Stochastic and Economic Optimal Control

Donald J. Chmielewski
and
Jin Zhang

Illinois Institute of Technology

Copyright (2017)
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>vii</td>
</tr>
<tr>
<td><strong>I Preliminaries</strong></td>
<td></td>
</tr>
<tr>
<td>1 Motivation and Overview</td>
<td>3</td>
</tr>
<tr>
<td>1.1 Level Control Example: Impact of Constraints</td>
<td>5</td>
</tr>
<tr>
<td>1.1.1 Open-loop Operation</td>
<td>5</td>
</tr>
<tr>
<td>1.1.2 Closed-loop Operation</td>
<td>6</td>
</tr>
<tr>
<td>1.1.3 Operating Point Selection</td>
<td>9</td>
</tr>
<tr>
<td>1.2 Reactor Control Example: Considering Profit</td>
<td>9</td>
</tr>
<tr>
<td>1.2.1 CSTR Profit and the Expected Dynamic Operating Region</td>
<td>10</td>
</tr>
<tr>
<td>1.2.2 CSTR Constraints and Backed-off Operating Point Selection</td>
<td>12</td>
</tr>
<tr>
<td>1.2.3 CSTR Profit with Equipment Modifications</td>
<td>15</td>
</tr>
<tr>
<td>1.3 Impact of Disturbance Characteristics</td>
<td>18</td>
</tr>
<tr>
<td>1.4 Chapter Summary</td>
<td>20</td>
</tr>
<tr>
<td>1.4.1 Review of Literature on Economic Based Controller Design</td>
<td>20</td>
</tr>
<tr>
<td><strong>2 Modeling of Dynamic Systems</strong></td>
<td>23</td>
</tr>
<tr>
<td>2.1 Nonlinear State-Space Models</td>
<td>23</td>
</tr>
<tr>
<td>2.2 Deviation Variables</td>
<td>29</td>
</tr>
<tr>
<td>2.3 Linearization of Nonlinear Models</td>
<td>31</td>
</tr>
<tr>
<td>2.4 Analytic Solution of a Linear State Space Model</td>
<td>39</td>
</tr>
<tr>
<td>2.5 Discrete-time Models</td>
<td>41</td>
</tr>
<tr>
<td>2.5.1 The Explicit Euler Method</td>
<td>43</td>
</tr>
<tr>
<td>2.5.2 The Sample and Hold Method</td>
<td>45</td>
</tr>
<tr>
<td>2.6 Case Study Processes</td>
<td>50</td>
</tr>
<tr>
<td>2.6.1 Furnace Reactor Process</td>
<td>50</td>
</tr>
<tr>
<td>2.6.2 System of Three Masses</td>
<td>52</td>
</tr>
<tr>
<td>2.6.3 Endothermic Reactor</td>
<td>53</td>
</tr>
<tr>
<td>2.6.4 Manufacturing Process</td>
<td>53</td>
</tr>
<tr>
<td>2.6.5 Building HVAC System</td>
<td>54</td>
</tr>
<tr>
<td>2.6.6 Vapor Product Reactor</td>
<td>55</td>
</tr>
<tr>
<td>2.6.7 Two CSTR Process</td>
<td>56</td>
</tr>
<tr>
<td>2.7 Chapter Summary</td>
<td>58</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Review of Linear Algebra</td>
<td>67</td>
</tr>
<tr>
<td>3.1</td>
<td>Sets, Subsets and Linear Vector Spaces</td>
<td>67</td>
</tr>
<tr>
<td>3.2</td>
<td>Inner Products and Orthogonality</td>
<td>69</td>
</tr>
<tr>
<td>3.3</td>
<td>Linear Transformations</td>
<td>71</td>
</tr>
<tr>
<td>3.4</td>
<td>Linear Independence and Orthogonalization</td>
<td>76</td>
</tr>
<tr>
<td>3.5</td>
<td>Matrix Inverse and the Determinant</td>
<td>82</td>
</tr>
<tr>
<td>3.6</td>
<td>Eigenvector Decomposition and Diagonalization</td>
<td>85</td>
</tr>
<tr>
<td>3.7</td>
<td>Non-Square Systems of Equations</td>
<td>90</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Least Squares Solutions and the Pseudo-Inverse</td>
<td>91</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Non-uniqueness and the Minimum Norm Solution</td>
<td>94</td>
</tr>
<tr>
<td>3.7.3</td>
<td>Singular Value Decomposition and Solutions to $Mx = b$</td>
<td>97</td>
</tr>
<tr>
<td>3.8</td>
<td>Positive Definite Transformations</td>
<td>101</td>
</tr>
<tr>
<td>3.9</td>
<td>Chapter Summary</td>
<td>103</td>
</tr>
<tr>
<td>3.9</td>
<td>Exercises</td>
<td>104</td>
</tr>
<tr>
<td>II</td>
<td>Modern Control Theory</td>
<td>111</td>
</tr>
<tr>
<td>4</td>
<td>Linear System Theory</td>
<td>113</td>
</tr>
<tr>
<td>4.1</td>
<td>Stability of a Linear Dynamic System</td>
<td>113</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Stability in the Discrete-time Framework</td>
<td>113</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Stability in the Continuous-time Framework</td>
<td>116</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Stability in the Sense of Lyapunov</td>
<td>117</td>
</tr>
<tr>
<td>4.2</td>
<td>Linear Feedback and State Observers</td>
<td>119</td>
</tr>
<tr>
<td>4.3</td>
<td>Controllability and Stabilizability</td>
<td>121</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Complete Controllability</td>
<td>121</td>
</tr>
<tr>
<td>4.3.2</td>
<td>The Controllable Subspace</td>
<td>124</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Necessary and Sufficient Conditions for Stabilizability</td>
<td>126</td>
</tr>
<tr>
<td>4.4</td>
<td>Observability and Detectability</td>
<td>128</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Observability and the Unobservable Subspace</td>
<td>128</td>
</tr>
<tr>
<td>4.4.2</td>
<td>First Experience with Duality</td>
<td>130</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Necessary and Sufficient Conditions for Detectability</td>
<td>131</td>
</tr>
<tr>
<td>4.5</td>
<td>Chapter Summary</td>
<td>132</td>
</tr>
<tr>
<td>4.5</td>
<td>Exercises</td>
<td>133</td>
</tr>
<tr>
<td>5</td>
<td>Stochastic Processes</td>
<td>139</td>
</tr>
<tr>
<td>5.1</td>
<td>Review of Probability and Random Variables</td>
<td>139</td>
</tr>
<tr>
<td>5.2</td>
<td>Introduction to Stochastic Sequences</td>
<td>144</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Mean, Variance and Autocorrelation</td>
<td>145</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Impact of Autocorrelation</td>
<td>148</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Disturbance Modeling via a Shaping Filter</td>
<td>151</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Combining the Disturbance and Process Models</td>
<td>152</td>
</tr>
<tr>
<td>5.3</td>
<td>Analysis of Stochastic Processes</td>
<td>154</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Process Mean and Deviation Variables</td>
<td>154</td>
</tr>
<tr>
<td>5.3.2</td>
<td>The Variance Sequence</td>
<td>157</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Autocorrelation and Variance Settling Time</td>
<td>159</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Vector Valued Stochastic Processes</td>
<td>162</td>
</tr>
<tr>
<td>5.3.5</td>
<td>The Expected Dynamic Operating Region</td>
<td>166</td>
</tr>
</tbody>
</table>
# Contents

5.4 Continuous-time Stochastic Processes ........................................ 168  
5.5 Spectral Interpretations .......................................................... 172  
5.6 Design of Simple Disturbance Models ....................................... 176  
5.6.1 First Order Shaping Filters ................................................ 176  
5.6.2 Second Order Resonant Filters ........................................... 178  
5.7 Gramians and Covariance Equation Solutions .............................. 179  
5.8 Chapter Summary ................................................................. 183  
Exercises ...................................................................................... 183

6 Linear Estimation ................................................................ 189  
6.1 State Estimation in Continuous-time ......................................... 189  
6.2 Estimation Theory .................................................................. 193  
6.2.1 The Bayesian Approach ....................................................... 193  
6.2.2 Estimation with Gaussian Random Variables ......................... 195  
6.2.3 Data Reconciliation ............................................................. 197  
6.3 The Discrete-time Kalman Filter .............................................. 200  
6.3.1 Evolution of Estimates and Predictions ................................. 200  
6.3.2 The Steady-State Kalman Filter ........................................... 206  
6.3.3 Orthogonality and the Innovations Process ............................ 211  
6.4 Multi-step Prediction and Smoothing ........................................ 213  
6.4.1 Multi-step Predictions of the State ...................................... 213  
6.4.2 Smoothing of the State Estimates ....................................... 215  
6.5 Chapter Summary .................................................................. 216  
Exercises ...................................................................................... 217

7 Linear Quadratic Optimal Control ............................................... 221  
7.1 Finite-time LQOC .................................................................. 221  
7.1.1 The Batch Approach ........................................................... 223  
7.1.2 Dynamic Programming ......................................................... 227  
7.2 Infinite-time LQOC ................................................................. 232  
7.3 LQOC in Continuous-time ....................................................... 237  
7.4 Chapter Summary .................................................................. 239  
Exercises ...................................................................................... 239

8 Stochastic LQOC .................................................................... 245  
8.1 Stochastic LQOC with Full State Information ............................. 245  
8.2 Stochastic LQOC with Partial State Information ......................... 255  
8.3 Stochastic LQOC with PSI and Delay ....................................... 258  
8.4 Stochastic LQOC in Continuous-time ....................................... 261  
8.5 Chapter Summary .................................................................. 264  
Exercises ...................................................................................... 264

9 Model Predictive Control ............................................................. 269  
9.1 The Receding-Horizon Framework ......................................... 269  
9.2 Constrained LQOC .................................................................. 274  
9.2.1 CLQOC with a Finite-horizon .......................................... 275  
9.2.2 CLQOC with an Infinite-horizon ....................................... 277  
9.2.3 Stability of CLQOC with an Receding Horizon ...................... 280  
9.3 MPC with Stochastic Disturbances ......................................... 287  
9.3.1 MPC with Soft Constraints ............................................... 289
### Contents

9.3.2 MPC with Computational Delay and PSI ................. 296
9.4 Chapter Summary ............................................ 297
Exercises ......................................................... 298

#### III Economic Based Controller Design

10 Matrix Inequalities ............................................. 303
  10.1 Preliminaries .................................................. 303
  10.2 Stabilizability and Detectability ............................. 306
    10.2.1 LMI Conditions for Continuous-time Systems ........ 306
    10.2.2 LMI Conditions for Discrete-time Systems ............ 308
  10.3 Convexification of Covariance Equations .................... 310
    10.3.1 Open-Loop Stochastic Processes ....................... 310
    10.3.2 Closed-Loop Stochastic Processes ..................... 314
  10.4 Riccati Equations ............................................ 321
  10.5 Chapter Summary ............................................. 326
Exercises ......................................................... 327

11 Constrained Minimum Variance Control ........................ 331
  11.1 Covariance Based Controller Design ......................... 331
    11.1.1 Minimum Variance Control .............................. 333
    11.1.2 Constrained MV Control ................................ 337
    11.1.3 EDOR Interpretation of CMV Control ................... 338
  11.2 CMV in Discrete-time ......................................... 341
    11.2.1 CMV with Computational Delay .......................... 341
  11.3 CMV with Partial State Information ........................ 342
    11.3.1 Discrete-time CMV with Partial State Information .... 344
    11.3.2 CMV with One-Step Delay and Partial State Information 345
  11.4 CMV and the LQOC Problem ................................... 346
    11.4.1 Preliminary Results from Optimization ................ 346
    11.4.2 The Scalar Case ....................................... 347
    11.4.3 The Multivariate Case .................................. 350
    11.4.4 LQOC Equivalence in Discrete-time .................... 353
  11.5 Inverse Optimality and MPC Tuning .......................... 354
    11.5.1 Inverse Optimality ..................................... 355
    11.5.2 MPC Tuning ............................................ 358
  11.6 Chapter Summary ............................................. 362
Exercises ......................................................... 362

12 Hardware Selection and the GBD ............................... 367
  12.1 Examples of Hardware Section ............................... 368
    12.1.1 Data Reconciliation Based Sensor Selection ........... 368
    12.1.2 Sensor Selection for Open-loop Dynamic Systems ...... 370
    12.1.3 Actuator Selection for Closed-loop Dynamic System .... 372
  12.2 Generalized Benders Decomposition .......................... 373
    12.2.1 General Framework ..................................... 374
    12.2.2 Problem Q ............................................. 375
    12.2.3 L-dual Adequacy ....................................... 376
    12.2.4 GBD if $f_1(x) = 0$ .................................. 376
Contents

12.2.5 Summary of the GBD Algorithm ....................................... 378
12.3 GBD Applied to Hardware Selection ................................... 379
  12.3.1 Data Reconciliation Based Sensor Selection .......................... 379
  12.3.2 Sensor Selection for Open-loop Dynamic Systems ................. 384
  12.3.3 Actuator Selection for Closed-loop Dynamic Systems .......... 386
12.4 Closed-loop Sensor and Actuator Selection ............................ 390
  12.4.1 Actuator Selection with PSI .......................................... 390
  12.4.2 Closed-loop Sensor Selection: A First Attempt .................... 391
  12.4.3 Closed-loop Sensor Selection: A Second Attempt ................. 391
  12.4.4 Simultaneous Sensor and Actuator Selection ....................... 396
12.5 Hardware Selection for Discrete-time Systems ....................... 400
12.6 Chapter Summary .......................................................... 404
Exercises ................................................................. 404

13 Economic Linear Optimal Control .............................. 405
  13.1 Fixed Controller BOP Selection ......................................... 405
  13.2 Economic Linear Optimal Control ...................................... 405
  13.3 ELOC Solution Methods .................................................. 405
    13.3.1 The Heuristic Solution Method ..................................... 405
    13.3.2 ELOC and the Generalized Bender's Decomposition .......... 405
  13.4 ELOC with PSI and in Discrete-time .................................. 405
  13.5 Constrained ELOC ........................................................ 405
  13.6 Chapter Summary .......................................................... 405

14 Economic MPC ............................................................. 407

15 Controller Integrated Process Design ............................ 409

Bibliography ............................................................... 411
Preface

A recent trend within the chemical engineering control community is to incorporate process economics into the design of control systems. While such efforts are welcome, the analysis of these economic based controllers requires a broad and fairly deep understanding of modern control theory. As such, this book has two broad objectives. The first is to provide a succinct, but self-contained, development of modern control theory. Then, given this foundation, the second objective is to provide a compilation of recent results on the subject of economic based control system design.

In terms of prerequisite knowledge, the book assumes the reader to have a technical background in engineering or mathematics at the undergraduate level - in essence a familiarity with ordinary differential equations. While a familiarity with the concept of feedback will be helpful, the Laplace domain methods of classic control will not be discussed, as nearly all developments will take place in the time domain.

There are two intended audiences - first-year graduate students and self-study practitioners. If used within a course, the text is constructed such that the instructor should have a great deal of flexibility. The most basic option is to cover the material of Part II in a one semester survey course. This first option assumes students have taken a graduate level engineering mathematics course that has provided the modeling and linear algebra foundation given in Part I. If a survey treatment of topics is the goal, a fair amount of depth will need to be sacrificed if one would like to cover all 6 chapters of Part 2. To this end each chapter begins with a very simple sketch of the main idea, usually in a scalar context, with the goal of helping the student develop a bit of intuition about the more detailed material to follow. Thus, it is feasible for lectures to focus on the derivation given in first part of the chapter, and then skip to the results portion for the more general case. This is the approach used at IIT. A second option, is to offer a two semester sequence of courses, the first covering Chapters 2-5 and the second covering Chapters 6-9. In this case, there should be sufficient time to work through the more detailed derivations. The material of Part III is more appropriate for self-study practitioners and advanced graduate students. If offered as a course, a small group setting (such as that found in a special problems course) is likely best. In either case, it will be essential to work though the computational examples given within the chapters and reproduce the given results. In nearly all examples, sample MATLAB code is provided to get one started.

Each chapter ends with a set of exercises. While many of these are theoretical in nature (usually simple extensions or special cases of results developed in the chapters), most are numeric or computational based. These computational exercises are critical to developing an initial understanding of the theoretic material and are strongly encouraged, especially for self-study practitioners. Once this initial understanding has been achieved, the theoretic developments of the chapter should be revisited and the more theoretic exercises should be attempted, so as to develop a deeper understanding of the theory. In the last section of Chapter 2, a number of moderately sized case study systems are introduced.
- each containing 3-6 states, multiple inputs and several performance outputs. In the exercises of subsequent chapters, each of these case study systems is revisited with a focus on the concepts of that chapter. While teaching the course at IIT, groups of 2-3 students are assigned to each of these case study systems. Then, at the midpoint and end of the semester, each group will present a summary of their individualized exercises to the rest of the class. In addition to forcing each group to explain their results, the entire class receives the benefit of seeing the theoretic notions within a variety of applications. For the self-study practitioner, it is recommended to select two of these case study systems and attempt the corresponding exercises of each chapter. It is highlighted that these case study systems continue to be utilized within the chapters of Part III, and represent the best way to learn how to implement the newly developed techniques of those chapters.

Chapter 1 begins by discussing the conceptual relationship between the design of a process and the control of a process. This discussion will illustrate the need for more integration of these two activities. The connecting concept is that of process uncertainty and the fact that a realization of this uncertainty cannot occur until after the process design phase is complete. While the control system will mitigate the impact of this uncertainty, the level to which this mitigation occurs depends upon the capabilities of the process, which of course were engendered during the design phase. The remainder of Chapter 1 focuses on a couple of simple examples intended to illustrate the multifaceted relationships between uncertainty characteristics, mitigation capabilities, process constraints and process economics. While the topic of process economics is reserved for Part III, these examples provide motivation for many of the subjects discussed in Part II.

Chapter 2 introduces the state space modeling framework to be used throughout the text. Of particular interest is the notion of a steady-state operating point and the deviation variables defined with respect to that operating point. While most texts on advanced control will gloss over this rudimentary topic, such concepts will be an essential component of Part III. Specifically, the relationship between dynamic operation (in deviation variables) and the steady-state operating point (used to define deviation variables) is used to account for process economics. Procedures for model linearization and conversion of continuous-time models to the discrete-time framework are also provided in Chapter 2.

Chapter 3 provides an introduction to linear algebra, and while not comprehensive, an effort has been made to make this chapter self-contained in the sense that all developments are based on previous results. While the material of Chapter 3 will be useful throughout the text, most will be employed in the linear system theory derivations of Chapter 4. In addition to the usual topics of stability, controllability and observability, Chapter 4 gives a fairly detailed development of the conditions for stabilizability and detectability. The need for these conditions is motivated by a rather intuitive introduction to state feedback and state observer design. Looking ahead to Chapter 10, the conditions for stabilizability and detectability will be revisited and serve as the first examples of using matrix inequalities to answer control theoretic questions. Chapter 4 will also introduce Lyapunov based stability conditions for linear systems, which will be central to the proofs of Chapter 10.

In Chapter 5 the notion of a stochastic process is introduced and distinguished from an arbitrary collection of random variables through the concept of autocorrelation. Then, using the discrete-time framework an intuitive derivation of the covariance equation is presented. The chapter also emphasizes the difference between colored and white noise, and illustrates that a colored noise disturbance model can be constructed by driving a linear system with white noise. Chapter 5 concludes with a discussion of relationship between the covariance equation and it associated Gramian, which again will be required for the proofs of Chapter 10. Chapter 6 begins with a derivation of the optimal state
estimator (or Kalman filter) as a simple application of the continuous-time covariance equation. The development of the discrete-time version, however, begins with a review of the more general topic of estimation theory. This more involved derivation serves to expose the deeper characteristics of the Kalman filter, specifically the orthogonality principle and the whiteness of the innovations process. This discrete-time development also emphasizes the subtle difference between the estimator and the more commonly found one-step predictor. Chapter 6 concludes with a short introduction to multi-step prediction and smoothing, both of which will be of utility in Part III.

In Chapter 7, the deterministic linear quadratic optimal control (LQOC) problem is introduced by first highlighting the fact that the finite-time version is just a quadratic program that can be solved using standard optimization software. In addition to being an intuitively attractive approach, this perspective sets a foundation for the development of model predictive control, to be introduced in Chapter 9. To arrive at a deeper understanding of the LQOC problem, its solution is also derived using dynamic programming, which give the solution in the form of a linear state feedback and provides the gateway to solving the infinite-time version. Of course, this dynamic programming perspective will serve as the foundation for solving the stochastic LQOC problem, to be discussed in Chapter 8. The dynamic programming perspective also leads to the Riccati equation, which is of course identified as the dual of the Riccati equation associate with the Kalman filter of Chapter 6. The concept of model predictive control (MPC) is introduced in Chapter 9 by simply advocating a receding horizon implementation of the deterministic LQOC and highlighting the fact that a quadratic programming solution approach will allow for a straightforward incorporation of inequality constraints. Chapter 9 concludes with a discussion of MPC in the context of stochastic disturbances. The examples of that section will highlight the need for more sophisticated tuning procedures and will serve to motivate many of the developments of Part III.

Part III begins by laying a computational foundation for the remaining chapters. Chapter 10 will show that many of the control related challenges of Part II can be expressed as feasibility questions in the context of a linear matrix inequality (LMI). The advantage of this LMI form is that these conditions are convex and thus will result in a significant reduction in computational effort and in the more advanced cases will provide the only viable path to obtaining a global solution. Building on the LMI results of Chapter 10, Chapter 11 introduces a new perspective on control system design. In contrast to the LQOC, which relies on objective function weights for tuning, the proposed constrained minimum variance (CMV) controller design scheme is enhanced by a set of closed-loop variance bounds. Since these variance bounds have greater physical meaning, as compared to the LQOC objective function weights, the CMV approach is expected to be more attractive to the control engineer. While the controller generated by the CMV approach cannot enforce point-wise-in-time constraints, Chapter 11 will illustrate how this controller can be used for the tuning of an MPC controller. The basic idea is to show that the variance constraints within CMV control serve to simply modify the quadratic objective function weights of the LQOC. Once these modified weights are obtained - using inverse optimality - they are used within a standard application of MPC.

The downside of the LMI form is that enforcement of these constraints requires the use of specialized semi-definite programming (SDP) optimization routines, which in most cases are incapable of enforcing general nonlinear constraints. Since this feature will be require in later chapters, Chapter 12 provides an introduction to the generalized Bender’s decomposition (GBD) algorithm. This algorithm serves to segregates the LMI portion of the problem from the general nonlinear portion and allows each part to be addressed using an optimization routine that is appropriate for that class of problems. In Chapter
12, the GBD approach is illustrated in the context of the fairly simple hardware selection problem.

In Chapter 13, the CMV concept is extended to an economic framework. The first step is to recognize that the steady-state operating condition (usually selected prior to the controller design phase) dominates the economics of the process. Thus, the proposed economic linear optimal control (ELOC) scheme advocates performing both tasks (operating point selection and controller design) simultaneously. While the controller design part is nearly identical to the CMV problem, the operating point selection part requires a steady-state (possibly nonlinear) process model. As such the solution to the ELOC design problem will in most cases require use of the GBD algorithm. Once the ELOC policy has been calculated, the method of inverse optimality can once again be used for the tuning of an MPC controller to be denoted as constrained ELOC.

Chapter 14 turns to a recent extension of MPC known as economic MPC (EMPC). EMPC is nearly identical to the original MPC algorithm, in that a receding horizon framework is employed. The main difference is in the objective function, where the usual quadratic objective (which has little physical meaning) is replaced by an economic objective in the sense that the stage cost of each time step reflects actual operating costs of the process during that time period. In the second half of Chapter 13, the relationship between EMPC and ELOC will be illustrated. It will be shown that controllers derived from the ELOC problem (the constrained ELOC policy of Chapter 13 and a newly developed approximate infinite-horizon EMPC policy) overcome the shortcomings of EMPC, specifically stability concerns and inventory creep. A particularly attractive feature of EMPC is the ability to address processes with time-varying economics, especially those that vary on a timescale that is shorter than the characteristic time of the process. Chapter 14 will illustrate how the ELOC problem can be extended to address this type of situation and show how it and its derivatives compare to EMPC.

Chapter 15 considers the question of controller integrated process design. Conceptually, the problem is simple in that the operating cost objective function of the ELOC is augmented with the capital costs of equipment purchase or upgrade. However, the level of complexity involved in calculating a global solution to such problems will depend on the characteristics of the process dynamics - most notably how much the process design variables influence the dynamics of the process? As such, the structure of the final chapter is such that various classes of problems are identified each with a solution scheme of increasing complexity. It is finally highlighted that the approaches of this chapter will be essential for the design of processes with time-varying economics, since these types of situations require a consideration of process dynamics and a failure to do so would be incongruent with the objectives of the process.
Part I

Preliminaries
Chapter 1
Motivation and Overview

The fundamental challenge of process design is to specify equipment (and operating conditions of that equipment) such that a physical task may be performed. In nearly all cases, the design process is guided by economic motives in the sense that the goal is to minimize a combination of capital and operating costs. As one would imagine, the complexity of these design questions can be immense and typically will result in the formulation of a large scale, mixed integer, nonlinear, optimization problem. As such, enormous effort has been put toward simplifying problem formulations and the development of computational tools. A point of particular relevance is the role of equipment models. Specifically, a flawed equipment model will, necessarily, result in a sub-optimal (and possibility a non-viable) system design. The unfortunate fact, however, is that all equipment models are flawed regardless of complexity or perceived fidelity. This fact stems from the inevitable deviations that will occur between the design phase and the construction of the physical system. Consider, for example, the uncertainty associated with the physical characteristics of materials or the inconsistency of fabrication methods.

The primary task of a control system is to mitigate the impact of uncertainties. That is, the controller is expected to drive key operating variables (the control variables) to the values specified by the original design, and do so with virtually no model information. Controllers are able to achieve this seemingly impossible task by simply implementing the notion of feedback. In particular, the controller has the advantage of on-line measurements, which it uses to infer the impact of uncertainty. The feedback loop is closed by endowing the controller with an ability to manipulate certain elements of the system, which it does until the impact of uncertainty is canceled out. The important point being that concerns about model uncertainty during the system design phase can be reduced (and in many cases ignored) by assuming the existence of sufficient controllers during the operational phase.

While the above paradigm has historic precedence, the term “sufficient controller” should give one pause. What is a sufficient controller? One way to answer this question is to focus on the manipulated variables or more appropriately the actuation equipment. Do the actuators have enough range to be able to drive the control variables back to the design conditions? To answer this question, one would need additional information about the expected uncertainty - the types one should expect as well as the set of possible values for each. Another way to view the sufficient controller question is to ask if the set of measurements contain enough information to conclude that control variables have been driven back to design conditions. To answer this question, an uncertainty model
will again be needed, this time with respect to the accuracy and precision of measurement equipment.

While the above questions seem like a natural extension of the traditional design challenge, they carry an implicit assumption: implementation of a sufficient controller will have little or no impact on system economics. Though one could consider the cost of measurement and actuation equipment, this aspect only scratches the surface. The more fundamental concept is that actuation of manipulated variables will likely change the system operating conditions, even if the control variables are driven back to design conditions. Thus, the deeper question concerns the impact manipulated variables will have on system economics. Consider the simple example of using a furnace to manipulate the temperature of a material flow. The action of increasing stream temperature will, of course, result in greater fuel costs. Due to the nature of uncertainty, the conceptual challenge during the design phase is that one cannot know where the manipulated variables will end up during system operation. At best an uncertainty model can be used to determine the range of values one would expect. Based on this region, one could then make design decisions based on average economics or alternatively based on worst case economics. Notions similar to the about have been developed in the literature under the subject titles of flexibility analysis and stochastic optimization. See Section 1.4 for citations to those efforts.

A broader perspective is to relinquish the notion of always driving control variables back to the design conditions. The motivation being that for certain realizations of the uncertainty, the economic penalty of moving a manipulated variable may be greater than the penalty of allowing a control variable to be off-set from its design condition. The modern control system architecture is, in fact, an implementation of such notions. Figure 1-1 illustrates the cascade structure typically used. The inner regulatory loops are seen as actuators by the real-time optimization loop. These on-line optimization type controllers choose set-point commands such that the economic performance of the system will be maximized, given a current estimate of uncertainty. Thus, the model used within the system design phase should attempt to capture the actions and performance of the real-time optimization loop. Clearly, the performance of this outer loop will depend on the quality of the models used for optimization as well as those used for uncertainty estimation.

One way to classify uncertainty models is with regard to time. The two extremes are constant uncertainties and time-dependent uncertainties. For example, an aspect of the original system model may be unknown during the design phase, but once the system
1.1. Level Control Example: Impact of Constraints

Consider the surge tank depicted in Figure 1.2. The flow controller (or servo-loop) is intended to regulate flow from the pump. Specifically, this controller (FC) receives a set-point command, \( v^{(sp)} \), and then modifies valve position until the measured flow (FT) is equal to the given set-point. These low level loops (typically mass flow controllers) serve to simplify process modelling for the higher level controllers. That is, rather than model valve and pump characteristics, the high level controller will simply give a set-point command, which will be executed with a high degree of certainty. Thus, it is reasonable to assume that \( v = v^{(sp)} \) for all time, and that \( v \) can be selected as a manipulated variable for the higher level controller. It is also noted that the actual manipulated variable, valve position, is not expected to have any bearing on the economics of the process. However, the manipulated variable of the higher level controller, flow from the tank, will have an impact on economics; in that power to the pump will be a function of flow rate. However, in many cases pumping power is small enough that it can be ignored as well.

1.1.1. Open-loop Operation

Before we consider the higher level controller, let us highlight some of the objectives of our surge tank. Assume the inlet flow, \( v_{in} \), is from a reactor that varies its throughput to achieve temperature regulation. Thus, a reasonable characterization is to assume an average value and a range of variability for this inlet flow, say \( v_{in} = 30 \pm 3 \text{ m}^3/\text{min} \). Assume the exit flow, \( v \), will be sent to a separation unit that demands its inlet flow is less than 31 \( \text{m}^3/\text{min} \). Finally, the volume of liquid in the tank, \( V \), should not exceed the total tank volume (which would cause an overflow), and the tank should not be allowed to run dry (which would damage the pump). If the maximum volume of the tank is 20 \( \text{m}^3 \), then these specifications can be stated as \( 0 \leq V \leq 20 \text{m}^3 \). Thus, the objective of the surge tank is to deliver a downstream flow less than 31 \( \text{m}^3/\text{min} \), in the face of upstream variations, \( v_{in} = 30 \pm 3 \text{ m}^3/\text{min} \), while avoiding tank faults \( 0 \leq V \leq 20 \text{ m}^3 \).
Chapter 1. Motivation and Overview

We begin by assuming a constant exit flow \( v = 30 \text{ m}^3/\text{min} \) for all time, and test the system under an arbitrarily selected inlet flow scenario (depicted in the left plot of Figure 1.3). To determine the impact of this scenario it is common to construct a model of the surge tank. A simple volume balance yields:

\[
\dot{V} = v_{in} - v \tag{1.1}
\]

The dot notation indicates the derivative with respect to time: \( \dot{V} = dV/dt \). If \( v \) is constant and \( v_{in} \) is a known function of time, then \( V(t) \) may be determined by simple integration.

\[
V(t) = V(0) + \int_{0}^{t} v_{in}(\tau)d\tau - vt \tag{1.2}
\]

The right plot of Figure 1.3, shows that result of a constant exit flow policy and illustrates that an over-flow of the tank is expected.

1.1.2 Closed-loop Operation

Now consider the installation of a level controller (Figure 1.4), where tank volume is the control variable (CV) and the manipulated variable (MV) is exit flow \( v \). In this cascade configuration, the level (or volume) controller is at a higher level than the flow controller (or equivalently LC is master to the slave FC).

For simplicity assume the level controller is of the Proportional Integral (PI) variety. As an aid to the tuning of a PI controller, it is common to convert our existing time-domain plant model (Equation 1.1) to the Laplace domain to arrive at a transfer function model. If the Steady-State Operating Point (SSOP) of the tank is assumed to be \( V^{SSOP} = V^{sp} = 10 \text{ m}^3 \), \( v_{in}^{SSOP} = 30 \text{ m}^3/\text{min} \) and \( v^{SSOP} = 30 \text{ m}^3/\text{min} \), then one can define deviation variables as: \( x = V - V^{SSOP} \), \( w = v_{in} - v_{in}^{SSOP} \) and \( u = v - v^{SSOP} \), which results in the following deviation variable form of the original plant model:

\[
\dot{x} = w - u \tag{1.3}
\]

Taking the Laplace transform yields the following:

\[
sX(s) = w(s) - u(s) \tag{1.4}
\]
Then the open-loop transfer function is:

\[ X(s) = G_p(s)u(s) + G_d(s)w \]  

(1.5)

where the transfer function corresponding to the process is \( G_p(s) = -1/s \) and that of the disturbance is \( G_d(s) = 1/s \). The block diagram of the closed-loop system is depicted in Figure 1.5, where the transfer function of the PI controller is:

\[ G_c(s) = K_c \left(1 + \frac{1}{\tau I s}\right) \]  

(1.6)

![Figure 1.4. Surge tank with level controller](image)

![Figure 1.5. Block diagram of level controller](image)

Now employ a commonly recommended controller tuning from the literature (Table 12.1, entry E of Seborg et al., [1]):

\[ K_c = -2/\tau_c \quad \text{and} \quad \tau_I = 2\tau_c \]  

(1.7)

where \( \tau_c \) is the desired closed-loop time constant (to be selected by the user). As a point of interest, the resulting closed-loop transfer functions are:

\[ \frac{x(s)}{w(s)} = \frac{1}{s} \left(\frac{\tau_c s}{\tau_c s + 1}\right)^2 \quad \text{and} \quad \frac{u(s)}{w(s)} = \frac{1}{s^2} \left(\frac{\tau_c s}{\tau_c s + 1}\right)^2 \]  

(1.8)

However, it should be noted that (1.8) was not used to generate the simulations below. Subsequent chapters will illustrate alternative and more general methods to simulate a system subject to such inputs.
Given this tuning method, we can now consider the impact of different $\tau_c$ selections:

- If one selects $\tau_c = 1 \text{ min}$, then the top plots of Figure 1.6 illustrates tight regulation with respect to tank volume. However, the objective of keeping $v < 31 \text{ m}^3/\text{min}$ is violated. It is important to highlight that the top plots of Figure 1.6 are those one would typically find in a classic control textbook - the selected CV (tank volume) is being regulated at the expense of the MV (exit flow).
- If one selects $\tau_c = 10 \text{ min}$, then the middle plots of Figure 1.6 shows a similar situation in that tank hold-up is within limits and exit flow is not. However, the exit flow violations are less severe than in the previous case.
If one selects $\tau_c = 50 \text{ min}$, then from the bottom plots it is seen that exit flow does satisfy the $v < 31 \text{ m}^3/\text{min}$ constraint, but now the volume constraint is violated.

### 1.1.3 Operating Point Selection

The result of our first example is not too promising. We have yet to find a controller tuning that achieves all of our constraint satisfaction objectives. So, what is the solution? Well, we must broaden the set of alternatives. In fact, there are many options available. The first that come to mind is to install a larger tank. Certainly, the controller with $\tau_c = 50 \text{ min}$ would have been sufficient, if the tank was $30 \text{ m}^3$ in size. A similar option is to upgrade the downstream separation unit so that it can accept inlet flows larger than $31 \text{ m}^3/\text{min}$. However, from a capital cost perspective, it will likely be much cheaper to enlarge the surge tank. A third option is to change the average flow from the reactor feeding the surge tank. For example, one could set $v_{in} = 29 \pm 3 \text{ m}^3/\text{min}$. While this change may cause other problems with respect to reaction conversion or yield, such a change is certainly plausible. In this case and with $\tau_c = 10 \text{ min}$, the volume curve of Figure 1.7 would be unchanged, while the $v$ and $v_{in}$ in curves would be shifted down by one unit, which would result in satisfaction of the constraint $v < 31$. To assess the revenue impact of this change in nominal throughput, one could employ the following simple relation:

$$R_{actual} = R_{design} + K(v_{ss op}^{actual} - v_{ss op}^{design})$$

where $K$ indicates the value of the product (per m$^3$), $v_{ss op}^{design}$ is the throughput designated during the design phase and $v_{ss op}^{actual}$ is the actual throughput specified by the operator. Thus, if the actual throughput is equal to the design value, then there will be no change in profit. If the actual throughput is different, then process revenue will change in the expected way (i.e., increased throughput gives more revenue and decreased gives less).

To summarize the example, two extreme situations have been identified:

- The controller of the top plots of Figure 1.6 completely ignores and violates the exit flow constraint.
- The uncontrolled case of Figure 1.2 assumes the tank is sufficiently large that volume constraints can be ignored.

While many systems can be put into one of these categories, many others cannot. This is especially true when the design of the plant is highly focused on economic concerns, which tends to reduce equipment sizes and tighten constraint limits. Thus, an objective of the text is to illustrate how to handle the gray areas of controller tuning. That is, address the cases in which the constraints of both the MV and CV are of concern.

### 1.2 Reactor Control Example: Considering Profit

Let us now expand on the notion of profit by exploring the operational aspects of an exothermic reactor; specifically, the interplay between operating point selection, controller tuning and the influence of constraints.
Table 1.1. Parameter values for CSTR example

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactor volume</td>
<td>$V$</td>
<td>5</td>
<td>$m^3$</td>
</tr>
<tr>
<td>Inlet concentration</td>
<td>$C_{A_{in}}$</td>
<td>1</td>
<td>kmole/m$^3$</td>
</tr>
<tr>
<td>Pre-exponential factor</td>
<td>$k_0$</td>
<td>$10^{10}$</td>
<td>1/s</td>
</tr>
<tr>
<td>Activation energy</td>
<td>$E/R$</td>
<td>1006.5</td>
<td>1/K</td>
</tr>
<tr>
<td>Inlet temperature</td>
<td>$T_{in}$</td>
<td>383</td>
<td>K</td>
</tr>
<tr>
<td>Heat of reaction / heat capacity</td>
<td>$\Delta H_r / \rho C_p$</td>
<td>10</td>
<td>K m$^3$/kmole</td>
</tr>
</tbody>
</table>

Figure 1.7. Feasible steady-state operating points for unconstrained CSTR

1.2.1 CSTR Profit and the Expected Dynamic Operating Region

Consider the following model of non-isothermal Continuous Stirred Tank Reactor (CSTR) intended to convert a species $A$ to a species $B$:

\[
0 = \frac{\psi}{V} (C_{A_{in}} - C_A) - k_0 e^{(-E/RT)} C_A
\]

\[
0 = \frac{\psi}{V} (T_{in} - T) - \left( \frac{\Delta H_r}{\rho C_p} \right) k_0 e^{(-E/RT)} C_A
\]

where $V$ is the volume of the reactor, $\psi$ is the volumetric flow through the reactor, $C_{A_{in}}$ is the inlet concentration of species $A$, $C_A$ is exit concentration and $T_{in}$ and $T$ are the inlet and exit temperatures. The parameters of the process are given in Table 1.1. One approach to the selection of a steady-state operating point is to select a volumetric flow rate, $\psi$, and then determine the resulting exit concentration and temperature as the solution of (1.10). The plots of Figure 1.7 illustrate this solution for a variety of volumetric flow values.

While minimization of the exit concentration of $A$ may seem like the objective, as this would maximize the exit concentration of species $B$ (defined as $C_{A_{in}} - C_A$), one should also consider the impact of volumetric flow. To this end, assume the instantaneous operating profit of the reactor is defined as the difference between product revenue (proportional to molar flow of species $B$ exiting the reactor) and feed costs (proportional to molar flow of species $A$ entering the reactor):

\[
P = c_{prod} \psi (C_{A_{in}} - C_A) - c_{feed} \psi C_{A_{in}}
\]

(1.11)
1.2. Reactor Control Example: Considering Profit

The plot of Figure 1.8 shows this profit function if \( c_{\text{prod}} = 10 \ $/\text{mole} \) of \( B \) and \( c_{\text{feed}} = 6 \ $/\text{mole} \) of \( A \). The point of maximum profit is then found to be at \( v = 0.09 \ \text{m}^3/\text{s} \). The corresponding exit temperature and concentration are 390.9K and 0.216 kmole/m^3, and are depicted in Figure 1.9. This operating point will be denoted as the Optimal Steady-State Operating Point (OSSOP).

![Figure 1.8. Instantaneous profit of exothermic CSTR as a function of volumetric flow](image)

Now consider the impact of a disturbance. Assume the inlet concentration, \( C_{A\text{in}} \), varies as in the top left plot of Figure 1.10. To respond to this disturbance (and others), a PI controller is typically installed. In this case, the PI controller uses exit temperature as the control variable and volumetric flow as the manipulated variable. The controller tuning parameters were selected to be as follows: a gain \( K_c = -0.005 \ \text{m}^3/\text{s} \ K \), an integral time-constant \( \tau_I = 50 \ \text{s} \) and a temperature set-point \( T_{sp} = 390.9K \). Then, simulation of a dynamic version of the CSTR model with the above controller results in the remaining plots of Figure 1.10 - subsequent chapters will illustrate how to determine this dynamic model and implement this simulation. While the time-series plots of Figure 1.10 seem to be the natural depiction of closed-loop behaviour, the phase-plane representation will give additional insight. The bottom two plots of Figure 1.10 show that the reactor will operate in a region surrounding the operating point. We will denote this region as the Expected Dynamic Operating Region (EDOR). The utility of a EDOR is greatest in the context of process constraints, as will be illustrated next.

![Figure 1.9. Instantaneous profit of exothermic CSTR as a function of volumetric flow](image)
1.2.2 CSTR Constraints and Backed-off Operating Point Selection

Consider again the CSTR model of (1.10) and the profit function of (1.11), but now require that the following operational constraints are observed:

\[ T \leq 390K \quad \text{and} \quad v \leq 0.35 \text{ m}^3/\text{s} \]  

The temperature constraint could be due to protection of the catalysts or avoidance of side reactions, while the flow constraint is likely due to a pumping limit. Examination of the new set of feasible steady-state operating points (Figure 1.11) indicates that the
operating point of the previous section, \( v = 0.09 \ \text{m}^3/\text{s} \), is no longer available, as the temperature constraint would be violated at this flow. A re-examination of the profit function plot indicates that a new OSSOP will need to be selected. Based on Figure 1.12, this point corresponds to \( v = 0.1325 \ \text{m}^3/\text{s} \) and \( T = 390K \).

Now consider the impact of the inlet concentration disturbance of Figure 1.10. If the PI controller used in Figure 1.10 is re-applied to the system (with the same tuning parameters and the new temperature set-point of 390K), then the EDOR of the top plots of Figure 1.13 will result. Clearly, this type of operation has frequent violations of the temperature constraint at 390. One way to address this concern is to select a lower temperature set-point. The middle plots of Figure 1.13 use this approach by selecting a set-point of 388.5K. The result is that the time-averaged operating point (the center of the EDOR) will now be backed-off from the OSSOP. As one would expect this Backed-off Operating Point (BOP) is still on the steady-state operating line (the solid line of the right plot) and results in an average volumetric flow of 0.225. In this case, the EDOR is pretty much in observance of the temperature constraint. However, the instantaneous profit is significantly reduced. In fact, the average profit is negative \( = -0.1 \ $/\text{s} \). Now consider a retuning of the controller. The bottom plots of Figure 1.13 result from a controller gain of \(-0.5\) (the previous gain was \(-0.005\)) and a temperature set-point of 389.85K. In this case, the size of the EDOR in the temperature direction is significantly reduced, which
is what allows the set-point to be selected so close the constraint. In this case the average profit is just below that predicted by the OSSOP calculation.

It is highlighted that reducing EDOR size in the temperature direction came at the cost of expanding the EDOR in the volumetric flow direction. Fortunately, the pump is sufficiently large and the flow constraint does not become an issue. However, if the pump was smaller, for example with a limit of 0.25, then a violation of this constraint would be expected. In this case, one would want to reduce the controller gain magnitude, to make the EDOR in the flow direction smaller. However, this would cause the temperature direction of the EDOR to increase and require further back-off from the OSSOP.

![Figure 1.13. EDOR of constrained CSTR. Top plots: \( T_{sp} = 390 \) and \( K_c = -0.005 \), middle plots: \( T_{sp} = 388.5 \) and \( K_c = -0.005 \), bottom plots: \( T_{sp} = 389.85 \) and \( K_c = -0.5 \)](image_url)

While this iteration between controller tuning (to observe constraints) and back-off
1.2. Reactor Control Example: Considering Profit

Selection (to improve profit) is intuitive for this single-input single-out example, things will become much less so for more complicated multi-input multi-output system. In these cases, the EDOR will expand from a flat region in a two dimensional space to a multi-dimension volume in a multi-dimensional space. Furthermore, the set of feasible steady-state operating points will expand from a single curve to a hyper-surface in the multi-dimensional space. With these types of problems in mind, the goal of the text is to develop computational methods that will automate the iterative decision process and arrive at a method that simultaneously selects the controller tuning parameters along with the BOP. The details of such a procedure will be given in Chapter 11.

Figure 1.14. Optimal steady-state operating point of constrained CSTR with recovery

1.2.3 - CSTR Profit with Equipment Modifications

While the previous CSTR shows an ability to observe constraints and be profitable, the level of profitability seems a bit low. As such, one might ask if an equipment modification will improve the situation. The profit function of (1.11) suggests that unconverted portions of the reactant (species $A$) in the exit stream will be lost with the product. However, if one were to add a downstream separation unit to recover species $A$ from the product stream, then the profit function would change to

$$ P = c_{prod} \sigma(C_{A, in} - C_A) - c_{feed} \sigma(C_{A, in} - RC_A) $$

(1.13)

where $R$ is the recovery factor (the fraction of $A$ in the exit stream that is recovered).

Figure 1.14 illustrates this new profit as a function of volumetric flow (with $R = 0.95$) and indicates that the new OSSOP is located along the flow constraint (right plot). The plots of Figures 1.15 illustrate various choices for the controller tuning and temperature set-point. The top plots show significant violation of the flow constraint and as such this set of tuning values are deemed unacceptable. The back-off of the middle plots shows little constraint violation, but the back-off is such that a fair amount of profit will be lost. The profit predicted by the OSSOP is $0.53 \$/s, while the backed-off profit is $0.39 \$/s - a loss of 26%. The controller retuning of the bottom plots shows a reduction of the EDOR in the volumetric flow direction, which allows the back-off to be very small and thus realize most of the profit predicted by the OSSOP. The profit for this case is $0.493 \$/s, a loss of only 7.5% with respect to the OSSOP.

An interesting aspect of this last controller is that it is barely a controller. That is, the gain of this controller is so small that it looks like the system is operating in open-loop. However, this is the appropriate controller for the given economic situation. This
Figure 1.15. EDOR of CSTR with recovery. Top plots: $T_{sp} = 387.2$ and $K_c = -0.5$, Middle plots: $T_{sp} = 389$ and $K_c = -0.5$, Bottom plots: $T_{sp} = 387.3$ and $K_c = -0.0005$

A non-intuitive result stems directly from the notion of using the profit function in the context of operating constraints as the guide for controller design. In other words, no self-respecting control engineer would suggest such a controller, unless there was an economic incentive to do so. While the current case is a bit extreme, it illustrates that the process of using BOP selection to tune a controller gives insight into the important operational aspects of the system. In many cases, this insight will be more valuable to the control engineer than the specific controller tuning parameters generated by the algorithm.

Let us now return to the controller of the top plots of Figure 1.15 and ask about the validity of our analysis in the sense that the simulation made no effort to model the equipment limitation. That is, if that controller was implemented on the real system...
1.2. Reactor Control Example: Considering Profit

and the controller asked for a volumetric flow greater than 0.35, then the pump would simply saturate and implement the limit. A simulation of this scenario is given in the top plot of Figure 1.16 and results in a profit of 0.488$/s. With the exception of a few exotic excursions - likely due to wind-up of the integral portion of the PI controller - the response to this policy looks to be almost identical to that of the bottom plots of Figure 1.15. That is, a volumetric flow that is almost always at a value very close to 0.35. The time series plots of Figure 1.16 highlight this fact.

![Graphs showing profit and reactor temperature over time.](image)

**Figure 1.16.** EDOR (top) and time series (bottom) of constrained CSTR with recovery: $T_s = 387.2$ and $K_c = -0.5$ and volumetric flow saturation at $0.35\ m^3/s$

The question at this point is why all the fuss? If a passive enforcement of the constraints will yield a nearly identical closed-loop response, then why bother with the EDOR analysis? One problem is that the only way to determine the response of a system with hard constraints is through simulation. However, if we ignore the constraints initially, then techniques exist to determine the EDOR analytically. These analytic methods can then be used for BOP selection and an unconstrained tuning procedure. Finally, once the unconstrained tuning parameters have been determined, one can impose hard constraints. However, if the tuning parameters of the unconstrained controller are such that constraint violations are rare, then one can be fairly confident that the EDOR of this controller with constraints imposed will be very close to that of the original. For example, consider the impact of imposing the saturation on the controller of the bottom plots of Figure 1.15. Clearly, one would expect the response to be almost identical for the constrained and unconstrained case. This near equality between the constrained and unconstrained controller will be referred to as an alignment between the controller tuning parameters and its constraints.
The above discussion could also be thought of from the BOP selection perspective. If, in the end, one will use a constrained controller, then the BOP selection procedure should search over all constrained controllers. However, due to the computational complexity of a constrained controller, one option is to use an unconstrained control as a surrogate to reduce computational burden. However, the quality of the surrogate will depend on how well its tuning is aligned with the constraints.

It should also be highlighted that the saturation simulation could not be performed on the controller of the top plot of Figure 1.13, where the temperature constraint is being violated. This is because a PI controller has no ability to enforce constraints on control variables. To enforce such constraints (on any variable that is not a manipulated variable) a Model Predictive Control (MPC) type controller will be required. Details on the implementation of MPC will be given in Chapter 8, but suffice it to say that MPC has an unconstrained formulation from which analytic methods exist to design the shape of the EDOR. Then, similar to the above discussion, imposition of constraints onto an unconstrained MPC that has tuning parameters aligned with the constraints will result in an EDOR that is almost identical to the unconstrained EDOR.

### 1.3 Impact of Disturbance Characteristics

![Figure 1.17. Surge tank simulations under uncontrolled policy](image)

![Figure 1.18. Surge tank simulations with PI control ($\tau_c = 10$)](image)
In the opening comments of the chapter, it was suggested that the type of uncertainty facing a process will influence expected performance. In this section, we return to the level control example of Section 1.1 and explore how the characteristics of the disturbance will impact performance and what recourse can be taken.

We begin by revisiting the simulation of Figure 1.3, where the exit flow from the tank was held constant (at 30 \( m^3/min \)) and the inlet flow, \( v_{in} \), was in the range 30 ± 3 \( m^3/min \). The left plot of Figure 1.17, reiterates that the tank is expected to over-flow under this scenario. However, the right plot of Figure 1.17, shows no over-flow, even though the exit flow is still fixed at 30 \( m^3/min \) and the inlet is 30 ± 3 \( m^3/min \). The only difference is that the time duration between inlet flow step changes is smaller in the right plot than in the left. In subsequent chapters, this notion of duration will be denoted as correlation time.

In the controlled case, Figure 1.18 illustrates that disturbance correlation time will also have an impact the manipulated variables. In Section 1.1, it was concluded that a reduction in nominal throughput (by 1 \( m^3/min \)) would be required to satisfy all the constraints. However, when the disturbance has a smaller correlation time (the right plot of Figure 1.18), all of the constraints seem to be satisfied and thus no change to the nominal throughput will be required. In fact, one may even wish to de-tune the controller further, as in Figure 1.19. In this case, on could increase the nominal throughput (by 0.75 \( m^3/min \)) and generate revenue greater than the design condition. Figures 1.20 and 1.21 illustrate these notions from the phase-plane perspective.
Chapter 1. Motivation and Overview

1.4 Chapter Summary

The objective of economic optimal control is to maintain the highest plant profitability in the face of a changing environment. However, this changing environment is not limited to just the short term variations in the disturbances, but includes the longer term changes in the process and its disturbance characteristics. In response to this evolving environment the desired recourse is to redesign (or retune) the controllers. To enable this retuning activity, two technologies will be required:

- A method to characterize the process (the plant and the disturbances).
- An economic based tuning method that can respond to a changing environment.

It should be highlighted that the proposed notion of disturbance characterization goes beyond simply modeling the impact of measured disturbances on process outputs. The models desired will need to make predictions about future disturbances based on measurements of the past. In this book we will begin by review existing methods for the first need area (process characterization) and then propose new control system design methods that make full use of the economic, plant and disturbance characterizations. The envisioned economic based controller design scheme is one in which a changing environment can be estimated on-line. Then, this updated environment is sent to an automated controller retuning algorithm. The net result is a system that can adapt, not just to changing output measurements, but to a changing scenario.

The notions outlined in this chapter indicate a destination for the text. Unfortunately, a significant amount of preliminary material will be required to arrive at this target. That is, the development of a general methodology capable of addressing a wide class of processes will require the use of significant background material. This background material will be provided in Part II. We will then return to the subject of plant economics in Part III. Fortunately, much of the material of the background chapters is expected to be of independent interest to the reader and can be thought of as a survey of the control theory literature. It should be highlighted that these background chapters will focus on presenting results rather than providing in-depth derivations of the results. However, at the end of each chapter efforts have been made to guide the reader to the source and documentation of the full derivation of the presented results.

1.4.1 Review of Literature on Economic Based Controller Design

It should be highlighted that the notion of profit guided control system design is not new or novel. In fact, this question has been around for many decades and many researchers have contributed significantly to answering such questions. The following is not intended to be an exhaustive review, but merely an attempt to provide context.
We begin with the process design and optimization community. This community is clearly focused on the topic of process economics. However, their attraction to controller design stems from an interest in mimicking how plant operations would deal with process uncertainty. While many approaches have been proposed, two are of particular interest; flexibility analysis and chance constrained optimization. Flexibility analysis uses nominal values in the objective function, but enforces constraints based on the full range of uncertain parameters. Deterministic versions include: Nishida et al. [2], Grossmann & Sargent [3], Polak & Sangiovanni-Vincentelli [4], Halemaane & Grossmann [5], Lasserre & Roubellat [6], Saboo et al. [7], Swaney & Grossmann [8, 9], Grossmann & Floudas [10], Raspania et al. [11], Ostrovsky et al. [12], Chakraborty & Linninger [13], Banerjee & Ierapetritou [14] and Malcolm et al. [15]. The stochastic extensions include: Pistikopoulos & Grossmann [16], Pistikopoulos & Mazzuchi [17], Straub & Grossmann [18], Pistikopoulos & Ierapetritou [19], Mohideen et al. [20], Ahmed & Sahinidis [21], Rooney & Biegler, [22] and Bansal et al. [23]. Most of these methods are primarily interested in a design based on a reasonable representation of the control system. However, in some the cases a feedback element is specified. Chance constrained optimization begins by calculating the probability density function of key process variables, and then seeks to ensure that constraint violations are below a pre-specified percentage. General descriptions can be found in Charnes & Cooper [24], Miller & Wagner [25], Uryasev [26], Cooper et al. [27], Henrion & Römisch [28] and Nemirovski & Shapiro [29]. Application to process systems type problems include: Petkov & Maranas [30, 31], Schwarm & Nikolaou [32], Li et al. [33, 34, 35, 36] and Henrion & Möller [37]. A handful of these efforts focus on controller synthesis.

Another important (but very large) group is the control structure selection community. While process economics remains the motive, the objective changes to which MV’s and CV’s should one use to mitigate uncertainty. Broad discussions on the motives and high level procedures can be found in: Morari et al. [38], Fisher et al. [39], Stephanopoulos & Ng [40], Skogestad [41, 42], van de Wal & de Jager [43] and Ward et al. [44]. Measurement and CV selection efforts include: Mellefent & Sargent [45], Harris et al. [46], Morari & O’Dowd [47], Morari & Stephanopoulos [48], Romagnoli et al. [49], Lee & Morari [50], Hovd & Skogestad [51], Wisnewski & Doyle [52] and van de Wouwer et al. [53], Antoniades & Christofides [54], Muske & Georgakis [55], Alonso et al. [56], Peng & Chmielewski [179], Singh & Hahn [58] and Armaou & Demetriou [59]. Recognizing the computational challenge of investigating all possible MV/CV set combinations, screening methods have been proposed: Lee et al. [60], Lewin [61], Cao & Rossiter [62], Zheng & Mahajanam [63], Figueroa [64], Vinson & Georgakis [65], Seferlis & Grievink [66], Chodavarapu & Zheng [67], Georgakis et al. [68], Hovd et al. [69], Biagiola et al. [70] and Cao & Saha, [71]. Finally, Perkins and colleagues (Narraway et al., [72]; Narraway & Perkins, [73, 74]; Loeblein & Perkins, [75, 76, 77, 78]; Heath et al., [79]; Kookos & Perkins, [80]) have used Backed-off Operating Point (BOP) selection to provide economic bearing. While most schemes address only the structure question, many generate good controllers as a by-product.

If the process design and control structure are given, then the next question is to determine a BOP. BOP selection schemes based on flexibility analysis include: Bahri et al. [81], Young et al. [82], Bahri et al. [83], Figueroa et al. [84], Contreras-Dordelly & Marlin, [85], Zhang & Forbes, [86], Figueroa & Desages, [87], Rooney & Biegler [88], Arbiza et al. [89], Young et al. [90] and Soliman et al. [91]. Most of these assume steady-state disturbances (the time characteristic of the disturbance is much larger than the largest time constant of the process). Approaches based on chance constraints or stochastic control include: Loeblein & Perkins [77, 78], van Hessem et al. [92], Muske,
Finally, we arrive at the economic based controller synthesis community. Engell \cite{Engell2003} and Rawlings \& Amrit \cite{Rawlings2009} advocate an economic objective function within MPC. van Hessem et al. \cite{vanHessem2010}, Peng et al. \cite{Peng2011} and Zhao et al. \cite{Zhao2012} extend BOP selection to simultaneously arrive at economically motivated controllers.

Finally, it is interesting to note that the relation between process control and process economics is highlighted in many introductory text on process control, see for example Seborg et al., \cite{Seborg2004}, Riggs \& Karim \cite{Riggs2005} and Romagnoli and Palazoglu \cite{Romagnoli2007}. For an interesting historical perspective on the subject, see Edgar \cite{Edgar2002}.
Chapter 2
Modeling of Dynamic Systems

The process model will be the starting point for all of our controller design efforts. This model should be of sufficient fidelity that the relationship between process inputs and outputs can be understood. It should be emphasized that a dynamic model of extremely high fidelity is not likely required and in many cases is not desired. This is due to the fact that the dominant features of a dynamic process can usually be captured by a relatively simple model. Furthermore, use of an overly complicated model may needlessly preclude utilization of some extremely powerful controller design methods. One should also remember that the model used to design the controller (or in some cases used within the controller) is not the actual process. Thus, in the absence of the actual process, the controller should be tested on a model with the highest fidelity available. It is finally noted that this chapter on the modeling of processes is only half of the modeling challenge. The second half - on the modeling of process disturbances - will be addressed in Chapter 5. Since disturbance attenuation is usually the objective of a control system, having a sufficient disturbance model will be essential to the design of a successful controller.

Throughout the text we will employ a state variable perspective in process modeling. For those unfamiliar with the state variable approach, the following high level description should provide some perspective. The state of a system at a given time is the minimum information required to completely specify the condition of the system. Said another way, if given the system state and the future values of the input signals, all future outputs can be predicted. This is significantly different from classic input-output models, which require all of the past inputs to the system to predict the future outputs. In essence, the system state contains all of the important information pertaining to the history of the process. In the development of a state space model, the central task is to appropriately select a set of process variables to be the state variables. Then, the value of these variables at a given time will be the system state. While the choice of state variables is not unique, they are usually chosen to correspond to physically meaningful quantities.

2.1 • Nonlinear State-Space Models

A reasonably general class of dynamic models is that of a nonlinear finite-dimensional state-space process. Such models will contain the following variables (or signals): a state vector $s$, a vector of manipulated variables $m$, a vector of disturbance variables $p$, a vector of performance outputs $q$ and a vector of measured outputs $\theta$, which are corrupted by measurement noise $n$. 
where the dot above the $s$ in equation (2.1) denotes the first derivative of $s$ with respect to time. The function $f$ is usually due to the physical aspects of the process. The fact that a first derivative term exists in (2.1) for each variable in the state vector, provides significant guidance in the identification of state variables. The function $h$ is a construct of the process engineer intended to reflect the important variables of the process. The vector of performance outputs, $q$, should not be confused with the set of measured outputs, $\theta$. The difference being that measured outputs, $\theta$, are physically obtained signals available to the controller for feedback purposes. In contrast, the performance outputs, $q$, may or may not be physically measured and may contain any variable of interest. Similarly, the set of manipulated variables, $m$, are those available to the controller to influence the process. In contrast, the disturbance variables, $p$, will influence the process, but the controller will have no influence over these signals. It is also noted that the disturbance variables and the measurement noise vector, $n$, are usually distinct.

**Example 2.1.** Consider a Continuous Stirred Tank Reactor (CSTR) in which a second order reaction $2A \rightarrow B$ is the only reaction taking place. More specifically, the kinetic expression for production of species $A$ is $r_A = -2kC_A^2$, where $C_A$ is the molar concentration.
2.1. Nonlinear State-Space Models

of species $A$ in the reactor and $k$ is the rate constant. ($r_A$ has units of moles of $A/m^3s$.) A mole balance over the reactor is:

Accumulation = Moles In - Moles Out + Moles Generated

which gives:

$$\frac{dN_A}{dt} = F_{A,in} - F_{A,out} + Vr_A$$ (2.4)

where $N_A$ is the number of moles of species $A$ within the reactor of volume $V$, $F_{A,in}$ is the molar flow of $A$ into the reactor and $F_{A,out}$ is the molar flow out. Noting that $N_A = VC_A$, $F_{A,in} = vC_{A,in}$ and $F_{A,out} = vC_A$ (where $C_A$ is the concentration of $A$ within the reactor, $C_{A,in}$ is the concentration of the inlet flow and $v$ is the volumetric flow through the reactor, with units of $moles/m^3$ and $m^3/s$, respectively) one arrives at the following differential equation model of the reactor.

$$\dot{C}_A = \frac{v}{V}C_{A,in} - \frac{v}{V}C_A - 2kC_A^2$$ (2.5)

From this expression it is clear that $C_A$ should be selected as the state variable. The selection of manipulated and disturbance variables will be up to the designer of the control system. Let us assume the volumetric flow rate through the reactor, $v$, can be manipulated and the inlet concentration of $A$, $C_{A,in}$, is outside the influence of the controller and is thus classified as the disturbance (assume $k$ and $V$ are constant for all time). Based on these definitions the function $f$ of equation (2.1) is found to be

$$f(s, m, p) = \frac{m}{V}p - \frac{m}{V}s - 2ks^2$$ (2.6)

Definition of the performance output is again up to the designer of the control system. If the only variables of interest are the state and manipulated variables, then the definition of $b$ of equation (2.2) is

$$b(s, m, p) = \begin{bmatrix} s \\ m \end{bmatrix}$$ (2.7)

The simplicity of the performance equation usually makes it somewhat difficult to determine. The procedure should become clearer when we address more complicated processes. The concept to note at this point is that since we are interested in two output signals the function $b$ will be a vector function.

Let us further assume there will be physical measurements available to the controller. In particular, assume $C_A$, concentration within the reactor, is measured and possesses additive measurement noise. In this case, the definition of $l$ of (2.3) would be:

$$l(s, n) = s + n$$ (2.8)

If there is also a measurement of the inlet concentration, $C_{A,in}$, then it would appear that the structure of Equation (2.3) precludes inclusion of this measurement. However, one may postulates a new state variable, $C^m$, to incorporate a small time constant, $\tau_m$, associated with the physical measurement: $\dot{C}^m = (C_{A,in} - C^m)/\tau_m$. In this case, the state vector would have two elements

$$s = \begin{bmatrix} s^{(1)} \\ s^{(2)} \end{bmatrix} = \begin{bmatrix} C_A \\ C^m \end{bmatrix}$$ (2.9)
and the full nonlinear process model can be stated as:

\[
\begin{align*}
    f(s, m, p) &= \begin{bmatrix}
        m \dot{p} - m \ddot{s}^{(1)} - 2k(s^{(1)})^2 \\
        (p - s^{(2)})/\tau_m
    \end{bmatrix} \\
    h(s, m, p) &= \begin{bmatrix}
        s^{(1)} \\
        m
    \end{bmatrix} \\
    l(s, n) &= \begin{bmatrix}
        s^{(1)} + n^{(1)} \\
        s^{(2)} + n^{(2)}
    \end{bmatrix}
\end{align*}
\]

(2.10) (2.11) (2.12)

Example 2.2. Consider the following system of differential equations describing a DC motor.

\[
\begin{align*}
    L_f \frac{d i_f}{dt} &= -R_f i_f + v_f \\
    L_a \frac{d i_a}{dt} &= -R_a i_a - K_v \omega i_f + v_a \\
    J \frac{d \omega}{dt} &= -B \omega + K_v i_a i_f + T_L
\end{align*}
\]

(2.13)

where \(i_f\) and \(i_a\) are the currents through the field and armature coils, respectively and \(\omega\) is the angular speed of the motor. In addition, \(v_f\) and \(v_a\) are the voltage applied to the field and armature coils, respectively and \(T_L\) is the torque applied to the motor. (If \(T_L < 0\), then the motor is applying the torque.) All of the other terms are assumed to be constant parameters of the motor. Due to the derivative terms of (2.13) it is clear that the state vector should be

\[
    s = \begin{bmatrix}
        s^{(1)} \\
        s^{(2)} \\
        s^{(3)}
    \end{bmatrix} = \begin{bmatrix}
        i_f \\
        i_a \\
        \omega
    \end{bmatrix}
\]

(2.14)
where $s^{(i)}$ simply denotes the $i^{th}$ element of the vector $s$. Once again determination of the manipulated and disturbance variables is application dependent. Assume the field and armature voltages are at the service of the controller. In addition, assume the torque will change over time, but the controller has no influence on the amount of torque applied to the motor. These assumptions yield the following definitions.

$$m = \begin{bmatrix} m^{(1)} \\ m^{(2)} \end{bmatrix} = \begin{bmatrix} v_f \\ v_a \end{bmatrix} \quad p = [T_L]$$  \hspace{1cm} (2.15)

Then, the vector function $f$ is expressed as

$$f(s, m, p) = \begin{bmatrix} \frac{1}{L_f} (-R_f s^{(1)} + m^{(1)}) \\ \frac{1}{L_a} (-R_a s^{(2)} - K_v s^{(3)} s^{(1)} + m^{(2)}) \\ \frac{1}{J} (-B s^{(3)} + K_v s^{(2)} s^{(1)} + p) \end{bmatrix}$$  \hspace{1cm} (2.16)

Assume there are two indicators of process performance; total electrical and mechanical power sent to the motor. If the two power signals are defined as $P_e = i_f v_f + i_a v_a$, and $P_m = T_L \omega$, then $h$ should be defined as:

$$h(s, m, p) = \begin{bmatrix} s^{(1)} m^{(1)} + s^{(2)} m^{(2)} \\ p s^{(3)} \end{bmatrix}$$  \hspace{1cm} (2.17)

$$h(s, m, p) = \begin{bmatrix} s^{(1)} m^{(1)} + s^{(2)} m^{(2)} \\ p s^{(3)} \end{bmatrix}$$  \hspace{1cm} (2.17)

**Example 2.3.** Consider the following second order differential equation describing a mass-spring damper system (depicted in Figure 2.4).

$$M \frac{d^2 r}{dt^2} = -b \frac{dr}{dt} - k r + f_a + f_d + f_g$$  \hspace{1cm} (2.18)

where $r$ is the position of the mass, $f_a$ and $f_d$ are forces applied to the mass (where $f_a$ is under the influence of the controller and $f_d$ is not) and $M$, $b$, $k$ and $f_g$ are constant parameters. In this system, the appearance of a second order derivative term seems to preclude connection with equation (2.1). However, if we simply define a new variable
$v = dr/dt$, then the following system of differential equations will arise:

\[
\frac{dr}{dt} = v \\
\frac{dv}{dt} = \frac{1}{M} \left(-br + fr_a + f_d + f_g\right)
\] (2.19) (2.20)

From this point it is clear that one should select the state, manipulated and disturbance vectors as:

\[
s = \begin{bmatrix} s^{(1)} \\ s^{(2)} \end{bmatrix} = \begin{bmatrix} r \\ v \end{bmatrix} \quad m = [f_a] \quad p = [f_d]
\] (2.21)

Then, the vector function $f$ should be expressed as:

\[
f(s, m, p) = \begin{bmatrix} s^{(2)} \\ \frac{1}{M} \left(-bs^{(2)} - ks^{(1)} + m + p + f_g\right) \end{bmatrix}
\] (2.22)

As indicated in Figure 2.4, the position of the mass is restricted, which may be expressed as $r_{\text{min}} \leq r \leq r_{\text{max}}$. Assume similar restrictions will be enforced for the manipulated variable ($f_{a,\text{min}} \leq f_a \leq f_{a,\text{max}}$) as well as for mass acceleration ($a_{\text{min}} \leq a \leq a_{\text{max}}$) where $a$ is defined as:

\[
a = \frac{1}{M} \left(-br + fr_a + f_d + f_g\right)
\] (2.23)

In this case, we should define the performance vector, $q$, as:

\[
q = \begin{bmatrix} q^{(1)} \\ q^{(2)} \\ q^{(3)} \end{bmatrix} = \begin{bmatrix} r \\ f_a \\ a \end{bmatrix}
\] (2.24)

Then, the vector function $h$ should be defined as:

\[
h(s, m, p) = \begin{bmatrix} s^{(1)} \\ m \\ \frac{1}{M} \left(-bs^{(2)} - ks^{(1)} + m + p + f_g\right) \end{bmatrix}
\] (2.25)

We can now express the inequality restrictions in the following compact form

\[
q_{\text{min}} \leq q \leq q_{\text{max}}
\] (2.26)

where $q_{\text{min}}$ and $q_{\text{max}}$ are defined as:

\[
q_{\text{min}} = \begin{bmatrix} r_{\text{min}} \\ f_{a,\text{min}} \\ a_{\text{min}} \end{bmatrix} \quad q_{\text{max}} = \begin{bmatrix} r_{\text{max}} \\ f_{a,\text{max}} \\ a_{\text{max}} \end{bmatrix}
\] (2.27)
2.2. Deviation Variables

In many cases it will be desired to operate the process near a steady-state operating condition. In general, a steady-state operating point (SSOP) is defined by the nonlinear state-space model evaluated with the time derivative terms set equal to zero:

\[ 0 = f(s_{SSOP}, m_{SSOP}, p_{SSOP}) \]  
\[ q_{(SSOP)} = h(s_{SSOP}, m_{SSOP}, p_{SSOP}) \]  
\[ \theta_{(SSOP)} = l(s_{SSOP}, n_{SSOP}) \]  

(2.28)  
(2.29)  
(2.30)

where the triple \((s_{SSOP}, m_{SSOP}, p_{SSOP})\) is any set of vector values satisfying equation (2.28). If the following deviation variables are defined:

\[ x = s - s_{SSOP} \]  
\[ u = m - m_{SSOP} \]  
\[ w = p - p_{SSOP} \]  
\[ z = q - q_{SSOP} \]  
\[ y = \theta - \theta_{SSOP} \]  
\[ v = n - n_{SSOP} \]  

(2.31)  
(2.32)  
(2.33)  
(2.34)  
(2.35)  
(2.36)

then, the following deviation variable model can be constructed:

\[ \dot{x} = f(x + s_{SSOP}, u + m_{SSOP}, w + p_{SSOP}) \]  
\[ z = h(x + s_{SSOP}, u + m_{SSOP}, w + p_{SSOP}) - q_{SSOP} \]  
\[ y = l(x + s_{SSOP}, v + n_{SSOP}) - \theta_{SSOP} \]  

(2.37)  
(2.38)  
(2.39)

The advantage of the deviation variable form is that the desired condition of \((s, m, p, q, \theta, n) = (s_{SSOP}, m_{SSOP}, p_{SSOP}, q_{SSOP}, \theta_{SSOP}, n_{SSOP})\) is achieved when \((x, u, w, z, y, v) = (0, 0, 0, 0, 0, 0)\).

Example 2.4. Reconsider the mass-spring damper system of Example 2.3 and assume a SSOP triple \((s_{SSOP}, m_{SSOP}, p_{SSOP})\) has been identified. Specifically, if \(p_{SSOP}\) is given and \(m_{SSOP}\) is arbitrarily selected, then by using the relations (2.28)-(2.29), \(s_{SSOP}\) and \(q_{SSOP}\) are calculated to be

\[ s_{SSOP} = \begin{bmatrix} 1 \kappa (m_{SSOP} + p_{SSOP} + f_g) \\ 0 \end{bmatrix} \]  
\[ q_{SSOP} = \begin{bmatrix} 1 \kappa (m_{SSOP} + p_{SSOP} + f_g) \\ m_{SSOP} \\ 0 \end{bmatrix} \]  

(2.40)  
(2.41)

If we now employ the definitions of (2.31)-(2.33) and apply (2.37) to arrive at the following
deviation variable model of the state equation:

\[
\dot{x} = \begin{bmatrix}
\frac{1}{M} \left( -b(x(2) + s^{(2)}_{SSOP}) - k(x(1) + s^{(1)}_{SSOP}) 
+ (u + m_{SSOP}) + (w + p_{SSOP}) + f_g \right) \\
\frac{1}{M} \left( -b x(2) - k x(1) + u + w 
- k s^{(1)}_{SSOP} + m_{SSOP} + p_{SSOP} + f_g \right) \\
\frac{1}{M} \left( -b x(2) - k x(1) + u + w \right)
\end{bmatrix}
\]

(2.42)

where the equalities utilize the identities of (??). Application of the definitions of (2.31)-(2.34) to (2.38) yields:

\[
z = \begin{bmatrix}
x^{(1)} + s^{(1)}_{SSOP} \\
u + m_{SSOP} \\
- b(x(2) + s^{(2)}_{SSOP}) \\
-k(x(1) + s^{(1)}_{SSOP}) \\
+ (u + m_{SSOP}) + (w + p_{SSOP}) + f_g \end{bmatrix} - \begin{bmatrix}
x^{(1)}_{SSOP} \\
m_{SSOP} \\
-b s^{(2)}_{SSOP} \\
-k s^{(1)}_{SSOP} + m_{SSOP} + p_{SSOP} + f_g \end{bmatrix}
\]

(2.43)

Equations (2.42)-(2.43) can be further manipulated into the following form:

\[
\dot{x} = Ax + Bu + Gw
\]

(2.44)

\[
z = Dx + Du u + Dw w
\]

(2.45)

where the matrices \(A, B, G, D_x, D_u,\) and \(D_w\) are defined as:

\[
A = \begin{bmatrix}
0 & 1/M \\
-k/M & -b/M
\end{bmatrix} \quad B = \begin{bmatrix}
0 \\
1/M
\end{bmatrix} \quad G = \begin{bmatrix}
0 \\
1/M
\end{bmatrix}
\]

(2.46)

\[
D_x = \begin{bmatrix}
1 & 0 \\
-k/M & -b/M
\end{bmatrix} \quad D_u = \begin{bmatrix}
0 \\
1/M
\end{bmatrix} \quad D_w = \begin{bmatrix}
0 \\
1/M
\end{bmatrix}
\]

(2.47)

The following two definitions indicate when a state-space model will be within the special class of linear models.

**Definition 2.1.** A function \(f(s)\) is linear with respect to \(s\), if the following conditions hold:
(i) \( f(0) = 0 \)
(ii) \( f(\alpha s) = \alpha f(s) \) for all \( s \) and all scalars \( \alpha \)
(iii) \( f(s_1 + s_2) = f(s_1) + f(s_2) \) for all \( s_1 \) and \( s_2 \)

**Definition 2.2.** A state-space process model (2.1)-(2.3) is linear if \( f(s, m, p) \), \( h(s, m, p) \) and \( l(s, n) \) are all linear with respect to all three variables \( s, m, p \) and \( n \).

**Example 2.5.** Reconsider the deviation variable model of the mass-spring damper system developed in Example 2.4. If Equations (2.44)-(2.45) are expressed as:

\[
\begin{align*}
\dot{x} &= f(x, u, w) \\
z &= h(x, u, w)
\end{align*}
\]

then, \( f(x, u, w) = Ax + Bu + Gw \) and \( h(x, u, w) = Dx + Du + Dw \). Then, one can easily verify that the functions \( f \) and \( h \) satisfy Definition 2.2. If one looks back to the original mass-spring-damper model of Example 2.3, it is easily observed that the second term of Equation (2.16) does not satisfy Definition 2.2 (due to the constant term \( f_e \)). Fortunately, the conversion to deviation variables yields a linear form.

### 2.3 Linearization of Nonlinear Models

In most cases the conversion to deviation variable form will not yield a linear model. In these cases one will need to resort to linearization. The method of linearization is simply an application of the Taylor series expansion. Specifically, a function \( f(s) \) is equal to the following expression:

\[
f(s) = f(s_0) + \frac{\partial f}{\partial s}
|_{s=s_0} (s - s_0) + \frac{1}{2} \frac{\partial^2 f}{\partial s^2}
|_{s=s_0} (s - s_0)^2 + \cdots
\]

(2.50)

The basic idea is that use of a finite number of terms will yield a “good” approximation of \( f(s) \) for values of \( s \) that are “close” to \( s_0 \). The quotes around good and close are intended to indicate the qualitative nature of this statement. In practice, it will be important to validate this statement quantitatively. To arrive at a linear approximation of \( f(s) \) one should retain only the first two terms of the expansion. In the vector case, one finds that a function \( f(s, m, p) \) can be approximated by the following expression:

\[
f(s, m, p) \approx f(s_0, m_0, p_0) + \frac{\partial f}{\partial s}
|_{s=s_0, m=m_0, p=p_0} (s - s_0)
+ \frac{\partial f}{\partial m}
|_{s=s_0, m=m_0, p=p_0} (m - m_0) + \frac{\partial f}{\partial p}
|_{s=s_0, m=m_0, p=p_0} (p - p_0)
\]

(2.51)

If the triple \((s_0, m_0, p_0)\) is selected to be \((s^{SSOP}, m^{SSOP}, p^{SSOP})\), then \( f(s^{SSOP}, m^{SSOP}, p^{SSOP}) = 0 \) can be applied along with the definitions of (2.31)-(2.33) to find:

\[
f(s, m, p) \approx \frac{\partial f}{\partial s}
|_{s=s^{SSOP}, m=m^{SSOP}, p=p^{SSOP}} (s - s^{SSOP}) + \frac{\partial f}{\partial m}
|_{s=s^{SSOP}, m=m^{SSOP}, p=p^{SSOP}} (m - m^{SSOP})
+ \frac{\partial f}{\partial p}
|_{s=s^{SSOP}, m=m^{SSOP}, p=p^{SSOP}} (p - p^{SSOP})
\]

(2.52)
Since the partial derivatives are evaluated at the SSOP, each of these coefficients will be constants. Then, recalling that the time derivative of $s$ is equal to the time derivative of $x$, yields 
\[ \dot{x} = Ax + Bu + Gw \]
as linear deviation variable approximation of (2.1), where

\[
A = \frac{\partial f}{\partial s} \bigg|_{s=SSOP \atop m=m_{SSOP} \atop p=p_{SSOP}} \quad B = \frac{\partial f}{\partial m} \bigg|_{m=m_{SSOP} \atop p=p_{SSOP}} \quad G = \frac{\partial f}{\partial p} \bigg|_{m=m_{SSOP} \atop p=p_{SSOP}}
\]

Similar use of the Taylor expansion on the performance output $q = h(s, m, p)$ gives

\[
h(s, m, p) \approx h(s_o, m_o, p_o) + \frac{\partial h}{\partial s} \bigg|_{s=s_o \atop m=m_o \atop p=p_o} (s - s_o) + \frac{\partial h}{\partial m} \bigg|_{s=s_o \atop m=m_o \atop p=p_o} (m - m_o) + \frac{\partial h}{\partial p} \bigg|_{s=s_o \atop m=m_o \atop p=p_o} (p - p_o) \tag{2.53}
\]

If the triple $(s_o, m_o, p_o)$ is selected to be $(s_{SSOP}, m_{SSOP}, p_{SSOP})$, then $q_{SSOP} = h(s_{SSOP}, m_{SSOP}, p_{SSOP})$ can be applied along with the definitions of (2.31)-(2.33) to find:

\[
q \approx q_{SSOP} + \frac{\partial h}{\partial s} \bigg|_{s=s_{SSOP} \atop m=m_{SSOP} \atop p=p_{SSOP}} x + \frac{\partial h}{\partial m} \bigg|_{s=s_{SSOP} \atop m=m_{SSOP} \atop p=p_{SSOP}} u + \frac{\partial h}{\partial p} \bigg|_{s=s_{SSOP} \atop m=m_{SSOP} \atop p=p_{SSOP}} w \tag{2.54}
\]

Thus, $z = q - q_{SSOP} \approx D_x x + D_u u + D_w w$ where

\[
D_x = \frac{\partial h}{\partial s} \bigg|_{s=s_{SSOP} \atop m=m_{SSOP} \atop p=p_{SSOP}} \quad D_u = \frac{\partial h}{\partial m} \bigg|_{s=s_{SSOP} \atop m=m_{SSOP} \atop p=p_{SSOP}} \quad D_w = \frac{\partial h}{\partial p} \bigg|_{s=s_{SSOP} \atop m=m_{SSOP} \atop p=p_{SSOP}} \tag{2.55}
\]

If the physical measurements possess only additive disturbances, i.e., the measurement equation is of the form $\theta = \theta(s) + n$, then $\theta_{SSOP} = \theta(s_{SSOP}) + n_{SSOP}$ can be applied along with (2.31), (2.35)-(2.36) to find $y = \theta - \theta_{SSOP} \approx C x + v$ where $C = \frac{\partial \theta}{\partial s} \bigg|_{s=s_{SSOP}}$.

**Figure 2.5. Simulations of CSTR model of Example 2.6**
Example 2.6. Reconsider the CSTR process of Example 2.1. Assume the volume of the reactor is 3 m$^3$ and rate constant is 0.1 m$^3$ per mole per sec. If the nominal inlet concentration is 1 mole per m$^3$ and the exit concentration is desired to be 0.3 moles per m$^3$, then (based on a steady-state version of Equation (2.5)) the volumetric flow rate should be selected as 0.0771 m$^3$ per sec. Thus, the SSOP triple is determined to be $(s^{SSOP}, m^{SSOP}, p^{SSOP}) = (0.3, 0.0771, 1)$. If $f(s, m, p) = \frac{m}{V} p - \frac{m}{V} s - 2ks^2$, then

\[
A = \frac{\partial f}{\partial s} \big|_{s=s^{SSOP}, m=m^{SSOP}, p=p^{SSOP}} = \left( -\frac{m}{V} - 4ks \right) \bigg|_{s=0.3, m=0.0771, p=1} = -0.146 \quad (2.56)
\]

\[
B = \frac{\partial f}{\partial m} \bigg|_{s=s^{SSOP}, m=m^{SSOP}, p=p^{SSOP}} = \left( \frac{p}{V} - \frac{s}{V} \right) \bigg|_{s=0.3, m=0.0771, p=1} = 0.233 \quad (2.57)
\]

\[
G = \frac{\partial f}{\partial p} \bigg|_{s=s^{SSOP}, m=m^{SSOP}, p=p^{SSOP}} = \left( \frac{m}{V} \right) \bigg|_{s=0.3, m=0.0771, p=1} = 0.0257 \quad (2.58)
\]

Let us now compare the simulations of the CSTR process using the nonlinear model and the linearized model. Assume the initial condition is $C_A(0) = 0.2$ mole per m$^3$. Furthermore, assume $C_{A_{in}} = 1$ for $t = [0 \ 50], C_{A_{in}} = 0.5$ for $t = [50 \ 150], \nu = 0.771$ for $t = [0 \ 100]$, and $\nu = 1.5$ for $t = [100 \ 150]$. Using the MATLAB code of Tables 2.1 and 2.2, the plot of Figure 2.5 was generated. While the two simulations are not identical, they do possess a great deal of similarity.

**Table 2.1: MATLAB code used in calculations for Example 2.6**

```matlab
clear
V=3;k=0.1;CInbar=1;Cbar=0.3;
nubar=2*k*Cbar^2*V/(CInbar-Cbar); x0=0.2;
% Calculate solution with nonliner ODE
linearize='n';
[tt1,xxodel]=ode23(@(t,x) cstr(t,x,linearize,V,k,Cbar,nubar,CInbar),...[0 150],x0);
% Calculate solution with linearized ODE
linearize='y';
[tt2,xxode2]=ode23(@(t,x) cstr(t,x,linearize,V,k,Cbar,nubar,CInbar),...[0 150],x0);
% Plot the results
plot(tt1,xxodel,'-*b',tt2,xxode2,'-*k','MarkerSize',8)
legend('Nonlinear Solution','Linearized Solution')
ylabel('Concentration of A'), xlabel('time')
```

**Table 2.2: MATLAB code used in calculations for Example 2.6**

```matlab
function dsdt = cstr(t,s,linearize,V,k,Cbar,nubar,CInbar)
% This function should be in a file named 'cstr.m'
CA=s(1); nu=nubar; CAin=CInbar;
if t > 50 CAin = 0.5, end
if t > 100 nu=0.15, end
if linearize=='y'
    A=-nubar/V-4*k*Cbar, B=(CInbar-Cbar)/V, G=nubar/V,
x=CA-Cbar; u=nu-nubar; w=CAin-CInbar;
    dsdt=A*x+B*u+G*w;
else
    dsdt=CAin*nu/V-CA*nu/V-2*k*CA^2;
end
```
Example 2.7. Reconsider the DC motor model of Example 2.2. Assume a SSOP triple \((s^{SSOP}, m^{SSOP}, p^{SSOP})\) satisfying (2.28) has been determined and the deviation variable definitions of (2.31)-(2.36) are to be used. Recall the nonlinear state-space model

\[
\begin{align*}
L_f \frac{di_f}{dt} &= -R_f i_f + v_f \\
L_a \frac{di_a}{dt} &= -R_a i_a - K_v \omega + v_a \\
J \frac{d\omega}{dt} &= -B \omega + K_v i_f + T_L
\end{align*}
\]  

Along with the performance outputs \(P_e = i_f v_f + i_a v_a\), and \(P_m = T_L \omega\). Then, a linear state-space model approximating the original is found to be of the form:

\[
\dot{x} = Ax + Bu + Gw
\]

\[
z = Dx + Du + Dw
\]

where

\[
A = \left. \frac{\partial f}{\partial s} \right|_{s=s^{SSOP} \atop m=m^{SSOP} \atop p=p^{SSOP}} = \begin{bmatrix}
\frac{-R_f}{L_f} & 0 & 0 \\
\frac{-K_v}{L_a} & -\frac{R_a}{L_a} & -\frac{K_v i_f}{L_a} \\
0 & -\frac{B}{J} & 0
\end{bmatrix}
\]  

\[
B = \left. \frac{\partial f}{\partial m} \right|_{s=s^{SSOP} \atop m=m^{SSOP} \atop p=p^{SSOP}} = \begin{bmatrix}
1/L_f & 0 & 0 \\
0 & 1/L_a & 0 \\
0 & 0 & 1/J
\end{bmatrix}
\]  

\[
G = \left. \frac{\partial f}{\partial p} \right|_{s=s^{SSOP} \atop m=m^{SSOP} \atop p=p^{SSOP}} = \begin{bmatrix}
0 \\
0 \\
1/J
\end{bmatrix}
\]  

\[
D_x = \left. \frac{\partial h}{\partial s} \right|_{s=s^{SSOP} \atop m=m^{SSOP} \atop p=p^{SSOP}} = \begin{bmatrix}
v_f^{SSOP} & v_a^{SSOP} & 0 \\
0 & 0 & 0 \\
0 & 0 & T_L^{SSOP}
\end{bmatrix}
\]  

\[
D_u = \left. \frac{\partial h}{\partial m} \right|_{s=s^{SSOP} \atop m=m^{SSOP} \atop p=p^{SSOP}} = \begin{bmatrix}
i_f^{SSOP} & i_a^{SSOP} & 0 \\
0 & 0 & 0
\end{bmatrix}
\]  

\[
D_w = \left. \frac{\partial h}{\partial p} \right|_{s=s^{SSOP} \atop m=m^{SSOP} \atop p=p^{SSOP}} = \begin{bmatrix}
\omega^{SSOP}
\end{bmatrix}
\]

Example 2.8. Reconsider the DC motor model of Example 2.7. Assume the system has the following parameters: \(L_f = 50 \text{H} \), \(R_f = 25 \Omega\), \(L_a = 2 \text{H}\), \(R_a = 0.1 \Omega\), \(R_{int} = 0.4 \Omega\), \(Kv = 0.91 V/(A \text{ rad}/s)\), \(J = 5 \text{Nm}\) and \(B = 0.3 \text{Nm}/(\text{rad}/s)\).

(i) Assume the following nominal conditions are required: \(m^{SSOP} = [36 V \ 360 V]^T\) and \(p^{SSOP} = [-280 \text{Nm}]\). Determine \(s^{SSOP}\) and \(q^{SSOP}\).

(ii) Linearize the nonlinear model around the SSOP of part (i).

(iii) Using a MATLAB ODE solver, compare the linearized model solution with that of the original nonlinear model. Assume the initial condition is the SSOP and \(v_a\)
and \( T_t \) remain at their nominal values for all time and consider the following two scenarios:

Case 1: \( v_f = 36 \text{ V} \) for \( t = [0, 10] \), \( 38 \text{ V} \) for \( t = [10, 50] \) and \( 34 \text{ V} \) for \( t = [50, 100] \).

Case 2: \( v_f = 36 \text{ V} \) for \( t = [0, 10] \), \( 56 \text{ V} \) for \( t = [10, 50] \) and \( 26 \text{ V} \) for \( t = [50, 100] \).

Solution: Solving the first steady state equation yields:

\[
i_{SSOP} = \frac{v_{SSOP}}{R_f}.
\]

Substitution into the third yields:

\[
\omega = \frac{(K_v i_{SSOP}^f v_{SSOP}^f / R_f + T_{SSOP}^L)}{B}.
\]

Substitution of both into the second steady state equation yields:

\[
i_{SSOP}^a = v_{SSOP}^a - K_v T_{SSOP}^a v_{SSOP}^f / (R_f B)
\]

Using these formula one finds:

\[
\begin{bmatrix}
i_{SSOP}^f \\
i_{SSOP}^a \\
\omega_{SSOP}
\end{bmatrix} = \begin{bmatrix}
1.44 A \\
254 A \\
178 \text{ rad/s}
\end{bmatrix}
\]

\[
q_{SSOP} = \begin{bmatrix}
p_{SSOP}^e \\
p_{SSOP}^m
\end{bmatrix} = \begin{bmatrix}
91.2 \text{ kW} \\
49.7 \text{ kW}
\end{bmatrix}
\]

Using Equations (2.62)-(2.65) one finds

\[
A = \frac{\partial f}{\partial s} = \begin{bmatrix}
-0.50 & 0 & 0 \\
-80.8 & -0.250 & -0.655 \\
46.3 & 0.262 & -0.060
\end{bmatrix}
\]

\[
B = \frac{\partial f}{\partial m} = \begin{bmatrix}
0.02 & 0 & 0 \\
0 & 0.5 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
G = \frac{\partial f}{\partial p} = \begin{bmatrix}
0 & 0
\end{bmatrix}
\]

\[
D_x = \frac{\partial h}{\partial s} = \begin{bmatrix}
36 & 360 & 0 \\
0 & 0 & -280
\end{bmatrix}
\]

\[
D_u = \frac{\partial h}{\partial m} = \begin{bmatrix}
1.44 & 254 \\
0 & 0
\end{bmatrix}
\]

\[
D_w = \frac{\partial h}{\partial p} = [177]
\]

The plots of Figures 2.6 and 2.7 were made with the code of Tables 2.3, 2.4 and 2.5. Notice that in case 1 the discrepancies are quite small, but are much larger in case 2. This is due to the fact that in case 2 operation is much further from the point of linearization. Also, note that since the first equation is liner, there is no discrepancy in either of the field current plots.
Figure 2.6. Simulations of the State Variables for DC motor of Example 2.8

Table 2.3: Top level MATLAB code for Example 2.8

```matlab
clear all

% Motor Parameters
Lf=50; %H
Rf=25; %ohms
La=2; %H
Rint=0.4; %ohms
Ra=0.1; %ohms
Kv=0.91; %V/(A rad/s)
J=5; %Nm
B=0.3; %J/(rad/s)
% Inputs at Steady-State
Vf=36; %V
```
2.3. Linearization of Nonlinear Models

Figure 2.7: Simulations of the Output Variables for DC-motor of Example 2.8

Va=360; %V
TL=-280; %Nm=J
%Steady-state Calcs
if_st=[Vf/Rf];
lalive=inv([((Ra+Rint) Kv*if_ss;-Kv*if_ss B)]*[Va;TL];
ia_ss=(lalive(1)); w_ss=alive(2);
s_ss=[if_ss; ia_ss; w_ss];
q_ss=[if_ss*Vf+ia_ss*Va; w_ss*TL]

[t_nlin, s_nlin]=ode45('motor_mod_ode', [0 100], s_ss);
[t_lin, x_lin]=ode45('motor_mod_ode_lin', [0 100], s_ss-s_ss);
[NN, dumb]=size(t_nlin); q_nlin=zeros(NN,2);
for ii=1:NN
    [dsdt, qout]=motor_mod_ode(t_nlin(ii), s_nlin(ii,:));
    q_nlin(ii,:)=qout';
end

[NN, dumb]=size(t_lin); z_lin=zeros(NN,2);
for ii=1:NN
    [dxdt, zout]=motor_mod_ode_lin(t_lin(ii), x_lin(ii,:));
    z_lin(ii,:)=zout';
end

plot(t_nlin, s_nlin(:,1), 'k', t_lin, x_lin(:,1)+s_ss(1), 'k--');
title('Case 1', 'FontSize', 14, 'FontName', 'Times New Roman');
ylabel('Field Current (A)', 'FontSize', 14, 'FontName', 'Times New Roman');
xlabel('Time (s)', 'FontSize', 14, 'FontName', 'Times New Roman');
legend('Nonlinear', 'Linear'); pause
plot(t_nlin, s_nlin(:,2), 'k', t_lin, x_lin(:,2)+s_ss(2), 'k--');
title('Case 1', 'FontSize', 14, 'FontName', 'Times New Roman');
Table 2.4: Nonlinear model for Example 2.8

```matlab
function [dsdt, qout] = motor_mod_ode(t, s)
    % Motor Parameters
    Lf=50; % H
    Rf=25; % ohms
    La=2; % H
    Rint=0.4; % ohms
    Ra=0.1; % ohms
    Kv=0.91; % V/(A rad/s)
    J=5; % Nm
    B=0.3; % J/(rad/s)

    % Inputs at Steady-State
    m1=36; % V
    m2=360; % V
    p=-280; % Nm

    Case=1;
    if (Case == 1) % Case 1
        if t > 10; m1=38; end
        if t > 50; m1=34; end
    else % Case 2
        if t > 10; m1=56; end
        if t > 50; m1=26; end
    end
    ds1dt=( - Rf *s(1) + m1 )/Lf;
    ds2dt=( - (Ra+Rint) *s(2) - Kv *s(3)*s(1) + m2 )/La;
    ds3dt=( - B *s(3) + Kv *s(1)*s(2) + p )/J;
    dsdt=[ds1dt;ds2dt;ds3dt];
    qout=[s(1)*m1+s(2)*m2; s(3) *p];
```

Table 2.5: Linear model for Example 2.8

```matlab
function [dxdt, zout] = motor_mod_ode_lin(t, x)
    AA=[-0.5 0 0; ... 
        -80.8 -0.25 -0.655; ... 
        46.3 0.262 -0.06];
    BB=[ 0.02 0; 0 0.5; 0 0];
    GG=[ 0; 0; 0.2];
    Dx=[36 360 0; 0 0 -280];
    Du=[1.44 254; 0 0];
    Dw=[177];
    u=[0; 0];
    w=0;
    Case=1;
    if (Case == 1) % Case 1
if \( t > 10 \); \( u=[2; 0] \); end
if \( t > 50 \); \( u=[-2; 0] \); end
else
  \%
  Case 2
  if \( t > 10 \); \( u=[20; 0] \); end
  if \( t > 50 \); \( u=[-10; 0] \); end
end
dxdt=AA*x+ BB *u+GG*w;
zout=Dx*x+Du*u+Dw*w;

In summary, the linearization process will result in the following linear process model:

\[
\dot{x} = Ax + Bu + Gw \quad (2.68)
\]
\[
z = Dx \dot{x} + Du u + Dw w \quad (2.69)
\]
\[
y = C x + v \quad (2.70)
\]

Note that all of the system matrices \((A, B, G, Dx, Du, Dw, \text{ and } C)\) will be constant. The dimension of these system matrices will depend on the size of the vector signals. Throughout the text the following notation will be used to indicate size of these vectors:

\[
x \text{ (and } s) \sim n_x \times 1 \quad (2.71)
\]
\[
u \text{ (and } m) \sim n_u \times 1 \quad (2.72)
\]
\[
w \text{ (and } p) \sim n_w \times 1 \quad (2.73)
\]
\[
z \text{ (and } q) \sim n_z \times 1 \quad (2.74)
\]
\[
y \text{ (and } \theta) \sim n_y \times 1 \quad (2.75)
\]
\[
v \text{ (and } n) \sim n_v \times 1 \quad (2.76)
\]

Based on these definition the dimensions of the system matrices are as follows:

\[
A \sim n_x \times n_x \quad (2.77)
\]
\[
B \sim n_x \times n_u \quad (2.78)
\]
\[
G \sim n_x \times n_w \quad (2.79)
\]
\[
D_x \sim n_z \times n_x \quad (2.80)
\]
\[
D_u \sim n_z \times n_u \quad (2.81)
\]
\[
D_w \sim n_z \times n_w \quad (2.82)
\]
\[
C \sim n_y \times n_x \quad (2.83)
\]

### 2.4 Analytic Solution of a Linear State Space Model

The primary benefit of the linear model approximation is the ability to apply analytic methods in analyzing the process. The first example of these methods is in obtaining a closed-form solution of a system of linear differential equations. Before giving the matrix form of the solution, the following scalar case will illustrate the steps. Consider a first order differential equation \( \dot{x} = ax + bu \) with initial condition \( x(t_o) = x_o \). Assuming \( a \) and \( b \) are scalar constants, determine \( x(t) \) for any signal \( u(t) \). Begin by defining a new signal \( \ddot{x}(t) \) such that \( \ddot{x}(t) = e^{-a(t-t_o)}x(t) \). Then substitution into the original differential equation yields \( \ddot{x}(t) = e^{-a(t-t_o)}bu(t) \), which can be solved by separation of variables to yield:

\[
\ddot{x}(t) - \ddot{x}(t_o) = \int_{t_o}^{t} e^{-a(t-\tau)}b u(\tau)d\tau \quad (2.84)
\]
which is equivalent to 

\[ e^{-a(t-t_0)}x(t) - x_0 = \int_{t_0}^{t} e^{-a(\tau-t_0)} b u(\tau)d\tau \]

and finally yields:

\[ x(t) = e^{a(t-t_0)}x_0 + \int_{t_0}^{t} e^{a(t-\tau)} b u(\tau)d\tau \] (2.85)

In the more general case of \( \dot{x} = Ax + Bu + Gw \) where \( x, u, \) and \( w \) are vector functions and \( A, B \) and \( G \) are constant matrices, the central challenge is to interpret the exponential term when the scalar \( a \) is replaced by the matrix \( A \).

**Definition 2.3.** Given a square matrix \( A \), the exponential of that matrix is defined as

\[ e^{At} = I + At + \frac{1}{2}(At)^2 + \frac{1}{3}(At)^3 + \cdots \]

**Definition 2.4.** If a matrix \( A \) has elements \( a_{ij} \) at row \( i \) and column \( j \), then the transpose of \( A \), denoted \( A^T \), has elements \( a_{ji} \) at row \( i \) and column \( j \).

**Definition 2.5.** If a matrix \( A \) has an inverse, denoted \( A^{-1} \), then \( A^{-1} \) is such that

\[ AA^{-1} = A^{-1}A = I \]

**Corollary 2.1.** The exponential of a matrix has the following properties (all of which are easily verified from Definition 2.3).

\[ \frac{d}{dt}e^{At} = Ae^{At} = e^{At}A \]

\[ e^{A(t_1+t_2)} = e^{At_1}e^{At_2} \] (2.86)

\[ (e^{At})^{-1} = e^{-At} \]

\[ (e^{At})^T = e^{A^Tt} \] (2.87)

Chapter 3 will provide a method for determining a closed form (non-series) expression for \( e^{At} \). However, at this point it is sufficient to simply know that this term can be evaluated for any finite value of \( t \). In particular, the function `expm` in MATLAB will calculate this term with enough accuracy and speed for our current purposes. Thus, if one follows the same steps as the scalar case, it is found that the solution to \( \dot{x} = Ax + Bu + Gw \) is:

\[ x(t) = e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-\tau)} Bu(\tau)d\tau + \int_{t_0}^{t} e^{A(t-\tau)} Gw(\tau)d\tau \] (2.88)

Then, the output signals are simply calculated as

\[ z(t) = D_x x(t) + D_u u(t) + D_w w(t) \] (2.89)

\[ y(t) = C x(t) + v(t) \] (2.90)

where \( x(t) \) is from (2.88).
2.5 • Discrete-time Models

The discrete-time framework is one of convenience from a simulation standpoint and will be of great utility from a practical implementation standpoint. Concerning dynamic simulation, the recursive form of the discrete-time model will greatly simplify the coding effort - ode solvers can be replaced by simple loops. Concerning implementation, all modern controllers are computer implemented and at some level must operate with respect to sampled time intervals. As such, a discrete-time model is likely to be a more accurate representation of the process under the influence of a time-sampled controller. From a theoretical standpoint, nearly all results developed for the continuous-time case will have a parallel relation in the discrete-time framework. Throughout the text, design formulas will be provided for both cases, however to illustrate the development of these relations a single perspective will be employed, typically the one requiring the least technical detail.

In general, a discrete-time model will be of the following recursive form:

\[ s_{k+1} = f_d(s_k, m_k, p_k) \]  \hspace{1cm} (2.91)
\[ q_k = h(s_k, m_k, p_k) \] \hspace{1cm} (2.92)
\[ \theta_k = l(s_k, n_k) \] \hspace{1cm} (2.93)

The term recursive is used to emphasize that the state at the next time-step, \( s_{k+1} \), can be calculated if given the current state, \( s_k \), and the current input values, \( m_k \) and \( p_k \).

![Figure 2.8. Single-echelon inventory system of Example 2.9](image)

**Example 2.9.** Consider the following model of product inventory at a retail location:

\[ l_{k+1} = l_k + r_{k-\theta} - d_k \] \hspace{1cm} (2.94)

where \( l_k \) is the inventory at the end of day \( k \), \( d_k \) is the product sold during day \( k \) (denoted as demand) and \( r_k \) is the amount of product ordered at the end of day \( k \) (denoted as starts or reorders). To capture delivery delay, \( r_{k-\theta} \) is the amount ordered \( \theta \) days earlier and arriving at the beginning of day \( k \). This arrangement is depicted in Figure 2.8. Consider the case of \( \theta = 3 \). To put the system (2.94) into the form of (2.91) begin by defining state,
manipulated and disturbance vectors as follows:

\[
\begin{bmatrix}
s_k^{(1)} \\
s_k^{(2)} \\
s_k^{(3)} \\
s_k^{(4)} \\
\end{bmatrix} =
\begin{bmatrix}
I_k \\
r_{k-3} \\
r_{k-2} \\
r_{k-1} \\
\end{bmatrix}
\]

\[m_k = [r_k] \quad p_k = [d_k]\] \hspace{1cm} (2.95)

then \( f_d \) can be expressed as: \( f_d(s_k, m_k, p_k) = A_d s_k + B_d m_k + G_d p_k \) where

\[
A_d = \begin{bmatrix}
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\end{bmatrix} \quad B_d = \begin{bmatrix}
0 \\
0 \\
0 \\
1 \\
\end{bmatrix} \quad G_d = \begin{bmatrix}
-1 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\] \hspace{1cm} (2.96)

The inventory and reorders are both restricted to be positive and must be less than maximum values, \( I_{\text{max}} \) and \( r_{\text{max}} \). As such, the performance output should be selected as

\[
q_k = \begin{bmatrix}
q_k^{(1)} \\
q_k^{(2)} \\
\end{bmatrix} = \begin{bmatrix}
I_k \\
r_k \\
\end{bmatrix}
\] \hspace{1cm} (2.97)

Then, the \( h \) can be expressed as: \( h(s_k, m_k, p_k) = D_x s_k + D_u m_k + D_w p_k \) where

\[
D_x = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix} \quad D_u = \begin{bmatrix}
0 \\
0 \\
1 \\
\end{bmatrix} \quad D_w = \begin{bmatrix}
0 \\
0 \\
0 \\
\end{bmatrix}
\] \hspace{1cm} (2.98)

Given this performance equation the relations \( q_{\text{min}} \leq q \leq q_{\text{max}} \) can be used to define the operating constraints, where \( q_{\text{min}} \) and \( q_{\text{max}} \) are defined as:

\[
q_{\text{min}} = \begin{bmatrix}
0 \\
0 \\
\end{bmatrix} \quad q_{\text{max}} = \begin{bmatrix}
I_{\text{max}} \\
r_{\text{max}} \\
\end{bmatrix}
\] \hspace{1cm} (2.99)

The notion of deviation variables extends quite naturally to the discrete-time framework. In comparison to the continuous-time procedure, the main difference is with regard to the determination of the steady-state operating point (SSOP). Rather than the relation of (2.28), the discrete-time definition of the SSOP is based on the following relation:

\[
s_{\text{SSOP}} = f_d(s_{\text{SSOP}}, m_{\text{SSOP}}, p_{\text{SSOP}})
\] \hspace{1cm} (2.100)

Then, using definitions similar to those of Section 2.2 the following deviation variable model can be constructed:

\[
x_{k+1} = f_d\left( x_k + s_{\text{SSOP}}, m_k + m_{\text{SSOP}}, n_k + p_{\text{SSOP}} \right) - s_{\text{SSOP}}
\] \hspace{1cm} (2.101)

\[
z_k = h\left( x_k + s_{\text{SSOP}}, m_k + m_{\text{SSOP}}, n_k + p_{\text{SSOP}} \right) - q_{\text{SSOP}}
\] \hspace{1cm} (2.102)

\[
y_k = l\left( x_k + s_{\text{SSOP}}, n_k + p_{\text{SSOP}} \right) - q_{\text{SSOP}}
\] \hspace{1cm} (2.103)
Example 2.10. Reconsider the inventory system of Example 2.9 and again assume \( \theta = 3 \). Assume \( p^{SSOP} \) is equal to the average product demand (denoted as \( d_{av} \)). Then the relations obtained by applying the steady-state equation (2.100) to (2.94) indicate that the average reorder value must also be equal to \( d_{av} \), but no restriction is put on the average inventory. As such, the SSOP can be defined as:

\[
\begin{bmatrix}
  s^{(1)SSOP} \\
  s^{(2)SSOP} \\
  s^{(3)SSOP} \\
  s^{(4)SSOP}
\end{bmatrix}
= \begin{bmatrix}
  I_{max}/2 \\
  d_{av} \\
  d_{av} \\
  d_{av}
\end{bmatrix}
\]

\[
q^{SSOP} = \begin{bmatrix}
  q^{(1)SSOP} \\
  q^{(2)SSOP}
\end{bmatrix}
= \begin{bmatrix}
  I_{max}/2 \\
  d_{av}
\end{bmatrix}
\]

\[
m^{SSOP} = [d_{av}]
\]

\[
p^{SSOP} = [d_{av}]
\]

where \( s^{(1)SSOP} \) is arbitrarily selected to be half of the inventory maximum. Application of this SSOP to equation (2.101) yields:

\[
x_{k+1} = f\left(x_k + s^{SSOP}, u_k + m^{SSOP}, w_k + p^{SSOP}\right) - s^{SSOP}
\]

\[
= A_d x_k + B_d u_k + G_d w_k + (A_d s^{SSOP} + B_d m^{SSOP} + G_d p^{SSOP} - s^{SSOP})
\]

where the last equality is due to equation (2.100). Similarly, application of (2.102) yields:

\[
z_k = h\left(x_k + s^{SSOP}, u_k + m^{SSOP}, w_k + p^{SSOP}\right) - q^{SSOP}
\]

\[
= D_x x_k + D_u u_k + D_w w_k + (D_x s^{SSOP} + D_u m^{SSOP} + D_w p^{SSOP} - q^{SSOP})
\]

Finally, it is found that

\[
z_{min} = q_{min} - q^{SSOP} = -q^{SSOP}
\]

\[
z_{max} = q_{max} - q^{SSOP} = \left[ \frac{I_{max}}{2} \right]
\]

If the functions \( f, h \) or \( l \) are nonlinear then a linearization procedure similar to that of Section 2.3 can be applied to (2.91)-(2.93). The details of this linearization procedure are left to the reader.

2.5.1 The Explicit Euler Method

While some discrete-time models exist purely within a discrete framework, such systems are the exception. The more common scenario is to start with a continuous-time (or differential equation) model and convert to a discrete-time model - the process of discretization.

Among the simplest of the conversion methods is the explicit Euler method. The method begins with the continuous-time nonlinear system of Equation (2.1): \( \dot{s} = f(s, m, p) \). Next define a time sequence \( s_k \) to approximate the function \( s(t) \). Specifically, \( s_k = \)
\[ s(k + 1) - s(k) \approx \Delta t \hat{s} = f(s(k), m_k, p_k) \] (2.110)

Simple rearrange of this expression finally yields:

\[ s(k + 1) = s(k) + \Delta t f(s(k), m_k, p_k) \] (2.111)

Given this recursion, the output equations are identical to the originals, but with the continuous variables replaced by the new discrete variables: \( q_k = h(s_k, m_k, p_k) \) and \( \theta_k = b(s_k, n_k) \).

If the continuous-time model is linear and in deviation variable form, (2.68)-(2.70), then application of the explicit Euler method results in the following discrete-time model:

\[ x(k + 1) = A_d x_k + B_d u_k + G_d w_k \] (2.112)
\[ z_k = D x_k + D u_k + D w_k \] (2.113)
\[ y_k = C x_k + v_k \] (2.114)

where

\[ A_d = (I + \Delta t A) \quad B_d = \Delta t B \quad G_d = \Delta t G \] (2.115)

and \( D_x, D_u, D_w, \) and \( C \) are unaltered from (2.68)-(2.70).

**Example 2.11.** Reconsider to the surge tank example of Chapter 1. Recall that the original continuous-time process model for the surge tank is (\( V \) is volume and \( v \) is volumetric flow):

\[ \dot{V} = v_{in} - v \] (2.116)

Using the explicit Euler method, one can approximate the original process as:

\[ \frac{V(k + 1) - V_k}{\Delta t} \approx \dot{V} = v_{in}^{(k)} - v_k \] (2.117)

which yields:

\[ V(k + 1) = V_k + \Delta t v_{in}^{(k)} - \Delta t v_k \] (2.118)

Since (2.116) is a linear system, the formulas of (2.115) can be applied to arrive at:

\[ V(k + 1) = a_d V_k + b_d v_k + g_d v_{in}^{(k)} \] (2.119)

where \( a_d = 1, b_d = -\Delta t \) and \( g_d = \Delta t \).

To illustrate how (2.119) can be used to simulate the surge tank process consider the following scenario: Select \( \Delta t = 0.1 \) and assume the tank volume is known to be 1 at time sample \( k = 2 \) (\( t = 0.2 \)), which defines \( V_2 = 1 \). Additionally, assume the input sequences
are expected to be as follows: \( v_2 = 6 \) and \( v_k = 0 \) for \( k > 2 \), \( v_2^{(in)} = 0 \), \( v_3^{(in)} = 10 \) and \( v_k^{(in)} = 0 \) for \( k > 3 \). Then, application of (2.114) yields:

\[
\begin{align*}
V_2 &= 1.0 \\
V_3 &= 1.0 - 0.6 + 0.0 = 0.4 \\
V_4 &= 0.4 - 0.0 + 1.0 = 1.4 \\
V_5 &= 1.4 - 0.0 + 0.0 = 1.4
\end{align*}
\]

and so on.

2.5.2 The Sample and Hold Method

The downside of the explicit Euler method is that the state of the system is implicitly assumed to be constant over the sample interval. While this may be a reasonable assumption for the input signals, especially the manipulated variable if under the influence of a discrete-time controller, it is likely a poor assumption for the state variable. If the continuous-time process of interest is linear, then one may employ the sample-and-hold method. Since the sample-and-hold approach is based on the analytic solution of a linear continuous-time process, derived in Section 2.4, it provides an exact representation of the continuous-time solution under the assumption of the input signals being constant between the sample intervals. Although the method can be applied any linear system, the following derivation is with respect to the deviation variable form of a linear continuous-time model, i.e., Equation (2.68). Similar to the Euler method, we begin with the define discrete-time variables: \( x_k = x(\Delta t \cdot k) \), \( u_k = u(\Delta t \cdot k) \) and \( w_k = w(\Delta t \cdot k) \), where the input signals are again assumed to be held at constant during each sample interval. Now assume we are given \( x_k \), \( u_k \) and \( w_k \) and would like to calculate \( x_{k+1} = x(\Delta t \cdot (k+1)) \). Application of Equation (2.88) with \( t = \Delta t \cdot (k+1) \) and \( t_o = \Delta t \cdot k \) yields:

\[
x(\Delta t \cdot (k+1)) = e^{A(\Delta t \cdot (k+1) - \Delta t \cdot k)}(\Delta t \cdot k) + \int_{\Delta t \cdot k}^{\Delta t \cdot (k+1)} e^{A(\Delta t \cdot (k+1) - \tau)}B u(\tau) d\tau + \int_{\Delta t \cdot k}^{\Delta t \cdot (k+1)} e^{A(\Delta t \cdot (k+1) - \tau)}G w(\tau) d\tau \quad (2.120)
\]

Recalling that the inputs are held constant over the interval of interest, \( u(\tau) = u_k \) and \( w(\tau) = w_k \) for \( \tau = \Delta t \cdot k \) to \( \tau = \Delta t \cdot (k+1) \), indicates that these terms can be moved outside the integration operators. Thus, one finds:

\[
x_{k+1} = (e^{A\Delta t}) x_k + \left( \int_{\Delta t \cdot k}^{\Delta t \cdot (k+1)} e^{A(\Delta t \cdot (k+1) - \tau)} d\tau \right) B u_k + \left( \int_{\Delta t \cdot k}^{\Delta t \cdot (k+1)} e^{A(\Delta t \cdot (k+1) - \tau)} d\tau \right) G w_k \quad (2.121)
\]

Next define \( \tau' = \Delta t \cdot (k+1) - \tau \) and perform a change of variables in the integration terms:

\[
x_{k+1} = (e^{A\Delta t}) x_k + \left( \int_0^{\Delta t} e^{A\tau'} d\tau' \right) B u_k + \left( \int_0^{\Delta t} e^{A\tau'} d\tau' \right) G w_k \quad (2.122)
\]
Thus, the final form of the sample and hold method is

\[ x_{k+1} = A_d x_k + B_d u_k + G_d w_k \]  

(2.123)

where

\[ A_d = (e^{A \Delta t}) \quad B_d = \left( \int_0^{\Delta t} e^{A \tau} d\tau \right) B \quad G_d = \left( \int_0^{\Delta t} e^{A \tau} d\tau \right) G \]  

(2.124)

The important point to note is that for a given sample time \( \Delta t \), the matrices \( A_d, B_d, \) and \( G_d \) will be constant and need only be calculated once prior to the discrete-time simulation. If the matrix \( A \) has an inverse, then the integrals of (2.124) can be performed analytically:

\[ B_d = \left( \int_0^{\Delta t} e^{A \tau} d\tau \right) B = A^{-1}(e^{A \Delta t} - I)B \]  

(2.125)

\[ G_d = \left( \int_0^{\Delta t} e^{A \tau} d\tau \right) G = A^{-1}(e^{A \Delta t} - I)G \]  

(2.126)

An alternative approach that does not require \( A \) to be invertible is the following relation

\[ e^{M \Delta t} = \begin{bmatrix} A_d & B_d & G_d \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \quad \text{where} \quad M = \begin{bmatrix} A & B & G \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \]  

(2.127)

It is instructive to compare the sample-and-hold method with the explicit Euler method. Recall Definition 2.3 (the exponential of a matrix) and note that the Euler approach, \( A_d = (I + \Delta t A) \), is simply the first two terms of the sample-and-hold approach. Thus, if \( \Delta t \) is sufficiently small, then one can expect the Euler method to be a good approximation of sample-and-hold. Similarly, if \( \Delta t \) is reasonably small then the integration terms in (2.125) and (2.126) will be about equal to \( \Delta t I \), which is the approximation given by the Euler method.

**Example 2.12.** Reconsider the mass-spring damper system of Example 2.4. At the end of that example the continuous-time deviation variable model of the process was concluded to be of the following form:

\[ \dot{x} = Ax + Bu + Gw \]  

(2.128)

If the parameters \((M, b, k)\) are selected to be \((1 \text{ kg}, 2 \text{ Ns/m}, 3 \text{ N/m})\), then matrices \(A, B\) and \(G\) will take the following values:

\[ A = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]  

(2.129)

Application of the explicit Euler method (with \( \Delta t = 0.5 \text{ seconds} \)) gives

\[ A_d = \begin{bmatrix} 1 & 0.5 \\ -1.5 & 0 \end{bmatrix} \quad B_d = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \quad G_d = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \]  

(2.130)

Application of the sample-and-hold method (with \( \Delta t = 0.5 \text{ seconds} \)) gives

\[ A_d = \begin{bmatrix} 0.74 & 0.28 \\ -0.84 & 0.18 \end{bmatrix} \quad B_d = \begin{bmatrix} 0.087 \\ 0.28 \end{bmatrix} \quad G_d = \begin{bmatrix} 0.087 \\ 0.28 \end{bmatrix} \]  

(2.131)
Application of the explicit Euler method (with $\Delta t = 0.2$ seconds) gives

$$A_d = \begin{bmatrix} 1 & 0.2 \\ -0.6 & 0.6 \end{bmatrix} \quad B_d = \begin{bmatrix} 0 \\ 0.2 \end{bmatrix} \quad G_d = \begin{bmatrix} 0 \\ 0.2 \end{bmatrix}$$

(2.132)

Application of the sample-and-hold method (with $\Delta t = 0.2$ seconds) gives

$$A_d = \begin{bmatrix} 0.95 & 0.16 \\ -0.48 & 0.62 \end{bmatrix} \quad B_d = \begin{bmatrix} 0.017 \\ 0.16 \end{bmatrix} \quad G_d = \begin{bmatrix} 0.017 \\ 0.16 \end{bmatrix}$$

(2.133)

Clearly, (2.132) is a better approximation of (2.133) than (2.130) is of (2.131). Now consider a simulation of the mass-spring-damper process with the following inputs:

$$u(t) = \begin{cases} 0 & t < 5 \\ 0 & t \geq 5 \end{cases} \quad u(t) = \begin{cases} 0 & t < 10 \\ -2 & t \geq 10 \end{cases}$$

(2.134)

and an initial condition $x(0) = [1 \ 0]^T$. The plots of Figure 2.6 show the simulation for the Euler approximated case. While the sample rate of 0.2 seconds provides reasonable accuracy, unacceptable degradation occurs if the sample rate is 0.5 seconds. In the sample-and-hold case, Figure 2.7, both sample rates provide an exact sampling of the continuous-time process.

Figure 2.9. Discrete-time simulation of mass-spring damper process using the explicit Euler method (left plot - $\Delta t = 0.5$; right plot - $\Delta t = 0.2$)

Figure 2.10. Discrete-time simulation of mass-spring damper process using the sample-and-hold method (left plot - $\Delta t = 0.5$; right plot - $\Delta t = 0.2$)
Table 2.6: MATLAB code used in calculations for Example 2.12

clear

% Continuous-time Model of Mass-Spring-Damper
A=[0 1; -3 -2]; B=[0; 1]; C=[0; 1]; nx=2; nu=1; nw=1;
% Conversion to Discrete-time
dt=0.5; Euler=1;
if (Euler==1)
    Ad=eye(nx)+dt*A; Bd=dt*B; Gd=dt*G;
else
    M=[A B G; zeros(nu+nw,nx+nu+nw)];
    M=expm(M*dt);
    Ad=M(1:nx,1:nx);
    Bd=M(1:nx,nx+1:nx+nu);
    Gd=M(1:nx,nx+nu+1:nx+nu+nw);
end
% Simulate Forced Process
NNN=15/dt; ttt=zeros(1,NNN); xxx=zeros(2,NNN);
xx=(0; 0); xxx(:,1)=xx;
for kk=1:NNN-1
    ttt(kk+1)=dt*kk; uk=0; wk=0;
    if (ttt(kk) >= 5) uk=1; end
    if (ttt(kk) >= 10) wk=-2; end
    xxx(:,kk+1)=Ad*xxx(:,kk)+Bd*uk+Gd*wk;
end
plot(ttt,xxx(1,:),'-k*',ttt,xxx(2,:),'-ko')
legend('Mass Position','Mass Velocity')
xlabel('Time (seconds)')
ylabel('State Variables (m or m/s)')

Figure 2.11. Comparison of continuous- and discrete-time simulations of the linearized DC-motor system using the sample and hold method, from Example 2.13
Example 2.13. Reconsider the DC motor model of Example 2.8. If we discretize the linearized model using the sample-and-hold method and a sample time of 5 seconds, then a repeat of the scenario of case 2 of part (iii) of the Example 2.8 results in the plots of Figure ?? (made with the code of Table 2.7). Notice that the sample and hold method mimics the continuous-time solution of the linear system exactly, even though the sample time is quite large. Of course, this is due to the fact that we are comparing with the linearized simulation. If compared with the nonlinear simulation, the discrete-time result would appear to be significantly degraded. If accuracy with the respect to the nonlinear simulation is desired then the implicit Euler method applied to the original nonlinear system might be more appropriate, but has the downside of resulting in a nonlinear discrete-time model. See section 2.5.3 for additional discussion.

Table 2.7. Discrete-time simulation code for Example 2.13

```
clear all
s_ss=[1.44; 254; 178]; m_ss=[36; 360]; p_ss=-280; q_ss=[91200; 49700];

% Linear Continuous-time Model
AA=[-0.5 0 0; -80.8 -0.25 -0.655; 46.3 0.262 -0.06];
BB=[ 0.02 0; 0 0.5; 0 0]; GG=[ 0; 0; 0.2];
Dx=[36 360 0; 0 0 -280]; Du=[1.44 254; 0 0]; Dw=[177];

% Discretization
dt=5; AAd=expm(AA *dt);
sum=zeros(3); Ndt=2000; ddt=dt/Ndt;
for ii=1:Ndt; sum=sum+expm(AA *ii*ddt);
end
BBd=sum*BB*ddt; GGd=sum *GG*ddt;

% Discrete-time Simulation
NNN=round(100/dt); t_disc=zeros(1,NNN); x_disc=zeros(3,NNN); z_disc=zeros(2,NNN);
uu=[0; 0]; ww=0;
for ii=1:NNN-1
t_disc(ii+1)=ii*dt;
    Case=2; t=t_disc(ii);
    if (Case == 1) %Case 1
        if t >= 10; uu=[2; 0]; end
        if t >= 50; uu=[-2; 0]; end
    else %Case 2
        if t >= 10; uu=[20; 0]; end
        if t >= 50; uu=[-10; 0]; end
    end
    x_disc(:,ii+1)=AAd*x_disc(:,ii)+BBd*uu+GGd*ww;
    z_disc(:,ii)=Dx*x_disc(:,ii)+Du*uu+Dw*ww;
end
z_disc(:,NNN)=z_disc(:,NNN-1);

%Continuous-time Simulation
[t_lin,x_lin]=ode45('motor_mod_ode_lin',[0 100],s_ss-s_ss);
[NN,dumb]=size(t_lin); z_lin=zeros(NN,2); for ii=1:NN
    [dxdt,zout]=motor_mod_ode_lin(t_lin(ii),x_lin(ii,:))'; z_lin(ii,:)=zout';
end

%Plots
plot(t_disc,x_disc(1,:)+s_ss(1),'k*'); t_lin,x_lin(:,1)+s_ss(1),'k');
title('Case 2','FontSize',14,'FontName','Times New Roman');
ylabel('Field Current (A)','FontSize',14,'FontName','Times New Roman');
xlabel('Time (s)','FontSize',14,'FontName','Times New Roman');
legend('Discrete-time','Continuous-time'); pause
plot(t_disc,x_disc(2,:)+s_ss(2),'k*'); t_lin,x_lin(:,2)+s_ss(2),'k');
title('Case 2','FontSize',14,'FontName','Times New Roman');
ylabel('Armature Current (A)','FontSize',14,'FontName','Times New Roman');
xlabel('Time (s)','FontSize',14,'FontName','Times New Roman');
```
It is interesting to note both of the discussed discrete-time conversion methods preserve the set of steady-state operating conditions. That is, if a triple \((s^{SSOP}, m^{SSOP}, p^{SSOP})\) satisfies relation (2.28), then it will also satisfy relation (2.100). For the explicit Euler method it is clear that the relation \(s^{SSOP} = s^{SSOP} + \Delta t f(s^{SSOP}, m^{SSOP}, p^{SSOP})\) is equivalent to \(0 = f(s^{SSOP}, m^{SSOP}, p^{SSOP})\). In the sample and hold case, it is a bit more challenging to illustrate equivalence. To simplify the analysis assume the inverse of \(A\) exists, so that the expressions of (2.125) and (2.126) can be applied. Starting with the discrete-time steady-state relation:

\[
s^{SSOP} = A_d s^{SSOP} + B_d m^{SSOP} + G_d p^{SSOP}
\]

\[= e^{A \Delta t} s^{SSOP} + A^{-1}(e^{A \Delta t} - I)B_m s^{SSOP} + A^{-1}(e^{A \Delta t} - I)G_p s^{SSOP}
\]

(2.135)

Then, Definition 2.3, indicates that \(A e^{A \Delta t} = e^{A \Delta t} A\), which can be used to rearranged (2.135) into the following form:

\[
0 = A^{-1} (e^{A \Delta t} - I) (A_s s^{SSOP} + B_m s^{SSOP} + G_p s^{SSOP})
\]

(2.136)

Thus, any triple \((s^{SSOP}, m^{SSOP}, p^{SSOP})\) that satisfies \(0 = A_s s^{SSOP} + B_m s^{SSOP} + G_p s^{SSOP}\) will also satisfy (2.136).

This equivalence result will be quite useful in the later chapters of the book, when the steady-state relation is incorporated into the design of the feedback controller. In particular, since the continuous-time steady state relation is likely simpler, owing to greater sparsity, it will be beneficial to use it vouchers the discrete-time relation of (2.100).

2.6 - Case Study Processes

In this section, we introduce a number of process models that will used throughout the text as exercises. It is suggested that the reader select a few of these case study processes and work through the associated exercises given in each chapter. In addition to being a bit more involved than the regular exercise problems, use of the same case study system within multiple chapters will serve to connect the concepts of each chapter.

2.6.1 - Furnace Reactor Process

Consider the furnace reactor process of Figure 2.12. The process state, manipulation and disturbance vectors are: 

\[
s = \begin{bmatrix} T_F & T_R & C_{O_2} & C_{CO} \end{bmatrix}^T, \quad m = [F_{feed} \ F_{fuel} \ P_v]^T, \quad p = [T_0].
\]
Based on step tests (around the operating point $s^{\text{nom}} = [375\, K \ 500\, K \ 4\% \ 100 \, p\, p\, m]^T$, $n^{\text{nom}} = [10000 \, \text{bbl/day} \ 10 \, \text{bbl/day} \ 0.1\%]^T$, $p^{\text{nom}} = [300\, K]$) the following dynamic model has been determined:

$$
A = \begin{bmatrix}
-8000 & 0 & 0 & 0 \\
2000 & -1500 & 0 & 0 \\
0 & 0 & -5000 & 0 \\
0 & 0 & 0 & -5000 \\
\end{bmatrix}
$$

$$
B = \begin{bmatrix}
-75 & 75000 & 0 \\
-25 & 0 & 0 \\
0 & -8500 & 8.5 \times 10^5 \\
0 & 0 & -5 \times 10^7 \\
\end{bmatrix}
$$

$$
G = \begin{bmatrix}
10000 \\
0 \\
0 \\
0 \\
\end{bmatrix}
$$

Note that the unit of time for this model is days, but the sample time of the controller is expected to be on the order of seconds. Limitations on the state and manipulated variables are as follows:

- $350\, K \leq T_f \leq 400\, K$
- $495\, K \leq T_f \leq 505\, K$
- $4\% \leq C_{O_2} \leq 6\%$
- $40 \, p\, p\, m \leq C_{CO} \leq 130 \, p\, p\, m$
- $950200 \, \text{bbl/day} \leq F_{\text{Feed}} \leq 101000 \, \text{bbl/day}$
- $7 \, \text{bbl/day} \leq F_{\text{Fuel}} \leq 12 \, \text{bbl/day}$
- $0.09\% \leq P_v \leq 0.11\%$

The economic objective function (to be maximized) is $g = 10F_{\text{feed}} - 30F_{\text{fuel}} - 0.1C_{CO}$. 

Figure 2.12. Process schematic of furnace reactor process
2.6.2 System of Three Masses

Consider the three mass system of Figure 2.13, described by the following equations:

\[
\frac{d r_0}{d t} = v_0 \\
\frac{d r_1}{d t} = v_1 \\
\frac{d r_2}{d t} = v_2 \\
M_0 \frac{d v_0}{d t} = -b_0(v_0 - v_1) - k_0(r_0 - r_1 - \Delta \hat{r}_0) - M_0 g_0 \\
M_1 \frac{d v_1}{d t} = b_0(v_0 - v_1) - b_1(v_1 - v_2) + k_0(r_0 - r_1 - \Delta \hat{r}_0) - k_1(r_1 - r_2 - \Delta \hat{r}_1) \\
M_2 \frac{d v_2}{d t} = b_1(v_1 - v_2) + k_1(r_1 - r_2 - \Delta \hat{r}_1) - k_2(r_2 - \Delta \hat{r}_2) - M_2 g_0 - F_m + F_d
\]

The parameters of the systems are:

\[M_0 = 75 \text{ kg}, \quad b_0 = 100 \text{ kg/s}, \quad k_0 = 10000 \text{ kg/s}^2, \quad \Delta \hat{r}_0 = 0.2 \text{ m}, \quad g_0 = 9.8 \text{ m/s}^2\]
\[M_1 = 500 \text{ kg}, \quad b_1 = 10000 \text{ kg/s}, \quad k_1 = 50000 \text{ kg/s}^2, \quad \Delta \hat{r}_1 = 0.28 \text{ m}, \quad M_2 = 10 \text{ kg}\]
\[k_2 = 5 \times 10^3 \text{ kg/s}^2, \quad \Delta \hat{r}_2 = 0.03 \text{ m}\]

The manipulated variable is \(F_m\) and the sample time of the controller is expected to be 0.1 seconds. The disturbance, \(F_d\), has a nominal value of \((M_0 + M_1 + M_2)g_0\). The limitations of the process are as follows: \(q_{\text{min}} \leq q \leq q_{\text{max}}\) where

\[
q_{\text{min}} = \begin{bmatrix} -5 \text{ m/s} \\ 0.1 \text{ m} \\ -0.45 \text{ m/s} \\ -1 \text{ m/s}^2 \\ 0.075 \text{ m} \\ -4.5 \text{ kN} \end{bmatrix}
q = \begin{bmatrix} a_0 \\ r_1 \\ v_1 \\ a_1 \\ r_1 - r_2 \\ F_m \end{bmatrix}
q_{\text{max}} = \begin{bmatrix} 5 \text{ m/s} \\ 0.2 \text{ m} \\ 0.45 \text{ m/s} \\ 1 \text{ m/s}^2 \\ 0.3 \text{ m} \\ 0 \text{ kN} \end{bmatrix}
\]

The acceleration terms are defined as:

\[a_0 = -b_0(v_0 - v_1)/M_0 - k_0(r_0 - r_1 - \Delta \hat{r}_0)/M_0 - g_0\]
\[a_1 = b_0(v_0 - v_1)/M_1 - b_1(v_1 - v_2)/M_1\]
\[+ k_0(r_0 - r_1 - \Delta \hat{r}_0)/M_1 - k_1(r_1 - r_2 - \Delta \hat{r}_1)/M_1 - g_0 + F_m/M_1\]

The economic objective function (to be minimized) is \(g = r_1\).
2.6.3 • Endothermic Reactor

Consider the following model of an endothermic CSTR with preheating heat exchangers.

\[
\rho C_p V_1 \frac{dT_1}{dt} = \nu_0 \rho C_p (T_0 - T_1) + UA(T_4 - T_1)
\]

\[
V_3 \frac{C_A}{dt} = \nu_0 (C_{A0} - C_A) - k_0 e^{-E/RT_3} C_A V_3
\]

\[
V_3 \frac{C_B}{dt} = -\nu_0 C_B + k_0 e^{-E/RT_3} C_A V_3
\]

\[
\rho C_p V_3 \frac{dT_3}{dt} = \nu_0 \rho C_p (T_2 - T_3) + (-\Delta H) k_0 e^{-E/RT_3} C_A V_3
\]

\[
\rho C_p V_4 \frac{dT_4}{dt} = \nu_0 \rho C_p (T_3 - T_4) - UA(T_4 - T_1)
\]

\[
T_2 = T_1 + Q/\nu_0 \rho C_p
\]

The manipulated variable is the amount of heat added to stream 1, \( Q \), and the disturbance is the inlet concentration, \( C_{A0} \). The sample time of the controller is expected to be 0.1 hour. The process parameters are as following: \( V_1 = V_3 = V_4 = 4 \text{ m}^3 \), \( \nu_0 = 10 \text{ m}^3/\text{hr}, k_0 = 1.5 \times 10^3 /\text{hr}, E = 2 \times 10^5 \text{ kcal/mol} K, R = 1.987 \text{ kcal/mol} K, \rho = 1000 \text{ kg/m}^3, C_p = 1 \text{ kcal/kg} K, T_0 = 298 K, (-\Delta H) = -2 \times 10^5 \text{ kcal/mol}, U = 550 \text{ kcal/hr m}^2 K \text{ and } A = 50 \text{ m}^2 \). The steady-state operating point can be determined from \( Q_{SSOP}^{SSOP} = 2.845 \times 10^6 \text{ kcal/hr} \) and \( C_{A0}^{SSOP} = 1 \text{ mole/m}^3 \). Limitations on the state and manipulated variables are as follows: \( 350 K \leq T_3 \leq 450 K \) and \( 2 \times 10^6 \text{ kcal/hr} \leq Q \leq 2.9 \times 10^6 \text{ kcal/hr} \). The economic objective function (to be maximized) is \( g = 100 \nu_0 C_B - 5Q \).

2.6.4 • Manufacturing Process

Consider the multi-echelon manufacturing system of Figure 2.14. The process is inherently discrete-time with a sample period of 1 week. The disturbances to the process are the two consumer demand signals, \( d_1 \) and \( d_2 \), which must be met at all times, each with a mean of 200 items per week and 100 items per week, respectively. The manipulated

Figure 2.14. Multi-echelon item manufacturing process.
variables are the raw material orders \( m_0 \) with units of items per week), the production rates of products 1 and 2 \( (m_1 \text{ and } m_2 \text{ with units of items per week}), \) and purchases of products 1 and 2 from outside vendors \( (m_3 \text{ and } m_4 \text{ with units of items per week}). \) The processing time of product 2 is twice that of product 1, which is reflected in the production limitation constraints of the processing unit: \( m_1 + 2m_2 \leq 450 \text{ items per week.} \) Other constraints on the manipulated variables are: \( 0 \leq m_0 \leq 500, \text{ } 0 \leq m_1 \leq 450, \text{ } 0 \leq m_2 \leq 225, \text{ } 0 \leq m_3 \leq 200, \text{ } 0 \leq m_4 \leq 200. \) The three storage units are also constrained: \( 0 \leq S_0 \leq 150 \text{ items, } 0 \leq S_1 \leq 500 \text{ items, and } 0 \leq S_2 \leq 25 \text{ items.} \) If there is insufficient inventory of product 1 or 2 at the beginning of the week, then a purchase of product 1 or 2 from an outside vendor must occur. Clearly, one would like to avoid such scenarios, as the economic penalty of purchasing from a competitor is large.

If the state vector is defined as \( s_k = [S_{0,k} \ r_{3,k} \ r_{2,k} \ r_{1,k} \ S_{1,k} \ S_{2,k}]^T, \) the manipulated variable vector as \( m_k = [m_{0,k} \ m_{1,k} \ m_{2,k} \ m_{3,k} \ m_{4,k}]^T, \) and the disturbance as \( d_k = [d_{1,k} \ d_{2,k}]^T, \) then the following system matrices could be used to model the process.

\[
A_d = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \quad B_d = \begin{bmatrix}
0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0
\end{bmatrix}
\]

\[
G_d = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

The economic objective function (to be minimized) is \( g = 50m_3 + 100m_4 + 0.5S_0 + 1.0S_1 + 1.5S_2. \)

### 2.6.5 - Building HVAC System

![Figure 2.15. Schematic of a single room HVAC system.](image)

Consider a single volume of perfectly mixed indoor air, along with a volume solid material (see Figure 2.16). The disturbances acting on the room are assumed to be the temperature of the outside air, \( T_{\text{outside}} \) (mean of 29°C) as well as the rate of contaminant generation, \( S_c \) (mean of 0.25 p p m / s). The control variables are temperature and contaminant concentration, each should satisfy: \( 29^\circ \text{C} \leq T_{\text{room}} \leq 26^\circ \text{C} \) and \( C_{\text{room}} \leq 400 \text{ p p m.} \)
The sample time of the controller is expected to be 15 minutes. The manipulated variables are the volumetric flows through the air processing unit; $F_{rcy}$ being the flow of air sourced from the room (recycled air) and $F_{fresh}$ being air from the outside (fresh air). Each manipulated variable should satisfy for all time: $0 \leq F_{rcy} \leq 2 \text{ m}^3/\text{s}$ and $0 \leq F_{fresh} \leq 0.3 \text{ m}^3/\text{s}$. From an energy standpoint, the intake of fresh air increases energy use, due to the larger cooling duty required to arrive at $T_{cool}$ from $T_{outside}$. However, from an indoor air quality standpoint, fresh air is the only means of reducing contaminant levels. Thus, the basic challenge is to identify the minimum flow of fresh air required to keep the contaminant level below the requirement and then use the recycled air for temperature regulation. Material and energy balances around the room yield the following nonlinear dynamic model of the process:

$$
\dot{T}_{room} = \frac{(F_{rcy} + F_{fresh})}{V_{room}}(T_{cool} - T_{room}) \\
+ \frac{UA_{room}}{V_{room} \rho C_p}(T_{outside} - T_{room}) + \frac{UA_{solid}}{V_{room} \rho C_p}(T_{solid} - T_{room})
$$

$$
\dot{T}_{solid} = \frac{UA_{solid}}{V_{solid} (\rho C_p)_{solid}}(T_{room} - T_{solid})
$$

$$
\dot{C}_{room} = \frac{F_{fresh}}{V_{room}}(C_{fresh} - C_{room}) + S_c
$$

where $V_{room} = 60 \text{ m}^3$, $V_{solid} = 10 \text{ m}^3$, $A_{room} = 200 \text{ m}^2$, $A_{solid} = 5 \text{ m}^2$, $U = 5.69 \text{ W/m}^2\text{K}$, $\rho = 40 \text{ mole/m}^3$, $C_p = 29 \text{ J/moleK}$, $(\rho C_p)_{solid} = 2000 \text{ J/m}^3\text{K}$, $T_{cool} = 20^\circ\text{C}$, $C_{fresh} = 0 \text{ ppm}$. The cost (or negative profit) of running the HVAC system of the process is defined by the following expression: $g = \rho C_p [F_{rcy}(22 - T_{cool}) + F_{fresh}(29 - T_{cool})]$.

### 2.6.6 Vapor Product Reactor

![Figure 2.16. Schematic of a single room HVAC system.](image)
Consider a jacket cooled CSTR with vapour product.

\[ \dot{V} = v_0 - v \]
\[ \dot{C}_A = \frac{v_0}{V} C_{Ain} - \frac{v}{V} C_A - k_0 e^{-E/RT} C_A \]
\[ \dot{C}_B = -\frac{v}{V} C_B + k_0 e^{-E/RT} C_A \]
\[ \dot{T} = \frac{v_0}{V} T_{in} - \frac{v}{V} T + \frac{(-\Delta H)}{\rho C_p} k_0 e^{-E/RT} C_A - \frac{UA(T - T_j)}{V \rho C_p} \]
\[ \dot{T}_j = \frac{v_j}{V_j} (T_{jin} - T_j) + \frac{UA(T - T_j)}{V_j \rho_j C_{pj}} \]
\[ \dot{p} = \frac{R_k}{64 - V} (V k_0 e^{-E/RT} C_A - F_g) \]

The manipulated variables are the volumetric flow of liquid out of the reactor, \( v \), the volumetric flow through the cooling jacket, \( v_j \), and the molar flow of gas out of the reactor, \( F_g \). The nominal values of \( v \), \( v_j \) and \( F_g \) are 40 ft\(^3\)/hr, 56.6 ft\(^3\)/hr and 10.6 lb mole/hr, respectively. The sample time of the controller is expected to 2 minutes. Limitations on the state and manipulated variables are as follows:

\[
\begin{align*}
40 & \leq V \leq 55 \text{ ft}^3 \\
0.05 & \leq C_A \leq 0.35 \text{ lb mole/ft}^3 \\
560 & \leq T \leq 620 \text{ °R} \\
560 & \leq T_j \leq 650 \text{ °R} \\
445 & \leq T_j \leq 475 \text{ lb ft}^2/\text{ft}^3 \\
0 & \leq v \leq 50 \text{ ft}^3/\text{hr} \\
0 & \leq v_j \leq 100 \text{ ft}^3/\text{hr} \\
0 & \leq F_g \leq 16 \text{ lb mole/\text{hr}}
\end{align*}
\]

The process disturbances are volumetric flow into the reactor, \( v_0 \), and inlet concentration of species \( A \), \( C_{Ain} \). The \( v_0 \) disturbance has a mean value of 40 ft\(^3\)/hr and the \( C_{Ain} \) disturbances has a mean of 0.5 lb mole/ft\(^3\). The profit of the process is defined as:

\( g = 0.375 \rho C_B - 0.015 v_j - 0.00225 F_g \)  

The parameters of the systems are:

\[
\begin{align*}
V_j & = 3.85 \text{ ft}^3 \\
T_{in} & = 530 \text{ °R} \\
T_{jin} & = 530 \text{ °R} \\
E & = 2.99 \times 10^4 \text{ btu/lb mole} \\
R & = 1.99 \text{ btu/lb mole} \text{ °R} \\
R_j & = 10.73 \text{ ft}^3 \text{ psi/lb mole} \text{ °R} \\
k_0 & = 7.08 \times 10^{10} \text{ /hr} \\
-\Delta H & = -3 \times 10^4 \text{ btu/lb mole} \\
\rho C_p & = 37.5 \text{ btu/ft}^3 \text{ °R} \\
\rho_j C_{pj} & = 62.3 \text{ btu/ft}^3 \text{ °R} \\
UA & = 2.25 \times 10^4 \text{ btu/hr} \text{ °R}
\end{align*}
\]

2.6.7 Two CSTR Process

Consider two non-isothermal CSTRs with cooling jackets in series with the following reactions: \( A(l) \rightarrow B(l) \) and \( B(l) \rightarrow C(l) \), where \( B \) is the product and \( C \) is an undesired side product. Using material and energy balances around each of the CSTRs, along with an arithmetic mean cooling water temperature approximation, yields the following differen-
The nominal values and parameters of the systems are

\[ F_1 = 0.283 \text{ m}^3/\text{s} \quad F_m = 0.229 \text{ m}^3/\text{s} \quad F_{c,1} = 0.7 \text{ m}^3/\text{s} \quad F_{c,2} = 0.7 \text{ m}^3/\text{s} \quad E_1/R = 6000 \text{ K} \quad E_2/R = 4500 \text{ K} \quad V \text{(both reactors)} = 5 \text{ m}^3 \quad k_{o,1} = 2.7 \times 10^8 \text{ s}^{-1} \quad k_{o,2} = 160 \text{ s}^{-1} \]

\[ \Delta H_{R,1}/(C_p \rho) = -5 \text{ m}^3/\text{kmol} \quad \Delta H_{R,2}/(C_p \rho) = -5 \text{ m}^3/\text{kmol} \quad U_1 = 0.35 \text{ m}^3/\text{s} \]

\[ U_2 = 0.35 \text{ m}^3/\text{s} \quad T_{c,1,in} = 300 \text{ K} \quad T_{c,2,in} = 275 \text{ K} \]

Here the manipulated variables are the volumetric flow rate and the makeup volumetric flow rate \( F_1 \), \( F_m \) and the volumetric flow rate going to each cooling jacket \( F_{c,1} \), and \( F_{c,2} \).

Process limitations include:

\[ T_1 \leq 350 \text{ K} \]
\[ T_2 \leq 350 \text{ K} \]
\[ F_1 + F_m \leq 0.8 \text{ m}^3/\text{s} \]
\[ T_{c,1,in} \leq 330 \text{ K} \]
\[ T_{c,2,in} \leq 330 \text{ K} \]
\[ F_1 \geq 0.05 \text{ m}^3/\text{s} \]
\[ F_m \geq 0.05 \text{ m}^3/\text{s} \]
\[ C_{A,2} \leq 0.3 \text{ kmol}/\text{m}^3 \]

The disturbance variables are the temperatures being fed into the CSTRs \( T_1 \), \( T_m \) and the concentration of \( A \) being fed in \( C_{1,in} \), and \( C_{m,in} \). Assume the temperature disturbances are colored noise with a mean of 300 K, a standard deviation of 3 K and a correlation time of 10 sec. Assume the concentration disturbances are colored noise with a mean of 20 kmol/m³, a standard deviation of 1 kmol/m³ and a correlation time of 60 sec.
The profit of the process is defined by the following expression, given in $/hr: \Phi = 10F_2C_{B2} - 0.01q_{cool,1} - q_{cool,2} - 0.1F_1 - 0.1F_m \text{ where } q_{cool,1} = \frac{2U_{a1}F_1(T_1 - T_{c,1,\text{in}})}{2F_c + U_{a1}} \text{ and } q_{cool,2} = \frac{2U_{a2}F_2(T_2 - T_{c,2,\text{in}})}{2F_c + U_{a2}}.

### 2.7 - Chapter Summary

In the remainder of the text, the linear state-space model in deviation variable form will play a central role and will serve as the starting point of the controller design methods to be developed. The general procedure for developing such a model is to begin with a continuous-time nonlinear state-space model, developed from the physical characteristics of the process. Based on the steady-state version of the model, one must identify an appropriate SSOP - possibly based on the performance and/or economic objectives of the process. Then, the nonlinear model can be linearized around this SSOP. As we have seen the linearization procedure will also serve to re-cast the model into deviation variable form. These deviation variables will serve to quantify how much the process deviates from the SSOP and will be central to the construction of a feedback controller. Finally, one has the option of converting to the discrete-time framework. The advantage of the discrete-time model is that it more appropriately captures the response of a system under the influence of a time sampled controller. Clearly, each step of the above model development procedure will introduce a level of error. The point, however, is not that the resulting model is exceedingly accurate, but that the linear model will provide access to a variety of powerful design tools. Then, once a controller has been designed (using the linear model), the final step in the design procedure is to test that controller on the continuous-time nonlinear model.

It is highlighted that material similar to that given in this chapter can be found in numerous textbooks (see for example Kwakernaak & Sivan, [103]; Stengle, [104]; Burl, [105]), including those intended for an undergraduate course on process control (see for example Stephanopoulos, [106]; Oggunnaike & Ray, [107]; Bequette, [108]; Seborg et al., [1]); Romagnoli & Palazoglu, [109]).

### Exercises

2.1. Consider the following model of a surge tank for which we plan to implement a liquid level control system:

\[
\begin{align*}
A \frac{dh}{dt} &= v_{in} - v_{out} \\
v_{out} &= p_v \sqrt{\rho h}
\end{align*}
\]

where $A = 1 \text{ m}^2$ is the cross-sectional area of the tank, $h$ is the liquid level, $v_{out}$ is the volumetric flow rate out of the tank, $\rho = 1.2 \text{ g/cm}^3$ is the fluid density and $p_v$ is proportional to the position of the valve on the exit stream.

(i) Develop a 1-dimensional nonlinear state space model of the system.

(ii) If $v_{in}$ is considered the disturbance input and $p_v$ is the manipulated variable, calculate the linearized model of the process. Assume nominal values of $v_{in}^{\text{nom}} = 1 \text{ m}^3/\text{min}$ and $p_v^{\text{nom}} = 0.01 \text{ m}^3/(\text{min kg}^{1/2})$. 

2.2. Consider the following first order differential equation \( \dot{x} = ax + bu \) where \( a = -2 \), \( b = 10 \), \( u(t) = e^{-2t} \) and the initial condition is \( x(0) = 5 \).

(i) Determine the analytic solution of this differential equation by identifying the expression \( x(t) \).

(ii) Calculate \( x(t) \) at \( t = 0.5 \).

(iii) Using the explicit Euler method with a sample time of 0.25, determine the discrete-time model \( x_{k+1} = ax_k + bu_k \).

(iv) Calculate \( x_k \) for the time corresponding to \( t = 0.5 \).

(v) Comment on the accuracy (or lack thereof) of the discrete-time model and why this is the case.

(vi) Repeat parts (iii), (iv) and (v) with a sample time of 0.1

2.3. Repeat Exercise 2.2, but this time use the sample and hold method to arrive at a discrete-time model.

2.4. Consider the differential equation \( m\ddot{r} = F \), where \( r \) is the position of a particle with mass \( m \) and \( F \) is an external force applied to the particle.

(i) Convert the above differential equation model into a state space form: \( \dot{x} = Ax + Bu \) and \( z = Dx + Du \). Assume the input signal, \( u \), is the applied force and the output, \( z \), is the particle position. Specifically, you should indicate the matrix values (\( A, B, D_x \) and \( D_u \)) that should be used. (Hint: You may find it helpful to consider of the system \( m\ddot{r} + b\dot{r} + kr = F \) and set \( b \) and \( k \) equal to zero.)

(ii) Convert the continuous-time state-space model of part (i) into a discrete-time model, using the Euler method. Specifically, find \( A_d \) and \( B_d \) of the model \( x_{k+1} = A_dx_k + B_d u_k \). Assume the mass, \( m \), equals 1 and the sample time, \( \Delta t \), is 1. (Hint: don't forget about the output equation.)

(iii) Assume the particle is initially 25 units away from the origin and is at rest. Determine the initial condition, \( x_0 \), that should be used.

2.5. Consider the following \( n^{th} \) order differential equation (\( u \) the input and \( z \) the output):

\[
\frac{d^n z}{dt^n} + a_{n-1} \frac{d^{n-1} z}{dt^{n-1}} + \cdots + a_1 \frac{dz}{dt} + a_0 z = bu
\]

Verify that a state space representation of this process can be obtained as:

\[
\dot{x} = Ax + Bu \quad z = D_x x + D_u u
\]

where

\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 1 & 0 \\
a_0 & a_1 & a_2 & \cdots & a_{n-1}
\end{bmatrix} \quad B = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
b
\end{bmatrix}
\]

\[
D_x = [1 \ 0 \ 0 \ \cdots \ 0] \quad D_u = [0]
\]
2.6. Using Definition 2.3 verify the identities

(i) \[ \frac{d}{dt} e^{At} = A e^{At} = e^{At} A \]
(ii) \[ e^{A(t_1 + t_2)} = e^{At_1} e^{At_2} \]
(iii) \[ (e^{At})^{-1} = e^{-At} \]
(iv) \[ (e^{At})^T = e^{A^T t} \]

2.7. Consider the pair of surge tanks depicted in Figure 2.17. Assume the first objective is to deliver a constant exit flow, \( v_2 \), to the downstream unit, in the face of upstream variations at \( v_0 \). Similar to the tank example of Chapter 1, it is reasonable to assume that \( v_1 = v_1^{(sp)} \) and \( v_2 = v_2^{(sp)} \) for all time and that \( v_1, v_2 \) can be selected as manipulated variables (through their relation to \( v_1^{(sp)} \) and \( v_2^{(sp)} \)). For each tank the volume of liquid should not exceed the total tank volume, and neither tank should be allowed to run dry. A volume balance around each tank yields:

\[ \dot{V}_1 = v_0 - v_1 \]
\[ \dot{V}_2 = v_1 - v_2 \]

(i) If the nominal inlet flow, \( v_0^{SSOP} \), is 30 m³/min, determine appropriate values for the other variables associated with the SSOP (\( V_1^{SSOP}, V_2^{SSOP}, v_1^{SSOP}, v_2^{SSOP} \)). It is also noted that the maximum volume allowed in each tank is 20 m³.

(ii) Assuming the performance output is \( q = [V_1 \ V_2 \ v_1 \ v_2]^T \), define the vector signals and matrices associated with the linear model of Equations (2.34)-(2.35).

(iii) For the linear system of part (ii), show that the explicit Euler and sample-and-hold methods generate the same discrete-time model.

2.8. Consider the three mass system of Figure 2.18, described by the following set of differential equations.

\[ M_1 \frac{d^2 r_1}{dt^2} + b_1 \left( \frac{d r_1}{dt} - \frac{d r_2}{dt} \right) + k_1 (r_1 - r_2) = u_1 \]
\[ M_2 \frac{d^2 r_2}{dt^2} - b_1 \left( \frac{d r_1}{dt} - \frac{d r_2}{dt} \right) - k_1 (r_1 - r_2) + b_2 \left( \frac{d r_2}{dt} - \frac{d r_3}{dt} \right) + k_2 (r_2 - r_3) = u_2 - u_1 \]
\[ M_3 \frac{d^2 r_3}{dt^2} - b_2 \left( \frac{d r_2}{dt} - \frac{d r_3}{dt} \right) - k_2 (r_2 - r_3) = w - u_2 \]
with parameters $M_1 = 1$, $M_2 = 100$, $M_3 = 10$, $b_1 = 0.05$, $b_2 = 3$, $k_1 = 0.1$ and $k_2 = 20$.

(i) Verify that a state space representation of this process can be obtained as:

$$
\dot{x} = Ax + Bu + Gw \\
z = Dx + Du + Dw w
$$

where

$$
A = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
-0.1 & 0.1 & 0 & -0.05 & 0.05 & 0 \\
0.001 & -0.201 & 0.2 & 0.0005 & -0.0305 & 0.03 \\
0 & 2 & -2 & 0 & 0.3 & -0.3
\end{bmatrix}
$$

$$
B = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0 \\
-0.01 & 0.01 & 0 \\
0 & -0.1 & 0
\end{bmatrix} \quad G = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
D_x = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \quad D_u = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix} \quad D_w = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

(ii) Using the sample-and-hold method (with a sample period $\Delta t = 0.5$), determine the discrete-time model.

(iii) Simulate the system using both the continuous and the discrete-time models. Assume $u_1$, $u_2$, and $w$ equal zero for all time and the system initial condition is $x(0) = [1 \ 0.1 \ 0.5 \ 0 \ 0 \ 0]^T$.

(iv) Simulate the system using the discrete-time model. Assume $x(0) = 0$, $w(t) = \sin(10t)$ and $u_1$ and $u_2$ equal zero for all time.
2.9. Consider the following model of an exothermic CSTR with cooling jacket.

\[
\begin{align*}
&V_r \frac{dC_A}{dt} = v_0(C_{A0} - C_A) - k_0 e^{-E/RT} C_A V_r \\
&\rho C_p V_r \frac{dT}{dt} = \rho C_p v_0 (T_0 - T) + (-\Delta H) k_0 e^{-E/RT} C_A V_r - UA(T - T_j) \\
&\rho_j C_{pj} V_j \frac{dT}{dt} = \rho_j C_{pj} v_j (T_{j0} - T_j) + UA(T - T_j)
\end{align*}
\]

with the following parameters: \(V_r = 1 \ m^3\), \(V_j = 0.08 \ m^3\), \(v_0 = 3 \ m^3/h\), \(k_0 = 3 \times 10^6 \ /hr\), \(E = 2 \times 10^4 \ kcal/kmol\), \(R = 1.987 \ kcal/kmolK\), \(\rho = \rho_j = 1000 \ kg/m^3\), \(C_p = C_{pj} = 0.2 \ kcal/kgK\), \(T_{j0} = 300 \ K\), \(\Delta H = 5 \times 10^5 \ kcal/kmol\), \(U = 800 \ kcal/hr \ m^2 K\) and \(A = 6 \ m^2\). The manipulated variable will \(v_j\) and two disturbances are expected: \(C_{A0}\) and \(T_{j0}\). The nominal conditions of the inputs are as follows: \(v_j^{SSOP} = 120 \ m^3/h\), \(C_{A0}^{SSOP} = 4 \ kmol/m^3\) and \(T_{j0}^{SSOP} = 300K\).

(i) Determine the three steady state operating points of the reactor. (Hint: solve the first and third equations for \(C_A^{SSOP}\) and \(T_j^{SSOP}\) respectively. After substituting these into the second equation, find the three zeros of the nonlinear equation graphically.)

(ii) Using an ode solver in MATLAB, simulate the nonlinear system, using the initial condition:

\[
s(0) = \begin{bmatrix} C_A(0) \\ T(0) \\ T_j(0) \end{bmatrix} = \begin{bmatrix} 2.8 \ kmole/m^3 \\ 684 \ K \\ 364 \ K \end{bmatrix}
\]

Assume \(v_j = v_j^{SSOP}\), \(C_{A0} = C_{A0}^{SSOP}\) and \(T_{j0} = T_{j0}^{SSOP}\) for all time and simulate for a period of 0.5 hours. Repeat the simulation using the initial condition

\[
s(0) = \begin{bmatrix} C_A(0) \\ T(0) \\ T_j(0) \end{bmatrix} = \begin{bmatrix} 2.8 \ kmole/m^3 \\ 687 \ K \\ 364 \ K \end{bmatrix}
\]

Write a paragraph explaining the simulation results. Do they have any relation to the results of (i)? Can these be used to say something about the stability of the three steady-states?

(iii) Add a simple proportional controller to the process. Use reactor temperature as the control variable (CV) and jacket flow rate as the manipulated variable. That is, add to the simulation: \(v_j(t) = v_j^{SSOP} + K_c (T_j^{SSOP} - T(t))\) where \(T_j^{SSOP} = 685.5K\) and \(K_c\) is the controller gain (should be \(\sim -40 \ m^3/hrK\)). (Assume \(C_{A0} = C_{A0}^{SSOP}\) and \(T_{j0} = T_{j0}^{SSOP}\) for all time.) Simulate this closed-loop system using a variety of initial conditions, say for example \(s(0) = [2.0 \ 690 \ 364]^T\). Write a paragraph analyzing the results. (Hint: change the simulation time to about 0.05 hours.)
(iv) Define the performance output to be reactant conversion \( q = (1 - C_A/C_{A0}) \).

Linearize the system around the steady-state condition:

\[
\begin{align*}
s_{SSOP} &= \begin{bmatrix} 2.8 \text{ kmole/m}^3 \\ 685.5 \text{ K} \\ 364 \text{ K} \end{bmatrix} \\
m_{SSOP} &= [120 \text{ m}^3/\text{hr}] \\
p_{SSOP} &= \begin{bmatrix} 4 \text{ kmole/m}^3 \\ 300 \text{ K} \end{bmatrix} \\
q_{SSOP} &= [0.429]
\end{align*}
\]

Determine the stability characteristics of the linearized model. (Use the MATLAB function ‘ss2tf’ to determine the open-loop characteristic equation. Then, use ‘roots’ to determine poles of the open-loop system.)

(v) Discretize the system using the sample-and-hold approach, using a sample period, \( \Delta t \), of 1sec. Using this discrete-time model, simulate the open-loop process and verify the stability result of part (iv).

(vi) Add a simple proportional controller to the process and simulate conditions identical to those used in (iii). Compare the results with those of the nonlinear simulation. (Remember to change to deviation variables for the new simulations. Additionally, remember to convert back to the original variables when making the plots.)

(vii) Using the closed-loop system of part (vi), simulate the response to step changes at the disturbances. (Regarding the size of these step changes, you should select something reasonable).

2.10. Consider the Furnace Reactor process of Section 2.6.1. (i) Indicate the deviation variable form of the process model. (ii) Using a MATLAB ode solver, simulate the following scenario. The initial condition is \( s(0) = [350K \ 525K \ 6\% \ 130 \text{ ppm}] \), \( m(t) = [10000 \text{ bbl/day} \ 10 \text{ bbl/day} \ 0.1\%] \) for \( t < 5 \text{ minutes} \) and \( = [10100 \text{ bbl/day} \ 9 \text{ bbl/day} \ 0.11\%] \) for \( t > 5 \text{ minutes} \) and \( p(t) = 300K \) for \( t < 10 \text{ minutes} \) and \( = 325K \) for \( t > 10 \text{ minutes} \). While this simulation should be performed using the model in deviation variables, all plots should display the results in natural variables. (iii) Convert the continuous-time linear model in deviation variables to a discrete-time form using the sample and hold method and a sample time of 30 seconds \( (= 0.000347 \text{ days}) \). Report the values of \( A_d, B_d \) and \( G_d \). (iv) Repeat the simulation of part (ii) using the discrete-time model and compare with the continuous-time results.

2.11. Consider the Three Mass system of Section 2.6.2.

(i) If \( r_1 \) is desired to be 0.15 \( m \), determine value of \( F_m \) required. This will require you to solve the steady-state relations of the system using \( r_1 = 0.15 \) and \( F_m \) as a variable to be determined. Note that the nominal value of \( F_d \), given in Section 2.6.2 should be assumed.

(ii) Using the steady-state point of part (i), determine the deviation variable form of the process model. Report \( A, B, G, D_x, D_u \) and \( D_w \).

(iii) Using a MATLAB ode solver, simulate the following scenario. The initial condition is \( s(0) = s^{\text{nom}}, m(t) = m^{\text{nom}} \) for \( t < 1 \text{ second} \) and \( = m^{\text{nom}} + 1 \text{ kN} \) for \( t > 1 \text{ second} \) and \( p(t) = p^{\text{nom}} \) for \( t < 3 \text{ seconds} \) and \( = p^{\text{nom}} + 1.5 \text{ kN} \) for \( t > 3 \text{ seconds} \). While this simulation should be performed using the model in deviation variables, all plots should display the results in natural variables.

(iv) Convert the continuous-time linear model in deviation variables to a discrete-
time form using the sample and hold method and a sample time of 0.1 seconds. Report the values of \( A_d, B_d \) and \( G_d \).

(v) Repeat the simulation of part (iii) using the discrete-time model and compare with the continuous-time results.

2.12. Consider the Endothermic CSTR process of Section 2.6.3.

(i) Put system model into the form of Equation (2.1)-(2.2).

(ii) Verify that the SSOP is given by:

\[
\begin{bmatrix}
  s^{(1)SSOP} \\
  s^{(2)SSOP} \\
  s^{(3)SSOP} \\
  s^{(4)SSOP}
\end{bmatrix} =
\begin{bmatrix}
  352.04 \\
  0.0174 \\
  425.53 \\
  371.63
\end{bmatrix},
\]

where

\[
s =
\begin{bmatrix}
  s^{(1)} \\
  s^{(2)} \\
  s^{(3)} \\
  s^{(4)}
\end{bmatrix} =
\begin{bmatrix}
  T_1 \\
  C_A \\
  T_3 \\
  T_4
\end{bmatrix},
\]

(iii) Linearize the model around the SSOP. Report \( A, B \) and \( G \).

(iv) Using a MATLAB ode solver, compare the linearized model with the original nonlinear model. Assume the initial condition is the SSOP and implement the following scenario:
- \( t < 0 \): \( C_A = 1.0 \text{ kmole/m}^3 \) and \( Q = 2.7 \times 10^6 \text{ kcal/hr} \)
- \( t > 0 \) and \( t < 10 \): \( C_A = 1.2 \text{ kmole/m}^3 \) and \( Q = 2.7 \times 10^6 \text{ kcal/hr} \)
- \( t > 10 \) and \( t < 20 \): \( C_A = 1.2 \text{ kmole/m}^3 \) and \( Q = 3.1 \times 10^6 \text{ kcal/hr} \)

(v) Discretize the linear model of part (iii) using a sample time 0.1 hr. Report the values of \( A_d, B_d \) and \( G_d \).

(vi) Repeat the scenario of part (iv) and compare the discrete-time simulation with the linear continuous-time simulation.

2.13. Consider the Manufacturing process of Section 2.6.4.

(i) Using the nominal values given in Section 2.6.4, determine the nominal values of all other variables. Note that the nominal values of the storage inventory will be indeterminate, so you should select reasonable values.

(ii) Indicate the deviation variable form of the process model.

(iii) Simulate the discrete-time process using the following scenario. The initial condition is \( s_0 = s^{\text{nom}}, w_k = w^{\text{nom}} \) for all \( k \), \( m_k = m^{\text{nom}} - [50 25 25 0 0]^T \) for \( k < 10 \), \( m_k = m^{\text{nom}} + [100 25 25 0 0]^T \) for \( 10 \leq k < 20 \) and \( m_k = m^{\text{nom}} + [100 25 25 15 15]^T \) for \( 20 \leq k < 30 \). While this simulation should be performed using the model in deviation variables, all plots should display the results in natural variables.

2.14. Consider the HVAC system of Section 2.6.5.

(i) If \( T_{\text{room}} \) and \( C_{\text{room}} \) are desired to be 25.5°C and 400 ppm, determine required values of \( F_{\text{recy}} \) and \( F_{\text{fresh}} \). This will require you to solve the steady-state relations of the system using \( F_{\text{recy}} \) and \( F_{\text{fresh}} \) as variables to be determined. Note that the nominal and parameter values given in Section 2.6.5.
(ii) Using the steady-state point of part (i), determine the deviation variable form of the process model. Report $A$, $B$ and $G$.

(iii) Using a MATLAB ode solver, simulate the following scenario. The initial condition is $s(0) = s^{nom}$, $m(t) = m^{nom}$ for $t < 1$ hour and $m(t) = m^{nom} + [0.5 \ -0.1]^T$ for $t > 1$ hour and $p(t) = 29^\circ C$ for $t < 2$ hours and $= 32^\circ C$ for $t > 2$ hours. While this simulation should be performed using the model in deviation variables, all plots should display the results in natural variables.

(iv) Convert the continuous-time linear model in deviation variables to a discrete-time form using the sample and hold method and a sample time of 15 minutes. Report the values of $A_d$, $B_d$ and $G_d$.

(v) Repeat the simulation of part (iii) using the discrete-time model and compare with the continuous-time results.
Chapter 3

Review of Linear Algebra

In this chapter we will review a number of the fundamental concepts related to linear algebra. While many of these concepts are of independent interest, they will serve as a foundation for our subsequent analysis of linear dynamic systems. In particular, the fundamental theorem of linear algebra will be used in Chapter 4 to gain a deeper understanding of the concepts of controllability and observability. Similarly, eigenvector decomposition will be used expose the inner workings of stability analysis. In addition, the notion of similar matrices will lead to the canonical form of a dynamic system and ultimately to the concepts of stabilizability and detectability, which will be prerequisites for the design of a feedback controller. Finally, the Lyapunov stability theorem of Chapter 4 will serve as the basis for many of the subsequent control design methods (especially those of Part III). However, this theorem rests heavily upon the notion of a positive definite matrix, which will be introduced in the current chapter.

3.1 Sets, Subsets and Linear Vector Spaces

Let us begin with the notion of a set. In essence, a set is just a collection of elements. The collection may have a finite number of elements, say the set of all engineering graduate students, or may have an infinite number of elements, say all real numbers in the interval between 0 and 1. A particularly relevant example is the set of all vectors in the two dimensional plane. This set will be denoted as $\mathbb{R}^2$ and is defined as follows

$$\mathbb{R}^2 = \left\{ x = \begin{bmatrix} a \\ b \end{bmatrix} \ | \ a \text{ and } b \text{ are real numbers} \right\} \quad (3.1)$$

(For the definition of transpose, $^T$, see Definition 2.4.) Equation 3.1 is read as “$\mathbb{R}^2$ is the set of all $x$ with a form $\begin{bmatrix} a \\ b \end{bmatrix}^T$ such that $a$ and $b$ are real numbers.” Note the symbol “|$” is shorthand for “such that”. A subset is just a set where its elements are all members of the original set. For example, one could define the following set:

$$S = \left\{ x = \begin{bmatrix} a \\ b \end{bmatrix}^T \ | \ a \text{ and } b \text{ are real numbers}, 0 \leq a \leq 1 \text{ and } 0 \leq b \leq 1 \right\} \quad (3.2)$$

In this case $S$ would be a subset of $\mathbb{R}^2$, denoted as $S \subseteq \mathbb{R}^2$.

In this text, we will mostly be interested in sets that have been endowed with two elementary operations - addition and scalar multiplication. Clearly, this requirement will exclude the set of all engineering graduate students from our consideration, as it will be difficult to define such operations for this collection of elements. However, the set $\mathbb{R}^2$
is certainly amenable to the definition of such operators. For example if \( x \) is an element of \( \mathbb{R}^2 \) (denoted as \( x \in \mathbb{R}^2 \)) and \( \alpha \) is a real number then scalar multiplication can be defined as \( \alpha x = [\alpha a \ a b]^T \). Similarly, if given two elements \( x_1 \) and \( x_2 \), both in \( \mathbb{R}^2 \) (denoted \( x_1, x_2 \in \mathbb{R}^2 \)), then the addition of the two can be defined as \( x_1 + x_2 = [a_1 + a_2 \ b_1 + b_2]^T \).

It should be highlighted that the set of scalars used for scalar multiplication could be any subset of \( \mathbb{R} \) or \( \mathbb{C} \). Sets for which addition and scalar multiplication are defined are typically denoted as vector sets. An important class of vector sets is those in which the operations of scalar multiplication and/or addition using any elements of the set will result in an element that is guaranteed to be within the original set. This property of never leaving the original set (under the operations of scalar multiplication and/or addition) provides an important starting point in the sense that all the linear combinations of elements are guaranteed to be elements of the set. Think of it as defining the universe (or space of elements) for the subsequent analysis.

**Definition 3.1.** A linear vector space is a set \( X \) such that \( \alpha_1 x_1 + \alpha_2 x_2 \in X \) for all \( x_1, x_2 \in X \) and all \( \alpha_1, \alpha_2 \in \mathbb{R} \).

Returning to the previous example, consider the set \( \mathbb{R}^2 \). Since for all \( \alpha_1, \alpha_2 \in \mathbb{R} \) and all \( x_1, x_2 \in X \) it is found that \( \alpha_1 x_1 + \alpha_2 x_2 = [\alpha_1 a_1 + \alpha_2 a_2 \ \alpha_1 b_1 + \alpha_2 b_2]^T \in \mathbb{R}^2 \). Thus, \( \mathbb{R}^2 \) is shown to be a linear vector space. In fact, if we generalize \( \mathbb{R}^2 \) to the set of elements with \( n \)-tuple real values, \( \mathbb{R}^n \), then it is easily concluded that this is also a linear space.

However, the following counter example illustrates that the set \( S \) (of Equation 3.2) is not a linear vector space. Let \( x_1 = [0.5 \ 0.5]^T, x_2 = [0 \ 0]^T, \alpha_1 = 4 \) and \( \alpha_2 = 0 \) then \( \alpha_1 x_1 + \alpha_2 x_2 = [2 \ 2]^T \notin S \). The reader should note the difference in proving that a set is or is not a linear vector space. In the former case, we must show the property for all scalars and all elements of the set, while in the latter case one need only find a single example for which the property fails.

**Example 3.1.** Let \( X \) be the set of all second order polynomials. In general an element of \( X \) is a function \( x_1(t) = a_1 t^2 + b_1 t + c_1 \), where \( a_1, b_1, c_1 \in \mathbb{R} \). If \( x_2 \) is defined similarly then it is easily verified that \( \alpha_1 x_1 + \alpha_2 x_2 \in X \) for all \( \alpha_1, \alpha_2 \in \mathbb{R} \).

In many cases, Definition 3.1 is expanded such that \( \mathbb{R} \) is replaced by \( \mathbb{C} \). In these cases, the vector space \( X \) will need to consist of complex elements, e.g. \( X = \mathbb{C}^n \). The following two definitions and associate corollary are direct consequences of Definition 3.1.

**Definition 3.2.** A subset \( S \) of a linear vector space \( X \) (denoted as \( S \subseteq X \)) is a subspace, if \( S \) is also a linear vector space.

**Definition 3.3.** Let \( S \) be a subset of a linear vector space \( X \). The span of \( S \) (denoted \( s \text{pan} \{S\} \)) is the set of all elements resulting from linear combinations of elements in \( S \).

**Corollary 3.1.** If \( S \) is a subset of a linear space \( X \), then \( s \text{pan} \{S\} \) is a subspace.

**Example 3.2.** Let \( X = \mathbb{R} \) and consider the subsets of Figure 3.1. Subset \( S_1 \) is contained completely in the positive quadrant and thus is not a subspace. Subset \( S_2 \) contains all points along the identified line and is easily shown to be a subspace. Subset \( S_3 \) also contains all points along a line, but does not contain the origin and thus is not a subspace.
3.2 Inner Products and Orthogonality

Subset $S_3$ is frequently denoted as a linear variety. The span of subset $S_1$ is equal to subset $S_2$. The span of subset $S_2$ is equal to itself. The span of subset $S_3$ equals the entire space.

In the context of Chapter 2, the obvious application of the linear vector space notion is toward the concept of a state space model. Specifically, if given a model of the form of Equation 2.1, then the state vector, $s$, is an element of $\mathbb{R}^n$, where $n$ is the dimension of the state vector. More specifically, if the solution to Equation 2.1 is $s(t)$, then at any given time $t_1$, $s(t_1) \in \mathbb{R}^n$. Similar notions can be applied to the space of inputs, associated with $m$ and $p$, as well as the space of outputs, associated with $q$ and $\theta$.

### 3.2 Inner Products and Orthogonality

In the two-dimensional space, $\mathbb{R}^2$, there is an intuitive understanding of directionality, and specifically the notion of orthogonality, in that the intersection of two vectors forms a right angle, or the two are perpendicular. In higher dimensional spaces, the generalized notion of directionality will need to be formalized. We will find this formality within the notion of an inner product.

**Definition 3.4.** Let $X$ be a linear vector space and $\overline{\alpha}$ be the complex conjugate of $\alpha$. The **inner product**, denoted $\langle x, y \rangle$, is a scalar function that satisfies the following five properties for all $x, y, z \in X$:

1. $\langle y, x \rangle = \overline{\langle x, y \rangle}$.
2. If $\alpha \in \mathbb{C}$, then $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$ and $\langle x, \alpha y \rangle = \overline{\alpha} \langle x, y \rangle$.
3. $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$.
4. $\langle x, x \rangle$ is real and non-negative.
5. $\langle x, x \rangle = 0$ only if $x = 0$.

Let $X = \mathbb{C}^n$, then an element $x \in X$ is denoted as $x = [x^{(1)} \ x^{(2)} \ \cdots \ x^{(n)}]^T$. For this linear vector space, a commonly defined inner product is

$$\langle x, y \rangle = \sum_{i=1}^{n} x^{(i)} \overline{y^{(i)}} \tag{3.3}$$

This is easily shown to satisfy criteria (i) – (v) of Definition 3.4.
Example 3.3. Let $X = \mathbb{C}^2$ and denote an element $x \in X$ as $x = [x^{(1)} \ x^{(2)}]^T$. Then, define the function:

$$[x, y] = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} = y^{(1)}x^{(1)} - y^{(2)}x^{(2)} - y^{(1)}x^{(2)} + 4y^{(2)}x^{(2)}$$

To see if this a valid inner product, we will need to check if this function satisfies the five criteria of Definition 3.4.

(i) $[x, y] = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} = [y, x]$.

(ii) $[x, \alpha y] = \begin{bmatrix} \alpha y^{(1)} \\ \alpha y^{(2)} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} = \alpha [x, y]$.

(iii) $[x + y, z] = \begin{bmatrix} z^{(1)} \\ z^{(2)} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix} \begin{bmatrix} x^{(1)} + y^{(1)} \\ x^{(2)} + y^{(2)} \end{bmatrix} = [x, z] + [y, z]$.

(iv) First, recall that $\alpha \overline{\alpha} = |\alpha|^2 \geq 0$. Then,

$$[x, x] = \overline{\alpha}^{(1)}x^{(1)} - \overline{\alpha}^{(1)}x^{(2)} - \overline{\alpha}^{(2)}x^{(1)} + 4\overline{\alpha}^{(2)}x^{(2)}$$

$$= (\overline{\alpha}^{(1)} - \overline{\alpha}^{(2)})(x^{(1)} - x^{(2)}) + 3\overline{\alpha}^{(2)}x^{(2)} = |(\overline{\alpha}^{(1)} - \overline{\alpha}^{(2)})|^2 + 3|x^{(2)}|^2 \geq 0$$

(v) The only way $[x, x] = |(x^{(1)} - x^{(2)})|^2 + 3|x^{(2)}|^2 = 0$ is if $x^{(1)} = 0$ and $x^{(2)} = 0$.

Let us now return to the intuitive nature of $\mathbb{R}^2$. If the inner product is defined as in Equation (3.3), over $\mathbb{R}^2$, then one finds that the notion of two vectors being perpendicular is equivalent to the inner product being equal to zero (see Example 3.4 below). Thus, the generalization to any linear vector space is the following - two vectors are perpendicular (or orthogonal) if the inner product of the two vectors is zero. One may also gain intuition from the three-dimensional space $\mathbb{R}^3$. Specifically, a vector is said to be perpendicular to a plane if it is at a right angle to all vectors in that plane. One may additionally ask: find the plane that perpendicular to a given vector or set of vectors. The following definitions formalize all of the above notions.

**Definition 3.5.** Vectors $x, y \in X$ are **orthogonal**, denoted $x \perp y$, if $[x, y] = 0$.

**Definition 3.6.** A vector $x \in X$ is **orthogonal to a set** $S$, denoted $x \perp S$, if $x \perp s$ for all $s \in S$.

**Definition 3.7.** Let $S$ be a subset of a linear vector space $X$, then the **orthogonal complement** of $S$ is defined as: $S^\perp = \{ x \in X \mid x \perp S \}$. $S^\perp$ is read as “$S$ perpendicular” or just “$S$ perp”. Also, $(S^\perp)^\perp$ is read as “$S$ double perp”.

**Corollary 3.2.** If $S$ is a subset of a linear vector space $X$, then $S^\perp$ is a subspace and $(S^\perp)^\perp = \text{span}\{S\}$. 

Definition 3.3. Let \( S_1 \) and \( S_2 \) be subspaces of a linear vector space \( X \). The direct sum of \( S_1 \) and \( S_2 \) is defined as: \( S_1 \oplus S_2 = \{ x \in X \mid x = s_1 + s_2 \text{ where } s_1 \in S_1 \text{ and } s_2 \in S_2 \} \).

Corollary 3.3. If \( S \) is a subset of a linear vector space \( X \), then \( S \perp \perp S \perp \perp = X \).

Example 3.4. Let \( X = \mathbb{R}^2 \) and employ the inner product of Equation (3.3).

(a) If \( x = [1 \ 2]^T \), \( y_1 = [-2 \ 1]^T \), \( y_2 = [2 \ -1]^T \) and \( y_3 = [-20 \ 10]^T \), then \( x \perp y \) for \( i = 1, 2, 3 \).

(b) Consider the set \( S = \{ y \in \mathbb{R}^2 \mid y = \alpha [-2 \ 1]^T, \alpha \in \mathbb{R} \} \). If \( x = [1 \ 2]^T \), then \( x \perp S \).

(c) Consider the singleton set \( S = \{ [-2 \ 1]^T \} \), then \( S \perp = \{ x \in \mathbb{R}^2 \mid x = \alpha [1 \ 2]^T, \alpha \in \mathbb{R} \} \) and \( S \perp \perp = \{ x \in \mathbb{R}^2 \mid x = \alpha [-2 \ 1]^T, \alpha \in \mathbb{R} \} \). Clearly, \( S \perp \oplus S \perp \perp = \mathbb{R}^2 \).

Example 3.5. Let \( X = \mathbb{R}^3 \) and employ the inner product of Equation (3.3). If the set is just \( S = \{ [1 \ 0 \ 0]^T \} \), then \( S \perp = \{ x \in \mathbb{R}^3 \mid x = \alpha_1 [0 \ 1 \ 0]^T + \alpha_2 [0 \ 0 \ 1]^T, \alpha_1, \alpha_2 \in \mathbb{R} \} \) and \( S \perp \perp = \{ x \in \mathbb{R}^3 \mid x = \alpha [1 \ 0 \ 0]^T, \alpha \in \mathbb{R} \} \). Clearly, \( S \perp \oplus S \perp \perp = \mathbb{R}^3 \).

3.3 Linear Transformations

Recall the model of a discrete-time process, given in Equation (2.91): \( s_{k+1} = f_d (s_k) \). In this model, a vector at time zero, \( s_0 \), is transformed by the mapping \( f_d \) to generate a new vector \( s_k \). This iteration continues until all of the \( s_k \)'s are calculated. As one might expect, being able to characterize such mappings will be an important tool in the analysis of a dynamic system. However, rather than consider the general class of mappings, we will focus on the special class known as linear transformations. Our first effort to characterize a linear transformation will be the definition of a pair of intuitively simple subspaces - the null space and range space. Using these subspaces, the Fundamental Theorem Linear Algebra will give conditions for the existence and uniqueness of a solution to a linear equation based on a linear transformation. To help utilize the Fundamental Theorem, a transformation companion to the original, known as the adjoint, will be introduced and shown to lead to the four fundamental subspaces of a linear transformation.

Definition 3.9. Let \( X \) and \( Y \) be linear vector spaces, then a transformation \( L: X \rightarrow Y \) (commonly denoted as \( y = Lx \), where \( y \in Y \) and \( x \in X \)) is a linear transformation if for all \( x_1, x_2 \in X \) and \( \alpha_1, \alpha_2 \in \mathbb{C} \) the following holds:

\[
L(\alpha_1 x_1 + \alpha_2 x_2) = \alpha_1 Lx_1 + \alpha_2 Lx_2
\]

Definition 3.10. Let \( X \) and \( Y \) be linear vector spaces, then the null space of linear transformation \( L: X \rightarrow Y \) is defined as:

\[
\mathbb{N}(L) = \{ x \in X \mid Lx = 0 \}
\]
Definition 3.11. Let $X$ and $Y$ be linear vector spaces, then the range space of linear transformation $L : X \to Y$ is defined as:

$$\mathcal{R}(L) = \{ y \in Y \mid y = Lx \text{ where } x \in X \}$$

Corollary 3.4. If $L : X \to Y$, where $X$ and $Y$ are linear vector spaces, then $\mathcal{N}(L)$ is a subspace of $X$ and $\mathcal{R}(L)$ is a subspace of $Y$.

Example 3.6. Let $X$ and $Y$ both be the set of second order polynomials. If the mapping $L : X \to Y$ is the first derivative operator $(d/dt)$, then $L$ is easily shown to be a linear transformation. The null space of $L$ is the subspace of zero order polynomial (constants). Also, the range space of $L$ is the subspace of first order polynomials. □

Example 3.7. Let $X = \mathbb{R}^3$, $Y = \mathbb{R}^2$, and define $L : X \to Y$ by the matrix $\begin{bmatrix} 5 & 1 & 0 \\ 10 & 2 & 0 \end{bmatrix}$.

(i) Show that $L$ is a linear transformation

(ii) Find the null space of $L$

(iii) Find the range space of $L$

Solution: (i) If an element $x \in X$ is $x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{bmatrix}$, then using the usual convention of matrix multiplication, $y = Lx$ is found to be $\begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} = \begin{bmatrix} 5x^{(1)} + 1x^{(2)} + 0x^{(3)} \\ 10x^{(1)} + 2x^{(2)} + 0x^{(3)} \end{bmatrix}$.

(ii) The problem is to find $x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{bmatrix}^T$ such that $Lx = 0$, or equivalently $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$ such that $\begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} = \begin{bmatrix} 5x^{(1)} + 1x^{(2)} + 0x^{(3)} \\ 10x^{(1)} + 2x^{(2)} + 0x^{(3)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. By inspection one may easily conclude that $[0 \ 0 \ 1]^T \in \mathcal{N}(L)$. In fact, for all $\alpha_1 \in \mathbb{R}$, $[0 \ 0 \ \alpha_1]^T \in \mathcal{N}(L)$. Similarly, for all $\alpha_2 \in \mathbb{R}$, $[\alpha_2 \ -5\alpha_2 \ 0]^T \in \mathcal{N}(L)$. In addition, any vector that is a linear combination of these two subspaces will also be in the null space of $L$. Specifically,

$$\alpha_1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ -5 \\ 0 \end{bmatrix} \in \mathcal{N}(L) \text{ for all } \alpha_1, \alpha_2 \in \mathbb{R}$$

This is equivalently stated as: $\mathcal{N}(L) = span \{ [0 \ 0 \ 1]^T, [1 \ -5 \ 0]^T \}$.

(iii) If an element $x \in X$ is denoted by $x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{bmatrix}^T$, then

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} = \begin{bmatrix} 5x^{(1)} + 1x^{(2)} + 0x^{(3)} \\ 10x^{(1)} + 2x^{(2)} + 0x^{(3)} \end{bmatrix} = 2 \begin{bmatrix} 5x^{(1)} + 1x^{(2)} \\ 5x^{(1)} + 1x^{(2)} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$
Since all vectors $y$ generated by $Lx$ will be of the form $\alpha [1 \ 2]^T$, $\mathcal{N}(L) = \text{span}\{[1 \ 2]^T\}$. Another way to see the result is to express the transformation as a linear combination of the matrix columns:

$$
\begin{bmatrix}
  y^{(1)} \\
  y^{(2)}
\end{bmatrix} = \begin{bmatrix}
  5 \\
  10
\end{bmatrix} x^{(1)} + \begin{bmatrix}
  5 \\
  10
\end{bmatrix} x^{(2)} + \begin{bmatrix}
  0 \\
  0
\end{bmatrix} x^{(1)}
$$

Then one would conclude that $\mathcal{N}(L) = \text{span}\{[5 \ 10]^T, [1 \ 2]^T, [0 \ 0]^T\}$, which is clearly equal to $\text{span}\{[1 \ 2]^T\}$. \qed

**Corollary 3.5.** If a linear transformation $L : \mathbb{C}^m \rightarrow \mathbb{C}^n$ is defined by a matrix $M$ with columns $c_1, c_2, \ldots, c_m$, then $\mathcal{N}(L) = \text{span}\{c_1, c_2, \ldots, c_m\}$.

The concepts of range space and null space are particularly useful for determining if a solution to a linear equation exists and if that solution is unique. The following theorem, termed the **Fundamental Theorem of Linear Algebra**, summarizes the result.

**Theorem 3.1.** If $L : X \rightarrow Y$, where $X$ and $Y$ are linear vector spaces, then a solution $x$ will exist to the equation $Lx = b$ if and only if $b \in \mathcal{N}(L)$. Furthermore, the solution will be unique if and only if $\mathcal{N}(L)$ is empty (i.e., the only $x$ in $\mathcal{N}(L)$ is $x = 0$).

**Proof.** The existence part stems directly from the definition of range space. That is, for every element $b \in \mathcal{N}(L)$, there must exist an element $x \in X$ such that $b = Lx$ (i.e., $b \in \mathcal{N}(L)$ implies existence). In addition, if $x$ is such that $Lx = b$, then $b$ must be in the range space of $L$ (i.e., existence implies $b \in \mathcal{N}(L)$). To prove the uniqueness part, assume $\mathcal{N}(L)$ is not empty, which implies there exists $x_N \in X$ such that $Lx_N = 0$. Then, if $x_N \in X$ is such that $Lx_N = b$, one can also conclude that $x_N + x_N \in X$ is also a solution since $L(x_N + x_N) = Lx_N = b$. Thus, $\mathcal{N}(L)$ not being empty, implies the solution is not unique, or equivalently that uniqueness implies $\mathcal{N}(L)$ is empty. To prove the other direction, assume we have two points $x_1, x_2 \in X$ are such that $Lx_1 = b$ and $Lx_2 = b$, then subtracting these two equations gives $L(x_1 - x_2) = b - b = 0$ or $x_1 - x_2 \in \mathcal{N}(L)$. Thus, non-uniqueness implies $\mathcal{N}(L)$ not empty, or equivalently empty implies uniqueness. \qed

While Theorem 3.1 provides significant motivation to calculate the range space and null space, the “by inspection” methods of Example 3.7 will quickly fail us, especially if the vector spaces $X$ and $Y$ are of large dimension and the linear transformation is not quite so simple. If the linear transformation is just a matrix, then the end of Example 3.7 indicates that the simplest method of calculating the range space is to recognize that it is just the span of the columns of the matrix, as indicated by Corollary 3.5. However, Example 3.7 also indicates that many of the columns of the matrix $M$ may be redundant, and the issue is to remove these redundancies. This will be of particular interest if one is trying to use Theorem 3.1 to show that a solution to the equation $Lx = b$ exists for all $b \in Y$. To do so, one will need to show that the range space is the entire space (i.e., $\mathcal{N}(L) = Y$). In this case, finding redundant columns will be of critical importance. In Section 3.4, a systematic method of finding and removing these redundancies will be given.

In the remainder of the current section an alternative approach will be investigated. Specifically we will consider the orthogonal complement of the range space: $\mathcal{N}(L)^\perp$. If one can show that $\mathcal{N}(L)^\perp$ is empty, then this will be equivalent to showing that $\mathcal{N}(L) = Y$.
and ultimately a solution to \( Lx = b \) exists regardless of the value used for \( b \). This characterization of \( \Re(L)^\perp \) will require the definition of a companion linear operator known as the adjoint (which should not be confused with the “classical adjoint” associated with matrix inversion and detailed in Definition 3.22). Using the notion of an adjoint, a rather elegant decomposition of the vector spaces \( X \) and \( Y \) can be achieved that leads to a characterization of \( \Re(L)^\perp \) as well as \( \Re(L)^\perp \).

**Definition 3.12.** Let \( X \) and \( Y \) be linear vector spaces and \( L : X \to Y \) be a linear transformation, then the **adjoint** of \( L \) is \( L^* \) such that \( [Lx, y] = [x, L^*y] \) holds for all \( x \in X \) and \( y \in Y \).

**Example 3.8.** If \( L : \mathbb{R}^3 \to \mathbb{R}^2 \) is defined by the matrix of Example 3.7, and the inner product is defined as in Equation (3.3), determine the adjoint, \( L^* : \mathbb{R}^2 \to \mathbb{R}^3 \).

**Solution:** Since \( Lx = \begin{bmatrix} 5x^{(1)} + x^{(2)} + 0x^{(3)} \\ 10x^{(1)} + 2x^{(2)} + 0x^{(3)} \end{bmatrix} \) one finds that

\[
[Lx, y] = \begin{bmatrix} 5 \end{bmatrix} \begin{bmatrix} 5x^{(1)} + x^{(2)} + 0x^{(3)} \\ 10x^{(1)} + 2x^{(2)} + 0x^{(3)} \end{bmatrix} = \begin{bmatrix} 5 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} = [x, L^*y]
\]

Thus,

\[
L^*y = \begin{bmatrix} 5 \end{bmatrix} \begin{bmatrix} 5x^{(1)} + 10x^{(2)} \\ 10x^{(1)} + 25x^{(2)} \\ 3x^{(1)} + 5x^{(2)} \end{bmatrix} = \begin{bmatrix} 5 & 10 \\ 1 & 2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix}
\]

and indicates that the matrix associated with \( L^* \) is \( \begin{bmatrix} 5 & 10 \\ 1 & 2 \\ 0 & 0 \end{bmatrix} \).

**Corollary 3.6.** If a linear transformation \( L^* : \mathbb{C}^m \to \mathbb{C}^n \) is defined by a matrix \( M \) and the inner product is defined as in Equation (3.3), then the matrix defining \( L^* \) is the complex conjugate transpose of \( M \) or \( M = M^\dagger \).

**Theorem 3.2.** Let \( X \), \( Y \) be linear vector spaces and \( L : X \to Y \) be a linear transformation. Then the following hold:

(i) \( \Re(L)^\perp = \Re(L^*) \) and \( \Re(L) \oplus \Re(L^*) = Y \).

(ii) \( \Re(L)^\perp = \Re(L^*) \) and \( \Re(L) \oplus \Re(L^*) = X \)

**Proof.** Part (i): If \( y \in \Re(L)^\perp \), then \( [Lx, y] = 0 \) for all \( x \in X \), since \( Lx \in \Re(L) \). Then, the definition of adjoint indicates that \( [x, L^*y] = 0 \). Since this holds for all \( x \in X \), one
must conclude that \( L^* y = 0 \), or \( y \in \mathbb{K}(L^*) \). Thus, \( \mathbb{K}(L)^\perp \subseteq \mathbb{K}(L^*) \). To show that \( \mathbb{K}(L^*) \subseteq \mathbb{K}(L)^\perp \), assume \( y \in \mathbb{K}(L^*) \) which implies \( 0 = [x, L^* y] = [L x, y] \) for all \( x \in X \), and finally \( y \in \mathbb{K}(L)^\perp \). Thus, \( \mathbb{K}(L)^\perp = \mathbb{K}(L^*) \). Then, since \( \mathbb{K}(L) \oplus \mathbb{K}(L)^\perp = Y \), one also concludes that \( \mathbb{K}(L) \oplus \mathbb{K}(L^*) = Y \).

Part (ii): see Exercise 3.7

**Example 3.9.** If \( L^* : \mathbb{R}^3 \to \mathbb{R}^2 \) is defined by the matrix of Example 3.7, and the inner product is defined as in Equation (3.3), determine the \( \mathbb{K}(L) \) and \( \mathbb{K}(L)^\perp \).

**Solution:** From Example 3.8, we know that the adjoint of \( L \) is the matrix

\[
M^T = \begin{bmatrix} 5 & 10 \\ 1 & 2 \\ 0 & 0 \end{bmatrix}
\]

By inspection we find that \( \mathbb{K}(L^*) = s\text{pan}\{[-2 \ 1]^T\} \). Thus, any \( y \) that is perpendicular to \([-2 \ 1]^T\) will be an element of \( \mathbb{K}(L) \). Clearly, the set of vectors perpendicular to \([-2 \ 1]^T\) is \( s\text{pan}\{[1 \ 2]^T\} \), which indicates that \( s\text{pan}\{[1 \ 2]^T\} = \mathbb{K}(L)^\perp = \mathbb{K}(L) \). Using Corollary 3.5, one finds that \( \mathbb{K}(L^*) = s\text{pan}\{[5 \ 1 \ 0]^T, [10 \ 2 \ 0]^T\} = s\text{pan}\{[5 \ 1 \ 0]^T\} \). Clearly, the set of vectors perpendicular to \([5 \ 1 \ 0]^T\) is \( s\text{pan}\{[0 \ 0 \ 1]^T, [1 \ -5 \ 0]^T\} = \mathbb{K}(L^*)^\perp = \mathbb{K}(L) \). □

It should be emphasized that the calculation of the adjoint and the orthogonal complement are both dependent on the specific inner product that is used. For example, use of the inner product of Example 3.3 will result in the adjoint of a matrix being something other than just the complex conjugate transpose. However, Theorem 3.2 will hold regardless of the inner products used. This is because both the adjoint and orthogonal complement simultaneously will change accordingly when using a different inner product (see Exercise 3.22).

**Corollary 3.7.** If a linear transformation is defined by a matrix \( M \), and the inner product is defined as in Equation (3.3), then using Corollaries 3.5 and 3.6 along with Theorem 3.2 one concludes that \( \mathbb{K}(L) = (s\text{pan}\{\overline{\mathbb{T}}_1, \overline{\mathbb{T}}_2, \ldots, \overline{\mathbb{T}}_n\})^\perp \), where \( \overline{\mathbb{T}}_i \) are the complex conjugates of the rows of \( M \). More specifically, \( x \in \mathbb{K}(L) \) if \( x \perp \overline{\mathbb{T}}_i \), for all \( i \), or equivalently \( [x, \overline{\mathbb{T}}_i] = 0 \) for all \( i \).

Summarizing the sub-section, Theorem 3.1 tells us that a solution \( x \) will exist for all \( b \), if \( \mathbb{K}(L) = Y \). If \( L \) is defined by a matrix \( M \), then Corollary 3.5 indicates that \( \mathbb{K}(L) = Y \) is equivalent to \( s\text{pan}\{c_1, c_2, \ldots, c_n\} = Y \), where \( c_i \) are the columns of \( M \). Similarly, Theorem 3.1 tells us that a solution \( x \) will be unique, if \( \mathbb{K}(L) = 0 \), which by Theorem 3.2 is equivalent to \( \mathbb{K}(L^*) = X \). Thus, if \( L \) is defined by a matrix \( M \), then Corollary 3.5 indicate that \( \mathbb{K}(L^*) = X \) is equivalent to \( s\text{pan}\{\overline{\mathbb{T}}_1, \overline{\mathbb{T}}_2, \ldots, \overline{\mathbb{T}}_n\} = X \), where \( \overline{\mathbb{T}}_i \) are the complex conjugates of the rows of \( M \).

Thus, for the case of \( L^* : \mathbb{C}^m \to \mathbb{C}^n \), we have boiled the conditions of Theorem 3.1 down to verifying if the span of a set of vectors is equal to the entire space (or equivalently if the set of vectors spans the entire space). In the next section, we will describe two systematic methods of performing this verification. However, before doing so, let us develop a simple pretest exclusion condition. That is, if the pretest condition is satisfied, then we will know immediately that the span of this set of vectors cannot be equal to the entire space. If \( L^* : \mathbb{C}^m \to \mathbb{C}^n \), then \( Y = \mathbb{C}^n \) and \( Y \) has a dimension on \( n \). Then, since at least \( n \) vectors are required to span all of \( Y \), it is easily concluded that \( m \) must be greater than or equal to \( n \) before there is any hope of these \( m \) vectors spanning all of \( Y \).
Thus, if $m < n$, then we know that $span\{c_1, c_2, \ldots, c_n\} \neq Y$. Similarly, since $X = \mathbb{C}^m$, the condition $n < m$ will guarantee that $span\{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n\} \neq X$.

As a final point, the condition $m < n$ does not preclude there being a solution to $Lx = b$, since the actual condition is $b \in \mathbb{R}(L)$. If $span\{c_1, c_2, \ldots, c_m\} \neq Y$, then one may still want to check if $b \in span\{c_1, c_2, \ldots, c_m\}$. While the first method of the next section (the Gram-Schmidt procedure) can be used to check if $span\{c_1, c_2, \ldots, c_m\} = span\{c_1, c_2, \ldots, c_m, b\}$, the second method (Singular Value Decomposition) will characterize $\mathbb{R}(L)$ directly. Given this characterization, along with Theorem 3.2, it will be easy to verify if by checking if $b \in \mathbb{R}(L^*)^\perp$, or equivalently if $b \perp \mathbb{R}(L^*)$.

### 3.4 - Linear Independence and Orthogonalization

While the characterization of the subspace spanned by a set of vectors has its origin in the notion of linear independence and more specifically the notion of a basis set, we will find that the procedure of orthogonalization will provide computationally based answers to these questions. Specifically, what is the smallest number of vectors that can be used to span a subspace? Given this number we will be able to ascertain if the span of a set of vectors is the entire space, and if not what is the most compact characterization of the subspace. At the end of the section a method will be provided that gives directly the four characteristic subspaces of Theorem 3.2.

**Definition 3.13.** A set of vectors $\phi_1, \phi_2, \ldots, \phi_n$ are **linearly independent** if the only way to arrive at $\sum_{i=1}^{n} \alpha_i \phi_i = 0$ is to set $\alpha_i = 0$ for all $i$.

**Theorem 3.3.** Consider a linear transformation $L^*: \mathbb{C}^m \rightarrow \mathbb{C}^n$ defined by a matrix $M$ with columns $c_1, c_2, \ldots, c_m$ and rows $r_1, r_2, \ldots, r_n$. Then,

1. $\mathbb{R}(L)$ is empty if and only if $c_1, c_2, \ldots, c_m$ are linearly independent.
2. $\mathbb{R}(L) = \mathbb{C}^n$ if and only if $\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n$ are linearly independent.

**Proof.** Part (i): results directly from the definition linear independence and that of the null space. Part (ii): From Theorem 3.2, we know that $\mathbb{R}(L) = \mathbb{R}(L^*)^\perp$. Thus, $\mathbb{R}(L) = \mathbb{C}^n$ if and only if $\mathbb{R}(L^*)^\perp$ is empty. And, $\mathbb{R}(L^*)^\perp$ is empty if and only if $\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n$ are linearly independent.

While Theorem 3.3 gives another perspective on the conditions of Theorem 3.1 (the Fundamental Theorem of Linear Algebra), it shifts the original question to that of linear independence. One way to answer the question of linear independence is through orthogonalization. We begin with a couple of definitions.

**Definition 3.14.** A set of vectors, $\phi_1, \phi_2, \ldots, \phi_n$, is **orthogonal** if $[\phi_i, \phi_j] = 0$ for all $i \neq j$.

**Definition 3.15.** A set of vectors, $\phi_1, \phi_2, \ldots, \phi_n$, is **orthonormal** if

$$[\phi_i, \phi_j] = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

**Theorem 3.4.** If a set of vectors $\phi_1, \phi_2, \ldots, \phi_n$ is orthogonal, then they are linearly independent.
3.4. Linear Independence and Orthogonalization

**Proof.** Assume $\phi_1, \phi_2, \ldots, \phi_n$ are orthogonal. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be such that $\sum_{i=1}^{n} \alpha_i \phi_i = 0$. Then, for all $j$, $\left[ \sum_{i=1}^{n} \alpha_i \phi_i, \phi_j \right] = [0, \phi_j] = 0$. Since $\phi_1, \phi_2, \ldots, \phi_n$ are orthogonal, $\alpha_j [\phi_j, \phi_j] = 0$. Since each $\phi_j$ is non-zero, we conclude that $\alpha_j = 0$ for all $j$. □

It should be emphasized that the converse of Theorem 3.4 is not true. Specifically, linear independence does not guarantee orthogonality.

![Figure 3.2. Candidate vectors for Example 3.10](image)

**Example 3.10.** Let $X = \mathbb{R}^2$ and consider the vectors of Figure 3.2. In case (a), the two vectors are linearly independent. It is noted that the two vectors are not orthogonal and need not be to be linearly independent. In case (b), the two vectors occupy the same subspace and thus are not linearly independent. In the context of Definition 3.13, take $\alpha_1 = 1/|\phi_1|$ and $\alpha_2 = -1/|\phi_2|$ to find the sum equal to zero, where $|\phi_j| = [\phi_j, \phi_j]^{1/2}$ is the length of each vector. In case (c), the three vectors are not linearly independent. In general, if the number of vectors is greater than the dimension of the space, then they are not linearly independent. In cases (d), the two vectors are orthogonal and as indicated by Theorem 3.4, must be linearly independent. Of course, the orthogonality of case (d), assumes the inner product of Equation 3.3 is used. If some other inner product was used, then the convention of a right angle may not indicate orthogonality. In fact, for an appropriately defined inner product, the vectors of case (a) could be orthogonal. However, those of cases (b) and (c) could never be orthogonal, regardless of the inner product used. ■

**Definition 3.16.** Consider a set of vectors $\phi_1, \phi_2, \ldots, \phi_n$, then the following **Gram-Schmidt Orthogonalization procedure** will generate a set of orthogonal vectors $q_1, q_2, \ldots, q_n$ such that
span\{q_1, q_2 \ldots q_n\} = \text{span}\{\phi_1, \phi_2 \ldots \phi_n\}. The procedure is as follows:

\begin{align*}
q_1 &= \phi_1 \\
q_2 &= \phi_2 - s_{2,1} \\
q_3 &= \phi_3 - s_{3,1} - s_{3,2} \\
&\vdots \\
q_n &= \phi_n - s_{n,1} - s_{n,2} \cdots - s_{n,n-1}
\end{align*}

where

\[ s_{i,j} = \begin{cases} 
[\phi_i, q_j] / [q_j, q_j] q_j & \text{if } [q_j, q_j] \neq 0 \\
0 & \text{if } [q_j, q_j] = 0 
\end{cases} \]

More compactly the algorithm can be stated as

\[ q_i = \phi_i - \sum_{j=1}^{i-1} s_{i,j} \quad i = 1 \ldots n \]

The particular benefit of the Gram-Schmidt procedure is that during execution one may find that one or more of the resulting \( q_i \) vectors is zero. If this is the case, then the original set of vectors \( \phi_1, \phi_2 \ldots \phi_n \) are not linearly independent. However, if all of the resulting \( q_i \) vectors are non-zero, then the original set is linearly independent.

**Example 3.11.** Let \( X = \mathbb{R}^2 \) and reconsider the vectors of Figure 3.2. If the inner product of Equation 3.3 is used and the Gram-Schmidt procedure is applied to cases (a), (b) and (c), then the result will be cases (d), (e) and (f), respectively.

To be more specific, consider case (a) and let \( \phi_1 = [2 \ 1]^T \) and \( \phi_2 = [-4 \ 1]^T \). Since \( q_1 = [2 \ 1]^T \) we find that

\[ s_{2,1} = \frac{[\phi_2, q_1]}{[q_1, q_1]} q_1 = \frac{-7}{5} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -14/5 \\ -7/5 \end{bmatrix} \]

and

\[ q_2 = \phi_2 - s_{2,1} = \begin{bmatrix} -4 \\ 1 \end{bmatrix} - \begin{bmatrix} -14/5 \\ -7/5 \end{bmatrix} = \begin{bmatrix} -6/5 \\ 12/5 \end{bmatrix}. \]

Then, one can verify that \( q_1 = [2 \ 1]^T \) and \( q_2 = [-6/5 \ 12/5]^T \) are orthogonal by calculating \( [q_1, q_2] = -12/5 + 12/5 = 0 \).

In case (b), \( \phi_1 = [2 \ 1]^T \) and \( \phi_2 = [4 \ 2]^T \). Since, \( q_1 = [2 \ 1]^T \) we find that

\[ s_{2,1} = \frac{[\phi_2, q_1]}{[q_1, q_1]} q_1 = \frac{10}{5} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \]

and

\[ q_2 = \phi_2 - s_{2,1} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} - \begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \]

As such, \( q_2 \) is not included in case (e). If there were an additional vector, say \( \phi_3 = [-4 \ 1]^T \), then in the calculation of \( q_3 \) one would set \( s_{3,2} = 0 \), since \( [q_2, q_2] = 0 \).
In case (c), \( \phi_1 = [2 \ 1]^T \), \( \phi_2 = [-4 \ 1]^T \) and \( \phi_3 = [-2 \ -2]^T \). From case (a) we know that \( q_1 = [2 \ 1]^T \) and \( q_2 = [-6/5 \ 12/5]^T \). Then,

\[
s_{3,1} = \frac{[\phi_3, q_1]}{[q_1, q_1]} q_1 = \frac{-6}{5} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -12/5 \\ -6/5 \end{bmatrix}
\]

and

\[
s_{3,2} = \frac{[\phi_3, q_2]}{[q_2, q_2]} q_2 = \frac{-12/5}{36/5} \begin{bmatrix} -6/5 \\ 12/5 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} -6/5 \\ 12/5 \end{bmatrix} = \begin{bmatrix} 2/5 \\ -4/5 \end{bmatrix}
\]

Finally,

\[
q_3 = \phi_3 - s_{3,1} - s_{3,2} = \begin{bmatrix} -2 \\ -2 \end{bmatrix} - \begin{bmatrix} -12/5 \\ -6/5 \end{bmatrix} - \begin{bmatrix} 2/5 \\ -4/5 \end{bmatrix} = \begin{bmatrix} -10/5 \\ -10/5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.
\]

As such, \( q_3 \) is not included in case (f).

Now return to the question of determining the range space of a matrix \( M : \mathbb{C}^m \to \mathbb{C}^n \), \( \mathcal{R}(M) = \text{span}\{c_1, c_2 \ldots c_m\} \), and if that range space is equal to \( \mathbb{C}^n \). Begin by applying the Gram-Schmidt procedure to the columns of \( M \) and remove all of the zero vectors from the resulting \( q_i \) set. If the number of remaining vectors is \( r \), then the set will be \( q_1, q_2 \ldots q_r \). If \( r \) is less than \( n \), then \( \mathcal{R}(M) = \text{span}\{c_1, c_2 \ldots c_m\} = \text{span}\{q_1, q_2 \ldots q_r\} \) is a subset of \( \mathbb{C}^n \). If \( r \) is equal to \( n \), then Theorem 3.4 tells us that \( \mathcal{R}(M) = \mathbb{C}^n \). The third option, of \( r \) being greater than \( n \), is not possible, since the number of linear independent vectors cannot be larger than the dimension of the space. It is noted that the number \( r \), is actually equal to the number of linear independent columns of \( M \). The following definition gives a name to this quantity.

**Definition 3.17.** The **rank** of a matrix \( M : \mathbb{C}^m \to \mathbb{C}^n \), denoted \( \text{rank}(M) \), is equal to the number of linearly independent columns of \( M \), \( c_1, c_2 \ldots c_m \).

Our second computational method, the singular value decomposition, is quite a bit more convenient, as it will give directly the four fundamental subspaces of Theorem 3.2, if the linear transformation is a matrix. While the computational details of this procedure are outside of our scope, the conceptual utility of this decomposition is undeniable. The following definitions will be essential to understanding the singular value decomposition.

**Definition 3.18.** A linear transformation, \( U : X \to X \), is **unitary** if \( UU^* = U^*U = I \).

**Definition 3.19.** A set of vectors \( \phi_1, \phi_2 \ldots \phi_n \) in a linear vector space \( X \) is a **basis set for \( X \)**, if \( \phi_1, \phi_2 \ldots \phi_n \) are linearly independent and \( \text{span}\{\phi_1, \phi_2 \ldots \phi_n\} = X \).

Based on Definitions 2.5 and 3.18, it is clear that the inverse of a unitary transformation is equal to its adjoint, i.e., \( U^{-1} = U^* \). Furthermore, if given a unitary matrix \( U : \mathbb{C}^n \to \mathbb{C}^n \), then it is easily concluded that the columns of \( U \) are an orthonormal basis set for \( \mathbb{C}^n \).
Example 3.12. Reconsider the vectors of Figure 3.2. Since the span of the two vectors of case (a) is $X$, these two may serve as a basis set. The vectors of the other two cases may not serve as a basis set for $X$. In case (b), the span of the vectors is not $X$, and in case (c) the vectors are not linearly independent.

Definition 3.20. The **Singular Value Decomposition** of a matrix, $M : \mathbb{C}^m \to \mathbb{C}^n$, is composed of three matrices $U$, $V$ and $S$ with the following properties:

(i) $M = U S V^*$

(ii) $U : \mathbb{C}^n \to \mathbb{C}^n$ and is unitary ($U U^* = U^* U = I$)

(iii) $V : \mathbb{C}^m \to \mathbb{C}^m$ and is unitary ($V V^* = V^* V = I$)

(iv) $S : \mathbb{C}^m \to \mathbb{C}^n$ and is of the form

$$
\Sigma = \begin{bmatrix}
\sigma_1 & 0 & 0 \\
0 & \ddots & 0 \\
0 & 0 & \sigma_r 
\end{bmatrix}
$$

where $\Sigma$ is a square diagonal matrix

and $r = \text{rank}(M)$.

The diagonal elements of $\Sigma$ are known as the singular values of $M$. These singular values are all real and are usually arranged in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r > 0$. Of greater interest is the fact that the columns of $V$ form an orthonormal basis for $\mathbb{C}^m$. Then it is easily concluded that the span of last $m - r$ columns of $V$ is the null space of $M$. Then, Theorem 3.2 tells us that the span of the remaining columns of $V$, the first $r$ columns, is the range space of $M$. A similar analysis of $M^* = V S U^*$ indicates that the first $r$ columns of $U$ span the range space of $M^*$ and the remaining span the null space of $M^*$. This decomposition of the four fundamental spaces of Theorem 3.2 is summarized by the following Corollary.

Corollary 3.8. Consider a singular value decomposition of a matrix, $M : \mathbb{C}^m \to \mathbb{C}^n$, as described in Definition 3.20. If the columns of $U$ are denoted as $u_i$, $i = 1 \ldots m$ and the columns of $V$ are $v_i$, $i = 1 \ldots m$, then

- (i) $\mathbb{R}(M) = \text{span}\{u_1, \ldots, u_r\}$ and $\mathbb{R}(M^*) = \text{span}\{u_{r+1}, \ldots, u_m\}$
- (ii) $\mathbb{R}(M^*) = \text{span}\{v_1, \ldots, v_r\}$ and $\mathbb{R}(M) = \text{span}\{v_{r+1}, \ldots, v_m\}$

It is noted that the MATLAB function `svd` provides the Singular Value Decomposition.

Example 3.13. Consider a matrix

$$
M = \begin{bmatrix}
5 & 1 & 0 \\
10 & 2 & 0 
\end{bmatrix}
$$
3.4. Linear Independence and Orthogonalization

Use of the MATLAB function ‘[U,S,V]=svd(M)’, gives:

\[
U = \begin{bmatrix}
-0.4472 & -0.8944 \\
-0.8944 & 0.4472 \\
-0.9806 & -0.1961 \\
-0.1961 & 0.9806 \\
0 & 0
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
11.4 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
-0.9940 & -0.1091 & 0.9940 \\
-0.1091 & 0.9940 & 0.3227 & 0.8452 \\
0.78581 & 0.5950 & 0.1690 \\
0.4483 & -0.7361 & 0.5071
\end{bmatrix}
\]

Clearly, both \(U\) and \(V\) are unitary. Based on the value determined for \(S\), we conclude that \(r = \text{rank}(M) = 1\). Thus,

\[
\Re(M) = \text{span}\{u_1, \ldots, u_r\} = \text{span}\{[-0.4472 -0.8944]^T\} = \text{span}\{[1 -2]^T\}
\]

\[
\Im(M^*) = \text{span}\{v_{r+1}, \ldots, v_n\} = \text{span}\{[-0.3227 0.8452]^T\} = \text{span}\{[5 1 3]^T\}
\]

\[
\Re(M^*) = \text{span}\{v_1, \ldots, v_r\} = \text{span}\{[0.9940 -0.1091]^T, [0 0 1]^T\}
\]

These results are in alignment with those of Examples 3.7 and 3.9.

Example 3.14. Consider a matrix

\[
M = \begin{bmatrix}
2 & -4 & -2 \\
1 & 1 & -2
\end{bmatrix}
\]

Use of the MATLAB function ‘[U,S,V]=svd(M)’, gives:

\[
U = \begin{bmatrix}
-0.9940 & -0.1091 \\
-0.1091 & 0.9940 \\
-0.4261 & 0.3227 & 0.8452 \\
0.78581 & 0.5950 & 0.1690 \\
0.4483 & -0.7361 & 0.5071
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
4.9213 & 0 & 0 \\
0 & 2.4043 & 0
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
-0.9940 & -0.1091 & 0.9940 \\
-0.1091 & 0.9940 & 0.3227 & 0.8452 \\
0.78581 & 0.5950 & 0.1690 \\
0.4483 & -0.7361 & 0.5071
\end{bmatrix}
\]

Based on the value determined for \(S\), we conclude that \(r = \text{rank}(M) = 2\). Thus,

\[
\Re(M) = \text{span}\{u_1, \ldots, u_r\} = \text{span}\{[-0.9940 -0.1091]^T, [-0.1091 0.9940]^T\} = \Re^2
\]

\[
\Im(M^*) = \text{span}\{v_{r+1}, \ldots, v_n\} = \{0\}
\]

\[
\Re(M^*) = \text{span}\{v_1, \ldots, v_r\} = \text{span}\{[0.8452 0.1690 0.5071]^T\}
\]

\[
\Re(M^*) = \text{span}\{v_1, \ldots, v_r\} = \text{span}\{[5 1 3]^T\}^\perp
\]

\[
\Im(M^*) = \text{span}\{v_{r+1}, \ldots, v_m\} = \text{span}\{[5 1 3]^T\}^\perp
\]

These results are in alignment with those of Examples 3.7 and 3.9.


Matrix Inverse and the Determinant

Let us return to the fundamental theorem of linear algebra (Theorem 3.1). It provides conditions such that a unique solution to the equation \( Lx = b \) will exist for all \( b \). Specifically, the two conditions for an operator \( L : X \to Y \) are "\( \mathcal{R}(L) = Y \) implies existence for all \( b \)" and "\( \mathcal{N}(L) = \{0\} \) implies uniqueness of the solution". If a unique solution will exist for each \( b \), then there must be a unique transformation \( L^{-1} \) such that \( x = L^{-1}b \). In addition, based on the original equation, \( Lx = b \), this \( L^{-1} \) must be such that \( L^{-1}L = I \), where \( Ix = x \) for all \( x \in X \). For the case of \( L \) being defined by a matrix \( M : \mathbb{C}^m \to \mathbb{C}^n \), the first condition, \( \mathcal{R}(M) = Y \), translates to \( \text{span}\{c_1, c_2, \ldots, c_m\} \) must be equal to \( \mathbb{C}^n \). Furthermore, the second condition, \( \mathcal{N}(M) = \mathcal{N}(M^\dagger) = \{0\} \), translates to \( \text{span}\{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n\} \) must be equal to \( \mathbb{C}^m \). Clearly, the first condition requires \( m \geq n \). That is, if \( m < n \), then it would be impossible for \( \text{span}\{c_1, c_2, \ldots, c_m\} \) to be equal to \( \mathbb{C}^n \). Similarly, the second condition requires \( n \geq m \). Thus, the prerequisite requirement for the inverse, \( M^{-1} \), to exists is that the matrix must be square \( (m = n) \) or \( M \) be \( \mathbb{C}^n \to \mathbb{C}^n \).

For a square matrix, \( M \), the condition \( \text{span}\{c_1, c_2, \ldots, c_m\} = \mathbb{C}^n \) is equivalent to requiring that the columns of \( M \) are linearly independent. However, Theorem 3.3 indicates that the condition of linearly independent columns is equivalent to \( \mathcal{N}(M) = \{0\} \). Thus, in the case a square matrix, the one condition of linearly independent columns will satisfy both conditions of the fundamental theorem of linear algebra and guarantee the existence of a matrix inverse.

While the previous paragraph is sufficient to state the desired result, it will be instructive to also consider \( \mathcal{N}(M) = \{0\} \), or equivalently \( \mathcal{N}(M^\dagger) = \text{span}\{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n\} = \mathbb{C}^m \). In this case, we find that linearly independent complex conjugate rows will satisfy this condition, and thus guarantee uniqueness of a solution to \( Mx = b \). However, referring again to Theorem 3.3 we find that \( \overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n \) being linearly independent is equivalent to \( \mathcal{R}(M) = \mathbb{C}^n \) and guarantee existence of a solution to \( Mx = b \), regardless of \( b \). Thus, we find that linearly independent complex conjugate rows is also enough to guarantee the existence of a matrix inverse. In fact, it can be shown (see Theorem 3.12) that the number of linearly independent columns is always equal to the number of linearly independent complex conjugate rows.

The above discussion is summarized by the following Theorem.

Theorem 3.5. Given a matrix \( M : \mathbb{C}^n \to \mathbb{C}^n \), the following conditions are equivalent.

(i) The inverse of \( M \), \( M^{-1} \), exists.
(ii) The columns of \( M \) are linearly independent.
(iii) \( \mathcal{N}(M) \) is empty.
(iv) \( \det(M) \neq 0 \)

Condition (iv) is likely the most common condition for the existence of a matrix inverse. To show that \( \det(M) \neq 0 \) should be part of the list we will show that it is equivalent to condition (ii). However, the definition of the \( \det(M) \) must first be provided.

Definition 3.21. The determinant of a matrix \( M : \mathbb{C}^n \to \mathbb{C}^n \), denoted \( \det(M) \), is given by the following formula, where \( m_{ij} \) is the matrix element at the \( i \)th row and \( j \)th column.

\[
\det(M) = \sum_{i=1}^{n} m_{ij} C_{ij} \text{ for any } j, \text{ or }
\]
The determinant of a matrix $M$ is defined as:

$$
\det(M) = \sum_{i=1}^{n} m_{ij} C_{ij} \text{ for any } i
$$

The terms $C_{ij}$ are denoted as cofactors and are given by:

$$
C_{ij} = (-1)^{i+j} \det(M_{ij})
$$

The terms $M_{ij}$ are denoted as minors and each is equal to the matrix $M$, with the $i$th row and $j$th column removed. Finally, $\det(M_{ij}) = m_{ij}$.

It is immediately observed that $\det(M^T) = \det(M)$ and $\det(M^*) = \overline{\det(M)}$. For the case of $n = 2$ and $j = 1$:

$$
\det(M) = m_{11} m_{22} - m_{21} m_{12}.
$$

For the case of $n = 3$ and $j = 1$:

$$
\det(M) = m_{11}(m_{22} m_{33} - m_{32} m_{23}) - m_{21}(m_{12} m_{33} - m_{32} m_{13}) + m_{31}(m_{12} m_{23} - m_{22} m_{13})
$$

Since each cofactor will require another determinant calculation, the nested nature of the procedure indicates that the calculation will quickly become tedious for $n$ larger than 3. However, for matrices with convenient structures the calculations can be significantly reduced. For example, consider a matrix of the following block form:

$$
M = \begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix}
$$

(3.4)

where the submatrices, $M_{11}$, $M_{12}$, $M_{21}$, $M_{22}$ are of appropriate dimension. If $M$ is block triangular (i.e., either $M_{12}$ is a zero matrix or $M_{21}$ is a zero matrix) then, the following is easily verified: $\det(M) = \det(M_{11}) \det(M_{22})$. This result can be extended to the case of a full triangular matrix (i.e., one with all zeros above and/or below the diagonal). In this case the determinant is just the product of the diagonal elements. An immediate consequence being: $\det(I) = 1$.

Another convenient property is that the determinant of the product is equal to the product of the determinants: $\det(M_1 M_2) = \det(M_1) \det(M_2)$. Of course this requires both matrices to be square. Using these relations, one finds that $\det(M^{-1}) = 1/\det(M)$, since $1 = \det(I) = \det(M^{-1} M) = \det(M^{-1}) \det(M)$.

Returning to the proof of Theorem 3.5, the following Lemma will be of great utility

**Lemma 3.1.** Consider a matrix $M : \mathbb{C}^n \to \mathbb{C}^n$. Then the following hold:

$$
\sum_{i=1}^{n} m_{ij} C_{ik} = \begin{cases} 
\det(M) & \text{if } j = k \\
0 & \text{if } j \neq k
\end{cases}
$$

and

$$
\sum_{j=1}^{n} m_{ij} C_{kj} = \begin{cases} 
\det(M) & \text{if } i = k \\
0 & \text{if } i \neq k
\end{cases}
$$

As an illustration, consider the case of $n = 3$, $k = 1$ and $j = 2$. In this case, the resulting sum

$$
\sum_{i=1}^{3} m_{i2} C_{i1} = m_{12}(m_{22} m_{33} - m_{32} m_{23}) - m_{22}(m_{12} m_{33} - m_{32} m_{13}) + m_{32}(m_{12} m_{23} - m_{22} m_{13})
$$

Chapter 3. Review of Linear Algebra

is easily shown to be zero.

**Proof of Theorem 3.5.** Based on the discussion preceding Theorem 3.5, it is concluded that parts (i), (ii) and (iii) are equivalent. To complete the proof we will need to show that (ii) and (iv) are equivalent. Essentially, we will need to prove the following two statements: (1) If \( \det(M) \neq 0 \), then the columns of \( M \) are linearly independent, and (2) If the columns of \( M \) are linearly independent, then \( \det(M) \neq 0 \). However, it will be more convenient to show the following equivalent statements: (1a) If the columns of \( M \) are not linearly independent, then \( \det(M) = 0 \), and (2a) If \( \det(M) = 0 \), then the columns of \( M \) are not linearly independent. Starting with (1a), assume the columns of \( M \), denoted \( c_1, c_2, \ldots, c_n \), are not linearly independent. This implies there exists \( \alpha_1, \alpha_2, \ldots, \alpha_n \) such that \( c_1 = \sum_{j=2}^{n} \alpha_j c_j \). This is equivalent to \( m_{i1} = \sum_{j=2}^{n} \alpha_j m_{ij} \), where \( m_{ij} \) is the matrix element at the \( i \)th row and \( j \)th column. Thus,

\[
\det(M) = \sum_{i=1}^{n} m_{i1} C_{i1} = \sum_{i=1}^{n} \left( \sum_{j=2}^{n} \alpha_j m_{ij} \right) C_{i1} = \sum_{i=1}^{n} \alpha_j \left( \sum_{i=1}^{n} m_{ij} C_{i1} \right)
\]

Since Lemma 3.1 tells us that \( \sum_{i=1}^{n} m_{ij} C_{i1} = 0 \) for all \( j \neq 1 \), we conclude that \( \det(M) = 0 \).

To prove (2a), begin by assuming \( \det(M) = 0 \), which implies \( m_{i1} = -\frac{1}{c_{i1}} \sum_{j=2}^{n} m_{ij} C_{ij} \). If \( \alpha_j = C_{ij}, j = 1 \ldots n \), then \( \sum_{j=1}^{n} \alpha_j c_j = 0 \) is easily verified using \( \sum_{j=1}^{n} m_{ij} C_{ij} = 0 \) if \( i \neq 1 \).

The following Theorem makes good use of Theorem 3.5 and is of independent interest.

**Theorem 3.6.** If \( T \) is invertible, then \( \text{rank}(TM) = \text{rank}(M) \).

**Proof.** Let us define \( \text{rank}(M) \overset{\triangle}{=} r_1 \) and \( \text{rank}(TM) \overset{\triangle}{=} r_2 \). Then, we must show that \( r_1 = r_2 \).

Let \( c_i \) be the columns of \( M \), then \( Tc_i \) are the columns of \( TM \). Now assume \( r_2 < r_1 \), which requires that for all rearrangements of the columns of \( TM \), there exist non-zero \( \alpha_i \) such that \( \sum_{i=1}^{r_2} \alpha_i Tc_i = 0 \). However, this implies that \( T \left( \sum_{i=1}^{r_1} \alpha_i c_i \right) = 0 \) and since \( T \) is invertible and does not have a null space, it is concluded that for all rearrangements of the columns of \( M \), there exist non-zero \( \alpha_i \) such that \( \sum_{i=1}^{r_1} \alpha_i c_i = 0 \), which contradicts \( \text{rank}(M) = r_1 \). Thus, \( r_2 \geq r_1 \). Using similar arguments one may establish that \( r_1 \geq r_2 \).

We are now in a position to define a formula for the calculation of the inverse of a matrix. This calculation will require the use of the classical adjoint, which should not be confused with the adjoint given in Definition 3.12.

**Definition 3.22.** The classical adjoint of a matrix \( M : \mathbb{C}^n \rightarrow \mathbb{C}^n \), denoted \( \text{adj}(M) \), is given by the transpose of the matrix of cofactors

\[
\text{adj}(M) = \begin{bmatrix}
C_{11} & \cdots & C_{1n} \\
\vdots & \ddots & \vdots \\
C_{n1} & \cdots & C_{nn}
\end{bmatrix}
\]

Using Lemma 3.1, the following Corollary is easily proved.

**Corollary 3.9.** The inverse of a matrix, \( M : \mathbb{C}^n \rightarrow \mathbb{C}^n \), is \( M^{-1} = \text{adj}(M)/\det(M) \).
3.6 • Eigenvector Decomposition and Diagonalization

The eigenvector decomposition will be central to our understanding of how a linear dynamic system behaves. Most importantly, the associated eigenvalues will characterize the stability of the system, as will be discussed in Section 4.1.

Definition 3.23. Consider a square linear transformation \( L : X \rightarrow X \). An eigenvalue-eigenvector pair consists of a scalar \( \lambda \in \mathbb{C} \) and a vector \( \phi \in X \) such that \( L\phi = \lambda \phi \).

If one is given an eigenvalue-eigenvector pair, \((\lambda, \phi)\), then by Definition 3.23 one finds \( \phi \in \mathbb{N}(\lambda I - L) \). Similar to the definition of the null space, the zero vector cannot be an eigenvector. It is also customary to normalize the eigenvectors (i.e., \( [\phi, \phi] = 1 \)), although this is certainly not required. In the case of the linear transformation being a matrix, \( M : \mathbb{C}^n \rightarrow \mathbb{C}^n \), we are looking for the null space of \( \lambda I - M \) to not be empty (i.e., the exists \( \phi \) such that \( \phi \in \mathbb{N}(\lambda I - L) \)). Theorem 3.5 tells us that this condition is equivalent to \( \det(\lambda I - M) = 0 \).

Definition 3.24. Consider a square matrix transformation \( M : \mathbb{C}^n \rightarrow \mathbb{C}^n \). Then, the nth order polynomial \( \Delta(\lambda) = \det(\lambda I - M) = 0 \) is known as the characteristic equation.

Since the characteristic equation is an nth order polynomial, there should be \( n \) eigenvalue-eigenvector pairs, \((\lambda_j, \phi_j)\), \( j = 1 \ldots n \). Furthermore, since an nth order polynomial may have complex roots, both the eigenvalues and eigenvectors can have complex values, even if the matrix contains only real elements. However, if all the elements of the matrix \( M \) have real values, then the all the coefficients of the characteristic polynomial will be real. If a polynomial has real coefficients, then it is well-known that all of its complex roots must appear as complex conjugate pairs. Thus, if a matrix has real elements, then its eigenvalues must be real or appear as complex conjugate pairs.

The following three Theorems will be of great utility in Chapter 4.

Theorem 3.7. If a matrix \( M : \mathbb{C}^n \rightarrow \mathbb{C}^n \) has all real elements, then \( M \) and \( M^* \) have the same eigenvalues.

Proof. If \( M \) has all real elements, then
\[
\det(\lambda I - M^*) = \det(\lambda I - M^T) = \det((\lambda I - M)^T) = \det((\lambda I - M))
\]
Thus, the characteristic polynomial of \( M \) and \( M^\ast = M^T \) are the same and will generate the same eigenvalues. \( \Box \)

Theorem 3.8. Consider a matrix of the following block form:
\[
M = \begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix}
\]
with submatrices, \( M_{11}, M_{12}, M_{21}, M_{22} \), of appropriate dimension. If \( M \) is block triangular (i.e., either \( M_{12} \) is a zero matrix or \( M_{21} \) is a zero matrix), then the eigenvalues of \( M \) are equal to the eigenvalues of \( M_{11} \) combined with those of \( M_{22} \).

Proof. If \( M \) is block triangular, then \( \det(M) = \det(M_{11})\det(M_{22}) \) and the characteristic
Definition 3.25. If $T$ is invertible, then $TMT^{-1}$ is denoted as similar to $M$.

Theorem 3.9. Similar matrices have the same eigenvalues.

Proof. Consider $(\lambda, \phi)$ such that $TMT^{-1}\phi = \lambda\phi$. Then, clearly $(\lambda, T^{-1}\phi)$ is an eigenvalue-eigenvector pair for $M$. □

Example 3.15. Consider the following matrix $M : \mathbb{C}^2 \rightarrow \mathbb{C}^2$, $M = \begin{bmatrix} -1 & 2 \\ -1 & -4 \end{bmatrix}$. Then, the characteristic equation is

$$0 = \Delta(\lambda) = (\lambda + 1)(\lambda + 4) - (-1)(2) = \lambda^2 + 5\lambda + 6 = (\lambda + 2)(\lambda + 3)$$

Thus, the eigenvalues are $\lambda_1(M) = -2$ and $\lambda_2(M) = -3$. Then, $\phi_1$ is found as the non-zero (or non-trivial) solution to $0 = (\lambda_1 I - M)\phi_1 = \begin{bmatrix} -1 & -2 \\ 1 & 2 \end{bmatrix} \phi_1$, which is $\phi_1 = [2 \ -1]^T$.

Similarly, $0 = (\lambda_2 I - M)\phi_2 = \begin{bmatrix} -2 & -2 \\ 1 & 1 \end{bmatrix} \phi_2$, implies $\phi_2 = [1 \ -1]^T$. □

Example 3.16. Consider a matrix $TMT^{-1}$, where $M = \begin{bmatrix} -1 & 2 \\ -1 & -4 \end{bmatrix}$ and $T = \begin{bmatrix} 2 & 1 \\ -1 & -1 \end{bmatrix}$. It is easily verified that $T^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix}$, and thus $TMT^{-1} = \begin{bmatrix} -3 & -3 \\ 0 & -2 \end{bmatrix}$. Using Theorem 3.8, we conclude that $\lambda_1(TMT^{-1}) = -2$ and $\lambda_2(TMT^{-1}) = -3$. Of course, given Theorem 3.9 it is no surprise that the similar matrices $M$ and $TMT^{-1}$ have the same eigenvalues. It is also not surprising to find that another similar matrix $TMT^{-1}$ also has the same eigenvalues. The more interesting point, however, is the fact that $TMT^{-1} = \begin{bmatrix} -2 & 0 \\ 0 & -3 \end{bmatrix}$ is diagonal, which can be exploited to achieve a computational advantage. □

Motivated by Example 3.16, let us consider a square matrix transformation $M : \mathbb{C}^n \rightarrow \mathbb{C}^n$ and arrange the eigenvalue-eigenvector pairs $(M\phi_i = \lambda_i \phi_i)$ into a matrix form: $M\Phi = \Phi \Lambda$, where $\Phi$ has columns consisting of the eigenvectors, $\Phi = [\phi_1 \ \phi_2 \ \cdots \ \phi_n]$, and $\Lambda$ is a diagonal matrix with elements consisting of the eigenvalues; $\Lambda = \text{diag} \{ [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_n] \}$. Then, the eigenvector decomposition is achieved by simply multiplying both sides by $\Phi^{-1}$:

$M = \Phi \Lambda \Phi^{-1}$ or equivalently $\Lambda = \Phi^{-1}M\Phi$

Given the decomposition $M = \Phi \Lambda \Phi^{-1}$, a number of matrix calculations become rather simple. The following two are easily verified:

$M^i = \Phi \Lambda^i \Phi^{-1}$, where $\Lambda^i = \text{diag} \{ (\lambda_1)^i \ (\lambda_2)^i \ \cdots \ (\lambda_n)^i \}$

e$^M = \Phi e^\Lambda \Phi^{-1}$, where $e^\Lambda = \text{diag} \{ e^{\lambda_1} \ e^{\lambda_2} \ \cdots \ e^{\lambda_n} \}$
Of course, this decomposition assumes that the inverse of $\Phi$ exists, in which case the matrix $M$ is said to be \textit{diagonalizable}. The following theorem, in combination with Theorem 3.5, gives a sufficient condition for a matrix to be diagonalizable (i.e., $\Phi^{-1}$ exists).

\textbf{Theorem 3.10.} Consider a square matrix transformation $M : \mathbb{C}^n \rightarrow \mathbb{C}^n$, and assume it has $n$ distinct eigenvalues (i.e., $\lambda_i \neq \lambda_j$ for all $i \neq j$). Then, the collection of $n$ eigenvectors is linearly independent.

\textbf{Proof.} Assume the eigenvectors are not linearly independent. Then we will need to show that the eigenvalues are not distinct (i.e., there exists $i \neq j$ such that $\lambda_i = \lambda_j$). Denote the largest number of linearly independent eigenvectors as $p$. That is, any collection of $p + 1$ eigenvectors will be linearly dependent. Since the eigenvectors are not linearly independent, this $p$ must be such that $1 \leq p \leq n$. Thus, there exists non-zero $\alpha_j$ such that $\sum_{j=1}^{p+1} \alpha_j \phi_j = 0$. If multiplied by $M$, one finds $\sum_{j=1}^{p+1} \alpha_j \lambda_j \phi_j = 0$. Multiplying the first equation by $\lambda_{p+1}$ and subtracting from the second equation, gives $\sum_{j=1}^{p} \alpha_j (\lambda_j - \lambda_{p+1}) \phi_j = 0$. Since this collection of $p$ eigenvectors is linearly independent, $\alpha_j (\lambda_j - \lambda_{p+1})$ must equal zero for all $j$. Since at least one $\alpha_j$ must be non-zero, we conclude that $\lambda_j = \lambda_{p+1}$ for that $j$. Thus, there exists $i \neq j$ such that $\lambda_i = \lambda_j$, which completes the proof. \hfill $\Box$

If a matrix does not have distinct eigenvalues, then it is still possible for the matrix to be diagonalizable. Again, this hinges on the linear independence of the eigenvectors associated with the repeated eigenvalues. For cases in which the repeated eigenvalues actually share the same eigenvector, one can construct what is known as the \textit{Jordan form}, which is close to a diagonal form. Specifically, $M = \Phi J \Phi^{-1}$ (or equivalently $J = \Phi^{-1} M \Phi$) where the columns of $\Phi$ are composed of generalized eigenvectors, which are constructed to be linearly independent. In this case, the matrix $J$ will also have eigenvalues along the diagonal, but may also have “1” values in the diagonal just above the main diagonal. In this case, we will again have $M' = \Phi J' \Phi^{-1}$ and $e^{\Phi} = \Phi e^{\Phi} \Phi^{-1}$, but $J'$ and $e'$ will be more difficult to calculate. However, given the particular form of this calculation is far from tedious.

\textbf{Example 3.17.} (a) Consider a matrix

$$M = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 3 & -1 \\ 0 & 0 & 2 \end{bmatrix}.$$  

Clearly the eigenvalues of $M$ are $\lambda_1 = 1$, $\lambda_2 = 2$ and $\lambda_3 = 3$. In addition it is found that $\phi_1 = [1 \ 0 \ 0]^T$, $\phi_2 = [1 \ 1 \ 1]^T$ and $\phi_3 = [1 \ 1 \ 0]^T$. Thus, one would construct the eigenvalue decomposition as

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad \Phi = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$  

A similar result is obtained using the MATLAB function `[Phi, Lam] = eig(M)`, which
results in:

\[
\text{Lam} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 3 & 0 \\
0 & 0 & 2 \\
\end{bmatrix} \quad \text{Phi =} \begin{bmatrix}
1 & 0.7071 & 0.5774 \\
0 & 0.7071 & 0.5774 \\
0 & 0 & 0.5774 \\
\end{bmatrix}.
\]

Although that the MATLAB function normalized the eigenvectors and switched the order of the eigenvalues, the result is the same. In both cases, one can verify that \( M \) is equal to \( \Phi \Lambda \Phi^{-1} \) and \( \Phi_i \text{ norm} \text{Phi} \).

(b) Now consider the matrix

\[
M = \begin{bmatrix}
1 & 1 & 0 \\
0 & 2 & 0 \\
0 & 0 & 2 \\
\end{bmatrix}.
\]

In this case use the MATLAB function ‘[Phi, Lam] = eig(M)’, gives

\[
\text{Lam} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 2 \\
\end{bmatrix} \quad \text{Phi =} \begin{bmatrix}
1 & 0.7071 & 0 \\
0 & 0.7071 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}.
\]

However, since the eigenvalues are not distinct, there is a chance that \( M \) may not be diagonalizable. However, since the columns of \( \Phi_i \) are linearly independent (i.e., \( \text{inv}(\Phi_i) \) exists), these matrices can be used for the eigenvalue decomposition.

(c) Now consider the matrix

\[
M = \begin{bmatrix}
1 & 2 & -1 \\
0 & 3 & -1 \\
0 & 1 & 1 \\
\end{bmatrix}.
\]

In this case, use of the MATLAB function ‘[Phi, Lam] = eig(M)’, gives

\[
\text{Lam} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 2 \\
\end{bmatrix} \quad \text{Phi =} \begin{bmatrix}
1 & 0.5774 & 0.5774 \\
0 & 0.5774 & 0.5774 \\
0 & 0.5774 & 0.5774 \\
\end{bmatrix}.
\]

Again, we find repeated eigenvalues, but this time the two eigenvectors are the same. Thus, the columns of \( \Phi_i \) are not linearly independent and \( \text{inv}(\Phi_i) \) does not exist. To determine the Jordan form, one may use the MATLAB function ‘[Phi,J]=jordan(M)’, which gives

\[
J = \begin{bmatrix}
1 & 0 & 0 \\
0 & 2 & 1 \\
0 & 0 & 2 \\
\end{bmatrix} \quad \text{Phi =} \begin{bmatrix}
-1 & 1 & 1 \\
0 & 1 & 1 \\
0 & 1 & 0 \\
\end{bmatrix}.
\]

Clearly, this \( \Phi_i \) is invertible and one may easily verify that \( \Phi_i \ast J \ast \text{inv}(\Phi_i) \) is equal to \( M \).

As a final note, we could have used the function ‘jordan’ in parts (a) and (b) of this example. In fact, doing so would have results in more natural eigenvectors, in that they would not have been normalized. However, the downside of the function ‘jordan’ is that the algorithm it uses is based on symbolic manipulations. In contrast, ‘eig’ uses a purely numeric algorithm, and thus runs much faster, especially for large matrices. ☐
The following much celebrated Cayley-Hamilton Theorem will be an essential tool for the developments of Chapter 4.

**Theorem 3.11.** A matrix $M : \mathbb{C}^n \to \mathbb{C}^n$ satisfies its own characteristic equation. That is, if

$$\Delta(\lambda) = \det(\lambda I - M) = \lambda^n - a_{n-1}\lambda^{n-1} + \cdots + a_1\lambda + a_0 = 0$$

then

$$\Delta(M) = M^n - a_{n-1}M^{n-1} + \cdots + a_1M + a_0I = 0_{n \times n}$$

**Proof.** To simplify the proof, we will additionally assume the matrix $M$ is diagonalizable (for the full proof please see Chen (1999)). As such, each power of $M$ can be written as $M^i = \Phi\Lambda^i\Phi^{-1}$. Thus, $\Delta(M) = \Phi\Delta(\Lambda)\Phi^{-1}$, where $\Delta(\Lambda) = \text{diag} \{\Delta(\lambda_1), \Delta(\lambda_2), \ldots, \Delta(\lambda_n)\}$. Since each eigenvalue satisfies the characteristic equation, one finds $\Delta(\Lambda) = 0_{n \times n}$, which leads to $\Delta(M) = 0_{n \times n}$ and completes the proof. $\square$

The utility of the Cayley-Hamilton Theorem is the identity

$$M^n = -\sum_{i=0}^{n-1} a_i M^i.$$ 

Thus, any matrix polynomial $\rho(M)$, regardless of degree, can converted to the form of

$$\rho(M) = b_{n-1}M^{n-1} + \cdots + b_1M + b_0I$$

Most notably, since $e^M$ is just a matrix polynomial, though with infinite degree, it too can be converted to a polynomial of degree $n-1$. It is also noted that the Cayley-Hamilton Theorem is particularly useful in calculating the powers of a Jordan form matrix. Specifically, the $J^i$ and $e^J$ found in $M^i = \Phi J^i \Phi^{-1}$ and $e^M = \Phi e^J \Phi^{-1}$. For the details of these useful conversions, please see Chen (1999).

In subsequent chapters following trace operator will serve to simplify some of the developed relations.

**Definition 3.26.** The trace of a square matrix $M$ is defined as the sum of the diagonal elements:

$$\text{Tr} \{M\} = \sum_{i=1}^{n} m_{ii} \quad (3.6)$$

Among the most useful properties of the trace operator is that the order of matrix products can be ‘wrapped around’ if within the trace operator. Specifically:

$$\text{Tr} \{ABC\} = \text{Tr} \{BCA\} = \text{Tr} \{CAB\} \quad (3.7)$$

Using the (generalized) eigenvector decomposition, one can easily prove the following two corollaries.

**Corollary 3.10.** The trace of a matrix is equal to the sum of its eigenvalues:

$$\text{Tr} \{M\} = \sum_{i=1}^{n} \lambda_i \quad (3.8)$$
Corollary 3.11. The determinant of a matrix is equal to the product of its eigenvalues:

\[ \det(M) = \prod_{i=1}^{n} \lambda_i \]  

(3.9)

3.7 Non-Square Systems of Equations

In this section, we will focus on relations of the form \( Mx = b \), but remove the assumption of the matrix \( M \) being square. We will start with a few technical results that will be required later in the section. The following corollary is a generalization of a concept that is commonly encountered in an introductory physics course.

Corollary 3.12. If \( \phi_1, \phi_2, \ldots, \phi_n \) is a basis set for \( X \), then all \( x \in X \) can be represented as a linear combination of the basis vectors \( \phi_1, \phi_2, \ldots, \phi_n \).

Corollary 3.12 states that for all \( x \in X \) there exists a projection, \( \alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T \), such that:

\[ x = \sum_{i=1}^{n} \alpha_i \phi_i \]  

(3.10)

If \( X = \mathbb{C}^n \) and \( \Phi = [\phi_1, \phi_2, \ldots, \phi_n] \), then Equation (3.11) can be written as \( x = \Phi \alpha \). Since a set of basis vectors must be linearly independent, it is known that \( \Phi^{-1} \) exists and the projection can be calculated as \( \alpha = \Phi^{-1}x \). If the basis set is orthonormal, then \( \Phi^{-1} = \Phi^* \), \( \alpha = \Phi^* x \), and the elements of \( \alpha \) are calculated as \( \alpha_i = [x, \phi_i] \), assuming the inner product is defined as in Equation (3.3). This case of using an orthonormal basis is equivalent to the introductory physics notion of “projecting a vector into unit vector directions”. In this case of an orthonormal basis set, Equation (3.11) is usually stated as

\[ x = \sum_{i=1}^{n} [x, \phi_i] \phi_i \]  

(3.11)

If a vector is in the subspace spanned by a subset of the basis vectors, then the projection coefficients associated with the other vectors will be zero. Specifically, if \( x \in \text{span}\{\phi_1, \phi_2, \ldots, \phi_m\} \), \( m < n \), then \( \alpha = 0 \) for \( i > m \). This convenient property will be used in the proof of the following Lemma.

Lemma 3.2. Consider a matrix, \( M : \mathbb{C}^m \rightarrow \mathbb{C}^n \). Then, the following hold

(i) \( \mathbb{R}(MM^*) = \mathbb{R}(M) \)

(ii) \( \mathbb{N}(M^*M) = \mathbb{N}(M) \)

Proof. (i) The range space is defined as the set of vectors \( y = Mx \), where \( x \) can be any vector in \( \mathbb{C}^m \). However, the set of available vectors, \( x \), need not be the entire space \( \mathbb{C}^m \). Consider a basis set for the null space of \( M \), along with a basis set its orthogonal complement. In the notation of Corollary 3.8, these are \( \mathbb{N}(M) = \text{span}\{v_{r+1}, \ldots, v_m\} \) and \( \mathbb{N}(M)^\perp = \text{span}\{v_1, \ldots, v_r\} \). Then, from Corollary 3.12 we know that for any \( x \in \mathbb{C}^m \), there exists \( \alpha_i \) such that \( x = \sum_{i=1}^{m} \alpha_i v_i \). Thus, \( y = Mx = \sum_{i=1}^{r+1} \alpha_i M v_i = Mx' \) where \( x' \in \mathbb{R}(M^*) \). Thus, when calculating \( \mathbb{R}(M) \), one need only consider vectors \( x \in \mathbb{R}(M^*) \). Since
the operation $x = M^* y$ can reach all vectors in $\mathfrak{R}(M^*)$, we find that $\mathfrak{R}(M M^*) = \mathfrak{R}(M)$. To show (ii), begin by observing that $x \in \mathfrak{N}(M)$ implies $x \in \mathfrak{N}(M^* M)$. Thus, it only remains to show that $x \in \mathfrak{N}(M^* M)$ implies $x \in \mathfrak{N}(M)$, which is equivalent to $x \notin \mathfrak{N}(M)$ implies $x \notin \mathfrak{N}(M^* M)$. If $x \notin \mathfrak{N}(M)$, then $Mx \in \mathfrak{R}(M)$. Since $\mathfrak{R}(M) = \mathfrak{N}(M)^\perp$, we observe that $Mx \notin \mathfrak{N}(M^*)$, which implies $x \notin \mathfrak{N}(M^* M)$ and completes the proof.

Using Lemma 3.2 we can return to the question of the number of linearly independent columns and row of a non-square matrix.

**Theorem 3.12.** For any matrix, $M : \mathbb{C}^m \to \mathbb{C}^n$, the number of linearly independent columns is equal to the number of linearly independent complex conjugate rows.

**Proof.** The statement is equivalent to ‘$M$ and $M^*$ have an equal number of linearly independent columns’. From Lemma 3.2, we know that $\mathfrak{R}(M) = \mathfrak{R}(M M^*)$ and $\mathfrak{R}(M^*) = \mathfrak{R}(M^* M)$. Thus, the statement is equivalent to ‘$M M^*$ and $M^* M$ have an equal number of linearly independent columns’. Now, let $\lambda$ be a non-zero eigenvalue of $M M^*$ with $\phi$ as the associated eigenvector, which gives: $M M^* \phi = \lambda \phi$. Multiplying by $M^*$ results in $M M^* (M^* \phi) = \lambda (M^* \phi)$, which indicates that $M^* \phi$ is an eigenvector of $M^* M$ with an eigenvalue of $\lambda$. This indicates that $M M^*$ and $M^* M$ have the same non-zero eigenvalues, and thus have the same number of non-zero eigenvalues. The proof is completed by noting that for a square matrix the number of linearly independent columns is equal to the number of non-zero eigenvalues.

### 3.7.1 Least Squares Solutions and the Pseudo-Inverse

We can now return to the subject of a non-square system of equations, $Mx = b$, where $M : \mathbb{C}^m \to \mathbb{C}^n$. In general, there are two cases. The first is when the number of equations, $n$, is greater than the number of variables, $m$. Such a set of equations is denoted as overdetermined, and while a solution might exist (if $b \in \mathfrak{R}(M)$), there are physically meaningful situations in which no solution exists. If this is the case, then we will look for a vector $\hat{x}$ that results in $M \hat{x}$ being as close as possible to $b$. The second case is when the number of equations, $n$, is less than the number of variables, $m$. Equations of this class are denoted as underdetermined, and while the set of solutions might be unique (if $\mathfrak{N}(M) = \{0\}$), there are physically meaningful situations in which an infinite number of solutions are possible. If this is the case, then a systemic methodology of selecting one of these solutions will be desired, and one may also be interested in characterizing all possible solutions.

![Figure 3.3: Simple process flow diagram.](image-url)
To help illustrate the overdetermined case, consider the flow network of Figure 3.3. The material balance equations are found to be: \( s_3 + s_4 - s_3 = 0 \) and \(-s_1 - s_2 - s_3 + s_5 = 0 \). Or equivalently:

\[
0 = \begin{bmatrix}
0 & 0 \\
-1 & -1 
\end{bmatrix} s_1 + \begin{bmatrix}
0 & \quad 1 \\
-1 & \quad -1 
\end{bmatrix} s_2 + \begin{bmatrix}
1 & \quad 0 \\
0 & \quad 1 
\end{bmatrix} s_3 + \begin{bmatrix}
-1 
\end{bmatrix} s_5
\] (3.12)

Let us now suppose that four of these flow rates are measured (\( s_1 = 5 \), \( s_2 = 7 \), \( s_3 = 2 \) and \( s_4 = 12 \)) and we would like to calculate \( s_5 \). In this case, the set of equations to be solved can be stated as \( M \hat{x} = b \), where

\[
M = \begin{bmatrix}
1 \\
-1 
\end{bmatrix}, \quad x = s_5,
\]

and

\[
b = \begin{bmatrix}
0 \\
-1 
\end{bmatrix} s_1 + \begin{bmatrix}
0 \\
-1 
\end{bmatrix} s_2 + \begin{bmatrix}
1 \\
0 
\end{bmatrix} s_3 + \begin{bmatrix}
14 \\
-14 
\end{bmatrix}
\] (3.13)

Since \( b \in \mathbb{R}(M) \), Theorem 3.1 tells us that a solution will exist, which we trivially find to be \( x = 14 \). Let us now assume there is a bit of error in each of the measurements \( s_1 = 5.2 \), \( s_2 = 6.9 \), \( s_3 = 2.1 \) and \( s_4 = 11.5 \). In this case, we find that \( b = [13.6 \; -14.2]^T \). Clearly, this \( \hat{b} \) is not in the range space of \( M \), and thus we should not expect there to be a solution. As an alternative, we seek \( \hat{x} \) such that \( M \hat{x} \) is as close as possible to \( \hat{b} \). In particular, one could form an optimization problem to minimize the quantity \((M \hat{x} - b)^T(M \hat{x} - b)\). Since this scalar quantity is just the inner product of the residual, \((M \hat{x} - b)\), with itself, this optimization is frequently denoted as the least squares problem (i.e., minimizing the sum of the squared errors). Taking the derivative of this function, equal to \( \hat{x}^T M^* M \hat{x} - 2 \hat{x}^T M^* b + b^T b \), with respect to \( \hat{x} \), one finds \( 2 M^* M \hat{x} = 2 M^* b \). Setting this derivative equal to zero, one arrives at the following set of relations, known as the normal equations:

\[
M^* M \hat{x} = M^* b
\] (3.14)

If the inverse of \( M^* M \) exists, then \( \hat{x} = (M^* M)^{-1} M^* b \). The expression \((M^* M)^{-1} M^* \) is known as the Moore-Penrose pseudo-inverse. Of course, the pseudo-inverse will exist only if the inverse of \( M^* M \) exists.

**Theorem 3.13.** Consider a matrix, \( M : \mathbb{C}^m \rightarrow \mathbb{C}^n \). The inverse of \( M^* M \) will exist if and only if \( M \) has linearly independent columns.

**Proof.** From Theorem 3.5 it is known that inverse of \( M^* M \) will exist if and only if \( \mathbb{N}(M^* M) = 0 \). However, from Lemma 3.2 it is known that \( \mathbb{N}(M^* M) = \mathbb{N}(M) \). Finally, from Theorem 3.3 it is known that \( \mathbb{N}(M) \) is empty if and only if \( M \) has linearly independent columns. \( \square \)

While the optimization based derivation of the pseudo-inverse is fairly simple, the following graphical interpretation gives much more insight into the origin of the normal equations of Equation (3.14). Begin by noting that for all \( \hat{x}, M \hat{x} \) must be in the range space of \( M \). In the example above, this is calculated as \( M \hat{x} \in \mathbb{R}(M) = \text{span} \{[1 \; -1]^T \} \), which is indicated as the dashed of Figure 3.4. If the vector \( b \) is not in the range space of \( M \), then it will not coincide with this line. Then, the question is to select \( \hat{x} \) such that \( M \hat{x} \) is as close as possible to \( b \). Based on Figure 3.4, this will be the point such that the residual, \( M \hat{x} - b \), is perpendicular to \( \mathbb{R}(M) \), or equivalently \( M \hat{x} - b \in \mathbb{R}(M)^\perp \). However, from Theorem 3.2 we know that \( \mathbb{R}(M)^\perp = \mathbb{R}(M^*) \). Thus, \( \hat{x} \) is such that \( M \hat{x} - b \in \mathbb{R}(M^*) \), or
3.7. Non-Square Systems of Equations

equivalently the normal equations: \( M^* (M\hat{x} - b) = 0 \). The fact that \( M\hat{x} - b \) is orthogonal (or normal) to \( \mathcal{R}(M) \) is the origin of the terminology ‘normal equations’. In Chapter 5 we will find a generalization of the least squares problem and find that similar notions of orthogonality will provide significant insight into the problem characteristics.

Table 3.1. Test Data for Example 3.18.

<table>
<thead>
<tr>
<th>Independent Variable</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1: ( f(t) )</td>
<td>2</td>
<td>7</td>
<td>16</td>
<td>29</td>
<td>46</td>
</tr>
<tr>
<td>Test 2: ( f(t) )</td>
<td>2.2</td>
<td>7.5</td>
<td>14.5</td>
<td>30.5</td>
<td>44.5</td>
</tr>
</tbody>
</table>

Example 3.18. Suppose you have collected the data of Table 3.1. It is postulated that the function \( f(t) \) is of quadratic form \( f(t) = a_2 t^2 + 4a_1 t + 16a_0 \) and the question is to determine the coefficients \( a_2, a_1, \) and \( a_0 \). If we assume \( f(t) \) is of quadratic form, then the data of Test 1 can be written in the following matrix form: \( Mx = b_1 \), where

\[
M = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
4 & 2 & 1 & 1 & 1 \\
9 & 3 & 1 & 1 & 1 \\
16 & 4 & 1 & 1 & 1 \\
25 & 5 & 1 & 1 & 1 \\
\end{bmatrix}, \quad x = \begin{bmatrix}
a_2 \\
a_1 \\
a_0 \\
\end{bmatrix}, \quad \text{and} \quad b_1 = \begin{bmatrix}
2 \\
7 \\
16 \\
29 \\
46 \\
\end{bmatrix}
\]

If by some miracle you select \( a_2 = 2, a_1 = -1 \) and \( a_0 = 1 \), it can be shown that \( b_1 \in \mathcal{R}(M) \). Of course, selecting these values by inspection is nearly impossible. However, one may employ the pseudo-inverse:

\[
(M^* M)^{-1} M^* = \begin{bmatrix}
979 & 225 & 55 \\
225 & 55 & 15 \\
55 & 15 & 5 \\
\end{bmatrix} M^*
\]

\[
= \begin{bmatrix}
0.143 & -0.0714 & -0.143 & -0.0714 & 0.143 \\
-1.057 & 0.3286 & 0.857 & 0.5286 & -0.657 \\
1.800 & 0.000 & -0.800 & -0.600 & 0.600 \\
\end{bmatrix}
\]
Then, one would find that

$$\hat{x}_1 = (M^*M)^{-1}M^*b_1 = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

If this $\hat{x}_1$ is then multiplied by $M$, then the original $b_1$ would be recovered indicating that indeed $b_1 \in \mathbb{R}(M)$. If we now look at Test 2, then the matrix form is $M\hat{x}_2 = b_2$, where $M$ is as before and $b_2 = [2.2 \ 7.5 \ 14.5 \ 30.5 \ 44.5]^T$. In this case, application of the pseudo-inverse yields:

$$\hat{x}_2 = (M^*M)^{-1}M^*b_2 = \begin{bmatrix} 1.886 \\ -0.554 \\ 0.760 \end{bmatrix}$$

To illustrate the graphical interpretation, note that:

$$M\hat{x}_2 - b_2 = \begin{bmatrix} -0.1086 \\ -0.3057 \\ 1.5686 \\ -1.7857 \\ 0.6314 \end{bmatrix}$$

Then, it is easily verified that $M^*(M\hat{x}_2 - b_2) = 0$, indicating that $M\hat{x}_2 - b_2 \in \mathbb{R}(M^*)$. As a final note, the pseudo-inverse is made possible by the fact that the columns of $M$ are linearly independent. If this were not the case, then we would find that $(M^*M)^{-1}$ does not exists. In a more practical sense, we would likely be warned by MATLAB that our matrix $M^*M$ is ‘close to singular or badly scaled’ and the inversion results are likely to be inaccurate. If such a warning occurs, then it is likely that the columns of $M$ are not linearly independent. As an illustration, consider the case of taking only the first 2 data points of both tests, for a total of 4 data points. In this case, $M$ would be defined as

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 1 & 1 & 1 \\ 4 & 2 & 1 \end{bmatrix}$$

Clearly the first two columns of $M$ are not linearly independent. This can be verified by trying to calculate $(M^*M)^{-1}$, which will give a close to singular warning. Section 3.7.3 will discuss how to interpret such a system and how to find a least squares solution. However, such methods should only be used if one is sure that an appropriate formulation of the problem has been given. In most cases, this issue of linearly dependent columns is the result of a flawed formulation of the problem, a typographical error or most likely bug in the code used to generate the system of equations.

3.7.2 • Non-uniqueness and the Minimum Norm Solution

Now return to the case of an underdetermined set of equations: $Mx = b$ ($M : \mathbb{C}^m \rightarrow \mathbb{C}^n$), where the number of equations, $n$, is less than the number of variables, $m$. In this case, it is likely that there will be an infinite number of solutions and one may be interested in characterizing all possible solutions. From the fundamental theorem of linear algebra
3.7. Non-Square Systems of Equations

(Theorem 3.1), we know that the set of solutions will be non-unique if and only if the null space of $M$ is not empty (or $\mathbb{N}(M) \neq \{0\}$). So, one way to characterize all solutions is to start with a solution, $\hat{x}$, such that $M\hat{x} = b$. Then, all $x$ such that $Mx = b$ can be defined as:

$$x = \hat{x} + \sum_{i=1}^{p} \alpha_i \phi_i$$  \hspace{1cm} (3.16)

where $\alpha_i$ are arbitrarily selected scalars and $\phi_i$ are basis vectors for the null space, i.e., $\mathbb{N}(M) = \text{span}\{\phi_1, \phi_2, \ldots, \phi_p\}$. The first question one might ask about this characterization is how does one go about finding the first solution $\hat{x}$. To develop a procedure, let us begin by assuming that $\hat{x}$ is known and that the $\phi_i$ vectors are orthonormal. Then, we can look for $\alpha_i$ values such that the size of $x = \hat{x} + \sum_{i=1}^{p} \alpha_i \phi_i$ is as small as possible. More appropriately, find $\alpha_i$ such that the size of the inner product of $x$ with itself (the square of the norm of $x$) is as small as possible. Specifically, seek to minimize

$$f(\alpha_1, \alpha_1, \ldots, \alpha_p) = x^* x = \left( \hat{x} + \sum_{i=1}^{p} \alpha_i \phi_i \right)^* \left( \hat{x} + \sum_{i=1}^{p} \alpha_i \phi_i \right)$$

$$= \hat{x}^* \hat{x} + 2 \sum_{i=1}^{p} \alpha_i \phi_i^* \hat{x} + \sum_{i=1}^{p} \alpha_i^2$$  \hspace{1cm} (3.17)

Taking the derivative of this expression with respect to each $\alpha_i$ and setting each equal to zero we find the following relations

$$0 = \frac{\partial f}{\partial \alpha_i} = 2 \phi_i^* \hat{x} + 2 \alpha_i$$  \hspace{1cm} (3.18)

Thus, the $\alpha_i$’s that minimize Equation (3.17) are those such that $\alpha_i = -\phi_i^* \hat{x}$ for all $i = 1 \ldots p$.

If we now turn the problem around and let $\hat{x}$ be the variable while setting the $\alpha_i$’s to zero, we find that the smallest $x$ is equal to $\hat{x}$ (from Equation (3.16) with $\alpha_i = 0$), and $\hat{x}$ is characterized as being such that $\phi_i^* \hat{x} = 0$. Of course, these relations indicate that must be orthogonal to the null space of $M$, or equivalently $\hat{x} \in \mathbb{N}(M)^\perp = \mathbb{R}(M^*)$, where the last equality is from Theorem 3.2. Thus, there must exist a $y$ such that $M^* y = \hat{x}$. If this relation is substituted into the original expression $M\hat{x} = b$, we find the dual of the normal equations: $MM^* y = b$. If the inverse of $MM^*$ exists, then $y = (MM^*)^{-1} b$ and ultimately

$$\hat{x} = M^* (MM^*)^{-1} b$$  \hspace{1cm} (3.19)

Note the similarity of (3.19) with the pseudo-inverse. Application of Theorem 3.13 to $M^*$ indicates that the inverse of $MM^*$ will exist if and only if the rows of $M$ are linearly independent.

To illustrate the minimum norm procedure graphically, let use consider the simplest case of one equation and two variables. Specifically, let $M = [1 \ 1]$ and $b = 1$. Then, the linear variety that characterizes all solutions to $Mx = b$ is the dashed line of Figure 3.5, which is parallel to the null space of $M$. Then, the smallest vector in the linear variety is clearly indicated as $\hat{x}$, which by inspection should be perpendicular to the null space ($\hat{x} \in \mathbb{N}(M)^\perp = \mathbb{R}(M^*)$), while also satisfying $M\hat{x} = b$. Based on the Figure 3.5, this point...
should be \( \hat{x} = [0.5 \ 0.5]^T \), which is verified by the minimum norm calculation:

\[
\hat{x} = M^*(MM^*)^{-1}b = \left[ \begin{array}{c} \frac{1}{1} \\ \frac{1}{1} \end{array} \right] \left( [1 \ 1] \left[ \begin{array}{c} \frac{1}{1} \\ \frac{1}{1} \end{array} \right] \right)^{-1} [1] = \left[ \begin{array}{c} 0.5 \\ 0.5 \end{array} \right]
\]

Since the null space of \( M \) is found to be the span of \([-1 \ 1]^T\), we find that the set of all solution is characterized as:

\[
x = \left[ \begin{array}{c} 0.5 \\ 0.5 \end{array} \right] + \alpha \left[ \begin{array}{c} -1 \\ 1 \end{array} \right]
\]

where \( \alpha \) can be any number.

**Example 3.19.** Consider a set of differential equations: \( \dot{s} = As + B m + G p \), where

\[
A = \left[ \begin{array}{cc} -1 & 0 \\ 1 & 0 \end{array} \right], \ B = \left[ \begin{array}{cc} 1 & 0 \\ 1 & -1 \end{array} \right] \quad \text{and} \quad G = \left[ \begin{array}{c} 0 \\ -1 \end{array} \right]
\]

Suppose \( p^{SSOP} \) is given to be 3 and we would like to characterize all possible steady-state conditions \( (s^{SSOP}, m^{SSOP}) \), i.e., such that \( 0 = As^{SSOP} + B m^{SSOP} + G p^{SSOP} \). This can be restated as: find all \( x \) such \( Mx = b \), where

\[
M = \left[ \begin{array}{ccc} -1 & 0 & 1 \\ 1 & 0 & -1 \end{array} \right], \ x = \left[ \begin{array}{c} s^{SSOP} \\ m^{SSOP} \end{array} \right] \quad \text{and} \quad b = \left[ \begin{array}{c} 0 \\ 3 \end{array} \right]
\]

Since the rows of \( M \) are clearly linearly independent, we can use Equation (3.19) to find

\[
\hat{x} = M^*(MM^*)^{-1}b = \left[ \begin{array}{ccc} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{array} \right] \left( \left[ \begin{array}{cc} 2 & 0 \\ 0 & 3 \end{array} \right] \right)^{-1} \left[ \begin{array}{c} 0 \\ 0 \end{array} \right]
\]

\[
= \left[ \begin{array}{ccc} -1/2 & 1/3 & 0 \\ 0 & 0 & 1 \\ 1/2 & 1/3 & 0 \\ 0 & -1/3 & 0 \end{array} \right] \left[ \begin{array}{c} 0 \\ 0 \end{array} \right] = \left[ \begin{array}{c} 1 \\ 0 \\ 0 \\ -1 \end{array} \right]
\]
By inspection one may conclude that the following vectors are in the null space of $M$: \[ 0 1 0 0 \] and \[ 1 0 1 2 \]. Thus, the set of all solutions is found to be
\[
x = \begin{bmatrix} \mathbf{s}_{SSOP}^T \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 0 \\ 1 \\ 2 \end{bmatrix}
\] (3.25)

An alternative approach to solving Example 3.19 is to rearrange the steady-state relation \( 0 = A_s^SSOP + B m^SSOP + G p^SSOP \) to be \( m^SSOP = -B^{-1} (A_s^SSOP + G p^SSOP) \) and find
\[
m^SSOP = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix} \mathbf{s}_{SSOP}^T - \begin{bmatrix} 0 \\ 3 \end{bmatrix}
\]
Then, if \( \mathbf{s}_{SSOP}^T \) is defined as \( \mathbf{s}_{SSOP} = [\alpha_1 \ \alpha_2]^T \), one can conclude that
\[
x = \begin{bmatrix} \mathbf{s}_{SSOP}^T \\ m_{SSOP}^T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -3 \end{bmatrix} + \alpha_1 \begin{bmatrix} 1 \\ 0 \\ 1 \\ 2 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\] (3.26)

While this approach appears to be simpler, it requires the inverse of \( B \) to exist. We did not even attempt to solve for \( \mathbf{s}_{SSOP}^T = -A^{-1} (B m^SSOP + G p^SSOP) \), since the inverse of \( A \) clearly does not exist. If \( B \) was defined as \( B = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \), then its inverse would not exist either, and the above "intuitive" approach would require the use of column 1 from \( A \) and either column from \( B \) to arrive at linearly independent columns and thus an invertible matrix. In the general case, where there may be many large columns to choose from, the selection of appropriate columns to create an invertible matrix is likely to be a process of trial and error. In contrast, the generalized method of Equation (3.16), with defined by Equation (3.19), can be completely automated. The only challenge is to identify a basis set for the null space of \( M \). However, the singular value decomposition along with Corollary 3.8 can always be used to find such a basis set. Using the MATLAB function \('[U,S,V]=svd(M)'\) and noting from the resulting \( S \) that the rank of \( M \) is 2, we conclude that the null space of \( M \) is spanned by columns 3 and 4 of \( V \). Using these columns, the characterization of all solutions would be stated as:
\[
x = \begin{bmatrix} \mathbf{s}_{SSOP}^T \\ m_{SSOP}^T \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0.2357 \\ -0.8165 \\ 0.2357 \\ 0.4714 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0.3333 \\ 0.5773 \\ 0.3333 \\ 0.6667 \end{bmatrix}
\] (3.27)

While this characterization has less intuitive appeal as compared to Equations (3.25) or (3.26), it is exactly equivalent and can be completely automated using MATLAB. The next subsection will illustrate the utility (and in some case the necessity) of the singular value decomposition in solving set of equations.

### 3.7.3 Singular Value Decomposition and Solutions to $Mx = b$

In this subsection, we will begin by illustrating how the singular value decomposition can be used to reduce the computational effort of calculating both the pseudo-inverse and
Chapter 3. Review of Linear Algebra

the minimum norm solution. It will be concluded by showing that appropriate solutions
to $Mx = b$ can be found regardless of rank if given the singular value decomposition.
However, first we will introduce a bit of terminology. In Section 3.7.1, it was noted
that the pseudo-inverse will exist if and only if the matrix $M : \mathbb{C}^m \to \mathbb{C}^n$ is such that its
$m$ columns are linearly independent. A more technical way of stating the condition of
linearly independent columns is that the rank of $M$ is equal to $m$. Such a matrix will be
denoted as having full column rank. In Section 3.7.2, it was found that the minimum
norm solution will exist if and only if its $n$ rows are linearly independent, or its rank is
equal to $n$. Such a matrix will be denoted as having full row rank. Alternatively, we can
combine the above two definitions to arrive at the following generalization that works
for all square and non-square matrices:

**Definition 3.27.** A matrix $M : \mathbb{C}^m \to \mathbb{C}^n$ is denoted as full rank if $\text{rank}(M) = \min(n, m)$.

If $m < n$, then the condition of full rank is equal to full column rank and is sufficient
for the existence of the pseudo-inverse. In addition, if $n < m$, then full rank is the same
as full row rank and will guarantee the existence of a minimum norm solution. In fact, if
$n = m$, then the condition of full rank indicates that the regular inverse will exist (recall
Theorem 3.5).

With the full rank terminology in hand, recall the singular value decomposition of
Definition 3.20: $M = U \Sigma V^*$, where $S$ is of the form $\begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}$ and $\Sigma$ is a square diagonal
matrix with dimension $r = \text{rank}(M)$ and only non-zero values on the diagonal. In
addition, both $U$ and $V$ are unitary and based on Corollary 3.8 have the form of
$U = [U_1 \quad U_2]$ and $V = [V_1 \quad V_2]$, where $\Re(M) = s \text{pan}\{U_1\}$, $\Re(M^*) = s \text{pan}\{U_2\}$, $\Re(M^*) = s \text{pan}\{V_1\}$ and $\Re(M) = s \text{pan}\{V_2\}$.

Let us begin with the overdetermined case: $M : \mathbb{C}^m \to \mathbb{C}^n$ and the number of columns
is less the number of rows ($m < n$). If $M$ is full column rank (i.e., $\Re(M) = \{0\}$), then $S$
will of the form $\begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}$ and $M = U_1 \Sigma V^*$. Substituting this decomposition into the
pseudo-inverse, one finds:

$$(M^*M)^{-1}M^* = (V \Sigma U_1^* U_1 \Sigma V^*)^{-1}V \Sigma U_1^*$$

$$= (V \Sigma^2 V^*)^{-1}V \Sigma U_1^* = V \Sigma^{-2} V^* V \Sigma U_1^* = V \Sigma^{-1} U_1^*$$

(3.28)

Since $\Re(M) = 0$ the resulting least squares solution (which may actually be a solution,
depending on the value of $b$) will be unique. The one questionable part of Equation (3.28)
is that $U_1^* U_1 = I$. You should be able to convince yourself this is true by considering
the example of $U$ equal to a $3 \times 3$ identity matrix and $r = 2$. Then, evaluate $U^* U$ and $U U^*$,
both of which will be equal to identity but will get there in a different ways.

Now consider the underdetermined case: $M : \mathbb{C}^m \to \mathbb{C}^n$ and the number of rows is
less than the number of columns ($n < m$). If $M$ is full row rank (i.e., $\Re(M) = \mathbb{C}^n$), then
$S$ will of the form $\begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}$ and $M = U \Sigma V_1^*$. Substituting this decomposition into the
minimum norm matrix of Equation 3.19, one finds:

$$M^*(M^*M)^{-1} = V_1 \Sigma U_1^*(U \Sigma V_1^* V_1 \Sigma U_1^*)^{-1}$$

$$= V_1 \Sigma U_1^*(U \Sigma^2 U^*)^{-1} = V_1 \Sigma U_1^* U \Sigma^{-2} U^* = V_1 \Sigma^{-1} U^*$$

(3.29)

Similar to Equation (3.28), the one questionable part of Equation (3.29) is that $V_1^* V_1 = I$,
which you should be able to convince yourself is true. It is noted, that since $\Re(M) = \mathbb{C}^n$
there will always be at least one solution to $Mx = b$. However, given that $n < m$ and the number of linearly independent columns must be equal to the number of linearly independent rows (recall Theorem 3.12), we know that $\mathcal{N}(M) = \text{span}\{V_2\} \neq \{0\}$, and we can be confident that this solution will not be unique.

At this point, it will be helpful to review the cases we have considered in the context of the fundamental theorem of linear algebra (Theorem 3.1). If we consider only the matrix $M : \mathbb{C}^m \to \mathbb{C}^n$ (i.e., exclude the impact of the $b$ vector), then there are four possible cases.

1) The first case is when $\mathcal{R}\{M\} = \mathbb{C}^n$ and $\mathcal{N}(M) = \{0\}$. As discussed is Section 3.5, such a situation is only possible if $M$ is square and full rank. In this case, the inverse of $M$ will exist and the solution will be unique.

2) If $\mathcal{R}\{M\} \neq \mathbb{C}^n$ and $\mathcal{N}(M) = \{0\}$, then the set of equation must be overdetermined ($m < n$) and $M$ must be full (column) rank. In this case, a solution may not exist, but we will always be able to calculate a least squares solution, using the pseudo-inverse or Equation (3.28). In either case, the result will be unique, since the pseudo-inverse matrix will be unique.

3) If $\mathcal{R}\{M\} = \mathbb{C}^n$ and $\mathcal{N}(M) \neq \{0\}$, then the set of equation must be underdetermined ($n < m$) and $M$ must be full (row) rank. In this case, an infinite number of solutions will always exist and we can always find the minimum norm solution using Equations (3.19) or (3.29).

4) If $\mathcal{R}\{M\} \neq \mathbb{C}^n$ and $\mathcal{N}(M) \neq \{0\}$, then we have the worst of both worlds - the solution may not exist and it (or the least squares solution) will not be unique. This type of situation can occur for square, underdetermined and overdetermined systems. The following theorem can be applied to all three cases.

**Theorem 3.14.** Consider a non-full rank matrix, $M : \mathbb{C}^m \to \mathbb{C}^n$ (i.e., $\mathcal{R}\{M\} \neq \mathbb{C}^n$ and $\mathcal{N}(M) \neq \{0\}$). Then, the following hold.

(i) The set of all possible least squares solutions to $Mx = b$ can be characterized as $\hat{x} = V_1\Sigma^{-1}U_1^*b + V_2\alpha_2$, where $\alpha_2 = [\alpha^{(r+1)} \alpha^{(r+2)} \ldots \alpha^{(n)}]^T$ is arbitrarily selected and $r$ is the rank of $M$.

(ii) The minimum norm element, selected from the infinite set of least squares solutions, is found to be $\hat{x} = V_1\Sigma^{-1}U_1^*b$ (i.e., when $\alpha_2 = 0$).

**Proof.** (i) The approach is to define $x = V\alpha$ and find the least squares solution with respect to $\alpha$. Then, the appropriate $\hat{x}$ can be determined from $x = V\alpha$, since $V = (V^*)^{-1}$.

Since $M$ is not full rank, $S$ must be of the form $\begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}$ and $M = U_1\Sigma V_1^*$. Using this along with $x = V\alpha$, it is found that:

$$Mx = U_1\Sigma V_1^*V\alpha = U_1\Sigma V_1^*[V_1 \ V_2] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = U_1\Sigma[V_1 \ V_2] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = U_1\Sigma\alpha_1$$

Thus, we are left with an overdetermined system $U_1\Sigma\alpha_1 = b$. However, since $U_1\Sigma$ is full (column) rank, the unique least squares solution is found to be $\alpha_1 = \Sigma^{-1}U_1^*b$. Since $\alpha_2$ is
unspecified we conclude that
\[
\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \Sigma^{-1} U_1^* b \\ \alpha_2 \end{bmatrix} \quad \text{and} \quad \hat{x} = V \begin{bmatrix} \Sigma^{-1} U_1^* b \\ \alpha_2 \end{bmatrix} = V_1 \Sigma^{-1} U_1^* b + V_2 \alpha_2
\]

To verify that this \( \hat{x} \) is indeed the least squares solution, let us verify that the residual, \( M \hat{x} - b \), is indeed in the null space of \( M^* \). First, note that the residual is:
\[
M \hat{x} - b = U_1 \Sigma V_1^*(V_1 \Sigma^{-1} U_1^* b + V_2 \alpha_2) - b = U_1 U_1^* b + U_1 \Sigma V_1^* V_2 \alpha_2 - b = (U_1 U_1^* - I)b
\]

Then, we find that \( M^*(M \hat{x} - b) = V_1 \Sigma U_1^* (U_1 U_1^* - I)b = (V_1 \Sigma U_1^* U_1 U_1^* - V_1 \Sigma U_1^*)b = 0 \).

(ii) To find the minimum norm element, form the scalar function
\[
f(\alpha_2) = \hat{x}^* \hat{x} = (V_1 \Sigma^{-1} U_1^* b + V_2 \alpha_2)^* (V_1 \Sigma^{-1} U_1^* b + V_2 \alpha_2)
\]
\[
= b^* U_1 \Sigma^{-1} V_1^* V_1 \Sigma^{-1} U_1^* b + \alpha_2^* V_2^* V_2 \alpha_2
\]
\[
= b^* U_1 \Sigma^{-1} U_1^* b + \alpha_2^* \alpha_2
\]

Thus, the minimum value of \( f(\alpha_2) \) is obviously achieved at \( \alpha_2 \), which completes the proof. \( \square \)

To help understand Theorem 3.14, let us think about the various cases. If the system is overdetermined and full (column) rank case, then \( \mathcal{N}(M) = 0 \) and the (least squares) solution will be unique. If the full rank assumption is removed (\( \mathcal{N}(M) \neq 0 \)), then it is natural to conclude that there will be an infinite number of least squares solutions. If the system is square, then the logic is similar. The main difference is that a square matrix must be either case 1 (\( \mathcal{N}(M) = C^n \) and \( \mathcal{N}(M) = 0 \)) or case 4 (\( \mathcal{N}(M) \neq C^n \) and \( \mathcal{N}(M) \neq 0 \)). If the system is underdetermined, then the logic is bit strange. If it is full (row) rank, then \( \mathcal{N}(M) = C^n \) and we are guaranteed an infinite number of solutions. However, if the full rank assumption is removed (\( \mathcal{N}(M) \neq C^n \)), then it may be that none of these infinite solutions exist. The remedy is to find the infinite least squares solutions. While this may seem confusing, it is just the steps of the overdetermined case, but in the reverse order. The good news is that Theorem 3.14 covers all three cases. In fact, Theorem 3.14 can be used to address all cases regardless of rank. To generalize Theorem 3.14 just apply the following: (1) if \( \mathcal{N}(M) = C^n \), then \( U_1 = U \) and \( U_2 = 0 \) and (2) if \( \mathcal{N}(M) = 0 \), then \( V_1 = V \) and \( V_2 = 0 \). Using these along with Theorem 3.14, the results of Equations (3.28) and (3.29) can be recovered. In addition, for a square matrix, \( M = USV^* \), it is easily verified that \( M^{-1} = VS^{-1}U^* \) if \( M \) (or equivalently \( S \)) is full rank.

**Example 3.20.** Consider a system of equations \( Mx = b \) where:
\[
M = \begin{bmatrix} 5 & 1 & 0 \\ 10 & 2 & 0 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 6 \\ 12 \end{bmatrix}
\]

By inspection, we see that one possible solution is \( x = [1 \ 1]^T \). Using the singular value decomposition results of Example 3.13, it is concluded that \( \Sigma = 11.4 \),
\[
U_1 = \begin{bmatrix} -0.4472 \\ -0.8944 \end{bmatrix}, \quad U_2 = \begin{bmatrix} -0.8944 \\ 0.4472 \end{bmatrix},
\]
\[
V_1 = \begin{bmatrix} 0.9806 \\ -0.1961 \\ 0 \end{bmatrix} \quad \text{and} \quad V_2 = \begin{bmatrix} -0.1961 \\ 0.9806 \\ 0 \end{bmatrix}
\]
Since $U_2 \neq 0$ and $V_2 \neq 0$, we know that $\Re(M) \neq \mathbb{C}^2$ and $\Im(M) \neq \{0\}$. Thus, Theorem 3.14 should be applied to find the minimum norm least squares solution:

$$
\hat{x} = V_1 \Sigma^{-1} U_1^T b = \begin{bmatrix}
0.9806 \\
-0.1961 \\
0
\end{bmatrix}
\begin{bmatrix}
1/(11.4) \\
-0.4471 \\
-0.8944
\end{bmatrix}
\begin{bmatrix}
6 \\
12
\end{bmatrix}
= \begin{bmatrix}
1.1538 \\
0.2307 \\
0
\end{bmatrix}
$$

This is clearly different than the ‘by inspection’ solution $x = [1 \ 1 \ 1]^T$. However, we do find that $M \hat{x}$ does equal $b$. Thus, we have at least two solutions. The set of all solutions is given by:

$$
\hat{x} = V_1 \Sigma^{-1} U_1^T b + V_2 \alpha_2 = \begin{bmatrix}
1.1538 \\
0.2307 \\
0
\end{bmatrix}
+ \alpha^{(2)} \begin{bmatrix}
-0.1961 \\
0.9806 \\
0
\end{bmatrix}
+ \alpha^{(3)} \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix}
$$

If $\alpha^{(2)}$ is selected to be 0.7845 and $\alpha^{(3)}$ is 1, then we find that $\hat{x} = [1 \ 1 \ 1]^T$, which aligns with our ‘by inspection’ solution. Strictly speaking none of these are least squares solutions, since each is actually a solution (though one could argue that a solution is a least squares with a zero residual). If one were to select a $b$ such that $b \in \Re(M)$, then all would be proper least squares solutions.


3.8 Positive Definite Transformations

If $x$ is a complex scalar (i.e., $x \in \mathbb{C}^1$), then the function $f(x) = px \bar{x} (\in \mathbb{R}^1)$ is positive for any positive value of $p$. In this subsection, such a notion is generalized through the definition of a positive definite transformation. These transformations are actually a subset of the class of self-adjoint transformations, which will have properties of independent interest.

**Definition 3.28.** A linear transformation $L : X \rightarrow X$ is self-adjoint if $L^* = L$.

**Theorem 3.15.** A self-adjoint linear transformation, $L$, has real eigenvalues and its eigenvectors are orthogonal.

**Proof.** Let $(\lambda, \phi)$ be any eigenvalue-eigenvector pair of $L$. If $L = L^*$, then $L \phi = L^* \phi$ and $[L \phi, \phi] = [L^* \phi, \phi]$. Then, the definition of adjoint indicates that $[L \phi, \phi] = [\phi, L \phi]$. Since $(\lambda, \phi)$ is an eigenvalue-eigenvector pair $[\lambda \phi, \phi] = [\phi, \lambda \phi]$. This implies $\lambda [\phi, \phi] = \lambda^*[\phi, \phi]$ and finally $(\lambda - \bar{\lambda})[\phi, \phi]$. Since an eigenvector cannot be the zero vector, it must be concluded that $(\lambda - \bar{\lambda}) = 2j \Im{\lambda}$, where $\Im{\lambda}$ denotes the imaginary part of $\lambda$. Thus, $\lambda$ must be real. To prove that the eigenvectors are orthogonal, let us further assume the eigenvalues are distinct. Let $(\lambda_1, \phi_1)$ and $(\lambda_2, \phi_2)$ be two eigenvalue-eigenvector pairs. Then, $[L \phi_1, \phi_2] = [\phi_1, L^* \phi_2] = [\phi_1, L \phi_2]$, which implies $\lambda_1 [\phi_1, \phi_2] = \lambda_2 [\phi_1, \phi_2]$ and leads to $(\lambda_1 - \lambda_2)[\phi_1, \phi_2] = 0$. Since the eigenvalues are assumed distinct the only possible conclusion is that $[\phi_1, \phi_2] = 0$. Graham and Rawlings [113] provide a proof without the assumption of distinct eigenvalues, but only for the case of matrix transformations. □

Corollary 3.13 is based on the fact that orthogonal eigenvectors are also linearly independent (Theorem 3.4), which guarantees that the matrix \( \Phi = [\phi_1 \phi_2 \ldots \phi_n] \) will have an inverse. If the eigenvectors are normalized (i.e., defined such that \([\phi_i, \phi_j]^* = 1\)), then \( \Phi \) will additionally be unitary (i.e., \( \Phi^{-1} = \Phi^* \)).

Now that we know that all of the eigenvalues of a self-adjoint matrix are real, let us further consider the cases of all eigenvalues having the same sign. These notions are captured by the following definition and subsequent Theorem.

Definition 3.29. A linear transformation \( P : X \to X \) is

(i) **positive definite** \( (P > 0) \), if \( P^* = P \) and \( [Px, x] > 0 \) for all nonzero \( x \in X \).

(ii) **positive semi-definite** \( (P \geq 0) \), if \( P^* = P \) and \( [Px, x] \geq 0 \) for all nonzero \( x \in X \).

(iii) **negative definite** \( (P < 0) \) if \( -P > 0 \).

(iv) **negative semi-definite** \( (P \leq 0) \) if \( -P \geq 0 \).

It should be emphasized that a positive definite matrix does not imply or require that all of the elements of the matrix are positive. The following theorem provides the actual criteria for a transformation to be positive definite.

Theorem 3.16. \( P = P^* \) is positive definite if and only if \( P \) has positive eigenvalues.

**Proof.** We will need to prove two statements: (1) If \( P > 0 \), then \( P \) has positive eigenvalues, and (2) If \( P = P^* \) has positive eigenvalues, then \( P > 0 \). (Statement (1) is known as the “only if direction” and statement (2) is known as the “if direction”.)

Only if direction: Assume \( P \) is positive definite and let \( \lambda \in \mathbb{R} \) and \( \phi \in X \) be an eigenvalue-eigenvector pair. Since \([Px, x] > 0\) for all \( x \in X \) and \( P\phi = \lambda\phi \), one finds that \([P\phi, \phi] = [\lambda\phi, \phi] = \lambda[\phi, \phi] > 0\). Since \( \phi \) cannot be the zero vector, it must be that \( \lambda > 0 \).

If direction: To simplify the proof, let us additionally assume \( P \) is a matrix (i.e., \( X = \mathbb{C}^n \)). Assume all eigenvalues of \( P \) are positive. This combined with the \( P \) being self adjoint indicates that \( P = \Phi\Lambda\Phi \) where \( \Lambda \) is a diagonal matrix containing the positive eigenvalues and the columns of \( \Phi \) contain the orthonormal eigenvectors. Then, for any \( x \in \mathbb{C}^n \), one finds:

\[
[Px, x] = [(\Phi\Lambda\Phi)x, x] = [\Lambda\Phi^* x, \Phi^* x] = [\Lambda\alpha, \alpha] = \sum_{i=1}^n \lambda_i \alpha_i^2 > 0
\]

where \( \alpha = \Phi^* x \). Since \( \lambda_i > 0 \) for all \( i \), the only conclusion is that \([Px, x] > 0\). \( \square \)

In the case of a positive semi-definite transformation, the only distinction is that zero eigenvalues are allowed. That is \( P = P^* \) is positive semi-definite if and only if it has non-negative eigenvalues. In this case, the null space of \( P \) can be non-empty.

Corollary 3.14. If \( P > 0 \), then \( L^* PL > 0 \) if \( \text{null}(L) \) is empty. Otherwise, \( L^* PL \geq 0 \).

Example 3.21. Consider the matrices of Table 3.2. Since all are symmetric, all have real eigenvalues. Cases 1 and 2 are positive definite and Case 3 is positive semi-definite. Case 6
is negative definite and Cases 4 and 5 have no definiteness properties. In the case of \( n = 2 \)

<table>
<thead>
<tr>
<th>Case</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>
| Matrix | \[
\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \] | \[
\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \] | \[
\begin{pmatrix} 2 & 1 \\ 1 & 0.5 \end{pmatrix} \] |
| Eigenvalues | \( \lambda_1 = 1.0 \) | \( \lambda_1 = 1.0 \) | \( \lambda_1 = 0 \) |
|            | \( \lambda_2 = 3.0 \) | \( \lambda_2 = 3.0 \) | \( \lambda_2 = 2.5 \) |

<table>
<thead>
<tr>
<th>Case</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
</table>
| Matrix | \[
\begin{pmatrix} 2 & 1 \\ 1 & 0.25 \end{pmatrix} \] | \[
\begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix} \] | \[
\begin{pmatrix} -2 & 1 \\ 1 & -2 \end{pmatrix} \] |
| Eigenvalues | \( \lambda_1 = -0.204 \) | \( \lambda_1 = -2.24 \) | \( \lambda_1 = -1 \) |
|            | \( \lambda_2 = 2.454 \) | \( \lambda_2 = 2.24 \) | \( \lambda_2 = -3 \) |

and \( P \) having elements \( p_{ij} \), positive definiteness is guaranteed by the following conditions \( p_{11} > 0, p_{22} > 0 \) and \( p_{11}p_{22} - p_{12}p_{21} > 0 \). The conditions for positive semi-definiteness are the same but with the strict inequalities replaced by regular inequalities. This method can be applied to the matrices of Example 3.21, to arrive at the same conclusions. Unfortunately, this simple rule does not generalize well for \( n > 2 \).

In some cases, we will find it convenient to split a positive semi-definite matrix into two roots. The following definition indicates one approach to doing so.

**Definition 3.30.** Consider a positive semi-definite matrix \( P \), then the principal square root of \( P \) is the unique positive semi-definite matrix defined as \( P^{1/2} = \Phi \Lambda^{1/2} \Lambda^{-1} \), where \( \Phi \) and \( \Lambda \) are from the eigenvalue-eigenvector decomposition of \( P \).

### 3.9 - Chapter Summary

In this chapter we have focused on three main topics, all related to linear transformations: \( L : X \rightarrow Y \). The first concerns the solution to a linear equation \( Lx = b \). The Fundamental Theorem of Linear Algebra indicates that the existence and uniqueness of a solution can be determined by the characteristics of the range and null space of the operator \( L \). Then, for the case of matrix transformation we have described methods that can be used to calculate both of these subspaces. In Chapter 4, the Fundamental Theorem will be used to derive conditions for some the most basic properties desired by a dynamic system - controllability and observability.

The second major topic of the chapter concerns matrix polynomial calculations, including the matrix exponential. In this case, it was found that the eigenvector decomposition could be used to greatly simplify these calculations by converting general matrix powers into powers of a diagonal matrix. It was also shown that the Cayley-Hamilton Theorem could be used to achieve similar ends. In Chapter 4, the eigenvector decomposition will be used to derive another fundamental property of a controlled system -
stability - and the Cayley-Hamilton Theorem will serve as an important theoretical tool in the advanced topic of connecting stability to controllability and observability.

The final main topic of the chapter concerns positive definite matrices. While this topic appears to be just a generalization parabolas, we will see that matrix inequalities are among the most powerful tools we have available for the analysis and design of linear control systems. As such, the reader should expect to see this topic appear in some form or another in all subsequent chapters. In particular, these concepts will be core to the economic based design and analysis methods to be discussed in Part III of the book.

As one might expect, much of the material presented in this chapter can be found in other texts, see for example Luenburger [110], Akhiezer and Glazman [111], Debnath and Mikusinski [112] and Graham and Rawlings [113]. Many of the proofs provided in this chapter can also be found in Grossman [114], Balakrishnan [115], Chen [116] and Anton [117].

Exercises

3.1. Consider a set of two differential equations in state-space form: \( \dot{x} = Ax \), with the initial conditions \( x(0) = x_0 \). Prove the following:

“If we are allowed to choose the initial conditions arbitrarily (i.e., \( x_0 \) can be selected as any value in \( \mathbb{R}^2 \)), then the set of functions defined by \( y(t) = Cx(t) \), where \( C \) has 1 row and 2 columns, is a linear vector space.”

(Hint: Assume \( e^{At} \) is a matrix of the form
\[
\begin{bmatrix}
    a_1(t) & a_2(t) \\
    a_3(t) & a_4(t)
\end{bmatrix}
\]
where the \( a_i(t) \) functions are known.)

3.2. For the linear transformations: \( M : \mathbb{X} \rightarrow \mathbb{Y} \) in Table 3.3, determine (i) \( \mathbb{X} \), (ii) \( \mathbb{Y} \), (iii) If the range space is the entire space (iv) If the null space is empty. You should not need to use a computer for these determinations.

3.3. For the linear transformations: \( M : \mathbb{X} \rightarrow \mathbb{Y} \) in Table 3.4, determine (i) the range space, (ii) the null space. Your answers should be either \( \mathbb{X} \), \( \mathbb{Y} \) “empty” or make use of the “span” operator.

3.4. Reconsider the inner product defined in Example 3.3:

\[
[x, y]_P = \bar{y}^T P x
\]

where \( P = \begin{bmatrix} 1 & 0 \\ -1 & 4 \end{bmatrix} \)

For each of the following vectors find a vector that is orthogonal under this inner product:

\[
x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad x_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad x_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

For each case, sketch the pair of orthogonal vectors and show that they do not conform to our usual understanding of being at a right angle, which would be the case if the inner product was defined as in Equation 3.3 (i.e., with \( P = I \)).
### Table 3.3. Table for Exercise 3.2.

<table>
<thead>
<tr>
<th></th>
<th>$X = ?$</th>
<th>$Y = ?$</th>
<th>$\mathbb{R}(M) = Y$</th>
<th>$\mathbb{N}(M)^{\perp} = {0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>1 0 0 1 2 2 0 2</td>
<td>Yes or No</td>
<td>Yes or No</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>1 1 1 1 1 1 1 2</td>
<td>Yes or No</td>
<td>Yes or No</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>1 2 3 4</td>
<td>Yes or No</td>
<td>Yes or No</td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>1 0 1 1 0 1 1 1</td>
<td>Yes or No</td>
<td>Yes or No</td>
<td></td>
</tr>
<tr>
<td>(e)</td>
<td>0 1 2 2 1 0 0 2</td>
<td>Yes or No</td>
<td>Yes or No</td>
<td></td>
</tr>
<tr>
<td>(f)</td>
<td>1 0 1 1 1 0 0 1</td>
<td>Yes or No</td>
<td>Yes or No</td>
<td></td>
</tr>
<tr>
<td>(g)</td>
<td>1 2 1 1 1 2 1 2</td>
<td>Yes or No</td>
<td>Yes or No</td>
<td></td>
</tr>
</tbody>
</table>

### 3.5. Consider a linear operator $M : X \to Y$ defined by the matrix

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

(i) Determine the spaces $X$ and $Y$

(ii) Determine the range space of $M : \mathbb{R}(M) = \{ y \in Y \text{ such that } y = Mx \text{ where } x \in X \}$

(iii) Determine the null space of $M^* : \mathbb{N}(M^{*}) = \{ y \in Y \text{ such that } M^*y = 0 \}$

(iv) Determine a vector $b$ such that a solution to $Mx = b$ exist. Verify this using the null space of $M^*$. (Your verification must use the $\mathbb{N}(M^{*})$)

(v) Determine a vector $b$ such that a solution to $Mx = b$ does NOT exist. Verify this using the null space of $M^*$. (Your verification must use the $\mathbb{N}(M^{*})$)
3.6. Find the four fundamental subspaces of the following matrices. (Do this by hand. Then verify your results with the MATLAB commands “null” and “orth”.)

(i) \( M = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \)
(ii) \( M = \begin{bmatrix} 2 & -1 & -1 \\ 1 & -2 & 3 \end{bmatrix} \)
(iii) \( M = \begin{bmatrix} 5 & 1 \\ 3 & 4 \\ 1 & 2 \end{bmatrix} \)
(iv) \( M = \begin{bmatrix} 5 & 0 & 1 \\ 0 & 5 & -7 \\ 1 & -7 & 10 \end{bmatrix} \)

3.7. Prove part (ii) of Theorem 3.2

3.8. (i) Using the Gram-Schmidt procedure, orthogonalize the following vectors

\( \phi_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \phi_2 = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \quad \phi_3 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \)

(ii) Using the Gram-Schmidt procedure, orthogonalize the following vectors

\( \phi_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \phi_2 = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \quad \phi_3 = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} \)

(iii) Using the Gram-Schmidt procedure, orthogonalize the following vectors

\( \phi_1 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} \quad \phi_2 = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \quad \phi_3 = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} \)

(iv) How is the null space of \( M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 2 \end{bmatrix} \) related to the solution of part (i)?

3.9. Consider the following set of vectors:

\( \phi_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \phi_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \phi_3 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \)
(i) Use Gram-Schmidt to determine an orthonormal basis for $span\{\phi_1, \phi_2, \phi_3\}$.  
(ii) What is the dimension of $span\{\phi_1, \phi_2, \phi_3\}$?

3.10. Consider a set of algebraic equations $Mx = b$ where

$$M = \begin{bmatrix}
0 & 1 & 2 & 0 \\
1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}$$

The Singular Value Decomposition of $M$ is $M = USV^*$ where

$$U = \begin{bmatrix}
0 & 1 & 0 & 0 \\
-0.365 & 0 & -0.447 & 0.816 \\
-0.913 & 0 & 0 & -0.408 \\
-0.183 & 0 & 0.894 & 0.408 \\
\end{bmatrix}$$

$$S = \begin{bmatrix}
2.45 & 0 & 0 & 0 \\
0 & 2.24 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}$$

$$V = \begin{bmatrix}
-0.894 & 0 & -0.447 & 0 \\
0 & 0.447 & 0 & 0.894 \\
0 & 0.894 & 0 & -0.447 \\
-0.447 & 0 & 0.894 & 0 \\
\end{bmatrix}$$

For which of the following $b$ vectors will a solution to $Mx = b$ exist?

$$(a) \quad b = \begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
\end{bmatrix}$$

$$(b) \quad b = \begin{bmatrix}
1 \\
2 \\
1 \\
0 \\
\end{bmatrix}$$

$$(c) \quad b = \begin{bmatrix}
0 \\
1 \\
0 \\
2 \\
\end{bmatrix}$$

$$(d) \quad b = \begin{bmatrix}
1 \\
0.5 \\
0 \\
-1 \\
\end{bmatrix}$$

3.11. Show that the $U$ and $V$ matrices of Exercise 3.10 are unitary.


3.13. Consider the following matrix and three vectors.

$$M = \begin{bmatrix}
2 & -2 & 2 \\
-1 & 1 & 1 \\
1 & -1 & 3 \\
\end{bmatrix} \quad u_1 = \begin{bmatrix}
0 \\
1 \\
1 \\
\end{bmatrix} \quad u_2 = \begin{bmatrix}
1 \\
1 \\
0 \\
\end{bmatrix} \quad u_3 = \begin{bmatrix}
1 \\
0 \\
-1 \\
7 \\
\end{bmatrix}$$

(i) Are any of these vectors in the Null Space of $M$?

(ii) Are any of these vectors eigenvectors of $M$?

(iii) Are any of these vectors in the Range Space of $M^T$?

3.14. Use Definition 2.3 show the following:

(i) If $D = \begin{bmatrix}
\lambda_1 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \lambda_n \\
\end{bmatrix}$, then $e^{Dt} = \begin{bmatrix}
e^{\lambda_1 t} & 0 & 0 & 0 \\
0 & e^{\lambda_2 t} & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & e^{\lambda_n t} \\
\end{bmatrix}$

(ii) If $A = \Phi D \Phi^{-1}$, then $e^{At} = \Phi e^{Dt} \Phi^{-1}$
3.15. Consider a set of algebraic equations \( Mx = b \) where

\[
M = \begin{bmatrix}
-1 & 1 & 0 & -1 \\
1 & -1 & 1 & 0 \\
0 & 1 & -1 & 1 \\
-1 & 0 & 1 & 0
\end{bmatrix} \quad b = \begin{bmatrix}
1 \\
0 \\
1 \\
0
\end{bmatrix}
\]

An eigenvector decomposition of \( A \) is

\[
A \Phi = \Phi A
\]

where

\[
\Phi = \begin{bmatrix}
0.5 & -0.577 & -0.5 & 0.408 \\
-0.707 & 0 & -0.707 & 0 \\
0.5 & 0.577 & -0.5 & -0.408 \\
0 & -0.577 & 0 & -0.817
\end{bmatrix}
\]

\[
\Lambda = \begin{bmatrix}
-2.414 & 0 & 0 & 0 \\
0 & -2 & 0 & 0 \\
0 & 0 & 0.414 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(i) Show that the matrix \( \Phi \) is unitary. Is there a way you could have known this without doing any calculations?

(ii) Determine if a solution to \( Mx = b \) will exist.

(iii) If a solution exists, calculate such a solution without the aid of a computer.

(iv) Is this the only solution?

3.16. Prove Corollaries 3.10 and 3.11.

3.17. Consider a set of algebraic equations \( Mx = b \) where

\[
M = \begin{bmatrix}
1 & -0.778 & -1.222 \\
0 & -0.889 & -0.889 \\
-1 & -0.111 & 2.111
\end{bmatrix}
\]

The singular value decomposition of \( M \) is \( M = USV^* \) where

\[
U = \begin{bmatrix}
-0.540 & 0.613 & 0.577 \\
-0.260 & -0.774 & 0.577 \\
0.800 & 0.161 & 0.577
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
2.91 & 0 & 0 \\
0 & 1.30 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
-0.461 & 0.348 & -0.816 \\
0.034 & -0.912 & -0.408 \\
0.887 & 0.216 & -0.408
\end{bmatrix}
\]

(i) Based on this decomposition, show that if an exact solution to \( Ax = b \) exists, then it will not be unique.

(ii) Determine any value for the vector \( b \) such that a solution to \( Ax = b \) will exist?

(iii) What is the set of all \( b \) such that a solution to \( Ax = b \) will exist?

(iv) For the \( b \) of part (ii), determine a solution to \( Ax = b \).
Exercises 109

(v) Using the b of part (ii), determine all solutions to $Ax = b$. Rather than perform the calculations, based on the above numbers, please use the notation below to develop a formula for the solution.

$$U = [u_1 \ u_2 \ u_3] \quad S = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad V = [v_1 \ v_2 \ v_3]$$

where $u_i$ and $v_i$ are the columns of $U$ and $V$.

3.18. Consider a set of algebraic equations $Mx = b$

where $M = \begin{bmatrix} 1 & 1 \\ 2 & 4 \\ -1 & 1 \end{bmatrix}$ and $b = \begin{bmatrix} 0 \\ -2 \\ -2 \end{bmatrix}$

(i) Determine if a solution to this problem exists.

(ii) If a solution exists, determine that solution. If a solution does not exist, determine the least squares solution.

(iii) Repeat parts (i) and (ii) using $b = [2 \ 5 \ 0]^T$.

(iv) Determine an orthonormal basis set for $\mathfrak{R}(M)$.

3.19. Consider a linear operator $M : X \rightarrow Y$ defined by the matrix

$$M = \begin{bmatrix} 0.48 & 0.64 \\ -0.40 & 0.30 \\ 0.36 & 0.48 \end{bmatrix}$$

The singular value decomposition of $M$ is $M = USV^*$ where

$$U = \begin{bmatrix} 0.8 & 0 & 0.6 \\ 0 & 1 & 0 \\ 0.6 & 0 & -0.8 \end{bmatrix} \quad S = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \\ 0 & 0 \end{bmatrix} \quad V = \begin{bmatrix} 0.6 & -0.8 \\ 0.8 & 0.6 \end{bmatrix}$$

(i) Determine the pseudo-inverse operator associated with the least squares solution.

(ii) If $b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}^T$, determine the least squares solution to $Mx = b$.

3.20. Determine the definiteness (either positive definite, positive semi-definite, negative definite, negative semi-definite or none of the above) for the following matrices: (Feel free to use any MATLAB commands)

(i) $P = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix}$

(ii) $P = \begin{bmatrix} 4 & -2 \\ -2 & -1 \end{bmatrix}$

(iii) $P = \begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix}$

(iv) $P = \begin{bmatrix} 5 & 1 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$

(v) $P = \begin{bmatrix} 5 & 0 & 1 \\ 0 & 5 & -7 \\ 1 & -7 & 10 \end{bmatrix}$

(vi) $P = \begin{bmatrix} 5 & 0 & 1 \\ 0 & 5 & -7 \\ 1 & -7 & 12 \end{bmatrix}$

3.21. In Corollary 3.6 it was stated that the adjoint of a matrix $M$ is found to be $M^* = M^T$, if the inner product is defined by Equation 3.3.

(i) Prove Corollary 3.6
(ii) If the inner product is defined as \( [x, y]_p = \bar{y}^T P x \), where \( P \) is a positive definite matrix, show that the adjoint of a matrix \( M \) is \( M^* = P^{-1} M^T P \)

(iii) Verify that Theorem 3.2 holds true if \( [x, y]_p = \bar{y}^T P x \) where \( P = \begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix} \) and \( M = \begin{bmatrix} 5 & 1 & 0 \\ 10 & 2 & 0 \end{bmatrix} \)

3.22. Consider the manufacturing process given in section 2.6.4. The process model is 
\[ s_{k+1} = A_d s_k + B_d m_k + G_d p_k \]
where the system matrices are
\[
A_d = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix} \quad B_d = \begin{bmatrix}
0 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
\end{bmatrix} \\
G_d = \begin{bmatrix}
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
-1 & 0 \\
0 & -1 \\
\end{bmatrix}
\]

The steady-state value of the disturbance, \( p_{ssop} \), is given to be \( \begin{bmatrix} 200 & 100 \end{bmatrix} \). Determine all the steady-state condition \( (s_{ssop}, m_{ssop}) \). (Hint: feel free to use the MATLAB function ‘svd’ to find the null space that is needed.)
Part II

Modern Control Theory
In Chapter 2 it was shown that the method of linearization can be used to approximate the dynamics of a nonlinear state-space process. The resulting linear model (in combination with the tools of linear algebra) can then be used to characterize the abilities of the process. Specifically, one would like to know answers to the following questions: How well will the system respond to feedback? Do the process measurements provide sufficient information? This chapter will address these questions by introducing the concepts of stability, controllability, observability and will focus on presenting tests to determine if a given system possesses these properties. Most importantly, the chapter will address the question of the existence of a feedback element capable of yielding a stable closed-loop system by introducing the concepts of stabilizability and detectability. If a given process does not possess these most basic criteria, then one will know that it is futile to attempt to design a controller and the only remaining option is to redesign of the open-loop process such that such conditions are satisfied.

4.1 Stability of a Linear Dynamic System

In the design of a control system, the first property of interest is to ensure the closed-loop system behaves reasonably well. Specifically, one would like to avoid situations in which one or more of the state variables becomes an unbounded function of time. As such, the notion of stability is concerned with the fate of the state trajectory (i.e., where the state will tend toward as time goes to infinite). This section will focus on autonomous systems (i.e., those in which all inputs are held constant at their nominal values). Subsequent sections will illustrate how these notions apply to system subject to feedback.

4.1.1 Stability in the Discrete-time Framework

In the discrete-time framework, the linear state-space model of interest is

\[ x_{k+1} = A_d x_k \]  

(4.1)

The solution to this recursion is easily found as \( x_k = (A_d)^k x_0 \). As discussed above the property desired from the sequence \( x_k \) is that none of its elements becomes unbounded as \( k \) tends to infinity. However, our definition of stability will be the stronger condition of requiring all elements of the state to approach zero for all initial conditions.
Definition 4.1. A discrete-time linear system, \( x_{k+1} = A_d x_k \), is stable if \( \lim_{k \to \infty} \{ x_k \} = 0 \) for all initial conditions \( x_0 \in \mathbb{R}^n \).

To illustrate the characteristics required to achieve a stable system, let us assume \( A_d \) is diagonalizable. Then, as indicated in Section 3.8, \( A_d = \Phi \Lambda \Phi^{-1} \) where \( \Phi \) has the eigenvectors of as its columns (\( \Phi = [\phi_1, \phi_2, \ldots, \phi_n] \)) and \( \Lambda \) is a diagonal matrix with the eigenvalues of \( A_d \) as its element (\( \Lambda = \text{diag} \{ \lambda_1, \lambda_2, \ldots, \lambda_n \} \)). Using this eigenvector decomposition it is easily concluded that:

\[
x_k = \Phi (\Lambda)^k \Phi^{-1} x_0 = \sum_{i=1}^{n} \lambda_i^k \alpha_i \phi_i
\]

(4.2)

As indicated in Section 3.7, the \( \alpha_i \)'s are elements of the projection of the initial condition, \( x_0 \), into the eigenvector directions, or simply \( \alpha = \Phi^{-1} x_0 \). This is also observed by setting \( k = 0 \).

The importance of Equation (4.2) is that it elucidates the relation between the eigenvalues and the fate of the state vector. The first point to note is that the fate of \( \lambda_i^k \) depends on the magnitude of \( \lambda_i \), where magnitude of a complex number is defined as \( |\lambda_i| = \sqrt{\lambda_i \overline{\lambda}_i} \). The salient points are captured by the following identity:

\[
\lim_{k \to \infty} \{ |\lambda_i^k| \} = \begin{cases} 0 & \text{if } |\lambda_i| < 1 \\ 1 & \text{if } |\lambda_i| = 1 \\ \infty & \text{if } |\lambda_i| > 1 \end{cases}
\]

Thus, if \( |\lambda_i| < 1 \) for all \( i \), then it is guaranteed that \( x_k \) will tend to zero, for all values of \( \alpha = \Phi^{-1} x_0 \). On the other hand, if \( |\lambda_i| \geq 1 \) for one of more \( i \), then there will exist an initial condition such that \( x_k \) will not tend to zero. Specifically, if \( |\lambda_i| \geq 1 \), then \( x_0 = \Phi \alpha \) with \( \alpha_i \not= 0 \) will cause the limit of \( x_k \) to be non-zero. These observations are made precise by the following fact.

Theorem 4.1. The system \( x_{k+1} = A_d x_k \) is stable, if and only if all eigenvalues of the matrix \( A_d \) have a magnitude strictly less than one (i.e., \( |\lambda_i| < 1 \) for all \( i \)).

Note that Theorem 4.1 does not require to be diagonalizable. To illustrate, consider the following non-diagonalizable matrix, with repeated eigenvalues at 0.9.

\[
A_d = \begin{bmatrix} 1.9 & -1 \\ 1 & -0.1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0.9 & 1 \\ 0 & 0.9 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \Phi J \Phi^{-1}
\]

Then, once again \((A_d)^k = \Phi J^k \Phi^{-1}\) and it is easily concluded that

\[
J^k = \begin{bmatrix} (0.9)^k & k(0.9)^{k-1} \\ 0 & (0.9)^k \end{bmatrix}
\]

and

\[
x_k = [\alpha_1 (0.9)^k + \alpha_2 k (0.9)^{k-1}] \phi_1 + \alpha_2 (0.9)^k \phi_2
\]

Thus, it is guaranteed that \( \lim_{k \to \infty} \{ x_k \} = 0 \) for all \( \alpha = \Phi^{-1} x_0 \). The unique feature of this system is that the term \( k (0.9)^{k-1} \) will initially increase with \( k \), and then eventually decrease for large values of \( k \). (For this example the transition occurs at \( k = 10 \).)

It should be emphasized that Theorem 4.1 cannot be used to conclude that \( \lim_{k \to \infty} \{ x_k \} = 0 \) implies that all eigenvalues of \( A_d \) are within the unit circle of the complex plane.
4.1. Stability of a Linear Dynamic System

This will be true only if \( \lim_{k \to \infty} \{x_k\} = 0 \) occurs for all initial conditions \( x_0 \). In fact, if \( A_d \) is unstable (i.e., there exists an eigenvalues of \( A_d \) outside the strict unit circle), then it is entirely possible to find an initial condition such that \( \lim_{k \to \infty} \{x_k\} = 0 \). The following definition indicates which initial conditions will generate such a result.

**Definition 4.2.** Consider a matrix \( A_d \) and organize the generalized eigenvalue-eigenvectors pairs \( (\lambda_i, \phi_i) \), \( i = 1 \ldots n \), to be in increasing order with respect to \( |\lambda_i| \), so that \( \lambda_1 \) has the smallest magnitude and \( \lambda_n \) has the largest. Define \( s \) such that \( \lambda_s \) is the largest eigenvalue with a magnitude strictly less than 1. Then, the **stable subspace** is defined as \( M_s(A_d) = \text{span} \{ \phi_1 \ldots \phi_s \} \) and the **unstable subspace** is defined as \( M_{Us}(A_d) = \text{span} \{ \phi_{s+1} \ldots \phi_n \} \).

The implication of Definition 4.2 is that if the initial condition, \( x_0 \), is in the stable subspace then the \( \alpha \)'s associated with the unstable eigenvalues will be zero and will be guaranteed. On the other hand, if \( x_0 \) is not in \( M_s(A_d) \), then some of the \( \alpha \)'s associated with unstable eigenvalues will not be zero and \( \lim_{k \to \infty} \{x_k\} \neq 0 \).

Another way to interpret the stable and unstable subspaces is to project the state sequence in the eigenvector directions. Specifically, define \( z_k = \Phi^{-1}x_k \), where the columns of \( \Phi \) are the eigenvectors of \( A_d \). Then, since \( \Phi^{-1}A_d\Phi = \Lambda \), \( x_{k+1} = A_d x_k \) becomes \( z_{k+1} = \Lambda z_k \). Then, if an element of the vector \( z_k \) is denoted as \( z_k^{(i)} \), then one finds the scalar recursions: \( z_{k+1}^{(i)} = \lambda_i z_k^{(i)} \), or equivalently \( z_k^{(i)} = (\lambda_i)^k z_0^{(i)} \). Thus, if \( |\lambda_i| \geq 1 \) and \( z_0^{(i)} \neq 0 \) then \( \lim_{k \to \infty} \{z_k^{(i)}\} \neq 0 \). Then, recalling the discussion of Section 3.1.9, the only way to arrive at \( z_0^{(i)} = 0 \) (from \( z_0 = \Phi^{-1}x_0 \)) is for \( x_0 \) to be a linear combination of the stable eigenvalues, or equivalently \( x_0 \) is in \( M_s(A_d) \).

**Example 4.1.** Consider the following matrix

\[
A_d = \begin{bmatrix}
1 & 0.2 \\
-0.1 & 0.7
\end{bmatrix}
\]

Using the MATLAB function \('\text{eig}(A)'\), one finds that the eigenvalues are 0.8 and 0.9, indicating that the open-loop process is stable.

**Example 4.2.** Consider the following matrix

\[
A_d = \begin{bmatrix}
1.4 & 0.6 \\
-0.3 & 0.5
\end{bmatrix}
\]

Using the MATLAB function \('[\Phi, \text{Lam}] = \text{eig}(A)'\), one finds the following eigenvalue and eigenvector matrices (as defined in Section 3.6)

\[
\Lambda = \begin{bmatrix}
1.1 & 0 \\
0 & 0.8
\end{bmatrix} \quad \Phi = \begin{bmatrix}
0.8944 & -0.7071 \\
-0.4472 & 0.7071
\end{bmatrix}
\]

The system is clearly unstable (since one eigenvalue is outside the unit circle). In this case, the **stable subspace** is \( M_s(A) = \text{span} \{ [ -0.7071 \quad 0.7071 ]^T \} \) and the **unstable subspace** is \( M_{Us}(A) = \text{span} \{ [0.8944 \quad -0.4472]^T \} \).
4.1.2 • Stability in the Continuous-time Framework

In the continuous-time framework, the linear state-space model of interest is
\[ \dot{x} = Ax \] (4.3)

In Chapter 2 it was asserted that the solution to Equation (4.3) is
\[ x(t) = e^{At}x(0). \]

**Definition 4.3.** A continuous-time linear system, \( \dot{x} = Ax \), is **stable** if
\[ \lim_{t \to \infty} \{ x(t) \} = 0 \]
for all initial conditions \( x(0) \in \mathbb{R}^n \).

If the matrix \( A \) is diagonalizable, (i.e., \( A = \Phi \Lambda \Phi^{-1} \)), then \( x(t) \) is easily found to be
\[ x(t) = \Phi e^{\Lambda t} \Phi^{-1}x(0) = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t} \phi_i \] (4.4)
where \( (\lambda_i, \phi_i) \) are the eigenvalue-eigenvector pairs of \( A \), and the \( \alpha_i \)'s are elements of the projection \( \alpha = \Phi^{-1}x(0) \). As compared to the discrete-time framework, Equation (4.4) gives a slightly different relation between the stability of the system and the eigenvalues of \( A \). Specifically, if the real part of an eigenvalue, \( \lambda_i \), is strictly negative (i.e., \( \text{Re}\{\lambda_i\} < 0 \)), then \( e^{\lambda_i t} \) will converge to zero for large \( t \). On the other hand, if \( \text{Re}\{\lambda_i\} \geq 0 \), then \( e^{\lambda_i t} \) will not converge to zero.

**Theorem 4.2.** The system \( \dot{x} = Ax \) is stable if and only if all eigenvalues of the matrix \( A \) have negative real parts.

Again, the condition for stability does not require \( A \) to be diagonalizable. If \( A \) is not diagonalizable and the Jordan form is employed, then solution will include the matrix \( e^{Jt} \). Then, using the Cayley-Hamilton Theorem, it can be shown that this matrix will include the expected \( e^{\lambda_i t} \) terms, but will also include \( t^{q-1}e^{\lambda_i t} \) terms where \( q \) is the number of repeated eigenvalues. Such a term should be familiar to those well versed in the subject of Laplace transforms, as the inverse transform of \( (q-1)!/(s-\lambda_i)^q \).

**Definition 4.4.** Consider a matrix \( A \) and organize the generalized eigenvalue-eigenvectors pairs \( (\lambda_i, \phi_i), i = 1 \ldots n, \) to be in increasing order with respect to \( \text{Re}\{\lambda_i\} \), so that \( \lambda_1 \) has the smallest real part and \( \lambda_n \) has the largest. Define \( s \) such that \( \lambda_s \) is the largest eigenvalue with a real part strictly less than \( 0 \). Then, the **stable subspace** is defined as \( M_{S}(A) = \text{span}\{\phi_1 \ldots \phi_s\} \) and the **unstable subspace** is defined as \( M_{US}(A) = \text{span}\{\phi_{s+1} \ldots \phi_n\} \).

**Example 4.3.** Consider the following matrix
\[ A = \begin{bmatrix} 0 & 2 \\ -1 & -3 \end{bmatrix} \]
Using the MATLAB function ‘eig(A)’, one finds that the eigenvalues are -1 and -2, indicating that the open-loop process is stable.

**Example 4.4.** Consider the following matrix
\[ A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix} \]

Using the MATLAB function ‘[Phi, Lam] = eig(A)’, one finds the following eigenvalue and eigenvector matrices (as defined in Section 3.6)

\[
\Lambda = \begin{bmatrix}
1 & 0 \\
0 & -2
\end{bmatrix}, \quad \Phi = \begin{bmatrix}
0.8944 & -0.7071 \\
-0.4472 & 0.7071
\end{bmatrix}
\]

The system is clearly unstable (since one eigenvalue is outside the unit circle). In this case, the stable subspace is \(M_S(A) = \text{span}\{\begin{bmatrix}-0.7071 \\ 0.7071\end{bmatrix}\}\) and the unstable subspace is \(M_{US}(A) = \text{span}\{\begin{bmatrix}0.8944 \\ -0.4472\end{bmatrix}\}\). □

### 4.1.3 Stability in the Sense of Lyapunov

The Lyapunov Stability Theorem is among the most celebrated result in all of control theory. Specifically, it provides a stability test alternative to the simple eigenvalue condition. While this approach appears to be more complicated, it is central to the controller synthesis methods to be described in later chapters.

**Theorem 4.3.** The following three statements are equivalent (in the sense that if any one holds then the other two also hold).

1. \( \dot{x} = Ax \) is stable
2. There exists \( P > 0 \) such that \( A^* P + PA < 0 \)
3. There exists \( P > 0 \) such that \( AP + P^* A < 0 \)

Before giving the proof of Theorem 4.3, the discrete-time version is stated to illustrate the similarities of the two. The proof of Theorem 4.4 is left as Exercise 4.19.

**Theorem 4.4.** The following three statements are equivalent (in the sense that if any one holds then the other two also hold).

1. \( x_{k+1} = A_d x_k \) is stable
2. There exists \( P > 0 \) such that \( A_d^* P A_d - P < 0 \)
3. There exists \( P > 0 \) such that \( A_d^* P A_d - P < 0 \)

**Proof of Theorem 4.3:** Let us begin by noting that the real part of the eigenvalues of \( A \) and \( A^* \) are the same. Using this fact it is concluded that if (1) is equivalent to (2), then one can replace \( A \) in (1) with \( A^* \) and find it equivalent to (3). Thus, the remaining challenge is to show that (1) implies (2) and that (2) implies (1). We will begin with the later.

Assume the existence of \( P > 0 \) such that \( A^* P + PA < 0 \) and let \( \lambda \in \mathbb{C} \) and \( \phi \in \mathbb{C}^n \) be an eigenvalue-eigenvector pair for \( A \). This indicates that

\[
0 > [(A^* P + PA)\phi, \phi] = [P \phi, A\phi] + [PA\phi, \phi] = [P \phi, \lambda \phi] + [P \lambda \phi, \phi] = (\lambda + \lambda)\phi, \phi
\]

Thus, it is found that \( 2\text{Re}\{\lambda\} |P\phi, \phi| < 0 \). Since \( P > 0 \) and \( \phi \) cannot be the zero vector, it is concluded that \( \text{Re}\{\lambda\} < 0 \), and it is confirmed that (2) implies (1).

For the other case, (1) implies (2), a \( P > 0 \) will need to be constructed and shown to satisfy \( A^* P + PA < 0 \), using only the assumption of stability. Consider the matrix

\[
P = \int_0^\infty e^{A^* t} Q e^{A t} dt
\]
where $Q$ is any positive definite matrix of appropriate dimension. Using Corollary 2.1, the following equalities are easily obtained:

\[ A^*P = \int_0^\infty A^* e^{A^t} Q e^{A^t} d t = \int_0^\infty \frac{d}{dt} \left( e^{A^t} \right) Q e^{A^t} d t \]

\[ PA = \int_0^\infty e^{A^t} Q e^{A^t} d t = \int_0^\infty e^{A^t} Q \frac{d}{dt} \left( e^{A^t} \right) d t \]

Then, application of the product rule for derivatives (in reverse) as well as the fundamental theorem of calculus yields:

\[ A^*P + PA = \int_0^\infty \frac{d}{dt} \left( e^{A^t} \right) Q e^{A^t} + \frac{d}{dt} \left( e^{A^t} \right) Q \frac{d}{dt} \left( e^{A^t} \right) d t \]

\[ = \left( e^{A^t} Q e^{A^t} \right) \Bigg|_0^\infty = \lim_{t \to \infty} \left\{ e^{A^t} Q e^{A^t} \right\} - Q \]

The stability assumption tells us that \( \lim_{t \to \infty} e^{At} = 0 \), and indicates the desired result:

\[ A^*P + PA = -Q \]

The final issue is to show that \( P > 0 \). Extending Definition 3.4(iii) to integration gives:

\[ [Px, x] = \int_0^\infty [Qe^{A^t}x, e^{A^t}x] d t \]

Then noting that the null space of \( e^{At} \) is empty for all \( t \) (since \((e^{At})^{-1} = e^{-At}\) exists for all \( t \)), indicates that \( [Px, x] > 0 \) for all non-zero \( x \in \mathbb{C}^n \).

**Example 4.5.** Consider the following matrix

\[
A = \begin{bmatrix}
0 & 2 \\
-1 & -3
\end{bmatrix}
\]

Using the MATLAB function ‘lyap(A’,eye(2))’, one finds that the solution to the Lyapunov equation \( A^* P + PA + I = 0 \) is

\[
P = \begin{bmatrix}
1 & 0.5 \\
0.5 & 0.5
\end{bmatrix}
\]

Since this \( P \) is also positive definite (with eigenvalues 0.191 and 1.309), the result of Example 4.3 is confirmed. Next consider the matrix of Example 4.4.

\[
A = \begin{bmatrix}
4 & 6 \\
-3 & -5
\end{bmatrix}
\]

Using the MATLAB function ‘lyap(A’,eye(2))’ again, one finds that the solution to the Lyapunov equation \( A^* P + PA + I = 0 \) is

\[
P = \begin{bmatrix}
-8 & -10.5 \\
-10.5 & -12.5
\end{bmatrix}
\]
However, this $P$ matrix is not positive definite (with eigenvalues -20.99 and 0.488). Thus, a strict conclusion about stability cannot be made. However, if one works from the other direction (the knowledge that $A$ is not stable based on the eigenvalues), then Theorem 3.3 tells us that there does not exist a positive definite $P$ satisfying the Lyapunov equation, so we should stop looking.

### 4.2 Linear Feedback and State Observers

Consider the following linear state-space process

\[
\dot{x} = Ax + Bu
\]
\[
y = Cx
\]

where $u$ is the manipulated variable and $y$ is a physical measurement. Let us initially assume $C = I$. That is the controller has full access to all the state variables at all times. In this case, it is reasonable to assume the controller is of the following form: $u(t) = -Lx(t)$. Substitution of this linear feedback policy into Equation (4.11) yields:

\[
\dot{x} = Ax - BLx = (A - BL)x
\]

Based on the previous section, the selection of $L$ should be such that the closed-loop system matrix, $(A - BL)$, has eigenvalues with negative real parts, or equivalently the closed-loop system is stable.

**Example 4.6.** Consider the matrix pair

\[
A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix}, \quad B = \begin{bmatrix} 2 \\ -1 \end{bmatrix}
\]

If the feedback gain is selected to be $L = [1 \ 0]$, then the closed-loop system becomes

\[
\dot{x} = (A - BL)x = \begin{bmatrix} 2 & 6 \\ -2 & -5 \end{bmatrix}x
\]

Using the MATLAB function `eig(A-B*L)`, one finds that the eigenvalues are -1 and -2, indicating that the closed-loop system is stable.

Before beginning the design of a controller (i.e., the selection of $L$), one would like a answer to the following question: “Does there exist a linear feedback $L$ such that $(A - BL)$ is stable?” Section 4.3 will develop a set of conditions aimed at answering this question. As one may expect, these conditions will be based solely upon the characteristics of the matrices $A$ and $B$. The following definition gives a name to this question.

**Definition 4.5.** A matrix pair $(A, B)$ is **stabilizable** if there exists a matrix, $L$, such that the closed-loop system matrix, $(A - BL)$, is stable.

Let us now return to the more general case, where $C$ is not equal to $I$. In this case, one would like to use a policy similar to $u = -Lx$, but a substitute for $x$ must be found since $x$ cannot be measured. One option is to use measurement feedback ($u = -Ly = -LCx$). Unfortunately, such an approach greatly narrows the design freedom with respect to $L$ and ultimately will limit the performance capabilities of the closed-loop system.
alternative is to construct a state observer to generate an estimate of the state vector. This
estimate will be denoted as \( \hat{x}(t) \) and will eventually be used to construct the following
control policy: \( u = -L\hat{x} \). However, before analyzing this new feedback policy, the
characteristic of the observer must be investigated.

A state observer is defined by the following linear state space process:

\[
\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x}) \quad (4.14)
\]

where \( y \) is from Equation (4.12). The basic idea being that data collected from the mea-


surement \( y(t) \) will be used to drive \( \hat{x} \) to the current value of \( x(t) \) regardless of the initial

condition of each. To see this, define the following error signal:

\[
e(t) = x(t) - \hat{x}(t)
\]

Then, by combining (4.14) with (4.11)-(4.12), one arrives at the following model for the time
evolution of the error signal:

\[
\dot{e} = (A - KC)e \quad (4.15)
\]

Thus, if \( K \) is selected such that \( (A - KC) \) is stable, then the error signal will go to zero
regardless of the initial condition \( e(0) \). This issue of stability of the error system is at the
heart of observer design.

Example 4.7. Consider the matrix pair

\[
A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix}, \quad C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\]

If the observer gain is selected to be \( K = \begin{bmatrix} 2 & 2 \end{bmatrix}^T \), then the error system becomes

\[
\dot{e} = (A - KC)e = \begin{bmatrix} 2 & 4 \\ -5 & -7 \end{bmatrix} e
\]

Using the Matlab function ‘eig(A-K*C)’, one finds that the eigenvalues are -2 and -3, indi-
cating that the error system is stable.

Similar to the state feedback case, one would like to an answer to the following ques-
tion: “Does there exist an observer gain \( K \) such that \( (A - KC) \) is stable?” Section 4.4 will
develop a set of conditions for the following definition.

Definition 4.6. A matrix pair \( (A, C) \) is detectable if there exists a matrix, \( K \), such that the
closed-loop system matrix, \( (A - KC) \), is stable.

Now return to the question of using the state estimate in the feedback policy: \( u = -L\hat{x} \). For this analysis one will need to construct a compound system to capture the
dynamics of both the original process and the observer. That is, if \( u = -L\hat{x} \), then \( \dot{x} = Ax - BL\hat{x} \). Thus, one will need to know the dynamics of \( \hat{x} \), from \( \dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x}) \) which is equal to \( \dot{\hat{x}} = A\hat{x} - BL\hat{x} + KCx - KC\hat{x} \). Thus, we arrive at the compound system

\[
\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A & -BL \\ KC & (A - KC - BL) \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} \quad (4.16)
\]

Then, the question is to find matrices \( L \) and \( K \) such that this compound system is stable.
Using the form of Equation (4.16), this will be a challenging task indeed. However, the
task is made much simpler by considering one of two possible variable transformations.
4.3. Controllability and Stabilizability

The first is arrived at by defining the feedback policy $u = -L\hat{x}$ as $u = -L(x - e)$, since $e = x - \hat{x}$. Then, we quickly arrive at the following compound system.

\[
\begin{bmatrix}
\dot{x} \\
\dot{\hat{x}}
\end{bmatrix} =
\begin{bmatrix}
(A - BL) & BL \\
0 & (A - KC)
\end{bmatrix}
\begin{bmatrix}
x \\
e
\end{bmatrix}
\] (4.17)

The second form, is based on the observation that Equation (4.14) is the same as $\dot{\hat{x}} = A\hat{x} + Bu + KCe$. Then, the compound system is found to be

\[
\begin{bmatrix}
\dot{x} \\
\dot{\hat{x}}
\end{bmatrix} =
\begin{bmatrix}
(A - BL) & KC \\
0 & (A - KC)
\end{bmatrix}
\begin{bmatrix}
\hat{x} \\
e
\end{bmatrix}
\] (4.18)

Since the compound system matrix of each case is similar to the others, Theorem 3.8 tells us that the eigenvalues of each are the same. Thus, stability for one is equivalent to stability for all. Then, since the compound system matrices of Equation (4.17) or (4.18) are block triangular the eigenvalues of the compound system are those of $(A - BL)$ combined with those of $(A - KC)$. Thus, the compound system is stable if and only if $(A - BL)$ and $(A - KC)$ are both stable.

The importance of this separation principle is that the question of the existence of gains, $L$ and $K$, such that the compound system is stable is addressed by the independent questions of stabilizability and detectability (Definitions 4.5 and 4.6).

Example 4.8. If the feedback and observer gains of Examples 4.6 and 4.7 are applied to Equation (4.17), one finds:

\[
\begin{bmatrix}
\dot{x} \\
\dot{\hat{x}}
\end{bmatrix} =
\begin{bmatrix}
2 & 6 & -2 & 0 \\
-2 & -5 & 1 & 0 \\
0 & 0 & 2 & 4 \\
0 & 0 & -5 & -7
\end{bmatrix}
\begin{bmatrix}
x \\
e
\end{bmatrix}
\]

As expected the eigenvalues of this system matrix are -1, -2, -2 and -3, indicating that the compound system is stable.

4.3 • Controllability and Stabilizability

The notions introduced in this section represent the most basic properties one would desire from a process that is to be controlled. That is, does the open-loop system possess the linear algebraic ability to respond to feedback favorably? More specifically, is the system stabilizable? If the system does not possess this property, then one will know with certainty that a controller will never exist that is capable of meeting this most modest control objective. In such cases, the appropriate action is to go back and re-design the open-loop process such that this condition is meet. As will be shown in subsequent chapters, the properties of this section are only the first step in determining the existence of a controller that is capable of meeting the desired objectives.

4.3.1 • Complete Controllability

We again start with the discrete-time framework and consider the properties of the following linear state-space process

\[
x_{k+1} = A_dx_k + B_d u_k
\] (4.19)
**Definition 4.7.** A linear system $x_{k+1} = A_d x_k + B_d u_k$ is **completely controllable**, if any final state, $\dot{x} \in \mathbb{R}^n$, can be reached in finite time, starting from an initial state of zero.

Let us begin by looking more closely at Equation (4.19).

$$
\begin{align*}
    x_1 &= A_d x_0 + B_d u_0 \\
    x_2 &= A_d x_1 + B_d u_1 = A_d^2 x_0 + A_d B_d u_0 + B_d u_1 \\
    x_3 &= A_d x_2 + B_d u_2 = A_d^3 x_0 + A_d^2 B_d u_0 + A_d B_d u_1 + B_d u_2
\end{align*}
$$

Thus, in general

$$
    x_k = A_d^k x_0 + \sum_{i=0}^{k-1} A_d^{k-1-i} B_d u_i = A_d^k x_0 + \sum_{i=0}^{k-1} A_d^i B_d u_{k-1-i}
$$

(4.20)

This is the discrete-time version of Equation (2.57). If the initial condition is zero and $x_k$ is set equal to $\dot{x}$, then Equation (4.20) can be written as

$$
    \dot{x} = L_{c,k} z_k \quad \text{where} \quad L_{c,k} = \begin{bmatrix} B_d & A_d B_d & A_d^2 B_d & \cdots & A_d^{n-1} B_d \end{bmatrix} \quad \text{and} \quad z_k = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_k \end{bmatrix}
$$

Thus, for a fixed $k$, the question is: Does there exist a solution to $\dot{x} = L_{c,k} z_k$ for all values of $\dot{x} \in \mathbb{R}^n$? The answer to the question is given by the Fundamental Theorem of Linear Algebra (Theorem 3.1) and states that a solution to $\dot{x} = L_{c,k} z_k$ will exist for all values of $\dot{x} \in \mathbb{R}^n$ if and only if $\mathfrak{N}(L_{c,k}) = \mathbb{R}^n$. Thus, the remaining question is to select value $a$ for $k$. Since we are allowed to select any finite value of $k$, let us begin with $k = n + 1$. In this case, the last set of columns of $L_{c,n+1}$ will be $A_d^n B_d$. However, the Cayley-Hamilton Theorem tell us that $A_d^n$ can be expressed as a linear combination of lower powers of $A_d$. This means that the columns of $A_d^n B_d$ are not linearly independent of the other columns of $L_{c,n+1}$, and thus can be omitted from the range space calculation. Clearly, the same will occur for any value of $k > n$. In sum, it is sufficient to select $k \leq n$. Let us now look at the other extreme of $B_d$ being a column vector. In this case, each $A_d^k B_d$ will also be a column vector, and if $k < n$, then $L_{c,k}$ will have less than $n$ columns. Under such a scenario, it will be impossible for the range space of $L_{c,k}$ to be the entire space, $\mathbb{R}^n$. Thus, to generate a condition that will work for all $B_d$ while having the fewest redundant columns we should select $k = n$. The above discussion is summarized in the following theorem, where $L_{c,n}$ is denoted as just $L_c$.

**Theorem 4.5.** A linear state space system $x_{k+1} = A_d x_k + B_d u_k$ is completely controllable if and only if $\mathfrak{N}(L_c) = \mathbb{R}^n$ where $L_c = \begin{bmatrix} B_d & A_d B_d & A_d^2 B_d & \cdots & A_d^{n-1} B_d \end{bmatrix}$.

Stated more simply, the system is completely controllable if and only if the range space of $L_c$ is the entire space. The continuous-time results are nearly identical, although...
the proof is a bit more complicated, see for example Chen, [116], or Kwakernaak & Sivan, [103].

**Definition 4.8.** A linear system $\dot{x} = Ax + Bu$ is **completely controllable**, if any final state, $\tilde{x} \in \mathbb{R}^n$, can be reached in finite time, starting from an initial state of zero.

**Theorem 4.6.** A linear state space system $\dot{x} = Ax + Bu$ is completely controllable if and only if $\mathcal{N}(L_c) = \mathbb{R}^n$ where $L_c = [B \ AB \ A^2B \ \cdots \ A^{n-1}B]$.

The fact that the controllability matrix is of identical form for the discrete-time and continuous-time cases indicates that the property has more to do with the structure of the matrices than the values of the non-zero elements.

**Example 4.9.** Consider the matrix pair

$$
A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}
$$

In this case the controllability matrix $L_c$ is

$$
L_c = [B \ AB] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}
$$

Clearly, the two columns of the controllability matrix are linearly dependent, indicating the pair is not completely controllable. Now, consider a slight modification of $A$

$$
A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{then} \quad L_c = [B \ AB] = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}
$$

Now, the two columns are linearly independent, indicating that the pair is completely controllable and any state can be reached from zero. Now, consider another modification of $B$

$$
A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{then} \quad L_c = [B \ AB] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}
$$

Again, the columns are linearly dependent and the pair is not completely controllable.

The concept of controllability can now be used to answer the stabilizability question of whether or not there exists a linear feedback, $L$, such that $A - BL$ is stable. The answer lies in the much celebrated **Pole Placement Theorem**.

**Theorem 4.7.** Consider real matrices $A$ and $B$. The pair $(A, B)$ is completely controllable if and only the eigenvalues of $(A - BL)$ can be assigned arbitrarily (assuming the complex conjugate of each is also an eigenvalue) by appropriate selection of a real matrix $L$.

The proof of the pole placement theorem is a bit involved, so the interested reader is referred to Chen, [116], for details. To help the reader develop some insight, the following example is provided.
Example 4.10. Consider the following pair:

\[
A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]

These are easily shown to be completely controllable, since the controllability matrix is the identity matrix. If the linear feedback is \(L = [l_1 \ l_2]\) then, the closed-loop system matrix is \(A - BL = \begin{bmatrix} -l_1 & -l_2 \\ 1 & 0 \end{bmatrix}\) and the characteristic equation is \(\lambda^2 + l_1 \lambda + l_2 = 0\).

This indicates complete freedom in the section of the coefficients of the characteristic equation, which gives complete freedom in the selection of its roots. In contrast, consider

\[
A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]

In this case, the characteristic equation of \(A - BL\) is \(\lambda^2 + l_1 \lambda = 0\). While one of the roots can be arbitrarily selected the other root will be zero regardless of the values of \(L\). As one should expect from the pole placement theorem, the pair \((A, B)\) is not completely controllable, which is verified to be true from \(\Re(L_c) \supseteq \text{span}\{[1 \ 0]^T\}\).

### 4.3.2 The Controllable Subspace

Using the pole placement theorem, we now have a sufficient condition for stabilizability. Specifically, we know that if the pair \((A, B)\) is completely controllable, then one can assign closed-loop eigenvalues to be such that the closed-loop system is stable, either with a magnitude strictly less than 1, or with a real part strictly less than 0. Unfortunately, this observation is not the complete answer to the stabilizability question. Consider the example of \(B = 0\), which guarantees the pair \((A, B)\) will not be completely controllable, regardless of \(A\). Thus, the pole placement theorem cannot be applied. However, if \(A\) is already stable then we know that any feedback will result in a stable closed-loop system. The bottom line is that we would prefer a condition that is not just sufficient, but also necessary. To find such a condition (given in the next subsection) we will need to expand our perspective on controllability. We begin with a few simple examples in the discrete-time framework.

Consider the following system:

\[
\begin{bmatrix}
    x_{k+1}^{(1)} \\
    x_{k+1}^{(2)}
\end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix}
    x_k^{(1)} \\
    x_k^{(2)}
\end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k
\]

If the initial condition is zero, then \(x_k^{(1)} = \sum_{i=0}^{k} u_i\) and \(x_k^{(2)} = 0\). Thus, the only set of final conditions that can be reached are those in \(\text{span}\{[1 \ 0]^T\}\), which happens to be the only linearly independent column of \(L_c\). Now consider the following slightly different system:

\[
\begin{bmatrix}
    x_{k+1}^{(1)} \\
    x_{k+1}^{(2)}
\end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix}
    x_k^{(1)} \\
    x_k^{(2)}
\end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u_k
\]

Similarly, an inspection this system indicates that if the initial condition is zero, then the two elements of the state vector must be equal, \(x_k^{(1)} = \sum_{i=0}^{k} u_i\) and \(x_k^{(2)} = \sum_{i=0}^{k} u_i\). As
you might expect, it is not a coincidence that the set of reachable states, \( \text{span}\{[1 \ 1]^T\} \), is the same as the only linearly independent column of \( L_c \). These observations are made precise by the following definition and fact.

**Definition 4.9.** The **controllable subspace** of the pair \((A, B)\), denoted \( M_C(A, B) \), is the set of final conditions that can be reached within a finite time, assuming a zero initial condition.

**Theorem 4.8.** The controllable subspace is the range space of the controllability matrix. That is, \( M_C(A, B) = \mathbb{R}(L_c) \).

The controllable subspace represents the region of influence for the input \( u \). That is, the structure of the open-loop process will preclude the input, \( u \), from modifying the portion of the state that is outside of the controllable subspace. The following fact, known as the controllable canonical form, is central to illustrating this point.

**Theorem 4.9.** The pair \((A, B)\) is not completely controllable if and only if there exists a projection \( z = T^{-1}x \), such that

\[
T^{-1}AT = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \quad \text{and} \quad T^{-1}B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}
\]

(4.21)

and the pair \((A_{11}, B_1)\) is completely controllable.

To understand the implication of Theorem 4.9, consider the continuous-time system \( \dot{x} = Ax + Bu \). Using of the projection of Theorem 4.9 (or equivalently \( x = Tz \)) leads to the following system in the projected state variable \( \dot{z} \equiv T^{-1}ATz + T^{-1}Bu \), or equivalently:

\[
\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} u
\]

(4.22)

Given this form it is clear that the input, \( u \), is unable to modify the \( z_2 \) portion of the state. The following theorem will be used in the proof of Theorem 4.9 and is of independent interest.

**Theorem 4.10.** The controllable subspace is invariant to the matrix \( A \). That is, if \( y \in \mathbb{R}(L_c) \) then \( Ay \in \mathbb{R}(L_c) \).

The proof of Theorem 4.10 boils down to showing that the columns of \( AL_c \) are a linear combination of the columns of \( L_c \), which is easily observed from the Cayley-Hamilton theorem.

**Proof of Theorem 4.9:** if portion Assume there exists \( T \) such that (4.21) is found and the pair \((A_{11}, B_1)\) is completely controllable. In this case, the only issue is to show that the dimension of \( M_C(A, B) \) is equal to the dimension of \( M_C(T^{-1}AT, T^{-1}B) \). In the later case, the controllability matrix is \( T^{-1}L_c \), where \( L_c \) is the controllability matrix of the former. Using Theorem 3.6, it is concluded that \( \text{rank}(T^{-1}L_c) = \text{rank}(L_c) \). Then, by inspection we conclude that \( (T^{-1}AT, T^{-1}B) \) is not completely controllable or \( \text{rank}(T^{-1}L_c) \) is less than the number of columns of \( A \). Thus, we conclude that \((A, B)\) cannot be completely controllable.

For the only if portion, one will need to construct a matrix \( T \) that gives the desired result. Since \((A, B)\) is not completely controllable, \( M_C(A, B) = \mathbb{R}(L_c) \) is not the full space
and \( \mathcal{M}(L_c) \) is not empty. Thus, the projection can be constructed as \( T = [T_1 \ T_2] \), where the columns of \( T_1 \) form an orthonormal basis for \( \mathcal{M}(L_c) \) and the columns of \( T_2 \) form an orthonormal basis for \( \mathcal{M}(L_c) \). Since all of the columns of \( T \) are orthonormal

\[
T^{-1} = T^* = \begin{bmatrix} T_1^* \\ T_2^* \end{bmatrix}
\]

Then, evaluation of the similarity matrices gives:

\[
T^{-1}AT = \begin{bmatrix} T_1^*AT_1 & T_1^*AT_2 \\ T_2^*AT_1 & T_2^*AT_2 \end{bmatrix} \quad \text{and} \quad T^{-1}B = \begin{bmatrix} T_1^*B \\ T_2^*B \end{bmatrix}
\]

Since each row of \( T_2 \) is orthogonal to \( \mathcal{M}(L_c) \) and each column of \( B \) is in \( \mathcal{M}(L_c) \), it is readily concluded that \( T_2^*B = 0 \). Similarly, \( T_2^*AT_1 \), if each column of \( AT_1 \) is in \( \mathcal{M}(L_c) \). This is guaranteed by Theorem 4.10, since each column of \( T_1 \) is in \( \mathcal{M}(L_c) \). Finally, we will need to show that the pair \( (A_{11}, B) = (T_1^*AT_1, T_1^*B) \) is completely controllable. To do so, consider controllability matrix for the pair \( (T^{-1}AT, T^{-1}B) \)

\[
\tilde{L}_c = \begin{bmatrix}
T_1^{-1}B & (T_1^{-1}AT)^{-1}B & (T_1^{-1}AT)^2T_1^{-1}B & \cdots & (T_1^{-1}AT)^{n-1}T_1^{-1}B \\
\end{bmatrix}
\]

The last equality is due to the fact that each row of \( T_2 \) is orthogonal to \( \mathcal{M}(L_c) \) and each column of \( A^kB \) is in \( \mathcal{M}(L_c) \). Noting that if \( x \in \mathcal{M}(L_c) \), then \( T^{-1}Tx = T_1^*T_1x \) it is concluded that

\[
\tilde{L}_c = \begin{bmatrix}
T_1^*B & (T_1^*AT_1)^{-1}T_1^*B & (T_1^*AT_1)^2T_1^*B & \cdots & (T_1^*AT_1)^{n-1}T_1^*B \\
0 & 0 & 0 & \cdots & 0 \\
\end{bmatrix}
\]

If \( rank(L_c) = r \) then another use of the Cayley-Hamilton theorem indicates that the columns of \( (T_1^*AT_1)^kT_1^*B, k \geq r \) are linearly dependent on the others. Thus, it is sufficient to consider the controllability matrix of \( (A_{11}, B_1) = (T_1^*AT_1, T_1^*B) \)

\[
\tilde{\tilde{L}}_c = \begin{bmatrix}
T_1^*B & (T_1^*AT_1)^{-1}T_1^*B & (T_1^*AT_1)^2T_1^*B & \cdots & (T_1^*AT_1)^{r-1}T_1^*B \\
0 & 0 & 0 & \cdots & 0 \\
\end{bmatrix}
\]

That is, \( \mathcal{M}(\tilde{\tilde{L}}_c) = \mathcal{M}(\tilde{L}_c) \). Then, recalling Theorem 3.6 and that \( \tilde{L}_c = T^{-1}L_c \), we conclude that \( r = rank(L_c) = rank(\tilde{L}_c) = rank(\tilde{\tilde{L}}_c) \) and finally that \( \tilde{L}_c \) is full rank or equivalently \( (A_{11}, B_1) = (T_1^*AT_1, T_1^*B) \) is completely controllable, which completes the proof.

### 4.3.3 Necessary and Sufficient Conditions for Stabilizability

Given the canonical form of Equation (4.22), it is clear that the input \( u \) will have no influence on the state \( z_2 \). To see this more clearly, consider a linear feedback \( u = -[\tilde{L}_1 \ \tilde{L}_2] \)
4.3. Controllability and Stabilizability

Consider the matrix pair $(L, A)$, where

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \text{ which gives } \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} A_{11} - B_1 \tilde{L}_1 & A_{12} - B_1 \tilde{L}_2 \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}.$$ 

Thus, if $A_{22}$ is not stable, then we know that there will not exist a feedback that will make the system stable. On the other hand, if $A_{22}$ is stable, then we need only find an $\tilde{L}_1$ such that $A_{11} - B_1 \tilde{L}_1$ is stable. Since $(A_{11}, B_1)$ is completely controllable, the pole placement theorem tells us that a stabilizing $\tilde{L}_1$ will always exist. In sum, a necessary and sufficient condition for stabilizability of the pair $(A, B)$ is for $A_{22}$ to be stable. The only problem is that to verify this condition one will need to compute $A_{22}$ as $T_2^* A T_2$, which is a bit tedious. The following theorem gives conditions directly related to the pair $(A, B)$.

**Theorem 4.11.** The pair $(A, B)$ is stabilizable if and only if the unstable subspace is contained within the controllable subspace, or $M_{US}(A) \subseteq M_C(A, B)$.

From the preceding paragraph, it has been established that $(A, B)$ is stabilizable if and only if $A_{22}$ is stable. Furthermore, since the eigenvalues of $A$ and $T^{-1} A T$ are the same, $A_{22}$ is stable if and only if all unstable eigenvalues of $A$ are also eigenvalues of $A_{11}$. To simplify the remainder of the proof, let us assume the eigenvalues of $A$ are distinct. In this case, $\lambda$ is an eigenvalue of $A_{11}$ only if it is not an eigenvalue of $A_{22}$ and the associated eigenvector of $T^{-1} A T$ must be of the form $\vec{\phi} = [\vec{\phi}_1^* \ 0]^T$. If the eigenvector associated with $\lambda$ and $A$ is denoted as $\phi$, then the two eigenvectors are related by the projection $\vec{\phi} = T^{-1} \phi$. This tells us that if $\lambda$ is an eigenvalue of $A_{11}$, then $\phi \in \mathbb{R}(L_e) = M_C(A, B)$, since $[\vec{\phi}_1^* \ 0] = [T_1^* \ 0] \phi$. Thus, if all unstable eigenvalues are associated with $A_{11}$, then $M_{US}(A) \subseteq M_C(A, B)$. On the other hand, if $M_{US}(A) \subseteq M_C(A, B)$, then any unstable eigenvector, $\phi$, will be such that $[\vec{\phi}_1^* \ 0] = [T_2^* \ 0] \phi$, which indicates its associated eigenvalue must be an eigenvalue of $A_{11}$.

**Example 4.11.** Consider the matrix pair

$$A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix}, \quad B = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

In this case the controllability matrix $L_e$ is

$$L_e = [B \ AB] = \begin{bmatrix} 2 & 2 \\ -1 & -1 \end{bmatrix}$$

Clearly, the two columns of the controllability matrix are not linearly independent, indicating that the pair is not completely controllable. Using the MATLAB function ‘$\text{orth}(L_e)$’, one finds that the range space is just a single dimension (in the direction of $[-0.8944 \ 0.4472]^T$). Thus, the controllable subspace is

$$M_e(A, B) = \text{span}\{ [-0.8944 \ 0.4472]^T \}$$

Using the MATLAB function ‘$[\text{Phi}, \text{Lam}] = \text{eig}(A)$’, one finds the following eigenvalue and eigenvector matrices (as defined in Section 3.6)

$$\Lambda = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}, \quad \Phi = \begin{bmatrix} 0.8944 & -0.7071 \\ -0.4472 & 0.7071 \end{bmatrix}$$
Thus, the unstable subspace $M_{US}(A)$ is exactly aligned with the controllable subspace, indicating that the pair is stabilizable. The following pair is similarly shown to be not stabilizable since $M_c(A, B) = \text{span}\{[-0.7071 \quad 0.7071]^T\}$

$$A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix} \quad B = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

### 4.4 Observability and Detectability

The notions of the previous section assume full information with regard to the initial condition of the process and focus on the ability to take actions to influence the process. We now turn the second half of feedback - the ability to gather information about the process. Clearly, both abilities (to influence the process as well as gather useful information about its conditions) are required to arrive at an effective controller. This section will focus on the following question: Does the open-loop system possess the ability to estimate the state of the process? More specifically, is the system detectable? If the system does not possess this property, then one will know with certainty that an estimator will never exist that is capable of meeting this most modest objective. In such cases, the appropriate action is to go back and re-design the open-loop process such that these conditions are met. The material of this section also represents our first encounter with the notion of duality. Specifically, we will see that the tests for observability and detectability are very similar to those of controllability and stabilizability. Additional examples of duality between control and observation will occur throughout the text.

#### 4.4.1 Observability and the Unobservable Subspace

We again start with the discrete-time framework, but this time focus on the measurement portion rather than the input.

$$x_{k+1} = A_dx_k$$

$$y_k = Cx_k$$

(4.23)

Before giving the definition of observability consider the following scenario. Assume two experiments are performed, both using the above system and each starting at a different, but unknown, initial condition. In this case, one would expect the measurement sequences, $y_k$, from each experiment to be different. However, if the measurement sequences from each experiment are exactly the same, then we know that it will be impossible to distinguish between the two initial conditions, even if we know they must be different. Avoiding this type of situation motivates the following definition.

**Definition 4.10.** A linear system $x_{k+1} = A_dx_k; \ y_k = Cx_k$ is **completely observable**, if any initial condition, $x_0 \in \mathbb{R}^n$, can be uniquely determined using a finite number of measurements $y_k$. 
Let us begin by looking more closely at Equation (4.23).

\[
\begin{align*}
y_0 &= Cx_0 \\
y_1 &= Cx_1 = CA_dx_0 \\
y_2 &= Cx_2 = CA^2_dx_0 \\
&\vdots \\
y_k &= Cx_k = CA^k_dx_0
\end{align*}
\]

This set of equations can be written as:

\[
z_k = L_{O,k}x_0 \quad \text{where} \quad L_{O,k} = \begin{bmatrix} C & CA_d & CA^2_d & \cdots & C A^{k-1} \\ \end{bmatrix}
\]

and

\[
z_k = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{k-1} \end{bmatrix}
\]

Thus, for a fixed \( k \), the question is: Does there exist a unique solution, \( x_0 \), to the linear equation \( z_k = L_{O,k}x_0 \)? The existence part of the question is answered trivially, due to the fact that the sequence \( y_k \) is assumed to be generated by the system (4.23) using some (although unknown) initial condition. The more interesting part of the question concerns the uniqueness of the solution. To answer this part of the question, we again turn to the Fundamental Theorem of Linear Algebra (Theorem 3.1), which states that a solution to \( z_k = L_{O,k}x_0 \) will be unique if and only if \( \mathbb{N}(L_{O,k}) \) is empty. Thus, the remaining question is to select a value for \( k \). Since we are allowed to select any finite value of \( k \), let us begin with \( k = n + 1 \). In this case, the last set of rows of \( L_{O,n+1} \) will be \( CA^n_d \). However, the Cayley-Hamilton Theorem tell us that \( A^n_d \) can be expressed as a linear combination of lower powers of \( A_d \). This means that the rows of \( CA^n_d \) are not linearly independent of the other rows of \( L_{O,n+1} \), and thus can be omitted from the null space calculation. Clearly, the same will occur for any value of \( k > n \). In sum, it is sufficient to select \( k \leq n \). Let us now look at the other extreme of \( C \) being a single row vector. In this case, each \( CA^k_d \) will also be a single row vector, and if \( k < n \), then will have less than \( n \) rows. Under such a scenario, it will be impossible for the null space of \( L_{O,k} \) to be empty. Thus, to generate a condition that will work for all \( C \) while having the fewest redundant rows, one should select \( k = n \). The above discussion is summarized in the following theorems, where \( L_{O,n} \) is denoted as just \( L_O \).

**Theorem 4.12.** A linear system \( x_{k+1} = A_d x_k; \ y_k = C x_k \) is completely observable if and only if \( \mathbb{N}(L_O) \) is empty, where

\[
L_O = \begin{bmatrix} 
C \\
CA_d \\
CA^2_d \\
\vdots \\
CA^{n-1}_d 
\end{bmatrix}
\]
To bring this result back to the original definition of observability, assume \( \mathbb{N}(L_O) \) is not empty and consider a non-zero initial condition in \( \mathbb{N}(L_O) \). Then, that initial condition can be added to any other non-zero initial condition to arrive at two distinct initial conditions that generate the same output sequence \( y_k \). However, if \( \mathbb{N}(L_O) \) is empty, then this situation can never occur.

The continuous-time result is again nearly identical to the discrete-time version.

**Theorem 4.13.** A linear system \( \dot{x} = Ax; \ y = Cx \) is completely controllable if and only if \( \mathbb{N}(L_O) \) is empty, where

\[
L_O = \begin{bmatrix}
  C \\
  CA \\
  CA^2 \\
  \vdots \\
  CA^{n-1}
\end{bmatrix}
\]

If the pair \((A, C)\) is not completely observable, then it will be of interest to determine the set of initial condition that cannot be observed.

**Definition 4.11.** The unobservable subspace of the pair \((A, C)\), denoted \( M_{UO}(A, C) \), is the set initial condition such that the output is zero for all time.

**Theorem 4.14.** The unobservable subspace is calculated as: \( M_{UO}(A, C) = \mathbb{N}(L_O) \).

### 4.4.2 First Experience with Duality

It is expected that most readers will find a great deal of similarity between the concepts of Sections 4.3 and 4.4. This is by far no coincidence. In fact, the notions of controllability and observability are intimately linked. As such, the question of observability is frequently denoted as the dual of controllability. To see this duality, let us begin by recalling Theorem 3.2, which indicates that \( \mathbb{N}(L_O) \) will be empty if and only if \( \mathbb{N}(L_{*O}) \) is the full space. However, the adjoint of the observability matrix bears a striking resemblance to the controllability matrix.

\[
L_{*O} = \begin{bmatrix}
  C^* & A^*C^* & A^{2*}C^* & \cdots & A^{*(n-1)}C^*
\end{bmatrix}
\]

Given this observation, the following corollaries are easily proven:

**Corollary 4.1.** The pair \((A, C)\) is completely observable if and only if the pair \((A^*, C^*)\) is completely controllable.

**Corollary 4.2.** The unobservable subspace of the pair \((A, C)\) is equal to the controllable subspace of the pair \((A^*, C^*)\). That is, \( M_{UO}(A, C) = M_{C}(A^*, C^*) \).

The following estimator version of the pole placement theorem can be easily proven, given the original theorem and the notion of duality.

**Theorem 4.15.** Consider real matrices \( A \) and \( C \). The pair \((A, C)\) is completely observable if and only if the eigenvalues of \( A - KC \) can be assigned arbitrarily (assuming the complex conjugate of each is also an eigenvalue) by appropriate selection of a real matrix \( K \).
The proof hinges on the fact that the eigenvalues of a real matrix and its adjoint are identical (recall Theorem 3.6). Thus, assignment of eigenvalues for \((A^* - C^*K^*)\) is the same as assignment for \(A - KC\). Then, given Corollary 4.1, it is clear that Theorem 4.15 is identical to Theorem 4.7, except that \((A, B)\) is replace by \((A^*, C^*)\).

### 4.4.3 Necessary and Sufficient Conditions for Detectability

Similar to the discussion of Section 4.3.2, the pole placement theorem indicates that complete observability is a sufficient condition for detectability, but it is clearly not necessary. For example, if \(C = 0\), then \((A, C)\) will not be completely observable regardless of \(A\). However, if that \(A\) is stable, then any matrix \(K\) will render \(A - KC\) stable and thus the system detectable. To arrive at necessary and sufficient conditions for detectability, we will follow a path similar to that of Section 4.3.3. Specifically, the system will be transformed into the observable canonical form.

**Theorem 4.16.** The pair \((A, C)\) is not completely observable if and only if there exists a projection \(z = T^{-1}x\), such that

\[
T^{-1}AT = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \quad \text{and} \quad CT = \begin{bmatrix} C_1 \\ 0 \end{bmatrix}
\]

and the pair \((A_{11}, C_1)\) is completely observable.

To understand the implication of Theorem 4.16, consider the continuous-time system \(\dot{x} = Ax; \ y =Cx\). Use of the projection of Theorem 4.16 (or equivalently \(x = Tz\)) leads to the following system in the projected state variable \(\dot{z} = T^{-1}ATz; \ y = CTz\), or equivalently:

\[
\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}
\]

\[
y = \begin{bmatrix} C_1 \\ 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}
\]

(4.25)

Given this form it is clear that the output, \(y\), is unable to observe the \(z_2\) portion of the state. The proof of Theorem 4.16 (see Exercise 4.20) is based on constructing the projection as \(T = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}\), where the columns of \(T_2\) form an orthonormal basis for \(\mathbb{N}(L_O)\) and the columns of \(T_1\) form an orthonormal basis for \(\mathbb{N}(L_O)^\perp\). It will also require the following Theorem concerning invariance of the unobservable subspace.

**Theorem 4.17.** The unobservable subspace is invariant to the matrix \(A\). That is, if \(x \in \mathbb{N}(L_O)\) then \(Ax \in \mathbb{N}(L_O)\).

Using the observable canonical form of Equation (4.25) to construct the error equation of the associated state estimator, one finds

\[
\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} = \left( \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} - \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} \begin{bmatrix} C_1 \\ 0 \end{bmatrix} \right) \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}
\]

\[
= \begin{bmatrix} A_{11} - K_1C_1 & 0 \\ A_{21} - K_2C_1 & A_{22} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}
\]
Since the pair \((A_{11}, C_1)\) is completely observable, one may employ the pole placement theorem to determine a stabilizing matrix \(K_1\). Then, the question of detectability will boil down to the stability of \(A_{22}\). Specifically, \(A_{22}\) is stable if and only if the pair \((A, C)\) is detectable. Given this initial argument, the proof of Theorem 4.18 is left as an exercise.

**Theorem 4.18.** The pair \((A, C)\) is detectable if and only if the unobservable subspace is contained within the stable subspace, or \(M_{uo}(A, C) \subseteq M_S(A)\).

**Example 4.12.** Consider the matrix pair

\[
A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix}, \quad C = \begin{bmatrix} 2 \\ 3 \end{bmatrix}
\]

In this case the observability matrix \(L_O\) is

\[
L_O = \begin{bmatrix} C \\ CA \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ -1 & -3 \end{bmatrix}
\]

Since the columns of \(L_O\) are linearly independent, the null space of \(L_O\) is the empty. Thus, the pair is completely observable.

**Example 4.13.** Consider the matrix pair

\[
A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix}, \quad C = \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\]

In this case the observability matrix \(L_O\) is

\[
L_O = \begin{bmatrix} C \\ CA \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}
\]

Since the columns of \(L_O\) are not linearly independent, the pair is not completely observable. Using the MATLAB function ‘null(Lo)’, one finds that the unobservable subspace is \(M_{uo}(A, C) = \text{span}\{ \begin{bmatrix} -0.7071 \\ 0.7071 \end{bmatrix} \} \), which is exactly aligned with the stable subspace, indicating that the pair is detectable. The following pair is similarly shown to be not detectable.

\[
A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix}, \quad C = \begin{bmatrix} 1 \\ 2 \end{bmatrix}
\]

### 4.5 - Chapter Summary

This chapter has presented a set of analysis tools representing a preliminary stage of control system design. If a proposed open-loop system (designated by a triple of system matrices: \(A, B, C\)) fails to pass the stabilizability and detectability tests, then one is confident that further analysis of this system is futile, as it has been revealed to be such that there does not exist a feedback element capable of achieving the most basic criterion of a control system - the property of being a stable closed-loop system. In the design methods
<table>
<thead>
<tr>
<th></th>
<th>Continuous-time</th>
<th>Discrete-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability</td>
<td>Theorem 4.2</td>
<td>Theorem 4.1</td>
</tr>
<tr>
<td>Controllability</td>
<td>Theorem 4.6</td>
<td>Theorem 4.5</td>
</tr>
<tr>
<td>Stabilizability</td>
<td>Theorem 4.11</td>
<td>Theorem 4.11</td>
</tr>
<tr>
<td>Observability</td>
<td>Theorem 4.13</td>
<td>Theorem 4.12</td>
</tr>
<tr>
<td>Detectability</td>
<td>Theorem 4.18</td>
<td>Theorem 4.18</td>
</tr>
</tbody>
</table>

of subsequent chapters, the tests of this chapter will be implicit. That is, while additional performance criteria may be added, the resulting procedures will be guaranteed to fail if the open-loop system is not stabilizable and detectable. It is also emphasized that all of the presented results are conceptually valid in both the continuous- and discrete-time frameworks. Table 4.1 highlights this fact along with the notion of duality with respect to manipulated variable influence (controllability and stabilizability) and measurement variable information (observability and detectability). This notion of duality in both frameworks will continue to occur throughout the text.

For expanded discussions of the material of this chapter, especially results concerning the continuous-time cases, appropriate references are Brockett [118], Kwakernaak & Sivan [103], Balakrishnan [115] and Chen [116]. It should be noted that the controllability proof of this chapter was adapted from Graham and Rawlings [113] while the stabilizability proof was adapted from Chen [116].

### Exercises

4.1. Consider the following second order differential equation

\[
\dot{x} = Ax \quad \text{where} \quad A = \begin{bmatrix} -2 & 0 \\ -1 & -1 \end{bmatrix} \quad \text{and} \quad x(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]

Determine an analytic expression for \( x(t) = [x_1(t) \ x_2(t)]^T \). It will be helpful to know that the eigenvalue/eigenvector pairs for \( A \) are as follows (you should verify this by hand)

\[
\lambda_1 = -1 \quad \text{and} \quad \phi_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{and} \quad \lambda_2 = -2 \quad \text{and} \quad \phi_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\]

4.2. Consider the following state space model of a physical process \( \dot{x} = Ax \), where

\[
A = \begin{bmatrix} 1 & 3 \\ 0 & -2 \end{bmatrix}
\]

Determine the eigenvalues and eigenvectors of this process. Do not use the MATLAB function ‘eig’. Is the process stable?

4.3. Repeat Exercise 4.2 with \( A = \begin{bmatrix} 1 & 5 \\ 0 & -2 \end{bmatrix} \)

4.4. Repeat Exercise 4.2 with \( A = \begin{bmatrix} -4 & \sqrt{3} \\ \sqrt{3} & -2 \end{bmatrix} \)
4.5. Consider the following model of a mass spring system 

\[ \dot{x} = Ax + Bu; \quad A = \begin{bmatrix} 0 & 1 \\ -3 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \]

where the first state of \( x \) is mass position, the second is mass velocity and the manipulated variable, \( u \), is applied force.

(i) Determine the eigenvalues of \( A \) (feel free to use the MATLAB function ‘eig’). Is the open-loop system stable?

(ii) Convert this continuous-time model to discrete-time using the sample-and-hold method and a sample period of 0.2.

(iii) Determine the eigenvalues of \( A_d \). Verify that these eigenvalues correspond to those of the continuous-time system.

(iv) Simulate the open-loop discrete-time system using an initial condition \( x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \). Make a plot of the output signal, \( y_k \).

(v) Using the linear feedback \( u_k = -Lx_k \) with \( L = \begin{bmatrix} 0 & 0.2 \end{bmatrix} \), determine the eigenvalues of the closed-loop system: \( x_{k+1} = (A_d - B_d L)x_k \).

(vi) Simulate the closed-loop system. Make a plot comparing the open-loop and closed-loop simulations.

4.6. Consider the following model of a mass spring system 

\[ \dot{x} = Ax; \quad y = Cx; \quad A = \begin{bmatrix} 0 & 1 \\ -3 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \end{bmatrix}, \]

where the first state of \( x \) is mass position, the second is mass velocity, the input, \( u \), is applied force and the output, \( y \), is the sum of the two state variables.

(i) Determine the eigenvalues of \( A \) (feel free to use the MATLAB function ‘eig’). Is the open-loop system stable?

(ii) Convert this continuous-time model to discrete-time using the sample-and-hold method and a sample period of 0.2.

(iii) Determine the eigenvalues of \( A_d \). Is the open-loop discrete-time system stable? Verify that these eigenvalues of the discrete-time system correspond to those of the continuous-time system.

(iv) Simulate the open-loop discrete-time system using an initial condition \( x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \). Make a plot of the output signal, \( y_k \).

(v) Using the gain \( K = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \), determine the eigenvalues of the error system, 
\[ e_{k+1} = (A_d - KC)e_k, \]
which is with associated the discrete-time state observer 
\[ \hat{x}_{k+1} = A_d\hat{x}_k + K(y_k - C\hat{x}_k). \]

(vi) Using the output signal, \( y_k \), and observer gain \( K = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \), simulate the discrete-time state observer \( \hat{x}_{k+1} = A_d\hat{x}_k + K(y_k - C\hat{x}_k) \), using an initial condition \( \hat{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \). Make a plot comparing the open-loop state signals (from part iv) and the state estimate signals (of this part).

(vii) Make a plot to verify that a simulation using the error system, \( e_{k+1} = (A_d - KC)e_k \), will be identical to the error signals calculated manually (i.e., \( e_k = x_k - \hat{x}_k \) where \( x_k \) and \( \hat{x}_k \) are from part vi).

4.7. Consider the following model of a mass spring system 

\[ \dot{x} = Ax + Bu; \quad y = Cx; \quad A = \begin{bmatrix} 0 & 1 \\ -3 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \end{bmatrix}, \]

where the first state of \( x \) is mass position, the second is mass velocity, the input, \( u \), is applied force and the output, \( y \), is the
sum of the two state variables.

(i) Convert this continuous-time model to discrete-time using the sample-and-hold method and a sample period of 0.2. Determine the eigenvalues of $A_d$ (feel free to use the MATLAB function ‘eig’).

(ii) Using the linear feedback $u_k = -Lx_k$ with $L = \begin{bmatrix} 0 & 0.2 \end{bmatrix}$, determine the eigenvalues of the closed-loop system: $x_{k+1} = (A_d - B_dL)x_k$. Using the gain $K = \begin{bmatrix} 0 & 0.2 \end{bmatrix}$, determine the eigenvalues of the error system, $e_{k+1} = (A_d - KC)\hat{e}_k$, associated the discrete-time state observer $\hat{x}_{k+1} = A_d\hat{x}_k + K(y_k - C\hat{x}_k)$.

(iii) Calculate the eigenvalues of the system matrix of the discrete-time version of Equation (4.16) (i.e., with $A$ and $B$ replaced by $A_d$ and $B_d$). Verify that these eigenvalues are identical to those of part ii.

(iv) Calculate the eigenvalues of the system matrices of the discrete-time versions of Equations (4.17) and (4.18). Verify that these eigenvalues are identical to those of part iii.

(v) Simulate the discrete-time version of the system of Equation (4.16), using initial conditions $x_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}^*$ and $\hat{x}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^*$ to determine the signals $x_k$ and $\hat{x}_k$. Calculate and plot $e_k = x_k - \hat{x}_k$ to verify that it is tending toward zero.

4.8. Consider the following state space model of a physical process:

$$\dot{x} = Ax + Bu, \quad y = Cx,$$

where $A = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $C = \begin{bmatrix} 0 & 1 \end{bmatrix}$

(i) Is the pair $(A; B)$ stabilizable?

(ii) Is the pair $(A; C)$ detectable?

4.9. Consider the following state space model of a physical process:

$$\dot{x} = Ax + Bu, \quad y = Cx,$$

where $A = \begin{bmatrix} 1 & 3 \\ 0 & -2 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

(i) Is the pair $(A; B)$ stabilizable?

(ii) Is the pair $(A; C)$ detectable?

4.10. You have been assigned the task of creating state observers for the following systems. The person previously in your position seems to remember that for one of these systems, a state observer could not be implemented. Is she correct?

(i) $\dot{x} = \begin{bmatrix} -3 & 1 \\ 0 & -3 \end{bmatrix} x$, $y = \begin{bmatrix} 0 & 1 \end{bmatrix} x$

(ii) $x_{k+1} = \begin{bmatrix} 1.5 & 1 \\ 0 & 0.5 \end{bmatrix} x_k$, $y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k$

(iii) $\dot{x} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} x$, $y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$
4.11. Consider the linear system $\dot{x} = Ax + Bu$, with

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 & -3 & -1 \end{bmatrix}$$

You are given a choice of two actuators. Each result in the following $B$ matrices

$$B_1 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \quad B_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

Which of these would you choose? Why?

4.12. Consider the linear system

$$x_{k+1} = \begin{bmatrix} 1 & -2 & 0 \\ -3 & -1 & 2 \\ 0 & 0 & -2 \end{bmatrix} x_k + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} u_k$$

Is the system Stable? Controllable? Stabilizable?

4.13. Determine the abilities of the linear system associated with the following triples (take note of the subscript notation for discrete-time systems). More specifically, determine if the system is stable, completely controllable, completely observable, stabilizable or detectable? (Feel free to use the MATLAB commands ‘orth’ and ‘null’)

(i) \(A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}\)

(ii) \(A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}\)

(iii) \(A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\)

(iv) \(A = \begin{bmatrix} -1 & -2 & 0 \\ -3 & -1 & 2 \\ 0 & 0 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\)

(v) \(A = \begin{bmatrix} -1 & -2 & 0 \\ -3 & -1 & 2 \\ 0 & 0 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\)

(vi) \(A_d = \begin{bmatrix} 1.1 & -0.2 & 0 \\ -0.3 & 0.9 & 0.2 \\ 0 & 0 & 0.8 \end{bmatrix} \quad B_d = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\)

4.14. Repeat the stabilizable and detectable parts of Exercise 4.7, by constructing the controllable and observable canonical forms of the system.

4.15. Consider an isothermal continuously stirred tank reactor (CSTR) in which a reversible reaction is being carried out, where the forward and reverse kinetic constants ($k_a$ and $k_b$, respectively) are defined with respect to species $B$.

(i) Determine the nonlinear dynamic model for this system assuming the inlet concentration of species $A$ is the manipulated variable. Other relevant information: $k_a = 0.05$, $k_b = 0.01$, $F/V = 0.02$ and $C_{SSOP}^{Ai} = 10$ (all in SI units).
(ii) Under these operating conditions verify that the steady-state concentrations are equal to $C_{SSOP}^A = 1.59$ and $C_{SSOP}^B = 4.21$.

(iii) Determine the linearized dynamic model of the process.

(iv) Fill in the following table:

<table>
<thead>
<tr>
<th>Is the linearized process</th>
<th>Yes</th>
<th>No</th>
<th>Cannot be determined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controllable?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stabilizable?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observable?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detectable?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.16. Consider the matrix equality $A^* P + PA + I = 0$ with

$$A = \begin{bmatrix} -1 & 12 \\ 0 & -2 \end{bmatrix}$$

If the elements of the matrix $P$ are denoted as

$$P = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix}$$

Solve the matrix equality without the aid of a computer. You should arrive at 3 linear equations that can easily be solved by hand. Verify your answer using the MATLAB function ‘lyap’.

4.17. Consider the matrix equality $A^* P + PA + I = 0$ with

$$A = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix}$$

If the elements of the matrix $P$ are denoted as

$$P = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix}$$

Solve the matrix equality without the aid of a computer. You should arrive at 3 linear equations that can easily be solved by hand. Verify your answer using the MATLAB function ‘lyap’.

4.18. A matrix $A$ is denoted $\sigma$-stable if $\text{Re}(\lambda_i(A)) < \sigma$. Prove that $A$ is $\sigma$-stable if and only if there exists $P > 0$ such that $A^* P + PA < 2\sigma P$.

4.19. Prove the discrete-time version of the Lyapunov theorem (Theorem 4.4):

(i) If there exists $P > 0$ such that $A_d^* P A_d - P < 0$ then $A_d$ is stable (i.e., $|\lambda_i(A_d)| < 1$).

(ii) If $A_d$ is stable then there exists $P > 0$ such that $A_d^* P A_d - P < 0$. (Hint: Define $P$ as $\sum_{k=0}^{\infty} A_d^{-k} Q A_d^k$.)
4.20. Prove Theorem 4.16.
4.22. Prove Theorem 4.18.
4.23. Answer the following questions as either True or False
   (i) If $A^* + A < 0$, then $A$ is stable.
   (ii) If $A$ is stable then $A^* + A < 0$.
   (iii) The linear system $\dot{x} = Ax + u$ is completely controllable.
Chapter 5
Stochastic Processes

As illustrated in the example of Section 1.4, the characteristics of the disturbances acting on a process will significantly influence the performance one should expect. One approach to disturbance modeling is the notion of a stochastic process. Such processes are a natural extension of traditional random variable notions to the time dependent systems. In general, the discrete-time framework results in a fairly a simple exposition of these entities. As such, this framework will be used to develop the main results, and then analogous results will be presented for the continuous-time framework. The continuous-time framework will also be used to illustrate the frequency domain perspective of stochastic processes. Using this frequency domain perspective, methods to design simple stochastic based disturbance models will be presented.

5.1 Review of Probability and Random Variables

Consider a scalar random variable, \( x \), with a probability density function \( p(x) \). Then, the probability that a realization of the random variable will be between the values \( x_1 \) and \( x_2 \) (\( x_2 > x_1 \)) is the integral of \( p(x) \) from \( x_1 \) to \( x_2 \):

\[
\Pr [x_1 < x < x_2] = \int_{x_1}^{x_2} p(x) \, dx \tag{5.1}
\]

Based on this definition it is concluded that all probability density functions must be positive (to ensure that the probability of all intervals is positive) and that the total area under the function is equal to 1, so that:

\[
\Pr [-\infty < x < \infty] = \int_{-\infty}^{\infty} p(x) \, dx \triangleq 1 \tag{5.2}
\]

This identity is frequently denoted as the zeroth moment of the random variable. The average, mean or expected value (all synonyms) of a random variable, \( x \), is frequently denoted as the first moment:

\[
\bar{x} = E[x] \triangleq \int_{-\infty}^{\infty} x \, p(x) \, dx \tag{5.3}
\]
The mean removed second moment will then provide the variance of the random variable:

$$\sigma^2 = E[(x - \overline{x})^2] = \int_{-\infty}^{\infty} (x - \overline{x})^2 p(x) dx \quad (5.4)$$

The square root of the variance, $\sigma$, is denoted as the standard deviation. In general, one could continue on and calculate the third, fourth and higher order moments. In the end, the infinite sequence of moments will provide information equivalent to the original probability density function. One exception is the special class of Gaussian (or normal) probability density functions. In this case, the first two moments are sufficient to specify the entire density function. Specifically, the Gaussian density function is defined as:

$$p(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left[ -\frac{1}{2} \left( \frac{x - \overline{x}}{\sigma} \right)^2 \right] \quad (5.5)$$

Example 5.1. Consider a ball being shot from a canon and we are interested in the distance the ball travels. If the angle of the canon is exactly the same for each shot then one would expect distance traveled would be the same. However, due to slight differences in ball weight, amount and quality of powder and wind conditions each ball will travel a slightly different distance. Assume this variability in distance shot is captured by the Gaussian density function. The parameter $\overline{x}$ is set to 100 and $\sigma$ is 5, then the density function of Figure 5.1 will result. Notice that the probability of the shot landing within the range of 95 to 105 (one standard deviation in each direction from the mean $\overline{x}$) is 0.683 (or 68.3%). Similarly, the probability of landing within two standard deviations of the mean (from 90 to 110) is 95.5%. Of course, these statistics are only valid for the Gaussian distribution.

![Figure 5.1. Probability density function for Example 5.1](image-url)
function \( p(x, y) \). The relationship between \( p(x) \), \( p(y) \) and \( p(x, y) \) is as follows:

\[
p(x) = \int_{-\infty}^{\infty} p(x, y) dy \quad \text{and} \quad p(y) = \int_{-\infty}^{\infty} p(x, y) dx
\]

(5.6)

If the two random variables are put into vector notation, \( z = [x \ y]^T \), then the mean of \( z \) is denoted as \( \bar{z} = [\bar{x} \ \bar{y}]^T \) and the mean removed covariance matrix of \( z \) is

\[
\Sigma_z = E[(z - \bar{z})(z - \bar{z})^T] = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}
\]

(5.7)

The individual terms of the covariance matrix are calculated as extensions of (5.4):

\[
\sigma_x^2 = E[(x - \bar{x})^2] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^2 P(x, y) dxdy = \int_{-\infty}^{\infty} (x - \bar{x})^2 p(x) dx
\]

(5.8)

\[
\sigma_y^2 = E[(y - \bar{y})^2] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (y - \bar{y})^2 P(x, y) dxdy = \int_{-\infty}^{\infty} (y - \bar{y})^2 p(y) dy
\]

(5.9)

\[
\sigma_{xy} = E[(x - \bar{x})(y - \bar{y})] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})(y - \bar{y}) P(x, y) dxdy
\]

(5.10)

Note that the diagonal elements of \( \Sigma_z \) are the variances of the individual random variables. The off-diagonal term is denoted as the cross-covariance (or just covariance), and indicates the relationship between the two random variables. Specifically, if the magnitude of \( \sigma_{xy} \) is large then the two random variables are expected to have similar realizations, and if close to zero the two realization will be unrelated. A normalized version of covariance is the (cross-)correlation coefficient, defined as \( \rho_{xy} = \sigma_{xy}/\sigma_x \sigma_y \). This unit-less parameter can range from -1 to 1. If \( \rho_{xy} = 1 \), then \( x \) and \( y \) will be equal for all realizations. If \( \rho_{xy} = -1 \), then \( x \) will equal \(-y\) for all realizations. If \( \rho_{xy} = 0 \), then \( x \) and \( y \) will be unrelated for all realization.

In the case of two Gaussian random variables, the joint density function is defined as:

\[
p(z) = \frac{1}{\sqrt{\Sigma_z} \sqrt{2\pi}} \exp \left[ -\frac{1}{2} (z - \bar{z}) \Sigma_z^{-1} (z - \bar{z})^T \right]
\]

(5.11)

Again, the characterization of this density function requires only the specification of the mean, \( \bar{z} \), and the mean removed covariance matrix, \( \Sigma_z \).

**Example 5.2.** Continuing Example 5.1, let us additionally consider the impact of wind speed. Specifically, if the wind speed is high, then we expect the distance the ball will travel to be larger (assuming the wind is from behind). If the wind speed is considered a second random variable, then one may assume the joint probability density function is again of the Gaussian form. Assume the mean of the two random variable are given by the parameters \( \bar{x} = 100 \) and \( \bar{y} = 10 \). Furthermore, assume the standard deviations are \( \sigma_x = 5 \) and \( \sigma_y = 2 \). The last item to specific is the correlation coefficient, which we set to 0.8. Then, from the formula \( \rho_{xy} = \sigma_{xy}/\sigma_x \sigma_y \), we find that \( \sigma_{xy} = 8 \). Thus, the
covariance matrix is found to be 
\[
\Sigma_x = \begin{bmatrix} 
\sigma_x^2 & \sigma_{xy} \\
\sigma_{xy} & \sigma_y^2 
\end{bmatrix} = \begin{bmatrix} 
25 & 8 \\
8 & 4 
\end{bmatrix}
\]
and has a determinant value of 36. The top plot of Figure 5.2 shows a mesh plot of this joint density function. Notice that the probability of the shot landing within a box defined by 95 to 105 and 8 to 12 (one standard deviation in each direction from the mean \(\bar{z}\)) is again 0.683 (or 68.3%). Similarly, the probability of landing within two standard deviations of the mean (from 90 to 110 and 6 to 14) is 95.5%. The more important issue is the correlation between the two
variables. For example, if one finds that the ball travels a distance of 85, then from the top plot of Figure 5.2 one concludes that the wind speed was likely low, somewhere around 5.5. This can also be seen in the contour plot of Figure 5.2 (curves of equal density value). The dashed line is the major axis of each ellipse, and indicates the correlation direction. The size of the ellipse in the minor axis direction (the dotted line), indicates how much we can trust this prediction. That is, if the ellipse was more narrow (due to a larger correlation coefficient), then we would have more trust in the prediction of a wind speed of 5.5. In the limit of a correlation coefficient approaching 1, the ellipse would collapse to the major axis and we could take the wind speed prediction with utmost confidence. As the correlation coefficient approaches zero, the ellipses become circles and indicate then there is no relation between wind speed and distance. If the correlation coefficient was negative, then the major axis would have a negative slope (essentially, the dashed and dotted lines would be swapped). In this case the predictions would be reversed. That is, a large wind speed would be correlated with shorter distances, which would be appropriate if the wind was not from behind, but into our face.

If we have a collection \( n_z \) random variables, then using vector notation the random vector \( \mathbf{z} \), defined as:

\[
\mathbf{z} = \begin{bmatrix} \mathbf{z}^{(1)} \\ \mathbf{z}^{(2)} \\ \vdots \\ \mathbf{z}^{(n_z)} \end{bmatrix}
\]

and a covariance matrix defined as:

\[
\Sigma_z = E[(\mathbf{z} - \overline{\mathbf{z}})(\mathbf{z} - \overline{\mathbf{z}})^\mathsf{T}] = \begin{bmatrix} E[\mathbf{z}^{(1)}\mathbf{z}^{(1)}] & E[\mathbf{z}^{(1)}\mathbf{z}^{(2)}] & \cdots & E[\mathbf{z}^{(1)}\mathbf{z}^{(n_z)}] \\ E[\mathbf{z}^{(2)}\mathbf{z}^{(1)}] & E[\mathbf{z}^{(2)}\mathbf{z}^{(2)}] & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ E[\mathbf{z}^{(n_z)}\mathbf{z}^{(1)}] & E[\mathbf{z}^{(n_z)}\mathbf{z}^{(2)}] & \cdots & E[\mathbf{z}^{(n_z)}\mathbf{z}^{(n_z)}] \end{bmatrix}
\]

\[
= \begin{bmatrix} \sigma^2_1 & \sigma_{12} & \cdots & \sigma_{1n_z} \\ \sigma_{21} & \sigma^2_2 & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n_z1} & \sigma_{n_z2} & \cdots & \sigma^2_{n_z} \end{bmatrix}
\]

where \( \mathbf{z}^{(j)} = \mathbf{z}^{(j)} - \overline{\mathbf{z}}^{(j)} \) is the mean removed version of \( \mathbf{z}^{(j)} \), \( \sigma^2_j \) is the mean removed variance of \( \mathbf{z}^{(j)} \) and \( \sigma_{ij} \) is the covariance between \( \mathbf{z}^{(i)} \) and \( \mathbf{z}^{(j)} \). If the random vector is also Gaussian, then the density function is:

\[
p(\mathbf{z}) = \frac{1}{|\Sigma_z|^{1/2} (\sqrt{2\pi})^{n_z}} \exp \left[ -\frac{1}{2} (\mathbf{z} - \overline{\mathbf{z}})^{\mathsf{T}} \Sigma_z^{-1} (\mathbf{z} - \overline{\mathbf{z}}) \right]
\]

(5.14)

It should be highlighted that this collection of \( n_z \) random variables is in an arbitrary order. That is, one could rearrange the order of the random variables within the vector \( \mathbf{z} \) and the only change would be with regard to notation. In the following section, we will see that the order of the random variables within a stochastic sequence is of paramount importance.
5.2 Introduction to Stochastic Sequences

In the discrete-time framework, a stochastic process is simply a sequence of random variables and can be thought of as an extension of the two random variable case of the previous section. The distinctive aspect of a stochastic process (in contrast to an arbitrary collection of random variables) is that the index of the sequence is associated with time. As we will see, this notion of a time index will provide an important physical interpretation in the sense that the time distance between two random variables will have an impact on the statistics of those variables. While this distance (or time difference) may (or may not) have an impact on the mean and variance of each random variable of the stochastic process, we will definitely find that the amount of correlation between two variables of the sequence will depend heavily on the time difference and is actually at the core of our characterization of a stochastic process. The following example provides a first illustration of these concepts.

Example 5.3. In the example of Section 1.3, it is given that the inlet flow disturbance is anticipated to be \( v^{(\text{in})} = 30 \pm 3 \text{ m}^3/\text{min} \). If this disturbance is assumed to be a stochastic process, denoted \( \{v^{(\text{in})}_k\} \), then it is clearly appropriate to assume the mean of each random variable, \( v^{(\text{in})}_k \), to be equal to 30 \text{ m}^3/\text{min} \). Similarly, it is reasonable to assume that the variance of each random variable, \( v^{(\text{in})}_k \), to be equal to 9 \text{ m}^6/\text{min}^2 \), which would yield a standard deviation of 3 \text{ m}^3/\text{min} \). In this case, the mean and variance statistics do not change with the time index.

In Section 1.3 it was also concluded that the time structure of the disturbance greatly influenced its impact on process outputs. Recall how the disturbance of the left plot of Figure 1.18 caused larger variations at \( v \) and \( V \) as compared to those of the right plot, even though each disturbance had the same mean and variance. A general observation about both disturbances is the following. If the disturbance is at a particular value at a particular time, then one expects the disturbance to be at that same value in the near future. However, as we go farther into the future, this expectation (of being at the same value) will diminish. Looking at the left plot disturbance, the term “near future” should correspond to about 3-5 minutes, while for the right plot disturbance, the notion of near future would only be about 1-3 minutes. To capture similar (duration type) characteristics within the stochastic framework, we will need to extend the simple notion of correlation between two random variables to the more general concept of autocorrelation. That is,
how are two elements of a stochastic process correlated with each other, and how does this correlation depend on the time distance between the two elements of the sequence.

5.2. Introduction to Stochastic Sequences

5.2.1 Mean, Variance and Autocorrelation

Let us denote a stochastic sequence as \( p_k \), and assume (for the moment) each of these random variables is a scalar. Define the mean and variance of the random variable sequences by the following sequences:

\[
\bar{p}_k = E[p_k] \quad \text{and} \quad \Sigma_{p,k} = E[(p_k - \bar{p}_k)^2]
\]  

(5.15)

As indicate by Example 5.3, the mean and variance sequences may be constant over time, though this is not required. The more important question is how much each random variable is correlated with others in the sequence? This relationship is defined by the autocorrelation function:

\[
R_{p,k,i} = E[(p_k - \bar{p}_k)(p_{k+i} - \bar{p}_{k+i})]
\]  

(5.16)

Let us now recall the disturbances of Figure 1.18. Intuition suggests that we should expect high correlation in the near future and lower correlation in the distant future. Thus, the distinction between the two disturbances of Figure 1.18 should be how quickly correlation decreases as the time difference between two random variables increases. That is, the autocorrelation, \( R_{p,k,i} \), as a function of time difference, \( i \), will decrease faster for the left disturbance and slower for the right. The central point is that if the autocorrelation function, \( R_{p,k,i} \), is different for two stochastic processes, then the time series characteristics of each stochastic process will be different, even if both have identical mean and variance sequences.

For a given time \( k \), the maximum of the autocorrelation function is found at \( i = 0 \):

\[
R_{p,k,0} = \Sigma_{p,k}
\]

That is, each random variable is most correlated with itself. However, as \( i \) increases, \( R_{p,k,i} \) will decrease, and approach zero as \( i \) approaches infinity. In other words, as the time difference between two random variables increases the correlation between the two will decrease to the limit of no correlation for very large time differences.

In the sequel it will be important to consider the limiting case of absolutely no correlation between any of the random variables, regardless of their time difference. In terms of autocorrelation, this type of stochastic process is described as:

\[
R_{p,k,i} = \begin{cases} 
\Sigma_k & \text{if } i = 0 \\
0 & \text{if } i \neq 0 
\end{cases}
\]  

(5.17)

This class of processes is denoted as **white noise**. The concept of white noise holds a special place in stochastic analysis, in the sense that such a process can serve as a blank canvas from which other stochastic processes can be generated. Any process that does not satisfy (5.17) is frequently denoted as colored noise. The origin of the white vs. colored noise terminology stems the spectral interpretation of stochastic processes (see Section 5.5) and is related to white light being composed of all light frequencies and colored light containing only a subset of frequencies.

If the statistics of a stochastic process do not change with time (i.e., \( \bar{p}_k = \bar{p}, \Sigma_{p,k} = \Sigma_p \) and \( R_{p,k,i} = R_{p,i} \) for all \( k \)), then the process is denoted as **stationary**. It is emphasized that
A stationary stochastic process is still composed of a sequence of random variables and will take random realizations over time. The only distinction is that the statistics of each random variable do not change with time. Also, note that the autocorrelation function of a stationary process is still a function of the time difference $i$. The distinction is that the autocorrelation function will be the same for all times $k$. If a stochastic process is both white noise and stationary, then it is common to denote as independent and identically distributed (i.i.d.). The idea being that the random variables of a white process are all independent of each other and all have exactly the same probability density functions (or distributions).

**Example 5.4.** Consider a stationary, Gaussian, white noise process, $p_k$, with a mean $\bar{p}$ and variance $\Sigma_p$. Assume $\bar{p} = 30$, $\Sigma_p = (1.5)^2$ and the sample time is 0.1 minute. A realization of this sequence is provided in the top plot of Figure 5.3. Although the mean and standard deviation characteristics of this process are similar to the disturbance of Example 5.3, the correlation characteristics do not match well. Specifically, the correlation between the samples is very low. Another interpretation is that the disturbance signal contains frequencies that are too high.

![Figure 5.3. Stochastic sequences for Example 5.4. Top plot – white noise; middle and bottom – colored noise with 2 and 20 minute correlation times.](image-url)
Now consider the colored noise processes depicted in the middle and bottom plots of Figure 5.3. Clearly, both have a mean and variance equal to the previous ($\bar{p} = 30$ and $\Sigma_p = (1.5)^2$). However, the observed correlation between samples at various time differences (the autocorrelation) is distinct for the two processes. Specifically, the characteristic correlation time of the middle plot has been specified to be 2 minutes, while that of the bottom was set to 20 minutes. Looking at the sequence, one should see that neighbors within this time period will have realizations of similar value, while those outside of this characteristic time period are virtually uncorrelated.

The autocorrelation functions of our three stochastic processes are depicted in Figure 5.4. The first point to notice is that all three have the same value of $(1.5)^2$ at $i = 0$, since $R_{p,0} = \Sigma_p = (1.5)^2$. For the white noise case, $R_{p,i} = 0$ for all values of $i$ other than zero. In the colored noise cases, the characteristic correlation time ($\tau_c = 2$ and 20 minutes) is the time difference required for the autocorrelation function drop by 63%. 

Figure 5.4. Autocorrelation functions for Example 5.4. Bottom plot is a “zoom-in” of top plot.
It should be noted that (at this point) the reader is not expected to be able to generate the figures of this example. The intent is for the reader to simply appreciate the relation between a stochastic realization (Figure 5.3) and its autocorrelation function (Figure 5.4). However, by the end of this chapter it is expected that the reader will be able to generate the plots of this example.

**Figure 5.5.** Stochastic process used as disturbance input for process analysis

### 5.2.2 Impact of Autocorrelation

In many cases a stochastic process is used to represent the disturbance acting on a physical process, as indicated in Figure 5.5. Following the example of Section 1.3, the natural next question is how will the autocorrelation statistics of a disturbance impact the outputs of the physical process?

**Figure 5.6.** Disturbance inputs for Example 5.5
5.2. Introduction to Stochastic Sequences

Example 5.5. Reconsider the discrete-time representation of the mass spring damper process of Example 2.12¹.

\[ s_{k+1} = A_d s_k + G_d p_k \]
\[ q_k = D_x s_k + D_w p_k \]  

Using sample-and-hold with \( \Delta t = 0.2s \) the discrete-time system matrices are calculated to be:

\[ A_d = \begin{bmatrix} 0.95 & 0.16 \\ -0.48 & 0.62 \end{bmatrix}, \quad G_d = \begin{bmatrix} 0.017 \\ 0.16 \end{bmatrix} \]
\[ D_x = \begin{bmatrix} 1 & 0 \\ -3 & -2 \end{bmatrix}, \quad D_w = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]  

Next, assume the disturbance input, \( p_k \), is a stationary stochastic process with a mean of zero and a standard deviation of 0.5. All three disturbances of Figure 5.6 clearly exhibit these statistics — the average of each realization is about zero and about 95% of the realization is in the range of ±1 (two standard deviations from the mean). However, it is also clear that each has a different amount of autocorrelation. Disturbance A has very little autocorrelation and appears to be close to white noise. Disturbance B has a correlation time on the order of 2-3 seconds, while C has a correlation time of about 10 seconds.  

Application of the three disturbances of Figure 5.6 to the mass spring damper system (with an initial condition of \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \)), results in the time-series plots Figure 5.7. It is

¹In example 2.12, the process model was in deviation variable form. However, to be consistent with current development, the \((s, p, q)\) notation will be used in this example. To be consistent with the example 2.12, simply assume \( g = 0 \). A discussion of deviation variables for stochastic processes is given in the next section.
quickly observed that the variance of the acceleration output is strongly influenced by the autocorrelation of the disturbance — large for case A and progressively smaller for cases B and C. Based on the time-series plots of Figure 5.7, it is a bit difficult to observe the differences in the mass position output. This is in part due to the fairly short duration of the plots, only 30 seconds. In the scatter plots of Figure 5.8, the simulation data is given in the phase-plane and as such a longer simulation period can be depicted. From these plots it is clear that variance of the mass position output will increase significantly from case A to B and increase just slightly from case B to C. The region defined by the scatter plot will be denoted as the Expected Dynamic Operating Region (EDOR), see Section 5.3.5 for additional details. In summary, it is clear that the EDOR size and shape depends on the amount of autocorrelation contained within the disturbance.

Figure 5.8. Scatter plot responses for Example 5.5 (top left — no disturbance; top right—disturbance A; bottom left — disturbance B; bottom right — disturbance C)

Figure 5.9. Shaping Filter Approach to Disturbance Modeling
5.2.3 Disturbance Modeling via a Shaping Filter

An important point of Example 5.5 is that the output signals (mass acceleration and mass position) can be considered as stochastic processes in their own right. In this case, a re-evaluation of mass position plots of Figure 5.7 would indicate that the correlation time of case C is larger than that of cases A and B. It should also be noted that the stochastic characteristics of the output signal can be changed by changing the parameters of the physical process. For example, if one of the parameters of the matrix $A_d$ were changed, then one would expect each output signal to change.

The above observation leads us to a procedure for the modeling of disturbance signals as stochastic processes. Specifically, select the parameters of a dynamic system such that the output signal (a stochastic process) will have the same characteristics as the disturbance of interest. However, the first issue associated with this procedure is to select the stochastic process to be used as the input to the system. As suggested previously, the white noise process is special in the sense it can serve as a blank canvas from which other colored noise processes can be generated. The idea is that white noise contains all frequencies, from which a dynamic process can be used to filter out all undesired frequencies and retain only those (colors) that one would expect to find in the disturbance of interest. As such, the dynamic process used in such a procedure is typically denoted a shaping filter. The procedure is summarized in Figure 5.9. In most cases, the shaping filter is described by a linear state-space model:

$$s(f)_{k+1} = A_d s(f)_k + G_d p(f)_k$$
$$q(f)_k = D_x s(f)_k + D_w p(f)_k$$

where the super-script $(f)$ indicates that the matrices and resulting signals are associated with the shaping filter. Of course, the input signal $p(f)_k$ should be selected to be white noise.

**Table 5.1.** MATLAB code used to generate plots Figure 5.6.

```matlab
clear
% Define Shaping Filter
Adf=0.8; Gdf=1; Dxf=0.3; Dwf=0; pfbar=0; Sig_pf=1;
% Simulate Stochastic Process
randn('state',2^6-1) dt=0.2; NN=100/dt;
tt=zeros(1,NN); sf=zeros(1,NN); qf=zeros(1,NN); sf0=[0]; sf(:,1)=sf0;
for kk=1:NN-1
    tt(kk+1)=dt*kk;
    pfk=sqrt(Sig_pf)*randn+pfbar;
    sf(:,kk+1)=Adf*sf(:,kk)+Gdf*pfk;
    qf(:,kk)=Dxf*sf(:,kk)+Dwf*pfk;
end
qf(:,NN)=qf(:,NN-1);
pk=qf;
plot(tt,pk)
xlabel('Time (seconds)', 'FontSize',18); ylabel('p_{k}', 'FontSize',18)
```

**Example 5.6.** Consider the following scalar shaping filter:

$$s(f)_{k+1} = a_d s(f)_k + p(f)_k$$
$$q(f)_k = d_x s(f)_k$$

where $p(f)_k$ is a zero mean white noise process with unit variance. By appropriately selecting the parameters of this filter, the three disturbances of Figure 5.6 can be generat-
ed. Note that $p_k$ of the plots are set equal to $q_k^{(f)}$ (the output of the shaping filter) and should not be confused with $p_k^{(f)}$ (the white noise input to the shaping filter). To be more specific, if $a_d^{(f)} = 0$ and $d_x^{(f)} = 0.5$, then disturbance A will be generated. These parameters also indicate that disturbance A is in fact white noise, since the output is just: $q_k^{(f)} = d_x^{(f)}q_k^{(f)} = d_x^{(f)}p_{k-1}^{(f)}$. For case B, the parameters are set as follows: $a_d^{(f)} = 0.8$ and $d_x^{(f)} = 0.3$ and for case C they are $a_d^{(f)} = 0.95$ and $d_x^{(f)} = 0.156$. The details of how these parameters are selected such that $q_k^{(f)}$ is always zero mean with a standard deviation of 0.5 will be given in Section 5.6. As indicated by the code of Table 5.1, the generation of a white noise process in discrete-time is achieved by simply drawing random numbers from a random number generator. It should also be added that the stochastic processes of Figure 5.5 are also Gaussian, since the shaping filter is linear and the white noise is also Gaussian (i.e., from the MATLAB function ‘randn’).

![Diagram](image)

**Figure 5.10.** Compound system that combines the physical model with the disturbance model

### 5.2.4 - Combining the Disturbance and Process Models

In many cases, it will be required to combine the disturbance model with the model of the physical process to arrive at a single compound process model. The main advantage of this compound system is that it will be driven by white noise, even if the physical process is driven by colored noise. As we will see in the next section, this assumption of the process being driven by white noise is a prerequisite of the analysis. However, based on the above shaping filter discussion it is clear that such an assumption can always be met.

Assume the physical process is characterized by a linear state-space model:

$$
\begin{align*}
  s_{k+1}^{(p)} &= A_d^{(p)} s_k^{(p)} + B_d^{(p)} m_k^{(p)} + G_d^{(p)} P_k^{(p)} \\
  q_k^{(p)} &= D_s^{(p)} s_k^{(p)} + D_u^{(p)} m_k^{(p)} + D_w^{(p)} P_k^{(p)}
\end{align*}
$$

(5.26) (5.27)

where the super-script $(p)$ indicates that the matrices and resulting signals are associated with the physical process. If we now assume the disturbance into this physical process is equal to the output of the shaping filter ($p_k^{(f)} = q_k^{(f)}$), then the two systems can be combined (as in Figure 5.10) to arrive at the following compound system:

$$
\begin{bmatrix}
  s_{k+1}^{(p)} \\
  s_k^{(f)}
\end{bmatrix} =
\begin{bmatrix}
  A_d^{(p)} & G_d^{(p)} D_s^{(f)} \\
  0 & A_d^{(f)}
\end{bmatrix}
\begin{bmatrix}
  s_k^{(p)} \\
  s_k^{(f)}
\end{bmatrix} +
\begin{bmatrix}
  B_d^{(p)} \hfill 0 \\
  B_d^{(f)} \hfill 0
\end{bmatrix}
\begin{bmatrix}
  m_k^{(p)} \\
  m_k^{(f)}
\end{bmatrix} +
\begin{bmatrix}
  G_d^{(p)} D_s^{(f)} \hfill G_d^{(f)} D_u^{(f)} \hfill G_d^{(f)} D_w^{(f)}
\end{bmatrix}
\begin{bmatrix}
  P_k^{(p)} \\
  P_k^{(f)}
\end{bmatrix}
$$

(5.28)
\[
q_k^{(p)} = \begin{bmatrix} D_{x}^{(p)} D_{w}^{(f)} x_k^{(p)} \\ D_{w}^{(f)} x_k^{(p)} \end{bmatrix} + \begin{bmatrix} D_{u}^{(p)} m_k^{(p)} + [D_{w}^{(p)} D_{w}^{(f)}] p_k^{(f)} \end{bmatrix} \tag{5.29}
\]

Then, we will frequently express this compound system compactly using our previous (or more common) notation:

\[
s_{k+1} = A_d s_k + B_d m_k + G_d p_k \\
q_k = D_x s_k + D_u m_k + D_w p_k
\tag{5.30}
\]

\[
q_k = D_x s_k + D_u m_k + D_w p_k
\tag{5.31}
\]

where

\[
s_k = \begin{bmatrix} s_k^{(p)} \\ s_k^{(f)} \end{bmatrix}, m_k = m_k^{(p)} p_k = p_k^{(f)} q_k = q_k^{(p)}
\tag{5.32}
\]

\[
A_d = \begin{bmatrix} A_{d}^{(p)} & G_{d}^{(p)} D_{x}^{(f)} \\ 0 & A_{d}^{(f)} \end{bmatrix}, \\
B_d = \begin{bmatrix} B_{d}^{(p)} \\ 0 \end{bmatrix}, G_d = \begin{bmatrix} G_{d}^{(f)} D_{w}^{(f)} \\ G_{d}^{(f)} \end{bmatrix}
\tag{5.33}
\]

\[
D_x = \begin{bmatrix} D_{x}^{(p)} D_{w}^{(f)} x_k^{(p)} \\ D_{w}^{(f)} x_k^{(p)} \end{bmatrix}, D_u = \begin{bmatrix} D_{u}^{(p)} \end{bmatrix}, D_w = \begin{bmatrix} D_{w}^{(p)} D_{w}^{(f)} \end{bmatrix}
\tag{5.34}
\]

Table 5.2. MATLAB code used to generate simulations of Example 5.5.

```
clear
% Mass-Spring-Damper Model in Continuous-time
Ap=[0 1; -3 -2]; Bp=[0; 0]; Gp=[0; 1];
Dxp=[1 0; -3 -2]; Dup=[0; 0]; Dwp=[0; 1];

% Conversion to Discrete-time
dt=0.2; Ndt=200; ddt=dt/Ndt; sum=[0 0; 0 0];
for jjj=1:Ndt sum=sum+expm(Ap *jjj*ddt);
end
Adp=expm(Ap*dt); Bdp=sum *Bp*ddt; Gdp=sum *Gp*ddt;

% Define Shaping Filter
Adf=0.8; Gdf=1; Dxf=0.3; Dwf=0; pfbar=0; Sig_pf=1;

% Compound System
Ad=[Adp Gdp *Dxf; 0 0 Adf]; Bd=[Bdp; 0]; Gd=[Gdp *Dwf; Gdf];
Dx=[Dxp Dwp *Dxf]; Du=Dup; Dw=Dwp *Dwf;

% Simulate Compound Process
randn('state',2^6-1); dt=0.2; NN=100/dt;
tt=zeros(1,NN); ss=zeros(3,NN); qq=zeros(2,NN);
ss0=[1; 0; 0]; ss(:,1)=ss0;
for kk=1:NN-1
tt(kk+1)=dt*kk;
pk=sqrt(Sig_pf)*randn+pfbar;
ss(:,kk+1)=Ad*ss(:,kk)+Gd*pk;
qq(:,kk)=Dx*ss(:,kk)+Dw*pk;
end
qq(:,NN)=qq(:,NN-1);
plot(tt,qq(1,:),'-k')
plot(tt,qq(2,:),'-ko'); axis([0 30 -3 2])
legend('Mass Position', 'Mass Acceleration', 'Location', 'SouthEast')
xlabel('Time (seconds)'); ylabel('Performance Outputs (m or m/s^2)')
```

```
Example 5.7. Consider the process model of Example 5.5. Furthermore consider the case B shaping filter of Example 5.6. Using these two models the following compound system will result:

\[
\begin{bmatrix}
  s_k^{(p)} \\
  s_k^{(f)}
\end{bmatrix}
+ \begin{bmatrix}
  0.9478 & 0.1616 & 0.0052 \\
  -0.4847 & 0.6246 & 0.8
\end{bmatrix}
\begin{bmatrix}
  s_k^{(p)} \\
  s_k^{(f)}
\end{bmatrix}
+ \begin{bmatrix}
  0 \\
  0
\end{bmatrix}
\begin{bmatrix}
  m_k^{(p)} \\
  p_k^{(f)}
\end{bmatrix}
= \begin{bmatrix}
  0.9478 & 0.1616 & 0.0052 \\
  -0.4847 & 0.6246 & 0.8
\end{bmatrix}
\begin{bmatrix}
  0 \\
  0
\end{bmatrix}
\begin{bmatrix}
  m_k^{(p)} \\
  p_k^{(f)}
\end{bmatrix}
\]

(5.35)

\[
q_k^{(p)} = \begin{bmatrix}
  1 & 0 & 0 \\
  -3 & -2 & 0.3
\end{bmatrix}
\begin{bmatrix}
  s_k^{(p)} \\
  s_k^{(f)}
\end{bmatrix}
+ \begin{bmatrix}
  0 \\
  0
\end{bmatrix}
\begin{bmatrix}
  m_k^{(p)} \\
  p_k^{(f)}
\end{bmatrix}
\]

(5.36)

Given this compound system, along with the fact that \( p_k^{(f)} \) is white noise, it is fairly simple to assemble the MATLAB code of Table 5.2, which was used to generate the plots of Figures 5.7 and 5.8.

The next two sections will focus on calculating the mean, variance and autocorrelation of a process generated by a white noise excited linear system. Given these characterization methods, the subject of disturbance design can be addressed. Specifically, how does one select a linear system such that the resulting output has properties similar to those of the disturbance input?

5.3 Analysis of Stochastic Processes

As illustrated in Example 5.7, one can use simulation to determine the statistical characteristics of an output signal. However, in the subsequent controller design methods it will be important to obtain analytic expressions that allow one to calculate the output characteristics. To illustrate the development the following scalar discrete-time process will be considered:

\[
s_{k+1} = a_ds_k + g_dp_k
\]

(5.37)

\[
q_k = d_s s_k + d_w p_k
\]

(5.38)

5.3.1 Process Mean and Deviation Variables

Define the mean of \( s_k, p_k \) and \( q_k \) as: \( \bar{s}_k = E[s_k], \bar{p}_k = E[p_k] \) and \( \bar{q}_k = E[q_k] \). Then application of the expectation operator to (5.37)-(5.38), yields:

\[
\bar{s}_{k+1} = a_d \bar{s}_k + g_d \bar{p}_k
\]

(5.39)

\[
\bar{q}_k = d_s \bar{s}_k + d_w \bar{p}_k
\]

(5.40)

where \( \bar{s}_0 = E[s_0] \) is the expected value of the initial condition, \( s_0 \). The conclusion being that the average of the process outputs respond in a way that is identical to the deterministic process. To isolate deviations from the average signals, it is common to define the following deviation variables with respect to the average signals:

\[
\tilde{p}_k = p_k - \bar{p}_k, \quad \tilde{s}_k = s_k - \bar{s}_k \quad \text{and} \quad \tilde{q}_k = q_k - \bar{q}_k
\]

(5.41)

If these definitions are substituted into (5.37)-(5.38) and the identities (5.39)-(5.40) are employed, then the following deviation variable model is determined:

\[
\tilde{s}_{k+1} = a_d \tilde{s}_k + g_d \tilde{p}_k
\]

(5.42)

\[
\tilde{q}_k = d_s \tilde{s}_k + d_w \tilde{p}_k
\]

(5.43)
If $p_k$ is a stationary process, then $\bar{p}_k$ will be a constant for all $k$ (denoted $\overline{p}$). If in addition the process is stable (i.e., $|a_d| < 1$), then $\bar{s}_k$ and $\bar{q}_k$ will eventually arrive at the SSOP defined by:

$$s^{SSOP} = a_d s^{SSOP} + g_d p^{SSOP}$$

$$q^{SSOP} = d_x s^{SSOP} + d_w p^{SSOP}$$

where $p^{SSOP} = \overline{p}$. Deviation variables with respect to average sequences ($\bar{p}_k = p_k - \overline{p}_k$, $\bar{s}_k = s_k - \bar{s}_k$ and $\bar{q}_k = q_k - \bar{q}_k$) should not be confused with deviation variables with respect to the SSOP:

$$\tilde{w}_k = p_k - p^{SSOP}, \quad \tilde{x}_k = s_k - s^{SSOP} \quad \text{and} \quad \tilde{z}_k = q_k - q^{SSOP}$$

which yields the familiar process model of Chapters 2 and 4:

$$\tilde{x}_{k+1} = a_d x_k + g_d \tilde{w}_k$$

$$\tilde{z}_k = d_x \tilde{x}_k + d_w \tilde{w}_k$$

A stochastic system in deviation variables with respect to the SSOP, can also be put into deviation variable form with respect to its average sequence. First, define $\tilde{w}_k = E[\omega_k]$. If $p_k$ is stationary, then $\overline{w}_k = E[\bar{\omega}_k] = 0$ for all $k$. Similarly, define the mean of $x_k$ and $z_k$ as: $\bar{x}_k = E[x_k]$ and $\bar{z}_k = E[z_k]$, and apply the expectation operator to (5.47)-(5.48) to arrive at:

$$\bar{\tilde{x}}_{k+1} = a_d \bar{x}_k$$

$$\bar{\tilde{z}}_k = d_x \bar{x}_k$$

where of course $\bar{x}_0 = E[x_0] = E[s_0 - s^{SSOP}]$.

Next, one can again define deviation variables with respect to the average sequence:

$$\tilde{w}_k = w_k, \quad \tilde{z}_k = x_k - \bar{x}_k \quad \text{and} \quad \tilde{z}_k = z_k - \bar{z}_k$$

If substituted into (5.47)-(5.48), the following deviation of the deviation variables form is found:

$$\bar{\tilde{x}}_{k+1} = a_d \bar{x}_k + g_d \bar{\tilde{w}}_k$$

$$\bar{\tilde{z}}_k = d_x \bar{x}_k + d_w \bar{\tilde{w}}_k$$

It should be highlighted that if $E[s_0] = s^{SSOP}$, then $\tilde{x}_0 = 0$ and from Equation (5.49) $\bar{\tilde{x}}_k = 0$ for all $k$. In this case, (5.52)-(5.53), (5.47)-(5.48) and (5.42)-(5.43) will be identical (i.e., $\tilde{x}_k = \bar{x}_k = x_k$ and $\tilde{z}_k = \bar{z}_k = z_k$ for all $k$).

**Example 5.8.** Consider the following discrete-time process:

$$s_{k+1} = 0.75s_k + p_k$$

where $p_k$ is stationary white noise with $\overline{p} = 2.5$ and $\Sigma_p = 0.25$. Assume the sample time is 0.2 seconds. If the initial condition, $s_0$, is known to be 15, then the expected value of the initial condition should be set to 15. A realization of the stochastic process is given in Figure 5.11, along with the process average. This average sequence, $\bar{s}_k$, quickly approaches the SSOP, which from (5.44) is known to be $s^{SSOP} = 10$. The second plot of Figure 5.11, shows the same realization, but from the deviation variable (with respect to the mean) perspective.
Example 5.9. Reconsider Example 5.8. The simulations of Example 5.8 are repeated, but in the context of deviation variables with respect to the SSOP (i.e., using Equation (5-47)). In the top plot of Figure 5.12, the sequences are simply shifted by a distance equal to the $s^{SSOP} = 10$. The bottom plot of the figure is identical to that of Figure 5.11, which
5.3. Analysis of Stochastic Processes

The Variance Sequence

The point being that both \( \tilde{s}_0 \) and \( \tilde{q}_0 \) can be calculated via the following recursive relations:

\[
\tilde{s}_k = \tilde{x}_k, \quad \tilde{q}_k = \tilde{q}_k \quad \text{and} \quad \tilde{w}_k = \tilde{p}_k
\]  

(5.55)

In fact, if the expected value of the initial condition, \( \tilde{s}_0 = E[\tilde{s}_0] \), is equal to \( s^{SSOP} \), then the following holds:

\[
x_k = \tilde{s}_k \quad \text{and} \quad \tilde{x}_k = \tilde{q}_k
\]  

(5.56)

Assuming \( \tilde{s}_0 = s^{SSOP} = 10 \) (and not 15) would cause both plots of Figure 5.12 to be the same as the bottom plot of Figure 5.11.

5.3.2 The Variance Sequence

The following development assumes the disturbance sequence, \( p_k \), is white noise. Recall that this assumption was justified by the compound plus shaping filter form discussed in Section 5.2.4. Such assumption implies that \( \tilde{p}_k, \tilde{w}_k, \) and \( \tilde{w}_k \) are also white noise processes. Begin by defining the variance of each signal as:

\[
\Sigma_{x,k} = E[(s_k - \tilde{s}_k)^2] = E[(x_k - \tilde{x}_k)^2] = E[\tilde{x}_k^2]
\]  

(5.57)

\[
\Sigma_{z,k} = E[(q_k - \tilde{q}_k)^2] = E[(\tilde{x}_k - \tilde{z}_k)^2] = E[\tilde{z}_k^2]
\]  

(5.58)

\[
\Sigma_{w,k} = E[(p_k - \tilde{p}_k)^2] = E[(\tilde{w}_k - \tilde{w}_k)^2] = E[\tilde{w}_k^2]
\]  

(5.59)

Next calculate \( \Sigma_{x,k} \), assuming \( \Sigma_{x,k} \) is given:

\[
\Sigma_{x,k} = E[\tilde{x}_k^2] = E[(d_x \tilde{x}_k + d_w \tilde{w}_k)^2]
\]  

\[
= d_x^2 E[\tilde{x}_k^2] + 2 d_x d_w E[\tilde{x}_k \tilde{w}_k] + d_w^2 E[\tilde{w}_k^2]
\]  

(5.60)

where the cross term, \( E[\tilde{x}_k \tilde{w}_k] \), is zero due to the assumption of \( p_k \) being a white noise process. More specifically, the independence of each \( \tilde{w}_k \) with respect to all other \( \tilde{w}_k \)'s along with Equation (5.52) indicates that \( \tilde{w}_k \) is independent of \( \tilde{x}_i \) for all \( k \geq i \). Similarly, \( \Sigma_{z,k+1} \) is calculated as:

\[
\Sigma_{z,k+1} = E[\tilde{z}_k^2] = E[(a_d \tilde{x}_k + g_d \tilde{w}_k)^2]
\]  

\[
= a_d^2 E[\tilde{x}_k^2] + 2 a_d g_d E[\tilde{x}_k \tilde{w}_k] + g_d^2 E[\tilde{w}_k^2]
\]  

(5.61)

In summary, if \( \Sigma_{x,0} \) and \( \Sigma_{w,k} \) are both given, then all future values of \( \Sigma_{x,k} \) and \( \Sigma_{z,k} \) can be calculated via the following recursive relations:

\[
\Sigma_{x,k+1} = a_d^2 \Sigma_{x,k} + g_d^2 \Sigma_{w,k}
\]  

(5.62)

\[
\Sigma_{z,k} = d_x^2 \Sigma_{x,k} + d_w^2 \Sigma_{w,k}
\]  

(5.63)

The point being that both \( \Sigma_{x,k} \) and \( \Sigma_{z,k} \) will evolve over time.

If \( p_k \) is assumed stationary, then \( \Sigma_{w,k} \) will be constant with respect to \( k \) (denoted \( \Sigma_{w} \)). Furthermore, if the system is stable \((|a_d| < 1)\), then one expects \( \Sigma_{x,k} \) and \( \Sigma_{z,k} \) to arrive at some steady-state condition. In this case, the steady-state variances are defined
as $\Sigma_x \overset{\Delta}{=} \lim_{k \to \infty} \Sigma_{x,k}$ and $\Sigma_z \overset{\Delta}{=} \lim_{k \to \infty} \Sigma_{z,k}$, where the values of each can be identified from the solutions to the following algebraic relations:

$$
\Sigma_x = a_d^2 \Sigma_x + g_d^2 \Sigma_w \quad (5.64)
$$

$$
\Sigma_z = d_x^2 \Sigma_x + d_w^2 \Sigma_w \quad (5.65)
$$

Equation (5.62) is frequently denoted as the discrete-time variance equation while (5.64) is the steady-state version of the discrete-time variance equation.

Example 5.10. Reconsider Example 5.8. In that example, it was assumed that the initial condition, $s_0$, was known to be equal to 15. This assumption of perfect knowledge of the initial condition suggests that the initial condition to the recursive relation (5.62) should be $\Sigma_{x,0} = 0$. That is, the variance of the initial condition is zero. Recall that the interpretation of $\Sigma_{x,k}$ is that it is the mean removed variance of $s_k$ at time $k$. Thus, the square root of $\Sigma_{x,k}$ will be the standard deviation of $s_k$ at time $k$. In Figure 5.13, the 2 standard deviation envelope is applied to the original simulations of Example 5.8. Notice how the envelope starts at zero and then grows to eventually settle at some steady-state. The steady-state variance could also be calculated from Equation (5.64). In this example the steady-state variance turns out to be $\Sigma_x = 0.5714$, giving a standard deviation of 0.7559, which corresponds to the 2 standard deviation envelope observed in the plots of Figure 5.13.

If one has less certainty with respect to the initial condition, then it is inappropriate to set the initial condition of (5.62) to zero. In the plots of Figure 5.14, $\Sigma_{x,0}$ was set equal to 9, and the initial condition of the state was selected by the random number generator based on this initial variance (which turned out in this realization to be 19.5028 (≈ 4.5028...
5.3. Analysis of Stochastic Processes

Figure 5.14. Stochastic process of Example 5.10 with unknown initial condition, natural variables (top) and deviation (bottom). Both augmented by a 2 standard deviation envelope.

+ 15). In this case, the two standard deviation envelope starts out large, but then decreases to the same steady-state observed in Figure 5.13.

5.3.3 - Autocorrelation and Variance Settling Time

The definition of the autocorrelation function is $R_{x,k,i} = E[\tilde{x}_k \tilde{x}_{k+i}]$. Let us first consider the case of $i = 1$.

$$R_{x,k,1} = E[\tilde{x}_k \tilde{x}_{k+1}] = E[\tilde{x}_k (a_d \tilde{x}_k + g_d \tilde{w}_k)]$$
$$= a_d E[\tilde{x}_k \tilde{x}_k] + g_d E[\tilde{x}_k \tilde{w}_k]$$
$$= a_d \Sigma_{x,k}$$  \hspace{1cm} (5.66)

Using similar calculations for other values of $i$, one finds:

$$R_{x,k,i} = a_d^{|i|} \Sigma_{x,k}$$  \hspace{1cm} (5.67)

If $\tilde{w}_k$ is stationary and $a_d$ is stable, then in the limit one finds:

$$R_{x,i} = \lim_{k \to \infty} \{R_{x,k,i}\} = a_d^{|i|} \Sigma_x$$  \hspace{1cm} (5.68)

Clearly, this function has properties identical to those described in Section 5.2. That is, as $|i|$ increases $R_{x,i}$ will decrease to zero. The parameter $a_d$ can now be used to quantify the amount of autocorrelation expected within the stochastic process. Specifically, assume $|a_d| < 1$, then as $|a_d|$ gets closer to 1, the amount of autocorrelation will increase. This is
reflected in the autocorrelation function, in that $R_{x,i}$ will decrease more slowly as $|a_d|$ gets closer to 1. Such property was observed in Example 5.6. Specifically, Disturbance C Figure 5.6 had high autocorrelation and an $a_d$ value of 0.95, while Disturbance B had lower autocorrelation and an $a_d$ value of 0.8. Finally, the white noise process (Disturbance A) had an $a_d$ value of zero.

![Figure 5.15](image)

**Figure 5.15.** Variance sequences for different values of $a_d$.

**Example 5.11.** Consider the following discrete-time process with a sample-time of 0.2 seconds:

$$x_{k+1} = a_d x_k + g_d w_k$$  \hspace{1cm} (5.69)

where $g_d = (1 - a_d)$, $x_0$ is known to be zero and $w_k$ is stationary white noise with zero mean. Now define the variance of $w_k$ as:

$$\Sigma_w = \left(1 - a_d^2\right) \Sigma_{(s_p)}$$  \hspace{1cm} (5.70)

Then, application of Equation (5.64) yields the following convent result:

$$\Sigma_x = \frac{g_d^2}{1 - a_d^2} \Sigma_w = \frac{g_d^2}{1 - a_d^2} \left(\frac{1 - a_d^2}{1 - a_d^2} \Sigma_{(s_p)}\right) = \Sigma_{(s_p)}$$  \hspace{1cm} (5.71)

That is, the steady-state variance of $x_k$ will be equal to $\Sigma_{(s_p)}$, regardless of the $a_d$ value selected. (In the plots to follow $\Sigma_{(s_p)}$ is set to 0.25). The next question concerns the path $\Sigma_{x,k}$ takes to arrive at its steady-state value. Figure 5.15, indicates the relationship between $a_d$ and the rate of convergence of (5.62). Specifically, as $a_d$ gets closer to 1, the recursion of (5.62) will converge ever more slowly.

The plots of Figure 5.16 illustrate realizations for each of the three cases as well as the 2 standard deviation envelope corresponding to each. These realizations illustrate once again the relation between $a_d$ and the autocorrelation. Specifically, as $a_d$ gets closer to 1, the correlation between samples will increase. This is also observed in the autocorrelation function plots of Figure 5.17.
Figure 5.16. Stochastic processes of Example 5.11, augmented by a 2 standard deviation envelope. (top \( a_d = 0.75 \); middle \( a_d = 0.95 \); bottom \( a_d = 0.995 \)).

Figure 5.17. Autocorrelation functions for Example 5.11.
5.3.4 Vector Valued Stochastic Processes

The analysis of the previous three sub-sections assumed the stochastic process was a sequence of scalar random variables. We now turn to the more general case of a stochastic process defined by a sequence of vector valued random variables. Having used the scalar case to illustrate the derivation details, the following will simply provide the main results.

Consider a state-space process excited by a stochastic process $p_k$:

$$s_{k+1} = A_d s_k + G_d p_k$$
$$q_k = D_x s_k + D_w p_k$$

Assume $p_k$ is a stationary process with mean $\bar{p} \equiv p^{SSOP}$. Then, one can define deviation variables with respect to the SSOP as

$$w_k = p_k - p^{SSOP}, \quad x_k = s_k - s^{SSOP}, \quad \text{and} \quad z_k = q_k - q^{SSOP}$$

where $s^{SSOP}$ and $q^{SSOP}$ are determined as solutions to:

$$s^{SSOP} = A_d s^{SSOP} + G_d \bar{p}$$
$$q^{SSOP} = D_x s^{SSOP} + D_w \bar{p}$$

The resulting process in SSOP deviation variables is:

$$x_{k+1} = A_d x_k + G_d w_k$$
$$z_k = D_x x_k + D_w w_k$$

Then, the stationary assumption on $p_k$, implies $\bar{w}_k = E[w_k] = 0$ for all $k$. Thus, $\bar{x}_k = E[x_k]$ and $\bar{z}_k = E[z_k]$ will be governed by the following relations:

$$\bar{x}_{k+1} = A_d \bar{x}_k$$
$$\bar{z}_k = D_x \bar{x}_k$$

where $\bar{x}_0 = E[x_0]$. In the following development we will assume $E[s_0] = s^{SSOP}$, which makes $\bar{x}_k = 0$ for all $k$. The net result being that all deviation variables will be same: $\bar{s}_k = \bar{x}_k = \bar{z}_k$ and $\bar{q}_k = \bar{z}_k = \bar{z}_k$, recall Example 5.9. If this assumption is removed, then all $x_k$ and $z_k$ terms would need to be replaced with $\bar{x}_k$ and $\bar{z}_k$. However, in either case the main results, Equations (5.91)-(5.94), will remain unchanged.

We begin the following definitions:

$$\Sigma_{x,k} = E[x_k x_k^*], \quad \Sigma_{z,k} = E[z_k z_k^*], \quad \text{and} \quad \Sigma_{w,k} = E[w_k w_k^*]$$

In this more general case, the each element of the sequence $\Sigma_{x,k}$ will be denoted as the state covariance matrix at time $k$. The covariance terminology is used to emphasize that the matrix contains not only the variance of each state variable but also the cross covariance terms, which express the amount of correlation between each state variable. To be more specific let $x_k = [x_k^{(1)} x_k^{(2)} \ldots x_k^{(n_k)}]^*$, then $\Sigma_{x,k}$ will be of the following form:

$$\Sigma_{x,k} = E[x_k x_k^*] = \begin{bmatrix}
E[x_k^{(1)} x_k^{(1)}] & E[x_k^{(1)} x_k^{(2)}] & \cdots & E[x_k^{(1)} x_k^{(n_k)}] \\
E[x_k^{(2)} x_k^{(1)}] & E[x_k^{(2)} x_k^{(2)}] & \cdots & E[x_k^{(2)} x_k^{(n_k)}] \\
\vdots & \vdots & \ddots & \vdots \\
E[x_k^{(n_k)} x_k^{(1)}] & E[x_k^{(n_k)} x_k^{(2)}] & \cdots & E[x_k^{(n_k)} x_k^{(n_k)}]
\end{bmatrix}$$

(5.82)
5.3. Analysis of Stochastic Processes

The disturbance and output covariance matrices, $\Sigma_{w,k}$ and $\Sigma_{z,k}$, are defined similarly:

$$\Sigma_{w,k} = E[w_k w_k^*] = \begin{bmatrix}
E[w_k^{(1)} w_k^{(1)}] & E[w_k^{(1)} w_k^{(2)}] & \cdots & E[w_k^{(1)} w_k^{(n_w)}] \\
E[w_k^{(2)} w_k^{(1)}] & E[w_k^{(2)} w_k^{(2)}] & \cdots & E[w_k^{(2)} w_k^{(n_w)}] \\
\vdots & \vdots & \ddots & \vdots \\
E[w_k^{(n_w)} w_k^{(1)}] & E[w_k^{(n_w)} w_k^{(2)}] & \cdots & E[w_k^{(n_w)} w_k^{(n_w)}]
\end{bmatrix}$$

(5.83)

$$\Sigma_{z,k} = E[z_k z_k^*] = \begin{bmatrix}
E[z_k^{(1)} z_k^{(1)}] & E[z_k^{(1)} z_k^{(2)}] & \cdots & E[z_k^{(1)} z_k^{(n_z)}] \\
E[z_k^{(2)} z_k^{(1)}] & E[z_k^{(2)} z_k^{(2)}] & \cdots & E[z_k^{(2)} z_k^{(n_z)}] \\
\vdots & \vdots & \ddots & \vdots \\
E[z_k^{(n_z)} z_k^{(1)}] & E[z_k^{(n_z)} z_k^{(2)}] & \cdots & E[z_k^{(n_z)} z_k^{(n_z)}]
\end{bmatrix}$$

(5.84)

In all three cases, it is clear that the diagonal elements of the covariance matrix represents the variance of the individual signals. In the sequel, it will be necessary to specify individual elements of the covariance matrices. To facilitate these specifications, the following notation will be employed. Let $\rho_j$, be the $j$th row of an appropriately dimensioned identity matrix:

$$I = \begin{bmatrix}
\rho_1 \\
\rho_2 \\
\vdots \\
\rho_n
\end{bmatrix}$$

(5.85)

or

$$\rho_j = [0 \ 0 \ \cdots \ 1 \ \cdots \ 0 \ 0]$$

(5.86)

Now, consider the $j$th performance output at time $k$, $z_k^{(j)}$. Using the above notation, the variance and standard deviation of this signal at time $k$ are defined as:

$$\zeta_k^{(j)} = \rho_j \Sigma_{z,k} \rho_j^* \quad \text{and} \quad \sigma_k^{(j)} = \sqrt{\zeta_k^{(j)}}$$

(5.87)

Similarly, the variance and standard deviation of the $j$th state variable are defined as:

$$\xi_k^{(j)} = \rho_j \Sigma_{x,k} \rho_j^* \quad \text{and} \quad \sigma_k^{(j)} = \sqrt{\xi_k^{(j)}}$$

(5.88)

Under the assumption of $w_k$ being a white noise process (i.e., $E[w_k w_{k+i}^*] = \Sigma_{w,k}$ if $i = 0$ and zero otherwise), $\Sigma_{x,k}$ is calculated as follows:

$$\Sigma_{x,k} = E[z_k z_k^*] = E[(D_x x_k + D_w w_k)(D_x x_k + D_w w_k)^*]$$

$$= D_x E[x_k x_k^*] D_x^* + D_x E[x_k w_k^*] D_w^* + D_w E[w_k x_k^*] D_x^* + D_w E[w_k w_k^*] D_w^*$$

$$= D_x \Sigma_{x,k} D_x^* + D_w \Sigma_{w,k} D_w^*$$

(5.89)
Similarly, $\Sigma_{x,k+1}$ is calculated as:

$$
\Sigma_{x,k+1} = E[x_{k+1}x_{k+1}^T] = E[(A_d x_k + G_d w_k)(A_d x_k + G_d w_k)^T] \\
= A_d E[x_k x_k^T]A_d^* + A_d E[x_k w_k^T]G_d^* + G_d E[w_k x_k^T]A_d^* + G_d E[w_k w_k^T]G_d^* \\
= A_d \Sigma_{x,k}A_d^* + G_d \Sigma_{w,k}G_d^* 
$$

(5.90)

Thus, if $\Sigma_{x,0}$ and $\Sigma_{w,k}$ are both given then all future values of $\Sigma_{x,k}$ and $\Sigma_{z,k}$ can be calculated via the following recursive relations:

$$
\Sigma_{x,k+1} = A_d \Sigma_{x,k}A_d^* + G_d \Sigma_{w,k}G_d^* \\
\Sigma_{z,k} = D_x \Sigma_{x,k}D_x^* + D_w \Sigma_{w,k}D_w^* 
$$

(5.91) (5.92)

If $w_k$ is additionally assumed stationary (i.e., $\Sigma_{w,k} = \Sigma_w$ for all $k$) and $A_d$ is stable, then the steady-state covariance matrices $\Sigma_x \triangleq \lim_{k \to \infty} \Sigma_{x,k}$ and $\Sigma_z \triangleq \lim_{k \to \infty} \Sigma_{z,k}$ will exist and can be determined as the solutions to the following algebraic relations:

$$
\Sigma_x = A_d \Sigma_x A_d^* + G_d \Sigma_w G_d^* \\
\Sigma_z = D_x \Sigma_x D_x^* + D_w \Sigma_w D_w^* 
$$

(5.93) (5.94)

Equation (5.91) is denoted as the discrete-time covariance equation while (5.93) is the steady-state version of the discrete-time covariance equation. The form of Equation (5.93) suggests that it is a member of a wider class: discrete-time Lyapunov equations. To solve such equations, the Control Systems Toolbox of MATLAB provides the function ‘dlyap’. To indicate steady-state conditions the following notation will be employed for steady-state variance and steady-state standard deviation:

$$
\zeta^{(j)} = \rho_j \Sigma_x \rho_j^* \quad \text{and} \quad \sigma^{(j)} = \sqrt{\zeta^{(j)}} \\
\xi^{(j)} = \rho_j \Sigma_z \rho_j^* \quad \text{and} \quad \sigma^{(j)} = \sqrt{\xi^{(j)}} 
$$

(5.95) (5.96)

Example 5.12. Reconsider the shaping filter augmented mass-spring-damper process of Example 5.7, where the shaping filter associated with Disturbance B is augmented to the physical process. In that case, the following compound system was determined:

$$
\begin{bmatrix}
    s^{(p)}_{k+1} \\
    s^{(f)}_{k+1}
\end{bmatrix}
= 
\begin{bmatrix}
    0.95 & 0.16 & 0.0051 \\
    -0.48 & 0.62 & 0.048 \\
    0 & 0 & 0.8
\end{bmatrix}
\begin{bmatrix}
    s^{(p)}_k \\
    s^{(f)}_k
\end{bmatrix}
+ 
\begin{bmatrix}
    0 \\
    0 \\
    0
\end{bmatrix}
\begin{bmatrix}
    m^{(p)}_k \\
    m^{(f)}_k
\end{bmatrix}
+ 
\begin{bmatrix}
    0 \\
    1
\end{bmatrix}
\begin{bmatrix}
    p^{(f)}_k
\end{bmatrix}
\quad (5.97)

q^{(p)}_k = 
\begin{bmatrix}
    1 & 0 & 0 \\
    -3 & -2 & 0.3
\end{bmatrix}
\begin{bmatrix}
    s^{(p)}_k \\
    s^{(f)}_k
\end{bmatrix}
+ 
\begin{bmatrix}
    0 \\
    0
\end{bmatrix}
\begin{bmatrix}
    m^{(p)}_k \\
    m^{(f)}_k
\end{bmatrix}
+ 
\begin{bmatrix}
    0 \\
    1
\end{bmatrix}
\begin{bmatrix}
    p^{(f)}_k
\end{bmatrix}
\quad (5.98)

Given this model the steady-state covariance matrices are determined to be

$$
\Sigma_x = 
\begin{bmatrix}
    0.0201 & 0.0001 & 0.1154 \\
    0.0001 & 0.0219 & 0.1256 \\
    0.1154 & 0.1256 & 2.778
\end{bmatrix}
\quad (5.99)

\Sigma_z = 
\begin{bmatrix}
    0.0201 & -0.026 \\
    -0.026 & 0.1618
\end{bmatrix}
\quad (5.100)
The MATLAB code of Table 5.3 illustrates how to calculate the steady-state covariance matrices of Equations (5.93)-(5.94) using the function ‘dlyap’ as well as through the recursive relations of Equations (5.91)-(5.92).

**Table 5.3. MATLAB code of Example 5.12.**

```matlab
% Mass-Spring-Damper Model in Continuous-time
Ap=[0 1; -3 -2]; Bp=[0; 0]; Gp=[0; 1];
Dxp=[1 0; -3 -2]; Dup=[0; 0]; Dw=[0; 1];
% Conversion to Discrete-time
dt=0.02; Ndt=200; ddt=dt/Ndt; sum=[0 0; 0 0];
for jj=1:Ndt sum=sum+expm(Ap *jj*ddt);
end
Adp=expm(Ap*dt); Bdp=sum *Bp*ddt; Gdp=sum *Gp*ddt;
% Define Shaping Filter
Adf=0.8; Gdf=1; Dxf=0.3; Dwf=0; pfbar=0; Sig_pf=1;
% Define Compound System
Ad=[Adp Gdp*Dxf; 0 0 Adf]; Bd=[Bdp; 0]; Gd=[Gdp*Dwf; Gdf];
Dx=[Dxp Dwp*Dxf]; Du=Dup; Dw=Dwp*Dwf; Sigw=Sig_pf
% Solve Covariance Equation with dlyap
Sigx=dlyap(Ad,Gd*Sigw*Gd');
Sigz=Dx*Sigx*Dx' + Dw*Sigw*Dw';
% Solve Covariance Equation recursively
Sigx2=zeros(3);
for ii=1:10
  Sigx2=Ad*Sigx2*Ad' + Gd*Sigw*Gd';
end
Sigx2=Sigx2
Sigz2=Dx*Sigx2*Dx' + Dw*Sigw*Dw';
% Variance of Output Signals
I22=eye(2);
zeta1=I22(1,:)*Sigz*I22(1,:)
  zeta2=I22(2,:)*Sigz*I22(2,:)
```

In the vector case, the autocorrelation functions are found to be:

\[
\begin{align*}
R_{x,k,i} &= E[x_k x_{k+i}^*] = \begin{cases} 
\Sigma_x (A_d^*)^i & \text{if } i \geq 0 \\
(A_d)^{-i} \Sigma_x & \text{if } i < 0 
\end{cases} \\
R_{z,k,i} &= E[z_k z_{k+i}^*] = \begin{cases} 
D_x \Sigma_x D_x^* + D_w \Sigma_w D_w^* & \text{if } i = 0 \\
D_x R_{x,k,i} D_x^* & \text{if } i \neq 0 
\end{cases}
\end{align*}
\]

**Example 5.13.** Applying the results of Example 5.12 to Equation (5.102) yields the plot of Figure 5.18.

In some cases, it may be of interest to calculate the non-mean removed variance of a signal. That is, how does one calculate \( \Sigma_{q,k} \stackrel{\wedge}{=} E[q_k q_k^*]? \) If the mean and mean removed
variance are both known, then the following can be applied:

\[
\Sigma_{q,k} = E[q_k q_k^*] = E[(q_k - \bar{q}_k + \bar{q}_k)(q_k - \bar{q}_k + \bar{q}_k)^*]
\]

\[
= E[(\bar{z}_k + \bar{q}_k)(\bar{z}_k^* + \bar{q}_k^*)]
\]

\[
= E[\bar{z}_k \bar{z}_k^*] + E[\bar{q}_k \bar{z}_k] + E[\bar{z}_k \bar{q}_k^*] + E[\bar{q}_k \bar{q}_k^*]
\]

\[
= \Sigma_{x,k} + \bar{q}_k \bar{q}_k^*
\]

(5.103)

The interpretation is that retention of the mean in the signal will cause the variance calculation to increase by an amount \( \bar{q}_k \bar{q}_k^* \).

![Figure 5.18. Autocorrelation function for Example 5.13.](image)

5.3.5 • The Expected Dynamic Operating Region

In Examples 5.10 and 5.11, the two standard deviation envelope was plotted to illustrate the connection between the calculated \( \Sigma_{z,k} \) and the time-series realization. In the vector case, this connection can also be observed through the scatter plot. Let us begin with the steady-state covariance matrix \( \Sigma_x \). Then, the \( \alpha \)-standard deviation region can be defined as:

\[
S_\alpha = \left\{ z \in \mathbb{R}^n \mid (z - \bar{z})^T \Sigma_x^{-1} (z - \bar{z}) \leq \alpha^2 \right\}
\]

(5.104)
This multidimensional elliptical region is denoted as the *Expected Dynamic Operating Region* (EDOR), since it indicates where one would expect to find the trajectory $z_k$.

One can think of the EDOR as an analytical representation of a numerically determined scatter plot.

**Example 5.14.** Reconsider the shaping filter augmented mass-spring-damper process of Example 5.7, where the shaping filter associated with Disturbance B is augmented to the physical process. Using the steady-state covariance matrix calculated in Example 5.12 and $\alpha = 2$, Figure 5.20 shows the resulting EDOR added to the appropriate scatter plot from Figure 5.8. The MATLAB code used to generate this EDOR plot is given in Table 5.4.

**Table 5.4.** MATLAB code of Example 5.14.

```
clear
% Define EDOR
zbar=[0; 0]; SigZ=[0.0201 -0.0260; -0.0260 0.1618]; alpha = 2;
SigZinv=inv(SigZ);
% Determine range of independent variable
xind_max=sqrt(alpha^2*SigZinv(2,2)/det(SigZinv)); xind_min=-xind_max
N=200; xxxind=zeros(1,N); yyyup=zeros(1,N); yyylo=zeros(1,N);
step = (xind_max-xind_min)/(N-1)
% Calculate upper and lower curve values
for iii=1:N
    xind=xind_min+step*(iii-1);
    xxxind(iii)=xind;
    bbb=SigZinv(1,2)*xind/SigZinv(2,2);
    ccc=(SigZinv(1,1)*xind*xind-alpha^2)/SigZinv(2,2);
    yyyup(iii)=-bbb+sqrt(bbb^2-ccc);
    yyylo(iii)=-bbb-sqrt(bbb^2-ccc);
end
% Plot upper and lower curves
xxxind=xxxind+zbar(1); yyyup=yyyup+zbar(2); yyylo=yyylo+zbar(2);
plot(zbar(1),zbar(2),'*k', xxxind,yyyup, 'k--', xxxind,yyylo, 'k--')
```

![Figure 5.20: Scatter plot and corresponding EDOR for Example 5.14.](image-url)
5.4 - Continuous-time Stochastic Processes

In the continuous-time framework, all of the notions of the previous section carry through, but with the obvious change in time index. The following gives a survey of the main results, without derivation. For detailed developments consult McGarty, [127], Papoulis [128] or Burl, [105]. For a stochastic process, \( x(t) \), the following definitions may be applied

\[
\begin{align*}
\overline{x}(t) &= E[x(t)] \\
\Sigma_x(t) &= E[(x(t) - \overline{x}(t))(x(t) - \overline{x}(t))^*] \\
R_x(t, \tau) &= E[(x(t) - \overline{x}(t))(x(t + \tau) - \overline{x}(t + \tau))^*]
\end{align*}
\]

(5.105)-(5.107)

One exception concerns white noise, which to define precisely is outside our current scope. However, it will be sufficient to assume that white noise is just as a stochastic process with the following autocorrelation function:

\[
R_w(t, \tau) = S_w(t) \delta(\tau)
\]

(5.108)

where \( S_w(t) \) is termed the spectral density and \( \delta(\tau) \) is the Dirac delta function, which is zero for all \( \tau \neq 0 \) and undefined \((= \infty)\) at \( \tau = 0 \). In addition, \( \delta(\tau) \) is such that

\[
\int_{-\epsilon}^{\epsilon} \delta(\tau) d\tau = 1 \text{ for all } \epsilon > 0
\]

(5.109)

Additional details concerning white noise processes in continuous-time will be given in Section 5.5.

If the continuous-time process model is

\[
\begin{align*}
\dot{x} &= Ax + Gw \\
z &= Dx
\end{align*}
\]

(5.110)-(5.111)

and \( w \) is assumed a white noise process with spectral density \( S_w \), the covariance equation is a matrix valued differential equation:

\[
\begin{align*}
\dot{\Sigma}_x &= A\Sigma_x + \Sigma_x A^* + GS_w G^* \\
\dot{\Sigma}_z &= D_x\Sigma_x D_x^*
\end{align*}
\]

(5.112)-(5.113)

If \( w \) is also stationary and \( A \) is stable, then \( \Sigma_x \overset{\Delta}{=} \lim_{t \to \infty} \Sigma_x(t) \) and \( \Sigma_z \overset{\Delta}{=} \lim_{t \to \infty} \Sigma_z(t) \), will exist and will be the solutions to the following algebraic relations:

\[
\begin{align*}
0 &= A\Sigma_x + \Sigma_x A^* + GS_w G^* \\
\Sigma_z &= D_x\Sigma_x D_x^*
\end{align*}
\]

(5.114)-(5.115)

Equation (5.114) is denoted as the continuous-time steady-state covariance equation, which is also a continuous-time Lyapunov equation. To solve such equations, the Control Systems Toolbox of MATLAB provides the function ‘lyap’.

Using the \( \rho_j \) notation of the previous section, the variance and standard deviation of the \( j^{\text{th}} \) output signal at time \( t \) are defined as:

\[
\zeta_j(t) = \rho_j \Sigma_z(t) \rho_j^* \text{ and } \sigma_j(t) = \sqrt{\zeta_j(t)}
\]

(5.116)
Similarly, the variance and standard deviation the $j^{th}$ state variable at time $t$ are defined as:

$$
\xi^{(j)}(t) = \rho_j \Sigma_x(t) \rho_j^* \quad \text{and} \quad \sigma_x^{(j)}(t) = \sqrt{\xi^{(j)}(t)}
$$

(5.117)

To indicate steady-state conditions the following notation will be employed for steady-state variance and standard deviation:

$$
\zeta^{(j)} = \rho_j \Sigma_z \rho_j^* \quad \text{and} \quad \sigma_z^{(j)} = \sqrt{\zeta^{(j)}}
$$

(5.118)

$$
\xi^{(j)} = \rho_j \Sigma_x \rho_j^* \quad \text{and} \quad \sigma_x^{(j)} = \sqrt{\xi^{(j)}}
$$

(5.119)

The autocorrelation functions are:

$$
R_x(t, \tau) = \begin{cases} 
\Sigma_x(t) (e^{At})^* & \text{if } \tau \geq 0 \\
(e^{-At}) \Sigma_x(t) & \text{if } \tau < 0
\end{cases}
$$

(5.120)

$$
R_x(t, \tau) = D_x R_x(t, \tau) D_x^*
$$

(5.121)

If $w$ is stationary and $A$ is stable, then

$$
R_x(\tau) = \lim_{t \to \infty} R_x(t, \tau) = \begin{cases} 
\Sigma_x (e^{At})^* & \text{if } \tau \geq 0 \\
(e^{-At}) \Sigma_x & \text{if } \tau < 0
\end{cases}
$$

(5.122)

$$
R_x(\tau) = \lim_{t \to \infty} R_x(t, \tau) = D_x R_x(\tau) D_x^*
$$

(5.123)

Assuming the white noise process is stationary, the relationship between $S_w$ and the discrete-time covariance matrix, $\Sigma_w$, is as follows (assuming application of the sample and hold discretization method with a sample period of $\Delta t$):

$$
\Sigma_w = S_w / \Delta t
$$

(5.124)

**Example 5.15.** Consider the following scalar process:

$$
\dot{s} = as + gp
$$

(5.125)

$$
q = d_x s
$$

(5.126)

where $a = -1.1157$, $g = 5.5816$ and $d_x = 0.3$. If $p$ is zero mean, white noise with spectral density $S_w = 0.2$, then application of Equations (5.112)-(5.113) gives

$$
\Sigma_x = -2.2314 \Sigma_x + 6.231 \quad \text{and} \quad \Sigma_z = 0.09 \Sigma_x
$$

(5.127)

If $\Sigma_x(0) = 0$, then the analytic solution to this linear first order differential equation is:

$$
\Sigma_x(t) = 2.7923(e^{-2.2314t} - 1) \quad \text{and} \quad \Sigma_z = 0.09 \Sigma_x
$$

(5.128)

Thus, the steady-state variance of the output is determined to be $\Sigma_z = 0.2513$.

If the sample and hold discretization method is applied to Equation (5.125)-(5.126) with $\Delta t = 0.2$, one arrives at:

$$
x_{k+1} = 0.8x_w + \omega_k \quad \text{and} \quad z_k = 0.3x_k
$$

(5.129)
where \( \Sigma_w = S_w / \Delta t = 1 \). Then application of Equation (5.93)- (5.94) gives

\[
\Sigma_x = 0.64 \Sigma_x + 1 \quad \text{and} \quad \Sigma_z = 0.09 \Sigma_x
\]  

(5.130)

Thus, the steady-state variance of the output of the discrete-time model is \( \Sigma_z = 0.2500 \).

Example 5.15 illustrates the importance of Equation (5.124) in converting between the continuous-time and discrete-time frameworks. Specifically, appropriate conversion will yield nearly equal signal variances from both modelling frameworks, assuming the sample time is sufficiently small.

As in Section 5.2.5, let us assume the physical process is characterized by a linear state-space model, but this time in continuous-time form:

\[
\dot{s}(p) = A(p) s(p) + B(p) m(p) + G(p) \eta(p)
\]

(5.131)

\[
q(p) = D_x(p) s(p) + D_u(p) m(p) + D_w(p) \eta(p)
\]

(5.132)

If we now assume the disturbance into this physical process is equal to the output of the following continuous-time shaping filter

\[
\dot{s}(f) = A(f) s(f) + G(f) \eta(f)
\]

(5.133)

\[
q(f) = D_x(f) s(f)
\]

(5.134)

then the two systems can be combined (as in Figure 5.10, with \( p(p) = q(f) \)) to arrive at the following compound system:

\[
\begin{bmatrix}
\dot{s}(p) \\
\dot{s}(f)
\end{bmatrix} =
\begin{bmatrix}
A(p) & G(p) D_x(f) \\
0 & A(f)
\end{bmatrix}
\begin{bmatrix}
s(p) \\
s(f)
\end{bmatrix} +
\begin{bmatrix}
B(p) \\
0
\end{bmatrix}
m(p) +
\begin{bmatrix}
G(p) D_w(f) \\
G(f)
\end{bmatrix}
p(f)
\]

(5.135)

\[
q(p) =
\begin{bmatrix}
D_x(p) & D_u(p)
\end{bmatrix}
\begin{bmatrix}
s(p) \\
s(f)
\end{bmatrix} +
\begin{bmatrix}
D_u(p)
\end{bmatrix}
m(p)
\]

(5.136)

Then, the compound system is compactly expressed as:

\[
\dot{x} = As + Bm + Gp
\]

(5.137)

\[
q = D_x s + D_u m
\]

(5.138)

where

\[
A =
\begin{bmatrix}
A(p) & G(p) D_x(f) \\
0 & A(f)
\end{bmatrix} 
\quad B =
\begin{bmatrix}
B(p) \\
0
\end{bmatrix} 
\quad G =
\begin{bmatrix}
G(p) D_w(f) \\
G(f)
\end{bmatrix}
\]

(5.140)

\[
D_x =
\begin{bmatrix}
D_x(p) & D_x(p) D_x(f)
\end{bmatrix} 
\quad D_u =
\begin{bmatrix}
D_u(p)
\end{bmatrix}
\]

(5.141)

Example 5.16. Reconsider the continuous-time mass-spring-damper process of Example 2.12. If augmented with the shaping filter of Equation (5.125)-(5.126), then the following
5.4. Continuous-time Stochastic Processes

Continuous-time compound process model will result.

\[
\begin{bmatrix}
    s^{(p)} \\
    s^{(f)}
\end{bmatrix} = \begin{bmatrix}
    0 & 1 & 0 \\
    -3 & -2 & 0.3
\end{bmatrix} \begin{bmatrix}
    s^{(p)} \\
    s^{(f)}
\end{bmatrix} + \begin{bmatrix}
    0 \\
    0
\end{bmatrix} P^{(f)}
\]

\[q^{(p)} = \begin{bmatrix}
    1 & 0 & 0 \\
    -3 & -2 & 0.3
\end{bmatrix} \begin{bmatrix}
    s^{(p)} \\
    s^{(f)}
\end{bmatrix}
\]

Application of Equations (5.114)-(5.115), with \( S_w = 0.2 \), yields

\[
\Sigma_x = \begin{bmatrix}
    0.0202 & 0.0000 & 0.1294 \\
    0.0000 & 0.0216 & 0.1443 \\
    0.1294 & 0.1443 & 2.7924
\end{bmatrix}
\]

\[
\Sigma_z = \begin{bmatrix}
    0.0202 & -0.0216 \\
    -0.0216 & 0.1132
\end{bmatrix}
\]

Application of sample and hold to (5.142)-(5.143) with \( \Delta t = 0.2 \) gives:

\[
\begin{bmatrix}
    s^{(p)}_{k+1} \\
    s^{(f)}_{k+1}
\end{bmatrix} = \begin{bmatrix}
    0.9478 & 0.1616 & 0.0048 \\
    -0.4847 & 0.6246 & 0.0431
\end{bmatrix} \begin{bmatrix}
    s^{(p)}_k \\
    s^{(f)}_k
\end{bmatrix} + \begin{bmatrix}
    0.0019 \\
    0.0271
\end{bmatrix} P^{(f)}_k
\]

\[q^{(p)}_k = \begin{bmatrix}
    1 & 0 & 0 \\
    -3 & -2 & 0.3
\end{bmatrix} \begin{bmatrix}
    s^{(p)}_k \\
    s^{(f)}_k
\end{bmatrix}
\]

Application of Equations (5.93)-(5.94), where \( \Sigma_w = S_w / \Delta t = 1 \), yields

\[
\Sigma_x = \begin{bmatrix}
    0.0201 & 0.0000 & 0.1299 \\
    0.0000 & 0.0213 & 0.1449 \\
    0.1299 & 0.1449 & 2.7777
\end{bmatrix}
\]

\[
\Sigma_z = \begin{bmatrix}
    0.0201 & -0.0213 \\
    -0.0213 & 0.1082
\end{bmatrix}
\]

Let us now take a slightly different route to a discrete-time model of the compound system. If one were to discretize the process model and disturbance model separately, and then apply Equations (5.32)-(5.34), then the following compound model would result:

\[
\begin{bmatrix}
    s^{(p)}_{k+1} \\
    s^{(f)}_{k+1}
\end{bmatrix} = \begin{bmatrix}
    0.9478 & 0.1616 & 0.0052 \\
    -0.4847 & 0.6246 & 0.0484
\end{bmatrix} \begin{bmatrix}
    s^{(p)}_k \\
    s^{(f)}_k
\end{bmatrix} + \begin{bmatrix}
    0 \\
    0
\end{bmatrix} P^{(f)}_k
\]

Then, application of Equations (5.93)-(5.94) to this compound system yields a modestly different solution

\[
\Sigma_x = \begin{bmatrix}
    0.0201 & 0.0001 & 0.1154 \\
    0.0001 & 0.0213 & 0.1256 \\
    0.1154 & 0.1256 & 2.7777
\end{bmatrix}
\]

\[
\Sigma_z = \begin{bmatrix}
    0.0201 & -0.0260 \\
    -0.0260 & 0.1618
\end{bmatrix}
\]

A comparison of these covariance matrices illustrates the small discrepancy that results from appending the shaping filter before and after the discretization step.
5.5 - Spectral Interpretations

Given the continuous-time framework, it is now convenient to return to the concept of frequency content. Consider a stationary stochastic process, \( x(t) \), along with its autocorrelation function, \( R_x(\tau) \). The spectral density function, \( S_x(\omega) \), is then defined as the Fourier transform of \( R_x(\tau) \):

\[
S_x(\omega) = \int_{-\infty}^{\infty} R_x(\tau)e^{-j\omega\tau}d\tau \tag{5.151}
\]

In addition, if given the spectral density function, the autocorrelation can be recovered via the inverse Fourier transform:

\[
R_x(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_x(\omega)e^{j\omega\tau}d\omega \tag{5.152}
\]

Using relation (5.152) with \( \tau = 0 \), one finds that

\[
\Sigma_x = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_x(\omega)d\omega \tag{5.153}
\]

Based on relation (5.153), \( S_x(\omega) \) can interpreted as a measure of the amount of variance contained in a frequency interval. Specifically, the integral

\[
\Sigma_x(\omega_1, \omega_2) = \frac{1}{2\pi} \int_{\omega_1}^{\omega_2} S_x(\omega)d\omega \tag{5.154}
\]

indicates the amount of variance with frequency characteristics in the interval \([\omega_1, \omega_2]\). Turning this statement around one could ask for the frequency interval containing the majority of the process variance. If such an interval is determined, then one would say that this interval describes the frequency content of the process.

Example 5.17. Consider a scalar process

\[
\tau_c \dot{x} = w - x \tag{5.155}
\]

where \( w \) is a stationary, zero mean white noise process with spectral density \( S_w \). In this case and based on (5.114), the steady-state variance of \( x \) is determined to be

\[
\Sigma_x = \frac{S_w}{2\tau_c} \tag{5.156}
\]

and the autocorrelation function is

\[
R_x(\tau) = \frac{S_w}{2\tau_c} e^{-|\tau|/\tau_c} \tag{5.157}
\]

Then application of relation (5.151) indicates that the spectral density function of \( x \) is

\[
S_x(\omega) = \frac{S_w}{(\tau_c \omega)^2 + 1} \tag{5.158}
\]
Plots of $S_x(\omega)$ for various values of $\tau_c$ are given in Figure 5.21, $S_w = 36$. From these curves it is clear that the bulk of the frequency content of $x$ is in the interval $[0, 1/\tau_c]$. Thus, as $\tau_c$ is made smaller, $x$ will contain larger amounts of the high frequency content.

To gain some insight into the characteristics of a continuous-time white noise process, let us consider the limit of $\tau_c$ in Example 5.17. As $\tau_c$ approaches zero, Equation (5.155) will become $x = w$ (i.e., $x(t)$ will become white noise). However, based on Figure 5.21, we see that as the signal $x$ will possess ever larger amounts of high frequency content. In fact, if $\tau_c = 0$, then the signal will contain equal amounts of all frequencies, which is reflected by the fact that its spectral density function, $S_x(\omega)$, becomes a constant, $S_w$, as $\tau_c \to 0$. Alternatively, one could take the Fourier transform of the autocorrelation function of white noise, Equation (5.108), to find that the spectral density function of white noise is a constant: $S_w(\omega) = S_w$. Also note that if the limit is applied to Equation (5.157), then $R_x(\tau) \to S_w \delta(\tau)$ since $\lim_{\tau_c \to 0} \{ e^{-|\tau|/2\tau_c} \} = \delta(\tau)$.

The part that makes continuous-time white noise difficult to understand stems from the limit of (5.156). Specifically, as $\tau_c \to 0$ the variance of $x(t)$ will approach infinity. Thus, the counter-intuitive nature of a continuous-time white noise process is that it has infinite variance, but still exists\(^2\). This result can also be seen from Equation (5.153), in that the infinite integral of a constant, $S_x(\omega) = S_w/((\tau_c\omega)^2 + 1) \to S_w$, will of course be infinite. Maybe the simplest way to observe the infinite variance conclusion is to just evaluate the autocorrelation function (5.108) at $\tau = 0$.

One may then ask why discrete-time white noise does not possess infinite variance. This is because the conversion of white noise to a discrete-time framework is essentially a process of filtering. Specifically, the continuous-time spectral density function is cut-off at the frequency $\omega_c = \pi/\Delta t_s$ where $\Delta t_s$ is the sample period of the discrete-time process. Application of this notion to Equation (5.154), provides insight into the origin

\(^2\)Strictly speaking, continuous-time white noise does not exist. The best one can do is work with a Wiener process, the derivative of which has characteristics similar to what we expect from white noise. For a detailed discussion see McGarty [127]. However, since we are concerned only with linear systems, one may forgo these technicalities since in the linear system case both approaches will lead to the same result, namely Equations (5.112)-(5.115).
of Equation (5.124)

\[
\Sigma_{w, DT} = \frac{1}{\pi} \int_0^{\omega_c} S_{w, CT}(\omega) d\omega = \frac{S_w}{\pi} \int_0^{\pi/\Delta t} d\omega = \frac{S_w}{\Delta t}
\]

(5.159)

For additional details on the sampling of continuous-time white noise, see Balakrishnan, [120].

You may have noticed that in the continuous-time model of Equations (5.110)-(5.111), the term \(D_x p\) was omitted (or equivalently \(D_x\) was set equal to zero). This is because we do not want white noise to be a direct component of the performance output and cause it to have infinite variance. However, in the process model of Equations (5.131)-(5.132) we have included the term \(D_x^{(p)} p^{(p)}\), since \(p^{(p)}\) is expected to be a physically motivated colored noise disturbance with finite variance. To ensure this is the case, the form of the shaping filter, Equations (5.133)-(5.134), also exclude the term \(D_w^{(f)} p^{(f)}\), to ensure the output of the shaping filter will have finite variance and correspond to a physically realizable signal. In this case, the compound system of (5.137)-(5.138) does not contain the \(D_x p\) and we are in no danger of an infinite variance at the performance output. Again, in the discrete-time case, this sort of technical issue is not of concern and one is free to add discrete-time white noise directly to the performance output equation, as in Equations (5.26)-(5.27) and (5.72)-(5.73). However, while the discrete-time framework has no mathematical objection to feeding white noise directly into a physical process, a better practice is to always employ a shaping filter so as to properly capture the spectral characteristic of the disturbance and avoid having the variance of the disturbance dependent on the sample time of the process.

It should also be noted that in subsequent chapters we will find another signal possessing infinite variance. In the case of physical measurements, indicated by the measurement equation \(y = C x\), there will be a need to add measurement noise. In this case, the measurement equation will be of the form \(y = C x + v\), where \(v\) is a white noise process that is independent of the white noise disturbance \(w\). Thus, the signal \(y\) will have infinite variance. However, if the signal is being used as the input of a state observer of Chapter 4 (or a state estimator of Chapter 6), then the resulting state estimate of interest will have finite variance. It is additionally noted that if converting the continuous-time measurement equation, \(y = C x + v\), to a discrete-time form, \(y_k = C x_k + v_k\), then the variance of the white noise signal \(v_k\) is given by

\[
\Sigma_v = S_v / \Delta t
\]

(5.160)

where \(S_v\) is the spectral density of the continuous-time process. Clearly, this relation possesses great similarity to that of Equation (5.124).

An alternate approach to calculating Equation (5.123) is to determine the \(w\) to \(x\) transfer function of (5.155): \(G(s) = 1/(\tau c s + 1)\), where \(s\) in this case is the Laplace variable. Then, \(S_x(\omega)\) can be calculated as \(S_w|G(j \omega)|^2\). To generalize this procedure let \(z\) be the output of a linear system excited by a stochastic process \(w\). If the transfer function of the linear system is \(G(s)\) and the spectral density function of the input \(w\) is \(S_w(\omega)\), then the spectral density function of output \(z\) is

\[
S_z(\omega) = G(j \omega) S_w(\omega) G(j \omega)^\ast
\]

(5.161)

Note that both the transfer function and the spectral density functions may be matrix valued. If matrix valued then the off diagonal terms may be complex valued. (Also, recall
5.5. Spectral Interpretations

that the adjoint operation on the right transfer function includes complex conjugation.) Equation (5.161) is depicted in Figure 5.22.

If the linear system is excited by white noise then a simpler version of (5.161) will result:

\[ S_x(\omega) = G(j\omega)S_w G(j\omega)^* \] (5.162)

This perspective illustrates the importance of white noise in the construction of disturbance models. Since white noise contains all frequencies, the transfer function \( G(s) \) can be used to attenuate (or filter out) undesired frequencies and amplify others. Equation (5.162) illustrates the shaping filter terminology (discussed in Section 5.2), as a linear system used for the construction of a disturbance model. As discussed in Section 5.2, the output of a shaping filter is frequently termed colored noise. This terminology is based on an analogy with the frequency content of light. Specifically, white light contains an equal amount of all light frequencies, while colored light contains only a portion of the light frequencies.
**Example 5.18.** Reconsider the shaping filter augmented mass-spring-damper process of Example 5.16 (Equations 5.142 and 5.143). For a state space system $\dot{x} = Ax + Gw$ and $z = D_x x + D_w w$, the $w$ to $z$ transfer function can be calculated as

$$G(s) = D_x (sI - A)^{-1} G + D_w$$ (5.163)

Then, application of Equation (5.162) results in the plot of Figure 5.23. If the entire process of Equations (5.142) and (5.143) is thought of as a shaping filter, then the plot illustrates that the spectral density function of the input, $S_w(\omega)$, has been filtered by the process to create the output spectral density shapes, $S_{z1}(\omega)$ and $S_{z2}(\omega)$, both of which are proportional to the square of the Bode plots of the process. The MATLAB code used to generate Figure 5.23 is given in Table 5.5.

Table 5.5. MATLAB code of Example 5.18.

```matlab
clear
% Mass-Spring-Damper Model in Continuous-time
Ap=[0 1; -3 -2]; Bp=[0; 0]; Gp=[0; 1];
Dxp=[1 0; -3 -2]; Dup=[0; 0]; Dwp=[0; 1];

% Shaping Filter in Continuous-time
Af=-1.1157; Bf=0; Gf=5.5816; Dxf=0.3; Dwf=0; pfbar=0; S_pf=0.2;

% Calculate Compound System
Ac=[Ap Gp*Dxf; 0 0 Af]; Bc=[Bp; 0]; Gc=[Gp*Dwf; Gf]; Sw=S_pf;
Dxc=[Dxp Dwp*Dxf]; Duc=Dup; Dwc=Dwp*Dwf;

% Calculate and plot Spectral Density of the Output
ww=logspace(-2,2,500);
for ii=1:500
    s=i*ww(ii); Gplus=Dxc*inv(s*eye(3)-Ac)*Gc;
    Sz(:,ii)=diag(Gplus*Sw*Gplus');
end
loglog(ww,Sz(1,:),ww,Sz(2,:),[1e-2 1e2],[Sw Sw])
axis([1e-2 1e2 1e-5 1e0])
```

### 5.6 Design of Simple Disturbance Models

In simple terms, the design of a disturbance model is the selection of a linear system (or shaping filter) such that if excited by a white noise process will have an output with statistical characteristics equal to those of the disturbance of interest. Unfortunately, a detailed review of data-based disturbance modeling is beyond our scope. For detailed descriptions the reader should consult: Bendat and Piersol [121], Soderstrom and Stoica [122], Ljung and Glad [123], Larimore [124], Zhu [125] and Qin [126].

The goal of this section is much more modest. That is, to present some simple model structures that will allow one to specify the mean, variance and autocorrelation of the desired disturbance model. While this simple set of modeling tools will be rather useful in subsequent examples and homework problems, they should not be thought of as a strong substitute for the rigorous data drive methods available in the literature.

#### 5.6.1 First Order Shaping Filters

The simplest disturbance model is the first order system:

$$\dot{x} = ax + gp$$ (5.164)

where $p$ is a stationary white noise process with mean $\overline{p}$ and spectral density $S_w$. Since the goal is to design the disturbance $s(t)$, one is free to select the parameters $a$, $g$, $\overline{p}$ and
$S_w$, where $a < 0$ is required to ensure stability. As a first selection, define $g$ as negative $a$. In this case it is found that $s^{SSOP} = \lim_{t \rightarrow \infty} \tilde{s}(t)$ will be equal to $\tilde{p}(= p^{SSOP})$, regardless of the value selected for $a$. Let us now convert to deviation variables ($x = s - s^{SSOP}$ and $w = p - p^{SSOP}$) and calculate the steady-state variance using Equation (5.114):

$$0 = 2a \Sigma_x + a^2 S_w$$

(5.165)

If $S_w$ is selected to be

$$S_w = \frac{-2}{a} \Sigma_x^{(sp)}$$

(5.166)

then $\Sigma_x$ will be equal to $\Sigma_x^{(sp)}$ regardless of the value selected for $a$. Finally, one can select the value of $a$ to correspond to the characteristic autocorrelation time, $\tau_c$. That is select $a = -1/\tau_c$. The particular advantage of using the above form is that a change to any one of the new physically meaningful, parameters ($\tilde{p}$, $\Sigma_x^{(sp)}$, and $\tau_c$) will influence only the output characteristic related to that parameter (mean, variance and autocorrelation, respectively).

**Example 5.19.** Assume the disturbance of interest has a mean of 2, a variance of 18 and a frequency content in the interval $[0 \ 10]$ rad/s. If a first order shaping filter is employed then one should set $\tilde{p} = 2$. Based on the frequency content, the characteristic correlation time is set to $\tau_c = 0.1$. Thus, one should set $g = -a = 10/s$. Finally, the spectral density should be $S_w = 3.6$.

If the desired frequency content is changed to the interval $[0 \ 0.01]$ rad/s, then $\tau_c = 100$ and $S_w$ should be set to 0.036. Figure 5.24 illustrates the spectral density functions for the two cases (as well as the $\tau_c = 1$ case). The key point being that the area under each spectral density function is the same, all equal to $2\pi \times 18$. Thus, changes to the correlations time will change the frequency content, but not change the total variance of the process.

![Figure 5.24. Spectral density functions for Example 5.19.](image)
5.6.2 Second Order Resonant Filters

Consider a second order process of the following form

\[
\dot{s} = \begin{bmatrix} 0 & 1 \\ -\omega_c^2 & -2\chi\omega_c \end{bmatrix} s + \begin{bmatrix} 0 \\ 0 \end{bmatrix} p \\
q = \begin{bmatrix} 1 & 0 \end{bmatrix} s
\]

(5.167)

Again, the task is to select the parameters of this model \((p, S_w, g, \omega_c, \chi)\) such that the statistics of \(q\) match the disturbance of interest. If \(g = \omega_c^2\), then \(q_{SSOP} = \lim_{t \to \infty} q(t) = \bar{p}\). Furthermore, if input spectral density is set to \(S_w = (4\chi/\omega_c)\Sigma_z^{(sp)}\), then \(\Sigma_z = \Sigma_z^{(sp)}\).

This can be verified by solving Equation (5.114) analytically. If \(\chi\) is selected to be \(\geq 1\), then the shaping filter will act as two first order filters in series, where the correlation times of the two filters are the two roots of \(s^2 + 2\chi\omega_c s + \omega_c^2 = 0\). However, if \(\chi\) is selected to be \(< 1\) (but \(> 0\) to ensure stability), then the filter will resonate at the frequency \(\omega_c\) and serve to amplify the white noise around that frequency.

**Example 5.20.** Consider the filter of Equation (5.167) with \(\bar{p} = 10, \Sigma_z^{(sp)} = 4, g = \omega_c^2\), and \(\omega_c = 2\pi/5 = 1.257\). Then, Figure 5.25 illustrates the spectral density function of \(z\) for various values of \(\chi\). For all three case (but especially \(\chi = 0.05\)), the resonant frequency is observed to be at \(\omega = 1.257\). Also, note that the area under all three curves is the same and presumably equal to 4. A time realization of \(q(t)\) is given in Figure 5.26. The left plot shows that the average is 10 and the standard deviation is 2. The right plot shows that the period of the captured oscillations is about 5, which stems from our assignment of \(\omega_c = 2\pi/5\).

![Figure 5.25. Spectral density functions for Example 5.20.](image)
5.7 Gramians and Covariance Equation Solutions

In the proof of the Lyapunov Stability Theorem of Section 4.1.3, it was shown that if the matrix $A$ is stable, then

$$P = \int_{0}^{\infty} e^{A^t} Q e^{At} \, dt$$

is the solution to $A^* P + PA + Q = 0$ (5.168)

and

$$P = \int_{0}^{\infty} e^{At} Q e^{A^* t} \, dt$$

is the solution to $AP + PA^* + Q = 0$ (5.169)

Furthermore, if $A_d$ is stable then

$$P = \sum_{k=0}^{\infty} A_d^k QA_d^* d \, t$$

is the solution to $P = A_d^* PA_d + Q$ (5.170)

and

$$P = \sum_{k=0}^{\infty} A_d^k QA_d^* d \, t$$

is the solution to $P = A_d PA_d^* + Q$ (5.171)

Clearly, relations (5.169) and (5.171) will be of interest when looking for solutions to the steady-state covariance equations in continuous and discrete-time, respectively.

$$0 = A \Sigma_x + \Sigma_x A^* + G S_w G^*$$

(5.114)

$$\Sigma_x = A_d \Sigma_x A_d^* + G_d \Sigma_w G_d^*$$

(5.93)

As part of the proof of Theorem 4.3 it was shown that $P$ would be positive definite if $Q$ was known to be positive definite. As such, it is trivial to conclude that $G S_w G^* > 0$ is a sufficient condition for the covariance matrix $\Sigma_x$ to also be positive definite. However, in many cases $G S_w G^*$ will not be positive definite, but merely positive semi-definite. For example, reconsider Example 5.16, where based on Equation (5.142) we find that

$$G S_w G^* = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 6.231 \end{bmatrix}$$

(5.172)
Clearly, this matrix is not positive definite. Yet, the resulting covariance matrix, of Equation (5.144) is positive definite (with eigenvalues of 0.0075, 0.0209 and 2.8058). Thus, there must be some other (necessary and sufficient) condition for $\Sigma_x$ to be positive definite. Before giving this condition let us consider some lemmas that could have been discussed in Chapter 4.

**Lemma 5.1.** Assume $A$ is stable. Then, the range space of the controllability gramian:

$$P_C = \int_0^\infty e^{At}BB^* e^{A^*t} \, dt$$  \hspace{1cm} (5.173)

is equal to the controllable subspace of pair $(A, B)$. That is, $\Re(P_C) = M_C(A, B)$.

**Lemma 5.2.** Assume $A$ is stable. Then, the null space of the observability gramian:

$$P_O = \int_0^\infty e^{A^*t} C^* C e^{At} \, dt$$  \hspace{1cm} (5.174)

is equal to the unobservable subspace of pair $(A, C)$. That is, $\N(P_O) = M_{UO}(A, C)$.

**Lemma 5.3.** Assume $A_d$ is stable. Then, the range space of the controllability gramian:

$$P_C = \sum_{k=0}^\infty A_d^k B_d B_d^* A_d^{-k}$$  \hspace{1cm} (5.175)

is equal to the controllable subspace of pair $(A_d, B_d)$. That is, $\Re(P_C) = M_C(A_d, B_d)$.

**Lemma 5.4.** Assume $A_d$ is stable. Then, the null space of the observability gramian:

$$P_O = \sum_{k=0}^\infty A_d^k C^* C A_d^k$$  \hspace{1cm} (5.176)

is equal to the unobservable subspace of pair $(A_d, C)$. That is, $\N(P_O) = M_{UO}(A_d, C)$.

The proofs these lemmas can be found in most texts on Linear System Theory (see for example Chen [116] or Duller and Paganini [119]). Definition 3.30 (repeated here for convenience) will help connect these lemmas to the desired condition.

**Definition 3.30.** Consider a positive semi-definite matrix $Q$, then the principal square root of $Q$ is the unique positive semi-definite matrix defined as $Q^{1/2} = \Phi \Lambda^{1/2} \Phi^{-1}$, where $\Phi$ and $\Lambda$ are from the eigenvalue-eigenvector decomposition of $Q$ (see Section 3.6).

Given this definition and Lemma 5.1 it is easily concluded that the $P$ of Equation (5.169) will be positive definite (i.e., $\Re(P)$ is the whole space or $\N(P)$ is zero) if and only if the pair $(A, Q^{1/2})$ is completely controllable. Furthermore, if the pair $(A, Q^{1/2})$ is not completely controllable then $P$ will be merely positive-semi-definite. Of course, all of these statements require $A$ to be stable, otherwise it is unlikely a positive semi-definite solution, $P$, will even exist (see discussion below).

Turning back to the solution to Equation (5.114), the condition $(A, B)$ is completely controllable can be implemented in a number of equally valid ways. An obvious choice would be to use the pair $(A, (G_S w G^*)^{1/2})$, as done in the preceding paragraph. Another
choice is to use the pair \((A, GS_w^{1/2})\). However, if \(S_w\) is guaranteed positive definite (which is a natural assumption), then it would be appropriate to use just the pair \((A, G)\).

**Theorem 5.1.** Assume \(S_w\) is positive definite. If \(A\) is stable, then Equation (5.114) will possess a unique positive semi-definite solution, \(\Sigma_x\). Furthermore, \(\Sigma_x\) is positive definite if and only if the pair \((A, G)\) is completely controllable.

**Theorem 5.2.** Assume \(\Sigma_w\) is positive definite. If \(A_d\) is stable, then Equation (5.93) will possess a unique positive semi-definite solution, \(\Sigma_w\). Furthermore, \(\Sigma_x\) is positive definite if and only if the pair \((A_d, G_d)\) is completely controllable.

While the above discussion of the properties of a solution to the Lyapunov equation could have been given in Chapter 4, the notion of stochastic analysis provides a good physical interpretation of the results. Consider a discrete-time system

\[
x_{k+1} = A_d x_k + B_d u_k,
\]

where the pair \((A_d, B_d)\) is completely controllable. In that case, there will exist an input sequence, \(u_k\), such that any value of the state (or any direction of the state space) can be reached. Now consider

\[
x_{k+1} = A_d x_k + G_d w_k,
\]

where the pair \((A_d, G_d)\) is completely controllable. Since \(w_k\) is a white noise sequence, the assumption of \((A_d, G_d)\) being completely controllable indicates that all directions of the state space will be reached (or excited) eventually (as \(k\) approaches infinity). This eventual excitation of all directions of \(x_k\) is what causes \(\Sigma_x = \lim_{k \to \infty} E[x_k x_k^*]\) to be positive definite.

On the other hand, if \((A_d, G_d)\) is not completely controllable, then one will find directions of the state space that cannot be excited by the sequence \(w_k\). In that case, there will be a linear combination of the state vector that eventually becomes zero (or will stay at zero if the initial condition is zero) and will cause \(\Sigma_x\) to have a null space or equivalently be merely positive semi-definite.

Before concluding, let us consider some pathological cases. Suppose \(A_d\) is not stable. In this case, it is possible that the covariance equation could still have a solution. Specifically, if the pair \((A_d, G_d)\) is not stabilizable (i.e., there are unstable modes that cannot be excited by \(w_k\)), then a solution to the covariance equation might exist and that solution might not be unique. To see this, let us consider the following form of Equation (5.91):

\[
\Sigma_{x,k} = A_d^k \Sigma_{x,0} A_d^{-k} + \sum_{i=0}^{k} A_d^i (G_d \Sigma_w G_d^*) A_d^{i*}
\]

(5.177)

If \(A_d\) is not stable, then we know that \(A_d^k\) will not converge to zero. However, if \((A_d, G_d)\) is not stabilizable, then it is possible that \(A_d^i (G_d \Sigma_w G_d^*) A_d^{i*}\) may still converge to zero. If this is the case, and \(\Sigma_{x,0} = 0\), then there will definitely be a positive semi-definite solution to the covariance equation. If \(A_d\) is marginally stable (i.e., at worst there are eigenvalues with a magnitude equal to unity) and \((A_d, G_d)\) is not stabilizable, then the assumption of \(\Sigma_{x,0} = 0\) is not required and we can conclude that non-unique, possibly positive definite, solutions will exist.

The good news is that if \((A_d, G_d)\) is, in fact, stabilizable and \(A_d\) is not stable (marginally or otherwise), then we know that a positive semi-definite solution will not exist (i.e., \(\Sigma_{x,k}\) of (5.177) will definitely go to infinity). This fact will be important in subsequent chapters when we are looking to design controllers and observers. That is, we will always want our disturbance to excite any unstable modes. If not, then the controller design procedure may choose to ignore those modes (since the associated covariance is not infinite) and leave them unstable. While a simulation of the dynamic model may not
have a problem with this instability, the real system will likely experience small amounts of excitement in these unstable directions and likely suffer greatly.

**Example 5.21.** Consider the following scalar version of canonical form:

\[
A_d = \begin{bmatrix} a_1 & 1 \\ 0 & a_2 \end{bmatrix} \quad G_d = \begin{bmatrix} g_1 & 1 \\ 0 & g_2 \end{bmatrix}
\]

In this case, the controllability matrix \( L_c \) is

\[
L_c = \begin{bmatrix} G_d & A_d G_d \end{bmatrix} = \begin{bmatrix} g_1 & 0 & a_1 g_1 & g_2 \\ 0 & g_2 & 0 & a_2 g_2 \end{bmatrix}
\]

Clearly, this system will be completely controllable, unless \( g_2 = 0 \). If \( g_2 = 0 \) and \( |a_2| < 1 \), then the pair is stabilizable, and if \( g_2 = 0 \) and \( |a_2| \geq 1 \), then the pair is not stabilizable.

If \( \Sigma_w = I \), then

\[
A_d^i G_d \Sigma_w G_d^* A_d^{i,i} = \begin{bmatrix} (a_1^i g_1)^2 + (x_i g_2)^2 & (x_i g_2)(a_1^i g_1) \\ (a_2^i g_2)(x_i g_2) & (a_2^i g_2)^2 \end{bmatrix}
\]

where \( x_i = \sum_{j=0}^i a_1^{i-j} a_2^j \). If \( \Sigma_{x,0} = \text{diag}(s_{1,0}, s_{2,0}) \), then one finds that

\[
\Sigma_{x,k} = A_d^k \Sigma_{x,0} A_d^k + \sum_{i=0}^{k} A_d^i G_d \Sigma_w G_d^* A_d^{i,i}
\]

\[
= \begin{bmatrix} a_1^{2k} s_{1,0} + (x_k s_{2,0})^2 & x_k a_2^{2k} s_{2,0} \\ x_k a_1^{2k} s_{2,0} & a_2^{2k} s_{2,0} \end{bmatrix} + \sum_{i=0}^{k} \begin{bmatrix} (a_1^i g_1)^2 + (x_i g_2)^2 & (x_i g_2)(a_1^i g_1) \\ (a_2^i g_2)(x_i g_2) & (a_2^i g_2)^2 \end{bmatrix}
\]

Let us now consider the various cases.

1) \( A_d \) **stable:** If both \( |a_1| < 1 \) and \( |a_2| < 1 \), then it is easily concluded \( \Sigma_{x,k} \) that will converge since \( A_d^i G_d \Sigma_w G_d^* A_d^{i,i} \) will converge to zero.

2) \( A_d \) **stable** and \( (A_d, G_d) \) **not completely controllable:** If \( |a_1| < 1, |a_2| < 1 \) and \( g_2 = 0 \), then it is obvious that \( \Sigma_{x,k} \) and its limit are not positive definite, since

\[
A_d^i G_d \Sigma_w G_d^* A_d^{i,i} = \begin{bmatrix} (a_1^i g_1)^2 & 0 \\ 0 & 0 \end{bmatrix}
\]

3) \( A_d \) **stable** and \( (A_d, G_d) \) **completely controllable:** If \( |a_1| < 1, |a_2| < 1 \) and \( g_2 \neq 0 \), then with the aid of the Schwarz Inequality (see Debnath and Mikusinski, [112]), it can be shown that \( \Sigma_{x,k} \) and its limit are indeed positive definite.

4) \( A_d \) **not stable** and \( (A_d, G_d) \) **stabilizable:** If \( |a_1| \geq 1, |a_2| < 1 \) and \( g_2 = 0 \), then the limit of \( A_d^i G_d \Sigma_w G_d^* A_d^{i,i} = \begin{bmatrix} (a_1^i g_1)^2 & 0 \\ 0 & 0 \end{bmatrix} \) will be infinite.

5) \( A_d \) **not stable** and \( (A_d, G_d) \) **not stabilizable:** If \( |a_1| < 1, |a_2| > 1 \) and \( g_2 = 0 \), then the limit of \( A_d^i G_d \Sigma_w G_d^* A_d^{i,i} = \begin{bmatrix} (a_1^i g_1)^2 & 0 \\ 0 & 0 \end{bmatrix} \) is zero and \( \Sigma_{x,k} \) will converge to a positive semi-definite solution if \( s_{2,0} = 0 \).
6) $A_d$ marginally stable and $(A_d, G_d)$ not stabilizable: If $|a_1| < 1$, $|a_2| = 1$ and $g_2 = 0$, then the limit of $A_d^i G_d \Sigma_w G_d^i A_d^i = \begin{bmatrix} (a_1^i g_1)^2 & 0 \\ 0 & 0 \end{bmatrix}$ will be zero and $\Sigma_{\epsilon,k}$ will converge to a non-unique solution

$$\Sigma_\epsilon = \begin{bmatrix} (\bar{x}_\infty)^2 & \bar{x}_\infty \\ \bar{x}_\infty & 1 \end{bmatrix} s_{2,0} + \sum_{i=0}^{k} \begin{bmatrix} (a_1^i g_1)^2 \\ 0 \end{bmatrix}$$

where $\bar{x}_\infty = \sum_{j=0}^{\infty} a_1^j$ and $s_{2,0}$ can be arbitrarily selected.

7) $A_d$ marginally stable and $(A_d, G_d)$ stabilizable: If $|a_1| < 1$, $|a_2| = 1$ and $g_2 \neq 0$, then $A_d^i G_d \Sigma_w G_d^i A_d^i = \begin{bmatrix} (a_1^i g_1)^2 + (\epsilon_1 g_2)^2 & (\epsilon_1 g_2)(a_1^i g_2) \\ (a_1^i g_2)(\epsilon_1 g_2) & (a_1^i g_2)^2 \end{bmatrix}$ will converge not to zero, but to $\begin{bmatrix} (\bar{x}_\infty)^2 & \bar{x}_\infty \\ \bar{x}_\infty & 1 \end{bmatrix} \bar{g}_2^2$, and consequentially $\Sigma_{\epsilon,k}$ will not converge.

### 5.8 - Chapter Summary

The chapter begins with a short review of probability and simple random variables. While the notions of mean and variance are central throughout the chapter the most important aspect of this first section was to review the concept of correlation between two random variables. Then, the concept of a stochastic process was introduced as a sequence of random variables. However, in contrast to an arbitrary collection, the index of this set of random variables is associated with time. This time-indexing of a stochastic process engenders it with a unique physical interpretation that is quantified by the autocorrelation function. It is then shown that changes in the autocorrelation function will result in distinct characteristics from the stochastic process realization. If the stochastic process is used as a disturbance input to a physical process, it was shown that the autocorrelation function of the disturbance will have a strong influence on the output signals of the physical process. It was also shown that one of the most convenient ways to model a stochastic process is to construct an appropriate shaping filter that is driven by white noise. This approach was supported by a discussion of the spectral perspective of a stochastic process, which also gave useful insight into the nature of the white noise process. The chapter also discussed the special care one must use in converting from continuous-time to a discrete-time framework when dealing with stochastic systems.

However, by far, the most important aspect of the chapter was the derivation of the steady-state covariance equation in discrete-time (5.93)-(5.94) and continuous-time (5.114)-(5.115). As fundamental tools of stochastic analysis, these relations will be revisited in subsequent chapters and will form the basis of optimal state estimation (Chapter 6) as well as covariance based controller design, which is at the heart of Part III of the book.

### Exercises

5.1. Consider the following discrete-time shaping filter: $s_{k+1} = 0.925 s_k + 0.1 p_k$ where $p_k$ is stationary white noise with $\overline{p} = 5$ and $\Sigma_p = 16$ (that is the standard deviation is equal to 4).
(i) Determine the steady-state average and steady-state variance of $s_k$.

(ii) Using the MATLAB function ‘randn’, simulate a realization of this process. The interval of the simulation should be 100 samples. Assume the initial condition is known to be zero.

(iii) Plot the average sequence and the 2 standard deviation envelope for the conditions of part (ii).

(iv) Combine the plots of parts (ii) and (iii) to arrive at a plot similar to that of Figure 5.13 (top).

(v) Consider the following numeric approaches to estimate steady-state average and steady-state variance:

\[
\bar{s}_N = \frac{1}{N} \sum_{k=1}^{N} s_k \quad \text{and} \quad \Sigma_{x,N} = \frac{1}{N-1} \sum_{k=1}^{N} (s_k - \bar{s}_N)^2
\]

Repeat the simulation of part (ii) and calculate $\bar{s}_N$ and $\Sigma_{x,N}$. Compare these estimates to the values obtained by the analytic approach. (To get an accurate estimate, the simulation will likely need to be extended to 10,000 samples or more.)

5.2. Consider the following first-order discrete time system:

\[
x_{k+1} = 0.8x_k + u_k + w_k
\]

\[
y_k = 0.1x_k + v_k
\]

where $w_k$ and $v_k$ are independent white-noise sequences with zero-mean and variances $\Sigma_w = 2$, $\Sigma_v = 1$, respectively.

(i) Determine the open-loop ($u_k = 0$ for all $k$) steady-state variance $\Sigma_x = \lim_{k \to \infty} E[x_k^2]$.

(ii) Assume $u_k = -0.3x_k$ and re-calculate the steady-state variance $\Sigma_x = \lim_{k \to \infty} E[x_k^2]$.

(iii) Now assume $u_k = -0.3y_k$ and re-calculate the steady-state variance $\Sigma_x = \lim_{k \to \infty} E[x_k^2]$. In this case you may wish to define a variable $r_k = v_k + w_k = \begin{bmatrix} 1 & 1 \end{bmatrix} v_k w_k$ and then ask what is $E[r_k r_k^\top]$, keeping in mind that $w_k$ and $v_k$ are independent of each other.

(iv) Assume the measurement feedback of part (iii) must be used. However, you are given the option of selecting a new sensor. Select a new measurement variance $\Sigma_v$ such that the closed-loop variance is smaller than the open-loop. At what value of $\Sigma_v$ will the open-loop variance equal the closed-loop?

5.3. Consider the following continuous-time process:

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w
\]

(i) Discretize this system using the sample and hold method (with $\Delta t = 0.1$).

(ii) Apply the controller $u_k = Lx_k = [-7 \quad -3]x_k$ and simulate the open and closed-loop responses assuming $w_k = 0$ for all $k$ and an initial condition $x_0 = [1 \quad -2]^\top$. Use a time interval of 10 time units ($k = 0$ to 100).
(iii) Assume the disturbance, \( w_k \), is a white noise sequence with zero mean and simulate the open- and closed-loop responses for the following three cases:

a) \( \Sigma_w = 0.01 \); b) \( \Sigma_w = 1 \); c) \( \Sigma_w = 100 \)

(iv) Comment on any observations.

5.4. Consider the following continuous-time process:

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w
\]

Assume \( w(t) \) is a stationary white noise process with zero mean and a power spectral density, \( S_w \), equal to 0.1.

(i) Assume an open-loop configuration \( (u = 0) \) and determine the state covariance matrix at steady-state, using the MATLAB function ‘lyap’ to solve Equation (5.114).

(ii) Repeat part (i) assuming a closed-loop configuration with \( u(t) = Lx(t) = \begin{bmatrix} -7 & -3 \end{bmatrix} x(t). \)

(iii) Discretize this system using the sample and hold method (with \( \Delta t = 0.1 \)). Make sure to convert \( S_w \) to \( \Sigma_w \) by using Equation (5.124).

(iv) Assume an open-loop configuration and determine the state covariance matrix at steady-state, using the MATLAB function ‘dlyap’ to solve Equation (5.93). Repeat for the closed-loop case \( (u_k = Lx_k = \begin{bmatrix} -7 & -3 \end{bmatrix} x_k). \) Compare these covariance matrices with those obtained in parts (i) and (ii).

(v) Repeat part (iv) by implementing the recursion of Equation (5.91) with a MATLAB ‘for loop’. Plot the variance of both state variables (the diagonal elements of the covariance matrix sequence) to observe the convergence of each. Compare the steady-state covariance matrices with those obtained in part (iv).

(vi) Repeat parts (iii) and (iv) with \( \Delta t = 0.01 \).

(vii) Repeat parts (iii) and (iv) with \( \Delta t = 0.5 \).

5.5. Consider the following process driven by zero-mean white noise:

\[
\dot{x} = Ax + Gw; \quad y = Cx \quad \text{where} \quad A = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad S_w = 5
\]

Calculate analytically \( \sigma_y^2(t) \), the time dependant variance of the output \( y \). (Hint: Given the structure of the matrices: \( A \), \( G \) and \( C \) you should be able find a scalar version of the differential equation required to solve this problem.)

5.6. Consider the following process driven by zero-mean white noise:

\[
\dot{x} = Ax + Bu + Gw \quad \quad u = Lx
\]

where

\[
A = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad L = \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad S_w = 5
\]

And the controller is defined as \( u = -Lx \). Determine the following covariance matrix values:
\begin{enumerate}
\item \[ \Sigma_x = \lim_{t \to \infty} E[x(t)x(t)^*] \]
\item \[ \Sigma_u = \lim_{t \to \infty} E[u(t)u(t)^*] \]
\end{enumerate}

First perform these calculations by hand (the structure of \(A - BL\) is such that an analytic solution to the covariance is easily obtained), and then verify using the MATLAB function 'lyap'.

5.7. Consider the following continuous-time stochastic process:
\[ \dot{s}_1 = 10(p - s_1) \]
\[ \dot{s}_2 = 10(s_1 - s_2) \]
where \(p\) is white noise with mean \(\bar{p} = 13\) and mean removed spectral density \(S_w = 4\). Determine the mean and mean removed variance of \(s_2\) at steady-state.

5.8. Consider the following unstable process:
\[ \dot{x} = Ax + Bu \]
\[ y = Cx + v \]
where \(A = \begin{bmatrix} -3 & 0 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 2 \end{bmatrix} \)

(i) Using a measurement feedback structure, \(u = -ly\), select a gain, \(l\), that will stabilize the plant.

(ii) If the sensor noise, \(v(t)\), is a zero mean white noise process with a spectral density of \(S_v = 4\), determine the following covariance values at steady-state:
\[ a. \quad \Sigma_x = \lim_{t \to \infty} E[x(t)x(t)^*] \]
\[ b. \quad \Sigma_u = \lim_{t \to \infty} E[u(t)u(t)^*] \]
\[ c. \quad \Sigma_y = \lim_{t \to \infty} E[y(t)y(t)^*] \]
Feel free to use the MATLAB function 'lyap' to determine \(\Sigma_x\).

5.9. Consider the following band-pass shaping filter:
\[ \dot{x}_1 = w - x_1 \]
\[ \dot{x}_2 = w - x_1 - x_2 \]
\[ z = 10x_2 \]
where \(w\) is white noise with a mean of \(\bar{w}\) and a mean removed spectral density of \(S_w\).

(i) Determine the mean of \(z\) in steady-state.

(ii) Determine the value of \(S_w\) that must be selected such that the steady-state variance of \(z\) will equal 50.

5.10. Consider the following process driven by zero-mean white noise:
\[ \dot{x} = Ax + Bu + Gw \]
\[ u = -Lx \]
\[ y = Cx + v \]
where
\[ A = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad L = \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]
\[ C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad S_w = 5, \quad S_v = 1 \]

It is desired to implement an observer for this system. It is proposed to use an observer of the form \(\hat{x} = (A - BL)\hat{x} + K(y - C\hat{x})\) where \(K = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\).
Exercises 187

(i) Determine the steady-state variance of the error signal, defined as \(e(t) = x(t) - \hat{x}(t)\). In other words find: \(\Sigma_e = \lim_{t \to \infty} E[e(t)e(t)^*]\)

(ii) Unfortunately, there is a conceptual flaw with the observer above. Please identify this flaw and describe why such an observer is a poor reflection of reality. Also, indicate if there is any hope to obtaining a more meaningful observer. (Hint: Recall the notion of Detectability from Chapter 4.)

5.11. Consider the following continuous-time process:

\[
\dot{s} = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
-1 & -2 & -3
\end{bmatrix} s + \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix} p
\]

Assume \(p(t)\) is a stationary white noise process with a mean of 10 and a power spectral density, \(S_w\), equal to 0.1.

(i) Determine the steady-state average and steady-state covariance matrix of the state vector \(s\). Feel free to use the MATLAB function ‘lyap’.

(ii) Plot the continuous-time EDOR ellipse corresponding to the first and third state variables. Set the number of standard deviations parameter, \(\alpha\), equal to 2.

(iii) Discretize this system using the sample and hold method (with \(\Delta t = 1.0\)). Make sure to convert \(S_w\) to \(\Sigma_w\) using Equation (5.124). Determine the steady-state average and steady-state covariance matrix of the state vector \(s\). Feel free to use the MATLAB function ‘dlyap’.

(iv) Plot the discrete-time EDOR ellipse corresponding to the first and third state variables. Use the MATLAB command ‘hold on’ to put the two ellipses on one figure, so you can compare them.

(v) Simulate the discrete-time process. Add the appropriate outputs to the EDOR plot and verify the scatter plot corresponds to the discrete-time ellipse.

(vi) Repeat parts (iii), (iv) and (v) with \(\Delta t = 0.5\).

5.12. Consider the continuous-time process of Exercise 5.11 with a first order shaping filter:

\[
\dot{s} = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
-1 & -2 & -3 & 1 \\
0 & 0 & 0 & a
\end{bmatrix} s + \begin{bmatrix}
0 \\
0 \\
a \\
g
\end{bmatrix} p
\]

where \(p(t)\) is a stationary white noise process with a mean of \(\bar{p}\) a power spectral density \(S_w\).

(i) Determine values of \(a\), \(g\), \(\bar{p}\) and \(S_w\) such that the output of the shaping filter has the following statistics: a mean of 10, a variance of 0.1 and a correlation time of 0.01 time units.

(ii) Repeat parts (i)-(v) of Exercise 5.11.

(iii) Repeat parts (i) and (ii) of this problem with a correlation time of 10 time units.
Chapter 6

Linear Estimation

In this chapter the state observer notions of Section 4.2 are extended to the stochastic framework. The development of a state estimator requires the introduction of a measurement information structure. If the process is linear and in deviation variable form, then the system of interest is:

\[
\dot{x} = Ax + Gw \\
y = Cx + v
\]  

(6.1)  
(6.2)

where (6.2) is the measurement equation. The term \(v\) is the measurement noise, assumed to be Gaussian white noise with zero mean and spectral density \(S_v\). The notion of Partial State Information (PSI) is that the state, \(x(t)\), is unavailable to the user. In addition, the state is under the influence of the stochastic disturbances, \(w\), and as such the state will change in a stochastic fashion over time. Thus, the objective of a state estimator is to use the measured data, \(y\), to track the ever changing values of \(x\). As one would expect the quality of the measurements (inversely proportional to the level of corruption or the spectral density of the noise, \(S_v\)) along with the size of the disturbances (indicated by \(S_w\)) will influence the performance one should expect from a state estimator.

6.1 State Estimation in Continuous-time

A state estimator is defined by the following linear state space process:

\[
\dot{\hat{x}} = A\hat{x} + K(y - C\hat{x})
\]  

(6.3)

where \(\hat{x}(t)\) is the state estimate and \(K\) is the estimator gain. The basic idea is that data collected from the measurement \(y\) will be used to drive \(\hat{x}(t)\) and ultimately track \(x(t)\). Based on (6.3), the design parameters are contained in the estimator gain matrix, \(K\). To guide the selection of this matrix, a quantitative measure of estimator performance must be developed. To this end the following error signal is defined: \(e(t) = x(t) - \hat{x}(t)\). Then, by combining (6.3) with (6.1)-(6.2), the time evolution of the error signal can be characterized by the following linear system:

\[
\dot{e} = (A - KC)e + Gw - Kv
\]  

(6.4)

The covariance of the estimation error is calculated by applying equation (5.112):

\[
\dot{\Sigma}_e = (A - KC)\Sigma_e + \Sigma_e(A - KC)^* + GS_wG^* + KS_vK^*
\]  

(6.5)
where it has been assumed that the process disturbance, \( w \), and the measurement noise, \( v \), are independent of each other. The interpretation is that achievement of a small \( \Sigma_e \) will be an indication of good performance. Thus, the first requirement is to select \( K \) such that \((A - KC)\) is stable, otherwise the error signal and its covariance matrix will become unbounded. In Chapter 4, it was concluded that there exists a matrix \( K \) such that \((A - KC)\) is stable if and only if the pair \((A, C)\) is detectable. As such, it will be assumed that the pair \((A, C)\) is detectable, which will guarantee that at least one stabilizing estimator gain exists. Using such a gain, the covariance matrix characterized by (6.5) will converge to a steady state and satisfy:

\[
0 = (A - KC)\Sigma_e + \Sigma_e(A - KC)^* + GS_wG^* + KS_vK^*
\]  

(6.6)

Example 6.1. Consider a scalar process \( \dot{x} = ax + gw \), with \( a = -0.1 \) and \( g = 1 \). Assume \( w \) is a zero-mean white noise process with spectral density \( S_w = 1 \). Furthermore, define the measurement equation as \( y = cx + v \), where \( c = 1 \) and \( S_v = 4 \).

Figure 6.1 compares the simulated state with the state estimate for various values of the estimator gain \( K \). The top plot of Figure 6.2 shows the error between the simulated state and the estimate for the same three cases. This plot along with those of Figures 6.1 it is observed that \( K = 0.5 \) yields the smallest estimation error of the three cases.

In this scalar example, the steady-state estimation error variance of Equation (6.6) can
be calculated analytically:

\[ \Sigma_e = -\left( g^2 S_w - K^2 S_v \right) / (2(a - Kc)) \]  

(6.7)

Using this relation the bottom plot of Figure 6.2 is constructed. From this plot, it is clear that a minimum exists around 0.4. At \( K = 0.1 \) and 1.5 the estimation error variance increases substantially, which supported by the other plots of the example.

Example 6.1 clearly illustrates a criterion for selecting the estimator gain. That is, select a value such that the estimation error variance is minimized. To illustrate the procedure of identifying such a gain, let us again consider a scalar version of Equation (6.6):

\[ 0 = 2(a - Kc) \Sigma_e + g^2 S_w + K^2 S_v \]  

(6.8)

Then, one can apply an implicit derivative to Equation (6.8):

\[ \frac{d}{dK} \left( 0 \right) = \frac{d}{dK} \left( 2(a - Kc) \Sigma_e + g^2 S_w + K^2 S_v \right) \]

\[ 0 = 2(a - Kc) \frac{d \Sigma_e}{dK} + 2 \Sigma_e \frac{d(a - Kc)}{dK} + 0 + 2K S_v \]  

(6.9)

But then we are looking for the point at which \( d \Sigma_e / dK = 0 \), which gives \( 0 = -2 \Sigma_e c + 2K S_v \) and ultimately \( K = \Sigma_e c / S_v \). If this gain is substituted into Equation (6.8), then one arrives at:

\[ 0 = 2(a - \left( \Sigma_e c / S_v \right) c) \Sigma_e + g^2 S_w + \left( \Sigma_e c / S_v \right)^2 S_v \]  

\[ 0 = 2a \Sigma_e + g^2 S_w - \Sigma_e c^2 / S_v \]  

(6.10)
Thus, the procedure for optimal estimation is to solve Equation (6.10) for $\Sigma_e$, substitute into $K = \Sigma_e C / S_v$ and apply this gain to the estimator process of Equation (6.3).

In the non-scalar, non-steady-state case, the procedure is identical, but the formulas (and their derivation) are a bit more complicated. The optimal state estimate, $\hat{x}(t)$, is the state of the optimal state estimator - also known as the Kalman Filter - and is driven by measurement signal, $y$.

$$\dot{\hat{x}} = A\hat{x} + K(y - C\hat{x}) \quad (6.11)$$

where $K$ is the Kalman gain

$$K = \Sigma_e C^* S_v^{-1} \quad (6.12)$$

and $\Sigma_e$ is the solution to the Riccati equation

$$d\Sigma_e / dt = A\Sigma_e + \Sigma_e A^* + G S_w G^* - \Sigma_e C^* S_v^{-1} C \Sigma_e$$

with $\Sigma_e(0) = \Sigma_e(0)$. It is interesting to note that the estimation error covariance matrix, $\Sigma_e(t)$, can be calculated prior to the collection of the measurement signal, $y$. In addition, if any of the matrix parameters, $A, G, C, S_w, S_v$ change with time, then Equations (6.11 - 6.13) will remain valid as the optimal estimate.

In most cases, the matrix parameters will not change with time and Equation (6.13) will reach a steady-state condition rather quickly. In such cases, one will be interested in the following Algebraic Riccati Equation (ARE):

$$0 = A\Sigma_e + \Sigma_e A^* + G S_w G^* - \Sigma_e C^* S_v^{-1} C \Sigma_e$$

(6.14)

However, there are cases for which Equation (6.13) does not converge and one should not expect the ARE of Equation (6.14) to have a positive definite solution. The following theorem provides conditions for convergence through the existence of a positive definite solution to the ARE.

**Theorem 6.1.** If $(A, C)$ detectable and $(A, G)$ stabilizable, then the differential equation (6.13) will converge to a positive definite $\Sigma_e$ and use of (6.12) will result in the matrix $(A - KC)$ being stable.

The proof of this Theorem can be found in [103]. To get a feel for the proof, consider the following (easily verified) alternate form of the ARE:

$$(A - KC)\Sigma_e + \Sigma_e (A - KC)^* = -G S_w G^* - \Sigma_e C^* S_v^{-1} C \Sigma_e$$

(6.15)

Then, recall the discussion of Section 5.7 (especially Theorem 5.1), which gives the relation between $(A, G)$ stabilizable and a positive definite solution to the Lyapunov equation. This along with Definition 4.6, which states that $(A, C)$ must be detectable for there to exist a stable $A - KC$, indicates the necessity of these two conditions.
6.2 • Estimation Theory

Having outlined the optimal state estimator in the continuous-time framework, we now take a step back and review some fundamental concepts from the more general field of estimation theory. These foundations will then be used in a more detailed derivation of the discrete-time Kalman filter. In addition to providing the reader with an alternate formulation of the optimal estimator, a number of technical results of great utility to subsequent chapters will be presented.

6.2.1 • The Bayesian Approach

Consider two vector valued random variables, \( x \) and \( y \), where \( x \sim n_x \times 1 \) and \( y \sim n_y \times 1 \). It is emphasized that \( x \) and \( y \) are not stochastic processes. However, one could think of these random variables as a specific time instant taken from two stochastic processes. Assume the probability density functions \( p(x) \) and \( p(y) \) are known as well as the joint probability density function \( p(x, y) \). The relationship between \( p(x) \), \( p(y) \) and \( p(x, y) \) are as follows:

\[
p(x) = \int_{\mathbb{R}^n_x} p(x, y) \, dy \quad \text{and} \quad p(y) = \int_{\mathbb{R}^n_y} p(x, y) \, dx
\]

(6.16)

Furthermore, assume the random variable \( y \) is measured. Then, the central question of estimation theory is to construct some function of this measurement as an estimator of the unknown random variable \( x \). We will denote an arbitrary estimator as a function \( f(\bullet) \).

To guide the selection of \( f(\bullet) \), one must identify some measure of performance. One approach is to define the estimation error as \( e = x - f(y) \) and then seek to minimize the covariance matrix of this error \( \Sigma_e = E[(x - f(y))(x - f(y))^*] \). Thus, the Minimum Mean Square Error (MMSE) problem is defined as

\[
\min_{f(\bullet)} \{ E[(x - f(y))^*(x - f(y))] \} = \min_{f(\bullet)} \{ Tr(\Sigma_e) \}
\]

(6.17)

where the expectation is over both random variables \( x \) and \( y \). To be more specific, if \( \Sigma_e(f) = E[(x - f(y))(x - f(y))^*] \) and \( f^*(\bullet) \) is the solution to the MMSE problem then

\[
\Sigma_e(f) - \Sigma_e(f^*) \geq 0
\]

(6.18)

for all \( f(\bullet) \), where \( \geq \) indicates that the left side is positive semi-definite.

Before giving the solution to (6.17), some additional notation will be needed. Define the conditional density function of \( x \) given \( y \) as

\[
p(x \mid y) \triangleq \frac{p(x, y)}{p(y)}
\]

(6.19)

This density function is proper in the sense that \( \int p(x \mid y) \, dx = 1 \). Given the conditional density function, one can define the conditional mean as:

\[
E[x \mid y] = \int_{\mathbb{R}^n_x} x \, p(x \mid y) \, dx
\]

(6.20)

The following theorem illustrates a fundamental property of the conditional mean and will be of great utility in subsequent analysis.
Theorem 6.2. Let \( f(y) \) be an arbitrary function of \( y \). Then, the following will hold:

\[
E \left[ f(y)(x - E[x \mid y])^* \right] = 0
\]  
(6.21)

Proof.

\[
E[f(y)x^*] = \int_{\mathbb{R}^n} \int_{\mathbb{R}^m} f(y)x^* p(x,y) dx dy \tag{6.22}
\]

and

\[
E[f(y)E[x^* \mid y]] = \int_{\mathbb{R}^n} \int_{\mathbb{R}^m} f(y)E[x^* \mid y] p(x,y) dx dy
\]

\[
= \int_{\mathbb{R}^n} f(y)E[x^* \mid y] \int_{\mathbb{R}^m} p(x,y) dx dy
\]

\[
= \int_{\mathbb{R}^n} f(y)E[x^* \mid y] p(y) dy
\]

\[
= \int_{\mathbb{R}^n} f(y) \left( \int_{\mathbb{R}^m} x^* p(x \mid y) dx \right) p(y) dy
\]

\[
= \int_{\mathbb{R}^n} \int_{\mathbb{R}^m} f(y)x^* p(x \mid y) p(y) dx dy
\]

\[
= \int_{\mathbb{R}^n} \int_{\mathbb{R}^m} f(y)x^* p(x,y) dx dy
\]

Thus, (6.22) is equal to (6.23). \( \square \)

Corollary 6.1. If \( \hat{x}(y) \overset{\wedge}{=} E[x \mid y] \), then \( E[\hat{x}(y)] = E[x] \).

Proof. Let \( f(y) = 1 \) in Theorem 6.2. \( \square \)

Theorem 6.2 is frequently denoted as the Orthogonality Theorem. This is due to the fact that the conditional mean is such that its estimation error is orthogonal to all possible estimators, \( f(y) \). Corollary 6.1 indicates that the conditional mean, \( \hat{x}(y) \), is an unbiased estimator. Thus, if the estimation error is defined as \( e = x - \hat{x}(y) \), then the expected value of the estimation error is zero, i.e., \( E[e] = 0 \).

Theorem 6.3. The solution to problem (6.17) is \( \hat{x}(y) \overset{\wedge}{=} E[x \mid y] \). Furthermore, the minimum value of the objective function is:

\[
\Sigma_e \overset{\wedge}{=} E[(x - \hat{x}(y))(x - \hat{x}(y))^*] = E[xx^*] - E[\hat{x}(y)\hat{x}(y)^*]
\]  
(6.24)
6.2. Estimation Theory

Proof. Consider an arbitrary estimator $f(y)$. Then,

$$
\Sigma_e(f) = E[(x - f(y))(x - f(y))^*]
$$

$$
= E[(x + \hat{x}(y) - \hat{x}(y) - f(y))(x + \hat{x}(y) - \hat{x}(y) - f(y))^*]
$$

$$
= E[(x - \hat{x}(y))(x - \hat{x}(y))^*] + E[(x - \hat{x}(y))(\hat{x}(y) - f(y))^*]
$$

$$
+ E[(\hat{x}(y) - f(y))(x - \hat{x}(y))^*] + E[(\hat{x}(y) - f(y))(\hat{x}(y) - f(y))^*]
$$

$$
= \Sigma_e + E[(\hat{x}(y) - f(y))(\hat{x}(y) - f(y))^*]
$$

Then, since $E[(\hat{x}(y) - f(y))(\hat{x}(y) - f(y))^*]$ must be positive semi-definite, the inequality of (6.18) is concluded. Equation 6.24 can then be verified by using the following identity:

$$
E[\hat{x}(y)x^*] = E[\hat{x}(y)(x - \hat{x}(y) + \hat{x}(y))^*] = E[\hat{x}(y)\hat{x}(y)^*]
$$

(6.25)

An alternate method of selecting an estimator is the maximum likelihood approach. Such an estimator is frequently denoted as the Maximum Likelihood Estimator (MLE). Assume again the joint probability density function $p(x, y)$ is given and the random variable $y$ is measured to a value $Y$, the MLE is the solution to

$$
\max_x \{ p(x, Y) \}
$$

(6.26)

A necessary condition for the solution to this problem is

$$
\frac{\partial}{\partial x} \{ p(x, Y) \} = 0
$$

(6.27)

In many cases, it will be more convenient to search for $x$ such that

$$
\frac{\partial}{\partial x} \{ \ln(p(x, Y)) \} = 0
$$

(6.28)

The MLE has intuitive appeal in the sense that it is natural to select the estimate as the value of $x$ that has the highest probability for a given value of $y = Y$.

6.2.2 Estimation with Gaussian Random Variables

The previous sub-section made no assumptions with regard to the joint density function $p(x, y)$. Unfortunately, in most cases the calculation of $E[x \mid y]$ will require one to employ numeric methods that are likely to be computationally intensive, especially if $x$ and $y$ are of large dimension. One exception is if $x$ and $y$ are jointly Gaussian. Under such an assumption the joint density function is completely specified by the first and second order statistics. Specifically, if $z = [x^* y^*]^*$, the mean of $z$ is denoted as $\bar{z} = [\bar{x}^* \bar{y}^*]^*$ and the mean removed covariance matrix of $z$ is

$$
\Sigma_z = E[(z - \bar{z})(z - \bar{z})^*] = \begin{bmatrix}
\Sigma_x & \Sigma_{xy} \\
\Sigma_{yx} & \Sigma_y
\end{bmatrix}
$$

(6.29)
then the joint density function is

\[
p(z) = \frac{|\Sigma_x|^{-\frac{1}{2}}}{(\sqrt{2\pi})^{n_x+n_y}} e^{-\frac{1}{2}(x-z)\Sigma_x^{-1}(x-z)^T}
\]  \hspace{1cm} (6.30)

This particular form of the joint density function possesses a number of convenient properties. Consider the relationship between independent and uncorrelated random variables. Two random variables, \(x\) and \(y\), are said to be independent if \(p(x, y) = p(x)p(y)\) and uncorrelated if \(E[(x - \bar{x})(y - \bar{y})] = 0\). While independence is easily shown to imply zero correlation, the reverse is not true for generic random variables. In the case of Gaussian random variables, \(\Sigma_{xy} = 0\) implies \(|\Sigma_x| = |\Sigma_x||\Sigma_y|\) and

\[
\Sigma_x^{-1} = \begin{bmatrix}
\Sigma_x^{-1} & 0 \\
0 & \Sigma_y^{-1}
\end{bmatrix}
\]  \hspace{1cm} (6.31)

Thus,

\[
p(z) = \frac{|\Sigma_x|^{-\frac{1}{2}}}{(\sqrt{2\pi})^{n_x}} e^{-\frac{1}{2}(x-z)\Sigma_x^{-1}(x-z)^T} \times \frac{|\Sigma_y|^{-\frac{1}{2}}}{(\sqrt{2\pi})^{n_y}} e^{-\frac{1}{2}(y-\bar{y})\Sigma_y^{-1}(y-\bar{y})^T} = p(x)p(y)
\]  \hspace{1cm} (6.32)

Another convenient property of a Gaussian distribution is that the conditional density, \(p(x \mid y)\), is always Gaussian. To see this, first note that \(|\Sigma_y| = |\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T||\Sigma_y|\). In addition, the following identity is easily verified.

\[
\Sigma_x^{-1} = \begin{bmatrix}
I - \Sigma_y^{-1}\Sigma_{xy} & 0 \\
-\Sigma_y^{-1}\Sigma_{xy} & I
\end{bmatrix} \begin{bmatrix}
(\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T)^{-1} & I \\
0 & \Sigma_y^{-1}
\end{bmatrix} \begin{bmatrix}
I & -\Sigma_x\Sigma_y^{-1} \\
0 & I
\end{bmatrix}
\]  \hspace{1cm} (6.33)

Thus,

\[
p(x, y) = \frac{|\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T||\Sigma_y|}{(\sqrt{2\pi})^{n_x}(\sqrt{2\pi})^{n_y}} e^{-\frac{1}{2}(x-\bar{x})\left((\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T)^{-1}(x-\bar{x})\right)^T} \times e^{-\frac{1}{2}(y-\bar{y})\Sigma_y^{-1}(y-\bar{y})^T} p(y)
\]  \hspace{1cm} (6.34)

where \(\hat{x} = \bar{x} + \Sigma_{xy}\Sigma_y^{-1}(y - \bar{y})\). Since the conditional probability density is defined as \(p(x \mid y) = p(x, y) / p(y)\) it is concluded that

\[
p(x \mid y) = \frac{|\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T|^{-\frac{1}{2}}}{(\sqrt{2\pi})^{n_x}} e^{-\frac{1}{2}(x-\bar{x})\left((\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T)^{-1}(x-\bar{x})\right)^T} p(y)
\]  \hspace{1cm} (6.35)

From this density function it is easily concluded that the conditional mean, \(E[x \mid y]\), is:

\[
\hat{x}(y) = \bar{x} + \Sigma_{xy}\Sigma_y^{-1}(y - \bar{y})
\]  \hspace{1cm} (6.36)
Returning to Equation (6.24), the above indicates that

\[
E[xx^*] = E[(x - \hat{x})(x - \hat{x})^*] = \Sigma_x + \hat{x}\hat{x}^* \tag{6.37}
\]

\[
E[\hat{x}(y)\hat{x}(y)^*] = E[(\hat{x} - \Sigma_{xy}\Sigma_y^{-1}(y - \hat{y}))((\hat{x} - \Sigma_{xy}\Sigma_y^{-1}(y - \hat{y}))^*)]
= \hat{x}\hat{x}^* + \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^* \tag{6.38}
\]

\[
\Sigma_e \hat{x} \overset{\Delta}{=} E[(x - \hat{x})(x - \hat{x})^*] = \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^* \tag{6.39}
\]

**Example 6.2.** Consider two jointly Gaussian random variables \(x\) and \(y\) with statistics \(\hat{z} = [\hat{x}\hat{y}]^\top = \begin{bmatrix} 2 & 15 \end{bmatrix}^\top\) and

\[
\Sigma_x = E[(z - \hat{z})(z - \hat{z})^\top] = \begin{bmatrix}
\Sigma_x & \Sigma_{xy} \\
\Sigma_{xy} & \Sigma_y
\end{bmatrix} = \begin{bmatrix}
1 & 2 \\
2 & 10
\end{bmatrix}
\]

Then, the optimal estimator is

\[
\hat{x}(y) \overset{\Delta}{=} E[x \mid y] = \hat{x} + \Sigma_{xy}\Sigma_y^{-1}(y - \hat{y}) = \begin{bmatrix}
2 \\
15
\end{bmatrix} + \begin{bmatrix}
1 \\
5
\end{bmatrix} \frac{1}{25}(y - 3)
\]

and the error covariance of this estimate is

\[
\Sigma_e = \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^* = \begin{bmatrix}
1 & 2 \\
2 & 10
\end{bmatrix} - \begin{bmatrix}
1 \\
5
\end{bmatrix} \frac{1}{25} \begin{bmatrix}
1 \\
5
\end{bmatrix} = \begin{bmatrix}
0.96 & 1.8 \\
1.8 & 9
\end{bmatrix}
\]

If \(Y = 8\), then \(E[x \mid Y = 8] = \hat{x}(8) = \begin{bmatrix}
2 \\
15
\end{bmatrix} + \begin{bmatrix}
1 \\
5
\end{bmatrix} \frac{1}{25}(8 - 3) = \begin{bmatrix}
2.2 \\
16
\end{bmatrix}
\]

Now return to the MLE, where the goal is find \(x\) such that \(\frac{\partial}{\partial x} \{\ln(p(x \mid Y))\} = 0\). Using (6.35) it is found that

\[
0 = \frac{\partial}{\partial x} \{\ln(p(x \mid Y))\} = \frac{\partial}{\partial x} \left\{-\frac{1}{2}(x - \hat{x})(\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^*)^{-1}(x - \hat{x})^*\right\}
= -(x - \hat{x})(\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^*)^{-1}
\]

(6.40)

Since \((\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^*)^{-1}\) must be full rank, it is found that the MLE must be such that \(x \overset{\Delta}{=} \hat{x}\) and thus the conditional mean is also the MLE in the case of Gaussian random variables.

6.2.3 - Data Reconciliation

Consider a vector, \(s\), indicating material flows within a chemical plant. Due to material balance constraints, the vector must satisfy a set of linear equalities, which can be stated in the following form: \([M - I]s = 0\). In addition, one may define a subset of the vector \(s\) as random variables, \(x = [I \ 0]s\), where the dimension of \(x\) is equal to the number of material balance equality constraints. Then, one will arrive at the following: \(s = Cx\), where \(C = [I \ M^\top]\). The challenge of data reconciliation is to estimate the material flow vector, \(s\), given noise corrupted measurements of these flows.

\[
y = s + v = Cx + v \tag{6.41}
\]
where \( x \) and \( v \) are assumed to be Gaussian random variables. In addition, \( v \) is assumed to be zero mean and independent of \( x \). From (6.41) one concludes that \( \hat{y} = C\hat{x}, \Sigma_y = C\Sigma_x C^* + \Sigma_v \) and \( \Sigma_{xy} = \Sigma_x C^* \). Thus, \( \hat{s} = C\hat{x} \) where from Equation (6.36) and (6.39) we find

\[
\hat{x} = \hat{x} + \Sigma_x C^*(C\Sigma_x C^* + \Sigma_v)^{-1}(y - C\hat{x}) \tag{6.42}
\]

\[
\Sigma_x = \Sigma_x - \Sigma_x C^*(C\Sigma_x C^* + \Sigma_v)^{-1}C\Sigma_x \tag{6.43}
\]

In many cases, one would like to assume a complete absence of knowledge with respect to \( x \), i.e., assume \( \Sigma_x \to \infty \). To address this case, an alternate form of the estimator must be determined. This conversion requires use of the Matrix Inversion Lemma (MIL):

\[
A - AC^*(CAC^* + B)^{-1}CA = (A^{-1} + C^*B^{-1}C)^{-1} \tag{6.44}
\]

Application of the MIL to (6.43) directly leads to

\[
\Sigma_x = (\Sigma_x^{-1} + C^*\Sigma_v^{-1}C)^{-1} \tag{6.45}
\]

Then, application of the MIL to \((C\Sigma_x C^* + \Sigma_v)^{-1}\) results in:

\[
\Sigma_x C^*(C\Sigma_x C^* + \Sigma_v)^{-1} = \Sigma_x C^* \left[ \Sigma_v^{-1} - \Sigma_v^{-1}C(C\Sigma_v^{-1}C^* + \Sigma_x^{-1})^{-1}C\Sigma_v^{-1} \right] \tag{6.46}
\]

\[
= \Sigma_x \left[ I - C^*\Sigma_v^{-1}C(C\Sigma_v^{-1}C^* + \Sigma_x^{-1})^{-1}C\Sigma_v^{-1} \right] C\Sigma_v^{-1}
\]

\[
= \Sigma_x \left[ \Sigma_v^{-1}C - C^*\Sigma_v^{-1}C\Sigma_v^{-1} \right] C\Sigma_v^{-1}
\]

\[
= \Sigma_x \left[ (\Sigma_x^{-1} + C^*\Sigma_v^{-1}C) - C^*\Sigma_v^{-1}C \right] \Sigma_x C^*\Sigma_v^{-1}
\]

\[
= \Sigma_x C^*\Sigma_v^{-1} \tag{6.47}
\]

Thus, an alternate form of the optimal estimator is

\[
\hat{x} = \hat{x} + \Sigma_x C^*\Sigma_v^{-1}(y - C\hat{x}) \tag{6.48}
\]

Given this alternate form, one may now take the limit \( \Sigma_x \to \infty \), to find \( \Sigma_x = (C^*\Sigma_v^{-1}C)^{-1} \). Notice that the resulting estimator, \( \hat{x} = \Sigma_x C^*\Sigma_v^{-1}y \), does not utilize \( \hat{x} \), which of course is likely to be unknown if \( \Sigma_x \to \infty \).

In practice, the data reconciliation problem is usually solved numerically due to a desire to enforce element-wise inequality constraints: \( \hat{s}^{\text{min}} \leq \hat{s} \leq \hat{s}^{\text{max}} \). To arrive at a numeric procedure the data reconciliation problem is stated as follows:

\[
\min_{\hat{x}} \left\{ (y - C\hat{x})^\top \Sigma_v^{-1}(y - C\hat{x}) \right\} \tag{6.49}
\]

s.t. \( \hat{s}^{\text{min}} \leq C\hat{x} \leq \hat{s}^{\text{max}} \)

To illustrate the equivalence, we will solve this problem ignoring (6.49):

\[
0 = \frac{\partial}{\partial \hat{x}} \left\{ (y - C\hat{x})^\top \Sigma_v^{-1}(y - C\hat{x}) \right\} = 2\Sigma_v^{-1}(y - C\hat{x}) \tag{6.50}
\]

Multiplying by \( C^* \) and then \((C^*\Sigma_v^{-1}C)^{-1}\), yields \( \hat{x} = (C^*\Sigma_v^{-1}C)^{-1}C^*\Sigma_v^{-1}y \), the desired result.
Example 6.3. Consider the process flow sheet of Figure 6.3. Material balances indicate that \( s_4 = s_1 + s_2 \) and \( s_5 = s_1 + s_2 + s_3 \). If \( s = [s_1 \ s_2 \ s_3 \ s_4 \ s_5]^T \), then

\[
M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}
\]

If \( y = Cx + v \) and \( v \) is a Gaussian random variable with zero mean and a variance

\[
\Sigma_v = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix}
\]

then

\[
\Sigma_e = (C^*\Sigma_v^{-1}C)^{-1}
\]

\[
= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}^{-1}
\]

\[
= \begin{bmatrix} 0.787 & -0.426 & -0.033 \\ -0.426 & 1.15 & 0.066 \\ -0.033 & -0.066 & 0.918 \end{bmatrix}
\]
and the optimal estimator, $E[x \mid y]$, is

$$
\hat{x}(y) = (C^*\Sigma^{-1}_v C)^{-1} C^*\Sigma^{-1}_v y
$$

$$
= \begin{bmatrix}
0.787 & -0.426 & -0.033 \\
-0.426 & 1.15 & 0.066 \\
-0.033 & -0.066 & 0.918
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0
\end{bmatrix}^* \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.5 \\
0 & 0 & 0 & 0 & 0.1
\end{bmatrix}^* y
$$

If $y = [y_1, y_2, y_3, y_4, y_5]^* = [6.2, 3.75, 2.05, 11.1, 19]^*$, then $\hat{x}(y) = [6.64, 4.62, 2.57]^*$ and $\hat{s} = C\hat{x} = [6.64, 4.62, 2.57, 11.3, 13.8]^*$. It is interesting to note that if one of the measurements did not exist, then one could communicate this to the estimator by setting the variance of that measurement to infinity. However, if too many streams are unmeasured then the system will be unobservable. For example, if streams 4 and 5 were unmeasured then the optimal estimator would degenerate to the original material balance equations.

$$
\hat{s}(y) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0
\end{bmatrix} y
$$

If 3 streams were unmeasured then, $(C^*\Sigma^{-1}_v C)^{-1}$ would fail to exist, since there would be too many degrees of freedom.

### 6.3 - The Discrete-time Kalman Filter

Now that the fundamentals of estimation theory have been exposed, the development of the discrete-time Kalman filter is relatively straightforward. The approach is to simply apply the previously derived conditional mean and minimum error covariance calculations:

$$
\dot{x}(y) = \bar{x} + \Sigma_{xy} \Sigma_y^{-1} (y - \bar{y}) \tag{6.36}
$$

$$
\Sigma_x = \Sigma_x - \Sigma_{xy} \Sigma_y^{-1} \Sigma_{xy} \tag{6.39}
$$

The only additional concept is that of one-step prediction.

#### 6.3.1 - Evolution of Estimates and Predictions

Consider a discrete-time process with an associated measurement equation:

$$
x_{k+1} = A_d x_k + B_d u_k + G_d w_k \tag{6.51}
$$

$$
y_k = C x_k + v_k \tag{6.52}
$$

Assume that each step of input sequence, $u_k$, is known at or before time $k$. Also, assume $w_k$ and $v_k$ are Gaussian zero mean white noise processes that are independent of each
other and the initial condition. At time zero the statistics of the initial condition, \(x_0\), are \(E[x_0] = \hat{x}_0\) and \(E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] = \Sigma_{x,0}\). If we are given a measurement at time zero, \(y_0\), then the statistics of this measurement are:

\[
\hat{y}_0 = C\hat{x}_0 \\
\Sigma_{y,0} = E[(y_0 - \hat{y}_0)(y_0 - \hat{y}_0)^T] \\
= E[(C(x_0 - \hat{x}_0) + v)(C(x_0 - \hat{x}_0) + v)^T] \\
= C\Sigma_{x,0}C^* + \Sigma_v \\
\Sigma_{xy,0} = E[(x_0 - \hat{x}_0)(y_0 - \hat{y}_0)^T] \\
= E[(x_0 - \hat{x}_0)(C(x_0 - \hat{x}_0) + v)^T] \\
= \Sigma_{x,y}C^*
\]

Thus, the optimal estimate of \(x_0\) given \(y_0\), \(\hat{x}_0 = E[x_0 \mid y_0]\), is calculated from (6.36) as:

\[
\hat{x}_0 = \hat{x}_0 + \Sigma_{x,0}C^*(C\Sigma_{x,0}C^* + \Sigma_v)^{-1}(y_0 - C\hat{x}_0) \\
\]

Note the similarity with Equation (6.42). The resulting estimation error covariance is \(\Sigma_{e,0} = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T \mid y_0]\), which is calculated from (6.39) as:

\[
\Sigma_{e,0} = \Sigma_{x,0} - \Sigma_{x,0}C^* (C\Sigma_{x,0}C^* + \Sigma_v)^{-1}C\Sigma_{x,0} \\
\]

Note the similarity with Equation (6.43). At time \(k = 0\), a prediction of \(x_1\) may be made as \(\hat{x}_1 = E[x_1 \mid y_0]\), where

\[
\hat{x}_1 = A_d\hat{x}_0 + B_d u_0 \\
\]

The estimation error covariance of this prediction is defined as \(\Sigma_{e,1}^+ = E[e_1^+e_1^{*+}]\), where \(e_1^+ = x_1 - \hat{x}_1^+\), and is calculated to be

\[
\Sigma_{e,1}^+ = A_d\Sigma_{e,0}A_d^* + G_d\Sigma_v G_d^* \\
\]

Once the measurement at time \(k = 1\) arrives, we would like to calculate the optimal estimate of \(x_1\) given \(y_1\) and \(y_0\), \(\hat{x}_1 = E[x_1 \mid y_1, y_0]\). It is important to emphasize that in calculating the statistics to be used within (6.36) and (6.39), we must include the fact that the measurement \(y_0\) is available and has been used.

\[
\hat{x}_1 = E[x_1 \mid y_0] = \hat{x}_1^+ \\
\hat{y}_1 = E[y_1 \mid y_0] = C\hat{x}_1^+ \\
\Sigma_{x,1} = E[(y_1 - \hat{y}_1)(y_1 - \hat{y}_1)^T \mid y_0] \\
= E[(C(x_1 - \hat{x}_1^+ + v_1)(C(x_1 - \hat{x}_1^+ + v_1)^+ \mid y_0) \\
= C\Sigma_{e,1}^+ C^* + \Sigma_v \\
\Sigma_{xy,1} = E[(x_1 - \hat{x}_1)(y_1 - \hat{y}_1)^T \mid y_0] \\
= E[(x_1 - \hat{x}_1^+)(C(x_1 - \hat{x}_1^+) + v_1)^T \mid y_0] \\
= \Sigma_{e,1}^+ C^*
\]

Thus, the optimal estimate, \(\hat{x}_1\), is again calculated from (6.36):

\[
\hat{x}_1 = \hat{x}_1^+ + \Sigma_{e,1}^+ C^*(C\Sigma_{e,1}^+ C^* + \Sigma_v)^{-1}(y_1 - C\hat{x}_1^+) \\
\]
and the estimation error covariance, $\Sigma_{e,1} \triangleq E[(x_1 - \hat{x}_1)(x_1 - \hat{x}_1)^* | y_1, y_0]$, is again calculated from (6.39):

$$\Sigma_{e,1} = \Sigma_{e,1}^+ - \Sigma_{e,1}^+ C^* (C \Sigma_{e,1}^+ C^* + \Sigma_w)^{-1} C \Sigma_{e,1}^+$$

(6.65)

Continuing the procedure to a generic time $k$ results in the following statement of the discrete-time Kalman filter:

$$\hat{x}_k = \hat{x}_k^+ + \Sigma_{e,k}^+ C^* (C \Sigma_{e,k}^+ C^* + \Sigma_w)^{-1} (y_k - C \hat{x}_k)$$

(6.66)

$$\Sigma_{e,k} = \Sigma_{e,k}^+ + \Sigma_{e,k}^+ C^* (C \Sigma_{e,k}^+ C^* + \Sigma_w)^{-1} C \Sigma_{e,k}^+$$

(6.67)

$$\hat{x}_{k+1}^+ = A_d \hat{x}_k + B_d u_k$$

(6.68)

$$\Sigma_{e,k+1}^+ = A_d \Sigma_{e,k} A_d^* + G_d \Sigma_w G_d^*$$

(6.69)

$$\hat{x}_{e,0} = \hat{x}_0$$

(6.70)

As a matter of notion, $\hat{x}_k$ is the estimate and $\hat{x}_{k+1}^+$ is the one-step prediction, both based on measurements up to time $k$, $y_0, y_1, \ldots, y_k$. In addition, $\Sigma_{e,k}$ and $\Sigma_{e,k+1}^+$ are the corresponding estimation error covariance matrices (i.e., those associated with $e_k = x_k - \hat{x}_k$ and $e_{k+1}^+ = x_{k+1} - \hat{x}_{k+1}^+$).

**Example 6.4.** Consider a scalar process $\hat{x} = ax + gw$ and $y = cx + v$, with $a = -0.1$, $g = 1$ and $c = 1$. Assume $w$ and $v$ are a zero-mean white noise process with spectral densities $S_w = 1$ and $S_v = 4$. Applying sample-and-hold with $\Delta t = 0.5$, results in $a_d = 0.95$, $g_d = 0.49$, $\Sigma_w = 2$, $c = 1$ and $\Sigma_v = 8$. If $x_0 = 0$ and $\Sigma_{x,0} = 0.5$, then $\hat{x}_0^+ = 0$ and $\Sigma_{e,0}^+ = 0.5$. Now suppose that the actual initial condition, $x_0$, is -0.81, though you do not know this fact. However, you do know the measurement $y_0 = c x_0 + v_0 = -4.14$. Then the optimal estimate of the initial state is

$$\hat{x}_0 = 0 + 0.5(0.5 + 8)^{-1}(-4.14 - 0) = 0.0588(-4.14) = -0.244$$

It is highlighted that despite a fairly large negative value for the measurement, the estimate is fairly close to zero. This is due to the large size of $\Sigma_v$ in comparison to $\Sigma_{e,0}^+$, indicating to the optimal estimator that the measurement is fairly noisy. Also, note that if $\Sigma_{e,0}^+$ were equal to zero then the estimate would be $\hat{x}_0 = 0$, since $\Sigma_{e,0}^+ = 0$ indicates complete confidence in $x_0$. Finally, note that if $\Sigma_{e,0}^+$ were very large (indicating very little confidence in $\hat{x}_0$), then the estimate would be approximately equal to $y_0$. The error variance of the estimate is calculated as

$$\Sigma_{e,0} = 0.5 - 0.5(0.5 + 8)^{-1}0.5 = 0.47$$

Finally, the prediction of the state at time $k = 1$ is

$$\hat{x}_1 = a_d \hat{x}_0 = 0.95(-0.244) = -0.232$$

and the error variance of the one-step prediction is

$$\Sigma_{e,1}^+ = 0.95(0.47)0.95 + 0.49(2)0.49 = 0.904$$

Now suppose $w_0$ is 1.68, which makes $x_1 = 0.95(-0.81) + 0.49(1.68) = 0.0537$ and further assume $v_1 = 3.36$, all of which you do not know. What you do know is that $y_1 = 0.0537 + 3.36 = 3.414$. Then,

$$\hat{x}_1 = -0.232 + 0.904(0.904 + 8)^{-1}(3.414 - (-0.232)) = 0.138$$
Since $\Sigma_{e,1}^+$ has increased compared to $\Sigma_{e,0}^+$, the estimate puts more weight to the measurement $y_1$, or more appropriately more weight to $y_1 - C\hat{x}_1^+$. The error variance of this estimate is

$$\Sigma_{e,1} = 0.904 - 0.904(0.904 + 8)^{-1}0.904 = 0.822$$

It is highlighted that the sequences $\Sigma_{e,k}^+$ and $\Sigma_{e,k}$ can be calculated for all values of $k$, prior to the realizations of $y_k$. That is, the error variances, $\Sigma_{e,k}$ and $\Sigma_{e,k}^+$, are known before any measurement is taken, assuming the statistics of $w_k$ and $v_k$ are known.

To arrive at a more compact form of the Kalman filter, begin by noting that application of the MIL to (6.67) gives

$$\Sigma_{e,k} = ((\Sigma_{e,k}^+)^{-1} + C^*\Sigma_{v}^{-1}C)^{-1}$$

Then, application of the MIL to $(C\Sigma_{e,k}^+ C^* + \Sigma_{v})^{-1}$ results in:

$$\Sigma_{e,k}^+ C^*(C\Sigma_{e,k}^+ C^* + \Sigma_{v})^{-1}$$

Using these two identities in (6.66) and (6.67), followed by substitution of (6.68) and (6.69) into (6.66) and (6.67) results in the following optimal estimator form of the Kalman filter:

$$\dot{\hat{x}}_k = A_d \hat{x}_{k-1} + B_d u_{k-1} + K_k (y_k - C(A_d \hat{x}_{k-1} + B_d u_{k-1}))$$

$$K_k = \Sigma_{e,k} C^*\Sigma_{v}^{-1}$$

$$\Sigma_{e,k} = ((A_d \Sigma_{e,k-1} A_d^* + G_d \Sigma_w G_d^*)^{-1} + C^*\Sigma_{v}^{-1}C)^{-1}$$

The filter can also be stated with respect to only the predictions. Substitution of (6.66) and (6.67) into (6.68) and (6.69) results in the following one-step predictor form of the Kalman filter:

$$\dot{\hat{x}}_{k+1}^+ = A_d \hat{x}_k^+ + B_d u_k + K_k^+(y_k - C \hat{x}_k^+)$$

$$K_k^+ = A_d \Sigma_{e,k}^+ C^*(C\Sigma_{e,k}^+ C^* + \Sigma_{v})^{-1}$$

$$\Sigma_{e,k+1}^+ = A_d \left(\Sigma_{e,k}^+ - \Sigma_{e,k} C^*(C\Sigma_{e,k}^+ C^* + \Sigma_{v})^{-1}C\Sigma_{e,k}^+\right) A_d^* + G_d \Sigma_w G_d^*$$

The existence of two forms of the discrete-time Kalman filter is a common source of confusion to those new to the subject. If the objective is to only monitor the process, then the optimal estimator form is likely the best choice since it will have the lowest estimation error. However, if the objective is to use the reconstructed state within a feedback controller, then it is likely that the one-step predictor form will be required. This is due to the fact that the controller is likely to possess computational delay. Due
to this delay, the controller will need the reconstructed state one time step before the control action is to be implemented. In such a case, it is better to give the controller a prediction of the state, which is associated with the time instant that the control action will be implemented. In subsequent chapters, we will analyze (and design) controllers with and without the assumption of computational delay, which will require both forms of the discrete-time Kalman filter.

![Figure 6.4](image)

**Figure 6.4.** All plots are for Example 6.5 with $\Sigma_v = 8$. Top: Comparison of true state and measurements. Middle: Comparison of true state and the optimal estimates. Bottom: Comparison of estimation errors of the Kalman filter with the two standard deviation envelope of the error variance.

**Example 6.5.** Continuing Example 6.4, a time-series realization of the (unknown) state and its measurement are given in the top plot of Figure 6.4. If one applies (6.72), then the state estimates of the 2nd plot in Figure 6.4 will result. The 3rd plot of Figure 6.4 illustrates the (unknown) estimation error signal. As a reference, the 2 standard deviation envelope determined from (6.74) is also included in Figure 6.4. It is noted that $\Sigma_{e,k}$ quickly reaches a steady-state.

Now consider the case of improving the measurement equipment. Specifically, assume the standard deviation of the measurement noise is reduced by an order of magnitude, $\Sigma_v = 0.08$. The plots in Figure 6.5 indicate the new results. Under this scenario, the bottom two plots of Figure 6.5 illustrate a substantial difference between error variance of the optimal estimator and the one-step predictor. Note that the variance of the prediction is always larger than that of original estimate, a fact that is easily concluded from
(6.67) and the fact that $\Sigma_{e,k}$, $\Sigma_{e,k}^{-1}$ and $\Sigma_{e,k}^{-1} C \Sigma_{e,k}^{-1} C^* (C \Sigma_{e,k}^{-1} C^* + \Sigma_v)^{-1} C \Sigma_{e,k}^{-1}$ must all be positive definite matrices. □

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6.5.png}
\caption{All plots are for Example 6.5 with $\Sigma_v = 0.08$. Top: Comparison of true state and measurements. 2nd: Comparison of true state and the optimal estimates. 3rd and 4th: Comparison of estimation errors of the Kalman filter and predictor with the two standard deviation envelopes of the error variances.}
\end{figure}
6.3.2 • The Steady-State Kalman Filter

An alternate form of (6.77) is

\[\Sigma_{e,k+1} = (A_d - K_k^+ C)\Sigma_{e,k} + K_k^+ (C\Sigma_{e,k}^+ + \Sigma_v)^{-1} + \Sigma_v + G_d \Sigma_w G_d^* \quad (6.78)\]

\[K_k^+ = A_d \Sigma_{e,k}^+ C (C\Sigma_{e,k}^+ + \Sigma_v)^{-1} \quad (6.79)\]

This can be verified by substituting (6.79) into (6.78) to recover (6.77). Alternatively, Equation (6.78) is observed to be the covariance equation of the error system. Specifically, if \(e_k^+ = x_k - \hat{x}_k\), then

\[e_{k+1}^+ = (A_d - K_k^+ C)e_k^+ - K_k^+ v_k + G_d w_k \quad (6.80)\]

Then, use of Equation (5.93) gives (6.78). Concerning the stability of this error system, the following theorem applies, see Balakrishnan, [120], for proof.

**Theorem 6.4.** If \((A_d, C)\) detectable and \((A_d, G_d)\) stabilizable, then the recursion (6.78)-(6.79) will converge generating a positive definite \(\Sigma_{e,k}^+\) as the limit of \(\Sigma_{e,k}^+\) and \(K_k^+\) as the limit of \(K_k^+\) and the matrix \((A_d - K^+ C)\) will be stable.

In steady-state, the **one-step prediction form** of the Kalman filter is:

\[\hat{x}_{k+1}^+ = A_d \hat{x}_k^+ + B_d u_k + K^+ (y_k - C \hat{x}_k^+) \quad (6.81)\]

\[K^+ = A_d \Sigma_{e}^+ C (C\Sigma_{e}^+ + \Sigma_v)^{-1} \quad (6.82)\]

\[\Sigma_{e}^+ = A_d \left( \Sigma_{e}^+ - \Sigma_{e}^+ C (C\Sigma_{e}^+ + \Sigma_v)^{-1} C\Sigma_{e}^+ \right) A_d^* + G_d \Sigma_w G_d^* \quad (6.83)\]

Equation (6.77), or equivalently (6.78), is commonly denoted as the Riccati equation, which can be used to determine the solution to (6.83), the Algebraic Riccati Equation (ARE), through the limit of iterations. If using MATLAB, then the solution to (6.83) can be determined from the MATLAB function `care(A,B,Q,R)` to solve (6.83). However, it is important to note that this MATLAB function is intended to solve the dual version of the ARE (see `help care` for details). Thus, to use `care` the inputs \((A,B,Q,R)\) should be set to \((A_d^*, C^*, G_d \Sigma_w G_d^*, \Sigma_v)\) and the filter gain of (6.82), \(K^+\), will be the transpose of the gain output from `care`. Similarly, the solution to (6.14), the continuous-time version of the ARE can be found through the MATLAB function `care`.

In steady-state, the **optimal estimator form** of the Kalman filter is:

\[\hat{x}_{k+1} = A_d \hat{x}_k + B_d u_k + K (y_{k+1} - C (A_d \hat{x}_k + B_d u_k)) \quad (6.84)\]

\[K = \Sigma_{e} C (C\Sigma_{e} + \Sigma_v)^{-1} \quad (6.85)\]

\[\Sigma_{e} = \left( (A_d \Sigma_{e} A_d^* + G_d \Sigma_w G_d^*)^{-1} + C^+ (C\Sigma_{e} + \Sigma_v)^{-1} C \right)^{-1} \quad (6.86)\]

To determine \(\Sigma_{e}\), the iteration limit of Equation (6.74) could be used. Alternatively, if \(\Sigma_{e}^+\) is determined, then the following equivalent relations can be used:

\[\Sigma_{e} = \left( (\Sigma_{e}^+)^{-1} + C^+ (C\Sigma_{e} + \Sigma_v)^{-1} C \right)^{-1}\]

or

\[\Sigma_{e} = \Sigma_{e}^+ - \Sigma_{e}^+ C (C\Sigma_{e} + \Sigma_v)^{-1} C \Sigma_{e}^+ \quad (6.87)\]

Also, the filter gain could be calculated using the following alternate form:

\[K = \Sigma_{e}^+ C (C\Sigma_{e} + \Sigma_v)^{-1} \quad (6.88)\]
6.3. The Discrete-time Kalman Filter

Example 6.6. Continuing Example 6.5, the ARE is found to be

$$\Sigma_e^+ = (0.95^2)(\Sigma_e^+ - \Sigma_e^+(\Sigma_e^+ + 8)^{-1}\Sigma_e^+ + 2(0.49^2))$$ (6.89)

which can be rearranged to a quadratic equation:

$$0 = (\Sigma_e^+)^2 + [8(1 - 0.95^2) - 2(0.49^2)]\Sigma_e^+ - 16(0.49^2)$$ (6.90)

The roots of (6.90) are: -2.11 and 1.82, of which the positive one should be selected. Alternatively, one could use the MATLAB command `dare(0.95, 1, 0.49^2 + 2, 8)` to find $\Sigma_e^+ = 1.816$. Then the gain of the prediction form of the filter is found to be $K^+ = 0.95(1.816)/(1.816 + 8) = 0.176$. Based on the results of the prediction form, the error covariance of the original filter is found to be $\Sigma_e = [1.816^{-1} + 8^{-1}]^{-1} = 1.48$ or $\Sigma_e = 1.816 - 1.816^2(1.816 + 8)^{-1} = 1.48$. Finally, the gain of the original filter is calculated as $K = (1.48)8^{-1} = 0.185$ or $K = 1.816(1.816 + 8)^{-1} = 0.185$.

Now return to Example 6.1. The solution to the continuous-time ARE can be found from the MATLAB command `care(-0.1, 1, 1, 4)` which gives $\Sigma_e = 1.640$ and $K = 0.410$. This continuous-time error variance could also be found as the limit of the discrete-time error variance as the sample-time, $\Delta t$, approaches zero, where $\Sigma_e^+$ approaches from above and $\Sigma_e$ from below. Under the same limit of letting the sample-time approach zero, $K^+/\Delta t$ and $K/\Delta t$ will both approach the continuous-time gain.

Example 6.7. Consider the following continuous-time model of a mass-spring-damper damper with a shaping filter model for the disturbance:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ -3 & -2 & 0.3 \\ 0 & 0 & -1.1157 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 5.5816 \end{bmatrix} w$$ (6.91)

where the first state is position and the second is velocity. Assume $w(t)$ is white noise processes with zero mean and spectral density $S_w = 20$. Application of sample and hold with a sample time of 0.2 results in

$$x_{k+1} = \begin{bmatrix} 0.9478 & 0.1616 & 0.0048 \\ -0.4847 & 0.6246 & 0.4031 \\ 0 & 0 & 0.8 \end{bmatrix} x_k + \begin{bmatrix} 0.0019 \\ 0.0271 \\ 1.0003 \end{bmatrix} w_k$$

with $\Sigma_w = S_w/\Delta t = 100$. Furthermore assume the measurement equation is

$$y = \begin{bmatrix} 1 & 0 & 0 \\ -3 & -2 & 0.3 \end{bmatrix} x + v$$ (6.92)

with $S_v = diag([1, 0.2])$. Then, $\Sigma_v = S_v/\Delta t = diag([5, 1])$. The solution to the continuous-time ARE, Equation (6.12), is found from `care($A^*, C^*, GS_w G^*, S_v$)`:

$$\Sigma_e = \begin{bmatrix} 0.5706 & 0.2413 & 6.7235 \\ 0.2413 & 0.2868 & 4.4303 \\ 6.7235 & 4.4303 & 122.75 \end{bmatrix}$$ (6.93)

The solution to the discrete-time ARE, Equation (6.83), is found from `dare($A_d^*, C^*, G_d \Sigma_w G_d^*, \Sigma_v$)`:

$$\Sigma_e^+ = \begin{bmatrix} 0.6339 & 0.2327 & 6.2902 \\ 0.2327 & 0.3247 & 5.9747 \\ 6.2902 & 5.9747 & 168.01 \end{bmatrix}$$ (6.94)
Then, application of (6.87) gives:

\[
\Sigma_e = \begin{bmatrix}
0.5380 & 0.2333 & 6.7433 \\
0.2333 & 0.2858 & 4.4486 \\
6.7433 & 4.4486 & 106.17
\end{bmatrix}
\]  

(6.95)

As suggested in Example 6.6, (6.94) minus (6.93) and (6.94) minus (6.95) are both positive definite, and if \( \Delta t \to 0 \), then both (6.94) and (6.95) will converge to (6.93).

![Figure 6.6. Simulation of the true states and measurements for Example 6.8.](image)

**Example 6.8.** Consider the process of Example 6.7. A simulation of the discrete-time process results in the top plot of Figure 6.6. Of course, in the real process these state trajectories would be unknown. However, the measurements of bottom two plots of Figure 6.6 will be available to the state estimator. To illustrate the impact of measurement noise, these plots also include the uncorrupted measurements, which of course are unavailable to the state estimator.

Given the value of \( \Sigma_e \), from Example 6.7, one can use Equation (6.85) to calculate the optimal estimator gain:

\[
K = \begin{bmatrix}
0.1076 & -0.0577 \\
0.0467 & 0.0629 \\
1.3487 & 2.7246
\end{bmatrix}
\]
Then, application of this gain to the Kalman filter of Equation (6.84) results in the plots of Figure 6.7.

![Figure 6.7. Comparison of optimal estimates with the true state for Example 6.8.](image1)

Given these state estimates, $\hat{x}_k$, one can easily calculate the one-step state prediction as: $\hat{x}_{k+1}^+ = A_0 \hat{x}_k$. An alternate approach to finding the sequence $\hat{x}_{k+1}^+$ is to start with $\Sigma_e^+$ of Example 6.7. Then, calculate $K^+$ from Equation (6.82) or obtain $K^+$ directly from the MATLAB function ‘dare’. Then, the one-step predictor of Equation (6.81) could be implemented. For this particular example, a plot of $\hat{x}_{k+1}^+$ will be virtually indistinguishable from $\hat{x}_k$, at least visually. This is supported by the fact that $\Sigma_e$ and $\Sigma_e^+$ are almost the
same.

If the spectral density of the measurement noise is changed to \( S_v = \text{diag}([0.01 \ 0.002]) \) then the error covariances are calculated to be

\[
\Sigma_e = \begin{bmatrix}
0.0087 & 0.0042 & 0.1156 \\
0.0042 & 0.0041 & 0.0733 \\
0.1156 & 0.0733 & 1.7822
\end{bmatrix}
\quad \text{and} \quad
\Sigma_e^+ = \begin{bmatrix}
0.0108 & 0.0093 & 0.2951 \\
0.0093 & 0.0768 & 2.7616 \\
0.2951 & 2.7616 & 101.2043
\end{bmatrix}
\]

Since the diagonal values of these covariance matrices are much smaller we expect better performance from the estimator, which is supported by the top plot Figure 6.8. However, note the discrepancy between \( \Sigma_e \) and \( \Sigma_e^+ \) in the second diagonal element, that corresponding to the error variance of the velocity variable. This difference is also observed in bottom (error) plot of Figure 6.8.

Table 6.1: Example code for Example 6.7 and 6.8

```matlab
clear all
%
Continuous-time model
A=[0 1 0; -3 -2 0.3;0 0 -1.1157]; G=[0; 0; 5.5816]; Sw=20; nx=3;
C=[1 0 0; -3 -2 0.3]; Sv=[1 0; 0 0.2];
%
Convert to discrete-time
dt=0.2; Ndt=500; sum=zeros(nx); ddt=dt/Ndt;
for jjj=1:Ndt; sum=sum+expm(A * jjj * ddt); end
Ad=expm(A* dt); Gd=sum*expm(A*jjj*ddt); end
%
Simulate with Stochastic Disturbance
NNN=round(40/dt); ttt=zeros(1, NNN); xxx=zeros(nx, NNN); xxx(:,1)=zeros(nx,1);
yyy=zeros(2, NNN); randn('state',2^6-1);
for ii=1: NNN-1;
    ttt(ii+1)=dt*ii;
    ww=randn*sqrt(Sigw); xxx(:,ii+1)=Ad*xxx(:,ii)+Gd*ww;
    vv=sqrt(Sigv)*randn(2,1); yyy(:,ii)=C*xxx(:,ii)+vv;
end
%
Design Optimal Filter
Sige_plus=dare(Ad',C',Gd*Sigw*Gd',Sigv)
Sige=inv(inv(Sige_plus)+C'*inv(Sigv)*C)
K=Sige*C'* inv(Sigv)
%
Implement Optimal Filter
xxx_hat=zeros(nx, NNN); xxx_hat_plus=zeros(nx, NNN);
for ii=1: NNN-1
    xxx_hat(:,ii)=xxx_hat_plus(:,ii)+K*(yyy(:,ii)-C*xxx_hat_plus(:,ii));
    xxx_hat_plus(:,ii+1)=Ad*xxx_hat(:,ii);
end
xxx_hat(:,NNN)=xxx_hat_plus(:,NNN)+K*(yyy(:,NNN)-C*xxx_hat_plus(:,NNN));
%
Plot Estimates
figure(1), plot(ttt, xxx(1,:), 'k-', ttt, xxx_hat(1,:), 'k--', 'linewidth', 2)
legend('Mass Position', 'Estimate of Mass Position')
xlabel('Time (seconds)', 'FontSize', 14, 'FontName', 'Times New Roman');
figure(2), plot(ttt, xxx(2,:), 'k-', ttt, xxx_hat(2,:), 'k--', 'linewidth', 2)
legend('Mass Velocity', 'Estimate of Mass Velocity')
xlabel('Time (seconds)', 'FontSize', 14, 'FontName', 'Times New Roman');
```
6.3.3 • Orthogonality and the Innovations Process

Since the Kalman filter is an application of conditional mean, Theorem 6.2 indicates that the error signal is orthogonal to any estimator. Specifically,

$$E[f(y_k, y_{k-1}, \ldots, y_0)(\hat{x}_k - \hat{x}_k^*)] = E[f(y_k, y_{k-1}, \ldots, y_0)\epsilon_k^*] = 0$$

and

$$E[f(y_k, y_{k-1}, \ldots, y_0)(\hat{x}_{k+1}^+ - \hat{x}_{k+1}^*)] = E[f(y_k, y_{k-1}, \ldots, y_0)\epsilon_{k+1}^+] = 0$$

for all functions $f(\bullet)$. Furthermore, application of Equation (6.24) to the Kalman filter gives the following very useful identities: $\Sigma_{e, k} = \Sigma_{x, k} - \Sigma_{\hat{x}, k}$ and $\Sigma_{e, k}^+ = \Sigma_{x, k} - \Sigma_{\hat{x}, k}^+$. In steady-state these become:

$$\Sigma_e = \Sigma_x - \Sigma_{\hat{x}} \quad \text{and} \quad \Sigma_e^+ = \Sigma_x - \Sigma_{\hat{x}}^+$$

(6.98)

The interpretation of $\Sigma_{\hat{x}}$ and $\Sigma_{\hat{x}}^+$ is elucidated by notion of the innovations sequence defined as:

$$\eta_k = y_k - C\hat{x}_k^+ = y_k - C(A_d\hat{x}_{k-1} + B_d u_{k-1})$$

(6.99)

If the innovations notation is used within (6.72) and (6.75), the Kalman filter and one-step predictor are found to be

$$\hat{x}_k = A_d\hat{x}_{k-1} + B_d u_{k-1} + K_k \eta_k$$

(6.100)

and

$$\hat{x}_{k+1}^+ = A_d\hat{x}_k^+ + B_d u_k + K_k^+ \eta_k$$

(6.101)

For the moment, let us assume $u_k = 0$ for all $k$ and that $\eta_k$ is a zero-mean, white noise process with covariance $\Sigma_{\eta_k}$. If this is the case, then one can apply Equation (5.91) to find that: $\Sigma_{\hat{x}, k+1} = A_d \Sigma_{\hat{x}, k} A_d^* + K_k \Sigma_{\eta} K_k^*$. In steady-state, this is found to be

$$\Sigma_{\hat{x}} = A_d \Sigma_{\hat{x}} A_d^* + K \Sigma_{\eta} K^*$$

(6.102)

In Chapter 8, the manipulated variable will be a linear feedback of state estimate, $u_k = -L\hat{x}_k$. Then, the covariance of the state estimate, $\Sigma_{\hat{x}}$, can be determined from Equation (6.102), but with $A_d$ replaced by $(A_d - B_d L)$. Then, with a known value for $\Sigma_e$, Equation (6.98) can be used to determine the closed-loop value of $\Sigma_e$. Of course, these relations will be valid only if the innovations sequence is a white noise process.

**Theorem 6.5.** The innovations sequence as defined in (6.99) is zero-mean with the following autocorrelation matrix

$$R^{(\gamma)}_{k, i} = E[\eta_k \eta_{k-i}^*] = \begin{cases} C \Sigma_{e, k}^+ C^* + \Sigma_{\omega} & i = 0 \\ 0 & \text{otherwise} \end{cases}$$

(6.103)

**Proof.** First note that $\eta_k = Ce_k^* + v_k$. Then, use of Corollary 6.1 indicates the Kalman estimate is unbiased and that $E[\epsilon_k^*] = 0$. Thus, $\eta_k$ is zero-mean. For the autocorrelation matrix, note that $\eta_{k-i}$ is a function of $y_{k-i}, y_{k-i-1}, \ldots, y_0$. Thus, if $i \geq 1$, then
\[ E[\eta_k \eta_{k-i}] = E[C e_k^+ \eta_{k-i}^* + v_k \eta_{k-i}^*] = 0 + 0, \]
where the first is due to orthogonality of \( e_k^+ \) with a function of the measurements, \( \eta_{k-i} \), and the second is due to independence of \( \eta_{k-i} \) with respect to future measurement noise \( v_k \). The case of \( i \leq -1 \) will generate the similar results. If \( i = 0 \), then \( e_k^+ \) will not be orthogonal to the innovation, \( \eta_k \), but will be independent of the measurement noise, \( v_k \). □

![Figure 6.9.](image)

**Figure 6.9.** Innovations process of Example 6.9 along with the two standard deviation envelope.

**Example 6.9.** Continuing Example 6.5, the innovations process is given in Figure 6.9. Clearly, it is zero mean and appears to be white. (To test for whiteness, one would need to apply a numeric Fourier transform to the data to estimate the spectral density and verify that this density is indeed uniform.) From Example 6.6, it is concluded that \( \Sigma^+ \eta = 1.816. \) Thus, \( \Sigma_\eta = 9.816 \) and the 2 standard deviation envelope should be 6.28, as indicated in the figure. □

If the controller is a linear feedback of state prediction, \( u_k = -L \hat{x}_k^+ \), then the covariance of the state prediction, \( \Sigma_{\hat{x}^+} \), can be determined from

\[ \Sigma_{\hat{x}^+} = A_d \Sigma_{\hat{x}^+} A_d^+ + K^+ \Sigma_\eta K^{+*} \]

(6.104)

with \( A_d \) replaced by \( (A_d - B_d L) \). Then, with a known value for \( \Sigma_{\hat{x}^+} \), the second half of Equation (6.98) can be used to determine the closed-loop value of \( \Sigma_\xi \).

In the continuous-time case, the innovations process is defined similarly

\[ \eta(t) = y(t) - C \hat{x}(t) \]

(6.105)

In this case, the innovations process is also zero mean white noise. However, the spectral density is equal to that of the measurement noise:

\[ S_\eta = S_v \]

(6.106)

If the controller is a linear feedback of continuous-time state estimate, \( u = -L \hat{x} \), then the covariance of the state estimate, \( \Sigma_\hat{x} \), can be determined as the covariance of the error system \( \dot{e} = (A - BL)e + K \eta \):

\[ 0 = (A - BL) \Sigma_\hat{x} + \Sigma_\hat{x} (A - BL)^* + KS_\eta K^{**} \]

(6.107)

Then, with a known value for \( \Sigma_\varepsilon \), the continuous-time relation \( \Sigma_\varepsilon = \Sigma_\xi - \Sigma_\hat{x} \) can be used to determine the closed-loop value of \( \Sigma_\xi \).
6.4 Multi-step Prediction and Smoothing

Once the optimal state estimate at the current time has been determined, one may then ask about future values of the state. This is the notion of prediction or forecasting. This section will expand upon the one-step predictor of Section 6.3. Similarly, one may ask if estimates of the state can be improved once additional measurements have arrived. Intuitively, one would expect that access to measurements in the future, with respect to the time of interest, will improve our estimate of the state. This is the notion of smoothing, which we will find to be a simple extension of the original optimal filtering result.

Using the notation of Section 6.3.2, optimal filtering is the process of calculating the conditional mean of the state at time \( \hat{x}_k \), given measurements up to time \( k \), \( \hat{x}_k = E[x_k | y_0, y_1 \ldots y_k] \). Prediction is to obtain the conditional mean of state at time \( k+i \), \( \hat{x}_{k+i} = E[x_{k+i} | y_0, y_1 \ldots y_k] \) \((i \geq 1)\). The smoothing problem is to determine the conditional mean of the state at time \( k \), \( x_k \), given the measurements up to time \( k+i \), \( \hat{x}_{k|k+i} = E[x_k | y_0, y_1 \ldots y_{k+i}] \) \((i \geq 1)\).

6.4.1 Multi-step Predictions of the State

In the case of having only measurements of the past, state prediction is simply a simulation of the system model assuming the white noise input is equal to its expected value of zero. Specifically, define \( \hat{x}_{k+i|k} \) as the prediction of the state at a future time \( k+i \), where \( k \) is the present time and \( i \geq 1 \). Then \( \hat{x}_{k+i|k} \) is calculated from

\[
\hat{x}_{k+i|k} = A_{d} \hat{x}_{k+i|k} + B_{d} u_{k+i|k}, \quad i = 0, 1, \ldots 
\]

(6.108)

where \( u_{k+i|k} \) is the prediction of the manipulated variable. The initial condition of predictor at \( i = 0 \) is \( \hat{x}_{k|k} \). Clearly this should be set equal to the Kalman filter estimate at time \( k \), \( \hat{x}_{k|k} = \hat{x}_k \), where \( \hat{x}_k \) is from (6.72) or (6.84). Since \( \hat{x}_{k+i|k} = \hat{x}_{k+i} \), we see that (6.108) is just a generalization of the one-step predictor.

A more compact representation of the optimal predictor is to define \( j \) as the time of the prediction, which would make \( i \) the difference between the prediction time and actual time \( i = j - k \). In this case, \( \hat{x}_{j|k} = E[x_j | y_0, y_1 \ldots y_k] \), \( j > k \) is given by

\[
\hat{x}_{j+1|k} = A_{d} \hat{x}_j + B_{d} u_{j|k}, \quad j = k, k+1, \ldots 
\]

(6.109)

with initial condition \( \hat{x}_{k|k} = \hat{x}_k \). If one is interested in the estimation error covariance,

\[
\Sigma_{e_j|k} = E[(x_j - \hat{x}_j)(x_j - \hat{x}_j)^T | y_0 \ldots y_k],
\]

then it can be calculated from

\[
\Sigma_{e,j+1|k} = A_d \Sigma_{e,j|k} A_d^T + G_d \Sigma_w G_d^T, \quad j \geq k 
\]

(6.110)

The initial condition of (6.110) is \( \Sigma_{e,k|k} = \Sigma_{e,k} \), where \( \Sigma_{e,k} \) is from (6.67) or if appropriate could be set to the steady-state covariance \( \Sigma_{e} \) from (6.86). Again we see that \( \Sigma_{e,k+1|k} \) is equal to the error covariance of the one-step predictor \( \Sigma_{e,k+1} \).

Example 6.10. Reconsider Example 6.5 with \( \Sigma_{e} = 8 \), and assume the measurements are known up to time \( t = 15 \), as indicated in the top plot of Figure 6.10. If one applies (6.72) and (6.108) - the state estimator for \( t = 0 \) to 15 and the multi-step predictor for \( t = 15 \) to 29 - the middle plot of Figure 6.10 is obtained. The bottom plot of Figure
6.10 illustrates the (unknown) estimation error signal and that of the prediction (also unknown) as well as the 2 standard deviation envelope determined from (6.74) and (6.110), both of which are known. As expected, the error variance of the multi-step prediction is larger than that of Kalman filter, increasing with time until it reaches a steady state value of $\Sigma_x$.

**Example 6.11.** Consider the following continuous-time model of a mass-spring damper:

$$
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w
$$

$$
y = \begin{bmatrix} 0 & 1 \end{bmatrix} x + v
$$

where the first state is position and the second is velocity. Assume $w(t)$ and $v(t)$ are both white noise processes with zero mean and power spectral densities: $S_w = 0.1$ and $S_v = 0.001$. Now assume the measurements are known up to time $t = 5$, as indicated in the top plot of Figure 6.11. Application of (6.72) and (6.108) yields the state estimate for $t = 0$ to 5 and the multi-step prediction of state for $t = 5$ to 10, as shown in the two middle plots of Figure 6.11. The bottoms plots in Figure 6.11 illustrate the error signal of the estimator along with that of multistep predictor. The 2 standard deviation envelopes are determined from (6.74) and (6.110).
6.4. Multi-step Prediction and Smoothing

6.4.2 Smoothing of the State Estimates

Smoothing is similar to prediction in that measurements up to the present are given, but we would like an optimal estimate of a past state. Specifically, the smoothed estimate is \( \hat{x}_{k-i|k} = E[x_{k-i} | y_0, \ldots, y_k], \ i \geq 1 \). The key to calculating this quantity is to have a model that keeps track of state values from the past. As such define \( x^{(j)}_k = x_{k-j} \). Then, we conclude that \( x^{(1)}_{k+1} = x^{(0)}_{k+1} = x_k \). Similarly, we define \( x^{(j)}_{k+1} = x^{(j-1)}_k \). Now collect the past state variables into a compound vector \( \chi_k \), defined:

\[
\chi_k = \begin{bmatrix}
  x^{(0)}_k \\
  x^{(1)}_k \\
  \vdots \\
  x^{(j)}_k
\end{bmatrix}
\]  

(6.111)
Then, it is easily concluded that:

$$
\chi_{k+1} = \begin{bmatrix}
A_d & 0 & \cdots & 0 & 0 \\
I & 0 & \cdots & 0 & 0 \\
0 & I & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & I & 0
\end{bmatrix}
\chi_k +
\begin{bmatrix}
G_d \\
0 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix}
\omega_k
$$

(6.112)

Using this new compound state, the measurement equation, \( y_k = C x_k + v_k \), takes the form of

$$
y_k = C \chi_k + v_k
$$

(6.113)

Thus, the compound system used as the basis of the smoother is

$$
\chi_{k+1} = A^{(sm)} \chi_k + G^{(sm)} w_k
$$

(6.114)

$$
y_k = C^{(sm)} \chi_k + v_k
$$

(6.115)

Then, the previous optimal estimation results, (6.84)-(6.86), can be used to find \( \hat{x}_k \) as:

$$
\hat{x}_{k+1} = A^{(sm)} \hat{x}_k + K^{(sm)} (y_{k+1} - C^{(sm)} A^{(sm)} \hat{x}_k)
$$

(6.116)

$$
K^{(sm)} = \Sigma^{(sm)} C^{(sm)} \Sigma_v^{-1}
$$

(6.117)

$$
\Sigma^{(sm)} = \left[ \left( A^{(sm)} \Sigma_c^{(sm)} A^{(sm)^\ast} + G^{(sm)} \Sigma_c^{(sm)} G^{(sm)^\ast} \right)^{-1} + C^{(sm)} \Sigma_v^{-1} C^{(sm)^\ast} \right]^{-1}
$$

(6.118)

Finally, it is found that \( E[\chi_{k-i} \mid y_0 \cdots y_k] = \hat{x}_{k-i} = \chi^{(i)} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix} \hat{x}_k \).

It is noted that the smoother described above (the fixed lag smoother) is just one of three possible forms. Descriptions of the other two forms (fixed point and fixed interval) can be found in [132].

### 6.5 Chapter Summary

The primary contribution of this chapter is the development of the Kalman filter as the optimal state estimator. While the initial derivation, in continuous-time, characterized the estimates as minimizing the estimation error covariance, the discrete-time derivation exposed a number of deeper meanings. Specifically, it was shown that the optimal estimate is also the condition mean of the state variable and as such its error signal is orthogonal to all estimators. This, along with the whiteness of the innovations process, will be used extensively in subsequent developments of measurement information based controller design schemes.

However, the most notable aspect of the chapter is the steady-state form of the Kalman filter, Equation (6.84)-(6.86), along with the one-step predictor, Equations (6.81)-(6.83), as these will be the most commonly used in applications as well as in subsequent chapters. At the heart of the steady-state development one finds the Algebraic Riccati Equation (ARE), Equation (6.83), along with its continuous-time incarnation, Equation (6.14). We will encounter the dual version of both of these ARE’s in the following chapter on linear quadratic optimal control.

Much of the material of this chapter can be found in other sources, see for example Kalman [129], Jazwinski [130], Gelb [131], Stengel [104]. The development of estimation theory was adapted from Anderson and Moore, [132], and Van Trees [133],...
Exercises

6.1. Consider the following continuous-time scalar process:

\[
\dot{x} = w \\
y = x + v
\]

where \( S_w = 4 \), \( S_v = 1 \) and the initial estimation error variance, \( \Sigma_e(0) \), is 0. Using separation of variables, solve the differential equation of Equation (6.13). Generate plots of \( \Sigma_e(t) \) and \( K(t) \) as a function of time. Identify the steady-state value of \( K \) and determine the eigenvalue of the error process of Equation (6.4).

6.2. Consider the following discrete-time scalar process:

\[
x_{k+1} = x_k + 0.001 w_k \\
y_k = x_k + v_k
\]

where \( \Sigma_w = 4000 \), \( \Sigma_v = 1000 \) and the initial estimation error variance, \( \Sigma_e(0) \), is 0. Using the recursive relation of Equation (6.76) and (6.77), generate plots of \( \Sigma_e(k) \) and \( K_k \). Compare this result with the solution to Exercise 6.1.

6.3. Consider the following scalar process: \( \dot{x} = -x + w, \ y = x + v \). Assume \( w(t) \) and \( v(t) \) are zero mean white noise processes with spectral densities given as: \( S_w = 1 \) and \( S_v = 1 \). Construct the following observer: \( \hat{x} = A \hat{x} + K(y - C \hat{x}) \) where \( K = 1 \).

(i) Determine the steady-state variance of the error signal, defined as \( e(t) = x(t) - \hat{x}(t) \). In other words find: \( \Sigma_e = \lim_{t \to \infty} E[e(t)^2] \)

(ii) Determine the optimal gain \( K \), as suggested by the Kalman filter, and again determine the steady-state variance of the error signal.

6.4. Using the sample code of Table 6.1, reproduce the results of Examples 6.7 and 6.8.

6.5. Consider the following continuous-time model of a mass-spring damper:

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w \\
y = \begin{bmatrix} 0 & 1 \end{bmatrix} x + v
\]

where the first state is position and the second is velocity. Assume \( w(t) \) and \( v(t) \) are both white noise processes with zero mean and power spectral densities: \( S_w = 0.1 \) and \( S_v = 0.001 \).

(i) Determine the steady-state error covariance matrix, \( \Sigma_e \), of the continuous-time Kalman filter using the MATLAB function ‘care’.

(ii) Discretize the system using the sample and hold method (with \( \Delta t = 0.01 \)).

(iii) Determine the steady-state error covariance matrix of the one-step predictor, \( \Sigma_e^+ \), using the MATLAB function ‘dare’.

(iv) Determine the error covariance of the discrete-time optimal estimator, \( \Sigma_e \).

6.6. Using the process of Exercise 6.5, perform the following tasks:
(i) Using the discrete-time model (sample time $\Delta t = 0.01$) simulate the process over 10 seconds with an initial condition of zero. Plot the two state variables. In a real system, this plot would be unavailable to you. However, in this simulated environment the actual state values will be useful in illustrating the performance of the estimator.

(ii) Make a plot of the noise corrupted measurement, $y$, and compare with the uncorrupted measurement, $Cx$.

(iii) Implement a sub-optimal state estimator using an estimator gain $K = [0.000143 \ 0.0142]^T$. Make plots (one for the position and one for the velocity) comparing the actual state with the state estimate.

(iv) Repeat part (iii), but with the optimal estimator gain.

(v) Repeat part (iv) with $S_v = 0.01$

(vi) Repeat part (iv) with $S_v = 0.1$

(vii) Repeat part (iv) with the following measurement equation $y = [1 \ 0]x + v$ and $S_v = 0.001$.

(viii) For each of the above cases, calculate the steady-state standard deviation of the error signal. Comment on any trends you observe.

6.7. Repeat Exercise 6.6 assuming the disturbance, $w$, is colored noise with a correlation time of 0.5 time units and a variance of 0.1 (Recall the shaping filter design methods of Chapter 5). What if the correlation time was 10 time units?

6.8. Consider the following scalar continuous-time system: $\dot{x} = w$ where $w$ is a zero mean white noise process with spectral density $S_w = 4$.

   a) If the measurement equation is $y = x + v$, where $v$ is zero mean white noise with spectral density $S_v = 1$, determine the steady-state error variance of the optimal estimator.

   b) Now assume two sensors are used at all times. In this case the measurement equation is $y = \begin{bmatrix} 1 & 1 \end{bmatrix} x + v$, where $v$ is zero mean white noise with spectral density $S_v = \text{diag}[s_1 \ s_2]$.

      (i) Determine the steady-state error variance of the optimal estimator for this case, and with $s_1$ and $s_2$ as parameters.

      (ii) Calculate the steady-state error variance of the optimal estimator under the following conditions:

            (1) $s_1 = 1$ and $s_2 \to \infty$ (2) $s_1 = 1$ and $s_2 \to 0$ (3) $s_1 \to \infty$ and $s_2 \to \infty$ (4) $s_1 \to 0$ and $s_2 \to 0$

      (iii) Discuss the results of part b(ii), and indicate why each should be expected.

6.9. Consider the differential equation $\ddot{r} + \dot{r} + r = F$, where $r$ is the position of a particle and $F$ is an external force applied to the particle.

   (i) Convert this differential equation into state space form $\dot{x} = Ax + Gw$, $y = Cx + v$. Assume the output $y$ is a noise corrupted measurement of particle velocity, and the input force is a zero mean white noise process with spectral density $S_w = 2$. Determine the steady-state variance of the particle position without a computer.
(ii) If the measurement noise is a zero mean white noise process with spectral density \( S_v = 1 \), determine the appropriate Kalman gain and the resulting estimation error variance for both the position and velocity of the particle without a computer.

(iii) If the external force was actually colored noise with a variance \( \Sigma_w = 2 \) and a correlation time of 10 time units, repeat the calculations of parts (i) and (ii) using MATLAB.

6.10. Consider the following continuous-time process.

\[
\begin{align*}
\dot{x}_1 &= -x_1 + w_1 \\
\dot{x}_2 &= -x_2 + w_2
\end{align*}
\]

with measurements

\[
\begin{align*}
y_1 &= x_1 + v_1 \\
y_2 &= x_2 + v_2
\end{align*}
\]

Assume both disturbances and each measurement noise terms are white noise processes each with a spectral density of 1.

(i) Determine the optimal estimator gain for this system as well as the estimation error variance one should expect from the application of this gain.

(ii) Consider the following continuous-time process, with disturbances and measurement noise characteristics the same as part (i).

\[
\begin{align*}
\dot{x}_1 &= -x_1 + w_1 \\
\dot{x}_2 &= -x_2 + 0.707w_1 + 0.707w_2
\end{align*}
\]

Determine the estimation error variance one should expect, if the optimal estimator gain from part 1 is applied to this process. (Note: the spectral density of \( 0.707w_1 + 0.707w_2 \) is 1.)

(iii) Using the MATLAB function ‘care’ it was determined that the optimal estimation gain for the process of part (ii) is:

\[
K = \begin{bmatrix}
0.391 & 0.254 \\
0.254 & 0.391
\end{bmatrix}
\]

Compare and contrast this result with the solutions you obtained in parts (i) and (ii).

6.11. Consider the following scalar continuous-time stochastic process: \( \dot{x} = -x + w \) with a measurement equation: \( y = x + v \). If \( w \) and \( v \) are zero mean white noise processes with spectral densities \( S_w = S_v = 1 \), then the gain of the optimal steady-state estimator is \( K = 0.414 \), which results in a steady-state estimation error of \( \Sigma_e = 0.414 \).

(i) If \( \hat{x} \) is the optimal estimate and \( e \) is the error of that estimate \( (e = x - \hat{x}) \), then show that \( E[ee] = 0 \) in the steady-state. Hint: Form a compound system containing the estimator and the error system and calculate the compound covariance matrix

\[
\Sigma = \begin{bmatrix}
E[ee] & E[e\hat{x}] \\
E[\hat{x}e] & E[\hat{x}\hat{x}]
\end{bmatrix}
\]
6.12. Consider a continuous-time process:
\[
\dot{x} = Ax + Gw \\
y = Cx + v
\]
(6.119)
where \(w\) and \(v\) are zero mean white noise processes with spectral densities \(S_w\) and \(S_v\). The optimal state estimate, \(\hat{x}\), is given by the following process \(\dot{\hat{x}} = A\hat{x} + K(y - C\hat{x})\) where the estimator gain is \(K = \Sigma_e C^* S_v\) and \(\Sigma_e\) is the positive definite solution to the Riccati equation: \(\dot{\Sigma_e} = A\Sigma_e + \Sigma_e A^* + GS_w G^* - \Sigma_e C^* S_v^{-1} C\Sigma_e\). Now assume you are given the optimal estimate at a time \(t\), \(\hat{x}(t)\), and define the optimal state prediction at time \(\tau > t\), \(\hat{x}_p(\tau|t)\), as the solution to the following process:
\[
\dot{\hat{x}_p}(\tau|t) = A\hat{x}_p(\tau|t) \\
\hat{x}_p(t|t) = \hat{x}(t)
\]
(i) If the prediction error is defined as \(e_p(\tau|t) = x(\tau) - \hat{x}_p(\tau|t)\), show that the covariance of the prediction error, \(\Sigma_p(\tau|t) = E[e_p(\tau|t)e_p(\tau|t)^*]\), is governed by the differential equation:
\[
\dot{\Sigma_p}(\tau|t) = A\Sigma_p(\tau|t) + \Sigma_p(\tau|t) A^* + GS_w G^* \\
\Sigma_p(t|t) = \Sigma_e(t)
\]
(Hint: find the differential equation governing \(e_p(\tau|t)\))

(ii) Consider a scalar continuous-time process (with time units of seconds):
\[
\dot{x} = -0.5x + w \\
y = x + v
\]
where \(w\) and \(v\) are zero mean white noise processes with spectral densities \(S_w = 1\) and \(S_v = 0.02\). Assume \(t = 100\) sec, and determine the variance of the prediction error at time \(\tau = 101s\).

6.13. Reproduce the results of Example 6.11.
Chapter 7
Linear Quadratic Optimal Control

One finds the origins of optimal control theory in the physically motivated subject of calculus of variations. One of the earliest problems to be cast in the form of a calculus of variations problem is the “brachistochron” (Johann Bernoulli - 1696). The problem is to find the curve such that a bead under the influence of gravity and restricted to the curve would traverse the curve in a minimum time. The modern version of the Linear Quadratic Optimal Control (LQOC) problem (i.e., using a state-space process model) was first solved by Kalman, [134]. While a variety of methods can be used to develop a solution to the LQOC we will focus on the simplistic batch approach (Section 7.1.1) as well as the method of dynamic programming (Section 7.1.2.). The former will provide a foundation to our subsequent development of model predictive control (Chapter 9), while the latter provides a route to solving the infinite-time LQOC problem (Section 7.2) and is central to solving the stochastic version of the LQOC (Chapter 8).

7.1 Finite-time LQOC

Let us begin with a few simple examples.

Example 7.1. Consider just the first step of a scalar discrete-time process: $x_1 = x_0 + u_0$ with initial condition $x_0 = 10$. Now assume you will be paid $(100 - x_1^2)$ for selecting the input $u_0$. Clearly, the best you can do is to select $u_0 = -10$, which will get you the full $100$. Now let us make the problem harder. Assume you will have to pay for the control action in an amount of $3u_0^2$. In this case, your original policy (of $u_0 = -10$) will allow you to collect $100$, but you would also need to pay $300$ for the control action – a net loss to you of $200$. Now suppose you select $u_0 = -5$. In this case, you would at least break even. If $u_0 = -1$, then you would receive $18$. Since you are likely interested in maximizing your profit from the transaction, it makes good sense to plot your objective function:

$$ H(u_0) = 100 - (10 + u_0)^2 - 3u_0^2 $$

where the relation $x_1 = x_0 + u_0$ has been substituted into the revenue expression: $100 - x_1^2$

Based on the plot of Figure 7.1, it is clear that the most profit (of $25) will be achieved by selecting $u_0 = -2.5$. Rather than put in the effort to plot this function, we could have also solved the problem analytically. From the plot it is clear that the maximum value occurs at the point where the slope of $H$ is zero. Thus, we should look for the point $u_0$
such that $dH / du_0 = 0$. In the above case, $dH / du_0 = -2(10 + u_0) - 6u_0$, giving again the result that the best action is $u_0 = -2.5$.

A formal definition of the above optimization problem would be

$$
\max_{u_0} \left\{ 100 - x_1^2 - 3u_0^2 \right\}
$$

s.t. $x_1 = x_0 + u_0$

Since the constant term in the objective function, 100, played no role in determining our analytic solution, it is customary to move it outside of the max operator (or even ignore it). In addition, since maximization of a function is equivalent to minimization of the negative of that function, the solution to the original problem can be obtained by the following simpler problem definition:

$$
\Phi(x_0) = \min_{u_0} \left\{ x_1^2 + 3u_0^2 \right\}
$$

s.t. $x_1 = x_0 + u_0$

In this case, the profit would be $100 - \Phi(x_0)$. We also took the step of leaving $x_0$ as a parameter. Thus, one would find the solution to this problem as $u_0 = -0.25x_0$. If this is substituted back into the objective function (and the modest algebra is carried out) one would find $\Phi(x_0) = 0.75x_0^2$.

Example 7.2. Now consider an extension of Example 7.1, in that the system of interest has two time steps ($x_1 = x_0 + u_0$ and $x_2 = x_1 + u_1$) and your payment is based on the state at the final time ($100 - x_2^2$). The relevant optimization problem can be stated as:

$$
\Phi(x_0) = \min_{u_0,u_1} \left\{ 3u_0^2 + 3u_1^2 + x_2^2 \right\}
$$

s.t. $x_1 = x_0 + u_0$

$x_2 = x_1 + u_1$

Then, after substitution of the equality constraints, the objective is found to be a function of two variables: $H(u_0, u_1) = (x_0 + u_0 + u_1)^2 + 3u_0^2 + 3u_1^2$. In this case, there will be two conditions for optimality $\partial H / \partial u_0 = 0$ and $\partial H / \partial u_1 = 0$, which will result in the following two equations with two unknowns.

$$
2(x_0 + u_0 + u_1) + 6u_0 = 0 \quad \text{and} \quad 2(x_0 + u_0 + u_1) + 6u_1 = 0
$$
Solving these one finds: \( u_0 = u_1 = -0.2x_0 \) and
\[
\Phi(x_0) = \left( 3(0.2)^2 + 3(0.2)^2 + (1 - 0.2 - 0.2)^2 \right) x_0^2 = 0.6x_0^2
\]
If \( x_0 = 10 \), then \( u_0 = u_1 = -2 \), \( \Phi(10) = 60 \) and your profit would be $40. □

The above two examples are special cases of the Linear Quadratic Optimal Control (LQOC) problem in discrete-time. If the dynamic system is a scalar, then a general form of the LQOC problem is stated as:

\[
\Phi(x_0) = \min_{u_0,u_1,\ldots,u_{N-1}} \left\{ \sum_{k=0}^{N-1} \left( q_k x_k^2 + r_k u_k^2 + q_f x_N^2 \right) \right\}
\]

s.t. \( x_{k+1} = a_d x_k + b_d u_k, \quad k = 0 \ldots N - 1 \)

To be equivalent to Example 7.1, just set \( a_d = b_d = q_f = 1, q = 0, r = 3 \) and \( N = 1 \). For Example 7.2 the only difference is \( N = 2 \). As suggested by the examples, the objective is to drive the system state, \( x_k \), toward zero, while keeping the manipulated variable, \( u_k \), fairly small. The weighting parameters \( q \) and \( r \) are intended to reflect the relative importance of each objective. The ability to select \( q_f \) as a value different than \( q \) is simply for convenience, as it is common to put more or less weight on the state at the final time. In fact, if all of the weighting terms are time dependent (\( q \) and \( r \) being replaced with \( q_k \) and \( r_k \)), the subsequent results would require only a slight modification. However, there are few cases where such a problem formulation will be required. If the dynamic model is vector valued, then the LQOC problem is defined as:

\[
\Phi(x_0) = \min_{u_0,u_1,\ldots,u_{N-1}} \left\{ \sum_{k=0}^{N-1} \left( x_k^* Q x_k + u_k^* R u_k \right) \right\}
\]

s.t. \( x_{k+1} = A_d x_k + B_d u_k, \quad k = 0 \ldots N - 1 \)

Clearly, this problem will revert back to (7.1), if using a scalar valued dynamic system.

### 7.1.1 The Batch Approach

The batch solution method is to simply restate the LQOC problem as a standard Quadratic Program (QP) and pass the problem to a generic QP solver, for example the ‘quadprog’ routine in MATLAB. Typically, the general formulation of a QP problem is stated as

\[
\min_{\theta} \{ M \theta^* M + m^* \theta \} \quad \text{s.t.} \quad A_0 \theta = b_0, \quad \text{and} \quad A_1 \theta \leq b_1
\]

To put the problem of Example 7.1 into this form, one could define the vector of ‘to be determined’ (or optimization) variables as \( \theta = \left[ \begin{array}{c} u_0 \\ x_1 \end{array} \right]^* \). Then, the remaining parameters are defined as:

\[
M = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}, \quad m = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad A_0 = \begin{bmatrix} 1 & -1 \end{bmatrix}, \quad b_0 = \begin{bmatrix} -x_0 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0 \end{bmatrix}
\]

For Example 7.2, the optimization variable could be defined as \( \theta = \left[ \begin{array}{c} u_0 \\ u_1 \\ x_1 \\ x_2 \end{array} \right]^* \), and the remaining parameters would be defined as:

\[
M = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_0 = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 \end{bmatrix}, \quad b_0 = \begin{bmatrix} -x_0 \\ 0 \end{bmatrix}
\]
As the number of time steps (or time period) increases, a pattern will begin to emerge. Consider the problem of (7.1) with \( N = 3 \). In this case, the vector of optimization variables could be defined as \( \theta = [u_0 \ u_1 \ u_2 \ x_1 \ x_2 \ x_3]^* \) and

\[
M = \begin{bmatrix}
r & 0 & 0 & 0 & 0 & 0 \\
r & 0 & 0 & 0 & 0 & 0 \\
0 & r & 0 & 0 & 0 & 0 \\
0 & 0 & q & 0 & 0 & 0 \\
0 & 0 & 0 & q & 0 & 0 \\
0 & 0 & 0 & 0 & q & 0 \\
\end{bmatrix}, \quad
A_0 = \begin{bmatrix}
b_d & 0 & 0 & -1 & 0 & 0 \\
0 & b_d & 0 & a_d & -1 & 0 \\
0 & 0 & b_d & 0 & a_d & -1 \\
\end{bmatrix}, \quad
b_0 = \begin{bmatrix}
-a_d x_0 \\
0 \\
0 \\
\end{bmatrix}
\] (7.4)

**Example 7.3.** Let \( a_d = 0.95, \ b_d = 0.5, \ q = 1, \ r = 5, \ q_f = 25, \ N = 4 \) and \( x_0 = 100 \). In this case, one could define \( \theta = [u_0 \ u_1 \ u_2 \ u_3 \ x_1 \ x_2 \ x_3 \ x_4]^* \). Then, the \( M \) matrix is easily determined as:

\[
M = \begin{bmatrix}
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 \\
\end{bmatrix}
\]

To determine the \( A_0 \) and \( b_0 \) matrices, it will be helpful to rearrange the four equality constraints into the following form:

\[
\begin{align*}
0.5u_0 - x_1 &= -0.95x_0 \\
0.5u_1 + 0.95x_1 - x_2 &= 0 \\
0.5u_2 + 0.95x_2 - x_3 &= 0 \\
0.5u_3 + 0.95x_3 - x_4 &= 0
\end{align*}
\]

Then, the matrices are easily constructed as:

\[
A_0 = \begin{bmatrix}
0.5 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0.5 & 0 & 0 & 0.95 & -1 & 0 & 0 \\
0 & 0 & 0.5 & 0 & 0 & 0.95 & -1 & 0 \\
0 & 0 & 0 & 0.5 & 0 & 0 & 0.95 & -1 \\
\end{bmatrix}, \quad
b_0 = \begin{bmatrix}
-95 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]

Table 7.1 illustrates how to construct these matrices in MATLAB as well as the use of the ‘quadprog’ routine to obtain the LQOC solution, given in Table 7.2.

---

**Table 7.1:** MATLAB code used in calculations for Example 7.3.

```matlab
clear
% Problem parameters
ad=0.95; bd=1/2; q=1; r=5; qf=25; N=4; x0=100;
% Define QP matrices
M=[r*eye(N) zeros(N); zeros(N) q*eye(N)]; M(2*N,2*N)=qf;
A0=[bd*eye(N) -eye(N)]
for k=1:N-1 A0(k+1,N+k)=ad; end
b0=zeros(N,1); b0(1,1)=-ad*x0;
```
7.1. Finite-time LQOC

\[ f = \text{zeros}(2 \times N, 1); A_1 = []; b_1 = []; \]
\[
\begin{align*}
\textbf{Check QP matrices} \\
M, A_0, b_0
\end{align*}
\]
\[
\textbf{Solve QP} \\
[z, \phi, eflag] = \text{quadprog}(2 \times M, f, A_1, b_1, A_0, b_0)
\]
\[
\textbf{Display Solution} \\
uk = z(1:N), x_k = [x_0; z(N+1:2 \times N)]
\]

Table 7.2. Optimal LQOC solution for Example 7.3.

<table>
<thead>
<tr>
<th>k</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_k)</td>
<td>100</td>
<td>73.4</td>
<td>50.9</td>
<td>31.2</td>
<td>13.2</td>
</tr>
<tr>
<td>(u_k)</td>
<td>-43.2</td>
<td>-37.7</td>
<td>-34.4</td>
<td>-32.9</td>
<td>-</td>
</tr>
</tbody>
</table>

In the case of a vector valued system, the QP matrices will be constructed as blocks from the matrices of the dynamic system. Thus, the QP formulation of Problem (7.2) of arbitrary horizon size is:

\[
\theta = \begin{bmatrix} u_0^* & u_1^* & \ldots & u_{N-1}^* & x_1^* & x_2^* & \ldots & x_N^* \end{bmatrix}
\]

and

\[
M = \begin{bmatrix} R & 0 & \cdots & 0 & 0 \\ 0 & R & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & R & 0 \\ 0 & 0 & \cdots & 0 & R \end{bmatrix}
\]

(7.5)

\[
A_0 = \begin{bmatrix} B_d & 0 & 0 & \cdots & 0 & -I & 0 & 0 & 0 & \cdots & 0 \\ 0 & B_d & 0 & \cdots & 0 & A_d & -I & 0 & 0 & \cdots & 0 \\ 0 & 0 & B_d & \cdots & 0 & 0 & A_d & -I & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \vdots & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & B_d & 0 & \cdots & A_d & -I & 0 \\ 0 & 0 & 0 & \cdots & B_d & 0 & \cdots & 0 & 0 & A_d & -I \\ -A_dx_0 & 0 & 0 & \cdots & 0 & B_d & 0 & \cdots & 0 & 0 & A_d & -I \end{bmatrix}
\]

(7.6)

\[
b_0 = \begin{bmatrix} -A_dx_0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}
\]

(7.7)
Example 7.4. Consider the following continuous-time model of a mass-spring-damper:

\[
A = \begin{bmatrix} 0 & 1 \\ -3 & -0.2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

Application of the sample-and-hold method (with \(\Delta t = 0.2\) seconds) gives

\[
A_d = \begin{bmatrix} 0.941 & 0.192 \\ -0.576 & 0.903 \end{bmatrix}, \quad B = \begin{bmatrix} 0.020 \\ 0.192 \end{bmatrix}
\]
If the LQOC parameters are selected as: \( Q = I \), \( R = 0.5 \), \( Q_f = 0 \), \( N = 26 \) and \( x_0 = [1 \ 0]^T \),
then the state trajectories resulting from application of the LQOC are depicted in the top left plot of Figure 7.2. If the weight on the manipulated variable is increased (from \( R = 0.5 \) to \( R = 5 \)), then the right plots of Figure 7.2 indicate that the controller is reluctant to select large values for and will settle for a slow convergence of the states to zero. If the MV weight is decreased to \( R = 0.05 \), then we are telling the controller that large values of \( u_k \) are acceptable, as is observed in the bottom plots of Figure 7.2, and convergence of the state variables to zero will be faster than in the other cases.

Now suppose we would like the second state (the velocity) to remain small. This can be achieved by increasing only the second diagonal term of \( Q \). For example, setting \( Q \) as diag([1 50]) and \( R = 0.5 \) will result in the plots of Figure 7.3. Conversely if one sets \( Q \) as diag([50 1]), then Figure 7.4 indicates that the first state (position) will be moved very quickly to zero and the controller has little concern about the size of the second state. It is additionally, noted that the matrices \( Q \) and \( R \) need not be diagonal, but they do need to be positive definite, or at the very least positive semi-definite. However, the selection of non-diagonal weighting matrices is not nearly as intuitive, so in most cases one will stick to diagonal weighting matrices. In Chapter 11, a systematic procedure to exploit off-diagonal terms will be discussed.

It is highlighted that the LQOC problem of (7.2) does not require the use of \( m \), \( A_1 \) or \( b_1 \). In our subsequent development of the MPC controller (Chapter 9) we will see \( A_1 \) and \( b_1 \) come into play. In addition, note that the objective function of (7.2) could be augmented with a cross-term: \( x_k^T S u_k + u_k^T S^* x_k \). This generalization of (7.2) can also be addressed by the QP of (7.3), by modification of (7.5) and utilization of the \( m \) term (see Problems 7.10 and 7.11). This generalization will be utilized in Chapter 11.

### 7.1.2 Dynamic Programming

While the batch solution approach is simple to implement, its numeric basis gives very little insight into the characteristic of the LQOC problem. The method of dynamic programming will serve to uncover many of the rich characteristics inherent to the LQOC problem.
Example 7.5. Consider the following scalar two step problem

$$\Phi(x_0) = \min_{u_0, u_1} \left\{ \begin{array}{l} x_0^2 + u_0^2 + x_1^2 + u_1^2 + x_2^2 \\ \text{s.t. } x_1 = x_0 + u_0 \text{ and } x_2 = x_1 + u_1 \end{array} \right\}$$

(7.8)

The first point to note is that the term $x_0^2$ can be moved outside of the min operator since it is a constant parameter to the optimization problem. The second point to note is that the min operator can be divided into a nested pair of operations, with min over $u_1$ as the inner min and min over $u_0$ as the outer. Thus, problem (7.8) can be rewritten as:

$$\Phi(x_0) = x_0^2 + \min_{u_0} \left\{ \min_{u_1} \left\{ \begin{array}{l} u_0^2 + x_1^2 + u_1^2 + x_2^2 \\ \text{s.t. } x_1 = x_0 + u_0 \\ x_2 = x_1 + u_1 \end{array} \right\} \right\}$$

(7.9)

The idea being that the inner minimization (over $u_1$) will think of $u_2$ as a constant parameter. Specifically, as the outer minimization searches over $u_0$, the inner minimization will change its solution. Based on this perspective of the problem, it is clear that $u_0^2 + x_1^2$ and $x_1 = x_0 + u_0$ can be removed from the inner minimization, as all of these will appear as constants to the inner min operator. Thus, problem (7.8) can be further rewritten as:

$$\Phi(x_0) = x_0^2 + \min_{u_0} \left\{ \min_{u_1} \left\{ u_1^2 + x_2^2 \right. \right. \left. \text{s.t. } x_1 = x_0 + u_0 \right\} \right\}$$

(7.10)

To emphasize the parameter dependency of the inner minimization, (7.8) can be further restated as:

$$\Phi(x_0) = x_0^2 + \min_{u_0} \left\{ u_0^2 + x_1^2 \right. \left. \text{s.t. } x_1 = x_0 + u_0 + \phi_1(x_1) \right\}$$

(7.11)

where $\phi_1(x_1) = \min_{x_1} \left\{ u_1^2 + x_2^2 \right. \text{s.t. } x_2 = x_1\right\}$. Given this perspective, one can solve the $\phi_1$ problem assuming $x_1$ is a known parameter. To do so, one can simply substitute to arrive at $u_1^2 + (x_1 + u_1)^2$, as the expression to be minimized. Then, the solution is quickly found to be $u_1 = -0.5x_1$. Then, substitution of this solution back into the objective function, $u_1^2 + (x_1 + u_1)^2$ results in $\phi_1(x_1) = 0.5x_1^2$. Thus, (7.11) can be rewritten as:

$$\Phi(x_0) = x_0^2 + \min_{u_0} \left\{ u_0^2 + x_1^2 \right. \left. \text{s.t. } x_1 = x_0 + u_0 + 0.5x_1^2 \right\}$$

(7.12)

The process can then be repeated to find the objective as $u_0^2 + 1.5(x_0 + u_0)^2$ with a solution $u_0 = -0.6x_0$. And finally, $\Phi(x_0)$ is found to be $x_0^2 + 1.5(0.4)^2x_0^2 = 1.24x_0^2$. In summary, we have found that the optimal inputs have the following feedback form: $u_0 = -0.6x_0$ and $u_1 = -0.5x_1$, which would have been quite difficult to glean from the batch procedure.

Looking back, we see that the most important step of Example 7.5 was the definition of $\phi_1(x_1)$ and that this minimization could be solved independent of $u_0$, as long as we assume $x_1$ would eventually be known. This separation of the problem into stages...
is the central aspect of dynamic programming, which is made possible by the staged aspect of the dynamic model. Said another way, our state-space model engenders $x_1$ with all the necessary information about $u_0$. Thus assuming knowledge about $x_1$ is equivalent to knowing $u_0$. The impact of this observation expands greatly when we think about problems with larger periods. Specifically, if $x_k$ contains all of the important information about past inputs, then assuming knowledge about $x_k$ is equivalent to knowing $u_0, u_1, \ldots, u_{k-1}$.

The 'cost-to-go' function at time $k$, assuming knowledge of the state, $x_k$, for problem (7.1), is defined as:

$$
\phi_k(x_k) = \min_{u_k, \ldots, u_{N-1}} \left\{ \sum_{i=k}^{N-1} \left( q x_i^2 + r u_i^2 \right) + q_f x_N^2 \right\} \quad \text{s.t.} \quad x_{i+1} = a_d x_i + b_d u_i, \ i = k, \ldots, N-1
$$

(7.13)

Notice that $\Phi_0(x_0)$ is equal to the original value function $\Phi(x_0)$. Now consider separating the minimization over $u$, and pulling out the terms not needed for the remaining inner minimizations, similar to the steps of Example 7.5.

$$
\phi_k(x_k) = \min_{u_k} \left\{ \min_{x_{i+1}=a_d x_i + b_d u_i, \ i = k, \ldots, N-1} \left\{ q x_k^2 + r u_k^2 + \sum_{i=k}^{N-1} \left( q x_i^2 + r u_i^2 \right) + q_f x_N^2 \right\} \right\}
$$

$$
= \min_{u_k} \left\{ q x_k^2 + r u_k^2 \right\} + \min_{x_{i+1}=a_d x_i + b_d u_i, \ i = k, \ldots, N-1} \left\{ \sum_{i=k+1}^{N-1} \left( q x_i^2 + r u_i^2 \right) + q_f x_N^2 \right\}
$$

(7.14)

$$
= \min_{u_k} \left\{ q x_k^2 + r u_k^2 \right\} + \phi_{k+1}(x_{k+1})
$$

The third equality results from the observation that the inner min is the definition of $\phi_{k+1}(x_{k+1})$. Thus, one arrives at the much celebrated Hamilton-Jacobi-Bellman (HJB) equation:

$$
\phi_k(x_k) = q x_k^2 + \min_{u_k} \left\{ r u_k^2 + \phi_{k+1}(x_{k+1}) \right\} \quad \text{s.t.} \quad x_{k+1} = a_d x_k + b_d u_k
$$

(7.15)

where $\phi_N(x_N)$ is set equal to $q_f x_N^2$. This recursive relation tells us that the only thing needed to determine $u_k$ is the function $\phi_{k+1}(\cdot)$, and that $\phi_{k+1}(\cdot)$ can be determined from $\phi_{k+2}(\cdot)$.

Using the HJB, the solution to (7.1) can be determined analytically for any period $N$. Begin by calculating $\phi_{N-1}(x_{N-1})$.

$$
\phi_{N-1}(x_{N-1}) = q x_{N-1}^2 + \min_{u_{N-1}} \left\{ r u_{N-1}^2 + q_f \left( a_d x_{N-1} + b_d u_{N-1} \right)^2 \right\}
$$

(7.16)

Taking the derivative of the objective function with respect to $u_{N-1}$ and setting the result equal to zero gives:

$$
2 r u_{N-1} + 2 q_f \left( a_d x_{N-1} + b_d u_{N-1} \right) b_d = 0
$$

$$
\Rightarrow \ u_{N-1} = - l_{N-1} x_{N-1} \ \text{where} \ l_{N-1} = \frac{a_d q_f b_d}{r + b_d q_f b_d}
$$

(7.17)
Substituting this back into the minimization, \( \phi_{N-1}(x_{N-1}) \) is determined as

\[
\phi_{N-1}(x_{N-1}) = q x_{N-1}^2 + \left\{ r(-l_{N-1} x_{N-1}^2) + q f (a_d x_{N-1} - b_d l_{N-1} x_{N-1})^2 \right\}
\]

\[
= q x_{N-1}^2 + r l_{N-1}^2 x_{N-1}^2 + q f (a_d - b_d l_{N-1})^2 x_{N-1}^2
\]

\[
= [q + r l_{N-1}^2 + q f (a_d - b_d l_{N-1})^2] x_{N-1}^2
\]

\[
= p_{N-1} x_{N-1}^2
\]

Since \( \phi_{N-1}(x_{N-1}) \) has the same quadratic form as \( \phi_N(x_N) \), one can extrapolate the calculation of \( \phi_{N-2}(x_{N-2}) \) by replacing \( q_f \) with \( p_{N-1} \) in (7.17) and (7.18). The result is that \( u_{N-2} = -l_{N-2} x_{N-2} \) where \( l_{N-2} = (r + b_d p_{N-1}/b_d)^{-1} b_d p_{N-1} a_d \) and \( \phi_{N-2}(x_{N-2}) = p_{N-2} x_{N-2}^2 \). This procedure can be repeated to calculate all of the linear feedback policies: \( u_k = -l_k x_k, k = 0 \ldots N - 1 \).

Summarizing, we find the analytic procedure of Table 7.3 can be used to calculate the optimal sequence of linear feedbacks. After this procedure, one will have calculated the sequence \( l_k, k = 1 \ldots N - 1 \) and can then simulate the system \( x_{k+1} = (a_d - b_d l_k) x_k \) to finally arrive at the optimal set of inputs as \( u_k = -l_k x_k, k = 0 \ldots N - 1 \).

**Table 7.3. Dynamic programming solution of the scalar LQOC problem**

1) Set \( p_N = q_f \) and \( k = N - 1 \)
2) Calculate \( l_k = (r + b_d p_{k+1}/b_d)^{-1} b_d p_{k+1} a_d \)
3) Calculate \( p_k = q + r l_k^2 + p_{k+1}(a_d - b_d l_k)^2 \)
4) If \( k = 0 \), stop, otherwise set \( k = k - 1 \) and go to step 2)

**Table 7.4. Optimal LQOC feedback gains for Example 7.6.**

<table>
<thead>
<tr>
<th>( k )</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_k )</td>
<td>1.06</td>
<td>0.675</td>
<td>0.514</td>
<td>0.432</td>
<td></td>
</tr>
<tr>
<td>( p_k )</td>
<td>25</td>
<td>11.03</td>
<td>7.42</td>
<td>5.88</td>
<td>5.10</td>
</tr>
</tbody>
</table>

**Table 7.5. Optimal LQOC solution for Example 7.6.**

<table>
<thead>
<tr>
<th>( k )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_k )</td>
<td>100</td>
<td>73.4</td>
<td>50.9</td>
<td>31.2</td>
<td>13.2</td>
</tr>
<tr>
<td>( u_k )</td>
<td>-43.2</td>
<td>-37.7</td>
<td>-34.4</td>
<td>-32.9</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example 7.6.** Let us return to the scenario of Example 7.3 (\( a_d = 0.95, b_d = 0.5, q = 1, r = 5, q_f = 25, N = 4, \) and \( x_0 = 100 \)). Implementation of the procedure of Table 7.3 for these parameters is given in Table 7.4. Remember the procedure starts at \( k = 4 \) and works backwards in time, as suggested by the table. Using the initial condition \( x_0 = 100 \),
Table 7.5 was generated by simulating the system with the feedback sequence, \( l_k \), of Table 7.4. In this case, the procedure starts at \( k = 0 \). Comparison of Table 7.5 with Table 7.2, indicates that the dynamic programming method generates exactly the same input sequences as the batch approach.

For vector valued dynamic systems the development is nearly identical. In this case, the Hamilton-Jacobi-Bellman (HJB) equation is:

\[
\phi_k(x_k) = x_k^*Qx_k + \min_{u_k} \left\{ u_k^*Ru_k + \phi_{k+1}(x_{k+1}) \right\} \quad \text{s.t.} \quad x_{k+1} = A_d x_k + B_d u_k
\]

(7.19)

If we assume the cost-to-go function is quadratic (i.e., \( \phi_{k+1}(x_{k+1}) = x_{k+1}^*P_{k+1}x_{k+1} \)), one finds:

\[
\phi_k(x_k) = x_k^*Qx_k + \min_{u_k} \left\{ u_k^*Ru_k + (A_d x_k + B_d u_k)^* P_{k+1} (A_d x_k + B_d u_k) \right\}
\]

(7.20)

The minimization term can be evaluated by setting the partial derivative equal to zero:

\[
0 = \frac{\partial}{\partial u_k} \left\{ u_k^*Ru_k + (A_d x_k + B_d u_k)^* P_{k+1} (A_d x_k + B_d u_k) \right\}
\]

(7.21)

Solving for \( u_k \) one finds:

\[
u_k = -\left( R + B_d^*P_{k+1}B_d \right)^{-1} B_d^*P_{k+1}A_d x_k
\]

(7.22)

Substituting back into (7.20) results in:

\[
\phi_k(x_k) = x_k^* \left[ Q + L_k^*RL_k + (A_d - B_d L_k)^* P_{k+1} (A_d - B_d L_k) \right] x_k = x_k^*P_k x_k
\]

(7.23)

Thus, the following backwards in time recursion is found:

\[
P_N = Q_f
\]

(7.24)

\[
L_k = \left( R + B_d^*P_{k+1}B_d \right)^{-1} B_d^*P_{k+1}A_d
\]

(7.25)

\[
P_k = Q + L_k^*RL_k + (A_d - B_d L_k)^* P_{k+1} (A_d - B_d L_k)
\]

(7.26)

Along the way we have found each of the cost-to-go functions is: \( \phi_k(x_k) = x_k^*P_k x_k \). Then, looking back at the original definition of cost-to-go (the vector version of Equation 7.13), it is concluded that the value function of problem (7.2) is:

\[
\Phi_0(x_0) = \phi_0(x_0) = x_0^*P_0 x_0
\]

(7.27)

**Example 7.7.** Reconsider the scenario of Example 7.4. Using the MATLAB code of Table 7.6, the recursion of Equations (7.25)-(7.26) was implemented. The elements of the resulting matrix sequences are depicted in Figure 7.5. It is highlighted that the evolution is backwards in time and starts from \( k = 26 \) for \( P_k \) and \( k = 25 \) for \( L_k \).
Table 7.6: MATLAB code used in calculations for Example 7.7.

```matlab
clear

% Problem parameters
Ad=[0.941 0.192; -0.576 0.903]; Bd=[.020; 0.192];
Q=[1 0;0 1]; R=.5; Qf=zeros(2); N=26;

% Initialize arrays
NN=N+1; LL=zeros(NN-1,2);
PPPk=zeros(NN,3); PPPk(NN,:)=[Qf(1,1) Qf(1,2) Qf(2,2)];

% Calculate sequences
Pk=Qf
for k=NN-1:-1:1
  kk(k)=(k-1);
  LL(k,:)=inv(R+Bd'*Pk*Bd)*Bd'*Pk*Ad;
  Pk=Q+LL(k,:)'*R*LL(k,:)+(Ad-Bd*LL(k,:))'*Pk*(Ad-Bd*LL(k,:))
  PPPk(k,:)=[Pk(1,1) Pk(1,2) Pk(2,2)];
end

% Plot sequences
figure(1), plot(kk,LL), figure(2), plot([kk NN],PPPk)
```

Figure 7.5. Time evolution of the LQOC cost-to-go matrix $P_k$ (left) and the optimal linear feedback matrix $L_k$ (right) with $N = 26$

Expansion of Equation (7.26) will result in the following:

$$P_k = Q + L_k^*RL_k + A_d^*P_{k+1}A_d - L_k^*B_d^*P_{k+1}B_d + L_k^*B_d^*P_{k+1}B_dL_k$$
$$= A_d^*P_{k+1}A_d - L_k^*B_d^*P_{k+1}A_d - A_d^*P_{k+1}B_dL_k + L_k^* \left( R + B_d^*P_{k+1}B_d \right) L_k + Q \quad (7.28)$$

Then, substitution of Equation (7.25) into (7.28) will yield:

$$P_k = A_d^*P_{k+1}A_d - A_d^*P_{k+1}B_d \left( R + B_d^*P_{k+1}B_d \right)^{-1} B_d^*P_{k+1}A_d + Q$$
$$= A_d^* \left( P_{k+1} - P_{k+1}B_d \left( R + B_d^*P_{k+1}B_d \right)^{-1} B_d^*P_{k+1} \right) A_d + Q \quad (7.29)$$

Equation (7.29) is known as the **discrete-time Riccati equation** for the LQOC problem. This Riccati equation is the dual of the Kalman filter Riccati equation of Equation (6.76).

### 7.2 Infinite-time LQOC

The primary observation of the dynamic programming perspective is that the optimal sequence of inputs can be stated in the form of a linear feedback of the state. An additional
observation is that the linear feedback gain matrix will be fairly constant for the early portion the time period. For example, in the right plot of Figure 7.5 it is observed that $L_k$ is nearly constant for sample times less than 15. Thus, for nearly half of the time period the feedback will appear to be a regulator — a feedback policy is the same for all time. If one increases the time period of the LQOC problem, then the fraction of time in regulator mode will increase. For example, reconsider Example 7.7 but with $N = 51$. The resulting plots are given in Figure 7.6 and illustrate that the regulator mode will be for about three fourths of the time period. From these plots, it is easy to imagine what will happen for very large values of $N$ — the regulator mode will occur for nearly all of the time period. Thus, if the process is to be run indefinitely, then the appropriate feedback gain should be a regulator.

The infinite-time LQOC problem — also known as the Linear Quadratic Regulator (LQR) problem — is stated as:

$$\Phi(x_0) = \min_{u_0,u_1,...} \left\{ \sum_{k=0}^{\infty} \left( x_k^T Q x_k + u_k^T R u_k \right) \right\}$$

s.t. $x_{k+1} = A_d x_k + B_d u_k$, $k = 0, 1, ...$  \hfill (7.30)

The solution to the LQR problem is found as the limit of the finite-time problem. Let us begin by introducing the notation: $L_k^{(N)}$ and $P_k^{(N)}$ as the $k^{th}$ points of the iteration (7.25)-(7.26) with $P_N = Q_f$. Then, define new matrix variables $L_{N-k}^# = L_k^{(N)}$ and $P_{N-k}^# = P_k^{(N)}$, and substitute into (7.25)-(7.26) to arrive at the following forward in time recursion:

$$P_0^# = Q_f$$

$$L_k^# = \left( R + B_d^* P_{k-1}^# B_d \right)^{-1} B_d^* P_{k-1}^# A_d$$

$$P_k^# = Q + L_k^# R L_k^# + \left( A_d - B_d L_k^# \right)^* P_{k-1}^# \left( A_d - B_d L_k^# \right)$$

One can, then, take the limit as $k$ goes to infinity — which will require $N$ to also approach infinity since $N$ must be $\geq k$. If these limits are defined as $L = \lim_{k \to \infty} L_k^#$ and $P = $
\[
\lim_{k \to \infty} P^k_k, \text{ then each must satisfy}
\]
\[
L = \left( R + B_d^* P B_d \right)^{-1} B_d^* P A_d \tag{7.34}
\]
\[
P = Q + L^* R L + (A_d - B_d L)^* P (A_d - B_d L) = A_d^* \left( P - P B_d \left( R + B_d^* P B_d \right) B_d^* P \right) A_d + Q \tag{7.35}
\]

Relation (7.35) is known as the Algebraic Riccati Equation (ARE), while (7.34) is the feedback gain of the LQR. Given the solution to the ARE, the value function of the infinite-time LQOC problem is calculated as:

\[
\Phi(x_0) = x_0^* P x_0 \tag{7.36}
\]

**Example 7.8.** Let us return to the scenario of Example 7.6 \((a_d = 0.95, b_d = 0.5, q = 1, r = 5, q_f = 25, N = 4, \text{ and } x_0 = 100)\). Figure 7.7 illustrates the relationship between \(L_k\) and \(L^*_k\) and \(P_k\) and \(P^*_k\). The plot of Figure 7.8 shows the sequence \(P_k\) for various time periods and how \(P^*_k\) does not change with respect to \(N\) and seems to converge to the same point, somewhere near 4. The exact value can be calculated by application of Equation (7.35):

\[
P = a_d^2 P - a_d^2 P^2 b_d^2 / \left( r + b_d^2 P \right) + q \Rightarrow a_d^2 P^2 b_d^2 = \left( (a_d^2 - 1)P + q \right) \left( r + b_d^2 P \right)
\]
\[
\Rightarrow 0 = P^2 b_d^2 - \left[ (a_d^2 - 1) r + q b_d^2 \right] P - q r
\]
\[
\Rightarrow 0 = 0.25 P^2 + 0.2375 P + 5
\]
\[
\Rightarrow P = -0.475 \pm 4.4973
\]

Since we are only interested in the positive value, \(P = 4.0223\) is the solution of interest. It is additionally noted that \(q_f\) did not appear the calculation of \(P\). Figure 7.9 illustrates that the sequence will converge to the same value regardless of \(q_f\). 

![Figure 7.7. Comparison of \(L_k\) and \(L^*_k\) and \(P_k\) and \(P^*_k\) for Example 7.8](image)

In the matrix valued case, it is nearly impossible to perform calculations similar to those of Example 7.8 to determine the solution to the ARE. One option is to perform the iterations of (7.32)-(7.33) until the sequence converges. Another option is to use the MATLAB function ‘dare’. 


Example 7.9. Let us return to the scenario of Example 7.7. Small modifications of the code from Table 7.6 can be used to generate the plot of Figure 7.10, which illustrates that the solution to the ARE is independent of $Q_f$. Using the code of Table 7.7, the dare function finds that $P = \begin{bmatrix} 12.6 & 0.858 \\ 0.858 & 4.15 \end{bmatrix}$, which matches with the iterative solution found in Figure 7.10.
**Table 7.7:** Matlab code used in calculations for Example 7.9.

```matlab
clear
% Problem parameters
Ad=[0.941 0.192; -0.576 0.903]; Bd=[0.020; 0.192]; Q=[1 0 0 1]; R=.5;
% Find ARE solution
P=dare(Ad,Bd,Q,R)
```

While Examples 7.8 and 7.9 give antidotal evidence that the recursion (7.32)-(7.33) will converge, the following theorem gives appropriate conditions. Since Theorem 7.1 is essentially the dual of Theorem 6.3, the proof of Theorem 7.1 is immediate.

**Theorem 7.1.** Assume $R$ is positive definite and $Q$ is positive semi-definite. If $(A_d, B_d)$ stabilizable and $(A_d, Q)$ detectable, then the recursion (7.32)-(7.33) will converge generating a positive definite $P$ as the limit of $P^k$ and $L$ as the limit of $L^k$ and the matrix $(A_d - B_d L)$ will be stable.

![Figure 7.11](image1.png)

**Figure 7.11.** Comparison of LQOC and LQR solutions for Example 7.10. Solid LQOC and dashed LQR. Top left $-R=0.5$; top right $-R=5$; bottom left $-R=0.05$; bottom right $-manipulated variable

---

**Example 7.10.** Reconsider the scenario of Example 7.4 ($Q = I$, $Q_f = 0$, $N = 26$, $x_0 = [1 0]^*$ and $R = 0.5, 5$ and 0.05). The plots of Figure 7.11 illustrate that the trajectories of the LQOC and the LQR solutions are nearly identical. Notice how the plots had to be zoomed in for the $R = 0.5$ and 0.05 cases, just to be able to see a difference. Also, notice that the difference is mostly near the final time. However, this should be no surprise, since the backward recursion of the Riccati equation puts the transients of the controller...
7.3. LQOC in Continuous-time

In continuous-time, the LQOC problem is stated as

\[
\Phi(x_0) = \min_{u(t)} \left\{ \int_0^T (x^* Q x + u^* R u) \, dt + x^*(T) Q_f x(T) \right\}
\]

s.t. \( \dot{x} = Ax + Bu, \ t = [0, T], \ x(0) = x_0 \)

The solution to this finite-time problem is

\[
u(t) = -L(t)x(t) \quad (7.38)\]

\[
L(t) = R^{-1}B^*P(t) \quad (7.39)\]

\[
\frac{dP}{dt} = -A^*P - PA - Q + PBR^{-1}B^*P \quad (7.40)\]

\[
P(T) = Q_f \quad (7.41)\]

\[
\Phi(x_0) = x_0^*P(0)x_0 \quad (7.42)\]

Similar to the discrete-time case, the continuous-time solution requires a backwards solution of the Riccati equation (7.40) given the final condition of (7.41). This is then used to determine the feedback gain as a function of time and ultimately the feedback policy (7.38).

**Example 7.11.** Consider a simple integrator process \( \dot{x} = u \). If \( q = 9, r = 1, \) and \( q_f = 0 \), then the continuous-time Riccati equation is

\[
\frac{dP}{dt} = -9 + P^2; \quad P(T) = 0
\]

This quadratic ordinary differential equation is solved using separation of variables

\[
\frac{dP}{9 - P^2} = -dt \Rightarrow \int_{P(T)}^{P(t)} \frac{dP}{9 - P^2} = -\int_T^t dt
\]

\[
\Rightarrow \frac{1}{6} ln\left(\frac{3 + P}{3 - P}\right) \bigg|_0^{P(t)} = -(t - T) \Rightarrow ln\left(\frac{3 + P(t)}{3 - P(t)}\right) = -6(t - T)
\]

\[
P(t) = \frac{3(e^{-6(t-T)} - 1)}{(e^{-6(t-T)} + 1)}
\]

The plot of Figure 7.12 illustrates the curve for various values of \( T \).

The regulator case is simply the limit as \( T \) approaches infinity, which results in the steady-state version of the previous relations:

\[
u(t) = -Lx(t) \quad (7.43)\]

\[
L(t) = R^{-1}B^*P \quad (7.44)\]

\[
0 = -A^*P - PA - Q + PBR^{-1}B^*P \quad (7.45)\]

\[
\Phi(x_0) = x_0^*P x_0 \quad (7.46)\]
Figure 7.12. Riccati equation solution for Example 7.10

Equation (7.45) is the continuous-time version of the ARE, which can be written as:

\[ 0 = (A - BL)^*P + P(A - BL) + Q - PB R^{-1} B^*P \]  

(7.47)

This form provides a hint to the origin of the following theorem.

**Theorem 7.2.** Assume \( R \) is positive definite and \( Q \) is positive semi-definite. If \((A, B)\) stabilizable and \((A, Q)\) detectable, then (7.45) will have a positive definite solution \( P \) and the resulting \( L \) (from 7.44) will be such that \((A - BL)\) is stable.

**Example 7.12.** Reconsider the continuous-time mass-spring-damper of Example 7.4:

\[
A = \begin{bmatrix} 0 & 1 \\ -3 & -0.2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

Application of the ‘care’ function in MATLAB, with \( Q = I \) and \( R = 0.5 \) gives the following:

\[
P = \begin{bmatrix} 2.41 & 0.158 \\ 0.158 & 0.718 \end{bmatrix} \quad L = \begin{bmatrix} 0.3166 & 1.435 \end{bmatrix}
\]

If the system model is converted to discrete-time with \( \Delta t = 0.2 \) and the ‘dare’ function is applied (with \( Q = I \) and \( R = 0.5 \)) the following will result:

\[
P = \begin{bmatrix} 12.6 & 0.858 \\ 0.858 & 4.15 \end{bmatrix} \quad L = \begin{bmatrix} -0.1212 & 1.226 \end{bmatrix}
\]

Clearly, there is a substantial discrepancy. However, if one defines the discrete-time \( Q \) and \( R \) as \( Q_d = Q \Delta t \) and \( R_d = R \Delta t \), then the following will result from dare:

\[
P = \begin{bmatrix} 2.52 & 0.172 \\ 0.172 & 0.830 \end{bmatrix} \quad L = \begin{bmatrix} -0.1212 & 1.226 \end{bmatrix}
\]

In this case, the discrete-time \( P \) is much closer to the continuous-time version, and as one would expect \( L \) is unchanged by rescaling by both \( Q \) and \( R \) by the same amount. To arrive at the same linear feedback as the continuous-time, the sample-time will need to be reduced. The feedbacks with \( \Delta t = 0.02 \) and 0.002 are as follows:

\[
L_{0.02} = \begin{bmatrix} 0.2694 & 1.414 \end{bmatrix} \quad L_{0.002} = \begin{bmatrix} 0.3119 & 1.433 \end{bmatrix}
\]
Table 7.8: Matlab code used in calculations for Example 7.12.

```matlab
clear
%A = [0 1; -3 -0.2]; B = [0; 1]; G = [0; 1];
% Conversion to Discrete-time
dt = 0.02; Ndt = 200; ddt = dt / Ndt; sum = [0 0; 0 0];
for jjj = 1:Ndt; sum = sum + expm(A * jjj * ddt);
end
Ad = expm(A * dt); Bd = sum * B * ddt; Gd = sum * G * ddt;
[P, Lam, L] = care(A, B, eye(2), 0.5);
[P, Lam, L] = dare(Ad, Bd, eye(2), 0.5);
[P, Lam, L] = dare(Ad, Bd, eye(2) * dt, 0.5 * dt);
```

7.4 • Chapter Summary

The chapter begins by introducing the LQOC problem as an optimization based method to determine a sequence of manipulated variable inputs that will drive the process state toward the nominal operating condition (i.e., zero in deviation variables). The quadratic objective function weights of the LQOC indicate the urgency at which state variables should be driven to the origin. Since the finite-time LQOC is just quadratic program, the batch solution method can be readily applied. While the batch method is conceptually simple, it fails to uncover the rich solution structure engendered by the stage-wise nature of the LQOC. Subsequent investigation of the dynamic programming based solution method revealed that an analytic solution to the LQOC is possible. This solution was found to be in the form of a linear feedback, where the sequence of optimal feedback gains is determined through use of a second form of the Riccati equation (the dual of the Riccati equation associated with the optimal state estimator of Chapter 6). This analytic solution to the finite-time LQOC was also found to be integral to determining the solution to the infinite-time LQOC problem. This infinite-time problem is unique in the sense that the feedback gain is a constant for all time and is determined from the Algebraic Riccati Equation (ARE). In Section 7.3, the LQOC problem statement and solution are presented in the continuous-time framework.

For additional details and alternative solution methods for the LQOC problem one should consult Athans and Falb [135], Kirk [136], Russell [137] or Anderson and Moore [138]. Detailed discussions of dynamic programming can be found in Bellman and Kalaba [139] or Dreyfus [140].

Exercises

7.1. Consider the following optimization problem

\[ \Phi(x_0) = \min_{\mu_0} \left\{ 5x_0^2 + 2\mu_0^2 + x_1^2 \right\} \]

s.t. \( x_0 = 100 \)

\( x_1 = 0.95x_0 + 2\mu_0 \)

(i) Solve this problem manually by substituting the constraints into the objective function to arrive at a function of only one variable.

(ii) Using the MATLAB function ‘quadprog’, determine the solution of the unconstrained QP determined in part (i).
(iii) Using the MATLAB function ‘quadprog’, determine the solution of the equality constrained problem directly. In this case, the vector $\theta$ of Equation (7.3) should contain 2 variables $u_0$ and $x_1$.

(iv) Verify that all three methods generate the same solution.

7.2. Consider the following optimization problem

$$\Phi(x_0) = \min_{u_0,u_1} \left\{ 5x_0^2 + 2u_0^2 + 5x_1^2 + 2u_1^2 + x_2^2 \right\}$$

s.t. $x_0 = 100$
$$x_1 = 0.95x_0 + 2u_0$$
$$x_2 = 0.95x_1 + 2u_1$$

(i) Solve this problem manually by substituting the constraints into the objective function to arrive at a function of only two variables.

(ii) Using the MATLAB function ‘quadprog’, determine the solution of the unconstrained QP determined in part (i).

(iii) Using the MATLAB function ‘quadprog’, determine the solution of the equality constrained problem directly. In this case, the vector $\theta$ of Equation (7.3) should contain 4 variables $u_0$, $u_1$, $x_1$ and $x_2$.

(iv) Verify that all three methods generate the same solution.

7.3. Consider the following optimization problem

$$\Phi(x_0) = \min_{u_k} \left\{ \sum_{k=0}^{N-1} \left( 5x_k^2 + 2u_k^2 \right) + x_N^2 \right\}$$

s.t. $x_0 = 100$
$$x_{k+1} = 0.95x_k + 2u_k$$

Using the MATLAB function ‘quadprog’, determine the solution of the equality constrained problem with $N = 5$.

7.4. Reproduce the plots of Example 7.4, using the batch solution method.

7.5. Consider the following continuous-time model of a mass-spring damper:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -0.25 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

where the first state is mass position, the second is velocity and the manipulated variable is an applied force. Discretize the system using the sample and hold method (with $\Delta t = 0.25$). Using the MATLAB function ‘quadprog’, determine the solution to the deterministic LQOC problem (with $N = 25$, $Q_f = 0$ and initial condition $x_0 = [1 \ 0]^T$) for the following three cases: a) $Q = I$ and $R = 0.1$, b) $Q = I$ and $R = 1$ and c) $Q = I$ and $R = 10$.

7.6. Consider the following optimization problem

$$\Phi(x_0) = \min_{u_0,u_1} \left\{ 5x_0^2 + 2u_0^2 + 5x_1^2 + 2u_1^2 + x_2^2 \right\}$$

s.t. $x_0 = 100$
$$x_1 = 0.95x_0 + 2u_0$$
$$x_2 = 0.95x_1 + 2u_1$$
(i) Solve this problem using the method of Dynamic Programming.
(ii) Verify that the solution is the same as Exercise 7.2.

7.7. Reproduce the results of Example 7.6.

7.8. Reproduce the plots of Example 7.4, using the dynamic programming solution method.

7.9. Consider the following continuous-time model of a mass-spring damper:

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -0.25 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
\]

where the first state is mass position, the second is velocity, and the manipulated variable is an applied force. Discretize the system using the sample and hold method (with \(\Delta t = 0.25\)). Using the dynamic programming method, determine the solution to the deterministic LQOC problem (with \(N = 25\), \(Q_f = 0\) and initial condition \(x_0 = [1 \ 0]^T\)) for the following three cases: a) \(Q = I\) and \(R = 0.1\), b) \(Q = I\) and \(R = 1\) and c) \(Q = I\) and \(R = 10\). Verify these are the same as those found in Exercise 7.5.

7.10. Consider the following generalization of the scalar LQOC problem

\[
\Phi(x_0) = \min_{u_0,u_1,\ldots,u_{N-1}} \left\{ \sum_{k=0}^{N-1} \left( q x_k^2 + 2 s x_k u_k + r u_k^2 \right) + q_f x_N^2 \right\}
\]

s.t. \(x_{k+1} = a_d x_k + b_d u_k, \ k = 0 \ldots N - 1\)

\(x_0\) is a given parameter

In this case, the optimal controller and associate Riccati equation are

\[
u_k = -l_k x_k; \quad l_k = (s + b_d p_{k+1} a_d) / (r + b_d^2 p_{k+1})
\]

\[
p_k = a_d^2 p_{k+1} + q - (s + b_d p_{k+1} a_d)^2 / (r + b_d^2 p_{k+1}) ; \quad p_N = q_f
\]

(i) Using the method of dynamic programming, derive equations (2)-(3). Hint: the appropriate HJB equation would be:

\[
\phi_k(x_k) = q x_k^2 + \min_{u_k} \left\{ 2 s x_k u_k + r u_k^2 + \phi_{k+1}(x_{k+1}) \right\}
\]

s.t. \(x_{k+1} = a_d x_k + b_d u_k\)

and \(\phi_N(x_N) = q_f x_N^2\). If it is helpful, you may set \(a_d = b_d = 1\).

(ii) If \(N = 3\), determine the matrices one would need to give the MATLAB routine ‘quadprog’ to solve problem (1) numerically. You should leave \(a_d\), \(b_d\), \(q\), and \(s\) as parameters. Hint: Pay attention to the time \(k = 0\).

7.11. Derive the vector version of the generalized discrete-time ARE (i.e., with cross-terms in the objective function).

7.12. In this problem we consider the impact of sampling on the feedback gain. Consider the continuous-time process: \(\dot{x} = 2x + u\)

(i) Apply the infinite-time LQOC method to the continuous-time process to determine the feedback gain \(l\). Assume both \(q\) and \(r\) to be equal to one.

(ii) Using the Euler method, discretize the above system and recalculate the feedback gain by applying the discrete-time version of the infinite-time LQOC method. Fill in the following table.
7.13. Consider the following continuous-time model of a mass-spring damper:

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -0.25 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
\]

where the first state is mass position, the second is velocity and the manipulated variable is an applied force.

(i) Using the MATLAB function ‘care’, determine the optimal controller gains and eigenvalues of the closed-loop system for the following three cases: a) \(Q = I\) and \(R = 0.1\), b) \(Q = I\) and \(R = 1\) and c) \(Q = I\) and \(R = 10\).

(ii) Discretize the system using the sample and hold method (with \(\Delta t = 0.25\)). Using the MATLAB function ‘dare’, determine the optimal controller gains and closed-loop system eigenvalues for the following three cases: a) \(Q = I\) and \(R = 0.1\), b) \(Q = I\) and \(R = 1\) and c) \(Q = I\) and \(R = 10\). Comment on any similarities with part (i).

7.14. Reconsider Example 2.9. In deviation variable form the following system matrices will arise (see Example 2.9 for the interpretation of the \(x\), \(u\) and \(w\) variables).

\[
A_d = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad G_d = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

Determine the controller suggested by the infinite-time FSI stochastic LQOC problem. Do this for the following three cases:

(i) \(Q = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad R = [1] \)

(ii) \(Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad R = [1] \)

(iii) \(Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad R = [0.01] \)

For all three cases solve the ARE using two methods: (a) Recursive implementation of Equations (7.25)-(7.26) with \(N = 100\) and \(P_N = 0\). Then, \(L_0\) will be the steady-state optimal controller. (b) Use the MATLAB function ‘dare’.
7.15. The physical process of a frictionless, unit mass confined to a single spatial dimension can be described by the following state-space representation, where the first state is position and the second is velocity.

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
\]

If \( R = 1 \) and the solution to the continuous-time ARE is \( P = \begin{bmatrix} \sqrt{2} & 1 \\ 1 & \sqrt{2} \end{bmatrix} \), verify that \( Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \). Describe the meaning of this particular value of \( Q \). In other words, what are we communicating to the optimization problem by using this value of \( Q \).

7.16. Consider again the process of Exercise 7.15

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
\]

(i) Given the following infinite-time objective function

\[
\int_{0}^{\infty} \left( x^* x + r^2 u^2 \right) dt
\]

Verify that the solution of the continuous-time ARE is:

\[
P = \begin{bmatrix} \sqrt{2r+1} & r \\ r & r\sqrt{2r+1} \end{bmatrix}
\]

(ii) Determine the resulting steady-state feedback law.

(iii) Determine the values of \( r \) for which the closed-loop system exhibits oscillatory behavior.

7.17. Consider the following process

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
\]

\[
y = \begin{bmatrix} \alpha & 0 \\ 0 & (1-\alpha) \end{bmatrix} x
\]

along with the infinite-time objective function

\[
\int_{0}^{\infty} \left( y^* y + u^2 \right) dt
\]

(i) Determine the solution to the ARE for all values of \( 0 < \alpha \leq 1 \).

(ii) Determine the values of \( \alpha \) for which the closed-loop system exhibits oscillatory behavior. (At the very least evaluate at \( \alpha = 1 \) and \( \alpha = 0 \).)

(iii) If \( \alpha = 0 \), then the resulting solution to the ARE is not positive definite. Please show that an additional consequence of \( \alpha = 0 \) is that the closed-loop system is not stable.
(iv) Explain from the perspective of the objective function why the results of part (iii) occur when $\alpha = 0$. (Hint: consider the initial condition $x(0) = [1 \ 0]^*$)

(v) Assuming $\alpha = 0.01$ and the initial condition is $x(0) = [5 \ 5]^*$, sketch time plots of position and velocity. (Hint: You need not apply any equations or calculations. The concepts from your answer to part (iv) should suffice.)

7.18. Consider a continuous-time process: $\dot{x} = Ax + bu$. Prove the following statement: If there exists a positive definite solution, $P$, to the ARE: $0 = A^*P + PA + Q - PBR^{-1}B^*P$ and the matrix $Q$ is positive definite, then application of a controller $u = -R^{-1}B^*Px$ will result in a stable closed-loop system. (Hint: Recall the Lyapunov Theorem of Chapter 4.)
Chapter 8
Stochastic LQOC

In this chapter, the LQOC problem is extended to a stochastic framework. This extension will take two forms. The first, denoted the Full State Information (FSI) problem, assumes the feedback controller has perfect knowledge of the process state. As a bit of foreshadowing, it will be shown that the solution to the FSI stochastic LQOC problem is identical to the feedback form of the deterministic LQOC of Section 7.1.2. This rather convenient result is known as the Certainty Equivalence Principle. The second form of the stochastic LQOC assumes that only noise corrupted measurements are available for feedback and is denoted the Partial State Information (PSI) problem. Similar to the results of Section 4.2, it will be shown that the PSI stochastic LQOC problem possesses a Separation Principle, in that the PSI problem can be decomposed into a pair of independent design problems — the optimal control problem and the optimal estimator problem. It will then be shown that the separated control problem has a feedback solution identical to the FSI problem and is ultimately equal to the deterministic problem.

8.1 Stochastic LQOC with Full State Information

The discrete-time version of the finite-time FSI stochastic LQOC problem is defined as:

\[
\Psi^{(N)} = \min_{u_0(x_0), u_1(x_1), \ldots, u_{N-1}(x_{N-1})} \left\{ E \left[ \sum_{k=0}^{N-1} \left( x_k^T Q x_k + u_k^T R u_k \right) + x_N^T Q_f x_N \right] \right\} \quad (8.1)
\]

subject to

\[
x_{k+1} = A_d x_k + B_d u_k + G_d w_k, \quad k = 0 \ldots N - 1 \quad (8.2)
\]

Let us begin by noting the differences between Problem (8.1) and the deterministic version, Problem (7.2). The first and most important difference is that the process model includes the disturbance term \( G_d w_k \). As in previous chapters \( w_k \) is assumed to be zero mean white noise with a covariance of \( \Sigma_w \). If the actual disturbance is not white, then one should construct an appropriate shaping filter and then assemble a compound system to be used for controller design (recall Chapter 5). The primary impact of this disturbance term is that the process state will become a stochastic process. As such, if the objective function of (7.2) were used, then it would become a random variable (i.e., a function of the realization of \( w_k \)). Thus, to arrive at a meaningful problem statement, the expected value of the objective function is used in Problem (8.1). Due to the stochastic nature of the process, the optimization variables are changed from a simple sequence of numbers, \( u_k \), to a sequence functions, \( u_k(x_k) \), denoted as the feedback policies. In the current FSI
Chapter 8. Stochastic LQOC

problem, these are functions of the state, \( x_k \), which is assumed to be known. In the PSI case, these will change to functions of the measurements, \( y_k \). Finally, the notation used to indicate the value function is changed to \( \Psi^{(N)} \), where \( \Psi \) is used to highlight that this is the stochastic problem and the time period, \( N \), is made explicit. However, since the initial condition is a random variable, it is inappropriate for \( \Psi^{(N)} \) to be a function of the initial condition, though it will be a function of the mean of the initial condition, as we will see below.

As done in Section 7.1, we begin with the scalar single stage \((N = 1)\) problem, which should give some insight into the problem characteristics.

\[
\Psi^{(1)} = \min_{u_0(x_0)} \left\{ E\left[ q x_0^2 + r u_0^2 + q f x_1^2 \right] \right\} \quad \text{s.t. } x_1 = a_d x_0 + b_d u_0 + g_d \omega_0
\]  

(8.3)

It is highlighted that the expectation operator is with respect to all random variables and one can use the notion of a conditional density to convert to a nested expectation:

\[
E_{x_1,x_0} \left[ q x_0^2 + r u_0^2 + q f x_1^2 \right] = \int \int \left( q x_0^2 + r u_0^2 + q f x_1^2 \right) p(x_1, x_0) d x_1 d x_0
\]

\[
= \int \int \left( q x_0^2 + r u_0^2 + q f x_1^2 \right) p(x_1 | x_0) p(x_0) d x_1 d x_0
\]

(8.4)

\[
= \int \left[ \left( q x_0^2 + r u_0^2 \right) \int p(x_1 | x_0) d x_1 + \int q f x_1^2 p(x_1 | x_0) d x_1 \right] p(x_0) d x_0
\]

\[
= E_{x_0} \left[ q x_0^2 + r u_0^2 + E_{x_1|x_0} \left[ q f x_1^2 | x_0 \right] \right]
\]

where the identity \( \int p(x_1 | x_0) d x_1 = 1 \) for all \( x_0 \) is used in the fourth equality and the subscript on the expectation operators are to highlight the density function used. If the (8.4) identity is used in Problem (8.3), one finds

\[
\Psi^{(1)} = \min_{u_0(x_0)} \left\{ E_{x_0} \left[ q x_0^2 + r u_0^2 + E_{x_1|x_0} \left[ q f x_1^2 | x_0 \right] \right] \right\} \quad \text{s.t. } x_1 = a_d x_0 + b_d u_0 + g_d \omega_0
\]

(8.5)

\[
= E_{x_0} \left[ q x_0^2 + \min_{u_0(x_0)} \left\{ r u_0^2 + E_{x_1|x_0} \left[ q f x_1^2 | x_0 \right] \right\} \right] \quad \text{s.t. } x_1 = a_d x_0 + b_d u_0 + g_d \omega_0
\]

The exchange of the expectation and minimization operators should give pause. So, let us look a bit more closely.

\[
\min_{u_0(x_0)} \left\{ E_{x_0} \left[ r u_0^2 + E_{x_1|x_0} \left[ q f x_1^2 | x_0 \right] \right] \right\} = \min_{u_0(x_0)} \left\{ \int \int \left( r u_0^2 + E_{x_1|x_0} \left[ q f x_1^2 | x_0 \right] \right) p(x_0) d x_0 \right\}
\]

(8.6)

\[
= \int \left( \min_{u_0(x_0)} \left( r u_0^2 + E_{x_1|x_0} \left[ q f x_1^2 | x_0 \right] \right) \right) p(x_0) d x_0
\]

\[
= E_{x_0} \left[ \min_{u_0(x_0)} \left( r u_0^2 + E_{x_1|x_0} \left[ q f x_1^2 | x_0 \right] \right) \right]
\]
To understand why the second equality holds, think of the integral as a summation and consider an arbitrary function of two variables: \( f(a, b) \). Then, it should be clear that the following is true

\[
\min_{a(b)} \{ f(a, b_1) + f(a, b_2) \} = \min_{a(b)} \{ f(a(b_1), b_1) + f(a(b_2), b_2) \} \\
= \min_{a} \{ f(a, b_1) \} + \min_{a} \{ f(a, b_2) \}
\]

(8.7)

Finally, the problem can be stated as:

\[
\Psi^{(1)} = E_{x_0} \left[ q x_0^2 + \min_{u_0(x_0)} \left\{ r u_0^2 + E_{x_1|x_0} \left[ q_f x_1^2 \mid x_0 \right] \right\} \right]
\]

\[
= E_{x_0} \left[ q x_0^2 + \min_{u_0(x_0)} \left\{ r u_0^2 + E_{x_1|x_0} \left[ q_f (a_d x_0 + b_d u_0 + g_d w_0)^2 \mid x_0 \right] \right\} \right]
\]

\[
= E_{x_0} \left[ q x_0^2 + \min_{u_0(x_0)} \left\{ r u_0^2 + q_f (a_d x_0 + b_d u_0)^2 + E_{x_1|x_0} \left[ q_f (g_d w_0)^2 \mid x_0 \right] \right\} \right]
\]

(8.8)

where the cross terms containing \( w_0 \) (of the third equality) are zero due to the fact that \( a_d x_0 + b_d u_0 \) is a constant in the eyes of the conditional expectation and \( w_0 \) is zero mean. Then, from the necessary conditions for optimality one finds that:

\[
2 r u_0 + 2 q_f (a_d x_0 + b_d u_0) b_d = 0
\]

\[
\Rightarrow u_0(x_0) = -l_0 x_0 \quad \text{where} \quad l_0 = \frac{a_d q_f b_d}{r + b_d q_f b_d}
\]

(8.9)

Substituting back into (8.8) one finds:

\[
\Psi^{(1)} = E_{x_0} \left[ (q + r l_0^2 + q_f (a_d - b_d l_0)^2) x_0^2 \right] + q_f g_d^2 \Sigma_w
\]

\[
= p_0 E_{x_0} x_0^2 + q_f g_d^2 \Sigma_w = p_0 \left( \Sigma_{x,0} + \bar{x}_0^2 \right) + q_f g_d^2 \Sigma_w
\]

\[
= p_0 \bar{x}_0^2 + p_0 \Sigma_{x,0} + q_f g_d^2 \Sigma_w
\]

(8.10)

where \( p_0 = q + r l_0^2 + q_f (a_d - b_d l_0)^2 \). With the exception of the terms \( p_0 \Sigma_{x,0} + q_f g_d^2 \Sigma_w \), the above development should be reminiscent of the dynamic programming results of Section 7.1.2. In fact, if the initial condition is known perfectly (i.e., \( \Sigma_{x,0} = 0 \) and \( \bar{x}_0 = x_0 \)) and \( \Sigma_w = 0 \), then (8.10) would be identical to (7.18) with \( N = 1 \). The more remarkable observation is that the feedback policy of (8.9) is identical to (7.17), with \( N = 1 \), when \( \Sigma_w \neq 0 \) and the initial condition is not known. In fact, the feedback policy is actually independent of \( \Sigma_w \).

Now consider the two stage scalar problem. In this case, the identity analogous to (8.4) is

\[
E_{x_1|x_0} \left[ q x_0^2 + r u_0^2 + q x_1^2 + r u_1^2 + q_f x_2^2 \right]
\]

\[
= E_{x_0} \left[ q x_0^2 + r u_0^2 + E_{x_2|x_1|x_0} \left[ q x_2^2 + r u_1^2 + E_{x_1|x_0} \left[ q_f x_2^2 \mid x_1, x_0 \right] \mid x_0 \right] \right]
\]

\[
= E \left[ q x_1^2 + r u_1^2 + E \left[ q x_2^2 \mid x_1 \right] \right]
\]

(8.11)
where the third expression of (8.11) is the same as the second, but with less detailed notation. Then, the two-stage problem can be stated as

\[
\Psi^{(2)} = E \left[ q x_2^2 + \min_{u_2(x_2)} \left\{ ru_2^2 + E \left[ q x_1^2 + \min_{u_1(x_1)} \left\{ ru_1^2 + E \left[ q f \mid x_1 \right] \mid x_0 \right\} \right] \mid x_2 \right\} \right] \tag{8.12}
\]

where \( x_1 = a_d x_0 + b_d u_0 + g_d \omega_0 \) and \( x_2 = a_d x_1 + b_d u_1 + g_d \omega_1 \) are of course enforced, but dropped from the notation for convenience. Now define

\[
\Psi^{(2)} = E [ \phi_0 (x_0) ] \tag{8.13}
\]

\[
\phi_0 (x_0) = q x_0^2 + \min_{u_0(x_0)} \left\{ ru_0^2 + E \left[ \phi_1 (x_1) \mid x_0 \right] \right\} \tag{8.14}
\]

\[
\phi_1 (x_1) = q x_1^2 + \min_{u_1(x_1)} \left\{ ru_1^2 + E \left[ \phi_2 (x_2) \mid x_1 \right] \right\} \tag{8.15}
\]

\[
\phi_2 (x_2) = q f x_2^2 \tag{8.16}
\]

Then, a substitution of (8.14)-(8.16) into (8.13), indicates that (8.13) is the same as (8.12). This nesting of optimization stages leads to the following stochastic version of the HJB equation:

\[
\phi_k (x_k) = q x_k^2 + \min_{u_k(x_k)} \left\{ ru_k^2 + E \left[ \phi_{k+1} (x_{k+1}) \mid x_k \right] \right\} \tag{8.17}
\]

In fact, (8.17) can be used for any time period \( N \) by defining \( \phi_N (x_N) = q f x_N^2 \). An alternate, but more descriptive, definition of \( \phi_k (x_k) \) is as the cost-to-go function:

\[
\phi_k (x_k) = q x_k^2 + \min_{u_k(x_k)} \left\{ ru_k^2 + E \left[ \sum_{i=1}^{N-1} (q x_i^2 + ru_i^2) + q f x_N^2 \right] \right\} \tag{8.18}
\]

s.t. \( x_{i+1} = a_d x_i + b_d u_i + g_d \omega_i, \ i = k \ldots N - 1 \)

The vector version of the HJB equation is easily concluded to be:

\[
\phi_k (x_k) = x_k^T Q x_k + \min_{u_k(x_k)} \left\{ u_k^T R u_k + E \left[ \phi_{k+1} (x_{k+1}) \mid x_k \right] \right\} \tag{8.19}
\]

s.t. \( x_{k+1} = A_d x_k + B_d u_k + G_d \omega_k \)

The solution procedure begins by evaluating (8.19) at \( k = N - 1 \):

\[
\phi_{N-1} (x_{N-1}) = x_{N-1}^T Q x_{N-1} + \min_{u_{N-1}(x_{N-1})} \left\{ u_{N-1}^T R u_{N-1} + E \left[ x_{N-1}^T Q x_{N-1} \mid x_{N-1} \right] \right\} \tag{8.20}
\]

s.t. \( x_{N-1} = A_d x_{N-1} + B_d u_{N-1} + G_d \omega_{N-1} \)

Evaluation of \( E \left[ x_N^T Q x_N \mid x_{N-1} \right] \) yields:

\[
E \left[ (A_d x_{N-1} + B_d u_{N-1} + G_d \omega_{N-1})^T Q_N (A_d x_{N-1} + B_d u_{N-1} + G_d \omega_{N-1}) \mid x_{N-1} \right] = (A_d x_{N-1} + B_d u_{N-1})^T Q_N (A_d x_{N-1} + B_d u_{N-1}) + E \left[ \omega_{N-1} G_d^* Q f G_d \omega_{N-1} \right] \tag{8.21}
\]

where the cross terms containing \( \omega_{N-1} \) are zero due to the fact that \( A_d x_{N-1} + B_d u_{N-1} \) is a constant in the eyes of the conditional expectation and \( \omega_{N-1} \) is zero mean. The last
equality of (8.21) is due to the following:

\[ E \left[ w_{N-1}^{*} G_d^2 Q_f G_d w_{N-1} \right] = E \left[ Tr \left( w_{N-1}^{*} G_d^2 Q_f G_d w_{N-1} \right) \right] \]

\[ = E \left[ Tr \left( Q_f G_d w_{N-1}^{*} G_d^2 \right) \right] \]

\[ = Tr \left\{ Q_f G_d E \left[ w_{N-1}^{*} w_{N-1} \right] G_d^2 \right\} \]

\[ = Tr \left\{ Q_f G_d \Sigma_w G_d^2 \right\} \]

(8.22)

Substituting (8.21) into (8.20) and rearranging yields:

\[ \psi_{N-1}(x_{N-1}) = x_{N-1}^{*} Q x_{N-1} + Tr \left\{ Q_f G_d \Sigma_w G_d^2 \right\} \]

\[ + \min_{u_{N-1}(x_{N-1})} \left\{ u_{N-1}^{*} R u_{N-1} + (A_d x_{N-1} + B_d u_{N-1})^T Q_f (A_d x_{N-1} + B_d u_{N-1}) \right\} \]

(8.23)

Now the minimization term can be evaluated by setting the partial derivative equal to zero:

\[ 0 = \frac{\partial}{\partial u_{N-1}} \left\{ u_{N-1}^{*} R u_{N-1} + (A_d x_{N-1} + B_d u_{N-1})^T Q_f (A_d x_{N-1} + B_d u_{N-1}) \right\} \]

\[ = 2R u_{N-1} + 2B_d^T Q_f (A_d x_{N-1} + B_d u_{N-1}) \]

(8.24)

Solving for \( u_{N-1} \) one finds:

\[ u_{N-1} = -L_{N-1} x_{N-1} \quad \text{where} \quad L_{N-1} = (R + B_d^T Q_f B_d)^{-1} B_d^T Q_f A_d \]

(8.25)

Substituting back into (8.23) results in:

\[ \psi_{N-1}(x_{N-1}) \]

\[ = x_{N-1}^{*} \left( Q + L_{N-1}^* R L_{N-1} + (A_d - B_d L_{N-1})^T Q_f (A_d - B_d L_{N-1}) \right) x_{N-1} + Tr \left\{ Q_f G_d \Sigma_w G_d^2 \right\} \]

\[ = x_{N-1}^{*} P_{N-1} x_{N-1} + \gamma_{N-1} \]

(8.26)

where

\[ P_{N-1} = Q + L_{N-1}^* R L_{N-1} + (A_d - B_d L_{N-1})^T Q_f (A_d - B_d L_{N-1}) \]

(8.27)

\[ L_{N-1} = (R + B_d^T P_{N-1} B_d)^{-1} B_d^T P_{N-1} A_d \]

(8.28)

\[ \gamma_{N-1} = Tr \left\{ Q_f G_d \Sigma_w G_d^2 \right\} \]

(8.29)

and \( P_N = Q_f \). Returning to (8.19), but now with \( k = N - 2 \) one finds:

\[ \psi_{N-2}(x_{N-2}) = x_{N-2}^{*} Q x_{N-2} + \min_{u_{N-2}(x_{N-2})} \left\{ u_{N-2}^{*} R u_{N-2} + E \left[ \psi_{N-1}(x_{N-1}) \mid x_{N-2} \right] \right\} \]

(8.30)

Due to the quadratic structure of \( \psi_{N-1}(x_{N-1}) \), evaluation of (8.30) will be nearly identical to the steps used to evaluate (8.20). Thus,

\[ \psi_{N-2}(x_{N-2}) = x_{N-2}^{*} P_{N-2} x_{N-2}^* + \gamma_{N-2} + \gamma_{N-1} \]

\[ P_{N-2} = Q + L_{N-2}^* R L_{N-2} + (A_d - B_d L_{N-2})^T P_{N-1} (A_d - B_d L_{N-2}) \]

(8.31)

\[ L_{N-2} = (R + B_d^T P_{N-1} B_d)^{-1} B_d^T P_{N-1} A_d \]

(8.32)

\[ \gamma_{N-2} = Tr \left\{ Q_f G_d \Sigma_w G_d^2 \right\} \]

(8.33)
and $\gamma_{N-2} = \text{Tr} \left\{ P_{N-1} G_d \Sigma_w G_d^* \right\}$. In the general case, one finds:

$$
\psi_k(x_k) = x_k^* P_k x_k + \sum_{i=k}^{N-1} \gamma_i
$$

(8.34)

$$
P_k = Q + L_k^* R L_k + (A_d - B_d L_k)^\dagger P_{k+1} (A_d - B_d L_k)
$$

(8.35)

$$
L_k = (R + B_d P_{k+1} B_d^*)^{-1} B_d P_{k+1} A_d
$$

(8.36)

$$
P_N = Q_f \quad \text{and} \quad \gamma_k = \text{Tr} \left\{ P_{k+1} G_d \Sigma_w G_d^* \right\}
$$

(8.37)

Evaluation of (8.34) at $k = 0$ yields the minimum cost as requested by (8.1)

$$
\Psi^{(N)} = E[\psi_0(x_0)] = x_0^* P_0 x_0 + \text{Tr} \left\{ P_0 \Sigma_{x,0} \right\} + \sum_{k=0}^{N} \gamma_k
$$

(8.38)

It must be highlighted that the recursive relations of (8.35)-(8.36) are identical to the relations of (7.25)-(7.26). And, similar to the deterministic LQOC, the controller gains, $L_k$, are calculated independent of the trajectory, most notably independent of the disturbance realization, $w_k$. However, the realization will have a large impact on the state and manipulated variable sequences $x_k$ and $u_k$, since $x_{k+1} = A_d x_k + B_d u_k + G_d w_k$ and $u_k = -L_k x_k$. Thus, the closed-loop system will be:

$$
x_{k+1} = (A_d - B_d L_k) x_k + G_d w_k, \quad k = 1 \ldots N - 1
$$

(8.39)

### Table 8.1. Stochastic LQOC trajectory for Example 8.1.

<table>
<thead>
<tr>
<th>$k$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k$</td>
<td>100</td>
<td>76.6</td>
<td>53.1</td>
<td>33.2</td>
<td>16.4</td>
</tr>
<tr>
<td>$u_k$</td>
<td>-43.2</td>
<td>-39.4</td>
<td>-35.8</td>
<td>-35.0</td>
<td>-</td>
</tr>
</tbody>
</table>

**Example 8.1.** Let us return to the scenario of Example 7.6 ($a_d = 0.95$, $b_d = 0.5$, $q = 1$, $r = 5$, $q_f = 25$, $N = 4$, and $x_0 = 100$) and assume $g_d = 1$ and $\Sigma_w = 4$. Since the recursion of (8.35)-(8.37) is identical to that of the deterministic LQOC, one should implement the procedure of Table 7.3 and arrive again at the parameters of Table 7.4. Using the initial condition $x_0 = 100$ and appropriate random numbers for $w_k$, Table 8.1 was generated by simulating the system with the feedback sequence, $l_k$, of Table 7.4. Comparison of Table 8.1 with Table 7.5, indicates that the stochastic trajectory is similar to the deterministic, but certainly different.

### Table 8.2. MATLAB code used in calculations for Example 8.1.

```matlab
clear
ad=0.95; bd=1/2; q=1; r=5; qf=25; gd=1; Sigw=4; x0=100;
N=5; l=zeros(1,N-1); p=zeros(1,N); p(N)=qf;
for k=N-1:-1:1
    l(k)=inv(r+bd*p(k+1)*bd)*bd*p(k+1)*ad;
    p(k)=q+r*l(k)^2+p(k+1)*(ad-bd*l(k))^2;
end
```
8.1. Stochastic LQOC with Full State Information

\[ l, p, x(1) = x_0; \text{randn('state',2^6-1)}; \]
\[ \text{for } k=1:N-1 \]
\[ w_k = \sqrt{\text{Sigw}} \times \text{randn}; \]
\[ u(k) = -l(k) \times x(k); \]
\[ x(k+1) = a_d \times x(k) + b_d \times u(k) + g_d \times w_k; \]
\[ \text{end} \]
\[ x, u \]

Figure 8.1. Comparison of deterministic (solid) and stochastic (dashed) LQOC trajectories for Example 8.2.

Figure 8.2. Comparison of deterministic and stochastic LQOC trajectories with a longer time period for Example 8.2.
Example 8.2. Reconsider the scenario of Example 7.4 \((Q = I, Q_f = 0, N = 26, x_0 = [1 \ 0]^\top, R = 0.5)\), with \(G_d = B_d\) and \(\Sigma_w = 2.5\). The plots of Figure 8.1 compare the trajectories of the deterministic (solid) and stochastic (dashed) LQOC. Clearly, there is a similarity between the two, with the obvious difference being the random nature of the stochastic case. If the time period is extended \((N = 100)\), then Figure 8.2 illustrates a much greater difference. Specifically, the deterministic trajectory will converge to zero, while the stochastic continues to be excited.

As in the deterministic case, the recursions of (8.35)-(8.36) will converge as the time period, \(N\), approaches infinity.

\[
P = Q + L^* RL + (A_d - B_d L)^* P (A_d - B_d)
\]

(8.40)

\[
L = (R + B_d^* P B_d)^{-1} B_d^* P A_d
\]

(8.41)

It is additionally noted that substitution of (8.41) into (8.40) results in the more compact (and more commonly encountered) algebraic Riccati equation (see ‘dare’ in MATLAB):

\[
P = A_d^* P A_d + Q - A_d^* P B_d (B_d^* P B_d + R)^{-1} B_d^* P A_d
\]

(8.42)

While the linear feedback is found to converge, the value function \(\Psi(N)\) does not, due to the summation of \(\gamma_k\). To resolve this minor issue, one can define a new infinite time problem as the limit of the finite time problem divided by the time period of the problem:

\[
\Psi = \lim_{N \to \infty} \frac{1}{N} \Psi(N) = \lim_{N \to \infty} \frac{1}{N} \left( \bar{x}_0 P \bar{x}_0 + Tr \{P \Sigma_w \} + \sum_{k=0}^{N} \gamma_k \right)
\]

(8.43)

\[
= \lim_{N \to \infty} \frac{1}{N} \left( \sum_{k=0}^{N} Tr \{P_k G_d \Sigma_w G_d^* \} \right) = Tr \left\{ P G_d \Sigma_w G_d^* \right\}
\]

where \(P\) is from (8.42).

Example 8.3. Reconsider the scenario of Example 8.2 \((Q = I, Q_f = 0, x_0 = [1 \ 0]^\top, R = 0.5, G_d = B_d\) and \(\Sigma_w = 2.5\)), but with a much larger time period, \(N = 1000\). While the time series plots of Figure 8.3 are intuitive, it will be more instructive to consider the phase plane (scatter) plots of Figure 8.4. The top two plots of Figure 8.4 correspond to the plots of Figure 8.3, \(R = 0.5\). The middle and bottom plots of Figure 8.4 show the impact of changing \(R\) to 50 and 0.005, respectively. With \(R = 50\) the LQOC is reluctant to use the manipulated variable. The impact of this tuning is that the manipulated variable remains very small and as such the state variables are essentially the same as the open-loop process. In contrast, if \(R = 0.005\), then the manipulated variable is very active and quite successful at reducing the variance of both state variables.
An alternative to the numerically determined scatter plot is an analytic calculation of the Expected Dynamic Operating Region (EDOR). Specifically, one may use the following closed-loop versions of the steady-state covariance equations to determine the covariance matrix of the performance outputs $z$.

\[ \Sigma_x = (A_d - B_d L) \Sigma_x (A_d - B_d L)^T + G_d \Sigma_w G_d^T \]  
\[ \Sigma_z = (D_x - D_u L) \Sigma_x (D_x - D_u L)^T + D_w \Sigma_w D_w \]

In Example 8.3 one would define $z_k = D_x x_k + D_u u_k + D_w w_k$ where

\[
D_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D_u = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{and} \quad D_w = 0
\]

Then, using the LQOC gain from the $R = 0.005$ case, along with the plotting methods of Section 5.3.5, the EDOR ellipse of the closed-loop system was added to the bottom plot of Figure 8.4.

It is interesting to note that the infinite-time LQOC problem could have been stated
Figure 8.4. FSI Stochastic LQOC scatter plots for Example 8.3.

as follows:

\[
\Psi = \min_{u(x)} \left\{ \lim_{N \to \infty} \left\{ \frac{1}{N} \sum_{k=0}^{N-1} \left( x_k^* Q x_k + u_k^* R u_k \right) \right\} \right. \\
\quad \text{s.t. } x_{k+1} = A_d x_k + B_d u(x_k) + G_d w_k, \ k = 0 \ldots N - 1 \right\} \\
= \min_{u(x)} \left\{ \lim_{k \to \infty} \left\{ E \left[ x_k^* Q x_k + u_k^* R u_k \right] \right\} \right. \\
\quad \text{s.t. } x_{k+1} = A_d x_k + B_d u(x_k) + G_d w_k \right\} \\
= \min_{u(x)} \left\{ \lim_{k \to \infty} \left\{ T \left[ \Sigma x_{k, k} \right] + T \left[ \Sigma u_{k, k} \right] \right\} \right. \\
\quad \text{s.t. } x_{k+1} = A_d x_k + B_d u(x_k) + G_d w_k \right\} \\
= \min_{u(x)} \left\{ \lim_{k \to \infty} \left\{ T \left[ \Sigma x_{k, k} \right] + T \left[ \Sigma u_{k, k} \right] \right\} \right. \\
\quad \text{s.t. } x_{k+1} = A_d x_k + B_d u(x_k) + G_d w_k \right\} \\
\tag{8.47}
\]
8.2. Stochastic LQOC with Partial State Information

The finite-time PSI LQOC problem is stated as:

$$\Psi^{(N)} = \min_{u_0(y_0), u_1(y_1), \ldots, u_N(y_N)} \left\{ E \left[ \sum_{k=0}^{N-1} \left( x_k^* Q x_k + u_k^* R u_k \right) + x_N^* Q f x_N \right] \right\}$$

(8.48)

subject to

$$x_{k+1} = A_d x_k + B_d u_k + G_d w_k, \quad k = 0 \ldots N - 1$$

(8.49)

$$y_k = C x_k + v_k, \quad k = 0 \ldots N - 1$$

(8.50)

where \( \eta_k \) is the innovations process of Section 6.3.3 and is known to be white noise with a covariance \( \Sigma_{w,k} = C \Sigma_{x,k}^+ + \Sigma_v \). Also, recall from Section 6.3.3 that the error signal is orthogonal to any estimator (i.e., \( E \left[ (x_k - \hat{x}_k) \hat{x}_k^* \right] = E \left[ e_k^* \hat{x}_k^* \right] = 0 \)). Thus, evaluation of any state term of the summation gives:

$$E \left[ x_k^* Q x_k \right] = E \left[ \left( (x_k - \hat{x}_k) + \hat{x}_k^* \right)^* Q \left( (x_k - \hat{x}_k) + \hat{x}_k^* \right) \right]$$

$$= E \left[ (e_k + \hat{x}_k)^* Q \left( e_k + \hat{x}_k \right) \right]$$

$$= E \left[ e_k^* Q e_k \right] + E \left[ \hat{x}_k^* Q \hat{x}_k \right] + E \left[ \hat{x}_k^* Q e_k \right] + E \left[ e_k^* Q \hat{x}_k \right]$$

(8.54)

From (8.54) it is observed that \( x_k \) can be removed entirely from the objective function, indicating that the constraints of (8.52) are not actually needed. Thus, the PSI problem can be further simplified to:

$$\Psi^{(N)} = \min_{u_0(\hat{x}_0), u_1(\hat{x}_1), \ldots, u_N(\hat{x}_{N-1})} \left\{ E \left[ \sum_{k=0}^{N-1} \left( \hat{x}_k^* Q \hat{x}_k + u_k^* R u_k \right) + \hat{x}_N^* Q f \hat{x}_N \right] + \sum_{k=0}^{N-1} Tr \left\{ Q \Sigma_{e,k} \right\} \right\}$$

(8.55)

subject to

$$\hat{x}_{k+1} = A_d \hat{x}_k + B_d u_k + K_{k+1} \eta_{k+1}, \quad k = 0 \ldots N - 1$$

(8.56)
Since Problem (8.55) is of exactly the same form as (8.1) — the only difference being the nomenclature ($\hat{x}_k$ replacing $x_k$ and $K_{k+1} \eta_{k+1}$ replacing $G_d \omega_k$) — the linear feedback solution will be identical to that of the FSI LQOC, which was just concluded in Section 8.1 to be identical to the feedback solution of the deterministic LQOC. More specifically, the optimal controller gains, $L_k$, are identical for both cases: $u_k = -L_k \hat{x}_k$ for FSI and $u_k = -L_k \hat{x}_k$ for PSI. This much celebrated result is known as the Separation Principle and indicates that the PSI controller design problem can be solved as two separate steps — the FSI optimal control problem (to determine $L_k$) and the PSI optimal estimation problem (to determine $\hat{\eta}_k$).

Identical to the previous subsection, the solution to infinite-time PSI problem is the linear feedback calculated from (8.40)-(8.41). To arrive at a finite value function for the infinite-time problem it stated as the limit of the finite-time problem divided by the time period:

$$\Psi = \lim_{N \to \infty} \frac{1}{N} \Psi^{(N)}$$

$$= \lim_{N \to \infty} \frac{1}{N} \left( \bar{x}_0 P \bar{x}_0 + Tr \left\{ P_0 \Sigma_{x,0} \right\} + \sum_{k=0}^{N} Tr \left\{ P_k G_d \Sigma_w G_d^* \right\} + \sum_{k=0}^{N} Tr \left\{ Q \Sigma_{x,k} \right\} \right)$$

$$= Tr \left\{ PG_d \Sigma_w G_d^* \right\} + Tr \left\{ Q \Sigma_e \right\}$$

(8.57)

where $P$ is from (8.42) and $\Sigma_e$ is the steady-state error covariance from (6.86) or (6.87).

In the PSI case, it is a bit more challenging to calculate the closed-loop covariance matrix of the performance outputs. One option is to form a compound system similar to that of Section 4.2. For example, using (8.49) and (6.84) one could construct:

$$\begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} = \begin{bmatrix} A_d & -B_d L \\ KC & (A_d - B_d L)(I - KC) \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} + \begin{bmatrix} G_d & 0 \\ 0 & K \end{bmatrix} \begin{bmatrix} w_k \\ \omega_k \end{bmatrix}$$

(8.58)

$$z_k = \begin{bmatrix} D_x & -D_u L \end{bmatrix} \begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} + D_w \omega_k$$

(8.59)

Then, application of standard covariance analysis to this compound system will give the covariance matrix, $\Sigma_z$, needed to plot the EDOR, using the methods of Section 5.3.5. However, a simpler approach can be found by exploiting orthogonality and the innovations process. Begin by considering the state estimator in innovations form with $u = -L \hat{x}_k$:

$$\hat{x}_{k+1} = (A_d - B_d L) \hat{x}_k + K \eta_{k+1}$$

(8.60)

Recall that $\eta_k$ is white noise with a covariance $\Sigma_\eta = C \Sigma_e^+ C^* + \Sigma_v$. Then, covariance analysis with respect to (8.60) gives:

$$\Sigma_z = (A_d - B_d L) \Sigma_z (A_d - B_d L)^* + K (C \Sigma_e^+ C^* + \Sigma_v) K^*$$

$$= (A_d - B_d L) \Sigma_z (A_d - B_d L)^* + \Sigma_e^+ C^* (C \Sigma_e^+ C^* + \Sigma_v)^{-1} C \Sigma_e^+$$

(8.61)

$$= (A_d - B_d L) \Sigma_z (A_d - B_d L)^* + (\Sigma_e^* - \Sigma_v)$$

where the second equality is from (6.88) and the third is from (6.87). Similarly, the performance output equation can be written as

$$z_k = D_x x_k - D_u L \hat{x}_k + D_w \omega_k$$

$$= (D_x - D_u L) \hat{x}_k + D_u \hat{e}_k + D_w \omega_k$$

(8.62)
Then, noting that $\omega_k$ is white noise and that $\hat{x}_k$ and $e_k$ are orthogonal to each other, one finds:

$$\Sigma_z = (D_x - D_u L)\Sigma_{\hat{x}_k} (D_x - D_u L)^* + D_x \Sigma_e D_x^* + D_w \Sigma_w D_w^* \quad (8.63)$$

Using (8.61) and (8.63), along with the plotting methods of Section 5.3.5, one can easily display the EDOR ellipse of the closed-loop system.

![Figure 8.5: Stochastic LQOC scatter plots for Example 8.4.](image)

**Example 8.4.** Reconsider the scenario of Example 8.3 ($Q = I$, $Q_f = 0$, $x_0 = [1 \ 0]^*$, $N = 1000$, $G_d = B_d$, $\Sigma_w = 2.5$ and $R = 0.005$). Let us additionally assume that both states are measured ($C = I$) and $\Sigma_v = 0.025 I$. Since the measurement noise is quite small,
we should expect performance that is about the same FSI case. The top two plots of Figure 8.5 confirm this expectation, with the resulting EDOR just a bit larger than the FSI case. The remaining plots of Figure 8.5 illustrate that performance will degrade as the measurement noise increases. However, it should be emphasized that all three cases use the same LQOC feedback policy. The difference in the plots stems from the fact that \( \tilde{x}_k \) is different for each case.

### 8.3 - Stochastic LQOC with PSI and Delay

In many cases, the sample-time of the controller is determined by the time required to compute the control action. That is, the sample-time is selected to be as small as possible and the limiting factor is that selection of a smaller sample-time will result in the controller not being able to complete its computations in time for the control action to be implemented. Thus, if the sample-time is ‘small’, then once a measurement is taken the controller will use most of the given sample-time interval for calculations. Then, at the beginning of the next sample-time it will implement the control action. In essence, there will be a delay (a computational delay) associated with the calculation of the control actions. In the context of the stochastic LQOC problem, this computational delay manifests as the manipulated variable no longer having the measurement vector available to it at the current time (the time of implementation). It will, however, have all of the previous measurements, including the measurement from the previous time-step. Thus, the PSI stochastic LQOC problem with computational delay is stated as:

\[
\Psi^{(N)+} = \min_{u_k(y_{0:k-1}), u_k(y_{0:k-1})} \left\{ E \left[ \sum_{k=0}^{N-1} \left( x_k^* Q x_k + u_k^* R u_k \right) + x_N^* Q_f x_N \right] \right\}
\]

subject to

\[
x_{k+1} = A_d x_k + B_d u_k + G_d w_k, \quad k = 0 \ldots N - 1
\]

\[
y_k = C x_k + v_k, \quad k = 0 \ldots N - 1
\]

The difference being that the manipulated variable can only be a function of the past measurements: \( u_k(y_{0:k-1}) \). However, since the one-step predictor is a linear function of only past measurements (i.e., \( \hat{x}_k^+(y_{0:k-1}) \)) an equivalent statement is that the manipulated variable is a function of the one-step prediction: \( u_k(\hat{x}_k^+) \). Also, recall that the one-step predictor enjoys orthogonality (i.e., \( E \left[ (x_k - \hat{x}_k^+) \hat{x}_k^+ \right] = E \left[ e_k^+ \hat{x}_k^+ \right] = 0 \)) and thus:

\[
E \left[ x_k^* Q x_k \right] = E \left[ (e_k^+ + \hat{x}_k^+) \right] Q (e_k^+ + \hat{x}_k^+)
\]

\[
= E \left[ e_k^+ Q e_k^+ \right] + E \left[ e_k^+ Q \hat{x}_k^+ \right] + E \left[ \hat{x}_k^+ Q e_k^+ \right] + E \left[ \hat{x}_k^+ Q \hat{x}_k^+ \right]
\]

\[
= E \left[ e_k^+ Q e_k^+ \right] + E \left[ \hat{x}_k^+ Q \hat{x}_k^+ \right] = T r \left\{ Q \Sigma_{e,k}^+ \right\} + E \left[ \hat{x}_k^+ Q \hat{x}_k^+ \right]
\]

Thus, the PSI problem can be stated as:

\[
\Psi^{(N)+} = \min_{u_k(\hat{x}_k^+), u_k(\hat{x}_k^+), \ldots, u_{N-1}(\hat{x}_{N-1}^+)} \left\{ E \left[ \sum_{k=0}^{N-1} \left( \hat{x}_k^+ Q \hat{x}_k^+ + u_k^* R u_k \right) + \hat{x}_N^+ Q_f \hat{x}_N^+ \right] \right\}
\]

subject to

\[\hat{x}_{k+1}^+ = A_d \hat{x}_k^+ + B_d u_k + K_d^+ \eta_k, \quad k = 0 \ldots N - 1\]
8.3. Stochastic LQOC with PSI and Delay

Given this form, we once again conclude that the optimal solution is just the feedback generated by the deterministic LQOC problem. Of course, the value function will be a bit larger as illustrated by the infinite-time case, where $\Sigma^e$ replaces $\Sigma_e$

$$
\Psi^+ = \lim_{N \to \infty} \frac{1}{N} \Psi^{(N)} + Tr \left\{ PG_d \Sigma_w G_d^* \right\} + Tr \left\{ Q \Sigma^e \right\} 
$$

(8.70)

In addition, the calculation of the performance output covariance is slightly different. Specifically, the closed-loop system of interest is:

$$
\hat{x}_{k+1} = (A_d - B_d L) \hat{x}_k + K^+ \eta_k
$$

(8.71)

$$
z_k = (D_x - D_u L) \hat{x}_k + D_x e^+_k + D_w w_k
$$

(8.72)

Then, covariance analysis leads to:

$$
\Sigma_{\hat{x}} = (A_d - B_d L) \Sigma_{\hat{x}} (A_d - B_d L)^* + K^+ (C \Sigma^e \Sigma^e + \Sigma_v) K^{**}
$$

$$
= (A_d - B_d L) \Sigma_{\hat{x}} (A_d - B_d L)^* + A_d \Sigma^e \Sigma^e (C \Sigma^e \Sigma^e + \Sigma_v)^{-1} C \Sigma^e A_d^*
$$

(8.73)

$$
= (A_d - B_d L) \Sigma_{\hat{x}} (A_d - B_d L)^* + A_d (\Sigma^e - \Sigma_v)^{-1} A_d^*
$$

$$
\Sigma_v = (D_x - D_u L) \Sigma_{\hat{x}} (D_x - D_u L)^* + D_x \Sigma^e D_x^* + D_w \Sigma_w D_w^*
$$

(8.74)

Example 8.5. Reconsider the scenario of Example 8.3 ($Q = I$, $Q_f = 0$, $x_0 = [1 \ 0]^*$, $N = 1000$, $G_d = B_d^*$, $\Sigma_w = 2.5$ and $R = 0.005$), but this time select $C = [1 \ 0]$ and $\Sigma_v =$
0.0025). Figures 8.6 and 8.7 illustrate the impact of the selected measurement information structure (FSI, PSI without delay and PSI with delay). Clearly, the two PSI cases result in larger state variances. However, the variance of the manipulated variable is reduced. This must be due to the fact that \( \hat{x}_k \) and \( \hat{x}_k^+ \) are different, since the controller gain is the same for all three cases. The MATLAB code used to generate the plots of Figures 8.6 and 8.7 is given in Table 8.3.

Table 8.3. MATLAB code used in calculations for Example 8.5.

```matlab
clear

% Mass spring damper model
Ad=[0.941 0.192; -0.576 0.903]; Bd=[0.020; 0.192]; Gd=Bd;
Dx=[1 0; 0 1]; Du=[0; 0]; Q=[1 0; 0 1]; R=0.005; C=[1 0];
dt=0.2; NN=200/dt; Sigw=2.5; Sigv=0.0025*eye(1);

% Find LQOC Gain
P=dare(Ad,Bd,Q,R); L=inv(R+Bd'*P*Bd)*Bd'*P*Ad;

% Find Optimal Estimation Gains
Sige_plus=dare(Ad'*C',Gd*Sigw*Gd'*Sigv);
K_plus=Ad*Sige_plus*C'*inv(C*Sige_plus*C)+Sigv;
Sige=inv(inv(Sige_plus)+C'*inv(Sigv)*C); K=Sige*C'*inv(Sigv);

% Generate White Noise Sequences
randn('state',2^6-1); ww=sqrt(Sigw)*randn(NN,1); vv=sqrt(Sigv)*randn(1,NN);

% Simulte with FSI
tt=zeros(1,NN+1); xx=zeros(2,NN+1); uu=zeros(1,NN); xx(:,1)=[1;0];
for kk=1:NN
    tt(kk)=(kk-1)*dt;
    uu(:,kk)=-L*xx(:,kk); xx(:,kk+1)=Ad*xx(:,kk)+Bd*uu(:,kk)+Gd*ww(kk);
end
tt(NN+1)=(NN)*dt; uu(:,NN+1)=-L*xx(:,NN+1);
figure(1); plot(xx(2,:),xx(1,:),'k');
figure(2); plot(uu(1,:),xx(1,:),'k');

% Simulte with PSI and without computational delay
xx=zeros(2,NN+1); xhat=zeros(2,NN+1); xhat_plus=zeros(2,NN+1);
for kk=1:NN
    yyyk=C*xx(:,kk)+vv(:,kk);
exhat(:,kk)=xhat_plus(:,kk)+K*(yyyk-C*xhat_plus(:,kk));
    uu(:,kk)=-L*xhat(:,kk); xx(:,kk+1)=Ad*xx(:,kk)+Bd*uu(:,kk)+Gd*ww(kk);
exhat_plus(:,kk+1)=Ad*xhat(:,kk)+Bd*uu(:,kk);
end
```
8.4 Stochastic LQOC in Continuous-time

In the continuous-time framework both the Certainty Equivalence and Separation Principles hold true. Thus, to arrive at the appropriate feedback policy one need only consult Section 7.3 to find the optimal feedback gain and if a PSI problem, consult Section 6.1 to construct the optimal estimate. The one point of interest is to calculate the covariance matrix of the performance output. If the problem has a FSI feedback structure, then the closed-loop system of interest is:

\[
\dot{x} = (A - BL)x + Gw
\]  (8.75)

\[
z = (D_x - D_u L)x
\]  (8.76)

Then, covariance analysis leads to:

\[
0 = (A - BL)\Sigma_x + \Sigma_x (A - BL)^* + GS_w G^* 
\]  (8.77)

\[
\Sigma_x = (D_x - D_u L)\Sigma_x (D_x - D_u L)^* 
\]  (8.78)

If the problem has a PSI feedback structure, then the closed-loop system of interest is:

\[
\dot{\hat{x}} = (A - BL)\hat{x} + K\eta 
\]  (8.79)

\[
z = (D_x - D_u L)\hat{x} + D_x e
\]  (8.80)

Then, recalling that \( K = \Sigma_e C^* S_v^{-1} \) and \( S_\eta = S_v \), covariance analysis leads to:

\[
0 = (A - BL)\Sigma_\hat{x} + \Sigma_\hat{x} (A - BL)^* + \Sigma_e C^* S_v^{-1} \Sigma_e 
\]  (8.81)

\[
\Sigma_x = (D_x - D_u L)\Sigma_x (D_x - D_u L)^* + D_x \Sigma_e D_x^*
\]  (8.82)
Example 8.6. Recall the continuous-time model of Example 7.4. Assume $Q = I$, $R = 0.005$, $G = B$, $S_w = 0.5$ and a FSI feedback structure. Figure 8.8 illustrates the outstanding performance that can be achieved when the instantaneous feedback of the continuous time case is applied and how that performance will quickly degrade in the discrete-time case. If we now consider the PSI case in continuous-time (assume $C = [1 \ 0]$ and $S_v = 0.0005$). Figure 8.9 illustrates the performance degradation one should expect when going from FSI to PSI. Figure 8.10 illustrates the impact of sample time in the PSI case. As one would expect the computational delay case is quite sensitive to sample time, while the no delay case is nearly identical to the continuous-time case, even with a fairly large sample time of 0.1. The MATLAB code used to generate the covariance matrices is given in Table 8.4.
8.4. Stochastic LQOC in Continuous-time

Figure 8.10. PSI Stochastic LQOC EDOR plots for Example 8.6. Top plots are without computational delay and bottom plots are with delay.

Table 8.4. MATLAB code used in calculations for Example 8.6.

```matlab
clear

% Mass-Spring-Damper Model In Continuous-time
A=[0 1; -3 -.2]; B=[0; 1]; Sw=0.5;
Dx=[1 0; 0 0]; Du=[0; 0]; C=[1 0]; Sv=0.0005;
Q=[1 0; 0 1]; R=0.005;

% Conversion to discrete-time
dt=0.2; Ndt=200; ddt=dt/Ndt; sum=[0 0; 0 0];
for jjj=1:Ndt; sum=sum+expm(A *jjj*ddt);
end
Ad=expm(A*dt); Bd=sum *B*ddt; Gd=sum *G*ddt;
Sigw=Sw/dt; Sigv=Sv/dt;
Qd=Q *dt; Rd=R *dt;

% LQOC gain in continuous-time
[P,Lam,L]=care(A,B,Q,R);

% Output covariance in continuous-time with FSI
Sigx=lyap(A-B*L,G*Sw*G');
Sigz_FSI_CT=(Dx-Du*L)*Sigx*(Dx-Du*L)';

% Output covariance in continuous-time with PSI
[Sige,Lam,K]=care(A',C',G*Sw*G',Sv); K=K';
Sigxhat=lyap(A-B*L,Sige*C' *inv(Sv)*C*Sige);
Sigz_PSI_CT=(Dx-Du*L)*Sigxhat*(Dx-Du*L)'+Dx*Sige*Dx';

% LQOC gain in discrete-time
[Pd,Lamd,Ld]=dare(Ad,Bd,Qd,Rd);

% Output covariance in discrete-time with FSI
Sigx=dlyap(Ad-Bd*Ld,Gd*Sigw*Gd');
```
\[ \text{Sigz\_FSI\_DT} = (\text{Dx} - \text{Du}\cdot\text{Ld}) \cdot \text{Sigx} \cdot (\text{Dx} - \text{Du}\cdot\text{Ld})' \]

\% Optimal estimator gains
[Sige\_plus, Lam, K\_plus] = dare(Ad', C', Gd\_Sigw\_Gd', Sigv); K\_plus = K\_plus';
Sige = inv(inv(Sige\_plus) + C'\cdot inv(Sigv) \cdot C); K = Sige \cdot C' \cdot inv(Sigv);

\% Output covariance in discrete-time with PSI and no delay
Sigxhat = dlyap(Ad-Bd\_Ld, Sige\_plus - Sige);
Sigz\_PSI\_noD = (\text{Dx} - \text{Du}\cdot\text{Ld}) \cdot \text{Sigxhat} \cdot (\text{Dx} - \text{Du}\cdot\text{Ld})' + \text{Dx} \cdot \text{Sige} \cdot \text{Dx}

\% Output covariance in discrete-time with PSI and with delay
Sigxhat\_plus = dlyap(Ad-Bd\_Ld, Ad\_\text{(Sige\_plus - Sige)} \cdot Ad');
Sigz\_PSI\_withD = (\text{Dx} - \text{Du}\cdot\text{Ld}) \cdot \text{Sigxhat\_plus} \cdot (\text{Dx} - \text{Du}\cdot\text{Ld})' + \text{Dx} \cdot \text{Sige\_plus} \cdot \text{Dx}'

**8.5 • Chapter Summary**

This chapter focuses on the stochastic version of the LQOC problem. This version is distinct in the sense that a white noise disturbance is continually exciting the process state. Using again the method of dynamic programming it was shown that the optimal feedback gain for the FSI version of the stochastic LQOC problem is identical to that of the deterministic problem. This notion of certainty equivalence was shown to hold regardless of disturbance size, \( \Sigma_v \), or direction, \( G_d \). In the PSI version of the stochastic LQOC problem it was shown that the optimal feedback policy is a function of the optimal state estimate (i.e., the separation principle can be applied) and the optimal feedback gains are again equal to those of the deterministic case. In the PSI with computational delay case, it was shown that the the optimal feedback gains are again equal to those of the deterministic case, but the feedback policy is now a function of the one-step predictor.

It should be emphasized that despite the diversity of LQOC problem formulations (deterministic, FSI stochastic, PSI stochastic and PSI stochastic with computational delay) the feedback gains for each case are the same – the only difference being that the state, state estimate or one-step prediction is used for feedback. While many herald this result as the crowning jewel of optimal control theory, it does seem a bit counter-intuitive. That is, one would expect that the presences of disturbances along with the characteristics of those disturbances should influence the gain of the feedback controller. It should be highlighted that the certainty equivalence result hinges on the fact that a linear process is assumed and that the objective is quadratic. If either of these assumptions is removed, then the result will fail to hold. In Part III, we will consider a broader perspective of the stochastic control problem. Within this new framework we will find that disturbance size and direction will have a strong influence on the “optimal” feedback gain.

As one would expect, much of the material on the stochastic LQOC problem can be found in other texts. For additional details on the problem, please consult Astrom [142], Kushner [142], Stengel [104], Balakrishnan [120], Burl [105] or Speyer and Chung [143].

**Exercises**

8.1. Consider the following continuous-time process.

\[
\begin{align*}
\dot{x}_1 &= u \\
\dot{x}_2 &= -x_1 + w
\end{align*}
\]
Exercises 265

Design a full state information linear quadratic stochastic optimal controller using the following ARE:

\[ 0 = A^*P + PA + Q - PBR^{-1}B^*P, \]

which defines a controller

\[ u = -Lx \]

where \( L = R^{-1}B^*P. \)

(i) If \( R = 1 \), determine the value of \( Q \) such that \( P \) will be equal to \[
\begin{bmatrix}
2 & -1 \\
-1 & 2
\end{bmatrix}
\]

(ii) Determine the resulting controller.

(iii) If the controller of part (ii) is used, determine the eigenvalues of the closed-loop process.

(iv) Is the original plant stable? Is it stabilizable?

(v) Using the closed-loop system of part (iv), determine the standard deviation of each state variable as well as the standard deviation of the manipulated variable. Assume the disturbance, \( w \), is a zero mean white noise process with a spectral density of 4.

8.2. Consider the following scalar continuous-time stochastic process: \( \dot{x} = u + w \) where \( w \) is a zero mean white noise process with spectral density \( S_w = 2 \).

(i) Assuming a full state information feedback structure, \( u = -Lx \), calculate a feedback gain, \( L \), such that the steady-state variance of \( x \) is less than or equal to 0.5

(ii) Assuming a full state information feedback structure, \( u = -Lx \), calculate a feedback gain, \( L \), such that the steady-state variance of \( u \) is less than or equal to 2

(iii) Now consider the infinite-time LQOC problem, and identify objective function weights (\( Q \) and \( R \)) such that the controller generated by the LQOC problem is identical to the controller of part (i).

8.3. Reproduce the plots of Example 8.3.

8.4. Reproduce the plots of Example 8.4.

8.5. Reproduce the plots of Example 8.5.

8.6. Consider the following continuous-time model of a mass-spring damper:

\[
\begin{align*}
\dot{x} &= \begin{bmatrix} 0 & 1 \\ -1 & -0.25 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w \\
z &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \end{bmatrix} w
\end{align*}
\]

where the first performance output is mass position and the second is the force applied by the manipulated variable. Assume \( w(t) \) is a stationary white noise process with zero mean and a power spectral density: \( S_w = 0.1 \).

(i) Discretize the system using the sample and hold method (with \( \Delta t = 0.25 \)).

(ii) Using an initial condition \( x_0 = [0 \ 0]^T \) and assuming a FSI structure, simulate the process using the 3 controllers of Exercise 7.13(ii). (Make one plot for each performance output that includes the 3 trajectories associated with the 3 controllers.) Based on these plots, qualitatively comment on the on the performance output standard deviations as a function of \( R \) values. A simulation time of 100 time units (400 sample points) should be sufficient.

(iii) Using the MATLAB function ‘dlyap’, calculate the steady-state standard deviation of the two performance outputs for each of the controllers determined is problem 7-13(ii). Make a plot of \( \sigma_z^{(1)} \) vs. \( \sigma_z^{(2)} \). While this plot will
only have three points, it should give a quantitative description of the trends observed in part (ii).

(iv) Determine a value of $R$ such that the steady-state standard deviation of the manipulated variable is 0.15.

8.7. Consider the following scalar, open-loop *unstable*, continuous-time process.

\[ \dot{x} = x + u + w \]
\[ y = x + v \]

where $w$ and $v$ are zero mean white noise processes with spectral densities $S_w = 1$ and $S_v = 1$.

(i) Determine the optimal estimator gain for this system as well as the estimation error variance one should expect from the application of this gain.

(ii) If the manipulated variable is set to $u = -l \hat{x}$, then the state estimator will be governed by

\[ \dot{\hat{x}} = (1 - l)\hat{x} + K \eta \]

where $K$ is the optimal estimator gain of part 1 and $\eta$ is the innovation process, which is a zero mean white noise process with spectral density $S_\eta$. It is also noted that application of the optimal filter gain gives rise to the following identity: $\Sigma_x = \Sigma_{\hat{x}} + \Sigma_e$. Determine the controller gain, $l$, required to achieve a state variance, $\Sigma_x$, of 4. Using this controller, determine the variance of the manipulated variable.

8.8. Consider the following scalar, discrete-time process:

\[ x_{k+1} = a_d x_k + b_d u_k + g_d w_k \]

where $w_k$ is a zero mean white noise processes with variance $\Sigma_w$.

(i) Assume a full state information structure, $u_k = -l x_k$, and prove that the smallest possible steady-state, state variance is $\Sigma_x = g_d^2 \Sigma_w$. (It is not enough to identify a controller that yields this steady-state variance. You must prove that this is the smallest one could possibly achieve.)

(ii) Show that the discrete-time, infinite-time LQOC will yield the controller of part (i), if $Q = 1$ and $R = 0$. (Hint: for this case the solution of the ARE is irrelevant.)

8.9. Consider the following continuous-time process.

\[ \dot{x} = x - u + 2w \]
\[ y = x + v \]

where $w$ and $v$ are a zero mean, white noise processes with spectral densities $S_w = 0.2$ and $S_v = 0.04$. Determine the controller advocated by the continuous-time PSI LQOC with objective function weights $Q = 1$ and $R = 3$.

(i) Calculate the optimal controller and estimator gains: $L$ and $K$.

(ii) Determine $\Sigma_x$, the closed-loop variance of the state, as well as $\Sigma_u$, the closed-loop variance of the manipulated variable.
8.10. Consider the following continuous-time, unstable, scalar process:

\[
\begin{align*}
\dot{x} &= 0.5x - 2u + 0.5w \\
z &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\
y &= x + v
\end{align*}
\]

where \(w\) and \(v\) are stationary white noise processes with zero mean and spectral densities \(S_w = 24\) and \(S_v = 3\).

(i) If \(Q = 2\) and \(R = 4\), determine the controller advocated by the stochastic Full State Information (FSI) infinite-time LQOC problem. Show that the closed-loop system is stable.

(ii) Assume the FSI controller of part (i) is used. Then, determine the covariance matrix associated with \(z\).

(iii) Determine the steady-state optimal estimator gain associated with this process. If this gain were used, what would the estimation error variance be?

(iv) Assume the controller of the stochastic Partial State Information (PSI) infinite-time LQOC problem (with \(Q = 2\) and \(R = 4\)) is used. Determine the steady-state covariance matrix associated with \(z\).

8.11. Reconsider the process of Exercise 8.10. If the process model is converted to discrete-time using the Euler method (with a sample time of 0.2), the following model would result:

\[
\begin{align*}
x_{k+1} &= 1.1x_k - 0.4u_k + 0.1w_k \\
z_k &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k \\
y_k &= x_k + v_k
\end{align*}
\]

where \(w_k\) and \(v_k\) are stationary white noise sequences with zero mean and variances \(\Sigma_w = 120\) and \(\Sigma_v = 15\).

(i) If \(Q = 0.4\) and \(R = 0.8\), determine the controller advocated by the stochastic FSI infinite-time LQOC problem. You may assume the solution to the associated Algebraic Riccati Equation is \(P = 2.3\). Using this FSI controller determine the steady-state covariance matrix associated with \(z_k\).

(ii) Determine the steady-state gain of the one-step predictor for this process. You may assume the solution to the associated Algebraic Riccati Equation is \(\Sigma^+ = 6.9\).

(iii) Determine the steady-state gain of the optimal estimator for this process. If this gain were used, what would the estimation error variance be?

(iv) Assume the controller of the stochastic PSI infinite-time LQOC problem (with \(Q = 0.4\) and \(R = 0.8\)) is used. Determine the steady-state covariance matrix associated with \(z_k\).

(v) Assume the controller of part (iv) is used, but with computational delay. Determine the steady-state covariance matrix associated with \(z_k\).
Chapter 9

Model Predictive Control

Our primary interest in Model Predictive Control (MPC) is that such controllers have the ability to observe point-wise-in-time constraints (i.e., constraints that are enforced at each instant of time). The need for constraint enforcement stems from the fact that all physical systems must (at some point) consider such limitations. For example, an actuator element will be required to observe maximum and minimum bounds. Consider a simple valve, where the position of this valve can be no greater than fully open and no less than fully closed. Similarly, limits on the response time of an actuator can be enforced by putting bounds on the derivative of the control action. As we will see, the more interesting class of constraints are those with respect to state variables (or outputs that are a function of the state). Such constraints are usually created by the control engineer so as to satisfy indirect control objectives. For example, one may want to limit plant operation to a subset of the state space. This subset could be defined as the region in which the process model is guaranteed to be valid. Consider the model of a spring, $F = -kx$. Clearly, this model will not be valid for large values of $x$. A more important scenario is when the constraints represent safety limits. Continuing the spring example, if $x$ is too large, then the spring may fail. In a chemical system, these safety limits usually manifest as maximum temperatures, pressures or other limits on species concentrations.

Unfortunately, the mathematical framework of the LQOC does not allow for an incorporation of point-wise-in-time constraints (i.e., incorporation constraints precludes the development of an analytic solution). MPC is able to get around this theoretical issue by combining two novel concepts that are both computational in nature. The first is that the batch solution method for the LQOC can be easily modified to incorporate point-wise-in-time constraints. However, the batch approach is inherently an open-loop formulation. That is, the batch approach assumes that the initial state is known, but does not expect to receive any additional measurements during the time period of the problem. Thus, the second novel concept of MPC is to create an element of feedback through application of the so called ‘receding-horizon’ framework.

9.1 - The Receding-Horizon Framework

The basic idea behind MPC is to utilize a dynamic model to make predictions about future outcomes (including potential constraint violations). Based on these predicted outcomes, the manipulated variable is selected. To highlight the predictive aspect of MPC, two time indices are utilized. The index $k$ represents actual time, while the index $i$
denotes predictive time. Specifically, $x_{i|k}$, $i = k \ldots k + N$ is the sequence of state predictions, indexed by $i$, but determined at the current time $k$. Thus, a linear predictive model can be compactly stated as

\begin{align}
x_{i+1|k} &= A_d x_{i|k} + B_d u_{i|k}, \quad i = k \ldots k + N - 1 \\
z_{i|k} &= D_x x_{i|k} + D_u u_{i|k}, \quad i = k \ldots k + N \\
z_{i|k}^{\min} &\leq z_{i|k} \leq z_{i|k}^{\max}, \quad i = k \ldots k + N \\
x_{k|k} &= \hat{x}_k
\end{align}

(9.1) - (9.4)

The parameter $\hat{x}_k$ is the estimate of the state, $x_k$, which is governed by the actual process:

$$x_{k+1} = f_d(x_k, u_k, \omega_k), \quad k = 0 \ldots$$

Equation (9.5) suggests that we should not expect the predicted trajectory to be exactly the same as that of the actual process. In addition to likely being nonlinear, we expect the process to be excited by disturbances. In the unlikely event of having perfect measurements, the state estimate, $\hat{x}_k$, can be replaced by the true state, $x_k$. However, in spite of all the potential errors, we should expect the estimate of (9.4) to be reasonably close to the actual state.

Given this predictive model, the first objective is to select a sequence, $u_{i|k}$, $i = k \ldots k + N - 1$, such that the constraints of (9.3) are satisfied. Since it is likely that more than one sequence $u_{i|k}$ is capable of satisfying these constraints, the selection process is cast as a staged optimization problem with an objective function typically equal to that of the LQOC.

$$\Phi(\hat{x}_k) = \min_{u_{i|k}, u_{i+1|k}, \ldots, u_{i+N|k}} \left\{ \sum_{i=k}^{k+N-1} \left( x_{i|k}^T Q x_{i|k} + u_{i|k}^T R u_{i|k} \right) + x_{i+N|k}^T Q_f x_{i+N|k} \right\}$$

(9.6)

subject to \(9.1\) - \(9.4\)

(9.7)

Once an optimal sequence of predicted inputs, $u_{i|k}$, has been calculated, the controller does something that seems counterintuitive — the MPC algorithm utilizes only the first value, $u_{k|k}$, of the optimal sequence $u_{i|k}$, $i = k \ldots k + N - 1$. That is, the actual manipulated variable at time $k$ is set equal to the first step of the prediction: $u_k = u_{k|k}$. While this approach may seem like a waste of computational effort, the other elements of $u_{i|k}$, will play an important role, especially if a constraint violation is imminent. The primary reason for implementing only the first time-step is to introduce feedback into the algorithm. Specifically, as measurements about the process become available (during each time-step), one will find that the actual state is not where the predictive model predicted — recall Equation (9.5). Thus, the appropriate action is to replace the initial condition of problem (9.6), namely $x_{k|k}$, with the current estimate of the state, $\hat{x}_k$. In summary, the receding-horizon algorithm of MPC is as follows: At $k = 0$, $u_{0|0}$ is determined by (9.6), and $u_0$ is set to this value. Then, at the next time step ($k = 1$) the initial condition of the MPC calculation, $x_{1|1}$, is set equal to the estimated value of the state, $\hat{x}_1$. Then, the process is repeated by setting $u_1$ equal to $u_{1|1}$. The basic idea is that we know that the predicted trajectory, $x_{i|k}$, contains errors. So, before starting to solve Problem (9.6), we update the initial condition, $x_{k|k}$, with an estimate based on the latest measurement information, which is $\hat{x}_k$. 
A second reason to apply only the first time-step of the prediction is to reduce computational effort. Consider a scenario in which the process of interest is to be run for a long period of time. Furthermore, assume the model is perfect and all future disturbances are known (i.e., there is no discrepancy between the realized state and the predictions). In
Chapter 9. Model Predictive Control

In this case, it would seem reasonable to implement the entire open-loop policy generated at the first time step, \( u_{i|0} \). However, if the time period is very large and the number of state and manipulated variables is large, then the optimization problem one would need to solve would be enormous — potentially intractable. The alternative is to decouple the prediction time period from the operational time period. In this case, the prediction time indicates how much of the future the controller will consider, which could be significantly smaller than the operational period. Then, at each time step, when time index, \( k \), is increased, to \( k + 1 \), the final point of the prediction is also be increased by one. Typically, the prediction time period is denoted as the prediction horizon. This is analogous to a ship on the ocean. The ship can see as far as the horizon (to identify icebergs), travel toward the horizon (hopefully avoiding icebergs), but will never reach the horizon. This analogy illustrates a possible origin of the receding-horizon (or rolling-horizon) terminology commonly used to describe MPC.

Example 9.1. Reconsider the scalar process of Example 7.3 (\( a_d = 0.95, b_d = 0.5 \) and \( x_0 = 100 \)) along with the LQOC parameters of the same example (\( q = 1, r = 5, q_f = 25, N = 4 \)). If these parameters are assembled within an MPC framework, the problem would be stated as:

\[
\Phi(x_k) = \min_{u_{i|k}, u_{i+1|k}, u_{i+2|k}, u_{i+3|k}} \left\{ \sum_{i=k}^{k+3} \left( x_{i|k}^2 + 5u_{i|k}^2 + 25x_{i+4|k}^2 \right) \right\}
\] (9.8)

s.t. \( x_{i+1|k} = 0.95x_{i|k} + 0.5u_{i|k} \) \hspace{1cm} (9.9)

\( x_{k|k} = x_k \) \hspace{1cm} (9.10)

Equation (9.10) suggests that we will have full state information when providing feedback at each time step. In addition, assume the process model is perfect and there are no disturbances acting on the system (i.e., \( x_{k+1} = 0.95x_k + 0.5u_k \) for all \( k \)).

Recalling the batch LQOC solution approach used in Example 7.3, and noting the absence of inequality constraints, the quadratic program to be solved at each time step is exactly the same as the one solved in Example 7.3, except for a change in the initial condition. A comparison of the MATLAB code of Tables 7.1 and 9.1 indicate that the same \( M \) and \( A_0 \) matrices are used at each time step and the only update needed is to the \( b_0 \) vector.

The plots of Figure 9.3 illustrate the rolling horizon nature of MPC. The top two plots indicate the QP solution at time 0, and show a prediction horizon of 4. In the second two plots, the prediction horizon is again 4, but the subjected time points are shifted (or rolled) forward, when calculating the QP. A similar rolling forward is indicated in the bottom four plots. Figure 9.4 compares the predictions with the actual, assuming no disturbance or plant model mismatch. In spite of this assumption, there is clearly a discrepancy between the prediction and the realization. The reason for this discrepancy will be discussed in Section 9.2.2.

Table 9.1. MATLAB code used in calculations for Example 9.1.

clear
% Problem parameters
ad=0.95; bd=1/2; q=1; r=5; qf=25; N=4; x0=100;
% Define QP matrices
M=[r*eye(N) zeros(N); zeros(N) q*eye(N)]; M(2*N,2*N)=qf;
A0=[bd*eye(N) -eye(N)]; M(2*N,2*N)=qf;
for k=1:N-1 A0(k+1,N+k)=ad; end
9.1. The Receding-Horizon Framework

```matlab
b0=zeros(N,1); b0(1,1)=-ad*x0;
f=zeros(2*N,1); A1=[]; b1=[];

% Simulate MPC
NN=10; xxx=zeros(1,NN); xxx(1)=x0; uuu=zeros(1,NN-1);
for k=1:NN
    b0(1,1)=-ad*xxx(k);
    % Solve QP
    [z, phi, eflag]=quadprog(2*M,f,A1,b1,A0,b0);
    ui=z(1:N); xi=[xxx(k); z(N+1:2*N)];
    % Display Predicted Trajectories
    iix=linspace(k-1,k-1+N,N+1), figure(11), plot(iix,xi);
    iiu=linspace(k-1,k-1+N-1,N), figure(22), plot(iiu,ui);
    pause
    % Implement control action
    uuu(k)=ui(1); xxx(k+1)=ad*xxx(k)+bd*uuu(k);
end

% Display Actual Trajectory
kkx=linspace(0,NN,NN+1), figure(111), plot(kkx,xxx);
kku=linspace(0,NN-1,NN), figure(222), plot(kku,uuu);
```

Figure 9.3. Rolling horizon nature of MPC from Example 9.1
9.2 - Constrained LQOC

In Example 9.1, point-wise-in-time inequality constraints were not enforced. As such, the batch method of Section 7.1.1 could be applied directly. If inequality constraints are desired, then a simple modification of the batch method will be required. However, before doing so, let us reduce the notational burden. Notice that in Problem (9.6), the optimization to be solved at each time step \( k \) is of identical form. Thus, when considering the QP to be solved, we will assume \( k = 0 \) and then suppress the \( k \) notation, which will leave us with a QP in terms of only the predictive time index \( i \).

\[
\Phi(x_0) = \min_{u_0, u_1, \ldots, u_{N-1}} \left\{ \sum_{i=0}^{N-1} \left( x_{i+1}^T Q x_{i+1} + u_{i+1}^T R u_{i+1} \right) + x_N^T Q x_N \right\} 
\]

\[
\text{s.t. } x_{i+1} = A_d x_i + B_d u_i, \quad i = 0 \ldots N - 1 
\]

\[
z_i = D_d x_i + D_u u_i, \quad i = 0 \ldots N - 1 
\]

\[
z_{\min} \leq z_i \leq z_{\max}, \quad i = 0 \ldots N - 1 
\]

When implementing the MPC algorithm, the feedback aspect can then be incorporated by the assignment: \( u_k = u_{i|i=0} \). Then, in the next time step the new state estimate is used to as the new initial condition \( x_{i|i=0} = \hat{x}_{k+1} \).

\[
\Phi(x_0) = \min_{x_0, u_0, \ldots, u_{N-1}} \left\{ \sum_{i=0}^{N-1} \left( x_{i+1}^T Q x_{i+1} + u_{i+1}^T R u_{i+1} \right) + x_N^T Q x_N \right\} 
\]

\[
\text{s.t. } x_{i+1} = A_d x_i + B_d u_i, \quad i = 0 \ldots N - 1 
\]

\[
z_i = D_d x_i + D_u u_i, \quad i = 0 \ldots N - 1 
\]

\[
z_{\min} \leq z_i \leq z_{\max}, \quad i = 0 \ldots N - 1 
\]

\[
x_{i|i=0} = \hat{x}_{k+1} 
\]

Figure 9.4. Comparison of prediction and realization from Example 9.1

Figure 9.5. Illustration of MPC with the actual time index, \( k \), suppressed in the QP.
9.2.1 CLQOC with a Finite-horizon

In Section 7.1.1 it was stated that the general form of a quadratic program is:

$$\min_\theta \left\{ \sum \theta^T M\theta + \sum m^T \theta \right\} \quad \text{s.t.} \quad A_0 \theta = b_0 \quad \text{and} \quad A_1 \theta \leq b_1$$

(9.15)

And, the vector $\theta$ was designated as $\theta = [u_0^* \ u_1^* \ ... \ u_{N-1}^* \ x_1^* \ x_2^* \ ... \ x_N^*]^T$.

While the $z_i$ signal could be added to $\theta$, a more compact representation is to substitute (9.13) into (9.14) to find

$$z_{\text{min}} \leq D_x x_i + D_u u_i \leq z_{\text{max}} , \ i = 0 \ldots N - 1.$$  

Then, the matrices $A_1$ and $b_1$ are found to be:

$$A_1 = \begin{bmatrix} D_u & 0 & \cdots & 0 & 0 \\ 0 & D_u & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & D_u & 0 \\ 0 & 0 & \cdots & 0 & D_u \\ -D_u & 0 & \cdots & 0 & 0 \\ 0 & -D_u & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -D_u & 0 \\ 0 & 0 & \cdots & 0 & -D_u \end{bmatrix}$$

$$b_1 = \begin{bmatrix} z_{\text{max}} \\ z_{\text{max}} \\ \vdots \\ z_{\text{max}} \\ z_{\text{max}} \\ -z_{\text{min}} \\ -z_{\text{min}} \\ \vdots \\ -z_{\text{min}} \\ -z_{\text{min}} \end{bmatrix}$$

(9.16)

Figure 9.6. Comparison of LQOC with CLQOC for Example 9.2 ($-0.5 \leq u_i \leq 0.5$)

Figure 9.7. Comparison of LQOC with CLQOC for Example 9.2 ($-0.5 \leq x_i^{(2)} \leq 0.5$)
Example 9.2. Consider the mass-spring-damper system of Example 7.4:

\[
A = \begin{bmatrix} 0 & 1 \\ -3 & -0.2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

Application of the sample-and-hold method (with \(\Delta t = 0.2\) seconds) gives

\[
A_d = \begin{bmatrix} 0.941 & 0.192 \\ -0.576 & 0.903 \end{bmatrix} \quad B_d = \begin{bmatrix} 0.020 \\ 0.192 \end{bmatrix}
\]

If the LQOC parameters are selected as

\[
Q = I, \quad R = 0.5, \quad Q_f = 0, \quad N = 26 \quad \text{and} \quad x_0 = [1 \ 0]^T,
\]

then the trajectories resulting from the unconstrained LQOC are depicted in the plots of Figures 9.6 and 9.7, which use the \(M, A_0\) and \(b_0\) matrices of Example 7.4. If one would like to impose the constraints:

\[-0.5 \leq u_i \leq 0.5,\]

then \(D_x = \begin{bmatrix} 0 & 0 \end{bmatrix}^T, D_u = 1\) one could apply Equation (9.16) to determine the \(A_1\) and \(b_1\) to be used within Problem (9.15). The resulting \(A_1\) and \(b_1\) are given in Equation (9.17). The resulting Constrained LQOC (CLQOC) trajectory is depicted in Figure 9.6. If instead one would like to impose the constraints:

\[-0.5 \leq x_{(2)}^i \leq 0.5,\]

then \(D_x = \begin{bmatrix} 0 & 1 \end{bmatrix}^T, D_u = 0\) and the \(A_1\) matrix should be changed to Equation (9.18), but \(b_1\) would remain as in (9.17). The resulting CLQOC trajectory is depicted in Figure 9.7.

\[
A_1 = \begin{bmatrix} +1 & 0 & \ldots & 0 & 0 \\ 0 & +1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & +1 & 0 \\ 0 & 0 & \ldots & 0 & +1 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0.5 \\ 0.5 \\ \vdots \\ 0.5 \\ 0.5 \end{bmatrix}
\]

(9.17)

\[
A_1 = \begin{bmatrix} 0 & +1 & 0 & \ldots & 0 & 0 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0 \end{bmatrix}
\]

(9.18)
9.2.2 • CLQOC with an Infinite-horizon

Similar to the LQOC, significant conceptual insight will be gained if the horizon is stretched to infinity. The infinite-horizon CLQOC problem is stated as:

\[
\Phi(x_0) = \min_{u_0, u_1, \ldots} \left\{ \sum_{i=0}^{\infty} x_i^* Q x_i + u_i^* R u_i \right\}
\]

s.t. \( x_{i+1} = A_d x_i + B_d u_i, \ i = 0 \ldots \) \( \) \( z_i = D_x x_i + D_u u_i, \ i = 0 \ldots \) \( \) \( z_{\text{min}} \leq z_i \leq z_{\text{max}}, \ i = 0 \ldots \) \( \)

(9.19)

Table 9.2. Solution approaches for the LQOC and CLQOC problems.

<table>
<thead>
<tr>
<th></th>
<th>Finite-horizon</th>
<th>Infinite-horizon</th>
</tr>
</thead>
<tbody>
<tr>
<td>LQOC</td>
<td>Numeric or Analytic</td>
<td>Analytic only</td>
</tr>
<tr>
<td>CLQOC</td>
<td>Numeric only</td>
<td>Numeric and Analytic</td>
</tr>
</tbody>
</table>

Table 9.2 summarizes the solution methods required to solve the LQOC and CLQOC problems. In the case of CLQOC with an infinite-horizon, a combination of numeric and analytic procedures will be required. The basic idea is to convert the infinite-horizon problem to an equivalent finite-horizon problem. This begins by recalling the dynamic programming notion of a 'cost-to-go' function (see Section 7.1.2). If applied to Problem (9.19) one finds:

\[
\Phi_j(x_j) = \min_{u_j, u_{j+1}, \ldots} \left\{ \sum_{i=j}^{\infty} x_i^* Q x_i + u_i^* R u_i \right\}
\]

s.t. \( x_{i+1} = A_d x_i + B_d u_i, \ i = j \ldots \) \( z_i = D_x x_i + D_u u_i, \ i = j \ldots \) \( z_{\text{min}} \leq z_i \leq z_{\text{max}}, \ i = j \ldots \)

(9.23)

A quick inspection of Problem (9.23) reveals that the cost-to-go function, \( \phi_j(\cdot) \), is the same for all \( j \) (due to the fact that the horizon is infinite). Given this observation, we can further conclude that the cost-to-go function is equal to the value function of the original problem: \( \phi_j(x) = \phi_0(x) = \Phi(x) \). Of course, convergence of the cost-to-go functions to the value function for the infinite-time case should not be too surprising, since the same occurred in Section 7.2, but with respect to the Riccati equation. Using developments similar to those used to construct the Hamilton-Jacobi-Bellman (HJB) equation, the following generalization of the HJB is easily constructed:

\[
\Phi(x_0) = \min_{u_0, u_1, \ldots, u_{N-1}} \left\{ \sum_{i=0}^{N-1} (x_i^* Q x_i + u_i^* R u_i) + \Phi(x_N) \right\}
\]

s.t. \( x_{i+1} = A_d x_i + B_d u_i, \ i = 0 \ldots N - 1 \) \( z_i = D_x x_i + D_u u_i, \ i = 0 \ldots N - 1 \) \( z_{\text{min}} \leq z_i \leq z_{\text{max}}, \ i = 0 \ldots N - 1 \)

(9.27)
Problem (9.27) appears to have achieved our goal of converting the numerically intractable infinite-horizon LQOC problem to a finite-horizon form. However, before implementation, one will need the value function, \( \Phi(\cdot) \), so that \( \Phi(x_N) \) can be evaluated by the optimization routine. In general, the value function of Problem (9.19) is quite complicated and difficult to calculate, though not impossible (see Bemporad et al. [144]). Fortunately, for a particular region of the state space the value function can be determined trivially.

Consider the unconstrained infinite-horizon LQOC problem, essentially Problem (9.19), but without (9.22). In this case, the solution can be stated as

\[
x_i^* = (A_d - B_d L)^i x_0, \quad u_i^* = -L(A_d - B_d L)^i x_0
\]

and

\[
z_i^* = (D_x - D_u L)(A_d - B_d L)^i x_0
\]

where \( L \) is the optimal gain of the infinite-time LQOC problem, given by Equations (7.34). In addition, the value function of the unconstrained problem is \( \Phi(x_0) = x_0^* P x_0 \), where \( P \) is the solution to the ARE of Equation (7.35).

Let us now suppose that this unconstrained optimal trajectory satisfies the constraints of (9.22). That is, \( z_{min} \leq (D_x - D_u L)(A_d - B_d L)^i x_0 \leq z_{max} \) for all \( i \). In this case, the trajectory of (9.31) will not only be feasible for the constrained problem, it will be the solution. Thus, for that initial condition, \( x_0 \), the unconstrained value function, \( \Phi(x_0) = x_0^* P x_0 \), will be valid for the constrained problem. We can now turn the question around and ask for the set of all initial conditions such that application of the unconstrained infinite-time LQOC solution will result in no constraint violations. This set is denoted as the “maximal admissible” set (Gilbert and Tan, [145]):

\[
O_\infty = \{ x_0 \in \mathbb{R}^n, z_{min} \leq (D_x - D_u L)(A_d - B_d L)^i x_0 \leq z_{max} \quad \forall i \geq 0 \}
\]

Thus, if \( x \in O_\infty \), then \( \Phi(x) = x^* P x \). Using this insight, the following finite-horizon CLQOC problem can be stated:

\[
\min_{u_0, u_1, \ldots, u_{N-1}} \left\{ \sum_{i=0}^{N-1} (x_i^* Q x_i + u_i^* R u_i) + x_N^* P x_N \right\}
\]

\[\text{s.t.} \quad x_{i+1} = A_d x_i + B_d u_i, \quad i = 0 \ldots N - 1 \]

\[z_i = D_x x_i + D_u u_i, \quad i = 0 \ldots N - 1 \]

\[z_{min} \leq z_i \leq z_{max}, \quad i = 0 \ldots N - 1 \]

If the solution to Problem (9.33) is such that \( x_N^* \in O_\infty \), then one can safely conclude that the solution to Problem (9.19) has been determined, since the \( \Phi(x_N) \) of (9.23) can be correctly replaced by \( x_N^* P x_N \). If \( x_N^* \notin O_\infty \), then one will need to increase \( N \) until the solution is such that \( x_N^* \in O_\infty \).

An alternative approach to arriving at Problem (9.33) is via the finite-constrained infinite-time LQOC problem:

\[
\min_{u_0, u_1, \ldots} \left\{ \sum_{i=0}^{\infty} (x_i^* Q x_i + u_i^* R u_i) \right\}
\]

\[\text{s.t.} \quad x_{i+1} = A_d x_i + B_d u_i, \quad i = 0 \ldots \]

\[z_i = D_x x_i + D_u u_i, \quad i = 0 \ldots \]

\[z_{min} \leq z_i \leq z_{max}, \quad i = 0 \ldots N - 1 \]
In this case, the objective function can be restated as

\[
\min_{u_0, u_1, \ldots} \left\{ \sum_{i=0}^{\infty} x_i^* Q x_i + u_i^* R u_i \right\}
\]

\[= \min_{u_0, u_1, \ldots} \left\{ \sum_{i=0}^{N-1} (x_i^* Q x_i + u_i^* R u_i) + \sum_{i=N}^{\infty} (x_i^* Q x_i + u_i^* R u_i) \right\}
\]

\[= \min_{u_0, u_1, \ldots, u_{N-1}} \left\{ \sum_{i=0}^{N-1} (x_i^* Q x_i + u_i^* R u_i) + \min_{u_N, u_{N+1}, \ldots} \left\{ \sum_{i=N}^{\infty} (x_i^* Q x_i + u_i^* R u_i) \right\} \right\}
\]

\[= \min_{u_0, u_1, \ldots, u_{N-1}} \left\{ \sum_{i=0}^{N-1} (x_i^* Q x_i + u_i^* R u_i) + x_N^* P x_N \right\}
\] (9.41)

The last equality of (9.41) is due to the fact that the constraints of (9.40) are not enforced beyond time \(N - 1\), and thus the inner min is the unconstrained infinite-time LQOC with a value function equal to \(x_N^* P x_N\). Thus, (9.37) is actually equivalent to Problem (9.33). The advantage of the (9.37) form is that it explicitly shows the trajectory after the time \(N - 1\), which must be equal to:

\[
x_i^* = (A_d - B_d L)^{-N} x_N, \quad u_i^* = -L(A_d - B_d L)^{-N} x_N
\]

\[z_i^* = (D_x - D_u L)(A_d - B_d L)^{-N} x_N
\] (9.42)

where \(x_N^*\) is from the numerically calculated solution to (9.33) and \(i \geq N\). Thus, if \(z_i^*\) of (9.42) does not satisfy (9.22) for all \(i \geq N\), then we know that the solution to (9.37) is infeasible for (9.19) and thus cannot be a solution. On the other hand, if \(z_i^*\) of (9.31) does satisfy (9.22) for all \(i \geq N\) (i.e., \(x_N^* \in O_\infty\)), then we know the solution to (9.37) is the solution to (9.19). One way to think about (9.42) is that it is the implicit infinite tail associated with the solution to (9.33). Then, the test for optimality is if the implicit tail satisfies the constraints of (9.22), which is equivalent to testing if \(x_N^* \in O_\infty\). While one can explicitly calculate the \(O_\infty\) region, see Gilbert and Tan, [145], a more practical approach is to just implement the simulation of (9.42) a sufficient distance and check if (9.22) are satisfied or not.
9.2.3 Stability of CLQOC with an Receding Horizon

Let us now turn to the question of stability. As discussed in Chapter 4, the first criterion with regard to stability is that the open-loop plant must be stabilizable. However, in the constrained case, where the inequalities of (9.22) are enforced, the definition of stabilizability must be generalized. In particular, the set of initial conditions that can be stabilized while observing the constraints will be a subset of the stabilizable subspace. The following definition quantifies this set.

\[
X_{\text{max}} = \left\{ x_0 \in \mathbb{R}^n \mid \exists \{u_k\} \text{ s.t. } z_{\text{min}} \leq D_x x_k + D_u u_k \leq z_{\text{max}}, \right. \\
x_{k+1} = A_d x_k + B_d u_k, \forall k \geq 0, \text{ and } \lim_{k \to \infty} x_k = 0 \right\} \tag{9.43}
\]

Thus, a prerequisite to showing that a constrained system is stable is that the initial condition is in \(X_{\text{max}}\). Then, under the assumptions of no model mis-match (that is assuming the constraints of (9.20)-(9.21) are equal to the actual plant), it can be shown that the closed-loop system will be stable, if \(x_N^* \in O_{\infty}\) for each iteration of the receding horizon algorithm. Thus, using (9.33) within a receding horizon framework while ensuring a sufficiently large \(N\) (i.e., such that \(x_N^* \in O_{\infty}\)) will provide (in addition to optimality) a guarantee of stability. In Chmielewski & Manousiouthakis, [146] it is shown that if \(x_0 \in X_{\text{max}}\) there will always exits a finite \(N\) such that \(x_N^* \in O_{\infty}\). While one could just select an enormous \(N\), doing so will also result in an enormous (and likely unnecessary) computation burden. Thus, there is an interest in finding the smallest \(N\), denoted \(N^*\), such that \(x_N^* \in O_{\infty}\). Unfortunately, determination of \(N^*\) prior to solving (9.33) is a bit of a challenge, since \(N^*\) will depend on the initial condition \(x_0\). While estimates of \(N^*\) as a function of \(x_0\) exist, they are usually rather conservative, see for example Chmielewski & Manousiouthakis (1996). Thus, a more practical approach is to solve (9.33) and then test if \(x_N^* \in O_{\infty}\). If \(x_N^* \in O_{\infty}\), then repeatedly solve (9.33) with ever larger horizons until it is found that \(N \geq N^*\) or equivalently \(x_N^* \in O_{\infty}\). If re-optimization is undesired or impossible, then an alternative is to add \(x_N \in O_{\infty}\) as an additional constraint in Problem (9.33), which will force \(x_N^*\) into the set \(O_{\infty}\). While such an approach will ensure stability (by guaranteeing that for each iteration), it will be impossible to check if the \(N\) used is such that \(N \geq N^*\). The net result is a guarantee of stability, even if \(N < N^*\), but a loss...
of optimality, especially if \( N < N^* \). That is, if \( N < N^* \), then the closed-loop trajectory will deviate from the solution to (9.19). It should be emphasized that the additional constraint, \( x_N \in O_\infty \), may result in the optimization problem becoming infeasible. If this occurs, then one can just increase the size of \( N \) until the problem becomes feasible.

Example 9.3. Reconsider the discrete-time mass-spring-damper of Example 9.2. Recall that the LQOC parameters are selected as \( Q = I \) and \( R = 0.5 \) and the initial conditions is \( x_0 = [1 \ 0]^T \). Let us assume the constraints are \( -0.8 \leq x_i^{(2)} \leq 0.05 \). If the prediction horizon is set to \( N = 10 \), then the plots of Figure 9.10 illustrate the solution to problem (9.33), i.e., the predicted trajectory along with its implicit tail.

![Figure 9.10. CLQOC with Implicit Tail for Example 9.3 (N = 10)](image)

Notice that the tail is in violation of the constraint \( x_i^{(2)} \leq 0.05 \). Thus, we conclude that \( x_N^* \not\in O_\infty \) when \( N = 10 \). If the horizon is changed to \( N = 25 \), then Figure 9.11 indicates that \( x_N^* \in O_\infty \). However, since no constraints are active at the very end of the predicted trajectory, it is postulated that horizon could be made smaller. Figure 9.12 indicates that \( N^* = 20 \), since it is the smallest horizon such that \( x_N^* \in O_\infty \). Also, notice that the full trajectory (predicted plus tail) is the same for \( N = 25 \) and \( N = 20 \). In fact, the trajectories are the same for all \( N \geq N^* = 20 \). Let us now consider the case of adding the terminal constraint \( x_N \in O_\infty \) to Problem (9.33). This is done by adding constraints to the final state \( x_N \). Specifically, we add the constraints: 

\[
 z_i^{\min} \leq (D_x - D_u L)(A_d - B_d L)^i x_N \leq z_i^{\max} \quad i = 0 \ldots N_{\text{tail}} \quad \text{with } N_{\text{tail}} \text{ selected arbitrarily large, say } 20 .
\]

In this case, the resulting optimization problem will be infeasible for \( N < 6 \). That is, there does not exist inputs \( u_i \) such that \( x_N^* \not\in O_\infty \). Figure 9.13, illustrate the resulting trajectory for the case of \( N = 6 \).
Notice the rather erratic trajectory (compared to the true solution of Figures 9.11) that results from forcing the final value of the state into the terminal constraint $x_N^* \in O_\infty$. It should be noted that this is an extreme case and a small increase of the horizon (say $N = 8$) will greatly reduce the erratic nature of the trajectory.
It should be emphasized that all of the plots of Example 9.3 are open-loop trajectories. That is, Problem (9.33) is given no information about the process other than the initial condition, and as such implicitly assumes that no feedback information will be available. However, the receding horizon framework of Section 9.1 indicates that the MPC controller will use these open-loop optimal trajectories to construct a closed-loop policy, denoted the Open-Loop Optimal Feedback (OLOF) policy. It is emphasized that the OLOF policy is not optimal in a closed-loop sense. Specifically, if one assumes that a measurement of the state will be available to the controller at each time-step, then the optimal policy is likely to be different than the OLOF policy. This is especially true in the case of stochastic disturbances and/or model mismatch. In the case of stochastic disturbances, the closed-loop optimal policy must be calculated using dynamic programming. However, the presence of constraints significantly complicates the formulation and will require a numeric solution procedure that suffers from the “curse of dimensionality,” Bellman and Kalaba, [139] and Dryfus, [147]. For additional discussion on the topic of MPC with stochastic disturbances, please see Chmielewski & Manousiouthakis, [147]. This result is in distinct contrast with the stochastic LQOC (without inequality constraints), which possess certainty equivalence with its deterministic counterpart.

However, if there are no disturbances and there is no model mismatch (i.e., the actual process is $x_{k+1} = A_d x_k + B_d u_k$), then it is possible for the OLOF to be closed-loop optimal. The conditions are those indicated earlier in this subsection, namely: the weight in the final cost term is the solution the ARE (i.e., $Q_f = P$) and the horizon is sufficiently large (i.e., $x_N^* \in O_{\infty}$). If all of these conditions are satisfied, then the closed-loop trajectory resulting from application of the OLOF policy will be identical to the initial open-loop trajectory. This fact along with deterministic dynamic programming arguments indicates that the closed-loop trajectory resulting from the OLOF policy is indeed optimal. Using this optimal trajectory as a baseline, let us now investigate the impact of violating these assumptions. The following example illustrates that the OLOF policy performs remarkably well, even if the initial open-loop optimal trajectory is quite dissimilar to the closed-loop optimal trajectory.

**Example 9.4.** Reconsider the scenario of Example 9.3 and assume no disturbances and no model mismatch. In this case and assuming this initial condition, the closed-loop optimal policy is equal to the OLOF with a horizon $N \geq N^* = 20$. The resulting closed-loop optimal trajectory is identical to those depicted in Figures 9.11 and 9.12. In the following plots, this optimal trajectory will be denoted as the dashed line. Let us begin with a horizon $N = 3$ and a final cost weight equal to the ARE solution, $Q_f = P$. 

---

**Figure 9.13. CLQOC with Terminal Constraint $x_N \in O_{\infty}$ for Example 9.3 ($N = 6$)**
In this case, the plots of Figure 9.14 indicate that the resulting closed-loop trajectory is actually quite similar to the optimal closed-loop, though it approaches the origin a bit later. However, if the final cost weight is $Q_f = 0$, then there is a significant degradation in performance, with the trajectory approaching the origin significantly later. Table 9.3 illustrates the degradation quantitatively by calculating the objective function value the closed-loop trajectories. If the horizon is increased to $N = 6$, Figure 9.15 suggests that performance improves a bit for both cases. If the horizon is $N = 6$, then we can also test the impact of the terminal constraint $x_N \in O_\infty$, since the open-loop problem will be feasible, as indicated in Example 9.3. In this case, the closed-loop trajectory approaches the origin sooner than the closed-loop optimal. While this trajectory appears to be closer to the optimal than the other two, Table 9.3 indicates a quantitative performance that is actually poorer. This is likely due to the fairly erratic input used at the first time step — approximately equal to $-0.9$. Given the open-loop trajectory of Figure 9.13, this should not be a surprise. If the horizon is increased to $N = 9$, Figure 9.16 illustrates that the trajectories of all three begin to approach that of the closed-loop optimal, and Table 9.3 indicates a quantitative performance that is nearly identical for all four. The MATLAB code used to generate the plots of this example and Example 9.3 is given in Table 9.4.
Table 9.3. Solution approaches for the LQOC and CLQOC problems.

<table>
<thead>
<tr>
<th></th>
<th>$Q_f = P$</th>
<th>$Q_f = 0$</th>
<th>$x_N \in O_\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 3$</td>
<td>13.990</td>
<td>19.453</td>
<td>-</td>
</tr>
<tr>
<td>$N = 6$</td>
<td>13.863</td>
<td>15.614</td>
<td>14.907</td>
</tr>
<tr>
<td>$N = 9$</td>
<td>13.696</td>
<td>13.794</td>
<td>13.700</td>
</tr>
</tbody>
</table>

Table 9.4. MATLAB code used in calculations for Example 9.3 and 9.4.

```matlab
clear
Ad=[0.941 0.192; -0.576 0.903]; Bd=[0.020; 0.192];
Dx=[1 0;0 1;0 0; Du=[0;0;1];
zmin=[-1e6; -0.8; -1e6]; zmax=[1e6; 0.05; 1e6];
nx=2;nu=1,nw=1,nz=3; Q=[1 0;0 1]; R=0.5; S=[0; 0];
Find ARE solution
```
\[ P = \text{dare}(A_d, B_d, Q_d, R_d); \quad L = \text{inv}(R_d + B_d' * P * B_d) * B_d' * P * A_d; \quad Q_f = P; \]
\[
\text{mpc} = 1; \quad \text{if} (\text{mpc} == 1)
\]
\[ N = 25; \quad \% \text{horizon size} \]
\[ M_u = \text{zeros}(nu+N); M_x = \text{zeros}(nx+N); M_{ux} = \text{zeros}(nu+N, nx+N); \]
\[ \text{Mu} = \text{zeros}(nu+N); M_x = \text{zeros}(nx+N); M_{ux} = \text{zeros}(nu+N, nx+N); \]
\[ \text{for } i = 1:N - 1 \]
\[ M_u(1+(i-1)*nu:nu+(i-1)*nu, 1+(i-1)*nu:nu+(i-1)*nu) = R; \]
\[ M_x(1+(i-1)*nx:nx+(i-1)*nx, 1+(i-1)*nx:nx+(i-1)*nx) = Q; \]
\[ M_{ux}(1+i*nu:nu+i*nu, 1+(i-1)*nx:nx+(i-1)*nx) = S'; \]
\[ M_u(1+(N-1)*nu:nu+(N-1)*nu, 1+(N-1)*nu:nu+(N-1)*nu) = R; \]
\[ M_x(1+(N-1)*nx:nx+(N-1)*nx, 1+(N-1)*nx:nx+(N-1)*nx) = Q_f; \]
\[ M = [M_u, M_x; M_x', M_x]; \]
\[ A_0u = \text{zeros}(nx*N, nu*N); A_0x = \text{zeros}(nx*N, nx*N); \]
\[ A_0u(1:nx, 1:nu) = B_d; A_0x(1:nx, 1:nx) = -\text{eye}(nx); \]
\[ \text{for } i = 2:N \]
\[ A_0u(1+(i-1)*nx:nx+(i-1)*nx, 1+(i-1)*nu:nu+(i-1)*nu) = B_d; \]
\[ A_0x(1+(i-1)*nx:nx+(i-1)*nx, 1+(i-1)*nx:nx+(i-1)*nx) = -\text{eye}(nx); \]
\[ A_0x(1+(i-1)*nx:nx+(i-1)*nx, 1+(i-2)*nx:nx+(i-2)*nx) = A_d; \]
\[ A_0 = [A_0u, A_0x; -A_0u, -A_0x; \text{zeros}(nz*N, N*(nx+nu)-nx)]; \]
\[ b_0 = \text{zeros}(nx*N, 1); \]
\[ A_{lu} = \text{zeros}(nz*N, nu*N); A_{lx} = \text{zeros}(nz*N, nx*N); \]
\[ b_{lmax} = \text{zeros}(nz*N, 1); b_{lmin} = \text{zeros}(nz*N, 1); \]
\[ A_0(l_{1:nx, 1:nu}) = D_u; \]
\[ b_{lmax}(1:nz) = z_{max} - D_x * x_0; \]
\[ b_{lmin}(1:nz) = -z_{min} + D_x * x_0; \]
\[ \text{for } i = 2:N \]
\[ A_{lu}(1+(i-1)*nz:nz+(i-1)*nz, 1+(i-1)*nu:nu+(i-1)*nu) = D_u; \]
\[ A_{lx}(1+(i-1)*nz:nz+(i-1)*nz, 1+(i-1)*nx:nx+(i-1)*nx) = D_x; \]
\[ b_{lmax}(1+(i-1)*nz:nz+(i-1)*nz, 1) = z_{max}; \]
\[ b_{lmin}(1+(i-1)*nz:nz+(i-1)*nz, 1) = -z_{min}; \]
\[ A_l = [A_{lu}, A_{lx}; -A_{lu}, -A_{lx}; \text{zeros}(nz*N, N*(nx+nu)-nx)] A_l + \ldots \]
\[ \text{zeros}(nz*N_{tail}, N*(nx+nu)-nx) A_l l; \ldots \]
\[ \text{zeros}(nz*N_{tail}, N*(nx+nu)-nx) -A_l l; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{Simulate Closed-loop System} \]
\[ dt = 0.2; \quad \text{NN} = 10/dt; \quad \text{tt} = \text{zeros}(1, NN+1); \quad \text{xxx} = \text{zeros}(nx, NN+1); \]
\[ \text{for } kk = 1:NN \]
\[ \text{tt}(kk) = (kk-1) \times dt; \]
\[ \text{if } \text{mpc} == 1 \]
\[ x_0 = \text{xxx}(:, 1:kk); \]
\[ m(1:nu) = 2 * x_0' * S; b_0(1:nx) = -A_d * x_0; \]
\[ b_{lmax}(1:nz) = z_{max} - D_x * x_0; b_{lmin}(1:nz) = -z_{min} + D_x * x_0; \]
\[ \text{if } \text{constraints on initial condition are needed} \]
\[ b_{lmax}(1:nz) = z_{max}; b_{lmin}(1:nz) = -z_{min}; \]
\[ \theta = \text{quadprog}(2 * M, A_1, b_1, A_0, b_0); \]
\[ \text{if } (0) \]
\[ x_{1i}(:, 1) = x_0; \quad \text{tt} = \text{linspace}(0, N \times dt, N + 1); \]
\[ \text{for } i = 1:N \]
9.3. MPC with Stochastic Disturbances

While the previous section tells us that the OLOF policy of MPC is not closed-loop optimal, especially with regard to the constrained stochastic problem, it is possible (and certainly of interest) to apply MPC to stochastic systems. In particular, the severe computational burden associated with the closed-loop optimal policy (i.e., the curse of dimensionality) suggests that the OLOF policy may be a reasonable substitute.

Example 9.5. Consider the following continuous-time model of a mass-spring-damper:

\[
A = \begin{bmatrix} 0 & 1 \\ -3 & -0.2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad S_w = 0.5
\]

Application of the sample-and-hold method (with \( \Delta t = 0.2 \) seconds) gives

\[
A_d = \begin{bmatrix} 0.941 & 0.192 \\ -0.576 & 0.903 \end{bmatrix}, \quad B_d = G_d = \begin{bmatrix} 0.020 \\ 0.192 \end{bmatrix}, \quad \Sigma_w = 2.5
\]
In addition, let $Q = I$, $R = 0.5$, $Q_f = P$ and $x_0 = [0 \ 0]^T$. In this example we will test the MPC policy using the constraints $-0.4 \leq x_i^{(1)} \leq 0.4$ and a horizon $N = 10$. The plots of Figures 9.17 and 9.18 show that the controller is making a good effort to keep the mass position within the $-0.4 \leq x_i^{(1)} \leq 0.4$ constraint and does so by significantly

![Figure 9.17. Stochastic MPC trajectories for Example 9.5. The bottom plot is unconstrained (i.e., the result of the LQOC policy)](image-url)
increasing the input force when needed to do so. This can be seen by comparing the manipulated variable plot with the unconstrained case of the bottom plot. Notice that when the unconstrained case violates the constraints, the MPC will use large values of the manipulated variable to hold the mass position within the constraints. The scatter plots of Figure 9.18 illustrate that constraints are not observed perfectly. This is due to the fact that the new stochastic disturbance, $w_k$, occurs at the same time as the manipulated variable ($x_{k+1} = A_d x_k + B_d u_k + G_d w_k$). Figures 9.19 and 9.20 show the one-step predicted state trajectory, $\bar{x}_{k+1} = A_d x_k + B_d u_k$ (i.e., the state values the MPC expected to see). Using this perspective, the constraints appear to be observed perfectly, though we know that they are not since the new stochastic disturbance, $w_k$, may push the state outside of the constraint after the manipulated variable, $u_k$, has made the effort to hold it inside the constraints.

9.3.1 MPC with Soft Constraints

The one-step prediction trajectories of Example 9.5 illustrate that, as far as the MPC is concerned, the manipulated variable can always get the state to satisfy the constraints $-0.4 \leq x_i^{(1)} \leq 0.4$. This is because no matter how far the mass position is from the constraint, the controller is always able to select a large enough input force to drive the position into the constraint set. However, if the input force was also constrained (say,
−3 ≤ \( u^{(1)}_i \) ≤ 3, then there will be cases in which a return to the set −0.4 ≤ \( x^{(1)}_i \) ≤ 0.4 will be impossible while observing the −3 ≤ \( u^{(1)}_i \) ≤ 3 constraint (i.e., initial condition is infeasible or \( x_0 \notin X_{\text{max}} \)). While the condition \( x_0 \notin X_{\text{max}} \) suggests that the problem is illposed (and it is), the controller will need to give some sort of command to the process (the command ‘infeasible solution’ is not an option). Given that \( x_0 \notin X_{\text{max}} \), the only option is to give up on enforcing one or more of the constraints. The question is how do we communicate to the MPC that constraint violations are allowed, but only if absolutely required (i.e., \( x_0 \notin X_{\text{max}} \))? In addition, how do we tell the MPC which constraints can be relaxed and if there are multiple options which has the higher priority? To achieve these goals, one could employ a soft-constrained formulation (Zheng and Morari, [148]). The basic procedure is to introduce a vector of slack variables, \( \theta \). Then, the original ‘hard’ constraints of (9.36) are replaced by the following soft constraints

\[
\begin{align*}
  z_{\text{min}} - \theta & \leq z_i \leq z_{\text{max}} + \theta, \quad i = 0 \ldots N - 1 \\
  \theta & \geq 0
\end{align*}
\] (9.44)

Then, a weighted sum of the slack variables, \( c^\ast \theta \), is added to the MPC objective function. The idea is that if the elements of \( c \) are set to sufficiently large values, then the optimization will select \( \theta \approx 0 \), except for the cases in which the hard constraints are infeasible. That is, the constraints of controller are relaxed, \( \theta > 0 \), only if the optimization has no other options. In sum, the soft constrained MPC formulation is as follows:

\[
\Phi(x_0) = \min_{u_0, u_1, \ldots, u_{N-1}} \left\{ \sum_{i=0}^{N-1} \left( x_i^r Q x_i + u_i^r R u_i \right) + x_N^r P x_N + c^\ast \theta \right\}
\] (9.45)

s.t. \[
\begin{align*}
  x_{i+1} &= A_d x_i + B_d u_i, \quad i = 0 \ldots N - 1 \\
  z_i &= D_x x_i + D_u u_i, \quad i = 0 \ldots N - 1 \\
  z_{\text{min}} - \theta & \leq z_i \leq z_{\text{max}} + \theta, \quad i = 0 \ldots N - 1 \\
  \theta & \geq 0
\end{align*}
\] (9.46) (9.47) (9.48) (9.49)

**Example 9.6.** Reconsider the scenario of Example 9.5, but add the constraints −3 ≤ \( u_i \) ≤ 3. In this case, there will be many cases in which Problem (9.33) will be infeasible. However, application of the soft-constrained MPC (Problem 9.45), with \( c = 10^{12} \begin{bmatrix} 1 & 1 & 10 \end{bmatrix} \), results in the plots of Figures 9.21 and 9.22. Notice that the constraint violations occur on the mass position variable and not the input force. This is because the \( c \) weight on the input...
force is an order of magnitude larger than the others, which is communicating that this constraint is more important than the others. If $c$ is changed to $10^{12}[10\ 1 \ 1]$, then it would allow the violations on the input force and not the mass position. In this case, the plots would be similar to those of Figure 9.20. Note that Figures 9.21 and 9.22 are both plotting the one-step prediction, so that activation of the slack variables is not confused with a violation resulting from the new value of the disturbance.

![Figure 9.21](image1.png)  
**Figure 9.21.** Stochastic MPC prediction trajectories with soft constraints for Example 9.6.

![Figure 9.22](image2.png)  
**Figure 9.22.** Stochastic MPC prediction scatter plots with soft constraints for Example 9.6.
Table 9.5. MATLAB code used in calculations for Example 9.6 and 9.7.

```
clear
Ad=[0.941 0.192; -0.576 0.903]; Bd=[.020; 0.192]; Gd=[.020; 0.192]; Sw=0.5;
Dx=[1 0;0 1]; Du=[0;0]; zmin=[-0.4; -1e6; -3]; zmax=[0.4; 1e6; 3];
nx=2;nu=1,nw=1,nz=3; Q=[1 0;0 1]; R=0.5; S=[0;0];
P=dare(Ad,Bd,Q,R); L=inv(R+Bd'*P*Bd)*Bd'*P*Ad; Qf=P;
mpc=1; % set to zero for unconstrained LQOC
delay=0; % set to zero no delay
if (mpc == 1)
    N=10; % horizon size
    Mu=zeros(nu*N); Mx=zeros(nx*N); Mux=zeros(nu*N,nx*N);
    for i=1:N-1
        Mu(1+(i-1)*nu:nu+(i-1)*nu,1+(i-1)*nu:nu+(i-1)*nu)=R;
        Mx(1+(i-1)*nx:nx+(i-1)*nx,1+(i-1)*nx:nx+(i-1)*nx)=Q;
        Mux(1+i*nu:nu+i*nu,1+(i-1)*nx:nx+(i-1)*nx)=S';
    end
    Mu(1+(N-1)*nu:nu+(N-1)*nu,1+(N-1)*nu:nu+(N-1)*nu)=R;
    Mx(1+(N-1)*nx:nx+(N-1)*nx,1+(N-1)*nx:nx+(N-1)*nx)=Q;
    M=[Mu Mux; Mux' Mx];
    m=zeros((nu+nx)*N,1);
    A0u=zeros(nx*N,nu*N); A0x=zeros(nx*N,nx*N);
    A0u(1:nx,1:nu)=Bd; A0x(1:nx,1:nx)=-eye(nx);
    for i=2:N
        A0u(1+(i-1)*nx:nx+(i-1)*nx,1+(i-1)*nu:nu+(i-1)*nu)=Bd;
        A0x(1+(i-1)*nx:nx+(i-1)*nx,1+(i-1)*nx:nx+(i-1)*nx)=-eye(nx);
        A0x(1+(i-1)*nx:nx+(i-1)*nx,1+(i-2)*nx:nx+(i-2)*nx)=Ad;
    end
    A0=[A0u A0x];
    b0=zeros(nx*N,1);
    A0=[A0 zeros(nx*N,nz)]; % Add Slack variables
Alu=zeros(nx*N,nu*N); Alx=zeros(nx*N,nx*N); Alu(1:nx,1:nu)=Du;
blmax=zeros(nx*N,1); blmin=zeros(nx*N,1);
AIslack=zeros(nx*N,nz); % Add Slack variables
AIslack(1:nx,1:nz)=-eye(nz); % Add Slack variables
    for i=2:N
        Alu(1+(i-1)*nx:nx+(i-1)*nx,1+(i-1)*nu:nu+(i-1)*nu)=Du;
        Alx(1+(i-1)*nx:nx+(i-1)*nx,1+(i-2)*nx:nx+(i-2)*nx)=Dx;
        AIslack(1+(i-1)*nx:nx+(i-1)*nx,1:nz)=-eye(nz);
        blmax(1+(i-1)*nx:nx+(i-1)*nx,1)=zmax;
        blmin(1+(i-1)*nx:nx+(i-1)*nx,1)=zmin;
    end
    A1=[Alu Alx;-Alu -Alx]; % Add Slack variables
    A1=[A1 [AIslack; AIslack];zeros(nx*N*(nu+nx)) -eye(nz)];
end

% Simulate Closed-loop System
T=0.2, NN=T/dc; Sigw=Sw/dc; randn('state',2^6-1);
tt=zeros(1,NN+1); xxx=zeros(nx,NN+1); xbar=zeros(nx,NN+1);
xxxbar=zeros(nx,NN+1); uu=zeros(nu,NN); OPTIONS = optimset('MaxIter',500);
for kk=1:NN
    tt(kk)=(kk-1)*dt; kk
if mpc == 1
    if (delay == 0) x0=xxx(:,kk), else x0=xxxbar(:,kk); end
    b0(1:nx)=Ad*x0; m(1:nu)=2*x0'*S;
    blmax(1:nz)=zmax-Dx*x0; blmin(1:nz)=-zmin+Dx*x0;
    % Use above if constraints on initial condition are needed
    blmax(1:nz)=zmax; blmin(1:nz)=-zmin; bl=[blmax;blmin];
    bl=[bl; zeros(nx,1)]; % Add Slack variables
    theta=quadprog(2*N,m,A1,b1,A0,b0,[],[],[],OPTIONS);
```

uuu(:,kk)=theta(1,nu); 
else 
    uuu(:,kk)=-L*xxx(:,kk); 
end 
ww=sqrt(Sigw)*randn; 
xxx(:,kk+1)=Ad*xxx(:,kk)+Bd*uuu(:,kk)+Gd*ww; 
xbar(:,kk)=Ad*xxx(:,kk)+Bd*uuu(:,kk); 
xxxbar(:,kk+1)=Ad*xxx(:,kk)+Bd*uuu(:,kk); 
end 
tt(NN+1)=(NN)*dt; uuu(:,NN+1)=uuu(:,NN); 
figure(2); hold on; plot(tt,xbar(1,:),'k--*',... 
    [0 200], [zmin(1) zmin(1)],'--k',... 
    [0 200], [zmax(1) zmax(1)],'--k') 
figure(3); hold on; plot(tt,uuu(1,:),',k--*',... 
    [0 200], [zmin(3) zmin(3)],'--k',... 
    [0 200], [zmax(3) zmax(3)],'--k') 
figure(4); hold on; plot(uuu(1,:),xbar(1,:),',k--*',... 
    [-zmin(3) zmin(3)], [zmin(1) zmin(1)],'--k',... 
    [-zmin(3) zmin(3)], [-zmin(1) -zmin(1)],'--k',... 
    [zmin(3) zmin(3)], [-zmin(1) zmin(1)],'--k',... 
    [-zmin(3) -zmin(3)], [zmin(1) zmin(1)],'--k') 
figure(5); hold on; plot(xbar(2,:),xbar(1,:),',k--*',... 
    [-2 2], [zmin(1) zmin(1)],'--k',... 
    [-2 2], [-zmin(1) -zmin(1)],'--k--') 
figure(14); hold on; plot(uuu(1,:),xxx(1,:),',k--*',... 
    [-zmin(3) zmin(3)], [zmin(1) zmin(1)],'--k',... 
    [-zmin(3) zmin(3)], [-zmin(1) -zmin(1)],'--k',... 
    [zmin(3) zmin(3)], [-zmin(1) zmin(1)],'--k',... 
    [-zmin(3) -zmin(3)], [zmin(1) -zmin(1)],'--k--') 

Figure 9.23. Stochastic MPC prediction scatter plots with soft constraints for Example 9.7. 
Top plots $R = 50$, bottom plots $R = 0.005$. 
Example 9.7. Continuing Example 9.6, let us investigate the impact of the objective function weights, $Q$ and $R$. Begin by changing $R$ from 0.5 to 50. The top plots of Figure 9.23 illustrate that this re-tuning will result in many more infeasible situations. This is observed from the significant number of points outside of the $-0.4 \leq x_i^{(1)} \leq 0.4$ constraint, which is a result of the slack variable being activated. If $R$ is changed to 0.005, then the bottom plots of Figure 9.23 indicate that there will be almost no infeasible cases, approximately 3 out of the 10,000 time steps. The reason for this dramatic change in performance can be found by looking at the unconstrained (LQOC) operating region (i.e., the one resulting purely from the objective function weights). The top left plot of Figure 9.24 shows this region for the $R = 50$ case. Note that all of the plots of Figure 9.24 are of the actual state variables and not of the one-step predictions. In this case, the
manipulated variable takes almost no action (due to the large weigh on $R$). So, the state trajectories are essentially those of the open loop. Turning to the soft constrained case (the top right plot), the MPC does take action, but only when it is required to observe (or attempt to observe) the $-0.4 \leq x_{j}^{(1)} \leq 0.4$ constraint. The problem is that in many case this action comes too late, and the slack variables must be activated. In the middle plots ($R = 0.5$), the situation is better because the objective function is telling the controller to act even if the state is not near a constraint. In the bottom plots ($R = 0.005$), we see that the LQOC is quite aggressive. This early action allows the MPC to easily observe the $-0.4 \leq x_{j}^{(1)} \leq 0.4$ constraints.

Example 9.7 shows that the MPC objective function weights can have a dramatic impact on performance. It also illustrates that the operating region of the stochastic LQOC will give insight into the performance one can expect. Specifically, if the LQOC operating region is aligned with the constraint set, then one should expect good MPC performance in the sense that the closed-loop trajectories should be sensible and rational while keeping the number of infeasible situations to a minimum. In contrast, if the operating region is not aligned with the constraints, then the MPC objective function and its constraints will each be telling the controller to do different things. This internal struggle will make the MPC ill prepared for constraint enforcement, which will likely result in erratic behavior. One way to think about the two situations is that there is a natural behavior induced by the objective function weights. If the constraints are not aligned with this natural tendency, then the policy must be twisted and reshaped to conform to the constraints.

Thus, the question is: How does one select objective function weights so that the unconstrained operating region will be in harmony with the constraints? In Example 9.7, the simplicity of system allowed for a rather intuitive analysis: make the controller more or less aggressive by changing the scalar $R$. However, in the more complicated multi-input multi-output case, it is unlikely one will be able to rely on such intuition. As such, much of Part III of the text will be devoted to answering this question and providing a scheme for determining suitable weights.

In contrast to Example 9.7, it will be unreasonable to rely on simulations to calculate the unconstrained operating region. Fortunately, the results of previous chapters have provided us with the tools necessary to calculate this operating region analytically. Recall the covariance analysis of Chapters 5 and 8. As an example of these analytic calculations, the grey ellipse of the bottom right plot of Figure 9.24 represents the EDOR of that case. The method of calculating such EDORs is described in the text just after Example 8.3.

Of course, being able to calculate the EDOR analytically is only half of the solution. Given this tool, one will still need to select the $Q$ and $R$ matrices and then test manually if the EDOR is aligned with the constraint set. If it is not, then a trial and error exercise will be required to find suitable objective function weights. Clearly, one would prefer to have an automated procedure. One that is given the constraint set and automatically generates objective function weights that will result in a closed-loop EDOR that is aligned with the constraints. Details of such a procedure will be provided in Chapter 11. In Chapter 13, a similar procedure will be provided, but with the added feature of incorporating economic tuning into the procedure.

By the same logic, one should not expect reasonable performance from the MPC, regardless of objective function weights, if there is no EDOR capable of fitting into the constraint set. Under such a scenario, it would be ill advised to implement the MPC, though with the soft-constraint formulation it is certainly possible. A better option is
to re-evaluate the constraints imposed and possibly the design of the process. One could think of this as a constrained version of the stabilizability question. Thus, maybe a more important utility of an automated tuning procedure is to give a definitive answer as to the existence (or lack thereof) of objective function weights such that the EDOR fits within the constraint set.

### 9.3.2 MPC with Computational Delay and PSI

Another important MPC issue is the time required to solve Problem (9.33) or (9.45). The above rolling horizon perspective of MPC suggests that the computational time allowed to solve these optimization problems must be less than the controller sample-time, $\Delta t$. The fact that the QP cannot be solved instantaneously suggests that a modification of the MPC algorithm will be required. The issue is that at time $k$ one will receive a measurement of $x_k$. Then, based on this state the optimization will determine $u_k = u_k|k$. However, if the optimization takes one sample-time to determine $u_k$ (i.e., solve the QP), then it will be determined at time $k + 1$, one sample period later than required. Thus, to determine $u_k$ such that it is available at time $k$, one will need to start the optimization at time $k - 1$. However, at time $k - 1$, one will not have a measurement of $x_k$. At best, one can expect to have a measurement of $x_{k-1}$.

The logical solution is to make a prediction of $x_k$ based on $x_{k-1}$,

$$
\hat{x}_k = A_d x_{k-1} + B_d u_{k-1}
$$

(9.50)

where $x_{k-1}$ is measured and $u_{k-1} = u_{k-1}|k-1$ is the manipulation calculated at the previous time-step (and just implemented). In sum, the MPC algorithm assuming computational delay is stated as:

$$
\Phi(\hat{x}_k) = \min_{u_k, u_{k+1}, \ldots, u_{k+N-1}} \left\{ \sum_{i=k}^{k+N-1} \left( x_i^s Q x_i^k + \sum_{i=k}^{k+N-1} R u_i^k \right) + x_{k+N}^* P x_{k+N}^k \right\} 
$$

(9.51)

subject to:

- $x_{k+1} = A_d x_{k+1} + B_d u_{i=k}, \quad i = k \ldots k + N - 1$
- $z_{i=k} = D_d x_{i=k} + N u_{i=k}, \quad i = k \ldots k + N - 1$
- $z_{i=k} \leq z_{i=k} \leq z_{i=k}^* \leq z_{i=k}^*, \quad i = k \ldots k + N - 1$
- $x_{k+1} = \hat{x}_k$

The implicit assumption is that this optimization will be started at time $k - 1$ and its solution will be determined before time $k$, so that $u_k|k$ will be available for a timely implementation of $u_k$.

If the MPC is being implemented with a partial state information feedback structure, then Problem (9.33) would again be implemented, but with the initial condition, $x_k$, replaced by the optimal estimate, $\hat{x}_k$. If the system also has computational delay, then Problem (9.51) would be implemented, but with the deterministic prediction, $\bar{x}_k$, replaced by the one-step prediction from the optimal estimator, $\hat{x}^f$. Again, the intention is that Problem (9.51) is started at time $k - 1$, which should not be a problem since $\hat{x}^f$ requires the measurement obtained at time $k - 1, y_{k-1}$, but not the measurement from time $k$. Of course, each of these changes to the feedback structure will result in a degradation of performance as compared to the full state information case without delay.

**Example 9.8.** Reconsider the last simulation ($R = 0.005$) of Example 9.7. However, implement the MPC under the assumption of computational delay. In this case, the scatter
plots of Figure 9.25 will result. As expected greater variability is found as compared to the no delay case of Figure 9.24.

Figure 9.25. Stochastic MPC scatter plots with soft constraints and computational delay for Example 9.8.

9.4 - Chapter Summary

The chapter begins by describing the core concept of MPC — the notion of a receding horizon implementation, also known as open-loop optimal feedback. The idea being that after one solves a multi-stage optimal control problem, the only part of that optimal trajectory implemented is the initial step of the manipulated variable. More importantly, the state (or state estimate) of the actual process is used as the initial condition of the multi-stage optimal control problem solved at each time step. This mechanism of incorporating feedback into an algorithm based on open-loop optimization is the key element behind the unexpected success of MPC.

The chapter then narrows its scope to the constrained LQOC framework — essentially LQOC, but with a requirement to enforce point-wise-in-time inequality constraints. In this case, it was shown that the batch method of Chapter 7 could be extended to find the numeric solution to the CLQOC, under the assumption of a finite horizon. To address the infinite-horizon case, which is required to prove MPC stability, it was shown that the numeric solution required the batch method to be combined with the analytic solution to the infinite-time LQOC. The net result was the construction of a finite-horizon CLQOC problem, but with an appropriately selected final cost term along with a sufficiently large horizon size.

The chapter then turned to a discussion of MPC in the context of stochastic disturbances. Under the influence of such disturbances, it is likely that one will eventually find an initial condition such that the CLQOC problem will be infeasible. Since the actual process cannot proceed without a command to the manipulated variables, the only option in this case is to relax some of the inequality constraints. To this end, the soft-constrained version of MPC was shown to provide the required relaxations, but only if absolutely necessary and was able to designate priority of the constraints by appropriately selecting the soft-constraint weights. While the net result is the possibility of the actual process violating the inequality constraints, Example 9.7 shows that the likelihood of these violations will be strongly influenced by the MPC tuning parameters (i.e., the
LQOC objective function weights $Q$ and $R$). The conclusion being that an alignment between the LQOC EDOR and the constraint set will serve to reduce the likelihood of constraint violations, possible to the point of virtually zero occurrences. This notion of tuning the MPC objective function to be aligned with the constraint set, sets the stage for subsequent chapters and will be a guiding principle in the development of the economic based controllers of Part III.

The notion of MPC was first introduced in the seminal work of Culter and Ramaker, [149]. Based on that work, a large number of implementations occurred within the chemical industry, see Froisy, [150], and Qin and Badgwell, [151], for details. Original developments of the infinite-horizon problem (along with stability proofs) can be found in Sznaier and Damborg [152], Rawlings and Muske [153], Chmielewski and Manousiouthakis [146] and Scokaert and Rawlings [154]. While the literature on MPC is vast (to the point of being overwhelming) a number of high quality tutorial papers can be a great help those new to the subject (see for example, Garcia et al., [155], Rawlings [156], Rawlings [157], or Mayne, et al. [158]). For additional detail on the subject, it is recommended to consult the following textbooks: Camacho and Bordons [160] and Rawlings and Mayne [159].

**Exercises**

9.1. Reproduce the plots of Example 9.2.
9.2. Reproduce the plots of Example 9.4.
9.3. Reproduce the plots of Example 9.5.
9.4. Reproduce the plots of Example 9.6.
9.5. Consider the following continuous-time model of a mass-spring damper:

$$
\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -0.25 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u
$$

where the first state is mass position, the second is velocity and the manipulated variable is an applied force. Discretize the system using the sample and hold method (with $\Delta t = 0.25$). Using the MATLAB function ‘quadprog’, determine the solution to the deterministic LQOC problem (with $N = 25$ and $Q = 0$) for the following three cases: a) $Q = I$ and $R = 0.1$, b) $Q = I$ and $R = 1$ and c) $Q = I$ and $R = 10$.

9.6. We desire to apply a predictive controller to the following system

$$
x_{k+1} = A_d x_k + B_d u_k \\
y_k = C x_k
$$

so as to track a predefined trajectory, $y^{(sp)}(s, p)$. To do so we will utilize the velocity form of the predictive controller, which will solve the following optimization
problem at each time step.

\[ \Phi(x_k) = \min_{u_{i|k}, u_{i+1|k}, \ldots, u_{i+N-i|k}} \left\{ \sum_{i=k}^{k+N-1} \left( y_{i|k} - y_{i}^{(sp)} \right)^* Q \left( y_{i|k} - y_{i}^{(sp)} \right) + r_{i|k}^* R r_{i|k} \right\} \]

s.t. \[
\begin{align*}
x_{i+1|k} &= A_d x_{i|k} + B_d u_{i|k}, \quad i = k \ldots k + N - 1 \\
y_{i|k} &= C x_{i|k}, \quad i = k \ldots k + N \\
r_{i|k} &= u_{i|k} - u_{i-1|k}, \quad i = k \ldots k + N \\
x_{min} \leq x_{i|k} \leq x_{max}, \quad i = k \ldots k + N \\
u_{min} \leq u_{i|k} \leq u_{max}, \quad i = k \ldots k + N \\
r_{min} \leq r_{i|k} \leq r_{max}, \quad i = k \ldots k + N \\
x_{k|k} &= x_k
\end{align*}
\]

where the initial condition, \( x_k \), the desired trajectory, \( y_k^{(sp)} \), and the input at the previous time, \( u_{k-1} \), are all known.

(i) Please describe the characteristics of this velocity form of the predictive controller. How is it different from the predictive controller discussed in Section 9.2? Why is it particularly useful for the tracking problem? (Hint: Consider the case where \( y_k^{(sp)} \) does not go to zero in the steady-state and think about where \( u_k \) will end up at steady-state.)

(ii) Determine the matrices one should provide to the Matlab function ‘quadprog’ to solve the velocity form of the predictive controller. Recall that \( x_k, y_k^{(sp)} \), and \( u_{k-1} \) are all known. If it is helpful to you, you may assume the value of \( N \) is 3.

9.7. Consider the following constrained infinite horizon optimal control problem

\[ \Phi(x_0) = \min_{u_0, u_1} \left\{ \sum_{k=0}^{\infty} x_k^2 + u_k^2 \right\} \]

s.t. \[
\begin{align*}
x_{k+1} &= 2x_k + u_k \\
u_k &\leq 1
\end{align*}
\]

Determine the sets \( X_{max} \) and \( O_{\infty} \).
Part III

Economic Based Controller Design
Chapter 10

Matrix Inequalities

The notion of a matrix inequality has been around since the time of Lyapunov and has implications far beyond the simple linear system notions discussed here. In recent decades a number of computational advances have been made in the area of Linear Matrix Inequalities (LMIs), a subset of the more general field of Semi-Definite Programming (SDP). In fact, some would say that LMI methods have revolutionized control system design. While this may be an overstatement, one can certainly assert that SDP methods have had a strong impact on a variety of areas within the wider field optimization. The material of this chapter serves as an introduction to the concepts required to understand the most basic applications of LMIs. The results of this chapter will provide the theoretic foundation to the economic based control system design methods of subsequent chapters. In contrast to previous chapters, formal proofs are provided for most of the results as these are perceived to be integral to the reader’s understanding of the material.

10.1 Preliminaries

We begin with a restatement of the definition of an inequality in the context of a matrix.

**Definition 3.29.** A linear transformation $P : X \to X$ is

(i) **positive definite** ($P > 0$), if $P^* = P$ and $[Px, x] > 0$ for all nonzero $x \in X$.

(ii) **positive semi-definite** ($P \geq 0$), if $P^* = P$ and $[Px, x] \geq 0$ for all nonzero $x \in X$.

(iii) **negative definite** ($P < 0$) if $-P > 0$.

(iv) **negative semi-definite** ($P \leq 0$) if $-P \geq 0$.

It should be emphasized that a positive definite matrix does not imply or require that all the elements of the matrix are positive (see Example 3.21). Theorem 3.16 indicates that the necessary and sufficient conditions for a matrix to be positive definite are be self-adjoint and have all positive eigenvalues.

The following **Schur Complement Theorem** will be an essential utility in the derivation of many of the subsequent results.
Theorem 10.1. Consider a matrix $M$, partitioned as follows, and assume $C^{-1}$ exists.

$$M = \begin{bmatrix} A & B \\ B^* & C \end{bmatrix}$$ (10.1)

Then, $M > 0$ if and only if $C > 0$ and $A - BC^{-1}B^*$. Furthermore, $M > 0$ if and only if $C > 0$ and $A - BC^{-1}B^* \geq 0$.

The proof of Theorem 10.1 stems from the following easily verified identity

$$\begin{bmatrix} A & B \\ B^* & C \end{bmatrix} = \begin{bmatrix} I & BC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} A - BC^{-1}B^* & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} I & 0 \\ C^{-1}B^* & I \end{bmatrix}$$ (10.2)

Maybe the simplest use of the Schur complement concerns the scalar constraint: $xy > 1$, where $x$ and $y$ are both scalars. A plot of the points in the $x - y$ plane satisfying this constraint indicates that the specified region is convex. One should visualize this region as the points above the curve $y = 1/x$. Using the Schur complement, the constraint $y - x^{-1} > 0$ is converted to:

$$\begin{bmatrix} y & 1 \\ 1 & x \end{bmatrix} > 0$$ (10.3)

The computational attractiveness of constraint (10.3) stems from the linear appearance of variables $x$ and $y$. As such, this inequality is denoted as a **Linear Matrix Inequality** (LMI). However, it must be emphasize that LMIs are not linear constraints. This is due to the fact that $> 0$ represents positive definiteness, and not term by term positivity. As stated above the guarantee of positive definiteness is to ensure positivity of all the eigenvalues of the expression. While such a condition would appear to be highly nonlinear, and quite challenging to enforce, we are fortunate in the sense that others (see for example Boyd et al., [161] or Vandenberghe and Boyd, [162]) have shown these conditions to be convex in nature and thus have laid the groundwork for highly efficient numeric routines aimed at enforcing LMI constraints. One such routine can be found in the Robust Control Systems Toolbox provided by MATLAB. Another option is YALMIP, which provides a MATLAB interface to numerous LMI solvers available as shareware. It should also be noted that LMI based optimization problems are a special case of the more general class of Semi-Definite Programming (SDP) problems, for which many additional solvers are available (see for example Benson, [163]; Borchers and Young, [164]; Fujisawa, et al., [165]; Yamashita, et al., [166]; and Boyd and Vandenberghe, [167]). However, most of these routines require a recasting of the problem definition from a control related LMI form to a generalized matrix inequality form. Consider the following matrix function of a vector $p = [p_1 \ p_2 \ ... \ p_n]^* \in \mathbb{R}^n$.

$$F(p) = F_0 + \sum_{i=1}^{n} p_i F_i$$ (10.4)

where $F_i$, $i = 0 \ ... \ n$ are all self-adjoint matrices. Then, one can define a matrix inequality constraint as the set of $p \in \mathbb{R}^n$ such that

$$F(p) > 0$$ (10.5)

In the case of constraint (10.3), the following expression would result:

$$F(p) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + p_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + p_2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} > 0$$ (10.6)
Fortunately, the LMI solvers provided by MATLAB and YALMIP do not require this conversion and will accept the LMI conditions in their natural form.

Another example of using the Schur complement theorem is with regard to data reconciliation, discussed in Chapter 6, and the notion of sensor network design. Let us begin with a short review of data reconciliation. If \( s \) is a set of flow variables associated with a process flow network and \( C \) captures the material balance, then the vector of noise corrupted flow measurements can be stated as \( y = s + v = Cx + v \), where \( v \) is the vector of measurement noise terms, assumed to be a zero-mean Gaussian and with a covariance matrix \( \Sigma_v \). As indicated in Chapter 6, the optimal estimate of \( s \), given the measurements \( y \), is \( \hat{s} = (C^*\Sigma_v^{-1}C)^{-1}C^*\Sigma_v^{-1}y \). The estimation error is defined as \( e = s - \hat{s} \), and that of an individual stream is defined as \( e^{(j)} = s^{(j)} - \hat{s}^{(j)} \). Then the estimation error variance of stream \( j \) can be calculated as \( \zeta^{(j)} = \rho_j C(C^*\Sigma_v^{-1}C)^{-1}C^*\rho_j^* \). Then, one can define the performance requirements of the sensor network as: “The individual estimation error variances of the targeted streams \( (q) \) must below a given set of bounding parameters: \( \zeta^{(j)} < \zeta^{(j),max}, j = 1 \ldots n_q \).” Now we can apply the Schur complement theorem to state the performance specifications as a set of matrix inequalities

\[
\begin{bmatrix}
\zeta^{(j),max} \\
C^*\rho_j^*
\end{bmatrix} > 0, \quad j = 1 \ldots n_q
\]

(10.7)

The sensor network design problem arises by allowing the covariance matrix of the noise vector to contain design variables. Specifically, if the each of the measurement noise terms, of \( v \), is independent of the others, then \( \Sigma_v^{-1} \) will be diagonal:

\[
\Sigma_v^{-1} = \begin{bmatrix}
\sigma_{v1}^{-2} & 0 & 0 & 0 \\
0 & \sigma_{v2}^{-2} & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \sigma_{vn}^{-2}
\end{bmatrix}
\]

(10.8)

where each \( \sigma_{vi}^{-2} \) is the variance associated with the measurement at stream \( i \). If one defines inverse variance as \( \sigma_{vi}^{-2} = \gamma_i / \bar{\sigma}_{vi}^2 \), where \( \gamma_i \in \{0, 1\} \) is a zero-one attendance variables and \( \bar{\sigma}_{vi}^2 \) is a constant (representing the actual noise variance of a candidate measurement at stream \( i \)), then \( C^*\Sigma_v^{-1}C \) can be written as

\[
C^*\Sigma_v^{-1}C = C^* \begin{bmatrix}
\gamma_1 / \bar{\sigma}_{v1}^2 & 0 & 0 & 0 \\
0 & \gamma_2 / \bar{\sigma}_{v2}^2 & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & 0 & \gamma_{n_v} / \bar{\sigma}_{vn}^2
\end{bmatrix} = C^* \left( \sum_{i=1}^{n_v} (\gamma_i / \bar{\sigma}_{vi}^2) \rho_i^* \rho_i \right) C
\]

(10.9)

\[
= \sum_{i=1}^{n_v} \gamma_i (C^*\rho_i^*\rho_i C / \bar{\sigma}_{vi}^2) = \sum_{i=1}^{n_v} \gamma_i \Theta_i
\]
Thus, the above matrix inequality becomes:

\[
\begin{bmatrix}
\zeta_{\epsilon}^{(j),\max} & \rho_j^C \\
C^* \rho_j^* & \sum_{i=1}^{n_i} y_i \Theta_i
\end{bmatrix} > 0, \quad j = 1 \ldots n_q
\]  

(10.10)

where \( \Theta_i = C^* \rho_i^* \rho_i C / \sigma^2_v \) is a constant. Clearly, this is a LMI for each \( j \), since

\[
F_j(\gamma) = \begin{bmatrix}
\zeta_{\epsilon}^{(j),\max} & \rho_j^C \\
C^* \rho_j^* & 0
\end{bmatrix} + \sum_{i=1}^{n_i} y_i \begin{bmatrix} 0 & 0 \\ 0 & \Theta_i \end{bmatrix} > 0
\]  

(10.11)

is of the proper form. We will return to the sensor network design problem in Chapter 12.

### 10.2 Stabilizability and Detectability

While Chapter 4 provides necessary and sufficient conditions for stabilizability and detectability, each requires the calculation of subspaces and the rather tedious step of verifying that one subspace is a subset of another. As such, this section will introduce an alternative approach that is an application of LMIs. Specifically, we will search directly for the matrix that is the subject of each definition.

#### 10.2.1 LMI Conditions for Continuous-time Systems

The Lyapunov theorem is the enabling result for the development of the desired alternate conditions. As such, the continuous-time version is restated here for convenience.

**Theorem 4.3.** The following three statements are equivalent.

1. \( \dot{x} = Ax \) is stable
2. There exists \( P > 0 \) such that \( A^* P + P A < 0 \)
3. There exists \( P > 0 \) such that \( A P + P^* A < 0 \)

Using Theorem 4.3, the following corollaries are immediate.

**Corollary 10.1.** The system \( \dot{x} = Ax + Bu \) is stabilizable if and only if there exists \( P > 0 \) and a matrix \( L \) such that \( (A - BL)P + P(A - BL)^* < 0 \).

**Corollary 10.2.** The system \( \dot{x} = Ax; \ y = Cx \) is detectable if and only if there exists \( P > 0 \) and a matrix \( K \) such that \( (A - KC)^* P + P(A - KC) < 0 \).

While these corollaries provide alternate conditions, their computational utility is greatly diminished by the presence of the nonlinear terms \( LP \) and \( PK \). The following Theorems alleviate this issue.

**Theorem 10.2.** The system \( \dot{x} = Ax + Bu \) is stabilizable if and only if there exists \( Z_0 > 0 \) and a matrix \( Z \) such that \( (AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* < 0 \).

**Theorem 10.3.** The system \( \dot{x} = Ax; \ y = Cx \) is detectable if and only if there exists \( Z_0 > 0 \) and a matrix \( Z \) such that \( (Z_0 A - Z_1 C)^* + (Z_0 A - Z_1 C) < 0 \).
10.2. Stabilizability and Detectability

Proof of Theorem 10.2: If direction: Assume there exists $Z_0 > 0$ and $Z_1$ such that $(AZ_0 - BZ_1) + (A^* Z_0^{-1} Z_1^*) < 0$. Then define $P = Z_0$ and $L = Z_1 Z_0^{-1}$, where $Z_0^{-1}$ is guaranteed to exist since $Z_0 > 0$. Then, evaluation of the condition of Corollary 10.1 gives:

$$
(A - BL)P + P(A - BL)^* = (A - BZ_1 Z_0^{-1})Z_0 + Z_0(A - BZ_1 Z_0^{-1})^*
= (AZ_0 - BZ_1) + (A^* Z_0^{-1} Z_1^*)
$$

(10.12)

Thus, $(A - BL)P + P(A - BL)^* < 0$ and $\dot{x} = Ax + Bu$ is stabilizable.

Only if direction: If $\dot{x} = Ax + Bu$ is stabilizable, then there exists $P > 0$ and $L$ such that $(A - BL)P + P(A - BL)^* < 0$. If $Z_0$ and $Z_1$ are defined as $Z_0 = P$ and $Z_1 = LP$, then

$$(AZ_0 - BZ_1) + (A^* Z_0^{-1} Z_1^*) = (AP - BLP) + (AP - BLP)^*$$

(10.13)

Thus, $(AZ_0 - BZ_1) + (A^* Z_0^{-1} Z_1^*) < 0$, which completes the proof.

Based on the proof of Theorem 10.2 it is clear that if given $Z_0 > 0$ and $Z_1$ such that $(AZ_0 - BZ_1) + (A^* Z_0^{-1} Z_1^*) < 0$, then a stabilizing controller can be constructed as $L = Z_1 Z_0^{-1}$. Similarly, if given $Z_0 > 0$ and $Z_1$ such that $(Z_0 A - Z_1 C)^* + (Z_0 A - Z_1 C) < 0$, then a stabilizing estimator gain can be constructed as $K = Z_0^{-1} Z_1$. The proof of Theorem 10.3 is left as Exercise 10.7.

Example 10.1. Consider the pair of Example 3.7

$$A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix} \quad B = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

(10.14)

The code of Table 10.1 uses Theorem 10.1 to verify that this pair is stabilizable. The code could also be used to verify that the following pair is not stabilizable.

$$A = \begin{bmatrix} 4 & 6 \\ -3 & -5 \end{bmatrix} \quad B = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

(10.15)

It should be highlighted that most YALMIP solvers do not enforce strict matrix inequalities. That is, even if the symbol ‘$<$‘ is used, the solver will interpret as ‘$\leq$‘. The exception is the ‘solver’ which can only enforce a strict inequalities (i.e., ‘$<$‘ is always used, and ‘$\leq$‘ is interpreted as strict). In either case, the numeric difference between positive definite and positive semi-definite can sometimes be vague. Based on the proof of Theorem 4.3, one could change statement (3) to “There exists $P > 0$ such that $AP + PA^* + I \leq 0$” and still find it equivalent to statement (1). As such the code of Table 10.1 actually enforces $(AZ_0 - BZ_1) + (A^* Z_0^{-1} Z_1^*) + I < 0$. This can be shown to be a valid substitute for the original that does not suffer from the numeric vagueness of positive definite verses positive semi-definite enforcement. This problem is actually an artefact of the true / false nature of the stabilizable question. In subsequent sections, the questions will be more quantitative and the subtle difference between positive definite and positive semi-definite will be less concerning.

Table 10.1. Matlab code used in calculations for Example 10.1.

```matlab
clear all
% Continuous-time Model
nx=2; nu=1; AA=[4 6; -3 -5]; BB=[2; -1]; % BB=[-1; 1];
```
10.2.2 LMI Conditions for Discrete-time Systems

In the discrete-time framework, analogous results will be found. The only difference is that the conditions and required proofs are a bit more complicated. Again, we begin with a restatement of the Lyapunov theorem.

**Theorem 4.4.** The following three statements are equivalent (in the sense that if any one holds then the other two also hold).

1. \( x_{k+1} = A_dx_k \) is stable
2. There exists \( P > 0 \) such that \( A_dPA_d^T - P < 0 \)
3. There exists \( P > 0 \) such that \( A_d^TPA_d - P < 0 \)

Using Theorem 4.4, the following corollaries are immediate.

**Corollary 10.3.** The system \( x_{k+1} = A_dx_k + B_du_k \) is stabilizable if and only if there exists \( P > 0 \) and a matrix \( L \) such that \((A_d - B_dL)P(A_d - B_dL)^* - P < 0\).

**Corollary 10.4.** The system \( x_{k+1} = A_dx_k, y_k = Cx_k \) is detectable if and only if there exists \( P > 0 \) and a matrix \( K \) such that \((A_d - KC)^*P(A_d - KC) - P < 0\).

While these corollaries provide alternate conditions, their computational utility is greatly diminished by the presence of the nonlinear terms, which seem to be even more nonlinear than the continuous-time case. The following theorems alleviate this issue.

**Theorem 10.4.** The system \( x_{k+1} = A_dx_k + B_du_k \) is stabilizable if and only if there exists \( Z_0 > 0 \) and a matrix \( Z_1 \) such that

\[
\begin{bmatrix}
Z_0 & (A_dZ_0 - B_dZ_1) \\
(A_dZ_0 - B_dZ_1)^* & Z_0
\end{bmatrix} > 0
\]
Theorem 10.5. The system \( x_{k+1} = A_dx_k, y_k = Cx_k \) is detectable if and only if there exists \( Z_0 > 0 \) and a matrix \( Z_1 \) such that
\[
\begin{bmatrix}
Z_0 & (Z_0A_d - Z_1C)^* \\
(Z_0A_d - Z_1C) & Z_0
\end{bmatrix} > 0
\]

Proof of Theorem 10.4: The Schur complement theorem tells us that
\[
\begin{bmatrix}
Z_0 & (A_dZ_0 - B_dZ_1) \\
(A_dZ_0 - B_dZ_1)^* & Z_0
\end{bmatrix} > 0
\]
is equivalent to \((A_dZ_0 - B_dZ_1)Z_0^{-1}(A_dZ_0 - B_dZ_1)^* - Z_0 < 0\). If direction: Assume there exists \( Z_0 > 0 \) and \( Z_1 \) such that \((A_dZ_0 - B_dZ_1)Z_0^{-1}(A_dZ_0 - B_dZ_1)^* - Z_0 < 0\). Then define \( P = Z_0 \) and \( L = Z_1Z_0^{-1} \), where \( Z_0^{-1} \) is guaranteed to exist since \( Z_0 > 0 \). Then, evaluation of the condition of Corollary 10.3 gives:
\[
\]
\begin{align}
&= (A_d - B_dZ_1Z_0^{-1})Z_0(A_d - B_dZ_1Z_0^{-1})^* - Z_0 \\
&= (A_d - B_dZ_1Z_0^{-1})Z_0Z_0^{-1}(A_d - B_dZ_1Z_0^{-1})^* - Z_0 \\
&= (A_d - B_dZ_1Z_0^{-1})(A_d - B_dZ_1Z_0^{-1})^* - Z_0
\end{align}
(10.16)
Thus, \((A_d - B_dL)P(A_d - B_dL)^* - P < 0\) and \( x_{k+1} = A_dx_k + B_du_k \) is stabilizable.

Only if direction: If \( x_{k+1} = A_dx_k + B_du_k \) is stabilizable, then there exists \( P > 0 \) and \( L \) such that \((A_d - B_dL)P(A_d - B_dL)^* - P < 0\). If \( Z_0 \) and \( Z_1 \) are defined as \( Z_0 = P \) and \( Z_1 = LP \), then
\[
(A_d - B_dZ_1Z_0^{-1})(A_dZ_0 - B_dZ_1)^* - Z_0 = (A_dP - B_dLP)^{-1}(A_dP - B_dLP)^* - P
\]
(10.17)
Thus, \((A_d - B_dL)P(A_d - B_dL)^* - P < 0\), which completes the proof.

Table 10.2. Matlab code used in calculations for Example 10.2.
Example 10.2. Consider the discrete-time pair

\[
A_d = \begin{bmatrix}
1.3916 & 0.5729 \\
-0.2864 & 0.5323
\end{bmatrix} \quad B_d = \begin{bmatrix}
0.2104 \\
-0.1052
\end{bmatrix}
\]  

(10.18)

The code of Table 10.2 uses Theorem 10.4 to verify that this pair is stabilizable. The code could also be used to verify that the following pair is not stabilizable.

\[
A_d = \begin{bmatrix}
1.3916 & 0.5729 \\
-0.2864 & 0.5323
\end{bmatrix} \quad B_d = \begin{bmatrix}
-0.0906 \\
0.0906
\end{bmatrix}
\]  

(10.19)

Similar to Example 10.1, the issue of positive definite versus positive semi-definite suggests that the code of Table 10.2 should enforce

\[
\begin{bmatrix}
Z_0-I & (A_dZ_0-B_dZ_1) \\
(A_dZ_0-B_dZ_1)^* & Z_0
\end{bmatrix} > 0
\]  

(10.20)

rather than the original inequality of Theorem 10.4. See Example 10.1 for additional discussion.

10.3 Convexification of Covariance Equations

This section will introduce the primary computational tools to be used in the remainder of the book. Specifically, it will be shown that the nonlinear covariance equations associated with controller design can be put into the form of a LMI. Given this convex form, the controller design methods of subsequent chapters will reap significant computational benefit. However, the section begins with the study of open-loop stochastic processes, the results of which can be thought of as a slight generalization of the Lyapunov theorem. While Theorems 10.6 and 10.7 are likely of little computational utility, since the original open-loop covariance equations are linear relations, the proof techniques for these theorems provide a conceptual foundation that will be essential for the closed-loop cases.

10.3.1 Open-Loop Stochastic Processes

Consider the following continuous-time stochastic process driven by zero-mean white noise, \( w \), with a spectral density \( S_w \).

\[
\dot{x} = Ax + Gw 
\]  

(10.21)

\[
z = Dx
\]  

(10.22)
As indicated in Chapter 5, the steady-state covariance equations for this process are:

\[
\begin{align*}
A\Sigma_x + \Sigma_x A^* + GS_w G^* &= 0 \quad (10.23) \\
\Sigma_x &= D_x \Sigma_x D_x^* \quad (10.24) \\
\zeta^{(j)} &= \rho_j \Sigma_x \rho_j^* \quad (10.25)
\end{align*}
\]

where \(\zeta^{(j)}\) is the steady-state variance of the \(j\)th element of \(z\) and \(\Sigma_x\) and \(\Sigma_z\) are the steady-state covariance matrices associated with \(x\) and \(z\), respectively. In many cases, one would like to know if the steady-state variances are less than a set of bounds: \(\zeta^{(j)} < \zeta^{(j),\text{max}}, j = 1 \ldots n_z\). The following theorem addresses this question.

**Theorem 10.6.** Assume \(A\) is stable. Then, there exists \(\Sigma_x \geq 0\) and scalars \(\zeta^{(j)}\), \(j = 1 \ldots n_z\) such that

\[
\begin{align*}
A\Sigma_x + \Sigma_x A^* + GS_w G^* &= 0 \quad (10.26) \\
\rho_j D_x \Sigma_x D_x^* \rho_j^* &= \zeta^{(j)} < \zeta^{(j),\text{max}}, j = 1 \ldots n_z
\end{align*}
\]

if and only if there exists \(Z > 0\) and scalars \(\mu^{(j)}\), \(j = 1 \ldots n_z\) such that

\[
\begin{align*}
AZ_0 + Z_0 A^* + GS_w G^* &= 0 \quad (10.28) \\
\rho_j D_x Z_0 D_x^* \rho_j^* &= \mu^{(j)} < \zeta^{(j),\text{max}}, j = 1 \ldots n_z
\end{align*}
\]

The key elements of the proof of Theorem 10.6 are highlighted in the following lemmas.

**Lemma 10.1.** Assume \(A\) is stable and \(Z = Z^*\) is such that \(AZ + ZA^* < 0\). Then, \(Z > 0\).

**Proof.** Define \(Q = -AZ - ZA^*\). Then, define \(Z' = \int_0^\infty e^{At} Q e^{A^*t} dt\), which is easily concluded to be positive definite. From Section 5.7 it is known that \(Z'\) is the solution to \(AZ' + Z'A + Q = 0\). However, the definition of \(Q\) indicates: \(A(Z' - Z) - (Z' - Z)A^* = 0\). Since \(A\) is stable and does not have a zero eigenvector, it is concluded that \(Z' - Z = 0\). Thus, \(Z = Z' > 0\).

**Lemma 10.2.** Consider scalars \(x, y, z\) and \(\epsilon > 0\), If \(x\) is strictly less than \(z\), \(y = x + \epsilon\) and \(\epsilon\) can be selected arbitrarily small, then there exists \(\epsilon > 0\) such that \(y\) is strictly less than \(z\).

The proof of Lemma 10.2 becomes self-evident if \(x, y, z\) are put on a number line.

**Proof of Theorem 10.6:** If direction: Assume there exists \(Z_0 > 0\) and scalars \(\mu^{(j)} > 0\), \(j = 1 \ldots n_z\) such that \(AZ_0 + Z_0 A^* + GS_w G^* < 0\) and \(\rho_j D_x Z_0 D_x^* \rho_j^* < \mu^{(j)} < \zeta^{(j),\text{max}}\), \(j = 1 \ldots n_z\). Since \(A\) is stable, we can define

\[
\Sigma_x = \int_0^\infty e^{At} GS_w G^* e^{A^*t} dt
\]

Then, from the proof of Theorem 4.3, we know that \(\Sigma_x \geq 0\) and it satisfies: \(A\Sigma_x + \Sigma_x A^* + GS_w G^* = 0\). If this equality is subtracted from \(AZ_0 + Z_0 A^* + GS_w G^* < 0\), then one arrives at

\[
A(Z_0 - \Sigma_x) + (Z_0 - \Sigma_x)A^* < 0 \quad (10.31)
\]
Chapter 10. Matrix Inequalities

Then Lemma 10.1 tells us $\Sigma_x < Z_0$, which indicates that $\rho_j D_x \Sigma_x D_x^* \rho_j^* < \rho_j D_x Z_0 D_x^* \rho_j^*$ and leads to:

$$\zeta^{(j)} = \rho_j D_x \Sigma_x D_x^* \rho_j^* < \rho_j D_x Z_0 D_x^* \rho_j^* < \mu^{(j)} < \zeta^{(j), \max}$$

(10.32)

Thus, $\zeta^{(j)} < \zeta^{(j), \max}$, $j = 1 \ldots n_x$, which is the desired result.

Only if direction: Assume there exists $\Sigma_x \geq 0$ and scalars $\zeta^{(j)} > 0$, $j = 1 \ldots n_x$ such that $A \Sigma_x + \Sigma_x A^* + G S_w G^* = 0$ and $\rho_j D_x \Sigma_x D_x^* \rho_j^* = \zeta^{(j)} < \zeta^{(j), \max}$, $j = 1 \ldots n_x$. Define $Z_0$ as

$$Z_0 = \int_0^\infty e^{At} (G S_w G^* + \epsilon_0 I) e^{A^*_t} \, dt$$

(10.33)

where $\epsilon_0$ is a small positive scalar. It is then concluded that $Z_0 > 0$, $Z_0 = \Sigma_x + \epsilon_0 Q_0$ where $Q_0 > 0$ and $Z_0$ satisfies $A Z_0 + Z_0 A^* + G S_w G^* = -\epsilon_0 I < 0$. Next define

$$\mu = \rho_j D_x Z_0 D_x^* \rho_j^*$$

$$= \rho_j D_x \Sigma_x D_x^* \rho_j^* + \epsilon_0 \rho_j D_x Q_0 D_x^* \rho_j^*$$

(10.34)

$$= \zeta^{(j)} + \epsilon_j$$

where $\epsilon_j = \epsilon_0 \rho_j D_x Q_0 D_x^* \rho_j^*$. Thus, $\zeta^{(j)} < \zeta^{(j), \max}$ and $\mu^{(j)} = \zeta^{(j)} + \epsilon_j$, where $\epsilon_j > 0$ can be selected arbitrarily small, which by Lemma 10.2 indicates that $\mu^{(j)} < \zeta^{(j), \max}$.

To gain a bit of insight in to Theorem 10.6, consider a scalar system: $\dot{x} = ax + g w$. First note that $a$ must be negative for the system to be stable, which is required to find a finite $\Sigma_x > 0$. Then, rearrangement of (10.28) yields $Z_0 > g^2 S_w / 2a \geq \Sigma_x$. Thus, the set of feasible $Z_0$ with respect to (10.28) is simply those that are greater than $\Sigma_x$. Then, (10.29) will serve to bound $Z_0$ from above. The following example illustrates the same concepts for a multi-dimensional case.

![Figure 10.1. EDOR plots corresponding to $\Sigma_x$ and $Z_0$ from Example 10.3](image)

Example 10.3. Consider the following continuous-time model of a mass-spring-damper:

$$A = \begin{bmatrix} 0 & 1 \\ -3 & -0.2 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad S_w = 0.5 \quad D_x = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \zeta^{max} = 0.5$$

(10.35)
Using ‘lyap’ it is found that the solution to (10.26)-(10.27) is \( \Sigma_x = \begin{bmatrix} 0.4167 & 0.0 \\ 0.0 & 1.25 \end{bmatrix} \) and \( \zeta = 0.4167 \). Using an SDP routine to check the feasibility of (10.28)-(10.29) it is found to be feasible (as expected) and returns \( Z_0 = \begin{bmatrix} 0.4644 & -0.0032 \\ -0.0032 & 1.3947 \end{bmatrix} \) and \( \mu = 0.5 \). A comparison of the two yields \( Z_0 - \Sigma_x = \begin{bmatrix} 0.0477 & -0.0032 \\ -0.0032 & 0.1448 \end{bmatrix} \), and since the difference has positive eigenvalues (0.0476 and 0.1448) it is confirmed to be positive definite. It is instructive to note that if one were to make EDOR plots using \( \Sigma_x \) and \( Z_0 \), the one from \( Z_0 \) would be slightly larger than the one from \( \Sigma_x \), see Figure 10.1. As expected, setting \( \zeta^{\text{max}} \) to 0.4, results in (10.28)-(10.29) being infeasible.

The discrete-time result is nearly identical, but slightly more complicated due to the change in the state covariance equation.

**Theorem 10.7.** Assume \( A_d \) is stable. Then, there exists \( \Sigma_x > 0 \) and scalars \( \zeta(j), j = 1 \ldots n_z \) such that

\[
\Sigma_x = A_d \Sigma_x A_d^* + G_d \Sigma_w G_d^* \\
\rho_j(D_x \Sigma_x D_x^* + D_w \Sigma_w D_w^*) \rho_j^* = \zeta(j) < \zeta(j)_{\text{max}}, \quad j = 1 \ldots n_z
\]

(10.36)

if and only if there exists \( Z_0 > 0 \) and scalars \( \mu(j), j = 1 \ldots n_z \) such that

\[
\begin{bmatrix}
Z_0 - G_d \Sigma_w G_d^* & A_d Z_0 \\
Z_0 A_d^* & Z_0
\end{bmatrix} > 0
\]

(10.38)

\[
\rho_j(D_x \Sigma_x D_x^* + D_w \Sigma_w D_w^*) \rho_j^* < \mu(j) < \zeta(j)_{\text{max}}, \quad j = 1 \ldots n_z
\]

(10.39)

**Proof of Theorem 10.7.** By the Schur complement theorem, the LMI of (10.38) is equivalent to \( Z_0 - G_d \Sigma_w G_d^* - A_d Z_0 Z_0^{-1} Z_0 A_d^* < 0 \) or \( Z_0 > A_d Z_0 A_d^* + G_d \Sigma_w G_d^* \).

If direction: Assume there exists \( Z_0 > 0 \) and scalars \( \mu(j), j = 1 \ldots n_z \) such that \( Z_0 > A_d Z_0 A_d^* + G_d \Sigma_w G_d^* \) and \( \rho_j(D_x \Sigma_x D_x^* + D_w \Sigma_w D_w^*) \rho_j^* < \mu(j) < \zeta(j)_{\text{max}}, j = 1 \ldots n_z \). Since \( A_d \) is stable, we can define

\[
\Sigma_x = \sum_{k=0}^{\infty} A_d^k G_d \Sigma_w G_d^* A_d^k
\]

(10.40)

Then, we know that \( \Sigma_x > 0 \) and it satisfies: \( \Sigma_x = A_d \Sigma_x A_d^* + G_d \Sigma_w G_d^* \). If this equality is subtracted from \( Z_0 > A_d Z_0 A_d^* + G_d \Sigma_w G_d^* \), then one arrives at

\[
A_d(Z_0 - \Sigma_x)A_d^* - (Z_0 - \Sigma_x) < 0
\]

(10.41)

Then using methods similar to the proof of Lemma 10.1, it is concluded that \( \Sigma_x < Z_0 \), which indicates that \( \rho_j(D_x \Sigma_x D_x^* \rho_j^* < \rho_j(D_x Z_0 D_x^* \rho_j^* and leads to \( \zeta(j) < \zeta(j)_{\text{max}}, j = 1 \ldots n_z \).

Only if direction: Assume there exists \( \Sigma_x > 0 \) and scalars \( \zeta(j), j = 1 \ldots n_z \) such that \( \Sigma_x = A_d \Sigma_x A_d^* + G_d \Sigma_w G_d^* \) and \( \rho_j(D_x \Sigma_x D_x^* + D_w \Sigma_w D_w^*) \rho_j^* = \zeta(j) < \zeta(j)_{\text{max}}, j = 1 \ldots n_z \).
Closed-Loop Stochastic Processes

Given the proof techniques of the previous theorems of this chapter, we are now in a position to tackle the closed-loop cases. Consider the following continuous-time stochastic process driven by zero-mean white noise, \( w \), with a spectral density \( S_w \).

\[
\dot{x} = Ax + Bu + Gw \\
z = Dx + Du
\]

If the feedback policy is assumed to be linear (i.e., \( u = -Lx \)) then the steady-state covariance equations become:

\[
(A - BL)\Sigma_x + \Sigma_x(A - BL)^* + GS_wG^* = 0 \\
\Sigma_x = (Dx - DuL)\Sigma_x(Dx - DuL)^* \\
\zeta^{(j)} = \rho_j \Sigma_x \rho_j^*
\]

Given the closed-loop perspective of (10.46)-(10.48), which provides freedom with respect to the linear feedback parameter, \( L \), the following theorem addresses a question that is central to the design of a control system. That is, does there exist a feedback that can achieve the variance targets of the performance outputs?

**Theorem 10.8.** There exists \( \Sigma_x > 0 \), a matrix \( L \) and scalars \( \zeta^{(j)}, j = 1 \ldots n_x \) such that

\[
(A - BL)\Sigma_x + \Sigma_x(A - BL)^* + GS_wG^* = 0
\]

\[
\rho_j(Dx - DuL)\Sigma_x(Dx - DuL)^* \rho_j^* = \zeta^{(j)} < \zeta^{(j),\text{max}}, \ j = 1 \ldots n_x
\]

if and only if there exists \( Z_0 > 0 \), a matrix \( Z_1 \) and scalars \( \mu^{(j)}, j = 1 \ldots n_x \) such that

\[
(AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_wG^* < 0
\]

\[
\left[ \begin{array}{c} \mu^{(j)} \\ \rho_j(DxZ_0 - DuZ_1) \end{array} \right] < 0, \ j = 1 \ldots n_x
\]

and \( \mu^{(j)} < \zeta^{(j),\text{max}}, j = 1 \ldots n_x \)

**Proof of Theorem 10.8:** If direction: Assume there exists \( Z_0 > 0 \), a matrix \( Z_1 \) and scalars \( \mu^{(j)}, j = 1 \ldots n_x \) such that (10.51), (10.52) and (10.51) hold. If \( L = Z_1Z_0^{-1} \), then
\( (AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* = (A - BL)Z_0 + Z_0(A - BL)^* + GS_w G^* < 0, \) indicating that \((A - BL)\) must be stable. Define

\[
\Sigma_z = \int_0^\infty e^{(A - BL)t} GS_w G^* e^{(A - BL)^*t} dt
\]

Then, \(\Sigma_z\) satisfies: \((A - BL)\Sigma_z + \Sigma_z(A - BL)^* + GS_w G^* = 0\). If this is subtracted from \((A - BL)Z_0 + Z_0(A - BL)^* + GS_w G^* < 0\), then \((A - BL)(Z_0 - \Sigma_z) + (Z_0 - \Sigma_z)(A - BL)^* < 0\), which indicates that \(Z_0 > \Sigma_z\). Using the Schur complement theorem (10.52) is equivalent to

\[
\zeta^{(j),\max} > \mu^{(j)} > \rho_j (D_x Z_0 - D_u Z_1)Z_0^{-1}(D_x Z_0 - D_u Z_1)^* \rho_j^*
\]

\[
= \rho_j (D_x - D_u L)Z_0^{-1}(D_x - D_u L)^* \rho_j^*
\]

\[
> \rho_j (D_x - D_u L)\Sigma_z^{-1}(D_x - D_u L)^* \rho_j^* = \zeta^{(j)}
\]

This indicates that \(\zeta^{(j)} < \zeta^{(j),\max}, \ j = 1 \ldots n_z\), which is the desired result.

Only if direction: Assume there exists a stabilizing matrix \(L\) such that (10.49) and (10.50) hold. Define \(Z_0\) as

\[
\Sigma_z = \int_0^\infty e^{(A - BL)t} (GS_w G^* + \epsilon_0 I) e^{(A - BL)^*t} dt
\]

where \(\epsilon_0\) is a small positive scalar. Then one finds that \(Z_0 > 0\) and \(Z_0\) satisfies \((A - BL)Z_0 + Z_0(A - BL)^* + GS_w G^* + \epsilon_0 I = 0\). Thus, if \(Z_1\) is defined as \(Z_1 = LZ_0\), then

\[
(A - BL)Z_0 + Z_0(A - BL)^* + GS_w G^* = -\epsilon_0 I.
\]

Also, \(Z_0 = \Sigma_z + \epsilon_0 Q_0\), where \(Q_0 \geq 0\). Next, define

\[
\tau^{(j)} = \rho_j (D_x Z_0 - D_u Z_1)Z_0^{-1}(D_x Z_0 - D_u Z_1)^* \rho_j^*
\]

\[
= \rho_j (D_x - D_u L)Q_0(D_x - D_u L)^* \rho_j^*
\]

\[
+ \epsilon_0 \rho_j (D_x - D_u L)Q_0(D_x - D_u L)^* \rho_j^*
\]

\[
= \zeta^{(j)} + \epsilon_j
\]

where \(\epsilon_j = \epsilon_0 \rho_j (D_x - D_u L)Q_0(D_x - D_u L)^* \rho_j^*\). By assumption \(\zeta^{(j)}\) is strictly less than \(\zeta^{(j),\max}\). Thus, \(\tau^{(j)} = \zeta^{(j)} + \epsilon_j\) indicates that one can always select a sufficiently small \(\epsilon_j > 0\) such that \(\tau^{(j)} < \zeta^{(j),\max}\). Since this inequality is strict, there exists \(\mu^{(j)}\) such that

\[
\tau^{(j)} < \mu^{(j)} < \zeta^{(j),\max}.
\]

Then, this inequality along with the definition of \(\tau^{(j)}\) and the Schur complement theorem can be used to arrive at (10.52) and (10.53).

Based on the proof of Theorem 10.8, the conditions (10.51)-(10.53) could have also been written as:

\[
(AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0
\]

\[
\left[\begin{array}{c}
Z_3 \\
(D_x Z_0 - D_u Z_1)^* \\
Z_0
\end{array}\right] > 0
\]

and \(\rho_j Z_3 \rho_j^* < \zeta^{(j),\max}, \ j = 1 \ldots n_z\)
The possible advantage of this alternate formulation is that the \( n_x \) LMI constraints of (10.52) are replaced by the single LMI of (10.56). Thus, the problem will require less computational memory, which can be an issue for large problems, especially those with a large state vector. The trade-off is that the number of LMI variables will increase, since the matrix \( Z_0 \) contains off-diagonal variables (the diagonals are off-set by the elimination of the \( m^{(j)} \) variables). As such, this alternate formulation is likely to provide the greatest computational advantage when dimension of \( x \) is large and that of \( z \) is small.

**Example 10.4.** Consider the following continuous-time model of a mass-spring-damper:

\[
A = \begin{bmatrix} 0 & 1 \\ -3 & -0.2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad S_w = 0.5
\]

\[
D_x = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad D_a = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \zeta^{(1),\text{max}} = 0.5^2 = 0.25 \quad \zeta^{(1),\text{max}} = 0.2^2 = 0.04
\]

Using (10.51)-(10.53) within a SDP solver it is found that \( Z_0 = \begin{bmatrix} 0.2201 & -0.0044 \\ -0.0044 & 0.6765 \end{bmatrix} \), \( Z_1 = \begin{bmatrix} 0.0134 & 0.1438 \end{bmatrix}, \mu^{(1)} = 0.2326 \) and \( \mu^{(2)} = 0.0363 \) are feasible. Using the formula \( L = Z_1 Z_0^{-1} \), it is found that \( L = \begin{bmatrix} 0.0653 & 0.2130 \end{bmatrix} \) is a feasible linear feedback for (10.49)-(10.50). Using (10.49) one can calculate \( \Sigma_x = \begin{bmatrix} 0.1975 & 0.0 \\ 0.0 & 0.6053 \end{bmatrix} \), \( \zeta^{(1)} = 0.1975 \) and \( \zeta^{(2)} = 0.0283 \). It is interesting to note that again \( Z_0 - \Sigma_x \geq 0 \), and that \( \mu^{(j)} - \zeta^{(j)} \geq 0 \).

Let us now consider changes to the performance targets. If changed to \( \zeta^{(1),\text{max}} = 0.4^2 \) and \( \zeta^{(2),\text{max}} = 0.2^2 \), then the SDP solver will quickly indicate that this problem is infeasible. Clearly, such a conclusion would be difficult (if not impossible) to verify if given only the nonlinear relations (10.49)-(10.50). This infeasibility is due to the fact that we are asking the controller to reduce the variance of the first state variable by too much (\( \zeta^{(1),\text{max}} = 0.4^2 \)), while not giving enough power to the manipulated variable (\( \zeta^{(2),\text{max}} = 0.2^2 \)). However, if we increase the power to the manipulated variable, by setting to \( \zeta^{(2),\text{max}} = 0.3^2 \), then the problem will again become feasible. Similarly, if \( \zeta^{(1),\text{max}} = 0.3^2 \) and \( \zeta^{(2),\text{max}} = 0.3^2 \), the problem is infeasible, while \( \zeta^{(1),\text{max}} = 0.3^2 \) and

---

**Figure 10.2.** EDOR plots of \( \Sigma_x \) for Example 10.4. Left plot case 1. Right plot all 3 cases. Case 1 – \( \zeta^{(1),\text{max}} = 0.5^2 \) and \( \zeta^{(2),\text{max}} = 0.2^2 \). Case 3 – \( \zeta^{(1),\text{max}} = 0.4^2 \) and \( \zeta^{(2),\text{max}} = 0.3^2 \). Case 5 – \( \zeta^{(1),\text{max}} = 0.3^2 \) and \( \zeta^{(2),\text{max}} = 0.4^2 \).
\( \zeta^{(2),\text{max}} = 0.4^2 \), the problem is feasible. The EDOR plots associated with the above feasible cases are given in Figure 10.2. The MATLAB code of Table 10.3 was used for the calculations of this example.

Table 10.3. Matlab code for Example 10.4.

```matlab
clear all
% Continuous-time Model
nx=2; nu=1; nz=2; AA=[0 1; -3 -0.2]; BB=[0; 1]; GG=[0; 1]; Sw=0.5;
Dx=[1 0; 0 0]; Du=[0;1]; zeta_max = [0.1 0.2]; \%zeta_max = [0.1 0.2]; \%zeta_max = [0.2 0.1]; \%zeta_max = [0.2 0.025];
%
% Clear Old Variables
yalmpip('clear'); C1=[]; C2=[]; C3=[]; C4=[];
% Define Variables
Z0=sdpvar(nx); Z1=sdpvar(nu,nx); for ii=1:nz mu(ii)=sdpvar(1);
% Z0 >= 0
C1=[0 <= Z0];
% (AZ0 - BB*Z1) + (AZ0 - BB*Z1) + GG *Sw*GG <= 0
C2=[(AA*Z0-BB*Z1)+(AA*Z0-BB*Z1) + GG *Sw*GG] <= 0;
% [mu rhoj*(Dx*Z0-Du*Z1) (Dx*Z0-Du*Z1)'] >= 0
Izz=eye(nz);
for ii=1:nz rho_i=Izz(ii,:); C3=[C3, [mu(ii) rho_i *(Dx*Z0-Du*Z1)';(Dx*Z0-Du*Z1)']*rho_i' Z0 >=0];
% muj < zeta_max
for ii=1:nz C4=[C4, mu(ii) <= zeta_max(ii)]; end
% Define Constraints and Objective
Constraints=[C1,C2,C3,C4]; Objective=[];
% Determine Feasibility
options = sdpsettings('verbose',0,'solver','mosek');
sol = optimize(Constraints,Objective,options);
if sol.problem == 0 \% problem is feasible
Z0feas=value(Z0); Z1feas=value(Z1);
for ii=1:nz mufeas(ii)=value(mu(ii)); end
ZLfeas=(Dx*Z0feas-Du*Z1feas)*inv(Z0feas)*(Dx*Z0feas-Du*Z1feas);
LLfeas=Z1feas/Z0feas, eig(AA-BB*LLfeas)
Sigx=lyap(AA-BB*LLfeas,GG*Sw*GG')
Sigz=(Dx-Du*LLfeas)*Sigx*(Dx-Du*LLfeas)
elseif sol.problem == 1 \% problem is infeasible
display('Infeasible Problem');
else
display('Hmm, something went wrong!'); sol.info; pause
end
```

Example 10.5. Consider the furnace reactor system of Figure 10.3. The system matrices of the process model are as follows:

\[
A = \begin{bmatrix}
-8000 & 0 & 0 & 0 & 10000 \\
2000 & -1500 & 0 & 0 & 0 \\
0 & 0 & -5000 & 0 & 0 \\
0 & 0 & 0 & -5000 & 0 \\
0 & 0 & 0 & 0 & -10
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
-75 & 75000 & 0 \\
-25 & 0 & 0 \\
0 & -8500 & -8.5 \times 10^5 \\
0 & 0 & -5.0 \times 10^7 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
G = \begin{bmatrix}
0 \\
0 \\
0 \\
S_w = 64 \\
0 \\
10
\end{bmatrix}
\]
Using these matrices, the performance outputs correspond to:

1) Feed Temperature
2) Furnace Temperature
3) Reactor Temperature
4) \(O_2\) Concentration
5) CO Concentration
6) Reactant Feed Rate
7) Fuel Feed Rate
8) Vent Position

If the performance bounds are set to \(\zeta_{\text{max}} = [50^2 \ 10^2 \ 5^2 \ 2.5^2 \ 100^2 \ 2.5^2 \ 0.01^2]\), then using (10.51)-(10.53) within a SDP solver it is found that use of the following feedback gain \((L = Z_1 Z_0^{-1})\) will result in a satisfaction of these bounds

\[
L = \begin{bmatrix}
0.499 & 1.21 & 8.94 & -0.0403 & -6.61 \\
-0.155 & 0.12 & -0.492 & -0.00263 & 0.183 \\
-0.0000419 & 0.000183 & 0.000316 & 0.0000993 & -0.0000723
\end{bmatrix}
\]

Specifically, if this gain is used within (10.49), then (10.50) will be satisfied. For a more graphical interpretation, consider the \(\Sigma_z\) calculated from (10.47). Figure 10.4 depicts the resulting EDOR plots along with the corresponding standard deviation bounds. To highlight the difference between \(\Sigma_z\) and the LMI variables, define \(Z_3\) similar to that of (10.56), \(Z_3 = (D_x Z_0 - D_u Z_1) Z_0^{-1} (D_x Z_0 - D_u Z_1)^*\). Figure 10.4 illustrates that the \(Z_3\) EDOR is closer to the constraints in the sense that it is between the \(\Sigma_z\) EDOR and constraints. In Chapter 10, we will find that the CMV approach significantly reduces the discrepancy between \(Z_3\) and the more realistic \(\Sigma_z\).

If one would like to reduce the standard deviation of reactor temperature from 5 to 0.5, the performance bounds could be set to \(\zeta_{\text{max}} = [50^2 \ 10^2 \ 0.5^2 \ 2.5^2 \ 50^2 \ 100^2 \ 2.5^2\)
In this second case, (10.51)-(10.53) is found to be infeasible, indicating that there is no controller capable of satisfying these bounds. To achieve the desired reduction in reactor temperature standard deviation, one of the other bounds will need to be increased. For example, that of reactant feed rate could be increased from 100 to 200. Case 3: $\zeta_{max} = [50^2 \ 10^2 \ 0.5^2 \ 2.5^2 \ 50^2 \ 200^2 \ 2.5^2 \ 0.01^2]$. Based on the EDOR plots of Figure 10.5, we see that such a change will provide this manipulated variable (reactant feed rate) with greater influence over the process.

If changes reactant feed rate variance should be avoided, then one could look to the other manipulated variables. For example, one could increase the standard deviation bound on fuel feed rate from 2.5 to 5. Case 4: $\zeta_{max} = [50^2 \ 10^2 \ 0.5^2 \ 2.5^2 \ 50^2 \ 100^2 \ 5^2 \ 0.01^2]$. In this case, the problem is found to be infeasible, and efforts to increase further do not change the situation. Based on Figure 10.4 this should make sense, since the bound on fuel feed rate is not active. The actual limitation is with regard to $O_2$ concentration. So, an increase in the bound on $O_2$ concentration should free up the fuel feed rate variable. If an increase at $O_2$ concentration is not possible (say for safety reasons), then one could look to vent position, which also influences $O_2$ concentration. However, that manipulated variable is limited by the CO concentration bound. Based on this logic, one could increase the bound on CO concentration to allow more freedom to the vent position. This additional freedom to the vent position variable can then be used to reduce variability of the $O_2$ concentration, so that the fuel feed rate will have more freedom to regulate reactor temperature. This is executed by simply changing the standard deviation bound on CO concentration from 50 to 70. Case 5: $\zeta_{max} = [50^2 \ 10^2 \ 0.5^2 \ 2.5^2 \ 70^2 \ 100^2 \ 2.5^2 \ 0.01^2]$. The result is found in Figure 10.6.
Figure 10.5. EDOR plots comparing Case 1 and Case 3 for Example 10.5.

Figure 10.6. EDOR plots comparing Case 1 and Case 5 for Example 10.5.
While the manipulations of this example seem intuitive (especially given the EDOR plots), remember that a feedback controller, \( L = Z_1 Z_0^{-1} \), has been synthesized for each feasible case. If one did not have the LMI conditions, then for each choice of one would need to find a controller gain such that Equations (10.49)-(10.50) would be satisfied. Even with the aid of the LQOC procedure, this would be an additional trial-and-error step that is at best tedious and at worst intractable.

The discrete-time version of Theorem 10.8 possesses a proof that is nearly identical and thus is left as Exercise 10.9.

**Theorem 10.9.** There exists \( \Sigma_x \geq 0 \), a matrix \( L \) and scalars \( \zeta^{(j)} \), \( j = 1 \ldots n_z \) such that

\[
\Sigma_x = (A_d - B_d L) \Sigma_x (A_d - B_d L)^* + G_d \Sigma_w G_d^*
\]

\[
\rho_j (D_x - D_u L) \Sigma_x (D_x - D_u L)^* \rho_j^*
\]

\[
+ \rho_j D_w \Sigma_w D_w^* \rho_j^* < \zeta^{(j),\text{max}}, \quad j = 1 \ldots n_z
\]

if and only if there exists \( Z_0 > 0 \), a matrix \( Z_1 \) and scalars \( \mu^{(j)} \), \( j = 1 \ldots n_z \) such that

\[
\begin{bmatrix}
Z_0 - G_d \Sigma_w G_d^* & (A_d Z_0 - B_d Z_1) \\
(A_d Z_0 - B_d Z_1)^* & Z_0
\end{bmatrix} > 0
\]  

\[
\begin{bmatrix}
\mu^{(j)} - \rho_j D_w \Sigma_w D_w^* \rho_j^* & \rho_j (D_x Z_0 - D_u Z_1) \\
(D_x Z_0 - D_u Z_1)^* \rho_j^* & Z_0
\end{bmatrix} > 0, \quad j = 1 \ldots n_z
\]

and \( \mu^{(j)} < \zeta^{(j),\text{max}}, \quad j = 1 \ldots n_z \)  

In some cases, it may be useful to use the Schur complement theorem to convert (10.61) to the following equivalent form:

\[
\begin{bmatrix}
Z_0 & (A_d Z_0 - B_d Z_1) & G_d \\
(A_d Z_0 - B_d Z_1)^* & Z_0 & 0 \\
G_d^* & 0 & \Sigma_w^{-1}
\end{bmatrix} > 0
\]

A similar form also exists for (10.51).

10.4 • Riccati Equations

This section addresses the question of converting the Riccati equation into a convex form. It should be highlighted that the utility of these results are greatly overshadowed by those of the previous section, especially in the control related case. In fact, in Chapter 10 it will be shown that the covariance related LMIs can be used to solve the LQOC problem. However, in the case of the state estimator based Riccati equation, the convex form will be of great utility in the formulation of the cost optimal sensor selection problem. For completeness, we begin with the control related form of the Riccati equation.

**Theorem 10.10.** Assume \( R > 0 \) and the pair \((A, Q)\) is detectable. Then, there exists \( P \geq 0 \) such that \( (A - BR^{-1}B^*P) \) is stable and

\[
A^* P + PA + Q - PBR^{-1}B^* P \leq 0
\]
if and only if there exists \( M_0 > 0 \), such that
\[
A^*M_0 + M_0A + Q - M_0BR^{-1}B^*M_0 < 0
\] (10.66)

**Proof of Theorem 10.10:**

**If direction:** Assume there exists \( M_0 > 0 \), such that (10.66) holds. Since (10.66) can be written as
\[
(A - BR^{-1}B^*M_0)^*M_0 + M_0(A - BR^{-1}B^*M_0) + Q + M_0BR^{-1}B^*M_0 < 0
\] (10.67)
it is found that \( (A - BR^{-1}B^*M_0)^*M_0 + M_0(A - BR^{-1}B^*M_0) < 0 \) and \( (A - BR^{-1}B^*M_0) \) must be stable. Thus, the pair \((A, B)\) is stabilizable, which is the only requirement for the existence of a positive semi-definite solution to the Riccati equation. This along with \((A, Q)\) being detectable, indicates that (10.66) is equivalent to (10.65) holds. Define \( M_0 \) as
\[
M_0 = \int_0^\infty e^{(A - BR^{-1}B^*)t}((Q + PBR^{-1}B^*P + \epsilon_0I)e^{(A - BR^{-1}B^*)t})^*dt
\] (10.68)
where \( \epsilon_0 \) is a small positive scalar. Then one finds that \( M_0 > 0 \) and \( M_0 \) satisfies \( (A - BR^{-1}B^*)^*M_0 + M_0(A - BR^{-1}B^*) + Q + PBR^{-1}B^*P + \epsilon_0I \) = 0. If \( M_0BR^{-1}B^*M_0 \) is added and subtracted from this equation and one completes the square the following is determined:
\[
A^*M_0 + M_0A + Q - M_0BR^{-1}B^*M_0 = -(P - M_0)BR^{-1}B^*(P - M_0) - \epsilon_0I < 0.
\]
If \( W_0 > 0 \) is defined as \( W_0 = W^{-1} \), then \( A^*M_0 + M_0A + Q - M_0BR^{-1}B^*M_0 < 0 \) is equivalent to \( W_0A^* + AW_0 + W_0QW_0 - BR^{-1}B^* < 0 \). Then, application of the Schur complement theorem indicates that (10.66) is equivalent to
\[
\begin{bmatrix}
BR^{-1}B^* - W_0A^* - AW_0 & W_0Q^{1/2} \\
Q^{1/2}W_0 & I
\end{bmatrix} > 0
\] (10.69)

The slight advantage provided by Theorem 10.10 is that if \( Q \) is not selected to be such that the pair \((A, Q)\) is detectable, then the LMI of (10.69) will generate a controller of the form \((A - BR^{-1}B^*)P\) that is guaranteed stable. This is because the strict inequality of (10.67) will implicitly include the term as a companion to \( Q \) and guarantee detectability of \( Q + \epsilon_0I \).

In the case of the dual of the Riccati equation, its association with the Kalman filter provides for greater physical significance and thus much greater utility. In particular, the following theorem, a slight generalization of 10.10, will form the basis of the open-loop sensor selection problem to be discussed in the next section.

**Theorem 10.11.** Assume \( S_v > 0, S_w > 0 \) and the pair \((A, G)\) is stabilizable. Then, there exists \( \Sigma_e \) such that \((A - \Sigma_eC^*S_v^{-1}C)\) is stable, \( \rho_j \Sigma_e \rho_j^* < \zeta_e^{(j)\text{max}}, j = 1 \ldots n_s \) and
\[
A\Sigma_e + \Sigma_eA^* + GS_wG^* - \Sigma_eC^*S_v^{-1}C\Sigma_e = 0
\] (10.70)

if and only if there exists \( M_0 > 0 \), such that \( \rho_j M_0 \rho_j^* < \zeta_e^{(j)\text{max}}, j = 1 \ldots n_s \) and
\[
AM_0 + M_0A^* + GS_wG^* - M_0C^*S_v^{-1}CM_0 < 0
\] (10.71)
Proof of Theorem 10.11: If direction: Assume there exists \( M_0 > 0 \), such that (10.71) holds. In this case, (10.71) can be written as

\[
(A - M_0 C^* S_v^{-1} C) M_0 + M_0 (A - M_0 C^* S_v^{-1} C)^* + G S_w G^* + M_0 C^* S_v^{-1} C M_0 < 0
\]

(10.72)

This indicates that \( (A - M_0 C^* S_v^{-1} C) M_0 + M_0 (A - M_0 C^* S_v^{-1} C)^* < 0 \) and \( (A - M_0 C^* S_v^{-1} C) \) must be stable. Thus, the pair \((A, C)\) is detectable, which is the only requirement for the existence of a positive semi-definite solution to (10.70). This along with \((A, G)\) being stabilizable, indicates that \((A - \Sigma_e C^* S_v^{-1} C)\) is stable (see Theorem 6.1). Subtracting (10.70) from (10.72), one finds:

\[
(A - M_0 C^* S_v^{-1} C) (M_0 - \Sigma_e) + (M_0 - \Sigma_e) (A - M_0 C^* S_v^{-1} C)^* + (M_0 - \Sigma_e) C^* S_v^{-1} C (M_0 - \Sigma_e) < 0
\]

(10.73)

Thus, \((A - M_0 C^* S_v^{-1} C) (M_0 - \Sigma_e) + (M_0 - \Sigma_e) (A - M_0 C^* S_v^{-1} C)^* < 0\). Using Lemma 10.1 along with the fact that \((A - M_0 C^* S_v^{-1} C)\) is stable, one concludes that \( M_0 - \Sigma_e > 0 \) and \( \rho_j \Sigma_e \rho_j^* < \rho_j M_0 \rho_j^* < \zeta_{e(j)}^{(j),\text{max}}, j = 1 \ldots n_j \).

Only if direction: Assume there exists \( \Sigma_e > 0 \) such that \((A - \Sigma_e C^* S_v^{-1} C)\) is stable, \( \rho_j \Sigma_e \rho_j^* < \zeta_{e(j)}^{(j),\text{max}}, j = 1 \ldots n_j \) and (10.71) holds. Define \( M_0 \) as

\[
M_0 = \int_0^\infty e^{(A - \Sigma_e C^* S_v^{-1} C) t} (G S_w G^* + \Sigma_e C^* S_v^{-1} C \Sigma_e + \epsilon_0 I) e^{(A - \Sigma_e C^* S_v^{-1} C)^*} \, dt
\]

(10.74)

where \( \epsilon_0 \) is a small positive scalar. Then one finds that \( M_0 > 0 \) and \( M_0 \) satisfies:

\[
(A - \Sigma_e C^* S_v^{-1} C) M_0 + M_0 (A - \Sigma_e C^* S_v^{-1} C)^* + G S_w G^* + \Sigma_e C^* S_v^{-1} C \Sigma_e + \epsilon_0 I = 0
\]

(10.75)

If one completes the square, with the use of \( M_0 C^* S_v^{-1} C M_0 \), then the following is determined:

\[
A M_0 + M_0 A^* + G S_w G^* - M_0 C^* S_v^{-1} C M_0 = -(\Sigma_e - M_0) C^* S_v^{-1} C (\Sigma_e - M_0) - \epsilon_0 I < 0
\]

(10.76)

Furthermore, the definition of \( M_0 \) indicates that \( M_0 = \Sigma_e + \epsilon_0 D \), where \( D > 0 \). Thus, \( \rho_j M_0 \rho_j^* = \rho_j \Sigma_e \rho_j^* + \epsilon_0 \), where \( \epsilon_0 = \epsilon_0 \rho_j D \rho_j^* \). Then, since \( \rho_j \Sigma_e \rho_j^* < \zeta_{e(j)}^{(j),\text{max}} \) one can select sufficiently small \( \epsilon_0 \) such that \( \rho_j M_0 \rho_j^* < \zeta_{e(j)}^{(j),\text{max}} \) (recall Lemma 10.2).

If \( W_0 \) is defined as \( W_0 = M_0^{-1} \), then \( A M_0 + M_0 A^* + G S_w G^* - M_0 C^* S_v^{-1} C M_0 < 0 \) is equivalent to \( W_0 A + A^* W_0 + W_0 G S_w G^* W_0 - C^* S_v^{-1} C < 0 \). Then, application of the Schur complement theorem indicates that (10.71) and \( \rho_j M_0 \rho_j^* < \zeta_{e(j)}^{(j),\text{max}}, j = 1 \ldots n_j \) are equivalent to

\[
\begin{bmatrix}
C^* S_v^{-1} C - W_0 A - A^* W_0 & W_0 G \\
G^* W_0 & S_w^{-1}
\end{bmatrix} > 0 \quad \text{and} \quad \begin{bmatrix}
\zeta_{e(j)}^{(j),\text{max}} & \rho_j \\
\rho_j & W_0
\end{bmatrix} > 0
\]

(10.77)

Clearly, these matrix inequalities are linear with respect to the variable \( W_0 \). They are also linear with respect to \( S_v^{-1} \). This fact will be central to hardware selection problems of Chapter 12.
Table 10.4. Matlab code for Example 10.6.

```matlab
clear all
% Continuous-time Model
nx=2; ny=1; AA=[0 1; -3 -0.2]; GG=[0; 1]; Sw=0.5; CC=[0 1]; Sv=0.1;
zeta_max = [0.1 0.5]; Ixx=eye(nx);

% ARE approach
Sige=care(AA',CC',GG*Sw*GG',Sv); diag(Sige)

% SDP approach
yalmp('clear'); C1=[]; C2=[]; C3=[]; C4=[];
W0=sdpvar(nx); Svinv=sdpvar(1);
C1=[W0>=0];
C2=[[CC'Svinv*CC=W0*AA-AA'W0; GG*W0 inv(Sw)]>=0];
for ii=1:nx rho_i=Ixx(ii,:);
C3=[C3, [zeta_max(ii) rho_i; rho_i'
C4=[Svinv >= 0];
Constraints=[C1,C2,C3,C4]; Objective=Svinv;
options = sdpsettings('verbose',0,'solver','mosek');
sol = optimize(Constraints,Objective,options);
if sol.problem == 0 % problem is feasible
    W0feas=value(W0); Svinvfeas=value(Svinv);
zeta=diag(inv(W0feas)), Svmax=inv(Sv).
else
    display('Found feasible solution!');
end
```

Example 10.6. Consider the following continuous-time model of a mass-spring-damper:

\[
\begin{align*}
A &= \begin{bmatrix} 0 & 1 \\ -3 & -0.2 \end{bmatrix}, \\
G &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \\
S_w &= 0.5 \\
C &= \begin{bmatrix} 0 & 1 \end{bmatrix}, \\
S_v &= 0.5 \\
\zeta_1 &= 0.1366, \\
\zeta_2 &= 0.40999
\end{align*}
\]  

Using (10.70), it is found that \( \zeta_1 = 0.1366 \) and \( \zeta_2 = 0.40999 \). Thus, it is clear that the conditions \( \rho_j M_0^+ \rho_j^* < \zeta_j^{(i),\text{max}} \) along with (10.71) will not be satisfied. However, if \( S_v \) is changed to 0.1, then \( \zeta_1 = 0.0682, \zeta_2 = 0.2045 \) and the estimation problem will be feasible. If one would like to know the largest value of \( S_v \) that will yield a feasible estimation problem, then the code of Table 10.4 could be employed. This code indicates this largest value of \( S_v \) to be 0.2368.

Theorem 10.12. Assume \( \Sigma_v > 0, \Sigma_w > 0 \) and the pair \((A_d, G_d)\) is stabilizable. Then, there exists \( \Sigma_e \geq 0 \) such that \((A_d - K^+ C)\) is stable, \( K^+ = A_d \Sigma_e^+ C^+ (C \Sigma_e^+ C^+ + \Sigma_v)^{-1}, \rho_j \Sigma_e^+ \rho_j^* < \zeta_j^{(i),\text{max}}, j = 1 \ldots n_s \) and

\[
-\Sigma_e^+ + A_d (\Sigma_e^+ - \Sigma_e^+ C^+ (C \Sigma_e^+ C^+ + \Sigma_v)^{-1} C \Sigma_e^+) A_d^* + G_d \Sigma_w G_d^* = 0
\]  

if and only if there exists \( M_0^+ > 0 \), such that \( \rho_j M_0^+ \rho_j^* < \zeta_j^{(i),\text{max}}, j = 1 \ldots n_s \) and

\[
-M_0^+ + A_d (M_0^+ - M_0^+ C^+ (C M_0^+ C^+ + \Sigma_v)^{-1} C M_0^+) A_d^* + G_d \Sigma_w G_d^* < 0
\]
10.4. Riccati Equations

**Proof of Theorem 10.12:** If direction: Assume there exists $M^+_0 > 0$, such that (10.80) holds. If $K^+ \triangleq A_d M^+_0 C^*(C M^+_0 C^* + \Sigma_v)^{-1}$, then (10.80) can be written as

$$-M^+_0 + (A_d - K^+_0 C)M^+_0 (A_d - K^+_0 C)^* + G_d \Sigma_w G_d^* + K^+_0 \Sigma_v K^+_0 < 0$$

(10.81)

This indicates that $-M^+_0 + (A_d - K^+_0 C)M^+_0 (A_d - K^+_0 C)^* < 0$ and must be stable. Thus, the pair $(A, C)$ is detectable, which is the only requirement for the existence of a positive semi-definite solution to (10.79). This along with $(A, G)$ being stabilizable, indicates that $(A_d - K^+_0 C)$ is stable. Subtracting (10.79) from (10.81), one finds:

$$-(M^+_0 - \Sigma^+_e) + (A_d - K^+_0 C)(M^+_0 - \Sigma^+_e)(A_d - K^+_0 C)^* + (K^+_0 - K^+)(C M^+_0 C^* + \Sigma_v)(K^+_0 - K^+) < 0$$

(10.82)

Thus $-(M^+_0 - \Sigma^+_e) + (A_d - K^+_0 C)(M^+_0 - \Sigma^+_e)(A_d - K^+_0 C)^* < 0$. Using the discrete-time version of Lemma 10.1 along with the fact that $(A_d - K^+_0 C)$ is stable, one concludes that $M^+_0 - \Sigma^+_e > 0$ and $\rho_j \Sigma^+_e \rho_j^* < \rho_j \Sigma^+_e \rho_j^* < \zeta_e^{(j), \text{max}}$, $j = 1 \ldots n_d$.

Only if direction: Assume there exists $\Sigma^+_e > 0$ such that $(A_d - K^+ C)$ is stable, $\rho_j \Sigma^+_e \rho_j^* = \zeta_e^{(j), \text{max}}$, $j = 1 \ldots n_d$ and (10.80) holds. Define $M^+_0$ as

$$M^+_0 = \sum_{k=0}^{\infty} (A_d - K^+ C)^k (G_d \Sigma_w G_d^* + K^+ \Sigma_v K^+ + \epsilon_0 I) (A_d - K^+ C)^k$$

(10.83)

where $\epsilon_0$ is a small positive scalar. Then one finds that $M^+_0 > 0$ and $M^+_0$ satisfies:

$$-M^+_0 + (A - K^+ C) M^+_0 (A - K^+ C)^* + G_d \Sigma_w G_d^* + K^+ \Sigma_v K^+ + \epsilon_0 I = 0$$

(10.84)

If one completes the square, with the use of $K^+_0 (C M^+_0 C^* + \Sigma_v) K^+_0$, then the following is determined:

$$-M^+_0 + A_d (M^+_0 - M^+_0 C^*(C M^+_0 C^* + \Sigma_v^{-1}) C M^+_0 A^*_d + G_d \Sigma_w G_d^*$$

$$= -(K^+_0 - K^+)(C M^+_0 C^* + \Sigma_v)(K^+_0 - K^+) - \epsilon_0 I < 0$$

(10.85)

Furthermore, the definition of $M^+_0$ indicates that $M^+_0 = \Sigma^+_e + \epsilon_0 D$, where $D > 0$. Thus, $\rho_j M^+_0 \rho_j^* = \rho_j \Sigma^+_e \rho_j^* + \epsilon_0 \rho_j D \rho_j^*$. Then, since $\rho_j \Sigma^+_e \rho_j^* < \zeta_e^{(j), \text{max}}$ one can select sufficiently small $\epsilon_0$ such that $\rho_j M^+_0 \rho_j^* < \zeta_e^{(j), \text{max}}$ (recall Lemma 10.2).

If the matrix inversion lemma is applied to (10.80) one finds:

$$-M^+_0 + A_d \left[(M^+_0)^{-1} + C^* \Sigma_v^{-1} C\right]^{-1} A^*_d + G_d \Sigma_w G_d^* < 0$$

(10.86)

If $W^+_0 > 0$ is $W^+_0 = (M^+_0)^{-1}$ and (10.86) is multiplied by $W^+_0$ on both sides, one finds:

$$-W^+_0 + W^+_0 A_d \left[W^+_0 + C^* \Sigma_v^{-1} C\right]^{-1} A^*_d W^+_0 + W^+_0 G_d \Sigma_w G_d^* W^+_0$$

$$= -W^+_0 + [W^+_0 A_d \ W^+_0 G_d] \left[C^* \Sigma_v^{-1} C\right]^{-1} [0 \Sigma_w] \left[W^+_0 A_d \ W^+_0 G_d\right]^* < 0$$

(10.87)
Then, use of the Schur complement theorem on (10.87) and \( \rho_j(W_0^+ \rho^+_j) \) gives:

\[
egin{bmatrix}
W_0^+ & W_0^+ A_d & W_0^+ G_d \\
(W_0^+ A_d)^\tau & W_0^+ + C^* \Sigma_v^{-1} C & 0 \\
(W_0^+ G_d)^\tau & 0 & \Sigma_w^{-1}
\end{bmatrix} > 0
\]  
(10.88)

\[
\begin{bmatrix}
\rho_j & \rho_j W_0^+ \\
\rho_j & W_0^+ + C^* \Sigma_v^{-1} C
\end{bmatrix} > 0
\]  
(10.89)

Clearly, these matrix inequalities are linear with respect to the variable \( W_0^+ \).

**Theorem 10.13.** Assume \( \Sigma_v > 0, \Sigma_w > 0 \) and the pair \( (A_d, G_d) \) is stabilizable. Then, there exists \( \Sigma_e \) such that \( (A_d - KC) \) is stable, \( K = \Sigma_e C^* \Sigma_v^{-1} \), \( \rho_j \Sigma_e \rho_j^* \leq \zeta_e(i), j = 1 \ldots n_s \), and

\[
-\Sigma_e + \left[ (A_d \Sigma_e A_d^* + G_d \Sigma_w G_d^*)^{-1} + C^* \Sigma_v^{-1} C \right]^{-1} = 0
\]  
(10.90)

if and only if there exists \( W_0^+ > 0 \), such that (10.88) holds along with

\[
\begin{bmatrix}
\zeta_e(i), \max \\
\rho_j \\
\rho_j \\
W_0^+ + C^* \Sigma_v^{-1} C
\end{bmatrix} > 0, \quad j = 1 \ldots n_s
\]  
(10.91)

**Proof of Theorem 10.13:** The first part of Theorem 10.13 is equivalent to the first part of Theorem 10.12, but with \( \rho_j \Sigma_e \rho_j^* \leq \zeta_e(i), \max, j = 1 \ldots n_s \) replaced by

\[
\rho_j \Sigma_e \rho_j^* = \rho_j \left( (\Sigma_e^+)^{-1} + C^* \Sigma_v^{-1} C \right)^{-1} \rho_j^* \leq \zeta_e(i), \max, j = 1 \ldots n_s
\]  
(10.92)

This is due to Equation (6.87). Theorem 10.12 can then be used to find the conditions of (10.88) along with \( \rho_j(W_0^+ + C^* \Sigma_v^{-1} C)^{-1} \rho_j^* < \zeta_e(i), \max \). Then, application of the Schur complement theorem leads to (10.91).

### 10.5 Chapter Summary

This chapter has provided a computational foundation for the economic based controller design methods to be discussed in subsequent chapters. The most important element being that all conditions are of the convex variety and can be enforced using numerically robust algorithms. Most notably, an optimization subject to convex constraints is guaranteed to arrive at a global solution. The set of conditions that will be of greatest use in subsequent chapters are those associated with covariance based controller design (Theorems 10.8 and 10.9). The main benefit of this design scheme is that variance based performance criteria are used for controller tuning. Since variance based criteria are more relatable to the operational objectives of a controlled system, the selection of criteria are more intuitive than the LQOC tuning method of selecting objective function weights.

While much of this chapter focuses on the development of proof techniques, most of the proofs presented can be found elsewhere. For a general overview of LMI methods in control theory on should consult Boyd et al. [168]. Concerning the more specific topic of covariance based controller design, please see Skelton et al. [169] as well as Dullard and Paganini [119] both of which provide a foundation for the proofs given in Chmielewski and Manthanwar [170].
Exercises

10.1. Consider the continuous-time matrix pair:

\[
A = \begin{bmatrix}
1 & -3 & 0 \\
0 & -2 & 0 \\
2 & -2 & -1
\end{bmatrix}
\quad \text{and} \quad
B = \begin{bmatrix}
1 & 1 \\
1 & 0 \\
0 & 1
\end{bmatrix}
\]

(i) Using the MATLAB function ‘eig’ verify that \(A\) is not stable (indicating that we cannot trivially conclude that the pair \((A, B)\) is stabilizable).

(ii) Form the controllability matrix \(L_c = [B \ A \ A^2 B]\) and use the MATLAB function ‘orth’ to verify that the range space of \(L_c\) is not the full space and the pair \((A, B)\) is not completely controllable (again indicating that we cannot trivially conclude that the pair \((A, B)\) is stabilizable).

(iii) Use Theorem 10.2 to show that the pair \((A, B)\) is in fact stabilizable.

(iv) Repeat part (ii) with

\[
B = \begin{bmatrix}
1 & 0 \\
1 & 0 \\
0 & 1
\end{bmatrix}
\]

and use Theorem 10.2 to show that this new pair \((A, B)\) is in fact not stabilizable.

10.2. Consider the continuous-time matrix pair:

\[
A = \begin{bmatrix}
1 & -3 & 0 \\
0 & -2 & 0 \\
2 & -2 & -1
\end{bmatrix}
\quad \text{and} \quad
B = \begin{bmatrix}
1 & 1 \\
1 & 0 \\
0 & 1
\end{bmatrix}
\]

(i) Discretize the matrix pair using sample and hold method (with \(\Delta t = 0.1\)) to find \(A_d\) and \(B_d\).

(ii) Using the MATLAB function ‘eig’ verify that \(A_d\) is not stable (indicating that we cannot trivially conclude that the pair \((A_d, B_d)\) is stabilizable).

(iii) Form the controllability matrix \(L_c = [B_d \ A_d B_d \ A_d^2 B_d]\) and use the MATLAB function ‘orth’ to verify that the range space of \(L_c\) is not the full space and the pair \((A_d, B_d)\) is not completely controllable (again indicating that we cannot trivially conclude that the pair \((A_d, B_d)\) is stabilizable).

(iv) Use Theorem 10.4 to show that the pair \((A_d, B_d)\) is in fact stabilizable.

(v) Repeat part (iii) with

\[
B_d = \begin{bmatrix}
0.0906 & 0 \\
0.0906 & 0 \\
0 & 0.0951
\end{bmatrix}
\]

and use Theorem 10.4 to show that this new pair \((A_d, B_d)\) is in not stabilizable.

10.3. Consider the linear system

\[
\dot{x} = \begin{bmatrix}
1 & -2 & 0 \\
-3 & -1 & 2 \\
0 & 0 & -2
\end{bmatrix} x + \begin{bmatrix}
1 & 0 \\
0 & 1 \\
0 & 0
\end{bmatrix} u
\]

Is the system (1) Stable? (2) Controllable? (3) Stabilizable?
10.4. You have been assigned the task of creating state observers for the following systems. The person previously in your position seems to remember that for one of these systems, a state observer could not be implemented. Is she correct?

(i) \[ \dot{x} = \begin{bmatrix} -3 & 1 \\ 0 & -3 \end{bmatrix} x \quad y = [0 \ 1] x \]

(ii) \[ x_{k+1} = \begin{bmatrix} 1.5 & 1 \\ 0 & 0.5 \end{bmatrix} x_k \quad y_k = [1 \ 0] x_k \]

(iii) \[ \dot{x} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} x \quad y = [1 \ 0] x \]

10.5. Consider the linear system \( \dot{x} = Ax + Bu \), with

\[ A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 & -3 & -1 \end{bmatrix} \]

You are given a choice of two actuators. Each result in the following \( B \) matrices

\[ B_1 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \quad B_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \]

Which of these would you choose? Why?

10.6. Using Theorems 10.2 through 10.5, determine if the following systems are stabilizable and/or detectable?

(i) \[ A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = [0 \ 1] \]

(ii) \[ A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad C = [1 \ 0] \]

(iii) \[ A = \begin{bmatrix} -1 & 0 \\ 1 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad C = [1 \ 0] \]

(iv) \[ A = \begin{bmatrix} 1 & -2 & 0 \\ -3 & -1 & 2 \\ 0 & 0 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad C = [1 \ 0 \ 0] \]

(v) \[ A = \begin{bmatrix} 1 & -2 & 0 \\ -3 & -1 & 2 \\ 0 & 0 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad C = [1 \ 0 \ 0] \]

(vi) \[ A_d = \begin{bmatrix} 1.1 & -0.2 & 0 \\ -0.3 & 0.9 & 0.2 \\ 0 & 0 & 0.8 \end{bmatrix} \quad B_d = \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix} \quad C = [1 \ 0 \ 0] \]

10.7. Prove Theorem 10.3.

10.8. Prove Theorem 10.5.


10.10. Reproduce the results of Example 10.4.

10.11. Convert the system of Example 10.4 to discrete-time using a sample time of 0.2. Reproduce the results of Example 10.4, but using the discrete-time framework. The \( \zeta (j)_{\max} \) values of cases 1 through 5 should give the same results and generate a plot almost identical to that of Figure 10.2.
10.12. Consider the following continuous-time model of a mass-spring-damper augmented with a shaping filter:

\[
A = \begin{bmatrix}
0 & 1 & 0 \\
-3 & -2 & 1 \\
0 & 0 & -0.25
\end{bmatrix} \quad B = \begin{bmatrix}
0 \\
1 \\
0
\end{bmatrix} \quad G = \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix} \quad S_w = 12.5
\]

\[
D_x = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
-3 & -2 & 1
\end{bmatrix} \quad D_u = \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix}
\]

Determine which of the following cases will yield a feasible FSI stochastic controller. If the case is feasible, determine a controller capable achieving the variance bounds.

(i) \(\zeta_{max} = \begin{bmatrix}
0.1^2 & 0.04^2 & 5^2 & 0.05^2
\end{bmatrix}\)

(ii) \(\zeta_{max} = \begin{bmatrix}
0.6^2 & 0.2^2 & 3.5^2 & 0.3^2
\end{bmatrix}\)

(iii) \(\zeta_{max} = \begin{bmatrix}
1.5^2 & 0.7^2 & 1^2 & 1.2^2
\end{bmatrix}\)

10.13. Convert the system of Exercise 10.12 to discrete-time using a sample time of 0.01. Verify that the three case of Exercise 10.12 hold within the discrete-time framework. If the sample time is increased sufficiently, then all three cases will be infeasible. Determine the smallest sample time such that all three cases are infeasible.

10.14. Reproduce the results of Example 10.5.

10.15. Convert the system of Exercise 10.14 to discrete-time using a sample time of 0.0001. Verify that the five case of Exercise 10.14 hold with the discrete-time framework. Show that if the sample time is 0.001, then only case 1 will be feasible and all others will be infeasible.

10.16. Convert the results of Example 10.6 to a discrete-time framework, using a sample-time of 2.0. That is, find the largest value of \(\Sigma_v\) that will yield a feasible estimation problem, with \(\zeta_{e(1),max} = 0.1\) and \(\zeta_{e(2),max} = 0.5\). Since the discrete-time estimation problem has two forms, there should be two solutions — one for the one-step predictor and a second for the optimal estimator. Intuition suggests that the one-step predictor will require a smaller measurement noise than the optimal estimator to achieve the same level of performance. In both cases, use the MATLAB function ‘dare’ to verify that the obtained largest value of \(\Sigma_v\) satisfies the performance criteria.
In this chapter a new perspective on controller synthesis is presented. The Minimum Variance (MV) and Constrained Minimum Variance (CMV) control problems are stated as direct extensions of the stochastic analysis of Chapter 5, but supported by the LMI results of Chapter 10. Then, it is shown that the controller generated by the CMV problem is equal to that generated by some Linear Quadratic Optimal Control (LQOC) problem. Since LQOC is the basis of the Model Predictive Control (MPC) policy, it will be shown that the CMV controller can be used for the tuning of MPC controllers.

### 11.1 Covariance Based Controller Design

Consider the continuous-time process

\[
\dot{x} = Ax + Bu + Gw
\]
\[
z = Dx + Du u
\]

Then, assuming the use of a full state information linear feedback, \( u = -Lx \), the covariance equations are (recall that \( \rho_j \) is the \( j \)-th row of an appropriately dimensioned identity matrix):

\[
0 = (A - BL)\Sigma_x + \Sigma_x (A - BL)^* + GS_w G^*
\]
\[
\Sigma_z = (D_x - D_u L)\Sigma_x (D_x - D_u L)^*
\]
\[
\zeta^{(i)} = \rho_i \Sigma_x \rho_i^*, \quad i = 1 \ldots n_z
\]

From a controller design perspective (the selection of \( L \)), the system of equations (11.3) - (11.5) can be of great utility. Specifically, one can evaluate the variance of each performance output, \( \zeta^{(i)} \), as a function of the choice of \( L \).

**Example 11.1.** Consider a scalar process \( \dot{x} = ax + bu + gw \), and assume \( w \) is zero-mean white noise with \( S_w = 16 \). Furthermore, let \( a = -1 \), \( b = 1 \) and \( g = 1 \) and define the performance output, \( z \), as the state and the manipulated variable: \( D_x = [1 0]^* \) and \( D_u = [0 1]^* \). If the controller is \( u = -Lx \), then by using the scalar version of (11.3),

\[
0 = 2\Sigma_x (a - bL) + g^2 S_w, \quad \text{we find} \quad \Sigma_x = 8/(L + 1).
\]

In addition, the variance of the two performance outputs are calculated as \( \zeta^{(1)} = 8/(L + 1) \) and \( \zeta^{(2)} = 8L^2/(L + 1) \). Plots of \( \zeta^{(1)} \) and \( \zeta^{(2)} \) as a function of \( L \) (Figure 11.1, left) indicate a general trend that will be seen throughout the chapter. Specifically, efforts to decrease the variance of the state can be
achieved only by increasing the variance of the manipulated variable. This can be seen more clearly in the parametric plot of $\zeta^{(1)}$ vs. $\zeta^{(2)}$ (Figure 11.1, right).

![Figure 11.1. Plots of output variance for Example 11.1](image1)

The scalar notions of Example 11.1 extend quite naturally to the multidimensional case. Specifically, if we investigate the positive quadrant of the $\zeta$ space and ask if there exists a linear controller that will generate this set of variances, a Pareto frontier can be constructed as the boundary between controller existence and non-existence. A Pareto frontier is a curve in the space of two or more objectives that defines the points where one objective cannot be improved without worsening at least one of the others. Figure 11.2 shows this for the 2-dimensional case. In the general case, the frontier will be a hypersurface in an $n_x$-dimensional space. In these cases, multiple plots are required to depict the hyper-surface.

![Figure 11.2. The Pareto frontier](image2)
Example 11.2. Consider a system of the form (11.1)-(11.2) with

\[
A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad G = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad D_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad D_u = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

and assume \(w\) is zero-mean white noise with \(\Sigma_w = 4\). If the controller is \(u = -[l_1 \ l_2]x\), then evaluation of (11.3)-(11.4) yields

\[
\Sigma_x = \begin{bmatrix} \frac{1}{l_1 l_2} & 0 \\ 0 & \frac{1}{l_2} \end{bmatrix} \quad \Sigma_z = \begin{bmatrix} \frac{1}{l_1 l_2} & 0 & \frac{1}{l_2} \\ 0 & \frac{1}{l_2} & 1 \\ \frac{1}{l_2} & 1 & \frac{1}{l_2} + l_2 \end{bmatrix}
\]

Thus, \(\zeta^{(1)} = 1/(l_1 l_2)\), \(\zeta^{(2)} = 1/l_2\) and \(\zeta^{(3)} = l_1/l_2 + l_2\). An evaluation of these expressions for a few thousand combinations \(l_1\) and \(l_2\) results in the plots of Figure 11.3. Clearly, a region of achievable variance triples is beginning to emerge from these plots. Of course, the point \((\zeta^{(1)}, \zeta^{(2)}, \zeta^{(3)}) = (0, 0, 0)\) is not achievable. Similarly, the point \((0.1, 0.2, 2)\) cannot be achieved, while \((0.2, 0.4, 5)\) is clearly possible.

Figure 11.3. Plots of output variance for Example 11.2

11.1.1 Minimum Variance Control

To characterize the Pareto frontier, one can define an optimization problem with an objective function constructed as the weighted sum of the \(\zeta^{(j)}\)s. To this end, the Minimum Variance (MV) control problem is defined as:

\[
\min_{\zeta^{(j)}, \Sigma_x, \Sigma_z, L} \left\{ \sum_{j=1}^{n_z} d_j \zeta^{(j)} \right\}
\]

s.t.

\[
(A - BL)\Sigma_x + \Sigma_x (A - BL)^* + GS_w G^* = 0 \tag{11.7}
\]

\[
\Sigma_x = (D_x - D_u L)\Sigma_x (D_x - D_u L)^* \tag{11.8}
\]

\[
\zeta^{(j)} = \rho_j \Sigma_x \rho_j^*, \quad j = 1 \ldots n_z \tag{11.9}
\]
where $d_j$ are weights intended indicate the relative importance of each output signal. It is easily observed that the solution to problem (11.6) will lie on the Pareto frontier and that the specific point of that solution is a function of the weights $d_j$. In addition, all controllers generated by (11.6) are efficient in the sense that variance reduction in one direction can only be achieved by increasing the variance in another.

![Figure 11.4](image)

**Figure 11.4.** Plots of output variance for Example 11.3

**Example 11.3.** Recall the system of Example 11.2, then the MV problem is defined as

$$\min_{\zeta^{(1)}, \zeta^{(2)}, \zeta^{(3)}, d_1, d_2} \left\{ d_1 \zeta^{(1)} + d_2 \zeta^{(2)} + d_3 \zeta^{(3)} \right\} \tag{11.10}$$

subject to

$$\zeta^{(1)} = 1/(l_1 l_2)$$

$$\zeta^{(3)} = l_1/(l_1 l_2)$$

If $d_1 = d_2 = 1$ and $d_3$ is set to a variety of values, then the Pareto frontier of Figure 11.4 can be constructed. The “o” points are the solutions to the MV control problem, with the dashed line simply connecting the solutions.

It is important to note that the (11.10) form of the MV control problem is not particularly amenable to a numeric solution. This is because the constraints of Problem (11.10) are rather nonlinear and thus there is no guarantee that a generic nonlinear solver will find the global minimum. The method used to solve the optimization problem of Example 11.3 (and generate the plots of Figure 11.4) is to recast the MV control problem in to the form of a Convex Program (CP). Specifically, Theorem 10.8 is used to arrive at the following equivalent form of the MV problem (i.e., equivalent to Problem (11.6)).

$$\min_{\mu^{(j)}, Z_0, Z_1} \sum_{j=1}^{n_x} d_j \mu^{(j)} \tag{11.11}$$

subject to

$$(AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0 \tag{11.12}$$

$$\begin{bmatrix} \mu^{(j)} & \rho_j (D_x Z_0 - D_u Z_1) \\ (D_x Z_0 - D_u Z_1)^* \rho_j^* & Z_0 \end{bmatrix} > 0, \quad j = 1 \ldots n_x$$ \tag{11.13}

Since the objective function and all constraints of Problem (11.11) are either linear or convex, we conclude that Problem (11.11) is a CP, and as such its numeric solution is...
guaranteed to be a global solution. Given the form of Problem (11.11), one may use a variety of Semi-Definite Programming solvers to obtain the solution to the MV control problem. The code used for Example 11.3 is given in Table 11.1.

Based on the proof of Theorem 10.8, we know that if the solution to Problem (11.11) is $Z_0^*$ and $Z_1^*$, then the linear feedback portion of the solution to (11.6) can be constructed as $L = Z_1^*(Z_0^*)^{-1}$. In addition, the proof of Theorem 10.8 indicates a relation between $\Sigma_x$ and $Z_0^*$, specifically $\Sigma_x \leq Z_0^*$. As such, $Z_0^*$ provides an indication of the value of $\Sigma_x$, but there is no guaranteed that the two will be equal. Thus, to calculate $\Sigma_x$ the best option is to use the Lyapunov Equation with the linear feedback calculated as $L = Z_1^*(Z_0^*)^{-1}$.

**Table 11.1:** *MATLAB/YALMIP code used in calculations for Example 11.3.*

```matlab
clear
nx=2; nu=1; nw=1;
nz=3; Izz=eye(nz);
A=[0 1;0 0]; B=[0;1];
G=[0;1]; Sw=2;
Dx=[1 0;0 0]; Du=[0;0;1];
% Solve MV control problem with different d3 weights
for kk=1:20
    d3=2^(kk-10);
dweight=[1 1 d3];
yalmip('clear');
C1=[]; C2=[];
mu=sdpvar(nz,1);
Z0=sdpvar(nx);
Z1=sdpvar(nu,nx);
C1=[(A*Z0-B*Z1)+(A*Z0-B*Z1)
    +G*Sw*G
    <=0];
for j=1:nz,
    rho_j=Izz(j,:);
    C2=[C2, [mu(j) rho_j*(Dx*Z0-Du*Z1);
    (Dx*Z0-Du*Z1)'*rho_j
    Z0]]>=0];
end
Constraints=[C1, C2];
Objective=dweight*mu;
options = sdpsettings('verbose',0,'solver','mosek');
sol = optimize(Constraints,Objective,options);
if sol.problem == 0
    Z0star=value(Z0);
    Z1star=value(Z1);
    LListar=Z1star/Z0star;
    % Determine the FSI Controller Covariances
    SigXc1=lyap(A-B*LListar,G*Sw*G');
    SigmaZ=(Dx-Du*LListar)*SigXc1*(Dx-Du*LListar)';
    ss1(kk)=SigmaZ(1,1);
    ss3(kk)=SigmaZ(2,2);
    su(kk)=SigmaZ(3,3);
else
    display('Hmm, something went wrong!'); sol.info, pause
end

figure(1), plot(ss1,ssu,'-or')
axis([-0.0 1 0 20])
figure(2), plot(ss3,ssu,'-or')
axis([-0.0 1 0 20])
```
Example 11.4. Let the matrices $A$, $B$, $G$, $D_x$ and $D_u$ be defined as follows:

$$
A = \begin{bmatrix}
0 & 1 & 0 \\
-3 & -2 & 1 \\
0 & 0 & -0.25
\end{bmatrix}
\quad B = \begin{bmatrix}
0 \\
1 \\
0
\end{bmatrix}
\quad G = \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix}
$$

(11.14)

$$
D_x = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
-3 & -2 & 1
\end{bmatrix}
\quad D_u = \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix}
$$

(11.15)

Assume $w$ is zero mean white noise with $S_w = 12.5 N^2 s$. Solutions to the MV control problem (11.11) for various objective function weights are given in Table 11.2. Recall that $\sigma_z^{(j)} = (\zeta^{(j)})^{1/2}$. As $d_3$ is increased from 0.01 to 1, there is greater interest in making $\sigma_z^{(3)}$ small. To arrive at these smaller values of $\sigma_z^{(3)}$, all the others ($\sigma_z^{(1)}$, $\sigma_z^{(2)}$ and $\sigma_z^{(4)}$) must increase.

Table 11.2. Solutions to the MV control problem for Example 11.4.

<table>
<thead>
<tr>
<th>Case</th>
<th>$\sigma_z^{(1)}$</th>
<th>$\sigma_z^{(2)}$</th>
<th>$\sigma_z^{(3)}$</th>
<th>$\sigma_z^{(4)}$</th>
<th>Optimal Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1 = d_2 = d_4 = 1 \quad d_3 = 0.01$</td>
<td>0.096</td>
<td>0.035</td>
<td>4.74</td>
<td>0.043</td>
<td>$L = [-1.96 \quad -0.250 \quad 0.977]$</td>
</tr>
<tr>
<td>$d_1 = d_2 = d_4 = 1 \quad d_3 = 0.1$</td>
<td>0.606</td>
<td>0.241</td>
<td>3.34</td>
<td>0.320</td>
<td>$L = [-1.69 \quad -0.168 \quad 0.824]$</td>
</tr>
<tr>
<td>$d_1 = d_2 = d_4 = 1 \quad d_3 = 1$</td>
<td>1.35</td>
<td>0.674</td>
<td>1.09</td>
<td>1.12</td>
<td>$L = [-0.764 \quad -0.007 \quad 0.363]$</td>
</tr>
</tbody>
</table>

Figure 11.5. Illustration of the CMV control problem
11.1. Covariance Based Controller Design

11.1.2 Constrained MV Control

In many cases, one would like to impose bounds on the variance of each output: \( \zeta^{(j)} < \zeta^{(j)}_{\text{max}} \). The result is the following Constrained Minimum Variance (CMV) control problem:

\[
\min \left\{ \sum_{j=1}^{n_z} d_j \zeta^{(j)} \right\} \quad \text{s.t.} \quad (A - BL)\Sigma_x + \Sigma_x(A - BL)^* + GS_w G^* = 0 \\
\Sigma_x = (D_x - D_u L)\Sigma_x(D_x - D_u L)^* + D_w S_w D_w^* \\
\zeta^{(j)} = \rho_j \zeta^{(j)}_\rho < \zeta^{(j)}_{\text{max}}, \quad j = 1 \ldots n_z 
\]

In the context of Pareto efficiency, the CMV control problem is visualized in Figure 11.5, where the feasible region is now a pie piece (above the frontier and less than the dashed lines) and again the solution will be found on the frontier. The important concept is that the imposition of variance bounds will override the intent of the objective function weights, \( d_j \). In fact, we will show later in the chapter that these bounds can be thought of as modifying the weights so that a new unconstrained problem will have the same solution. For the control engineer, the constrained formulation with its reduced focus on weights is a welcome change. Specifically, it is fairly easy to specify variance targets for most output signals, but the selection of objective function weights is in most cases a trial-and-error exercise. This is because objective function weights reflect the relative importance of each variable with respect to the other variables. So, a change in the weight of one variable will have impacts on all the other variables.

Similar to the MV problem, the CMV control problem can be recast as a CP. Specifically, Theorem 10.8 can be used again to arrive at the following equivalent form of the CMV control problem.

\[
\min \left\{ \sum_{j=1}^{n_z} d_j \mu^{(j)} \right\} \quad \text{s.t.} \quad (AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0 \\
\begin{bmatrix}
\mu^{(j)} \\
\mu^{(j)}_0
\end{bmatrix} = \begin{bmatrix}
(D_x Z_0 - D_u Z_1) \rho_j \\
Z_0
\end{bmatrix} > 0, \quad j = 1 \ldots n_z \\
\mu^{(j)} < \zeta^{(j)}_{\text{max}}, \quad j = 1 \ldots n_z
\]

**Table 10.3:** MATLAB/YALMIP code used in calculations for Example 11.5.

```matlab
clear
\%
Define System
nx=3; nu=1; nw=1; nz=4; Izz=eye(nz);
AA=[0 1 0; -3 -2 1;0 0 -.25]; BB=[0; 1; 0]; GG=[0; 0; 1];
Dx=[1 0 0;0 1 0;0 0 0; -3 -2 1]; Du=[0;0;1;1]; Sw=12.5;
dweight=[1 1 0.01 1];
sigzmax=[10 10 5 10];
alpha=1;
\%
Define and Solve CMV Optimization
yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[];
mu=sdpvar(nz,1); Z0=sdpvar(nx); Z1=sdpvar(nu,nx);
```
C1=[mu>=0]; C2=[Z0>=0];
for ii=1:nz,
    C3=[C3, alpha^2 *mu(ii)<=sigzmax(ii)^2];
end
C4=[(AA*Z0-BB*Z1)+(AA*Z0-BB*Z1)'+GG*Sw*GG']<=0; 
for ii=1:nz, rho_i=Izz(ii,:);
    C5=[C5, [mu(ii) rho_i*(Dx*Z0-Du*Z1);
         (rho_i*(Dx*Z0-Du*Z1))' Z0]>=0];
end
Constraints=[C1, C2, C3, C4, C5];
Objective=dweight*zeta;
options = sdpsettings('verbose',0,'solver','mosek');
sol = optimize(Constraints,Objective,options);
if sol.problem == 0
    Z0star=value(Z0); Z1star=value(Z1); LLstar=Z1star *inv(Z0star)
else
    display('Hmm, something went wrong!'); sol.info, pause
end
% Determine the FSI Controller Covariances
SigXcl=lyap(AA-BB*LLstar,GG*Sw*GG'); SigmaZ=(Dx-Du*LLstar)*SigXcl*(Dx-Du*LLstar)';
sqrt(diag(SigmaZ))

Table 11.4. Solutions to the CMV problem for Example 11.5.

<table>
<thead>
<tr>
<th>Case</th>
<th>$\sigma_z^{(1)}$</th>
<th>$\sigma_z^{(2)}$</th>
<th>$\sigma_z^{(3)}$</th>
<th>$\sigma_z^{(4)}$</th>
<th>Optimal Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.097</td>
<td>0.035</td>
<td>4.74</td>
<td>0.043</td>
<td>$L = [-1.96, -0.254, 0.977]$</td>
</tr>
<tr>
<td></td>
<td>0.584</td>
<td>0.231</td>
<td>3.4</td>
<td>0.306</td>
<td>$L = [-1.70, -0.172, 0.832]$</td>
</tr>
<tr>
<td></td>
<td>1.38</td>
<td>0.696</td>
<td>1.00</td>
<td>1.16</td>
<td>$L = [-0.706, -0.003, 0.335]$</td>
</tr>
</tbody>
</table>

Example 11.5. Reconsider the scenario of Example 11.4, with $d_1 = d_2 = d_4 = 1$ and $d_3 = 0.01$. If $\zeta^{(1),\max} = \zeta^{(2),\max} = \zeta^{(4),\max} = 10^2$, then for various values of $\sigma_z^{(3),\max} = \sqrt{\zeta^{(3),\max}}$, the solution to the CMV problem (11.20) is presented in Table 11.4. Once again Pareto efficiency is clearly observed, in that decreases in $\sigma_z^{(3)}$ can only be achieved if the others ($\sigma_z^{(1)}$, $\sigma_z^{(2)}$ and $\sigma_z^{(4)}$) increase. It is also noted that the solutions of Table 11.4 are approximately equal to those of Table 11.2. This suggests that $\sigma_z^{(3),\max} = 3.4$ is almost the same as setting $d_3 = 0.1$ and that $\sigma_z^{(3),\max} = 1$ is almost the same as setting $d_3 = 1$.

11.1.3 • EDOR Interpretation of CMV Control

While the Pareto interpretation is appealing with respect to optimization, from a process operations perspective a visualization of the Expected Dynamic Operating Region (EDOR) is more intuitive. Consider the multidimensional elliptical region discussed in Section 5.3.5: the set of vectors, $z$, such that $z^*\Sigma^{-1}z < 1$. This region indicates where one would expect to find the trajectory of $z(t)$, assuming the covariance matrix of $z(t)$ is $\Sigma_z$. As indicated in Figure 11.6, $\zeta^{(j)} < \zeta^{(j),\max}$ defines a multidimensional constraint box, centered at the origin and with sides of length $2\sigma_z^{(j),\max}$. The interpretation of Problem (11.20) is that a feasible controller is such that the resulting EDOR is contained in the
11.1. Covariance Based Controller Design

Figure 11.6. Feasible controllers for CMV problem (solid EDOR is infeasible, dashed is feasible).

Constraint box. One can think of the EDOR as a multidimensional balloon that can be squeezed in one direction, but due to its Pareto efficiency must also expand in some other direction. However, this balloon squeezing can only occur if there exists a controller $L$ such that Equations (11.17)-(11.18) are satisfied, or equivalently if there exists $Z_0$ and $Z_1$ such that Equations (11.21)-(11.22) are satisfied.

Figure 11.7. EDOR Visualizations for Example 11.6

Example 11.6. Continuing Example 11.5, the EDOR visualizations resulting from the controllers of Table 11.4 are given in Figure 11.7. As the constraint on the manipulated input force, $\sigma_z^{(3)\text{max}}$, decreases from 5, to 3.4 to 1, the EDOR expands in all other directions.

In Chapter 5 the EDOR was actually defined as the set of vectors, $z$, such that $z^*\Sigma_z^{-1}z < \alpha^2$, where $\alpha$ indicated the number of standard deviations associated with the EDOR. Thus, if $\alpha \neq 1$, then constraint 11.23 of the CMV problem should be changed to $\alpha^2 \mu^{(j)} < \zeta^{(j)\text{max}}, \ j = 1...n_z$. For example, if $\alpha = 2$, then the EDOR will be defined to be larger and the CMV will need to make appropriate changes to fit this larger EDOR within the constraint box. Said another way, the original EDOR (with $\alpha = 1$)
will need to be smaller to fit into a smaller constraint box. The smaller constraint box being exemplified by rewriting the constraint as $\mu^{(j)} < \zeta^{(j),\text{max}} / \alpha^2$, $j = 1 \ldots n_z$. If $\zeta^{(j),\text{max}}$ is replaced by $(\sigma^{(j),\text{max}}_z / \alpha)^2$, then the constraints take the more intuitive form of $\mu^{(j)} < (\sigma^{(j),\text{max}}_z / \alpha)^2$, $j = 1 \ldots n_z$. In this form we see clearly, that an increase in $\alpha$, decreases the constraint box that the originally defined EDOR (with $\alpha = 1$) must be fit into to be feasible.

Example 11.7. Continuing Example 11.6, consider the case of $\sigma^{(3),\text{max}}_z = 1$ and solve the CMV problem with $\alpha = 2$. Figure 11.8 depicts the solution, where the solid ellipse is the EDOR defined with $\alpha = 2$ and the dashed ellipse is defined with $\alpha = 1$. Given these definitions, the left plot shows that the standard deviation in the manipulated variable direction is equal to 0.5, or more directly from the plot that the two standard deviations in the manipulated variable direction is equal to $\sigma^{(3),\text{max}}_z = 1$. The right plot shows the same, but more directly (i.e., one standard deviation is equal to $\sigma^{(3),\text{max}}_z / \alpha = 0.5$). Recall that a Gaussian process will be found within two standard deviations of the mean 95% of the time, whereas it will be found within one standard deviation of the mean only 68% of the time. Thus, the $\alpha = 2$ EDOR definition (the solid ellipse) is likely a better representation of actual plant operation. As such, the right plot of Figure 11.8 summarizes the situation as follows: The dashed (one standard deviation) ellipse is found by the CMV problem with reduced constraints (i.e., $\zeta^{(j),\text{max}}_z = (\sigma^{(j),\text{max}}_z / \alpha)^2$). If this linear feedback is applied to the process, then the two standard deviation (solid) ellipse is a better representation of process operation and will satisfy the original constraints, $\sigma^{(j),\text{max}}_z$.

![Figure 11.8. EDOR Visualizations for Example 11.7](image)

In some cases, one may wish to define a different $\alpha$ for each direction. For example, the constraints could be defined as $\mu^{(j)} < (\sigma^{(j),\text{max}}_z / \alpha_j)^2$, $j = 1 \ldots n_z$. However, since the EDOR can only be defined with respect to one $\alpha$, the right plot interpretation is most appropriate. Specifically, the CMV puts the one standard deviation EDOR into the reduced constraint box (i.e., with bounds $(\sigma^{(j),\text{max}}_z / \alpha_j)^2$). If $\alpha_j < 2$, then the two standard deviation EDOR will violate the $\sigma^{(j),\text{max}}_z$ constraint in the $j$ direction. And, if $\alpha_j \geq 2$, the two standard deviation EDOR will satisfy the $\sigma^{(j),\text{max}}_z$ constraint. Of course, all the above can be addressed using the original constraint $\mu^{(j)} < \zeta^{(j),\text{max}}$, $j = 1 \ldots n_z$, by appropriately modifying each parameter $\zeta^{(j),\text{max}}$. Thus, in the theoretical results to follow the original constraint will be retained. Then, in the more practical framework of the examples, the $\alpha_j$ formulation will be reprised.
11.2 • CMV in Discrete-Time

In the discrete-time case, identical concepts exist, but with numerous changes to the details. Consider a discrete-time process:

\[ x_{k+1} = A_d x_k + B_d u_k + G_d w_k \]  
\[ z_k = D_x x_k + D_u u_k + D_w w_k \]  

(11.24, 11.25)

If the controller is assumed to be \( u_k = -L x_k \), then application of the discrete-time covariance relations of Chapter 5 results in the following CMV problem:

\[
\min_{\zeta^{(i)}, \Sigma_x, \Sigma_z \geq 0, L} \left\{ \sum_{j=1}^{n_z} d_j \zeta^{(j)} \right\} \quad \text{s.t.} \ 
\Sigma_x = (A_d - B_d L) \Sigma_x (A_d - B_d L)^* + G_d \Sigma_w G_d^* \\
\Sigma_z = (D_x - D_u L) \Sigma_x (D_x - D_u L)^* + D_w \Sigma_w D_w^* \\
\zeta^{(j)} = \rho_j \Sigma_z \rho_j^* < \zeta^{(j)}_{\max}, \ j = 1 \ldots n_z
\]

(11.26)

Convexification is achieved by application of Theorem 10.9, and results in the following convex form of the discrete-time CMV problem, from which a global solution is readily obtained:

\[
\min_{\mu^{(i)}, Z_0 \geq 0, Z_1} \left\{ \sum_{j=1}^{n_z} d_j \mu^{(j)} \right\} \quad \text{s.t.} \ 
\begin{bmatrix}
Z_0 - G_d \Sigma_w G_d^* & (A_d Z_0 - B_d Z_1)^* \\
(A_d Z_0 - B_d Z_1) & Z_0
\end{bmatrix} > 0 \\
\begin{bmatrix}
\mu^{(j)} - \rho_j D_w \Sigma_w D_w^* \rho_j^* \\
(D_x Z_0 - D_u Z_1) \rho_j^* & Z_0
\end{bmatrix} > 0, \ j = 1 \ldots n_z
\]

and \( \mu^{(j)} < \zeta^{(j)}_{\max}, \ j = 1 \ldots n_z \)

(11.30, 11.31, 11.32, 11.33)

From the proof of Theorem 10.9, we know that if \( Z_0^*, Z_1^* \) is the solution to (11.30), then the solution to (11.26) is reconstructed from \( L = Z_1^* (Z_0^*)^{-1} \) and the relations (11.27)-(11.28).

11.2.1 • CMV with Computational Delay

In the computational delay case, recall that the computations required to determine the input at time \( k, u_k \), must start at time \( k - 1 \). Thus, the feedback information available at the beginning of that calculation is the state at time \( k - 1 \) and not the state at time \( k \). In other words, \( u_k \) is a function of \( x_{k-1} \) and not \( x_k \). However, given \( x_{k-1} \), one can always predict the state at time \( k \), by assuming \( w_{k-1} = 0 \):

\[ \dot{x}_k = A_d x_{k-1} + B_d u_{k-1} \]  

(11.34)

Then, the appropriate feedback policy is \( u_k = -L \dot{x}_k \). Using Equation (11.24) one finds that \( x_k = \dot{x}_k + G_d w_{k-1} \) and Equation (11.34) can be written as:

\[
\dot{x}_{k+1} = A_d x_k + B_d u_k \\
= A_d \dot{x}_k + A_d G_d w_{k-1} - B_d L \dot{x}_k \\
= (A_d - B_d L) \dot{x}_k + A_d G_d w_{k-1}
\]

(11.35)
Similarly, the performance output can be written as 
\[ z_k = (D_x - D_u L) \hat{x}_k + D_w \bar{w}_{k-1} + D_w w_k. \]

Thus, the CMV problem with one-step computational delay is defined as:

\[
\min_{\zeta^{(i)}, \Sigma_z, \Sigma_d, L} \left\{ \sum_{j=1}^{n_z} d_j \zeta^{(j)} \right\} \quad \text{s.t.} \quad (11.36)
\]

\[
\Sigma_z = (A_d - B_d L) \Sigma_z (A_d - B_d L)^* + A_d G_d \Sigma_w \Sigma_d A_d^* \quad (11.37)
\]

\[
\Sigma_d = (D_x - D_u L) \Sigma_d (D_x - D_u L)^* + D_x G_d \Sigma_w \Sigma_d D_x^* + D_w \Sigma_w D_w^* \quad (11.38)
\]

\[
\zeta^{(j)} = \rho_j \Sigma_z \rho_j^* < \zeta^{(j), \text{max}}, \quad j = 1 \ldots n_z \quad (11.39)
\]

Then, re-application of Theorem 10.9, using the appropriate change of variable names, results in problem (11.36) being exactly equivalent to the following convex optimization problem:

\[
\min_{\mu^{(j)}, Z_0, Z_1} \left\{ \sum_{j=1}^{n_z} d_j \mu^{(j)} \right\} \quad \text{s.t.} \quad (11.40)
\]

\[
\begin{bmatrix}
Z_0 - A_d G_d \Sigma_w \Sigma_d A_d^* & (A_d Z_0 - B_d Z_1) \\
(A_d Z_0 - B_d Z_1)^* & Z_0
\end{bmatrix} > 0 \quad (11.41)
\]

\[
\begin{bmatrix}
\mu^{(j)} - \rho_j (D_x G_d \Sigma_w \Sigma_d D_x^* + D_w \Sigma_w D_w^*) \rho_j^* & \rho_j (D_x Z_0 - D_u Z_1) \\
(D_x Z_0 - D_u Z_1)^* \rho_j^* & Z_0
\end{bmatrix} > 0 \quad (11.42)
\]

\[
\mu^{(j)} < \zeta^{(j), \text{max}}, \quad j = 1 \ldots n_z \quad (11.43)
\]

Similar to the discrete-time case without delay, the proof of Theorem 9.9 tells us that if 
\( Z_0^*, Z_1^* \) is the solution to (11.40), then the solution to (11.36) is reconstructed from 
\( L = Z_1^* (Z_d^*)^{-1} \) and the relations (11.37)-(11.38).

Clearly, the CMV policy generated from (11.40) will be distinct from that of generated by problem (11.30). This is in stark contrast to the stochastic LQOC approach, where starting with the system of Equation (11.35), the stochastic LQOC would result in the same linear feedback as the case without computational delay, (11.24). This is because of the certainty equivalence result of Chapter 8, which tells us that the process disturbance term, either \( G_d \bar{w}_k \) or \( A_d G_d \bar{w}_{k-1} \), will play no role in the calculation of the stochastic LQOC gain. Thus, one feature of the CMV problem is that it does not possess certainty equivalence and the resulting policy will be tailored to the size and direction of the disturbance as well as to the particular feedback structure of the given situation.

### 11.3 CMV with Partial State Information

This section will consider the CMV problem under the assumption of measurement information, also known as the Partial State Information (PSI) case. In general, the results will parallel the previous two sections. We will begin with the continuous-time case, where the innovation process is used to achieve a separation of the estimation and control problems - a nearly identical development to that found in Chapter 8. In the discrete-time PSI cases, we will find results paralleling the computational delay cases of the previous section, but will also find a connection between the delay concept and that of the one-step predictor of Chapter 6.

Starting with the continuous-time case, the first step is to augment system (11.1)-(11.2) with a measurement equation: 
\( y = C x + v \). As usual, \( v \) is assumed to be zero.
mean, white noise with a spectral density \( S_v \). The PSI development of the CMV problem begins by assuming the controller to be a linear feedback of the of the optimal state estimate: \( u = -L\hat{x} \). Let us now recall a few facts from Chapter 6:

1. The state estimate, \( \hat{x} \), is the solution to the optimal estimator (or Kalman filter):
\[
\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x})
\]  
(11.45)

2. The optimal estimator gain in steady-state is \( K = \Sigma_e C^*S_v^{-1} \), where \( \Sigma_e \) is the positive definite solution to the estimation ARE:
\[
0 = A\Sigma_e + \Sigma_e A^* + GS_wG^* - \Sigma_e C^*S_v^{-1}C\Sigma_e
\]  
(11.44)

In addition, \( \Sigma_e \) is the covariance of the estimation error, defined as \( e = x - \hat{x} \).

3. The innovations process, defined as \( \eta = y - C\hat{x} \), is a zero mean, white noise process with spectral density \( S_\eta = S_v \).

4. The orthogonality principle indicates that \( E[e^*\hat{x}] = 0 \).

Thus, the closed-loop system can be stated as:
\[
\begin{align*}
\dot{\hat{x}} &= (A - BL)\hat{x} + K\eta \\
z &= (D_x - D_u L)\hat{x} + D_x e
\end{align*}
\]  
(11.46)

Applying standard covariance analysis, while utilizing the orthogonality principle, one finds the following steady-state covariance relations:
\[
0 = (A - BL)\Sigma_x + \Sigma_x (A - BL)^* + \Sigma_x C^*S_v^{-1}C\Sigma_e
\]  
(11.47)

\[
\Sigma_x = (D_x - D_u L)\Sigma_x (D_x - D_u L)^* + D_x \Sigma_e D_x^*
\]  
(11.48)

Thus, the continuous-time PSI CMV problem is stated as
\[
\begin{align*}
\min_{\zeta^{(i)}, \Sigma_x, \Sigma_e, L} & \left\{ \sum_{j=1}^{n_z} d_j \zeta^{(j)} \right\} \\
\text{s.t.} & (11.47), (11.48) \text{ and } \zeta^{(j)} = \rho_j \Sigma_x \rho_j^* < \zeta^{(j), \max}, \ j = 1 \ldots n_z
\end{align*}
\]  
(11.49)

Application of Theorem 10.8, using the appropriate change of variable names, results in problem (11.49) being exactly equivalent to the following convex optimization problem:
\[
\begin{align*}
\min_{\mu^{(j)}, \zeta^{(j)}, L_1} & \left\{ \sum_{j=1}^{n_z} d_j \mu^{(j)} \right\} \\
\text{s.t.} & (11.51) \text{ and } (11.52) \text{ and } \mu^{(j)} < \zeta^{(j), \max}, \ j = 1 \ldots n_z
\end{align*}
\]  
(11.53)

\[
(AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + \Sigma_x C^*S_v^{-1}C\Sigma_e < 0
\]  
(11.52)

\[
\left[ \begin{array}{c}
\mu^{(j)} - \rho_j D_x \Sigma_e D_x^* \rho_j^* \\
\rho_j (D_x Z_0 - D_u Z_1)
\end{array} \right] > 0
\]  
(11.53)

Similar to the full state information case, the proof of Theorem 10.8 indicates that if \( Z_0^*, Z_1^* \) is the solution to (11.51), then the solution to (11.49) is reconstructed from \( L = Z_1^*(Z_0^*)^{-1} \) and the relations (11.47)-(11.48).
Chapter 11. Constrained Minimum Variance Control

11.3.1 Discrete-time CMV with Partial State Information

In the discrete-time case, the steady-state Kalman filter, as stated in Section 6.3.2, is:

\[
\hat{x}_k = A_d \hat{x}_{k-1} + B_d u_{k-1} + K \eta_k
\]

\[
K = \Sigma^{+}_{e} C^{T} (C \Sigma^{+}_{e} C^{*} + \Sigma_{w})^{-1}
\]

\[
\Sigma^{+}_{e} = A_d \Sigma_{e} A^{*} + G_d \Sigma_{w} G^{*}_{d}
\]

\[
\Sigma_{e} = \Sigma^{+}_{e} - \Sigma^{+}_{e} C^{*} (C \Sigma^{+}_{e} C^{*} + \Sigma_{w})^{-1} C \Sigma^{+}_{e}
\]

where \( \eta_k = y_k - C (A \hat{x}_{k-1} + B_d u_{k-1}) \) is the innovations sequence, which is zero mean, white noise with covariance \( \Sigma_{\eta} = C \Sigma^{+}_{e} C^{*} + \Sigma_{w} \). Recall that \( \Sigma_{e} \) is the error covariance of the state estimator and \( \Sigma^{+}_{e} \) is that of the one-step predictor. If the controller is \( u_k = -L \hat{x}_k \), then the closed-loop estimator can be written as:

\[
\hat{x}_{k+1} = (A_d - B_d L) \hat{x}_k + K \eta_{k+1}
\]

\[
z_k = (D_x - D_u L) \hat{x}_k + D_x e_k + D_w w_k
\]

Using, again, the orthogonality principle \( E[e^{T}_k \hat{x}_k] = 0 \), one finds the following covariance relations:

\[
\Sigma_{\hat{x}} = (A_d - B_d L) \Sigma_{\hat{x}} (A_d - B_d L)^{*} + \Sigma^{+}_{e} C^{*} (C \Sigma^{+}_{e} C^{*} + \Sigma_{w})^{-1} C \Sigma^{+}_{e}
\]

\[
\Sigma_{x} = (D_x - D_u L) \Sigma_{\hat{x}} (D_x - D_u L)^{*} + D_x \Sigma_{e} D^{*} + D_w \Sigma_{w} D^{*}
\]

From the ARE of Equation (11.59) one finds that \( \Sigma^{+}_{e} C^{*} (C \Sigma^{+}_{e} C^{*} + \Sigma_{w})^{-1} C \Sigma^{+}_{e} = \Sigma^{+}_{e} - \Sigma_{e} \), and (11.62) can be replaced with:

\[
\Sigma_{\hat{x}} = (A_d - B_d L) \Sigma_{\hat{x}} (A_d - B_d L)^{*} + \Sigma^{+}_{e} - \Sigma_{e}
\]

Thus, the discrete-time PSI CMV problem is stated as:

\[
\min_{\zeta^{(j)}, \Sigma_{\hat{x}}, \Sigma_{x}, L} \left\{ \sum_{j=1}^{n_x} d_j \zeta^{(j)} \right\} \quad \text{s.t.}
\]

\[
(11.64), (11.63), \text{and } \zeta^{(j)} = \rho_j \Sigma_{\hat{x}} \rho_j^{*} < \zeta^{(j), \text{max}}, \quad j = 1 \ldots n_z
\]
Then, re-application of Theorem 10.9, using the appropriate change of variable names, results in problem (11.65) being exactly equivalent to the following CP:

$$
\min_{\mu^{(i)}, Z_0, \hat{z}_1} \left\{ \sum_{j=1}^{n_z} d_j \mu^{(j)} \right\} \quad \text{s.t.} \quad (11.67)
$$

$$
\begin{bmatrix}
Z_0 - (\Sigma^+_e - \Sigma_e) & (A_d Z_0 - B_d Z_1) \\
(A_d Z_0 - B_d Z_1)^T & Z_0
\end{bmatrix} > 0
$$

$$
\begin{bmatrix}
\mu^{(j)} - \rho_j (D_w \Sigma_w D_w^* + D_x \Sigma_x D_x^*) \rho_j & \rho_j (D_x Z_0 - D_u Z_1) \\
(D_x Z_0 - D_u Z_1)^T \rho_j & Z_0
\end{bmatrix} > 0
$$

and \( \mu^{(j)} < \zeta^{(j), \max}, \ j = 1 \ldots n_x \) \hspace{1cm} (11.69)

Similar to the full state information case, the proof of Theorem 10.9 indicates that if \( Z_0^*, \ Z_1^* \) is the solution to (11.67), then the solution to (11.65) is reconstructed from \( L = Z_1^* (Z_0^*)^{-1} \) and the relations (11.62)-(11.63).

It is again interesting to note that if \( C = I \) and \( \Sigma_v \to 0 \), then \( \Sigma_e \to 0 \) and \( \Sigma^+_e \to G_d \Sigma_w G_d^* \), and problem (11.67) will converge to its full state information counterpart - problem (11.30).

### 11.3.2 CMV with One-Step Delay and Partial State Information

In the computational delay case, the controller is also a feedback of the Kalman estimate, but this time in the one-step predictor form: \( u_k = -L \dot{x}_k^+ \). (The reader is encouraged to seek out the similarities between of the pairs \((x_k, \dot{x}_k)\) of Section 11.2 and \((\hat{x}_k, \dot{x}_k^+)\) of Chapter 6). Thus, the closed loop system of interest is the one-step predictor in innovations form:

$$
\dot{x}_k^+ = (A_d - B_d L) \dot{x}_k^+ + K^+ \eta_k
$$

$$
z_k = (D_x - D_u L) \dot{x}_k^+ + D_x \epsilon_k^+ + D_w w_k
$$

$$
K^+ = A_d \Sigma^+_e C^* (C \Sigma_e C^* + \Sigma_v)^{-1}
$$

As in the previous section, the innovation sequence is \( \eta_k = y_k - C \dot{x}_k^+ \), with covariance \( \Sigma_{\eta} = C \Sigma^+_e C^* + \Sigma_v \). Using, again, the orthogonality principle \( E[e_k^+ \dot{x}_k^+] = 0 \) along with the ARE, one finds the following covariance relations:

$$
\Sigma_{\dot{x}} = (A_d - B_d L) \Sigma_{\dot{x}} + (A_d - B_d L)^T \Sigma_{\dot{x}} A_d^* + A_d (\Sigma^+_e - \Sigma_e) A_d^*
$$

$$
\Sigma_x = (D_x - D_u L) \Sigma_{\dot{x}} + (D_x - D_u L)^T \Sigma_{\dot{x}} D_x^* + D_x \Sigma_x D_x^* + D_w \Sigma_w \Sigma_x D_w^*
$$

Thus, the CMV problem with PSI and computational delay is stated as:

$$
\min_{\xi^{(i)}, \Sigma_x, \Sigma_{\dot{x}}, L} \left\{ \sum_{j=1}^{n_z} d_j \zeta^{(j)} \right\} \quad \text{s.t.} \quad (11.74), (11.75), \text{ and } \zeta^{(j)} = \rho_j \Sigma_x \rho_j^* < \zeta^{(j), \max}, \ j = 1 \ldots n_x
$$
Then, re-application of Theorem 10.9, using the appropriate change of variable names, results in problem (11.76) being exactly equivalent to the following CP:

\[
\min_{\mu^{(j)}, Z_0, Z_1} \left\{ \sum_{j=1}^{n_z} d_j \mu^{(j)} \right\} \quad \text{s.t.} \quad \begin{bmatrix}
Z_0 - A_d (\Sigma_e^+ - \Sigma_e) A_d^* (A_d Z_0 - B_d Z_1) \\
(A_d Z_0 - B_d Z_1)^* Z_0
\end{bmatrix} > 0
\] (11.78)

\[
\begin{bmatrix}
\mu^{(j)} (D_x Z_0 - D_u Z_1) [\rho_j (D_x Z_0 - D_u Z_1)]^* Z_0 \\
(D_x Z_0 - D_u Z_1) [\rho_j (D_x Z_0 - D_u Z_1)]^* Z_0
\end{bmatrix} > 0
\] (11.79)

and \( \mu^{(j)} < \zeta^{(j), \max}, \quad j = 1 \ldots n_z \) (11.81)

If \( C = I \) and \( \Sigma_e \to 0 \), then \( \Sigma_e \to 0 \) and \( \Sigma_e^+ \to G_d \Sigma_u G_d^T \), then problem (11.78) will converge to its full state information counterpart - problem (11.40).

11.4 CMV and the LQOC Problem

While the CMV problem appears to be distinct from the LQOC, the stochastic basis of the CMV problem suggests a connection. However, before we explore this relationship, it will be useful to review the necessary condition for optimality associated with nonlinear duality theory.

11.4.1 Preliminary Results from Optimization

Consider an unconstrained optimization problem over a vector \( x = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}]^* \):

\[ \Phi = \min_x f(x) \]

The necessary conditions for optimality stem from the partial derivative set equal to zero:

\[ \frac{\partial f}{\partial x} = \left[ \frac{\partial f}{\partial x^{(1)}} \frac{\partial f}{\partial x^{(2)}} \cdots \frac{\partial f}{\partial x^{(n)}} \right]^* = 0 \]

Consider a similar optimization problem, but this time with equality constraints

\[ \Phi = \min_x f(x) \quad \text{s.t.} \quad F(x) = 0 \]

The dual of this problem is an unconstrained max min problem:

\[ \Gamma = \max_{\lambda_{eq}} \left\{ \min_x \left\{ f(x) + \lambda_{eq}^* F(x) \right\} \right\} \]

In this case, the necessary conditions for optimality become:

\[ \frac{\partial (f + \lambda_{eq}^* F)}{\partial x} = \frac{\partial f}{\partial x} + \lambda_{eq}^* \frac{\partial F}{\partial x} = 0 \]

\[ \frac{\partial (f + \lambda_{eq}^* F)}{\partial \lambda_{eq}} = F(x) = 0 \]
Now, consider an optimization problem with both equality and inequality constraints:

$$\Phi = \min_x f(x) \text{ s.t. } \begin{cases} F(x) = 0 \\ G(x) \leq 0 \end{cases}$$

In this case, the dual is the following (slightly) constrained optimization problem:

$$\Gamma = \max_{\lambda_{eq}, \lambda_{in} \geq 0} \min_x \left\{ f(x) + \lambda_{eq}^* F(x) + \lambda_{in}^* G(x) \right\}$$

The idea being that if an element of $\lambda_{in}$ is equal to zero, then the corresponding constraint in $G$ is inactive for that value of $x$, or $G(x) < 0$. Otherwise, it is active ($G(x) = 0$) and acts as an equality constraint. The necessary conditions are found to be:

$$\frac{\partial f}{\partial x} + \lambda_{eq}^* \frac{\partial F}{\partial x} + \lambda_{in}^* \frac{\partial G}{\partial x} = 0, \quad F(x) = 0, \quad \lambda_{in}^* G(x) = 0, \quad \lambda_{in} \geq 0$$

It is emphasized that these conditions for optimality are for the dual problem, which may have a solution different than the original problem. This issue is expressed by the concept of a duality gap. Specifically, $\Gamma \leq \Phi$ must hold for all cases. If the inequality is strict then there will be a duality gap and the solutions to the original problem and its dual may be different. However, if $f$, $F$ and $G$ are convex functions of $x$, then it is guaranteed that no duality gap will exist ($\Gamma = \Phi$) and the solution to both problems will be equal.

### 11.4.2 • The Scalar Case

We will now show that the MV problem generates a linear feedback that is exactly equal to that generated by the LQOC problem. More specifically, we will show the MV controller to be equal to that of the stochastic LQOC problem, which is equal to the deterministic LQOC controller, due to the certainty equivalence results of Chapter 8. The critical challenge is to identify the LQOC objective function weights that will make it companion to the MV problem. As we will see below, this will be a trivial exercise for the MV problem. Building upon the MV results, it will then be shown that the CMV controller is also of the LQOC class. However, in contrast to the MV case, the determination of the LQOC companion (i.e., determination of its objective function weights) will be a bit more challenging.

**The Scalar MV Problem**

Consider the system $\dot{x} = ax + bu + gw$ with the output equation $z = Dx x + Du u$, where $D_x = [1 \ 0]^*$ and $D_u = [0 \ 1]^*$. In this case, the MV problem is stated as:

$$\min_{l, \Sigma_x, \zeta(j)} \left\{ d_1 \zeta^{(1)} + d_2 \zeta^{(2)} \right\} \text{ s.t. } \begin{align*} 0 &= 2(a - bl) \Sigma_x + g^2 \Sigma_w \\
\zeta^{(j)} &= \rho_j (D_x - Du l) \Sigma_x (D_x - Du l)^* \rho_j^*, \quad j = 1, 2 \end{align*} \quad (11.82)$$

Now consider the infinite-time stochastic LQOC problem

$$\min_{u(x)} \left\{ \lim_{T \to \infty} \frac{1}{T} \int_0^T E \left[ qx^2 + ru^2 \right] dt \right\} \text{ s.t. } \dot{x} = ax + bu + gw \quad (11.83)$$
We know from Chapter 8 that the solution to this problem is \( u(x) = -lx \), where \( l = r^{-1} b p \) and \( p \) is the solution to the ARE: \( 0 = 2a p + q - p^2 b^2 r^{-1} \). However, let us take problem (11.83) in a different direction. First, note that

\[
q x^2 + ru^2 = [x \ u] M \begin{bmatrix} x \\ u \end{bmatrix} = z^* M z \quad \text{where} \quad M = \begin{bmatrix} q & 0 \\ 0 & r \end{bmatrix}
\]

(11.84)

Thus, \( E[z^* M z] = Tr \{ M E[z z^*] \} = Tr \{ M \Sigma_x(t) \} \), where \( \Sigma_x(t) \) is the covariance matrix of the output \( z \) at time \( t \), \( \Sigma_x(t) = (D_x - D_u l) \Sigma_x(t) (D_x - D_u l)^T, \) \( \Sigma_x(t) \) is the variance of the state, which is the solution of the Riccati differential equation \( \dot{\Sigma_x}(t) = 2(a - bl) \Sigma_x(t) + g^2 S_w, \) and the controller is assumed to be of the form \( u(x) = -lx \), where the gain, \( l \), is to be determined. If the diagonal elements of \( \Sigma_x(t) \) are \( \zeta^{(1)}(t) \) and \( \zeta^{(2)}(t) \), then \( Tr \{ M \Sigma_x(t) \} = q \zeta^{(1)}(t) + r \zeta^{(2)}(t) \). Evaluation of the objective function of (11.83), then, leads to:

\[
\lim_{T \to \infty} \frac{1}{T} \int_0^T q \zeta^{(1)}(t) + r \zeta^{(2)}(t) \, dt = q \zeta^{(1)} + r \zeta^{(2)}
\]

(11.85)

Thus, problem (11.83) can be restated with respect to the steady-state statistics:

\[
\min_{l, \Sigma_x, \zeta^{(0)}} \left\{ q \zeta^{(1)} + r \zeta^{(2)} \right\} \quad \text{s.t.:}
\]

\[
0 = 2(a - bl) \Sigma_x + g^2 S_w
\]

\[
\zeta^{(j)}(t) = \rho_j (D_x - D_u l) \Sigma_x (D_x - D_u l)^T \rho_j^T, \quad j = 1, 2
\]

(11.86)

Clearly, this restatement of the stochastic LQOC, problem (11.86), is exactly the same as the MV problem (11.82), if the objective function parameters are set equal to each other (i.e., \( q = d_1 \) and \( r = d_2 \)). Thus, the solution to (11.82) must be that of the LQOC: \( l = r^{-1} b p \), where \( p \) is the solution to the ARE.

To illustrate further, the scalar configuration allows us to calculate the solution to problem (11.82) directly. Given this solution we can show that it is in fact equal to the LQOC. First observe that \( \zeta^{(1)} = \Sigma_x, \zeta^{(2)} = l^2 \Sigma_x \). Thus, problem (11.82) can be stated as:

\[
\Phi = \min_{l, \Sigma_x} \left\{ (d_1 + l^2 d_2) \Sigma_x \right\} \quad \text{s.t.} \quad 2(a - bl) \Sigma_x + g^2 S_w = 0
\]

(11.87)

Then, the dual problem is found to be:

\[
\Gamma = \max_{l, \Sigma_x} \min_{\lambda_{eq}} \left\{ H(l, \Sigma_x, \lambda_{eq}) \right\}
\]

(11.88)

where \( H(l, \Sigma_x, \lambda_{eq}) = (d_1 + l^2 d_2) \Sigma_x + \lambda_{eq} (2 \Sigma_x (a - bl) + g^2 S_w) \). The necessary conditions for optimality are then determined by setting the partial derivatives equal to zero:

\[
\frac{\partial H}{\partial l} = 2l d_2 \Sigma_x - \lambda_{eq} 2 \Sigma_x b = 0
\]

(11.89)

\[
\frac{\partial H}{\partial \Sigma_x} = (d_1 + l^2 d_2) + \lambda_{eq} (2(a - bl)) = 0
\]

(11.90)

\[
\frac{\partial H}{\partial \lambda_{eq}} = 2 \Sigma_x (a - bl) + g^2 S_w = 0
\]

(11.91)
From (11.89), it is concluded that $l = d_1^{-1} b \lambda_{eq}$. Substituting this into (11.90), one finds the scalar ARE, but in terms of $\lambda_{eq}, d_1$ and $d_2$:

$$0 = 2a\lambda_{eq} + d_1 - (\lambda_{eq})^2 b^2 d_2^{-1}$$

(11.92)

Thus, if we just redefine our parameters as $q = d_1$, $r = d_2$ and $p = \lambda_{eq}$, we will recover the LQOC problem. This result gives an interesting interpretation of the ARE solution, $p$, as the Lagrange multiplier associated with the covariance equation.

A conceptually attractive way of interpreting the above results is that all stochastic LQOC controllers can be found on the Pareto frontier generated by the MV problem (recall Figure 11.2). Since the stochastic LQOC policy is the same as the deterministic LQOC, we find that indeed there is something special about all LQOC policies, regardless of the objective function weights used.

It is also interesting to note that the equivalence between the MV and the LQOC occurs only for the infinite-time versions of the problems. If one were to consider the finite-time versions, then strictly speaking this equivalence would not hold. To see this, recall that the Riccati equation of the finite-time LQOC solves backward in time, while the covariance equation solves forward in time. It is only in the limit that the two algebraic equations meet.

**Example 11.8.** Reconsider the process of Example 11.1. If $d_1 = 1$ and $d_2 = 0.1$, then the solution to the MV problem is found to be $l^* = 2.317$. If the LQOC objective function weights are defined as $q = d_1 = 1$ and $r = d_2 = 0.1$, the the solution to the ARE will be found to be $p = 0.2317$, and the LQOC controller is $l = 0.2317/0.1 = 2.317$.

**The Scalar CMV Problem**

Let us now add an inequality constraint to the MV problem: $\zeta^{(1)} = \Sigma_x \leq \zeta^{(1),\text{max}}$. The resulting CMV problem is:

$$\Phi = \min_{l,\Sigma_x} \{ (d_1 + l^2 d_2) \Sigma_x \} \quad \text{ s.t.} \quad 2(a - bl) \Sigma_x + g^2 S_w = 0$$

$$\Sigma_x \leq \zeta^{(1),\text{max}}$$

(11.93)

Then, the dual of this CMV problem is found to be:

$$\Gamma = \max_{\lambda_{in} \geq 0} \left[ \max_{\lambda_{eq}} \min_{l,\Sigma_x} \left\{ \left( d_1 + l^2 d_2 \right) \Sigma_x + \lambda_{eq} \left( 2\Sigma_x (a - bl) + g^2 S_w \right) \right\} \right]$$

$$- \lambda_{in} \zeta^{(1),\text{max}}$$

(11.94)

$$= \max_{\lambda_{in} \geq 0} \left[ \max_{\lambda_{eq}} \min_{l,\Sigma_x} \left\{ \left( d_1 + \lambda_{in} \right) + l^2 d_2 \right\} \Sigma_x + \lambda_{eq} \left( 2\Sigma_x (a - bl) + g^2 S_w \right) \right]$$

(11.95)

The point to note is that the inner max min problem is identical problem (11.88), the dual of the MV problem. The only difference is that $d_1$ is replaced by $d_1 + \lambda_{in}$. This indicates that for any nonnegative value of $\lambda_{in}$, the inner max min problem will be equal to an associated LQOC problem. Once the solution to the outer max problem is obtained, denoted as $\lambda_{in}^*$, the CMV controller can be calculated as $l = d_2^{-1} b \lambda_{eq}$, where $\lambda_{eq}$...
is the positive definite solution to the following ARE with a modified objective function weight:

\[ 0 = 2a \lambda_{eq} + (d_1 + \lambda_{in}^*) - (\lambda_{eq})^2 b^2 d_2^{-1}. \]

Of course, the above result could have also been deduced from Figure 11.4. That is, if we know that all MV solutions are found on the Pareto frontier, then the imposition of inequality constraints on the \( \zeta^{(i)} \) variables will serve only to move the solution to a different point on the frontier, which will be associated with some LQOC problem.

Thus, it is found that the CMV problem is also equivalent to a LQOC problem. The difference is that it is bit more challenging to determine the objective function weights associated with the companion LQOC problem. If the the computational algorithm used to solve the CMV problem provides access the dual variable solution, \( \lambda_{in}^* \), then the above development can be used to construct these weights. If the algorithm does not provide the optimal value of the dual variables, then one will have only the CMV controller gain, \( l^* \), to work with. In this case, how does one find the corresponding \( q \) weight? A simple rearrangement of the ARE (back in terms of \( q \), \( r \), and \( p \)) gives

\[ 0 = 2ap + q - l^2r. \]

Then, any selection of \( p > 0 \) such that \( r = \frac{bp}{l^2} > 0 \) and \( q = l^2r - 2ap > 0 \) will give the desired result.

Example 11.9. Reconsider the process of Example 11.8, \( d_1 = 1 \) and \( d_2 = 0.1 \). If the following inequality constraints are added \( \zeta^{(1)}_{\text{max}} = 10^2 \) and \( \zeta^{(2)}_{\text{max}} = 1 \), then the solution to the resulting CMV problem is found to be \( l^* = 0.4215 \). If \( p \) is selected to be 0.04215, then \( r = 0.1 \) and \( q = (0.4215)^2(0.1) = (2)(-1)(0.04215) = 0.1021 \). If these objective function weights are used within the LQOC, then the CMV feedback would be regenerated. It should be highlighted that one could actually select any positive value of \( p \), say for example \( p = 1 \). Then, as long as the resulting \( q \) and \( r \) are positive, the resulting LQOC controller will again regenerate the CMV solution.

It is emphasized that the solution to problem (11.95), \( \lambda_{in}^* \), will depend on the value of \( \zeta^{(1)}_{\text{max}} \) and implicitly on the size of the disturbance magnitude, \( g^2S_w \). This is in distinct contrast to the certainty equivalence results of the stochastic LQOC. The difference is that inequality constraint of the CMV problem will serve to modify the LQOC objective function weights, so that the stochastic LQOC controller is able to observe the inequality constraints. Of course, this assumes the inequality constraints are feasible. Recall Figure 11.5 and assume the point \((\zeta^{(1)}_{\text{max}}, \zeta^{(1)}_{\text{max}})\) is below the Pareto frontier. In this case, no amount of changes to the objective function weights of the stochastic LQOC will result in satisfaction of the variance inequalities.

11.4.3 • The Multivariate Case

Before addressing the general case, we will need a slight generalization of the ARE. Consider the following LQOC problem

\[
\Phi(x_0) = \min_{u(t)} \left\{ \int_0^\infty \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} dt \right\} \tag{11.96}
\]

s.t. \( \dot{x} = Ax + Bu \) and \( x(0) = x_0 \)
The solution to this problem is
\begin{align*}
u(t) &= -Lx(t) \\
L &= R^{-1}(PB + S)^* \\
0 &= A^*P + PA + Q - (PB + S)R^{-1}(PB + S)^* \\
\Phi(x_0) &= x_0^*Px_0
\end{align*}
(11.97, 11.98, 11.99, 11.100)

Of course \( R > 0 \) and \( Q - SR^{-1}S^* \geq 0 \) is required to guarantee that the compound matrix of the objective function is positive semi-definite. The following generalization of Theorem 7.2 can be found in Russell (1979).

**Theorem 11.1.** If \((A - BR^{-1}S^*, B)\) stabilizable and \((A - BR^{-1}S^*, Q)\) detectable, then (11.99) will have a positive definite solution \( P \) and the resulting \( L \) (from 11.98) will be such that \((A - BL)\) is stable.

**The Multivariate MV problem**

The MV problem is stated as:

\[
\min_{\zeta} \sum_{j=1}^{n_z} d_j \zeta^{(j)} \quad \text{s.t.} \quad (A - BL)\Sigma_x + \Sigma_x(A - BL)^* + GS_w G^* = 0 \\
\Sigma_x = (D_x - D_u L)\Sigma_x(D_x - D_u L)^* \\
\zeta^{(j)} = \rho_j \Sigma_x \rho_j^*, \quad j = 1 \ldots n_z
\]
(11.101)

Then, \( \sum_{j=1}^{n_z} d_j \zeta^{(j)} \) can be re-written as \( T \tau(D\Sigma_x) \) where \( D \) is a diagonal matrix containing the \( d_j \) terms. Additionally, \( T \tau(D\Sigma_x) = T \tau(D E[z z^*]) = E[z^* Dz] = \lim_{T \to \infty} \frac{1}{T} T_0 T E[z^* Dz] \) \( d t \). Thus, the MV problem can be written as:

\[
\min_{u(x)} \left\{ \lim_{T \to \infty} \frac{1}{T} T_0 T E[z^* Dz] \right\} \quad \text{s.t.} \quad \dot{x} = Ax + Bu + Gw \\
z = D_x x + D_u u
\]
(11.102)

Then, substitution of \( z = D_x x + D_u u \) into the expected value term gives

\[
E[z^* Dz] = E \left[ x^* D_x^* DD_x x + 2x^* D_x^* DD_u u + u^* D_u^* DD_u u \right]
\]

\[
= E \left[ \begin{bmatrix} x \\ u \end{bmatrix}^* \begin{bmatrix} D_x^* DD_x & D_x^* DD_u \\ D_u^* DD_x & D_u^* DD_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \right]
\]
(11.103)

Thus, the MV problem is finally written in the form of the infinite-time stochastic LQOC problem:

\[
\min_{u(x)} \left\{ \lim_{T \to \infty} \frac{1}{T} T_0 T E \left[ \begin{bmatrix} x \\ u \end{bmatrix}^* \begin{bmatrix} Q & S \\ S^* & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \right] \right\} \\
\text{s.t.} \quad \dot{x} = Ax + Bu + Gw
\]
(11.104)

where

\[
Q = D_x^* DD_x, \quad R = D_u^* DD_u \quad \text{and} \quad S = D_x^* DD_u
\]
(11.105)
The Multivariate CMV problem

To show equivalence with the CMV problem, we start with the convex form:

\[
\begin{align*}
\min_{\mu^{(i)}, Z_0, Z_1} & \left\{ \sum_{j=1}^{n_x} d_j \mu^{(j)} \right\} \\
\text{s.t.} & \ (AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0 \\
& \left[ \begin{array}{c}
\mu^{(j)} \\
\rho_j (D_x Z_0 - D_u Z_1)
\end{array} \right] > 0, \ j = 1 \ldots n_x
\end{align*}
\]

(11.106)

Then, the dual problem can be written as

\[
\begin{align*}
\max_{\lambda_j \geq 0} & \left\{ \sum_{j=1}^{n_x} d_j \mu^{(j)} + \sum_{j=1}^{n_z} \lambda_j (\mu^{(j)} - \zeta^{(j)}_{\max}) \right\} \\
\text{s.t.} & \ (AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0 \\
& \left[ \begin{array}{c}
\mu^{(j)} \\
\rho_j (D_x Z_0 - D_u Z_1)
\end{array} \right] > 0, \ j = 1 \ldots n_x
\end{align*}
\]

(11.107)

Since the objective function and all of the constraints of Problem (11.106) are convex, it is guaranteed that there will be no duality gap between the solution of (11.106) and (11.107). Then, the objective function of the minimization part of (11.107) can be rewritten as

\[
\min_{\mu^{(i)}, Z_0, Z_1} \left\{ \sum_{j=1}^{n_x} (d_j + \lambda_j) \mu^{(j)} \right\} - \sum_{j=1}^{n_z} \lambda_j \zeta^{(j)}_{\max}
\]

(11.108)

Example 11.10. Reconsider Example 11.5 with \( d_1 = d_2 = d_4 = 1 \) and \( d_3 = 0.01 \). If \( \zeta^{(1)}_{\max} = \zeta^{(2)}_{\max} = \zeta^{(4)}_{\max} = 10^2 \) and \( \zeta^{(3)}_{\max} = 1 \), then the controller obtained by solving the CMV problem is \( L^* = [-0.706 - 0.003 0.335] \), and the dual variables associated with the constraints \( \mu^{(j)} < \zeta^{(j)}_{\max}, \ j = 1 \ldots n_z \) are \( \lambda^* = [0 0 1.13 0]^T \). If \( D = \text{diag}([1 1 0.01 1]) + \text{diag}([1 1 1.13 1]) \), then application of (11.105) yields:

\[
Q = \begin{bmatrix}
10 & 6 & -3 \\
6 & 5 & -2 \\
-3 & -2 & 1
\end{bmatrix} \quad R = [2.1399] \quad S = \begin{bmatrix}
-3 \\
-2 \\
1
\end{bmatrix}
\]

(11.109)

If these weights are used within (11.98) and (11.99), then the controller is found to be \( L_{\text{LQOC}} = [-0.706 - 0.003 0.335] \), the same as that obtained from the CMV problem. In the following section we will develop a method to obtain the LQOC weights in the absence of the optimal multipliers \( \lambda^* \).
11.4. CMV and the LQOC Problem

11.4.4 LQOC Equivalence in Discrete-time

Consider the following infinite-time LQOC problem in the discrete-time framework:

\[ \Phi(x_0) = \min_{u_k, u_{k+1}, \ldots} \left\{ \sum_{k=0}^{\infty} \begin{bmatrix} x_k & u_k \end{bmatrix} \begin{bmatrix} Q & S \\ S^* & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \right\} \]  

\[ \text{s.t.} \quad x_{k+1} = A_d x_k + B_d u_k, \quad k = 0, \ldots \]  

\[ z_k = D_x x_k + D_u u_k, \quad k = 0, \ldots \]  

\[ z_{\min} \leq z_k \leq z_{\max}, \quad k = 0, \ldots \]  

\[ x_{k|k} = x_k \]  

The solution to this problem is

\[ u_k = -L x_k \]  

\[ L = (R + B_d^* P B_d)^{-1} (S + A_d^* P B_d)^* \]  

\[ P = A_d^* P A_d + Q - (S + A_d^* P B_d)(R + B_d^* P B_d)^{-1} (S + A_d^* P B_d)^* \]  

\[ \Phi(x_0) = x_0^* P x_0 \]  

Again, \( R > 0 \) and \( Q - S R^{-1} S^* \geq 0 \) is required to guarantee that the compound matrix of the objective function is positive semi-definite. The following is the generalization of Theorem 7.1.

**Theorem 11.2.** If \( (A_d - B_d R^{-1} S^*, B_d) \) stabilizable and \( (A - B_d R^{-1} S^*, Q) \) detectable, then (11.117) will have a positive definite solution \( P \) and the resulting \( L \) (from 11.116) will be such that \( (A_d - B_d L) \) is stable.

To illustrate equivalence with the discrete-time LQOC, consider the following discrete-time version of the stochastic LQOC problem:

\[ \Phi = \min_{u_k(x_k)} \left\{ \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N-1} E[z_k^* D z_k] d t \right\} \]  

\[ x_{k+1} = A_d x_k + B_d u_k + G_d w_k \]  

\[ z_k = D_x x_k + D_u u_k + D_w w_k \]

where \( D \) is a diagonal matrix with elements \( d_j \) or equivalently \( D = \sum_{j=1}^{n_s} d_j \rho_j^* \rho_j \). Similar to the continuous-time case, the relationship between \( (D, D_x, D_u) \) and \( (Q, R, S) \) is again given by

\[ Q = D_x^* D_x, \quad R = D_u^* D_u \quad \text{and} \quad S = D_x^* D_u \]

Conversion of the objective function of (11.119) to the form of a CMV problem is nearly identical to the continuous-time case:
Given this conversion of the objective function, the remainder of the proof is nearly identical, with the exception of using the discrete-time covariance equations of (11.27)-(11.28).

11.5 **Inverse Optimality and MPC Tuning**

To this point we have developed three types of controllers; CMV, LQOC and MPC. The results of the previous section indicate a relation between CMV and LQOC and those of Chapter 9 indicate a relation between LQOC and MPC. This section will explore the relationship between CMV and MPC by using LQOC as an intermediary.

The relation between LQOC and MPC is that both are based on the solution to a quadratic program. While the objective function of the two is the same, MPC will usually add a set of point-wise-in-time inequality constraints to the quadratic program. In addition, MPC is distinguished by being implemented in a receding horizon fashion. While the LQOC could be implemented with a receding horizon, the lack of inequality constraints will result in a policy that is identical to the analytically determined linear feedback.

The relation between CMV and LQOC is that the controller generated by the CMV problem will always be in the family of LQOC controllers. The difference is that the CMV problem contains statistically enforced inequality constraints. If these inequality constraints are not active, then objective function weights of the associated LQOC are trivially determined from Equation (11.105). If the inequality constraints are active, then the objective function weights of the associated LQOC will be modified. In this case, one will need to employ the notion of inverse optimality (i.e., if given a linear feedback $u = -Lx$, determine the objective function weights such that if used in the LQOC will regenerate the original linear feedback. Once these inverse optimal weights are determined, the resulting LQOC will be such that it observes the statistical inequality constraints of the CMV.

At this point one may be wondering how the linear feedback of the inverse optimal LQOC is different from the linear feedback of the CMV. In actuality they are not different. The difference is that the LQOC (with its inverse optimal weights) can be implemented using a receding horizon framework. Using this receding horizon framework the point-wise-in-time constraints of the MPC can be easily appended to the quadratic program. The net result is the generation of MPC objective function weights that are in alignment with the MPC constraints (assuming the MPC constraints are defined to be analogous to the CMV constraints). For a preliminary discussion of the value of creating such an alignment between the objective function and the constraints, recall Example 9.7.
11.5. Inverse Optimality and MPC Tuning

11.5.1 Inverse Optimality

The problem of being given a feedback gain matrix, \( u(t) = -Lx(t) \), and trying to find objective function weights, \( Q \) and \( R \), such that the the solution to the LQOC is equal to the original gain is the classic notion of inverse optimality (Kalman, [171]; Boyd et al., [161]). In the following, a slight generalization of those results will be given. Specifically, given an \( L \), we will try to find \( Q \), \( R \), and \( S \) such that (11.98)-(11.99) are satisfied. Of course, before one begins this inverse optimality search, it will be good to know if there exist matrices \( Q \), \( R \), and \( S \) capable of generating the given \( L \). Fortunately, the results of the previous section have laid this groundwork by showing that all controllers generated by the CMV problem will be in the family of LQOC controllers. Thus, we are guaranteed that such matrices will exist for all CMV generated controllers.

Let us begin with the continuous-time perspective, where the following Theorem provides a numeric procedure stated as an LMI feasibility problem.

**Theorem 11.3.** If there exists \( P > 0 \) and \( R > 0 \) such that

\[
\begin{bmatrix}
L^* R L - A^* P - PA \\
(L^* R - PB)^* - R
\end{bmatrix} > 0
\]  

(11.121)

Then \( Q^\wedge = L^* R L - A^* P - PA \) and \( S^\wedge = -L^* R + PB \) will be such that

\[
\begin{bmatrix}
Q & S^*
\end{bmatrix}
\begin{bmatrix}
S & R
\end{bmatrix} > 0
\]

(11.122)

and \( P \) and \( L \) will satisfy

\[
L = R^{-1}(PB + S)^*
\]

(11.123)

\[
0 = A^* P + PA + Q - (PB + S)R^{-1}(PB + S)^*
\]

(11.124)

**Proof.** Begin by substituting Equation 11.123 into the last term of Equation 11.124 to find \( L^* R L \). Then, solve for \( R \) from Equation 11.123 and \( Q \) from Equation 11.124. Finally, substitute both into Equation 11.122 to find Equation 11.121. \( \square \)

It should be highlighted that the procedure suggested by Theorem 11.3 is not expected to give a unique set of objective function weights. The simplest example is to double all of the weights, which will have no impact on the LQOC gain.

**Example 11.11.** Reconsider Example 11.5. In the case of \( \sigma^{(3)}_{x,\max} = 5 \), the constraint on \( \zeta^{(3)} \) is not active and the CMV solution is identical to the associated MV problem:

\[
L = \begin{bmatrix}
-1.96 & -0.251 & 0.977
\end{bmatrix}
\]

(11.125)

Using Equation (11.105), it is found that

\[
Q = \begin{bmatrix}
10 & 6 & -3 \\
6 & 5 & -2 \\
-3 & -2 & 1
\end{bmatrix}, \quad R = \begin{bmatrix}
1.01
\end{bmatrix}, \quad S = \begin{bmatrix}
-3 \\
-2 \\
1
\end{bmatrix}
\]

(11.126)
Using Theorem 11.3, one finds
\[
Q = \begin{bmatrix}
2.26 & 1.06 & -0.62 \\
1.06 & 1.53 & -0.35 \\
-0.62 & -0.35 & 0.91
\end{bmatrix}
\quad R = \begin{bmatrix}
0.42 \\
-0.32 \\
-0.51
\end{bmatrix}
\quad S = \begin{bmatrix}
0.72 \\
-0.32 \\
-0.51
\end{bmatrix}
\]
(11.127)

In both cases, placement of \(Q, R\) and \(S\) into (11.123)-(11.124) will recover the controller of (11.125). In the case of \(\sigma_z^{(3),\text{max}} = 1\), Example 11.5 finds
\[
L = \begin{bmatrix}
-0.706 & -0.003 & 0.335
\end{bmatrix}
\]
(11.128)
and application of Theorem 11.3 yields:
\[
Q = \begin{bmatrix}
1.294 & 0.393 & -0.225 \\
0.393 & 1.350 & -0.186 \\
-0.225 & -0.186 & 0.707
\end{bmatrix}
\quad R = \begin{bmatrix}
0.844 \\
-0.408 \\
-0.333
\end{bmatrix}
\quad S = \begin{bmatrix}
0.451 \\
0.408 \\
0.333
\end{bmatrix}
\]
(11.129)

While these matrices differ from those of (11.109) (from Example 11.9), both sets of matrices will regenerate (11.128) if used within (11.123)-(11.124). The MATLAB code used to generate the above results are given in Table 10.5.

Table 10.5: MATLAB/YALMIP code used in calculations for Example 11.11.

```matlab
clear
% Define System
nx=3; nu=1; nw=1; nz=4;
AA=[0 1 0; -3 -2 1;0 0 -.25]; BB=[0; 1; 0]; GG=[0; 0; 1];
Dx=[1 0 0;0 1 0;0 0 0; -3 -2 1]; Du=[0;0;1;1];
LLstar= [-7.0620e-001 -2.6503e-003 3.3545e-001]

yalmip('clear');
PP=sdpvar(nx,nx);
RR=sdpvar(nu,nu);
C1=[]; C2=[];
C1=[PP>=0, RR>=0];
C2=[ LLstar'*RR*LLstar-AA*PP-PP*AA LLstar'*RR-PP*BB;
( LLstar'*RR-PP*BB)' RR] >=0];
Constraints=[C1, C2];

options = sdpsettings('verbose',0,'solver','mosek');
sol = optimize(Constraints,[],options);
if sol.problem == 0
  % Extract and display value
  PPstar = value(PP)
  RRstar=value(RR)
  QQstar=LLstar'*RRstar*LLstar-AA'*PPstar-PPstar*AA
  SSstar=-(LLstar'*RRstar-PPstar*BB)
  [PPback,A行政执法,LLback] = care(AA, BB, QQstar, RRstar, SSstar); LLback
else
display('Hmm, something went wrong!');
sol.info
yalmiperror(sol.problem)
pause
end
```
The inverse optimality result in discrete-time is achieved by the follow discrete-time version of Theorem 11.3:

**Theorem 11.4.** If there exists $P > 0$ and $R > 0$ such that

$$
\begin{bmatrix}
P - A_d^*PA_d + L^*(R + B_d^*PB_d)L - L^*(R + B_d^*PB_d)P - A_d^*PB_d
\end{bmatrix}
\begin{bmatrix}
R
\end{bmatrix} > 0
$$

(11.130)

then $Q = P - A_d^*PA_d + L^*(R + B_d^*PB_d)L$ and $S = L^*(R + B_d^*PB_d) - A_d^*PB_d$ are such that

$$
\begin{bmatrix}
Q & S
\end{bmatrix}
\begin{bmatrix}
S & R
\end{bmatrix} > 0
$$

(11.131)

and $P$ and $L$ will satisfy

$$
P = A_d^*PA_d + Q - (S + A_d^*PB_d)(R + B_d^*PB_d)^{-1}(S + A_d^*PB_d)^*$$

(11.132)

$$
L = (R + B_d^*PB_d)^{-1}(S + A_d^*PB_d)^*$$

(11.133)

**Example 11.12.** Consider the following continuous-time model of a mass-spring-damper:

$$
A = \begin{bmatrix}
0 & 1 \\
-3 & -0.2
\end{bmatrix} \\
B = G = \begin{bmatrix}
0 \\
1
\end{bmatrix} \\
S_w = 0.5
$$

$$
D_x = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \\
D_u = \begin{bmatrix}
0 \\
1
\end{bmatrix}
$$

Application of the sample-and-hold method ($\Delta t = 0.2$ seconds) gives

$$
A_d = \begin{bmatrix}
0.941 & 0.192 \\
-0.576 & 0.903
\end{bmatrix} \\
B_d = G_d = \begin{bmatrix}
0.020 \\
0.192
\end{bmatrix} \\
\Sigma_w = 2.5
$$

If the CMV parameters are selected as $d_1 = d_2 = d_3 = 1$, $\zeta^{(1),max} = 0.4^2$ and $\zeta^{(3),max} = 3^2$, then the solution to the discrete-time CMV is $L = [-0.120 \ 0.862]$. Application of Theorem 11.4 yields:

$$
Q = \begin{bmatrix}
0.390 & 0.057 \\
0.057 & 0.567
\end{bmatrix}
$$

$$
R = \begin{bmatrix}
0.786
\end{bmatrix} \\
S = \begin{bmatrix}
-0.104 \\
0.515
\end{bmatrix}
$$

$$
P = \begin{bmatrix}
2.512 & 0.376 \\
0.376 & 0.958
\end{bmatrix}
$$

Of course, use of these weights within Equations (11.132)-(11.133), will regenerate the original CMV linear feedback $L = [-0.120 \ 0.862]$. 

\[\blacksquare\]
11.5.2 • MPC Tuning

Consider the following MPC formulation:

\[
\min_{u_{ij k}} \left\{ \sum_{i=k}^{k+N-1} \left[ \begin{array}{c} Q \\ S^* \end{array} \right] \begin{bmatrix} x_{ij k} \\ \mu_{ij k} \end{bmatrix} + \begin{bmatrix} x_{ij k}^* \\ P x_{ij k} \end{bmatrix} \right\}
\]

\[
\text{s.t. } x_{i+1 k} = A_d x_{ij k} + B_d u_{ij k}, \ i = k \ldots k+N-1
\]

\[
z_{ijk} = D x_{ijk} + D_u u_{ijk}, \ i = k \ldots k+N
\]

\[
z^{min} \leq z_{ijk} \leq z^{max}, \ i = k \ldots k+N
\]

\[
x_{0 k} = x_k
\]

As illustrated in Section 9.3, an arbitrary selection of the objective function weights \((Q, R, S, P)\) is likely to result in a misalignment between the objective function and the constraints. To obtain a better set of tuning parameters (and achieve the desired alignment) the results of the current chapter suggest the following procedure:

1. Solve the CMV problem (11.30) using the constraint parameters \(\sigma_z^{(j),\text{max}}\) set equal to \(\min\{z^{(j),\text{max}}, -z^{(j),\text{min}}\}\). If this problem is feasible, it will provide a statistically constrained linear feedback.

2. Using this CMV generated linear feedback, use Theorem 11.4 to construct objective function weights \((Q, R, S, P)\). If applied to Problem 11.134, but with the point-wise-in-time constraints of (Equation 11.137) removed, then the resulting policy will be identical to the CMV generated linear feedback.

3. Finally, implement Problem 11.134, with the point-wise-in-time constraints enforced to arrive at an MPC that is tuned (i.e., \(Q, R, S, P\) are selected) such that the objective function and its constraints are in alignment.

It should be highlighted that if the CMV problem of Step 2 is infeasible, then it is likely that the inequality constraints of the original MPC are too stringent. While the MPC can be implemented with such constraints it is expected that performance will be erratic and result an unusual level constraint relaxations (assuming a soft-constraint formulation is being employed).

An alternative approach is to start with the inequality constraint parameters of the CMV problem, \(\sigma_z^{(j),\text{max}}\). Then, construct the MPC by setting \(z^{(j),\text{max}} = \sigma_z^{(j),\text{max}}\) and \(z^{(j),\text{min}} = -\sigma_z^{(j),\text{max}}\). Then, use of inverse optimality on the CMV feedback will also result in the MPC being self-aligned.

Example 11.13. Reconsider the scenario of Example 9.6. Recall that \(z^{(1),\text{max}} = -z^{(1),\text{min}} = 0.4\) and \(z^{(3),\text{max}} = -z^{(3),\text{min}} = 3\). Using the method described above, the CMV problem should be defined with constraints \(\sigma_z^{(1),\text{max}} = 0.4\) and \(\sigma_z^{(3),\text{max}} = 3\). These are exactly those used in Example 11.12, which resulted in a linear feedback \(L = [-0.120 \quad 0.862]\). Then, using the \(Q, R, S, P\) matrices of 11.12 within a soft constrained MPC, results in the scatter plot of Figure 11.9, which is depicting the one-step prediction (recall the discussion of Example 9.6. Notice that a significant number of constraint violations are occurring. This can be explained by looking at the EDOR resulting from the CMV linear feedback. The dashed ellipse indicates the one standard deviation EDOR, which clearly satisfies all of the constraints. In fact, none of the CMV constraints are even active. However, the two standard deviation (solid) ellipse indicates that we should have expected the MPC
11.5. Inverse Optimality and MPC Tuning

If the CMV problem is redefined to be with respect to a two standard deviation EDOR (by setting $\alpha = 2$), then the linear feedback $L = [0.713 \quad 1.582]$ will be generated. In this case the two standard deviation EDOR will be completely contained within the constraints. However, implementation of the corresponding MPC (with inverse optimal weights) results in the scatter plot of Figure 11.10, which again results in significant constraint violations, although fewer than the previous case.

Figure 11.9. CMV EDORs and MPC Scatter plot for Example 11.13 both with $\alpha = 1$

Figure 11.10. CMV EDOR and MPC Scatter plot for Example 11.13 both with $\alpha = 2$

Figure 11.11. CMV EDOR and MPC Scatter plot for Example 11.13 both with $\alpha_1 = 4$ and $\alpha_3 = 2$
Let us not define the CMV problem with different $\alpha$ values for each constraint. Specifically, let $\alpha_1 = 4$ and $\alpha_3 = 2$. This will push the two standard deviation EDOR an additional two standard deviations away from the mass position constraint, as indicated in Figure 11.11. The scatter plot of that figure indicates that the resulting MPC will encounter only a handful of constrain violations ($< 5$ out of $10,000$ samples).

\[ \begin{align*}
A &= \begin{bmatrix}
-8000 & 0 & 0 & 0 & 10000 \\
2000 & -1500 & 0 & 0 & 0 \\
0 & 0 & -5000 & 0 & 0 \\
0 & 0 & 0 & -5000 & 0 \\
0 & 0 & 0 & 0 & -10
\end{bmatrix},
B &= \begin{bmatrix}
-75 & 75000 & 0 \\
-25 & 0 & 0 \\
0 & -8500 & -8.5 \times 10^5 \\
0 & 0 & -500 \times 10^5 \\
0 & 0 & 0
\end{bmatrix},
G &= \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
10
\end{bmatrix},
D_x &= \begin{bmatrix}
0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix},
D_u &= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{align*} \]

Assume $w$ is zero mean white noise with $S_w = 64 K^2 s$. The system model is converted to discrete-time using a sample time of 0.001 day.

Consider the soft-constrained MPC with objective function weights $Q = diag\{1 1 1 1 1\}$ and $R = diag\{1 1 1\}$ and constraints $z^{max} = -z^{min} = [100 20 6 5 100 160 5 0.02]$. In this case, the LQOC solution is:

\[ L = \begin{bmatrix}
-0.0005 & -0.0016 & 0 & 0 & -0.0011 \\
0.0036 & 0.0115 & -0.001 & 0 & 0.1287 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} \]
The two standard deviation EDOR associated with that linear feedback is depicted in Figure 11.13 along with the MPC scatter plots. Clearly, there is a misalignment between the LQOC EDOR (representing the MPC objective function) and the point-wise-in-time constraints.

Now consider the CMV problem with $d_j = 1, j = 1, \ldots, 8$, $\sigma_{z^{(j)}} = z^{(j),max}$ and
α = 2. The resulting CMV linear feedback is

\[
L = \begin{bmatrix}
-0.155 & -0.727 & -0.193 & 0.0381 & -4.35 \\
0.0057 & 0.0191 & -0.0064 & 0.0002 & 0.108 \\
0 & 0 & -0.0001 & 0 & 0.0003
\end{bmatrix}
\]

Application of Theorem 11.4 yields corresponding matrices Q, R, S and P. Then, implementation of the soft constrained MPC with these objective function weights results in scatter plots and the EDORs depicted in Figure 11.14. Clearly, there is much greater alignment between the objective function weights and the point-wise-in-time constraints, which results in fewer constraint violations, especially with respect to reactor temperature.

11.6 • Chapter Summary

The CMV controller design scheme has been shown to be an expansion of the widely popular LQOC method. While all CMV controllers are in the LQOC family, the CMV methodology does not require the selection of quadratic objective function weights. Rather, the CMV problem requires the specification of physically meaningful closed-loop variance bounds. Of course, once the CMV controller has been synthesized, one can use inverse optimality to obtain the corresponding quadratic objective function weights. Given these weights, a predictive form of the CMV controller can be constructed, from which point-wise-in-time inequality constraints can be enforced. Pulling all of the above concepts together, one could characterize this sequence of actions as a tuning method for MPC controllers, in the sense that objective function weights are determined such that they are in alignment with the point-wise-in-time inequality constraints.

The CMV problem was originally developed in Chmielewski & Manthanwar [170]. This paper also established equivalence with the LQOC problem and proposed the inverse optimality result. However, developments similar to the CMV problem can be found in Skelton et al. [169], van Hessem et al. [92], Xu et al. [173, 174], Lee et al. [95], Zhao et al. [97]. The discrete-time and PSI extensions discussed in this chapter were developed in Omell & Chmielewski [172].

Exercises

11.1. Using the code of Table 10.4, reproduce the results of Examples 11.4 and 11.5.
11.2. Convert Example 11.5 to the discrete-time case using the following steps.
   (1) Convert the matrices A, B, G and Sw to discrete-time form.
   (2) Implement discrete-time CMV problem (Equation 11.30) by suitable modification of Table 10.4.
   (3) Recalculate the values for Table 11.4.
   (4) Simulate the closed-loop process for each case and numerically verify the new values calculated for Table 11.4.

11.3. Repeat Exercise 10.2, but this time assume the controller will have one step of computational delay.
11.4. Re-consider Example 2.8. In deviation variable form the following system matrices will arise (see Example 2.8 for the interpretation of the $x$, $u$, $w$, and $z$ variables).

\[
A_d = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad G_d = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad D_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad D_u = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad D_w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

(i) Determine the controller suggested by the discrete-time CMV control problem of Equation (11.30), for the following cases: (Assume $\Sigma_w = 100$ for all cases).

a) $d_1 = d_2 = d_3 = 1$ and $\bar{z}_1 = \bar{z}_2 = \bar{z}_3 = \infty$

b) $d_1 = d_2 = 1$, $d_3 = 0$ and $\bar{z}_1 = \bar{z}_2 = \bar{z}_3 = \infty$

c) $d_1 = 1$, $d_2 = 0.01$, $d_3 = 0$ and $\bar{z}_1 = \bar{z}_2 = \bar{z}_3 = \infty$

d) $d_1 = 1$, $d_2 = 0$, $d_3 = 0$ and $\bar{z}_1 = \bar{z}_2 = \bar{z}_3 = \infty$

e) $d_1 = 1$, $d_2 = 0$, $d_3 = 0$, $\bar{z}_1 = 3$ and $\bar{z}_2 = 3 = \infty$

(ii) Simulate the closed-loop process for each case and calculate the output variances numerically. Compare these variances with those obtained from the CMV optimization as well as those obtained by substituting the controller into Equations (11.3)-(11.5), use the Matlab function 'dllyap'.

(iii) For each controller of part (i), determine the LQOC control objective function weights that correspond to that controller. Using the matlab function 'dare', verify that these weights will generate the original controller.

11.5. Consider the pair of surge tanks depicted in Figure 11.16. The objective is to deliver a constant exit flow, $q_2$, to the downstream unit, in the face of upstream variations at $q_0$. For each tank the volume of liquid should not exceed the total tank volume, and neither tank should be allowed to run dry. A volume balance around each tank yields: $\dot{V}_1 = q_0 - q_1$ and $\dot{V}_2 = q_1 - q_2$. If the nominal operating condition is given to be $\{V_1^{\text{nom}}, V_2^{\text{nom}}, q_0^{\text{nom}}, q_1^{\text{nom}}, q_2^{\text{nom}}\}$, then one can define deviation variables as: $\dot{V}_1' = V_1 - V_1^{\text{nom}}$ and $q_1' = q_1 - q_1^{\text{nom}}$, which generates a deviation based plant model: $\dot{V}_1' = q_0' - q_1'$ and $\dot{V}_2' = q_1' - q_2'$. If one selects $q_1$ and
As the manipulated variables and \( q_0 \) as the disturbance, then the following state space model is generated:

\[
A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 \\ 1 & -1 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]

where \( x = [V_1' V_2' q_1']^T, u = [q_1' q_2']^T \) and \( w = [q_0'] \). Additionally, given the objectives of the process it is reasonable to define the performance output as \( z = [V_1' V_2' q_1' q_2']^T \), which gives

\[
D_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad D_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D_w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

If the disturbance \( q_0' \) is driven by a first order filter, the resulting compound system is:

\[
A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix}
\]

\[
D_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad D_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D_w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

where \( \tau = 20 \) and \( S_w = 360 \).

(i) Using minimum variance control, complete Table 11.5:

(ii) Using CMV (with \( d_1 = d_2 = d_3 = d_4 = 1 \)), complete Table 11.6:

While the disturbance model of Exercise 10.5 is enough for basic covariance analysis, it cannot be used to design a Kalman filter, due to insufficient excitement of the uncontrolled process (i.e., the pair \( (A, G) \) is not stabilizable). To find additional excitement, we look more closely at the mass flow controllers regulating \( q_i \). It is postulated that the actual flow will deviate from the set-point command: \( q_i = q_i^{(sp)} + w_i \). If we assume this deviation is zero-mean white noise with a standard deviation equal to 0.5% of the nominal flow, then the following disturbance
Table 11.5. Solution to the MV problem for Exercise 10.5.

<table>
<thead>
<tr>
<th>$d_1 = d_2 = d_3 = 1$, $d_4 = 1$</th>
<th>$\sigma_z^{(1)}$</th>
<th>$\sigma_z^{(2)}$</th>
<th>$\sigma_z^{(3)}$</th>
<th>$\sigma_z^{(4)}$</th>
<th>Optimal Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_1 = d_2 = d_3 = 1$, $d_4 = 10^4$</td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_1 = d_2 = d_3 = 1$, $d_4 = 10^8$</td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.6. Solution to the CMV problem for Exercise 10.5.

<table>
<thead>
<tr>
<th>$\sigma_z^{(1)}$</th>
<th>$\sigma_z^{(2)}$</th>
<th>$\sigma_z^{(3)}$</th>
<th>$\sigma_z^{(4)}$</th>
<th>Optimal Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{z}_4 = 2.25$</td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tilde{z}_4 = 1.5$</td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tilde{z}_4 = 0.75$</td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The augmented process model will result:

$$G = \begin{bmatrix} 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ \frac{1}{\tau} & 0 & 0 & 0 \end{bmatrix}, \quad S_w = \begin{bmatrix} 360 & 0 & 0 & 0 \\ 0 & 0.25^2 & 0 & 0 \\ 0 & 0 & 0.25^2 & 0 \\ 0 & 0 & 0 & 0.25^2 \end{bmatrix}$$

Turning to the measurement equation, we assume that both tank volumes are measured as well as the flow inlet to tank 1. We assume the measurement noise to be zero-mean white noise with a standard deviation equal to 0.5% of the nominal hold-up/flow. Thus,

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad S_v = \begin{bmatrix} 0.75^2 & 0 & 0 \\ 0 & 0.75^2 & 0 \\ 0 & 0 & 0.25^2 \end{bmatrix}$$

(i) Determine the steady-state estimation error covariance resulting from application of an optimal state estimator.

(ii) Using PSI minimum variance control, complete the following table:

(iii) Using PSI CMV (with $d_1 = d_2 = d_3 = d_4 = 1$), complete the following table:

11.7. Re-consider Exercise 11.5 and solve the following problems:

(i) Do discrete-time version of Exercise 10.5 with time delay.

(ii) Do discrete-time version of Exercise 10.5 without time delay.

(iii) Do the MPC implementation and obtain the resulting process simulations.

11.8. Re-consider Exercise 11.6 and solve the following problems:

(i) Do discrete-time version of Exercise 10.6 with time delay.
Table 11.7. Solution to the MV problem for Exercise 10.6.

<table>
<thead>
<tr>
<th></th>
<th>$\sigma^{(1)}_z$</th>
<th>$\sigma^{(2)}_z$</th>
<th>$\sigma^{(3)}_z$</th>
<th>$\sigma^{(4)}_z$</th>
<th>Optimal Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1 = d_2 = d_3 = 1, ; d_4 = 1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
</tr>
<tr>
<td>$d_1 = d_2 = d_3 = 1, ; d_4 = 10^4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
</tr>
<tr>
<td>$d_1 = d_2 = d_3 = 1, ; d_4 = 10^8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
</tr>
</tbody>
</table>

Table 11.8. Solution to the CMV problem for Exercise 10.6.

<table>
<thead>
<tr>
<th></th>
<th>$\sigma^{(1)}_z$</th>
<th>$\sigma^{(2)}_z$</th>
<th>$\sigma^{(3)}_z$</th>
<th>$\sigma^{(4)}_z$</th>
<th>Optimal Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{z}_4 = 2.25$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
</tr>
<tr>
<td>$\tilde{z}_4 = 1.5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
</tr>
<tr>
<td>$\tilde{z}_4 = 0.75$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$L = \begin{bmatrix} - &amp; - &amp; - \end{bmatrix}$</td>
</tr>
</tbody>
</table>

(ii) Do discrete-time version of Exercise 10.6 without time delay.

(iii) Do the MPC implementation and obtain the resulting process simulations.
Chapter 12
Hardware Selection and the GBD

In Chapter 13, the Generalized Benders Decomposition (GBD) algorithm will be used to solve the Economic Linear Optimal Control (ELOC) problem. However, as a way of introduction, this chapter will utilize the GBD to solve the conceptually simpler hardware selection problem.

The basic idea behind cost optimal hardware selection is to find the lowest cost set of hardware such that a set of performance criteria are satisfied. Foundations of the hardware selection problem can be found in the seminal paper Bagajewicz [175], as well as in Bagajewicz [176]. The theorems of the previous two chapters are ideally suited for this class of problems in that each theorem is equipped with a set of performance criteria: \( \zeta(j) < \zeta(j)_{max}, j = 1 \ldots n_x \). The missing part is the influence of hardware (sensors and actuators) on system performance. In the case of sensors, we know that each row of the \( C \) matrix represents a measurement. In addition, the corresponding diagonal value of \( S_v \) (or \( \Sigma_v \)) indicates the level of noise that measurement will experience. In Example 10.5, it was shown that \( S_v \) had a strong influence on the performance criteria being satisfied or not. Extending this notion, if one of the diagonal element of \( S_v \) was set to an extremely large value (or equivalently, if the inverse of that element were set to zero), then the estimator would act as if that sensor does not exist. In the case of actuators, we know that each column of the \( B \) matrix represents an actuator. In Example 10.4, it was shown that the amount of power afforded to a manipulated variable had a strong influence on the ability of satisfy performance criteria with respect to the state variables. Extending this notion, if the power given to a manipulated variable is zero, then the controller will act as if that actuator does not exist.

In all of hardware selection problems to be described, we will find a recurring structure. Specifically, there will be an objective function that is linear with respect to the hardware attendance variables. This objective will reflect the aggregate cost of the hardware. Then, the performance criteria will be enforced by a set of LMI constraints, which are of course convex with respect to the attendance variable. As such, we will expect an SDP solver to easily provide the solution to these design problems. However, this will not be the case, due to an important detail — the hardware attendance variables will be limited to be either zero or one, and any value between will be deemed infeasible. That is, we cannot allow \( \frac{1}{2} \) a sensor or \( \frac{1}{4} \) of an actuator. It is either the full piece of equipment or nothing.

This restriction to integer values is a non-convex constraint on the attendance variables and will significantly complicate the solution procedure. While SDP software such
as YALMIP can incorporate the integer constraints, they do so by using the branch-and-bound algorithm, similar to the method advocated in Chmielewski et al., \cite{177}, Chmielewski and Peng, \cite{178}, Peng and Chmielewski, \cite{179}, and Ahmed et al., \cite{180}. Unfortunately, use of branch-and-bound for these problems can lead to extremely slow convergence, especially if there is a large number of hardware options, due to the combinatorial nature of the problem. However, it was recently determined that application of the Generalized Benders Decomposition (GBD) algorithm will lead to significant advances in the convergence rate for this class of problems, see Zhang et al. \cite{181} for details.

In this chapter, the GBD algorithm will be described in detail along with its application to hardware selection.

### 12.1 Examples of Hardware Selection

In this section, we will introduce three simple examples of hardware selection; sensor selection for data reconciliation systems, sensor selection for open-loop dynamic processes, and actuator selection for closed-loop dynamic processes. These examples will be used to illustrate hardware selection class of problems as well as provide a context for the application of the GBD for the solution to these problems. In Section 12.4, extension of these simple formulations will be provided, including the discrete-time framework as well as the conceptually challenging sensor selection problem for dynamic system in the closed-loop.

#### 12.1.1 Data Reconciliation Based Sensor Selection

In the context of an estimation system, system performance is quantified by estimation error variance. As indicated in Section 10.1, the estimation error variance of stream $j$ of a data reconciliation estimator is:

$$
\zeta_j^{(e)} = \rho_j C (C^* \Sigma_v^{-1} C)^{-1} C^* \rho_j^*. 
$$

If performance is specified by the bounds:

$$
\zeta_j^{(e)} = \zeta_j^{(e), \text{max}}, \quad j = 1 \ldots n_q,
$$

then the Schur complement theorem can be used to arrive at the following LMI based performance bounds:

$$
\begin{bmatrix}
\zeta_j^{(e), \text{max}} & \rho_j C \\
C^* \rho_j^* & \sum_{i=1}^{n_y} \gamma_i \Theta_i
\end{bmatrix} > 0, \quad j = 1 \ldots n_q \tag{12.1}
$$

where $C^* \Sigma_v^{-1} C = \sum_{i=1}^{n_y} \gamma_i \Theta_i$, $\Theta_i = C^* \rho_j^* \rho_j C / \sigma_{v_i}^2$, and $\gamma_i$ is the attendance variable for a sensor at stream $i$, see Section 10.1 for details. The sensor selection aspect is if $\gamma_i = 1$, then the sensor is placed at location $i$, and the variance of the measurement noise is as it should be ($\sigma_{v_i}^2 = \bar{\sigma}_{v_i}^2$). However, if $\gamma_i = 0$, then the noise of that measurement will be infinite (i.e., $\sigma_{v_i} \rightarrow \infty$). In that case, the optimal estimator will ignore this measurement, which is confirmed by looking back at the optimal data reconciliation estimator, $\hat{x} = C (C^* \Sigma_v^{-1} C)^{-1} C^* \Sigma_v^{-1} y$, with $\sigma_{v_i}^2 = 0$. Using this performance specification, the cost optimal sensor selection problem is stated as:

$$
\min_{\gamma_i \in \{0, 1\}} \left\{ \sum_{i=1}^{n_y} c_i^{(s)} \gamma_i \right\} \quad \text{s.t.} \quad (12.1) \tag{12.2}
$$

where $c_i^{(s)}$ is the cost of sensor $i$. Although Problem (12.2) seems like a straightforward optimization problem, given its linear objective and convex constraints, the presence of
12.1. Examples of Hardware Section 369

non-convex integer constraints on the attendance variables, $\gamma_i \in \{0, 1\}$, indicates that this problem should be classified as an Integer Convex Program (ICP). In contrast to a regular convex optimization problem, or Convex Program (CP), an ICP requires a more complicated solution procedure if a global solution is desired. To address such problems, Chmielewski et al., (2002) advocate the branch-and-bound algorithm. However, in Section 12.3 we will see that the GBD approach will also provide a global solution, but with much greater efficiency.

![Figure 12.1. Process diagram for Example 12.1.](image)

**Example 12.1.** Consider the process flow network of Example 6.3, depicted in Figure 12.1. There are 5 streams, each with a performance bound: $\zeta_{e}^{\max} = [3 \ 3 \ 6 \ 3 \ 6]$. Also, there are 5 locations available to place sensors — one at each stream. Each location corresponds to a row of the measurement matrix:

$$C = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1
\end{bmatrix}$$

The noise variance of each sensor is given by: $\sigma_v^2 = [1.6 \ 1.3 \ 0.75 \ 1.5 \ 0.25]$. If all of the sensors were used then the covariance matrix would be:

$$\Sigma_v = \begin{bmatrix}
1.6 & 0 & 0 & 0 & 0 \\
0 & 1.3 & 0 & 0 & 0 \\
0 & 0 & 0.75 & 0 & 0 \\
0 & 0 & 0 & 1.5 & 0 \\
0 & 0 & 0 & 0 & 0.25
\end{bmatrix}$$

Finally, the cost of each sensor is: $c(\gamma) = [10 \ 15 \ 20 \ 12 \ 25]$, all in $. Notice that the sensors with the largest cost are those with the smallest noise variance. Also, notice that since there are 5 decision variables, each with a 0 or 1 option, there are $2^5 = 32$ possible configurations. While about half of these are feasible (i.e., will be such that $\zeta_e(\gamma) < \zeta_e^{(\gamma)\max}$), the one with the lowest cost is $\gamma = [1 \ 1 \ 1 \ 0 \ 0]$, which has a total cost of $45$. An alternative to the exhaustive search over all possible combinations is to use the branch-and-bound algorithm. Using the MATLAB code of Table 12.5, the same solution is determined rather efficiently, using just 27 iterations and 1.6 seconds of computational time (case 1 of Table 12.4).

In Chmielewski et al. (2002) it is shown that the placement of multiple sensors in the same location (each with the same noise variance) is achieved by simply allowing $\gamma_i$
to take integer values greater than one. For example, the integer constraint on \( \gamma \) to be \( \gamma_i = \{0, 1, 2, 3, 4\} \). In this case, an exhaustive search would require an investigation of \( 5^3 = 3125 \) configurations to find the solution: \( \gamma = [1 \ 1 \ 0 \ 0] \). In contrast, the branch-and-bound algorithm finds the same solution, but requires only 43 iterations and 1.7 seconds (case 2 of Table 12.4).

If one would like to include the option of multiple sensors in the same location, but each with a different noise variance, then a more involved problem setup is required. Consider the scenario of having lower quality sensors available, but at a lower cost. In this case, one could simply augment the data vectors. For example, one could define:

\[
\sigma_v^2 = \begin{bmatrix}
\sigma_v^2 & 2\sigma_v^2 & 4\sigma_v^2 & 6\sigma_v^2 & 8\sigma_v^2
\end{bmatrix}
\]

\[
c^{(i)} = \begin{bmatrix}
c^{(i)}/2 & c^{(i)}/4 & c^{(i)}/6 & c^{(i)}/8
\end{bmatrix}
\]

In addition to these augmentations, the \( C \) matrix and the \( \zeta_e \epsilon^{max} \) vector would also need to be augmented, see the code of Table 12.5 for details. An alternative, but equally valid formulation can also be found in Chmielewski et al. (2002). If using the above augmentations, the result is that there will be 25 zero-one variables, so that an exhaustive search will require the investigation of \( 2^{25} = 33,554,432 \) combinations, which is clearly intractable. Using the branch-and-bound code of Table 12.5, the solution requires only 13,793 iterations and 360 seconds to find the solution, \( \gamma_1 \) = 1, \( \gamma_{17} \) = 1, \( \gamma_{21} \) = 1, \( \gamma_{22} \) = 1, \( \gamma_{24} \) = 1, \( \gamma_{25} \) = 1 for a total cost of \$15.75 (case 3 of Table 12.4). If each of the 25 sensor options is allowed up to 4 sensors of that type (\( \gamma_i \) \in \{0, 1, 2, 3, 4\}), then there will be \( 5^{25} = 2.98 \times 10^{17} \) possible configurations. In this case, the branch-and-bound requires 13,793 iterations and 360 seconds to find the solution, \( \gamma_{17} \) = 1, \( \gamma_{19} \) = 2, \( \gamma_{21} \) = 3, \( \gamma_{22} \) = 1, \( \gamma_{25} \) = 1 for a total cost of \$15.25 (case 4 of Table 12.4).

While the branch-and-bound algorithm is quite fast, compared to the exhaustive search, we are beginning to reach the limit of its capabilities. For example if there are 7 sensor options at each stream (for a total of 35 zero-one decisions), then the solution time will jump to 4,977 seconds = 83 minutes (case 5 of Table 12.4). As we will see in Section 12.3, the GBD approach can be used to reduce computational effort even further, in some case by several orders of magnitude.

### 12.1.2 Sensor Selection for Open-loop Dynamic Systems

If the system of interest is dynamic, then one would employ a Kalman filter for state estimation. In this case, the estimation error covariance is calculated from the associated Riccati equation:

\[
A\Sigma_e + \Sigma_e A^* + GS_w G^* - \Sigma_e C^* S^{-1} C \Sigma_e = 0
\]

(12.3)

where the diagonal elements of \( \Sigma_e \) specify performance: \( \rho_j \Sigma_e \rho_j^* = \zeta_e^{(j)} < \zeta_e^{(j),max} \), \( j = 1 \ldots n_q \). Similar to the data reconciliation case, the attendance of a sensor is given by \( C^* S^{-1} C = \sum_{i=1}^{n_q} \gamma_i \Theta_i \), where \( \Theta_i = C^* \rho_i^* \rho_i^* \Sigma_e \Sigma_e \), where \( \Sigma_e \) is the spectral density of sensor \( i \). Thus, the cost optimal sensor selection problem in the dynamic case is

\[
\min_{\gamma \in \{0,1\}^n} \left\{ \sum_{i=1}^{n_q} \gamma_i \right\}
\]

(12.4)

\[
\text{s.t. } \rho_j \Sigma_e \rho_j^* < \zeta_e^{(j),max}, \ j = 1 \ldots n_q
\]

(12.5)
Consider the following mass-spring-damper system:

$$A\Sigma + \Sigma A^* + GS_wG^* - \Sigma e \left( \sum_{i=1}^{n_v} \gamma_i \Theta_i \right) \Sigma e = 0 \quad (12.6)$$

To remove the nonlinearity associated with the Riccati equation one can employ Theorem 10.11 along with Equation (10.77) to arrive at the following equivalent form of the cost optimal sensor selection problem:

$$\min_{\gamma_i \in \{0,1\}, W_0 > 0} \left\{ \sum_{i=1}^{n_v} c^{(i)}(s) \gamma_i \right\}$$

s.t. $$\begin{bmatrix} \zeta^{(j),\max} e_j & \rho_j^* W_0^\gamma \end{bmatrix} > 0, j = 1 \ldots n_q \quad (12.7)$$

$$\left[ \sum_{i=1}^{n_v} \gamma_i \Theta_i - W_0 A - A^* W_0 \right] G^* W_0 S^{-1} \geq 0 \quad (12.8)$$

Similar to Problem (12.2), Problem (12.7) has a linear objective and convex constraints, but the presence of the integer constraints, $$\gamma_i \in \{0,1\}$$, indicates that this problem should be classified as a Mixed Integer Convex Program (MICP). The mixed qualifier stemming from the fact that this problem also contains continuous variables (those in $$W_0$$), and thus is a mix of integer and continuous variables. Again, the most straightforward approach to solving such a problem is to apply the branch-and-bound algorithm, but the slightly more complicated GBD approach can also be used and as we will see in Section 12.3 will do so with much greater efficiency.

**Example 12.2.** Consider the following mass-spring-damper system:

$$A = \begin{bmatrix} 0 & 1 \\ -3 & 0.2 \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad S_w = 0.5$$

Assume the performance bounds are: $$\zeta^{(1),\max}_e = 0.4, \ zeta^{(2),\max}_e = 0.75$$. Further assume there is one sensor for each state (i.e., $$C = I$$), the noise spectral density of each is $$\tilde{\tau}_e = [0.2 \ 0.4]$$ and the cost of the two sensors is $$c^{(i)} = [15 \ 20]$$. Using the branch-and-bound portion of the MATLAB code of Table 12.8, the solution requires just 5 iterations and 1.24 seconds of computational time to find the solution to be $$\gamma = [1 \ 0]$$ with a cost of $15 (case 1 of Table 12.7).

Now consider the option of multiple sensors in the same location, each with a different noise variance. Similar to Example 12.1, let us consider the scenario of having available lower quality sensors, but at a lower cost:

$$\tilde{\tau}_e = \begin{bmatrix} \tilde{\tau}_e & 2\tilde{\tau}_e & 4\tilde{\tau}_e & 6\tilde{\tau}_e & 8\tilde{\tau}_e & 10\tilde{\tau}_e & 12\tilde{\tau}_e & 14\tilde{\tau}_e \end{bmatrix}$$

$$c^{(i)} = \begin{bmatrix} c^{(i)} / 2 & c^{(i)} / 4 & c^{(i)} / 6 & c^{(i)} / 8 & c^{(i)} / 10 & c^{(i)} / 12 & c^{(i)} / 14 \end{bmatrix}$$

In this case, there will be 16 zero-one variables, so that an exhaustive search will require the investigation of $$2^{16} = 65,536$$ combinations. Using the branch-and-bound code of Table 12.8, the solution requires 5583 iterations and 52.6 seconds to find the solution: $$\gamma_7 = 1, \gamma_9 = 1, \gamma_{10} = 1, \gamma_{12} = 1, \gamma_{14} = 1$$ for a total cost of $10.54 (case 2 of Table 12.7). If each of the 16 sensor options is allowed up to 4 sensors of that type ($$\gamma_i \in \{0,1,2,3,4\}$$), then there will be $$5^{16} = 1.53 \times 10^{11}$$ possible configurations. In this case, the branch-and-bound requires 16,951 iterations and 206 seconds to find the solution, $$\gamma_7 = 1, \gamma_{10} = 1,$$
\gamma_{12} = 2, \gamma_{15} = 2\) for a total cost of $10.52 (case 3 of Table 12.7). Again, in Section 12.3 we will see that the GBD approach can reduce this computational effort by several orders of magnitude.

12.1.3 Actuator Selection for Closed-loop Dynamic System

For the actuator selection problem, let us state the performance criteria with respect to the state variables only. (This is not required, but it will reduce our notational burden.)

\[
\rho_i \Sigma_x \rho_i^* = \zeta_x^{(l)}, \quad l = 1 \ldots n_x \tag{12.10}
\]

\[
(A - BL) \Sigma_x + \Sigma_x (A - BL)^* + GS_w G^* = 0 \tag{12.11}
\]

It is highlighted that the linear feedback, \(L\), is a free variable that can be selected to be any value in an effort to satisfy the performance criteria (12.10). To indicate attendance of an actuator we will employ the variance of (or power to) each manipulated variable:

\[
\zeta_u^{(j)} = E[(u^{(j)})^2] = \rho_j \Sigma_u \rho_j^* = \rho_j L \Sigma_x L^* \rho_j^* < \delta_j \zeta_u^{(j),\text{max}}, \quad j = 1 \ldots n_u \tag{12.12}
\]

where \(u^{(j)}\) is the \(j\)th element of the manipulated variable vector \(u\), \(\zeta_u^{(j),\text{max}}\) is the variance bound on actuator \(j\), and \(\delta_j\) is the zero-one attendance variable. That is, if \(\delta_j = 1\), then \(\zeta_u^{(j)}\) can be any value less than \(\zeta_u^{(j),\text{max}}\). However, if \(\delta_j = 0\), then the variance of that manipulated variable must be zero. This, of course, is equivalent to the absence of that actuator. Thus, the actuator selection problem can be stated as:

\[
\min_{\delta_j} \left\{ \sum_{j=1}^{n_u} c_j^{(a)} \delta_j \right\} \quad \text{s.t.} \quad (12.10), \ (12.11) \text{ and } (12.12) \tag{12.13}
\]

where \(c_j^{(a)}\) is the cost of actuator \(j\). Using the methods of Theorem 10.8 the actuator selection problem can be equivalently stated as:

\[
\min_{\delta_j, \Sigma_x \geq 0, L} \left\{ \sum_{j=1}^{n_u} c_j^{(a)} \delta_j \right\} \quad \text{s.t.} \quad (AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0 \tag{12.15}
\]

\[
\rho_i \Sigma_x \rho_i^* < \zeta_x^{(l),\text{max}}, \quad l = 1 \ldots n_x \tag{12.16}
\]

\[
\begin{bmatrix}
\delta_j \zeta_u^{(j),\text{max}} \\
Z_1 \rho_j^*
\end{bmatrix}
\begin{bmatrix}
Z_0
\end{bmatrix}
> 0, \quad j = 1 \ldots n_u \tag{12.17}
\]

Similar to the sensor selection problems, the branch-and-bound method may be used to solve the MICP of Problem (12.14) globally, but as we will see in Section 12.3 the GBD method can also be used, and is likely to be computationally more efficient.

Example 12.3. Consider the furnace reactor system of Example 10.5 along with the following parameters for the actuator selection problem

\[
\zeta^{(a)} = \begin{bmatrix}
25^2 & 0.5^2 & 2.5^2 & 100^2 & 50^2
\end{bmatrix}
\]

\[
\zeta^{(a)} = \begin{bmatrix}
1025^2 & 3^2 & 0.01^2
\end{bmatrix}
\quad c^{(a)} = \begin{bmatrix}
8 & 6 & 1
\end{bmatrix}
\]
Recall that the first actuator is reactant feed rate, the second is fuel feed rate and the third is vent position. Given the above performance criteria the set of feasible actuators networks is $\delta = [0\ 1\ 1]$, $[1\ 0\ 0]$, $[1\ 0\ 1]$, $[1\ 1\ 0]$ or $[1\ 1\ 1]$. Given these feasible options, the one with the lowest cost ($7$) is $\delta = [0\ 1\ 1]$. Using the branch-and-bound code of Table 12.11, this solution requires 4 iterations and 1.24 seconds (case 1 of Table 12.10).

Now suppose that lower power actuators are available at lower cost. Specifically, assume that each actuator location $j$ has available a set actuator types $i = 1,\ldots,n_a$. Specifically, for actuator type $i$ at location $j$ the maximum power and cost is defined as: $\zeta^{(i,j),\max}$ and $c^{(a)}$. In this case, the actuator selection problem could be restated as:

$$\min \sum_{j=1}^{n_u} \sum_{i=1}^{n_a} c^{(a)} \delta_{i,j}$$

s.t. (12.15), (12.16),

$$\sum_{i=1}^{n_a} \delta_{i,j} \zeta^{(i,j),\max} \rho_j Z_1 > 0, \ j = 1,\ldots,n_u$$

For the current example, let us assume that all of the actuator types are defined by a size scaling factor $sf_i$. That is, the maximum power and cost of actuator type $i$ at location $j$ defined as: $\zeta^{(i,j),\max} = sf_i \zeta^{(j),\max}$ and $c^{(a)} = (sf_i)^{0.6} c^{(a)}$. If the scaling factor is:

$$sf = [1\ 0.5\ 0.4\ 0.3\ 0.2\ 0.1\ 0.05]$$

Then, the branch-and-bound code of Table 12.11 requires 249 iterations and 5.22 seconds to find the following solution: actuator type 6 at location 1, type 4 at location 2 and type 7 at location 3 for a total cost of $5.09 (case 2 of Table 12.10). Now let the number of options to be increased to:

$$sf = [1\ 0.5\ 0.44\ 0.4\ 0.35\ 0.3\ 0.25\ 0.2\ 0.15\ 0.1\ 0.05\ 0.025]$$

Then, the branch-and-bound code of Table 12.11 requires 1249 iterations and 22.01 seconds to find the following solution: actuator type 12 at location 1, type 3 at location 2 and type 7 at location 3 for a total cost of $5.03 (case 2 of Table 12.10).

### 12.2 Generalized Benders Decomposition

As discussed in Section 10.1, LMI constraints have the computational advantage of being convex. As such, a number of LMI (or SDP) based optimization tools have been developed. However, the downside of these solvers is that they can only address SDP constraints, and if the problem of interest contains any non-convex constraints, then the routine will not be able to solve the problem. In the hardware selection problems of Section 12.1, just such a situation was found — one must enforce convex LMI constraints, while simultaneously enforcing integer constraints on the attendance variables. In the controller design problems of Chapter 13, a similar situation will be found, in that a more general class of non-convex constraints will need to be enforced along with the LMI constraints. In both cases, application of the Generalized Benders Decomposition (GBD) algorithm is advocated. This approach will not only lead to globally optimal solutions, it will do so with relatively little computational effort as compared to branch-and-bound.
12.2.1 • General Framework

Consider the following generic optimization problem of two sets of variables, \( x \) and \( y \):

\[
\min_{x \in X, \ y \in Y} \{ f(x, y) \} \quad \text{s.t.} \quad G(x, y) \leq 0
\]  

(12.22)

where \( X \) represents constraints involving only \( x \), \( Y \) represents constraints involving only \( y \), and \( G(x, y) \leq 0 \) are constraints involving both. The main idea behind this classification of variables is that \( y \) is the set of complicating (or ‘bad’) variables. That is, if \( y \) is set to a constant, \( \bar{y} \), then the remaining problem (denoted the primal problem) will be easily solved.

\[
\min_{x \in X} \{ f(x, \bar{y}) \} \quad \text{s.t.} \quad G(x, \bar{y}) \leq 0
\]  

(12.23)

In the context of the hardware selection problems, the integer variables are complicating. If these are set to a constant then the remaining problem would be an easily solved convex optimization. Thus, the idea is to place all non-convex constraints and associated variables into \( y \in Y \). It is important to highlight that some \( \bar{y} \) may result in (12.23) being infeasible. To remove this possibility the choice should be limited to:

\[
\bar{y} \in V = \{ y : G(x, \bar{y}) \leq 0 \text{ for some } x \in X \}
\]  

(12.24)

In the subsequent developments, we will see that the set \( V \) plays an important role.

If \( X \) is a convex set and both \( f(x, y) \) and \( G(x, y) \) are convex in \( x \), then the original problem (12.22) is equivalent to the following master problem (see Geoffrion, [182], for details):

\[
\min_{y \in Y, \gamma_0} \{ \gamma_0 \} \quad \text{s.t.} \quad L^*(y, u) \leq \gamma_0 \text{ and } L_*(y, \lambda) \leq 0 \text{ for all } u \geq 0 \text{ and } \lambda \in \Lambda
\]  

(12.25)

where

\[
L^*(y, u) = \min_{x \in X} \{ f(x, y) + u^* G(x, y) \}
\]  

(12.26)

\[
L_*(y, \lambda) = \min_{x \in X} \{ \lambda^* G(x, y) \}
\]  

(12.27)

\[
\Lambda = \{ \lambda \geq 0 \text{ s.t. } 1 - \lambda^* 1 = 0 \} \quad \text{and} \quad 1 = [1 \ 1 \ldots \ 1]^T
\]  

(12.28)

The advantage of the master problem, (12.25), is that the optimizations of (12.26) and (12.27) are all convex optimizations and thus should be ‘easy’ to solve. The downside of the master problem is that the conditions \( L^*(y, u) \leq \gamma_0 \) and \( L_*(y, \lambda) \leq 0 \) must be enforced for an infinite number of points – for all \( u \geq 0 \) and \( \lambda \in \Lambda \). Clearly, this will be an impossible task. The genius of Benders contribution, [183], is that one need not enforce these condition for all \( u \geq 0 \) and \( \lambda \in \Lambda \). Rather, it is sufficient to enforce these conditions at a finite number of points. This notion leads to the following relaxed master problem:

\[
\min_{y \in Y, \gamma_0} \{ \gamma_0 \}
\]

\[
\text{s.t.} \quad L^*(y, \mu^{(k_1)}) \leq \gamma_0 \quad k_1 = 1 \ldots K_1
\]

\[
L_*(y, \lambda^{(k_2)}) \leq 0 \quad k_2 = 1 \ldots K_2
\]  

(12.29)

The idea being that appropriate selection of \( \mu^{(k_1)} \) and \( \lambda^{(k_2)} \) will lead to (12.29) being a sufficient approximation of (12.25). In the current nonlinear case, Geoffrion (1972) suggests the method of Table 12.1 for the selection of \( \mu^{(k_1)} \) and \( \lambda^{(k_2)} \).
12.2. Generalized Benders Decomposition

Problem Q

1. Initialize by selecting a point \( \bar{y} \in Y \cap V \). Obtain the solution to the primal problem, (12.23), along with the optimal multipliers associated with the constraints \( G(x, \bar{y}) \leq 0 \). Denote the optimal solution as \( x^* \) and the optimal multipliers (or dual variable solution) as \( u^* \). Set \( K_1 = 1, K_2 = 0, u^{(1)} = u^* \), the upper bound \( \text{UBD} = f(x^*, \bar{y}) \) and determine \( L^*(y, u^{(1)}) \).

2. Obtain a global solution to the relaxed master problem, (12.29). Denote the solution as \( y_0^* \) and \( y^* \). Set the lower bound \( \text{LBD} = y_0^* \). If \( \text{UBD} < \text{LBD} + \epsilon \), then the problem has converged.

3. Set \( \tilde{y} = y^* \). If the primal is feasible (i.e., \( \tilde{y} \in V \)), then go to step 3a. If the primal is infeasible (i.e., \( \tilde{y} \notin V \)), then go to step 3b.

   a. Solve the primal problem, (12.23), and denote the solution and optimal multipliers as \( x^* \) and \( u^* \). Set \( K_1 = K_1 + 1, u^{(K_1)} = u^* \) and determine \( L^*(y, u^{(K_1)}) \).

   b. Determine multipliers \( \lambda \in \Lambda \) such that \( \min_{x \in X} \{ \lambda^* G(x, \bar{y}) \} > 0 \). Set \( K_2 = K_2 + 1, \lambda^{(K_2)} = \lambda \) and determine \( L_*(y, \lambda^{(K_2)}) \). Return to step 2.

12.2.2 • Problem Q

If \( \tilde{y} \notin V \), then in Geoffrion (1972) it has been shown that there must exist \( \lambda \in \Lambda \) such that \( \min_{x \in X} \{ \lambda^* G(x, \bar{y}) \} < 0 \). Thus, a systematic way of selecting \( \lambda \) in step 3b is to obtain the solution to max \( \min_{x \in X} \{ \lambda^* G(x, \bar{y}) \} \), denoted as Problem Q. While Problem Q is conceptually appealing, obtaining its solution can be a bit challenging. To convert Problem Q into a more manageable form, Floudas et al., [184] and Bagajewicz and Manousiouthakis, [185], suggest the following nonlinear duality analysis (recall Section 11.4.1):

\[
\begin{align*}
\max_{\lambda \geq 0} \left\{ \min_{x \in X} \left\{ \lambda^* G(x, \bar{y}) \right\} \right\} & \quad \text{s.t.} \quad 1 - \lambda^* \mathbf{1} = 0 \\
= \min_\theta \left\{ \max_{\lambda \geq 0} \left\{ \min_{x \in X} \{ \lambda^* G(x, \bar{y}) \} + \theta (1 - \lambda^* \mathbf{1}) \right\} \right\} & \quad \text{s.t.} \quad \lambda^* \mathbf{1} = 0 \\
= \min_\theta \left\{ \max_{x \in X} \left\{ \min_{\lambda \geq 0} \{ \theta + \lambda^* G(x, \bar{y}) - \theta \mathbf{1} \} \right\} \right\} & \quad \text{s.t.} \quad G(x, \bar{y}) - \theta \mathbf{1} \leq 0 \\
= \min_{x \in X, \theta} \{ \theta \} & \quad \text{s.t.} \quad G(x, \bar{y}) - \theta \mathbf{1} \leq 0 \tag{12.30}
\end{align*}
\]

The first equality holds due to the convexity of \( \Lambda \), while the third equality holds due to the convexity of \( G(x, y) \) with respect to \( x \). Then, since \( X \) is a convex set, the solution to the last problem is readily obtained and the optimal multipliers (or dual variables) associated with the constraints \( G(x, \bar{y}) - \theta \mathbf{1} \leq 0 \) satisfy the criteria of step 3b. As such, step 3b (of Table 12.1) can be restated as:

3b. (revised) Solve problem Q, \( \min_{x \in X, \theta} \{ \theta \} \) \quad \text{s.t.} \quad G(x, \bar{y}) - \theta \mathbf{1} \leq 0 \), and denote the solution and optimal multipliers as \( x^* \) and \( \lambda^* \). Set \( K_2 = K_2 + 1, \lambda^{(K_2)} = \lambda^* \) and determine \( L_*(y, \lambda^{(K_2)}) \). Return to step 2.
The purpose of step 3b and the constraint $L_*(y, \lambda^{(K)}) \leq 0$ is as follows. If the relaxed master gives a point $\overline{y} \notin V$, then this $\overline{y}$ cannot be the solution to the original problem, since it is infeasible for all $x \in X$, recall (12.24). Thus, one would like to remove this infeasible point, $\overline{y}$, from subsequent relaxed master searches. Step 3b does this by simply finding a $\lambda^{(K)}$ such that $L_*(y, \lambda^{(K)}) > 0$. If we are lucky, then the constraint $L_*(y, \lambda^{(K)}) \leq 0$ will also exclude many other $\overline{y} \notin V$. However, the more important point is that the constraint $L_*(y, \lambda^{(K)}) \leq 0$ does not exclude any $y \in V$. To see this, consider any $y \in V$. Then, there exists $x \in X$ such that $G(x, y) \leq 0$. Thus, for all $\lambda \geq 0$ (which includes $\lambda^{(K)}$, $k=1...K$) it is easily observed that $L_*(y, \lambda) = \min_{x \in X} \{\lambda^* G(x, y)\} \leq 0$.

It should also be noted that Problem Q can be used to determine the feasibility / infeasibility of the primal problem. Specifically, if the optimal $\theta$ (denoted $\theta^*$) is greater than zero, then the primal is infeasible. If $\theta^*$ is less than or equal to zero, then the primal is feasible. Since Problem Q is likely the easier of the two to solve, one should begin with this problem. If $\theta^*$ turns out to be less than or equal to zero, then one should go back and solve the primal problem. However, in some cases the formulation of Problem Q is such that $\theta^*$ less than zero is impossible or $\theta^*$ less than or equal to zero is numerically impossible (see Example 12.6). In these cases, one would need to check the feasibility of the primal problem first, and if infeasible then move to Problem Q.

### 12.2.3 L-dual Adequacy

It must be emphasized that computational tractability of the GBD method hinges on being able to obtain $L_*(y, u)$ and $L_*(y, \lambda)$ as explicit functions of $y$. This is known as the L-dual adequacy property. While there are many cases of L-dual adequacy, the simplest and most common is when $f$ and $G$ are linearly separable: $f(x, y) = f_1(x) + f_2(y)$ and $G(x, y) = G_1(x) + G_2(y)$. In this case, the solution of a single optimization will yield the desired expressions:

$$L_*(y, u) = \min_{x \in X} \{f_1(x) + f_2(y) + u^*(G_1(x) + G_2(y))\}$$
$$= \min_{x \in X} \{f_1(x) - u^*G_1(x)\} + f_2(y) + u^*G_2(y)$$

(12.31)

$$L_*(y, \lambda) = \min_{x \in X} \{\lambda^* (G_1(x) + G_2(y))\} = \min_{x \in X} \{\lambda^* G_1(x)\} + \lambda^* G_2(y)$$

(12.32)

### 12.2.4 GBD if $f_1(x) = 0$

In all subsequent applications of the GBD algorithm, the problem of interest will be linearly separable. In fact, most of the problems of interest will be such that $f_1(x) = 0$. In this case, the procedure of Table 12.1 can be simplified. This is due to the fact that the primal problem, (12.23), will degenerate into the simple form of a feasibility problem. To see the impact of $f_1(x) = 0$, consider the dual form of the primal problem.

$$\max \min_{u \geq 0} \{f_2(y) + u^*(G(x, y))\}$$
$$= f_2(\overline{y}) + \max \min_{u \geq 0} \{u^*(G(x, \overline{y}))\}$$

(12.33)

If $\overline{y}$ is feasible (i.e., $\overline{y} \in V$), there exists $x \in X$ such that $G(x, \overline{y}) \leq 0$. In this case, the max operator of (12.33) can select $u^* = 0$ (i.e., any other $u^* \geq 0$ will be equal or worse). Since step 1 begins with $\overline{y} \in Y \cap V$, one finds that $u^{(1)} = u^* = 0$, UBD = $f_2(\overline{y})$ and $L_*(y, u^{(1)}) = f_2(y)$. During the first iteration of step 2, the relaxed master problem is
12.2. Generalized Benders Decomposition

Obtain the global solution to the relaxed master problem of (12.36). Denote the solution as 

\[ \min_{y \in Y, y_0} \{ y_0 \text{ s.t. } f_2(y) \leq y_0 \} = \min_{y \in Y} \{ f_2(y) \} \tag{12.34} \]

Denote the solution to this first iteration of the relaxed master as \( \bar{y}^{(1)} \), which gives the first LBD = \( f_2(\bar{y}^{(1)}) \). During step 3 there is of course two options: \( \bar{y}^{(1)} \in V \) or \( \bar{y}^{(1)} \notin V \). If \( \bar{y}^{(1)} \in V \), then one finds again that \( u^* = 0 \) and most notably UBD = \( f_2(\bar{y}^{(1)}) \). However, since the previous step just set LBD = \( f_2(\bar{y}^{(1)}) \), it is concluded that the iterations should stop, since LBD = UBD. This observation is easily extended to the following statement: If at any point in the iteration, the solution to the relaxed master is such that the primal is feasible, then the iterations should stop since a global solution to the original problem has been found. Of course, this observation requires \( f_1(x) = 0 \).

Let us now return to the first iteration of step 3 and consider the case of \( \bar{y}^{(1)} \notin V \). In this case, the revised version of step 3b will be implemented. This will involve solving Problem Q to obtain its solution and the optimal multiplier, denoted as \( x^* \) and \( \lambda^* \). If \( x^{(1)} = x^* \) and \( \lambda^{(1)} = \lambda^* \), then the assumption of a linearly separable \( G \), finds \( L \) as:

\[ L_s(y, \lambda^{(1)}) = \chi^{(1)}(y) + (\lambda^{(1)})^t G_2(y) \tag{12.35} \]

where the term \( \chi^{(1)} = (\lambda^{(1)})^t G_1(x^{(1)}) \) is a constant. During the second iteration of step 2, the relaxed master is of the form

\[ \min_{y \in Y} \{ f_2(y) \} \text{ s.t. } \chi^{(1)} + (\lambda^{(1)})^t G_2(y) \leq 0 \]

During the second iteration of step 3, there is again two options: \( \bar{y}^{(2)} \in V \) or \( \bar{y}^{(2)} \notin V \). If \( \bar{y}^{(2)} \in V \), then the iterations should stop. If \( \bar{y}^{(2)} \notin V \), then one should return to step 3b and determine \( L_s(y, \lambda^{(2)}) = \chi^{(2)} + (\lambda^{(2)})^t G_2(y) \), which will be used in the next relaxed master. Extending this analysis, one finds the general form of the relaxed master problem to be:

\[ \min_{y \in Y} \{ f_2(y) \} \text{ s.t. } \chi^{(k_2)} + (\lambda^{(k_2)})^t G_2(y) \leq 0, \quad k_2 = 1 \ldots K_2 \tag{12.36} \]

Of course, (12.36) is valid only under the assumptions of linearly separable \( f \) and \( G \) and \( f_1(x) = 0 \). Under these assumptions the GBD algorithm is summarized in Table 12.2 and depicted in Figure 12.2.

**Table 12.2. GBD algorithm if \( f \) and \( G \) are linearly separable and \( f_1(x) = 0 \)**

1. Initialize by selecting a point \( \bar{y} \in Y \cap V \). Set \( K_2 = 0 \) and UBD = \( f_2(\bar{y}) \).
2. Obtain the global solution to the relaxed master problem of (12.36). Denote the solution as \( \bar{y} \). Set LBD = \( f_2(\bar{y}) \).
3. Solve \( \min_{x, \theta} \{ \theta \} \text{ s.t. } G(x, \bar{y}) - \theta 1 \leq 0 \) and denote the solution and multipliers as \( x^*, \theta^* \) and \( \lambda^* \). If \( \theta^* \leq 0 \) (or the primal is feasible), stop — the algorithm has converged. If \( \theta^* > 0 \), set \( K_2 = K_2 + 1 \), \( x^{(k_2)} = x^* \), \( \lambda^{(k_2)} = \lambda^* \) and \( \chi^{(k_2)} = (\lambda^{(k_2)})^t G_1(x^{(k_2)}) \). Return to step 2.
12.2.5 • Summary of the GBD Algorithm

Before returning to the hardware selection problems, a few comments with regard to global optimality are in order. First it should be emphasized that to obtain equality between the original problem (12.22) and the (unrelaxed) master problem (12.25), it is required that \( X \) be a convex set and \( f \) and \( G \) be convex in \( x \). If this is not the case, then there may exist a duality gap between the two problems. In this case, the GBD algorithm will converge to the master problem which may have a solution that is not equal to that of the original problem. Also, all sub-problems of the algorithm must be solved to a global optimum. In the case of the primal and Problem Q, this should not be an issue if \( X \) is a convex set and \( f \) and \( G \) are convex in \( x \). However, for the relaxed master problem it is likely that \( Y \) is not a convex set and/or \( f \) and \( G \) are not convex in \( y \). In this case, one must use an appropriate non-convex solution algorithm to arrive at a global solution. However, since the variable \( x \) has been removed from the relaxed master the challenge of finding a global solution should be greatly reduced. For addition discussion on the subject of global solution guarantees, please see Bagajewicz and Manousiouthakis (1991).

In summary, the basic idea behind the GBD algorithm is to remove the multitude of non-complicating variables from the problem, so that one can focus on finding a global solution over the complicating variables. That is, the heart of the algorithm is the relaxed master problem, which focuses on the complicating variables. In the case of \( f \) and \( G \) linearly separable and \( f_i(x) = 0 \), (12.36) represents the constraints associated with the non-complicating variables. In essence, (12.36) is an approximation of the intended constraints, or stated more precisely (12.36) provides a finite support of the set \( V \). The purpose of Problem Q is to decide which constraints should be used to approximate \( V \). As we will see in Chapter 13, this approximation need only be good near the solution to the original problem. Thus, the task of Problem Q is to find an approximate constraint near the latest solution to the relaxed master. The only hiccup is the fact that the relaxed master must be solved repeatedly, and since this is likely a non-convex problem it will be time consuming to find a global solution. However, since the relaxed master does not have the baggage of the non-complicating variables (a burden that could be quite large, especially if the number of non-complicating variables or non-complicating constraints is large), repeatedly solving the relax master will be faster than solving the original, which carries the non-complicating variable burden.
12.3 • GBD Applied to Hardware Selection

In this section, we return to the simple hardware selection examples of Section 12.1 and illustrate how to apply the GBD algorithm to those problem as well as compare the computational aspects of the algorithm with the branch-and-bound procedure.

12.3.1 • Data Reconciliation Based Sensor Selection

In the sensor selection problem assuming a data reconciliation estimator, the problem of interest is (12.2). However, before the GBD algorithm can be applied, problem (12.2) will need to be restated in the following equivalent form:

\[ \min \left\{ \sum_{i=1}^{n_y} c_i^{(s)} y_i^{(cc)} \right\} \quad (12.38) \]

\[ y_i^{(cc)} \in \{0, 1\} \quad i = 1 \ldots n_y \quad (12.39) \]

\[ y_i^{(sp)} - y_i^{(cc)} \leq 0 \quad i = 1 \ldots n_y \quad (12.40) \]

\[ \left[ \zeta^{(j)\max} \rho_j C \right] \left[ \sum_{i=1}^{n_y} y_i^{(sp)} \Theta_i \right] > 0, \quad j = 1 \ldots n_q \quad (12.41) \]

The idea is to split the attendance variables \( y_i \in \{0, 1\} \) into two parts — one part for the capital cost of the sensor (denoted \( y_i^{(cc)} \)) and another part to indicate its impact on system performance (denoted as \( y_i^{(sp)} \)). As indicated by constraint (12.39), \( y_i^{(cc)} \) is required to take integer values. In contrast, \( y_i^{(sp)} \) is free to be any value, as long as (12.41) can be satisfied.

The relation between the two variables is dictated by constraint (12.40), and requires \( y_i^{(cc)} \) to be greater than or equal to \( y_i^{(sp)} \). One can think of these relations as follows: the system performance one gets from a sensor \( y_i^{(sp)} \) must be less than or equal to the amount paid for that sensor \( y_i^{(cc)} \). While it seems intuitive that Problem (12.38) is equivalent to (12.2), the following theorem (from Zhang et al., [181]) provides a guarantee.

**Theorem 12.1.** The solution \( y_i^{(cc)} \) of Problem (12.38) is equal to the solution \( y_i \) of Problem (12.2).

**Proof of Theorem 12.1:** Define the feasible regions of Problems (12.2) and (12.38) as:

\[ S_1 = \{ y = \{ y_i \} \mid y_i \in \{0, 1\} \text{ and (12.1) is satisfied} \} \]

\[ S_2 = \left\{ y^{(cc)} = \{ y_i^{(cc)} \} \mid y_i^{(cc)} \in \{0, 1\} \text{ and there exists } y_i^{(sp)} \text{ such that } y_i^{(sp)} \leq y_i^{(cc)} \text{ and (12.41) is satisfied} \right\} \]

Since the two problems have identical objective functions, we need only show that the two feasible regions are the same: \( S_1 \subseteq S_2 \) and \( S_2 \subseteq S_1 \). If \( y \in S_1 \), then one may always select \( y_i^{(sp)} = y_i \) to conclude \( y \in S_2 \). Thus, \( S_1 \subseteq S_2 \). If \( y^{(cc)} \in S_2 \), then for the corresponding \( y_i^{(sp)} \) we know that (12.41) holds. Since \( \gamma_i = y_i^{(cc)} - y_i^{(sp)} \geq 0 \), one finds that
Then, evaluation of $\gamma_i^{(cc)}$ within (12.1) gives:

$$
\left[\begin{array}{c}
\zeta_{(j),\text{max}} \\
C^* \rho_j^w \\
\sum_{i=1}^n \gamma_i^{(cc)} \Theta_i
\end{array}\right] = 
\left[\begin{array}{c}
\zeta_{(j),\text{max}} \\
C^* \rho_j^w \\
\sum_{i=1}^n \gamma_i^{(sp)} \Theta_i
\end{array}\right] + 
\left[\begin{array}{c}
0 \\
0 \\
\sum_{i=1}^n \tilde{\gamma}_i \Theta_i
\end{array}\right], \quad j = 1 \ldots n_q
$$

Thus, (12.1) is clearly positive definite for each $j = 1 \ldots n_q$, indicating that $\gamma_i^{(cc)} \in S_1$ and $S_2 \subseteq S_1$. □

Now that the two problems are guaranteed to be equal, we can now proceed to apply the GBD method to Problem (12.38). The first step is to select the complicating and non-complicating variables. If the complicating variables are selected to be $\gamma_i^{(cc)}$, then the set $Y$ is just the integer constraints (12.39). If the non-complicating variables are $\gamma_i^{(sp)}$, then the set $X$ corresponds to constraints (12.41), which of course is convex. This leaves the constraints of (12.40), which contain both the complicating and non-complicating variables. In summary, we have the following assignments:

$$
Y = \left\{ y = \{\gamma_i^{(cc)} \} \mid \gamma_i^{(cc)} \in \{0, 1\} \right\}
$$

$$
X = \left\{ x = \{\gamma_i^{(sp)} \} \mid (12.41) \text{ is satisfied} \right\}
$$

$$
f(x, y) = f_2(y) = \sum_{i=1}^{n_y} e_i^{(s)} \gamma_i^{(cc)}
$$

$$
G(x, y) = G_1(x) + G_2(y) = \begin{bmatrix}
\gamma_1^{(sp)} \\
\gamma_2^{(sp)} \\
\vdots \\
\gamma_n^{(sp)} \\
\gamma_1^{(cc)} \\
\gamma_2^{(cc)} \\
\vdots \\
\gamma_n^{(cc)}
\end{bmatrix}
$$

Notice that both $f$ and $G$ are linearly separable, convex in $x$ (and $y$) and $f_1(x) = 0$. The only part of the problem that is non-convex is the set $Y$. Following the procedure of Table 12.2, the first step is to initialize with a $\mathbf{y} \in Y \cap V$. An appropriate choice at this stage is $\gamma_i^{(cc)} = 1, \ i = 1 \ldots n_y$. Clearly, this point is in the set $Y$. In addition, this point corresponds to all the sensors being selected. Thus, if $\gamma_i^{(cc)} = 1, \ i = 1 \ldots n_y$ is not feasible, then original problem is also infeasible. As such the upper bound is set to $\text{UBD} = \sum_{i=1}^{n_y} e_i^{(s)}$ and $K_2 = 0$. In the first instance of step 2, the relaxed master is just

$$
\min_{y \in \mathcal{T}} f_2(y) = \min_{y \in \mathcal{T}} \sum_{i=1}^{n_y} e_i^{(s)} \gamma_i^{(cc)}
$$

Clearly, the solution to this problem is $\gamma_i^{(cc)} = 0$. Given
12.3. GBD Applied to Hardware Selection

the definition of $G(x, y)$, from (12.45), the Problem Q is found to be:

$$
\min_{\gamma, \theta} \{\theta\}
$$

s.t. $\gamma_i^{(sp)} - \gamma_i^{(cc)} - \theta \leq 0, \ i = 1 \ldots n_y$

$\begin{bmatrix}
\xi_{(j), \max} \\
C^s \rho^* \\
n_i \gamma_i^{(sp)} \Theta_i
\end{bmatrix} > 0, \ j = 1 \ldots n_q$

(12.46)
(12.47)
(12.48)

Thus, in the first instance of step 3, this problem is to be solved with $\gamma_i^{(cc)} = 0$. Using an SDP solver one can easily obtain $\theta^*, \gamma^{(sp)*}$ and $\lambda^*$. Since the data reconciliation based estimator will require at least one sensor, it will be impossible for $\theta^* \leq 0$ in the first iteration. Thus, the following are set $K_2 = K_2 + 1$, $\gamma^{(K_2)} = \gamma^{(sp)*}$, $\lambda^{(K_2)} = \lambda^*$ and $\chi^{(K_2)} = (\chi^{(K_2)})^* \gamma^{(K_2)} = \sum_i \lambda^{(K_2)} \gamma_i^{(K_2)}$. The procedure should then return to step 2. Based on the definition of $G_2(y)$, from (12.45), the general form of the relaxed master is

$$
\min_{\gamma_i^{(cc)} \in [0, 1]} \sum_{i=1}^{n_y} c_i^{(s)} \gamma_i^{(cc)}
$$

s.t. $\chi^{(K_2)} - (\chi^{(K_2)})^* \begin{bmatrix}
\gamma_i^{(cc)} \\
\gamma_i^{(cc)} \\
\cdots \\
\gamma_i^{(cc)}
\end{bmatrix} \leq 0, \ K_2 = 1 \ldots K_2$

(12.49)

Since the relaxed master contains non-convex integer constraints, $\gamma_i^{(cc)} \in [0, 1]$, it will require a special procedure to obtain a global solution. However, in contrast to the ICP of Problem (12.2), (12.49) contains only linear constraints. This combined with its linear objective opens the door to highly efficient extensions of the branch-and-bound solution procedure, many of which are MATLAB compatible, especially if using YALMIP. This along with the fact that the non-complicating variables are not present in (12.49) indicates that the relaxed master can be solve to a global optimum almost instantaneously. In the following example, the solver GUROBI will be used to solve the ILP. Figure 12.3 and Table 12.3 summarize the GBD algorithm for the data reconciliation based sensor selection problem. To obtain the optimal multipliers from the solution to Problem Q, one should use the YALMIP command ‘dual’ with the argument being the constraint of interest, those of (12.47).

Table 12.3. GBD algorithm for data reconciliation based sensor selection problem.

1. Set $K_2 = 0$ and UBD = $\sum_{i=1}^{n_y} c_i^{(s)}$.

2. Obtain the global solution to the relaxed master problem of (12.49). Denote the solution as $\gamma_i^{(cc)}$. Set LBD = $\sum_{i=1}^{n_y} c_i^{(s)} \gamma_i^{(cc)}$

3. Solve Problem Q (12.46) and denote its solution and multipliers as $\theta^*$, $x^*$ and $\lambda^*$. If $\theta^* \leq 0$, stop – the algorithm has converged. If $\theta^* > 0$, set $K_2 = K_2 + 1$, $x^{(K_2)} = x^*$, $\lambda^{(K_2)} = \lambda^*$ and $\chi^{(K_2)} = (\chi^{(K_2)})^* x^{(K_2)}$. Return to step 2.
Example 12.4. Reconsider the data reconciliation based sensor selection problems of Example 12.1. A comparison of the GBD algorithm with that of branch-and-bound is given in Table 12.4. Notice that for small problems, the overhead associated with GBD algorithm makes it slower than the branch-and-bound method. However, for larger problems, the reduction in the number of iterations significantly reduces computational effort. The MATLAB code used to generate Table 12.4 is given in Table 12.5. It should be highlighted that in this sensor selection problem, Problem Q will have a negative solution if the primal is feasible. Thus, there is no need to check feasibility of the primal. ■

Table 12.4. Comparison of GBD and Branch-and-Bound for Examples 12.1 and 12.4

<table>
<thead>
<tr>
<th>Case Number</th>
<th>BNB Time (seconds)</th>
<th>BNB Iterations</th>
<th>GBD Time (seconds)</th>
<th>GBD Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6</td>
<td>27</td>
<td>2.7</td>
<td>8</td>
<td>-65</td>
</tr>
<tr>
<td>2</td>
<td>1.7</td>
<td>43</td>
<td>2.7</td>
<td>8</td>
<td>-61</td>
</tr>
<tr>
<td>3</td>
<td>105</td>
<td>4,201</td>
<td>5.1</td>
<td>10</td>
<td>95.1</td>
</tr>
<tr>
<td>4</td>
<td>360</td>
<td>13,793</td>
<td>5.0</td>
<td>10</td>
<td>98.6</td>
</tr>
<tr>
<td>5</td>
<td>4,977</td>
<td>83,605</td>
<td>9.1</td>
<td>13</td>
<td>99.8</td>
</tr>
</tbody>
</table>

Table 12.5. MATLAB code for Examples 12.1 and 12.4.

clear all

% Data reconciliation based sensor selection
ny=5; nz=5; C=[1 0 0; 0 1 0; 1 0 1; 1 1 0; 1 1 1];
sigmav2_bar=[1.6 1.3 0.75 1.5 0.25];
zeta_max=[3; 3; 6; 3; 6]; c_s=[10 15 20 12 25];

% --- Cases 1 and 2 ---
gamma_max=1; gamma_max=4;
% --- Cases 3 and 4 ---
% gamma_max=1; gamma_max=4;
% ny=ny*5; c=[C; C; C; C];
% sigmav2_bar=[sigmav2_bar sigmav2_bar*2 sigmav2_bar*4 sigmav2_bar*6 ... sigmav2_bar*8];
% zeta_max=[zeta_max; zeta_max; zeta_max; zeta_max; zeta_max];
% c_s=[c_s c_s/2 c_s/4 c_s/6 c_s/8];
% --- Case 5 ---
% gamma_max=1; ny=ny*7; C=[C; C; C; C; C];
% sigmav2_bar=[sigmav2_bar sigmav2_bar*2 sigmav2_bar*4 sigmav2_bar*6 ... 
% sigmav2_bar*8 sigmav2_bar*10 sigmav2_bar*12];
% zeta_max=[zeta_max; zeta_max; zeta_max; zeta_max; zeta_max; 
% zeta_max; zeta_max];
% c_s=[c_s c_s/2 c_s/4 c_s/6 c_s/8 c_s/10 c_s/12];

% Branch-and-Bound Solution Method
yalmip('clear'); C1=[]; C2=[]; CSinvC=0; Iyy=eye(ny); Izz=eye(nz); tic
for ii=1:ny gamma(ii)=intvar(1,1); end
for ii=1:ny C1=[C1, 0<=gamma(ii)<=gamma_max]; end
for ii=1:ny rho_i=Iyy(ii,:);
    CSinvC=CSinvC+gamma(ii)*((rho_i*C)'*(rho_i*C)/sigmav2_bar(ii)); end
for ii=1:ny rho_i=Iyy(ii,:);
    C2=[C2, [zeta_max(ii) rho_i*C; (rho_i*C)' CSinvC]>=0]; end
Constraints=[C1,C2]; Objective=c_s*gamma;
options=sdpsettings('solver','bnb','bnb.solver','mosek','bnb.maxiter',...
    30000,'verbose',1);
MIPCsol = optimize(Constraints,Objective,options);
if MIPCsol.problem ~= 0 MIPCsol.info; pause, else
display('BnB Solution'); gamma_star=double(gamma), BNBcost=c_s*gamma_star', 
BNBtime=toc, end

% Generalized Benders Decomposition Method
itr=0; Termination=0; Iyy=eye(ny); Izz=eye(nz); chi_k2=[]; lambda_k2=[];
tic;
while Termination == 0
    % Relaxed Master Problem
    clear gamma_cc; yalmip('clear'); C1=[]; C2=[];
    for ii=1:ny gamma_cc(ii)=intvar(1,1); end
    for ii=1:itr C2=[C2, chi_k2(ii)-lambda_k2(ii,:) *gamma_cc(ii)<=0]; end
    Constraints=[C1,C2]; Objective=c_s*gamma_cc';
    options=sdpsettings('solver','gurobi','verbose',0);
    RM_sol = optimize(Constraints,Objective,options);
    gammacc_bar=double(gamma_cc)

    % Problem Q
    clear theta gamma_sp, yalmip('clear'); C1=[]; C2=[]; CSinvC=0;
    theta=sdpvar(1);
    for ii=1:ny gamma_sp(ii)=sdpvar(1); end
    for ii=1:ny C1=[C1, gamma_sp(ii)-gammacc_bar(ii)-theta<=0]; end
    for ii=1:ny rho_i=Iyy(ii,:);
        CSinvC=CSinvC+gamma_sp(ii)*((rho_i*C)'*(rho_i*C)/sigmav2_bar(ii)); end
    for ii=1:ny rho_i=Iyy(ii,:);
        C2=[C2, [zeta_max(ii) rho_i*C; (rho_i*C)' CSinvC]>=0]; end
    Constraints=[C1,C2]; Objective=theta;
    options=sdpsettings('solver','mosek','verbose',0);
    Qsol = optimize(Constraints,Objective,options);
    if Qsol.problem ~= 0 Qsol.info; pause, else
        itr=itr+1; theta_star=double(theta),
        gamma_sp_star=double(gamma_sp);
        for ii=1:ny lambda_star(ii)=dual(C1(ii)); end
        lambda_k2(itr,:)=lambda_star; chi_k2(itr)=lambda_star*gamma_sp_star';
        if theta_star <= 0 Termination=1; end
    end
end
display('GBD Solution'); gammacc_bar, GBDcost=c_s*gammacc_bar', 
GBDtime=toc, itr, percentreduction=100*(BNBtime-GBDtime)/BNBtime
In the case of Kalman filter based sensor selection, the attendance variables will again be split into a capital cost part, \( \gamma_i^{(cc)} \), and a system performance part, \( \gamma_i^{(sp)} \), to arrive at the following equivalent formulation of problem (12.7):

\[
\min \left\{ \sum_{i=1}^{n_y} c_i^{(s)} y_i^{(cc)} \right\} \quad \text{s.t.} \left(12.50\right)
\]

\[
\gamma_i^{(cc)} \in \{0, 1\} \quad i = 1 \ldots n_y
\]

\[
\gamma_i^{(sp)} - \gamma_i^{(cc)} \leq 0 \quad i = 1 \ldots n_y
\]

\[
\left[ \begin{array}{cc}
\gamma^{(i),\text{max}} & \rho_j \\
\rho_j & W_0
\end{array} \right] > 0, \quad j = 1 \ldots n_q
\]

\[
\left[ \sum_{i=1}^{n_y} \gamma_i^{(sp)} \Theta_i - W_0 A - A^* W_0 - W_0 G \\
G^* W_0 \\
S_w^{-1}
\right] > 0
\]

Similar to the data reconciliation based problem, the complicating variables are selected to be \( \gamma_i^{(cc)} \), which leaves \( \gamma_i^{(sp)} \) and \( W_0 \) to be the non-complicating variables. These selections result in the following assignments:

\[
Y = \{ y = \{ \gamma_i^{(cc)} \} \mid \gamma_i^{(cc)} \in \{0, 1\} \}
\]

\[
X = \{ x = \{ \gamma_i^{(sp)} \} \and W_0 \mid (12.53), (12.54) \text{ are satisfied} \}
\]

\[
f(x, y) = f_2(y) = \sum_{i=1}^{n_y} c_i^{(s)} y_i^{(cc)}
\]

\[
G(x, y) = G_1(x) + G_2(y) = \begin{bmatrix}
\gamma_1^{(sp)} \\
\gamma_2^{(sp)} \\
\vdots \\
\gamma_{n_y}^{(sp)}
\end{bmatrix} + \begin{bmatrix}
-\gamma_1^{(cc)} \\
-\gamma_2^{(cc)} \\
\vdots \\
-\gamma_{n_y}^{(cc)}
\end{bmatrix}
\]

Again, both \( f \) and \( G \) are linearly separable, convex in \( x \) (and \( y \)) and \( f_1(x) = 0 \). Given (12.58) Problem Q is defined as:

\[
\min \{ \theta \} \quad \text{s.t.} \quad \gamma_i^{(sp)} - \gamma_i^{(cc)} - \theta \leq 0 \quad i = 1 \ldots n_y, \quad (12.53) \text{ and } (12.54)
\]

Then, the algorithm of Table 12.6 will apply.

**Example 12.5.** Reconsider the Kalman filter based sensor selection problems of Example 12.2. A comparison of the GBD algorithm with that of branch-and-bound is given in Table 12.7. The MATLAB code used to generate Table 12.7 is given in Table 12.8. Once again, Problem Q will have a negative solution if the primal is feasible. Thus, there is no need to check feasibility of the primal problem directly.  

\[\square\]
12.3. GBD Applied to Hardware Selection

Table 12.6. GBD algorithm for Kalman filter based sensor selection problem.

1. Set $K_2 = 0$ and $\text{UBD} = \sum_{i=1}^{n_y} c_i^{(s)}$.

2. Obtain the global solution to the relaxed master problem of (12.49). Denote the solution as $\bar{\gamma}^{(s)}$. Set $\text{LBD} = \sum_{i=1}^{n_y} c_i^{(s)} \bar{\gamma}^{(s)}$.

3. Solve Problem Q (12.59) and denote its solution and multipliers as $\theta^*, x^* $ and $\lambda^*$. If $\theta^* \leq 0$, stop – the algorithm has converged. If $\theta^* > 0$, set $K_2 = K_2 + 1$, $x^{(K_2)} = x^*$, $\lambda^{(K_2)} = \lambda^*$ and $\chi^{(K_2)} = (\lambda^{(K_2)})^* x^{(K_2)}$. Return to step 2.

Table 12.7. Comparison of GBD and Branch-and-Bound for Examples 12.2 and 12.5

<table>
<thead>
<tr>
<th>Case Number</th>
<th>BNB Time (seconds)</th>
<th>BNB Iterations</th>
<th>GBD Time (seconds)</th>
<th>GBD Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.3</td>
<td>5</td>
<td>0.79</td>
<td>2</td>
<td>39.6</td>
</tr>
<tr>
<td>2</td>
<td>49.9</td>
<td>5,583</td>
<td>1.70</td>
<td>4</td>
<td>96.6</td>
</tr>
<tr>
<td>3</td>
<td>137</td>
<td>10,704</td>
<td>2.28</td>
<td>5</td>
<td>98.3</td>
</tr>
</tbody>
</table>

Table 12.8. MATLAB code for Examples 12.2 and 12.5.

```matlab
clear all

% Kalman filter based sensor selection
nx=2; ny=2; A=[0 1; -3 -0.2]; G=[0; 1]; Sw=0.5; zeta_max = [0.4^2 0.75^2]; Ixx=eye(nx); C=[1 0;0 1]; sigmav2_bar=[0.2 0.4]; c_s=[15 20]; Iyy=eye(ny);

% --- Cases 1 ---
gamma_max=1;
% --- Cases 2 and 3 ---
gamma_max=4;
ny=ny*8; Iyy=eye(ny); C=[C; C; C; C; C; C; C];
sigmav2_bar=[sigmav2_bar sigmav2_bar*2 sigmav2_bar*4 sigmav2_bar*6 ...
sigmav2_bar*8 sigmav2_bar*10 sigmav2_bar*12 sigmav2_bar*14];
c_s=[c_s/2 c_s/2 c_s/6 c_s/8 c_s/10 c_s/12 c_s/14];

% Branch-and-Bound Solution Method
yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5invC=0; tic
for ii=1:ny gamma(ii)=intvar(1,1); end
for ii=1:ny C1=[C1, 0<=gamma(ii)<=gamma_max]; end
for ii=1:ny rho_i=Iyy(ii,:); C3=[C3, [zeta_max(ii) rho_i; rho_i' W0]>=0]; end
C4=[W0>0]; Constraints=[C1,C2,C3,C4]; Objective=c_s*gamma;
options=sdpsettings('solver','bnb','bnb.solver','mosek','bnb.maxiter',...
30000,'verbose',1);
MICPsol = optimize(Constraints,Objective,options);
if MICPsol.problem ~= 0 MICPsol.info; pause, else display('BnB Solution'); gamma_star=double(gamma),
BNBcost=c_s*gamma_star, BNBtime=toc, end

% Generalized Benders Decomposition Method
itr=0; Termination=0; chi_k2=[]; lambda_k2=[]; tic;
```
while Termination == 0
% Relaxed Master Problem
  clear gamma_cc; yalmip('clear'); C1=[]; C2=[];
  for ii=1:ny gamma_cc(ii)=intvar(1,1); end
  for ii=1:ny C1=[C1, 0<=gamma_cc(ii)<=gamma_max]; end
  for ii=1:itr C2=[C2, [chi_k2(ii)-lambda_k2(ii,:)*gamma_cc(ii)<=0]; end
  Constraints=[C1,C2]; Objective=c_s*gamma_cc';
  options=sdpsettings('solver','gurobi','verbose',0);
  RM_sol = optimize(Constraints,Objective,options);
  gammacc_bar=double(gamma_cc)

% Problem Q
  clear theta gamma_sp W0, yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[];
  CSinvC=0; theta=sdpvar(1); W0=sdpvar(nx);
  for ii=1:ny gamma_sp(ii)=sdpvar(1); end
  for ii=1:ny C1=[C1, gamma_sp(ii)-gammacc_bar(ii)-theta<=0]; end
  for ii=1:ny rho_i=Iyy(ii,:);
    CSinvC=CSinvC+gamma_sp(ii)*(rho_i*C)'*(rho_i*C)/sigmav2_bar(ii); end
  C2=[C2, [CSinvC-W0*A-A'*W0 W0*G; G'*W0 inv(Sw)] >= 0];
  for ii=1:nx rho_i=Ixx(ii,:);
    C3=[C3, [zeta_max(ii) rho_i; rho_i'*W0]>=0]; end
  C4=[W0>0]; Constraints=[C1,C2,C3,C4]; Objective=theta;
  options=sdpsettings('solver','mosek','verbose',0);
  Qsol = optimize(Constraints,Objective,options);
  if Qsol.problem ~= 0 Qsol.info; pause, else
    itr=itr+1; theta_star=double(theta); gamma_sp_star=double(gamma_sp);
    for ii=1:ny lamda_star(ii)=dual(C1(ii)); end
    lambda_k2(itr,:)=lamda_star; chi_k2(itr)=lamda_star*gamma_sp_star';
    if theta_star <= 0 Termination=1; end
  end
display('GBD Solution'); gammacc_bar, GBDcost=c_s*gammacc_bar', GBDtime=toc,
itr, PercentReduction=100*(BNBtime-GBDtime)/BNBtime

12.3.3 • Actuator Selection for Closed-loop Dynamic Systems

In the actuator selection problem, the attendance variables are again be split into a capital cost part, $\delta_{j}^{(cc)}$, and a system performance part, $\delta_{j}^{(sp)}$, to arrive at the following equivalent formulation of problem (12.14):

$$
\min_{\delta_{j}^{(cc)}, \delta_{j}^{(sp)}} \left\{ \sum_{j=1}^{N_u} c_j (a) \delta_{j}^{(cc)} \right\} \quad \text{s.t. (12.60)}
$$

$$
\delta_{j}^{(cc)} \in \{0, 1\} \quad j = 1...n_u
$$

$$
\delta_{j}^{(sp)} - \delta_{j}^{(cc)} \leq 0 \quad j = 1...n_u
$$

$$
(AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0
$$

$$
\rho_i Z_0^* \rho_i < \zeta_{x,max}^{(l)}, \quad l = 1...n_x
$$

$$
\left[ \delta_{j}^{(sp)} \zeta_{x,max}^{(j)} \rho_i Z_1 \right] > 0, \quad j = 1...n_u
$$

Similar to the sensor selection problems, the complicating variables are selected to be $\delta_{j}^{(cc)}$, which leaves $\delta_{j}^{(sp)}$, $Z_0$ and $Z_1$ to be the non-complicating variables. These selections
result in the following assignments:
\[
Y = \{ y = \{ \delta^{(cc)}_j \} \mid \delta^{(cc)}_j \in \{0, 1\} \} \quad (12.66)
\]
\[
X = \{ x = \{ \delta^{(sp)}_j, Z_0 \text{ and } Z_1 \} \mid (12.63), (12.64) \text{ and } (12.65) \text{ are satisfied} \} \quad (12.67)
\]
\[
f(x, y) = f_2(y) = \sum_{j=1}^{n_s} c_j^{(a)} \delta^{(cc)}_j \quad (12.68)
\]
\[
G(x, y) = G_1(x) + G_2(y) = \begin{bmatrix}
\delta^{(sp)}_1 \\
\delta^{(sp)}_2 \\
\vdots \\
\delta^{(sp)}_{n_s}
\end{bmatrix} + \begin{bmatrix}
-\delta^{(cc)}_1 \\
-\delta^{(cc)}_2 \\
\vdots \\
-\delta^{(cc)}_{n_s}
\end{bmatrix}
\quad (12.69)
\]

Again, both \( f \) and \( G \) are linearly separable, convex in \( x \) (and \( y \)) and \( f_1(x) = 0 \). Given \( (12.69) \), the relaxed master is found to be:
\[
\begin{align*}
\min_{\delta^{(cc)}_j \in \{0, 1\}} \sum_{j=1}^{n_s} c_j^{(a)} \delta^{(cc)}_j \quad \text{s.t.} \quad \chi^{(k_2)} - (\lambda^{(k_2)})^T \begin{bmatrix}
\delta^{(cc)}_1 \\
\delta^{(cc)}_2 \\
\vdots \\
\delta^{(cc)}_{n_s}
\end{bmatrix} 
\leq 0, \quad k_2 = 1 \ldots K_2
\end{align*}
\quad (12.70)
\]

Problem Q is defined as:
\[
\begin{align*}
\min_{\delta^{(sp)}_j, Z_0, Z_1, \theta} \{ \theta \} \quad \text{s.t.} \quad \delta^{(sp)}_j - \delta^{(cc)}_j - \theta \leq 0 \quad (12.63), (12.64) \text{ and } (12.65)
\end{align*}
\quad (12.71)
\]

In contrast to the sensor selection problems, the system performance variable, \( \delta^{(sp)}_j \), cannot be negative, due to the 1,1 element \( (12.65) \), which is required to be positive if the matrix is to be positive definite. As such \( \theta^* \leq 0 \) cannot be used as the stopping criteria. Thus, the stopping criteria of the algorithm of Table 12.9 checks for primal problem feasibility: Does there exist \( Z_0 > 0, Z_1 \text{ and } \delta^{(sp)}_j \) such that \( (12.62)-(12.65) \) are satisfied with \( \delta^{(cc)}_j = \delta^{(cc)}_j \).

Table 12.9. GBD algorithm for actuator selection problem.

1. Set \( K_2 = 0 \) and UBD = \( \sum_{j=1}^{n_s} c_j^{(a)} \).
2. Obtain the global solution to the relaxed master problem of \( (12.70) \). Denote the solution as \( \delta^{(cc)}_j \). Set LBD = \( \sum_{j=1}^{n_s} c_j^{(a)} \delta^{(cc)}_j \).
3. If the primal problem is feasible, stop — the algorithm has converged. If the primal is infeasible, solve Problem Q \( (12.71) \) and denote its solution and multipliers as \( \theta^*, x^* \) and \( \lambda^* \). Set \( K_2 = K_2 + 1 \), \( x^{(K_2)} = x^* \), \( \lambda^{(K_2)} = \lambda^* \) and \( \chi^{(K_2)} = (\lambda^{(K_2)})^T x^{(K_2)} \). Return to step 2.
Table 12.10. Comparison of GBD and Branch-and-Bound for Examples 12.3 and 12.6

<table>
<thead>
<tr>
<th>Case Number</th>
<th>BNB Time (seconds)</th>
<th>BNB Iterations</th>
<th>GBD Time (seconds)</th>
<th>GBD Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.24</td>
<td>4</td>
<td>1.40</td>
<td>2</td>
<td>-13</td>
</tr>
<tr>
<td>2</td>
<td>5.22</td>
<td>249</td>
<td>2.68</td>
<td>4</td>
<td>48.7</td>
</tr>
<tr>
<td>3</td>
<td>22.01</td>
<td>1249</td>
<td>4.81</td>
<td>7</td>
<td>78.1</td>
</tr>
</tbody>
</table>

Example 12.6. Reconsider the actuator selection problems of Example 12.3. A comparison of the GBD algorithm with that of branch-and-bound is given in Table 12.10. The MATLAB code used to generate Table 12.10 is given in Table 12.11.

MATLAB code for Examples 12.3 and 12.6.

```matlab
clear all % Actuator Selection of Furnace Reactor System
% Process Model
nx=5; nu=3; nw=1; Ixx=eye(nx); Iuu=eye(nu);
A=[-8000 0 0 0 10000; 2000 -1500 0 0 0; 0 0 -5000 0 0; 0 0 0 -5000 0; 0 0 0 -5000 0];
B1=[-75; -25; 0; 0; 0]; B2=[75000; 0; 8500; 0; 0];
B3=[0; 0; 8.5e5; -5.0e7; 0]; B=[B1 B2 B3];
G=[0; 0; 0; 0; 10]; Sw=64;
zeta_x_max=[25^2 0.5^2 2.5^2 100^2 50^2];
zeta_u_max=[1025^2 3^2 0.01^2]; c_a0=[8 6 1];
%
--- Case 1 ---
a_size_fact=[1]; na=1;
--- Case 2 ---
a_size_fact=[1 0.5 0.4 0.3 0.2 0.1 0.05]; na=7;
--- Case 3 ---
a_size_fact=[1 0.5 0.45 0.4 0.35 0.3 0.25 0.2 0.15 0.1 0.05 0.025 ... 0.0125 0.005]; na=12;
c_fact=a_size_fact.^(0.6); c_a=[];
for ii=1:nu c_a=[c_a c_fact(1:na) *c_a0(ii)]; end
%
Branch-and-Bound Solution Method
yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; Iuu=eye(nu); tic
for ii=1:nu na delta(ii)=intvar(1,1); end, Z0=sdpvar(nx); Z1=sdpvar(nu,nx);
C1=[Z0 >= 0]; C2=[(A *Z0-B*Z1)+(A*Z0-B*Z1)'+G*Sw*G']<=0;
for ii=1:nx rho_i=Ixx(ii,:); C3=[C3,zeta_x_max(ii) >= rho_i *Z0*rho_i']];
end
for ii=1:nu rho_i=Iuu(ii,:); Sum_delt=0;
    for jj=1:na
        Sum_delt=Sum_delt+delta((ii-1)*na+jj)*zeta_u_max(jj)*a_size_fact(jj);
    end
C4=[C4, [Sum_delt rho_i*Z1; Z1'rho_i' Z0 ]>=0];
end
for ii=1:nu na C5=[C5, 0<=delta(ii)<=1]; end
Constraints=[C1,C2,C3,C4,C5]; Objective=c_a*delta';
options=sdpt3settings('solver','bnb','bnb.solver','mosek',... 'bnb.maxiter',30000,'verbose',1);
MICPsol = optimize(Constraints,Objective,options);
if MICPsol.problem == 0 MICPsol.info, pause, else
    display('BnB Solution'); delta_star=double(delta);
    [delta_star(1:na);delta_star(na+1:2*na);delta_star(2*na+1:3*na)]
    BNBcost=c_a*delta_star', BNBtime=toc, end
```
12.3. GBD Applied to Hardware Selection

% Generalized Benders Decomposition Method
itr=0; Termination=0; chi_k2=[]; lambda_k2=[]; tic;
while Termination == 0
% Relaxed Master Problem
  clear delta_cc; yalmip('clear'); C1=[]; C2=[];
  for ii=1:nu*na delta_cc(ii)=intvar(1,1); end
  for ii=1:nu*na C1=[C1, 0<=delta_cc(ii)<=1]; end
  for ii=1:itr C2=[C2, chi_k2(ii)-lambda_k2(ii,:) *delta_cc <=0]; end
  Constraints=[C1,C2]; Objective=c_a*delta_cc';
  options=sdpsettings('solver','gurobi','verbose',0);
  RM_sol = optimize(Constraints,Objective,options);
  deltacc_bar=double(delta_cc);
  [deltacc_bar(1:na);deltacc_bar(na+1:2*na);deltacc_bar(2*na+1:3*na)]
% Primal Problem
  clear theta delta_sp, yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[];
  Z0=sdpvar(nx); Z1=sdpvar(nu,nx);
  C1=[Z0 >= 0]; C2=[(A*Z0-B*Z1)+(A*Z0-B*Z1)' +G*Sw*G'<=0];
  for ii=1:nx rho_i=Ixx(ii,:);
    C3=[C3,zeta_x_max(ii) >= rho_i *Z0*rho_i'];
  end
  for ii=1:nu rho_i=Iuu(ii,:); Sum_delt=0;
    for jj=1:na
      Sum_delt=Sum_delt+...
        deltacc_bar((ii-1)*na+jj)*zeta_u_max(ii)*a_size_fact(jj);
    end
    C4=[C4, [Sum_delt rho_i*Z1; Z1'*rho_i' Z0] >=0]; end
  Constraints=[C1,C2,C3,C4]; Objective=[];
  options=sdpsettings('solver','mosek','verbose',0);
  PPsol = optimize(Constraints,Objective,options);
  if PPsol.problem == 0 Termination=1; else
% Problem Q
    clear theta delta_sp Z0 Z1, yalmip('clear');
    C1=[]; C2=[]; C3=[]; C4=[]; C5=[];
    theta=sdpvar(1); Z0=sdpvar(nx); Z1=sdpvar(nu,nx);
    for ii=1:nu*na delta_sp(ii)=sdpvar(1,1); end
    for ii=1:nx rho_i=Ixx(ii,:);
      C3=[C3,zeta_x_max(ii) >= rho_i *Z0*rho_i'];
    end
    for ii=1:nu rho_i=Iuu(ii,:); Sum_delt=0;
      for jj=1:na
        Sum_delt=Sum_delt+...
          deltacc_bar((ii-1)*na+jj)*zeta_u_max(ii)*a_size_fact(jj);
      end
      C4=[C4, [Sum_delt rho_i*Z1; Z1'*rho_i' Z0] >=0]; end
    Constraints=[C1,C2,C3,C4,C5]; Objective=theta;
    options=sdpsettings('solver','mosek','verbose',0);
    Qsol = optimize(Constraints,Objective,options);
    if Qsol.problem == 0 Qsol.info, pause,
    else
      itr=itr+1; theta_star=double(theta),
      delta_sp_star=double(delta_sp);
      for ii=1:nu*na lambda_star(ii)=dual(C5(ii)); end
      lambda_k2(itr,:)=lamda_star;
      chi_k2(itr)=lamda_star*delta_sp_star';
    end
  end
end
display('GBD Solution');
[deltacc_bar(1:na);deltacc_bar(na+1:2*na);deltacc_bar(2*na+1:3*na)]
GBDcost=c_a*deltacc_bar, GBDtime=toc, itr,
PercentReduction=100*(BNBtime-GBDtime)/BNBtime
12.4 • Closed-loop Sensor and Actuator Selection

In the section we will develop the simultaneous sensor and actuator selection problem, which can be specialized to arrive at the partial state information actuator selection problem and more importantly the closed-loop sensor selection problem. As a starting point let us restate the full state information actuator selection problem, but with respect to general performance criteria. Following the framework of Section 10.3.2 and combining with the actuator selection notions of Section 12.1.3, we find:

\[
\begin{align*}
\min_{\delta_j \in \{0,1\}} \left\{ \sum_{j=1}^{n_u} c_j^{(a)} \delta_j \right\} \quad \text{s.t.} \\
(A - BL)\Sigma_x + \Sigma_x (A - BL)^* + GS_w G^* = 0 \\
\rho_j (D_x - D_u L)\Sigma_x (D_x - D_u L)^* \rho_j^* = \zeta^{(j)} < \zeta^{(j),max}, j = 1 \ldots n_z \\
\rho_j L \Sigma_x L^* \rho_j^* = \zeta^{(j)} < \delta \zeta^{(j),max}, j = 1 \ldots n_u
\end{align*}
\]

Then, application of Theorem 10.8 yields the following MICP:

\[
\begin{align*}
\min_{\delta_j \in \{0,1\}} \left\{ \sum_{j=1}^{n_u} c_j^{(a)} \delta_j \right\} \quad \text{s.t.} \\
(AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* = 0 \\
\left[ \begin{array}{c}
\zeta^{(j),max} \\
(D_x Z_0 - D_u Z_1)^* \rho_j^* \\
\delta \zeta^{(j),max} \\
Z_1 \rho_j^* \\
Z_0
\end{array} \right] > 0, j = 1 \ldots n_u
\end{align*}
\]

12.4.1 • Actuator Selection with PSI

Let us now consider the actuator selection problem assume that set of sensors are available and they cannot be changed (i.e., the measurement equation \(y = Cx + v\) is given and the spectral density of \(v, S_v\) is fixed). Following the argument of Section 11.3 and combining with the actuator selection notions, we find:

\[
\begin{align*}
\min_{\delta_j \in \{0,1\}} \left\{ \sum_{j=1}^{n_u} c_j^{(a)} \delta_j \right\} \quad \text{s.t.} \\
(A - BL)\Sigma_x + \Sigma_x (A - BL)^* + GS_w G^* = 0 \\
\rho_j (D_x - D_u L)\Sigma_x (D_x - D_u L)^* \rho_j^* + D_x \Sigma_x D_x^* = \zeta^{(j)} < \zeta^{(j),max}, j = 1 \ldots n_z \\
\rho_j L \Sigma_x L^* \rho_j^* = \zeta^{(j)} < \delta \zeta^{(j),max}, j = 1 \ldots n_u
\end{align*}
\]
Again following the logic of Section 11.3, this can be converted to the following MICP:

\[
\min_{\delta_j \in \{0, 1\}} \left\{ \sum_{j=1}^{n_z} c_j^{(s)} \delta_j \right\} \quad \text{s.t.}
\]

\[
(AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + \Sigma_e C^* S_v^{-1} C \Sigma_e < 0
\]  

(12.85)

\[
\begin{bmatrix}
\zeta_j^{(i),\text{max}} - \rho_j D_x \Sigma_e D_x^* \rho_j^* & \rho_j (D_x Z_0 - D_u Z_1) \\
(D_x Z_0 - D_u Z_1)^* \rho_j^* & Z_0
\end{bmatrix} > 0, \quad j = 1 \ldots n_z
\]

(12.86)

\[
\begin{bmatrix}
\delta_j \zeta_j^{(i),\text{max}} \\
Z_1 \rho_j^* \rho_j Z_0
\end{bmatrix} > 0, \quad j = 1 \ldots n_u
\]

(12.87)

It should be emphasized that \(\Sigma_e\) is the estimation error covariance calculated from the ARE associated with the Kalman filter:

\[
A \Sigma_e + \Sigma_e A^* + GS_v G^* - \Sigma_e C^* S_v^{-1} C \Sigma_e = 0
\]

(12.88)

Since \(S_v^{-1}\) is fixed, then \(\Sigma_e\) will be a constant. Thus problem 12.84 will be a proper MICP, since 12.85 will be an LMI.

### 12.4.2 Closed-loop Sensor Selection: A First Attempt

In the closed-loop case, the performance requirements are changed from bounds on the estimation error variances (of the Kalman filter state estimates) to bounds on the variances of the controller performance outputs, \(\zeta^j < \zeta^{(i),\text{max}}, j = 1 \ldots n_z\). Clearly, this new set of bounds are more meaningful physically in the sense that they are related directly to the performance of the closed-loop system:

\[
\min_{\gamma_i \in \{0, 1\}} \left\{ \sum_{i=1}^{n_c} c_i^{(s)} \gamma_i \right\} \quad \text{s.t.} \quad (12.81), (12.82), (12.88)
\]

(12.89)

where \(C^* S_v^{-1} C = \sum_{i=1}^{n_c} \gamma_i \Theta_i, \Theta_i = C^* \rho_i^2 \rho_i C / \sigma_{vi}^2\) and \(\gamma_i\) is the attendance variable. It is highlighted that Equation (12.88) must be included in the formulation, with \(\Sigma_e\) as a variable, because \(S_v^{-1}\) is assumed to be a variable. Unfortunately, when we try to convert problem 12.89 to a MICP, we find that it is impossible. While Equations (12.81)-(12.82) and (12.88) can be converted to LMI s separately, we cannot convert both sets simultaneously. As such, we will need to find an alternate stochastic analysis of the PSI system.

### 12.4.3 Closed-loop Sensor Selection: A Second Attempt

Consider the following compound system:

\[
\begin{bmatrix}
\dot{x}^{(c)} \\
\dot{e}^{(c)}
\end{bmatrix} =
\begin{bmatrix}
A - BL & BL \\
0 & A - KC
\end{bmatrix}
\begin{bmatrix}
x^{(c)} \\
e^{(c)}
\end{bmatrix} +
\begin{bmatrix}
G \\
0
\end{bmatrix}
\begin{bmatrix}
\omega^{(c)} \\
\sigma_{vi}
\end{bmatrix}
\]

(12.90)

with \(x^{(c)} = [x \quad e]^T\), \(\omega^{(c)} = [\omega_{vi} \quad \sigma_{vi}]^T\), \(K = \Sigma_e C^* S_v^{-1}\) and \(\Sigma_e\) as the solution to the ARE, Equation (12.88). To simplify the calculation of \(\Sigma_{x^{(c)}}\), we employ the orthogonality principle...
Thus, closed-loop sensor selection problem can be alternately stated as:

There exists

\[ \text{Theorem 12.2.} \]

can be found in Peng and Chmielewski (2006).

To convert (12.95) to a MICP, the following theorem will be required, the proof of which need to be in the following form

\[ \begin{align*}
0 &= A\Sigma_x + \Sigma_x A^* - BL(\Sigma_x - \Sigma_e) - (\Sigma_x - \Sigma_e)L^*B^* + GS_w G^* \\
\end{align*} \]

and all other blocks are the same as (12.88). In addition, the performance output will need to be in the following form

\[ z = D_x x - D_u L(x - e) \]

Using again the relation \( E[ex^*] = E[ee^*] \), one finds that

\[ \begin{align*}
\Sigma_x &= D_x \Sigma_x D_x - D_x(\Sigma_x - \Sigma_e)L^*D_u^* - D_u L(\Sigma_x - \Sigma_e)D_x^* \\
&\quad+ D_u L(\Sigma_x - \Sigma_e)L^*D_u^* \\
\end{align*} \]

Thus, closed-loop sensor selection problem can be alternately stated as:

\[ \min_{\gamma_i \in \{0,1\}} \sum_{i=1}^{n_1} c_{i}^{(i)} \gamma_i \quad \text{s.t.} \quad (12.88) \]

To convert (12.95) to a MICP, the following theorem will be required, the proof of which can be found in Peng and Chmielewski (2006).

**Theorem 12.2.** There exists \( \Sigma_x \geq 0, \Sigma_e \geq 0 \), and a matrix \( L \) such that (12.88), (12.96), (12.97) hold if and only if there exists \( Z_0 > 0, W_0 > 0, \) and a matrix \( Z_1 \) such that

\[ (AZ_0 - BZ_1) + (AZ_0 - BZ_1)^* + GS_w G^* < 0 \]

\[ \begin{bmatrix} C^* S^{-1} C - W_0 A - A^* W_0 & W_0 G \\ G^* W_0 & S^{-1} \end{bmatrix} > 0 \]

\[ \begin{bmatrix} \begin{pmatrix} D_x Z_0 D_x^* \\
-D_x Z_1^* D_u^* \\
-D_u Z_1 D_x^* \end{pmatrix} & \rho_j(D_u Z_1) \\
0 & \end{pmatrix} > 0, \quad j = 1 \ldots n_z \]
Given Theorem 12.2, the closed-loop sensor selection problem is stated as:

$$\min_{\gamma_i \in \{0, 1\}} \left\{ \sum_{i=1}^{n_s} c_i^{(s)} Y_i \right\} \quad \text{s.t.} \quad (12.98), (12.100)$$

\[
\begin{bmatrix}
\sum_{i=1}^{n_s} Y_i \Theta_i - W_0 A - A^* W_0 & W_0 G
\end{bmatrix} G^* W_0 \Sigma^{-1} > 0
\]

(12.102)

If \(Z_0^*, Z_1^*\) and \(W_0^*\) is the solution to (12.101), then the solution to (12.95) is reconstructed as \(\Sigma_x = Z_0^*\) and \(L = Z_1^* (Z_0^* - (W_0^*)^{-1})^{-1}\).

Table 12.12. Sensor data for Examples 12.7 and 12.8

<table>
<thead>
<tr>
<th>(i)</th>
<th>Type</th>
<th>Location</th>
<th>(\bar{s}_{vi})</th>
<th>(c_i^{(s)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Temperature</td>
<td>(e_1)</td>
<td>0.0625</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>Temperature</td>
<td>(e_1)</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Temperature</td>
<td>(e_2)</td>
<td>0.0625</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Temperature</td>
<td>(e_2)</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Temperature</td>
<td>(e_3)</td>
<td>0.0625</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Temperature</td>
<td>(e_3)</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>(O_2) Concentration</td>
<td>(e_4)</td>
<td>0.1</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>(O_2) Concentration</td>
<td>(e_4)</td>
<td>0.4</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>(CO) Concentration</td>
<td>(e_5)</td>
<td>0.1</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>(CO) Concentration</td>
<td>(e_5)</td>
<td>0.4</td>
<td>8</td>
</tr>
</tbody>
</table>

Example 12.7. Reconsider the pre-heating furnace reactor system of Example 10.5. The system matrices are changed into:

\[
A = \begin{bmatrix}
-10 & 0 & 0 & 0 & 0 \\
8000 & -8000 & 0 & 0 & 0 \\
0 & 2000 & -1500 & 0 & 0 \\
0 & 0 & 0 & -5000 & 0 \\
0 & 0 & 0 & 0 & -5000 \\
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0 & 0 \\
75000 & 0 \\
0 & 0 \\
-8500 & -8.5 \times 10^5 \\
0 & -5.0 \times 10^7 \\
\end{bmatrix}
\]

\[
G = \begin{bmatrix}
10 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]

\[
D_x = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
D_w = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

\[
D_u = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

The state and manipulated vectors are \(s = [T_0 \ T_F \ T_R \ CO_2 \ C_{CO}]^*\) and \(m = [F_{fuel} \ P_v]^*\). The disturbance has a spectral density of \(S_w = 200\). The imposed bounds on the performance output variable variance are \(\bar{\Sigma}_x = [500^2 \ 5^2 \ 7^2 \ 5^2 \ 10^2]\). The data for the
available sensor is given in Table 12.12 and generates a matrix \( C \) equal to 
\[
\begin{bmatrix}
c_1^* & c_1^* & c_2^* & c_2^* & c_3^* & c_3^* & c_4^* & c_4^* & c_5^* & c_5^*
\end{bmatrix},
\]
where \( c_i \) (\( i = 1 \ldots 5 \)) is the \( i \)th row of the \( n_x \) identity matrix. It is assumed each of the 10 sensor options is allowed up to 4 sensors of that type \( (\gamma_i = \{0,1,2,3,4\}) \).

Both branch-and-bound and GBD approaches obtain the same solution, \( \gamma_1 = 3 \) for a total cost of $4.5. A comparison of the GBD algorithm with that of branch-and-bound is given in Table 12.13. The MATLAB code used to generate Table 12.13 is given in Table 12.15.

### Table 12.13. Comparison of GBD and Branch-and-Bound for Example 12.7

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (seconds)</th>
<th>Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch-and-Bound</td>
<td>4.21</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>GBD</td>
<td>3.88</td>
<td>2</td>
<td>7.9%</td>
</tr>
</tbody>
</table>

### Example 12.8

Reconsider the pre-heating furnace reactor system of Example 10.5. The system matrices are changed into:
\[
A = \begin{bmatrix}
-10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8000 & -8000 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2000 & -1500 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -5000 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -5000 & 0 & 0 & 0
\end{bmatrix}
\]
\[
B = \begin{bmatrix}
0 & -75 & 75000 & 0 \\
-25 & 0 & 0 & -8500 \\
0 & -8.5 \times 10^5 & 0 & 0 \\
0 & 0 & -5.0 \times 10^7 & 0
\end{bmatrix}
\]
\[
G = \begin{bmatrix}
10 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]
\[
D_x = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
\[
D_u = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

The state and manipulated vectors are \( s = [T_0 \ T_F \ T_R \ C_{O_2} \ C_{CO}]^* \) and \( m = [F_{feed} \ F_{fuel} \ P_c]^* \). The disturbance has a spectral density of \( S_w = 200 \). The imposed bounds on the performance output variable variance are \( \zeta^{max} = [500^2 \ 5^2 \ 7^2 \ 5^2 \ 10^2] \). The data for the available sensor is given in Table 12.12 and generates a matrix \( C \) equal to 
\[
\begin{bmatrix}
c_1^* & c_1^* & c_2^* & c_2^* & c_3^* & c_3^* & c_4^* & c_4^* & c_5^* & c_5^*
\end{bmatrix},
\]
where \( c_i \) (\( i = 1 \ldots 5 \)) is the \( i \)th row of the \( n_x \) identity matrix. It is assumed each of the 10 sensor options is allowed up to 4 sensors of that type \( (\gamma_i = \{0,1,2,3,4\}) \).

### Table 12.14. Comparison of GBD and Branch-and-Bound for Example 12.8

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (seconds)</th>
<th>Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch-and-Bound</td>
<td>4.88</td>
<td>13</td>
<td>-</td>
</tr>
<tr>
<td>GBD</td>
<td>4.00</td>
<td>2</td>
<td>18.0%</td>
</tr>
</tbody>
</table>

Both branch-and-bound and GBD approaches obtain the same solution, \( \gamma_1 = 2 \) for a total cost of $3. A comparison of the GBD algorithm with that of branch-and-bound is
given in Table 12.14. The MATLAB code used to generate Table 12.14 is given in Table 12.15.

**Table 12.15.** MATLAB code for Examples 12.7 and 12.8.

```matlab
clear all % Sensor Selection of Furnace Reactor System
% Process model
nx=5; nw=1; nz=5; Izz=eye(nz);
A=[-10 0 0 0 0; 8000 -8000 0 0 0; 0 2000 -1500 0 0;
 0 0 -5000 0 0; 0 0 0 -5000 0];
B1=[0 -75 -25 0 0]';
B2=[0 75000 0 -8500 0]';
B3=1e5*[0 0 0 8.5 -500]';
B=[B2 B3]; nu=size(B,2);
% Example 12.7
B=[B1 B2 B3];
% Example 12.8
G=[10; 0; 0; 0; 0]; Sw=2e2;
Dx=eye(nx); Du=zeros(nz,nu);
zeta_z_max=[500^2; 5^2; 7^2; 5^2; 10^2];
gamma_max=4; ny=nx *2; Iyy=eye(ny);
c1=[1 0 0 0 0]; c2=[0 1 0 0 0]; c3=[0 0 1 0 0];
c4=[0 0 0 1 0]; c5=[0 0 0 0 1];
C=[c1; c1; c2; c2; c3; c3; c4; c4; c5; c5];
sigmav2_bar=[0.0625; 0.25; 0.0625; 0.25; 0.0625; 0.25; 0.1; 0.4; 0.1; 0.4];
c_s=[1.5 1 1.5 1 1.5 1 9 6 12 8];
% Branch-and-Bound Solution Method
yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; CSinvC=0; tic;
Z0=sdpvar(nx); W0=sdpvar(nx); Z1=sdpvar(nu,nx);
for ii=1:ny gamma(ii)=intvar(1,1);
end
for ii=1:ny C1=[C1, 0<=gamma(ii)<=gamma_max]; end
C2=[(A*Z0-B*Z1)+(A*Z0-B*Z1)'
     +G*Sw*G']<=0;
for ii=1:nz rho_i=Izz(ii,:);
C3=[C3,[zeta_z_max(ii)-rho_i(Dx*Z0*Dx'
     -Dx*Z1*Dx'*Du*Z1)*rho_i
     zeros(nx,1) eye(nx)
     (Du*Z1)'*rho_i'
     Z0 eye(nx)...
     zeros(nx,1) eye(nx) W0]>0]; end
for ii=1:ny rho_i=Iyy(ii,:);
CSinvC=CSinvC+gamma(ii)*(rho_i*C)
     *(rho_i*C)/sigmav2_bar(ii);
end
C4=[[CSinvC-W0*A-A'
     'G*W0 inv(Sw) >= 0];
C5=[20>0, W0>0];
Constraints=[C1,C2,C3,C4,C5]; Objective=c_s*gamma';
options=sdpsettings('solver','bnb',
     'bnb.solver','mosek',...
     'mosek.MSK_DPAR_INTPNT_CO_TOL_PFEAS', 5e-5,...
     'mosek.MSK_DPAR_INTPNT_CO_TOL_DFEAS', 5e-5,...
     'mosek.MSK_DPAR_INTPNT_CO_TOL_REL_GAP', 5e-5,...
     'bnb.maxiter',30000, 'verbose',1); MICPsol = optimize(Constraints,Objective,options);
if MICPsol.problem == 0 MICPsol.info; pause, else
    display('BnB Solution'); gamma_star=double(gamma),
    BNBcost=c_s*gamma_star, BNBtime=toc, end
% Generalized Benders Decomposition Method
itr=0; Termination=0; chi_k2=[]; lambda_k2=[]; tic;
while Termination == 0
% Relaxed Master Problem
    clear gamma_cc; yalmip('clear'); C1=[]; C2=[];
    for ii=1:ny gamma_cc(ii)=intvar(1,1);
    end
    for ii=1:ny C1=[C1, 0<=gamma_cc(ii)<=gamma_max]; end
    for ii=1:itr C2=[C2, [chi_k2(ii)-lambda_k2(ii,:)*gamma_cc(ii)]<=0]; end
    Constraints=[C1,C2]; Objective=c_s*gamma_cc';
    options=sdpsettings('solver','cplex', 'verbose',0);
    RM_sol = optimize(Constraints,Objective,options);
```
12.4.4 Simultaneous Sensor and Actuator Selection

Combining the closed-loop sensor selection and actuator selection problem, the simultaneous sensor and actuator selection problem is formulated as:

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n_s} c_i(i) \gamma_i + \sum_{j=1}^{n_a} c_j(a) \delta_j \\
\text{s.t.} & \quad (12.103) \\
& \quad A \Sigma_e + \Sigma_e A^e + G S_w G^* - \Sigma_e C^e \Sigma_e^{-1} C \Sigma_e = 0 \\
& \quad 0 = A \Sigma_x + \Sigma_x A^x - BL(\Sigma_x - \Sigma_e) - (\Sigma_x - \Sigma_e) L^x B^* + G S_w G^* \\
& \quad \rho_j \left( D_x \Sigma_x - D_x \Sigma_x - \Sigma_e L^* D_w^* - D_u L(\Sigma_x - \Sigma_e) D^*_x \right) \rho_j^* \\
& \quad + \rho_j D_u L(\Sigma_x - \Sigma_e) L^* D_w^* \rho_j^* = \zeta^j < \zeta^j(max), \, j = 1 \ldots n_x \\
& \quad \rho_j \left( I \Sigma_x - \Sigma_e L^* \right) \rho_j^* = \zeta^h < \zeta^h(max), \, j = 1 \ldots n_u 
\end{align*}
\]

Then, application of a slight modification of Theorem 12.2 results in the following MICP:
(AZ₀ - BZ₁) + (AZ₀ - BZ₁)⁺ + GS_w G⁺ < 0

\[
\begin{bmatrix}
(\zeta^{(j),\text{max}} - \rho_j) \\
-\rho_j(Z_0 G^* I)
\end{bmatrix}
\begin{bmatrix}
D_x Z_0 D_x^* \\
-D_x Z_1 D_u^* \\
-D_u Z_1 D_x^*
\end{bmatrix}
\begin{bmatrix}
\rho_j(D_u Z_1) \\
0
\end{bmatrix} > 0, \ j = 1 \ldots n_z
\]  

(12.98)

\[
\begin{bmatrix}
\sum_{i=1}^{n_y} \gamma_i \Theta_i - W_0 A - A^* W_0 \\
G^* W_0
\end{bmatrix}
\begin{bmatrix}
Z_0 \\
I
\end{bmatrix}
> 0
\]  

(12.100)

\[
\begin{bmatrix}
\delta_j \zeta^{(j),\text{max}} \\
\rho_j Z_1 \\
0
\end{bmatrix}
\begin{bmatrix}
\rho_j^* \\
Z_0 \\
I
\end{bmatrix}
> 0, \ j = 1 \ldots n_u
\]  

(12.106)

**Table 12.16. Sensor data for Examples 12.9**

<table>
<thead>
<tr>
<th>j</th>
<th>Location</th>
<th>ζ^{(j),\text{max}}</th>
<th>ζ^{(a)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b₁</td>
<td>10²</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>b₁</td>
<td>20²</td>
<td>150</td>
</tr>
<tr>
<td>3</td>
<td>b₂</td>
<td>10²</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>b₂</td>
<td>20²</td>
<td>75</td>
</tr>
<tr>
<td>5</td>
<td>b₃</td>
<td>0.1²</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>b₃</td>
<td>0.2²</td>
<td>30</td>
</tr>
</tbody>
</table>

**Example 12.9.** Reconsider the Example 12.8. In addition to the sensor selection, actuator selection is augmented. Specifically, the following vectors \(b₁, b₂\) and \(b₃\) correspond the actuator position of the reactor feed rate, fuel feed rate and vent position, respectively.

\[
b₁ = \begin{bmatrix} 0 \\ -75 \\ -25 \\ 0 \\ 0 \end{bmatrix}, \quad b₂ = \begin{bmatrix} 0 \\ 75000 \\ 0 \\ -8500 \\ 0 \end{bmatrix}, \quad b₃ = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 8.5 \times 10^5 \\ -5.0 \times 10^7 \end{bmatrix}
\]

(12.107)

Data for available actuators is provided in the Table 12.16 and generates a matrix \(B\) equal to \([b₁ \ b₁ \ b₂ \ b₂ \ b₃ \ b₃]\). It is assumed that the disturbance has a spectral density of \(S_w = 100\).

Both branch-and-bound and GBD approaches obtain the same solution: \(δ₃ = 1, γ₁ = 1\) for a total cost of $51.5. A comparison of the GBD algorithm with that of branch-and-bound is given in Table 12.17. The MATLAB code used to generate Table 12.17 is given in Table 12.18.
Table 12.17. Comparison of GBD and Branch-and-Bound for Example 12.9

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (seconds)</th>
<th>Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch-and-Bound</td>
<td>56.14</td>
<td>113</td>
<td>-</td>
</tr>
<tr>
<td>GBD</td>
<td>8.46</td>
<td>3</td>
<td>84.9%</td>
</tr>
</tbody>
</table>

Table 12.18. MATLAB code for Examples 12.9.

```matlab
clear all  % Sensor Selection of Furnace Reactor System
% Process model
nx=5; nu=6; nw=1; nz=5; Iuu=eye(nu); Izz=eye(nz);
A=[-10 0 0 0 0; 8000 -8000 0 0 0; 0 2000 -1500 0 0; 0 0 0 -5000 0; 0 0 0 0 -5000];
B1=[0 -75 -25 0 0]; B2=[0 75000 0 -8500 0]; B3=1e5*[0 0 8.5 -500];
B=[B1 B1 B2 B2 B3 B3];
G=[10; 0; 0; 0; 0]; Sw=100;
Dx=eye(nx); Du=zeros(nz,nu);

zeta_z_max=[500^2; 5^2; 7^2; 5^2; 10^2];
zeta_u_max=[10^2; 20^2; 10^2; 20^2; 0.1^2; 0.2^2];
c_a=[100 150 50 75 20 30]; % actuator cost

% Branch-and-Bound Solution Method
yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; CSinvC=0; tic
Z0=sdpvar(nx); W0=sdpvar(nx); Z1=sdpvar(nu,nx); delta=intvar(1,nu); gamma=intvar(1,ny);
for ii=1:ny C1=[C1, 0<=gamma(ii)<=gamma_max]; end
C2=[Z0>0, W0>0];
C3=[(A*Z0-B*Z1)+(A*Z0-B*Z1)' + G*Sw*G] <= 0;
for ii=1:nz rho_iu=Iuu(ii,:);
    C4=[C4, [zeta_u_max(ii)-rho_iu*Dx*Dx' - (rho_iu*Dx-Dx*Dx')*rho_iu]
        zeros(nu,1), eye(nu), eye(nu), W0] >= 0];
end
for ii=1:ny rho_i=Iyy(ii,:);
    C5=[C5, [delta(ii)*zeta_z_max(ii)-rho_i*Dx*Dx' - (rho_i*Dx-Dx*Dx')*rho_i]
        Z0, eye(nx),
        zeros(nx,1), eye(nx), W0] >= 0];
end
end
if MICPsol.problem ~= 0 MICPsol.info; pause, else
display('BnB Solution'); delta_star=double(delta), gamma_star=double(gamma),
BNBcost=c_t*[delta_star gamma_star], BNBtime=toc, end
```
% Generalized Benders Decomposition Method
itr=0; Termination=0; chi_k2=[]; lambda_k2=[]; tic;
while Termination == 0
% Relaxed Master Problem
clear alpha_cc delta_cc gamma_cc; yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; C6=[]; CSinvC=0;
alpha_cc=intvar(1,N); delta_cc=binvar(1,nu); gamma_cc=intvar(1,ny);
for ii=1:ny C1=[C1, 0<=gamma_cc(ii)<=gamma_max]
end
C2=[alpha_cc==[delta_cc gamma_cc]]
for ii=1:itr C3=[C3, [chi_k2(ii)-lambda_k2(ii,:) *alpha_cc] <=0]
end
Constraints=[C1,C2,C3]; Objective=c_t *alpha_cc;
options=sdpsettings('solver','gurobi','verbose',0);
RM_sol = optimize(Constraints,Objective,options);
deltacc_bar=double(delta_cc); gammacc_bar=double(gamma_cc);

% Primal Problem
clear Z0 Z1 W0, yalmip('clear');
C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; C6=[]; CSinvC=0;
theta=sdpvar(1); Z0=sdpvar(nx); W0=sdpvar(nx); Z1=sdpvar(nu,nx);
alpha_sp=sdpvar(1,N); delta_sp=sdpvar(1,nu); gamma_sp=sdpvar(1,ny);
for ii=1:N C1=[C1, alpha_sp(ii)-alphacc_bar(ii)-theta<=0]
end
C2=[alpha_sp==[delta_sp gamma_sp], Z0>0, W0>0]
C3=[(A*Z0-B*Z1)+(A*Z0-B*Z1)'+G*Sw*G] <=0;
for ii=1:nz rho_i=Izz(ii,);
C4=[C4, [zeta_z_max(ii)-rho_i*(Dx*Z0*Dx'-Dx*Z1*Du'-Du*Z1*Dx')*rho_i'... 
    (Du*Z1)'*rho_i'  Z0  eye(nx);... 
    zeros(nx,1)  eye(nx)  W0]>0]; end
for ii=1:nu rho_iu=Iuu(ii,);
C5=[C5, [delta_sp(ii) *zeta_u_max(ii) rho_iu *Z1 zeros(1,nx);... 
    (rho_iu*Z1)' Z0  eye(nx);... 
    zeros(nx,1)  eye(nx)  W0]>0]; end

% Problem Q
clear theta alpha_sp delta_sp gamma_sp 20 W0 Z1, yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; C6=[]; CSinvC=0;
theta=sdpsdpvar(1); Z0=sdpsdpvar(nx); W0=sdpsdpvar(nx); Z1=sdpsdpvar(nx,nu); W0=sdpsdpvar(nx,nu);
alpha_sp=sdpsdpvar(1,N); delta_sp=sdpsdpvar(1,nu); gamma_sp=sdpsdpvar(1,ny);
for ii=1:N C1=[C1, alpha_sp(ii)-alphacc_bar(ii)-theta<=0]
end
C2=[alpha_sp==[delta_sp gamma_sp], Z0>0, W0>0];
C3=[(A*Z0-B*Z1)+(A*Z0-B*Z1)'+G*Sw*G] <=0;
for ii=1:nz rho_i=Izz(ii,);
C4=[C4, [zeta_z_max(ii)-rho_i*(Dx*Z0*Dx'-Dx*Z1*Du'-Du*Z1*Dx')*rho_i'... 
    (Du*Z1)'*rho_i'  Z0  eye(nx);... 
    zeros(nx,1)  eye(nx)  W0]>0]; end
for ii=1:nu rho_iu=Iuu(ii,);
C5=[C5, [delta_sp(ii) *zeta_u_max(ii) rho_iu*Z1 zeros(1,nx);... 
    (rho_iu*Z1)' Z0  eye(nx);... 
    zeros(nx,1)  eye(nx)  W0]>0]; end

if PPsol.problem == 0 Termination=1; else
% Problem O
clear theta alpha_sp delta_sp gamma_sp 20 W0 Z1, yalmip('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; C6=[]; CSinvC=0;
theta=sdpsdpvar(1); Z0=sdpsdpvar(nx); W0=sdpsdpvar(nx); Z1=sdpsdpvar(nx,nu); W0=sdpsdpvar(nx,nu);
alpha_sp=sdpsdpvar(1,N); delta_sp=sdpsdpvar(1,nu); gamma_sp=sdpsdpvar(1,ny);
for ii=1:N C1=[C1, alpha_sp(ii)-alphacc_bar(ii)-theta<=0]
end
C2=[alpha_sp==[delta_sp gamma_sp], Z0>0, W0>0];
C3=[(A*Z0-B*Z1)+(A*Z0-B*Z1)'+G*Sw*G] <=0;
for ii=1:nz rho_i=Izz(ii,);
C4=[C4, [zeta_z_max(ii)-rho_i*(Dx*Z0*Dx'-Dx*Z1*Du'-Du*Z1*Dx')*rho_i'... 
    (Du*Z1)'*rho_i'  Z0  eye(nx);... 
    zeros(nx,1)  eye(nx)  W0]>0]; end
for ii=1:nu rho_iu=Iuu(ii,);
C5=[C5, [delta_sp(ii) *zeta_u_max(ii) rho_iu*Z1 zeros(1,nx);... 
    (rho_iu*Z1)' Z0  eye(nx);... 
    zeros(nx,1)  eye(nx)  W0]>0]; end
end
12.5 Hardware Selection for Discrete-time Systems

In the discrete-time case with computational delay, the covariance equations of interest to the closed-loop sensor selection problem are:

\[
\Sigma_x = A_d \Sigma_x A_d^* - A_d (\Sigma_x - \Sigma_e^+) L^* B_d^* - B_d L (\Sigma_x - \Sigma_e^+) A_d^* + B_d L (\Sigma_x - \Sigma_e^+) L^* B_d^* + G_d \Sigma_w G_d^* \quad (12.108)
\]

\[
\zeta^{(j)} = \rho_j \left( D_x \Sigma_x D_x^* - D_x (\Sigma_x - \Sigma_e^+ L^* D_a^* - D_a L (\Sigma_x - \Sigma_e^+) D_x^*) \rho_j^* + \rho_j (D_a L (\Sigma_x - \Sigma_e^+) L^* D_a^* + D_a \Sigma_w D_a^*) \rho_j^* \right) < \zeta^{(j),\text{max}}, \quad j = 1 \ldots n_z
\]

\[
\Sigma_e = A_d (\Sigma_e^+ - \Sigma_e) C^*(C \Sigma_e^+ C^* + \Sigma_w)^{-1} C \Sigma_e^+) A_d^* + G_d \Sigma_w G_d^* \quad (12.110)
\]

**Theorem 12.3.** There exists \( \Sigma_x \geq 0, \Sigma_e^+ \geq 0, \) a matrix \( L \) and scalars \( \zeta^{(j)}, j = 1 \ldots n_z \) such that (12.108), (12.109), (12.110) hold if and only if there exists \( Z_0 > 0, W_0 > 0, \) a matrix \( Z_1 \) and scalars \( \mu^{(j)}, j = 1 \ldots n_z \) such that

\[
\begin{bmatrix}
Z_0 - \left( G_d \Sigma_w G_d^* + A_d Z_0 A_d^* \right) (B_d Z_1) & 0 \\
(B_d Z_1)^* & Z_0 & I \\
0 & I & W_0
\end{bmatrix} > 0 \quad (12.111)
\]

\[
\begin{bmatrix}
\mu^{(j)} - \rho_j \left( D_w \Sigma_w D_w^* + D_x Z_0 D_x^* \right) \rho_j^* & \rho_j (D_a Z_1) & 0 \\
(D_a Z_1)^* \rho_j^* & Z_0 & I \\
0 & I & W_0
\end{bmatrix} > 0, \\

j = 1 \ldots n_z \quad (12.112)
\]

\[
\begin{bmatrix}
W_0 & (W_0 A_d) & (W_0 G_d) \\
(W_0 A_d)^* & W_0 + C^* \Sigma_w^{-1} C & 0 \\
(W_0 G_d)^* & 0 & \Sigma_w^{-1}
\end{bmatrix} > 0 \quad (12.113)
\]

and \( \mu^{(j)} < \zeta^{(j),\text{max}}, j = 1 \ldots n_z \)

(12.114)
Given Theorem 12.3, the closed-loop sensor selection problem for discrete-time systems with computational delay is stated as:

\[
\min_{\gamma_i \in \{0, 1\}} \mu(j), Z_0 > 0, W_0 > 0, Z_1 > 0, W_0 \left( \sum_{i=1}^{n_i} c_i \gamma_i \right) \quad \text{s.t.} \quad (12.111), (12.112) (12.114)
\]

(12.115)

\[
\begin{bmatrix}
W_0 & (W_0 A_d) & (W_0 G_d) \\
(W_0 A_d)^T & W_0 + \sum_{i=1}^{n_i} \gamma_i \Theta_i & 0 \\
(W_0 G_d)^T & 0 & \Sigma_w^{-1}
\end{bmatrix} > 0
\]

(12.116)

Example 12.10. Consider the discrete-time case with computational delay of Example 12.7. Application of the sample-and-hold method (\(\Delta t = 10\) seconds) gives:

\[
A_d = \begin{bmatrix}
0.9988 & 0 & 0 & 0 & 0 \\
0.6034 & 0.3962 & 0 & 0 & 0 \\
0.0757 & 0.1368 & 0.8406 & 0 & 0 \\
0 & 0 & 0 & 0.5606 & 0 \\
0 & 0 & 0 & 0 & 0.5606
\end{bmatrix} \quad B_d = \begin{bmatrix}
5.6610 & 0 \\
0.7101 & 0 \\
-0.7469 & 74.6938 \\
0 & -4.3938 \times 10^3
\end{bmatrix}
\]

\[
G_d = \begin{bmatrix}
0.0012 & 0.0004 \\
0.0000 & 0 \\
0 & 0
\end{bmatrix} \quad \Sigma_w = 1.7280 \times 10^6
\]

The imposed bounds on the performance output variable variance, \(\zeta_{z_{max}}\), the cost and the spectral density of the sensors remain unchanged. Again, it is assumed each of the 10 sensor options is allowed up to 4 sensors of that type (\(\gamma_i = \{0, 1, 2, 3, 4\}\)).

Both branch-and-bound and GBD approaches obtain the same solution, \(\gamma_1 = 4\) and \(\gamma_2 = 1\) for a total cost of $7. A comparison of the GBD algorithm with that of branch-and-bound is given in Table 12.19. The MATLAB code used to generate Table 12.19 is given in Table 12.21.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (seconds)</th>
<th>Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch-and-Bound</td>
<td>6.21</td>
<td>25</td>
<td>-</td>
</tr>
<tr>
<td>GBD</td>
<td>5.35</td>
<td>3</td>
<td>13.9%</td>
</tr>
</tbody>
</table>

Table 12.19. Comparison of GBD and Branch-and-Bound for Example 12.10

Example 12.11. Consider the discrete-time case with computational delay of Example
12.8. Application of the sample-and-hold method ($\Delta t = 10$ seconds) gives:

\[
A_d = \begin{bmatrix}
0.9988 & 0 & 0 & 0 & 0 \\
0.6034 & 0.3962 & 0 & 0 & 0 \\
0.0757 & 0.1368 & 0.8406 & 0 & 0 \\
0 & 0 & 0 & 0.5606 & 0 \\
0 & 0 & 0 & 0 & 0.5606 \\
\end{bmatrix}
\]

\[
B_d = \begin{bmatrix}
0 & 0 & 0 \\
-0.0057 & 5.6610 & 0 \\
-0.0034 & 0.7101 & 0 \\
0 & -0.7469 & 74.6938 \\
0 & 0 & -4.3938 \times 10^3 \\
\end{bmatrix}
\]

\[
G_d = \begin{bmatrix}
0.0012 \\
0.0004 \\
0.0000 \\
0 \\
0 \\
\end{bmatrix}
\]

\[
\Sigma_w = 1.7280 \times 10^6
\]

The imposed bounds on the performance output variable variance, $\zeta_z^{\text{max}}$, the cost and the spectral density of the sensors remain unchanged. Again, it is assumed each of the 10 sensor options is allowed up to 4 sensors of that type ($\gamma_i = \{0, 1, 2, 3, 4\}$).

Both branch-and-bound and GBD approaches obtain the same solution, $\gamma_i = 2$ for a total cost of $3$. A comparison of the GBD algorithm with that of branch-and-bound is given in Table 12.20. The MATLAB code used to generate Table 12.20 is given in Table 12.21.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (seconds)</th>
<th>Iterations</th>
<th>Percent Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch-and-Bound</td>
<td>4.33</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>GBD</td>
<td>4.05</td>
<td>2</td>
<td>6.3%</td>
</tr>
</tbody>
</table>

Table 12.20. Comparison of GBD and Branch-and-Bound for Example 12.11

Table 12.21. MATLAB code for Examples 12.10 and 12.11.

clear all  % Sensor Selection of Furnace Reactor System
% Process model
nx=5; nw=1; nz=5; Izz=eye(nz);
A=[-10 0 0 0 0; 8000 -8000 0 0 0; 0 2000 -1500 0 0; 0 0 -5000 0 0; 0 0 0 -5000 0];
B1=[0 -75 -25 0 0]'; B2=[0 75000 0 -8500 0]'; B3=1e5*[0 0 0 8.5 -500]';
% B=[B2 B3]; nu=size(B,2); % Example 12.9
B=[B1 B2 B3]; nu=size(B,2); % Example 12.10
G=[10; 0; 0; 0; 0]; Sw=200;
Dx=eye(nx); Du=zeros(nx,nu); Dw=zeros(nz,nw);
\[\text{zeta}_z^{\text{max}} = \{500^2, 5^2, 7^2, 5^2, 10^2\}\]
\[\text{gamma}_\gamma = \{1.5, 1, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 6, 12, 8\}\]
% Convert to discrete-time model
dt=10/3600/24; % seconds into days
M=[A B G; zeros(nu+nw,nx+nu+nw)];
M=expm(M*dt); Ad=M(1:nx,1:nx); Bd=M(1:nx,nx+1:nx+nu);
12.5. Hardware Selection for Discrete-time Systems

\begin{verbatim}
Gd=M(1:nx,nx+nu+1:nx+nu+nw);
Sigw=Sw/dt; sigmav2_bar=sigmav2_bar/dt;

% Branch-and-Bound Solution Method
yalmp('clear'); C1=[]; C2=[]; C3=[]; C4=[]; C5=[]; C6=[]; CSInvC=0; tic
t0=sdptvar(nx); W0=sdptvar(nx); Z1=sdptvar(nx,nu); mu=sdptvar(nz,1);
for i=1:ny gamma(ii)=intvar(1,1);
end
c2=[];
for i=1:ny C1=[C1, 0<=gamma(ii)<=gamma_max]; end
for i=1:ny if 0<=gamma(ii)<=gamma_max
C2=[[Z0-(Gd* Sigw*Gd' + Ad*Z0*Ad' - Ad*Z1*Bd' - Bd*Z1*Ad') ... 
  (Bd*Z1)*Z0 eye(nu); 
  zeros(nx); eye(nx) W0>=0];
for i=1:ny rho_i=Izz(ii,:);
C3=[C3, [mu(ii)-rho_i *(Dw* Sigw*Dw' + Dx*Z0'*Dx'- ... 
  Dx*Z1*Du'-Du*Z1*Dx')*rho_i' rho_i*(Du*Z1) zeros(1,nx); 
  (Du*Z1)*rho_i' Z0 eye(nx); 
  zeros(nx,1) eye(nx) W0>=0];
C4=[C4, mu(ii)<zeta_z_max(ii)];
end
for i=1:ny rho_i=Iyy(ii,:);
CSinvC=CSinvC+gamma_sp(ii)*(rho_i*C)'*(rho_i*C)/sigmav2_bar(ii); end
C5=[[W0 (W0*Ad) (W0*Gd); 
  (W0*Ad)' W0+CSinvC zeros(nw,nx); 
  (W0*Gd)' zeros(nx,nw) inv(Sigw)]>=0];
C6=[Z0>0, W0>0];
end
Constraints=[C1,C2,C3,C4,C5]; Objective=c_s *gamma';
end
end
% Generalized Benders Decomposition Method
t0=0; while Termination == 0
itr=0; theta=gamma_sp;
theta=gamma_sp;
mu=gamma_sp;
for i=1:ny rho_i=Izz(ii,:);
C3=[C3, [mu(ii)-rho_i*(Dw* Sigw*Dw' + Dx*Z0'*Dx'- ... 
  Dx*Z1*Du'-Du*Z1*Dx')*rho_i' rho_i*(Du*Z1) zeros(1,nx); 
  (Du*Z1)*rho_i' Z0 eye(nx); 
  zeros(nx,1) eye(nx) W0>=0];
C4=[C4, mu(ii)<zeta_z_max(ii)];
end
for i=1:ny rho_i=Iyy(ii,:);
CSinvC=CSinvC+gamma_sp(ii)*(rho_i*C)'*(rho_i*C)/sigmav2_bar(ii); end
C5=[[W0 (W0*Ad) (W0*Gd); 
  (W0*Ad)' W0+CSinvC zeros(nw,nx); 
  (W0*Gd)' zeros(nx,nw) inv(Sigw)]>=0];
end
end

% Relaxed Master Problem
clear gamma_sp; theta=gamma_sp;
mu=gamma_sp;
for i=1:ny rho_i=Izz(ii,:);
C3=[C3, [mu(ii)-rho_i*(Dw* Sigw*Dw' + Dx*Z0'*Dx'- ... 
  Dx*Z1*Du'-Du*Z1*Dx')*rho_i' rho_i*(Du*Z1) zeros(1,nx); 
  (Du*Z1)*rho_i' Z0 eye(nx); 
  zeros(nx,1) eye(nx) W0>=0];
C4=[C4, mu(ii)<zeta_z_max(ii)];
end
for i=1:ny rho_i=Iyy(ii,:);
CSinvC=CSinvC+gamma_sp(ii)*(rho_i*C)'*(rho_i*C)/sigmav2_bar(ii); end
C5=[[W0 (W0*Ad) (W0*Gd); 
  (W0*Ad)' W0+CSinvC zeros(nw,nx); 
  (W0*Gd)' zeros(nx,nw) inv(Sigw)]>=0];
end

\end{verbatim}
(W0*Ad)\' \ W0+CSinvC \ zeros(nx,nw); 
(W0*Gd)\' \ zeros(nw,nx) \ inv(Sigw)\geq0;

C6=[Z0>0, W0\geq0]; 
Constraints=[C1,C2,C3,C4; C5; C6]; Objective=theta; 
options=sdpsettings('solver', 'mosek', 'verbose', 0); 
Qsol = optimize(Constraints,Objective,options); 
if Qsol.problem \neq 0 Qsol.info; pause, else 
itr=itr+1; theta_star=double(theta); gamma_sp_star=double(gamma_sp); 
for ii=1:ny lamda_star(ii)=dual(C1(ii)); end 
lambda_k2(itr,:)=lamda_star; chi_k2(itr)=lamda_star*gamma_sp_star'; 
if theta_star \leq 0 Termination=1; end 
end 
display('GBD Solution'); gammacc_bar, GBDcost=c_s*gammacc_bar', 
GBDtime=toc, itr, PercentReduction=100*(BNBtime-GBDtime)/BNBtime

12.6 • Chapter Summary

This chapter introduced the notion hardware selection. While the attendance variables of a hardware selection problem are subject to non-convex integer constraints, it was shown that the convexity of the remaining constraints was instrumental to achieving a computationally tractable method of finding a global solution. Specifically, one of the YALMIP solvers can be used to implement the branch-and-bound algorithm to solve a MIPC. To achieve greater computational efficiency, the GBD approach was introduced. This solution method will be revisited in Chapter 13, where more general non-convex constraints will be combined with the covariance based controller design to arrive at the Economic Linear Optimal Control (ELOC) design method.

Exercises

12.1. Reproduce the results of Examples 12.2 and 12.5.
12.2. Reproduce the results of Example 12.2 and 12.5, but with the following changes:

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad \tilde{\beta}_p = \begin{bmatrix} 0.25 & 0.1 & 0.8 \end{bmatrix} \quad c^{(i)} = \begin{bmatrix} 15 & 40 & 10 \end{bmatrix}$$

12.3. Convert Exercise 12.1 to the case of a discrete-time optimal estimator with a sample time of 0.2. Determine the new optimal sensor configurations for the three cases. Repeat using the one-step predictor formulation.
Chapter 13

Economic Linear Optimal Control

13.1 · Fixed Controller BOP Selection
13.2 · Economic Linear Optimal Control
13.3 · ELOC Solution Methods
13.3.1 · The Heuristic Solution Method
13.3.2 · ELOC and the Generalized Bender’s Decomposition
13.4 · ELOC with PSI and in Discrete-time
13.5 · Constrained ELOC
13.6 · Chapter Summary
Chapter 15

Controller Integrated Process Design
Bibliography


<table>
<thead>
<tr>
<th>Citation</th>
<th>Title</th>
<th>Journal/Conference</th>
<th>Pages</th>
</tr>
</thead>
</table>


