Computational Analysis of Real-Time Convex Optimization for Control Systems

by

Lawrence Kent McGovern

B.S., Aeronautics and Astronautics, M.I.T., 1994
M.S., Aeronautics and Astronautics, M.I.T., 1996

Submitted to the Department of
Aeronautics and Astronautics
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
at the
Massachusetts Institute of Technology

June, 2000

© 2000 Lawrence K. McGovern. All rights reserved.

The author hereby grants to MIT permission to reproduce and to
distribute copies of this thesis document in whole or in part.

Signature of Author

Department of Aeronautics and Astronautics

May 4, 2000

Certified by

Eric Feron
Assistant Professor of Aeronautics and Astronautics
Thesis Advisor

Brent Appleby
Charles Stark Draper Laboratory
Committee Member

Robert Freund
Professor of Operations Research
Committee Member

James Paduano
Principal Research Engineer, Dept. of Aeronautics and Astronautics
Committee Member

Nesbitt Hagood
Chairman, Departmental Graduate Committee

Certified by

Accepted by
Computational Analysis of Real-Time Convex Optimization for Control Systems

by

Lawrence Kent McGovern

Submitted to the Department of Aeronautics and Astronautics on May 4, 2000 in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Abstract

Computational analysis is fundamental for certification of all real-time control software. Nevertheless, analysis of on-line optimization for control has received little attention to date. On-line software must pass rigorous standards in reliability, requiring that any embedded optimization algorithm possess predictable behavior and bounded run-time guarantees.

This thesis examines the problem of certifying control systems which utilize real-time optimization. A general convex programming framework is used, to which primal-dual path-following algorithms are applied. The set of all optimization problem instances which may arise in an on-line procedure is characterized as a compact parametric set of convex programming problems. A method is given for checking the feasibility and well-posedness of this compact set of problems, providing certification that every problem instance has a solution and can be solved in finite time. The thesis then proposes several algorithm initialization methods, considering the fixed and time-varying constraint cases separately. Computational bounds are provided for both cases.

In the event that the computational requirements cannot be met, several alternatives to on-line optimization are suggested. Of course, these alternatives must provide feasible solutions with minimal real-time computational overhead. Beyond this requirement, these methods approximate the optimal solution as well as possible. The methods explored include robust table look-up, functional approximation of the solution set, and ellipsoidal approximation of the constraint set.

The final part of this thesis examines the coupled behavior of a receding horizon control scheme for constrained linear systems and real-time optimization. The driving requirement is to maintain closed-loop stability, feasibility and well-posedness of the optimal control problem, and bounded iterations for the optimization algorithm. A detailed analysis provides sufficient conditions for meeting these requirements. A realistic example of a small autonomous air vehicle is furnished, showing how a receding horizon control law using real-time optimization can be certified.

Thesis Supervisor: Eric Feron
Title: Associate Professor of Aeronautics and Astronautics

Technical Supervisor: Dr. Brent Appleby
Title: Principal Member of the Technical Staff, Draper Laboratory
Acknowledgments

I owe a great deal of thanks to many people who have contributed to the successful completion of this thesis. It has certainly been a long road, but also a fruitful and enjoyable one due to these people.

Professor Eric Feron undoubtedly deserves the greatest thanks for the guidance he has offered me over the years. He has always been more than generous with his time, even through his own busy schedule. Eric has a marvelous sense for unique problems worth pursuing, and always finds unconventional methods of attacking them. Also, his sense of humor has helped me to keep the right perspective, even if some of that French humor gets lost in translation.

I am indebted to Professor Robert Freund for his invaluable technical input. He has always offered outstanding advice, and many of his tips have turned into important cornerstones of this thesis. He is also one of the finest educators I have encountered during my decade at MIT, both in the lecture hall and in his office. I feel privileged to have benefited from his presence on my thesis committee, and also thank him for his long treks across campus from the operations research center to attend my meetings.

Dr. Brent Appleby has been my advisor and mentor at Draper Laboratory for six years now, and deserves much recognition. He has been a great resource throughout my master's and doctoral work, providing both technical advice and helping me to manage the bureaucracy at Draper and MIT. I don't know how I could have navigated through graduate school without his help.

I also thank Professor James Paduano for his contributions as a member of my thesis committee. His critical questions have helped to make sure my thesis remained tied to reality and relevant to the field of aeronautics. Dr. Nicola Elia also deserves thanks for his input to my general examination, and for many interesting discussions in years past.

Thanks go to Dr. Marc McConley and Dr. William Hall for joining my committee at the very end, and giving my thesis a good critical read. Marc waded through the text with a fine tooth comb, and uncovered many mistakes which I had completely overlooked. And Bill has been my one true resource on optimization at Draper Laboratory, as well as a good friend.

Special thanks to my friend and colleague Chris Dever, who did a nice job editing several chapters in this thesis. We have also enjoyed many lively discussions over food truck food, on bike rides, and at the Kendall Café, which have helped to distract me from the daily toil of student life.

Thanks to Nancy Masley from the graduate office (and from the radio station), for keeping an eye out for me. And a big thank you to the whole crew at WMBR, for an unforgettable five years. Am I ever going to miss that place!
A heartfelt thanks goes to Erika Garlitz, who has given me an immeasurable amount of support over the past year. She has been as good a friend as anyone could ever want. And she will be family in less than a month! I also want to thank my housemates, Joel Sindelar, Rich Diaz, and Nick Levitt, who have definitely improved the quality of my graduate student life.

My final thank you is reserved for my family. Mom & Dad, how could this ever have been possible without your love and support? And thanks too to my dear sister Wendy, and her fiancé (and husband in just a few weeks) James Garlitz. Thank you all, who made this possible.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under IR&D Project #15071. Publication of this thesis does not constitute approval by Draper of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.
Contents

1 On-line Optimization in Feedback Systems ............................................. 13
  1.1 Complexity of Convex Programming ............................................. 17
    1.1.1 Linear Convergence and Iteration Bounds ............................ 20
  1.2 Applications of On-Line Optimization ......................................... 20
    1.2.1 Receding Horizon Control .............................................. 21
    1.2.2 Control Allocation ..................................................... 22
    1.2.3 Reconfigurable Control ............................................... 23
    1.2.4 Combinatorial Applications ......................................... 23
  1.3 Thesis Objectives and Organization ........................................... 24

2 Convex Programming ........................................................................... 27
  2.1 Conic Convex Programming ....................................................... 27
  2.2 Duality ......................................................................................... 30
  2.3 Symmetric Cones and Self-Scaled Barriers .................................... 32
  2.4 The Central Path ........................................................................... 36
  2.5 The Newton Step ........................................................................... 37
    2.5.1 MZ-Family for Semidefinite Programming ............................. 38
  2.6 Path-Following Algorithms ......................................................... 40
  2.7 Homogeneous Self-Dual Embeddings ............................................ 45
    2.7.1 The Homogeneous Self-Dual Model for LP ............................ 47
    2.7.2 The General Homogenous Self-Dual Model ............................ 49
    2.7.3 A Strongly Feasible Self-Dual Embedding ............................. 50

7
3 Parametric Convex Programming 53
   3.1 Linear Fractional Transformations ................................. 53
      3.1.1 A Special Class of Parametric Problems ..................... 56
   3.2 Robustly Feasible Solutions ....................................... 57
   3.3 Condition Measures .................................................. 59
      3.3.1 Structured Distance to Ill-Posedness ......................... 61
      3.3.2 Computation of the Condition Number ......................... 61
      3.3.3 Condition Measures for Sets of Problems ..................... 63
      3.3.4 Largest Condition Number for a Special Class of Parametric Problems 65
   3.4 Infeasibility Detection for Parametric Programs .................. 66
      3.4.1 Infeasibility Detection for a Special Class of Parametric Problems 74

4 Computational Bounds 75
   4.1 Bounds for Problems with Fixed Constraints ........................ 76
      4.1.1 Algorithm Initialization ......................................... 76
      4.1.2 Iteration Bounds ................................................ 79
   4.2 Bounds for Problems with Variable Constraints ....................... 83
      4.2.1 Initialization and Termination ................................... 85
      4.2.2 Iteration Bounds ................................................ 86
   4.3 Warm-Start Strategies ............................................... 103

5 Solution Approximations 107
   5.1 Convex Programming as Set-Valued Functions ....................... 107
   5.2 Table Look-Up via Robust Solutions .................................. 108
   5.3 Functional Solution Approximation .................................. 111
   5.4 Ellipsoidal Constraint Approximation ................................ 116
      5.4.1 Inscribed Ellipsoids ............................................. 119
      5.4.2 Parameterized Ellipsoids ....................................... 125
      5.4.3 Other Ellipsoidal Approximations ............................... 130
   5.5 Discussion ............................................................ 137
6 Application to Receding Horizon Control

6.1 Constrained Linear Systems ...................................... 143
6.2 Well-Posedness of Receding Horizon Control ...................... 146
6.3 Suboptimal RHC and Invariant Sets ............................... 148
6.4 Stability of Suboptimal RHC .................................... 151
  6.4.1 Asymptotic Stability ......................................... 155
  6.4.2 Disturbance Rejection ....................................... 155
6.5 UAV Example ................................................................ 157
  6.5.1 Admissible Set of Initial Conditions ......................... 162
  6.5.2 Ω-Stability ......................................................... 163
  6.5.3 Computational Bounds and Experimental Results ............ 164
  6.5.4 Conclusions ....................................................... 167

7 Concluding Remarks .................................................. 171

7.1 Contributions ........................................................ 173
7.2 Practical Considerations ............................................. 174
7.3 Further Research ..................................................... 175
Nomenclature

\( \mathbb{R} \) = set of real numbers
\( \mathbb{R}^n \) = set of real vectors of length \( n \)
\( \mathbb{R}^{m \times n} \) = set of real \( m \times n \) matrices
\( \mathbb{R}_+, \mathbb{R}_{++} \) = nonnegative and positive orthant
\( \mathbb{R}^+_Q, \mathbb{R}^{++}_Q \) = the second-order cone (for \( n \geq 2 \))
\( \mathbb{S}^n \) = the set of real symmetric \( n \times n \) matrices
\( \mathbb{S}_+^n, \mathbb{S}_{++}^n \) = the set of positive semidefinite and positive definite \( n \times n \) matrices
\( \mathbb{E} \) = Hilbert space for conic program (see page 27)
\( A \succeq B \) = \( A-B \) positive semidefinite, i.e., \( x^T(A-B)x \geq 0 \) for all \( x \in \mathbb{R}^n \)
\( A \succ B \) = \( A-B \) positive definite, i.e., \( x^T(A-B)x > 0 \) for all \( x \in \mathbb{R}^n \)
\( (\gamma, x) \succeq_0 \) = \( (\gamma, x) \in \mathbb{R}^{n+1}_Q \), meaning \( \gamma \geq (\sum_{i=1}^n x_i^2)^{1/2} \)
\( (\gamma, x) \succ_0 \) = \( (\gamma, x) \in \mathbb{R}^{n+1}_{Q++} \), meaning \( \gamma > (\sum_{i=1}^n x_i^2)^{1/2} \)
\( I \) = identity matrix (dimension clear from context)
\( e \) = vector of ones or \( 2.71828... \) (use clear from context)
\( (\cdot)^T \) = transpose
\( \text{det}(\cdot) \) = matrix determinant
\( \text{Det}(\cdot) \) = generalized determinant (see page 34)
\( A^+ \) = Moore-Penrose inverse
\( \lambda_{\text{min}}(\cdot), \lambda_{\text{max}}(\cdot) \) = minimum and maximum eigenvalue of a matrix
\( \sigma_{\text{min}}(\cdot), \sigma_{\text{max}}(\cdot) \) = minimum and maximum singular value of a matrix
\( \text{Tr}(\cdot) \) = trace of a matrix
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log(\cdot) )</td>
<td>natural logarithm (base ( e ))</td>
</tr>
<tr>
<td>( | \cdot | )</td>
<td>Euclidean vector norm, i.e., ( |a| = \sqrt{a^T a} )</td>
</tr>
<tr>
<td>( | \cdot |_H )</td>
<td>vector norm induced by ( H \succeq 0 ), i.e., ( |x|_H = \sqrt{x^T H x} )</td>
</tr>
<tr>
<td>( | \cdot |_\infty )</td>
<td>vector infinity norm, i.e., ( |a|_\infty = \max_i</td>
</tr>
<tr>
<td>( | \cdot |_F )</td>
<td>Frobenius matrix norm, i.e., ( |A|_F = \sqrt{\text{Tr} A^T A} )</td>
</tr>
<tr>
<td>( O(\cdot) )</td>
<td>complexity order notation</td>
</tr>
<tr>
<td>( \nabla B(\cdot) )</td>
<td>gradient of ( B(\cdot) )</td>
</tr>
<tr>
<td>( \nabla^2 B(\cdot) )</td>
<td>Hessian of ( B(\cdot) )</td>
</tr>
<tr>
<td>( \mathcal{X}, \mathcal{Y}, \mathcal{Z}, \ldots )</td>
<td>set</td>
</tr>
<tr>
<td>( \text{int } \mathcal{X} )</td>
<td>interior of ( \mathcal{X} )</td>
</tr>
<tr>
<td>( \mathcal{X}, \mathcal{Y}, \mathcal{Z}, \ldots )</td>
<td>element of Hilbert space ( \mathcal{E} )</td>
</tr>
<tr>
<td>( \mathcal{X} \circ \mathcal{Y} )</td>
<td>Jordan product</td>
</tr>
</tbody>
</table>
Chapter 1

On-line Optimization in Feedback Systems

The fields of optimization and control theory have long been interrelated. Quantitative performance criteria, such as the system $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norms, provide a measure of optimality for a control system. Coupled with the problem constraints, such as the system dynamics or modelling uncertainties, the control problem reveals itself as a constrained optimization problem. Over the last four decades, numerous controller synthesis techniques have emerged to solve these problems, a notable example being the linear quadratic Gaussian (LQG) optimal control problem [Mac89].

For some control problems, numerical optimization may be bypassed, and the best solution found analytically. This is the case for the LQG problem, in which a linear controller is derived from the solution to a set of algebraic Riccati equations. However, for slightly more general problems, an analytical solution is impossible, making numerical optimization a necessity. Historically, these problems have not received as much attention because the computational requirements can be prohibitive. Over the years, some researchers have pointed out that numerical optimization problems with convex formulations are becoming tractable due to advancements in computational technology and convex programming algorithms. Early work on control design using numerical optimization can be found in the publications of Fegley et al. [FBB+71], who used linear and quadratic programming. Subsequent work based on optimization of the Youla pa-
parameterization has been done by Polak [PS89] and Boyd [BBB+88]. Linear programming has been the central tool of the recently developed $\ell_1$ control theory [DDB94]. Finally, semidefinite programming has opened up a wide range of possibilities in system and control theory [BEFB94].

These recent control applications of mathematical programming can be seen as a direct offshoot of advances in computer technology. Problems which could not be realistically considered forty years ago are routinely solved today. The available computational resources will only continue to improve, so it makes sense to begin looking for the next step in taking advantage of these resources for control.

For most optimal control methodologies, including those mentioned above, the optimization phase is done entirely off-line. Because optimization tends to be fairly demanding computationally, on-line optimization has traditionally been viewed as impractical, especially for digital control systems which operate at relatively high frequencies. This aversion to on-line optimization has slowly been changing over the years, due to improved computational capabilities. Already, the chemical process industry has begun to embrace on-line optimization in the context of receding horizon control [GPM89]. On-line methods allow the controller to directly incorporate hard system constraints such as actuator saturation, as well as treat future commanded inputs in an optimal manner. These methods also enable the controller to adapt to unforeseen changes that may take place in the system or constraints. The end product is a less conservative, more aggressive, and more flexible control system than could be designed off-line. The reason on-line optimization has been a practical tool for the chemical process industry is because the system dynamics of interest usually have fairly slow time constants, giving a computer the needed time to solve an optimal control problem in real time. In principle, given a fast enough computer, the techniques employed on these systems could be applied to a system of any speed.

In the near future, on-line optimization will become much more prevalent on faster systems. Of particular interest are aerospace systems, where high performance is desirable in the face of hard system constraints. The 1990s witnessed many potential uses for on-line optimization in the aerospace industry, from self-designing control systems for
high-performance aircraft to air traffic control. There is even a recent case where a linear programming algorithm was implemented on-line as part of an F-16 flight control system [MWBB97]. This example and several other proposed aerospace applications are discussed in Section 1.2.

While there is a wealth of literature on off-line optimization, the subject of on-line optimization is relatively unexplored. Very little research has been invested into determining when it should be applied, how it should be applied, and the potential pitfalls. Philosophically, an on-line optimization algorithm is merely an information storage and retrieval mechanism, and therefore falls into the same category as look-up table or function evaluation. For any method of information retrieval, there is an inherent trade-off between accuracy of the information returned and the memory and computations required to find that information. The decision to apply on-line optimization over some other method is dictated by the required accuracy and available processing power. For control systems, solution accuracy impacts closed-loop stability and performance. Linear feedback is efficient computationally, but may encounter loss in performance or stability when faced with system nonlinearities. Gain scheduling, a mixture of linear feedback and table look-up, is an improvement, yet carries a greater burden on memory, especially if the table dimension is large. Representing a control law using a mathematical program may offer the greatest amount of potential accuracy, but at much higher costs in computation. Understanding these computational costs is important to evaluating the appropriateness of an optimization algorithm for a particular application.

Computational analysis is fundamental for certification of on-line algorithms, yet to date has received little attention. There is a tendency among those who propose on-line optimization strategies to assume that the optimization problem can always be fully solved in the allotted time. However, this assumption cannot be taken on faith, since its violation can lead to disastrous consequences for the on-line procedure. To illustrate the importance of certification, consider the scenario where a linear program is embedded in a flight control system, and must be solved every second. Suppose next that at a particular time instance, the linear programming algorithm is presented with an ill-posed problem, and is not able to find a feasible solution within the allotted second. If this scenario has
not been anticipated, it could lead to failure of the flight control system, or worse yet, failure of the aircraft. Another conceivable scenario is that the optimization algorithm would return a feasible solution on schedule, but one which is far from optimality. In Chapter 6 of this thesis, it will be seen that this, too, can lead to instability.

Certification means providing a guarantee that the optimization algorithm will deliver the desired solution by the appropriate deadline. By nature, optimization algorithms tend to vary in performance from problem to problem, and can be fairly unpredictable with few general guarantees. Even the best performing algorithms can run into unforeseen difficulties, and frequently an algorithm must be tuned to a particular problem. On the other hand, on-line software, especially flight code, must pass rigorous standards in reliability [Lev95]. In particular, on-line software must have a deterministic run time. There is a general reluctance in industry to implement iterative algorithms on-line. If an upper bound on the number of iterations is known in advance, then a \textit{while} loop can be replaced with a \textit{for} loop, which is much more acceptable for on-line code. The most important trait an optimization algorithm must possess to meet this requirement is predictability. It is for this reason that conservative, predictable algorithms are preferred in this thesis over the more aggressive algorithms which tend to be used for off-line applications.

Determining the reliability of an on-line optimization procedure requires the ability to analyze sets of optimization problems, rather than unique problem instances. Of primary importance to this analysis is feasibility. Before implementation of an on-line procedure, the question must be asked: will every problem encountered be feasible? If the answer is no, then this indicates potential failure of the on-line method. Alternatively, it is important to determine whether there exist problem instances for which the objective function is unbounded, which can lead to an infinite computation time for an optimization algorithm. Once it is verified that every problem instance in a set is feasible, a computational upper bound must be determined. Depending on the processing power available to the application, this bound either can or cannot be met in the scheduled time. This is the \textit{only} way in which an optimization algorithm can be certified for an on-line application.

The primary goal of this thesis is to develop the tools necessary to certify on-line
optimization using computational analysis. A careful computational analysis requires a clear understanding of how the optimization algorithm should be applied. This includes understanding how to initialize the algorithm intelligently, the theoretical worst-case convergence, and the accuracy at which the algorithm should terminate.

If stability and performance are taken as the driving requirements for the on-line optimization procedure, then certification must consider the optimization algorithm and dynamic system as a whole. While initialization and convergence of the algorithm can be considered separately from the dynamic system, algorithm accuracy cannot, since the solution accuracy has a strong influence on the performance of the control system. At the very minimum, the controller should have enough accuracy to guarantee closed-loop stability. Combining this requirement with a proper initialization of the algorithm and a knowledge of the algorithm's rate of convergence, one can find the computational bounds necessary to certify an on-line optimization algorithm.

Understanding worst-case algorithm convergence requires a basic knowledge of complexity theory. The next section gives some background on algorithm complexity, which motivates the choice for the algorithms analyzed in this thesis.

### 1.1 Complexity of Convex Programming

The appropriate metric for comparison of an algorithm's worst-case performance comes from complexity theory. An upper bound on the complexity of an algorithm is indicated using the order notation $O(\cdot)$. Given two sequences of scalars $\{N_n\}$ and $\{S_n\}$, the notation

$$N_n = O(S_n)$$

indicates that

$$N_n \leq CS_n$$

for some positive constant $C$ and for all $n$ sufficiently large. In the case of optimization algorithms, $N_n$ usually refers to the number of computations or iterations required of the
Algorithm for a problem of dimension \( n \), and \( S_n \) is some function of \( n \). Complexity theory generally distinguishes between two types of algorithms, polynomial and exponential. An algorithm is considered polynomial if \( S_n \) is a polynomial of \( n \) (e.g., \( n^p \)) and exponential if \( S_n \) is an exponential function (e.g., \( p^n \)). Since exponential functions grow at an enormously fast rate, exponential algorithms are intractable on the worst-case problem except when the dimension is very small.

Worst-case analysis is an important tool for certification. Since on-line algorithms must at least be tractable on the worst-case problem instance, only polynomial time algorithms are considered in this thesis. For the most part, this restricts the types of problems which may be considered to convex programs, although there are a few nonconvex problems which can be solved in polynomial time (e.g., many problems in network optimization have this property).

Complexity considerations also restrict the attention of this thesis to interior-point methods. The seminal work of Nesterov and Nemirovskii [NN94] has given polynomial complexity results for interior-point methods in a general convex programming framework. In practice, interior-point methods are also competitive with other optimization methods, and have been observed to outperform other commonly used methods (e.g., the active set method) as the problem dimension increases [GB98]. Due to this practical value, interior-point methods have been proposed for receding horizon control by several authors [GB98, Han00, RWR98].

Currently, there is no known algorithm for which the complexity of finding an exact solution to a convex program is polynomial in the problem dimension, even in the case of linear programming. Many complexity results for linear programming show a dependence on the bit-size of the problem. Alternatively, Renegar [Ren95a] has shown that the complexity can also be related to the condition number of the linear program (condition numbers for convex programs are analogous to matrix condition numbers, discussed more in Chapter 3).

Rather than determining the complexity of finding an exact solution to a convex programming problem, it is reasonable to consider the complexity of finding a solution which is within a tolerance \( \epsilon \) of optimality. Through the use of interior-point algorithms,
an $\epsilon$-optimal solution can be determined with polynomial complexity in the problem dimension and conditioning.

Interestingly, the most celebrated convex optimization algorithm, the simplex method for linear programming, is an exponential algorithm. This was noted by Klee and Minty [KM72], who discovered a now famous example requiring $2^n$ iterations to solve a linear program in $\mathbb{R}^n$. This is in spite of the fact that the simplex algorithm performs exceptionally well on most problems. Similarly, other boundary-traversing convex programming algorithms such as active set methods for quadratic programming also have exponential complexity in spite of good practical performance [Fle87]. The first polynomial-time algorithm was the ellipsoid algorithm, discovered in 1979 by Khachiyan [Kha79]. Unfortunately, this algorithm performs on average as slow as its theoretical worst case, and is not competitive with the simplex method on even the smallest problems.

Karmarkar was the first to propose a practical polynomial-time algorithm for linear programming [Kar84], which has an $O(n \log(1/\epsilon))$ iteration bound. Karmarkar’s method is the precursor to most current interior-point algorithms, the best of which have an $O(\sqrt{n} \log(1/\epsilon))$ complexity. The main distinction between Karmarkar’s method and modern interior-point algorithms is that most of today’s algorithms consider the primal and dual problems simultaneously, whereas Karmarkar’s method is primal-only. It is to this primal-dual formulation that the best interior-point methods owe their efficiency.

Since worst-case performance has great relevance to real-time optimization, the interior-point algorithms used in this thesis are those with the best known complexity bound $O(\sqrt{n} \log(1/\epsilon))$, rather than the algorithms that tend to perform best in practice. The $O(\sqrt{n} \log(1/\epsilon))$ algorithms tend to take short, conservative steps towards the optimal solution, and employ very few heuristics. Ironically, many of the algorithms which tend to perform better by taking much more aggressive steps towards the optimal solution actually have larger complexity values (e.g., the long-step algorithm is $O(n \log(1/\epsilon))$), and some of the best performing algorithms have no guarantees whatsoever. It is the contention of this thesis that these “more aggressive” interior-point methods are less suitable for on-line optimization, not only because of their larger complexity bounds, but also because they may be less predictable due to their heuristics.
1.1.1 Linear Convergence and Iteration Bounds

The algorithms discussed in the next chapter belong to a class known as path-following algorithms. Bounds in the number of iterations follow by proving a certain rate of linear convergence towards the optimal solution. The algorithms examined here are primal-dual, meaning the degree of optimality is measured by the duality gap \( \mu \geq 0 \), where \( \mu = 0 \) indicates primal-dual optimality. The best known convergence bounds satisfy

\[
\mu_{k+1} \leq \left( 1 - \frac{\Delta}{\sqrt{n}} \right) \mu_k
\]

where \( \mu_k \) is the duality gap at iteration \( k \), \( \Delta > 0 \) is a constant, and \( n \) is the problem dimension. Given this linear convergence rate, an objective \( \mu_k \leq \epsilon \mu_0 \) is satisfied for all

\[
k \geq K \triangleq \left\lceil \frac{\log(\epsilon)}{\log(1 - \Delta/\sqrt{n})} \right\rceil.
\]

Noting that \( \log(1 + \beta) \leq \beta \), it follows that

\[
K \leq \left\lceil \frac{\sqrt{n}}{\Delta} \log(1/\epsilon) \right\rceil
\]

therefore, \( K = \mathcal{O}(\sqrt{n} \log(1/\epsilon)) \).

Despite the fact that the predicted number of iterations is polynomial, it can still be very large when compared to actual algorithm performance. This is because the value of \( (1 - \Delta/\sqrt{n}) \) can be very close to 1. In reality, the ratio \( \mu_{k+1}/\mu_k \) is frequently much smaller than its theoretical bound. For some of the more aggressive algorithms, the ratio is often about 0.1. Convergence of the path-following algorithms used in this thesis is not quite that good, but still tends to be better than its theoretical worst case. It may be possible that smaller guaranteed ratios for \( \mu_{k+1}/\mu_k \) can be proved for particular problem instances, which would result in much more realistic iteration bounds.

1.2 Applications of On-Line Optimization

The potential applications of on-line optimization are virtually limitless. A few motivating examples are given in this section to show the importance of on-line optimization for control systems. These examples are real engineering control methodologies which
1.2. APPLICATIONS OF ON-LINE OPTIMIZATION

require on-line optimization, and the success of these methods depends on the delivery of the solution to a convex program at regularly scheduled time intervals.

1.2.1 Receding Horizon Control

At present, receding horizon control is the most widely used on-line optimization control method in industry. It has been the accepted standard for constrained multivariable problems by the chemical process industry for over two decades. Its strength is an ability to directly incorporate state and input constraints into the design, as well as future command inputs. In its most typical form, receding horizon control is solved as a linear or quadratic program, although recent developments in convex programming have allowed researchers to consider more sophisticated formulations. For example, control laws which are robust to modeling uncertainty are presented in [KBM96] using semidefinite programming and in [BCH98] using second-order cone programming. A more standard formulation is given below.

Given a linear time invariant system, a typical problem in receding horizon control is posed as the open-loop optimal control problem

$$\begin{align*}
\min & \quad \sum_{k=0}^{N-1} ((y_k - r_k)^T Q (y_k - r_k) + u_k^T R u_k) + x_N P x_N \\
\text{subject to} & \quad x_{k+1} = A x_k + B u_k \\
& \quad y_k = C x_k + D u_k \\
& \quad y_{\text{min}} \leq y_k \leq y_{\text{max}} \\
& \quad u_{\text{min}} \leq u_k \leq u_{\text{max}}
\end{align*}$$

where $r_k$ is a target trajectory for output $y_k$, and $P$, $Q$, and $R$ are weighting matrices. This problem is solved on-line as a quadratic program. Only the first control input $u_0$ is implemented, and the problem is resolved at the next time step.

The drawback to receding horizon control is the relatively high computational effort when compared to linear feedback control laws. This has traditionally limited the use of receding horizon control to the slow dynamic systems typically encountered in the chemical process industry. However, the range of applications is beginning to expand as greater computing power becomes available. In Chapter 6, the subject of receding
horizon control is explored in greater depth, and applied to a higher bandwidth aerospace example.

1.2.2 Control Allocation

As aircraft become increasingly sophisticated, the number of moment generating actuators increases far beyond the conventional number of three independent actuators. For example, the F-18 HARV (High Angle-of-attack Research Vehicle) has rudders, horizontal tails, ailerons, leading edge and trailing edge flaps (each left and right), as well as three thrust vectoring moment generators, for a total of 13 independent moment inputs. Current proposals for tailless aircraft show the possibility of up to 20 independent actuators. These systems are clearly overactuated, yet are subject to saturation constraints on the control surfaces. The control allocation problem is motivated by the desire to choose the optimal configuration of the actuators to meet a specified objective, subject to saturation constraints. Furthermore, it allows the possibility of mitigating actuator failure. An introduction to the control allocation problem is found in [Dur94, Enn98]. An application of control allocation to a tailless fighter aircraft is given in [Buf97].

The standard form for the control allocation problem follows below. The linear equation \( d = Bu \) maps the control input \( u \in \mathbb{R}^n \) to a moment \( d \in \mathbb{R}^m \) via the matrix \( B \), derived from the linearized dynamics of the aircraft. A set of constraints \( u_{\text{min}} \leq u \leq u_{\text{max}} \) defines the saturation constraints of the actuators. The objective is to achieve a desired moment \( d_{\text{des}} \). If \( d_{\text{des}} \) is not feasible, a weighted 2-norm of the error may be minimized instead, i.e., \( \min e^T We \) where \( e = d_{\text{des}} - Bu \) and \( W \) is a symmetric, positive definite matrix. Alternatively if \( d_{\text{des}} \) is feasible, then a weighted distance from \( u \) to a preferred control \( u_p \) may be minimized, i.e., \( \min (u - u_p)^T W (u - u_p) \).

As presented, the control allocation problem is an on-line quadratic program. However, control allocation is generally not implemented as a quadratic program due to the inherent difficulty in certification of on-line optimization. A number of alternatives are given in [Enn98]. One recently proposed strategy is to approximate the feasible space by an ellipsoid [OJMF99]. Chapter 5 explores alternatives to on-line optimization similar to this.
1.2.3 Reconfigurable Control

Over time, the closed-loop dynamics of a system are capable of changing in unexpected ways. Changes may happen either gradually or immediately, as in the case of actuator failure. The term “reconfigurable control” simply refers to any control system capable of redesigning itself to meet a new set of dynamics. It usually involves some form of on-line system identification, coupled with a control redesign procedure. Reconfigurable control has been of particular interest to high-performance fighter aircraft, for which an unanticipated change in the open-loop dynamics can have disastrous consequences. In [PCM95], a basic reconfigurable control strategy is proposed for aerospace applications.

In 1996, a reconfigurable control law was successfully flight tested on the VISTA/F-16 by Barron Associates, Inc. [MWBB97]. Part of this flight test included a landing of the aircraft in crosswind conditions with a simulated missing left horizontal tail. The control system included a parameter identification algorithm which identified the stability and control derivatives on-line, in concert with a receding horizon control algorithm based on linear programming. Since then, similar strategies have been suggested for tailless aircraft as part of the Reconfigurable Systems for Tailless Fighter Aircraft (RESTORE) program at Lockheed Martin [EW99] and for Vertical Take-Off/Landing (VTOL) Uninhabited Air Vehicle (UAV) automated shipboard landing [WMS99].

1.2.4 Combinatorial Applications

It might seem limiting that this thesis only considers convex optimization, since many interesting on-line applications are nonconvex in nature. Combinatorial problems are abundant in the real world, most of which fall into the class of NP-hard problems (a class of problems which do not appear to be solvable in polynomial time). The usual approach to attacking these problems is using heuristics, which carry little if any guarantees. Fortunately, many of these problems have convenient convex relaxations, and thus can be attacked using convex programming. Convex relaxations can be derived using Lagrangian duality, since the dual of the dual is always convex. Combinatorial problems involving constraints such as $x \in \{0,1\}$ are frequently replaced with quadratic con-
straints \(x^2 - x = 0\), which have semidefinite programming relaxations [LO99]. Many combinatorial and quadratic problems with good semidefinite relaxations have been noted in the literature, including MAX-CUT, graph coloring, and quadratic maximization [GW95, KMS98, Zha98, Ali95]. There are diverse real-world applications of these problems, from circuit design to cellular phone channel assignment.

One relevant aerospace application is aircraft conflict detection and resolution. As the skies become increasingly congested with aircraft, air traffic controllers face the difficult task of rerouting traffic such that aircraft remain separated by a minimum distance. Given a set of aircraft positions and trajectories, one would ideally like to find the minimum change in trajectory necessary for a conflict-free path. This problem is combinatorial because, for example, each aircraft must decide whether to steer to the left or right of each other aircraft. In [FMOF00], this problem is posed as a nonconvex quadratically-constrained quadratic program, and the convex relaxation is proposed as an on-line optimization technique.

1.3 Thesis Objectives and Organization

The ultimate goal of this thesis is to show that optimization can be certified for on-line applications. Certification is necessary for any software which must run in real time, but becomes especially important for aerospace applications, where software failure can mean loss of the vehicle and/or loss of life. For this reason, the emphasis of this thesis is about what can be proved about an on-line algorithm, rather than what can be inferred from simulation. The guarantees given in this thesis are conservative by nature, and reflect the current gap between theory and practice which exists in the field of optimization.

Chapter 2 introduces much of the relevant background material on convex optimization. Convex optimization is presented in a fairly general form, as conic convex programming over the class of symmetric cones. This allows the contents of this thesis to be applied to linear programming, second-order cone programming (to which convex quadratic programming belongs), and semidefinite programming. Chapter 2 focuses on one class of interior-point algorithms: primal-dual path-following algorithms. These al-
1.3. **THESIS OBJECTIVES AND ORGANIZATION**

Algorithms are used because they have a very predictable behavior, and they have the best known complexity bounds. These two qualities are of high importance to safety-critical on-line systems. Finally, homogeneous self-dual programming is described, which is very reliable and easy to initialize. Homogeneous self-dual programming will become useful in subsequent chapters.

On-line optimization problems are typically parameter dependent problems, in which the parameter changes from instance to instance, but always belongs to some bounded set. Chapter 3 takes the important step of placing convex programming in a parameter dependent framework. These problems are parameterized using linear fractional transformations, which are seen as fairly general means of representing rational functions. Using this parameterization, solutions which are robustly feasible to parameter variations are introduced. Background material is given for the convex programming condition number, which gives a normalized measure of the proximity of a problem to infeasibility. Both condition numbers and robustly feasible solutions are used in later sections. In the last part of this chapter, a methodology is presented for checking whether every problem in a parametric space is feasible. This technique uses a branch-and-bound technique, although it is seen that checking feasibility can be made considerably easier if the parameterization has a certain form.

The heart of the thesis is Chapter 4, where computational bounds are given for parameter-dependent convex programming. In the first part, the constraints are considered to be fixed, and only the objective function is free to vary. An initialization strategy is proposed for this problem, and corresponding computational bounds derived. The second part explores the case where both constraints and objective are time-varying. Both the big-$M$ method (a well-known, basic infeasible start method) and homogeneous self-dual programming are explored for initialization. The analysis focuses on homogeneous self-dual programming, since it tends to be the more efficient means of initialization. A branch-and-bound technique is used for deriving computational bounds.

On-line optimization can be computationally demanding. On top of that, the computational bounds derived in Chapter 4 tend to be conservative. For the instances when the computational demands of on-line optimization become too great, it is desirable to find
cheap alternatives. Chapter 5 explores several options for approximating the solution to a parametric convex program. A few of the obvious choices of solution approximation are explored, such as look-up tables and function approximation. A fair portion of this chapter is devoted to approximating a set of constraints with an ellipsoid. Because of the relative ease of optimizing a linear objective over an ellipsoid, this approximation strategy is very simple to implement as part of an on-line algorithm.

Chapter 6 is devoted to the most common control application of on-line optimization: receding horizon control. The most essential certification requirement of any on-line control algorithm is stability. For receding horizon control, the necessary theory is developed to show that on-line optimization can stabilize constrained linear systems. This result goes beyond the traditional assumption for receding horizon control: that an on-line optimization algorithm exists which delivers an optimal solution at the scheduled time. This stability requirement drives the minimum accuracy which must be requested of the optimization algorithm, ultimately resulting in a computational bound for the algorithm.

The final part of Chapter 6 demonstrates real-time optimization on a realistic control example. A receding horizon control approach is proposed for an uninhabited air vehicle (UAV) example, where the actuator saturation is significant enough to induce closed-loop instability. Computational bounds are derived for the on-line scheme, providing a means of certifying this control approach.

A summary of the technical contributions of this thesis is found in Chapter 7, but there is one contribution which is more philosophical that needs to be emphasized. There is currently a gap between the optimization and control communities which frequently goes unnoticed. Control engineers tend to avoid applying on-line optimization strategies out of a mistrust of iterative algorithms, which are seen as unpredictable. Operations researchers often approach optimization from an off-line perspective, devising heuristics which seem to improve performance of the algorithms, yet do not address the concerns of on-line optimization. Complexity theory has opened the door to certification of these algorithms for on-line use. The better this message is understood, the more likely on-line optimization will be trusted and used in future applications.
Chapter 2

Convex Programming

The term convex programming refers to any optimization problem \( \min_{x \in \mathcal{X}} f(x) \) in which the objective \( f(x) \) is a convex function and the constraint set \( \mathcal{X} \) is a convex set. Convexity is of great importance to optimization for many reasons, the most important being that any local optimum is immediately known to be globally optimal. In Chapter 1, it was stated that from a complexity point of view, the most efficient known algorithms for convex programming are primal-dual interior-point algorithms. This chapter provides the details on a few interior-point algorithms, and also presents some of the supporting theory needed for development of these algorithms.

2.1 Conic Convex Programming

This thesis is concerned primarily with the conic convex optimization problem

\[
\begin{align*}
\inf_x & \quad c^T x \\
\text{subject to} & \quad \mathcal{X} = \mathcal{F}_0 + \mathbf{F} x \\
& \quad \mathcal{X} \in \mathcal{K},
\end{align*}
\]

(2.1)

where \( c, x \in \mathbb{R}^m \), \( \mathcal{X} \) and \( \mathcal{F}_0 \) belong to some finite-dimensional Hilbert space \( \mathbb{E} \), \( \mathbf{F} \) is a linear operator from \( \mathbb{R}^m \) to \( \mathbb{E} \), and \( \mathcal{K} \subset \mathbb{E} \) is a convex cone. Later on, \( \mathcal{K} \) will be restricted to the class of symmetric cones, defined in Section 2.3, but at the moment this restriction is not needed. In convex programming, the space \( \mathbb{E} \) is generally a space of real or complex
vectors or matrices.

The conic form given by (2.1) is very general. In fact, Nesterov and Nemirovskii [NN94] point out that any convex programming problem can be written in this form. Several important types of convex programming problems are linear programming, second-order cone programming and semidefinite programming, which are introduced below.

**Linear Programming** The linear program is perhaps the most well-understood convex programming problem, often written as

\[
\min_x c^T x \\
\text{subject to } Ax \leq b
\]

The convex cone \( \mathcal{K} \) is simply the non-negative orthant \( \mathbb{R}^n_+ \). The most famous algorithm for solving a linear program is the simplex method, which produces iterates that travel along the vertices of the polyhedron defined by \( Ax \leq b \). While the simplex method tends to work well in practice, its exponential worst-case complexity has led researchers to consider alternative polynomial-time interior-point methods.

**Second-Order Cone Programming** The second-order cone, also known as the Lorentz cone or “ice-cream” cone, is described by the set \( \mathbb{R}^{n+1}_{Q^+} = \{ (\gamma, x) \in \mathbb{R} \times \mathbb{R}^n \mid \gamma \geq \| x \| \} \). The notation \( [\gamma \ x]^T \succeq_Q 0 \) is used to indicate \( (\gamma, x) \in \mathbb{R}^{n+1}_{Q^+} \). Linear programming, convex quadratic programming, and quadratically constrained quadratic programming are all special cases of second-order cone programming. An excellent overview of many other problems which may be formulated as a second-order cone program is found in [LVBL98].

**Semidefinite Programming** Semidefinite programming is simply the minimization of a linear objective subject to a linear matrix inequality, described by

\[
\inf_x c^T x \\
\text{subject to } F_0 + \sum_{i=1}^m F_i x_i \succeq 0,
\]

where \( F_0, \ldots, F_m \in \mathbb{S}^n \). The notation \( A \succeq 0 \) indicates that a symmetric matrix \( A \) is positive semidefinite. The cone of all semidefinite matrices of dimension \( n \) is indicated by \( \mathbb{S}^n_+ \).
Sometimes it will be convenient to use the vector-space notation
\[ F_0 = \text{svec}(F_0), \]
\[ F = \begin{bmatrix} \text{svec}(F_1) & \cdots & \text{svec}(F_m) \end{bmatrix}, \tag{2.2} \]
where \( \text{svec}(\cdot) \) maps the subspace of symmetric matrices in \( S^n \) to \( \mathbb{R}^{n(n+1)/2} \) with the convention
\[ \text{svec}(A) = \begin{bmatrix} a_{11}, \sqrt{2}a_{12}, \ldots, \sqrt{2}a_{1n}, a_{22}, \ldots, \sqrt{2}a_{2n}, \ldots, a_{nn} \end{bmatrix}^T, \]
and \( a_{ij} \) is the \( ij \) entry of \( A \). This convention ensures that the inner product \( \langle A, B \rangle \triangleq \text{Tr}AB \) satisfies
\[ \langle A, B \rangle = \text{svec}A^T\text{svec}B = \langle \text{svec}(A), \text{svec}(B) \rangle. \]
The notation \( \text{smat}(\cdot) \) indicates the inverse of \( \text{svec}(\cdot) \). The vectorization of the positive semidefinite cone \( S^n_+ \) is written as \( VS^n_+ \), i.e., \( v \in VS^n_+ \iff \text{smat}(v) \in S^n_+ \).

Semidefinite programming is the most general of the three convex programs considered here, since the linear constraint \( Ax \leq b \) can be written as \( \text{diag}(b - Ax) \succeq 0 \), and the second-order cone constraint \( \|x\| \leq \gamma \) as
\[ \begin{bmatrix} \gamma I & x \\ x^T & \gamma \end{bmatrix} \succeq 0. \]
However, from a complexity point of view it is better to consider a semidefinite constraint using linear or second-order conic constraints whenever possible. To find a solution which is \( \epsilon \) from optimality, the best known interior-point algorithms for semidefinite programming take \( O(\sqrt{n} \log(1/\epsilon)) \) iterations, with \( O(n^4) \) computations per iteration for a dense matrix. A linear program has the same iteration complexity, with only \( O(n^3) \) computations per iteration. This difference in per-iteration complexity is due to the block-diagonal structure of the linear program when posed as a semidefinite program.

The benefit of using a second-order cone is much more dramatic. To see this impact, consider the problem
\[ \min_{\gamma, x} \gamma \]
subject to \( \begin{bmatrix} \gamma \\ Ax + b \end{bmatrix} \succeq 0 \).
where $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$. If this problem is solved using a second-order cone programming interior-point method, it can be solved to $\epsilon$ accuracy in $O(\log(1/\epsilon))$ iterations, regardless of the dimensions $m$ and $n$ [LVBL98]. This is compared to the "equivalent" semidefinite program

$$\min_{\gamma, x} \gamma$$

subject to

$$\begin{bmatrix} \gamma & x^T A + b^T \\ A x + b & \gamma I \end{bmatrix} \succeq 0,$$

which is solved in $O(\sqrt{m + 1} \log(1/\epsilon))$ iterations. In addition, the amount of work per iteration of the second order cone program is $O(n^3)$, compared to $O(n^4)$ for the semidefinite program. From this assessment, the second-order cone format is preferable to the more general semidefinite programming format.

### 2.2 Duality

Associated with any convex conic program is another convex conic program known as the dual:

$$\sup_{Z} \quad -\langle \mathcal{F}_0, Z \rangle$$

subject to

$$\mathcal{F}^T Z = c$$

$$Z \in \mathcal{K}^*,$$

where $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$, defined as

$$\mathcal{K}^* \triangleq \{ Z \mid \langle X, Z \rangle \geq 0, \forall X \in \mathcal{K} \}.$$

Letting $\mathbb{E} = \mathbb{R}^N$, observe that the dual problem can be put in "primal" form (2.1) by writing

$$\inf_{y \in \mathbb{R}^N} b^T y + d$$

subject to

$$Z = S_0 + G y$$

$$Z \in \mathcal{K}^*,$$
where
\[
\begin{align*}
\mathcal{G}_0 &= (\mathbf{F}^+)^T c \\
\mathbf{G} &= I - (\mathbf{F}^+)^T \mathbf{F}^T \\
\mathbf{b} &= \mathbf{G}^T \mathcal{F}_0 \\
d &= \langle \mathcal{F}_0, \mathcal{G}_0 \rangle,
\end{align*}
\]
and \(\mathbf{F}^+\) indicates the Moore-Penrose inverse of \(\mathbf{F}\), e.g., \(\mathbf{F}^+ = (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T\) when \(\mathbf{F}\) has full column rank. Since \(\mathbf{G}\) is a rank \(N - r\) matrix, where \(r\) is the rank of \(\mathbf{F}\), the dimension of \(y\) can be reduced to \(N - r\), and the dimension of \(\mathbf{G}\) reduced from \(N \times N\) to \(N \times (N - r)\).

Let \(p^*\) denote the optimal value of the primal, and \(d^*\) the optimal value of the dual, i.e.,
\[
\begin{align*}
p^* &\triangleq \inf_x \{c^T x \mid \mathcal{F}_0 + \mathbf{F} x \in \mathcal{K}\} \\
d^* &\triangleq \sup_{\mathcal{Z}} \{-\langle \mathcal{F}_0, \mathcal{Z} \rangle \mid \mathbf{F}^T \mathcal{Z} = c, \mathcal{Z} \in \mathcal{K}^*\}.
\end{align*}
\]
In general, \(\inf\) and \(\sup\) cannot be replaced by \(\min\) and \(\max\), since the optimal values may not be achieved.

The duality gap between primal and dual feasible variables is defined as the difference between primal and dual objective values, and is seen to be non-negative by the relation
\[
x^T c + \langle \mathcal{F}_0, \mathcal{Z} \rangle = \langle x, \mathbf{F}^T \mathcal{Z} \rangle + \langle \mathcal{F}_0, \mathcal{Z} \rangle = \langle x, \mathcal{Z} \rangle \geq 0.
\]
From this, it is clear that weak duality holds for any conic problem, meaning \(p^* \geq d^*\).

A stronger duality result is possible given certain constraint qualifications. These qualifications are based on the feasibility of the primal and dual problems, so a few definitions are required. Let the primal and dual feasible sets be defined by
\[
\begin{align*}
\mathcal{F}_P &\triangleq \{ \mathcal{X} \mid \mathcal{X} = \mathcal{F}_0 + \mathbf{F} x, x \in \mathbb{R}^m \} \\
\mathcal{F}_D &\triangleq \{ \mathcal{Z} \mid \mathbf{F}^T \mathcal{Z} = c, \mathcal{Z} \in \mathcal{K}^* \},
\end{align*}
\]
and the interiors defined by
\[
\begin{align*}
\overset{\circ}{\mathcal{F}}_P &\triangleq \mathcal{F}_P \cap \text{int} \mathcal{K} \\
\overset{\circ}{\mathcal{F}}_D &\triangleq \mathcal{F}_D \cap \text{int} \mathcal{K}^*.
\end{align*}
\]
Definition. A primal or dual problem is said to be feasible if $\mathcal{F} \neq \emptyset$, otherwise it is infeasible. A feasible problem is strongly feasible if $\mathcal{F} \neq \emptyset$. Otherwise it is weakly feasible.

Strong feasibility is sometimes known as the Slater condition, and ensures strong duality, defined in the next theorem.

**Theorem 2.1.** If the primal or dual problem is strongly feasible then strong duality holds, meaning $p^* = d^*$. Furthermore, if both the primal and dual are strongly feasible, then the optimal values $p^*$ and $d^*$ are achieved by the primal and dual problems, i.e., inf and sup can be replaced by min and max.

This theorem is a well-known result of convex programming, see for instance Rockafellar [Roc70], also Luo et al. [LSZ97] for many other duality results. The zero duality gap of a strongly dual problem is also known as the complementary slackness condition, where optimizers $X^*$ and $Z^*$ satisfy the orthogonality relation $(X^*, Z^*) = 0$.

### 2.3 Symmetric Cones and Self-Scaled Barriers

In this thesis, it is assumed that the convex cone $\mathcal{K} \subset \mathbb{E}$ is symmetric, which is defined in [FK94] to be a homogeneous and self-dual cone in a Euclidean space. A cone is homogeneous if for every pair of points $X, Y \in \text{int} \mathcal{K}$, there exists a linear transformation $F$ such that $F X = Y$ and $F \mathcal{K} = \mathcal{K}$. It is self-dual if $\mathcal{K}$ and its dual $\mathcal{K}^*$ are isomorphic by some linear transformation $G$, i.e., $\mathcal{K}^* = G \mathcal{K}$. While this class may at first look fairly restrictive, it does include the cones of interest in this thesis: the non-negative orthant, the second-order cone, and the cone of semidefinite matrices. Indeed, each of these cones is equivalent to its dual, i.e., $\mathbb{R}^n_+ = (\mathbb{R}^n_+)^*$, $\mathbb{R}^n_{Q+} = (\mathbb{R}^n_{Q+})^*$, and $\mathbb{S}^n_+ = (\mathbb{S}^n_+)^*$.

Several relevant properties of symmetric cones are summarized below. The reader is referred to Faraut and Korányi [FK94] for a detailed treatment of symmetric cones, and to [SA98, Stu99a] for properties of symmetric cones as they relate to convex programming.

Associated with every symmetric cone is an order $\vartheta$. For the cones of interest to this thesis, both $\mathbb{R}^n_+$ and $\mathbb{S}^n_+$ have order $n$, and the second-order cone $\mathbb{R}^n_{Q+}$ has order 2 for all
2.3. SYMMETRIC CONES AND SELF-SCALED BARRIERS

$n \geq 2$. The order of the Cartesian product of cones is simply the sum of the cone orders, e.g., if $\mathcal{K}_1$ and $\mathcal{K}_2$ have orders $\vartheta_1$ and $\vartheta_2$, then the order of $\mathcal{K}_1 \times \mathcal{K}_2$ is $\vartheta_1 + \vartheta_2$.

A symmetric cone $\mathcal{K} \subset \mathcal{E}$ of order $\vartheta$ can be placed in the context of a Euclidean Jordan Algebra, for which an identity element $J \in \text{int} \mathcal{K}$ and a Jordan product $X \circ Y$ are defined, where $X \circ Y \in \mathcal{K}$ if $X \in \mathcal{K}$ and $Y \in \mathcal{K}$. Each element in $\mathcal{E}$ has a spectral decomposition $X = V(X)\lambda(X)$ where for each $X \in \mathcal{E}$, $\lambda(X) \in \mathbb{R}^\vartheta$ is the spectrum and $V(X)$ is referred to as the Jordan frame, which is a linear operator from $\mathbb{R}^\vartheta$ to $\mathcal{E}$ satisfying

$$V(X)^T V(X) = I \quad V(X)\mathbb{R}^\vartheta_+ \subseteq \mathcal{K}.$$ 

This is akin to the eigenvalue decomposition of a symmetric matrix in linear algebra. The vector $\lambda(X)$ is a unique, ordered vector which satisfies

$$X \in \mathcal{K} \iff \lambda(X) \geq 0$$

$$X \in \text{int} \mathcal{K} \iff \lambda(X) > 0.$$ 

A point $X$ on the boundary of $\mathcal{K}$ satisfies $\lambda_i(X) = 0$ for at least one index $i$.

The identity solution can be derived from any Jordan frame $V$ by $Ve = J$. Several other useful properties of the spectral decomposition are

$$X - \lambda_{\min}(X)J \in \mathcal{K}$$

$$\lambda(x + \alpha J) = \lambda(X) + \alpha e$$

$$\|X\| = \|\lambda(X)\|.$$ 

The class of symmetric cones are known to coincide with the self-scaled cones introduced by Nesterov and Todd [NT97, NT98]. This class is the set of cones which admit a self-scaled $\vartheta$-normal barrier function. The precise definition of this barrier is not needed here, but can be found in [NT97, NT98]. The $\vartheta$-normal barrier is a subset of the class of self-concordant barrier functions introduced by Nesterov and Nemirovskii in [NN94]. A more general discussion of barrier functions in the context of Euclidean Jordan Algebras can be found in [Giul96]. These barrier functions are of fundamental importance to interior-point algorithms. For the cones of interest to this thesis, a barrier satisfying this definition can be constructed as

$$B(X) = -\log \text{Det } X$$
CHAPTER 2. CONVEX PROGRAMMING

where \( \text{Det}(\cdot) \) represents the generalized determinant based on the spectral decomposition for the symmetric cone

\[
\text{Det}(\mathcal{X}) \triangleq \prod_{i=1}^{\vartheta} \lambda_i(\mathcal{X}).
\]

This barrier function is defined on the interior of \( \mathcal{K} \), approaching \( \infty \) for any sequence approaching the boundary of \( \mathcal{K} \). A \( \vartheta \)-normal barrier function is said to be logarithmically homogeneous, meaning it satisfies the identity

\[
B(\tau \mathcal{X}) = B(\mathcal{X}) - \vartheta \log \tau
\]

for all \( \mathcal{X} \in \text{int} \mathcal{K} \) and \( \tau > 0 \). Two important consequences of this property are

\[
\nabla^2 B(\mathcal{X}) \mathcal{X} = -\nabla B(\mathcal{X})
\]

\[
\langle \nabla B(\mathcal{X}), \mathcal{X} \rangle = -\vartheta
\]

for all \( \mathcal{X} \in \text{int} \mathcal{K} \), where \( \nabla B(\mathcal{X}) \) and \( \nabla^2 B(\mathcal{X}) \) represent the gradient and Hessian of \( B \) at \( \mathcal{X} \). It is also significant that \(-\nabla B(\mathcal{X}) \in \text{int} \mathcal{K}^* \) for all \( \mathcal{X} \in \text{int} \mathcal{K} \). These properties will become useful in Chapter 4.

One relevant point derived from the barrier function is the analytic center of a convex region. Supposing that a primal convex region \( \mathcal{F}_p \) is bounded and has a nonempty interior, the analytic center is defined as the unique minimizer of the barrier function, denoted as

\[
\mathcal{X}^* \triangleq \arg\min_{\mathcal{X} \in \mathcal{F}_p} B(\mathcal{X}).
\]

Some properties and barriers of three important symmetric cones are now stated.

**Non-negative orthant** The cone \( \mathbb{R}^n_+ \) is an order \( n \) symmetric cone, with identity element \( e \) (a vector of \( n \) ones) and Jordan product defined by the component-wise product of two vectors. The spectral decomposition is simply \( V(x) = I \) and \( \lambda(x) = x \). The barrier is defined by

\[
B(x) = -\log \prod_{i=1}^{n} x_i \quad \text{for} \ x \in \mathbb{R}^n_+.
\]
The gradient and Hessian for this barrier are
\[
\nabla B(x) = -\left[ \frac{1}{x_1} \cdots \frac{1}{x_n} \right]^T,
\]
\[
\nabla^2 B(x) = \text{diag} \frac{1}{x_i^2}.
\]

**Second-order cone** The Jordan product and identity element of the second-order cone \(\mathbb{R}^{n+1}_{Q+}\) are defined by
\[
(\gamma, x) \circ (\eta, y) = (\gamma \eta + x^Ty, \gamma y + \eta x)/\sqrt{2},
\]
\[
J = (\sqrt{2}, 0).
\]
The spectral decomposition of an element \((\gamma, x)\) is
\[
V(\gamma, x) = \frac{1}{\sqrt{2}} \begin{bmatrix}
1 & 1 \\
-x/\|x\| & x/\|x\|
\end{bmatrix}, \quad \lambda(\gamma, x) = \frac{1}{\sqrt{2}} \begin{bmatrix}
\gamma - \|x\| \\
\gamma + \|x\|
\end{bmatrix}.
\]
The second-order cone admits the barrier
\[
B(\gamma, x) = -\log(\gamma^2 - \|x\|^2) + \log 2 \quad \text{for } (\gamma, x) \in \mathbb{R}^{n+1}_{Q+}.
\]
This barrier has the gradient and Hessian
\[
\nabla B(\gamma, x) = \frac{2}{\gamma^2 - \|x\|^2} \begin{pmatrix}
-\gamma \\
x
\end{pmatrix},
\]
\[
\nabla^2 B(\gamma, x) = \frac{2}{\gamma^2 - \|x\|^2} \begin{pmatrix}
-1 & 0 \\
0 & I
\end{pmatrix} + \frac{4}{(\gamma^2 - \|x\|^2)^2} \begin{pmatrix}
\gamma^2 - \gamma x^T \\
-\gamma x & xx^T
\end{pmatrix}.
\]

**Semidefinite cone** The semidefinite cone \(\mathbb{S}^n_+\) is order \(n\). The Jordan product and identity element are
\[
X \circ Y = (XY + YX)/2
\]
\[
J = I.
\]
The spectrum \(\lambda(X)\) is the vector of eigenvalues, and the linear operation defining the Jordan frame is simply the summation of dyads
\[
X = \sum_{i=1}^n v_i v_i^T \lambda_i.
\]
where $v_i$ is the $i$th eigenvector. The barrier function is defined by

$$B(X) = -\log \det X \quad \text{for } X \in S^n_{++}.$$ 

The gradient of this barrier is

$$\nabla B(X) = -X^{-1}$$

while the Hessian is a linear operator defined by

$$\nabla^2 B(Y)X = Y^{-1}XY^{-1}.$$ 

**2.4 The Central Path**

The central path can be described as a path of strictly feasible points centered away from the constraint boundaries and leading directly to the analytic center of the solution set. Under the assumption that the Slater condition is satisfied, each point on the central path is uniquely parameterized by a scalar $\mu > 0$, and is seen as the analytic center of the constraints at a particular duality gap $\langle X, Z \rangle = \partial \mu$, i.e.

$$(X_\mu, Z_\mu) \triangleq \text{argmin } B(X \circ Z) = -\log \det X - \log \det Z$$

subject to $X \in \mathcal{F}_P, Z \in \mathcal{F}_D$

$$\langle X, Z \rangle = \partial \mu. \quad (2.4)$$

The limit point

$$(X^*, Z^*) = \lim_{\mu \to 0} (X_\mu, Z_\mu)$$

solves the conic convex program.

Definition (2.4) gives an intuitive feel for what the central path is, while a more useful form is given by the unique solution to

$$(X_\mu, Z_\mu) \in \mathcal{F}_P \times \mathcal{F}_D$$

$$X_\mu \circ Z_\mu = \mu I.$$
2.5. THE NEWTON STEP

The central path can also be described in terms of the barrier gradient:

\[ z_\mu = -\mu \nabla B(x_\mu) \]

or equivalently

\[ x_\mu = -\mu \nabla B(z_\mu). \]

This construction is handy when defining neighborhoods of the central path, used in Section 2.6.

Most interior-point methods generate iterates which are close to the central path. Path-following algorithms in particular will target a point on the central path, taking a scaled Newton step to bring the next iterate closer to that point. These algorithms have the quality that they restrict the iterates to a feasible neighborhood of the central path.

2.5 The Newton Step

Nesterov and Nemirovskii demonstrate that as long as a computable self-concordant barrier function (as introduced in Section 2.3) is known for a convex region, polynomial-time interior-point algorithms can be used to optimize over that region (see [NN94], Chapter 5). This important result stems from the behavior of Newton's method on these barrier functions. From (2.4) in the previous section, points on the central path were expressed as the minimizers of self-concordant barrier functions. Based on this observation, Newton's method is seen as a powerful tool in targeting points on the central path. This section demonstrates the computation of the Newton step towards points on the central path.

Given an initial strictly feasible point \((x, x, z)\) with duality measure \(\mu = \langle x, z \rangle / \theta\), a point \((x_{\sigma \mu}, z_{\sigma \mu}) \in C\) with \(\sigma \in [0, 1]\) is targeted by an updated point \((x + \Delta x, x + \Delta x, z + \Delta z)\). To reach the targeted point, the update must satisfy the conditions

\[ \Delta x - F \Delta x = 0 \]  
\[ F^T \Delta z = 0 \]  
\[ (x + \Delta x) \circ (z + \Delta z) = (\sigma \mu) J. \]
For linear, second-order cone, and semidefinite programming, equation (2.5c) is bilinear. Newton's method approximates these conditions by linearizing (2.5c). Solving the linearized equations is straightforward, yielding a Newton step towards the point \((\bar{x}_\sigma, \bar{z}_\sigma)\) on the central path. If \(\sigma = 1\), these equations define a centering direction, which finds an updated point closer to the central path while keeping the duality gap constant. Choosing \(\sigma = 0\) produces a step known as the affine-scaling direction which attempts to reduce the duality gap by as much as possible. However, a full step in this direction is rarely possible, since the updated point would be outside the primal-dual cone \(\mathcal{K} \times \mathcal{K}^*\).

In the case of second-order cone and semidefinite programming, there is some ambiguity in the linearization of (2.5c), since there are many equivalent ways of writing equation (2.5c) using scaling matrices. Consequently, there are many different valid Newton steps. For example, in [Tod99], Todd surveys 20 different directions for semidefinite programming. This thesis is concerned with those directions belonging to the Monteiro-Zhang (MZ) family [MZ98], which was originally conceived for semidefinite programming but can be put in the more general context of symmetric cones [SA98]. This family characterizes the most commonly used search directions in the literature. For simplicity, the next section presents only the MZ-family for semidefinite programming. The MZ-family for second-order cone programming can be found in [MT98].

### 2.5.1 MZ-Family for Semidefinite Programming

In [Zha98], Y. Zhang suggested an approach to finding symmetric directions in semidefinite programming which unified several directions proposed by other authors. Given a matrix \(M \in \mathbb{S}^n\), Zhang defines the linear transformation

\[
H_P(M) \triangleq \frac{1}{2} [PMP^{-1} + (PMP^{-1})^T]
\]

for a given nonsingular matrix \(P \in \mathbb{R}^{n \times n}\), and notes that

\[
H_P(XZ) = \mu I \iff XZ = \mu I. \tag{2.6}
\]
2.5. THE NEWTON STEP

Equation (2.5c) is therefore equivalent to \( H_P((X + \Delta X)(Z + \Delta Z)) = \sigma \mu I \), which is linearized as

\[
H_P(XZ + \Delta XZ + X\Delta Z) = \sigma \mu I.
\] (2.7)

Even though (2.6) is linearized, the duality gap is reduced exactly by a factor \( \sigma \) (i.e., \( \text{Tr}(X + \Delta X)(Z + \Delta Z) = n\alpha \mu \)) due to the orthogonality property \( \text{Tr} \Delta X \Delta Z = 0 \).

Previous to Zhang's work, this symmetrization was used by Monteiro for the cases \( P = X^{-1/2} \) and \( P = Z^{1/2} \) [Mon97]. The family of directions parameterized by (2.7) are therefore said to belong to the Monteiro-Zhang (MZ) family [MZ98]. Three Newton directions belonging to the MZ family are defined below: the Alizadeh-Haeberly-Overton (AHO) direction, the Helmberg-Rendl-Vanderbei-Wolkowicz/Kojima-Shindoh-Hara/Monteiro (H..K..M) direction, and the Nesterov-Todd (NT) direction.

**AHO Direction** The direction introduced by Alizadeh, Haeberly and Overton [AHO98] is defined by

\[
(X\Delta Z + \Delta XZ) + (Z\Delta X + \Delta ZX) = 2\sigma \mu I - XZ - ZX
\]

which results from choosing \( P = I \) in (2.7). This is the direct linearization of (2.5c). Computation of the AHO direction requires the solution of a Lyapunov equation, requiring roughly \( 8mn^3 + m^2n^2 \) computations [TTT98]. Since \( P \) is based neither on the primal or dual iterates, the AHO direction is said to have *primal-dual symmetry*.

**H..K..M Direction** The H..K..M direction is found by choosing \( P = X^{-1/2} \) or \( P = Z^{1/2} \). It is so named because it was first introduced by Helmberg, Rendl, Vanderbei and Wolkowicz [HRVW96], was later rediscovered by Kojima, Shindoh, and Hara [KSH97] as one member of a family of directions, and then rederived by Monteiro [Mon97]. The form introduced by Monteiro is derived by the equation

\[
2X^{1/2}\DeltaZX^{1/2} + X^{-1/2}\Delta XZX^{1/2} + X^{1/2}Z\Delta XX^{-1/2} = 2\sigma \mu I - 2X^{1/2}ZX^{1/2}.
\]

Its dual form is similarly found by substituting \( P = Z^{1/2} \), which in most cases will yield a different direction. Unlike the AHO direction, the H..K..M direction does not have
primal-dual symmetry, since it treats primal and dual directions differently. Derivation of the H..K..M direction requires approximately $4mn^3 + 0.5m^2n^2$ computations [TTT98].

**NT Direction** The NT direction was introduced in [NT97, NT98] for Nesterov and Todd’s general class of self-scaled cones. It is a member of the MZ-family with $P = (X^{1/2}ZX^{1/2})^{1/4}X^{-1/2}$, and is derived by the equation

$$W^{-1}AXW^{-1} + AZ = \sigma\mu X^{-1} - Z,$$

where $W = X^{1/2}(X^{1/2}ZX^{1/2})^{-1/2}X^{1/2}$. The matrix $W$ is known as the scaling matrix, which has the property $W^{-1}XW^{-1} = Z$ (this is the self-scaling property). From this property, it can be shown that (2.8) is equivalent to its dual form, thus the NT direction has primal-dual symmetry. Approximately $3mn^3 + 0.5m^2n^2$ computations are required to find this direction.

In [TTT98], Todd et al. have carried out numerical experiments on these directions using a Mehrotra [Meh92] predictor-corrector algorithm on various semidefinite programs. Todd et al. concluded that while all three are competitive, the AHO direction generally appears to converge the fastest. However, since computation of this direction is about twice as expensive as computation of the H..K..M and NT directions, in many situations it may be more favorable to use one of the two latter directions.

### 2.6 Path-Following Algorithms

This section is concerned with primal-dual path-following methods, which can be described as methods which generate a sequence of iterates that “follow” the central path in the direction of decreasing $\mu$. This path is followed in the sense that the iterates remain within a certain neighborhood of the central path while the duality gap converges to zero. Although there are a variety of different neighborhoods used in the optimization literature (see §4 in [NT98] for the definitions of seven central path proximity measures, the level sets of which are neighborhoods of the central path), this thesis will focus primarily
on the neighborhood

\[
\mathcal{N}(\beta) \triangleq \left\{ (X, Z) \in \overset{\circ}{\mathcal{F}}_P \times \overset{\circ}{\mathcal{F}}_D \mid \frac{1}{\mu} Z + \nabla B(X) \left\| \frac{1}{\mu} Z + \nabla B(X) \right\|_\mu^{-1} \leq \beta \right\}
\]  

(2.9)

where \( \beta \in [0, 1) \). Notice that \( \mathcal{N}(0) = \mathcal{C} \). For linear programming, this neighborhood is

\[
\mathcal{N}(\beta) \triangleq \left\{ (x, z) \in \overset{\circ}{\mathcal{F}}_P \times \overset{\circ}{\mathcal{F}}_D \mid \left( \sum_{i=1}^{n} \left( \frac{x_i z_i}{\mu} - 1 \right)^2 \right)^{1/2} \leq \beta \right\},
\]

whereas for semidefinite programming,

\[
\mathcal{N}(\beta) \triangleq \left\{ (X, Z) \in \overset{\circ}{\mathcal{F}}_P \times \overset{\circ}{\mathcal{F}}_D \mid \left\| \frac{1}{\mu} X^{1/2} ZX^{1/2} - I \right\|_F \leq \beta \right\}.
\]

The equivalent neighborhood for second-order cone programming is slightly more complicated, so the reader is referred to [Tsu00] for its definition.

Assuming that the Slater condition is satisfied, the path-following methods discussed here will keep the iterates within \( \mathcal{N}(\beta) \). The fact that these methods restrict the iterates to this neighborhood will be significant in Chapter 4, where convergence for a homogeneous self-dual problem depends on an algorithm which produces weakly centered iterates (see Section 4.2.2).

Four different path-following algorithms are briefly described in this section: the short-step, predictor-corrector, long-step, and Mehrotra’s predictor-corrector algorithm.

**Short-Step Path-Following Algorithm** The short-step method was developed for linear programming by Kojima, Mizuno, and Yoshise [KMY89a], and Monteiro and Adler [MA89], and later extended to semidefinite programming in [KSH97, Mon97, Mon98, NT98] for the Newton directions in Section 2.5.

The fundamental idea behind the short-step method is to choose parameters \( \beta \) and \( \Delta \) such that for any point \((X, Z) \in \mathcal{N}(\beta)\), the Newton step derived from \( \sigma = 1 - \Delta/\sqrt{\beta} \) satisfies

i.) \((X + \Delta X, Z + \Delta Z) \in \mathcal{N}(\beta)\)

ii.) \(\langle X + \Delta X, Z + \Delta Z \rangle = \sigma \langle X, Z \rangle\).
In [NT98], Nesterov and Todd show that these conditions hold for the NT direction with \( \beta = 1/10 \) and \( \Delta = 1/15 \). Monteiro proved that for semidefinite programming, these conditions hold for all Newton steps in the MZ family with \( \beta = \Delta = 1/20 \) [Mon98]. This provides the means for an algorithm which reduces the duality gap by \( \sigma \) at each iteration, and keeps each iterate feasible within the neighborhood \( \mathcal{N}(\beta) \).

Assuming that the primal-dual initialization satisfies \((X(0), Z(0)) \in \mathcal{N}(\beta)\), then the duality gap at iteration \( k \) satisfies an optimality condition \((X(k), Z(k)) \leq \epsilon\) whenever

\[
k \geq \left\lceil \frac{\log(\epsilon/(X(0), Z(0)))}{\log(1 - \Delta/\sqrt{\theta})} \right\rceil.
\]

This iteration requirement grows as \( \mathcal{O}(\sqrt{\theta} \log((X(0), Z(0))/\epsilon)) \). However, even though the convergence rate is polynomial, it can still be quite slow when compared to other practical implementations of interior-point algorithms. This is because the algorithm always reduces the duality gap by the same factor \( \sigma \), never more. The algorithm can be augmented with a line search, reducing the duality gap by as much as possible while restricting the updated iterate to remain in \( \mathcal{N}(\beta) \). This will speed up the convergence of the algorithm in practice, while the theoretical complexity bounds will remain the same.

Another algorithm which allows the possibility of larger duality gap reductions is the predictor-corrector algorithm, described next.

**Predictor-Corrector Algorithm** The predictor-corrector algorithm was introduced for linear programming by Mizuno, Todd and Ye in [MTY93], and extended to semidefinite programming for directions in the MZ family by [Mon98, Zha98]. Two neighborhoods of the central path are used, \( \mathcal{N}(\beta) \) and \( \mathcal{N}(\bar{\beta}) \), where \( 0 < \beta < \bar{\beta} < 1 \). Each iteration consists of two steps:

1. **Predictor step** — Given \((X(k), Z(k)) \in \mathcal{N}(\beta)\), compute a Newton step \((\Delta X, \Delta Z)\) for \( \sigma = 0 \). Find the maximum step length \( \alpha_k \) such that \((\hat{X}(k), \hat{Z}(k)) = (X(k), Z(k)) + \alpha_k (\Delta X, \Delta Z) \in \mathcal{N}(\beta)\).

2. **Corrector step** — Compute a Newton step \((\Delta \hat{X}, \Delta \hat{Z})\) for \( \sigma = 1 \) and update by

\[
(X(k+1), Z(k+1)) = (X(k), Z(k)) + (\Delta \hat{X}, \Delta \hat{Z}).
\]
2.6. PATH-FOLLOWING ALGORITHMS

The predictor step attempts to decrease the duality gap by as much as possible while remaining in $\mathcal{N}(\beta)$. In the literature, this is also known as the affine scaling direction (see [MAR90] for the use of this step in the primal-dual affine-scaling algorithm). The predictor step reduces the duality gap by a factor $1 - \alpha_k$. Under certain conditions on $\beta$ and $\bar{\beta}$, it is known that $\alpha_k > O(1/\sqrt{\beta})$. For example, this is true for any Newton step in the semidefinite programming MZ-family whenever $\bar{\beta} = 2\beta$ [Mon98]. During the corrector step, the duality gap remains constant, and the iterate is brought closer to the central path. The objective of the corrector step is to bring the iterate back in the neighborhood $\mathcal{N}(\beta)$. For the NT direction, this is possible if $\beta = 1/10$ and $\bar{\beta} = 1/6$, with $\alpha \geq 1/(10\sqrt{\beta})$ [NT98]. This achieves the same iteration complexity as the short-step algorithm, with the number of iterations required to guarantee an $\epsilon$-optimal solution determined by (2.10).

Like the short-step algorithm, predictor-corrector restricts the iterates to a neighborhood $\mathcal{N}(\beta)$ of the central path, thereby limiting the step size the algorithm may take at each iteration. The disadvantage of this neighborhood is that even when $\beta$ is close to its upper bound of 1, $\mathcal{N}(\beta)$ only contains a small fraction of the feasible points. Improvements may be made by considering path-following algorithms which use larger neighborhoods. Nesterov and Todd describe one such algorithm, called the functional proximity path-following scheme [NT98], which is the same as the predictor-corrector method except that it uses the neighborhood

$$
\mathcal{N}_\psi(\beta) \triangleq \{(X, Z) \in \bar{\mathcal{F}}_p \times \bar{\mathcal{F}}_D \mid \psi(X, Z) \leq \beta\},
$$

where

$$
\psi(X, Z) \triangleq \log \text{Det} X_\mu \circ Z_\mu - \log \text{Det} X \circ Z
$$

$$
= \vartheta \log \langle X, Z \rangle - \log \text{Det} X - \log \text{Det} Z - \vartheta \log \vartheta
$$

(2.11)

and $X_\mu$, $Z_\mu$ are the points on the central path corresponding to the duality measure $\mu$. The second line of (2.11) holds because $\log \text{Det} X \circ Z = \log \text{Det} X + \log \text{Det} Z$ and $\log \text{Det} X_\mu \circ Z_\mu = \log \text{Det} \mu J = \vartheta \langle X, Z \rangle - \vartheta \log \vartheta$. The proximity measure $\psi(X, Z)$ is the difference between the value of the barrier function at a pair $X, Z$ and the barrier function
at the closest point on the central path. Observe that for any point \((X, Z) \in \tilde{F}_P \times \tilde{F}_D\), it is possible to choose a \(\beta\) such that this point is in \(\mathcal{N}_\psi(\beta)\). This was not true for \(\mathcal{N}(\beta)\).

This algorithm also has a \(O(\sqrt{\beta} \log(1/\epsilon))\) complexity. It is expected that in practice it should exhibit faster convergence than the algorithms based on the neighborhood \(\mathcal{N}(\beta)\), since the neighborhood \(\mathcal{N}_\psi(\beta)\) is capable of covering a much larger portion of the feasible region than \(\mathcal{N}(\beta)\).

**Long-Step Algorithm** Another path-following algorithm which uses a large neighborhood of the central path is the long-step algorithm of Kojima, Mizuno and Yoshise, which was developed for linear programming in [KMY89b] and extended to semidefinite programming in [Mon97, Zha98]. This neighborhood limits the spectrum of \(X \circ Z\) to lie within some interval \([a\mu, b\mu]\) where \(0 < a < 1 < b\), which can come arbitrarily close to the feasible space \(\tilde{F}_P \times \tilde{F}_D\). For practical purposes, this algorithm tends to perform better than the short-step and predictor-corrector algorithms based on \(\mathcal{N}(\beta)\), since steps along the Newton direction may be taken over a larger range. However, the long-step method has an \(O(\sqrt{\beta} \log(1/\epsilon))\) complexity, versus \(O(\sqrt{\beta} \log(1/\epsilon))\) for short-step and predictor-corrector. Therefore, even though this method is very practical, the iteration bound on the worst-case problem grows at a much faster rate than is useful for this thesis.

**Mehrotra’s Predictor-Corrector Algorithm** Mehrotra’s predictor-corrector algorithm is introduced as a benchmark for other path-following algorithms. This algorithm is not to be confused with the generic predictor-corrector algorithm described on page 42. In its most commonly used form, Mehrotra’s algorithm does not have any guarantees on the iteration complexity. Despite this fact, it does tend to exhibit the best performance in practice. Most interior-point software uses some variant of this method due to this practical efficiency. Mehrotra developed his method in the early 1990s for linear programming [Meh92], and since then it has been expanded upon and generalized to convex programming. It can be roughly described by the following three steps:

1. Predictor step — Compute the affine scaling direction, i.e., the linearized solution to (2.5) with \(\sigma = 0\).
2. Adaptive choice for centering parameter — Determine the step length that can be taken along the affine scaling direction before the new iterate reaches the boundary. Apply heuristic to adaptively choose centering parameter \( \sigma \) based on step length.

3. Corrector step — Using adaptively chosen \( \sigma \), compute Newton step which also accounts for second-order term in affine scaling step.

The first step is the same as the affine scaling step discussed previously. However, rather than taking this step, the information derived from it is used to construct a new, more accurate Newton step. The centering term \( \sigma \) is chosen adaptively based on how “good” the affine scaling step was. If a long step length could be taken in the affine scaling direction before encountering the cone boundary, then \( \sigma \) is set close to 0, otherwise more centering is required so it is set closer to 1. Finally, a second-order correction is added to the linearization of (2.5), allowing the Newton step to better approximate the curvature of the central path.

This is only a loose description of Mehrotra’s predictor-corrector algorithm, which also employs various other heuristics to speed up the interior-point algorithm. The net result of these heuristics is an algorithm which works extremely well, but is difficult to analyze from a theoretical perspective. Since the worst-case performance of this algorithm is unknown, it is not suitable for on-line optimization. Its purpose here is more for comparison. In the UAV example presented in Chapter 6, the performance of Mehrotra’s predictor-corrector algorithm will be compared with the more conservative, predictable algorithms presented earlier.

2.7 Homogeneous Self-Dual Embeddings

A common feature of the path-following methods mentioned so far is that they must all be initialized with a strictly feasible primal-dual pair contained within the appropriate neighborhood of the central path. It cannot be assumed that this initialization is immediately available, and finding it from scratch generally has the same complexity as solving the optimization problem. To address this problem, several methods have
emerged for initialization. One common technique is to employ an infeasible-start algorithm, which modifies the Newton equations to allow infeasible iterates. All iterations of an infeasible-start algorithm remain infeasible, but converge to feasibility while simultaneously converging to optimality. An overview of infeasible-start strategies is given in [Wri96]. The other basic initialization strategy is to embed the optimization problem into a slightly larger problem for which a strictly feasible point is easy to identify. One embedding strategy is to "enlarge" the constraints using a common technique known as the big-M method (see [BSS93] or [VB96]). However, this method is not known to be computational efficient. A more promising embedding strategy known as the homogeneous self-dual embedding has received considerable attention in recent years. It is this category of initialization techniques which is of interest to this thesis. These embeddings will be seen later in Chapter 4.

Before describing homogeneous self-dual embeddings, it is first necessary to introduce some basic concepts about homogeneous self-dual systems. Consider a system of homogeneous linear relations $\mathcal{F}_P$ and a related dual system $\mathcal{F}_D$:

$\mathcal{F}_P = \{(x, y) \mid x \in \mathcal{K}_1, y = Ax \in \mathcal{K}_2\}$

$\mathcal{F}_D = \{(u, v) \mid u = -A^T v \in \mathcal{K}_1^*, v \in \mathcal{K}_2^*\}$,

where $\mathcal{K}_1$ and $\mathcal{K}_2$ are convex cones. The duality considered here can be thought of as the duality of a convex program with a zero objective function vector and homogeneous constraints, e.g., $\min \{0 \mid x \in \mathcal{K}_1, Ax \in \mathcal{K}_2\}$. An important property of dual systems is complementarity between elements in $\mathcal{F}_P$ and $\mathcal{F}_D$, specifically $x^T u = 0$ and $y^T v = 0$ for all $(x, y) \in \mathcal{F}_P$ and all $(u, v) \in \mathcal{F}_D$. This is easily proved, by noting that $x^T u \geq 0$ since $x \in \mathcal{K}_1$ and $u \in \mathcal{K}_1^*$. Also, $x^T u \leq 0$ since $x^T u = x^T (-A^T v) = -y^T v$ for some $y \in \mathcal{K}_2$ and $v \in \mathcal{K}_2^*$. Thus $x^T y = 0$. The same can be proved for $y^T v$.

A self-dual system has the form

$\mathcal{S} = \{(x, y) \mid x \in \mathcal{K}, y = Ax \in \mathcal{K}^*\}$ (2.12)

where $A = -A^T$. It is not hard to see that for such a system, $(x, y) \in \mathcal{F}_P$ if and only if $(y, x) \in \mathcal{F}_D$. Self-dual systems satisfy self-complementarity, where $x^T y = 0$ for all $(x, y) \in \mathcal{S}$.
Dual systems were considered by Tucker in [Tuc56], where he considered the cone \( \mathcal{K} = \mathbb{R}_+^{n_1} \times \mathbb{R}^{n_2} \times \{0\}^{n_3} \). By restricting consideration to only equality and nonnegativity constraints, Tucker proved that not only do elements of \( \mathcal{F}_P \) and \( \mathcal{F}_D \) satisfy complementarity conditions, there exist strictly-complementary solutions. This means that there exists a feasible primal-dual pair \((x, y)\) and \((u, v)\) where for every index \(i\) such that \(\mathcal{K}_1\) restricts \(x_i \geq 0\) and \(u_i \geq 0\), then \(x_i + u_i > 0\). Likewise, \(y_j + v_j > 0\) for all indices \(j\) where \(\mathcal{K}_2\) restricts \(y_j \geq 0\) and \(v_j \geq 0\). This fact will be useful for the homogeneous self-dual model introduced in the next section. Unfortunately, strict-complementarity does not hold in general for arbitrary convex cones.

### 2.7.1 The Homogenous Self-Dual Model for LP

In [GT56], Goldman and Tucker used the existence of strictly-complementary solutions to self-dual systems to prove many basic theorems of linear programming. Their results were derived using the homogeneous self-dual model, which will be presented shortly. Since the introduction of interior-point methods, it has become apparent that the homogeneous self-dual model can be used for solving linear programs, as well as general convex programming problems.

To motivate the homogeneous self-dual model, consider the following primal-dual pair of linear programs

\[
\begin{align*}
\min \{c^T x \mid Ax &= b, x \geq 0\} & \quad \text{(2.13a)} \\
\max \{b^T y \mid A^T y + s = c, s \geq 0\} & \quad \text{(2.13b)}
\end{align*}
\]

By introducing the variables \(\tau\) and \(\kappa\), this problem can be embedded in the homogeneous self-dual model

\[
\begin{bmatrix}
0 \\
\bar{s} \\
\kappa
\end{bmatrix} =
\begin{bmatrix}
0 & A & -b \\
-A^T & 0 & c \\
\bar{b}^T & -c^T & 0
\end{bmatrix}
\begin{bmatrix}
\bar{y} \\
\bar{x} \\
\tau
\end{bmatrix},
\]

\((\bar{x}, \bar{s}, \tau, \kappa) \geq 0.\)

It is easily verified that this system has the same form as (2.12). A strictly-complementary solution between \((\bar{x}, \tau)\) and \((\bar{s}, \kappa)\) is guaranteed to exist, and will satisfy either \((\tau^* >\)
0, \( \kappa^* = 0 \) or \( \tau^* = 0, \kappa^* > 0 \). Either condition will yield important information about the primal-dual pair (2.13).

First, consider the case where \( \tau^* > 0 \) and \( \kappa^* = 0 \). In this case, it is evident that \( \bar{x}^*/\tau^* \) and \( (\bar{s}^*/\tau^*, \bar{y}^*/\tau^*) \) will be feasible solutions to (2.13). Since \( \kappa^* = 0 \), the third equation in (2.14) will set the duality gap \( c^T \bar{x}^*/\tau^* - b^T \bar{y}^*/\tau^* \) equal to zero, thereby making these feasible solutions optimal.

Next, consider the alternative case where \( \tau^* = 0 \) and \( \kappa^* > 0 \). Since \( \tau^* = 0 \), \( \bar{x}^* \) lies in the nullspace of \( A \), and \( A^T \bar{y}^* \leq 0 \), making \( x^* \) and \( y^* \) both feasible directions in the original problem. Also, \( c^T \bar{x}^* - b^T \bar{y}^* < 0 \), which means that at least one of \( c^T \bar{x}^* \) and \( -b^T \bar{y}^* \) is negative. If \( -b^T \bar{y}^* < 0 \), then by Farkas' Lemma (which is found in almost any optimization textbook, e.g., [BSS93]), the primal problem (2.13a) is infeasible. Similarly, if \( c^T \bar{x}^* < 0 \) then \( \bar{x}^* \) is an unbounded direction of improvement in the primal problem, so the dual problem (2.13b) is infeasible.

Thus, a strictly-complementary solution to (2.14) will produce either an optimal solution to (2.13) or a certificate of infeasibility. Notice that (2.14) can also be solved trivially, therefore not all solutions to this system will be strictly-complementary. Fortunately, certain interior-point methods produce iterates which approach strictly-complementary solutions (e.g., see Theorem 5.14 in [Wri96]), so there exist algorithms for finding these points.

An important observation about (2.14) is that the set of feasible solutions has no interior. To see this, suppose such a point exists. Then \( \tau > 0 \), and this point would yield a strictly feasible points \( \bar{x}/\tau \) and \( (\bar{s}/\tau, \bar{y}/\tau) \) to (2.13). However, (2.14) would also imply that \( c^T \bar{x} - b^T \bar{y} < 0 \), i.e., a negative duality gap, which is clearly impossible. Consequently, an interior point cannot exist.

Most conventional interior-point methods must be initialized with an interior point. Since (2.14) does not have an interior, any iterative solution procedure must start from a boundary or infeasible point, or the problem itself must be modified. In [XHY96], Xu, Hung and Ye proposed an infeasible-interior-point algorithm to solve this problem. In Section 2.7.3, the homogeneous self-dual system will be modified to include a known feasible interior point.
2.7.2 The General Homogenous Self-Dual Model

While the self-dual model of Goldman and Tucker applies only to linear programming, it can be generalized to any conic convex programming problem. Luo, Sturm and Zhang present a general framework for the self-dual model in [LSZ98]. Potra and Sheng [PS98] developed a semidefinite programming formulation of this problem, and presented two infeasible-interior-point algorithms which can solve it.

For reference, the conic convex programming problem and its associated dual are restated:

\[
\begin{align*}
\min & \quad c^T x \mid \mathcal{F} x + F x \in \mathcal{K} \\
\max & \quad \langle \mathcal{F}_0, Z \rangle \mid F^T Z = c, Z \in \mathcal{K}^*.
\end{align*}
\]

When both (2.15a) and (2.15b) are strictly feasible, the optimizers \((x^*, Z^*)\) are characterized by a zero duality gap, i.e., \(c^T x^* + \langle \mathcal{F}_0, Z^* \rangle = 0\).

The homogeneous self-dual model is constructed by introducing scalar variables \(\tau \) and \(\kappa\) to the original primal-dual pair:

\[
\begin{align*}
\bar{\mathcal{X}} &= \tau \mathcal{F}_0 + F \bar{x} \\
F^T \bar{Z} - \tau c &= 0, \\
c^T \bar{x} + \langle \mathcal{F}_0, \bar{Z} \rangle &\leq 0 \\
\bar{\mathcal{X}} \in \mathcal{K}, \quad \bar{Z} \in \mathcal{K}^*, \quad \tau \geq 0.
\end{align*}
\]

The skew-symmetric form analogous to (2.14) is

\[
\begin{bmatrix}
0 \\
\bar{x} \\
\kappa
\end{bmatrix}
= \begin{bmatrix}
0 & -F^T & c \\
F & 0 & \mathcal{F}_0 \\
-c^T & -\mathcal{F}_0^T & 0
\end{bmatrix}
\begin{bmatrix}
\bar{x} \\
\bar{Z} \\
\tau
\end{bmatrix},
\]

\[
\bar{\mathcal{X}} \in \mathcal{K}, \quad \bar{Z} \in \mathcal{K}^*, \quad \tau \geq 0, \quad \kappa \geq 0.
\]

Since (2.16) is a self-dual system, all solutions are self-complementary. However, unlike the linear programming case, there is no guaranteed existence of a strictly-complementary solution. Thus it is possible that all solutions to (2.16) satisfy \(\tau^* = 0\) and
\( \kappa^* = 0 \) simultaneously. This reflects the greater variety of feasibility conditions which may arise in a general convex programming framework.

Recall that for linear programming, (2.14) has a solution with \( \tau^* > 0 \) if and only if (2.13) is feasible, and a solution exists with \( \kappa^* > 0 \) if and only if (2.13) is infeasible. In the general case, it is possible that \( \tau^* = \kappa^* = 0 \) for all solutions to (2.16). This is true for infeasible problem instances for which there exist arbitrarily small perturbations in the problem data which make the problem feasible. An infeasible problem with this property is said to be *weakly infeasible*, otherwise it is *strongly infeasible*. This is analogous to the definition of weak and strong feasibility introduced in Section 2.2. Problems which are weakly feasible or weakly infeasible are said to be *ill-posed*. For linear programming, all ill-posed problem instances are feasible, which is why there always exist strictly complementary solutions.

For the general convex programming case, the following conditions hold:

- (2.16) has a solution with \( \tau^* > 0 \) if and only if (2.15) has a complementary solution.

- (2.16) has a solution with \( \kappa^* > 0 \) if and only if (2.15) is strongly infeasible.

One of these conditions will always be satisfied if a problem is not ill-posed, although \( \tau^* > 0 \) may hold even if the problem is ill-posed. For example, every ill-posed linear program is feasible, and every feasible linear program has a complementary solution, therefore every linear program satisfies one of the two conditions above.

### 2.7.3 A Strongly Feasible Self-Dual Embedding

By the introduction of a new variable \( \theta \), the homogeneous self-dual system (2.16) can be transformed into a strongly feasible convex program which is easy to initialize. Consequently, the new problem does not require an infeasible-interior-point algorithm. This problem was originally introduced for linear programming by Ye, Todd and Mizuno [YTM94], and has been generalized to semidefinite programming in [dKRT97, dKRT98,
2.7. HOMOGENEOUS SELF-DUAL EMBEDDINGS

The self-dual embedding is defined by

\[
\min_{\theta} \ z \theta
\]

subject to

\[
\tilde{X} - \tau F_0 - F \tilde{x} + \tilde{F} \theta = 0
\]

\[
F^T \tilde{Z} - \tau c + \theta\tilde{c} = 0
\]

\[
c^T \tilde{x} + \langle F_0, \tilde{Z} \rangle - \alpha \theta + \kappa = 0
\]

\[
c^T \tilde{x} + \langle \tilde{F}, \tilde{Z} \rangle - \alpha \tau = -\zeta
\]

\[
\tilde{X} \in \mathcal{K}, \quad \tilde{Z} \in \mathcal{K}^*, \quad \tau \geq 0, \quad \kappa \geq 0
\]

where

\[
\zeta = \langle \tilde{X}^0, \tilde{Z}^0 \rangle + \tau^0 \kappa^0
\]

\[
\tilde{F} = \tau^0 F_0 + F \tilde{x}^0 - \tilde{X}^0
\]

\[
\tilde{c} = \tau^0 c - F^T \tilde{Z}^0
\]

\[
\alpha = \kappa^0 + c^T \tilde{x}^0 + \langle F_0, \tilde{Z}^0 \rangle
\]

and \(\tilde{X}^0 \in \text{int} \mathcal{K}, \tilde{Z}^0 \in \text{int} \mathcal{K}^*, \tilde{x}^0 \in \mathbb{R}^n, \tau^0 \in \mathbb{R}_{++}, \kappa^0 \in \mathbb{R}_{++}\) can be chosen arbitrarily as long as they lie in the interiors of their respective cones.

A strictly feasible initialization is found by setting the initial iterates \(\tilde{X}^{(0)}, \tilde{Z}^{(0)}, \tilde{x}^{(0)}, \tau^{(0)}, \kappa^{(0)}\) equal to \(\tilde{X}^0, \tilde{Z}^0, \text{etc.}\), and setting \(\theta^{(0)} = 1\). Furthermore, the initial iterates lie on the central path if \(\tilde{X}^0 \circ \tilde{Z}^0 = \mu \mathcal{J}\) and \(\tau^0 \kappa^0 = \mu\) for some scalar \(\mu > 0\). Thus, it becomes easy to initialize an interior-point algorithm with a feasible, centered starting point.

Since this problem can be initialized with an interior point, the problem is strongly feasible, and an optimal primal-dual point exists with zero duality gap. Because the problem is self-dual, it is not hard to see that the duality gap of (2.17) is \(2\zeta \theta\), which is equivalent to \(2(\langle \tilde{X}, \tilde{Z} \rangle + \tau \kappa)\). Therefore, a feasible point is optimal if and only if \(\theta = 0\). Under this condition, the optimal point to (2.17) becomes a feasible (and therefore optimal) point for the homogeneous self-dual problem (2.16). As long as the primal-dual problem (2.15) is well-posed, then a solution to (2.17) can be used to determine the optimal solution to the primal and dual problems, or determine that they are either infeasible or unbounded.

At first glance, it may appear that the advantages of the self-dual embedding come at the penalty of doubling the dimension size. However, primal-dual interior-point methods
can be applied to self-dual systems *without* this doubling, since half of the problem is dual to the other half. Thus, a self-dual embedding of a problem with dimension $\vartheta$ is posed as a convex program with dimension $\vartheta + 1$ rather than one of dimension $2\vartheta + 2$. 
Chapter 3

Parametric Convex Programming

Traditionally, parametric programming has been used to analyze the sensitivity of a convex program to variations in the problem data (e.g., see [FI90]). This thesis uses parametric programming in a slightly different setting. Here, parametric programming is used to characterize possible problem instances which may arise during implementation of an on-line optimization procedure. Throughout this thesis, the problem data \( p = (c, \mathcal{F}_0, F) \) is sometimes indicated as a parametric function \( p(\delta) \) of a vector \( \delta \), where \( \delta \) belongs to some domain \( D \subseteq \mathbb{R}^s \). Let \( \mathcal{P} = \mathbb{R}^m \times \mathbb{R}^N \times \mathbb{R}^{N \times m} \) denote the space of all feasible and infeasible problem instances \((c, \mathcal{F}_0, F)\). For a given on-line optimization problem, the function \( p(\delta) \) and domain \( D \) are chosen such that the image of \( D \), written \( \mathcal{P}_D = \{ p(\delta) \mid \delta \in D \} \subseteq \mathcal{P} \), contains all problem instances which may occur during implementation of the on-line procedure.

The parameterized problem data \( p(\delta) \) is assumed to be rational in \( \delta \), and well-defined over \( D \). The set \( D \) is assumed to be compact, but no other requirement is needed. In considering rational functions \( p(\delta) \), it will be more useful in some of the sections to come to consider \( p(\delta) \) as a linear fractional transformation.

3.1 Linear Fractional Transformations

The following lemma is a well-known property about the representation of rational matrix functions.
Lemma 3.1. Let $F : \mathbb{R}^s \to \mathbb{R}^{m \times n}$ be a rational matrix function with a finite value at the origin. Then $F$ can be written as a linear-fractional transformation (LFT)

$$F(\delta_1, \ldots, \delta_s) = \mathcal{F}_\ell(M, \Delta) \triangleq M_{11} + M_{12}\Delta(I - M_{22}\Delta)^{-1}M_{21},$$

(3.1)

where $\Delta = \text{diag}(\delta_{1r_1}, \ldots, \delta_{sr_s})$ and where $M_{11} \in \mathbb{R}^{m \times n}$, $M_{12} \in \mathbb{R}^{m \times L}$, $M_{22} \in \mathbb{R}^{L \times L}$, $M_{21} \in \mathbb{R}^{L \times n}$, and $L = r_1 + \ldots + r_k$ for some nonnegative integers $r_1, \ldots, r_s$.

Proof. See [ZDG96], §10.2.

The proof of this lemma is constructive, and can be used to find such a LFT representation for a rational matrix function. The LFT (3.1) has its roots in control theory (see [DPZ91] for its relationship with robust control and the structured singular value). In control theory, the LFT $\mathcal{F}_\ell(M, \Delta)$ is used to represent the map $F(\Delta) : w \mapsto z$ in the feedback loop

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix},$$

$$u = \Delta y.$$

The problem data $p(\delta)$ with parameter $\delta \in \mathbb{R}^s$ can then be placed in an equivalent form as a LFT function with matrix variable $\Delta \in \mathbb{R}^{L \times L}$ where $L \geq s$. The domain of $\Delta$ is a set $\Delta = \{D(\delta) \mid \delta \in \mathcal{D}\}$, where $D(\delta)$ is a rank $s$ linear transformation from $\mathbb{R}^s$ to $\mathbb{R}^{L \times L}$ (e.g., $D(\delta) = \text{diag}(\delta_{1r_1}, \ldots, \delta_{sr_s})$). Sometimes, this thesis will refer to the subspace occupied by $\Delta$ as $\mathbb{L}_s \subset \mathbb{R}^{L \times L}$. The LFT form of the primal parametric convex program is

$$\inf_{x \in \mathbb{R}^m} \ c(\Delta)^T x$$

subject to \ $X = \mathcal{F}_0(\Delta) + F(\Delta)x$

(3.2)

\[ \mathcal{X} \in \mathcal{K} \]

where $c(\Delta)$, $\mathcal{F}_0(\Delta)$ and $F(\Delta)$ are each LFTs of $\Delta$. Throughout this thesis, it is assumed that any parametric program can be placed in this form. The LFT form of the primal parametric program is especially important in Section 3.2, where it is used to find robustly
feasible solutions $x$ which satisfy the constraints for all $\Delta \in \Delta$. In Section 2.2, it was demonstrated that the dual program could be described in primal form, making the two problems entirely symmetric. This fact is generalized for the parameterized program below. The following two lemmas appeared in [DPZ91], and are easily verified.

**Lemma 3.2.** Suppose $F_t(M, \Delta)$ is square and well-defined for all $\Delta \in \mathcal{D}$, and $M_{11}$ is nonsingular. Then

$$F_t(M, \Delta)^{-1} = F_t(\hat{M}, \Delta)$$

with $\hat{M}$ given by

$$\hat{M} = \begin{bmatrix} M_{11}^{-1} & -M_{11}^{-1}M_{12} \\ M_{21}M_{11}^{-1} & M_{22} - M_{21}M_{11}^{-1}M_{12} \end{bmatrix}.$$ 

**Lemma 3.3.** Given two LFT realizations $F_t(M, \Delta_1)$, $F_t(N, \Delta_2)$, then the cascade has a realization

$$F_t(M, \Delta_1)F_t(N, \Delta_2) = F_t(P, \Delta)$$

with

$$P = \begin{bmatrix} M_{11}N_{11} & M_{12} & M_{11}N_{12} \\ M_{21}N_{11} & M_{22} & M_{21}N_{21} \\ N_{21} & 0 & M_{22} \end{bmatrix}, \quad \Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix}.$$ 

These lemmas enable a parametric dual problem to be written in the “primal” LFT form (3.2), in the same manner as given in Section 2.2. To see how this transformation from dual to “primal” LFT form can be done, consider the dual parametric program

$$\sup_z -\langle \mathcal{F}_0(\Delta), z \rangle$$

subject to

$$\mathbf{F}^T(\Delta)z = c(\Delta), \quad z \in K^*.$$ (3.3)

and assume that $\mathbf{F}(\Delta)$ does not lose rank on the domain $\Delta$. The parameterized hyperplane $\mathbf{F}^T(\Delta)z = c(\Delta)$ is equivalently described by $z = \mathcal{G}_0(\Delta) + \mathcal{G}(\Delta)y$ where

$$\mathcal{G}_0(\Delta) = (\mathbf{F}^+(\Delta))^T c(\Delta),$$

$$\mathcal{G}(\Delta) = I - (\mathbf{F}^+(\Delta))^T \mathbf{F}^T(\Delta).$$
The Moore-Penrose inverse is constructed as

\[ F(\Delta)^+ = (F^T(\Delta)F(\Delta))^{-1}F^T(\Delta). \]

This inverse is constructed using LFT cascading and square LFT inversion, assuming \( F(0) \) has rank \( m \). \( S_0 \) and \( G \) are therefore created by cascading four LFTs with domain \( \Delta \). Finally, \( d(\Delta) = -\langle F_0(\Delta), S_0(\Delta) \rangle \) and \( b(\Delta) = -G^T(\Delta)F_0(\Delta) \), cascading a fifth LFT with domain \( \Delta \). These new LFTs are well-defined since it is assumed that \( F(\Delta) \) does not lose rank on \( \Delta \). Thus, (3.3) is equivalently described as

\[
\inf_y b(\Delta)^T y + d(\Delta)
\]

subject to \( Z = S_0(\Delta) + G(\Delta)y \)

(3.4)

\[ Z \in K^*, \]

where \( b(\Delta), d(\Delta), S_0(\Delta), \) and \( G(\Delta) \) are all LFT functions. This shows constructively how a dual program might be placed in "primal" LFT form, although this construction does not in general lead to the minimal representation. Using this transformation, robustly feasible solutions to both the primal and dual problems can be found using the techniques described in Section 3.2.

### 3.1.1 A Special Class of Parametric Problems

The results given throughout this chapter will often become greatly simplified when restricting the parameterization to the case where \( c(\delta) \) and \( F_0(\delta) \) are affine functions \( \delta \), and \( F \) is constant. It is worth noting that a dual problem with this parameterization

\[
\sup_Z -\langle F_0(\delta), Z \rangle \\
\text{subject to } F^T Z = c(\delta), \\
Z \in K^*
\]

can be written in "primal" form, belonging to the same restricted class of parametric problems:

\[
\inf_y b(\delta)^T y + d(\delta) \\
\text{subject to } Z = S_0(\delta) + G y \\
Z \in K^*
\]

where \( b(\delta), d(\delta), S_0(\delta), \) and \( G(\delta) \) are all LFT functions.
3.2 Robustly Feasible Solutions

The robust convex optimization problem was independently addressed by El Ghaoui, Oustry and Lebret in [GOL99] and by Ben-Tal and Nemirovskii in [BTN96], who each derived a different set of conditions for the robustness problem. The robustness conditions proposed by El Ghaoui et al. are based on an LFT model of the parameter, whereas Ben-Tal and Nemirovskii use an affine parameter dependence. Although it is presently unknown which set of conditions yield more accurate results [BTGN00], the robustness conditions proposed by El Ghaoui are used in this thesis since they seem more general. These conditions have long been known in the control community, where they have been used to show stability robustness to linear fractional perturbations in the system dynamics (see [BEFB94] and the references within).

A robustly feasible solution to a parametric convex program is a point $x$ which satisfies

$$\mathcal{F}_0(\delta) + F(\delta)x \in \mathcal{K}, \text{ for all } \delta \in \mathcal{D}. $$

This section considers the LFT parameterization of a linear matrix inequality (LMI) rather than in a general symmetric cone form. The LMI form is not seen as restrictive in this section, since any constraint over a symmetric cone can be expressed by well-known reductions as semidefinite constraints, although these reductions will not in general yield the smallest cone order $\theta$. A symmetric matrix function $F(x, \Delta)$ is parameterized by the symmetric LFT

$$F(x, \Delta) = M_{11}(x) + M_{12}(I - M_{22}\Delta)^{-1}M_{21}(x) + M_{22}(x)\Delta(I - M_{22}\Delta)^{-T}\Delta^TM_{12}^T \quad (3.5)$$

where $M_{11} : \mathbb{R}^m \to \mathbb{S}^n$ and $M_{21} : \mathbb{R}^m \to \mathbb{R}^{L \times n}$ are affine functions, and $M_{12} \in \mathbb{R}^{n \times L}$, $M_{22} \in \mathbb{R}^{L \times L}$. The symmetric LFT can be written in standard LFT notation by means of the transformation

$$F(x, \Delta) = M_{11} + \tilde{M}_{12}\Delta(I - \tilde{M}_{22}\Delta)^{-1}\tilde{M}_{12}^T,$$
The following proposition presents the heart of the robust semidefinite programming work of El Ghaoui et al. Its proof is based on an application of the $S$-procedure, which is a method for determining if a quadratic form is nonnegative when other quadratic forms are nonnegative. (An overview and history of the $S$-procedure, especially as it relates to control problems, is given in Chapter 2 of [BEFB94]).

**Proposition 3.4.** Let $M_{11}(x) = M_{11}(x)^T$, $M_{12}$, $M_{22}$, $M_{21}(x)$ represent the parameterized LMI (3.5). Let $\mathcal{L}_s$ be a subspace of $\mathbb{R}^{L \times L}$, and associate to $\mathcal{L}_s$ the matrix subspaces

$$
\mathcal{S} = \{ S \in \mathbb{R}^{L \times L} \mid S\Delta = \Delta S \text{ for all } \Delta \in \mathcal{L}_s \}
$$

$$
\mathcal{T} = \{ T \in \mathbb{R}^{L \times L} \mid T\Delta = -\Delta^T T^T \text{ for all } \Delta \in \mathcal{L}_s \}.
$$

Then $\det(I - M_{22}\Delta) \neq 0$ and $F(x, \Delta) \succeq 0$ for every $\Delta \in \mathcal{L}_s$ such that $\sigma_{\text{max}}(\Delta) \leq 1$ if there exist matrices $S \in \mathcal{S}$, $T \in \mathcal{T}$ such that

$$
\begin{bmatrix}
M_{11}(x) - M_{12}SM_{12}^T & M_{21}(x)^T - M_{12}SM_{22}^T + M_{12}T \\
M_{21}(x) - M_{22}SM_{12}^T + TM_{12}^T & S - TM_{22}^T + M_{22}T - M_{22}SM_{22}^T
\end{bmatrix} \succeq 0, \quad S \succ 0.
$$

(3.6)

If $\mathcal{L}_s = \mathbb{R}^{L \times L}$, the condition is necessary and sufficient.

**Proof.** See [GL97].

Robustness conditions such as Proposition 3.4 are common in the control literature (e.g., see [FTD91]). The general problem is known to be NP-hard, therefore Proposition 3.4 will be conservative in many cases. The strict inequalities in (3.6) guarantee that $\det(I - M_{22}\Delta) \neq 0$ for all $\Delta \in \mathcal{L}_s$ with $\sigma_{\text{max}}(\Delta) \leq 1$.

Typically, the structure of $\mathcal{L}_s$ will be block diagonal. For example, the structure of $\mathcal{L}_s$ might be

$$
\mathcal{L}_s = \{ \text{diag}(\Delta_1, \ldots, \Delta_p, \delta_1I_{s_1}, \ldots, \delta_qI_{s_q}) \mid \Delta_i \in \mathbb{R}^{r_i \times r_i}, \delta_i \in \mathbb{R} \}.
$$
In this case, the subspaces $S$ and $T$ of the previous proposition would be

$$S = \{ \text{diag}(\tau_1 I_{r_1}, \ldots, \tau_p I_{r_p}, S_1, \ldots, S_q) \mid \tau_i \in \mathbb{R}, S_i = S_i^T \in \mathbb{R}^{s_i \times s_i} \},$$

$$T = \{ \text{diag}(0_{(r_1+\ldots+r_p)\times(r_1+\ldots+r_p)}, T_1, \ldots, T_q) \mid T_i = -T_i^T \in \mathbb{R}^{s_i \times s_i} \}.$$

### 3.3 Condition Measures

An ill-posed problem has the property that arbitrarily small changes in the problem data exist that will yield both feasible problem instances as well as infeasible instances. The set of all ill-posed problems can be regarded as the boundary between feasible problems and infeasible problems. In [Ren94], Renegar introduced the concept of the “distance to ill-posedness,” which is the minimum size perturbation to a problem necessary to make it ill-posed. Renegar then demonstrated some fundamental relationships between the distance to ill-posedness and the solution characteristics. Several of his main results are presented below.

Let $\mathcal{P}_{PF}$ be the set of all data instances for which the primal problem is feasible, and define $\mathcal{P}_{DF}$ similarly for the set of dual feasible data instances, i.e.,

$$\mathcal{P}_{PF} \triangleq \{(c, \mathcal{F}_0, F) \in \mathcal{P} \mid \exists x \in \mathbb{R}^m, \mathcal{F}_0 + Fx \in \mathcal{K}\},$$

$$\mathcal{P}_{DF} \triangleq \{(c, \mathcal{F}_0, F) \in \mathcal{P} \mid \exists z \in \mathcal{K}^*, F^Tz = c\}.$$

The complements of these feasible sets, $\mathcal{P}_{PF}^C$ and $\mathcal{P}_{DF}^C$, are the sets of primal and dual infeasible problems respectively.

The primal distance to ill-posedness is defined by the expression

$$\rho_F(p) \triangleq \begin{cases} 
\inf_{\tilde{p} \in \mathcal{P}_{PF}} ||p - \tilde{p}|| & \text{if } p \in \mathcal{P}_{PF}, \\
\inf_{\tilde{p} \in \mathcal{P}_{PF}} ||p - \tilde{p}|| & \text{if } p \in \mathcal{P}_{PF}^C.
\end{cases}$$

where the norm $||p||$ is defined by

$$||p|| \triangleq \max\{||c||, ||\mathcal{F}_0||, ||F||\}$$

and $||F|| \triangleq \max_{||x|| \leq 1} ||Fx||$ is the standard operator norm. It should be mentioned that the results which follow can be based on any choice of norm for $p$. 
The analogous definition exists for the dual distance to ill-posedness $\rho_D(p)$. The distance to ill-posedness is defined to be the smallest of the primal and dual distances, i.e.,

$$\rho(p) = \min\{\rho_P(p), \rho_D(p)\}.$$

Let $\text{Opt}_P(p)$ and $\text{Opt}_D(p)$ denote the optimal solution sets for the primal and dual problems. In general, it is possible that $\text{Opt}_P(p)$ or $\text{Opt}_D(p)$ are empty, even for $p \in \mathcal{P}_{PF} \cap \mathcal{P}_{DF}$. This can happen for ill-posed problems.

**Theorem 3.5.** The following statements are true for any problem $p \in \mathcal{P}$:

(i) If $p \in \mathcal{P}_{PF}$ with $\rho_P(p) > 0$, then there exists a feasible $x$ such that

$$\|x\| \leq \|\mathcal{F}_0\| \rho_P(p).$$

(ii) If $p \in \mathcal{P}_{DF}$ with $\rho_D(p) > 0$, then there exists a feasible $Z$ such that

$$\|Z\| \leq \|c\| \rho_D(p).$$

(iii) If $p \in \mathcal{P}_{PF} \cap \mathcal{P}_{DF}$ with $\rho_P(p) > 0$ and $\rho_D(p) > 0$, then $\text{Opt}_P(p) \neq \emptyset$, $\text{Opt}_D(p) \neq \emptyset$, and

$$\|x^*\| \leq \|\mathcal{F}_0\| \frac{\|p\|}{\rho_P(p)\rho_D(p)} \text{ for all } x^* \in \text{Opt}_P(p)$$

$$\|Z^*\| \leq \|c\| \frac{\|p\|}{\rho_P(p)\rho_D(p)} \text{ for all } Z^* \in \text{Opt}_D(p).$$

**Proof.** See [Ren94].

The condition number of a data instance is a scale-invariant reciprocal of the distance to infeasibility. It was introduced by Renegar in [Ren95a], and is defined as

$$C(p) = \frac{\|p\|}{\rho(p)}.$$  

This condition number is analogous to the matrix condition number, which measures how close a matrix is to singularity. Like the matrix condition number, the optimization condition number satisfies $C(p) \geq 1$ for all well-posed problems. This is because the data instance $(0, 0, 0)$ is ill-posed, therefore $\rho(p) \leq \|p\|$. 

3.3. CONDITION MEASURES

In some cases, it is more convenient to work with a problem's condition number rather than its distance to infeasibility, since \( C(p) \) does not depend on the size of \( p \). For example, given a problem \( p \in \mathcal{P}_{PF} \cap \mathcal{P}_{DF} \), the bounds in Theorem 3.5 can be recast as (i) \( \|x\| \leq C(p) \) for some feasible \( x \); (ii) \( \|Z\| \leq C(p) \) for some feasible \( Z \); and (iii) \( \|x^*\| \leq C^2(p) \) and \( \|Z^*\| \leq C^2(p) \) for all optimal solutions \( x^* \) and \( Z^* \).

### 3.3.1 Structured Distance to Ill-Posedness

The distance to ill-posedness introduced above is unstructured in the sense that it considers perturbations in any direction of \( \mathcal{P} \). Suppose now that perturbations are restricted to some subspace of \( \mathcal{P} \). This restriction leads to the concept of a structured distance to ill-posedness \( \rho_s(p) \) which would satisfy \( \rho_s(p) \geq \rho(p) \). The structured distance to ill-posedness is analogous to the structured singular value (see [ZDG96]), which measures the distance of a matrix to singularity, restricting perturbations to a subspace.

Problem structure is useful when considering parametric sets of convex programs. Suppose that \( p(\delta) \) spans a linear subspace of \( \mathcal{P} \), and that a structured distance to ill-posedness \( \rho_s(p(\delta)) \) is known for a particular data instance \( p(\delta) \). Then \( p(\delta) \) is feasible and well-posed for all \( \delta \) such that \( \|p(\delta) - p(\bar{\delta})\| < \rho_s(p(\bar{\delta})) \). This is useful for determining the feasibility of a bounded set of data instances \( \mathcal{P}_D \). An algorithm for determining the smallest distance to ill-posedness for a set of problems is presented in Section 3.3.3.

### 3.3.2 Computation of the Condition Number

To quantify the distance to ill-posedness, it is useful to place the problem constraints in a more general setting. Consider the constraints

\[
\mathcal{F}_0 + \mathbf{F}x \in \mathcal{K}_1 \\
x \in \mathcal{K}_2
\]

where \( \mathcal{K}_1 \) and \( \mathcal{K}_2 \) are convex cones. Note that (3.7) is equivalent to the "primal" constraints if \( \mathcal{K}_1 = \mathcal{K}, \mathcal{K}_2 = \mathbb{R}^n \), and (3.7) has an equivalent form to the "dual" constraints if \( \mathcal{K}_1 = \{0\}, \mathcal{K}_2 = \mathcal{K}^* \). Also, the feasibility of (3.7) is equivalent to the feasibility of the
homogenized constraints

\[ \mathcal{T}_0 r + F x \in \mathcal{K}_1 \]
\[ x \in \mathcal{K}_2 \]
\[ r \geq 0 \]
\[ r^2 + \|x\|^2 \leq 1. \]

(3.8)

In this setting, data perturbations can be represented by adding a perturbation to the first line in (3.8). Assuming that the cones \( \mathcal{K}_1 \) and \( \mathcal{K}_2 \) are closed, Renegar proved that the distance to ill-posedness can be found by the mathematical program (see Theorem 3.5 in \[Ren95b\])

\[ \rho(p) = \min_{\|v\| \leq 1} \max_{r,x,\theta} \theta \]
\[ \text{subject to } \mathcal{T}_0 r + F x + \theta v \in \mathcal{K}_1 \]
\[ x \in \mathcal{K}_2 \]
\[ r \geq 0 \]
\[ r^2 + \|x\|^2 \leq 1. \]

(3.9)

This min-max formulation is used by Freund and Vera [FV99b] to compute estimates of the distance to ill-posedness. The accuracy of their methods depend on the norms chosen for \( p \). For example, if the Euclidean norm is chosen for the space \( P \), then an estimate of \( \rho(p) \) can be characterized within a factor of \( \sqrt{\theta} \). Exact characterizations of \( \rho(p) \) can be provided by an appropriate choice of norms. A complexity analysis of these approximation schemes is presented in [FV99a], where the mathematical programming representation of the condition number is solved via an interior-point algorithm. The conclusion of this analysis is that condition number estimation is not much harder than solving the problem itself.

A different approach to computation of \( \rho(p) \) is given by Peña in [Peñ98]. Notice that (3.9) is essentially finding the smallest ball of perturbations \( \{ \theta v \mid \|v\| \leq 1 \} \) which is contained in the constraints. Peña constructs a self-concordant barrier function for the constraints in (3.9), and uses this barrier to find the Dikin's ellipsoid, which is an ellipsoid contained entirely within the constraint boundaries. A lower bound on \( \rho(p) \) is found from
the smallest eigenvalue of this ellipsoid. The Dikin's ellipsoid at the analytic center also has the property that it contains the constraints when enlarged by a factor \((3\bar{d} + 1)\) (see Proposition 2.3.2 in [NN94]), which Peña uses to find an upper bound for \(\rho(p)\). Although the theoretical bounds for this approximation of \(\rho(p)\) are modest, numerical experiments show that his method works extremely well in practice.

The structured distance to ill-posedness problem is posed by restricting the perturbation \(v\) in (3.9) to the appropriate subspace. Peña presents a detailed analysis of structured perturbations in [Peñ00]. The work of Freund and Vera [FV99b] can be extended to find approximate values of the structured distance to ill-posedness. The details of this extension are deferred for further research.

### 3.3.3 Condition Measures for Sets of Problems

In this thesis, sets of problems \(\mathcal{P}_D\) are considered rather than individual problem instances. While the condition number estimation techniques mentioned above apply to individual problems, they can be used to bound \(C(p)\) for any problem belonging to a compact set \(\mathcal{P}_D\). Assuming that all problems belonging to \(\mathcal{P}_D\) are feasible and well-posed, then the following proposition bounds \(C(p)\) for all \(p \in \mathcal{P}_D\) based only on a finite number of problems. This proposition is also valid for the case of structured distance to infeasibility, if \(\mathcal{P}_D\) lies in a linear subspace of \(\mathcal{D}\).

**Proposition 3.6.** Suppose that a set of problems \(\mathcal{P}_D\) is covered by a finite number of sets \(\mathcal{B}_{r_i}(p_i)\) where \(\mathcal{B}_{r_i}(p) \triangleq \{\tilde{p} \mid \|\tilde{p} - p\| \leq r\}\), i.e.,

\[
\mathcal{P}_D \subseteq \bigcup_{i=1}^{N} \mathcal{B}_{r_i}(p_i).
\]

Furthermore, suppose that \(r_i < \rho(p_i)\). Then

\[
\rho(p) \geq \min_{i=1,\ldots,N} \rho(p_i) - r_i,
\]

\[
C(p) \leq \max_{i=1,\ldots,N} \frac{\|p_i\| + r_i}{\rho(p_i) - r_i} \text{ for all } p \in \mathcal{P}_D.
\]

**Proof.** By the triangle inequality,

\[
\|p\| \leq \|p_i\| + r_i,
\]

\[
\rho(p) \geq \rho(p_i) - r_i > 0, \text{ for all } p \in \mathcal{B}_{r_i}(p_i), i = 1, \ldots, N,
\]
function $S := \text{cover}(\mathcal{P})$

$$S := \{B_{r_1}(p_1), \ldots, B_{r_N}(p_N)\} = \text{refine}(\mathcal{P})$$

for $i = 1$ to $N$

if $r_i \geq \rho(p_i)$

then $S := \{S \setminus B_{r_i}(p_i)\} \cup \text{cover}(B_{r_i}(p_i))$

end

Figure 3.1: Recursive function to find a valid cover for Proposition 3.6.

and thus

$$C(p) \leq \frac{||p_i|| + r_i}{\rho(p_i) - r_i} \text{ for all } p \in B_{r_i}(p_i), i = 1, \ldots, N.$$ 

It is important to observe that if $\mathcal{P}_D$ is a compact set of feasible, well-posed problems, then such a covering always exists, since $\inf_{p \in \mathcal{P}_D} \rho(p) > 0$. Also, the estimate improves as the radii of the cover go to zero, at the expense of requiring many more sets $B_{r_i}(p_i)$ to cover $\mathcal{P}_D$. Figure 3.1 describes a recursive function called $\text{cover}$ which finds a valid cover for Proposition 3.6. The subroutine $\text{refine}$ finds a cover $B_{r_i}$ for an input set $\mathcal{P}$ such that $\mathcal{P} \subseteq \bigcup_{i=1}^{N} B_{r_i}$, and that the radii $r_i$ converge to zero as a set is successively refined.

This method foreshadows a technique which will play an important role in this thesis, specifically decomposing a space of parameterized problems into a collection of smaller subsets. This technique is of fundamental importance to the infeasibility detection algorithm introduced in Section 3.4.

One other observation to make of this method is that the number of balls required to cover $\mathcal{P}_D$ grows exponentially with the dimension of $\mathcal{P}_D$. While this appears to grow forbiddingly high rather quickly, it will be recalled that the dimension of $\mathcal{P}_D$ grows according to the dimension of the parameter space $\mathcal{D}$, which is usually relatively small compared to the dimension of the problem space $\mathcal{P}$. Furthermore, the number of sets
3.3. CONDITION MEASURES

$B_r(p)$ required to cover a well-posed set of problem instances $\mathcal{P}_D$ may be drastically decreased if the structured distance to ill-posedness is used.

3.3.4 Largest Condition Number for a Special Class of Parametric Problems

For many parametric problems, $c$ and $\mathcal{F}_0$ have an affine dependence on the parameter $\delta$, and $\mathbf{F}$ is fixed, independent of $\delta$. In this case, bounding the condition number may be a great deal easier than the technique suggested by Proposition 3.6. The following proposition shows that bounding the condition number can be reduced to checking the vertices of a parameter polytope.

**Proposition 3.7.** Let $p(\delta) = (c(\delta), \mathcal{F}_0(\delta), \mathbf{F})$, where $c(\delta)$ and $\mathcal{F}_0(\delta)$ are affine functions of $\delta$, and $\mathbf{F}$ is constant. Let the domain of $p(\delta)$ be the polytope $\mathcal{D}$ with vertices $\delta_1, \ldots, \delta_L$, and suppose that $p(\delta_1), \ldots, p(\delta_L) \in \mathcal{P}_F \cap \mathcal{P}_D$. Then

$$C(p(\delta)) \leq \frac{\max_{1 \leq i \leq L} ||p(\delta_i)||}{\min_{1 \leq i \leq L} \rho(p(\delta_i))} \text{ for all } \delta \in \mathcal{D}.$$

**Proof.** Since $p(\delta)$ is an affine function, then the norm $||p(\delta)||$ is a convex function, and the maximum value on $\mathcal{D}$ is $\max_{1 \leq i \leq L} ||p(\delta_i)||$. Lemma 3.9 given below proves the quasi-concavity of $\rho(p(\delta))$ on $\mathcal{D}$, thus the minimum value on $\mathcal{D}$ is $\min_{1 \leq i \leq L} \rho(p(\Delta_i))$. The condition number bound follows from these two values. \qed

**Lemma 3.8.** Let $\mathcal{F}_0(\delta)$ be an affine function of $\delta$, $\mathbf{F}$ be constant, and $\mathcal{K}$ convex. Then the set $\mathcal{D}_F = \{ \delta \mid \exists x, \mathcal{F}_0(\delta) + \mathbf{F}x \in \mathcal{K} \}$ is convex.

**Proof.** Assume $\mathcal{D}_F$ is nonempty, and choose $\delta_1$ and $\delta_2$ in $\mathcal{D}_F$. Choose $x_1, x_2$ such that $\mathcal{F}_0(\delta_1) + \mathbf{F}x_1 \in \mathcal{K}$ and $\mathcal{F}_0(\delta_2) + \mathbf{F}x_2 \in \mathcal{K}$. Then $\lambda \mathcal{F}_0(\delta_1) + (1-\lambda)\mathcal{F}_0(\delta_2) + \mathbf{F}(\lambda x_1 + (1-\lambda)x_2) \in \mathcal{K}$ for all $\lambda \in [0, 1]$. Hence, $\lambda \delta_1 + (1-\lambda)\delta_2 \in \mathcal{D}_F$. \qed

Note that convexity does not hold in general, even when $p(\delta)$ is an affine function of $\delta$. This is demonstrated by Example 3.1 given near the end of this chapter.

**Lemma 3.9.** Let $p(\delta) = (c(\delta), \mathcal{F}_0(\delta), \mathbf{F})$, where $c(\delta)$ and $\mathcal{F}_0(\delta)$ are affine functions of $\delta$, and $\mathbf{F}$ is constant. Then $\rho(p(\delta))$ is a quasi-concave function of $\delta$ on the domain $\{ \delta \mid p(\delta) \in \mathcal{P}_F \cap \mathcal{P}_D \}$. 
Proof. Choose $\delta_1$ and $\delta_2$ such that $p(\delta_1), p(\delta_2) \in \mathcal{P}_PF \cap \mathcal{P}_DF$. To establish the quasi-concavity of $\rho(p(\delta))$, it is necessary to examine the behavior of $\rho(p(\delta))$. Suppose that $\rho(p(\delta))$ is not quasi-concave. Then there exists a $\tilde{\lambda} \in (0, 1)$ such that

$$\rho(p(\tilde{\lambda}\delta_1 + (1 - \tilde{\lambda})\delta_2)) < \min\{\rho(p(\delta_1)), \rho(p(\delta_2))\}.$$ 

This implies that there exists a perturbation $(\Delta \mathbf{F}_0, \Delta \mathbf{F})$ such that

$$\{x | \mathbf{F}_0(\tilde{\lambda}\delta_1 + (1 - \tilde{\lambda})\delta_2) + \Delta \mathbf{F}_0 + (\mathbf{F} + \Delta \mathbf{F})x \in \mathcal{K}\} = \emptyset$$ \hspace{1cm} (3.10)

$$\max\{||\Delta \mathbf{F}_0||, ||\Delta \mathbf{F}||\} < \min\{\rho(p(\delta_1)), \rho(p(\delta_2))\}.$$ \hspace{1cm} (3.11)

Let $\mathcal{D}_F = \{\delta | \exists x, \mathbf{F}_0(\delta) + \Delta \mathbf{F}_0 + (\mathbf{F} + \Delta \mathbf{F})x \in \mathcal{K}\}$. Line (3.10) implies that $\tilde{\lambda}\delta_1 + (1 - \tilde{\lambda})\delta_2 \not\in \mathcal{D}_F$, and (3.11) implies that $\delta_1, \delta_2 \not\in \mathcal{D}_F$. This contradicts Lemma 3.8, which states that $\mathcal{D}_F$ must be convex. Therefore, $\rho(p(\delta))$ must be quasi-concave. By the same line of reasoning, $\rho_D(p(\delta))$ is also quasi-concave, hence $\rho(p(\delta))$ is quasi-concave. \hfill $\square$

### 3.4 Infeasibility Detection for Parametric Programs

It is critical that all problem instances which might arise during the implementation of an on-line optimization procedure be feasible. Infeasible problem instances may be disastrous to operation, since the on-line optimization procedure is expected to provide suboptimal solutions at certain scheduled deadlines. It may be clear from the context of the problem that infeasible instances would never occur. For example, a receding horizon controller might have time-varying saturation constraints, yet always have the origin as a feasible control input. However, in general it will not be clear whether a parameterized problem will always be feasible. Considering the same receding horizon controller, suppose state constraints are added to the problem. In this situation, there may exist initial conditions for which every trajectory generated by a feasible control input eventually violates the state constraints. Such an initial condition leads to an infeasible optimization problem, and must be avoided during implementation.

In Section 3.3.3, a method was outlined for determining the conditioning of sets of problems. By showing that the (structured) distance to ill-posedness for every problem...
belonging to a compact, connected set is positive, it is inferred that every problem in this set is feasible and well-posed.

In this section, a different approach is suggested. An infeasibility detection algorithm is presented, which relies on the robust solutions presented in Section 3.2. This method uses LMI constraints parameterized by a symmetric LFT, defined by (3.5). For this reason, the constraints in this section are assumed to be the parametric LMI \( F(x, \Delta) \geq 0 \), parameterized by a matrix \( \Delta \) which lies in a rank \( s \) subspace \( L_s \subset \mathbb{R}^{L \times L} \) rather than a vector \( \delta \in \mathbb{R}^s \). Sometimes, it will be necessary to alternately refer to the vector \( \delta \) and matrix \( \Delta \), so the linear map \( D : \delta \mapsto \Delta \) is defined on a compact parameter set \( D \). Typically, \( D \) maps \( \delta \) into a diagonal matrix. The image of \( D \) is the set \( \Delta = \{ D(\delta) \mid \delta \in D \} \).

The objective of the infeasibility detection algorithm can be summarized as follows:

**Objective.** Determine that a parameterized LMI \( F(x, \Delta) \geq 0 \) is feasible for all \( \Delta \in \Delta \), or find a parameter \( \Delta' \in \Delta \) for which \( \{ x \mid F(x, \Delta') \geq 0 \} = \emptyset \).

As will be seen, the algorithm presented succeeds in this task, except can fail in the situation when there exist ill-posed problem instances in \( \Delta \), and every infeasible problem instance is also ill-posed.

In this section, the notation \( \rho(F(\cdot)) \) is used to indicate the distance from ill-posedness for the constraint \( F(x) \geq 0 \), analogous to the measure \( \rho_F(p) \) defined in the previous section.

Several definitions are made to facilitate an infeasibility detection algorithm. Some of these definitions can be found in [HT96].

**Definition.** A partition of a set \( \mathcal{A} \) is a collection of subsets \( \{ \mathcal{A}_i \mid i \in I \} \) referenced by a finite set of indices \( I \) with the properties

\[
\mathcal{A} = \bigcup_{i \in I} \mathcal{A}_i \quad \text{and} \quad \mathcal{A}_i \cap \mathcal{A}_j = \partial \mathcal{A}_i \cap \partial \mathcal{A}_j \quad \text{for all } i, j \in I, i \neq j,
\]

where \( \partial \mathcal{A}_i \) denotes the boundary of \( \mathcal{A}_i \).

The partitions used in this section are hyper-rectangles \( \mathcal{M} = \{ x \in \mathbb{R}^s \mid a \leq x \leq b \} \), \( a, b \in \mathbb{R}^s \), \( a < b \). These rectangles are mapped into the \( \Delta \) parameter space as \( D(\mathcal{M}) \).

The diameter \( d(\mathcal{M}) \) of a rectangle is defined to be the length of the longest edge.
A partition is refined by a subdivision operation, which replaces a partition element by a partition of that element. For the partitions used in this section, any partition element can be further refined. By successively refining partition elements, it is possible to generate infinitely decreasing sequences \( \{M_k\} \), i.e., infinite sequences of partition elements which satisfy \( M_{k+1} \subset M_k \).

**Definition.** A subdivision is called exhaustive if every infinitely decreasing sequence \( \{M_k\} \) of partition elements generated by the subdivision satisfies \( \lim_{k \to \infty} d(M_k) = 0 \), where \( d(M) \) indicates the diameter of \( M \) measured by some metric.

This section focuses on the bisect subdivision, although any exhaustive subdivision operation can be used in the upcoming algorithm. The subdivision bisect partitions a given rectangle \( M \) by means of the operation

\[
\text{bisect}(M) \triangleq \{M_\alpha, M_\beta\}
\]

where \( M_\alpha \) and \( M_\beta \) are rectangles of equal volume which form a partition of \( M \) by bisecting \( M \) at the midpoint of one of its longest edges. It is not difficult to show that bisect is exhaustive.

For each rectangle \( M \), define a related parameter set

\[
\Delta_M \triangleq \{\Delta \in L_s \mid \sigma_{\max}(\Delta - \Delta_M) \leq r_M\}
\]

where \( \Delta_M = D(\delta_M) \), \( \delta_M \) is the center of the rectangle \( M \), and \( r_M \) is the smallest radius such that \( \sigma_{\max}(D(\delta - \delta_M)) \leq r \) for all \( \delta \in M \). The parameter set \( \Delta_M \) is in the form required by Proposition 3.4 in Section 3.2 to find a robust solution. Note that \( r_M \to 0 \) as \( d(M) \to 0 \).

The infeasibility detection algorithm \( \text{INF-DET}(\gamma) \) algorithm is introduced in Figure 3.2. It decides the feasibility or infeasibility of a parameterized problem by partitioning the parameter space into a collection of sets for which a robust solution exists for each set. The algorithm input \( \gamma \) is used to set the tolerance of the algorithm to problems which are close to ill-posedness. If all problems belonging to a parameterized set are either feasible or ill-posed, then this set of problems may be undecidable by this algorithm.
Algorithm INF-DET(γ)

initialize $S := \{M_0\}$, such that $\Delta_{\mathcal{M}_0} \supseteq \Delta$; $\mathcal{F} := \emptyset$; $U := \emptyset$

while $S \neq \emptyset$

choose $\tilde{M} \in S$

$S := S \setminus \{\tilde{M}\}$

$\{\mathcal{M}_\alpha, \mathcal{M}_\beta\} = \text{bisect}(\tilde{M})$

for $i = \alpha, \beta$

if $\Delta \cap \Delta_{\mathcal{M}_i} \neq \emptyset$ {

choose $\Delta \in \Delta \cap \Delta_{\mathcal{M}_i}$

if $F(x, \Delta) \geq 0$ is infeasible then output(INF); stop

else if $\exists \tilde{x}$ such that $F(\tilde{x}, \Delta) \geq 0$ for all $\Delta \in \Delta_{\mathcal{M}_i}$

then $\mathcal{F} := \mathcal{F} \cup \{\mathcal{M}_i\}$

else if $d(\mathcal{D}_i) \leq \gamma$ then $U := U \cup \{\mathcal{M}_i\}$

else $S := S \cup \{\mathcal{M}_i\}$

}

end

end

if $U = \emptyset$ output(FEAS)

else output(UND)

Figure 3.2: Infeasibility Detection Algorithm.
This means that the algorithm will output “undecidable” (UND) for all $\gamma > 0$, and will not terminate in finite time for $\gamma = 0$. For all other cases, the algorithm will terminate in finite time for all $\gamma \geq 0$, and will decide “feasible” (FEAS) or “infeasible” (INF) if $\gamma$ is small enough. The algorithm also exits with sets of partition elements $\mathcal{F}$ and $\mathcal{U}$. If the algorithm terminates with “infeasible”, then an infeasible problem instance has been found. Otherwise, $\mathcal{F} \cup \mathcal{U}$ forms a partition which covers the $\delta$ parameter space $\mathcal{D}$, where the parameter sets in $\mathcal{F}$ are feasible, and the parameter sets in $\mathcal{U}$ are undecidable by the algorithm. These properties of the algorithm are clarified by the following propositions.

**Proposition 3.10.** For all $\gamma > 0$, $\text{INF-DET}(\gamma)$ terminates in finite time.

*Proof.* Assume $\text{INF-DET}(\gamma)$ does not terminate. Then the step $S := S \cup \{M_i\}$ must be called an infinite number of times. Let $S_k$ denote $S$ at iteration $k$, and $\tilde{S} = \bigcup_{k=2}^{\infty} S_k$ is an infinite family of sets. It is possible to choose a sequence of sets $\{M_k\}$ in $\tilde{S}$ such that

$$M_1 \supseteq M_2 \supseteq M_3 \supseteq \cdots.$$  

However, since these sets were generated by bisect, and therefore $\lim_{q \to \infty} d(M_q) = 0$, there must exist a set $M_k$ such that $\delta(M_q) < \gamma$. This is a contradiction, since this set could not be a member of $\tilde{S}$. \hfill \Box

**Proposition 3.11.** For any $\epsilon > 0$, there exists a $\gamma > 0$ such that if $M \in \mathcal{U}$ at the termination of $\text{INF-DET}(\gamma)$, then $\rho(F(\cdot, \Delta)) < \epsilon$ for all $\Delta \in \mathcal{D}_M$.

The following lemma is necessary for the proof of Proposition 3.11. It bounds the distance from ill-posedness for a set of problems if a robust solution cannot be found.

**Lemma 3.12.** Let $\tilde{F}(x, \delta F) = (F_0 + \delta F_0) + \sum_{i=1}^{m} (F_i + \delta F_i)x_i$. Suppose that $\tilde{F}(x, 0) \succeq 0$ is feasible, and that $\{x \mid \tilde{F}(x, \delta F) \succeq 0, \forall \|\delta F_i\| \leq \beta\} = \emptyset$, i.e., there does not exist a robust solution to $\tilde{F}(x, \delta F) \succeq 0$. Then

$$\rho(\tilde{F}(\cdot, \delta F)) < \beta \sqrt{n}(m + 2) + \sqrt{\beta^2 n + \beta \sqrt{n} \|F_0\|_F}.$$  

for all $\|\delta F_i\| \leq \beta$, $i = 0, \ldots, m$. 


Proof. A point \( x \) satisfies

\[
\tilde{F}(x, \delta F) \succeq 0 \quad \text{for all } \| \delta F_i \| \leq \beta, i = 0, \ldots, m
\]

if and only if

\[
\tilde{F}(x, 0) \succeq I \left( \max_{\| \delta F_i \| \leq \beta} \left\| \delta F_0 + \sum_{i=1}^m \delta F_i x_i \right\| \right) = I \beta \sqrt{\|x\|^2 + 1}.
\]

(3.12)

Therefore, (3.12) is infeasible by the lemma assumptions.

Alternatively, the problem \( \tilde{F}(x, 0) \succeq 0 \) can be perturbed by \( \tilde{\rho} \triangleq \rho(\tilde{F}(. , 0)) \) and still remain feasible, thus

\[
\tilde{F}(x, 0) \succeq I \frac{\tilde{\rho}}{\sqrt{n}}
\]

(3.13)

is feasible. By Theorem 3.5, (3.13) is feasible for some \( x \) with

\[
\|x\| \leq \frac{1}{\tilde{\rho}} \left\| F_0 - I \frac{\tilde{\rho}}{\sqrt{n}} \right\|_F \leq \frac{\|F_0\|_F}{\tilde{\rho}} + 1.
\]

Since (3.12) is infeasible and (3.13) feasible, it follows that there exists an \( \|x\| \leq \frac{1}{\tilde{\rho}} \|F_0\|_F \)

\[
\beta \sqrt{\|x\|^2 + 1} > \frac{\tilde{\rho}}{\sqrt{n}}.
\]

Therefore,

\[
\frac{\tilde{\rho}}{\sqrt{n}} < \beta \sqrt{\left( \frac{\|F_0\|_F}{\tilde{\rho}} + 1 \right)^2 + 1} < \beta \left( \frac{\|F_0\|_F}{\tilde{\rho}} + 2 \right),
\]

which implies

\[
\tilde{\rho} < 2\beta \sqrt{n} + \sqrt{\beta^2 n + \beta \sqrt{n} \|F_0\|_F}.
\]

By the triangle inequality, the distance from ill-posedness for all constraints perturbed by \( \| \delta F_i \| \leq \beta \) can be bounded by

\[
\rho(\tilde{F}(\cdot, \delta F)) \leq \tilde{\rho} + m \sqrt{n} \beta \quad \text{for all } \| \delta F_i \| \leq \beta,
\]

which completes the proof. \(\square\)
Proof of Proposition 3.11. Let $\dot{F}_\Delta(x, \delta F) = F(x, \Delta) + \delta F_0 + \sum_{i=1}^m \delta F_i x_i$. By Lemma 3.12, for any $\epsilon > 0$, there exists a $\beta > 0$ such that $\{x \mid \dot{F}_\Delta(x, \delta F) \geq 0, ||\delta F_i|| \leq \beta\} = \emptyset$ and $F(x, \Delta) \geq 0$ feasible implies that $\rho(\dot{F}_\Delta(\cdot, \delta F)) < \epsilon$ for all $||\delta F_i|| \leq \beta$.

Expanding $F(x, \Delta) = F_0(\Delta) + \sum_{i=1}^m F_i(\Delta)x_i$, note that for each $i$, $F_i(\Delta)$ is a continuous mapping from the compact metric space $\Delta$ to $\mathbb{R}^{n \times n}$. Therefore $F_i(\Delta)$ is uniformly continuous on $\Delta$, meaning that for every $\beta > 0$, there exists a $\gamma > 0$ such that

$$||F_i(\Delta_2) - F_i(\Delta_1)|| < \beta$$

for all $\Delta_1, \Delta_2$ in $\Delta$ such that $||\Delta_2 - \Delta_1|| < \gamma$. This implies that given a linear subspace $\mathcal{L}_s$, there exists a $\gamma > 0$ such that

$$\{F(\cdot, \Delta + \delta \Delta) \mid \delta \Delta \in \mathcal{L}_s, ||\delta \Delta|| \leq \gamma\} \subseteq \{\dot{F}_\Delta(\cdot, \delta F) \mid ||\delta F_i|| \leq \beta\}$$

for all $\Delta \in \Delta$.

Choosing this $\gamma$ for $\text{INF-DET}(\gamma)$, suppose that $\mathcal{M} \in \mathcal{U}$. Then $\tau_{\mathcal{M}} \leq \gamma$ and $\{x \mid F(x, \Delta) \geq 0, \Delta \in \mathcal{D}_{\mathcal{M}}\} = \emptyset$. This implies that $\{x \mid \dot{F}_{\mathcal{M}}(x, \delta F) \geq 0, ||\delta F_i|| \leq \beta\} = \emptyset$.

From this, it follows that $\rho(F(\cdot, \Delta)) \leq \epsilon$ for all $\Delta \in \Delta$. \qed

It is not hard to see that if $\text{INF-DET}(\gamma)$ outputs $\text{UND}$ for all $\gamma > 0$, then there exists a parameter in $\mathcal{D}$ which corresponds to an ill-posed problem. However, if $\gamma = 0$, it is possible that the algorithm will not terminate. The following two corollaries are important consequences of Proposition 3.11, which state cases for which $\text{INF-DET}$ will always terminate.

**Corollary 3.13.** If $F(x, \Delta) \geq 0$ is feasible and well-posed for all $\Delta \in \Delta$, then $\text{INF-DET}(0)$ will terminate with $\text{FEAS}$ in finite time.

*Proof.* Let $\rho_{\text{inf}} = \inf_{\Delta \in \Delta} \rho(F(\cdot, \Delta))$. Observe that $\rho_{\text{inf}} > 0$, since the parameterized LMI is feasible and well-posed over the compact region $\mathcal{D}$. Hence, there exists a $\gamma' > 0$ for which $\mathcal{U} = \emptyset$ at the termination of $\text{INF-DET}(\gamma')$, and will thus output $\text{FEAS}$. This must also hold for any $\gamma < \gamma'$. \qed

**Corollary 3.14.** If for some $\Delta \in \Delta$, $F(x, \Delta) \geq 0$ is infeasible and is not ill-posed, then $\text{INF-DET}(0)$ will terminate with $\text{INF}$ in finite time.
Proof. Let $F(x, \tilde{\Delta}) \geq 0$ be an infeasible and well-posed problem instance for some $\tilde{\Delta} \in \Delta$, with $\rho(F(\cdot, \tilde{\Delta})) > 0$. Hence, there exists a $\gamma' > 0$ such that at the termination of $\text{INF-DET}(\gamma')$, $\Delta \notin \Delta_M$ for all $M \in \mathcal{U}$. $\text{INF-DET}(\gamma')$ cannot exit with $\text{UND}$, since $D \subseteq \bigcup_{M \in \mathcal{F}} \Delta_M$ at termination, which leads to a contradiction. Hence, $\text{INF-DET}(\gamma')$ must exit with $\text{INF}$. This also holds for any $\gamma < \gamma'$.

One observation to make about the infeasibility detection algorithm is that it depends on the ability to test for robust solutions. However, it will be recalled that Proposition 3.4 provided only a sufficient condition, not a necessary condition, for the existence of a robust solution. What impact does this have on the infeasibility detection algorithm? Very little, as it turns out. Even though the robustness criterion can be conservative for structured parameterizations, it is still less conservative than the unstructured case, in which case Proposition 3.4 is necessary and sufficient. The proof of Proposition 3.11 still holds if the robustness test is relaxed to the unstructured case. The algorithm may simply have a longer running time, since $D$ may need to be broken into finer pieces before robust solutions are found.

Example 3.1. This example illustrates how $\text{INF-DET}(0)$ may fail to detect infeasibility given a problem with an infeasible ill-posed problem instance. Consider the LMI parameterized by the scalar $\delta$

$$
\begin{bmatrix}
\delta x_1 + (1 - \delta)x_2 & \delta x_2 + (1 - \delta)x_1 + 1 \\
\delta x_2 + (1 - \delta)x_1 + 1 & 1
\end{bmatrix} \succeq 0,
$$

(3.14)

where $\delta$ is restricted to $0 \leq \delta \leq 1$. This problem is feasible and well-posed for all $\delta \in [0, 1/2) \cup (1/2, 1]$ and infeasible for $\delta = 1/2$. (To understand why, notice that (3.14) has two degrees of freedom for all $\delta$ except when $\delta = 1/2$, where the constraint is equivalent to $(x_1 + x_2)^2 + (x_1 + x_2) + 1 \leq 0$ and $x_1 + x_2 \geq 0$.) From the bisection partitioning, this algorithm will test all intervals $[1/2 - 2^{-n}, 1/2]$, $n = 1, 2, \ldots$ for a robust solution or an infeasible center, of which it will find neither. Thus, $\text{INF-DET}(0)$ will never terminate. However, in the limit as the length of the intervals approaches zero, Lemma 3.12 indicates the presence of an ill-posed problem.
3.4.1 Infeasibility Detection for a Special Class of Parametric Problems

As was demonstrated by Example 3.1, the parametric space of feasible problems is not necessarily convex, even when the parameters enter the constraints linearly. However, when \( c(\delta) \) and \( \mathcal{I}_0(\delta) \) have an affine dependence on \( \delta \), and \( F \) is a fixed constant, then the set of feasible parameters \( \{ \delta \mid (c(\delta), \mathcal{I}_0(\delta), F) \in \mathcal{P}_{PF} \cap \mathcal{P}_{DF} \} \) is convex. This property was proved earlier in Lemma 3.8. As a corollary, checking the feasibility of all problems in \( \mathcal{P}_D \) may be considerably easier than applying the infeasibility detection algorithm.

**Corollary 3.15.** Let \( p(\delta) = (c(\delta), \mathcal{I}_0(\delta), F) \), where \( c(\delta) \) and \( \mathcal{I}_0(\delta) \) are affine functions of \( \delta \), and \( F \) is constant. If \( D \) is a polytope with vertices \( \delta_1, \ldots, \delta_L \), then \( p(\delta) \in \mathcal{P}_{PF} \cap \mathcal{P}_{DF} \) for all \( \delta \in D \) if and only if \( p(\delta_i) \in \mathcal{P}_{PF} \cap \mathcal{P}_{DF} \) for \( i = 1, \ldots, L \).

By the previous corollary, exploitation of convexity in the parametric space can significantly reduce the complexity of determining feasibility of a parametric optimization problem.
Chapter 4

Computational Bounds

The main contribution of this thesis is a method for certification of on-line optimization. This is achieved by delivering a provable upper bound on the number of iterations that the algorithm will ever require for a particular application. For the path-following algorithms presented in Chapter 2, finding these bounds requires knowing three different properties: i) how the algorithm is to be initialized; ii) the worst-case convergence rate; and iii) when to terminate the algorithm. Of these three, only the convergence rate has been discussed so far. In this chapter, a few initialization strategies are presented, and it is assumed that the algorithm will terminate once the iterates come within a certain predefined tolerance of the optimal solution. Selection of this tolerance is very problem dependent, and will be visited again in Chapter 6 for receding horizon control. It should also be mentioned that these computational guarantees are based on an assumption of infinite precision computations of the on-line algorithm. Additional assumptions are necessary for these proofs to hold for finite-precision arithmetic (e.g., assumptions on the terminating accuracy), but will not be mentioned for the remainder of this chapter.

Initialization strategies are sometimes separated into two categories: cold starts and warm starts. An initialization is a warm start if it is reasonably close to optimality and can be brought to optimality relatively quickly, otherwise it is a cold start. The initialization schemes proposed in this chapter can be described as cold start methods, but it should be qualified that they are intelligent cold starts. These initializations represent very dependable points from which to start the algorithm. Warm starts are
CHAPTER 4. COMPUTATIONAL BOUNDS

mentioned again at the end of this chapter, but a detailed analysis is deferred to further research.

This chapter considers on-line optimization problems with fixed constraints and variable constraints as separate cases. This is justified because the constraints of many on-line problems are not parameter dependent (e.g., control allocation problems). As the next section demonstrates, there is a rather simple initialization for problems with fixed constraints that leads to a nice iteration bound.

4.1 Bounds for Problems with Fixed Constraints

For many parameterized convex programs, only the objective function $c(\delta)^T x$ of the primal problem is parameter dependent, and the constraints remain independent of $\delta$, as in

$$\min_x c(\delta)^T x$$

subject to

$$X = F_0 + F x$$

$$X \in \mathcal{K},$$

where $\delta$ belongs to some compact set $\mathcal{D}$.

Two assumptions are made about the optimization problem. First, the primal feasible space $\mathcal{F}_P$ is bounded with a nonempty interior. Second, the parametric vector $c(\delta)$ is unrestricted in direction, but is bounded in magnitude, i.e., $r \triangleq \max ||c|| < \infty$. These assumptions ensure that feasibility and strong duality holds over all possible values of $c$. The problem of finding computational bounds for this problem was first discussed by McGovern and Feron in [MF98]. This original treatment was limited to semidefinite programming for a potential reduction algorithm. The bounds presented below are for a path-following algorithm, and is generalized for the class of symmetric cones.

4.1.1 Algorithm Initialization

Perhaps the most difficult part of solving an on-line optimization problem is finding a proper initialization for the algorithm. Most interior-point algorithms require strictly
feasible initializations, and prefer initial points which are close to the central path. This is true for the path-following algorithms given in Chapter 2, which require an initial primal-dual pair which are in the neighborhood $\mathcal{N}(\beta)$ of the central path. The complexity of finding this initialization from scratch is the same as the complexity of solving the optimization problem itself, so it is desirable to do the bulk of this work off-line.

For the fixed constraint case, initialization of the primal variable is far simpler than dual initialization. Not only does the primal feasible set remain constant for all time, there exists a point which is on the primal central path for all possible values of $c$: the analytic center. Recall that this point is defined by the unique minimizer of the barrier function:

$$
\mathcal{X}^* \triangleq \operatorname{argmin}_{\mathcal{X} \in \mathcal{K}} \{ B(\mathcal{X}) \mid \mathcal{X} = \mathcal{J}_0 + \mathbf{F} x, x \in \mathbb{R}^m \}.
$$

The analytic center can be rapidly computed within machine numerical precision using a variant of Newton’s method, which is done off-line. The primal variable is initialized at the analytic center, i.e., $\mathcal{X}(0) = \mathcal{X}^*$.

A good initial value of the dual variable $\mathcal{Z}$ is also required. This variable must be strictly feasible (i.e., $\mathcal{Z} \in \operatorname{int} \mathcal{K}^*$ and $\mathbf{F}^T \mathcal{Z} = c$) and it must be rapidly computable for different values of $c$. Also, initialization of path-following methods require that the pair $(\mathcal{X}(0), \mathcal{Z}(0))$ be within a neighborhood of the central path $\mathcal{N}(\beta)$.

The first step in constructing a dual initialization is to find a $\mathcal{Z}_0$ which lies in the hyperplane $\mathbf{F}^T \mathcal{Z} = c$. This point need not lie in the interior of $\mathcal{K}^*$. Later, a point in the subspace $\mathbf{F}^T \mathcal{Z} = 0$ and in the interior of $\mathcal{K}^*$ will be added to $\mathcal{Z}_0$ to make the initialization strictly feasible. For reasons which will later become clear, $\mathcal{Z}_0$ is defined as the least squares solution to $\mathbf{F}^T \mathcal{Z} = c$ in the norm induced by $[\nabla^2 B(\mathcal{X}^*)]^{-1}$, i.e.,

$$
\mathcal{Z}_0 \triangleq \operatorname{argmin}_{\mathbf{F}^T \mathcal{Z} = c} ([\nabla^2 B(\mathcal{X}^*)]^{-1} \mathcal{Z}, \mathcal{Z})
\quad = \nabla^2 B(\mathcal{X}^*) \mathbf{F} (\mathbf{F}^T \nabla^2 B(\mathcal{X}^*) \mathbf{F})^{-1} c.
$$

The linear transformation $\nabla^2 B(\mathcal{X}^*) \mathbf{F} (\mathbf{F}^T \nabla^2 B(\mathcal{X}^*) \mathbf{F})^{-1}$ is computed once off-line, after which $\mathcal{Z}_0$ can be rapidly computed from $c$.

The dual variable is initialized by

$$
\mathcal{Z}^{(0)}(\gamma) = \mathcal{Z}_0 - \gamma \nabla B(\mathcal{X}^*)
$$
for some $\gamma > 0$. To see that $\mathbf{F}^T \mathcal{Z}^{(0)}(\gamma) = c$ for all $\gamma$, note that from the optimality conditions for $\mathcal{X}^*$

$$
\left( \frac{\partial}{\partial x} B(\mathcal{F}_0 + \mathbf{F}x) \right)_{x=x^*} = \nabla B(\mathcal{X}^*)^T \mathbf{F} = 0, \quad (4.1)
$$

which shows that $\nabla B(\mathcal{X}^*)$ is orthogonal to the rows of $\mathbf{F}^T$. Also, recall the basic property of the barrier function

$$
-\nabla B(\mathcal{X}) \in \text{int} \mathbb{K}^* \text{ for all } \mathcal{X} \in \text{int} \mathbb{K},
$$

therefore $-\nabla B(\mathcal{X}^*) \in \text{int} \mathbb{K}^*$. Since $\mathbb{K}^* \subset \mathbb{E}$ is a convex cone with a nonempty interior, then for any $\mathcal{X} \in \mathbb{E}$ and $\mathbf{y} \in \text{int} \mathbb{K}^*$, there exists a $\gamma \geq 0$ such that $\mathcal{X} + \gamma \mathbf{y} \in \text{int} \mathbb{K}^*$. Hence, there exists a $\gamma$ such that $\mathcal{Z}^{(0)}(\gamma) \in \text{int} \mathbb{K}^*$. Furthermore, it will soon be demonstrated that $\gamma$ can be chosen such that $\langle \mathcal{X}^{(0)}, \mathcal{Z}^{(0)}(\gamma) \rangle$ is arbitrarily close to the central path.

First, the following lemma is required concerning the duality gap.

**Lemma 4.1.** Given initializations $\mathcal{X}^{(0)}$ and $\mathcal{Z}^{(0)}(\gamma)$ as defined above,

$$
\langle \mathcal{X}^{(0)}, \mathcal{Z}^{(0)} \rangle = \vartheta \gamma.
$$

**Proof.** Using the properties $\nabla^2 B(\mathcal{X}) \mathcal{X} = -\nabla B(\mathcal{X})$, $\langle \nabla B(\mathcal{X}), \mathcal{X} \rangle = -\vartheta$, and (4.1), the following equalities hold:

$$
\langle \mathcal{X}^{(0)}, \mathcal{Z}^{(0)} \rangle = \langle \mathcal{X}^*, \nabla^2 B(\mathcal{X}^*) \mathbf{F} (\mathbf{F}^T \nabla^2 B(\mathcal{X}^*) \mathbf{F})^{-1} c - \gamma \nabla B(\mathcal{X}^*) \rangle \\
= \langle \mathbf{F}^T \nabla^2 B(\mathcal{X}^*) \mathcal{X}^*, (\mathbf{F}^T \nabla^2 B(\mathcal{X}^*) \mathbf{F})^{-1} \rangle - \gamma \langle \mathcal{X}^*, \nabla B(\mathcal{X}^*) \rangle \\
= \langle \mathbf{F}^T \nabla B(\mathcal{X}^*), (\mathbf{F}^T \nabla^2 B(\mathcal{X}^*) \mathbf{F})^{-1} \rangle + \gamma \vartheta \\
= \vartheta \gamma.
$$

Initialization of the path-following algorithms presented in Section 2.6 requires that $\langle \mathcal{X}^{(0)}, \mathcal{Z}^{(0)} \rangle$ be contained in a neighborhood of the central path $\mathcal{N}(\beta)$. As the following proposition shows, an appropriate choice of $\gamma$ will guarantee this.

**Proposition 4.2.** Define

$$
\bar{\gamma}(\beta, c) \triangleq \frac{1}{\beta \sqrt{c^T (\mathbf{F}^T \nabla^2 B(\mathcal{X}^*) \mathbf{F})^{-1} c}}
$$
on $\beta \in (0, 1)$. Let $Z^{(0)}(\beta) \triangleq Z^{(0)}(\bar{\gamma}(\beta, c))$. Then

$$(X^{(0)}, Z^{(0)}) \in \mathcal{N}(\beta).$$

**Proof.** The neighborhood $\mathcal{N}(\beta)$ is defined by the set of primal and dual points which satisfy $\| \frac{1}{\mu} Z + \nabla B(X) \|_{\nabla^2 B(X)^{-1}} \leq \beta$ where $\mu \triangleq (X, Z)/\beta$. From Lemma 4.1, $\mu = \tilde{\gamma}(\beta, c)$, which leads to

$$\frac{1}{\mu} Z^{(0)}(\beta, c) + \nabla B(X^{(0)}) = \frac{1}{\tilde{\gamma}(\beta, c)} (Z_0 - \tilde{\gamma}(\beta, c) \nabla B(X^*)) + \nabla B(X^*)$$

$$= \frac{1}{\tilde{\gamma}(\beta, c)} \nabla^2 B(X^*) F (F^T \nabla^2 B(X^*) F)^{-1} c.$$ 

Then

$$\| \frac{1}{\mu} Z^{(0)}(\beta, c) + \nabla B(X^{(0)}) \|^2_{\nabla^2 B(X)^{-1}} = \frac{1}{\tilde{\gamma}^2(\beta, c)} c^T (F^T \nabla^2 B(X^*) F)^{-1} c$$

$$= \beta^2. \qed$$

Since $\mathcal{N}(\beta) \subset \text{int} \mathcal{K} \times \text{int} \mathcal{K}^*$ when $\beta \in (0, 1)$, this initialization is now valid for a path-following algorithm.

### 4.1.2 Iteration Bounds

Given an initialization in the central path neighborhood $\mathcal{N}(\beta)$, the number of iterations required to guarantee an $\epsilon$-optimal solution depends on the initial duality gap and the rate of convergence. Lemma 4.1 shows that the initial duality gap is exactly $\phi\gamma(\beta, c)$. Supposing that the objective vector $c$ is unrestricted in direction but bounded in magnitude by $\|c\| \leq r$, an upper bound for $\tilde{\gamma}(\beta, c)$ follows directly from a singular value decomposition of the matrix $(F^T \nabla^2 B(X^*) F)^{-1}$. An upper bound on the duality gap is summarized by the following proposition.

**Proposition 4.3.** Suppose $\|c\| \leq r$. Let $X^{(0)}$ and $Z^{(0)}$ be initialized as above. Then

$$(X^{(0)}, Z^{(0)}) \leq \frac{\phi r}{\beta} \sqrt{\sigma_{\max}(F^T \nabla^2 B(X^*) F)^{-1}}. \quad (4.2)$$

**Proof.** A direct consequence of Lemma 4.1. \qed
Note that this bound is nonconservative, i.e., there exists a \( c \) with \( ||c|| = r \) such that equality holds for (4.2). Assuming that a given optimization algorithm reduced this duality gap by a known factor at each iteration, then the number of iterations to reduce the duality gap to \( \epsilon \) could be computed nonconservatively. In this thesis, the theoretical ratio \( \mu_{k+1}/\mu_k \leq (1 - \Delta/\sqrt{\theta}) \) is used, which is usually fairly conservative. Using a path-following algorithm which reduces the duality gap by at least this ratio, then \( (X^{(k)}, Z^{(k)}) \leq \epsilon \) for all

\[
k \geq \left\lceil \frac{\log(\beta \epsilon) - \log(\vartheta \sqrt{\sigma_{\text{max}}((F^T V^2 B(X^*)F)^{-1}))}}{\log(1 - \Delta/\sqrt{\theta})} \right\rceil.
\]

Thus, the convex program is solvable in \( O(\sqrt{\theta} \log((\vartheta r/\beta \epsilon) \sqrt{\sigma_{\text{max}}((F^T V^2 B(X^*)F)^{-1}))}) \) iterations.

**Example 4.1.** Quadratic programming frequently arises in on-line optimization, so it makes a useful example for this section. The problem given in Example 4.2 as well as the UAV example given in Chapter 6 will make use of the derivations given here. The quadratic program considered here is

\[
\begin{align*}
\min_{x} \quad & g^T x + \frac{1}{2} x^T G^T G x \\
\text{subject to} \quad & Ax \leq b
\end{align*}
\]

where \( g \) is free to point in any direction, and \( G \in \mathbb{R}^{n \times n}, \ A \in \mathbb{R}^{m \times n}, \ b \in \mathbb{R}^m \) are fixed. The region defined by \( Ax \leq b \) is assumed to be bounded. A constraint with the form \( \gamma \geq ||Gx||^2 \) can be replaced with the second-order cone constraint

\[
\frac{\gamma + 1}{2} \geq \sqrt{\left(\frac{\gamma - 1}{2}\right)^2 + ||Gx||^2}.
\]

This constraint is used to construct a conic convex program which is equivalent to the quadratic program:

\[
\begin{align*}
\min_{x} \quad & g^T x + \frac{1}{2} \gamma \\
\text{subject to} \quad & \begin{bmatrix} \frac{\gamma + 1}{2} & \frac{\gamma - 1}{2} \ x^T G^T \end{bmatrix}^T \succeq 0 \\
& Ax \leq b, \quad \gamma \leq \gamma_{\text{max}}, \quad (4.3)
\end{align*}
\]
where $\gamma_{\text{max}}$ is an upper bound on the optimal value of $x^T G^T G x$. The purpose of $\gamma_{\text{max}}$ is to make the feasible space in (4.3) bounded. Given a bound on $\|x\|$ (this is bounded because the constraints $Ax \leq b$ are assumed to be compact), a valid $\gamma_{\text{max}}$ is readily computed from the maximum singular value of $G$. Supposing that the analytic center of $Ax \leq b$ is $x^* = 0$ then the analytic center of the constraints in (4.3) is $x^* = 0$, $\gamma^* = \gamma_{\text{max}}/2$. The primal variables are initialized at

$$
\begin{bmatrix}
\gamma^* + 1/2 \\
\gamma^* - 1/2 \\
(Gx^*)^T
\end{bmatrix}^T, \\
\begin{bmatrix}
b - Ax^* \\
\gamma_{\text{max}} - \gamma^*
\end{bmatrix}.
$$

The dual problem is

$$
\begin{array}{c}
\max \\
\text{subject to}
\end{array}
\begin{bmatrix}
1/2 & -1/2 & 0 \\
0 & 0 & G
\end{bmatrix}
\begin{bmatrix}
Z_Q^T \\
Z_L^T
\end{bmatrix}
\begin{bmatrix}
b^T \\
\gamma_{\text{max}}
\end{bmatrix}
\begin{bmatrix}
1/2 \\
g
\end{bmatrix},
\begin{array}{c}
Z_Q \in \mathbb{R}^{n+1}_+, \\
Z_L \in \mathbb{R}^{m+1}_+, \\
F_Q^T Z_Q + F_L^T Z_L = 1/2
\end{array}
$$

where

$$
F_Q^T = \begin{bmatrix}
1/2 & 1/2 & 0 \\
0 & 0 & G
\end{bmatrix}, \\
F_L^T = \begin{bmatrix}
0 & -1 \\
-A^T & 0
\end{bmatrix}
$$

The dual initialization is then given by

$$
\begin{bmatrix}
Z_Q^{(0)} \\
Z_L^{(0)}
\end{bmatrix} = \begin{bmatrix}
\nabla_Q^2 B(x^*_Q) F_Q \\
\nabla_L^2 B(x^*_L) F_L
\end{bmatrix} \left( F_Q^T \nabla_Q^2 B(x^*_Q) F_Q + F_L^T \nabla_L^2 B(x^*_L) F_L \right)^{-1}
\begin{bmatrix}
1/2 \\
g
\end{bmatrix} - \tilde{\gamma}(\beta, g)
\begin{bmatrix}
\nabla_Q B(x^*, \gamma^*, t^*) \\
\nabla_L B(x^*, \gamma^*, t^*)
\end{bmatrix}
$$

where

$$
\nabla_Q B(\gamma^*, x^*) = \frac{-2}{\gamma^* - \|Gx^*\|^2} \begin{bmatrix}
\gamma^* + 1/2 \\
-Gx^* T
\end{bmatrix},
$$

$$
\nabla_L B(x^*, \gamma^*, t^*) = -\begin{bmatrix}
1/(b_1 - a_1 x^*) \\
\cdots \\
1/(b_m - a_m x^*) \\
1/(\gamma_{\text{max}} - \gamma)
\end{bmatrix}^T.
$$
and

\[ \nabla^2_{q} B(x_Q) = \frac{2}{\gamma^* - \|Gx^*\|^2} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \]

\[ + \frac{4}{(\gamma^* - \|Gx^*\|^2)^2} \begin{bmatrix} \left(\frac{\gamma^*+1}{2}\right)^2 - \left(\frac{\gamma^*-1}{2}\right)^2 & -\frac{\gamma^*+1}{2} Gx^* \\ -\frac{\gamma^*+1}{2} Gx^* & \left(\frac{\gamma^*+1}{2}\right)^2 - \left(\frac{\gamma^*-1}{2}\right)^2 \end{bmatrix} \]

\[ \nabla^2_{l} B(x_L) = \text{diag}([b - Ax^*] (\gamma_{\text{max}} - \gamma^*)) \]

**Example 4.2.** This example is based on an F-18 control allocation example used in [Dur94]. The control input \( u \) and vehicle torque \( d \) are related by a linear transformation \( d = Bu \). At each time step, a commanded torque \( d_{\text{des}} \) is requested of the actuators. The system is assumed to be over-actuated, but the control input is constrained by actuator saturation limits. The problem to be solved at each time step is the quadratic program

\[ \min_{u \in \mathcal{U}} \|d_{\text{des}} - Bu\| \]

where \( \mathcal{U} \) is a polytope defining the actuator limits.

For the F-18 example, the linear transformation from control to torque is expressed as

\[ B = \begin{bmatrix} .0238 & -.0238 & .123 & -.123 & .0418 & -.0418 & .00358 \\ -.698 & -.698 & .0994 & .0994 & -.0552 & -.0552 & 0 \\ -.0309 & .0309 & 0 & 0 & -.0174 & .0174 & -.0562 \end{bmatrix} \]

\[ u^T = \begin{bmatrix} u_{\text{hlt}} & u_{\text{rht}} & u_{\text{lte}} & u_{\text{rte}} & u_{\text{la}} & u_{\text{ra}} & u_{\text{r}} \end{bmatrix} \]

\[ u_{\text{min}}^T = \begin{bmatrix} -24^\circ & -24^\circ & -8^\circ & -8^\circ & -30^\circ & -30^\circ & -30^\circ \end{bmatrix} \]

\[ u_{\text{max}}^T = \begin{bmatrix} 10.5^\circ & 10.5^\circ & 45^\circ & 45^\circ & 30^\circ & 30^\circ & 30^\circ \end{bmatrix} \]

\[ d = \begin{bmatrix} C_l & C_m & C_n \end{bmatrix} \]

where \( u_{\text{hlt}} \) and \( u_{\text{rht}} \) are the left and right horizontal tails, \( u_{\text{lte}} \) and \( u_{\text{rte}} \) are the left and right trailing edge flaps, \( u_{\text{la}} \) and \( u_{\text{ra}} \) are the left and right ailerons, and \( u_{\text{r}} \) is the rudder. The moments \( C_l, C_m, \) and \( C_n \) are torques about the roll, pitch, and yaw axes respectively.
For this problem, the desired moment is limited to \( \|d_{\text{des}}\|_\infty \leq 5 \). The following constants are derived for this problem:

\[
\gamma_{\text{max}} = (\sigma_{\text{max}}(G) \cdot \max_{\|u\| \leq b} \|u\|)^2 = 88.9049 \quad \gamma^* = 46.2147
\]

\[
u^* = \begin{bmatrix} 0.6864 & 0.6864 & 15.4704 & 15.4704 & 2.1733 & 2.1733 & 0 \end{bmatrix}^T
\]

\[\vartheta = 17.\]

Let \( \beta = 0.1 \) and compute \( \tilde{\gamma}(\beta, d_{\text{des}}) \) for the extreme points \( d_{\text{des}} = [\pm 5 \quad \pm 5 \quad \pm 5]^T \). Then the maximum value of \( \tilde{\gamma}(\beta, d_{\text{des}}) \) is

\[
\max_{\|d_{\text{des}}\|_\infty \leq 5} \tilde{\gamma}(\beta, d_{\text{des}}) = 325.0265,
\]

which yields a maximum initial duality gap of 5525.4.

Nesterov and Todd guarantee a duality gap reduction of \( (1 - 1/(15\sqrt{\vartheta})) \) at each iteration of the short-step method using the NT-direction. For this problem, this is a reduction of at least 0.9838. Suppose that the desired duality gap is \( \epsilon = 10^{-5} \). Using the conservative estimate of Nesterov and Todd, this is guaranteed after 1235 iterations.

### 4.2 Bounds for Problems with Variable Constraints

In this section, computational bounds are derived for convex programs where the constraint data as well as the objective function is time varying. For reference, the parametric problem is defined as

\[
\min_{x \in \mathbb{R}^m} \quad c(\delta)^T x
\]

subject to

\[
X = F_0(\delta) + F(\delta)x
\]

\[\mathcal{K}
\]

where \( \delta \) lies in the compact domain \( \mathcal{D} \). It is assumed that the above problem and its corresponding dual are feasible and well-posed for all \( \delta \in \mathcal{D} \).

Initialization with a strictly feasible point is more difficult for this case than it was in the previous section, since both the primal and dual feasible spaces change from problem instance to instance. When a strictly feasible primal-dual point is unknown, the
usual strategy is to employ an \textit{infeasible-start method}. Many infeasible-start methods bypass the need for a feasible point by modifying the Newton equation to account for the infeasibility. An overview of several such methods is given in [Wri96]. Other methods embed the convex program in a slightly larger problem for which an interior point is relatively easy to identify. The self-dual embedding technique introduced in Chapter 2 is one example of this. Due to its practical success and efficiency, initialization via self-dual embeddings is becoming increasingly prevalent among various software packages (e.g., SeDuMi 1.03 created by Sturm [Stu99b], and SDPT3 v2.1 by Toh, Todd, and Tütüncü [TTT99]). This section analyzes the computational requirements of solving a problem initialized using the strongly feasible self-dual embedding introduced in Section 2.7.3. The parametric self-dual embedding of (4.4) is defined by

\[
\begin{align*}
\min_{\theta} & \quad \zeta\theta \\
\text{subject to} & \quad \bar{X} - \tau F_0(\delta) - F(\delta)\bar{x} + \bar{F}(\delta)\theta = 0 \\
& \quad F(\delta)^T\bar{z} - \tau c(\delta) + \bar{c}(\delta)\theta = 0 \\
& \quad c(\delta)^T\bar{x} + \langle F_0(\delta), \bar{z} \rangle - \alpha(\delta)\theta + \kappa = 0 \\
& \quad \bar{c}(\delta)^T\bar{x} + \langle \bar{F}(\delta), \bar{z} \rangle - \alpha(\delta)\tau = -\zeta \\
& \quad \bar{X} \in \mathcal{K}, \quad \bar{z} \in \mathcal{K}^*, \quad \tau \geq 0, \quad \kappa \geq 0
\end{align*}
\]

where

\[
\begin{align*}
\zeta &= \langle \bar{X}^0, \bar{Z}^0 \rangle + \tau^0\kappa^0 \\
\bar{F}(\delta) &= \tau^0 F_0(\delta) + F(\delta)\bar{x}^0 - \bar{X}^0 \\
\bar{c}(\delta) &= \tau^0 c(\delta) - F(\delta)^T\bar{Z}^0 \\
\alpha(\delta) &= \kappa^0 + c(\delta)^T\bar{x}^0 + \langle F_0(\delta), \bar{Z}^0 \rangle
\end{align*}
\]

and $\bar{X}^0 \in \text{int} \mathcal{K}$, $\bar{Z}^0 \in \text{int} \mathcal{K}^*$, $\bar{x}^0 \in \mathbb{R}^n$, $\tau^0 \in \mathbb{R}_{++}$, $\kappa^0 \in \mathbb{R}_{++}$ are constants which are specified before implementation. These points can be chosen arbitrarily, as long as they lie in the interior of their respective cones.
4.2. Initialization and Termination

The homogeneous self-dual method is easy to initialize. A strictly feasible initialization is given by choosing the initial iterates $\bar{X}(0)$, $\bar{Z}(0)$, $\bar{x}(0)$, $\tau(0)$, $\kappa(0)$ equal to the constants $\bar{X}^0$, $\bar{Z}^0$, etc. in (4.6). Furthermore, by initializing the self-dual program with $\bar{X}^0 \circ \bar{Z}^0 = \mu^0 \gamma$ and $\tau^0 \kappa^0 = \mu^0$ for some scalar $\mu^0 > 0$, then the initial iterates lie exactly on the central path with a known initial duality gap $\zeta(0) = \langle \bar{X}^0, \bar{Z}^0 \rangle + \tau^0 \kappa^0 = (\vartheta + 1) \mu$.

In Section 4.1, knowing the initial duality gap of the problem was sufficient to bound the number of iterations for finding an $\epsilon$-optimal solution. For the self-dual embedding, initialization is only half the battle, since the desired terminating duality gap is still unknown. Since the real variables of interest are $X = \bar{X}/\tau$ and $Z = \bar{Z}/\tau$, the desired solution accuracy is $(\bar{X}, \bar{Z}) \leq \epsilon \tau^2$, which cannot be immediately inferred from the self-dual program duality gap $\zeta(\delta) = \langle \bar{X}, \bar{Z} \rangle + \tau \kappa$. While the usual difficulty for interior-point methods is knowing how to initialize the problem, homogeneous self-dual programming has transferred this difficulty to knowing when the algorithm should be terminated.

If (4.5) is initialized such that $\bar{X}^0/\tau^0$, $\bar{x}^0/\tau^0$ and $\bar{Z}^0/\tau^0$ are feasible for (4.4) and its dual, then $\bar{X}(\delta) = 0$, $\bar{c}(\delta) = 0$ and all iterates generated by the self-dual program yield strictly feasible points for (4.4). However, it is generally not possible to have $\bar{X}(\delta) = 0$, $\bar{c}(\delta) = 0$ for all $\delta \in \mathcal{D}$. In the case where $\bar{F}$ and/or $\bar{c}$ are nonzero, then it is expected that the self-dual program will produce iterates which approach feasibility asymptotically. A suitable termination requirement for the self-dual program is to find iterates $X^{(k)} \triangleq \bar{X}^{(k)}/\tau^{(k)}$, $x^{(k)} \triangleq \bar{x}^{(k)}/\tau^{(k)}$, and $Z^{(k)} \triangleq \bar{Z}^{(k)}/\tau^{(k)}$ which satisfy

\begin{align*}
\|X^{(k)} - F_0(\delta) - F(\delta)x^{(k)}\| &\leq \epsilon_1 \quad (4.7a) \\
\|F^T(\delta)Z^{(k)} - c(\delta)\| &\leq \epsilon_2 \quad (4.7b) \\
\langle c(\delta), x^{(k)} \rangle + \langle F_0(\delta), Z^{(k)} \rangle &\leq \epsilon_3 \quad (4.7c)
\end{align*}

for some prespecified tolerance $\epsilon_1, \epsilon_2, \epsilon_3 > 0$. Inequalities (4.7a) and (4.7b) constrain the infeasibility of the primal and dual problems respectively, and inequality (4.7c) constrains the duality gap, or distance from optimality. Note that not all optimal solutions to (4.5) will yield solutions that satisfy (4.7), since the trivial solution $\bar{X} = 0$, $\bar{x} = 0$, $\bar{Z} = 0$, $\tau = \kappa = \theta = 0$ is optimal for (4.5). Finding the appropriate solution therefore depends
on implementing the right the type of optimization algorithm. Assuming that (4.4) is feasible and well-posed, any path-following algorithm applied to (4.5) is guaranteed to generate iterates which eventually satisfy (4.7) [LSZ98].

4.2.2 Iteration Bounds

Since the duality gap of (4.5) is proportional to $\theta$, each iterate of the short-step or predictor-corrector algorithm will reduce $\theta$ by at least a factor $(1 - \Delta/\sqrt{\theta} + 1)$. The variable $\theta$ is initialized at $\theta^{(0)} = 1$. An iteration bound is easily derivable if the final value of $\theta$ is known. Therefore, it is necessary to find a threshold $\theta^{(k)} \leq \gamma$ which implies (4.7).

It is not obvious how to find a suitable constant threshold for $\theta$, since interpreting the results of the self-dual program depends on inferring the limits of $\tau$ and $\kappa$ as $\theta$ approaches zero. Indeed, if the problem is ill-posed, deciding the feasibility of the problem may not even be possible from the exact solutions to (4.5), since it may hold that $\tau^* = \kappa^* = 0$ for all optimal solutions (see Section 2.7.2).

Manipulation of the equations in (4.5) shows that

\[
\|X - \mathcal{F}(\delta) - F(\delta)x\| = \frac{\theta}{\tau} \|\mathcal{F}(\delta)\| \\
\|F^T(\delta)z - c(\delta)\| = \frac{\tau}{\theta} \|\bar{c}(\delta)\| \\
\langle c(\delta), x \rangle + \langle \mathcal{F}(\delta), z \rangle \leq \alpha(\delta).
\]

For the remainder of the chapter, it is assumed that $\|\mathcal{F}\| > 0$, $\bar{c} > 0$, and $\alpha > 0$, since otherwise, the corresponding constraint (4.7a), (4.7b), or (4.7c) would be satisfied for all iterations, and could be eliminated from the problem. Given this assumption, the termination condition (4.7) is satisfied if and only if

\[
\theta^{(k)} \leq \tau^{(k)} \Gamma(\delta) \quad (4.8)
\]

where

\[
\Gamma(\delta) \triangleq \min \left\{ \frac{\epsilon_1}{\|\mathcal{F}(\delta)\|}, \frac{\epsilon_2}{\|\bar{c}(\delta)\|}, \frac{\epsilon_3}{\alpha(\delta)} \right\} \quad (4.9)
\]
4.2. BOUNDS FOR PROBLEMS WITH VARIABLE CONSTRAINTS

The difficulty with condition (4.8) is that it constrains \( \theta \) to be less than a quantity which is a function of the variable \( \tau \), rather than an \textit{a priori} known constant. Furthermore, (4.8) may never be satisfied if the primal-dual problem is infeasible, in which case \( \tau \) will approach zero at least as fast as \( \theta \).

The key to finding a constant threshold for \( \theta \) rather than (4.8) is to keep the iterates close to the central path. In order for an iterate to be on the central path, the pair of scalars \( \tau^{(k)} \) and \( \kappa^{(k)} \) must satisfy \( \tau^{(k)} \kappa^{(k)} = \mu^{(k)} \), where \( \mu^{(k)} = \zeta^{(k)}/(\theta + 1) \). In [LSZ98], a sequence of feasible iterates \((\bar{x}^{(k)}, \bar{z}^{(k)}, \tau^{(k)}), \rho^{(k)}, \theta^{(k)}), k = 1, 2, \ldots \), is defined to be \textit{weakly centered} if there exists some constant \( \omega \) such that \( \kappa^{(k)} \tau^{(k)} \geq \omega \zeta^{(k)} > 0 \) for all \( k = 1, 2, \ldots \), and \( \lim_{k \to \infty} \theta^{(k)} = 0 \). This condition is true for all path-following algorithms, since iterates are restricted to a neighborhood of the central path. Consider the neighborhood \( \mathcal{N}(\beta) \) introduced as (2.9) in Section 2.6. Then

\[
(\bar{x}, \bar{z}, \tau, \kappa) \in \mathcal{N}(\beta)
\]

implies that

\[
\left| \frac{\tau \kappa}{\zeta \theta/(\theta + 1)} - 1 \right| \leq \beta
\]

from which it is clear that

\[
\tau \kappa \geq \frac{1 - \beta}{\theta + 1} \zeta \theta.
\]

Therefore, the path-following algorithms presented in Section 2.6 are weakly centered with \( \omega = (1 - \beta)/(\theta + 1) \).

Condition (4.8) can be replaced by a constant threshold for \( \theta^{(k)} \) if \( \tau^{(k)} \) can be bounded below for all \( k \). The following proposition is useful for bounding \( \tau \) given a weakly centered sequence.

**Proposition 4.4.** Let \((\bar{x}, \bar{z}, \tau, \kappa, \theta)\) be a feasible point such that \( \kappa \tau \geq \omega \zeta \theta > 0 \), and let \((\bar{x}^*, \bar{z}^*, \tau^*, \kappa^*, \theta^*)\) be an optimizer. Then \( \tau \geq \omega \tau^* \).

**Proof.** Choose \((\bar{x}, \bar{z}, \tau, \kappa, \theta)\) as a primal point and \((\bar{x}^*, \bar{z}^*, \tau^*, \kappa^*, \theta^*)\) as a dual point. It is assumed that \( \tau^* > 0 \) and \( \kappa^* = 0 \), since the case in which \( \tau^* = 0 \) would already satisfy
\( \tau \geq \omega \tau^* \). The duality gap between these two points is

\[
\langle \tilde{X}, \tilde{Z}^* \rangle + \langle \tilde{X}^*, \tilde{Z} \rangle + \kappa \tau^* + 0 \tau = \zeta \theta
\]

therefore \( \kappa \leq \zeta \theta / \tau^* \), from which follows

\[
\tau \geq \frac{\omega \zeta \theta}{\kappa} \geq \omega \tau^*.
\]

This proposition shows that the value of \( \tau \) at any iteration can be bounded by the optimal value \( \tau^* \). Thus, a weakly-centered sequence which restricts iterates to the neighborhood \( \mathcal{N}(\beta) \) satisfies

\[
\tau^{(k)} \geq \frac{1 - \beta}{\vartheta + 1} \tau^*
\]

(4.10)

for all iterates \( k \). The optimal value \( \tau^* \) can be computed directly from the equations in (4.5). At optimality, \( \theta^* = 0 \). Rearrangement of the equations in (4.5) yields

\[
\tau^0 \kappa + \langle \tilde{X}_0, \tilde{Z}^* \rangle + \langle \tilde{X}^*, \tilde{Z}_0 \rangle + \kappa^0 \tau^* = (\vartheta + 1) \mu^0
\]

(4.11)

where

\[
\mu^0 \triangleq (\langle \tilde{X}_0, \tilde{Z}_0 \rangle + \tau^0 \kappa^0) / (\vartheta + 1).
\]

Assuming that problem (4.4) and its dual are feasible and well-posed with optimal solutions \( \tilde{X}^*, x^*, Z^* \) (otherwise \( \tau^* = 0 \)), then the corresponding \( \tau^* \) is derived as

\[
\tau^* = \frac{(\vartheta + 1) \mu^0}{\langle \tilde{X}_0, \tilde{Z}^* \rangle + \langle \tilde{X}^*, \tilde{Z}_0 \rangle}.
\]

(4.12)

For a given feasible problem instance \( p(\delta) \), define the quantity

\[
\Psi(\delta) \triangleq \Psi_P(\delta) + \Psi_D(\delta),
\]

where

\[
\Psi_P(\delta) \triangleq \sup\{ \langle \tilde{X}^*, \tilde{Z}_0 \rangle \mid \tilde{X}^* \text{ is optimal for the primal of } p(\delta) \},
\]

\[
\Psi_D(\delta) \triangleq \sup\{ \langle \tilde{X}_0, \tilde{Z}^* \rangle \mid \tilde{Z}^* \text{ is optimal for the dual of } p(\delta) \}.
\]
4.2. BOUNDS FOR PROBLEMS WITH VARIABLE CONSTRAINTS

With this notation, $\tau^*$ is bounded by

$$\tau^* \geq \frac{(\theta + 1)\mu^0}{\Psi(\delta)}$$

By combining this inequality with (4.10), it is seen that every iterate in the neighborhood $\mathcal{N}(\beta)$ satisfies

$$\tau^{(k)} \geq \frac{(1 - \beta)\mu^0}{\Psi(\delta)}.$$  \hspace{1cm} (4.13)

Finally, this yields the desired threshold for $\theta$, as stated by the next proposition.

**Proposition 4.5.** At iterate $k$, let $(\tilde{x}^{(k)}, \tilde{z}^{(k)}, \tau^{(k)}, \kappa^{(k)}) \in \mathcal{N}(\beta)$ and suppose $\theta^{(k)}$ satisfies

$$\theta^{(k)} \leq (1 - \beta)\mu^0 \Gamma(\delta) \Psi(\delta).$$

Then $X^{(k)}, x^{(k)}, Z^{(k)}$ satisfy the optimality conditions (4.7).

**Proof.** Follows directly from (4.8) and (4.13). $\square$

The short-step and predictor-corrector interior-point methods from Section 2.6 guarantee a reduction of $\theta$ by at least a factor $(1 - \Delta/\sqrt{\theta + 1})$ at each iteration. Since $\theta^{(0)} = 1$, the optimality conditions (4.7) are satisfied for all iterations $k$ with

$$k \geq \left\lceil \frac{\log \left( \frac{(1 - \beta)\mu^0 \Gamma(\delta)/\Psi(\delta)}{\log(1 - \Delta/\sqrt{\theta + 1})} \right)}{\log(1 - \Delta/\sqrt{\theta + 1})} \right\rceil$$  \hspace{1cm} (4.14)

which grows as

$$O \left( \sqrt{\theta + 1} \log \left( \frac{\Psi(\delta)}{(1 - \beta)\mu^0 \Gamma(\delta)} \right) \right).$$

Note that every quantity in (4.14) is independent of the problem data except the ratio $\Psi(\delta)/\Gamma(\delta)$. Computing this ratio for a single problem instance $p(\delta)$ would ordinarily not be too difficult. However, the iteration bound for the on-line optimization problem must be valid over all $\delta \in \mathcal{D}$, which means an upper bound for $\Psi(\delta)/\Gamma(\delta)$ must be derived. Unfortunately, $\Psi(\delta)$ is not a concave function, nor is $\Gamma(\delta)$ convex, so it is unlikely that the maximum value of $\Psi(\delta)$ or the minimum of $\Gamma(\delta)$ can be computed in polynomial time. Conservative bounds on these numbers will need to suffice. Fortunately, the dependence
of the bound (4.14) on this ratio is logarithmic, making conservative estimates easier to tolerate.

Define $\Psi$ and $\Gamma$ over the set of problem instances as

$$
\Psi(D) = \max_{\delta \in D} \Psi(\delta) \\
\Gamma(D) = \min_{\delta \in D} \Gamma(\delta).
$$

Two different strategies for bounding the ratio $\Psi(D)/\Gamma(D)$ are given next.

**Iteration Bounds via the Condition Number**

While not very practical for providing tight iteration bounds, complexity bounds based on the condition number are appealing from a theoretical perspective. Complexity of convex programming has traditionally been related to the bit-size of the problem, which is customarily referred to as *bit complexity* (see [Kha79] for the first proof that linear programming can be solved in polynomial time in the bit complexity framework). However, Renegar was the first to point out that complexity can instead be judged relative to the problem condition measure, which does not restrict consideration to problems with rational data [Ren95a]. Condition number based complexity results seem to offer a much more realistic picture of a problem’s true complexity, since they tend to be much less conservative than bit complexity. Condition numbers have also been used to show the efficiency of the ellipsoid algorithm. This is discussed by Freund and Vera in [FV97, FV99b], who prove the existence of inscribing and intersecting ellipsoids of a feasible region, with radii depending on the condition number rather than bit size of a problem.

This section relates the complexity of solving the self-dual embedding to the conditioning of the original primal-dual problem. It is assumed that $\bar{x}^0 = \bar{z}^0 = \mathbf{3}$, $\tau^0 = \kappa^0 = 1$, and $\epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon$. These assumptions are simply for convenience of the results which follow.

Based on the definition of $\Gamma(\delta)$ (4.9), the following bound holds:

$$
\frac{1}{\Gamma(\delta)} \leq \frac{1}{\epsilon} \max \left\{ (\|F_0(\delta)\| + \sqrt{\vartheta}), (\|c(\delta)\| + \sqrt{\vartheta}\|F(\delta)\|), (1 + \sqrt{\vartheta}\|F_0(\delta)\|) \right\}
$$

$$
\leq \frac{\sqrt{\vartheta}(2\|p(\delta)\| + 1)}{\epsilon}.
$$
Also, the Cauchy-Schwartz inequality yields the bound

\[ \Psi(\delta) \leq \sup \{ \sqrt{\vartheta(\|X^*\| + \|Z^*\|)} \mid X^* \text{ and } Z^* \text{ are optimal for } p(\delta) \}. \]

It was seen in Section 3.3 that the optimal solution set can be bounded by a problem’s condition number by means of

\[ \|X^*\| \leq 2\|p\|C^2(p) \]
\[ \|Z^*\| \leq C^2(p), \]

therefore

\[ \Psi(\delta) \leq \sqrt{\vartheta(2\|p(\delta)\| + 1)C^2(p(\delta))}. \]

Finally, the ratio \( \Psi(\delta)/\Gamma(\delta) \) is bounded by

\[ \Psi(\delta)/\Gamma(\delta) \leq \frac{\vartheta}{\epsilon}[(2\|p(\delta)\| + 1)C(p(\delta))]^2. \]

Substitution of this ratio into (4.14) shows that the complexity grows as

\[ O \left( \sqrt{\vartheta + 1} \log \left( \frac{\vartheta[(2\|p(\delta)\| + 1)C(p(\delta))]^2}{(1 - \beta)\epsilon} \right) \right). \]

In principle, the ratio \( \Psi(\mathcal{D})/\Gamma(\mathcal{D}) \) could be bounded above using the condition number bounding techniques presented in Section 3.3.3. This ultimately leads to a valid iteration bound using (4.14). However, for most problems, iteration bounds based on the condition number will be very conservative. Relating the iteration bound to the condition number may be more interesting from a theoretical point of view. In some sense, this relation confirms something which is already known about homogeneous self-dual programming. Certainly, an ill-posed problem may require an infinite number of computations to solve in the homogeneous self-dual framework, since an infinite precision solution is required to decide the limits of \( \tau \) and \( \kappa \). Likewise, the condition number indicates that a problem which is very close to ill-posedness may require a very large number of computations to solve the self-dual program, relative to a problem which is well-conditioned.

Since condition number estimates yields conservative bounds for \( \Psi(\mathcal{D})/\Gamma(\mathcal{D}) \), it is appropriate to ask whether a tighter estimate of this ratio is possible. The next section considers this question.
A Branch and Bound Approach to Iteration Bounds

The value of $\Psi(D)/\Gamma(D)$ is the solution to a nonconvex global optimization problem. Exact computation of this ratio is in general NP-hard. This can be seen by noting that the problem of finding $\Gamma(D)$ is a general convex quadratic maximization problem over a polytope, which falls into the class of NP-complete problems (see [GJ79]). Fortunately, all that is required is an upper bound of the optimal value, not the optimal value itself. Also, the iteration bound dependence upon this ratio is logarithmic, lessening the impact of conservatism.

Since $D$ is a compact domain, it is not unreasonable to consider branch and bound approaches to this problem. Branch and bound is a widely used global optimization technique for non-convex problems, appearing frequently in the optimization literature. The analysis used in this section is similar to that in [HT96].

Finding $\Gamma(D)$ consists of solving a quadratic maximization problem, since $||\mathcal{F}(\delta)||$, $||\mathcal{C}(\delta)||$, and $\alpha(\delta)$ must be maximized over $\delta$. Although this problem is NP-hard, there exist relaxations which can provide valid upper bounds. For example, these norms can be bounded using Cauchy-Schwartz inequalities, or more sophisticated semidefinite relaxation techniques (see [Fer99, Zha98]). It is not too difficult to construct a branch and bound technique which finds the exact value of $\Gamma(D)$ in the limit. Since many techniques for finding these bounds are well known, they will not be discussed further.

Computation of $\Psi(D)$ is not as standard. To simplify the problem, the primal and dual parts of $\Psi$ are considered as separate optimization problems, with the relation

$$\Psi(D) \leq \max_{\delta \in D} \Psi_P(\delta) + \max_{\delta \in D} \Psi_D(\delta).$$

Since the primal and dual parametric problems can be represented in equivalent forms, this section considers only the problem of bounding $\max \Psi_P(\delta)$.

Analysis of branch and bound algorithms typically depends on the continuity of the objective function. Unfortunately, $\Psi_P(\delta)$ is not continuous in general. For example, consider $\Psi(\delta) = \max\{x_1 \mid (x_1, x_2) \in \arg\min \{\delta x_1 + x_2 \mid 0 \leq x_1 \leq 1, \ x_2 \geq 0\}\}$. Then $\Psi(\delta) = \{1 \text{ if } \delta \leq 0; 0 \text{ otherwise}\}$. However, observe that $\Psi(\delta)$ is upper semicontinuous, meaning that for each $\epsilon > 0$, there exists a $\gamma > 0$ such that $||\delta_2 - \delta_1|| < \gamma$ implies that
4.2. BOUNDS FOR PROBLEMS WITH VARIABLE CONSTRAINTS

\[ \Psi_P(\delta_2) - \Psi_P(\delta_1) < \epsilon. \] Upper semicontinuity allows the possibility of a branch and bound algorithm with a convergent upper bound. It will be seen in Corollary 4.8 that upper semicontinuity is a general property of \( \Psi_P(\delta) \).

The branch and bound algorithm presented in this section employs the same partitioning technique used for the infeasibility detection algorithm in Section 3.4. It will be recalled that this partition was constructed using hyper-rectangles \( \mathcal{M} = \{ \delta \in \mathbb{R}^s \mid a \leq \delta \leq b \} \), \( a, b \in \mathbb{R}^s \), \( a < b \).

Figure 4.1 shows the branch and bound algorithm \( \Psi\text{-BND} \). The key steps of the algorithm are briefly outlined below.

1. Start with a cover \( \bigcup_{i \in \mathcal{I}} \mathcal{M}_i \supseteq \mathcal{D} \) derived from a finite partition \( \{ \mathcal{M}_i \} \) indexed by \( \mathcal{I} \).

2. For each index \( i \), determine lower and upper bounds \( \ell(\mathcal{M}_i) \) and \( u(\mathcal{M}_i) \) respectively, which satisfy

\[ \ell(\mathcal{M}_i) \leq \max_{\delta \in \mathcal{M}_i} \Psi_P(\delta) \leq u(\mathcal{M}_i). \]

3. Set \( \ell_{\text{max}} = \max_{i \in \mathcal{I}} \ell(\mathcal{M}_i) \) and \( u_{\text{max}} = \max_{i \in \mathcal{I}} u(\mathcal{M}_i) \). Then

\[ \ell_{\text{max}} \leq \max_{\delta \in \mathcal{D}} \Psi_P(\delta) \leq u_{\text{max}}. \]

4. If \( u_{\text{max}} = \ell_{\text{max}} \) then stop, otherwise repeat the process for a refined partition.

The bounding operations \( \ell \) and \( u \) need to be specified for this algorithm. To define \( \ell(\mathcal{M}) \), choose any \( \delta \in \mathcal{D} \cap \mathcal{M} \), and set \( \bar{\lambda}(\delta) = \min_x \{ c(\delta)^T x \mid \mathcal{F}_0(\delta) + \mathbf{F}(\delta)x \in \mathcal{K} \} \). Then let

\[ \ell(\mathcal{M}) = \max_x \langle \mathcal{X}, \hat{\mathbf{Z}}^0 \rangle \]

subject to

\[ \mathcal{X} = \mathcal{F}_0(\delta) + \mathbf{F}(\delta)x \]

\[ c(\delta)^T x = \bar{\lambda}(\delta) \]

\[ \mathcal{X} \in \mathcal{K}. \]

Clearly,

\[ \ell(\mathcal{M}) = \Psi_P(\delta) \leq \max_{\delta \in \mathcal{M}} \Psi_P(\delta). \]
Algorithm $\Psi$-BND

**initialize** $S := \{M_0\}$ such that $M_0 \supseteq \mathcal{D}$

$u_{-1} := \infty$; $\ell_{-1} = -\infty$; $k = 0$

**while** $\ell_{k-1} \neq u_{k-1}$

$\ell_k := \max_{M \in S} \ell(M); u_k := \max_{M \in S} u(M)$

$\tilde{M} = \text{argmax}_{M \in S} u(M)$

$S := S \setminus \{\tilde{M}\}$

$\{M_\alpha, M_\beta\} := \text{bisect} (\tilde{M})$

**if** $\mathcal{D} \cap M_\alpha \neq \emptyset$ **and** $u(M_\alpha) \geq \ell_k$ **then** $S := S \cup \{M_\alpha\}$

**if** $\mathcal{D} \cap M_\beta \neq \emptyset$ **and** $u(M_\beta) \geq \ell_k$ **then** $S := S \cup \{M_\beta\}$

$k := k + 1$

**end**

Figure 4.1: Branch and Bound Algorithm for $\Psi(\mathcal{D})$, which is necessary for finding an iteration bound.
Finding an upper bound \( u(M) \) requires a little more finesse. Choose any \( \lambda \geq \bar{\lambda}(\delta) \). Then the value of \( \Psi_P(\delta) \) can be bounded above by the solution to the problem

\[
\max_{\tilde{X}} \langle \tilde{X}, \tilde{z}(0) \rangle
\]
subject to \( \tilde{X} = \mathcal{I}_0(\delta) + F(\delta)x \)
\( z = \lambda - c(\delta)^T x \)
\( \tilde{X} \in \mathcal{K}, \ z \geq 0 \). \hspace{1cm} (4.15)

This problem can be transformed into an equivalent form

\[
\max_{\tilde{X}} \langle S_0(\delta), \tilde{X} \rangle + d(\delta)
\]
subject to \( G(\delta)^T \tilde{X} = g(\delta, \lambda) \)
\( \tilde{X} \in \mathcal{K} \times \mathbb{R}_+ \)

where \( \tilde{X} = (X, z) \), and \( S_0(\delta), G(\delta), g(\delta), d(\delta) \) are rational functions of the parameter \( \delta \).

The specifics of the transformation from (4.15) to (4.16) are not related here, since this transformation is very similar to the LFT transformations of problem data presented in Section 3.1. Problem (4.16) is dual to

\[
\min_y \ g(\delta, \lambda)^T y + d(\delta)
\]
subject to \( S_0(\delta) + G(\delta)y \in \mathcal{K}^* \times \mathbb{R}_+ \).

For a given \( \delta \), problem (4.16) is feasible if and only if \( \lambda \geq \bar{\lambda}(\delta) \). The optimal value of (4.17) bounds \( \Psi_P(\delta) \) from above for all \( \lambda > \bar{\lambda}(\delta) \). Strong duality between (4.16) and (4.17) is necessary to show that the optimal value of (4.17) equals \( \Psi_P(\delta) \) when \( \lambda = \bar{\lambda}(\delta) \). Because (4.16) is ill-posed when \( \lambda = \bar{\lambda}(\delta) \), this fact is not immediately obvious. Nevertheless, strong duality does in fact hold, as demonstrated by the next lemma.

**Lemma 4.6.** Assume \( p(\delta) \triangleq (c(\delta), \mathcal{I}_0(\delta), F(\delta)) \) is well posed. Then strong duality holds between (4.16) and (4.17) for all \( \lambda \geq \bar{\lambda}(\delta) \).

**Proof.** Since \( p(\delta) \) is well posed, any unbounded sequence of feasible points \( x \) such that \( \mathcal{I}_0(\delta) + Fx \in \mathcal{K} \) will yield an unbounded sequence in \( c(\delta)^T x \). Therefore, the constraints in (4.16), which are identical to those in (4.15), define a nonempty, compact space for all
\( \lambda \geq \bar{\lambda}(\delta) \). Compactness of the primal constraints ensures strong feasibility of the dual problem (see [LSZ97]). Finally, strong feasibility of either the primal or dual problem guarantees strong duality.

The key to bounding \( \max_{\delta \in M} \Psi_P(\delta) \) from above is to find robust solutions to (4.15) and (4.17). Assume that the radius of \( M \) is chosen small enough and \( \lambda \) large enough such that both (4.15) and (4.17) are robustly feasible. Then the robust objective value of (4.17) bounds \( \psi_P(\delta) \) from above for all \( \delta \in M \). Formally, a valid bound \( u(M) \) can be derived from the robust convex programs

\[
\begin{align*}
\bar{\lambda}(M) & \triangleq \min_{x, \lambda} \{ \lambda | \mathcal{F}_0(\delta) + F(\delta)x \in K, \ c(\delta)^T x \leq \lambda, \ \forall \delta \in M \} \quad (4.18) \\
u(M) & \triangleq \min_{y, \gamma} \{ \gamma | \mathcal{G}_0(\delta) + G(\delta) \in K^*, \ g(\delta, \bar{\lambda}(M))^T y + d(\delta) \leq \gamma, \ \forall \delta \in M \}. \quad (4.19)
\end{align*}
\]

The cases \( \bar{\lambda}(M) = \infty \) and \( u(M) = \infty \) are allowed in the event that robust solutions to (4.18) and (4.19) cannot be found.

In this algorithm, the operation \textbf{bisect} is used to partition a rectangle \( M \) into two pieces of equivalent size. It can be specified to divide \( M \) at the midpoint of one of its longest edges, although other variations are possible. The important quality of the subdivision is that it is \textit{exhaustive} (see Section 3.4).

\textbf{Lemma 4.7.} Let the parameter \( \bar{\delta} \in \mathbb{R}^s \) correspond to a feasible well-posed problem \( p(\bar{\delta}) \). Suppose \( \{ M_k \} \) is an infinitely decreasing sequence of rectangles generated by an exhaustive subdivision, with \( \bigcap_{k=1}^{\infty} M_k = \delta \). Then

\[
\lim_{k \to \infty} u(M_k) = \Psi_P(\delta).
\]

\textbf{Proof.} Define

\[
\begin{align*}
\bar{\lambda}(r) & \triangleq \min_{x, \lambda} \{ \lambda | \mathcal{F}_0(\delta) + F(\delta)x \in K, \ c(\delta)^T x \leq \lambda, \ \forall ||\delta - \bar{\delta}|| \leq r \} \quad (4.20) \\
u(r) & \triangleq \min_{y, \gamma} \{ \gamma | \mathcal{G}_0(\delta) + G(\delta) \in K^*, \ g(\delta, \bar{\lambda}(r))^T y \leq +d(\delta)\gamma, \ \forall ||\delta - \bar{\delta}|| \leq r \}. \quad (4.21)
\end{align*}
\]

Since a sequence \( r_k \to 0 \) can be constructed such that \( \delta \in M_k \) implies \( ||\delta - \bar{\delta}|| \leq r_k \), this lemma can be proved by showing that \( \lim_{k \to \infty} u(r_k) = \psi_P(\delta) \). Also, because \( \psi_P(\delta) = \hat{u}(0) \) (from strong duality), it is sufficient to show that \( \hat{u} : \mathbb{R}^+ \to \mathbb{R} \) is continuous at 0.
4.2. BOUNDS FOR PROBLEMS WITH VARIABLE CONSTRAINTS

Define the interiors of the problems $\tilde{\lambda}(r)$ and $\tilde{u}(r)$ as $L(r)$ and $U(r)$. Since $p(\delta)$ is feasible and well posed, it follows that $L(0) \neq \emptyset$ and $U(0) \neq \emptyset$.

By continuity of $F_0(\delta)$, $F(\delta)$ and $c(\delta)$, each point in the interior $L(0)$ also belongs to $L(\epsilon)$ for some $\epsilon > 0$. By Theorem 2.1 in [FI90], this is sufficient to ensure upper semicontinuity of $\tilde{\lambda}(r)$ at 0 (in fact, $\tilde{\lambda}(r)$ is continuous). Continuity at 0 follows since $\tilde{\lambda}(r)$ is minimized at $r = 0$.

Now it is clear that $\mathcal{G}_0(\delta)$, $\mathcal{G}(\delta)$ and $g(\delta, \tilde{\lambda}(r))$ are continuous on $\delta$ and $r$. Like before, for each point in the interior $U(0)$, there exists an $\epsilon > 0$ such that this point is also in $U(\epsilon)$. Continuity at 0 again follows from Theorem 2.1 in [FI90]. Thus, $\lim_{k \to \infty} \tilde{u}(r_k) = \tilde{u}(0) = \Psi_P(\delta)$.

Corollary 4.8. The function $\Psi_P(\delta)$ is upper semicontinuous.

Proof. Assume that $\Psi_P(\delta)$ is not upper semicontinuous. Then there exists an $\epsilon > 0$, a point $\tilde{\delta}$, and sequence of neighborhoods $B(\tilde{\delta}, r_k) = \{\delta \mid \|\delta - \tilde{\delta}\| \leq r_k\}$ with $r_k \to 0$ such that

$$\sup_{\delta \in B(\tilde{\delta}, r_k)} \Psi_P(\delta) - \Psi_P(\tilde{\delta}) \geq \epsilon$$

for all $k$. Therefore, $\tilde{u}(r_k) - \Psi_P(\tilde{\delta}) \geq \epsilon$ for all $k$, which contradicts Lemma 4.7.

As it will be seen, Lemma 4.7 is necessary to establish the convergence of $u_k$ to the optimal solution. Unfortunately, $\Psi_P$ is only upper semicontinuous, not continuous. It is therefore possible that a convergent sequence of sets $\{M_k\} \to \tilde{\delta}$ exists for which $\lim_{k \to \infty} \ell(M_k) < \Psi_P(\tilde{\delta})$, even though $\ell(\tilde{\delta}) = \Psi_P(\tilde{\delta})$. This means that in some cases it is possible that $\lim_{k \to \infty} (u_k - \ell_k) > 0$.

Proposition 4.9. In the algorithm $\Psi$-BND, $u_k \to \Psi_P(D)$ as $k \to \infty$.

Proof. Assume the algorithm never terminates (otherwise $u_k = \Psi_P(D)$ for some $k$). Let $\{\tilde{M}_k\}$ be the sequence of sets selected by the line $\tilde{M} = \arg\max_{M \in S} u(M)$ at the $k$th iterate of the algorithm, and $\{u_k\}$ the sequence of upper bounds (i.e., $u_k = u(\tilde{M})$). Since $\{u_k\}$ is a monotonically decreasing sequence bounded below by $\Psi_P(D)$, the limit $\lim_{k \to \infty} u_k \geq \Psi_P(D)$ exists.
Let \( \{ \mathcal{M}_k \} \) be an infinite subsequence such that
\[
\mathcal{M}_1 \supseteq \mathcal{M}_{k_2} \supseteq \mathcal{M}_{k_3} \supseteq \cdots.
\]

Since \texttt{bisect} is an exhaustive subdivision, it follows that there exists a \( \delta \in \mathcal{D} \) such that \( \tilde{\delta} = \bigcap_{q=1}^{\infty} \mathcal{M}_{k_q} \). By Lemma 4.7, \( u_{k_q} \rightarrow \Psi_P(\tilde{\delta}) \leq \max_{\delta \in \mathcal{D}} \Psi_P(\delta) \). Therefore, \( \lim_{k \to \infty} u_k = \Psi_P(\mathcal{D}) \).

It should be pointed out that for most problems, \( \Psi-\text{BND} \) will probably never terminate. In fact, relaxing the termination criterion to \( u_k - \ell_k < \epsilon \) is not enough to guarantee termination either, since \( \ell_k \) may not converge to \( \Psi_P(\mathcal{D}) \). However, it is to be remembered that the ultimate goal is to provide an upper bound for \( \Psi(P_0) \), not necessarily a tight bound. A realistic implementation would terminate once the rate of change in \( u_k \) falls below a prespecified tolerance.

\textbf{Example 4.3.} Consider the parametric semidefinite program

\[
\begin{align*}
\min_{x \in \mathbb{R}^2} & \quad [\delta_4 \quad \delta_5] x \\
\text{subject to} & \quad \begin{bmatrix} 1 & \frac{1}{2}\delta_1 & (x_1 - 2\delta_2) \\
\frac{1}{2}\delta_1 & 1 & (1 - \frac{1}{2}\delta_3)x_2 \\
(x_1 - 2\delta_2) & (1 - \frac{1}{2}\delta_3)x_2 & 1 \end{bmatrix} \succeq 0.
\end{align*}
\tag{4.22}
\]

The parameters are restricted to \( \mathcal{D} = \{ \delta \in \mathbb{R}^5 \mid \|\delta\|_{\infty} \leq 1 \} \). Problem (4.22) is feasible and well-posed for all \( \delta \in \mathcal{D} \). This example shows how \( \Psi(\mathcal{D}) \) can be bounded above using the branch and bound algorithm just given.

Suppose \( \bar{X}^0 = \bar{Z}^0 = I \), therefore

\[
\Psi(\mathcal{D}) = \sup_{\delta \in \mathcal{D}} \{ \text{Tr}X^* + \text{Tr}Z^* \mid X^* \text{ and } Z^* \text{ are optimal for (4.22) and its dual} \}.
\]

Clearly, \( \text{Tr}X^* = 3 \) for all optimal \( X^* \), since there are only ones on the diagonal of the
semidefinite constraint in (4.22). To bound $\text{Tr}Z^*$, consider the dual to (4.22):

$$\min_Z \text{Tr} \begin{bmatrix} 1 & \frac{1}{2} \delta_1 & -2 \delta_2 \\ \frac{1}{2} \delta_1 & 1 & 0 \\ -2 \delta_2 & 0 & 1 \end{bmatrix} Z$$

subject to $2z_{13} = \delta_4$

$2(1 - \frac{1}{2} \delta_3)z_{23} = \delta_5$

$Z = Z^T \succeq 0,$

where $z_{ij}$ indicates the $(i, j)$ element of $Z$. A robust solution to this problem is

$$Z = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \delta_4 \\ 0 & 1 & \frac{\delta_5}{2-\delta_3} \\ \frac{1}{2} \delta_4 & \frac{\delta_5}{2-\delta_3} & \frac{3}{2} \end{bmatrix}$$

for which the objective function

$$\text{Tr} \begin{bmatrix} 1 & \frac{1}{2} \delta_1 & -2 \delta_2 \\ \frac{1}{2} \delta_1 & 1 & 0 \\ -2 \delta_2 & 0 & 1 \end{bmatrix} Z = 3 - 2 \delta_2 \delta_4 \leq \tilde{\lambda} \text{ for all } |\delta_i| \leq 1.$$

From this constraint, a valid upper bound $\tilde{\lambda}(\mathcal{D}) = 5$ can be derived. Analogous to (4.15),

the problem is relaxed to

$$\max_Z \text{Tr}Z$$

subject to $2z_{13} = \delta_4$

$2(1 - \frac{1}{2} \delta_3)z_{23} = \delta_5$

$Z = Z^T \succeq 0$

$$\text{Tr} \begin{bmatrix} 1 & \frac{1}{2} \delta_1 & -2 \delta_2 \\ \frac{1}{2} \delta_1 & 1 & 0 \\ -2 \delta_2 & 0 & 1 \end{bmatrix} Z \leq \tilde{\lambda}(\mathcal{D}),$$

$$Z \succeq 0.$$
with corresponding parametric dual

\[
\min_{x \in \mathbb{R}^3} \delta_4 x_1 + \delta_5 x_2 + \bar{\lambda}(\mathcal{D}) x_3
\]
subject to
\[
\begin{bmatrix}
    x_3 & \frac{1}{2} \delta_1 x_3 & x_1 - 2 \delta_2 x_3 \\
    \frac{1}{2} \delta_1 x_3 & x_3 & (1 - \frac{1}{2} \delta_3) x_2 \\
    x_1 - 2 \delta_2 x_3 & (1 - \frac{1}{2} \delta_3) x_2 & x_3
\end{bmatrix} - I \succeq 0.
\]

(4.24)

There does not exist a robust solution to (4.24) over the entire parameter space \( \mathcal{D} \). However, robust solutions can be found if the range of \( \delta_2 \) is broken into three segments \([-1, -1/3], [-1/3, 1/3], \) and \([1/3, 1]\). Robust solutions derived over these three parameter sets yield the bound \( \text{Tr} Z^* \leq 38 \). Also, it is known that \( \text{Tr} Z^* \geq 0 \) from the positive semidefiniteness of \( Z^* \), so the initial gap between upper and lower bounds is 38.

The algorithm \( \Psi\text{-BND} \) is used to reduce this gap. The initial partition is \( \mathcal{M}_0 = \{ \delta \mid |\delta_i| \leq 1 \} \), which is subdivided by the subroutine \( \text{biset} \). This subroutine bisects a given partition element into two equivalent pieces \( \mathcal{M}_\alpha \) and \( \mathcal{M}_\beta \) along the dimension which minimizes the average of \( u(\mathcal{D}_{\mathcal{M}_\alpha}) \) and \( u(\mathcal{D}_{\mathcal{M}_\beta}) \).

Figure 4.2 illustrates the gap \( u_k - \ell_k \) versus iteration on a log-log plot. Inspection of this plot suggests a convergence rate of approximately \( O(\epsilon^{-4}) \).
After the last iterate on this plot, $\ell_k = 2.0$ and $u_k = 4.1$. Since $\Psi_P(D) = 3$ and $\ell_k \leq \Psi_D(D) \leq u_k$, this results in the bound

$$5.0 \leq \Psi(D) \leq 6.1.$$ 

**Example 4.4.** This example shows the specifics of finding the bounds $\lambda(\cdot)$ and $u(\cdot)$ for a convex quadratic. It is supposed that the quadratic program will be solved in the second-order cone program setting. The results of this example will be useful for the UAV example in Chapter 6.

A quadratic program

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T H x + g^T x$$

subject to $Ax \leq b$

with $H \in \mathbb{S}^{n}_{++}, g \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ can be written as a second-order cone program

$$\min_{\gamma,x} \gamma$$

subject to

$$\begin{bmatrix} \gamma \\ Gx + G^{-1}g \end{bmatrix} \succeq 0$$

$$Ax \leq b$$

where $G = H^{1/2}$. The corresponding dual is

$$\max_y - (b + AG^{-2}g)^T y$$

subject to

$$\begin{bmatrix} 1 \\ G^{-1}A^T y \end{bmatrix} \succeq 0$$

$$y \succeq 0.$$ 

The primal and dual second-order cone programs are easily embedded into the homogeneous self-dual framework 4.5. The self-dual program is initialized at the primal interior points $\bar{x}_Q^{(0)} = (u_1, u_2) \in \mathbb{R}^{n+1}_Q, \bar{x}_L^{(0)} \in \mathbb{R}^m_L$ and dual interior points $\bar{z}_Q^{(0)} = (v_1, v_2) \in \mathbb{R}^{n+1}_Q, \bar{z}_L \in \mathbb{R}^m_L$ which satisfy the centrality conditions $\bar{x}_Q^{(0)} \circ \bar{z}_Q^{(0)} = \mu(\sqrt{2}, 0)$ and $\bar{x}_L^{(0)} \circ \bar{z}_L^{(0)} = \mu e$, where $\mu = ((\bar{x}_Q^{(0)}, \bar{z}_Q^{(0)}) + (\bar{x}_L^{(0)}, \bar{z}_L^{(0)}))/(m + 2)$. For this example,
these variables are initialized at identity: $\tilde{x}_Q(0) = \tilde{z}_Q(0) = (\sqrt{2}, 0)$ and $\tilde{x}_L(0) = \tilde{z}_L(0) = e$.

From these initializations, the functions $\Psi_P$ and $\Psi_D$ are written as

$$
\Psi_P(p) = \max\{\sqrt{2}\gamma^* + e^T(b - Ax^*) \mid \gamma^*, \ x^* \text{ are optimal for (4.25)}\}
$$

$$
\Psi_D(p) = \max\{\sqrt{2} + e^Ty^* \mid y^* \text{ is optimal for (4.26)}\}
$$

where $p$ represents the data in (4.25) and (4.26), assumed to be feasible and well-posed.

Robust solutions to (4.25) and (4.26) lead to objective function bounds $\bar{\lambda}_P$ and $\bar{\lambda}_D$ for the primal and dual problems respectively.

The function $\Psi_P(p)$ is bounded above by

$$
\max_{\gamma, x} \sqrt{2}\gamma + e^T(b - Ax) \quad \text{subject to} \quad \begin{bmatrix} \gamma \\ Gx + G^{-1}g \end{bmatrix} \geq_0 0
$$

$$
Ax \leq b
$$

$$
\gamma \leq \bar{\lambda}_P
$$

with corresponding dual

$$
\min_{\eta, w} \bar{\lambda}_P\eta + (AG^{-2}g + b)^Tw \quad \text{subject to} \quad \begin{bmatrix} \eta - \sqrt{2} \\ G^{-1}ATw \end{bmatrix} \geq_0 0
$$

$$
w \geq e. \quad (4.27)
$$

The function $\Psi_D(p)$ is bounded above by

$$
\max_y \sqrt{2} + e^Ty \quad \text{subject to} \quad \begin{bmatrix} 1 \\ G^{-1}ATy \end{bmatrix} \geq_0 0
$$

$$
y \geq 0
$$

$$
-(b + AG^{-2}g)^Ty \geq \bar{\lambda}_D
$$
4.3 WARM-START STRATEGIES

with corresponding dual

$$\min_{\alpha, \beta, v} \alpha - \bar{\lambda}_D \beta + \sqrt{2}$$

subject to

$$\begin{bmatrix} \alpha \\ v \end{bmatrix} \geq_0 Q 0$$

$$-AG^{-1} v + (b + AG^{-2} g) \beta \geq e$$

$$\beta \geq 0.$$ (4.28)

Upper bounds for $\Psi_P$ and $\Psi_D$ over parametric sets of problems are found by searching for robust solutions to (4.27) and (4.28). Note that in many cases, it is much easier to find robust solutions to (4.28) by making the substitution $v = \beta G^{-1} g + z$, and instead searching for robust solutions of

$$\min_{\alpha, \beta, z} \alpha - \bar{\lambda}_D \beta + \sqrt{2}$$

subject to

$$\begin{bmatrix} \alpha \\ \beta G^{-1} g + z \end{bmatrix} \geq_0 Q 0$$

$$-AG^{-1} z + \beta b \geq e$$

$$\beta \geq 0.$$ (4.29)

4.3 Warm-Start Strategies

For interior-point methods, a good initialization satisfies three conditions: it must be in the interior of the cone, it should be feasible, and it should be "close" to the central path. An initialization which meets these conditions and is also close to optimality is considered a warm start. So far, only cold starts have been considered. A warm start for a problem $p_2$ is usually derived from a previously solved problem $p_1$, where $\|p_2 - p_1\|$ is small. Notice that an optimal solution for problem $p_1$ cannot be considered a good warm start for $p_2$, since it is on the boundary of the cone. This is in contrast to warm-start strategies for boundary-traversing algorithms (e.g., the simplex algorithm), which are typically initialized at the boundary of the constraints.

Rather than choosing the optimizer of $p_1$, a better choice for a warm start is to select one of the iterates derived while solving $p_1$, which are interior and approximately follow
the central path. In [YW00], Yildirim and Wright examine how warm starts might be selected for a linear programming path-following algorithm. This initialization must be contained in a neighborhood of the central path of $p_2$. Their research relies on the work of Nunez and Freund [NF98], who demonstrate that as points along the central path get further away from optimality, the perturbation in the central path gets smaller. The optimal warm start is chosen by selecting the iterate which is closest to the optimizer of $p_1$, while still being contained in the neighborhood of the central path of $p_2$. Yildirim and Wright show that if there exists a sequence of iterations within a given neighborhood of the central path of $p_1$, and that $||p_1 - p_2||$ is small enough, that a warm start can be selected which solves $p_2$ in $O(\sqrt{n} \log(C(p)^2 \delta_p/\epsilon))$, where $\delta_p$ is a normalized measurement of the problem perturbation. It should be stated that these complexity results are valid only for very small perturbations in the problem data, although the initialization strategy may be valid for much larger data perturbations.

The self-dual embedding is very suitable for a warm-start strategy, since any point in the cone interior leads to a strictly feasible initial iterate for the embedded program. Centrality is also preserved between embeddings of different problems. To explain, consider a point $(\bar{X}, \bar{Z}, \tau, \kappa) \in \mathcal{N}(\beta)$ for a self-dual embedding of problem $p$. Suppose this point is to be used to initialize a problem $p' \neq p$. By initializing the embedding of $p'$ with $\bar{X}^0 = \bar{X}, \bar{Z}^0 = \bar{Z}$, etc., then this point will be strictly feasible and contained in $\mathcal{N}(\beta)$ for the embedded program of $p'$. Thus, iterates contained in $\mathcal{N}(\beta)$ for one problem are suitable initializations for perturbed problems.

As seen in Section 4.2.2, the complexity of solving the self-dual program depends logarithmically on the ratio $\Psi/\Gamma$. (The dependence of $\Psi$ and $\Gamma$ on $\delta$ is dropped in this section.) Suppose a self-dual embedding is posed with $\bar{X}^0, \bar{Z}^0, \bar{x}^0, \tau^0, \kappa^0$. Recall that $\bar{F}_0$, $\bar{c}$, and $\alpha$ represent the infeasibility and suboptimality of $\bar{X}^0, \bar{Z}^0$, etc. Define the measure

$$\eta \triangleq ||\bar{F}_0|| + ||\bar{c}|| + \max\{0, \alpha\}.$$  

In some sense, $\eta$ measures how "warm" the initial iterate is, where $\eta = 0$ indicates that the initial iterate is optimal. This measure can be related to the ratio $\Psi/\Gamma$ by the next proposition. For convenience, it is assumed that $\epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon$. 

Proposition 4.10. Assume that $p$ is primal-dual feasible. Then

$$\frac{\Psi}{\Gamma} \leq \frac{C^2(p)\eta}{\epsilon}.$$  

Proof. Let $X^*, x^*, Z^*$ be optimizers of $p$. Then

$$\langle X^*, \bar{Z}^0 \rangle + \langle \bar{X}^0, Z^* \rangle = \langle F_0, \bar{Z}^0 \rangle + \langle \tau^0 F_0 + F \bar{x}^0 - \bar{F}, Z^* \rangle$$

$$= \langle F_0, \bar{Z}^0 \rangle + \langle \tau^0 c - \bar{c} \rangle^T x^* - \tau^0 c^T x^* + c^T \bar{x}^0 - \langle \bar{F}, Z^* \rangle$$

$$= \langle F_0, \bar{Z}^0 \rangle + c^T \bar{x}^0 - \bar{c}^T x^* - \langle \bar{F}, Z^* \rangle$$

$$= \alpha - \kappa^0 - \bar{c}^T x^* - \langle \bar{F}, Z^* \rangle$$

Therefore

$$\Psi = \sup_{x^*, Z^*} \alpha - \kappa^0 - \bar{c}^T x^* - \langle \bar{F}, Z^* \rangle$$

$$\leq \alpha + C^2(p)(||\bar{c}|| + ||\bar{F}||)$$

$$\leq C^2(p)\eta.$$  

From the definition of $\Gamma$ (4.9), it is clear that $\Gamma \leq \frac{\xi}{\eta}$. The proposition follows from these bounds on $\Psi$ and $\Gamma$. \hfill \Box

This proposition shows that $\Psi/\Gamma \rightarrow 0$ as $\eta \rightarrow 0$. As should be expected, this indicates a benefit to using warm starts, since the complexity of a path-following algorithm applied to a self-dual embedding is

$$O \left( \sqrt{\beta + 1} \log \left( \frac{\Psi}{(1-\beta)\mu^0 \Gamma} \right) \right).$$

However, this is not the whole story, since the complexity also depends on $\mu^0 = \langle \bar{X}^0, \bar{Z}^0 \rangle + \tau^0 \kappa^0$. When initializing the self-dual problem, one should be careful that $\mu_0$ not be allowed to get too small, since the iteration bound can get arbitrarily large. To see how this can happen, choose any sequence of convergent iterates $\{\bar{X}^0_i\} \rightarrow \bar{X}^0$, $\{\bar{Z}^0_i\} \rightarrow \bar{Z}^0$, etc., such that the sequence $\mu^0_i \rightarrow 0$. If $\bar{X}^0$ and $\bar{Z}^0$ are not optimizers for a problem $p$, then it follows that $\lim_{i \rightarrow \infty} \Psi_i/\Gamma_i > 0$. Since $\mu^0_i \rightarrow 0$, the iteration bound goes to infinity. This is true, regardless of how close $\bar{X}^0$ and $\bar{Z}^0$ are to the real optimizers.

It is still unclear how to best choose a warm start from a sequence of iterates for the homogeneous self-dual method. From Proposition 4.10, it is inferred that the algorithm
complexity grows as $\mathcal{O}(\log(\eta^2/\mu^0))$, so perhaps it is best to choose the iterates which minimize $\eta^2/\mu^0$. This topic deserves more investigation.

Ultimately, the relevance of warm starts to on-line optimization must be addressed. There is no doubt that warm starts tend to speed up optimization. However, the added value may not be that great when compared to the certification issues warm starts would face for on-line optimization. In [Wri97], Wright observed that a warm start for a receding horizon control interior-point method tended to speed up the optimization by a factor of three compared to a cold start strategy. This is not a significant computational saving. Given that a cold start strategy is most likely much easier to certify, use of a warm start may be of limited value for on-line applications.
Chapter 5

Solution Approximations

From a complexity point of view, interior-point algorithms remain the best way to find an $\epsilon$-optimal solution to a general convex optimization problem. The previous chapter demonstrated that the number of computations required of these algorithms can be bounded, which is important for on-line applications. However, it still remains true that interior-point algorithms are fairly computationally intensive, and the bounds introduced in the previous chapter tend to be conservative. This chapter presents several alternatives to solving an interior-point algorithm on-line. The strategy taken here is to carry out any significant computations for an on-line application off-line, keeping the on-line computations to a minimum. Three different approaches are suggested here: table look-up, approximation of a parametric solution by an analytic function, and approximation of the constraints by ellipsoids. One caveat for these methods is that general guarantees are not provided for the accuracy of the solutions. In some cases, the approximation may be exact, and in others, arbitrarily poor.

5.1 Convex Programming as Set-Valued Functions

The problem of finding the minimizer(s) of a convex optimization problem can be represented as a point-to-set map

$$\mathcal{X}^* : \delta \mapsto \arg\min_{x \in \mathcal{X}(\delta)} c(\delta)^T x$$
where $\delta$ is the problem data parameter which specifies an objective vector $c(\delta)$ and a convex feasible region $X(\delta)$. Depending on the problem data, $X^*(\delta)$ may consist of a single point, a convex set of points, or the empty set.

For most applications, the goal is to find a single element contained in or close to the optimizing set $X^*(\delta)$ for any parameter in a given domain $D$. This problem is represented by another map $x^*: D \rightarrow X$. Typical requirements for a map $x^*$ might specify that the returned point is feasible and within a certain range of optimality, i.e., $x^*(\delta) \in X(\delta)$ and $c(\delta)^T x^*(\delta) \leq \inf_{x \in X(\delta)} c(\delta)^T x + \epsilon$ for some $\epsilon > 0$. Evaluating such a map might involve solving an optimization algorithm, evaluating an analytic expression, or even using a look-up table. Thus, the optimization algorithm is seen as one of several options for solving a convex program. Whether an optimization algorithm should be used versus a look-up table or other strategy depends on the available computational resources, the importance of optimality, and the ability to certify the optimization algorithm.

### 5.2 Table Look-Up via Robust Solutions

The table look-up strategy is perhaps the simplest approach to approximating the solutions of an optimization problem. The parameter domain $D$ is broken into a finite number of non-intersecting subsets, and each subset is associated with a single point $x^*(D_i)$. If an accurate solution approximation can be represented using a relatively small table (the amount of memory required to index the subsets of $D$), then the table look-up approach may be the best option due to its ease of implementation.

A basic requirement for the table look-up strategy considered in this section is feasibility of the returned solutions. The image $x^*(D_i)$ is defined to be a single point, therefore this point must be robustly feasible. The techniques presented in Section 3.2 are used here to find robust solutions.

It is assumed that the compact parameter space $D$ can be covered by a finite number of subsets, each of which has a robust solution. This is satisfied if the infeasibility detection algorithm from Section 3.4 outputs **FEAS**, which is guaranteed if all problem instances are feasible and well-posed.
Since the objective vector $c(\delta)$ is a parameterized function, several alternatives are available for defining the "optimal" robust solution. In this section, a solution is chosen either to minimize the worst case objective over a parameter subset $D_i$, or to minimize the average objective. Worst case objective minimization is formulated as the program

$$\min \lambda$$

subject to

$$F(\delta)x \in K$$

$$\lambda - c(\delta)^T x \geq 0 \text{ for all } \delta \in D_i.$$ 

This is identical to the robust semidefinite programming problem formulated in [GOL99], where they proved the uniqueness of the solution in the case of unstructured perturbations. Alternatively, minimization of the average cost is achieved by minimizing the cost function

$$\tilde{c}^T x = \left( \int_{\delta \in D_i} c(\delta) d\delta \right)^T x.$$

One drawback of using a table look-up map online is that as the parameter moves from one set $D_i$ to another $D_j$, the optimal solution "jumps" from $x^*(D_i)$ to $x^*(D_j)$, whereas it may be preferable to make a more continuous transition. An obvious solution is to associate the solution $x^*(D_i)$ only with the center $\delta_i$ of $D_i$. Then, the whole parameter space might be interpolated from these finite number of points, either using linear interpolation or splines. However, feasibility of the solutions is no longer guaranteed. This problem can be corrected by modifying the sets $D_i$ such that they do intersect. For example, suppose that $D_i$ and $D_j$ intersect such that $\delta_i \in D_j$ and $\delta_j \in D_i$, and associate with parameters $\lambda \delta_i + (1 - \lambda) \delta_j$ the solution $\lambda x^*(D_i) + (1 - \lambda) x^*(D_j)$ for all $\lambda \in [0, 1]$. Then by convexity of the feasible region, these solutions are feasible.

The access time for finding an element in an $n$-dimensional table with $m$ elements along each dimension is $n \log m$, which is polynomial. Unfortunately, the number of elements in the look-up table is $m^n$, which is exponential. This limits the tractability of table look-up to problems of small dimension.

**Example 5.1.** A common application of look-up tables in control is gain scheduling for linear time-varying systems. For example, consider the system

$$\dot{x} = A(\delta)x + B(\delta)u$$
dependent on parameter \( \delta \). The linear quadratic regulator (LQR) is defined as the control input \( u \) which minimizes

\[
J(x(0), \delta) = \int_0^\infty (x^T C^T C x + u^T R u) dt
\]

where \( R > 0 \). Given initial condition \( x(0) \) and fixed \( \delta \), it is well known [BEFB94] that the LQR problem is solved by the linear feedback law \( u = -R^{-1}B^T(\delta)Q^{-1}x \) where \( Q \) is the positive definite solution to

\[
\min_{Q \succeq 0} \quad x(0)^T Q^{-1} x(0)
\]

subject to

\[
\begin{bmatrix}
QA(\delta)^T + A(\delta)Q - B(\delta)R^{-1}B(\delta)^T & QC^T \\
CQ & -I
\end{bmatrix} \succeq 0.
\]

It turns out that the solution to this semidefinite program is independent of \( x(0) \), as long as \( x(0) \neq 0 \). Given a robust solution \( Q \) which satisfies the above constraints for all \( \delta \in \mathcal{D} \), then the linear feedback law is robustly stable over all parameterized linear systems, and the closed-loop performance satisfies \( J(x(0), \delta) \leq x(0)^T Q^{-1} x(0) \) for all \( \delta \in \mathcal{D} \). Thus, a gain scheduling table can be constructed using robust solutions to this semidefinite program. Of course, these gains only guarantee stability for linear time invariant systems, so this strategy may not be valid if \( \delta \) changes quickly over time.

**Example 5.2.** The optimization problem given in this example represents the minimization of a linear objective function over a parameterized ellipsoid. Although optimization problems given in this form can be solved analytically as a function of the parameter, the point here is to use the tools developed so far to construct a table look-up map. The problem data is given by

\[
\min_{\delta \in \mathbb{R}^5} \quad [\delta_4 \quad \delta_5] x
\]

subject to

\[
\begin{bmatrix}
1 & \frac{1}{2} \delta_1 & (x_1 - 2\delta_2) \\
\frac{1}{2} \delta_1 & 1 & (1 - \frac{1}{2} \delta_3) x_2 \\
(x_1 - 2\delta_2) & (1 - \frac{1}{2} \delta_3) x_2 & 1
\end{bmatrix} \succeq 0.
\] (5.2)

The parameters are restricted to \( \mathcal{D} = \{ \delta \in \mathbb{R}^5 \mid \|\delta\|_\infty \leq 1 \} \). This problem is feasible for all \( \delta \in \mathcal{D} \). However, there does not exist a robustly feasible solution over the whole...
5.3 Functional Solution Approximation

In the previous section, interpolation was mentioned as a possible “fix” for the discontinuous jumps between the solutions of a look-up table. Essentially, interpolation is used to

Table 5.1: Look-up table for Example 5.2. The solution $x^*$ and cost $c^T x^*$ is shown for each parameter segment of $\delta_2$ (indexed by column) and cost vector (indexed by row).

<table>
<thead>
<tr>
<th>$c \setminus \delta_2$</th>
<th>$[-1,-.6]$</th>
<th>$[-.6,-.2]$</th>
<th>$[-.2,.2]$</th>
<th>$[.2,.6]$</th>
<th>$[.6,1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1,0)$</td>
<td>$(-2.07,0.00)$</td>
<td>$(-1.27,0.00)$</td>
<td>$(-0.47,0.00)$</td>
<td>$(0.33,0.00)$</td>
<td>$(1.13,0.00)$</td>
</tr>
<tr>
<td></td>
<td>$-2.07$</td>
<td>$-1.27$</td>
<td>$-0.47$</td>
<td>$0.33$</td>
<td>$1.13$</td>
</tr>
<tr>
<td>$(1,1)/\sqrt{2}$</td>
<td>$(-1.96,-0.13)$</td>
<td>$(-1.16,-0.13)$</td>
<td>$(-0.36,-0.13)$</td>
<td>$(0.44,-0.13)$</td>
<td>$(1.24,-0.13)$</td>
</tr>
<tr>
<td></td>
<td>$-1.47$</td>
<td>$-0.91$</td>
<td>$-0.34$</td>
<td>$0.22$</td>
<td>$0.79$</td>
</tr>
<tr>
<td>$(0,1)$</td>
<td>$(-1.60,-0.40)$</td>
<td>$(-0.80,-0.40)$</td>
<td>$(0.00,-0.40)$</td>
<td>$(0.80,-0.40)$</td>
<td>$(1.60,-0.40)$</td>
</tr>
<tr>
<td></td>
<td>$-0.40$</td>
<td>$-0.40$</td>
<td>$-0.40$</td>
<td>$-0.40$</td>
<td>$-0.40$</td>
</tr>
</tbody>
</table>

parameter space. A robustly feasible solution exists if the range of $\delta_2$ is broken into three equivalent segments, i.e., $D_1 = \{ \delta \in D \mid \delta \in [-1,-1/3] \}$, $D_2 = \{ \delta \in D \mid \delta \in [-1/3,1/3] \}$, and $D_3 = \{ \delta \in D \mid \delta \in [1/3,1] \}$. Minimizing the worst case objective for each of these parameter sets via (5.1) produces the solutions $x^*(D_1) = (-1.13,0.00)$, $x^*(D_2) = (0.00,0.00)$, and $x^*(D_3) = (1.13,0.00)$, with objectives $\lambda(D_1) = 1.13$, $\lambda(D_2) = 0.00$, and $\lambda(D_1) = 1.13$. Note that the worst case objective $\lambda = 1.13$ is optimal for $\delta = (-1,-1,-1,-1,-0.75)$.

Average cost minimization is considered next. In this case, it is possible to work with a objective vector parameterized by an angle, $c(\theta) = [\sin(\theta) \ \cos(\theta)]$. Given a segment $[\theta_1, \theta_2]$ with $\theta_2 - \theta_1 < 2\pi$, the average (normalized) objective vector is represented by $c((\theta_1 + \theta_2)/2)$. For this example, eight objective vectors equally spaced around the unit circle are used. Table 5.1 shows a look-up table for this example. Only three objective vectors are shown here, since the remainder of the eight are symmetric to these three. The range of $\delta_2$ has been broken into five equal segments (indexed by the columns in the table), while the range of $\delta_1$ and $\delta_3$ are kept at $[-1,1]$. 

5.3 Functional Solution Approximation

In the previous section, interpolation was mentioned as a possible “fix” for the discontinuous jumps between the solutions of a look-up table. Essentially, interpolation is used to
find a continuous function which approximates a parameterized solution. This begs the question: is it possible to directly solve for such a function, rather than interpolate from a finite number of data points? For example, the data in Table 5.1 is a linear function of $\delta_2$, suggesting that directly searching for this linear function might be more appropriate.

There has been some exploration of representation of the solutions as analytic functions for parametric programming reported in the literature. In [BRT97], Berkelaar et al. showed that for certain parametric linear and quadratic programs with a single scalar parameter, the solutions can be characterized as a piecewise linear function of that parameter. Dua et al. [DBP+99] extended this work by showing how to compute piecewise linear solutions to a multi-dimensional parametric quadratic program in which the parameter $\delta$ appears linearly only in the right hand side of the constraints $Ax \leq b(\delta)$. These techniques characterize the solution exactly. However, they only apply to certain convex programming problems, so the goal of this section is more general.

Power series work exceptionally well at approximating analytic functions. Unfortunately, the functional relationship between $\delta$ and the corresponding solutions is generally not analytic. Even though this precludes the hope that solution might be approximated to arbitrary accuracy by a polynomial, a power series is suggested here for representation of the parameter to solution map.

In principle, the techniques which will be given in this section can be applied to any problem parameterized using the LFT introduced in Section 3.1. However, this section considers a simplified parametric problem. The constraints are assumed to be a parametric LMI, in which the data is a linear function of the parameter $\delta$ over the domain $\|\delta\|_\infty \leq 1$. Also, the highest polynomial order considered for the solution is quadratic. These restrictions are made here to simplify the notation. Two feasibility conditions are presented in the following propositions.

**Proposition 5.1.** Define the feasibility space as

$$\mathcal{X}_\delta \triangleq \{x \in \mathbb{R}^m \mid F(x) + \sum_{k=1}^L \delta_k G^k(x) \geq 0\}$$  \hspace{1cm} (5.3)
where

\[ G^k(x) \triangleq G^k_0 + \sum_{i=1}^{m} G^k_i x_i \]

and \( G^k_i \in \mathbb{R}^{m \times n} \) for \( i = 0, \ldots, m \) and \( k = 1, \ldots, L \). Suppose \( x^*(\delta) \) is a linear function with

\[ x^*(\delta) = \alpha + B\delta \]

where \( \alpha \in \mathbb{R}^m \) and \( B \in \mathbb{R}^{m \times L} \). Then \( x^*(\delta) \in X_\delta \) for all \( \|\delta\|_\infty \leq 1 \) if there exist symmetric positive definite matrices \( S_1, \ldots, S_L \) and skew-symmetric matrices \( T_i, U_{jk} \) for \( 1 \leq i \leq L \) and \( 1 \leq j < k \leq L \) such that

\[
\begin{bmatrix}
\Pi & \Xi_1 & \Xi_2 & \cdots & \Xi_L \\
\Xi_1^T & \Phi_1 & \Psi_{12} & \cdots & \Psi_{1L} \\
\Xi_2^T & \Psi_{12}^T & \ddots & \cdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \Psi_{(L-1)L} \\
\Xi_L^T & \Psi_{1L}^T & \cdots & \Psi_{(L-1)L}^T & \Phi_L
\end{bmatrix} \succeq 0,
\]

(5.4)

where

\[
\Pi = F_0 + \sum_{i=1}^{m} F_i \alpha_i - \sum_{i=1}^{L} S_i \\
\Xi_j = \frac{1}{2} \left( \sum_{i=1}^{m} F_i B_{ij} + G_0^j + \sum_{i=1}^{m} G_i^j \alpha_i \right) + T_j \\
\Phi_j = \sum_{i=1}^{m} G_i^j B_{ij} + S_j \\
\Psi_{jk} = \frac{1}{2} \sum_{i=1}^{m} (G_i^j B_{ik} + G_i^k B_{ij}) + U_{jk}
\]

and \( B_{ij} \) indicates the \((i, j)\) element of \( B \).

Proof. Let \( \delta \) be given such that \( \|\delta\|_\infty \leq 1 \). Define the matrices

\[
Q_1 = F(x^*(\delta)) + \sum_{k=1}^{L} \delta_k G^k(x^*(\delta)) \\
Q_2 = \sum_{i=1}^{L} (S_i(1 - \delta_i^2) + \delta_i T_i + \sum_{j=i+1}^{L} \delta_i \delta_j U_{ij}).
\]
Suppose that (5.4) is satisfied. Then $y^T(Q_1 - Q_2)y \geq 0$ for all $y \in \mathbb{R}^n$. Furthermore, since $S_i \succeq 0$, $T_i = -T_i^T$ and $U_{ij} = -U_{ij}$, it follows that $yQ_2y \geq 0$, therefore $Q_1 \succeq 0$. □

**Proposition 5.2.** Define the feasibility space as

$$
\mathcal{X}_\delta \triangleq \{ x \in \mathbb{R}^m \mid F(x) + \sum_{k=1}^{L} \delta_k G_k \succeq 0 \}
$$

where $G_k \in \mathbb{R}^{m \times n}$ for $k = 1, \ldots, L$. Suppose $x^*(\delta)$ is a quadratic function with

$$
x^*_i(\delta) \triangleq \alpha_i + \beta_{x_i}^T \delta + \delta^T \Gamma_{x_i} \delta
$$

where $\alpha_i \in \mathbb{R}$, $\beta_{x_i} \in \mathbb{R}^L$, and $\Gamma_{x_i} \in \mathbb{R}^{L \times L}$ for $i = 1, \ldots, m$. Then $x^*(\delta) \in \mathcal{X}_\delta$ for all $\|\delta\|_{\infty} \leq 1$ if there exist symmetric positive definite matrices $S_1, \ldots, S_L$ and skew-symmetric matrices $T_i, U_{jk}$ for $1 \leq i \leq L$ and $1 \leq j < k \leq L$ such that

$$
\begin{bmatrix}
\Pi & \Xi_1 & \Xi_2 & \cdots & \Xi_L \\
\Xi_1^T & \Phi_1 & \Psi_{12} & \cdots & \Psi_{1L} \\
\Xi_2^T & \Psi_{12}^T & \ddots & \cdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \Psi_{(L-1)L} \\
\Xi_L^T & \Psi_{1L}^T & \cdots & \Psi_{(L-1)L}^T & \Phi_L
\end{bmatrix} \succeq 0,
$$

(5.5)

where

$$
\Pi = F_0 + \sum_{i=1}^{m} F_i \alpha_i - \sum_{i=1}^{L} S_i,
$$

$$
\Xi_j = \frac{1}{2} (G_j + \sum_{i=1}^{m} F_i \beta_{x_i, j}) + T_j,
$$

$$
\Phi_j = \sum_{i=1}^{m} F_i \Gamma_{x_i, (j,j)} + S_j,
$$

$$
\Psi_{jk} = \frac{1}{2} \sum_{i=1}^{m} F_i \Gamma_{x_i, (j,k)} + U_{jk},
$$

and $\beta_{x_i, j}$ indicates the $j$th element of $\beta_{x_i}$, and $\Gamma_{x_i, (j,k)}$ indicates the $(j, k)$ element of $\Gamma_{x_i}$.

**Proof.** Similar to the proof of Proposition 5.1. □

Conditions (5.4) and (5.5) are sufficient for inclusion, but it is not known whether or under what conditions they are also necessary. Experience suggests that this method
5.3. FUNCTIONAL SOLUTION APPROXIMATION

works very well on many problems. The proof of Proposition 5.1 is fairly straightforward, giving little insight into the mechanism at work here. In Section 5.4, the same technique (which is really an application of the $S$-procedure) is used to find a condition for the inclusion of an ellipsoid in an LMI. The reader is referred to the proof of Proposition 5.4, which is much more detailed and hopefully gives more insight into this problem.

In a sense, any solution $x^*(\delta)$ satisfying (5.4) or (5.5) can be seen as a robust parametric solution over a compact parametric space $\mathcal{D}$, similar to the robust solutions introduced in Section 3.2. In fact, if condition (3.6) from Proposition 3.4 is satisfied for some solution $\bar{x}$, then (5.4) and (5.5) are feasible for the same solution, i.e., $\alpha = \bar{x}$, $B = 0$, $\beta_x = 0$, $\Gamma_x = 0$. This is easily shown, by noting that in Proposition 3.4, $M_{11}(x) = F(x)$, $M_{12} = [I \cdots I]$, $\Delta = \text{diag}(\delta_I, \ldots, \delta_L)$, $M_{21} = \frac{1}{2}[G^1(x) \cdots G^L(x)]^T$, $M_{22} = 0$, $S = \{\text{diag}(S_1, \ldots, S_L) : S_i = S_i^T \in \mathbb{R}^{n \times n}\}$, and $T = \{\text{diag}(T_1, \ldots, T_L) : T_i = -T_i^T \in \mathbb{R}^{n \times n}\}$, which turn out to be identical to conditions (5.4) and (5.5).

Since the robust parametric solution generalizes the robust solution, any place in this thesis which uses robust solutions can be replaced with robust parametric solutions with possibly less conservative results. For example, the infeasibility detection algorithm in Section 3.4 may have a faster run time if it searches for robust parametric solutions instead of robust solutions. Also, in the table look-up scheme proposed in Section 5.2, one can imagine a table of robust parametric solutions. Just as in Section 5.2, minimization of the cost function could mean minimization of the worst case cost by means of (5.1), or minimization of the average cost, achieved by minimizing

$$J(\alpha, \beta, \Gamma) = \int_{||\delta||_{\infty} \leq 1} c(\delta)^T x^*(\delta) d\delta,$$

which is linear in the variables $\alpha$, $\beta$, and $\Gamma$.

The next example shows the derivation of a quadratic solution to a problem with linear parameter dependent constraints. In this case, the parametric solution turns out to be exact.

**Example 5.3.** Consider the following optimization problem parameterized in the vector
\[ \delta \in \mathbb{R}^2: \]

\[
\min_{\gamma \in \mathbb{R}} \quad \gamma \\
\text{subject to} \quad \delta^T Q^{-1} \delta \leq 1 \\
\| QY \| \leq \gamma \\
Y = \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix},
\]

where \( Q \in \mathbb{R}^{2\times2} \) is a symmetric matrix. The matrix form of the constraints is

\[
\begin{bmatrix} 1 & \delta^T \\ \delta & Q \end{bmatrix} \succeq 0, \quad \begin{bmatrix} \gamma I & QY \\ Y^T Q & \gamma I \end{bmatrix} \succeq 0,
\]

which can be restated in standard form

\[
F_0 + \sum_{i=1}^{4} F_i x_i + G_1 \delta_1 + G_2 \delta_2 \succeq 0.
\]

The solutions \( \gamma(\delta) \) and \( Q(\delta) \) are parameterized as quadratic functions of \( \delta \), constrained according to (5.5). Minimizing the average \( \gamma(\delta) \) over all \( \| \delta \|_\infty \leq 1 \) yields the parametric solution

\[
\gamma(\delta) = 3.4862 + 4.4361 \delta^2 + 2.8562 \delta^2
\]

\[
Q(\delta) = \begin{bmatrix} 1.98 & -1.54 \\ -1.54 & 2.42 \end{bmatrix} \delta^2 + \begin{bmatrix} 2.52 & -1.96 \\ -1.96 & 3.08 \end{bmatrix} \delta_1 \delta_2 + \begin{bmatrix} 1.62 & -1.26 \\ -1.26 & 1.98 \end{bmatrix} \delta_2^2.
\]

Incidentally, this parameterized solution appears to yield the optimal solution to the original problem for all values of \( \delta \). This conjecture is not simple to confirm analytically, but has been supported by numerical experiments.

### 5.4 Ellipsoidal Constraint Approximation

Another attractive alternative to on-line optimization is to approximate the constraints with an inscribed ellipsoid. An ellipsoid can be described as the image of the unit ball under a unique linear transformation as

\[
\mathcal{E}(P, \bar{x}) \triangleq \{ Pd + \bar{x} \mid d \in \mathcal{B}^m \},
\]
5.4. ELLIPSOIDAL CONSTRAINT APPROXIMATION

where \( P = P^T > 0 \) and the unit ball is defined to be \( B^p \triangleq \{ x \in \mathbb{R}^p \mid \| x \| \leq 1 \} \). The volume of this ellipsoid is proportional to the determinant of \( P \), or \( \det P \).

The ellipsoid is a very natural convex constraint to work with, since optimizing a linear function over an ellipsoid can be performed analytically by the following expression:

\[
\arg\min_{x \in \mathcal{C}(P,\bar{x})} c^T x = \bar{x} - \frac{P^2 c}{\sqrt{c^T P^2 c}}.
\]

Clearly, computing the optimal solution analytically will generally be much quicker than any iterative optimization procedure such as an interior-point method.

In this section, an ellipsoid is considered to be valid for a set of constraints if it is contained within the constraints. The objective is to find a valid ellipsoid which best approximates a set of constraints. If the objective function \( c \) is free to point in any direction, then clearly the best valid ellipsoid is that of maximum volume. This is assumed for most of this section. However, if \( c \) is known to reside in a subspace, or points preferentially in one direction, then there may be better constraint approximations than the maximum volume ellipsoid. These considerations are addressed in Section 5.4.3.

The LMI constraints are considered to be a linear function of a parameter \( \delta \), using the parameterization defined by (5.3) from the previous section. In Section 5.4.1, a procedure for finding inscribed ellipsoids in a fixed feasible space is developed. Similar work has explored the problem of finding the maximum volume ellipsoid contained within a polytope (e.g., see [NN94]). Section 5.4.2 generalizes this to parameterized ellipsoids. Before these methods are presented, a brief example demonstrates the utility of ellipsoidal constraint approximation on a receding horizon control problem.

**Example 5.4.** This example examines the receding horizon control of a double integrator subject to fixed actuator saturation constraints. The plant is given by

\[
P(z) = \frac{z + 1}{2(z-1)^2} \sim \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & .5 \\ 0 & 1 & 0 \end{bmatrix}.
\]

The system is to be steered to a reference trajectory \( r = 1 \) (a step input), subject to the saturation constraints \(-0.5 \leq u \leq 0.5\). The problem is posed over a horizon length of 15
with actuator weight 0.1, defined by

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=1}^{15} [(r_j - y_j)^2 + 0.1 u_{j-1}^2] \\
\text{subject to} & \quad y_k = \sum_{j=1}^{k} [CA^j x_0 + CA^{j-1} Bu_{j-1}], \\
& \quad -0.5 \leq u_{k-1} \leq 0.5, \quad k = 1, \ldots, 15,
\end{align*}
\]  

more compactly represented by

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} u^T Gu + g^T u \\
\text{subject to} & \quad -0.5 \leq u_{k-1} \leq 0.5, \quad k = 1, \ldots, 15,
\end{align*}
\]

where \( g \) is a function of the initial state \( x_0 \). Assuming that the initial state of the system has a position at \(-1\) and velocity at \(-1\), i.e. \( x_0 = [-1 \ -1]^T \), the optimal trajectory and control input for this system is depicted in Figure 5.1.

The constraint for this quadratic program is a 15-dimensional hypercube centered at the origin. The largest volume ellipsoid inscribed in these constraints is the hypersphere described by \( \sum_{k=1}^{15} u_{k-1}^2 \leq 0.25 \).

The online optimization problem has now been reduced to the minimization of a
convex quadratic function over an ellipsoid, specifically
\[
\min_x \quad x^T T_0 x + 2u_0^T x
\]
subject to \( x^T T_1 x + 2u_1^T x + v_1 \leq 0 \).

The dual of this problem is
\[
\max_{\lambda \geq 0} \quad -(u_0 + \lambda u_1)^T (T_0 + \lambda T_1)^{-1} (u_0 + \lambda u_1) + \lambda v_1,
\]
which can be optimized using a simple line search over \( \lambda \). This objective can be quickly computed if the matrices \( T_0 \) and \( T_1 \) are first diagonalized, e.g. find a matrix \( X \) such that \( X^T T_0 X \) and \( X^T T_1 X \) are both diagonal. Furthermore, since \( T_0 \) and \( T_1 \) are both fixed in the receding horizon control problem, \( X \) only needs to be computed once off-line. Finally, once the optimal dual variable \( \lambda \) is known, the optimal primal variable can be recovered by \( x_{\text{opt}} = -(T_0 + \lambda T_1)^{-1} (u_0 + \lambda u_1) \).

Figure 5.1 compares the trajectory realized by this constraint approximation scheme to the optimal trajectory. Further optimization is possible along the direction derived from the hypersphere constraint, by multiplying the solution by a scale factor and optimizing the scale factor subject to the original hypercube constraints. This problem has an analytic solution, and slightly improves the performance of the control system, as shown by the third trajectory in Figure 5.1.

5.4.1 Inscribed Ellipsoids

For the moment, the constraint parameter \( \delta \) is ignored, and the feasible set is considered to be fixed, i.e., \( \mathcal{X} \nless \mathcal{X}_0 \). The set \( \mathcal{X} \) is assumed to be compact and nonempty, in order to prevent the existence of infinite volume ellipsoids.

The objective is to find the maximum volume ellipsoid contained in the feasible region, i.e., the solution to the problem
\[
\max_{\mathcal{E}(P, \bar{x}) \subseteq \mathcal{X}} \log \det P
\]
If the constraint \( \mathcal{E}(P, \bar{x}) \subseteq \mathcal{X} \) can be written as an LMI, then this problem may be solved as a \( \text{maxdet} \) problem [VBW98]. A problem in this form can be solved rapidly using
interior point methods (for example, see the software package maxdet [WVB96]). It is also possible to convert a maxdet problem into a standard semidefinite program with the introduction of additional variables (see [NN94]).

It turns out that the constraint \( E(P, \tilde{x}) \subseteq \mathcal{X} \) can be expressed (possibly conservatively) as an LMI. The constraint is equivalent to

\[
F(Pd + \tilde{x}) \succeq 0 \quad \forall d \in \mathcal{B}^m,
\]

which can in turn be characterized as

\[
y^T \left[ F_0 + \sum_{i=1}^{m} F_i \left( \tilde{x}_i + \sum_{j=1}^{m} p_{ij} d_j \right) \right] y \geq 0 \quad \forall y \in \mathbb{R}^n, d \in \mathcal{B}^m. \tag{5.8}
\]

where \( p_{ij} \) is the \((i,j)\) element of \( P \).

Variable \( z = (z_1, \ldots, z_m) \) where \( z_i \in \mathbb{R}^n \) is now introduced. The following lemma will aid in the development of a feasibility criterion.

**Lemma 5.3.** Let \( \mathcal{W} = \{ (y, z) \in \mathbb{R}^n \times (\mathbb{R}^n)^m \mid \exists d \in \mathcal{B}^m, z_i = d_i y \} \) Then \((y, z) \in \mathcal{W}\) if and only if

\[
\sum_{i=1}^{m} z_i^T S z_i \leq y^T S y, \quad \forall S = S^T \succeq 0 \tag{5.9}
\]

\[
z_i^T T y = 0, \quad \forall T = -T^T.
\]

This lemma is a direct corollary to a lemma appearing in [FAG96], so the proof is omitted here. The feasibility condition is stated in the following proposition. The proof of this proposition is intentionally stronger than it needs to be.

**Proposition 5.4.** Ellipsoid \( \mathcal{E}(P, \tilde{x}) \subseteq \mathcal{X} \) whenever there exists a symmetric positive definite matrix \( S \) and skew-symmetric matrices \( T_i, U_{jk} \), for \( 1 \leq i \leq m \) and \( 1 \leq j < k \leq m \) such that

\[
\begin{bmatrix}
F(\tilde{x}) - S & \Xi_1 & \Xi_2 & \cdots & \Xi_m \\
\Xi_1^T & S & U_{12} & \cdots & U_{1m} \\
\Xi_2^T & -U_{12} & \ddots & \ddots & \vdots \\
\vdots & \vdots & \ddots & \ddots & U_{(m-1)m} \\
\Xi_m^T & -U_{1m} & \cdots & -U_{(m-1)m} & S
\end{bmatrix} \succeq 0, \tag{5.10}
\]
where
\[ \Xi_j = \frac{1}{2} \left( \sum_{i=1}^{m} F_i p_{ij} + T_j \right). \]

Proof. Concatenate the variables \( M = (S, T_1, \ldots, T_m, U_{12}, \ldots, U_{(m-1)m}) \), and define the set
\[ \mathcal{L} = \{ M \mid S = S^T \succeq 0, T_i = -T_i^T, U_{jk} = -U_{jk}^T, 1 \leq i \leq m, 1 \leq j < k \leq m \}. \]
Let
\[ \mathcal{Q}_1(P, \bar{x}) = \{(y, z) \mid y^T(F_0 + \sum_{i=1}^{m} F_i \bar{x}_i)y + y^T \sum_{i=1}^{m} \sum_{j=1}^{m} p_{ij} F_i z_j \geq 0 \} \]
and
\[ \mathcal{Q}_2(M) = \{(y, z) \mid \sum_{i=1}^{m} z_i^T S z_i - y^T S y + \sum_{i=1}^{m} \sum_{j=1}^{m} z_i^T T_i z_i + 2 \sum_{i=1}^{m} \sum_{j=i+1}^{m} z_i^T U_{ij} z_j \leq 0 \}. \]

The following four statements comprise the proof, and are each demonstrated separately.

\begin{align*}
(5.10) \text{ feasible} & \iff \mathcal{Q}_2(M) \subseteq \mathcal{Q}_1(P, \bar{x}) \quad (5.11a) \\
\mathcal{Q}_2(M) \subseteq \mathcal{Q}_1(P, \bar{x}) & \implies \bigcap_{M \in \mathcal{L}} \mathcal{Q}_2(M) \subseteq \mathcal{Q}_1(P, \bar{x}) \quad (5.11b) \\
\bigcap_{M \in \mathcal{L}} \mathcal{Q}_2(M) \subseteq \mathcal{Q}_1(P, \bar{x}) & \iff \mathcal{W} \subseteq \mathcal{Q}_1(P, \bar{x}) \quad (5.11c) \\
\mathcal{W} \subseteq \mathcal{Q}_1(P, \bar{x}) & \iff \mathcal{E}(P, \bar{x}) \subseteq \mathcal{X} \quad (5.11d)
\end{align*}

(i) Proof of (5.11a). The \( \mathcal{S} \)-procedure [BEFB94] guarantees that \( \mathcal{Q}_2(M) \subseteq \mathcal{Q}_1(P, \bar{x}) \)
if and only if
\begin{align*}
y^T(F_0 + \sum_{i=1}^{m} F_i \bar{x}_i)y + y^T \sum_{i=1}^{m} \sum_{j=1}^{m} p_{ij} F_i z_j + \tau \left( \sum_{i=1}^{m} \sum_{j=1}^{m} z_i^T S z_i - y^T S y + \sum_{i=1}^{m} \sum_{j=i+1}^{m} z_i^T U_{ij} z_j \right) \geq 0 \quad \forall y, z_i \in \mathbb{R}^n \quad (5.12)
\end{align*}
for some $\tau \geq 0$. The LMI (5.10) considers this feasibility question over all $M \in \mathcal{L}$, and since $\mathcal{L}$ is a cone, $\tau$ can be dropped.

(ii) Proof of (5.11b). Trivial.

(iii) Proof of (5.11c). From Lemma 5.3, it follows that

$$\mathcal{W} = \bigcap_{M \in \mathcal{L}} Q_2(M).$$

(iv) Proof of (5.11d). Inequality (5.8) states that $\mathcal{E}(P, \bar{x}) \subseteq \mathcal{X}$ if and only if

$$y^T(F_0 + \sum_{i=1}^{m} F_i \bar{x}_i)y + y^T \sum_{i=1}^{m} \sum_{j=1}^{m} p_{ij} F_i z_j \geq 0 \quad \forall (y, z) \in \mathcal{W}. \quad (5.13)$$

The following propositions demonstrate that this condition is non-conservative if $\mathcal{X}$ is described by a polytope or intersection of ellipsoids (simpler non-conservative tests for these specific problems can be found in [BEFB94] and [NN94]).

**Proposition 5.5.** Suppose $\mathcal{X}$ is the feasible space of a polytope $A\bar{x} \leq b$. Then (5.10) can be satisfied if and only if $\mathcal{E}(P, \bar{x}) \subseteq \mathcal{X}$. Furthermore, if (5.10) is feasible, then it will be feasible for $S$ diagonal, $T_i = 0$ and $U_{jk} = 0$, $1 \leq i \leq m$, $1 \leq j < k \leq m$.

**Proof.** Suppose that (5.10) holds. Then $\mathcal{E}(P, \bar{x}) \subseteq \mathcal{X}$ by Proposition 5.4. Conversely, suppose that $\mathcal{E}(P, \bar{x}) \subseteq \mathcal{X}$. Let $a_i^T$ denote the rows of $A$ for $i = 1, \ldots, n$. Then

$$\max_{x \in \mathcal{E}(P, \bar{x})} a_i^T \bar{x} \leq b_i \quad \text{for} \quad i = 1, \ldots, n.$$ 

This can be restated as

$$a_i^T \bar{x} + \sqrt{a_i^T P^2 a_i} \leq b_i, \quad i = 1, \ldots, n. \quad (5.14)$$

Define $s_i = \frac{1}{2} \sqrt{a_i^T P^2 a_i}$. Then (5.14) is equivalent to

$$b_i - a_i^T \bar{x} - s_i - \frac{1}{4s_i} a_i^T P^2 a_i \geq 0, \quad i = 1, \ldots, n,$$

which can be rewritten in matrix form as

$$\begin{bmatrix} b_i - a_i^T \bar{x} - s_i & -\frac{1}{2} a_i^T P \\ -\frac{1}{2} P a_i & I s_i \end{bmatrix} \succeq 0, \quad i = 1, \ldots, n.$$
5.4. ELLIPSOIDAL CONSTRAINT APPROXIMATION

Substitution reveals that this is equivalent to (5.10) with

\[
S = \text{diag} \ s_i, \quad F_0 = \text{diag} \ b \\
T_i = 0, \quad U_{jk} = 0 \quad F_j = -\text{diag} \ a_{ij}, \quad j = 1, \ldots, m,
\]

where \( a_{ij} \) is the \( ij \) element of \( A \).

\[\square\]

**Proposition 5.6.** Suppose \( \mathcal{X} \) is the intersection of ellipsoids \( \mathcal{E}_1, \ldots, \mathcal{E}_N \). Then \( \mathcal{E}_0 \subseteq \mathcal{X} \) if and only if (5.10) is feasible. Furthermore, if (5.10) is feasible, then it will be feasible for \( U_{ij} = 0, 1 \leq i < j \leq m \).

**Proof.** Let the ellipsoids \( \mathcal{E}_i \) be described by \( \mathcal{E}_i = \{ P_i d + x_i \mid d \in \mathcal{B}^m \} \). The lemma in §3.7.3 of [BEFB94] demonstrates that \( \mathcal{E}_0 \subseteq \bigcap_{i=1}^{N} \mathcal{E}_i \) if and only if

\[
\begin{bmatrix}
-P_i^2 & x_i - x_0 & P_0 \\
(x_i - x_0)^T & \lambda_i - 1 & 0 \\
P_0 & 0 & -\lambda_i I
\end{bmatrix} \preceq 0, \quad i = 1, \ldots, N \tag{5.15}
\]

for some nonnegative \( \lambda_1, \ldots, \lambda_N \). To prove this proposition, it is only necessary to prove that the feasibility of (5.15) is equivalent to the feasibility of (5.10). Condition (5.10) implies \( \mathcal{E}_0 \subseteq \bigcap_{i=1}^{N} \mathcal{E}_i \), and thus also implies that (5.15) is feasible. Thus, it suffices to show that (5.15) implies (5.10).

Using Schur complements, \( \mathcal{E}_i = \{ x \mid (x - x_i)^T P_i^{-2}(x - x_i) \leq 1 \} \) is equivalent to the LMI

\[
\begin{bmatrix}
P_i^2 & x - x_i \\
(x - x_i)^T & 1
\end{bmatrix} \succeq 0. \tag{5.16}
\]

Consider the satisfaction of (5.10) for each individual ellipsoid \( \mathcal{E}_i, i = 1, \ldots, N \). Choose the following matrices:

\[
S_i = \begin{bmatrix} 0 & 0 \\ 0 & \lambda_i \end{bmatrix} \quad i = 1, \ldots, N \\
T_j = \begin{bmatrix} 0 & p_j \\ -p_j^T & 0 \end{bmatrix} \quad j = 1, \ldots, m, \\
U_{ij} = 0 \quad 1 \leq i < j \leq m
\]
where $p_j$ is the $j$th column of matrix $P_0$. Substituting the LMI (5.16), $S$, $T$ and $U_{ij}$ into (5.10), and removing the rows and columns containing only zeros, yields the constraint

$$
\begin{bmatrix}
P_i^2 & x_0 - x_i & P_0 \\
(x_0 - x_i)^T & 1 - \lambda_i & 0 \\
P_0 & 0 & \lambda_i I
\end{bmatrix} \succeq 0, \quad i = 1, \ldots, N,
$$

(5.17)

which is clearly equivalent to (5.15). Hence, if (5.15) is feasible, (5.10) will also be feasible.

**Remark 5.1.** While it has been proved that the feasibility of (5.10) is a sufficient condition for $E(P, \bar{x}) \subseteq \mathcal{X}$, Proposition 5.5 and Proposition 5.6 show that there are certain assumptions under which (5.10) is also a necessary condition. It is presently unknown under what other conditions if any, $E(P, \bar{x}) \subseteq \mathcal{X}$ only if feasibility holds for (5.10). In fact, a counter example to necessity has yet to be found, so it may be true that (5.10) is a necessary condition in general. Observe that the proof to Proposition 5.4 is at times stronger than it needs to be. In particular, only the ($\Rightarrow$) direction in (5.11a)-(5.11d) is necessary for the proof, yet the ($\Leftarrow$) direction is also present for three out of the four statements. It should be recognized that the ($\Leftarrow$) directions are not trivial results, especially for (5.11a) which is due to the non-conservatism of the $S$-procedure for two quadratic sets. The only ($\Leftarrow$) direction missing is for (5.11b), which has already been seen to be true for under some conditions, e.g., those in Proposition 5.5 and Proposition 5.6.

**Remark 5.2.** Notice that the variable $U$ could be dropped from Proposition 5.4 without affecting the proof. Nor is $U$ required for the linear and quadratically constrained problems in Proposition 5.5 and Proposition 5.6. It is tempting to think that the variable $U$ might not be required at all. The following example shows that this is not the case.

**Example 5.5.** The following cubic constraints and LMI are equivalent.

$$
x_1^3 + x_2^3 \leq x_1 x_2,
$$

$$
x_1 \geq 0, \quad x_2 \geq 0
$$

$$
\iff
$$

$$
\begin{bmatrix}
1 & x_1 & x_2 \\
x_1 & x_2 & 0 \\
x_2 & 0 & x_1
\end{bmatrix} \succeq 0.
$$

The feasible region of these constraints is shown in Figure 5.2. Also shown in this figure are two inscribed ellipsoids. The dash-dot ellipsoid represents the maximum volume
ellipsoid restricting $U_{12} = 0$. This ellipsoid is clearly not the maximum volume inscribed ellipsoid. Alternatively, the dashed ellipsoid was derived with $U_{12}$ as a free variable, with solution

$$P = \begin{bmatrix} 0.2357 & 0.0786 \\ 0.0786 & 0.2357 \end{bmatrix}, \quad \bar{x} = \begin{bmatrix} 0.2778 \\ 0.2778 \end{bmatrix}.$$  

Visual inspection verifies that this ellipsoid is much larger than the dash-dot ellipsoid, and is most likely the maximum volume inscribed ellipsoid. This example shows that the variables $U_{ij}$ are necessary in general, even though Proposition 5.5 and Proposition 5.6 demonstrated that they are not needed for linear and quadratic constraints.

### 5.4.2 Parameterized Ellipsoids

The previous section considered the problem of finding the maximum volume ellipsoid contained within a fixed convex region. This is a candidate method to simplify on-line optimization problems with fixed constraints and changing objective functions. However, it is more typical that the constraints of an on-line optimization problem will also change over time. The parameterization for the feasible space $\mathcal{X}_\delta$ given by (5.3) is used here.

Assume that $\mathcal{X}_\delta \neq \emptyset$ for all $\delta \in \mathcal{B}^L$, although it may be possible that $\bigcap_{\delta \in \mathcal{B}^L} \mathcal{X}_\delta = \emptyset$. 

![Figure 5.2: Maximum volume ellipsoids for cubic constraints.](image-url)
The objective is to find a parameterized ellipsoid $\mathcal{E}_\delta$ such that $\mathcal{E}_\delta \subseteq \mathcal{X}_\delta$ for all $\delta \in \mathcal{B}_L$.

Define the parameterized ellipsoid as

$$\mathcal{E}_\delta(P, \bar{x}) \triangleq \{ P(\delta)d + \bar{x}(\delta) \mid d \in \mathbb{B}^m \}$$

where $P(\delta) \succeq 0$ for all $\delta \in \mathcal{B}_L$. In this section, $P(\delta)$ and $\bar{x}(\delta)$ are restricted to affine functions of $\delta$, i.e.,

$$\bar{x}(\delta) \triangleq \bar{x}^0 + \sum_{i=1}^L \bar{x}^i \delta_i$$

$$P(\delta) \triangleq P_0 + \sum_{i=1}^L P_i \delta_i$$

with $\bar{x}^0, \ldots, \bar{x}^L \in \mathbb{R}^m$, $P_0, \ldots, P_L \in \mathbb{R}^{m \times m}$. Polynomial parameterizations will be considered later, although restricting the parameterizations to affine forms does simplify the conditions enormously.

A sufficient LMI condition for $\mathcal{E}_\delta \subseteq \mathcal{X}_\delta$ for all $\delta \in \mathcal{B}_L$ is given below.

**Proposition 5.7.** Ellipsoid $\mathcal{E}_\delta \subseteq \mathcal{X}_\delta$ for all $\delta \in \mathcal{B}_L$ whenever there exists symmetric positive definite matrices $S_1, S_2, Q, R_1, \ldots, R_L$, and skew-symmetric matrices $T_1, \ldots, T_m, U_1, \ldots, U_L, V_{11}, \ldots, V_{LL}$ such that

$$
\begin{bmatrix}
\Gamma & \Xi_1 & \cdots & \Xi_m & \Theta_1 & \cdots & \Theta_L & V_{11} & \cdots & V_{1L} & \cdots & V_{LL} \\
\Xi_1^T & S_1 & 0 & \Psi_{11} & \cdots & \Psi_{1L} & \Upsilon_{111} & \cdots & \Upsilon_{1LL} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\Xi_m^T & 0 & S_1 & \Psi_{m1} & \cdots & \Psi_{mL} & \Upsilon_{m11} & \cdots & \Upsilon_{m1L} & \cdots & \Upsilon_{mLL} \\
\Theta_1^T & \Phi_{11} & \cdots & \Phi_{1m} & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\Theta_L^T & \Phi_{L1} & \cdots & \Phi_{Lm} & \Phi_{LL} & 0 & \cdots & 0 \\
V_{11}^T & \Upsilon_{111} & \cdots & \Upsilon_{m11} & 0 & \cdots & 0 & \Omega_{11} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
V_{1L}^T & \Upsilon_{11L} & \cdots & \Upsilon_{m1L} & \cdots & \cdots & \Omega_{1L} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
V_{LL}^T & \Upsilon_{1LL} & \cdots & \Upsilon_{mLL} & 0 & \cdots & 0 & \Omega_{LL}
\end{bmatrix} \preceq 0,
$$

(5.18)
where

\[
\Gamma = F_0 + \sum_{i=1}^{m} F_i \bar{x}_i^0 - S_1 - S_2 - Q,
\]

\[
\Xi_r = \frac{1}{2} \left( \sum_{j=1}^{m} F_j \varphi_{rj0} + T_r \right)
\]

\[
\Theta_r = \frac{1}{2} \left( \sum_{i=1}^{m} F_i \bar{x}_i^r + G_0^r + \sum_{i=1}^{m} G_i^r \bar{x}_i^0 + U_r \right),
\]

\[
\Phi_{rs} = \begin{cases} 
\sum_{i=1}^{m} G_i^r \bar{x}_i^r + S_2 - R_r & \text{if } r = s, \\
\frac{1}{2} \sum_{i=1}^{m} (G_i^r \bar{x}_i^s + G_i^s \bar{x}_i^r) & \text{otherwise,}
\end{cases}
\]

\[
\Psi_{rs} = \frac{1}{2} \sum_{i=1}^{m} (F_i \varphi_{irs} + G_i^r \varphi_{is0}),
\]

\[
\Upsilon_{rst} = \frac{1}{2} \sum_{i=1}^{m} G_i^r \varphi_{irot},
\]

\[
\Omega_{rs} = Q + R_r.
\]

**Proof.** The requirement that \( \mathcal{E}_\delta \subseteq \mathcal{X}_\delta \) can be written as

\[
y^T \left[ F_0 + \sum_{i=1}^{m} F_i \left( \bar{x}_i^0 + \sum_{k=1}^{L} \delta_k \bar{x}_i^k + \sum_{j=1}^{m} \left( p_{ij0} + \sum_{k=1}^{L} p_{ijk} \delta_k \right) d_j \right) \right]
\]

\[
+ \sum_{h=1}^{L} \left( G_0^h + \sum_{i=1}^{m} G_i^h \left( \bar{x}_i^0 + \sum_{k=1}^{L} \delta_k \bar{x}_i^k + \sum_{j=1}^{m} \left( p_{ij0} + \sum_{k=1}^{L} p_{ijk} \delta_k \right) d_j \right) \right) \delta_h \right] y \geq 0,
\]

\[\forall y \in \mathbb{R}^n, d \in \mathcal{B}_m. \quad (5.19)\]

where \( p_{ijk} \) is the \((i, j)\) element of matrix \( P_k \).

Introduce variables \( z = (z_1, \ldots, z_m) \), \( u = (u_1, \ldots, u_L) \), and \( v = (v_{11}, \ldots, v_{LL}) \) with \( z_h, u_i, v_{jk} \in \mathbb{R}^n \), and the concatenated variable

\[
M = (S_1, S_2, Q, R_1, \ldots, R_L, T_1, \ldots, T_m, U_1, \ldots, U_L, V_{11}, \ldots, V_{LL}).
\]

Define the sets

\[
\mathcal{W} = \{ (y, z, u, v) \in \mathbb{R}^n \times (\mathbb{R}^n)^m \times (\mathbb{R}^n)^L \times (\mathbb{R}^n)^{L^2} | \exists d \in \mathcal{B}_m, \exists \delta \in \mathcal{B}_L, z_i = d_i y, u_j = \delta_j y, v_{ij} = \delta_i u_j \},
\]
\( \mathcal{L} = \{ M \mid S_1, S_2, Q, R_1, \ldots, R_L \text{ symmetric and positive definite}, \\
T_1, \ldots, T_m, U_1, \ldots, U_L, V_{11}, \ldots, V_{LL} \text{ skew-symmetric} \} \),

\[
\mathcal{Q}_1(P, \bar{x}) = \left\{ (y, z, u, v) \left| \begin{array}{l}
y^T \left( F_0 + \sum_{i=1}^m F_i \bar{x}_i^0 \right) y + \\
\sum_{j=1}^L \sum_{i=1}^m y^T F_i \bar{x}_i^j u_j + \sum_{j=1}^m y^T \sum_{j=1}^m F_ip_{ij0} z_j + \sum_{j=1}^m u_k^T \sum_{i=1}^m F_i p_{ijk} z_j + \\
\sum_{h=1}^L y^T \left( G_0^h + \sum_{i=1}^m G_i^h \bar{x}_i^0 \right) u_h + \sum_{h=1}^L \sum_{j=1}^m u_h^T \sum_{i=1}^m G_i^h \bar{x}_i^j u_j + \\
\sum_{h=1}^L \sum_{j=1}^m u_h^T \sum_{i=1}^m G_i^h p_{ij0} z_j + \sum_{j=1}^m \sum_{h=1}^L \sum_{k=1}^m z_j^T \sum_{i=1}^m G_i^h p_{ijk} v_{hk} \geq 0 \right\}
\]

and

\[
\mathcal{Q}_2(M) = \left\{ (y, z, u, v) \left| \begin{array}{l}
\sum_{i=1}^m z_i^T S_1 z_i - y^T S_1 y + \sum_{i=1}^L u_i^T S_2 u_i - \\
y^T S_2 y + \sum_{j=1}^L \left( \sum_{i=1}^L v_{ij}^T R_j v_{ij} - z_j^T R_j z_j \right) + \sum_{i=1}^L \sum_{j=1}^L v_{ij}^T Q v_{ij} - y^T Q y + \\
\sum_{i=1}^m y^T T_i z_i + \sum_{i=1}^L y^T U_i u_i + \sum_{i=1}^L \sum_{j=1}^L y^T V_{ij} v_{ij} \leq 0 \right\}
\]

With these new variables and sets defined, the proof is the same as the proof of Proposition 5.4, statements (5.11a)-(5.11d).

\( \square \)

**Remark 5.3.** Recall that for Proposition 5.4, the variables \( U_{ij} \) were not necessary for the proof, yet they can reduce the conservatism of the proposition in some cases. Similarly, new skew-symmetric matrix variables can be added to any of the zero blocks in the LMI (5.18) without affecting the proof. It is expected that the addition of these variables will reduce the conservatism of Proposition 5.7.

Attention should now be paid to choosing some “optimal” inscribed parameterized ellipsoid. Ultimately, the objective is to find an inscribed ellipsoid which provides the best possible approximation of the constraints, so it makes sense to treat this as a volume maximization problem. The volume of the parameterized ellipsoid can no longer
be described by the volume of a single ellipsoid, but does have a volume in a higher
dimensional space. This volume maximization problem can be posed as

\[
\max_{\delta \in \mathbb{R}^L} \int_{\delta \in \mathbb{R}^L} \det P(\delta) d\delta_L \cdots d\delta_1.
\]

Unfortunately, it is unlikely that this objective function can be expressed as a simple
\textit{maxdet} objective. A simple alternative is to only maximize the volume of the nominal
ellipsoid (when \( \delta = 0 \)), which leads to the maximization of the objective \( \det P_0 \). This
alternative is intuitively appealing, and seems to work well in practice. Example 5.6,
described next, uses this approach.

**Example 5.6.** Consider the following constraints parameterized in the scalar \( \delta \):

\[
(x_1 - 2\delta)^2 + 4(x_2 - 1)^2 \leq 4
\]

\[
x_1 + 2x_2 \leq 6
\]

\[
x_1 \geq 2
\]

\[
|\delta| \leq 1.
\]

Figure 5.3 shows the boundaries of these constraints for the values \( \delta = -1 \) and \( \delta = 1 \).
Notice that the intersection of these constraints is the single point \((0, 1)\), so the only
ellipsoid contained within the constraints for all values of \( \delta \) would be this point, with
zero volume. Instead, the goal is to find a maximum volume ellipsoid parameterized in
the variable \( \delta \). Applying the constraints (5.18) and maximizing the volume of \( \mathcal{E}_0 \) (the
determinant of \( P_0 \)) yields the parameterized ellipsoid

\[
\mathcal{E}_\delta = \left\{ \left( \begin{bmatrix} 1.3345 & -0.0196 \\ -0.0196 & 0.8286 \end{bmatrix} + \begin{bmatrix} 0.4879 & -0.0597 \\ -0.0597 & 0.0204 \end{bmatrix} \delta \right) d \right\} + \begin{bmatrix} 0.3583 \\ 0.9577 \end{bmatrix} + \begin{bmatrix} 1.5108 \\ -0.0248 \end{bmatrix} \delta \mid d \in \mathbb{R}^2 \right\}.
\]

Figure 5.3 shows the parameterized ellipsoid \( \mathcal{E}_\delta \) for \( \delta = -1 \) and \( \delta = 1 \).
5.4.3 Other Ellipsoidal Approximations

So far, it has been assumed that the maximum volume inscribed ellipsoid is the best possible ellipsoidal approximation of a set of constraints. This is the case if the objective vector \( c \) is allowed to point in any direction with equal likelihood. However, if \( c \) is restricted to a subspace, or if the average value of \( c \) has a nonzero component, then there are better alternatives to choosing the maximum volume ellipsoid. This section is broken into two parts which deal with these two possibilities: Volume Maximization of Projected Ellipsoids, and Projective Transformations.

**Volume Maximization of Projected Ellipsoids** Suppose that the objective vector of a linear objective function \( c^T x \) always lies in some subspace of \( \mathbb{R}^p \). In particular, let \( c \in \mathcal{A} \) where \( \mathcal{A} = \{ Am \mid m \in \mathbb{R}^q \} \) with \( A \in \mathbb{R}^{p \times q} \) and \( q < p \). Then the objective has the form \( m^T A^T x \). An excellent metric to gauge how well an ellipsoid approximates a set of constraints is the volume of the ellipsoid *projected* onto \( \mathbb{R}^q \) by means of the linear map \( A^T \). The projection of a \( p \)-dimensional ellipsoid \( \mathcal{E} = \{ Pz \mid \|z\| \leq 1 \} \) where \( P = P^T > 0 \)
5.4. ELLIPSOIDAL CONSTRAINT APPROXIMATION

is the $q$-dimensional ellipsoid

$$\tilde{E} \triangleq \{ A^T P z \mid \|z\| \leq 1 \} = \{ \tilde{P} w \mid \|w\| \leq 1 \}$$

where

$$\tilde{P} = (A^T P^2 A)^{1/2}.$$ 

The volume of $\tilde{E}$ is proportional to $\sqrt{\det(A^T P^2 A)}$, which, unlike $\det(P)$, is not necessarily concave over the positive definite variable $P$. This means that volume maximization of the projected ellipsoid is a nonconvex problem in general. Interestingly, when the constraints are a unit hypercube centered at the origin, the problem does in fact become convex, independent of $A$. This is the case for an important class of online optimization problems, the control allocation problem [Dur94, Enn98].

In control allocation, a redundant number of actuators are used to achieve a lower-dimensional control input, subject to saturation constraints on the actuators. The desired control input maps to an objective vector lying in some lower-dimensional subspace of the actuator space. In [OJMF99], a convex method was discovered for finding the maximum volume projected ellipsoid contained in the constraints of

$$\min_{\|x\|_\infty \leq 1} m^T A^T x$$

where $m \in \mathbb{R}^q$ and $x \in \mathbb{R}^p$. Due to the symmetry of the constraints $\|x\|_\infty \leq 1$ about the origin, the maximum volume projected ellipsoid has the form $Q E'$, where $E' = \{ P' z \mid \|z\| \leq 1 \}$ is an ellipsoid in $\mathbb{R}^q$, and $Q$ is a linear map from $\mathbb{R}^q$ to $\mathbb{R}^p$ with $A^T Q = I$.

The projected volume of $Q E'$ is proportional to $\sqrt{\det(A^T Q (P')^2 Q^T A)} = \det(P')$, which is concave in $P'$. It is shown in [OJMF99] that the volume maximization problem is equivalent to the semidefinite program

$$\max_{P \in \mathbb{R}^{q \times q}, R \in \mathbb{R}^{p \times q}} \det P$$

subject to

$$\begin{bmatrix} P & r_i^T \\ r_i & 1 \end{bmatrix} \succeq 0, \quad i = 1, \ldots, p,$$

$$A^T R = P,$$
where the rows of $R$ are $r_1, \ldots, r_p$. The matrix $Q$ is recovered by $Q = RP^{-1}$.

In spite of the fact that for this special case, the projected volume maximization problem is a convex program, in general the objective is nonconvex. The projected volume maximization problem can be posed as

$$\max \det A^TP^2A$$
subject to $E(P, \bar{x}) \subseteq \mathcal{X}$.

The constraints can be posed in LMI form (possibly conservatively) using Proposition 5.4. One possibility for maximizing the objective function is to linearize it and solve the problem as a sequence of semidefinite programs using a variation on Zoutendijk’s method (see [BSS93]). Given an initial feasible symmetric ellipsoid matrix $P_k$, an improving feasible direction $P_\Delta$ is generated by solving the problem

$$\max_{P_\Delta, \bar{x}} \text{Tr} \nabla f(P_k)P_\Delta$$
subject to $E(P_k + P_\Delta, \bar{x}) \subseteq \mathcal{X}$
$$\|P_\Delta\| \leq 1$$

where the gradient of $\det A^TP^2A$ at $P_k$ is $\nabla f(P_k) = A(A^TP^2A)^{-1}A^TP_k$. This technique yields an improving direction $P_\Delta$ if $P_k$ is not locally optimal, otherwise the maximum of this program is zero. A line search is then used along the improving direction to generate a new feasible point $P_{k+1} = P_k + \lambda P_\Delta$ with an improved objective. Although this method does not guarantee convergence to a locally optimal solution in general, this can be remedied with simple extensions such as the method of Topkis and Veinott [BSS93].

Example 5.7. The constraints

$$\|Bx\|_{\infty} \leq 1, \quad B = \frac{1}{2} \begin{bmatrix} 1 & 1 & -\sqrt{2} \\ -\sqrt{2} & -\sqrt{2} & 0 \\ 1 & 1 & \sqrt{2} \end{bmatrix}$$

represent the unit cube rotated by $45^\circ$ along the $x_1$ and $x_3$ axes. Suppose the objective is to find the inscribed ellipsoid whose volume projected on the $x_1$ and $x_3$ axes is maximized. The projected volume (area) of the unit sphere is $2\pi$. The maximum projected volume
is 2.97\pi, with an increase in area of nearly 50%. Figures 5.4–5.7 show several 3-d views of these ellipsoids.

**Projective Transformations** The maximum volume ellipsoid attempts to fill the volume of a constraint space \( \mathcal{X} \) in every direction. This is justified if the objective vector may point in any direction with equal likelihood. However, if the objective vector \( c \) has a bias towards a particular direction (e.g., \( c \in \{c_0 + \delta \mid \|\delta\| \leq 1\} \) where \( c_0 \) is a constant vector), then it makes more sense to weight the volume of \( \mathcal{X} \) in that particular direction. Projective transformations are essentially a method of amplifying the volume of a space in a certain direction.

A convex set \( \mathcal{X} \) in \( \mathbb{R}^n \) can be embedded in a convex cone in \( \mathbb{R}^{n+1} \) by

\[
\mathcal{K} = \{ (\lambda x, \lambda) \mid x \in \mathcal{X}, \lambda \geq 0 \}.
\]

Each ray in \( \mathcal{K} \) is equivalent to a point in \( \mathcal{X} \). The set \( \mathcal{X} \) is recovered by intersecting \( \mathcal{K} \) with the hyperplane \( \mathcal{H}_0 = \{(x, 1) \mid x \in \mathbb{R}^n \} \). Other sets equivalent to \( \mathcal{X} \) are found by intersecting \( \mathcal{K} \) with different hyperplanes. These intersections are considered to be projections of \( \mathcal{X} \). The volume of a space \( \mathcal{X} \) is weighted in a direction \( c \) by projecting each point \( x \in \mathcal{X} \) on a hyperplane \( t = c^T x + d \), resulting in a new point \( x_t \), with the volume in close proximity to this point also being multiplied by \( t \). The following lemma gives the specifics of these projections.

**Lemma 5.8.** Given a set \( \mathcal{X} \in \mathbb{R}^n \), define a hyperplane

\[
\mathcal{H} = \{ (x, t) \mid x \in \mathbb{R}^n, t = c^T x + d \}
\]

where \( d(1 - c^T x) > 0 \) for all \( x \in \mathcal{X} \). Let

\[
\mathcal{X}' = \{ x \mid (x, t) \in \mathcal{K} \cap \mathcal{H} \}.
\]

Then \( \mathcal{X} \sim \mathcal{X}' \) with the bijection \( f : \mathcal{X} \to \mathcal{X}' \) defined by

\[
f(x) = \frac{xd}{1 - c^T x}.
\]
Figure 5.4: Maximum volume ellipsoid.

Figure 5.5: Ellipsoid with maximum volume in $x_2-x_3$ axis.

Figure 5.6: View from $x_2-x_3$ axis, maximum volume ellipsoid.

Figure 5.7: View from $x_2-x_3$ axis, ellipsoid with maximum volume in $x_2-x_3$ axes.
Proof. Given \( x \in \mathcal{X} \), let \( t = c^T f(x) + d = d/(1 - c^T x) \). Then \((f(x), t) \in \mathcal{H}\). Observe that \( t > 0 \) and \( xt = f(x) \). Therefore \((f(x), t) \in \mathcal{K}\), thus \( f(x) \in \mathcal{X}'\).

Next, choose any \( x' \in \mathcal{X}' \). Then \((x', (c^T x' + d)) \in \mathcal{K}\) and \(\{(x', \lambda (c^T x' + d)) \mid \lambda \geq 0\}\) is a ray in \(\mathcal{K}\). It is required that \((c^T x' + d) > 0\). To see that this is the case, assume otherwise. From the construction of the cone \(\mathcal{K}\), this would mean that \((x', c^T x' + d) = (0, 0)\). However, \(d \neq 0\), which leads to a contradiction. Continuing with the proof, define the function \(g : \mathcal{X}' \to \mathcal{X}\) such that \(g(x') = x'/(c^T x' + d)\). Then \((g(x'), 1) \in \mathcal{K}\), and so \(g(x') \in \mathcal{X}\).

Finally, observe that \(g(f(x)) = x\) for all \(x \in \mathcal{X}\), and \(f(g(x')) = x'\) for all \(x' \in \mathcal{X}'\), thus \(f\) is bijective. \(\square\)

The condition \(d(1 - c^T x) > 0\) means that \(c^T x \neq 1\) and \(\text{sign}(1 - c^T x) = \text{sign}(d)\) for all \(x \in \mathcal{X}\). Intuitively, the condition that \(c^T x \neq 1\) for all \(x \in \mathcal{X}\) guarantees that no ray in \(\mathcal{K}\) will lie in the same subspace as \(\mathcal{H}\). The condition that given any \(x \in \mathcal{X}\), \(\text{sign}(1 - c^T x) = \text{sign}(d)\) guarantees that the ray in \(\mathcal{K}\) containing \(x\) will intersect the hyperplane. The hyperplane \(\mathcal{H}\) will therefore intersect every ray in \(\mathcal{K}\) at a unique point, effectively taking a cross-sectional “slice” of the cone. The bijective function in the above lemma says that all slices are equivalent to each other, and provides a way to transform from one to another. This transformation is known as a projective transformation, which is defined explicitly below.

Suppose that \(\mathcal{X}_1\) and \(\mathcal{X}_2\) are the projections of the set \(\mathcal{X}\) on the hyperplanes \(\mathcal{H}_1 = \{(x, t) \mid t = c_1^T x + d_1\}\) and \(\mathcal{H}_2 = \{(x, t) \mid t = c_2^T x + d_2\}\), where \(d_1(1 - c_1^T x) > 0\) and \(d_2(1 - c_2^T x) > 0\) for all \(x \in \mathcal{X}\). Then the projective transformation from \(\mathcal{X}_1\) to \(\mathcal{X}_2\) is denoted by \(\text{proj}_{1 \to 2} : \mathcal{X}_1 \to \mathcal{X}_2\), given by the expression

\[
\text{proj}_{1 \to 2}(x) = \frac{xd_2}{(c_1 - c_2)^T x + d_1}.
\]

From Lemma 5.8, it is not hard to see that this function is bijective.

One projective transformation which will become important later is the transformation of ellipsoids. The following proposition gives the transformation.

**Proposition 5.9.** Let \(\mathcal{X}\) be a set in \(\mathbb{R}^n\), and let \(\mathcal{X}_1\) and \(\mathcal{X}_2\) be the projections of \(\mathcal{X}\) on the hyperplanes \(\mathcal{H}_1 = \{(x, t) \mid t = c_1^T x + d_1\}\) and \(\mathcal{H}_2 = \{(x, t) \mid t = c_2^T x + d_2\}\), where
$d_1(1 - c_1^Tx) > 0$ and $d_2(1 - c_2^Tx) > 0$ for all $x \in \mathcal{X}$. Suppose an ellipsoid $\mathcal{E}_1 = \{P_1z + \bar{x}_1 \mid \|z\| \leq 1\}$ with $P > 0$ satisfies $\mathcal{E}_1 \subseteq \mathcal{X}_1$. Then the projective transformation of $\mathcal{E}_1$ onto the hyperplane $\mathcal{H}_2$ is the ellipsoid $\mathcal{E}_2 = \text{proj}_{1\rightarrow 2}(\mathcal{E}_1)$, defined by $\mathcal{E}_2 = \{P_2z + \bar{x}_2 \mid \|z\| \leq 1\}$, with

$$
\begin{align*}
\gamma &= \bar{x}_1 P_1^{-2} \bar{x}_1 - 1 \\
T &= d_2^2 P_1^{-2} + \tilde{d}(\tilde{c} \bar{x}_1^T P_1^{-2} + P_1^{-2} \bar{x}_1 \tilde{c}^T) + \gamma \tilde{c} \tilde{c}^T \\
\bar{x}_2 &= T^{-1}(\tilde{d} P_1^{-2} \bar{x}_1 + \gamma \tilde{c}) \\
P_2 &= \sqrt{\bar{x}_2^T T \bar{x}_2 - \gamma T^{-\frac{1}{2}}}
\end{align*}
$$

and $\tilde{c} = (c_1 - c_2)/d_2$, $\tilde{d} = d_1/d_2$.

**Proof.** The projective transformation is defined by $\text{proj}_{1\rightarrow 2} = x/(\tilde{c}^Tx + \tilde{d})$, with the range $\mathcal{X}_1$. The ellipsoid $\mathcal{E}_1$ is defined by the inequality $(x - \bar{x}_1)^T P_1^{-2} (x - \bar{x}_1) \leq 1$, which is the same as

$$
x^T P_1^{-2} x - 2 \bar{x}_1^T P_1^{-2} x + \gamma \leq 0
$$

Substituting $x = y \tilde{d}/(1 - \tilde{c}^Ty)$ and multiplying by $(1 - \tilde{c}^Ty)^2$ yields

$$
\tilde{d}^2 y^T P_1^{-2} y - 2 \tilde{d} \bar{x}_1^T P_1^{-2} y + 2 \tilde{d} \bar{x}_1 \tilde{c}^T P_1^{-2} y + \gamma (1 - 2 \tilde{c}^T y + y^T \tilde{c} \tilde{c}^T y) \leq 0,
$$

which is equivalent to

$$
y^T (\tilde{d}^2 P_1^{-2} + \tilde{d}(\tilde{c} \bar{x}_1^T P_1^{-2} + P_1^{-2} \bar{x}_1 \tilde{c}) + \gamma \tilde{c} \tilde{c}^T) y - 2y^T (\tilde{d} P_1^{-2} \bar{x}_1 + \gamma \tilde{c}) + \gamma \leq 0.
$$

It is necessary to see that the set defined by the above inequality is bounded. $\mathcal{E}_1 \subseteq \mathcal{X}_1$ is a compact set, and $\text{proj}_{1\rightarrow 2}$ is a continuous function on $\mathcal{X}_1$, therefore $\text{proj}_{1\rightarrow 2}(\mathcal{E}_1)$ will be a compact set and thus bounded.

Notice that $T$ appears in the quadratic of the above expression. Because $y$ is bounded in the inequality, $T$ must be positive definite and therefore invertible. This means that the expression for $\bar{x}_2$ given above in terms of $T^{-1}$ is defined. Substituting $T$ and $\bar{x}_2$ leads to

$$
y^T T y - 2y^T T \bar{x}_2 + \gamma \leq 0,
$$
which can be rearranged as
\[(y - \bar{x}_2)^T T(y - \bar{x}_2) \leq \bar{x}_2^T T \bar{x}_2 - \gamma.\]

It follows that \(\bar{x}_2^T T \bar{x}_2 - \gamma > 0\), otherwise \(E_2 = \emptyset\) or \(E_2 = \{0\}\), which would contradict the fact that \(\text{proj}_{1\rightarrow 2}\) is an injective map. This inequality can therefore be divided by \(\bar{x}_2^T T \bar{x}_2 - \gamma\) to arrive at
\[(y - \bar{x}_2)^T P_2^{-2}(y - \bar{x}_2) \leq 1,
\]
where \(P_2\) will be a positive definite matrix.

Example 5.8. Consider a cube \(C_0\) in \(\mathbb{R}^3\) defined by \(|x_i| \leq 1, i = 1, 2, 3\). If optimization over this cube includes an unbiased objective vector, the constraints might be appropriately approximated by the unit sphere, shown in Figure 5.8. Suppose an optimization problem over this cube is defined by a biased objective vector, with \(\max_{x \in C_0} (c_0 + \delta)^T x\) where \(c_0^T = [0 \ 0 \ 1]\), and \(\delta\) is bounded in magnitude but free to point in any direction. If \(\|\delta\|\) is known to be small, then the solution is likely to lie on the top facet of this cube. In this case, a more appropriate constraint approximation would favor the top facet. The volume of this cube can be weighted in the positive vertical direction by projecting the cube onto the hyperplane \(t = \alpha x_3 + 1\), where \(\alpha\) is chosen such that \(0 < \alpha < 1\). For this example, \(\alpha = 0.8\). Denote the projected cube \(C_1\) and the projection map \(\text{proj}_{0 \rightarrow 1} : C_0 \rightarrow C_1\). Let \(E_1\) be the maximum volume ellipsoid in \(C_1\). Figure 5.9 shows \(C_1\) and \(E_1\). Finally, let \(E_0\) be the projection of \(E_1\) back onto \(C_0\), i.e., \(E_0 = \text{proj}_{0 \rightarrow 1}^{-1}(E_1)\). As \(\alpha \rightarrow 1\), it can be shown that \(E_0\) converges to the maximum area circle on the top facet of the cube. Figure 5.10 illustrates \(C_0\) and \(E_0\) for \(\alpha = 0.8\).

5.5 Discussion

This chapter has discussed several alternatives to on-line optimization. The intended applications for these methods are those where on-line optimization is not an option due to limited available processing power, or applications where the certification requirements limit the complexity of the software. The concepts discussed here are not new. For
CHAPTER 5. SOLUTION APPROXIMATIONS

Figure 5.8: Cube with maximum volume ellipsoid.

Figure 5.9: Projected cube with maximum volume ellipsoid.

Figure 5.10: Cube with ellipsoid derived from projective transformation.
example, look-up tables have long been used for gain-scheduled controllers, and ellipsoidal constraint approximation is one of several techniques used for the control allocation problem. This chapter has sought to refine these optimization alternatives by enforcing robust feasibility and finding solutions which are close to optimality.

Of course, the curse of dimensionality limits these methods to very small problems. Look-up tables grow exponentially with parameter dimension. The same may be said for functional solution approximation, if the function is composed of piecewise linear or quadratic segments. Finally, ellipsoidal constraint approximation grows arbitrarily poor as the dimension increases (e.g., \( \lim_{n \rightarrow \infty} \frac{\text{vol}(S_n)}{\text{vol}(C_n)} = 0 \), where \( \text{vol}(S_n) \) is the volume of an \( n \) dimensional hypersphere with diameter 1 and \( \text{vol}(C_n) \) is the volume of an \( n \) dimensional unit hypercube). Nevertheless, these methods still remain useful for many real on-line applications in control.
Chapter 6

Application to Receding Horizon Control

Receding horizon control (RHC) [GPM89] is perhaps the most classic control application of on-line optimization, and thus deserves its own chapter in this thesis. Sometimes known as model predictive control, RHC has its roots in the chemical process industry, where it has been a subject of constant research since the late 1970s. RHC can roughly be described as a discrete-time control method in which an optimal constrained control problem is solved over a finite horizon at each time step. The strength of this method is that it allows constraints on the state and control to be integrated into the control design, as well as future command inputs. In its most general form, RHC can be applied to nonlinear systems (see [MM90, Nev97]). However, nonlinear systems usually lead to nonconvex problem formulations [May95], and finding a global minimum is generally not a tractable problem. Since this thesis is concerned with convex optimization problems, only linear RHC with a convex objective and convex constraints is considered.

Receding horizon control introduces feedback to a system by repeatedly resolving the open-loop problem at every time step. Closed-loop instability can therefore become a problem, even for open-loop stable systems. Past research has shown that global asymptotic stability is guaranteed by using an infinite horizon, or by using a finite horizon and constraining the terminal state to equal zero (e.g., see [RM93]). For constrained systems, an infinite horizon is impractical because it leads to an infinite-dimensional
optimization problem. Use of a terminal state constraint is also not realistic, since in practice the closed-loop will only approach zero asymptotically using this technique. This chapter avoids either of these stability techniques by choosing a sufficiently long horizon and using a Lyapunov stability argument in the same spirit as introduced in [PN97].

Along with stability, feasibility is also a critical issue for constrained linear systems. It is important that a receding horizon control law not allow the state to enter a region of the state space which cannot be brought to zero asymptotically. Such a state may lead the optimization problem to become infeasible. From an on-line optimization point of view, the optimization problem must be feasible at every time instance it is to be solved. This is especially important for systems with state constraints, which may conflict with control saturation constraints for some initial conditions. One practical solution is to use "soft" constraints for the state, which allow the constraints to be violated but penalize the magnitude of violation [ZM95]. This chapter does not consider the use of soft constraints, but instead considers the region of the state space for which the receding horizon control system is always feasible and asymptotically stable.

Before proceeding, it is instructive to give a small example which shows the benefits of using a receding horizon control law rather than a conventional linear control law which does not account for saturation.

**Example 6.1.** Suppose a command-following control system must be designed for the linear discrete-time system

\[
\begin{align*}
\mathbf{x}_{k+1} &= \begin{bmatrix} 1 & 2.1 \\ 0 & 1.1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} u_k \\
y_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k
\end{align*}
\]

which minimizes the cost function

\[
J = \sum_{k=0}^{\infty} \left[ \left( y_k - r_k \right)^2 + 0.01u_k^2 \right],
\]

where \( r_k \) is the reference trajectory. This is a linear-quadratic control problem, with optimal linear feedback \( u_k = 1.44r_k - [1.44 \ 3.59]x_k \). The closed loop is stable, with a fair amount of robustness (the LQ controller always has at least 6 dB of gain margin).
6.1. **Constrained Linear Systems**

Consider the linear discrete-time system

\[ x_{k+1} = Ax_k + Bu_k, \quad k = k_0, (k_0 + 1), \ldots \tag{6.2} \]

where \( x \in \mathbb{R}^{n_x} \), \( u \in \mathbb{R}^{n_u} \), and \( A, B \) are matrices of appropriate dimension. It is assumed that \([A, B]\) is controllable. The state and control are constrained by

\[ x_k \in C_x, \quad u_k \in C_u, \quad k = k_0, (k_0 + 1), \ldots \tag{6.3} \]

where \( C_x \subset \mathbb{R}^{n_x} \) and \( C_u \subset \mathbb{R}^{n_u} \) are polyhedral sets which contain the origin in their interiors.

However, since the open-loop system is unstable, the closed loop may become unstable if saturation limits exist on the control input. This is indeed the case if the output is commanded to \( y_k = 10 \) and the control is limited to \(|u_k| \leq 1\), as is seen by Figure 6.1.

On the other hand, receding horizon control takes control saturation limits into account, and can provide stability when other control systems fail. Figure 6.1 shows the response of a system controlled by a receding horizon controller with the same cost function (6.1) and a horizon length \( N = 10 \).

Figure 6.1: Response of LQ and RHC closed-loop system to command input \( r = 10 \).

Figure 6.2: Control inputs of LQ and RHC controllers.
Define \( v \) to be an infinite sequence of control inputs \( \{u_k\} \) for \( k = k_0, (k_0 + 1), \ldots \), and let \( v_N \) be a \( n_u \cdot N \) dimensional vector of control inputs \( [u_{k_0}^T \cdots u_{k_0+N-1}^T]^T \). The sets of all control inputs \( v \) and \( v_N \) which satisfy the dynamics (6.2) and constraints (6.3) for an initial state \( x_{k_0} \) are defined as the infinite and finite-dimensional sets \( U(x_{k_0}) \) and \( U_N(x_{k_0}) \) respectively.

For some states \( x \), the sets \( U(x) \) and \( U_N(x) \) may be empty. Notice that \( U_N(x) = \emptyset \) implies \( U(x) = \emptyset \), but not vice versa. This indicates that for certain initial conditions, the infinite-horizon problem may be infeasible even when the finite-horizon problem is feasible.

The admissible set of initial conditions is defined by Gutman and Cwikel in [GC86] to be the set of states for which a control sequence exists that asymptotically drives the state to zero and satisfies the state and control constraints (6.3). This set turns out to be the set of initial conditions which can be driven to zero in finite time. By defining \( \mathcal{X}_0 = \{0\} \) and \( \mathcal{X}_{k+1} = \{x \mid \exists u \in C_u, Ax + Bu \in \mathcal{X}_k \cap C_x\} \), the admissible set of initial conditions can be expressed as

\[ \mathcal{X}_\infty \triangleq \bigcup_{k=0}^{\infty} \mathcal{X}_k. \]

Note this definition does not imply that \( \mathcal{X}_\infty \) is contained in \( C_x \). Indeed, it will frequently be the case that \( \mathcal{X}_\infty \not\subset C_x \), as illustrated by Example 6.2.

Gutman and Cwikel [GC87] provide an algorithm for approximating \( \mathcal{X}_\infty \) with a polyhedral set. Their algorithm assumes that \( A \) is invertible, meaning there are no pure delays in the system. The approximation converges uniformly to \( \mathcal{X}_\infty \). Note that even though the sets \( \mathcal{X}_k \) are polyhedra for all \( k \), \( \mathcal{X}_\infty \) may not be, since it need not be a closed set. For example, if \( x_{k+1} = 2x_k + u_k \) with constraints \( |x_k| \leq 1 \) and \( |u_k| \leq 1 \), then \( \mathcal{X}_\infty \) is the open set \((-1, 1)\).

The infinite-horizon control problem is defined by minimizing the cost function

\[ J^*(x_{k_0}) \triangleq \min_{v \in U(x_{k_0})} \sum_{k=k_0}^{\infty} (x_k^T Q x_k + u_k^T R u_k) \quad (6.4) \]

where \( Q \) is symmetric and positive semidefinite, and \( R \) is symmetric and positive definite. The positive definiteness of \( R \) ensures that the cost function is strictly convex, which will
be important later. Given the convention that $J_\infty^*(x) = \infty$ if $\mathcal{U}_N(x) = \emptyset$, the infinite-horizon cost function has the property

$$J_\infty^*(x) < \infty \iff x \in \mathcal{X}_\infty.$$ 

Chmielewski and Manousiouthakis [CM96] prove that there exists a horizon length $N'$ such that for all $N \geq N'$, the following finite-horizon control problem is equivalent to the infinite-horizon problem:

$$J_N^*(x_{k_0}) \triangleq \min_{u_N \in \mathcal{U}_N(x_{k_0})} J_N(x_{k_0}, u_N)$$

(6.5)

where

$$J_N(x_{k_0}, u_N) \triangleq \sum_{k=k_0}^{k_0+N-1} (x_k^TQx_k + u_k^TRu_k) + x_{k_0+N}^T P x_{k_0+N}$$

(6.6)

and $P$ is the unique positive-definite solution to the algebraic Riccati equation

$$P = A^T(P - PB(B^T PB + R)^{-1}B^T P)A + Q.$$ 

(6.7)

The terminal cost $x_{k_0+N}^TPx_{k_0+N}$ in (6.6) is the optimal unconstrained cost at state $x_{k_0+N}$, which is achieved by applying the linear feedback law $u_k = -Kx_k$, where

$$K = (R + B^T PB)^{-1}B^T PA.$$ 

In general, $J_N^*(x) \leq J_\infty^*(x)$, but it holds that $J_N^*(x) = J_\infty^*(x)$ if the linear feedback law, starting at state $x_{k_0+N}$ produces a sequence of controls which satisfies (6.3). Thus, the state $x_{k_0+N}$ must belong to a set of states where the solutions to the unconstrained and constrained problems coincide. The maximal open set containing the origin exists which satisfies this property is denoted as $\mathcal{O}_\infty$. Gilbert and Tan [GT91] have proposed a method for finding $\mathcal{O}_\infty$. It is not difficult to construct a set contained in $\mathcal{O}_\infty$. Consider the region of unconstrained control

$$\mathcal{K} = \{ x \mid -Kx \in \mathcal{C}_u, \ x \in \mathcal{C}_x \},$$

which is a polyhedron that contains the origin. Let this polyhedron be defined by the linear inequality $Cx \leq d$. The parameter dependent level set

$$\mathcal{P}(q) = \{ x \mid x^TPx \leq q \}$$
will be invariant whenever $P(q) \subset \mathcal{K}$. Define

$$q^* = \max_{P(q) \subset \mathcal{K}} q.$$ 

The constraint is simply the condition that an ellipsoid is inscribed within a polyhedron. It can easily be verified that $q^*$ is computed by the expression

$$q^* = \min_i \left( \frac{d_i}{\|P^{-1/2}C_i\|} \right)^2, \quad (6.8)$$

where $(\cdot)_i$ indicates the $i$th row of a matrix. Since $\text{int} P(q^*)$ is invariant and unconstrained for the optimal linear feedback, it follows that

$$\text{int} P(q^*) \subseteq \mathcal{O}_\infty. \quad (6.9)$$

### 6.2 Well-Posedness of Receding Horizon Control

In order to solve the constrained problem in real time, it is important that the optimization problem be well-posed. In Chapter 4 as well as in [Ren95a], it was seen that the complexity of solving a convex program is related to the problem's distance to ill-posedness, where the complexity increases as the problem gets closer to infeasibility. It is desirable that the on-line algorithm never encounter an ill-posed problem instance. It should not be immediately obvious where the set of ill-posed problems lies (the boundary between feasible and infeasible problem instances), since $x \notin \mathcal{X}_\infty$ does not imply that the infinite horizon control problem is infeasible (e.g., consider an unstable plant with control constraints but no state constraints). The next proposition characterizes the set of all ill-posed problems in $\mathcal{X}_\infty$. Recall that a problem is said to be ill-posed if either its primal or dual constraints have an empty interior.

**Proposition 6.1.** Suppose $x \in \mathcal{X}_\infty$. Then $x \notin \text{int} \mathcal{X}_\infty$ if and only if there exists a horizon length $N$ such that the optimization problem defined by (6.5) is ill-posed.

**Proof.** Choose $x_0 \in \mathcal{X}_\infty \setminus \text{int} \mathcal{X}_\infty$. Since $x_0 \in \mathcal{X}_N$ for some $N$, there exists a control input $u_N \in \mathcal{U}_N(x_0)$ such that $x_N = 0$. Assume that the constraints $\mathcal{U}_N(x_0)$ are well-posed, i.e., $\text{int} \mathcal{U}_N(x_0) \neq \emptyset$. By continuity, for any open neighborhood of the origin $\mathcal{B}_N$ there
exists a control $\hat{v}_N \in \text{int}U_N(x_0)$ which brings the terminal state $\hat{x}_N \in B_N$. Again by continuity, there exists an open neighborhood $B_0$ of $x_0$ such that for all $\hat{x}_0 \in B_0$, it holds that $\hat{v}_N \in U_N(\hat{x}_0)$ and the terminal state $\hat{x}_N$ produced by $\hat{x}_0$ and control sequence $\hat{v}_N$ is in $B_N$. Since $x_0$ is on the boundary of $X_{\infty}$, a state $\hat{x}_0$ can be chosen such that $\hat{x}_0 \in B_0$ and $\hat{x}_0 \notin X_{\infty}$. This means that there exists a feasible control sequence $\hat{v}_N$ which brings this state within $B_N$. Since $B_N$ is arbitrary, it can be chosen inside $O_\infty$, the set of states where the optimal constrained and unconstrained solutions coincide. Thus, a feasible control sequence exists which asymptotically drives the state $\hat{x}_0$ to zero. This contradicts $\hat{x}_0 \notin X_{\infty}$, therefore $\text{int}U(x_0) = \emptyset$.

Next, choose $x_0 \in \text{int}X_{\infty}$. Then there exists a $\tilde{x}_0 \in X_{\infty}$ such that $x_0 = \lambda \tilde{x}_0$ and $0 \leq \lambda < 1$. Since $\tilde{x}_0 \in X_N$ for some $N$, there exists a $\tilde{v}_N$ which takes $\tilde{x}_0$ to the origin with $\tilde{u}_k \in C_u$, $\tilde{x}_k \in C_x$, $k = 0, \ldots, N - 1$. Therefore, $v = \lambda \tilde{v}$ takes $x_0$ to the origin with $u_k = \lambda \tilde{u}_k \in \text{int}C_u$, $x_k = \lambda \tilde{x}_k \in \text{int}C_x$, $k = 0, \ldots, N - 1$. Thus, $\text{int}U(x_0) \neq \emptyset$, which means that the primal constraints are well-posed.

For a problem to be considered well-posed, it must be shown that the dual constraints are also nonempty. This follows easily from the fact that the objective function of the quadratic program is strictly convex, which always results in a strictly feasible dual regardless of the feasibility of the primal constraints. $\square$

It has been shown that initial conditions on the boundary of $X_{\infty}$ result in an ill-posed problem instance for a sufficiently long horizon, and initial conditions in the interior of $X_{\infty}$ always result in a well-posed problem instance. As a corollary, the infinite-horizon control problem (6.4) is feasible and well-posed if and only if the initial condition is in the interior of $X_{\infty}$.

To use real-time optimization algorithms for the receding horizon control problem, it is critical that the optimization problems remain well-posed. To achieve this, all trajectories must remain in the interior of $X_{\infty}$. This is possible if an invariant set can be found in the interior of $X_{\infty}$. An invariant set is any set $\mathcal{W}$ for which $x_0 \in \mathcal{W}$ implies that the entire trajectory $x_k \in \mathcal{W}$, ($k = 1, 2, \ldots$). It is known that for some horizon length
CHAPTER 6. APPLICATION TO RECEIVING HORIZON CONTROL

\[ J_\mu \triangleq \{ x \mid J_\infty^*(x) \leq \mu \}, \]

become invariant sets for the finite-horizon control policy (see [PN00]). However, as is demonstrated by the next example, these level sets are not necessarily contained in the interior of \( \mathcal{X}_\infty \).

**Example 6.2.** Consider the discrete-time double integrator

\[
\begin{align*}
x_{k+1} &= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u_k \\
\end{align*}
\]

subject to the constraints

\[
\|x_k\| \leq 1, \quad |u_k| \leq 1/2
\]

and the cost function

\[
J_\infty(x_0) = \min_{k=0}^{\infty} \sum (x_k^T x_k + u_k^2).
\]

Figure 6.3 shows the level sets of \( J_\infty^*(x) \), where \( \mathcal{X}_\infty \) is a closed set, shown as the bold polygon on the figure. The value of \( J_\infty^*(x) \) ranges from 3.6 to 13.5 on the boundary of \( \mathcal{X}_\infty \), therefore not all the level sets are contained in the interior of \( \mathcal{X}_\infty \). Also, notice that \( \mathcal{X}_\infty \not\subset \mathcal{C}_x \).

### 6.3 Suboptimal RHC and Invariant Sets

The next important aspect to consider is how the accuracy of the solutions returned by the optimization algorithm affects the receding horizon control scheme. As a rule, interior-point algorithms do not return an optimal solution. Instead, they return suboptimal solutions which are within some tolerance of optimality. This will impact the performance of the receding horizon control policy, but more importantly, may affect stability.

To show the stability of suboptimal receding horizon control, it is necessary to show that a level set which satisfies \( \mathcal{J}_\mu \subset \text{int} \mathcal{X}_\infty \) is an invariant set for some finite-horizon,
6.3. SUBOPTIMAL RHC AND INVARIANT SETS

suboptimal receding horizon control policy. An $\epsilon$-optimal control sequence is one which satisfies

$$v_N^{\epsilon,\text{opt}}(x) \in \{v_N \mid J_N(x,v_N) \leq J_N^*(x) + \epsilon, \ v_N \in \mathcal{U}_N(x)\}.$$

The superscript is used to differentiate between optimal and suboptimal control sequences, i.e., $v_N^{\epsilon,\text{opt}}(x)$ is indicated as above, whereas $v_N^{\text{opt}}(x) = \arg\min J_N(x,v)$. In order to differentiate between the state produced by an optimal and suboptimal control sequence, the same superscripts are used for the state. The subsequent state is indicated as a function of the previous, using the convention:

$$y_{\infty}(x) = Ax + Bu_{\infty}(x)$$
$$y_N^{\text{opt}}(x) = Ax + Bu_N^{\text{opt}}(x)$$
$$y_N^{\epsilon,\text{opt}}(x) = Ax + Bu_N^{\epsilon,\text{opt}}(x)$$

where $u_{\infty}(x)$ is the first element of the optimal infinite-horizon control sequence, and $u_N^{\text{opt}}(x)$ and $u_N^{\epsilon,\text{opt}}(x)$ are the first elements in the control sequences $v_N^{\text{opt}}(x)$ and $v_N^{\epsilon,\text{opt}}(x)$.

To prove the existence of invariant sets for suboptimal receding horizon control, the following lemma is needed.

![Figure 6.3: Level sets of $J_N^*(x)$ for Example 6.2.](image-url)
Lemma 6.2. Given an $\epsilon$-optimal receding horizon control scheme,

$$
\|y_N^{opt}(x) - y_N^{c-opt}(x)\| \leq \left(\sigma_{max}(B)/\sigma_{min}(Q^{1/2}B)\right)\sqrt{\epsilon}
$$

for all $N$ and all $x \in X_\infty$.

Proof. The optimization problem to be solved from state $x_0$ has the form

$$
J_N^*(x_0) = \min_{V_N \in \mathcal{U}_N(x)} V_N^T G^T G V_N + g^T v_N,
$$

where

$$
G = \begin{bmatrix}
Q^{1/2}B & 0 & \cdots \\
Q^{1/2}AB & Q^{1/2}B \\
\vdots & \ddots & \ddots
\end{bmatrix}.
$$

Let $v_N^{opt}$ be the optimizer, and $c^T v_N \geq c^T v_N^{opt}$ the halfspace which contains the polyhedral constraints $\mathcal{U}_N(x)$ and supports the set $\{v_N \mid v_N^T H v_N + g^T v_N \leq J_N^*(x_0)\}$. The suboptimal control sequence $v^{c-opt}$ must lie in the set $\{v_N \mid v_N^T H v_N + g^T v_N \leq J_N^*(x_0) + \epsilon, c^T v_N \geq c^T v_N^{opt}\}$.

Consider the transformation of coordinates $\tilde{v}_N = G v$. Then

$$
\|\tilde{v}_N^{c-opt} - \tilde{v}_N^{opt}\| \leq \sqrt{\epsilon}.
$$

(6.10)

This can be seen by assuming (without loss of generality) that $g = 0$ and $\tilde{v}_N^{opt} = [r \ 0 \ \cdots \ 0]$, which can be achieved by a translation and rotation of coordinates. Then $\tilde{v}_N^{c-opt}$ must lie in the set $\{\tilde{v}_N \mid \tilde{v}_N^T \tilde{v}_N \leq r^2 + \epsilon, [1 \ 0 \ \cdots \ 0] \tilde{v} \geq r\}$, for which (6.10) is easy to verify.

The first $n_x$ elements of the vector $\tilde{v}_N$ is the vector $Q^{1/2}Bw_0$, therefore $\|Q^{1/2}B(u_0^{c-opt} - u_0^{opt})\| \leq \|\tilde{v}_N^{c-opt} - \tilde{v}_N^{opt}\|$. The distance between optimal and suboptimal control inputs is then bounded by

$$
\|u_0^{c-opt} - u_0^{opt}\| \leq (1/\sigma_{min}(Q^{1/2}B))\|\tilde{v}_N^{c-opt} - \tilde{v}_N^{opt}\| \leq \sqrt{\epsilon}/\sigma_{min}(Q^{1/2}B).
$$

Finally, the distance between the states $y_N^{opt}(x_0) = A x_0 + B u_0^{opt}$ and $y_N^{c-opt}(x_0) = A x_0 + B u_0^{c-opt}$ is bounded by $\|y_N^{c-opt}(x_0) - y_N^{opt}(x_0)\| \leq \sqrt{\epsilon} \sigma_{max}(B)/\sigma_{min}(Q^{1/2}B)$. \qed
6.4. STABILITY OF SUBOPTIMAL RHC

Proposition 6.3. Suppose $J_\mu \subset \text{int} \mathcal{X}_\infty$. Then there exists a horizon length $N'$ and tolerance $\epsilon$ such that $J_\mu$ is an invariant set for all finite-horizon suboptimal receding horizon control schemes with horizon length $N \geq N'$ and tolerance $\epsilon$.

Proof. $J^*_\infty(x)$ is a convex function on the compact set $J_\mu$ with $J^*_\infty(x) = \mu$ on the boundary of $J_\mu$. Since $J^*_\infty(y_\infty(x)) \leq J^*_\infty(x) - x^TQx \leq \mu - \eta$ with $\eta = \min_{J_\infty(x)=\mu} x^TQx > 0$, then $y_\infty(x)$ is confined to the level set $J_{\mu-\eta}$. Hence,

$$y_\infty(x) + \delta \in J_\mu \text{ for all } x \in J_\mu, \|\delta\| \leq \eta.$$

Since $u_N^{\text{opt}}(x)$ is a continuous function of $x$ (see [FI90]) which converges pointwise to the continuous function $u_\infty^{\text{opt}}(x)$ (see [CM96]) on the compact set $J_\mu$, it therefore converges uniformly. Hence, given $\beta > 0$, there exists an $N'$ such that

$$\|y_\infty(x) - y_N^{\text{opt}}(x)\| = \|B(u_\infty(x) - u_N^{\text{opt}}(x))\| \leq \beta \text{ for all } N \geq N'.$$

By Lemma 6.2, there exists an $\alpha$ such that $\|y_N^{\text{opt}}(x) - y_N^{e^{\text{opt}}}(x)\| \leq \alpha \sqrt{\epsilon}$ for all $x \in \mathcal{X}_\infty$. Choose $\beta$ and $\epsilon$ such that $\beta + \alpha \sqrt{\epsilon} \leq \eta$. Then

$$\|y_\infty(x) - y_N^{e^{\text{opt}}}(x)\| \leq \|y_\infty(x) - y_N^{\text{opt}}(x)\| + \|y_N^{\text{opt}}(x) - y_N^{e^{\text{opt}}}(x)\| \leq \beta + \alpha \sqrt{\epsilon} \leq \eta$$

from which follows $y_N^{e^{\text{opt}}}(x) \in J_\mu$ for all $x \in J_\mu$ and $N \geq N'$.

Proposition 6.3 demonstrates the existence of an $\epsilon$-optimal receding horizon control scheme such that a level set $J_\mu \subset \text{int} \mathcal{X}_\infty$ is invariant. The trajectories will therefore always remain feasible, and the optimization problems well-posed. The next important topic to examine is one of stability.

6.4 Stability of Suboptimal RHC

Traditionally, stability analysis is concerned with asymptotic stability, which requires the existence of arbitrarily small invariant sets containing neighborhoods of the origin, and that trajectories of a system satisfy $\lim_{k \to \infty} \|x_k\| = 0$. Stability can be shown for a
nonlinear system by the existence of a Lyapunov function. In discrete time, this function is defined as any continuously differentiable function $V : \mathbb{R}^n \rightarrow \mathbb{R}$ satisfying

\[
V(0) = 0 \\
V(x) > 0 \text{ for } x \neq 0
\]

and

\[
V(x_{k+1}) < V(x_k)
\]

for any trajectory of the system $\{x_k\}$.

A Lyapunov argument was used to show the stability of receding horizon control in [PN97]. Although they based their proof on the assumption that the terminal weight in (6.6) is equal to $x^T_{k_0+N}Qx_{k_0+N}$ rather than $x^T_{k_0+N}Px_{k_0+N}$ where $P$ is the solution to the Riccati equation (6.7), their proof can be generalized for the terminal weight $x^T_{k_0+N}Px_{k_0+N}$. Their results are summed up by the following theorem.

**Theorem 6.4.** For any level set $\mathcal{J}_\mu$, there exists a horizon length $N'$ such that for each $N \geq N'$ the optimal finite-horizon controller is asymptotically stable on $\mathcal{J}_\mu$, with $J_N^*(x)$ acting as a Lyapunov function satisfying

\[
J_N^*(y_N^{\text{opt}}(x)) \leq \gamma_N J_N^*(x)
\]

for some $\gamma_N < 1$.

Unfortunately, asymptotic stability is not possible for an $\epsilon$-optimal controller with $\epsilon > 0$, since the trajectory cannot be guaranteed to converge exactly to the origin. To see this, notice that for an $\epsilon$-optimal controller there cannot be an invariant set contained in the open set $\{x \mid J_N^*(x) < \epsilon\}$. A more relaxed notion of stability is needed here, given by the following definition. Later, it will be seen that asymptotic stability can be achieved if the receding horizon controller is appended with a linear feedback controller.

**Definition.** Let $\mathcal{W}$ and $\Omega \subset \mathcal{W}$ be invariant sets. Suppose that any trajectory $\{x_k\}$ of a system originating in $\mathcal{W}$ converges to some limit set contained in $\Omega$. Then the system is considered to be $\Omega$-stable on $\mathcal{W}$.
The following proposition states a condition for \( \Omega \)-stability.

**Proposition 6.5.** For every level set \( \mathcal{J}_\mu \subset \text{int} \mathcal{X}_\infty \), there exists a horizon length \( N \) and an accuracy \( \bar{\epsilon} \) such that the \( \epsilon \)-optimal receding horizon control law is \( \Omega(\epsilon) \)-stable on \( \mathcal{J}_\mu \) for all \( 0 \leq \epsilon \leq \bar{\epsilon} \), where

\[
\Omega(\epsilon) = \left\{ x \mid J^*_N(x) \leq \left( \frac{16c_1^2}{(1 - \gamma_N)^2} + \frac{4c_2}{1 - \gamma_N} \right) \epsilon \right\},
\]

and \( \gamma_N, c_1, c_2 \) are constants defined by Theorem 6.4 and Lemma 6.6, given following this proposition.

**Lemma 6.6.** Suppose \( \mathcal{J}_\mu \) is invariant for an \( \epsilon \)-optimal receding horizon control law with horizon length \( N \). Then there exist constants \( c_1 \) and \( c_2 \) such that

\[
|J^*_N(y_{N, \text{opt}}(x)) - J^*_N(y_{N, \text{opt}}(x'))| \leq c_1 \sqrt{J^*_N(x) \epsilon} + c_2 \epsilon
\]

for all \( x \in \mathcal{J}_\mu \).

**Proof of Lemma 6.6.** Since \( J^*_N(x) \) is the optimal value of a parametric quadratic program, it is a convex, piecewise quadratic function of \( x \) (see [DBP+99]) on the compact domain \( \mathcal{W} \). From this as well as the fact that \( J^*_N(0) = 0 \) and \( \nabla J^*_N(0) = 0 \), it is possible to construct a quadratic function \( \frac{1}{2} x^T G x \) with \( G > 0 \) such that the norm of the gradient \( \|Gx\| \geq \|\nabla J^*_N(x)\| \) for all \( x \in \mathcal{J}_\mu \). Thus, given \( x, x' \in \mathcal{J}_\mu \), the cost difference is bounded by

\[
|J^*_N(x') - J^*_N(x)| \leq \max\{\|\nabla J^*_N(x)\|, \|\nabla J^*_N(x')\|\} \cdot \|x' - x\|
\]

\[
\leq \sigma_{\max}(G) \cdot \max\{\|x\|, \|x'\|\} \cdot \|x' - x\|.
\]

Given states \( y_{N, \text{opt}}(x) \) and \( y_{N, \text{opt}}(x') \), Lemma 6.2 guarantees the existence of an \( \alpha \) such that

\[
\max\{\|y_{N, \text{opt}}(x), y_{N, \text{opt}}(x')\|\} \leq \|y_{N, \text{opt}}(x)\| + \alpha \sqrt{\epsilon}.
\]

Therefore,

\[
|J^*_N(x_{k+1, \text{opt}}) - J^*_N(x_{k+1, \text{opt}})| \leq \sigma_{\max}(2G)\|x_{k+1, \text{opt}}\|\alpha \sqrt{\epsilon} + \sigma_{\max}(2G)\alpha^2 \epsilon.
\]
The unconstrained finite-horizon optimal cost is a quadratic function $x^TPx \leq J_N^*(x)$ with $P \succ 0$ (see [Ber95]). The magnitude of $y_N^*(x)$ is bounded by

$$||y_N^*(x)|| \leq \frac{\sqrt{x^TPx}}{\sigma_{\min}(P^{1/2})} \leq \frac{\sqrt{J_N^*(y_N^*(x))}}{\sigma_{\min}(P^{1/2})} \leq \frac{\gamma_N}{\sigma_{\min}(P^{1/2})} \sqrt{J_N^*(x)}.$$

Combining this bound with the previous bound on the cost difference results in

$$|J_N^*(y_N^{c,\text{opt}}(x)) - J_N^*(y_N^*(x))| \leq \frac{\alpha \gamma_N \sigma_{\max}(G)}{\sigma_{\min}(P^{1/2})} \sqrt{J_N^*(x)} \epsilon + \sigma_{\max}(G)\alpha^2 \epsilon,$$

leading to the obvious choice

$$c_1 = \frac{\alpha \gamma_N \sigma_{\max}(G)}{\sigma_{\min}(P^{1/2})} \quad c_2 = \sigma_{\max}(G)\alpha^2.$$

\[\square\]

Proof of Proposition 6.5. Choose $x \in J_\epsilon \setminus \Omega(\epsilon)$. Then

$$J_N^*(x) > \left( \frac{16c_1^2}{(1 - \gamma_N)^2} + \frac{4c_2}{1 - \gamma_N} \right) \epsilon.$$

This implies that

$$(1 - \gamma_N)J_N^*(x) > 4 \left( \frac{4c_1^2}{1 - \gamma_N} + c_2 \right) \epsilon$$

$$> \left( \frac{2c_1}{\sqrt{1 - \gamma_N}} + \sqrt{\frac{4c_1^2}{1 - \gamma_N} + c_2} \right)^2 \epsilon$$

from which it is seen that

$$\sqrt{(1 - \gamma_N)J_N^*(x)} - \frac{2c_1 \sqrt{\epsilon}}{\sqrt{1 - \gamma_N}} > \sqrt{\left( \frac{4c_1^2}{1 - \gamma_N} + c_2 \right) \epsilon}.$$

Squaring both sides yields

$$(1 - \gamma_N)J_N^*(x) - c_1 \sqrt{J_N^*(x_k)\epsilon} + \frac{4c_1^2\epsilon}{1 - \gamma_N} > \left( \frac{4c_1^2}{1 - \gamma_N} + c_2 \right) \epsilon.$$

Hence,

$$(1 - \gamma_N)J_N^*(x) > c_1 \sqrt{J_N^*(x)\epsilon} + c_2 \epsilon.$$
Combining this result with Lemma 6.6 shows that

\[ J_N^*(y_N^{\text{opt}}(x)) \leq J_N^*(y_N^{\text{opt}}(x)) + c_1 \sqrt{J_N^*(x)} \varepsilon + c_2 \varepsilon \]
\[ < J_N^*(y_N^{\text{opt}}(x)) - \gamma_N J_N^*(x) + J_N^*(x) \]
\[ < J_N^*(x), \]

therefore \( J_N^*(x) \) is a Lyapunov function for the \( \epsilon \)-optimal receding horizon control law on the set \( \mathcal{J}_\mu \setminus \Omega(\epsilon) \). By a similar argument, it can be shown that \( \Omega(\epsilon) \) is an invariant set. Thus, trajectories originating in \( \mathcal{J}_\mu \setminus \Omega(\epsilon) \) converge to some limit set in \( \Omega(\epsilon) \).

\[ \square \]

### 6.4.1 Asymptotic Stability

So far, it has only been guaranteed that the trajectories will converge toward a compact invariant set \( \mathcal{O} \). In some cases, this performance may be adequate. However, if asymptotic stability is required of the system, then the approximate receding horizon controller is clearly not enough. To achieve asymptotic stability, the receding horizon controller can be augmented with the optimal linear feedback law when the state enters a region of unconstrained optimal control. This is much in the same spirit as the dual-mode receding horizon control method proposed by Michalska and Mayne [MM93].

In order to satisfy asymptotic stability of the receding horizon controller, the trajectories must enter \( \mathcal{O}_\infty \) in finite time. This is the case if \( \epsilon \) is chosen such that \( \Omega(\epsilon) \subset \mathcal{O}_\infty \). Suppose \( \Omega(\epsilon) = \{ x \mid J_N^*(x) \leq L\epsilon \} \) where \( L \) is a constant. Then \( \Omega(\epsilon) \subset \mathcal{P}(L\epsilon) \). Noting that \( \text{int} \mathcal{P}(q^*) \subset \mathcal{O}_\infty \) from (6.9), then \( \Omega(\epsilon) \subset \mathcal{O}_\infty \) if \( \epsilon < q^*/L \).

### 6.4.2 Disturbance Rejection

Up to this point, it has been assumed that the open-loop system is completely deterministic. Of course, real systems suffer from unknown disturbances, i.e., \( x_{k+1} = Ax_k + Bu_k + B_w w_k \) where \( w_k \) is unknown but may have known bounds or stochastic properties. Unknown disturbances generally prevent asymptotic stability, or in some cases can even drive the system unstable. For systems with disturbance inputs, it is more appropriate to deal with input/output stability, which bounds the output due to a bounded input.
CHAPTER 6. APPLICATION TO RECEDING HORIZON CONTROL

Constrained systems with disturbances face additional challenges. In general, the disturbance must be restricted to a bounded set, e.g., \( \|w\| \leq W \), since an unbounded signal can force the state to leave the admissible set. One immediate consequence of a disturbance signal is that the admissible set \( \mathcal{X}_\infty \) may get smaller, since this set must take into account the worst-case disturbance. Also, the set \( \mathcal{O}_\infty \) (i.e., the set of states for which the optimal linear feedback law is unconstrained) gets smaller, or may even be empty. A detailed analysis of \( \mathcal{O}_\infty \) under disturbances is given in [GK95].

A constrained system which is input/output stable must keep trajectories inside \( \mathcal{X}_\infty \). Receding horizon control techniques which guarantee that trajectories will remain in \( \mathcal{X}_\infty \) are usually based on computationally demanding min-max formulations and often are extremely conservative [LY97]. This is because these schemes must consider the worst-case open loop behavior due to a disturbance input, and choose the control input accordingly. As an alternative to min-max, the open loop can first be closed with a linear controller which will compensate for the disturbance, and then a receding horizon controller applied to the closed loop [Bem98]. This scheme is less computationally demanding than a min-max approach, since it can deal with the disturbance in a closed-loop setting.

Suppose a constrained system with a bounded input is input/output stable when the loop is closed by a receding horizon control law. The state is constrained to converge to some bounded limit set, which is conceptually the same as the concept of \( \Omega \)-stability defined previously. Let a disturbance input \( d \) on the system be defined by

\[
x_{k+1} = Ax_k + Bu_k + Bd_k.
\]

Assuming that the disturbance is bounded, i.e., \( \|d_k\| \leq 1 \), then the difference between the subsequent state with disturbance \( y(x) = Ax + Bu + Bd \) and without disturbance \( \hat{y}(x) = Ax + Bu \) is given by \( \|y(x) - \hat{y}(x)\| \leq \sigma_{\text{max}}(B_d) \). This is analogous to the result given by Lemma 6.2. Therefore, the same machinery that was developed in the previous section for considering the \( \Omega \)-stability of suboptimal receding horizon control can also be used to consider disturbances. Since the proof of \( \Omega \)-stability for the case with disturbances is nearly identical to that presented by Proposition 6.5, it is not repeated.
6.5 UAV Example

An unmanned air vehicle (UAV) example is now presented to illustrate the computational certification of a realistic receding horizon control application. Figure 6.4 shows the schematic of a small reconnaissance UAV currently under development at the Charles Stark Draper Laboratory. A preliminary design of this vehicle is reported in [Sco98]. The UAV is designed to fit inside the cargo chamber of a 155-mm ballistic projectile, in which it is to be delivered to a target surveillance area. To achieve this mission, the vehicle must survive a 16,000-g gun-launch acceleration, and the external surfaces must fold against the fuselage, allowing the vehicle to be packaged inside the artillery shell. These design requirements dictate severe limitations on the vehicle aerodynamics. One consequence is a restriction in the size of the control surfaces, which means limited control authority and frequent actuator saturation. If linear feedback is to be used, the gains must be decreased to avoid saturation and possible closed-loop instability, resulting in performance reduction. An alternative design approach is to use on-line optimization, which can directly compensate for the effect of actuator saturation.

The linearized equations of motion for the lateral dynamics of an aircraft are fairly
standard (e.g., see Chapter 11 in [PH49]). These dynamics are given by

\[ mV(\dot{\beta} + r) = qS(Y_\beta \beta + mg\phi + Y_\delta \delta_e) \]

\[ I_{zz} \ddot{\phi} + I_{xz} \ddot{\theta} = qSb \left[ C_{n_\beta} \beta + \left( \frac{b}{2V} \right) C_{n_r} r + \left( \frac{b}{2V} \right) C_{n_p} p + C_{n_\delta} \delta_r \right] \]

\[ I_{xx} \ddot{\phi} + I_{xz} \ddot{\theta} = qSb \left[ C_{\ell_\beta} \beta + \left( \frac{b}{2V} \right) C_{\ell_r} r + \left( \frac{b}{2V} \right) C_{\ell_p} p + C_{\ell_\delta} \delta_h \right] \]

\[ \dot{\psi} = r \]
\[ \dot{\phi} = p \]

where

\[ \beta = \text{sideslip angle}, \quad \psi = \text{yaw angle}, \quad \phi = \text{roll angle}, \]
\[ r = \text{yaw angular rate}, \quad p = \text{roll angular rate}, \]
\[ \delta_r = \text{rudder deflection}, \quad \delta_h = \text{differential horizontal tail deflection} \]
\[ V = \text{relative wind velocity (ft/sec)} \]

The units of all angles in this model are in radians. The control surfaces are subject to saturation limits of ±25° = ±0.436 rad.

The physical vehicle is defined by the parameters

\[ m = \text{vehicle mass} = 0.238 \text{ slugs} \]
\[ S = \text{wing area} = 0.886 \text{ ft}^2 \]
\[ b = \text{wing span} = 4.16 \text{ ft} \]
\[ I_{xx} = \text{inertia about roll axis} = 0.0297 \text{ slug-ft}^2 \]
\[ I_{zz} = \text{inertia about yaw axis} = 0.0682 \text{ slug-ft}^2 \]
\[ I_{xz} = \text{cross moment of inertia} = -0.0027 \text{ slug-ft}^2 \]
\[ g = \text{gravity} = 32.2 \text{ ft/sec}^2. \]

The dynamic pressure is \( q = \frac{1}{2} \rho V^2 \), where \( \rho = 0.002377 \text{ slug/ft}^3 \) is the atmospheric density at sea level. The aerodynamic coefficients were determined by wind tunnel testing,
and are given by

\[ Y_\beta = 0.716, \quad Y_{\delta_h} = 0.400 \]
\[ C_{n,\beta} = 0.6, \quad C_{n, r} = -0.082 \]
\[ C_{n, p} = -0.1C_L, \quad C_{n, \delta_r} = 0.103 \]
\[ C_{\ell, \beta} = 0.0, \quad C_{\ell, r} = 0.25C_L \]
\[ C_{\ell, p} = -0.720, \quad C_{\ell, \delta_h} = 0.0566 \]

where the coefficient of lift is approximated as \( C_L = mg/qS \).

This model is valid for small angle perturbations in roll and sideslip, in the velocity range 70–90 ft/s. Numerically, the state-space model is

\[
\frac{d}{dt} \begin{bmatrix} \beta \\ r \\ p \\ \psi \\ \phi \end{bmatrix} = \begin{bmatrix} 0.00316V & -1 & 0 & 0 & 0.0338V \\
0.0386V^2 & -0.011V + 22.23/V & -97.64/V - 0.00877V & 0 & 0 \\
0.00351V^2 & -0.0010V + 561.6/V & -0.221V - 8.877/V & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ p \\ \psi \\ \phi \end{bmatrix}
\]

\[
+ \begin{bmatrix} 0.00177V & 0 \\
0.00663V^2 & 0.00033V^2 \\
0.00060V^2 & 0.00837V^2 \\
0 & 0 \\
0 & 0 \end{bmatrix} \begin{bmatrix} \delta_r \\ \delta_h \end{bmatrix}.
\]

From this point forward, the discrete time dynamics of this model are considered, sampled at 0.1 seconds using a zero-order hold. The velocity is given as \( V = 80 \) ft/s. The dynamics are denoted as

\[
x_{k+1} = Ax_k + Bu_k
\]

where \( x = [\beta \ r \ p \ \psi \ \phi]^T \) and \( u = [\delta_r \ \delta_h]^T \).

The objective is to control the velocity angle, which is \( \psi + \beta \). Sideslip is regulated to
zero so the yaw angle matches the velocity vector. The cost function is defined by

\[ J_N(x_0, v_N) = \sum_{k=0}^{N-1} (100(\dot{\psi}_k + \beta_k)^2 + 25\beta_k^2 + 0.1(\delta_{\psi,k} + \delta_{\beta,k}^2)) + x_N^T P x_N \]

\[ = \frac{1}{2} v^T G^T G v + g(x_0)^T v + c(x_0) \]

where

\[ G = (Q^T Q + 0.1 I)^{1/2} \]

\[ g(x_0)^T = x_0^T \begin{bmatrix} A^T C^T & (A^T)^2 C^T & \cdots & (A^T)^N P^{1/2} \end{bmatrix} Q \]

\[ Q = \begin{bmatrix} C B & 0 & \cdots & \cdots & 0 \\
C A B & C B & \ddots & \ddots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
C A^{N-2} B & C A^{N-3} B & \cdots & C B & 0 \\
P^{1/2} A^{N-1} B & P^{1/2} A^{N-2} B & \cdots & P^{1/2} A B & P^{1/2} B \end{bmatrix} \]

\[ C = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 10 & 0 \\
0 & 0 & 0 & 0 & 1 \end{bmatrix} \]

The parameter dependent scalar \( c(x_0) \) does not affect the optimization problem, so it is not derived here. The terminal weight \( P \) is the solution to the discrete-time Riccati equation.

Figures 6.5–6.9 illustrate the control response of the UAV to a 45° commanded input. These figures compare a receding horizon controller with horizon length \( N = 30 \) with the linear quadratic regulator derived using the same cost function. In this case, the receding horizon controller is able to properly accommodate actuator saturation, while instability is induced by the LQR controller. Of course, it is possible to regain stability for the LQR controller by reducing the gains, resulting in some performance reduction. This simulation is only meant to show one possible pitfall of not considering saturation, and is in no way meant to claim the superiority of receding horizon control over LQR.
Figure 6.5: Heading response \((\psi + \beta)\) to 45\(^\circ\) commanded input.

Figure 6.6: Roll response.

Figure 6.7: Rudder input.

Figure 6.8: Differential horizontal stabilizer input.

Figure 6.9: Sideslip response.
6.5.1 Admissible Set of Initial Conditions

Since the state is unconstrained for this problem, the only property which will restrict the admissible set of initial conditions is instability of the open loop. The general problem of finding the admissible set for an unstable linear time-invariant system with input saturation is treated in [HQL97]. For the UAV dynamics, there is only a single unstable mode at eigenvalue $\lambda = 1.1073$, so finding the admissible set is rather straightforward.

To find the admissible set, transform the state into modal canonical form $z = Tx$, and partition the state-space system into semi-stable (might include poles with $|\lambda| = 1$) and anti-stable parts:

$$
\begin{bmatrix}
    z_s \\
    z_a
\end{bmatrix}
_{k+1} =
\begin{bmatrix}
    T_s A T_s^{-1} & 0 \\
    0 & \lambda_a
\end{bmatrix}
\begin{bmatrix}
    z_s \\
    z_a
\end{bmatrix}
_k +
\begin{bmatrix}
    T_s B \\
    T_a B
\end{bmatrix} u
$$

where $\lambda_a$ is the unstable pole. Then the system is controllable for all states $z$ belonging to the set

$$
\{ z | \exists u \in U, |\lambda_a z_a + T_a B u| < |z_a| \}.
$$

Assuming $U$ is symmetric about the origin, this is satisfied whenever

$$
|z_a| < \max_{u \in U} \frac{T_a B u}{\lambda_a - 1}.
$$

This in turn translates to a constraint on the state $x$. For the UAV, the admissible set is defined by the region

$$
|0.298\beta - 0.00291r + 0.0422p + 0.790\phi| < 1.
$$

Of course, this boundary between controllable and uncontrollable states probably does not exist in the actual aircraft. The instability in this model is directly attributed to the lateral acceleration dynamics $mV(\dot{\beta} + r) = \ldots$, sometimes referred to as the spiral mode. Since this acceleration really depends linearly on $\sin \phi$ rather than $\phi$, it is unlikely that the real vehicle would become uncontrollable for large angles.

In spite of the fact that the admissible region may be fictitious for the real system, it does give a boundary for which the linear system can be analyzed. Most likely, this linear
model would only be used in some bounded region containing the origin, and a different model linearized about a different trim point would be used once the state passed out of this region. Keeping this in mind, the following region is selected for analysis of the optimization algorithm, which is contained within the admissible set of the linear system:

\[
|\beta| \leq \frac{1}{5}\text{ rad, } |r| \leq 2\text{ rad/s, } |p| \leq 2\text{ rad/s, } \\
|\psi| \leq \pi\text{ rad, } |\phi| \leq 1\text{ rad.}
\]

### 6.5.2 \(\Omega\)-Stability

In Section 6.4, a detailed stability analysis of suboptimal receding horizon control was given. While the theory gives sufficient conditions for stability, the reality is that these conditions are difficult to check. For example, the condition given in Proposition 6.5 depends on knowing the guaranteed fractional decrease in cost \(\gamma_N\). Computation of a valid \(\gamma_N\) requires finding a global optimum of a nonconvex problem, and it is readily admitted in [PN97] that finding this \(\gamma_N\) is impractical for most problems.

When a detailed analysis proves intractable, the obvious fallback is to examine the system behavior in simulation. Proposition 6.5 tells us that \(\Omega(\epsilon) \to \{0\}\) as \(\epsilon \to 0\), so the strategy for choosing \(\epsilon\) is to determine how small \(\Omega(\epsilon)\) should be, and simulate with different values of \(\epsilon\) until \(\Omega(\epsilon)\) meets that requirement. Figure 6.10 shows a simulation of heading vs. roll for \(\epsilon = 0.0005\). In this plane, valid sets for \(\Omega(\epsilon)\) can be inferred by examining the level sets for which the trajectory enters but never leaves. Several level sets of \(J^*(x)\) are also superimposed for comparison.

The goal specified for this system is for the receding horizon controller to steer the trajectory into the region of unconstrained optimal control \(O_\infty\), after which the optimal linear control law can be implemented. As noted previously, this condition is met if \(\Omega(\epsilon) \subseteq P(q^*)\), where \(P(q) = \{x \mid x^TPx \leq q\}\), \(P\) is the solution to the discrete-time Riccati equation, and \(q^*\) is computed from (6.8) (for this problem, \(q^* = 0.0552\)). Computational simulations carried out on 1000 different initial conditions suggest that \(\Omega(\epsilon) \subseteq P(q^*)\) for \(\epsilon = 0.0001\).
6.5.3 Computational Bounds and Experimental Results

Implementation of a receding horizon control policy for the UAV requires that a quadratic program with 60 variables and 120 constraints be solved every 0.1 seconds. Given the types of embedded processors which might run on this vehicle, this is a relatively small window of time to solve a problem of this size. It is imperative that the computational requirements be known and bounded before implementation. This section uses the bounding techniques developed in Chapter 4. The bounds are compared to the actual run time of several interior-point algorithms. The short-step and predictor-corrector algorithms used are identical to those presented in [NT98], and thus are based on the NT search direction. The number of iterations for Mehrotra's predictor-corrector method is also presented, which in practice tends to be the interior-point algorithm with the fastest convergence. This comparison should give the reader a feel for how these run times and bounds compare to a typical interior-point algorithm used for off-line optimization, even though Mehrotra's algorithm does not have a known polynomial-time complexity guarantee. A complete computational comparison of algorithms would also include empirical
results from an active set method (see [Fle87, Wri97]), which would most likely be competitive with Mehrotra’s method. However, since the worst case iteration count of active set methods is known to grow exponentially, these methods lie outside the scope of this thesis.

**Initialization at Analytic Center**

The quadratic program is parameterized by the initial condition. Following the format given in Example 4.1 from Chapter 4, this quadratic program can be written as

\[
\begin{align*}
\min_v & \quad g(x_0)^T v + \frac{1}{2} \gamma \\
\text{subject to} & \quad \frac{\gamma + 1}{2} \geq \sqrt{\left(\frac{\gamma - 1}{2}\right)^2 + \|Gx\|^2} \\
& \quad Av \leq b, \quad \gamma \leq \gamma_{\text{max}}.
\end{align*}
\]

This form places the parameter varying term in the objective vector, and contains only linear and second-order cone constraints. The initialization strategy presented in Section 4.1 is applied to this problem. The constant \(\gamma_{\text{max}}\) bounds the maximum value of \(v^T G^T G v\). Using \(\|v\|_{\infty} \leq 0.4363\), then \(\gamma_{\text{max}} = 16.8167\) is valid for this problem.

Initialization follows exactly as presented in Example 4.1. Requiring that the initialization be contained in \(N(\beta)\), then the initial duality gap is

\[
\langle x^{(0)}, z^{(0)} \rangle = \frac{\beta}{\beta} \sqrt{c(x_0)^T (F^T \nabla^2 B(x^*) F)^{-1} c(x_0)}
\]

where \(c(x_0)\) is the parameter dependent objective vector. Let \(\beta = 1/10\). Since the duality gap is a convex function of \(x_0\), the maximum value is found at one of the vertices \(x_0 = [\pm 2 \pm 2 \pm 2 \pm \pi \pm 1]^T\). Computation of this maximum value leads to the bound

\[
\langle x^{(0)}, z^{(0)} \rangle \leq 5706.
\]

From Section 6.5.2, a final duality gap of \(\epsilon = 0.0001\) is recommended. Assuming that the path-following algorithm reduces the duality gap by at least \(1 - \Delta/\sqrt{\beta}\) with \(\Delta = 1/15\), then \(\langle x^{(k)}, z^{(k)} \rangle \leq \epsilon\) if

\[
k \geq \left\lceil \frac{\log(\epsilon/\langle x^{(0)}, z^{(0)} \rangle)}{\log(1 - \Delta/\sqrt{\beta})} \right\rceil = 2962.
\]
If the short-step algorithm is implemented without a line search, this is exactly the number of iterations required. If a line search is also used, the actual number of iterations is reduced to about 1500. The predictor-corrector algorithm takes no more than about 135 iterations. For comparison, Mehrotra’s predictor-corrector algorithm was also implemented on this problem. Using the version of Mehrotra’s algorithm found in the software package SDPPack 0.9 [AHN+97], a MATLAB based optimization package developed by Alizadeh et al, a maximum of 21 iterations was required.

**Initialization using Homogeneous Self-Dual Programming**

Now, suppose the quadratic program is posed as

\[
\begin{align*}
\min_{\gamma, x} & \quad \gamma \\
\text{subject to} & \quad \begin{bmatrix} \gamma \\ Gx + G^{-1}g(x_0) \end{bmatrix} \geq 0 \\
& \quad Ax \leq b
\end{align*}
\]

In this form, the constraints are parameter dependent, so it is appropriate to initialize the problem using homogeneous self-dual programming. In Section 4.2.2, it was shown that finding an iteration bound for a homogeneous self-dual program requires bounding the ratio \(\Psi(D)/\Gamma(D)\) from above, where \(D\) is the domain of the parameter \(x_0\).

Let the desired accuracy be \(\epsilon_1 = \epsilon_2 = \epsilon_3 = 10^{-4}\). From the problem data, the following bounds hold:

\[
\begin{align*}
\max_{x_0 \in D} ||\mathcal{F}_0(x_0)|| &= \max \sqrt{1 + ||G^{-1}g(x_0)||^2 + ||b - e||^2} = 8.1453 \\
||\bar{c}|| &= ||A^T e|| = 0 \\
\alpha &= 1 + b^T e = 53.3599
\end{align*}
\]

From this, it is seen that \(\Gamma(D) = \epsilon_3/\alpha = 1.87 \times 10^{-6}\).

Analysis of \(\Psi(D)\) proceeds in exactly the same way as developed in Example 4.4 from Chapter 4. The primal objective is bounded above by \(\bar{\lambda}_P = 1.3341\), and the dual objective bounded below by \(\bar{\lambda}_D = 0\). Using these numbers, we have the bound \(\Psi(D) \leq \Psi_P(D) + \Psi_D(D) \leq 54.25 + 13.37 = 67.62\). This leads to the bound on the
optimal value of $\tau$ over the parameter space:

$$\tau^* \geq \frac{\langle X^{(0)}, Z^{(0)} \rangle + \kappa^{(0)}}{\Psi(D)} = 1.8190.$$  

Assuming a path-following algorithm is implemented which keeps iterates within $N(1/10)$ of the central path and guarantees a duality gap reduction of $1 - 1/(15\sqrt{\theta} + 1)$ at each iteration, then any iteration satisfying

$$k \geq \left\lceil \log \left( \frac{\langle X^{(0)}, Z^{(0)} \rangle + \kappa^{(0)}(1 - 1/10)\Gamma(D)}{\Psi(D)(\theta + 1)} \right) / \log(1 - 1/(15\sqrt{\theta} + 1)) \right\rceil = 2905$$

is sufficient to guarantee the optimality requirements.

Experimentally, the smallest value of $\tau^*$ encountered was 1.8238, very close to the predicted value. In reality, the short-step method takes no more than about 2100 iterations when implemented on the actual problem for $x_0 = [\pm 2 \pm 2 \pm 2 \pm \pi \pm 1]^T$. When coupled with a line search, the short step algorithm took less than 1800 iterations. For the predictor-corrector method, the maximum number of iterations encountered was 76. Mehrotra’s predictor-corrector algorithm was also implemented on this problem for comparison. The software package SeDuMi 1.03 created by Sturm [Stu99b] is specifically designed to solve homogeneous self-dual programming problems over symmetric cones, which is ideal for this problem. Implementation using this software took a maximum of 12 iterations.

### 6.5.4 Conclusions

It should not come as a surprise that the computational bounds for the UAV seem so conservative, especially when compared to Mehrotra’s method. Currently, the theory for interior-point algorithms limits the guaranteed fractional decrease in duality gap to $(1 - 1/(15\sqrt{\theta}))$, which is approximately 0.994 for this problem. In contrast, the duality gap reduction achieved by Mehrotra’s algorithm is frequently as small as 0.1. Figure 6.11 shows the fractional decrease of the objective per iteration for a typical run of SeDuMi on this example. Reduction at each iteration tends to be between 0.3–0.4, although is as small as 0.006 at the last iteration.
Despite the vast difference between convergence of Mehrotra's algorithm and the conservative path-following algorithms, the path-following method used here does have one advantage for on-line applications: predictability. Path-following algorithms take steps which do not stray far from the central path, causing the progress towards the optimal solution to be nearly identical from iteration to iteration. A predictable algorithm is much more appropriate for on-line use, since it is not likely to encounter surprises during implementation. Further comments on the practicality of these methods for on-line optimization are found in the next chapter.

It is also worth considering whether such an algorithm is computationally realistic to consider for the UAV. To make this evaluation, the number of arithmetic operations per iteration of the algorithm must be estimated. For linear and second-order cone programming, the main computational effort in deriving the Newton step is spent solving a $n \times n$ linear system of equations, where $n$ is the number of variables in the convex program. For a dense system of equations, this takes roughly $n^3$ operations. Since $n = 60$ for the UAV example, each iteration takes roughly 200000 arithmetic operations. Multiplying this number by an iteration bound of 3000 yields $6 \times 10^8$ operations, which must be
computed in 0.1 seconds. In order to meet this computational requirement, a 6 gigaflop (6 billion floating point operations per second) computer would be needed! Currently, only massive parallel processors such as the 512-processor CRAY T3D supercomputer are capable of meeting this challenge. This is hardly the sort of computer that could be expected to run on a small autonomous air vehicle. Even Mehrotra’s algorithm, which appears to run over 100 times faster, would stress the computer resources of the fastest single-processor personal computers on the market today. However, it is expected that processing power will only get cheaper in the years to come, and it is conceivable that someday this power will be available as a small embedded microprocessor.
Chapter 7

Concluding Remarks

There is an established need for certification of on-line optimization for control systems. As computer technology moves into the 21st century, control methodologies utilizing on-line optimization are becoming increasingly viable for a wide range of systems. These methods open the door to much more aggressive control systems than are possible with traditional control methodologies. Yet without the ability to guarantee that the optimization algorithms will always deliver the desired solution on schedule, on-line optimization based control methods will not be able to meet the stringent safety requirements demanded of real-time software. These safety requirements are especially pronounced for aerospace control applications, where algorithm failure can have catastrophic consequences. Certification of on-line optimization means conducting a thorough computational analysis, providing a guarantee that the algorithm will always converge in the allotted time under all possible flight conditions. Depending on the severity of consequences resulting from algorithm failure, this guarantee may be relaxed to mean that the risk of failure be very small or negligible. This motivates some of the practical considerations raised in Section 7.2.

This thesis should be viewed as an important step in bridging the gap between the control and optimization communities. Control theorists often propose methods using on-line optimization without regard for whether these methods can be certified. Applied control engineers tend to ignore these methods out of an inherent mistrust of on-line optimization, since it necessarily contains iterative loops. Finally, operations researchers
are frequently unaware of the issues facing on-line control applications, and continue to pursue mathematical programming from an off-line perspective.

Applied control engineers have so far lacked the time-to-convergence numbers needed for certification, yet the required theory exists to provide these numbers. Convergence guarantees of interior-point algorithms have long been known by operations researchers, who often only emphasize the theoretical interest of these guarantees despite the obvious relevance to certification. Most optimization algorithms used in practice employ many heuristics which invalidate complexity results but tend to have extremely fast convergence. Often these heuristics must be tuned for a specific problem (e.g., in Mehrotra's algorithm, it is advantageous to limit the fraction of the step to the boundary of the feasible cone, where the choice of the fraction depends on the numerics of the problem). While these \textit{ad hoc} methods are very appropriate for off-line optimization, they are ill-suited for on-line problems which require predictable behavior for a variety of problems. This has motivated the focus of this thesis on algorithms which do have provable worst-case performance and predictable behavior.

This thesis provides a methodology for computational analysis of on-line convex optimization, and given a case where the computational requirements cannot be met by the on-line computer, suggest cheaper alternatives to numerical optimization. The convex problems considered here are kept in a general form, making the analysis applicable to the most useful convex programming problems: linear, second-order cone, and semidefinite programming. The class of algorithms considered are primal-dual path-following algorithms. It is the contention of this thesis that path-following algorithms, in their purest form, are appropriate for on-line optimization, since they are very predictable, have theoretical guarantees, and produce numerically stable iterates which stay close to the central path. Furthermore, primal-dual path-following algorithms have the best known complexity results. Thus, their theoretical worst case convergence is at least as good as any other algorithm.

This thesis also makes the first step towards considering the impact of real optimization algorithms on a closed-loop system. Interior-point methods do not return optimal solutions, but instead return solutions which are within a tolerance $\epsilon$ of optimality. To
the knowledge of this author, the stability of receding horizon control using an \( \epsilon \)-optimal optimization algorithm has never before been considered. Knowledge of the largest value of \( \epsilon \) which is sufficient for stability should drive the requirements for the algorithm accuracy, which in turn determines the computational requirements. These requirements are the key ingredients for on-line certification.

7.1 Contributions

A brief summary of the contributions made by this thesis is given below.

- A method for checking the feasibility and well-posedness of a compact parametric set of convex programming problems is provided. Although this method is based on a branch-and-bound procedure, it simplifies considerably for many relevant problems.

- An initialization strategy of a primal-dual path-following algorithm is given for on-line convex programs with fixed constraints. This initialization can be quickly computed on-line, and is sufficiently centered for any objective vector. Accompanying this initialization is an iteration bound which is easy to compute and is non-conservative, assuming the theoretical worst-case complexity.

- Computational bounds are derived for on-line convex programs with time-varying objective and constraints. These bounds are based on the use of homogeneous self-dual programming, and are computed using a branch-and-bound procedure.

- Several alternatives to on-line optimization are proposed. These methods require very few on-line computations, and always return a solution which meets the problem constraints. The proposed methods include robust table look-up, functional approximation of the solution set, and ellipsoidal approximation of the constraint set.

- A detailed analysis is provided for receding horizon control of constrained linear systems. This marks the first stability analysis of receding horizon control coupled
with a real-time optimization algorithm. Sufficient conditions are given for closed-loop stability, as well as feasibility, well-posedness, and bounded iterations for the optimal control problem.

- A realistic example of a UAV with saturating actuators is analyzed. Stability requirements are imposed, and computational bounds for an on-line optimization algorithm are derived.

7.2 Practical Considerations

Given the conservative nature of the iteration bounds derived in Chapter 4, it is tempting to question their practicality. For example, the iteration bounds derived for the UAV in Section 6.5 were on the order of 3000, while in practice, Mehrotra’s algorithm took roughly 12 iterations. This begs the question: does it make sense to spec a supercomputer for a job that only seems to require a personal computer? Of course the answer is no, but does that in turn mean the foregone analysis is without practical value for the real problem? The answer is again no, for reasons which will soon become clear.

The contribution of this thesis has largely been in bounding the difference in duality gap between initialization and termination of an on-line optimization procedure, and has relied on the theoretical estimate of the convergence rate to derive iteration bounds. When compared to actual algorithm performance, the large values of the iteration bounds are almost entirely due to the conservative theoretical estimate of the linear convergence rate.

The actual convergence rate is limited because iterates are restricted to a narrow neighborhood of the central path. Operations researchers have worked with these neighborhoods to prove guaranteed convergence rates which hold in general. In some cases, specific cones may have better convergence proofs than the general case. For example, the convergence rate used in this thesis has been the Nesterov-Todd result of \( \frac{\mu_{k+1}}{\mu_k} \leq (1 - \frac{1}{(15\sqrt{d})}) \), which is based on a restriction of the iterates to \( \mathcal{N}(1/10) \). Yet, in the case of linear programming, it can be proved that \( \frac{\mu_{k+1}}{\mu_k} \leq (1 - 0.4/\sqrt{d}) \) for iterates restricted to \( \mathcal{N}(0.4) \) [Wri96]. Supposing that this property held for the quadratic program
generated by the UAV example, the ratio \( \frac{\mu_{k+1}}{\mu_k} \) would be guaranteed to be less than 0.964 rather than 0.994, reducing the iteration bound from 2905 to 477. This reduction is significant, and it may be possible to do much better by looking beyond the theory.

It will be recalled that path-following algorithms were chosen for on-line optimization for two reasons: predictability and available convergence proofs. The convergence proofs relied on choosing a very narrow neighborhood of the central path. The neighborhood frequently referred to in this thesis is \( \mathcal{N}(1/10) \). However, suppose that requirement of having proofs bounding the ratio \( \frac{\mu_{k+1}}{\mu_k} \) is relaxed, such that it is only important to witness consistent behavior among iterations. Since \( \mathcal{N}(1/10) \) was only important because it enabled a convergence proof, it can be widened as long as there is still consistent behavior of the iterates. Suppose widening the neighborhood to \( \mathcal{N}(1/4) \) allowed a short-step algorithm which reduced the duality gap by a factor between 0.75–0.85 every time in experimental simulations. In fact, this is the observed behavior for the UAV example. Using the ratio 0.85 rather than 0.994 reduces the iteration bound to 108, which is within a factor of 10 of Mehrotra's algorithm. This would be a much more practical bound.

The truth is that no system can be designed to be 100% reliable. Systems engineers typically measure the reliability of a system in terms of likelihood of hazard occurrence per unit time, such as \( 10^{-12} \) incidents per year. This type of risk analysis can be done here, given a probability distribution of the number of iterations. Rather than saying the algorithm will never take more than 3000 iterations, it may be more useful to say the chance of the algorithm taking more than 150 iterations is one in a trillion. In principle, Monte Carlo simulations can be applied to the optimization algorithm to determine a probability distribution for the ratio \( \frac{\mu_{k+1}}{\mu_k} \). From this distribution, a probability distribution on the number of iterations can be derived. This may be the key to practical certification of on-line optimization algorithms.

### 7.3 Further Research

Now that it has been demonstrated that optimization algorithms can be certified for safety-critical on-line applications, the next step is to design optimization algorithms
specifically tailored for this use. These algorithms need to be simple, predictable, and should not be much more computationally intensive than efficient off-line algorithms. Several researchers have already suggested interior-point algorithms designed for receding horizon control (see [Han00, RWR98, Wri93, Wri97]). The methods proposed in these references have not yet addressed certification, but could easily be extended for this type of analysis. Ideally, an on-line algorithm would have similar traits to the short-step algorithm, taking consistent steps towards the optimal solution. Of course, the short-step algorithm converges very slowly, and therefore would not be useful for many applications. By widening the neighborhood of the central path and taking larger steps, the short-step algorithm can be accelerated while maintaining predictability, although the general theoretical results may be violated.

A more detailed study of the convergence ratio $\frac{\mu_{k+1}}{\mu_k}$ is merited. This thesis has assumed that the best value is $(1 - \Delta/\sqrt{\vartheta})$, which causes the iteration bound to grow as $O(\sqrt{\vartheta})$. However, numerous researchers have observed that in practice, the growth is much less than $O(\sqrt{\vartheta})$, and at times seems independent of $\vartheta$ [Wri93, VB96]. It is conceivable that for a specific problem instance, a numerical proof could be constructed which would find a much lower bound for $\frac{\mu_{k+1}}{\mu_k}$ than the general theoretical result. This would provide much tighter iteration bounds, and would be of more practical value. Even if these results prove difficult to find, there still remains the possibility of a statistical analysis, derived analytically or experimentally in simulation.
Bibliography


BIBLIOGRAPHY


