

ELLIPSOIDAL AND SEMI-ELLIPSOIDAL CONTROLLED  
INVARIANT SETS FOR CONSTRAINED  
LINEAR SYSTEMS

By

BRIAN DWAYNE O'DELL

Bachelor of Science  
Oklahoma State University  
Stillwater, Oklahoma  
1993

Master of Science  
Oklahoma State University  
Stillwater, Oklahoma  
1994

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
DOCTOR OF PHILOSOPHY  
July, 1999

Thesis  
1999D  
823e

COPYRIGHT

By

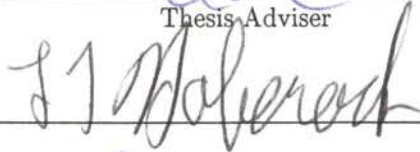
BRIAN DWAYNE O'DELL

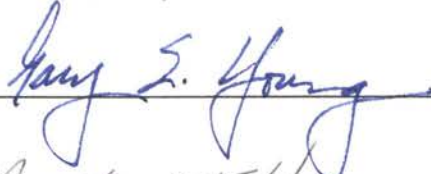
July, 1999

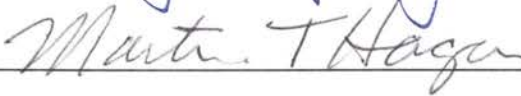
ELLIPSOIDAL AND SEMI-ELLIPSOIDAL CONTROLLED  
INVARIANT SETS FOR CONSTRAINED  
LINEAR SYSTEMS

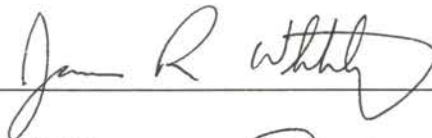
Thesis Approved:

  
\_\_\_\_\_  
Thesis Adviser

  
\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_

Dean of the Graduate College

## ACKNOWLEDGMENTS

I would first like to express my sincere appreciation to my advisor, Dr. Eduardo A. Misawa, for his wisdom, direction, and support during the course of this entire process. I also wish to acknowledge the other members of my advisory committee, Dr. Lawrence L. Hoberock, Dr. Gary E. Young, Dr. Martin T. Hagan, and Dr. James R. Whiteley, for agreeing to oversee my work and for providing a balanced perspective of engineering theory and practice.

In addition, this five years of work would certainly not have been possible without financial support in several areas. I would first like to thank the National Science Foundation, for this material is based upon work supported under a National Science Foundation Graduate Fellowship. Thanks also to Mr. and Mrs. J. Roy Dorrough for providing an Oklahoma State University Distinguished Graduate Fellowship. Finally, I would like to thank the Oklahoma Center for the Advancement of Science and Technology (OCAST) and Seagate Technology, Inc. of Oklahoma City, Oklahoma, for their sponsorship of the project which inspired this research.

Finally, sincerest thanks must be given to those who have provided advice, support, and encouragement on matters largely outside the realm of academia: to the people of Bible Baptist Church, Stillwater, Oklahoma, but foremost to my family, who have supported me in so many ways. Of all the lessons learned, the most important have been taught by these people, who by their instruction and example prepare me to walk circumspectly, as a man of wisdom and not merely of knowledge.

“Let us hear the conclusion of the whole matter: Fear God, and keep his commandments:  
for this is the whole duty of man.” - *Ecclesiastes 12:13*

## TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Comments on Invariance . . . . .	5
1.2	Outline of Thesis . . . . .	6
<b>2</b>	<b>Literature Review</b>	<b>8</b>
2.1	Motivation & Contributions . . . . .	10
<b>3</b>	<b>Ellipsoidal Sets</b>	<b>12</b>
3.1	Recoverable Ellipsoidal Set . . . . .	12
3.1.1	Objective Function . . . . .	15
3.1.2	Constraints . . . . .	16
3.1.3	Implementation Issues . . . . .	17
3.2	Reachable Ellipsoidal Set . . . . .	18
3.3	Controllable Ellipsoidal Set . . . . .	18
3.4	Summary . . . . .	19
<b>4</b>	<b>Ellipsoidal Set Examples</b>	<b>20</b>
4.1	Second Order Systems . . . . .	20
4.1.1	Stable . . . . .	21
4.1.2	Stable Focus . . . . .	22
4.1.3	Marginal . . . . .	25
4.1.4	Unstable . . . . .	27
4.1.5	Unstable Focus . . . . .	27
4.1.6	Saddle . . . . .	30
4.2	Third Order Systems . . . . .	32
4.2.1	Stable . . . . .	32
4.2.2	Unstable . . . . .	33
4.2.3	Marginal . . . . .	35

4.2.4	Mixed . . . . .	38
4.3	Comparison Against Published Result . . . . .	43
<b>5</b>	<b>Semi-Ellipsoidal Sets</b>	<b>45</b>
5.1	Recoverable Semi-Ellipsoidal Set . . . . .	46
5.1.1	Theory . . . . .	47
5.1.2	Design . . . . .	50
5.1.3	Main Result . . . . .	54
5.1.4	Implementation Issues . . . . .	57
5.2	Reachable Semi-Ellipsoidal Set . . . . .	59
5.3	Controllable Semi-Ellipsoidal Set . . . . .	60
5.4	Summary . . . . .	61
<b>6</b>	<b>Semi-Ellipsoidal Set Examples</b>	<b>63</b>
6.1	Second Order Systems . . . . .	63
6.1.1	Stable . . . . .	63
6.1.2	Stable Focus . . . . .	66
6.1.3	Marginal . . . . .	66
6.1.4	Unstable . . . . .	70
6.1.5	Unstable Focus . . . . .	70
6.1.6	Saddle . . . . .	73
6.2	Third Order Systems . . . . .	76
6.2.1	Stable . . . . .	76
6.2.2	Unstable . . . . .	79
6.2.3	Marginal . . . . .	81
6.2.4	Mixed . . . . .	84
6.3	Comparison Against Published Result . . . . .	87
<b>7</b>	<b>Concluding Remarks</b>	<b>89</b>
7.1	Future Research . . . . .	90
	<b>Bibliography</b>	<b>94</b>
<b>A</b>	<b>Existence of Controllable Ellipsoid</b>	<b>99</b>
<b>B</b>	<b>Derivation of Ellipsoid Volume Cost Function</b>	<b>102</b>

<b>C</b>	<b>Derivation of State Constraint Inequalities</b>	<b>106</b>
C.1	Level Sets	106
C.1.1	Ellipsoidal Set Boundary as a Level Set	106
C.1.2	Linear Constraint Boundary as a Level Set	107
C.2	Derivation of Inequality	108
C.3	Derivation of Control Constraint Inequalities	111
<b>D</b>	<b>Gradients of Cost Functions and Constraints</b>	<b>112</b>
D.1	Preliminaries	112
D.2	Identities	113
D.3	Analytic Derivatives	114
<b>E</b>	<b>Overlapping Ellipsoid Constraint Derivation for Variable Structure Control</b>	<b>120</b>
E.1	Derivation of Linear Transformation	122
E.2	Analytic Gradient of Constraint	123
<b>F</b>	<b>Matlab Code for Ellipsoidal Sets</b>	<b>126</b>
F.1	Function Files for Computing Recoverable Ellipsoidal Set	126
F.1.1	Optimization Routine	126
F.1.2	Cost Function and Constraints	133
F.1.3	Derivatives of Cost Function and Constraints	135
F.1.4	Plotting Routine	138
F.2	Function Files for Computing Reachable Ellipsoidal Set	140
F.2.1	Optimization Routine	140
F.2.2	Cost Function and Constraints	146
F.2.3	Derivatives of Cost Function and Constraints	148
F.2.4	Plotting Routine	151
F.3	Function Files for Computing Controllable Ellipsoidal Set	153
F.3.1	Optimization Routine	153
F.3.2	Cost Function and Constraints	160
F.3.3	Derivatives of Cost Function and Constraints	162
F.3.4	Plotting Routine	167
F.4	Miscellaneous Files	170
F.4.1	Plotting Point Generator for Ellipsoidal Set	170



F.4.2	Search Parameter/Matrix Parameter Mapping Routines . . . . .	173
<b>G</b>	<b>Matlab Code for Semi-Ellipsoidal Sets</b>	<b>175</b>
G.1	Function Files for Computing Recoverable	
Semi-Ellipsoidal Set . . . . .		175
G.1.1	Optimization Routine . . . . .	175
G.1.2	Cost Function and Constraints . . . . .	184
G.1.3	Derivatives of Cost Function and Constraints . . . . .	186
G.1.4	Cost Function and Constraints for Finding $\tilde{u}$ . . . . .	190
G.1.5	Derivatives of Cost Function and Constraints for Finding $\tilde{u}$ . . . . .	191
G.1.6	Plotting Routine . . . . .	192
G.2	Function Files for Computing Reachable	
Semi-Ellipsoidal Set . . . . .		194
G.2.1	Optimization Routine . . . . .	194
G.2.2	Cost Function and Constraints . . . . .	202
G.2.3	Derivatives of Cost Function and Constraints . . . . .	204
G.2.4	Cost Function and Constraints for Finding $\tilde{u}$ . . . . .	208
G.2.5	Derivatives of Cost Function and Constraints for Finding $\tilde{u}$ . . . . .	209
G.2.6	Plotting Routine . . . . .	210
G.3	Function Files for Computing Controllable	
Semi-Ellipsoidal Set . . . . .		212
G.3.1	Optimization Routine . . . . .	212
G.3.2	Cost Function and Constraints . . . . .	221
G.3.3	Derivatives of Cost Function and Constraints . . . . .	223
G.3.4	Cost Function and Constraints for Finding $\tilde{u}$ . . . . .	229
G.3.5	Derivatives of Cost Function and Constraints for Finding $\tilde{u}$ . . . . .	230
G.3.6	Plotting Routine . . . . .	231
G.4	Miscellaneous Files . . . . .	234
G.4.1	Plotting Point Generator for Semi-Ellipsoidal Set . . . . .	234
G.4.2	Search Parameter/Matrix Parameter Mapping Routines . . . . .	237

LIST OF TABLES

6.1 Comparison of Number of Required Parameters and Enclosed Area. . . . . 87

## LIST OF FIGURES

1.1 Illustration of Point Which is Reachable but Not Recoverable. . . . .	4
1.2 Illustration of Recoverable, Reachable, and Controllable Points. . . . .	4
3.1 Maximal Recoverable Set. . . . .	13
3.2 Approximate Maximal Recoverable Set. . . . .	14
3.3 Ellipsoidal Level Sets of $V$ . . . . .	15
4.1 Recoverable Ellipsoid for Second Order Stable System. . . . .	22
4.2 Reachable Ellipsoid for Second Order Stable System. . . . .	22
4.3 Controllable Ellipsoid for Second Order Stable System. . . . .	23
4.4 Recoverable Ellipsoid for Second Order Stable Focus System. . . . .	23
4.5 Reachable Ellipsoid for Second Order Stable Focus System. . . . .	24
4.6 Controllable Ellipsoid for Second Order Stable Focus System. . . . .	24
4.7 Recoverable Ellipsoid for Second Order Marginal System. . . . .	25
4.8 Reachable Ellipsoid for Second Order Marginal System. . . . .	26
4.9 Controllable Ellipsoid for Second Order Marginal System. . . . .	26
4.10 Recoverable Ellipsoid for Second Order Unstable System. . . . .	27
4.11 Reachable Ellipsoid for Second Order Unstable System. . . . .	28
4.12 Controllable Ellipsoid for Second Order Unstable System. . . . .	28
4.13 Recoverable Ellipsoid for Second Order Unstable Focus System. . . . .	29
4.14 Reachable Ellipsoid for Second Order Unstable Focus System. . . . .	29
4.15 Controllable Ellipsoid for Second Order Unstable Focus System. . . . .	30
4.16 Recoverable Ellipsoid for Second Order Saddle System. . . . .	31
4.17 Reachable Ellipsoid for Second Order Saddle System. . . . .	31
4.18 Controllable Ellipsoid for Second Order Saddle System. . . . .	31
4.19 Recoverable Ellipsoid for Third Order Stable System. . . . .	33
4.20 Reachable Ellipsoid for Third Order Stable System. . . . .	34

4.21 Controllable Ellipsoid for Third Order Stable System. . . . .	34
4.22 Recoverable Ellipsoid for Third Order Unstable System. . . . .	36
4.23 Reachable Ellipsoid for Third Order Unstable System. . . . .	37
4.24 Controllable Ellipsoid for Third Order Unstable System. . . . .	37
4.25 Recoverable Ellipsoid for Third Order Marginal System. . . . .	38
4.26 Reachable Ellipsoid for Third Order Marginal System. . . . .	39
4.27 Controllable Ellipsoid for Third Order Marginal System. . . . .	39
4.28 Recoverable Ellipsoid for Third Order Mixed System. . . . .	41
4.29 Reachable Ellipsoid for Third Order Mixed System. . . . .	42
4.30 Controllable Ellipsoid for Third Order Mixed System. . . . .	42
4.31 Comparison of Ellipsoidal and Polyhedral Methods for Double Integrator System. . . . .	44
5.1 Invariant Subset of an Invariant Ellipsoid. . . . .	45
5.2 Illustration of Trajectories Emanating from State Constraint. . . . .	46
5.3 Illustration of Ellipsoid Violating a State Constraint. . . . .	47
5.4 Derivative Function Subsets. . . . .	48
5.5 Acceptable and Unacceptable Regions of a Constraint Boundary. . . . .	49
5.6 Variable Structure Control Failing Invariance of $\hat{S}_v$ . . . . .	50
5.7 Comparison of Requirements for Theorems 5.1 and 5.2. . . . .	51
5.8 Illustration of Invariant Ellipse and Intersecting Hyperplane. . . . .	53
5.9 Illustration for Proof of Corollary 5.3. . . . .	54
5.10 Illustration of Reduced Dimension Ellipsoid and State Constraint Boundary in $R^2$ . . . . .	55
5.11 Illustration of Reduced Dimension Ellipsoid and State Constraint Boundary in $R^1$ . . . . .	55
5.12 Progression of Search Using Ellipsoid Volume as Objective Function. . . . .	58
5.13 Maximal Control Occurring Outside of $\hat{S}_v$ . . . . .	59
5.14 Maximal Control Occurring Inside of $\hat{S}_v$ . . . . .	60
5.15 Duality of Constraint Derivative Regions when $\Gamma_i B = 0$ . . . . .	61
6.1 Recoverable Semi-Ellipsoidal Set for Second Order Stable System. . . . .	64
6.2 Reachable Semi-Ellipsoidal Set for Second Order Stable System. . . . .	65
6.3 Controllable Semi-Ellipsoidal Set for Second Order Stable System. . . . .	65
6.4 Recoverable Semi-Ellipsoidal Set for Second Order Stable Focus System. . . . .	66
6.5 Reachable Semi-Ellipsoidal Set for Second Order Stable Focus System. . . . .	67
6.6 Controllable Semi-Ellipsoidal Set for Second Order Stable Focus System. . . . .	67

6.7 Recoverable Semi-Ellipsoidal Set for Second Order Marginal System. . . . .	68
6.8 Reachable Semi-Ellipsoidal Set for Second Order Marginal System. . . . .	69
6.9 Controllable Semi-Ellipsoidal Set for Second Order Marginal System. . . . .	69
6.10 Controllable Semi-Ellipsoidal Set for Second Order Marginal System using Increased Control Effort. . . . .	69
6.11 Recoverable Semi-Ellipsoidal Set for Second Order Unstable System. . . . .	70
6.12 Reachable Semi-Ellipsoidal Set for Second Order Unstable System. . . . .	71
6.13 Controllable Semi-Ellipsoidal Set for Second Order Unstable System. . . . .	71
6.14 Recoverable Semi-Ellipsoidal Set for Second Order Unstable Focus System. . . . .	72
6.15 Reachable Semi-Ellipsoidal Set for Second Order Unstable Focus System. . . . .	72
6.16 Controllable Semi-Ellipsoidal Set for Second Order Unstable Focus System. . . . .	73
6.17 Recoverable Semi-Ellipsoidal Set for Second Order Saddle System. . . . .	74
6.18 Reachable Semi-Ellipsoidal Set for Second Order Saddle System. . . . .	74
6.19 Controllable Semi-Ellipsoidal Set for Second Order Saddle System. . . . .	75
6.20 Recoverable Semi-Ellipsoidal Set for Third Order Stable System. . . . .	77
6.21 Reachable Semi-Ellipsoidal Set for Third Order Stable System. . . . .	77
6.22 Controllable Semi-Ellipsoidal Set for Third Order Stable System. . . . .	78
6.23 Controllable Semi-Ellipsoidal Set Trajectories for Third Order Stable System. . . . .	78
6.24 Recoverable Semi-Ellipsoidal Set for Third Order Unstable System. . . . .	79
6.25 Reachable Semi-Ellipsoidal Set for Third Order Unstable System. . . . .	80
6.26 Controllable Semi-Ellipsoidal Set for Third Order Unstable System. . . . .	80
6.27 Controllable Semi-Ellipsoidal Set Trajectories for Third Order Unstable System. . . . .	81
6.28 Recoverable Semi-Ellipsoidal Set for Third Order Marginal System. . . . .	82
6.29 Reachable Semi-Ellipsoidal Set for Third Order Marginal System. . . . .	82
6.30 Controllable Semi-Ellipsoidal Set for Third Order Marginal System. . . . .	83
6.31 Controllable Semi-Ellipsoidal Set Trajectories for Third Order Marginal System. . . . .	83
6.32 Recoverable Semi-Ellipsoidal Set for Third Order Mixed System. . . . .	85
6.33 Reachable Semi-Ellipsoidal Set for Third Order Mixed System. . . . .	85
6.34 Controllable Semi-Ellipsoidal Set for Third Order Mixed System. . . . .	86
6.35 Controllable Semi-Ellipsoidal Set Trajectories for Third Order Mixed System. . . . .	86
6.36 Comparison of Semi-Ellipsoidal, Ellipsoidal, and Polyhedral Methods for Double In- tegrator System. . . . .	88

C.1 Illustration of Boundary Intersection. . . . .	109
E.1 Illustration of Biased Constraint Derivative Boundary. . . . .	121

# Chapter 1

## Introduction

An inherent evil of model-based control is the inevitable discrepancy between the actual system and the system model for which the controller was designed. Failure to consider these discrepancies, or modeling errors, can lead to undesirable - and potentially unstable - results. Often, such errors are unavoidable due to insufficient information about the system. In many instances, however, they are intentional, resulting from simplifying assumptions in the model.

One such simplification is the approximation of a system with constraints on the states (saturation nonlinearities on the state variables) as a purely linear system. Suppose the controllable, single-input, linear, time-invariant (LTI) system, (1.1), (1.2), has limits on the magnitude of the state variables and on the available input.

$$\dot{x} = Ax + Bu, x \in R^n \quad (1.1)$$

$$y = Cx + Du \quad (1.2)$$

Specifically, the following linear inequality constraints exist, where each inequality is assumed to contain the origin.

$$G_i = \{x | \Gamma_i x \leq 1\}, i = 1, \dots, k \quad (1.3)$$

$$U = \{u | |u| \leq \bar{u}\} \quad (1.4)$$

**Definition 1.1** *The maximal state-space,  $G$ , is given by the intersection of the inequality constraints, (1.5).*

$$G = \bigcap_i G_i \quad (1.5)$$

Obviously, such a system is indeed linear so long as the states remain in the unsaturated mode, so one approach is to keep the state in this linear range. However, this method generally results in

overly conservative restrictions on the operating space, limiting both the range of operation and the achievable performance characteristics. The problems of performance and stability are compounded by the limit on the available control effort.

The problem of remaining inside a set of prescribed bounds on the states is non-causal, requiring prediction to determine if the available control effort is sufficient to keep the state within these bounds. Essentially, it is a two point boundary value problem since the initial condition,  $x(t_0)$ , and the final state,  $x(t_f)$  (e.g., the origin for the regulator problem), are known. If no realizable control signal exists to take the system from  $x(t_0)$  to  $x(t_f)$  without violating the state constraints, then the point  $x(t_0)$  is not admissible.

**Definition 1.2** *A trajectory,  $x(t)$ , or control signal,  $u(t)$ , is said to be **admissible** on  $t \in [t_1, t_2]$  if  $x(t) \in G$ ,  $u(t) \in \mathcal{U}$  ( $t \in [t_1, t_2]$ ), respectively.*

In general, the issue is whether or not two points,  $x_a, x_b \in G$ , can be accessed from one another along some admissible trajectory using some admissible control signal. The origin being of particular interest in most control system designs, the following definitions are stated with the note that  $x_2$  is taken as the origin for the remainder of the thesis. These definitions adapt those established by LeMay [32], who studied the constrained input problem, by including restrictions on the states, as well.

**Definition 1.3** *An element  $x_1 \in R^n$  is a **recoverable state** in  $(t_0, t_f)$  with respect to  $x_2$  if there exists an admissible control which will drive the system from state  $x_1$  at time  $t_0$  to  $x_2$  at time  $t_f$  along an admissible trajectory. The **maximum region of recoverability** with respect to  $x_2$ ,  $S_v$ , is the set of all recoverable states in  $(t_0, t_f)$  with respect to  $t_f$ .*

**Definition 1.4** *An element  $x_1 \in R^n$  is a **reachable state** in  $(t_0, t_f)$  with respect to  $x_2$  if there exists an admissible control which will drive the system from  $x_2$  at time  $t_0$  to state  $x_1$  at time  $t_f$  along an admissible trajectory. The **maximum region of reachability** with respect to  $x_2$ ,  $S_e$ , is the set of all reachable states in  $(t_0, t_f)$  with respect to  $t_f$ .*

**Definition 1.5** *Two elements  $x_1, x_2 \in R^n$ , are a **dual pair of states**, or **dual states**, in  $(t_0, t_f)$  if there exists an admissible control which will drive the system from  $x_1$  at  $t_0$  to  $x_2$  at  $t_f$  along an admissible trajectory, and if there exists an admissible control which will drive the system from  $x_2$  at  $t_0$  to  $x_1$  at  $t_f$  along an admissible trajectory. The **maximum region of controllability** in  $(t_0, t_f)$ ,  $S_c$ , is that set of states in  $R^n$  for which:*

1. Any pair of states in that set is a dual pair in  $(t_0, t_f)$ .



2. There does not exist a state in  $R^n$  not in the set which is the dual in  $(t_0, t_f)$  of each state in the set.

**Remark 1.1** *The problem of control of systems with constrained states and input first received widespread attention in the 1960's and early 1970's but apparently faded due to the complexity of the problem and the proposed solutions. Interest was revived by the availability of computers in the late 1980's and onwards. However, the terms used in the earlier publications often differ from those used in the more recent ones. In addition, technologies originating in other fields have been applied to the problem, introducing additional words and phrases to the mix. The result is a lack of consistency in the terminology. The terms **recoverable**, **reachable**, and **controllable** are used here because of their introduction in one of the earliest complete works in the field of constrained input set theory and because they help to describe what is physically occurring. In [16], **controllable**, **reachable**, and **maneuverable** are used to describe the same sequence of sets. Other terms may be encountered in the literature, most notably **viable** as an alternative for **recoverable** (e.g. [4],[13]) and **attainable** as an alternative for **reachable** (e.g. [1],[2]).*

To better illustrate the issues involved, one may imagine a simple, second order mass-spring-damper system traveling at some positive velocity (which for this case is assumed to be its maximum admissible velocity) and consider what control effort is necessary to maintain the system inside a position limit. The point designated  $x_e$  in Figure 1.1, the phase plane of the system's control canonical form (with state bounds denoted by the dotted lines), illustrates the point in time when the system state has reached the maximum position bound as a result of traveling at the maximum velocity.

To maintain the system state inside the prescribed limit, a negative impulse signal (control effort) must be sent to the system to drive the velocity to zero (point  $x_c$ ) instantaneously. Obviously, such a signal violates any finite constraint on the input. Consequently, one must determine the point,  $x^*$ , at which maximum negative control effort must be applied to ensure that the system state will remain inside the constraint.

From this illustration, it is seen that the point  $x_e$  is not *recoverable* in the sense that no admissible control signal exists which will maintain the system trajectory inside the state bounds. However, the point is *reachable* since the state can be accessed from at least one other point in  $G$  via some trajectory remaining in  $G$ .

These ideas of recoverability and reachability can be expanded by considering other points in the state-space. Figure 1.2 shows, in addition to  $x_e$ , two additional points,  $x_v$ ,  $x_c$ , and representative

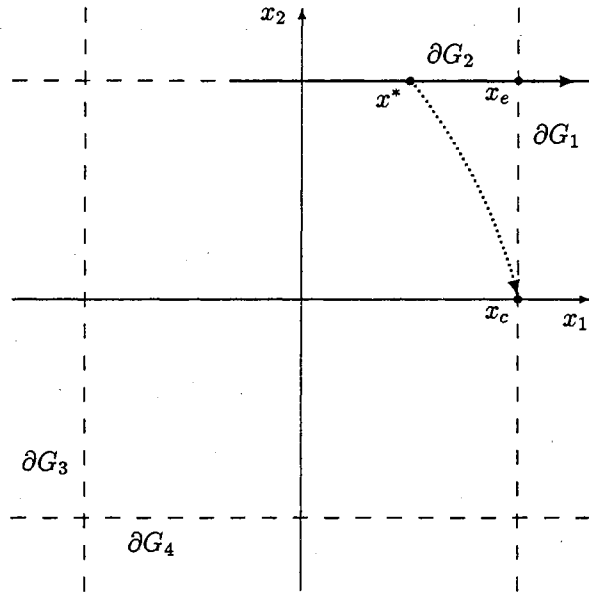


Figure 1.1: Illustration of Point Which is Reachable but Not Recoverable.

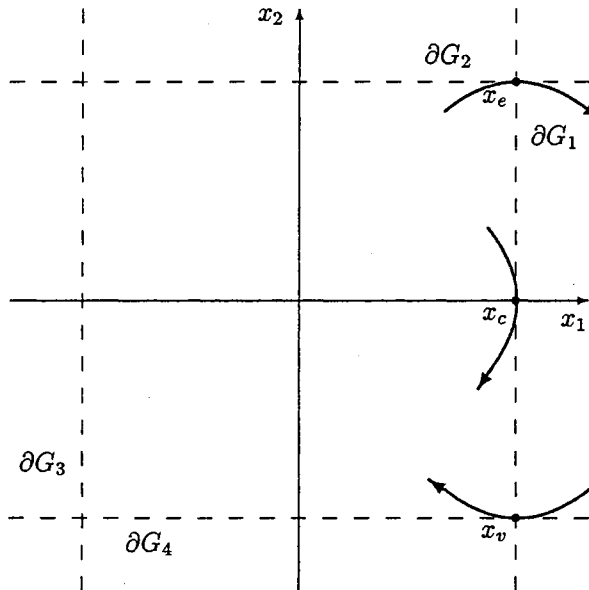


Figure 1.2: Illustration of Recoverable, Reachable, and Controllable Points.

trajectories through the points (The precise path is, of course, control dependent, but for this discussion it is only important to note that trajectories must tend to the right if the velocity is positive and to the left if the velocity is negative.) In contrast to  $x_e$ , the point  $x_v$  is recoverable but not reachable, while  $x_c$  is both reachable and recoverable, a property which is referred to as *controllable*.

## 1.1 Comments on Invariance

As discussed in Chapter 2, several options exist for solving the problem of control with constrained states and input. The method used in this thesis is that of identifying *a priori* a subset,  $\hat{S}$ , of the state-space so that, for every state in that subset, there exists an admissible control law such that the state is reachable/recoverable/controllable, depending on the desired set. This involves the notion of invariance, and it is important to clarify the terminology with a few definitions.

**Definition 1.6** [8] *The set  $S \subset R^n$  is said positively invariant for a system of the form*

$$\frac{d}{dt}x(t) = f(x(t))$$

*if for all  $x(0) \in S$  the solution  $x(t) \in S$  for  $t > 0$ . If  $x(0) \in S$  implies  $x(t) \in S$  for all  $t \in R$  then we say that  $S$  is invariant.*

**Definition 1.7** [8] *The set  $S \subset R^n$  is said controlled invariant for the system*

$$\begin{aligned} \frac{d}{dt}x(t) &= f(x(t)) \\ y(t) &= g(x(t)) \end{aligned}$$

*if there exists a continuous feedback control law*

$$u(t) = \Phi(y(t))$$

*which assures the existence and uniqueness of the solution on  $R^+$  and it is such that  $S$  is positively invariant for the closed loop system.*

**Remark 1.2** *As noted in [8], the formal definition of invariant, referring to time future and past, rarely finds use in engineering applications. In this thesis, when a set is said to be invariant, positive or controlled invariance is implied.*

The objective of this research is to identify a reasonable approximation to a maximal set, without regard for any *fixed* control law. The proposed approach specifies a structure for the set (ellipsoidal

or semi-ellipsoidal) *as well as for the control law* (state-feedback), then searches over the parameter space (matrix elements) of both the set *and* the control law to find maximal sets subject to the constraints of invariance, bounded states and input, etc. However, performance requirements (settling time, etc.) are not incorporated into the problem formulation; the state-feedback structure is assumed *solely to prove that there exists an admissible control making the set invariant*. In fact, response characteristics under the computed state-feedback law may be far from ideal even though admissibility of the trajectories is ensured, so the computed control law is simply discarded.

Adhering to the definitions of Blanchini, the algorithms presented in this thesis are technically solving a positive invariance problem. However, the only information of interest is the set itself, which can be said to be controlled invariant since there is known to exist at least one control law (the discarded state feedback rule) making the set positively invariant in the closed-loop, all of which adds unnecessary confusion to the discussion. For the sake of clarity, then, the phrase “controlled invariant” is used throughout, with the admission that this may not be in keeping with the strict definition in all instances.

**Remark 1.3** *If, for a given system, a state feedback control law has been computed which gives desirable performance characteristics, the Matlab code in the appendices could be easily adapted to solve the maximal (positively) invariant set problem for a fixed control law by allowing the algorithm to search over the set of ellipsoid/semi-ellipsoid parameters only.*

## 1.2 Outline of Thesis

The remainder of the thesis is organized as follows. Chapter 2 surveys the currently available literature related to control of constrained systems, motivates the problem addressed in this work, and highlights the contributions. Chapters 3 and 4 present the theory and examples of ellipsoidal set approximations, which provides the foundation for the semi-ellipsoidal sets. Chapter 5 presents the main contribution of the thesis, the semi-ellipsoidal approximating set, and Chapter 6 provides examples illustrating the application of this theory to second- and third-order systems. Chapter 7 makes some concluding remarks and comments on possible avenues for future research.

Several appendices are included to provide details of certain portions of the work. Appendix A addresses the existence of a solution to the pair of Ricatti equations for the controllable ellipsoid, while Appendices B and C derive the ellipsoid volume objective function and state constraint inequalities, respectively. Appendix D provides the analytic derivatives of the cost functions and constraints required for the Matlab algorithms of Appendices F and G. Finally, Appendix E presents a modified

semi-ellipsoidal state constraint inequality based on a variable structure rather than state feedback control.

## Chapter 2

# Literature Review

Research into the control of systems with state and input constraints can generally be classified into one of two broad categories: predictive schemes and set membership methods. Among those approaches falling into the first category are time-optimal and model predictive methods. Time-optimal control for constrained systems is a classic problem and has been investigated extensively (see, e.g., [30]). Using a model predictive scheme, Bemporad [5] proposes an alternative reference governor for discrete time systems. At each time step, an admissible control sequence is generated which minimizes a cost function measuring the sum squared error between the predicted system trajectory and the desired reference trajectory for  $N$  future time steps. Furthermore, the constraint is imposed that the system's state will remain inside the state bounds for at least the  $N$  time steps of the cost function. Instead of imposing a terminal state constraint, the author relaxes the constraint to membership in an ellipsoidal set. The primary drawback of this and other model predictive methods is the computational burden for predicting states at future time steps.

The set membership methods seek to identify *a priori* a subset of the state-space contained in the constraint set which is invariant under an admissible control. One of the earliest papers investigating the reachable, recoverable, and controllable regions of the state-space is the doctoral dissertation by LeMay [32], in which the maximal subsets for linear systems with input constraints (i.e., no state constraints) are computed using time-optimal trajectories. A large body of work by LeMay's doctoral committee chairman, Dr. Elmer G. Gilbert, attempts to extend the results to autonomous discrete time systems with both input and state constraints using polyhedral approximations to the maximal recoverable set. (Typically, only the recoverable set is treated in the literature, primarily because this is the set defining the safe range of operation.) Most notable is a paper co-authored by K. T. Tan [20], which identifies the recoverable set (or "output admissible set," as referred to

by the authors) for systems with state-feedback control. The set is characterized by taking the system forward in time and identifying those initial conditions which do not violate the constraints. Among the applications of this technology is a reference governor, which modifies the reference signal to the controller so that the system remains in the un-saturated region ([17], [18], [19]), and a multimode control, which switches the control law when necessary to ensure safety at the expense of performance ([31]).

Gutman and Cwikel [21], [22] investigated the use of polyhedra for discrete time non-autonomous systems. This work was extended in [29] and most recently in [35], in which an approach is proposed for discrete time systems based on level sets of time-optimal control. That is, the authors generate backward in time the set of all points (within the admissible range of the state variables) which can reach the origin in  $N$  steps, resulting in polyhedra characterizing each  $N$ th level set.

Bitsoris and Gravalou [7] adopt a linear programming approach to solving the constrained state and input problem for discrete time systems given an *a priori* specified, bounded convex polyhedral set,  $X_0$ , of initial conditions. Solution of the control problem lies in finding an admissible control for each vertex of  $X_0$ , then computing a control law for any other point as a linear combination of the vertices' control laws. However, this method does not (and cannot) guarantee that an admissible solution exists for each vertex.

Most relevant to the present note are the studies utilizing a Lyapunov or invariant ellipsoid approach. Suárez, et al., applies this method to linear systems with control constraints [44]. Among the first to employ the ellipsoid approach for systems with state constraints was Gutman and Haggander [23]. Similar works by Shewchun and Feron [42], [43] and Wredenhagen and Belanger [47] investigate the use of nested ellipsoids to regulate systems with bounds on the control and control rate. The state is moved from one ellipsoid to the next inner one using subsequently higher control gains, resulting in improved performance without saturating. Both papers rely on LQR theory to develop the control law, with the ellipsoid being the solution to the Riccati equation. The primary difference in the two papers is that, in the latter, the authors develop a recursive algorithm to compute the parameters for the nested ellipsoids, while, in the former, the authors rely on a linear matrix inequality (LMI) approach.

Hou and Michel [24] investigate the asymptotic and global stability of stable, linear, autonomous systems operating on the unit hypercube or some less restrictive partial state constraint set using ellipsoidal Lyapunov functions. However, the autonomous form is not assumed to be the closed loop structure resulting from state feedback, and, consequently, no conditions are placed on control effort.

To the author's knowledge, the work of LeMay [32] has not been formally extended to the case

of constrained input and control. However, the results of his study, as well as those of Mayne and Schroeder [35], suggest that the boundaries of the maximal regions are characterized by a combination of time-optimal (or maximal input) trajectories and the state constraints,  $G_i$ , themselves. Such a region would be difficult to characterize quantitatively for a generic system. Thus, approximations such as the polyhedra suggested by Gilbert, et al., or Mayne and Schroeder are more realistic for practical applications. Furthermore, Blanchini and Miani [9] have shown that the maximal set can be arbitrarily closely approximated by polyhedra. However, these approaches themselves become markedly more complex for systems higher than second or third order, due to the number of vertices needed to adequately describe maximal sets in higher dimensions. (As an indicator, see Table 1 of Mayne and Schroeder [35], which shows the total number of vertices for an  $N = 8$  level set design increasing from 78 for a second order system to 566 for a fifth order system, and note that each vertex requires  $n$  data points to describe, for a total of 156 and 1698 parameters, respectively.) Although some work has been done to minimize the number of vertices without degrading performance ([37]), data storage space remains an issue for many applications.

## 2.1 Motivation & Contributions

Blanchini makes the following statement in the conclusion of his survey paper on set invariance in control [8]:

The techniques based on ellipsoidal sets are *conservative*. This fact is well established in robustness analysis as well as in the determination of domains of attraction under constraints. Polyhedral sets provide non-conservative solutions but *they lead to computationally intensive algorithms*. This is one of the most serious troubles although the fast improving computer performances alleviate the problem.

...

We believe that there are still several open problems that are worth an investigation. For instance, we have seen that the only family of sets of practical use having a bounded complexity are the ellipsoids. For the reasons explained above it would be important to develop algorithms to find other classes of invariant sets to achieve a reasonable tradeoff between conservatism and complexity.

While computer performance has improved tremendously in recent years, numerous situations exist in which system resources (processing time, available storage space, etc.) are limited. For



such systems, the comments by Blanchini are even more applicable, and they motivate a need for approximations to the maximal sets with better conservativeness and complexity properties than those methods currently available.

For the primary contribution, it is proposed that the maximal sets be approximated by subsets of ellipsoids defined by the intersection of the ellipsoid itself and the state constraint sets. These *semi-ellipsoidal sets* are shown to be invariant via an admissible state-feedback control law. This approach addresses the concerns of Blanchini by offering an approximation to the maximal sets which is less conservative than the ellipsoidal approach but simpler than the polyhedral approach.

As a foundational step to the main result, the maximal sets are first approximated by invariant ellipsoids completely contained within the state constraint set. The more general parent problem, that of finding the largest invariant ellipsoid contained in a polyhedral set, has been formulated by Boyd, et al. [10, Sec. 5.2]. Consequently, the ellipsoidal approach is effectively an extension of this concept to the specific problem of control in the presence of state and input constraints.

In light of the published results, the research presented in this thesis offers the following contributions:

1. The ellipsoidal method provides approximations to the maximal recoverable, reachable, and controllable sets for continuous time systems with constraints on both the input and linear combinations of the states. The approximation is characterized by only  $n(n+1)/2$  parameters specifying an  $(n \times n)$  symmetric ellipsoid matrix.
2. The semi-ellipsoidal method provides approximations to the maximal recoverable, reachable, and controllable sets for continuous time systems with constraints on both the input and linear combinations of the states which are less conservative than the ellipsoidal approach but less complex than the polyhedral approach. The approximation is characterized by only  $n((2k+n)+1)/2$  parameters specifying an  $(n \times n)$  symmetric ellipsoid matrix and  $k$  state constraint matrices of dimension  $(n \times 1)$ .
3. The ellipsoidal and semi-ellipsoidal approximations transfer the problem of control with constrained states and input from one of point-in-time prediction to one of point-in-time set membership, greatly reducing the on-line computational burden.
4. In general, the ellipsoidal and semi-ellipsoidal methods are applicable for the constrained-input problem (i.e., no state constraints), as well. In certain cases, such as the recoverable set for a strictly stable system or the reachable set for a strictly unstable system, the algorithm will not converge, since the maximal set is the entire state-space.

# Chapter 3

## Ellipsoidal Sets

The objective of this chapter is twofold: (i) to extend, at least in a conservative sense, the concepts of recoverable, reachable, and controllable sets due to LeMay [32] to systems with state constraints and (ii) to provide a foundation for the semi-ellipsoidal sets to follow in Chapter 5. Consideration is first given to approximating the recoverable set, which can essentially be thought of as solving a constrained regulator problem. The reachable and controllable sets are then adapted from these results.

### 3.1 Recoverable Ellipsoidal Set

Suppose the maximal recoverable set for a particular system is as shown in Figure 3.1.

**Lemma 3.1** *The maximal recoverable set,  $S_v$ , is invariant under some control law,  $u_v(x)$ .*

*Proof.* Note that the maximal recoverable set is, by definition, complete (i.e., there are no points outside  $S_v$  which can reach the origin along some admissible trajectory using an admissible control signal,  $u_v(x)$ .) Suppose an admissible trajectory of some point in  $S_v$  passes outside  $\partial S_v$  using an admissible control. This implies that  $S_v$  is not complete, contradicting the definition. ■

**Lemma 3.2** *A sub-maximal recoverable set  $S_v \subset S_v \subseteq G$  satisfies the state and input constraints if  $S_v$  is invariant under some control law  $u_v(x)$ , where  $u_v \in \mathcal{U}$  for  $x \in S_v$ .*

*Proof.* Since  $S_v$  is invariant, then  $x(x_0, t) \in S_v$ ,  $t \in [0, \infty)$ , which implies that  $x(x_0, t) \in G$ ,  $t \in [0, \infty)$ . This, in turn, implies that  $u_v(x, t) \in \mathcal{U}$ ,  $t \in [0, \infty)$ . ■

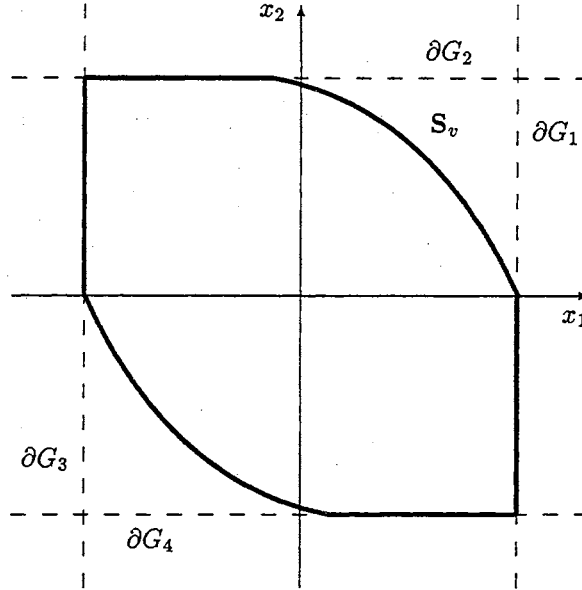


Figure 3.1: Maximal Recoverable Set.

The condition on  $S_v$  in Lemma 3.2 is sufficient but not necessary, since it may be possible, using another admissible control law,  $\hat{u}_v(x)$ , to generate another recovery trajectory which exits  $S_v$  for  $t \in [t_1, t_2]$  but remains inside  $G$ , as illustrated in Figure 3.2.

Suppose that the boundary of the sub-maximal set,  $\partial S_v$ , can be expressed as the level set of a Lyapunov function,  $V(x)$ . A necessary and sufficient condition for invariance of  $S_v$  is that (3.2) hold for some control law,  $u_v(x)$ , where  $\dot{V}$  denotes the time derivative of  $V$ .

$$\dot{V} \leq 0 \quad (3.1)$$

↓

$$\frac{dV}{dx} \frac{dx}{dt} \leq 0 \quad (3.2)$$

In general,  $V(x)$  and  $u(x)$  can be any of a number of functions (provided, of course, that the candidate  $V$  qualifies as a Lyapunov function). However, simplifying assumptions must be made to bring the generic problem into a tractable form. Specifically, the following structures are chosen, where  $P_v > 0$  implies that  $P_v$  is a positive definite matrix (Note:  $V$  satisfies the usual requirements for a Lyapunov function, in particular that it is positive definite):

$$V = x^T P_v x, P_v > 0 \quad (3.3)$$

$$u_v = -K_v x \quad (3.4)$$

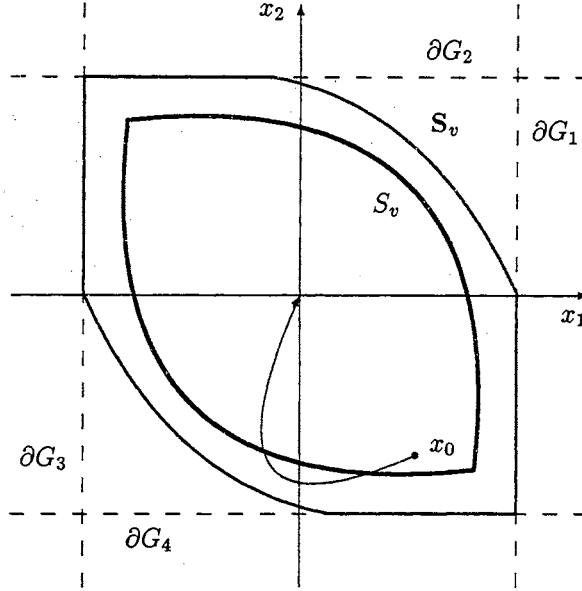


Figure 3.2: Approximate Maximal Recoverable Set.

Ellipsoids are quadratic functions of the states defined by positive definite matrices. Consequently, each level set of (3.3) defines an ellipse, as shown in Figure 3.3. In particular, attention may be focused (without loss of generality) on the level set  $V(x) = 1$ , leading to Problem 3.1.

**Problem 3.1** Find the largest ellipsoid,  $\bar{E}_v = \{x | x^T P_v x \leq 1\}$  and corresponding control law,  $u_v = -K_v x$ , such that (a)  $\bar{E}_v \subseteq G$ , (b)  $|u_v(x)| \leq \bar{u}$  for  $x \in \bar{E}_v$ , and (c)  $\bar{E}_v$  is a controlled invariant set for the system (1.1) and control law  $-K_v x$ .

**Remark 3.1** It is important to emphasize the fact that the computed state feedback gain,  $K_v$ , is used only as a tool to establish invariance of the maximal recoverable ellipsoid, and is not necessarily intended to serve as the implemented controller for the system. (The same holds for the discussion on the reachable and controllable sets to follow.)

**Remark 3.2** Suppose asymmetric constraints exist on a state (or input). Since the ellipsoid is symmetric and assumed centered on the origin, then its size is determined by the more restrictive condition, such that the asymmetric constraint effectively becomes a symmetric constraint. Consequently, state and input constraints are assumed symmetric for the duration of the thesis.

Invariance of the ellipsoid is ensured by forcing the time derivative of  $V$ , (3.5), to be negative semi-definite.

$$dV/dt = x^T P_v \dot{x} + \dot{x}^T P_v x$$

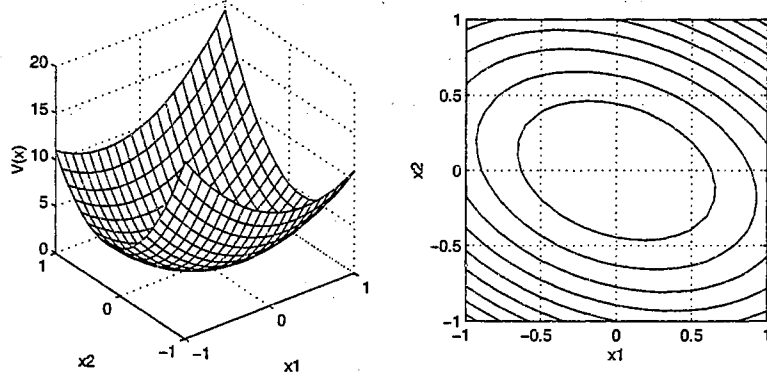


Figure 3.3: Ellipsoidal Level Sets of  $V$ .

$$\begin{aligned}
 &= x^T P_v (Ax + Bu_v) + (Ax + Bu_v)^T P_v x \\
 &= x^T P_v (A - BK_v) x + x^T (A - BK_v)^T P_v \\
 &= x^T \left[ P_v (A - BK_v) + (A - BK_v)^T P_v \right] x
 \end{aligned} \tag{3.5}$$

The requirement that this time derivative be negative semi-definite imposes the equivalent condition that the matrix in brackets be negative semi-definite, as shown in (3.6).

$$P_v (A - BK_v) + (A - BK_v)^T P_v \leq 0 \tag{3.6}$$

Note the similarity of (3.6) with the Control Algebraic Riccati Equation (CARE), shown in (3.7), along with the state feedback gain in (3.8).

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{3.7}$$

$$K = R^{-1}B^T P \tag{3.8}$$

Restricting the form of the state feedback matrix to (3.8) (as is done in [42], [48]) will result in a solution which, although optimal for the LQR problem, may not necessarily be “optimal” for the problem of finding the largest invariant set. Consequently, no such restrictions are placed on the form of the state feedback matrix.

Problem 3.1 can be formulated as a linear matrix inequality problem, or, more generally, as a constrained optimization problem. The following sections summarize the problem development.

### 3.1.1 Objective Function

For an ellipsoid (centered at the origin) defined by a given  $P_v$  matrix, it is known that the lengths of the principal axes of the ellipsoid are given by the eigenvalues of  $P_v^{-1}$ . Furthermore, the volume

of an ellipsoid is proportional to the product of these eigenvalues. Since the determinant of a matrix is equal to the product of its eigenvalues, (3.9) may be used as an objective function to find the largest ellipsoid, where the logarithm function is included to improve the search (See Appendix B for derivation.)

$$\max \log \det (P_v^{-1}) \quad (3.9)$$

### 3.1.2 Constraints

In this section, constraints on ellipsoid invariance, on the states, and on the control input are expressed as functions of the optimization parameters,  $P_v$  and  $K_v$ .

#### Invariance Constraint

To ensure that the ellipsoid is controlled invariant, the following matrix inequality is imposed. In comparison with (3.7), the parameter  $Q$  is set to zero. A positive-definite symmetric  $Q$  matrix forces a minimum decay rate on  $\dot{V}$ , yielding a certain level of robustness. For the present work, however, the robustness issue is not considered, hence the choice of  $Q$ .

$$P_v (A - BK_v) + (A - BK_v)^T P_v \leq 0 \quad (3.10)$$

Several options exist to test for negative semi-definiteness of matrices ([27]), most notably for this application that the eigenvalues must be negative semi-definite. Consequently, the constraint imposed is that the eigenvalues of the matrix  $L$ , where  $L$  is equal to the left hand side of (3.10), be less than or equal to zero (Note that they are guaranteed to be real since  $P_v$  is symmetric).

#### State Constraints

Constraints expressible as linear combinations of the states (as in (1.3)) can be written in the form of an inequality in the ellipsoid matrix,  $P_v$ , and constraint matrix,  $\Gamma_i$ , as derived in Appendix C.

$$\Gamma_i P_v^{-1} \Gamma_i^T \leq 1, \quad i = 1, \dots, k \quad (3.11)$$

#### Control Constraint

Since the control law is of the state feedback form, constraints on the input magnitude can be expressed as a modified version of (3.11).

$$K_v P_v^{-1} K_v^T \leq \bar{u}^2 \quad (3.12)$$

## Positive Definiteness Constraint

It has been noted that ellipsoids are described by positive definite, symmetric  $P_v$  matrices. Consequently, the following constraint is also imposed.

$$P_v > 0 \tag{3.13}$$

### 3.1.3 Implementation Issues

To demonstrate the concepts presented in this chapter, optimization routines were written for use with the Matlab software package (in particular, the function `constr.m`). The comments which follow describe several implementation problems encountered, as well as the solutions that were developed.

**Remark 3.3** *This constrained optimization problem fits nicely into the framework of the linear matrix inequality (LMI) problem (In fact, the text on LMI's by Stephen Boyd, et al., discusses application of the technology to finding an invariant ellipsoid for the constrained input problem [10].) In particular, freeware packages such as SDPSOL [49] and MAXDET [50] (specially developed for the maximum determinant problem, (3.9)), appear to be well suited and might be considered for future implementations of these algorithms.*

### Initial Conditions

Often, Matlab's search routine fails to find the optimal ellipsoid from a random initial guess of  $P_v$  and  $K_v$ . More consistent results are obtained by using the approach of Corollary A.2 (page 101) to initialize the search parameters. The resulting ellipsoid is scaled down (such that the ellipsoid boundary only comes to some fraction (say 90%) of the admissible control effort or state constraint bounds) and used as the initial condition for the combined state and input constrained problem.

### Unstable Systems

To improve the stability of the search for systems with positive eigenvalues, a nominal stabilizing state feedback control gain,  $K_v^0$ , is imposed. The search routine remains the same, with the exception of the following changes to (3.10) and (3.12).

$$P_v [A - B (K_v + K_v^0)] + [A - B (K_v + K_v^0)]^T P_v \leq 0 \tag{3.14}$$

$$(K_v + K_v^0) P_v^{-1} (K_v + K_v^0)^T \leq \bar{u}^2 \tag{3.15}$$

## 3.2 Reachable Ellipsoidal Set

Drawing from the previous results, the observation is made that a given ellipsoid is reachable if condition (3.16) holds for some control law  $u_e(x)$ .

$$dV/d\tau \leq 0, \tau = -t \quad (3.16)$$

In other words, if, in negative time, a set of points can reach a neighborhood of the origin under a given control law, then those same points can be reached from the neighborhood of the origin using the same control law in positive time. The problem is thus formally stated next.

**Problem 3.2** Find the largest ellipsoid,  $\bar{\mathcal{E}}_e = \{x|x^T P_e x \leq 1\}$  and corresponding control law,  $u_e = -K_e x$ , such that (a)  $\bar{\mathcal{E}}_e \subseteq G$ , (b)  $|u_e(x)| \leq \bar{u}$  for  $x \in \bar{\mathcal{E}}_e$ , and (c)  $\bar{\mathcal{E}}_e$  is a controlled invariant set for the system (1.1) and control law  $-K_e x$  in negative time.

Reevaluating the derivative of  $V$  with respect to  $\tau$  yields (3.17), the counterpart to (3.6).

$$-P_e(A - BK_e) - (A - BK_e)^T P_e \leq 0 \quad (3.17)$$

This relation serves as the invariance constraint (see (3.10)) for the reachable ellipsoid problem. All other inequalities remain the same, except that the nominal controller,  $K_e^0$ , must stabilize the system in negative time (i.e., it must stabilize the pair  $[-A, -B]$ ).

## 3.3 Controllable Ellipsoidal Set

A set is both recoverable and reachable if, for some control law(s), each point can both *reach* some neighborhood of the origin and *be reached from* some neighborhood of the origin along an admissible trajectory using an admissible control.

**Problem 3.3** Find the largest ellipsoid,  $\bar{\mathcal{E}}_c = \{x|x^T P_c x \leq 1\}$  and corresponding pair of control laws,  $u_v = -K_v x$ ,  $u_e = -K_e x$  such that (a)  $\bar{\mathcal{E}}_c \subseteq G$ , (b)  $|u_v(x)|, |u_e(x)| \leq \bar{u}$  for  $x \in \bar{\mathcal{E}}_c$ , and (c)  $\bar{\mathcal{E}}_c$  is a controlled invariant set for the system (1.1) and control law  $-K_v x$  in positive time and for the system (1.1) and control law  $-K_e x$  in negative time.

This problem is solved by imposing (3.10) and (3.17) on  $K_v, K_e$ , respectively. Existence of a solution to this pair of inequalities is treated in Appendix A.



### 3.4 Summary

The optimization problems for finding ellipsoidal approximations to the maximal recoverable, reachable, and controllable sets are summarized in Problems 3.4, 3.5, and 3.6, respectively. The elements of the ellipsoid and state feedback matrices are search parameters for the optimization routine, where the ellipsoid matrix is symmetric by construction.

#### Problem 3.4 (Recoverable Ellipsoidal Set)

Maximize

$$\log \det (P_v^{-1})$$

subject to

$$P_v (A - BK_v) + (A - BK_v)^T P_v \leq 0$$

$$-P_v^{-1} < 0$$

$$\Gamma_i P_v^{-1} \Gamma_i^T \leq 1, i = 1, \dots, k$$

$$K_v P_v^{-1} K_v^T \leq \bar{u}^2$$

#### Problem 3.5 (Reachable Ellipsoidal Set)

Maximize

$$\log \det (P_e^{-1})$$

subject to

$$-P_e (A - BK_e) - (A - BK_e)^T P_e \leq 0$$

$$-P_e < 0$$

$$\Gamma_i P_e^{-1} \Gamma_i^T \leq 1, i = 1, \dots, k$$

$$K_e P_e^{-1} K_e^T \leq \bar{u}^2$$

#### Problem 3.6 (Controllable Ellipsoidal Set)

Maximize

$$\log \det (P_c^{-1})$$

subject to

$$P_c (A - BK_v) + (A - BK_v)^T P_c \leq 0$$

$$-P_c (A - BK_e) - (A - BK_e)^T P_c \leq 0$$

$$-P_c < 0$$

$$\Gamma_i P_c^{-1} \Gamma_i^T \leq 1, i = 1, \dots, k$$

$$K_v P_c^{-1} K_v^T \leq \bar{u}^2$$

$$K_e P_c^{-1} K_e^T \leq \bar{u}^2$$

## Chapter 4

# Ellipsoidal Set Examples

This chapter presents example test cases which illustrate the maximal ellipsoids for different types of systems. As the results are reviewed, the following general comments should be kept in mind:

- Ellipsoid shape tends to be biased toward system modes that are easier to operate along (a stable mode is easier to recover along than an unstable one, while an unstable mode is easier to reach along than a stable one). This means that recoverable ellipsoids will be larger than reachable ellipsoids for strictly stable systems, and vice versa for strictly unstable systems.
- The more difficult the ellipsoid is to achieve, the more tightly the trajectories will follow the bound (e.g., the trajectories of an unstable ellipsoid will decay very little in the recovering mode if the system is near its operating limit).

### 4.1 Second Order Systems

Contained in this section are the computed maximal controlled invariant ellipsoids for six different second-order, linear systems in control canonical form: stable, stable focus, marginally stable, unstable, unstable focus, and saddle. Neither the theoretical development nor the Matlab code requires canonical form. However, this structure does allow for an easier interpretation of the imposed constraints, namely, a  $\pm 1$  limit on both the “position” and “velocity” and a  $\pm 1$  limit on the control effort.

Trajectories (coming *from* the ellipsoid boundary when recovering, going *to* the ellipsoid boundary when reaching) are plotted for all three cases. Note that for the controllable ellipsoid plots, one (or both) of the sets of trajectories cycles about the ellipsoid (compare with individual recoverable and reachable plots). It is difficult to show control effort on the plots without losing clarity, so for

simplicity it is merely stated that  $\max_{\varepsilon} |Kx| \leq 1$  in all cases (Note: The objective is to obtain the largest ellipsoid, which may or may not mean that maximal control is used, although, generally speaking, the maximum observed control is near 1.)

#### 4.1.1 Stable

The eigenvalues of the following  $A$  matrix are  $-1, -3$ .

$$A = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.1, 4.2, and 4.3, respectively. Ellipsoid matrices are given in Equations (4.1), (4.2), and (4.3).

$$P_v = \begin{bmatrix} 1.0019 & 0.0432 \\ 0.0432 & 1.0019 \end{bmatrix} \left( K_v = \begin{bmatrix} -1.0000 & -2.5000 \end{bmatrix} \right) \quad (4.1)$$

$$P_e = \begin{bmatrix} 16.912 & 0.0000 \\ 0.0000 & 22.950 \end{bmatrix} \left( K_e = \begin{bmatrix} -1.0000 & -5.5000 \end{bmatrix} \right) \quad (4.2)$$

$$P_c = \begin{bmatrix} 16.914 & 0.0000 \\ 0.0000 & 22.947 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -2.2629 & -1.9541 \end{bmatrix} \\ K_e = \begin{bmatrix} -2.2629 & -4.0000 \end{bmatrix} \end{array} \right) \quad (4.3)$$

Since the system is stable, it is expected that all of the points in the state-space can recover to the origin, although it is not clear that this can be done along trajectories which do not violate the state constraints. The purpose of the input, then, is to try to maximize the ellipsoidal set of states whose recovering trajectories do not violate the constraints. The trajectories originating from the ellipse boundary illustrate this.

In contrast, the stability of the system naturally limits the set of states which can be reached from the origin, since the control input must fight the natural dynamics of the system to move the state away from this critical point. Comparing Figures 4.1 and 4.2, it is seen that, indeed, the diameter of the reachable ellipse is roughly 0.25 that of the recoverable ellipse. Also, it is seen that trajectories oscillate near the boundary of the ellipse, indicating that maximal effort is needed to reach these points.

Finally, since the points in the controllable ellipsoid must be both reachable and recoverable, it is expected to be approximately equal to the intersection of these two sets. In this case, since the reachable set is contained entirely inside the recoverable set (all reachable points are recoverable,

but not all recoverable points are reachable), the controllable set is nearly the same as the reachable set, though it may differ somewhat in shape to remain invariant for recovery.

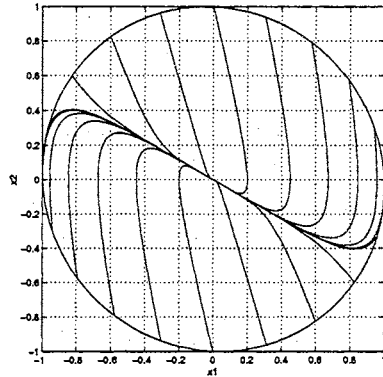


Figure 4.1: Recoverable Ellipsoid for Second Order Stable System.

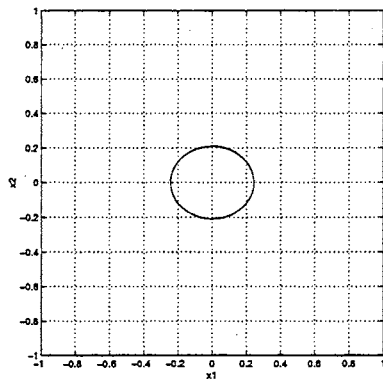


Figure 4.2: Reachable Ellipsoid for Second Order Stable System.

### 4.1.2 Stable Focus

The eigenvalues of the following  $A$  matrix are  $= -1 \pm 1.414i$ .

$$A = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.4, 4.5, and 4.6, respectively. Ellipsoid matrices are given in Equations (4.4), (4.5), and (4.6).

$$P_v = \begin{bmatrix} 1.0078 & 0.0888 \\ 0.0888 & 1.0078 \end{bmatrix} \left( K_v = \begin{bmatrix} -1.0000 & -0.5000 \end{bmatrix} \right) \quad (4.4)$$

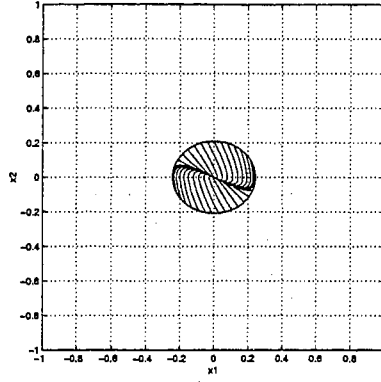


Figure 4.3: Controllable Ellipsoid for Second Order Stable System.

$$P_e = \begin{bmatrix} 9.3120 & 0.0000 \\ 0.0000 & 4.3404 \end{bmatrix} \left( K_e = \begin{bmatrix} -1.0000 & -3.5000 \end{bmatrix} \right) \quad (4.5)$$

$$P_c = \begin{bmatrix} 9.1717 & 0.0000 \\ 0.0000 & 4.4043 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -0.9172 & 1.9680 \end{bmatrix} \\ K_e = \begin{bmatrix} -0.9175 & -2.0000 \end{bmatrix} \end{array} \right) \quad (4.6)$$

The comments for the stable (negative real eigenvalue) system hold for the stable focus system as well. The eigenvalue locations for this system are generally closer to the imaginary axis, implying that a larger set of states can be reached from the origin, as seen in the figures.

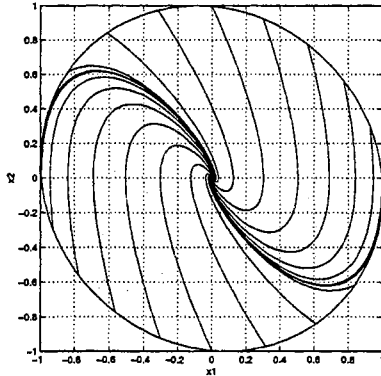


Figure 4.4: Recoverable Ellipsoid for Second Order Stable Focus System.

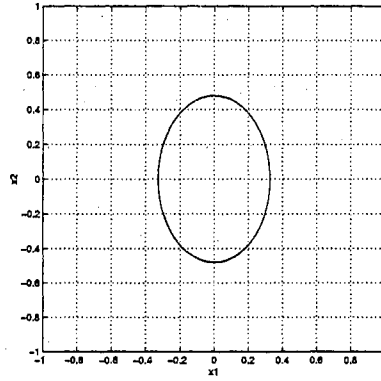


Figure 4.5: Reachable Ellipsoid for Second Order Stable Focus System.

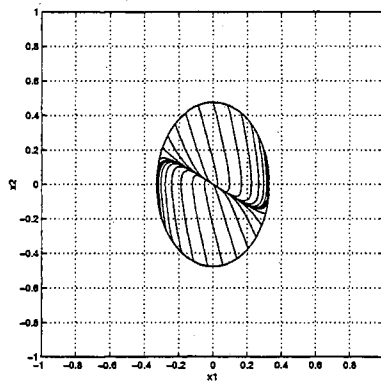


Figure 4.6: Controllable Ellipsoid for Second Order Stable Focus System.

### 4.1.3 Marginal

The eigenvalues of the following  $A$  matrix are 0, 0.

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.7, 4.8, and 4.9, respectively. Ellipsoid matrices are given in Equations (4.7), (4.8), and (4.9).

$$P_v = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} \left( K_v = \begin{bmatrix} 2.0000 & 1.5000 \end{bmatrix} \right) \quad (4.7)$$

$$P_e = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} \left( K_e = \begin{bmatrix} 2.0000 & -1.5000 \end{bmatrix} \right) \quad (4.8)$$

$$P_c = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 1.0000 & 0.0000 \end{bmatrix} \\ K_e = \begin{bmatrix} 1.0000 & 0.0000 \end{bmatrix} \end{array} \right) \quad (4.9)$$

Since the system's eigenvalues are on the imaginary axis, there is no bias towards the recoverable or reachable sets. Consequently, both sets are approximately the same size (the unit circle), the only difference being the direction of the spirals (one set of trajectories going to the origin, the other away from it). The slight change in the ellipsoid needed to create the controllable set dramatically alters the trajectories needed to make set invariant in both positive and negative time.

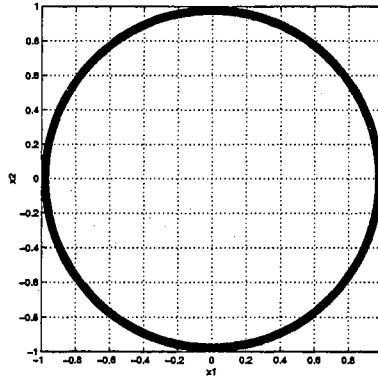


Figure 4.7: Recoverable Ellipsoid for Second Order Marginal System.

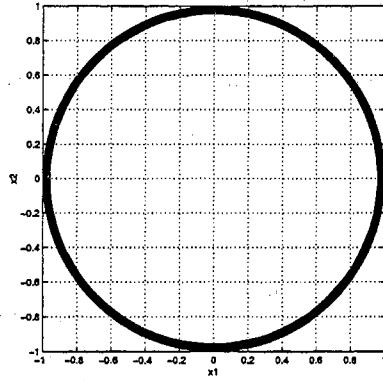


Figure 4.8: Reachable Ellipsoid for Second Order Marginal System.

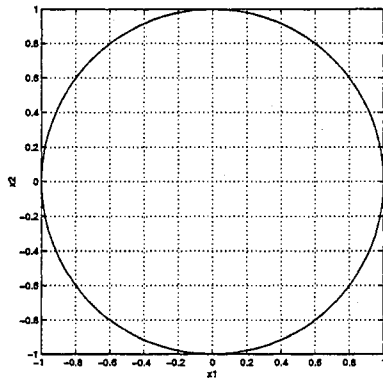


Figure 4.9: Controllable Ellipsoid for Second Order Marginal System.



#### 4.1.4 Unstable

The eigenvalues of the following  $A$  matrix are 1, 3.

$$A = \begin{bmatrix} 0 & 1 \\ -3 & 4 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.10, 4.11, and 4.12, respectively. Ellipsoid matrices are given in Equations (4.10), (4.11), and (4.12).

$$P_v = \begin{bmatrix} 16.910 & 0.0000 \\ 0.0000 & 22.953 \end{bmatrix} \left( K_v = \begin{bmatrix} -1.0000 & 5.5000 \end{bmatrix} \right) \quad (4.10)$$

$$P_e = \begin{bmatrix} 1.0019 & -0.0432 \\ -0.0432 & 1.0019 \end{bmatrix} \left( K_e = \begin{bmatrix} -1.0000 & 2.5000 \end{bmatrix} \right) \quad (4.11)$$

$$P_c = \begin{bmatrix} 16.912 & 0.0000 \\ 0.0000 & 22.950 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -2.2631 & 4.0000 \end{bmatrix} \\ K_e = \begin{bmatrix} -2.2628 & -2.7382 \end{bmatrix} \end{array} \right) \quad (4.12)$$

The reverse is true of the unstable system results compared with the stable system results. Here, the recoverable set is smaller than the reachable set.

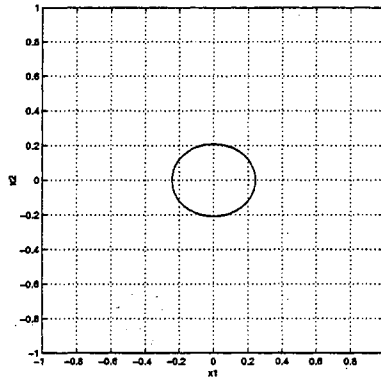


Figure 4.10: Recoverable Ellipsoid for Second Order Unstable System.

#### 4.1.5 Unstable Focus

The eigenvalues of the following  $A$  matrix are  $1 \pm 1.414i$ .

$$A = \begin{bmatrix} 0 & 1 \\ -3 & 2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

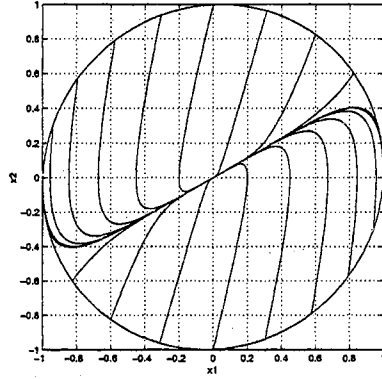


Figure 4.11: Reachable Ellipsoid for Second Order Unstable System.

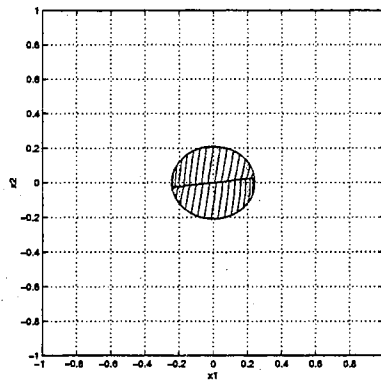


Figure 4.12: Controllable Ellipsoid for Second Order Unstable System.

The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.13, 4.14, and 4.15, respectively. Ellipsoid matrices are given in Equations (4.13), (4.14), and (4.15).

$$P_v = \begin{bmatrix} 9.2020 & 0.0000 \\ 0.0000 & 4.3895 \end{bmatrix} \left( K_v = \begin{bmatrix} -1.0000 & 3.5000 \end{bmatrix} \right) \quad (4.13)$$

$$P_e = \begin{bmatrix} 1.0078 & -0.0888 \\ -0.0888 & 1.0078 \end{bmatrix} \left( K_e = \begin{bmatrix} -1.0000 & 0.5000 \end{bmatrix} \right) \quad (4.14)$$

$$P_c = \begin{bmatrix} 9.2269 & 0.0000 \\ 0.0000 & 4.3778 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -0.8924 & 2.0000 \end{bmatrix} \\ K_e = \begin{bmatrix} -0.8924 & 0.8599 \end{bmatrix} \end{array} \right) \quad (4.15)$$

Again, the results can be contrasted to the stable focus system, where the recoverable set was larger than the reachable, opposite of what is seen here.

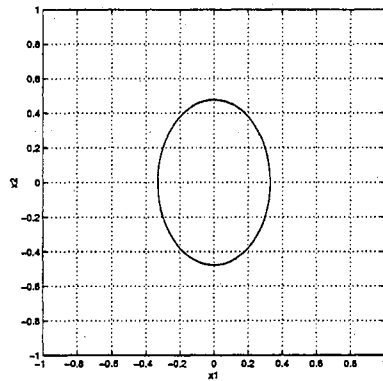


Figure 4.13: Recoverable Ellipsoid for Second Order Unstable Focus System.

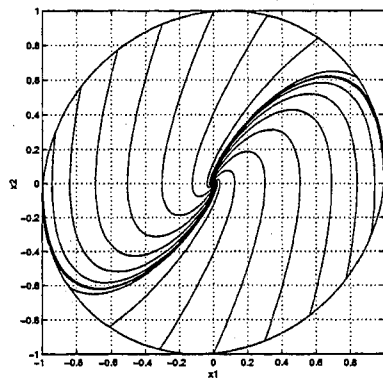


Figure 4.14: Reachable Ellipsoid for Second Order Unstable Focus System.

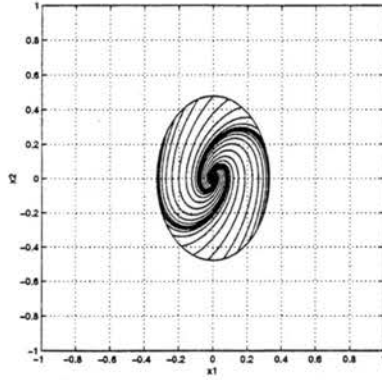


Figure 4.15: Controllable Ellipsoid for Second Order Unstable Focus System.

#### 4.1.6 Saddle

The eigenvalues of the following  $A$  matrix are 1,  $-3$ .

$$A = \begin{bmatrix} 0 & 1 \\ 3 & -2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.16, 4.17, and 4.18, respectively. Ellipsoid matrices are given in Equations (4.16), (4.17), and (4.18).

$$P_v = \begin{bmatrix} 9.0000 & 3.0000 \\ 3.0000 & 2.0000 \end{bmatrix} \left( K_v = \begin{bmatrix} 5.0000 & -0.5000 \end{bmatrix} \right) \quad (4.16)$$

$$P_e = \begin{bmatrix} 9.2744 & -8.7601 \\ -8.7601 & 9.2744 \end{bmatrix} \left( K_e = \begin{bmatrix} 5.0000 & -3.5000 \end{bmatrix} \right) \quad (4.17)$$

$$P_c = \begin{bmatrix} 16.955 & 0.0000 \\ 0.0000 & 22.892 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 3.7406 & -1.9590 \end{bmatrix} \\ K_e = \begin{bmatrix} 3.7407 & -2.0000 \end{bmatrix} \end{array} \right) \quad (4.18)$$

The results for this system are the most unique seen thus far. The eigenvectors of the system (and their associated eigenvalues) are  $v_1 = \begin{bmatrix} 0.707 & 0.707 \end{bmatrix}^T$  ( $\lambda_1 = 1$ ) and  $v_2 = \begin{bmatrix} -0.316 & 0.949 \end{bmatrix}^T$  ( $\lambda_2 = -3$ ). The recoverable set is strongly skewed along the stable eigenvector, while the reachable set is strongly skewed along the unstable eigenvector. The controllable set is significantly smaller than either of these, since the region of intersection is limited (Also, the trajectories seen in the controllable set's plot are for recovery, while those for reaching cycle close to the bound. This may be a consequence of the stable pole being "stronger" than the unstable pole due to the relative magnitudes. Consequently, less effort is required to recover from a state than to reach it.)

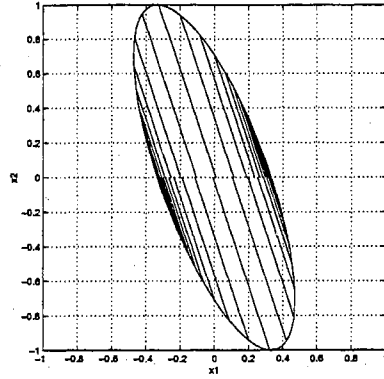


Figure 4.16: Recoverable Ellipsoid for Second Order Saddle System.

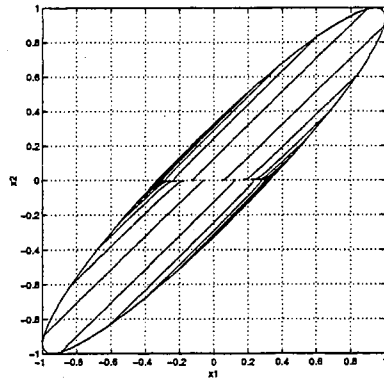


Figure 4.17: Reachable Ellipsoid for Second Order Saddle System.

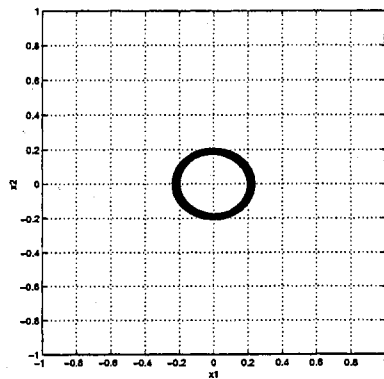


Figure 4.18: Controllable Ellipsoid for Second Order Saddle System.

## 4.2 Third Order Systems

Computational difficulty obviously increases with higher order systems, due to the increased number of parameters and constraints. However, the results for the third order systems in this section suggest that, in principle, the procedure is applicable to systems of any order, the only limitation being the increased computational load with higher dimensions and the stability of the optimization routine.

It is somewhat difficult to illustrate (in a limited number of figures) the nature of a third order ellipsoid and its associated trajectories. The figures in this section are of a quadrant of the three dimensional ellipsoid which allows for trajectories internal to the ellipsoid to be viewed. Additionally, the recovering and reaching trajectories are plotted in separate sub-figures (Note that the same quadrant is plotted for each, although some scaling differences may exist between the pairs.)

### 4.2.1 Stable

The eigenvalues of the following  $A$  matrix are  $-1, -1, -1$ .

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -2 & -1 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.19, 4.20, and 4.21, respectively. Ellipsoid matrices are given in Equations (4.19), (4.20), and (4.21).

$$P_v = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0012 \\ 0.0000 & 0.0012 & 1.0000 \end{bmatrix} \left( K_v = \begin{bmatrix} 0.0000 & 8.0000 & 2.0000 \end{bmatrix} \right) \quad (4.19)$$

$$P_e = \begin{bmatrix} 15.019 & -30.038 & -12.008 \\ -30.038 & 72.053 & 24.015 \\ -12.008 & 24.015 & 12.002 \end{bmatrix} \left( K_e = \begin{bmatrix} -6.0000 & 20.000 & 2.0000 \end{bmatrix} \right) \quad (4.20)$$

$$P_c = \begin{bmatrix} 16.841 & -33.683 & -13.546 \\ -33.683 & 78.893 & 27.092 \\ -13.546 & 27.092 & 13.151 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 1.2661 & -4.6526 & -1.4317 \end{bmatrix} \\ K_e = \begin{bmatrix} -3.0004 & 3.8809 & 2.0003 \end{bmatrix} \end{array} \right) \quad (4.21)$$

As with the second order system, the trajectories for the recoverable set draw towards the origin, while those of the reachable set appear to oscillate about the ellipsoid boundary (In this case, they

tend to outline the rest of the ellipsoid which was cut away for this figure.) The more restrictive of these two sets (recoverable/reachable) determines the size of the controllable ellipsoid. In this case, the controllable ellipsoid is approximately the size of the reachable ellipsoid, the limiting behavior further illustrated by the fact that the reaching trajectories oscillate about the ellipsoid boundary, while the recovering trajectories readily converge toward the interior. (Also, note that the reachable and controllable figures have been rotated 180° about the  $x_3$  axis relative to the recoverable ellipsoid to provide a better view of the trajectories.)

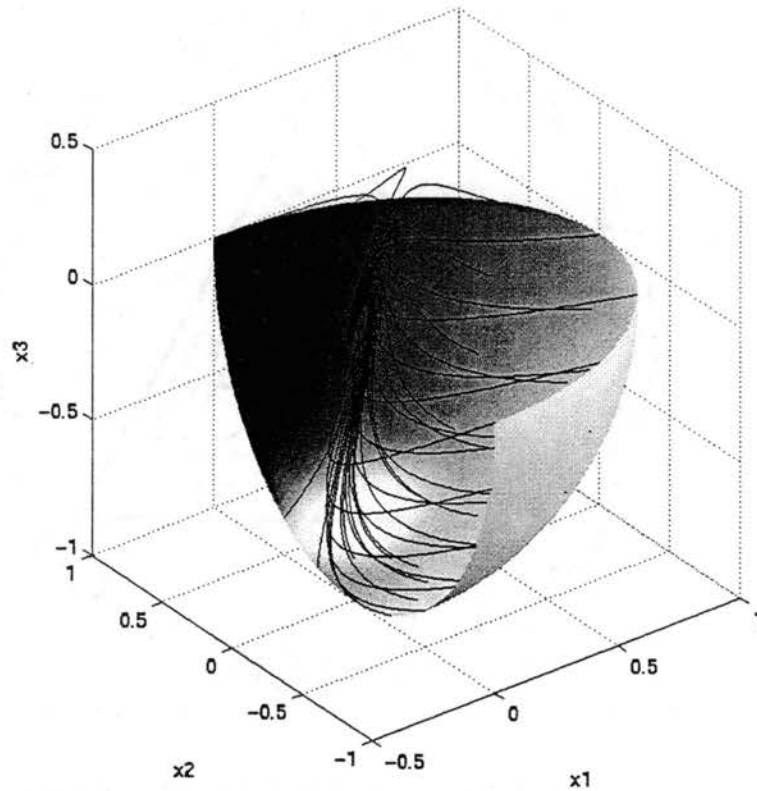


Figure 4.19: Recoverable Ellipsoid for Third Order Stable System.

#### 4.2.2 Unstable

The eigenvalues of the following  $A$  matrix are 1, 3, 5.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 15 & -23 & 9 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

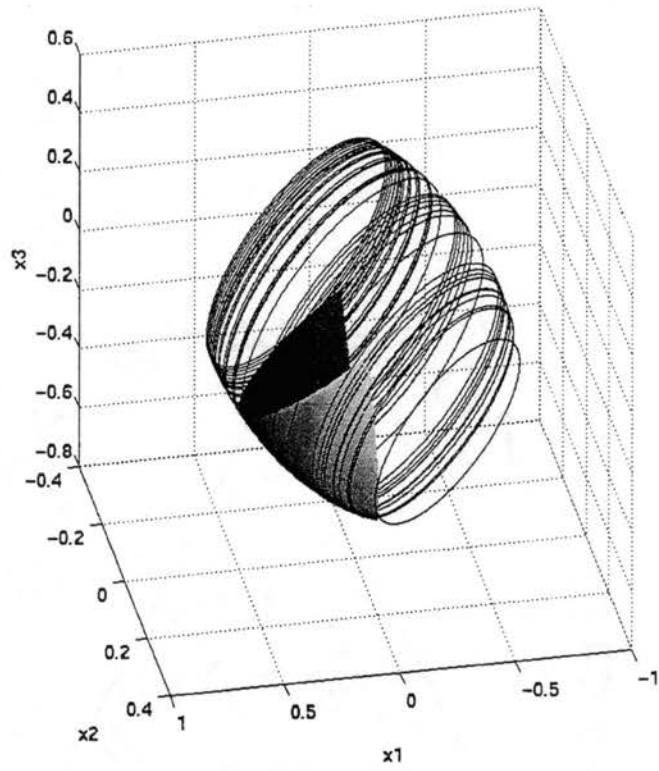


Figure 4.20: Reachable Ellipsoid for Third Order Stable System.

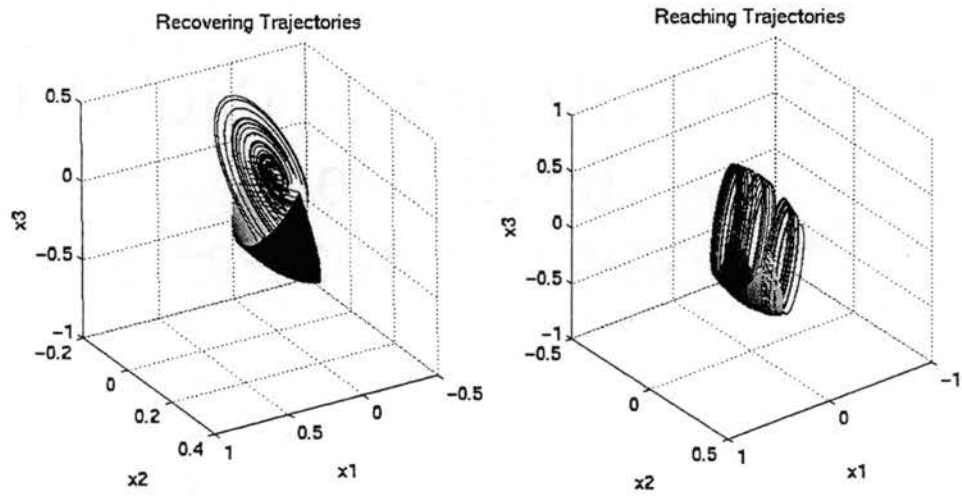


Figure 4.21: Controllable Ellipsoid for Third Order Stable System.



The ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 4.22, 4.23, and 4.24, respectively. Ellipsoid matrices are given in Equations (4.22), (4.23), and (4.24).

$$P_v = \begin{bmatrix} 676.08 & 0.0000 & 103.28 \\ 0.0000 & 847.32 & 0.0000 \\ 103.28 & 0.0000 & 145.22 \end{bmatrix} \left( K_v = \begin{bmatrix} 18.000 & -12.000 & 12.000 \end{bmatrix} \right) \quad (4.22)$$

$$P_e = \begin{bmatrix} 3.1951 & -3.7625 & 0.8911 \\ -3.7625 & 6.4608 & -1.4220 \\ 0.8911 & -1.4220 & 1.3169 \end{bmatrix} \left( K_e = \begin{bmatrix} 12.000 & -12.000 & 6.0000 \end{bmatrix} \right) \quad (4.23)$$

$$P_c = \begin{bmatrix} 686.92 & 0.0000 & 101.77 \\ 0.0000 & 853.81 & 0.0000 \\ 101.77 & 0.0000 & 141.57 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 15.000 & -16.250 & 9.0000 \end{bmatrix} \\ K_e = \begin{bmatrix} 11.006 & -16.250 & 3.4437 \end{bmatrix} \end{array} \right) \quad (4.24)$$

The results for the unstable system are opposite the results for the stable system. Here, the recovering trajectories appear to oscillate about the ellipsoid boundary, while the reaching trajectories clearly emanate from the origin.

Contrasting the stable system's controllable ellipsoid, the unstable system's ellipsoid is restricted by the recovering trajectories, which tend to oscillate about the boundary. The reaching trajectories, however, readily expand from the interior.

### 4.2.3 Marginal

The eigenvalues of the following  $A$  matrix are 0,0,0.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The ellipsoidal approximations of the recoverable and reachable sets are given in Figures 4.25, 4.26, and 4.27, respectively. Ellipsoid matrices are given in Equations (4.25), (4.26), and (4.27).

$$P_v = \begin{bmatrix} 1.5000 & 0.7500 & 0.7500 \\ 0.7500 & 1.5000 & 0.7500 \\ 0.7500 & 0.7500 & 1.5000 \end{bmatrix} \left( K_v = \begin{bmatrix} 3.0000 & 11.000 & 3.0000 \end{bmatrix} \right) \quad (4.25)$$

$$P_e = \begin{bmatrix} 1.5000 & -0.7500 & 0.7500 \\ -0.7500 & 1.5000 & -0.7500 \\ 0.7500 & -0.7500 & 1.5000 \end{bmatrix} \left( K_e = \begin{bmatrix} -3.0000 & 11.000 & -3.0000 \end{bmatrix} \right) \quad (4.26)$$

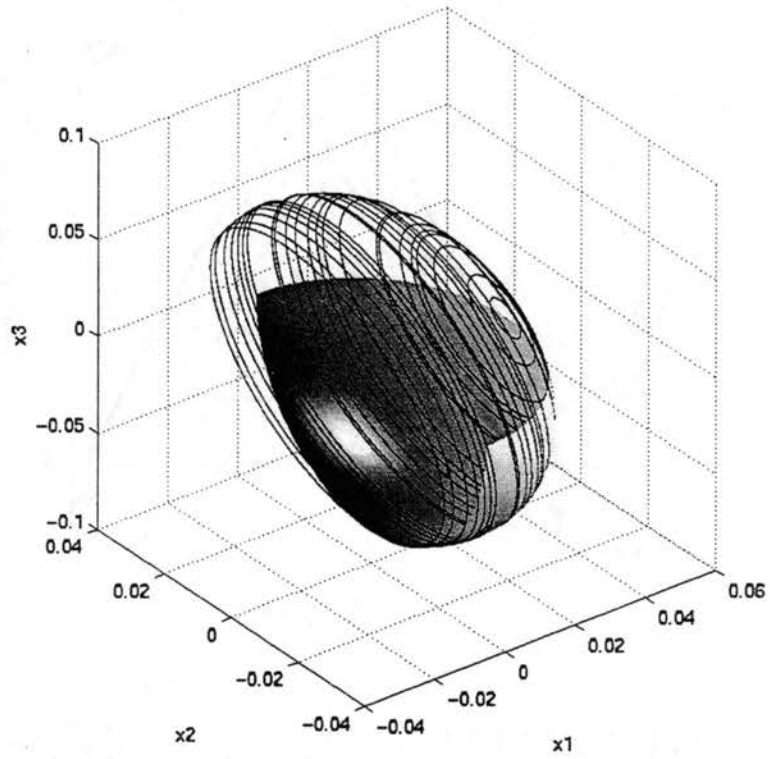


Figure 4.22: Recoverable Ellipsoid for Third Order Unstable System.

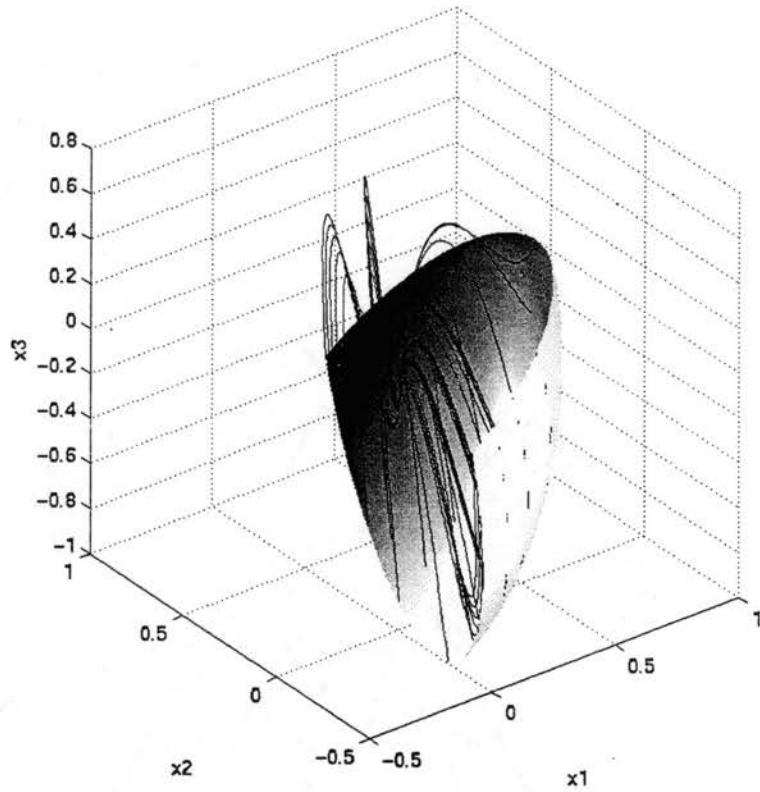


Figure 4.23: Reachable Ellipsoid for Third Order Unstable System.

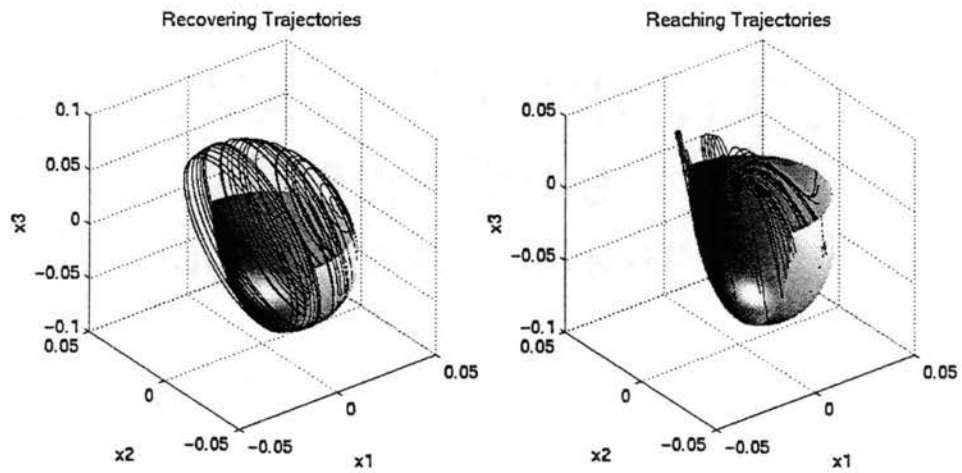


Figure 4.24: Controllable Ellipsoid for Third Order Unstable System.

$$P_c = \begin{bmatrix} 2.0001 & 0.0000 & 1.5875 \\ 0.0000 & 1.5873 & 0.0000 \\ 1.5875 & 0.0000 & 2.5200 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 0.0036 & 1.2599 & 0.0057 \end{bmatrix} \\ K_e = \begin{bmatrix} -0.0009 & 1.2599 & -0.0014 \end{bmatrix} \end{array} \right) \quad (4.27)$$

In both the recovering and reaching sets, the trajectories tend to a plane where they oscillate about the origin (in the reaching case, tend *from* a plane), although the shape of the ellipsoid and the position of this plane differ. The controllable ellipsoid is restricted by neither the recovering nor reaching conditions, with both sets of trajectories oscillating about the bound.

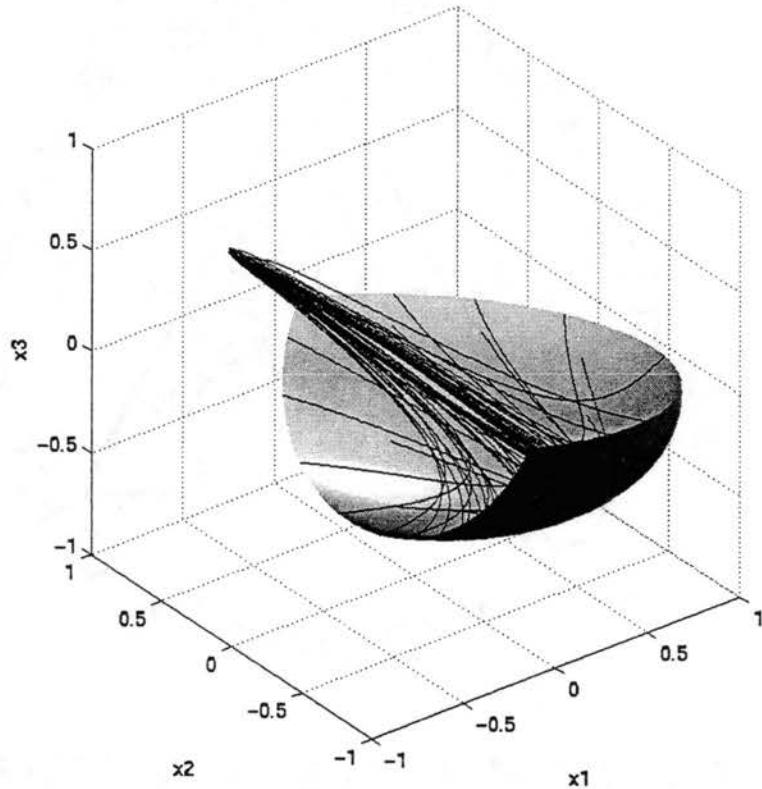


Figure 4.25: Recoverable Ellipsoid for Third Order Marginal System.

#### 4.2.4 Mixed

The eigenvalues of the following  $A$  matrix are  $-1.28, 0.14 \pm 1.53i$ .

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 & -2 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

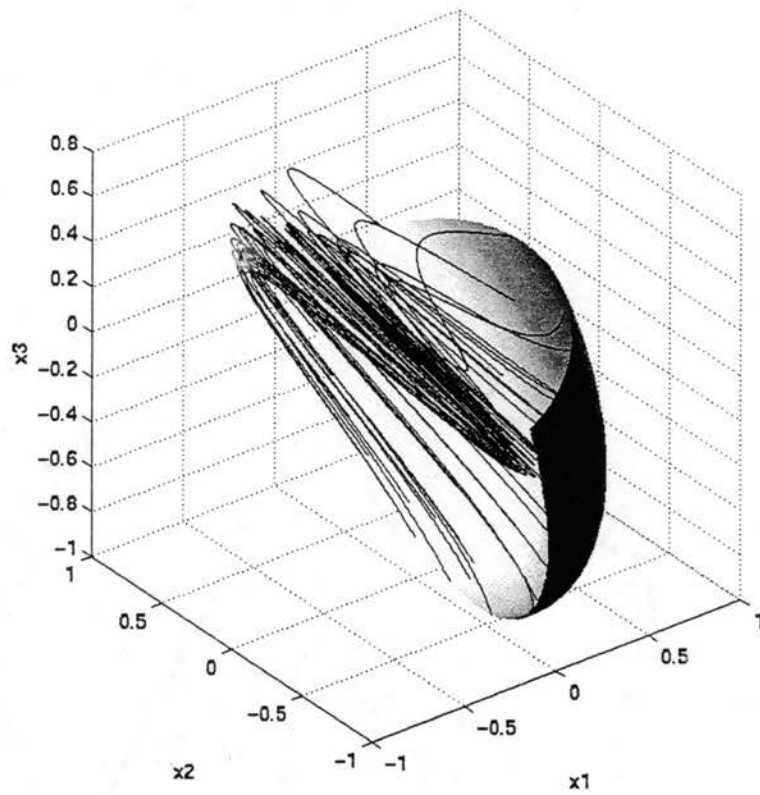


Figure 4.26: Reachable Ellipsoid for Third Order Marginal System.

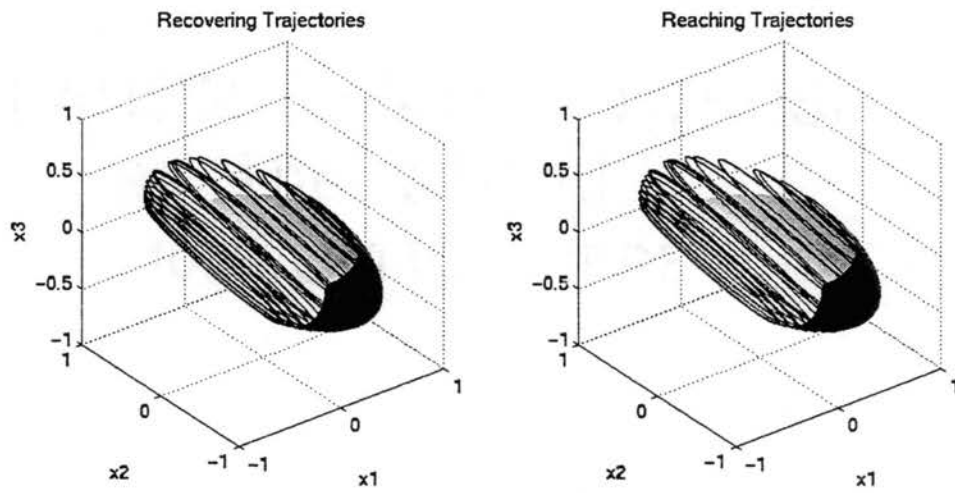


Figure 4.27: Controllable Ellipsoid for Third Order Marginal System.

The ellipsoidal approximations of the recoverable and reachable sets are given in Figures 4.28, 4.29, and 4.30, respectively. Ellipsoid matrices are given in Equations (4.28), (4.29), and (4.30).

$$P_v = \begin{bmatrix} 1.8577 & 1.2586 & 0.7255 \\ 1.2586 & 1.9398 & 0.7993 \\ 0.7255 & 0.7993 & 1.3705 \end{bmatrix} \left( K_v = \begin{bmatrix} 0.0000 & 9.0000 & 2.0000 \end{bmatrix} \right) \quad (4.28)$$

$$P_e = \begin{bmatrix} 9.6965 & -1.5952 & 4.6095 \\ -1.5952 & 1.9425 & -0.9067 \\ 4.6095 & -0.9067 & 3.2044 \end{bmatrix} \left( K_e = \begin{bmatrix} -6.0000 & 9.0000 & -4.0000 \end{bmatrix} \right) \quad (4.29)$$

$$P_c = \begin{bmatrix} 10.822 & 0.0000 & 5.0437 \\ 0.0000 & 2.1458 & 0.0000 \\ 5.0437 & 0.0000 & 3.3505 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -1.0255 & 0.1458 & 0.3117 \end{bmatrix} \\ K_e = \begin{bmatrix} -3.0000 & 0.1458 & -1.0000 \end{bmatrix} \end{array} \right) \quad (4.30)$$

Since the system contains both stable and unstable eigenvalues, neither the recoverable nor reachable trajectories restrict the ellipsoid in its entirety. Rather, it appears that they tend to restrict particular modes. Along an unstable mode, for example, it is “easier” to move away from the origin than towards it. Thus, in the direction of the unstable mode, it is expected that the range of recoverable trajectories should be more restrictive than the range of reachable trajectories, and vice versa for stable modes.

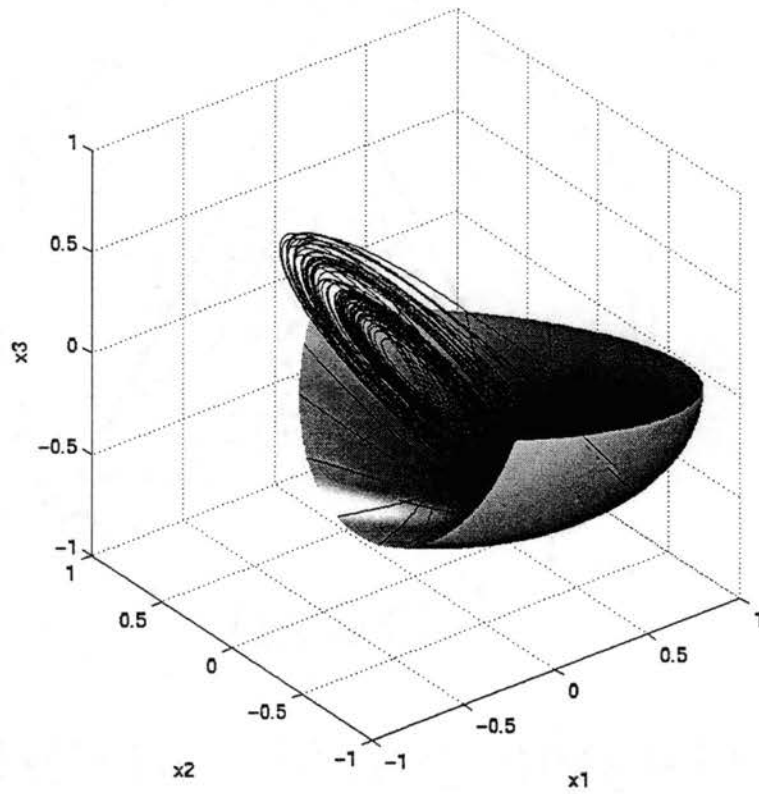


Figure 4.28: Recoverable Ellipsoid for Third Order Mixed System.

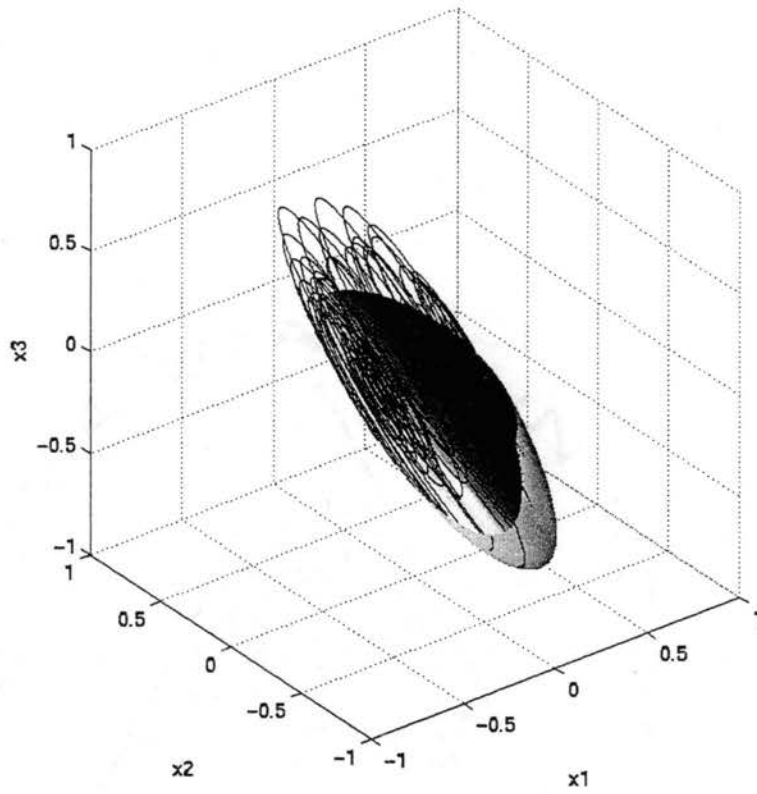


Figure 4.29: Reachable Ellipsoid for Third Order Mixed System.

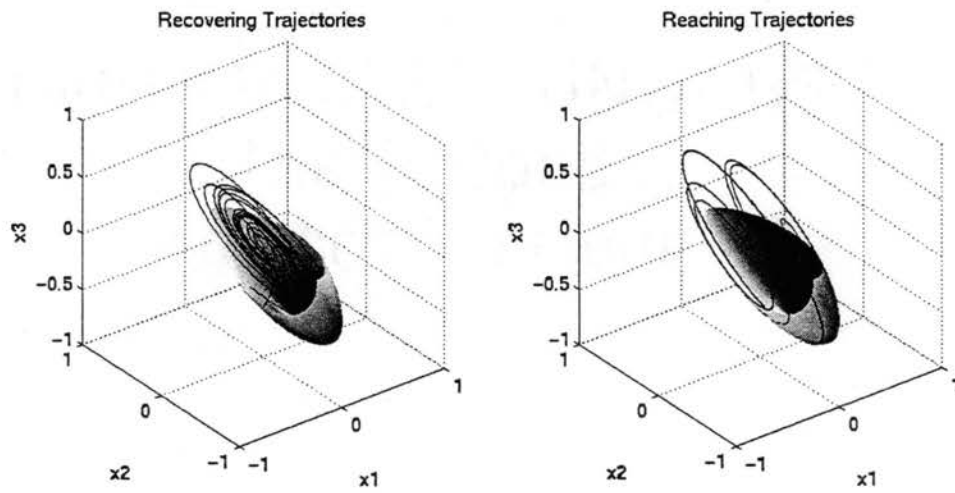


Figure 4.30: Controllable Ellipsoid for Third Order Mixed System.



### 4.3 Comparison Against Published Result

As mentioned in Chapter 2, the maximal set,  $S$ , can be approximated to any degree of accuracy using the polyhedral approach. Though it is also known that the ellipsoidal approach is inherently conservative, it is informative to provide an example illustrating this fact.

In [22], Gutman and Cwikel find a polyhedral approximation to the maximal (recoverable) set for the double integrator system, (4.31), where state constraints are  $x_1 \in [-25, 25]$ ,  $x_2 \in [-5, 5]$ , and the control constraint is  $u \in [-1, 1]$ . (For the purpose of discussions to follow, the states are treated as having units of (m) and (m/s), respectively).

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \quad (4.31)$$

Since the polyhedral approach requires a discrete time model, (4.31) was sampled with a period of  $T_s = 1$ (s), to give the discrete time model, (4.32).

$$\dot{x} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u \quad (4.32)$$

Using the algorithm presented in their paper, the authors find a maximal (recoverable) polygon with vertices given in (4.33).

$$X_{max} = \{\pm (12.5, 5), \pm (15, 4.5), \pm (19, 3.5), \pm (22, 2.5), \pm (24, 1.5), \pm (25, 0.5), \pm (25, -5)\} \quad (4.33)$$

Applying the approach of the preceding chapter, the recoverable ellipsoidal set is defined by (4.34).

$$P_v = \begin{bmatrix} 0.0016 & 0.0000 \\ 0.0000 & 0.0400 \end{bmatrix} \left( K_v = \begin{bmatrix} 2.0000 & 1.5000 \end{bmatrix} \right) \quad (4.34)$$

Figure 4.31 shows the vertices of the polygon (connected by a dotted line) and the ellipsoidal set. As expected, the ellipsoidal method does not capture as much of the state-space as the polyhedral method, particularly in the upper-left and lower-right quadrants, highlighting the conservativeness of the approach.

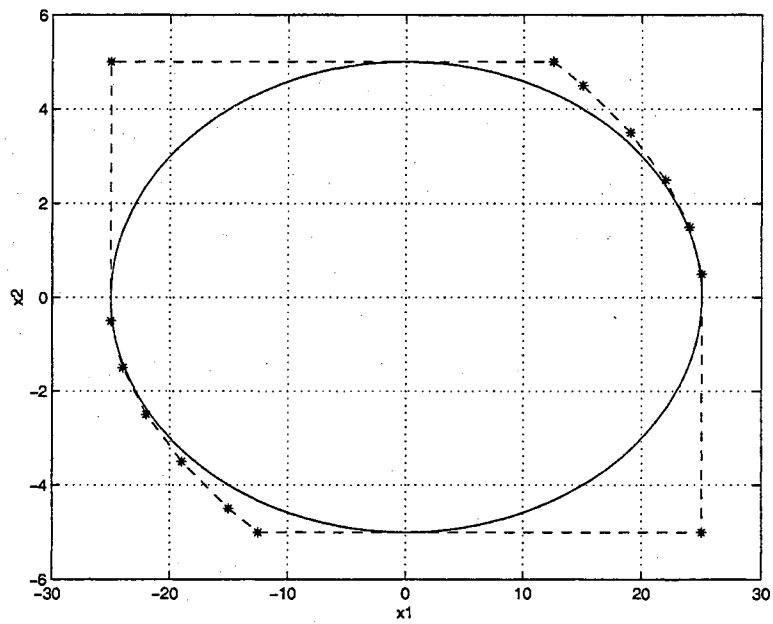


Figure 4.31: Comparison of Ellipsoidal and Polyhedral Methods for Double Integrator System.

## Chapter 5

# Semi-Ellipsoidal Sets

In this chapter, it is shown that, under certain conditions, the subset of an invariant ellipsoid satisfying the state constraints is also invariant (as illustrated in Figure 5.1 for state constraints of  $\pm 1$ ). This subset potentially provides a better approximation of the maximal operating set without resorting to polyhedra. As with Chapter 3, the recoverable case is addressed first, the results of which naturally lead to the reachable and controllable cases.

**Remark 5.1** *Figures are used heavily in this chapter to illustrate the ideas presented in the theorems. As in previous chapters, constraints are assumed symmetric throughout, although generally only one side is shown for clarity of the figure.*

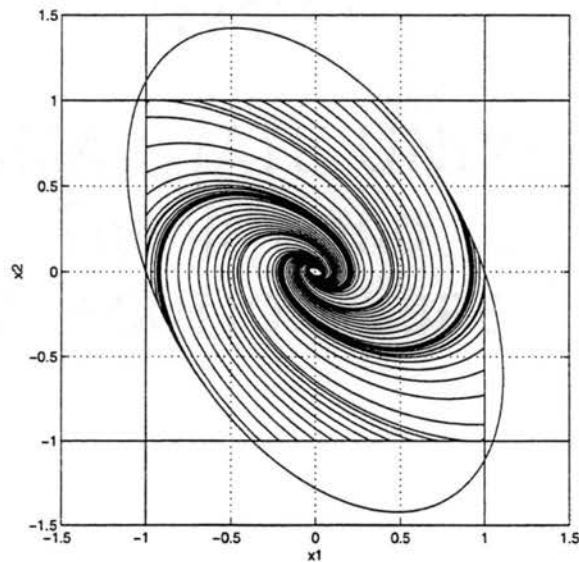


Figure 5.1: Invariant Subset of an Invariant Ellipsoid.

## 5.1 Recoverable Semi-Ellipsoidal Set

Consider a linear state constraint, (5.1),

$$G_i = \{x | \Gamma_i x \leq 1\} \quad (5.1)$$

and note that, to avoid violation of the constraint, states on the constraint boundary must satisfy the following condition.

$$\Gamma_i \dot{x} \leq 0 \quad (5.2)$$

This is stated formally in Lemma 5.1

**Lemma 5.1** *Given a linear state constraint,  $\Gamma_i x \leq 1$ , and a point,  $x_0$ , on the boundary of the constraint ( $\Gamma_i x_0 = 1$ ), the system trajectory satisfies the state constraint for  $x_0$  if and only if, for some control law,  $u_0$ ,  $\Gamma_i \dot{x}(x_0, u_0) \leq 0$ .*

The proof is intuitive, and stems from the fact that, if  $\Gamma_i \dot{x}(x_0, u_0) > 0$ , the function  $\Gamma_i x$  increases and the constraint is violated.

Figure 5.2, which shows a constraint bound,  $\partial G_i$ , as well as state trajectories emanating from that bound, illustrates the general concept. Since (5.2) is linear in  $x$ , there exists a point (more generally, a dimension  $(n - 1)$  hyperplane) on  $\partial G_i$  for which  $\Gamma_i \dot{x} = 0$ .

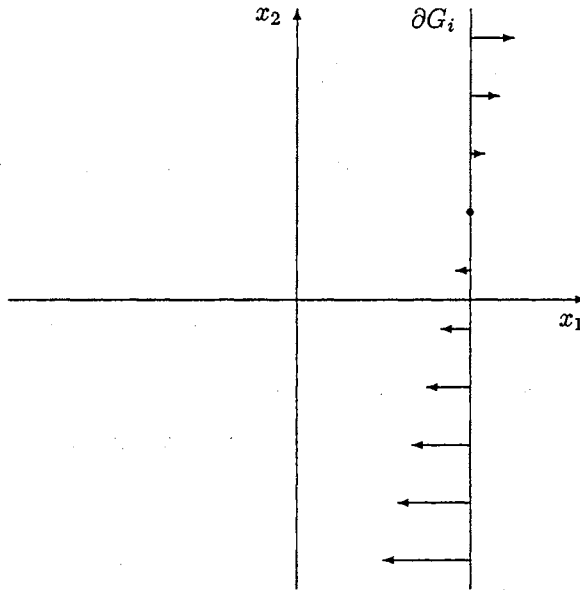


Figure 5.2: Illustration of Trajectories Emanating from State Constraint.

Suppose an ellipsoid  $\mathcal{E}$  exists which is controlled invariant under the state feedback control law  $u = -K_v x$ . Further, assume that  $\mathcal{E}$  is such that some of the states in the set violate a particular

state constraint, as shown in Figure 5.3. If, for the intersection of  $\mathcal{E}$  and  $\partial G_i$  (which may be the null set), relation (5.2) holds (which it does not, in this case), then the set defined by  $\mathcal{E} \cap G_i$  is also controlled invariant under  $u$ .

The objective, then, is to construct  $\mathcal{E}$  such that the subset  $\hat{S}_v$ , defined as the intersection of the ellipsoid and all state constraints, is controlled invariant under  $u$ . The procedure is to first develop the theory showing that an ellipsoid satisfying (5.2) for all regions of intersection with the constraints does indeed yield a controlled invariant set,  $\hat{S}_v$ , and then to apply the theory by defining rules for construction of such an ellipsoid, leading to the main result in Theorem 5.3.

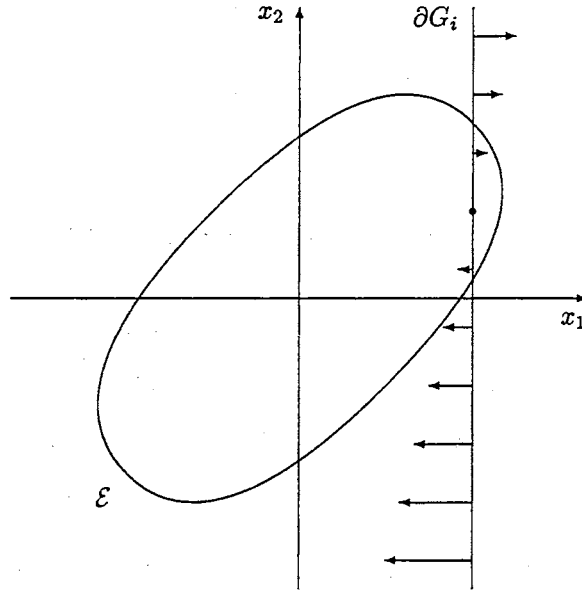


Figure 5.3: Illustration of Ellipsoid Violating a State Constraint.

### 5.1.1 Theory

For completeness, define the following subsets (see Figure 5.4) related to the derivative of the constraint boundary function (where  $u$  has been defined as the state feedback law,  $-K_v x$ , for the controlled invariant ellipsoid):

$$\partial \dot{G}_i = \{x | \Gamma_i (A - BK_v) x = 0\} \quad (5.3)$$

$$\dot{G}_i^+ = \{x | \Gamma_i (A - BK_v) x > 0\} \quad (5.4)$$

$$\dot{G}_i^- = \{x | \Gamma_i (A - BK_v) x < 0\} \quad (5.5)$$

These subsets define the acceptable regions of overlap (if any) of the ellipsoid and state constraints, as presented in Theorem 5.1.

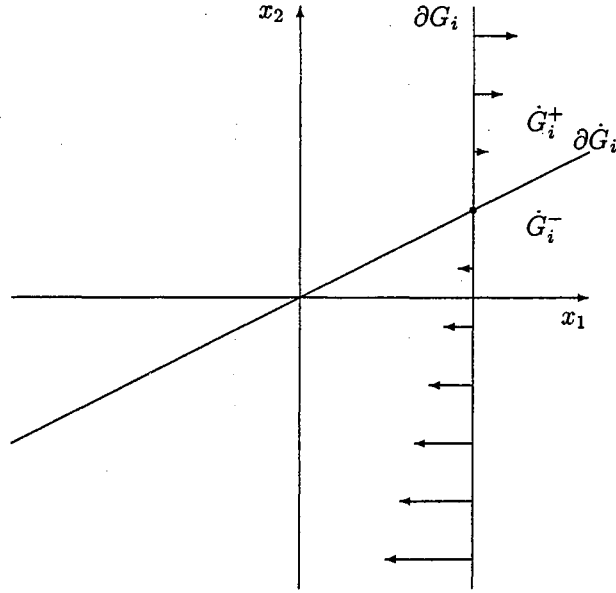


Figure 5.4: Derivative Function Subsets.

**Theorem 5.1** *Given an ellipsoid,  $\mathcal{E}$ , which is controlled invariant for system (1.1) under the control law  $u = -K_v x$ , and set of state constraints,  $G_i$  ( $i = 1, \dots, k$ ), whose intersection is  $G$ , define the subset of  $\mathcal{E}$  satisfying the state constraints as  $\hat{S}_v = \mathcal{E} \cap G$ . The set  $\hat{S}_v$  is controlled invariant under  $u$  if and only if  $\hat{S}_v \cap \partial G_i \cap \dot{G}_i^+ = \emptyset \forall i$ .*

Loosely interpreted, Theorem 5.1 says that the set  $\hat{S}_v$  cannot be invariant if, for any region of its boundary (particularly, those formed by the state constraints), the trajectories point “out.” In Figure 5.5, the region which is “unacceptable,”  $\partial G_i \cap \dot{G}_i^+$ , is given by the ray  $\vec{ba}$ , while the region which is “acceptable,”  $\partial G_i \cap (\partial \dot{G}_i \cup \dot{G}_i^-)$ , is given by the ray  $\vec{bc}$  (inclusive of point  $b$ ).

*Proof.* ( $\Rightarrow$ ) Suppose that for some constraint,  $G_i$ ,  $\hat{S}_v \cap \partial G_i \cap \dot{G}_i^+ \neq \emptyset$ . This implies that a portion of the boundary of  $\hat{S}_v$ ,  $\partial \hat{S}_v$ , is formed by a region of the state constraint boundary,  $\partial G_i$ , for which  $\frac{d}{dt}(\Gamma_i x) > 0$ , which further implies that the trajectories in this region violate the bound (i.e., do not remain within  $\hat{S}_v$ ), contradicting the assumption of invariance of  $\hat{S}_v$ .

( $\Leftarrow$ ) The trivial case here is  $\hat{S}_v \cap \partial G_i = \emptyset \forall i$  (i.e., the ellipsoid is entirely within the constraints), such that  $\hat{S}_v = \mathcal{E}$ . Since  $\mathcal{E}$  is invariant by construction, it follows that  $\hat{S}_v$  is invariant as well.

Assuming, then, that  $\hat{S}_v \cap \partial G_i \neq \emptyset \forall i$ , invariance must be shown for a set,  $\hat{S}_v$ , whose boundaries are defined by piecewise continuous regions of  $\partial \mathcal{E}$  and  $\partial G_i$ 's. Geometrically, this means that the time derivatives of  $x$  on the boundary must always be pointing “in.” Since the ellipsoid is controlled invariant under  $u = -K_v x$ , those portions of  $\partial \hat{S}_v$  belonging to  $\partial \mathcal{E}$  satisfy the condition  $\frac{d}{dt}(x^T P x) \leq$

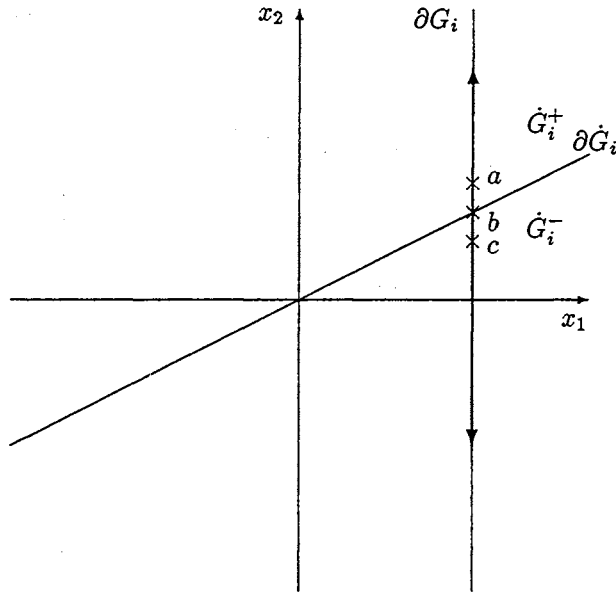


Figure 5.5: Acceptable and Unacceptable Regions of a Constraint Boundary.

0, implying that trajectories do not exit  $\hat{S}_v$  in regions defined by the boundary of the ellipsoid. Also, since  $\hat{S}_v \cap \partial G_i \cap G_i^+ = \emptyset \forall i$ , any portion of  $\hat{S}_v$  composed of a region of a state constraint bound,  $\partial G_j$  must satisfy  $\frac{d}{dt}(\Gamma_j x) \leq 0$ , implying that trajectories do not exit  $\hat{S}_v$  in these regions, either.

The last point of concern is those areas where the surface  $\partial \hat{S}_v$  is non-smooth (i.e., where regions of  $\partial \hat{S}_v$  are formed by the intersection of  $\partial \mathcal{E}$  and  $\partial G_j$ , or  $\partial G_j$  and  $\partial G_k$ , etc.) Since the same control law,  $u = -K_v x$ , is used everywhere, it follows that, for any combination of boundaries that might intersect,  $\frac{d}{dt}(f_1), \dots, \frac{d}{dt}(f_k) \leq 0$ , where  $f_i$  is any of the ellipsoidal or linear constraint functionals,  $x^T P x$ ,  $\Gamma_i x$ . Consequently,  $\hat{S}_v$  is invariant at these intersection areas, as well, completing the proof. ■

**Remark 5.2** *Although variable structure control could be used, in which separate regions of  $\hat{S}_v$  are made invariant using a different control law, it is generally more difficult to prove invariance of  $\hat{S}_v$ . For example, Figure 5.6 illustrates a section of  $\hat{S}_v$  where  $\partial \hat{S}_v$  is formed by two state constraint boundaries,  $\partial G_1$  and  $\partial G_2$ , with separate control laws defined for each. Although the trajectories of points lying on only one of the constraints satisfy the invariance requirement, the trajectory of the point lying at the intersection of the two does not (under either control law). This condition is avoided in Theorem 5.1 by using a single control law which simultaneously satisfies all constraints forming  $\hat{S}_v$ .*

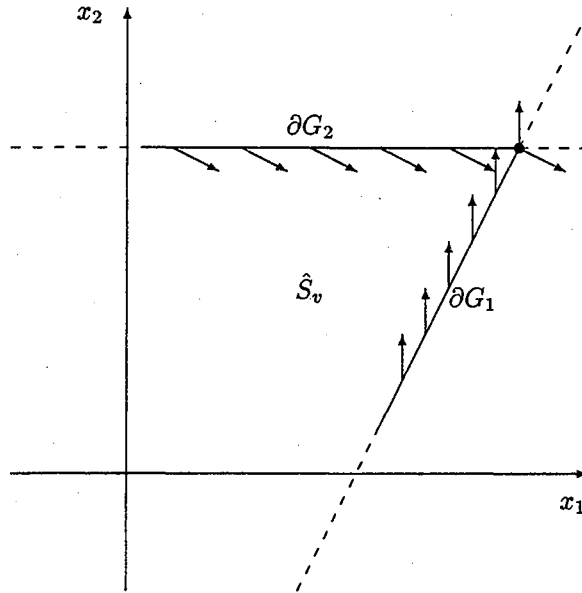


Figure 5.6: Variable Structure Control Failing Invariance of  $\hat{S}_v$ .

### 5.1.2 Design

Theorem 5.1 establishes the foundation by providing the *necessary and sufficient condition* for invariance of  $\hat{S}_v$ . Theorem 5.2 provides a *sufficient condition* for invariance of  $\hat{S}_v$ . Although more conservative, this leads to more feasible algorithms for the optimization routines.

**Theorem 5.2** *Given an ellipsoid,  $\mathcal{E}$ , which is controlled invariant for system (1.1) under the control law  $u = -K_v x$ , and set of state constraints,  $G_i$  ( $i = 1, \dots, k$ ), whose intersection is  $G$ , define the subset of  $\mathcal{E}$  satisfying the state constraints as  $\hat{S}_v = \mathcal{E} \cap G$ . The set  $\hat{S}_v$  is controlled invariant under  $u$  if  $\mathcal{E} \cap \partial G_i \cap \dot{G}_i^+ = \emptyset \forall i$ .*

The proof of this theorem is similar to that for Theorem 5.1, where the region of overlap was restricted to the points belonging to  $\hat{S}_v$ , not  $\mathcal{E}$ . Figure 5.7 illustrates the difference in the two theorems, in that  $\frac{d}{dt}(\Gamma_1 x) \leq 0$  must hold for points only on the line segment  $\overline{ab}$  for Theorem 5.1, but must hold for the segment  $\overline{ac}$  for Theorem 5.2. In the optimization routines to follow, this allows for a simple check on the ellipsoid matrix,  $P_v$ , rather than on the more complex set  $\hat{S}_v$ .

A relationship between  $\mathcal{E}$  and  $G$  is now sought to assist in the design of ellipsoids satisfying Theorem 5.2. As illustrated in Figure 5.5, the region of acceptable overlap of an ellipsoid and constraint is defined by  $\partial \hat{G}$ , (5.3). To determine the required relationship, then, three categories of constraints are identified by considering the properties of the matrix  $\Gamma_i (A - BK_v)$ . These categories are presented formally in Corollary 5.1, which addresses the case where trajectories lie in the hyperplane  $\partial G_i$ ,



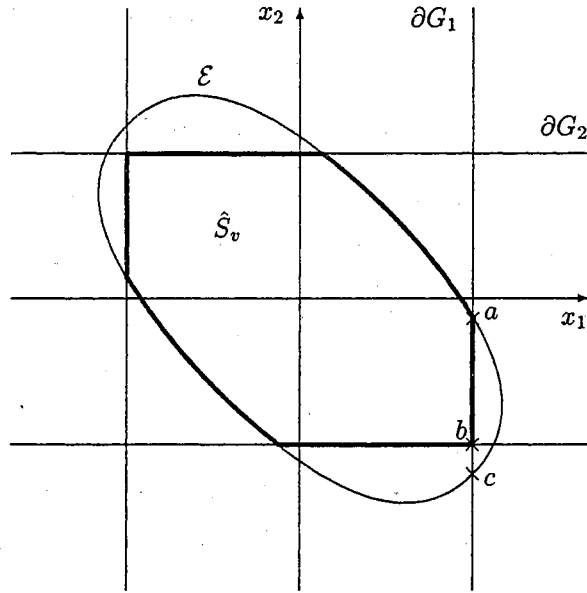


Figure 5.7: Comparison of Requirements for Theorems 5.1 and 5.2.

Corollary 5.2, which addresses the case where the trajectories are the same for the entire hyperplane  $\partial G_i$ , and Corollary 5.3, the most general case, in which trajectories are admissible for only certain regions of  $\partial G_i$ .

**Corollary 5.1** *If  $\Gamma_i(A - BK_v) = 0$ , then  $\mathcal{E} \cap G_i$  is controlled invariant for any ellipsoid  $\mathcal{E}$  which is controlled invariant under  $u = -K_v x$ .*

*Proof.* Here,  $\partial \dot{G}_i$  is  $\mathbb{R}^n$ . Consequently,  $G_i^+ = \emptyset$ , implying that  $\mathcal{E} \cap \partial G_i \cap G_i^+ = \emptyset$  for any  $\mathcal{E}$ , so that  $\mathcal{E} \cap G_i$  is controlled invariant if  $\mathcal{E}$  is controlled invariant. ■

**Corollary 5.2** *If  $\Gamma_i(A - BK_v) = \alpha_i \Gamma_i$  ( $\alpha_i$  a non-zero constant), then  $\mathcal{E} \cap G_i$  is controlled invariant for any ellipsoid  $\mathcal{E}$  which is controlled invariant under  $u = -K_v x$ .*

*Proof.* In this degenerate case,  $\partial \dot{G}_i$  is parallel to  $\partial G_i$ , so that  $\Gamma_i \frac{d}{dt} x$  has the same value (more importantly, the same sign) for all points on  $\partial G_i$ .

Since  $\Gamma_i(A - BK_v) = \alpha_i \Gamma_i$ , this implies that  $\Gamma_i$  is a left eigenvector and  $\alpha_i$  is an eigenvalue of  $A - BK_v$  [27, pg. 663]. If  $\alpha_i > 0$ , the closed loop system  $A - BK_v$  has at least one unstable mode, and  $(A - BK_v)^T P + P(A - BK_v) > 0$  for any  $P > 0$ , violating the invariance assumption on the ellipsoid. It therefore follows that  $\alpha_i \leq 0$ . Thus, for points on the state constraint boundary,  $\frac{d}{dt}(\Gamma_i x) = \alpha_i \Gamma_i x = \alpha_i \leq 0$  (since  $\Gamma_i x = 1$  on  $\partial G_i$ ), so that the constraint is met for any controlled invariant ellipsoid. ■

**Corollary 5.3** *If  $\Gamma_i(A - BK_v) \neq \beta\Gamma_i$  (where  $\beta$  is any real-valued constant) and  $\mathcal{E}$  is any ellipsoid which is controlled invariant under  $u = -K_v x$ , then  $\mathcal{E} \cap G_i$  is controlled invariant under  $u = -K_v x$  if and only if  $\text{int}\{\mathcal{E}\} \cap \partial G_i \cap \partial \dot{G}_i = \emptyset$ , where  $\text{int}\{\mathcal{E}\}$  denotes the interior of  $\mathcal{E}$ .*

The condition  $\Gamma_i(A - BK_v) \neq \beta\Gamma_i$  serves to eliminate from consideration those cases treated in Corollaries 5.1 and 5.2. Proof of this corollary requires a preliminary result presented in Lemma 5.2, which highlights a useful property of invariant ellipsoids in relation to an intersecting hyperplane. In this lemma, hyperplanes are considered in the general sense, but this is specialized in the proof of Corollary 5.3 to state constraints defined by hyperplanes.

**Lemma 5.2** *Suppose an ellipsoid,  $\mathcal{E}$ , where  $\mathcal{E} = \{x|x^T P x \leq 1\}$ , is controlled invariant under some control law,  $u = -Kx$ , and that the set is intersected by the hyperplane defined by  $\Gamma x = 1$ . Then it cannot be true that  $\frac{d}{dt}[\Gamma x(x_0, u)] > 0$  for all points satisfying  $\Gamma x_0 = 1$  and  $x_0 \in \mathcal{E}$ .*

*Proof.* To prove this lemma, it must be shown that (5.6) holds for at least one point  $x_0$  satisfying  $\Gamma x_0 = 1$  and  $x_0 \in \mathcal{E}$ .

$$\Gamma(A - BK)x_0 \leq 0 \quad (5.6)$$

Since the hyperplane intersects the ellipsoid, any number of points (or, a minimum of one point if the hyperplane just touches the boundary of the set) satisfy the conditions  $\Gamma x_0 = 1$  and  $x_0 \in \mathcal{E}$ . However, only existence need be proven, so consideration is limited to that point  $x_0$  where the hyperplane defined by  $\Gamma$  is tangent to a level set of the ellipsoid function, as in Figure 5.8 (i.e.,  $x_0$  satisfies  $\Gamma x_0 = 1$  and  $x_0^T P x_0 = \alpha$ , where  $0 < \alpha \leq 1$ ).

For the hyperplane to be tangent to a level set, the gradients of the hyperplane and ellipsoid functions must be parallel, as shown in the following relations.

$$\begin{aligned} \nabla(x^T P x) &\propto \nabla(\Gamma x) \\ &\Downarrow \\ 2P x_0 &= \beta \Gamma^T \\ &\Downarrow \\ x_0 &= \frac{\beta}{2} P^{-1} \Gamma^T \end{aligned}$$

Furthermore, since  $x_0$  must satisfy  $\Gamma x_0 = 1$ , it is found that  $\beta = 2 / (\Gamma P^{-1} \Gamma^T)$ , so that  $\beta$  is a positive constant (due to the fact that  $P$  is a positive definite matrix and  $\Gamma P^{-1} \Gamma^T$  is a quadratic form).

Since the ellipsoid is controlled invariant under  $u$ , the invariance inequality  $\frac{d}{dt}(x^T P x) \leq 0$  must

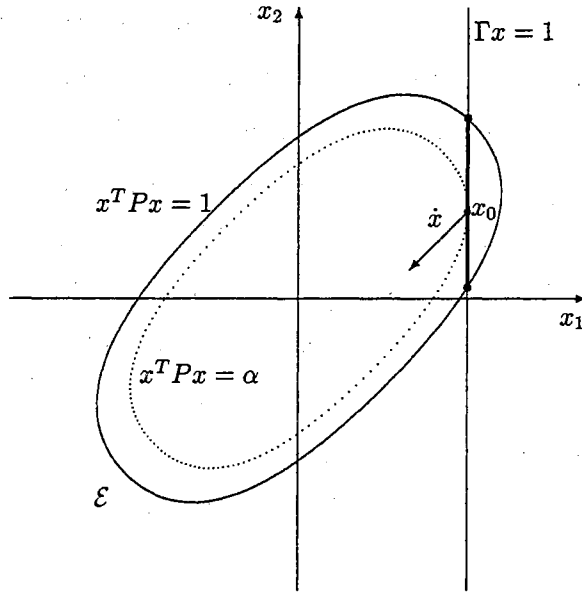


Figure 5.8: Illustration of Invariant Ellipse and Intersecting Hyperplane.

hold at  $x = x_0$  under the control law  $u$ , resulting in the following derivation.

$$\begin{aligned}
 & x_0^T P (A - BK) x_0 + x_0^T (A - BK)^T P x_0 \leq 0 \\
 & \quad \downarrow \\
 & \left( \frac{\beta}{2} P^{-1} \Gamma^T \right)^T P (A - BK) \left( \frac{\beta}{2} P^{-1} \Gamma^T \right) + \left( \frac{\beta}{2} P^{-1} \Gamma^T \right)^T (A - BK)^T P \left( \frac{\beta}{2} P^{-1} \Gamma^T \right) \leq 0 \\
 & \quad \downarrow \\
 & (P^{-1} \Gamma^T)^T P (A - BK) \left( \frac{\beta}{2} P^{-1} \Gamma^T \right) + \left( \frac{\beta}{2} P^{-1} \Gamma^T \right)^T (A - BK)^T P (P^{-1} \Gamma^T) \leq 0 \\
 & \quad \downarrow \\
 & \Gamma (A - BK) \left( \frac{\beta}{2} P^{-1} \Gamma^T \right) + \left( \frac{\beta}{2} P^{-1} \Gamma^T \right)^T (A - BK)^T \Gamma^T \leq 0 \\
 & \quad \downarrow \\
 & 2\Gamma (A - BK) \left( \frac{\beta}{2} P^{-1} \Gamma^T \right) \leq 0 \\
 & \quad \downarrow \\
 & \Gamma (A - BK) x_0 \leq 0
 \end{aligned}$$

Thus, it is shown that  $\frac{d}{dt} [\Gamma x(x_0, u)] \leq 0$  for the point  $x_0 = (\Gamma P^{-1} \Gamma^T)^{-1} P^{-1} \Gamma^T$ , completing the proof. ■

The proof of Corollary 5.3 can now be presented.

*Proof.* ( $\Rightarrow$ ) Assume  $\mathcal{E} \cap G_i$  is controlled invariant but that  $\text{int} \{ \mathcal{E} \} \cap \partial G_i \cap \partial \dot{G}_i = \Upsilon \neq \emptyset$ . It follows from the definition of  $\partial \dot{G}_i$  that, for any neighborhood of a point  $x_0 \in \Upsilon$ , there exist points in  $\text{int} \{ \mathcal{E} \} \cap \partial G_i$  which belong to the set  $\dot{G}_i^+$ , implying that the trajectories violate the constraint and that the set  $\mathcal{E} \cap G_i$  is not controlled invariant, contradicting the original assumption.

( $\Leftarrow$ ) As shown in Figure 5.9, three possibilities exist to describe the intersection of the ellipsoid with the constraint boundary: (i)  $\mathcal{E} \cap \partial G_i = \emptyset$  (they do not intersect), (ii)  $\mathcal{E} \cap \partial G_i \cap \dot{G}_i^- = \Upsilon_1 \neq \emptyset$  (the ellipsoid intersects the constraint in the region where derivatives of the constraint are negative), and (iii)  $\mathcal{E} \cap \partial G_i \cap \dot{G}_i^+ = \Upsilon_2 \neq \emptyset$  (the ellipsoid intersects the constraint in the region where derivatives of the constraint are positive). By Theorem 5.2, cases (i) and (ii) both result in  $\mathcal{E} \cap G_i$  being invariant. By Lemma 5.2, case (iii) cannot happen since  $\mathcal{E}$  is invariant. ■

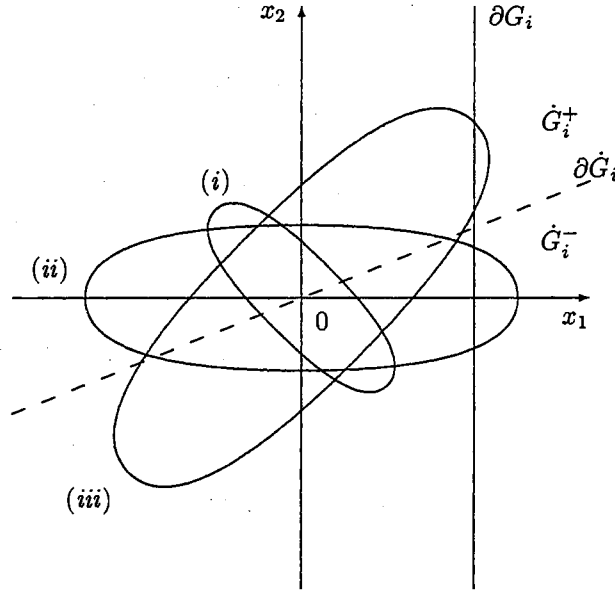


Figure 5.9: Illustration for Proof of Corollary 5.3.

### 5.1.3 Main Result

It has been established in the preceding section that state constraints giving the matrix conditions discussed in Corollaries 5.1 and 5.2 effectively impose no restriction on  $\mathcal{E}$ . Therefore, the final step is to develop a relationship between  $\Gamma_i$  and  $P_v$  such that Corollary 5.3 holds.

As Figure 5.10 illustrates for a second order system, the intersection of  $\partial \dot{G}_i$  with the ellipsoid and linear constraint bound defines a line segment and point, respectively, in  $R^2$  (more generally, dimension  $(n - 1)$  ellipsoid and hyperplane). Defining a new coordinate  $z_1 \in R^1$ , where  $z_1$  lies along

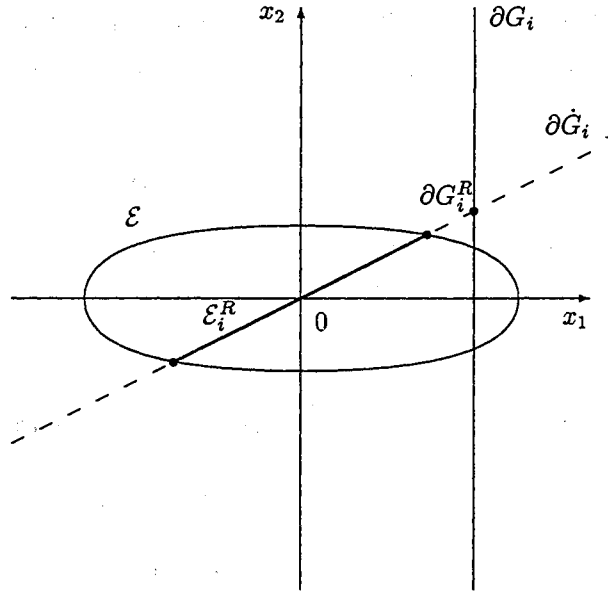


Figure 5.10: Illustration of Reduced Dimension Ellipsoid and State Constraint Boundary in  $R^2$ .

the line  $\partial\dot{G}_i$ , gives the “reduced dimension” ellipsoid,  $\mathcal{E}_i^R$ , and state constraint bound,  $\partial G_i^R$ , as shown in Figure 5.11.

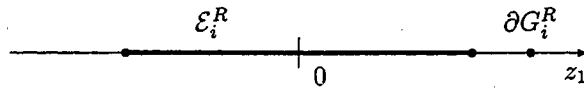


Figure 5.11: Illustration of Reduced Dimension Ellipsoid and State Constraint Boundary in  $R^1$ .

It is apparent from this example that the condition of Corollary 5.3 holds (i.e., no point in  $\partial G_i \cap \partial\dot{G}_i$  is contained in  $\text{int}\{\mathcal{E}\}$ ) if the reduced dimension ellipsoid does not violate the reduced dimension state constraint. Thus, if  $P_i^R$  and  $\Gamma_i^R$  are matrices defining the ellipsoid and constraint in the new coordinates, it follows that the result in (3.11) (cf. Appendix C) can be used to give the constraint (5.7) which satisfies Corollary 5.3 (For the remainder of this section, the subscript “ $v$ ” is omitted from the reduced-dimension ellipsoid matrix to avoid confusion with the state-constraint index.)

$$\Gamma_i^R (P_i^R)^{-1} (\Gamma_i^R)^T \leq 1 \quad (5.7)$$

To define the matrices  $P_i^R$  and  $\Gamma_i^R$ , the new coordinate system must first be identified. Since the boundary of the state constraint derivative,  $\partial\dot{G}_i$  is defined by the relation  $\Gamma_i (A - BK_v) x = 0$ , it follows that the null space of this matrix defines the vectors lying in the hyperplane,  $\partial\dot{G}_i$  (Note:

The minimum norm solution to this equation is 0. Also, an infinite number of vector sets could be chosen for  $W_i$ , but for computational purposes the orthonormal set is preferred, hence the condition  $W_i^T W_i = I$ ).

$$W_i = \text{null} \{ \Gamma_i (A - BK_v) \}, W_i^T W_i = I \quad (5.8)$$

Consequently, any vector,  $x^R$ , lying in this hyperplane is given by a linear combination of these vectors.

$$x^R = W_i z \quad (5.9)$$

The ellipsoid and state constraint can now be redefined in the new coordinate system:

$$\mathcal{E}_i^R = \{ z | z^T (W_i^T P_v W_i) z \leq 1 \} \quad (5.10)$$

$$G_i^R = \{ z | (\Gamma_i W_i) z \leq 1 \} \quad (5.11)$$

The ellipsoid and state constraint matrices for the reduced dimension coordinate system are taken as (5.12), (5.13).

$$P_i^R = W_i^T P_v W_i \quad (5.12)$$

$$\Gamma_i^R = \Gamma_i W_i \quad (5.13)$$

**Remark 5.3** For the case where  $\Gamma_i (A - BK_v) = \alpha_i \Gamma_i$  ( $\alpha_i$  a non-zero constant), it is apparent that  $W_i = \Gamma_i^\perp$  (where  $\Gamma_i \Gamma_i^\perp = 0$ ). Thus,  $\Gamma_i^R = \Gamma_i W_i = 0$ , so that  $\Gamma_i^R (P_i^R)^{-1} (\Gamma_i^R)^T = 0 \leq 1$  for any  $P_v$  defining a controlled invariant ellipsoid, which agrees with Corollary 5.2. The reduced-dimension inequality constraint, (5.7), can therefore be applied for the more general case  $\Gamma_i (A - BK_v) \neq 0$ .

The results of this section are summarized in the following theorem, the proof of which flows naturally from Theorem 5.2, Corollaries 5.1, 5.2, 5.3, and the preceding inequality derivation.

**Theorem 5.3** Given an ellipsoid,  $\mathcal{E}$ , controlled invariant for system (1.1) under  $u = -K_v x$ , and state constraints  $G_i$ , whose intersection is  $G$ , the set  $\hat{S}_v = \mathcal{E} \cap G$  is controlled invariant under  $u$  if, for  $\Gamma_i (A - BK_v) \neq 0$  ( $i = 1, \dots, k$ ), the following inequality holds:

$$\Gamma_i^R (P_i^R)^{-1} (\Gamma_i^R)^T \leq 1 \quad (5.14)$$

where

$$\Gamma_i^R = \Gamma_i W_i^v \quad (5.15)$$

$$P_i^R = (W_i^v)^T P_v W_i^v \quad (5.16)$$

$$W_i^v = \text{null} \{ \Gamma_i (A - BK_v) \} \quad (5.17)$$

### 5.1.4 Implementation Issues

The implementation issues discussed in Chapter 3 hold for the search for optimal semi-ellipsoidal sets, as well, as do the inequality constraints for invariance and positive definiteness of the ellipsoid matrix. However, the nature of  $\hat{S}_v$ 's construction requires modification of the objective function and control constraint.

#### Objective Function

Although the case may occur where  $\hat{S}_v = \mathcal{E}$  (the ellipsoid is entirely within the state constraints), the volume of  $\hat{S}_v$  is generally not equal to the volume of  $\mathcal{E}$ , where “volume” is used in the sense given in (5.18).

$$V_{\hat{S}_v} = \int \int \cdots \int_{\hat{S}_v} dx_1 dx_2 \cdots dx_n \quad (5.18)$$

However, it should also be apparent that analytically computing the exact volume of  $\hat{S}_v$ , which is defined by changing boundaries, would be extremely difficult, particularly for a general  $n$ -dimensional system.

Although a finite element approximation to the area could be computed, it was decided that a measure of the ellipsoid itself would be used as an indicator of the volume of  $\hat{S}_v$ . The objective function used for the ellipsoid search, (3.9), was tested, but the optimization routine generally failed to converge to a solution. Figure 5.12, which illustrates a hypothetical search progression  $a \rightarrow b \rightarrow c$ , shows the potential difficulty in using the volume as the objective function. As the ellipse is lengthened along one axis, it is “pinched” along the other by the time-optimal trajectories (which define the bounds of invariance under limited control). A typical search sequence with Matlab's `constr.m` would begin with initial conditions similar to ellipse  $a$ , then continue through  $c$  until it “collapsed” along the minor axis, followed by a restart with an ellipse similar to, but generally smaller than,  $a$ .

After evaluating several functions, (5.19) was selected as the objective function for the semi-ellipsoidal set optimization.

$$\min \log \text{tr}(P) \quad (5.19)$$

It is known that the trace of a matrix yields the sum of the eigenvalues. This objective function, then, seeks to minimize the sum of the inverse of the axes' lengths (since the eigenvalues of  $P^{-1}$  give the lengths of the axes), (5.20).

$$\text{tr}(P) = \frac{1}{\lambda_1(P^{-1})} + \frac{1}{\lambda_2(P^{-1})} + \cdots + \frac{1}{\lambda_n(P^{-1})} \quad (5.20)$$

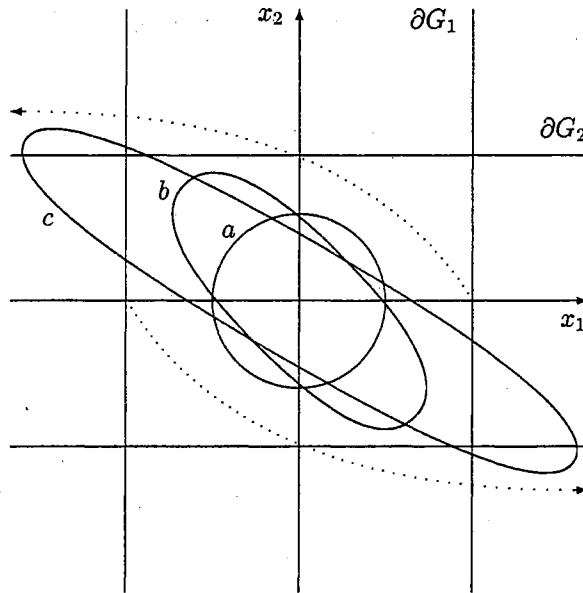


Figure 5.12: Progression of Search Using Ellipsoid Volume as Objective Function.

Since the value of this function grows quickly if any of the eigenvalues of  $P^{-1}$  approaches zero, the search is generally stable and cases such as that shown in Figure 5.12 are avoided. Although admittedly imprecise, searches using (5.19) do provide results comparable to what is available in the literature (as is shown in Chapter 6) and, most importantly, serve to establish proof-of-concept for the theory.

### Control Constraints

To this point, no mention has been made of the input constraint with respect to the semi-ellipsoidal set. If the constraint, (3.12), is imposed, the control may not reach its maximum value inside  $\hat{S}_v$ , even if though it is reached on  $\partial\mathcal{E}$ , as shown in Figure 5.13.

Computing the maximum control on  $\hat{S}_v$  is a difficult problem best done using a search technique. However, this approach is computationally inefficient if performed at each iteration of the optimization of (5.19). To solve this problem, several search passes are made using a pseudo-maximum control,  $\hat{u}$ , in place of  $\bar{u}$  in (3.12). For the first pass,  $\hat{u}$  is set to the value of  $\bar{u}$  and an optimal ellipsoid computed. A separate constrained optimization algorithm then finds the largest value of  $|Kx|$ ,  $\tilde{u}_1$ , within  $\hat{S}_v$ . If  $\tilde{u}_1$  is within some tolerance of  $\bar{u}$  the search is terminated. However, if  $\tilde{u}_1$  is not within this tolerance,  $\hat{u}$  is increased and a new ellipsoid and corresponding  $\tilde{u}_2$  computed. Again, if  $\tilde{u}_2$  is within some tolerance of  $\bar{u}$ , the search is terminated, with the second ellipsoid as a solution (see Figure 5.14). If not, the pairs  $(\hat{u}_1, \tilde{u}_1)$  and  $(\hat{u}_2, \tilde{u}_2)$  are used to select a new  $\hat{u}$  via a false-position



linear search formula [12, pg. 138], and the process repeated.

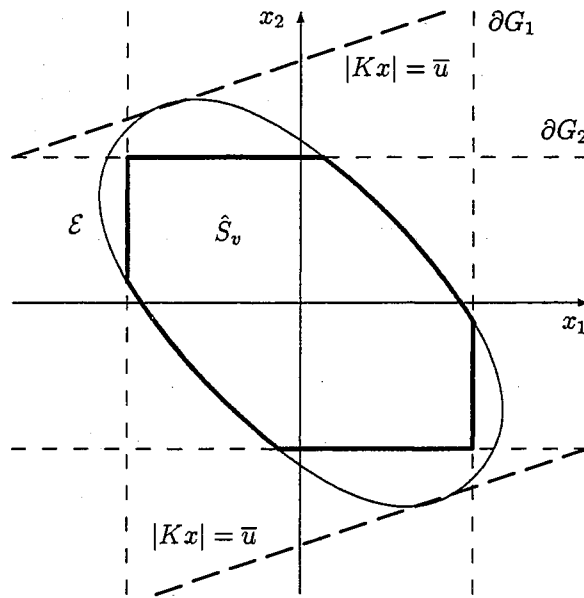


Figure 5.13: Maximal Control Occurring Outside of  $\hat{S}_v$ .

### Positive Definiteness of Ellipsoid Matrix

Even though a positive definiteness constraint is imposed on the ellipsoid matrix,  $P$ , it may occur during the search that one of the eigenvalues will be slightly negative (although within the tolerance of the search routine). To prevent this from happening,  $P$  is parametrized in terms of an auxiliary symmetric matrix,  $\hat{P}$ , where the two matrices are related by (5.21).

$$P = \hat{P} \cdot \hat{P} \quad (\hat{P} = \hat{P}^T) \quad (5.21)$$

By imposing the positive definiteness constraint on  $\hat{P}$ , positive (semi-) definiteness of  $P$  is ensured via the quadratic form of (5.21).

## 5.2 Reachable Semi-Ellipsoidal Set

As in Chapter 3, computation of the reachable set merely requires a sign change on any time derivatives involved (note that this means  $\partial \hat{G}_i$  changes, as well). For regions of  $\partial \hat{S}_v$  formed by the state constraints,  $\frac{d}{dt}(\Gamma_i x)$  must be negative for points on the boundary of the set to be reachable from points within the set. However, note that since (5.22) holds, no significant change need be made to Theorem 5.3. This is reflected in its reformulation for the reachable case in Theorem 5.4,

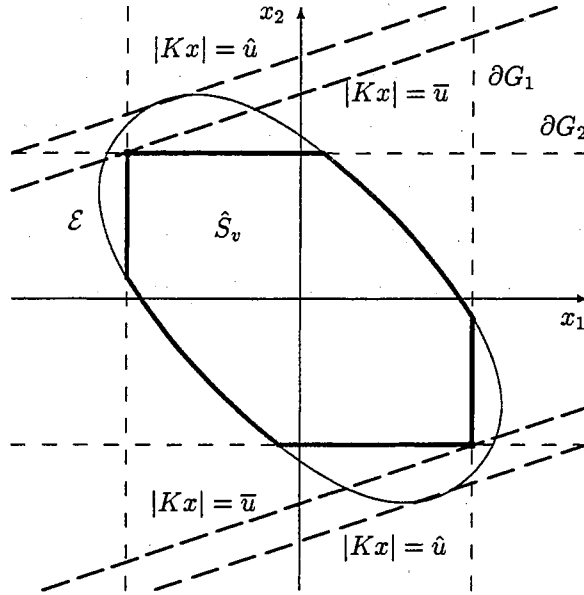


Figure 5.14: Maximal Control Occurring Inside of  $\hat{S}_v$ .

where the control gain is now  $K_e$ .

$$\text{null}\{-\Gamma_i(A - BK_e)\} = \text{null}\{\Gamma_i(A - BK_e)\} \quad (5.22)$$

**Theorem 5.4** *Given an ellipsoid,  $\mathcal{E}$ , controlled invariant in negative time for system (1.1) under  $u = -K_e x$ , and state constraints  $G_i$ , whose intersection is  $G$ , the set  $\hat{S}_e = \mathcal{E} \cap G$  is controlled invariant in negative time under  $u$  if, for  $\Gamma_i(A - BK_e) \neq 0$  ( $i = 1, \dots, k$ ), the following inequality holds:*

$$\Gamma_i^R P_i^R (\Gamma_i^R)^T \leq 1 \quad (5.23)$$

where

$$\Gamma_i^R = \Gamma_i W_i^e \quad (5.24)$$

$$P_i^R = (W_i^e)^T P_e W_i^e \quad (5.25)$$

$$W_i^e = \text{null}\{\Gamma_i(A - BK_e)\} \quad (5.26)$$

### 5.3 Controllable Semi-Ellipsoidal Set

Similarly to Chapter 3, the controllable semi-ellipsoidal set is computed by imposing the constraints of Theorems 5.3 and 5.4 on  $\mathcal{E}$  and  $K_v$ ,  $K_e$ . Reduced-dimension state constraints must be imposed for both reaching and recovering conditions, although in certain instances a simplification can be made, as noted in Remark 5.4.

**Remark 5.4** If  $\Gamma_i B = 0$ , then the inequality constraint for  $G_i$  is not a function of the control gains, and the derivative boundary,  $\partial \dot{G}_i$ , is the same for both positive and negative time. As illustrated in Figure 5.15, this implies that there is no region of  $\partial G_i$  which  $\mathcal{E}$  may overlap, since all points on  $\mathcal{E} \cap \partial G_i$  would be either recoverable but not reachable or reachable but not recoverable. Consequently, for the case  $\Gamma_i A \neq 0$ ,  $\Gamma_i B = 0$ , the state constraint inequality simplifies to (5.27), which was used in Chapter 3 and implies that  $\mathcal{E}$  must lie entirely within the state constraint.

$$\Gamma_i P_c^{-1} \Gamma_i^T \leq 1 \quad (5.27)$$

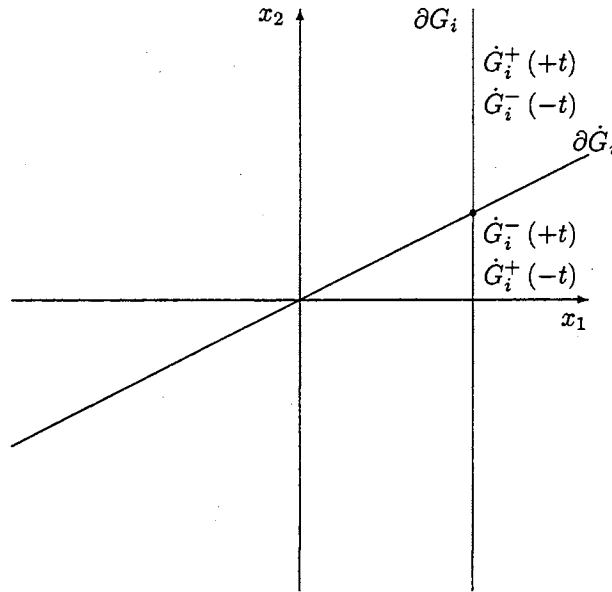


Figure 5.15: Duality of Constraint Derivative Regions when  $\Gamma_i B = 0$ .

## 5.4 Summary

The optimization problems finding the ellipsoid matrix for the semi-ellipsoidal set approximations to the maximal recoverable, reachable, and controllable sets are summarized in Problems 5.1, 5.2, and 5.3, respectively, where  $W_i^r$  is defined as in (5.17), and  $W_i^c$  is defined as in (5.26).

### Problem 5.1 (Recoverable Semi-Ellipsoidal Set)

Minimize

$$\log \text{tr} (P_v)$$

subject to

$$P_v (A - BK_v) + (A - BK_v)^T P_v \leq 0$$

$$\begin{aligned}
& -P_v < 0 \\
& (\Gamma_i W_i^v)^T \left[ (W_i^v)^T P_v W_i^v \right]^{-1} (\Gamma_i W_i^v)^T \leq 1, \Gamma_i (A - BK_v) \neq 0 \\
& K_v P_v^{-1} K_v^T \leq \hat{u}^2, \hat{u} \geq \bar{u}
\end{aligned}$$

where

$$\max_{\mathcal{E}(\hat{u}) \cap G} |K_v x| \leq \bar{u}$$

### Problem 5.2 (Reachable Semi-Ellipsoidal Set)

Minimize

$$\log \text{tr} (P_e)$$

subject to

$$-P_e (A - BK_e) - (A - BK_e)^T P_e \leq 0$$

$$-P_e < 0$$

$$(\Gamma_i W_i^e)^T \left[ (W_i^e)^T P_e W_i^e \right]^{-1} (\Gamma_i W_i^e)^T \leq 1, \Gamma_i (A - BK_e) \neq 0$$

$$K_e P_e^{-1} K_e^T \leq \hat{u}^2, \hat{u} \geq \bar{u}$$

where

$$\max_{\mathcal{E}(\hat{u}) \cap G} |K_e x| \leq \bar{u}$$

### Problem 5.3 (Controllable Semi-Ellipsoidal Set)

Minimize

$$\log \text{tr} (P_c)$$

subject to

$$P_c (A - BK_v) + (A - BK_v)^T P_c \leq 0$$

$$-P_c (A - BK_e) - (A - BK_e)^T P_c \leq 0$$

$$-P_c < 0$$

$$(\Gamma_i W_i^v)^T \left[ (W_i^v)^T P_c W_i^v \right]^{-1} (\Gamma_i W_i^v)^T \leq 1, \Gamma_i (A - BK_v) \neq 0$$

$$(\Gamma_i W_i^e)^T \left[ (W_i^e)^T P_c W_i^e \right]^{-1} (\Gamma_i W_i^e)^T \leq 1, \Gamma_i (A - BK_e) \neq 0$$

$$K_v P_c^{-1} K_v^T \leq \hat{u}^2, \hat{u} \geq \bar{u}$$

$$K_e P_c^{-1} K_e^T \leq \hat{u}^2, \hat{u} \geq \bar{u}$$

where

$$\max_{\mathcal{E}(\hat{u}) \cap G} |K_v x| \leq \bar{u}$$

$$\max_{\mathcal{E}(\hat{u}) \cap G} |K_e x| \leq \bar{u}$$

## Chapter 6

# Semi-Ellipsoidal Set Examples

This chapter presents the example test cases considered in Chapter 4 for ellipsoidal approximations. The same general comments apply to these results of semi-ellipsoidal sets. Additionally, it is noted that, if the ellipsoidal solution to a particular case does not touch the state constraints, the solution for the semi-ellipsoidal set will be the same (This is the condition where insufficient control effort exists to operate far beyond the origin.)

### 6.1 Second Order Systems

Contained in this section are the computed maximal semi-ellipsoidal sets for six different second-order, linear systems in control canonical form: stable, stable focus, marginally stable, unstable, unstable focus, and saddle (one stable, one unstable pole).

As in the previous example chapter, trajectories (coming from the ellipsoid when recovering, going to the ellipsoid when reaching) are plotted for all three cases. For simplicity, it is merely stated that  $\max_{\mathcal{E} \cap \mathcal{G}} |Kx| \leq 1$  in all cases.

#### 6.1.1 Stable

The eigenvalues of the following  $A$  matrix are  $-1, -3$ .

$$A = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 6.1, 6.2, and 6.3, respectively. Ellipsoid matrices are given in Equations (6.1), (6.2),

and (6.3).

$$P_v = \begin{bmatrix} 1.0000 & 0.1468 \\ 0.1468 & 0.0409 \end{bmatrix} \left( K_v = \begin{bmatrix} 0.6953 & 0.2019 \end{bmatrix} \right) \quad (6.1)$$

$$P_e = \begin{bmatrix} 18.128 & 0.0000 \\ 0.0000 & 21.532 \end{bmatrix} \left( K_e = \begin{bmatrix} -2.1581 & -4.0000 \end{bmatrix} \right) \quad (6.2)$$

$$P_c = \begin{bmatrix} 17.504 & 0.0000 \\ 0.0000 & 22.206 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -2.2118 & -1.7039 \end{bmatrix} \\ K_e = \begin{bmatrix} -2.2117 & -4.0000 \end{bmatrix} \end{array} \right) \quad (6.3)$$

Since the system is stable, it is expected that all of the points in the state-space can recover to the origin (even without actuator effort), although not necessarily along admissible trajectories. The recoverable semi-ellipsoidal set generated for this system (Figure 6.1) is considerably larger than the ellipsoidal set, Figure 4.1. The trajectories originating from the ellipse boundary illustrate invariance.

In contrast, the stability of the system naturally limits the set of state which can be reached from the origin. Since insufficient control effort exists to even reach the state constraint boundaries, the “semi-ellipsoidal set” of Figure 6.2 is in fact equivalent to the ellipsoidal set of Figure 4.2.

Finally, since the points in the controllable ellipsoid must be both reachable and recoverable, it is expected to be approximately equal to the intersection of these two sets, which in this case is dictated by the reachable set. Thus, the controllable semi-ellipsoidal set, Figure 6.3, does not increase in size compared to the ellipsoidal set, even though the recoverable set is expanded with the semi-ellipsoidal approach.

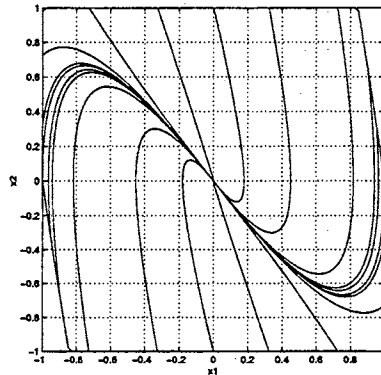


Figure 6.1: Recoverable Semi-Ellipsoidal Set for Second Order Stable System.

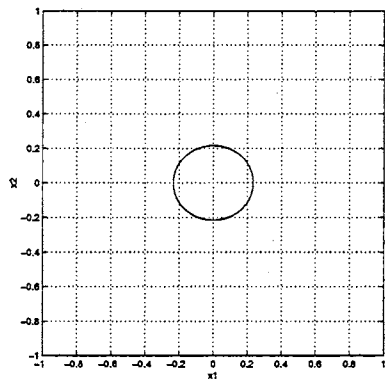


Figure 6.2: Reachable Semi-Ellipsoidal Set for Second Order Stable System.

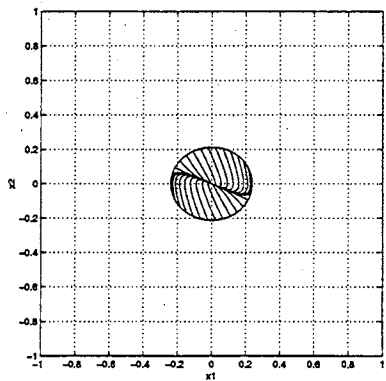


Figure 6.3: Controllable Semi-Ellipsoidal Set for Second Order Stable System.

### 6.1.2 Stable Focus

The eigenvalues of the following  $A$  matrix are  $-1 \pm 1.414i$ .

$$A = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 6.4, 6.5, and 6.6, respectively. Ellipsoid matrices are given in Equations (6.4), (6.5), and (6.6).

$$P_v = \begin{bmatrix} 1.0000 & 0.0799 \\ 0.0799 & 0.1716 \end{bmatrix} \left( K_v = \begin{bmatrix} -0.8161 & 0.1698 \end{bmatrix} \right) \quad (6.4)$$

$$P_e = \begin{bmatrix} 8.6982 & 0.0000 \\ 0.0000 & 4.7387 \end{bmatrix} \left( K_e = \begin{bmatrix} -1.1644 & -2.0000 \end{bmatrix} \right) \quad (6.5)$$

$$P_c = \begin{bmatrix} 8.7051 & 0.0000 \\ 0.0000 & 4.7318 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -1.1605 & 0.6164 \end{bmatrix} \\ K_e = \begin{bmatrix} -1.1603 & -2.0000 \end{bmatrix} \end{array} \right) \quad (6.6)$$

The comments for the stable system hold for this stable focus system as well. As with the ellipsoidal approximation, the eigenvalue locations for this system are generally closer to the imaginary axis, implying that a larger set of states can be reached from the origin, as seen in the figures.

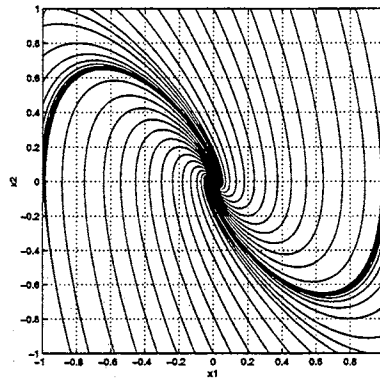


Figure 6.4: Recoverable Semi-Ellipsoidal Set for Second Order Stable Focus System.

### 6.1.3 Marginal

The eigenvalues of the following  $A$  matrix are  $0, 0$ .

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



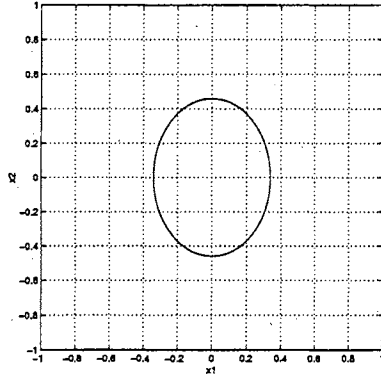


Figure 6.5: Reachable Semi-Ellipsoidal Set for Second Order Stable Focus System.

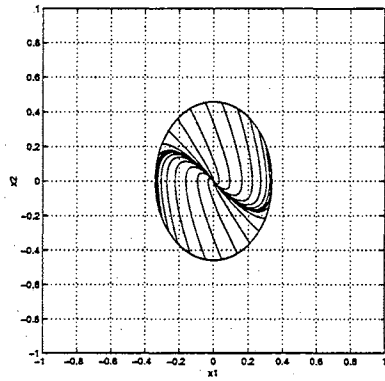


Figure 6.6: Controllable Semi-Ellipsoidal Set for Second Order Stable Focus System.

The semi-ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 6.7, 6.8, and 6.9, respectively. Ellipsoid matrices are given in Equations (6.7), (6.8), and (6.9).

$$P_v = \begin{bmatrix} 1.0000 & 0.3379 \\ 0.3379 & 0.6079 \end{bmatrix} \left( K_v = \begin{bmatrix} 0.7026 & 0.7374 \end{bmatrix} \right) \quad (6.7)$$

$$P_e = \begin{bmatrix} 1.0000 & -0.3379 \\ -0.3379 & 0.6079 \end{bmatrix} \left( K_e = \begin{bmatrix} 0.7026 & -0.7374 \end{bmatrix} \right) \quad (6.8)$$

$$P_c = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 1.0000 & 0.0001 \end{bmatrix} \\ K_e = \begin{bmatrix} 1.0000 & -0.0001 \end{bmatrix} \end{array} \right) \quad (6.9)$$

Since the system's eigenvalues are on the imaginary axis, there is no bias towards the recoverable or reachable sets. Consequently, both sets are approximately the same size though shifted in orientation. Although both recoverable and reachable sets share a common boundary along the maximal velocity limit ( $x_2 = \pm 1$ ,  $x_1 \in [-0.4, 0.4]$ ), the controllable ellipse barely touches this bound. Had a larger control been permitted, the controllable ellipsoid would have extended past this bound, as seen in Figure 6.10 (cf. (6.10)) for  $\bar{u} = 10$ , where recovering and reaching trajectories are shown in separate plots.

$$P_c = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 0.2154 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 4.6416 & 4.1113 \end{bmatrix} \\ K_e = \begin{bmatrix} 4.6416 & -4.1113 \end{bmatrix} \end{array} \right) \quad (6.10)$$

For this system, no amount of control effort would be sufficient to allow overlap of the controllable ellipsoid and the position constraint, via the arguments of Remark 5.4.

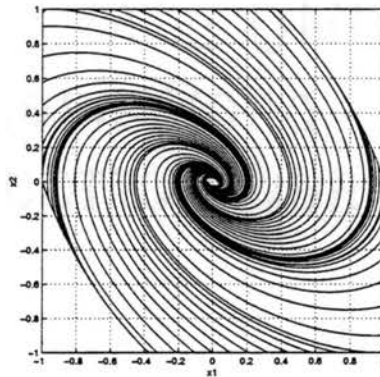


Figure 6.7: Recoverable Semi-Ellipsoidal Set for Second Order Marginal System.

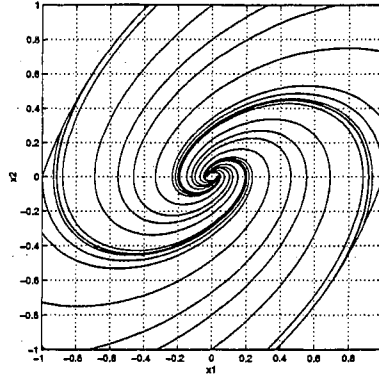


Figure 6.8: Reachable Semi-Ellipsoidal Set for Second Order Marginal System.

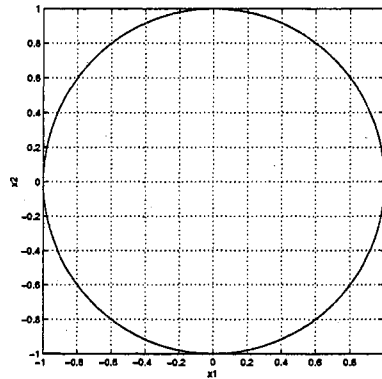


Figure 6.9: Controllable Semi-Ellipsoidal Set for Second Order Marginal System.

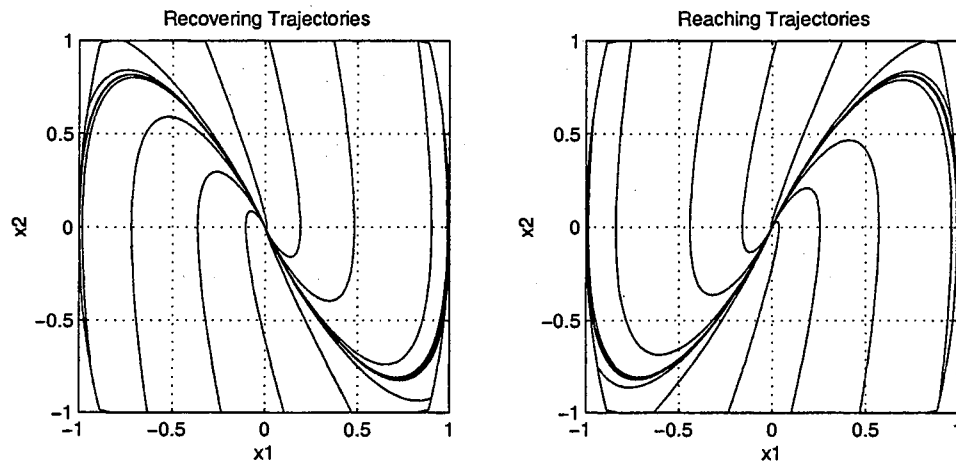


Figure 6.10: Controllable Semi-Ellipsoidal Set for Second Order Marginal System using Increased Control Effort.

### 6.1.4 Unstable

The eigenvalues of the following  $A$  matrix are 1, 3.

$$A = \begin{bmatrix} 0 & 1 \\ -3 & 4 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 6.11, 6.12, and 6.13, respectively. Ellipsoid matrices are given in Equations (6.11), (6.12), and (6.13).

$$P_v = \begin{bmatrix} 18.118 & 0.0000 \\ 0.0000 & 21.542 \end{bmatrix} \left( K_v = \begin{bmatrix} -2.1589 & 4.0000 \end{bmatrix} \right) \quad (6.11)$$

$$P_e = \begin{bmatrix} 1.0000 & -0.1423 \\ -0.1423 & 0.0392 \end{bmatrix} \left( K_e = \begin{bmatrix} 0.7975 & -0.3018 \end{bmatrix} \right) \quad (6.12)$$

$$P_c = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 0.3742 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -0.3278 & 6.1139 \end{bmatrix} \\ K_e = \begin{bmatrix} -0.3276 & 1.8538 \end{bmatrix} \end{array} \right) \quad (6.13)$$

The reverse is true of the unstable system results compared with the stable system results. Here, the recoverable set is smaller than the reachable set, which dictates the size of the controllable set.

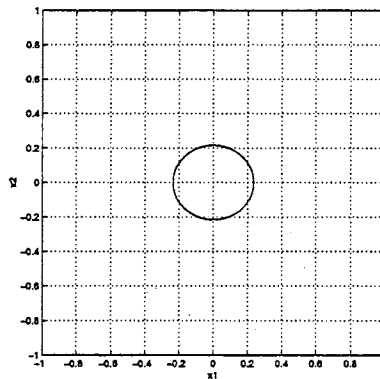


Figure 6.11: Recoverable Semi-Ellipsoidal Set for Second Order Unstable System.

### 6.1.5 Unstable Focus

The eigenvalues of the following  $A$  matrix are  $1 \pm 1.414i$ .

$$A = \begin{bmatrix} 0 & 1 \\ -3 & 2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

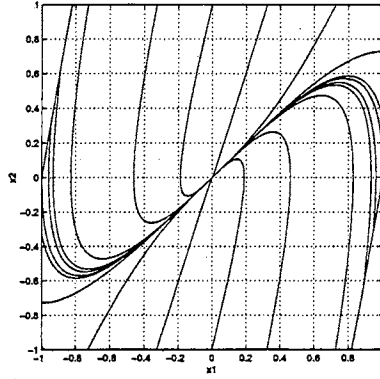


Figure 6.12: Reachable Semi-Ellipsoidal Set for Second Order Unstable System.

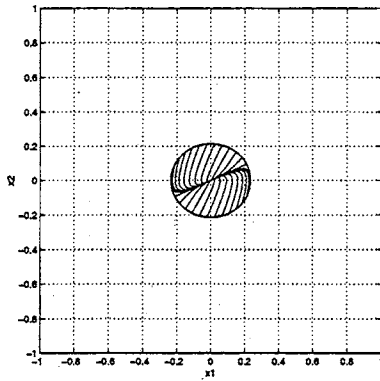


Figure 6.13: Controllable Semi-Ellipsoidal Set for Second Order Unstable System.

The semi-ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 6.14, 6.15, and 6.16, respectively. Ellipsoid matrices are given in Equations (6.14), (6.15), and (6.16).

$$P_v = \begin{bmatrix} 8.6977 & 0.0000 \\ 0.0000 & 4.7392 \end{bmatrix} \left( K_v = \begin{bmatrix} -1.1647 & 2.0000 \end{bmatrix} \right) \quad (6.14)$$

$$P_e = \begin{bmatrix} 1.0000 & -0.0799 \\ -0.0799 & 0.1716 \end{bmatrix} \left( K_e = \begin{bmatrix} -0.8161 & -0.1698 \end{bmatrix} \right) \quad (6.15)$$

$$P_c = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 0.2692 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 0.7143 & 5.1752 \end{bmatrix} \\ K_e = \begin{bmatrix} 0.7148 & -1.2137 \end{bmatrix} \end{array} \right) \quad (6.16)$$

Again, the results can be contrasted to the stable focus system, where the recoverable set was larger than the reachable, opposite of what is seen here.

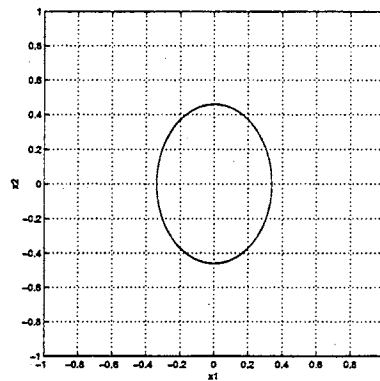


Figure 6.14: Recoverable Semi-Ellipsoidal Set for Second Order Unstable Focus System.

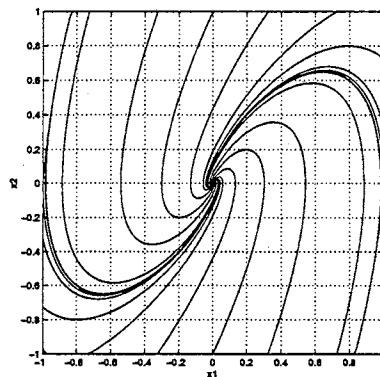


Figure 6.15: Reachable Semi-Ellipsoidal Set for Second Order Unstable Focus System.

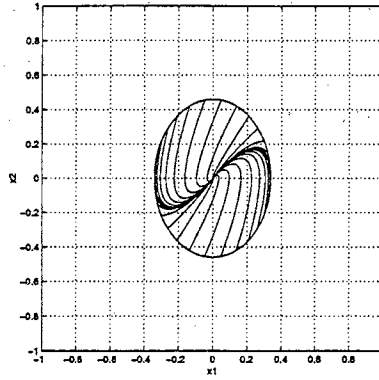


Figure 6.16: Controllable Semi-Ellipsoidal Set for Second Order Unstable Focus System.

### 6.1.6 Saddle

The eigenvalues of the following  $A$  matrix are 1,  $-3$ .

$$A = \begin{bmatrix} 0 & 1 \\ 3 & -2 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 6.17, 6.18, and 6.19, respectively. Ellipsoid matrices are given in Equations (6.17), (6.18), and (6.19).

$$P_v = \begin{bmatrix} 9.0000 & 3.0000 \\ 3.0000 & 1.0000 \end{bmatrix} \left( K_v = \begin{bmatrix} 3.0000 & 1.0000 \end{bmatrix} \right) \quad (6.17)$$

$$P_e = \begin{bmatrix} 9.0000 & -9.0000 \\ -9.0000 & 9.0000 \end{bmatrix} \left( K_e = \begin{bmatrix} 3.0000 & -3.0000 \end{bmatrix} \right) \quad (6.18)$$

$$P_c = \begin{bmatrix} 1.0000 & 0.0000 \\ 0.0000 & 0.3250 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 6.0770 & 0.5687 \end{bmatrix} \\ K_e = \begin{bmatrix} 6.0766 & -4.5276 \end{bmatrix} \end{array} \right) \quad (6.19)$$

The eigenvectors of the system (and their associated eigenvalues) are  $v_1 = \begin{bmatrix} 0.707 & 0.707 \end{bmatrix}^T$  ( $\lambda_1 = 1$ ) and  $v_2 = \begin{bmatrix} -0.316 & 0.949 \end{bmatrix}^T$  ( $\lambda_1 = -3$ ). The recoverable set is strongly skewed along the stable eigenvector, while the reachable set is strongly skewed along the unstable eigenvector. The ellipsoids found for the recoverable and reachable sets have axes lengths of  $(0.1, 1.5 \times 10^5)$  and  $(8.8 \times 10^3, 0.06)$ , respectively, with the principle axis lying along the stable (unstable) mode for the recoverable (reachable) case. This relatively large principle axis length may indicate that, under certain conditions, the optimal ellipsoid is unrestricted in one dimension (i.e., axis length is  $\infty$ ).

The controllable set using the semi-ellipsoidal method is the same as using the ellipsoidal method, since the region of intersection is limited (As before, the trajectories seen in the controllable set's plot are recovering, while the reaching trajectories cycle close to the bound.)

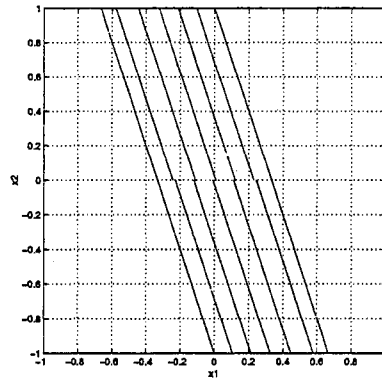


Figure 6.17: Recoverable Semi-Ellipsoidal Set for Second Order Saddle System.

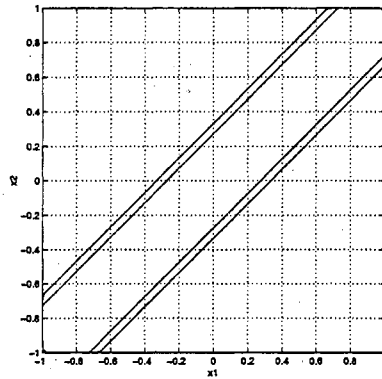


Figure 6.18: Reachable Semi-Ellipsoidal Set for Second Order Saddle System.



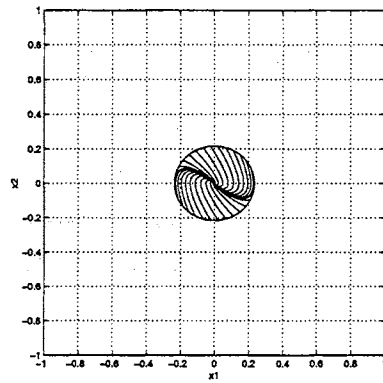


Figure 6.19: Controllable Semi-Ellipsoidal Set for Second Order Saddle System.

## 6.2 Third Order Systems

The figures in this section include both a complete view of the semi-ellipsoidal set and a quadrant of the three dimensional ellipsoid which allows the trajectories to be viewed. Additionally, the recovering and reaching trajectories of the semi-ellipsoidal set are plotted in separate sub-figures (Note that the same quadrant is plotted for each, although some scaling differences may exist between the pairs.)

### 6.2.1 Stable

The eigenvalues of the following  $A$  matrix are  $-1, -1, -1$ .

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -2 & -1 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable and reachable sets are given in Figures 6.20 and 6.21, while Figures 6.22 and 6.23 show the controllable set and representative trajectories. Ellipsoid matrices are given in Equations (6.20), (6.21), and (6.22).

$$P_v = \begin{bmatrix} 0.1302 & 0.1266 & -0.0304 \\ 0.1266 & 0.2281 & 0.1981 \\ -0.0304 & 0.1981 & 1.0071 \end{bmatrix} \left( K_v = \begin{bmatrix} 0.3312 & -0.0556 & -0.6952 \end{bmatrix} \right) \quad (6.20)$$

$$P_e = \begin{bmatrix} 9.7015 & -19.403 & -6.1485 \\ -19.403 & 54.236 & 12.297 \\ -6.1485 & 12.297 & 9.5471 \end{bmatrix} \left( K_e = \begin{bmatrix} -3.0000 & 4.9567 & 2.0000 \end{bmatrix} \right) \quad (6.21)$$

$$P_c = \begin{bmatrix} 9.7012 & -19.402 & -6.1741 \\ -19.402 & 54.239 & 12.348 \\ -6.1741 & 12.348 & 9.5409 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 2.7011 & -6.4480 & -1.6284 \end{bmatrix} \\ K_e = \begin{bmatrix} -3.0000 & 4.9546 & 2.0000 \end{bmatrix} \end{array} \right) \quad (6.22)$$

As seen in Figure 6.20, the boundary of the recoverable semi-ellipsoidal set is formed by the state constraints on the front and back surfaces and by a combination of the ellipsoid and state constraint on the top and bottom surfaces. As with the second order stable system, insufficient control effort exists for the reachable set to touch the constraint boundaries, so both the reachable and controllable sets are similar to those found using the ellipsoidal method (note that the orientation of the figures differs compared to Figures 4.20 and 4.21).

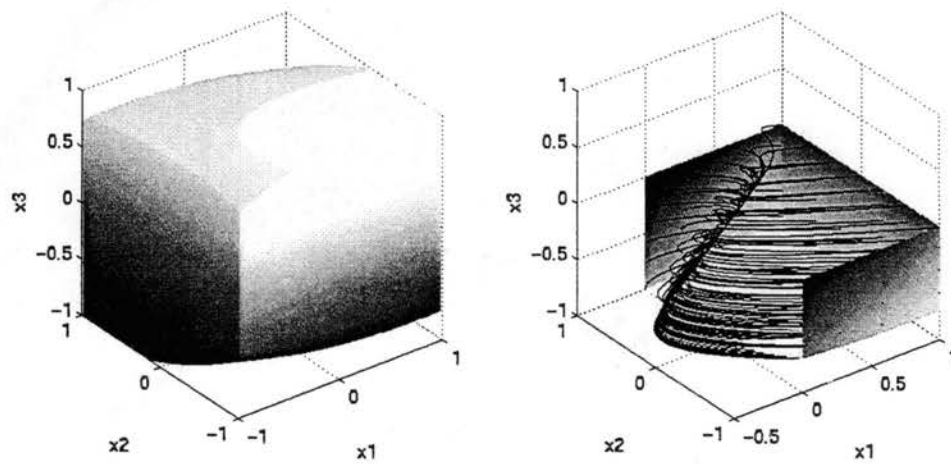


Figure 6.20: Recoverable Semi-Ellipsoidal Set for Third Order Stable System.

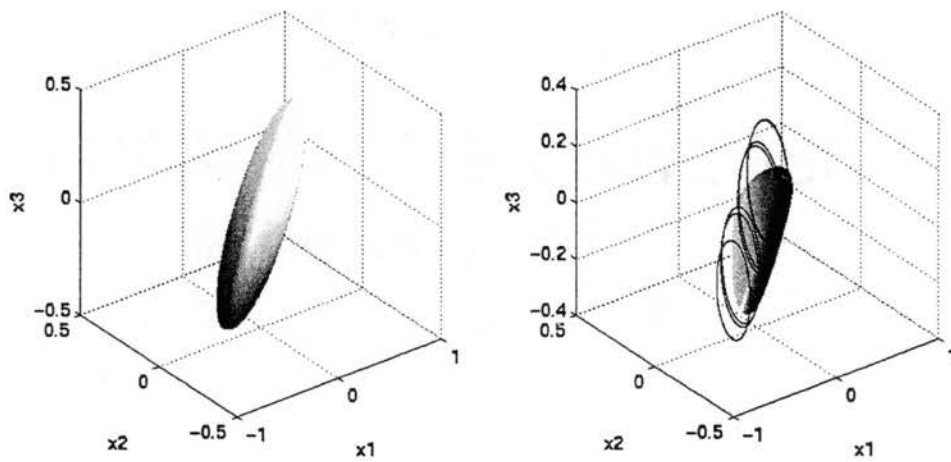


Figure 6.21: Reachable Semi-Ellipsoidal Set for Third Order Stable System.

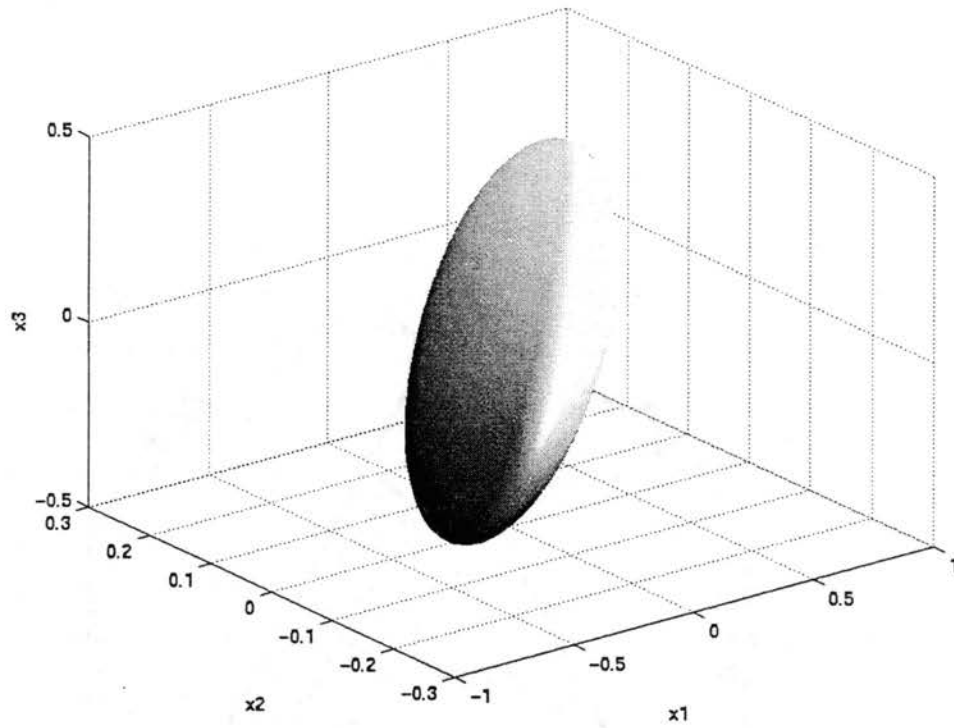


Figure 6.22: Controllable Semi-Ellipsoidal Set for Third Order Stable System.

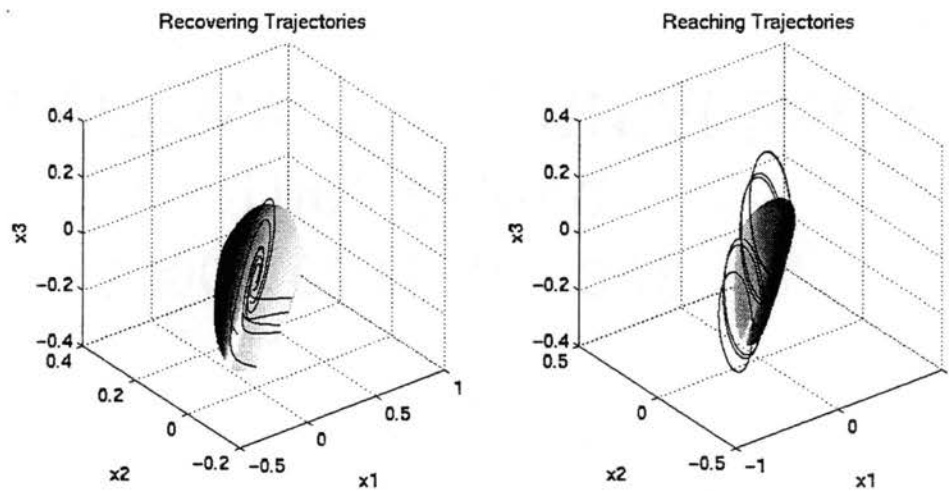


Figure 6.23: Controllable Semi-Ellipsoidal Set Trajectories for Third Order Stable System.

## 6.2.2 Unstable

The eigenvalues of the following  $A$  matrix are 1, 3, 5.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 15 & -23 & 9 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable, reachable, and controllable sets are given in Figures 6.24 and 6.25, while Figures 6.26 and 6.27 show the controllable set and representative trajectories. Ellipsoid matrices are given in Equations (6.23), (6.24), and (6.25).

$$P_v = \begin{bmatrix} 578.17 & 0.0000 & 139.83 \\ 0.0000 & 777.99 & 0.0000 \\ 139.83 & 0.0000 & 221.98 \end{bmatrix} \left( K_v = \begin{bmatrix} 15.000 & -18.865 & 9.0000 \end{bmatrix} \right) \quad (6.23)$$

$$P_e = \begin{bmatrix} 1.7304 & -2.4563 & 0.0891 \\ -2.4563 & 4.4868 & -0.2661 \\ 0.0891 & -0.2661 & 0.4165 \end{bmatrix} \left( K_e = \begin{bmatrix} 1.0550 & -1.6332 & -0.2911 \end{bmatrix} \right) \quad (6.24)$$

$$P_c = \begin{bmatrix} 535.37 & 0.0000 & 118.88 \\ 0.0000 & 839.85 & 0.0000 \\ 118.88 & 0.0000 & 212.88 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 15.000 & -18.496 & 9.0001 \end{bmatrix} \\ K_e = \begin{bmatrix} 4.9481 & -18.496 & -9.0004 \end{bmatrix} \end{array} \right) \quad (6.25)$$

Following the established pattern, the results for the unstable system are opposite the results for the stable system. Here, the recovery trajectories appear to oscillate about the ellipsoid boundary, while the reaching trajectories clearly emanate from the origin.

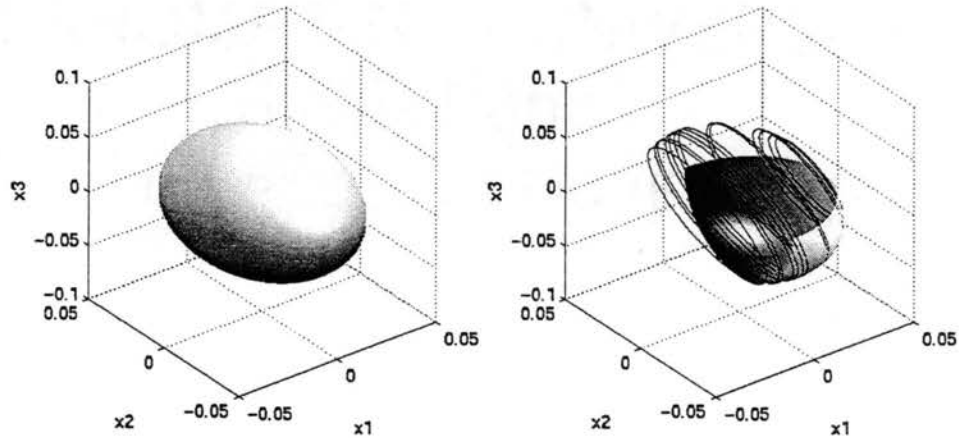


Figure 6.24: Recoverable Semi-Ellipsoidal Set for Third Order Unstable System.

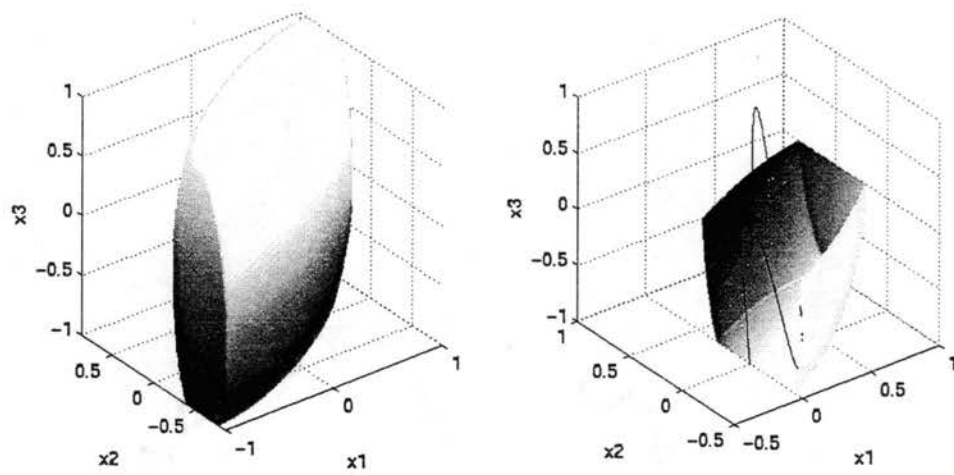


Figure 6.25: Reachable Semi-Ellipsoidal Set for Third Order Unstable System.

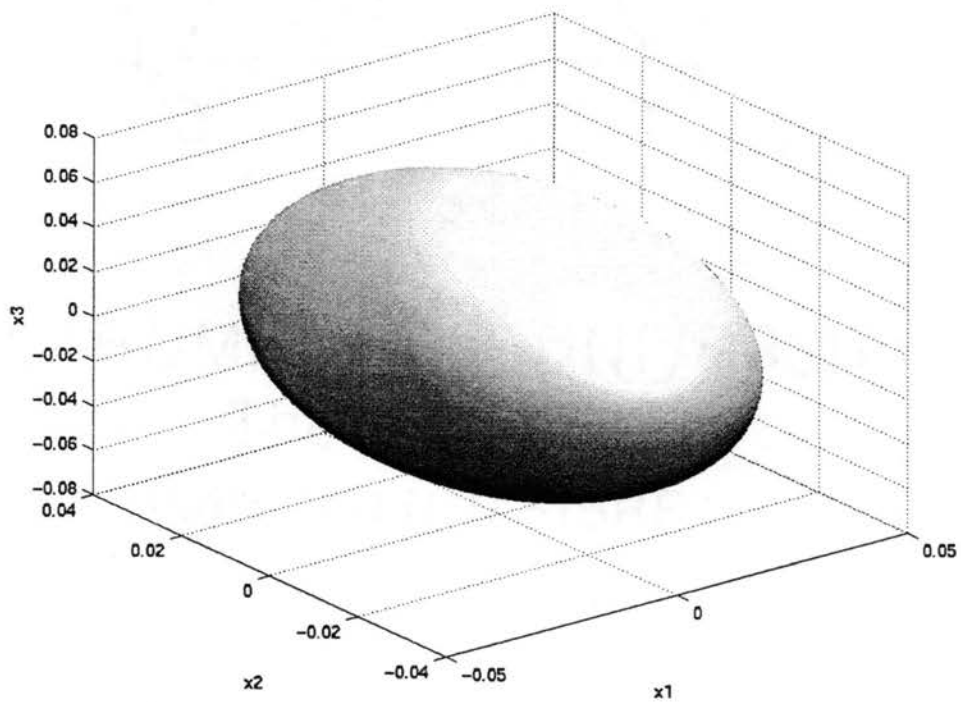


Figure 6.26: Controllable Semi-Ellipsoidal Set for Third Order Unstable System.

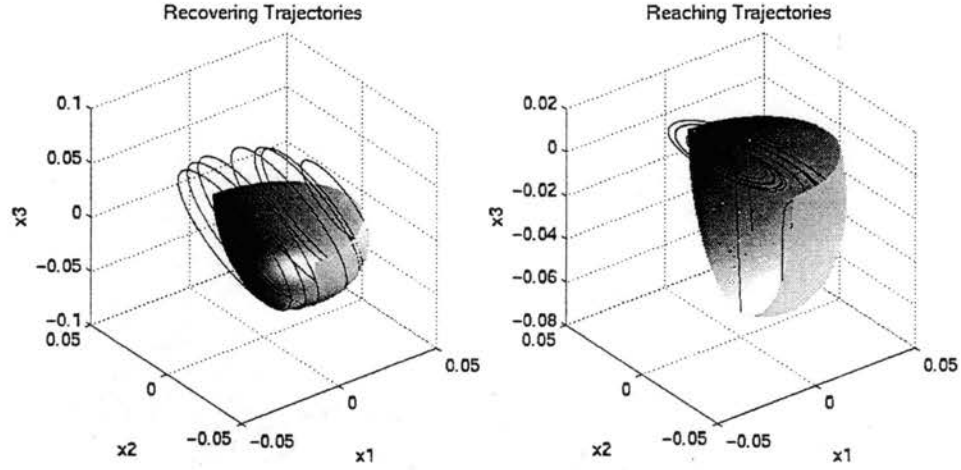


Figure 6.27: Controllable Semi-Ellipsoidal Set Trajectories for Third Order Unstable System.

### 6.2.3 Marginal

The eigenvalues of the following  $A$  matrix are  $0, 0, 0$ .

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable and reachable sets are given in Figures 6.28 and 6.29, while Figures 6.30 and 6.31 show the controllable set and representative trajectories. Ellipsoid matrices are given in Equations (6.26), (6.27), and (6.28).

$$P_v = \begin{bmatrix} 1.1478 & 0.5516 & 0.3859 \\ 0.5516 & 1.5179 & 0.7385 \\ 0.3859 & 0.7385 & 1.0078 \end{bmatrix} \left( K_v = \begin{bmatrix} 0.6527 & 0.8875 & 0.9545 \end{bmatrix} \right) \quad (6.26)$$

$$P_e = \begin{bmatrix} 1.1478 & -0.5519 & 0.3860 \\ -0.5519 & 1.5174 & -0.7383 \\ 0.3860 & -0.7383 & 1.0083 \end{bmatrix} \left( K_e = \begin{bmatrix} -0.6535 & 0.8875 & -0.9544 \end{bmatrix} \right) \quad (6.27)$$

$$P_c = \begin{bmatrix} 1.5477 & 0.0000 & 1.0947 \\ 0.0000 & 1.9988 & 0.0000 \\ 1.0947 & 0.0000 & 2.1881 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} 0.0007 & 1.4138 & 0.0014 \end{bmatrix} \\ K_e = \begin{bmatrix} -0.0020 & 1.4138 & -0.0040 \end{bmatrix} \end{array} \right) \quad (6.28)$$

Here, both the recoverable and reachable set boundaries are partially formed by the state constraints. In both cases, the trajectories tend to a plane where they oscillate about the origin. The controllable ellipsoid does not touch the state constraints with the given control limit.

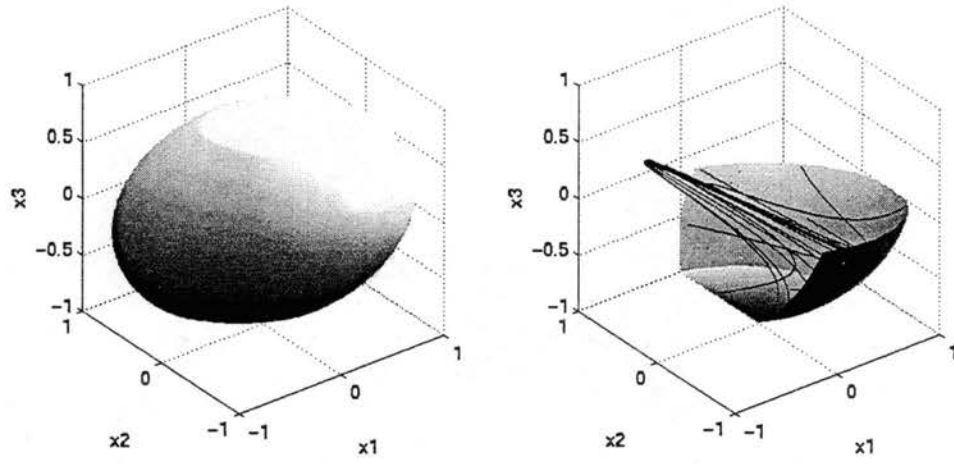


Figure 6.28: Recoverable Semi-Ellipsoidal Set for Third Order Marginal System.

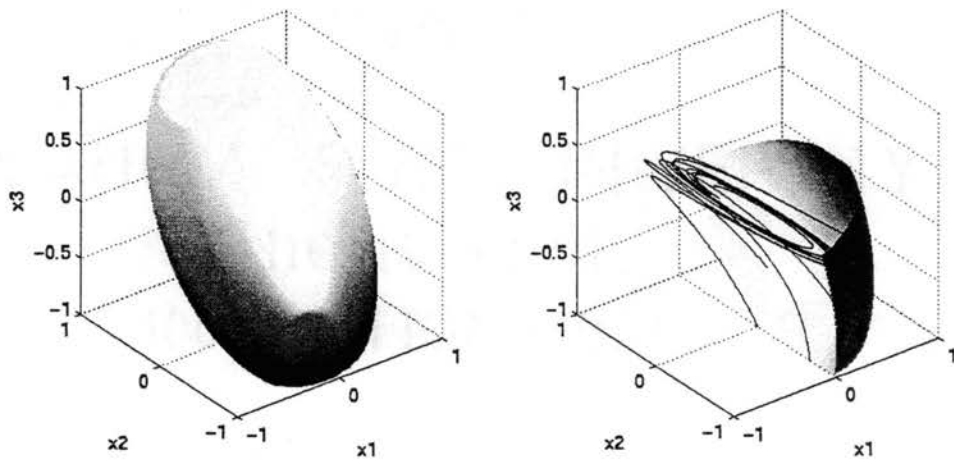


Figure 6.29: Reachable Semi-Ellipsoidal Set for Third Order Marginal System.



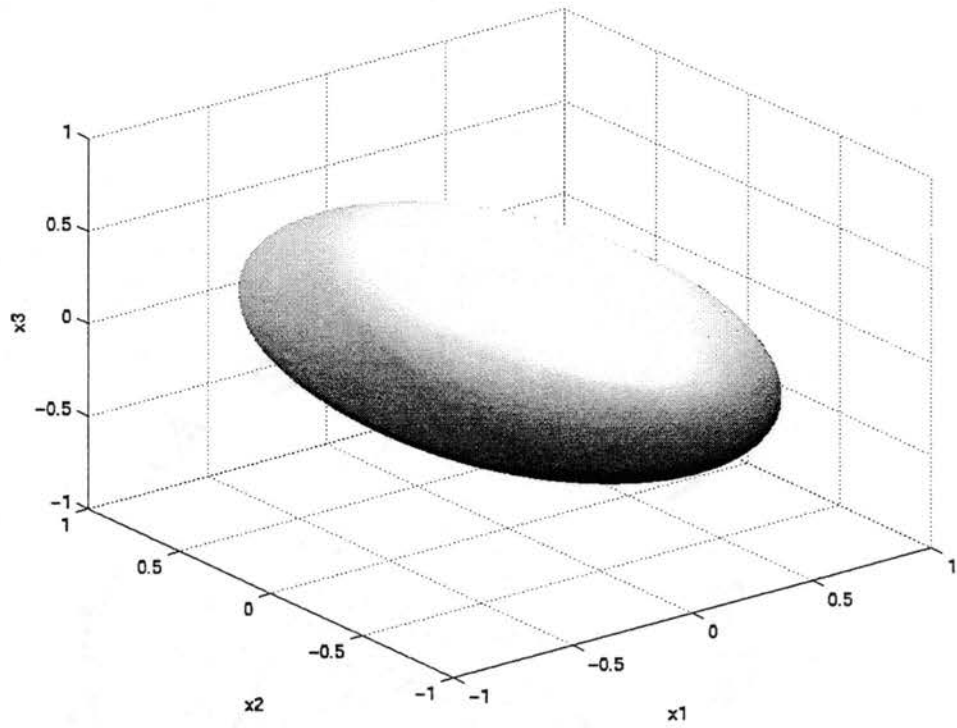


Figure 6.30: Controllable Semi-Ellipsoidal Set for Third Order Marginal System.

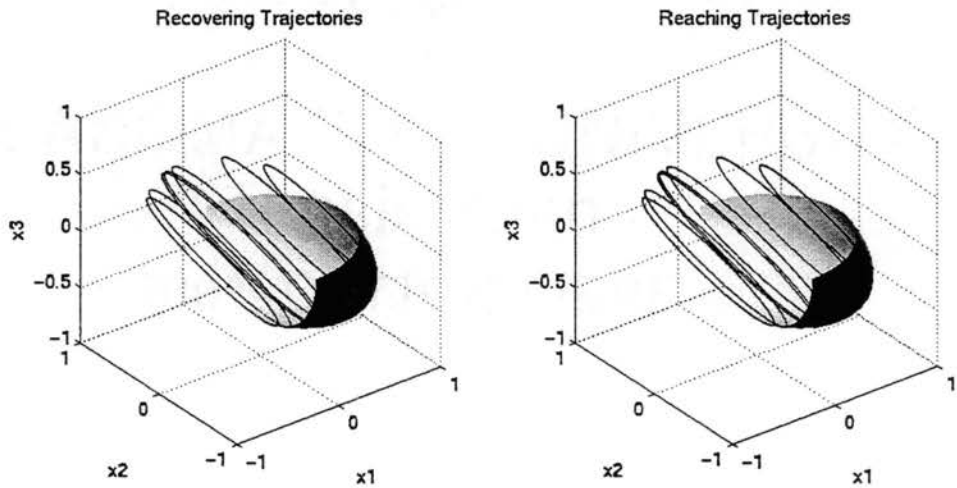


Figure 6.31: Controllable Semi-Ellipsoidal Set Trajectories for Third Order Marginal System.

## 6.2.4 Mixed

The eigenvalues of the following  $A$  matrix are  $-1.28, 0.14 \pm 1.53i$ .

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 & -2 & -1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The semi-ellipsoidal approximations of the recoverable and reachable sets are given in Figures 6.32 and 6.33, while Figures 6.34 and 6.35 show the controllable set and representative trajectories. Ellipsoid matrices are given in Equations (6.29), (6.30), and (6.31).

$$P_v = \begin{bmatrix} 1.3459 & 1.0517 & 0.1372 \\ 1.0517 & 1.9292 & 0.5561 \\ 0.1372 & 0.5561 & 0.7538 \end{bmatrix} \left( K_v = \begin{bmatrix} -0.6704 & -0.0417 & 0.6315 \end{bmatrix} \right) \quad (6.29)$$

$$P_e = \begin{bmatrix} 9.0105 & -1.0958 & 3.7675 \\ -1.0958 & 2.1179 & -0.5576 \\ 3.7675 & -0.5576 & 2.5709 \end{bmatrix} \left( K_e = \begin{bmatrix} -3.0003 & 0.3836 & -1.2839 \end{bmatrix} \right) \quad (6.30)$$

$$P_c = \begin{bmatrix} 10.332 & 0.0000 & 4.3167 \\ 0.0000 & 2.3934 & 0.0000 \\ 4.3167 & 0.0000 & 2.8035 \end{bmatrix} \left( \begin{array}{l} K_v = \begin{bmatrix} -1.6877 & 0.3934 & -0.1479 \end{bmatrix} \\ K_e = \begin{bmatrix} -3.0000 & 0.3934 & -1.0000 \end{bmatrix} \end{array} \right) \quad (6.31)$$

Although the system contains both stable and unstable eigenvalues, only the recoverable set touches the state constraints. In this case, the “strength” of the stable mode relative to the unstable pair apparently restricted the size of the reachable state-space.

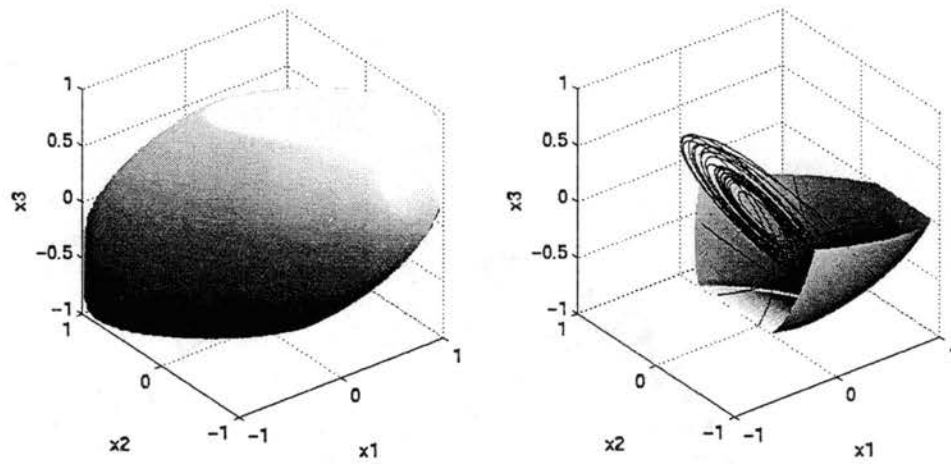


Figure 6.32: Recoverable Semi-Ellipsoidal Set for Third Order Mixed System.

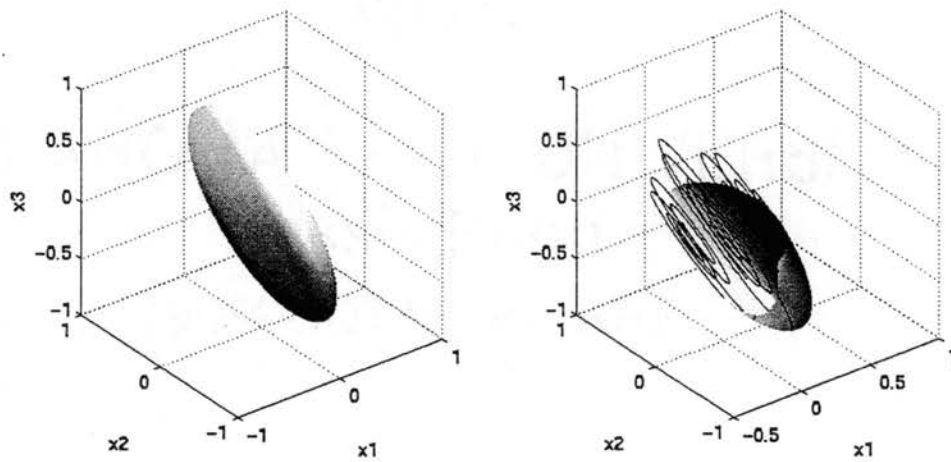


Figure 6.33: Reachable Semi-Ellipsoidal Set for Third Order Mixed System.

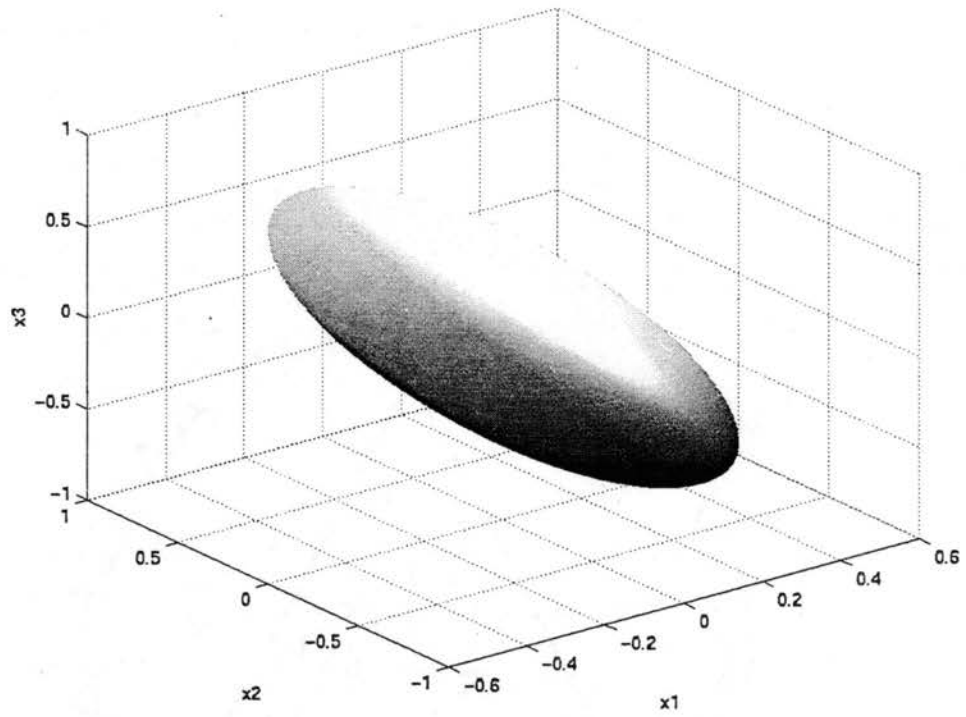


Figure 6.34: Controllable Semi-Ellipsoidal Set for Third Order Mixed System.

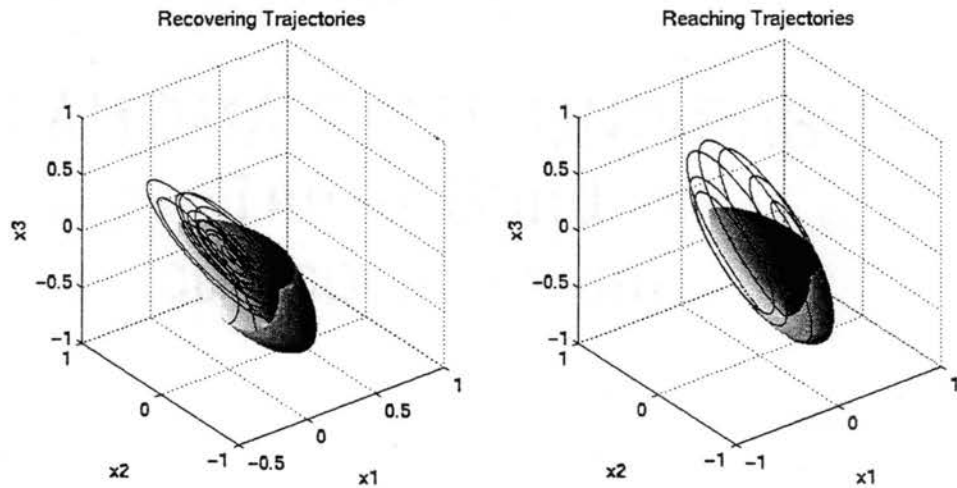


Figure 6.35: Controllable Semi-Ellipsoidal Set Trajectories for Third Order Mixed System.

### 6.3 Comparison Against Published Result

The recoverable semi-ellipsoidal algorithm was applied to the double integrator example of Section 4.3. The search was somewhat slow to converge, probably due to the fact that the state constraint limits differed fairly significantly. This implied that, more than in the preceding examples, much of the ellipsoid was contained outside of the state constraint bounds and contributed nothing to the volume of the semi-ellipsoidal set, although the present cost function does not accurately account for this. Nevertheless, the solution, (6.32), was eventually found.

$$P_v = \begin{bmatrix} 0.0016 & 0.0027 \\ 0.0027 & 0.0243 \end{bmatrix} \left( K_v = \begin{bmatrix} 0.0281 & 0.1475 \end{bmatrix} \right) \quad (6.32)$$

Figure 6.36 compares the semi-ellipsoidal set (solid line) with the ellipsoidal set (dashed line) and polyhedral set (dotted line). In general, the semi-ellipsoidal set captures a larger area than the ellipsoidal set. Furthermore, the semi-ellipsoidal set requires only 7 parameters to define (3 for the symmetric ellipsoid matrix and 2 for each of the state constraint matrices), while the polyhedral set requires 14 (cf. (4.33)). This information is summarized in Table 6.1 along with information on the enclosed area of the state-space relative to the polyhedral approach, which may be computed analytically for this relatively simple case.

Method	Parameters	Area ( $m^2/s$ )	Change (%)
Polyhedral	14	458.8	-
Ellipsoidal	3	392.7	-14.4
Semi-Ellipsoidal	7	435.3	-5.1

Table 6.1: Comparison of Number of Required Parameters and Enclosed Area.

For this double integrator system, the maximal set,  $S_v$ , is formed by the state constraints and a second order curve (the time-optimal curve,  $(\int \dot{x}(\bar{u}, t) dt, \int \int \dot{x}(\bar{u}, t) dt) \Rightarrow (\bar{u}t + c_0, \frac{1}{2}\bar{u}t^2 + c_0t + c_1)$ ). Thus, a modest number of vertices closely approximates the pure integrator system's maximal set for the polyhedral approach. More generally, though, the maximal set surfaces are described by much more complex functions. Consequently, the number of parameters needed to gain the accuracy advantage of the polyhedral form increases significantly with both system order and complexity, while the parameters for the ellipsoidal and semi-ellipsoidal remains modest ( $(n^2 + n)/2$  and  $(3n^2 + n)/2$ , respectively, assuming a constraint on each state).

**Remark 6.1** *The existence of asymmetric constraints on a state or input is one instance in which the polyhedral method has a distinct advantage, since the ellipsoidal and semi-ellipsoidal methods, as formulated here, cannot capture the additional freedom available in the one direction, while the polyhedral method can (cf. Remark 3.2 on page 14).*

Interestingly, the ellipsoidal set includes a portion of the state-space not belonging to the semi-ellipsoidal set. This shortcoming of the semi-ellipsoidal method might be minimized by the use of a different objective function (one which computes the actual enclosed volume, rather than using the trace of the ellipsoid matrix as an approximation) or by investigating the possibility of piecewise-quadratic ([25], [51]) or non-quadratic ([52]) Lyapunov functions (both of which could potentially better approximate a higher-order surface than a single quadratic function). However, both of these modifications introduce considerably more complexity to the problem.

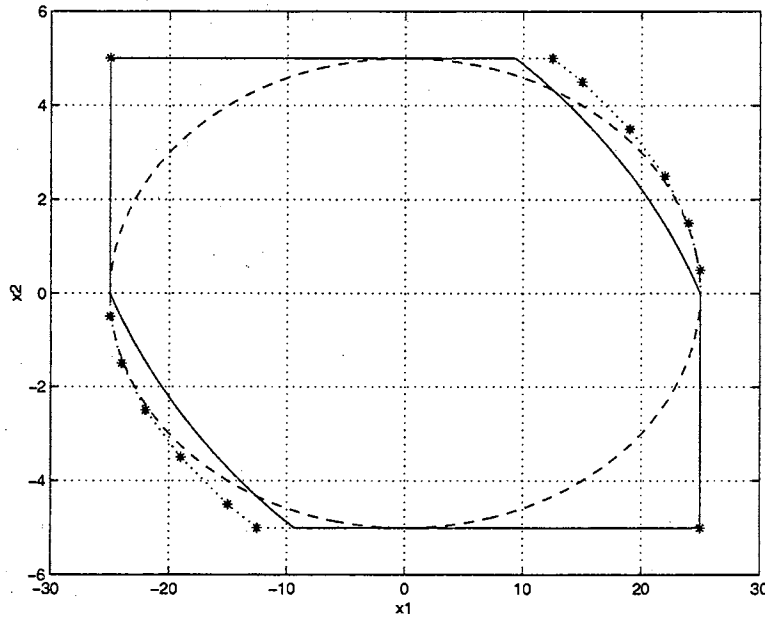


Figure 6.36: Comparison of Semi-Ellipsoidal, Ellipsoidal, and Polyhedral Methods for Double Integrator System.

## Chapter 7

# Concluding Remarks

This thesis investigates alternative approximations to the maximal operating sets for linear systems with constrained states and input. The need for such approximations is highlighted in the following comments by Blanchini [8]:

The techniques based on ellipsoidal sets are *conservative*. This fact is well established in robustness analysis as well as in the determination of domains of attraction under constraints. Polyhedral sets provide non-conservative solutions but *they lead to computationally intensive algorithms*. This is one of the most serious troubles although the fast improving computer performances alleviate the problem.

...

We believe that there are still several open problems that are worth an investigation. For instance, we have seen that the only family of sets of practical use having a bounded complexity are the ellipsoids. For the reasons explained above it would be important to develop algorithms to find other classes of invariant sets to achieve a reasonable tradeoff between conservatism and complexity.

As the primary contribution, it is shown that the intersection,  $\hat{S}$ , of an invariant ellipsoid,  $\mathcal{E}$ , and state constraints,  $G$ , is itself invariant under certain conditions. Specifically, the following theorem is proved for the recoverable case, and a similar theorem provided for the reachable case.

**Theorem 5.3** *Given an ellipsoid,  $\mathcal{E}$ , controlled invariant for system (1.1) under  $u = -K_v x$ , and state constraints  $G_i$ , whose intersection is  $G$ , the set  $\hat{S}_v = \mathcal{E} \cap G$  is controlled invariant under  $u$  if, for  $\Gamma_i(A - BK_v) \neq 0$  ( $i = 1, \dots, k$ ), the following inequality holds:*

$$\Gamma_i^R (P_i^R)^{-1} (\Gamma_i^R)^T \leq 1$$

where

$$\begin{aligned}\Gamma_i^R &= \Gamma_i W_i^v \\ P_i^R &= (W_i^v)^T P_v W_i^v \\ W_i^v &= \text{null} \{ \Gamma_i (A - BK_v) \}\end{aligned}$$

The proposed semi-ellipsoidal approach satisfies the original objective of providing an approximation which is less conservative than the ellipsoidal method but simpler than the polyhedral method, as illustrated by the example of Section 6.3. Furthermore, the algorithms are implemented in the form of Matlab script routines available in the appendices. These routines are evaluated on second- and third-order systems in Chapters 4 and 6, which serve as proof-of-concept of the approach. The only limitation for systems with higher dimension is the efficiency and stability of the optimization routine, since the theoretical development contains no inherent restriction on system order. Although application of the technology is not demonstrated in this thesis, the theory is now mature enough to allow that phase of work to begin.

## 7.1 Future Research

A number of avenues are available for future research, in both the theory and application of this new technology. The following list highlights some of the more prominent ones.

### Theory:

- *Discrete Time Systems.* To date, the research has focused solely on continuous time systems. However, practical application of the theory would almost certainly be implemented using digital controllers. Consequently, the theory must be adapted to discrete time systems. In principle, this could be accomplished by replacing the Lyapunov constraint, (3.1), with (7.1).

$$V(k+1) \leq V(k) \tag{7.1}$$

This relationship, in turn, implies a discrete time (on-sample) invariance constraint, (7.2), replacing the continuous time invariance constraint, (3.10).

$$(A_d - B_d K_v)^T P_v (A_d - B_d K_v) - P_v \leq 0 \tag{7.2}$$

A primary concern here is that this relation does not guarantee state-constraint compliance of the inter-sample points of a continuous time plant.



- *Robustness Issues.* In this thesis, no consideration is given to the robustness of the set's invariance to external disturbances, modeling errors, etc. For example, with many of the optimal ellipsoids found, maximal control was needed to maintain system trajectories just inside the bound, so that no control effort remains in reserve to counteract the effects of a disturbance. Consequently, application of the technology to actual control problems will require modification to compensate for this shortcoming, perhaps by specifying minimum decay rates on the Lyapunov function, (3.1).

Since the objective of the research presented in this thesis is not to design the control law, but to approximate the operating bounds, it is not within the context to specify a decay rate for the entire admissible state-space. A more appropriate specification is a minimum decay rate on the boundary of the ellipse, say  $-\eta$ . By construction,  $x^T P_v x = 1$  for points,  $x$ , on the boundary of the maximal ellipse. Consequently, the following derivation, leading to the modified invariance constraint, (7.3), ensures a decay rate of  $-\eta$  on the boundary of the ellipsoid.

$$\begin{aligned}
dV/dt &\leq -\eta \\
&\Downarrow \\
x^T \left[ P_v (A - BK_v) + (A - BK_v)^T P_v \right] x &\leq -\eta (x^T P_v x) \\
&\Downarrow \\
P_v (A - BK_v) + (A - BK_v)^T P_v &\leq -\eta P_v \\
P_v \left[ (A - \frac{\eta}{2} I) - BK_v \right] + \left[ (A - \frac{\eta}{2} I) - BK_v \right]^T P_v &\leq 0 \tag{7.3}
\end{aligned}$$

For the semi-ellipsoidal sets, though, (7.3) does not provide robustness for those portions of the boundary formed by the state constraints. Imposing the relation,  $\frac{d}{dt} (\Gamma_i x) \leq -\eta$  (as opposed to  $\frac{d}{dt} (\Gamma_i x) \leq 0$ ) introduces a bias (non-trivial solution) to the expression of  $W_i$  in (5.8). This requires a re-derivation of the reduced-dimension state constraint, similar to that of Appendix E.

- *Asymmetric State and Input Constraints.* If the bounds on a particular state or on the input are asymmetric, the quadratic nature of the state and input constraints on the ellipsoid, (3.11), (3.12), imply that smaller in magnitude of these two bounds will be restricting quantity (i.e., the proposed method can *handle* constraints of the form  $\underline{f} \leq f \leq \bar{f}$  - so long as these bounds contain the origin - but it cannot *take advantage* of the asymmetric bounds.)

- *Multi-Input Systems.* As derived, the ellipsoidal and semi-ellipsoidal methods are applicable only to single-input systems. The results need to be adapted to multi-input systems to have a broader range of application.
- *Characterization of Absolute Maximal Sets.* To the author’s knowledge, no work exists which adapts the theory in LeMay’s thesis [32] (true maximal sets for constrained input systems) to systems with constrained states and input. Intuitively, it seems that the direct application of this theory to control applications would be limited due to the complexity. However, a characterization of the true maximal sets could serve as a benchmark by which to quantify the quality of approximating sets.
- *Alternative Lyapunov Functions.* Several papers have recently been published investigating unique Lyapunov functions for special applications, including piecewise quadratic functions [51], [25], and non-quadratic functions [52]. If such functions could be substituted for the quadratic ellipsoidal function, the boundaries of the true maximal set could be more closely approximated (since more degrees of freedom would be available to approximate the maximal set’s surfaces). Such functions may also be necessary to fully take advantage of asymmetric constraints.
- *Output Feedback.* The ellipsoidal and semi-ellipsoidal methods presented in this thesis assume that full state feedback is available for making control decisions. For most applications, however, only a limited number of states (outputs) are available. Thus, an observer will most likely be required to provide estimates of the states, which will, in turn, require a more conservative ellipsoidal or semi-ellipsoidal set to account for the uncertainties of the estimates.

**Application:**

- *Reference Governor.* The concept of modifying a signal to avoid saturation of the control or control rate has been investigated extensively in a variety of forms (see, e.g., [19] for the “reference governor,” [41] for the “measurement governor”). McNamee [36] has presented a reference governor which ensures that the system state remains within a specified polyhedral set. This method could be adapted to use the semi-ellipsoidal set as a first application.
- *Reference Trajectory Generation.* Many existing control schemes, such as sliding mode control, utilize reference trajectories to move from set-point to set-point. If the semi-ellipsoid could be used to generate trajectories which naturally met state and input

constraints, then the theory could have immediate application to a number of different problems using these existing control techniques.

- *Modified Backstepping Control.* In [38], a modified backstepping approach is investigated in which constraints are placed on each of the states in the backstepping controller. It was discovered that this approach still suffered from the non-causal nature of the problem and could only avoid saturation of the control and control rate, but not of other states. The bounding semi-ellipsoidal set might be used to dynamically manipulate the saturation levels of the states to eliminate this flaw.
- *Modified Variable Structure Control.* The principle of the variable structure controller commonly known as sliding mode is to define an  $(n - 1)$  dimension hyperplane and then design the controller so that this surface is bi-directionally attractive (i.e., states to the “left” of the plane are drawn to the “right,” and states to the “right” of the plane are drawn to the “left”). Using a similar approach, it may be possible to design a variable structure controller in which the semi-ellipsoid’s boundary is uni-directionally attractive. In this scheme, points exterior to the ellipse are drawn to it, while points interior are not, but are, rather, governed by a different control law (state-feedback, sliding mode, etc.) Hence, the semi-ellipsoidal set merely acts to restrain trajectories resulting from pre-existing control law from violating state constraints.

# Bibliography

- [1] M. E. Acchab, F. M. Callier, and V. Wertz. Admissible controls and attainable states for a class of nonlinear systems with general constraints. *International Journal of Robust and Nonlinear Control*, 4:267–288, 1994.
- [2] M. E. Achhab and V. Wertz. A further result on attainable states for a class of nonlinear systems with constraints. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 4, pages 3819–3820, Lake Buena Vista, FL, December 14-16, 1994.
- [3] Michael Athans and Peter L. Faub. *Optimal Control: An Introduction to the Theory and Its Applications*. McGraw-Hill, New York, 1966.
- [4] Jean-Pierre Aubin. A survey of viability theory. *SIAM Journal of Control and Optimization*, 28(4):749–788, 1990.
- [5] Alberto Bemporad. A predictive controller with artificial Lyapunov function for linear systems with input/state constraints. *Automatica*, 34(10):1255–1260, 1998.
- [6] William H. Beyer, editor. *CRC Handbook of Mathematical Sciences*. CRC Press, Inc., Boca Raton, FL, 6th edition, 1987.
- [7] George Bitsoris and Eliana Gravalou. A design technique for the control of discrete-time systems subject to state and control constraints. In *Proceedings of the 35th IEEE Conference on Decision and Control*, pages 1503–1504, Kobe, Japan, December 1996.
- [8] F. Blanchini. Set invariance in control - a survey. Scheduled for publication in *Automatica*, 35(11), 1999.
- [9] Franco Blanchini and Stefano Miani. Constrained stabilization of continuous-time linear systems. *Systems & Control Letters*, 28(2):95–102, 1996.

- [10] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1994.
- [11] William L. Brogan. *Modern Control Theory*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 1991.
- [12] Steven C. Chapra and Raymond P. Canale. *Numerical Methods for Engineers*. McGraw-Hill, St. Louis, 2nd edition, 1988.
- [13] G. Colombo and V. Křivan. Robustness of viability controllers under small perturbations. *Journal of Optimization Theory and Applications*, 83(1):207–215, 1994.
- [14] R. Lane Dailey. Eigenvector derivatives with repeated eigenvalues. *AIAA Journal*, 27(4):486–491, 1989.
- [15] R. L. Fox and M. P. Kapoor. Rates of change of eigenvalues and eigenvectors. *AIAA Journal*, 6(12):2426–2429, 1968.
- [16] J. E. Gayek and T. L. Vincent. On the intersection of controllable and reachable sets. *Journal of Optimization Theory and Applications*, 50(2):267–278, 1986.
- [17] E. G. Gilbert and Ilya Kolmanovsky. Discrete-time reference governors for systems with state and control constraints and disturbance inputs. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 2, pages 1189–1194, New Orleans, December 13-15 1995.
- [18] Elmer G. Gilbert, Ilya Kolmanovsky, and Kok Tin Tan. Nonlinear control of discrete-time linear systems with state and control constraints: A reference governor with global convergence properties. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 1, pages 144–149, Lake Buena Vista, FL, December 14-16 1994.
- [19] Elmer G. Gilbert, Ilya Kolmanovsky, and Kok Tin Tan. Discrete-time reference governors and the nonlinear control of systems with state and control constraints. *International Journal of Robust and Nonlinear Control*, 5(5):487–504, 1995.
- [20] Elmer G. Gilbert and Kok Tin Tan. Linear systems with state and control constraints: The theory and application of output admissible sets. *IEEE Transactions on Automatic Control*, AC-36(9):1008–1020, 1991.

- [21] Per-Olaf Gutman and Michael Cwikel. Admissible sets and feedback control for discrete-time linear dynamical systems with bounded controls and states. *IEEE Transactions on Automatic Control*, AC-31(4):373–376, 1986.
- [22] Per-Olaf Gutman and Michael Cwikel. An algorithm to find maximal state constraint sets for discrete-time linear dynamical systems with bounded controls and states. *IEEE Transactions on Automatic Control*, AC-32(3):251–254, 1987.
- [23] Per-Olaf Gutman and Per Hagander. A new design of constrained controllers for linear systems. *IEEE Transactions on Automatic Control*, AC-30(1):22–33, 1985.
- [24] Ling Hou and Anthony N. Michel. Asymptotic stability of systems with saturation constraints. *IEEE Transactions on Automatic Control*, AC-43(8):1148–1154, 1998.
- [25] Mikael Johansson and Anders Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, AC-43(4):555–559, 1998.
- [26] Jer-Nan Juang and Kyong B. Lim. On the eigenvalue and eigenvector derivatives of a general matrix. Technical Memorandum NASA-TM-98127, NASA Langley Research Center, Hampton, VA, March 1987.
- [27] Thomas Kailath. *Linear Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.
- [28] Naoya Kawasaki and Etsujiro Shimemura. Determining quadratic weighting matrices to locate poles in a specified region. *Automatica*, 19(5):557–560, 1983.
- [29] S. S. Keerthi and E. G. Gilbert. Computation of minimum-time feedback control laws for discrete-time systems with state-control constraints. *IEEE Transactions on Automatic Control*, AC-32(5):432–435, 1987.
- [30] Donald E. Kirk. *Optimal Control Theory: An Introduction*. Prentice Hall, Englewood Cliffs, NJ, 1970.
- [31] Ilya Kolmanovsky and Elmer G. Gilbert. Multimode regulators for systems with state & control constraints and disturbance inputs. In A. Stephen Morse, editor, *Control Using Logic-Based Switching*, Lecture Notes in Control and Information Sciences (222), pages 104–117. Springer-Verlag, New York, 1997.

- [32] Joseph Louis LeMay. *Recoverable, Reachable, Controllable, and Maintainable Regions for Control Systems with Linear Plants and Bounded Controller Outputs*. Ph.D. Dissertation, The University of Michigan, Ann Arbor, MI, 1963.
- [33] K. B. Lim, J. L. Junkins, and B. P. Wang. Re-examination of eigenvector derivatives. *Journal of Guidance, Control, and Dynamics*, 10(6):581–587, 1987.
- [34] Jerrold E. Marsden and Anthony J. Tromba. *Vector Calculus*. W. H. Freeman and Company, New York, 3rd edition, 1988.
- [35] D. Q. Mayne and W. R. Schroeder. Robust time-optimal control of constrained linear systems. *Automatica*, 33(12):2103–2118, 1997.
- [36] Joe McNamee and Meir Pachter. The construction of the set of stable states for constrained systems with open-loop unstable plants. In *Proceedings of the American Control Conference*, volume 6, pages 3364–3368, Philadelphia, 1998.
- [37] Joe McNamee and Meir Pachter. Efficient nonlinear reference governor algorithms for constrained tracking control systems. In *Proceedings of the American Control Conference*, pages 3549–3553, San Diego, June 1999.
- [38] Eduardo A. Misawa, Gary E. Young, and Otis Funches. Nonlinear control of disc drives. Technical Report AR6-053, Oklahoma Center for the Advancement of Science and Technology (OCAST), Oklahoma City, OK, December 31, 1998.
- [39] R. H. Plaut and K. Huseyin. Derivatives of eigenvalues and eigenvectors in non-self-adjoint systems. *AIAA Journal*, 11(2):250–251, 1973.
- [40] Lynn C. Rogers. Derivatives of eigenvalues and eigenvectors. *AIAA Journal*, 8(5):943–944, 1970.
- [41] Jeff S. Shamma. Anti-windup via constrained regulation with observers. In *Proceedings of the American Control Conference*, pages 2481–2485, San Diego, June 1999.
- [42] J. Marc Shewchun and Eric Feron. High performance control with position and rate limited actuators. Submitted to the *International Journal of Robust and Nonlinear Control*.
- [43] J. Marc Shewchun and Eric Feron. High performance bounded control. In *Proceedings of the American Control Conference*, volume 5, pages 3250–3245, Albuquerque, NM, June 4-6 1997.

- [44] Rodolfo Suárez, Julio Solís-Daun, and Jesús Álvarez. Stabilization of linear controllable systems by means of bounded continuous nonlinear feedback control. *Systems & Control Letters*, 23(6):403–410, 1994.
- [45] J. C. Willems. Least squares stationary optimal control and the algebraic Riccati equation. *IEEE Transactions on Automatic Control*, AC-16(6):621–634, 1971.
- [46] W. H. Wittrick. Rates of change of eigenvalues, with reference to buckling and vibration problems. *Journal of the Royal Aeronautical Society*, 66(621):590–591, 1962.
- [47] G. F. Wredenhagen. A new method of controller design for systems with input constraints using interpolation functions. In *Proceedings of the 33rd Conference on Decision and Control*, volume 2, pages 1024–1029, Lake Buena Vista, FL, December 14–16 1994.
- [48] G. F. Wredenhagen and P. R. Bélanger. Piecewise-linear LQ control for systems with input constraints. *Automatica*, 30(3):403–416, 1994.
- [49] Shao-Po Wu and Stephen Boyd. SDPSOL, a parser/solver for semidefinite programming and determinant maximization problems with matrix structure. User’s guide, Stanford University, Stanford, CA, May 31, 1996.
- [50] Shao-Po Wu, Lieven Vandenberghe, and Stephen Boyd. MAXDET, software for determinant maximization problems. User’s guide, Stanford University, Stanford, CA, May 24, 1996.
- [51] Lin Xie, Serge Shishkin, and Minyue Fu. Piecewise Lyapunov functions for robust stability of linear time-varying systems. *Systems & Control Letters*, 31(3):165–171, 1997.
- [52] A. L. Zelentsovsky. Nonquadratic Lyapunov functions for robust stability analysis of linear uncertain systems. *IEEE Transactions on Automatic Control*, AC-39(1):135–138, 1994.
- [53] David Zwillinger, editor. *CRC Standard Mathematical Tables and Formulae*. CRC Press, Inc., Boca Raton, FL, 30th edition, 1996.



# Appendix A

## Existence of Controllable Ellipsoid

In this section, the issue of existence of a controllable ellipsoid is treated by solving the following problem (where  $A, B$  are real matrices and the pair  $[A, B]$  is assumed controllable throughout):

**Problem A.1** Find a triplet  $(P, K_v, K_e)$ ,  $P > 0$  such that the following holds:

$$\begin{aligned} P(A - BK_v) + (A - BK_v)^T P &\leq 0 \\ -P(A - BK_e) - (A - BK_e)^T P &\leq 0 \end{aligned}$$

Noting the similarity of these relations to the algebraic Riccati equation, the possibility of utilizing previously established results in linear optimal control theory is investigated.

The first requirement for the solution to Problem A.1 is that  $P$  must be positive definite. In general, though, the solution to the algebraic Riccati equation is not unique, so that nothing can be said about the sign definiteness of all solutions. However, Willems notes that, for the algebraic Riccati equation, (A.1), there exists<sup>1</sup> a unique real symmetric solution matrix,  $P_+$  (the maximum solution), having as one of its properties that  $P_+ \geq P$  (i.e.,  $P_+ - P \geq 0$ ) for any other solution matrix,  $P$  [45].

**Remark A.1** The eigenvalues of  $P_+$  are real via the property that  $P_+$  is a real, symmetric matrix [3].

Furthermore, Kawasaki and Shimemura [28] provides the following useful lemma related to this maximum solution.

**Lemma A.1** [28] Let  $\lambda_1^-, \lambda_2^-, \dots, \lambda_j^-$  be the left half plane eigenvalues ( $\text{Re}(\lambda_i^-) \leq 0$ ) of  $A$  and  $\xi_1^-, \xi_2^-, \dots, \xi_j^-$  be the corresponding eigenvectors. The maximum solution  $P_+$  of the equation

$$PA + A^T P - PBR^{-1}B^T P = 0 \tag{A.1}$$

---

<sup>1</sup>Existence of a solution, for this simplified form of the algebraic Riccati equation, is guaranteed so long as  $R \geq 0$  [45].

(where  $R > 0$ ) satisfies

$$\text{null}(P_+) = \text{span}(\xi_1^-, \xi_2^-, \dots, \xi_j^-) \quad (\text{A.2})$$

where  $\text{span}(\xi_1^-, \xi_2^-, \dots, \xi_j^-)$  denotes the linear subspace spanned by vectors  $\xi_1^-, \xi_2^-, \dots, \xi_j^-$ .

Lemma A.1 is now used to give the positive definiteness requirement in Corollary A.1.

**Corollary A.1** *If the eigenvalues of  $A$  are contained in the open right half of the complex plane ( $\text{Re}(\lambda_i^-) > 0$ ), then the maximum solution,  $P_+$ , to the algebraic Riccati equation*

$$PA + A^T P - PBR^{-1}B^T P = 0 \quad (\text{A.3})$$

*is a positive definite matrix.*

*Proof.* Since the matrix  $P_0 = 0$  is a solution to (A.3), then  $P_+ \geq P_0 = 0$  (i.e.,  $P_+$  is positive semi-definite) by definition of the maximum solution. This implies that the eigenvalues of  $P_+$  (which are real via Remark A.1) are at least greater than or equal to zero, so it must be shown that the eigenvalues of  $P_+$  are strictly non-zero to prove strict positive definiteness.

Define  $\Lambda^-$  as the set of all left half plane eigenvalues of  $A$  and  $\Xi^-$  as the set of corresponding eigenvectors. Since the eigenvalues of  $A$  are contained in the open *right* half of the complex plane,  $\Lambda^-$  and  $\Xi^-$  are empty sets. From Lemma A.1, this implies

$$\text{null}(P_+) = \text{span}\{\emptyset\}$$

which further implies that  $P_+$  is full rank and that its eigenvalues are strictly non-zero, completing the proof. ■

For a general system, no assumptions can be made about the location of the eigenvalues of  $A$ . However, since the system is assumed controllable, a state feedback matrix,  $K_0$ , may be defined such that the eigenvalues of  $A - BK_0$  are contained in the open right half plane. By defining  $A_0 \equiv A - BK_0$ , the maximum solution,  $P_+$ , to the modified Riccati equation, (A.4), is positive definite via Corollary A.1.

$$PA_0 + A_0^T P - PBR^{-1}B^T P = 0 \quad (\text{A.4})$$

To apply this result to the first inequality of Problem A.1, the recoverable state feedback matrix is assumed to be of the form  $K_v = K_0 + \alpha_v R^{-1} B^T P_+$ , where the scalar  $\alpha_v$  is a real constant, and the necessary restriction on  $\alpha_v$  is investigated such that the recoverable inequality, (A.5), holds.

$$P_+ (A - BK_v) + (A - BK_v)^T P_+ \leq 0 \quad (\text{A.5})$$

↓

$$P_+ A_0 + A_0^T P_+ - 2\alpha_v P_+ B R^{-1} B^T P_+ \leq 0$$

↓

$$(P_+ A_0 + A_0^T P_+ - P_+ B R^{-1} B^T P_+) - (2\alpha_v - 1) P_+ B R^{-1} B^T P_+ \leq 0$$

↓

$$-(2\alpha_v - 1) P_+ B R^{-1} B^T P_+ \leq 0$$

Since  $R > 0$ ,  $-P_+ B R^{-1} B^T P_+$  is negative semi-definite. Hence, (A.5) holds for  $\alpha_v \geq \frac{1}{2}$ .

Similarly, the form  $K_e = K_0 + \alpha_e R^{-1} B^T P_+$  is assumed for the reachable state feedback matrix and the necessary restriction on  $\alpha_e$  considered.

$$-P_+ (A - B K_e) - (A - B K_e)^T P_+ \leq 0 \tag{A.6}$$

↓

$$-P_+ A_0 - A_0^T P_+ + 2\alpha_e P_+ B R^{-1} B^T P_+ \leq 0$$

↓

$$-(P_+ A_0 + A_0^T P_+ - P_+ B R^{-1} B^T P_+) + (2\alpha_e - 1) P_+ B R^{-1} B^T P_+ \leq 0$$

↓

$$(2\alpha_e - 1) P_+ B R^{-1} B^T P_+ \leq 0$$

Here, (A.6) holds for  $\alpha_e \leq \frac{1}{2}$ .

The results of this section are summarized in Corollary A.2.

**Corollary A.2** *A family of solutions,  $(P, K_v, K_e)$ ,  $P > 0$ , to the set of matrix inequalities*

$$P (A - B K_v) + (A - B K_v)^T P \leq 0$$

$$-P (A - B K_e) - (A - B K_e)^T P \leq 0$$

*is given by  $(P_+, K_0 + \alpha_v R^{-1} B^T P_+, K_0 + \alpha_e R^{-1} B^T P_+)$ , where  $P_+$  is the maximum solution to the algebraic Riccati equation*

$$P (A - B K_0) + (A - B K_0)^T P - P B R^{-1} B^T P = 0$$

*and the parameters  $R, K_0, \alpha_v, \alpha_e$  satisfy*

$$R > 0$$

$$\text{Re}[\text{eig}(A - B K_0)] > 0$$

$$\alpha_v \geq \frac{1}{2}$$

$$\alpha_e \leq \frac{1}{2}$$

## Appendix B

# Derivation of Ellipsoid Volume

## Cost Function

The objective of the optimization routine for the ellipsoidal method is to maximize the volume of the ellipsoid subject to the state and control constraints. The ellipsoid is expressed in (B.2) in terms of the ellipsoid function, (B.1), where  $P$  is an  $n \times n$  real, positive definite, symmetric matrix.

$$\theta(x) = x^T P x \quad (\text{B.1})$$

$$\mathcal{E}_1 = \{x | \theta(x) \leq 1\} \quad (\text{B.2})$$

Strictly speaking, it is not necessary that  $P$  be symmetric, only that it be positive definite. Any real, square matrix can be written as the sum of a symmetric matrix and a skew-symmetric matrix, (B.3), where  $P_H = P_H^T$  and  $P_{ss} = -P_{ss}^T$ .

$$P = P_H + P_{ss} \quad (\text{B.3})$$

However, the skew-symmetric component contributes nothing to the ellipsoid function.

$$\begin{aligned} \theta(x) &= x^T P x \\ &= x^T P_H x + x^T P_{ss} x \\ &= x^T P_H x \end{aligned}$$

Consequently, no degrees of freedom are lost by restricting to the symmetric form, but this does allow for simplifications to be made in computing the volume, (B.4).

$$V_{\mathcal{E}_1} = \int \int \cdots \int_{\mathcal{E}_1} dx_1 dx_2 \cdots dx_n \quad (\text{B.4})$$

Specifically, a linear change of variables will be introduced to convert the integral of an ellipsoid to an integral of a spheroid, for which explicit solutions are available.

It is known that, for any real, positive definite symmetric matrix,  $P$ , there exists a unitary transformation,  $U$ , mapping  $P$  to its diagonal form.

$$U^T P U = \Lambda \quad (\text{B.5})$$

where

$$\det U = 1$$

$$U^T U = I$$

$$\Lambda = \text{diag} \{ \lambda_1, \lambda_2, \dots, \lambda_n \}$$

and the  $\lambda_i$  are the eigenvalues of  $P$ , all of which are positive, real numbers due to the properties of positive definite matrices and symmetric matrices, respectively.

Furthermore, it is possible to define a transformation,  $M$ , mapping  $P$  to the identity matrix.

$$M^T P M = I \quad (\text{B.6})$$

where

$$M = U \left( \Lambda^{1/2} \right)^{-1}$$

$$\Lambda^{1/2} = \text{diag} \{ \sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots, \sqrt{\lambda_n} \}$$

Invoking the property of determinants for two square matrices,  $P_1, P_2$ , that  $|P_1 P_2| = |P_1| |P_2|$  [11, pg. 128], relation (B.6) is noted, where the last step is possible since a positive definite, square matrix is invertible.

$$|M^T P M| = |I|$$

↓

$$|M| = 1/\sqrt{|P|} \quad (\text{B.7})$$

↓

$$|M| = \sqrt{|P^{-1}|} \quad (\text{B.8})$$

Define a state transformation as in (B.9),

$$z = M^{-1} x \quad (\text{B.9})$$

such that the ellipsoid can be written as in (B.10).

$$\theta(z) = (Mz)^T P (Mz) \quad (\text{B.10})$$

$$= z^T I z \quad (\text{B.11})$$

This change of variables transforms the ellipsoidal set,  $\mathcal{E}_1$ , into the special case of a spherical set,  $\mathcal{S}_1$ .

$$\mathcal{S}_1 = \{z | \theta(z) \leq 1\} \quad (\text{B.12})$$

Consequently, the integral equation for the ellipsoid volume may be rewritten.

$$V_{\mathcal{E}_1} = \int \int \cdots \int_{\mathcal{E}_1} dx_1 dx_2 \cdots dx_n \quad (\text{B.13})$$

$$= \int \int \cdots \int_{\mathcal{S}_1} \begin{vmatrix} \frac{\partial x_1}{\partial z_1} & \frac{\partial x_2}{\partial z_1} & \cdots & \frac{\partial x_n}{\partial z_1} \\ \frac{\partial x_1}{\partial z_2} & \frac{\partial x_2}{\partial z_2} & \cdots & \frac{\partial x_n}{\partial z_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial z_n} & \frac{\partial x_2}{\partial z_n} & \cdots & \frac{\partial x_n}{\partial z_n} \end{vmatrix} dz_1 dz_2 \cdots dz_n \quad (\text{B.14})$$

$$= \int \int \cdots \int_{\mathcal{S}_1} |M^T| dz_1 dz_2 \cdots dz_n \quad (\text{B.15})$$

$$= \sqrt{|P^{-1}|} \int \int \cdots \int_{\mathcal{S}_1} dz_1 dz_2 \cdots dz_n \quad (\text{B.16})$$

$$= \sqrt{|P^{-1}|} V_{\mathcal{S}_1} \quad (\text{B.17})$$

The explicit equation for the volume of a spheroid of dimension  $n$ , (B.18), has been previously established [53, pg. 315] (Note: To keep with standard notation, the symbol  $\Gamma$  is used uniquely in this section to represent the gamma function; elsewhere in the text it is used in reference to a linear state constraint. Also, the symbol  $r$  refers to the radius of the sphere, which, for this case, (B.10), (B.12), has a value of 1.)

$$V_{\mathcal{S}_1}^n(r) = \frac{2\pi^{n/2} r^n}{n\Gamma(n/2)} \quad (\text{B.18})$$

where

$$\Gamma\left(\frac{n}{2}\right) = \begin{cases} \left(\frac{n-1}{2}\right)!, & n = 2, 4, 6, \dots \\ \frac{(n-1)! \sqrt{\pi}}{\left(\frac{n-1}{2}\right)! 2^{(n-1)}}, & n = 1, 3, 5, \dots \end{cases} \quad (\text{B.19})$$

It is therefore apparent (as expected) that the volume of a unit spheroid is a constant for any fixed number of states,  $n$ , implying that the ellipsoid volume is proportional to the function, (B.20).

$$\hat{f} = \sqrt{|P^{-1}|} \quad (\text{B.20})$$

However, it is noted that  $\hat{f}$  maps the parameters of  $P$  to only positive values of  $R^1$ . In particular, resolution of  $\partial\hat{f}/\partial P$  (when searching for optimal parameters) becomes increasingly difficult for  $\hat{f} \leq 1$ . This suggests that the accuracy of a search would be affected by any pre-scaling of the states (for example, to set the state bounds as the unit hypercube). To minimize this effect, the following equivalent objective functions are used, where the  $1/2$  from the square root, which simply scales these functions, is omitted for simplicity.

$$\max \log |P^{-1}| \tag{B.21}$$

$$\min \{-\log |P^{-1}|\} \tag{B.22}$$

## Appendix C

# Derivation of State Constraint Inequalities

This section details how bounds on the states are converted to explicit bounds on the ellipsoidal approximating set's size and shape. Specifically, if the ellipsoidal set is denoted as  $\mathcal{E}$  and the linear state constraint by  $G$ , then the following problem is to be solved.

**Problem C.1** *Determine the restrictions on the size and shape of  $\mathcal{E}$  such that every point  $x \in \mathcal{E}$  satisfies the constraint  $G$ .*

### C.1 Level Sets

In essence, the constraint development centers on finding a tangent point on the boundary of the ellipsoidal set which is parallel to the affine boundary constraint, then determining the necessary restriction(s) so that this tangent lies interior to the constraint line. Formal development of the constraint involves the concept of level sets of functions.

#### C.1.1 Ellipsoidal Set Boundary as a Level Set

Define the ellipsoid function, ellipsoidal set, and boundary of the ellipsoidal set as in (C.1), (C.2), and (C.3), respectively.

$$\theta(x) = x^T P x \tag{C.1}$$

$$\mathcal{E}_\beta = \{x | \theta(x) \leq \beta\} \tag{C.2}$$

$$\partial\mathcal{E}_\beta = \{x | \theta(x) = \beta\} \tag{C.3}$$



With respect to the problem of finding the maximal ellipsoidal set, the parameter  $\beta$  is a scaling factor adjusting the size of the ellipsoid. Without loss of generality, then, the value  $\beta = 1$  is chosen to define the maximal ellipsoid.

### C.1.2 Linear Constraint Boundary as a Level Set

A linear state constraint can be defined as in (C.4),

$$\underline{\delta}_i \leq W_i x \leq \bar{\delta}_i \quad (\text{C.4})$$

where single-sided constraints can be described by appropriately defining one of the  $\delta$  parameters to  $\pm\infty$ , and where the following properties hold (Note that the strict inequalities on  $\underline{\delta}$ ,  $\bar{\delta}$  guarantee inclusion of the origin.)

$$\begin{aligned} |W_i| &= 1 \\ \underline{\delta}_i &< 0 \\ \bar{\delta}_i &> 0 \end{aligned}$$

Equation (C.4) can be split as in (C.5)

$$\begin{aligned} \Gamma_{i,1} x &\leq 1 \\ \Gamma_{i,2} x &\leq 1 \end{aligned} \quad (\text{C.5})$$

where

$$\begin{aligned} \Gamma_{i,1} &= W_i / \bar{\delta}_i \\ \Gamma_{i,2} &= -W_i / \underline{\delta}_i \end{aligned} \quad (\text{C.6})$$

In general, then, a linear inequality constraint (containing the origin) can be written as in (C.7).

$$\Gamma x \leq 1 \quad (\text{C.7})$$

Consequently, define the linear constraint function, constraint set, and boundary of the constraint set as in (C.8), (C.9), and (C.10), respectively.

$$\gamma(x) = \Gamma x \quad (\text{C.8})$$

$$G_\alpha = \{x | \gamma(x) \leq \alpha\} \quad (\text{C.9})$$

$$\partial G_\alpha = \{x | \gamma(x) = \alpha\} \quad (\text{C.10})$$

In light of (C.7), attention is focused on the unit level subset and its boundary ( $\alpha = 1$ ).

Lemma C.1 highlights a property of these regions which is necessary for the development of the constraints.

### Lemma C.1

$$G_{\alpha_1} \subseteq G_{\alpha_2} \Leftrightarrow \alpha_1 \leq \alpha_2$$

*Proof.* ( $\Rightarrow$ ) Assume that  $G_{\alpha_1} \subseteq G_{\alpha_2}$  but that  $\alpha_1 > \alpha_2$ . Then, there exists some  $x_0 \in G_{\alpha_1}$  for which  $\gamma(x_0) = \alpha_0$ ,  $\alpha_2 < \alpha_0 \leq \alpha_1$ . However, this implies that  $x_0 \notin G_2$ , violating the original assumption. Thus,  $G_1 \subseteq G_2 \Rightarrow \alpha_1 \leq \alpha_2$  is shown by contradiction.

( $\Leftarrow$ ) By definition,  $\gamma(x_0) \leq \alpha_1 \forall x_0 \in G_{\alpha_1}$ . Since  $\alpha_1 \leq \alpha_2$ , this implies that  $\gamma(x_0) \leq \alpha_2 \forall x_0 \in G_{\alpha_1}$ , hence  $G_{\alpha_1} \subseteq G_{\alpha_2}$ , completing the proof. ■

## C.2 Derivation of Inequality

For preciseness, the following lemma is stated to formalize the state constraint as a set relation. The proof is intuitive and, consequently, omitted here.

**Lemma C.2** *The set of points defining an ellipsoid,  $\mathcal{E}_1$ , satisfy a state constraint,  $G_1$ , if and only if  $\mathcal{E}_1 \subset G_1$ .*

The objective of this line of investigation, then, is to develop a constraint on  $P$  ensuring that Lemma C.2 holds. In light of Lemma C.1, the following definition is stated relating the boundary constraints to the ellipsoid,  $\mathcal{E}_1$ .

**Definition C.1** *Let  $\mathcal{A} = \{\alpha | \mathcal{E}_1 \subset G_\alpha\}$ . The minimal bounding value,  $\underline{\alpha}$ , is defined as  $\underline{\alpha} = \min \mathcal{A}$ .*

It is now possible to propose the following theorem.

### Theorem C.1

$$\mathcal{E}_1 \subset G_1 \Leftrightarrow \underline{\alpha} \leq 1$$

*Proof.* ( $\Rightarrow$ ) Assume that  $\mathcal{E}_1 \subset G_1$ . By definition of  $\underline{\alpha}$ ,  $G_{\underline{\alpha}} \subseteq G_1$ . Lemma C.1 then gives  $\underline{\alpha} \leq 1$ .

( $\Leftarrow$ ) Assume that  $\underline{\alpha} \leq 1$ . From Lemma C.1, this implies  $G_{\underline{\alpha}} \subseteq G_1$ . By definition of  $\underline{\alpha}$ ,  $\mathcal{E}_1 \subset G_{\underline{\alpha}}$ , which gives  $\mathcal{E}_1 \subset G_1$ , completing the proof. ■

The practical application of Theorem C.1 is to find some constraint on  $P$  such that  $\underline{\alpha}(P) \leq 1$ , ensuring that the corresponding ellipsoid,  $\mathcal{E}_1(P)$  is within  $G_1$ . The steps in converting an inequality constraint on the states to a constraint on the ellipsoid are, first, to determine the minimal bounding value,  $\underline{\alpha}(P) \leq 1$ , and, second, to develop a constraint on  $P$  such that  $\underline{\alpha} < 1$ .

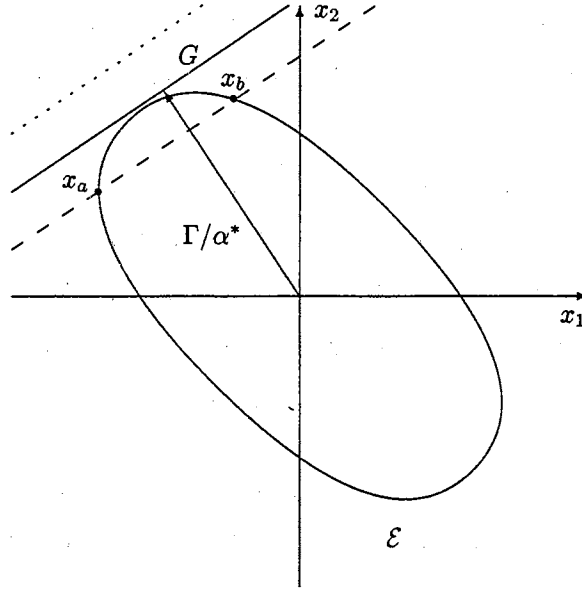


Figure C.1: Illustration of Boundary Intersection.

Lemma C.3 formally states the idea that the minimal bounding value,  $\underline{\alpha}$ , is equal to the value of  $\alpha$  for which the boundary of the linear constraint,  $\partial G_\alpha$ , is tangent to the boundary of the ellipse,  $\partial \mathcal{E}_1$ .

**Lemma C.3** For some  $\alpha^* > 0$ ,  $\alpha^* = \underline{\alpha}$  if and only if  $\partial G_{\alpha^*} \cap \partial \mathcal{E}_1$  is a single point.

This lemma implies that, for the minimal bounding value,  $\partial G_{\underline{\alpha}}$  is tangent to  $\partial \mathcal{E}_1$ . Proof of this lemma is somewhat intuitive, and utilizes concepts illustrated in Figure C.1.

*Proof.* ( $\Rightarrow$ ) It is known that the intersection of a hyperplane and the boundary of an ellipsoid defined by a *positive definite* matrix,  $P$ , is (i) the null set, (ii) a single point, or (iii) a “ring.” These cases are considered independently.

Suppose  $\alpha^* = \underline{\alpha}$  ( $\partial G_{\underline{\alpha}} = \partial G_{\alpha^*}$ ), but that  $\partial G_{\underline{\alpha}} \cap \partial \mathcal{E}_1$  is the null set. This implies (a) that  $\mathcal{E}_1 \subset G_{\underline{\alpha}}$  and (b) that there exists some positive quantity,  $\Delta\alpha$ , such that  $\partial \mathcal{E}_1 \subset G_{\underline{\alpha} - \Delta\alpha} \subset G_{\underline{\alpha}}$ . Consequently,  $\underline{\alpha}$  is not the minimum bounding value, contradicting the assumption.

Similarly, assume that  $\alpha^* = \underline{\alpha}$ , but that  $\partial G_{\underline{\alpha}} \cap \partial \mathcal{E}_1$  is more than one point. For a second order system, the intersection is two points,  $\{x_a, x_b\}$ , but for higher dimensions, the intersection is a “ring” formed by an infinite number of points, any two of which can be chosen as  $\{x_a, x_b\}$ . Defining a closed set,  $\bar{\mathcal{X}}$ , as  $\bar{\mathcal{X}} = \{\lambda x_a + (1 - \lambda) x_b | 0 \leq \lambda \leq 1\}$ , convexity of the ellipsoidal set implies that  $\bar{\mathcal{X}} \subset \mathcal{E}_1$ , and convexity of  $\partial G_{\underline{\alpha}}$  implies that  $\bar{\mathcal{X}} \subset \partial G_{\underline{\alpha}}$ . Furthermore, it can be shown that the open set  $\mathcal{X}$ , where  $\mathcal{X} = \{\lambda x_a + (1 - \lambda) x_b | 0 < \lambda < 1\}$ , is strictly interior to  $\partial \mathcal{E}_1$ , thus implying that there

are points in a neighborhood of elements of  $\bar{\mathcal{X}}$  which are interior to  $\mathcal{E}_1$  but exterior to  $G_{\underline{\alpha}}$  (i.e.,  $G_{\underline{\alpha}} \cap \mathcal{E}_1 \neq \mathcal{E}_1$ , see Figure C.1), violating the definition of  $\underline{\alpha}$ . Consequently,  $\partial G_{\underline{\alpha}} \cap \partial \mathcal{E}_1$  must be a single point.

( $\Leftarrow$ ) Assume that  $\partial G_{\alpha^*} \cap \partial \mathcal{E}_1$  is a single point but that  $\alpha^* \leq 0$ . This implies that  $\partial G_{\alpha^*}$  passes through the origin (in the case of equality) or that  $G_{\alpha^*}$  does not contain the origin, hence  $\alpha^*$  must be positive. If  $\partial G_{\alpha^*} \cap \partial \mathcal{E}_1$  is a single point and  $\alpha^* > 0$ , the proof follows similarly as above to show that  $\alpha^* = \underline{\alpha}$ . (Note: there are two solutions,  $\pm \underline{\alpha}$  yielding a single intersection point (on opposite sides of the ellipse), necessitating the positive restriction.) ■

It is widely known that the gradient of a function gives normals to the function's isoclines [34, pp. 149-150]. Consequently, the gradient of the function  $\theta(x)$  gives the inward pointing normals to the  $\partial \mathcal{E}_\beta$ 's, as defined in (C.11).

$$\nabla \theta(x) = 2Px \quad (\text{C.11})$$

Similarly, the gradient of the function  $\gamma(x)$  gives the inward pointing normal to the  $\partial G_\alpha$ 's.

$$\nabla \gamma(x) = \Gamma^T \quad (\text{C.12})$$

The linear constraint bound is parallel to the tangent hyperplane of the ellipse at  $x^*$  if the gradients are parallel, (C.13),

$$Px^* \propto \Gamma^T \quad (\text{C.13})$$

or, equivalently, if the gradient of one is normal to the null space of the gradient of the other, as in (C.14).

$$(\Gamma^\perp)^T Px^* = 0, \quad \Gamma^\perp = \text{null}\{\Gamma\} \quad (\text{C.14})$$

The linear constraint bound is the tangent hyperplane of the ellipse at  $x^*$  if (C.15) holds.

$$\begin{cases} (\Gamma^\perp)^T Px^* = 0 \\ (x^*)^T P(x^*) = 1 \end{cases} \quad (\text{C.15})$$

The solution to the first relation in (C.15) proceeds as follows:

$$\begin{aligned} (\Gamma^\perp)^T Px^* &= 0 \\ \Downarrow \\ (\Gamma^\perp)^T z &= 0 \\ \Downarrow \\ z^* &= \Gamma^T / \alpha^* \quad (\text{null}\{\Gamma^\perp\} = \Gamma) \end{aligned}$$

$$\begin{aligned}
& \Downarrow \\
& \Gamma^T / \alpha^* = Px^* \\
& \Downarrow \\
& x^* = P^{-1} \Gamma^T / \alpha^* \tag{C.16}
\end{aligned}$$

Substituting (C.16) into the second relation yields to intersection solution, (C.17), where, via Lemma C.3,  $\underline{\alpha}$  is substituted for  $\alpha^*$ .

$$\begin{aligned}
& (\Gamma P^{-1} / \underline{\alpha}) P (P^{-1} \Gamma^T / \underline{\alpha}) = 1 \\
& \Downarrow \\
& (\underline{\alpha})^2 = \Gamma P^{-1} \Gamma^T \tag{C.17}
\end{aligned}$$

Theorem C.1 requires  $\underline{\alpha} \leq 1$ , yielding the state constraint inequality, (C.18).

$$\Gamma P^{-1} \Gamma^T \leq 1 \tag{C.18}$$

### C.3 Derivation of Control Constraint Inequalities

Since linear state feedback is used, the control law is of the form  $u = -Kx$ . Consequently, the input amplitude constraint may be written as a pair of linear state constraints as in (C.19).

$$\begin{cases} \mathcal{U}_1^+ = \{x | [-K / (+\bar{u})] x \leq 1\} \\ \mathcal{U}_1^- = \{x | [-K / (-\bar{u})] x \leq 1\} \end{cases} \tag{C.19}$$

Substituting  $\pm K / \bar{u}$  for  $\Gamma$  in (C.18) yields the control constraint inequality, (C.20).

$$(K / \bar{u}) P^{-1} (K / \bar{u})^T \leq 1 \tag{C.20}$$

## Appendix D

# Gradients of Cost Functions and Constraints

Matlab's `constr.m` optimization routine uses gradients to calculate the search direction. If analytical gradients are not provided, the routine automatically implements a finite difference approximation. However, the search is generally more stable and convergence faster if the analytical gradients are supplied. In this chapter, expressions for the gradients of the cost functions and constraint inequalities are derived.

### D.1 Preliminaries

As implemented in the code of Appendices F and G, the parameter space is defined as a vector composed of the component elements of the pseudo-ellipse (cf. (5.21)) and state feedback matrices. The vector is formed by stacking the lower diagonal columns of the ellipse matrix on top of the feedback gain elements, as in (D.1).

$$\hat{P} = \begin{bmatrix} \hat{p}_{11} & \hat{p}_{21} \\ \hat{p}_{21} & \hat{p}_{22} \end{bmatrix}, K = \begin{bmatrix} k_{11} & k_{12} \end{bmatrix} \Rightarrow X = \begin{bmatrix} \hat{p}_{11} & \hat{p}_{21} & \hat{p}_{22} & k_{11} & k_{12} \end{bmatrix}^T \quad (\text{D.1})$$

Gradients of any of the scalar functions are expressed as column vectors the size of  $X$ , each element of which can be computed as a derivative with respect to a scalar value,  $X_i$ .

## D.2 Identities

Several matrix identities and special forms are used to derive the gradients. The following are all taken from [6, pp. 50, 59-60, 66-68]. In these identities, bold face letters denote matrices (upper case) and vectors (lower case), while normal type denotes a scalar. Subscripted letters refer to an individual matrix/vector element.

**Special Forms** The matrix  $(\mathbf{E}\mathbf{L})_{ij}^{\mathbf{Y}}$  denotes a matrix the same dimensions as  $\mathbf{Y}$  whose elements are all zeros except for the  $(ij)^{th}$  element, which is one.

### Traces of Matrices

$$\text{tr}(\mathbf{Y}) = \sum_i y_{ii} \quad (\text{D.2})$$

$$\text{tr}(\mathbf{X}\mathbf{Y}\mathbf{Z}) = \text{tr}(\mathbf{Y}\mathbf{Z}\mathbf{X}) = \text{tr}(\mathbf{Z}\mathbf{X}\mathbf{Y}) \quad (\text{D.3})$$

$$\text{tr}(\mathbf{X} + \delta\mathbf{Y}) = \text{tr}(\mathbf{X}) + \delta\text{tr}(\mathbf{Y}) \quad (\text{D.4})$$

$$\text{tr}[(\mathbf{E}\mathbf{L})_{ij}^{\mathbf{Y}} \mathbf{Y}] = \text{tr}[\mathbf{Y} (\mathbf{E}\mathbf{L})_{ji}^{\mathbf{Y}}] = y_{ij} \quad (\text{D.5})$$

### Derivatives of Matrices

$$\partial\mathbf{Y}/\partial y_{ij} = (\mathbf{E}\mathbf{L})_{ij}^{\mathbf{Y}} \quad (\text{D.6})$$

$$\partial\mathbf{Y}^T/\partial y_{ij} = (\mathbf{E}\mathbf{L})_{ji}^{\mathbf{Y}^T} \quad (\text{D.7})$$

$$\partial\mathbf{X}\mathbf{Y}/\partial x = (\partial\mathbf{X}/\partial x)\mathbf{Y} + \mathbf{X}(\partial\mathbf{Y}/\partial x) \quad (\text{D.8})$$

$$\partial\mathbf{W}\mathbf{Y}/\partial x = \mathbf{W}(\partial\mathbf{Y}/\partial x) \quad (\text{if } \mathbf{W} \text{ not a function of } x) \quad (\text{D.9})$$

$$\partial\mathbf{W}\mathbf{Y}\mathbf{W}^T/\partial x = \mathbf{W}(\partial\mathbf{Y}/\partial x)\mathbf{W}^T \quad (\text{D.10})$$

$$\partial\mathbf{Y}^T\mathbf{W}\mathbf{Y}/\partial x = (\partial\mathbf{Y}^T/\partial x)\mathbf{W}\mathbf{Y} + \mathbf{Y}^T\mathbf{W}(\partial\mathbf{Y}/\partial x) \quad (\text{D.11})$$

$$\partial\text{tr}(\mathbf{Y})/\partial\mathbf{Y} = \mathbf{I} \quad (\text{D.12})$$

$$\partial\text{tr}(\mathbf{Y}^T\mathbf{A}\mathbf{Y})/\partial\mathbf{Y} = \mathbf{A}\mathbf{Y} + \mathbf{A}^T\mathbf{Y} \quad (\text{D.13})$$

$$\partial\log_e |\mathbf{Y}|/\partial\mathbf{Y} = (\mathbf{Y}^T)^{-1} \quad (\text{D.14})$$

$$\partial\mathbf{Y}^{-1}/\partial x = -\mathbf{Y}^{-1}(\partial\mathbf{Y}/\partial x)\mathbf{Y}^{-1} \quad (\text{D.15})$$

$$\partial v/\partial x = \text{tr}[(\partial v/\partial\mathbf{Z})(\partial\mathbf{Z}^T/\partial x)] \quad (\text{D.16})$$

Also required are analytic derivatives of the eigenvalues of matrices with respect to the search parameters. A large body of work has been published investigating the derivatives of eigenvalues and eigenvectors with respect to matrix parameters, particularly as it is related to flexible structures

(see, e.g., [15], [26], [39], [40]). Wittrick [46] shows that, if  $\xi_i$  is a normalized ( $\xi_i^T \xi_i = 1$ ) eigenvector of a self-adjoint matrix  $\mathbf{Y}$ , and  $\lambda_i$  is its corresponding eigenvalue, then the derivative of the eigenvalue with respect to the  $(ij)^{th}$  element of  $\mathbf{Y}$  is given in (D.17).

$$\frac{\partial \lambda_i}{\partial y_{ij}} = \xi_i^T \left( \frac{\partial \mathbf{Y}}{\partial y_{ij}} \right) \xi_i \quad (\text{D.17})$$

Analytical expressions for the gradients of the positive definiteness and invariance constraints only require the derivatives of the eigenvalues. However, the reduced dimension state constraint matrix,  $\Gamma_i^R$ , in (5.7) is a function of the state feedback matrix (a search parameter) via  $W_i$ , the null space of  $\Gamma_i(A - BK_v)$  (or  $K_e$ ). The null space corresponds to the eigenvectors of  $\Gamma_i(A - BK_v)$  with eigenvalues of 0. Thus, the analytic derivative of the state constraint requires the derivative of these eigenvectors, which is complicated by the fact that  $\Gamma_i(A - BK_v)$  is non-self-adjoint (asymmetric) and the eigenvectors in question belong to repeated eigenvalues of multiplicity  $(n - 1)$ .

Eigenvector derivatives have been investigated in [33] for non-self-adjoint systems with distinct eigenvalues and in [14] for self-adjoint systems with repeated eigenvalues, but the case where both conditions occur is apparently still an open research issue. Furthermore, the algorithms presented in [14], [33] involve multiple steps, adding to the computational burden for an optimization search. Given the unresolved theoretical issue of eigenvector derivatives of non-self-adjoint systems with repeated eigenvalues as well as the desire to minimize the search time, a finite difference approximation to the eigenvector derivatives is used.

### D.3 Analytic Derivatives

The derivatives of the objective functions and constraints are presented here for the case of the recoverable ellipsoidal/semi-ellipsoidal set, though the subscript “ $v$ ” has been dropped to avoid confusion with the indexing subscripts. For the reachable case, all that is required is a sign change (and change of feedback matrix) on the invariance constraint. Before presenting the equations, a few remarks are necessary regarding the notation used.

- The ellipsoid matrix,  $P$ , is defined in terms of an auxiliary symmetric matrix,  $\hat{P}$  via the relation  $P = \hat{P}\hat{P}$ , (5.21). Because of the forced-symmetric structure of  $\hat{P}$  ( $\hat{p}_{ji} \equiv \hat{p}_{ij}$ ), the partial derivatives of  $P$  with respect to the elements of  $\hat{P}$  are different for on- and off-diagonal elements, as shown in (D.18).

$$\frac{\partial \hat{P}}{\partial \hat{p}_{ij}} = \begin{cases} (\mathbf{EL})_{ij}^{\hat{P}}, & i = j \\ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}}, & i \neq j \end{cases} \quad (\text{by (D.6),(D.7)}) \quad (\text{D.18})$$



This relation then implies that the derivatives of  $P$  are also dependent on the parameter location, (D.19).

$$\begin{aligned}
\frac{\partial P}{\partial \hat{p}_{ij}} &= \frac{\partial (\hat{P}\hat{P})}{\partial \hat{p}_{ij}} \\
&= \hat{P} \left( \frac{\partial \hat{P}}{\partial \hat{p}_{ij}} \right) + \left( \frac{\partial \hat{P}}{\partial \hat{p}_{ij}} \right) \hat{P} \quad (\text{by (D.8)}) \\
&= \begin{cases} \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P}, & i = j \\ \hat{P} [(\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}}] + [(\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}}] \hat{P}, & i \neq j \end{cases} \quad (\text{D.19})
\end{aligned}$$

- In all cases,  $A_{cl} = A - B(K + K^0)$ .
- The subscript “ $i$ ” has been dropped on the state constraint matrix,  $\Gamma$  to avoid confusion with the parameter indexing subscripts.
- Matlab requires constraints to be written in the form  $g(x) \leq 0$ , which means that the constraints derived in the preceding chapters must be expressed as  $f(x) - 1 \leq 0$ , but this constant does not alter the expression of the gradient of the constraint function.

#### Ellipsoidal Set Cost Function

$$\begin{aligned}
\frac{\partial (\log_e |P|)}{\partial \hat{p}_{ij}} &= \text{tr} \left\{ \left[ \frac{\partial (\log_e |P|)}{\partial P} \right] \left( \frac{\partial P}{\partial \hat{p}_{ij}} \right) \right\} \quad (\text{by (D.16)}) \\
&= \text{tr} \left( P^{-1} \frac{\partial P}{\partial \hat{p}_{ij}} \right) \quad (\text{by (D.14)}) \\
&= \begin{cases} \text{tr} \left[ (\hat{P}\hat{P})^{-1} \left( \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right) \right], & i = j \\ \text{tr} \left[ (\hat{P}\hat{P})^{-1} \left( \hat{P} [(\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}}] \right. \right. \\ \quad \left. \left. + [(\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}}] \hat{P} \right) \right], & i \neq j \end{cases} \\
&= \begin{cases} \text{tr} \left[ (\hat{P}^{-1} \hat{P}^{-1}) \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right] + \text{tr} \left[ (\hat{P}^{-1} \hat{P}^{-1}) (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right], & i = j \\ \text{tr} \left[ (\hat{P}^{-1} \hat{P}^{-1}) \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right] + \text{tr} \left[ (\hat{P}^{-1} \hat{P}^{-1}) \hat{P} (\mathbf{EL})_{ji}^{\hat{P}} \right] \\ \quad + \text{tr} \left[ (\hat{P}^{-1} \hat{P}^{-1}) (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] + \text{tr} \left[ (\hat{P}^{-1} \hat{P}^{-1}) (\mathbf{EL})_{ji}^{\hat{P}} \hat{P} \right], & i \neq j \end{cases} \quad (\text{by (D.4)}) \\
&= \begin{cases} 2 \text{tr} \left[ \hat{P}^{-1} (\mathbf{EL})_{ij}^{\hat{P}} \right], & i = j \\ 2 \text{tr} \left[ \hat{P}^{-1} (\mathbf{EL})_{ij}^{\hat{P}} \right] + 2 \text{tr} \left[ \hat{P}^{-1} (\mathbf{EL})_{ji}^{\hat{P}} \right], & i \neq j \end{cases} \quad (\text{by (D.3)}) \\
&= \begin{cases} 2 (\hat{P}^{-1})_{ij} & i = j \\ 2 (\hat{P}^{-1})_{ij} + 2 (\hat{P}^{-1})_{ji} & i \neq j \end{cases} \quad (\text{by (D.5)}) \\
&= \begin{cases} 2 (\hat{P}^{-1})_{ij} & i = j \\ 4 (\hat{P}^{-1})_{ij} & i \neq j \end{cases} \quad (\text{since } \hat{P} = \hat{P}^T) \quad (\text{D.20})
\end{aligned}$$

$$\frac{\partial (\log_e |P|)}{\partial k_{ij}} = 0 \quad (\text{D.21})$$

### Semi-Ellipsoidal Set Cost Function

$$\begin{aligned} \frac{\partial [\log_e \text{tr}(P)]}{\partial \hat{p}_{ij}} &= \left[ \frac{1}{\text{tr}(P)} \right] \left[ \frac{\partial \text{tr}(P)}{\partial \hat{p}_{ij}} \right] \\ &= \left[ \frac{1}{\text{tr}(P)} \right] \cdot \text{tr} \left[ \frac{\partial \text{tr}(P)}{\partial P} \frac{\partial P}{\partial \hat{p}_{ij}} \right] \quad (\text{by (D.16)}) \\ &= \left[ \frac{1}{\text{tr}(P)} \right] \cdot \text{tr} \left[ \frac{\partial P}{\partial \hat{p}_{ij}} \right] \quad (\text{by (D.12)}) \\ &= \begin{cases} \left[ \frac{1}{\text{tr}(P)} \right] \cdot \text{tr} \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right], & i = j \\ \left[ \frac{1}{\text{tr}(P)} \right] \cdot \text{tr} \left\{ \hat{P} \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] + \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \hat{P} \right\}, & i \neq j \end{cases} \\ &= \begin{cases} \left[ \frac{1}{\text{tr}(P)} \right] \left\{ \text{tr} \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right] + \text{tr} \left[ (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] \right\}, & i = j \\ \left[ \frac{1}{\text{tr}(P)} \right] \left\{ \text{tr} \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right] + \text{tr} \left[ (\mathbf{EL})_{ji}^{\hat{P}} \hat{P} \right] \right. \\ \quad \left. + \text{tr} \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right] + \text{tr} \left[ (\mathbf{EL})_{ji}^{\hat{P}} \hat{P} \right] \right\}, & i \neq j \end{cases} \quad (\text{by (D.4)}) \\ &= \begin{cases} \left[ \frac{1}{\text{tr}(P)} \right] \left\{ 2 \text{tr} \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right] \right\}, & i = j \\ \left[ \frac{1}{\text{tr}(P)} \right] \left\{ 2 \text{tr} \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right] + 2 \text{tr} \left[ (\mathbf{EL})_{ji}^{\hat{P}} \hat{P} \right] \right\}, & i \neq j \end{cases} \quad (\text{by (D.3)}) \\ &= \begin{cases} \left[ \frac{1}{\text{tr}(P)} \right] (2\hat{p}_{ij}), & i = j \\ \left[ \frac{1}{\text{tr}(P)} \right] (2\hat{p}_{ij} + 2\hat{p}_{ji}), & i \neq j \end{cases} \quad (\text{by (D.5)}) \\ &= \begin{cases} (2\hat{p}_{ij}) / \left[ \text{tr}(\hat{P}\hat{P}) \right], & i = j \\ (4\hat{p}_{ij}) / \left[ \text{tr}(\hat{P}\hat{P}) \right], & i \neq j \end{cases} \quad (\text{since } \hat{P} = \hat{P}^T) \end{aligned} \quad (\text{D.22})$$

$$\frac{\partial (\log_e |P|)}{\partial k_{ij}} = 0 \quad (\text{D.23})$$

**Invariance Constraint** Let  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  and  $\Xi = \{\xi_1, \xi_2, \dots, \xi_n\}$  be the set of eigenvalues and eigenvectors of  $PA_{cl} + A_{cl}^T P$ .

$$\begin{aligned} \frac{\partial \lambda_k}{\partial \hat{p}_{ij}} &= \xi_k^T \left[ \frac{\partial (PA_{cl} + A_{cl}^T P)}{\partial \hat{p}_{ij}} \right] \xi_k \quad (\text{by (D.10), (D.17)}) \\ &= \xi_k^T \left( \frac{\partial P}{\partial \hat{p}_{ij}} A_{cl} + A_{cl}^T \frac{\partial P}{\partial \hat{p}_{ij}} \right) \xi_k \quad (\text{by (D.9)}) \\ &= \begin{cases} \xi_k^T \left\{ \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] A_{cl} \right. \\ \quad \left. + A_{cl}^T \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] \right\} \xi_k, & i = j \\ \xi_k^T \left\{ \left[ \hat{P} \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \right. \right. \\ \quad \left. \left. + \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \hat{P} \right] A_{cl} \right. \\ \quad \left. + A_{cl}^T \left\{ \hat{P} \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \right. \right. \\ \quad \left. \left. + \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \hat{P} \right\} \right\} \xi_k, & i \neq j \end{cases} \end{aligned} \quad (\text{D.24})$$

$$\begin{aligned}
\frac{\partial \lambda_k}{\partial k_{ij}} &= \xi_k^T \left[ \frac{\partial (PA_{cl} + A_{cl}^T P)}{\partial k_{ij}} \right] \xi_k \quad (\text{by (D.10),(D.17)}) \\
&= \xi_k^T \left[ P \frac{\partial (A - BK)}{\partial k_{ij}} + \frac{\partial (A - BK)^T}{\partial k_{ij}} P \right] \xi_k \quad (\text{by (D.9)}) \\
&= \xi_k^T \left\{ (\hat{P}\hat{P}) \left[ -B(\mathbf{EL})_{ij}^K \right] + \left[ -B(\mathbf{EL})_{ij}^K \right]^T (\hat{P}\hat{P}) \right\} \xi_k \quad (\text{by (D.8),(D.6)}) \quad (\text{D.25})
\end{aligned}$$

**Positive Definiteness Constraint** Let  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  and  $\Xi = \{\xi_1, \xi_2, \dots, \xi_n\}$  be the set of eigenvalues and eigenvectors of  $-P$  (recall that Matlab requires constraints of the form  $f(x) \leq 0$ ).

$$\begin{aligned}
\frac{\partial \lambda_k}{\partial \hat{p}_{ij}} &= \xi_k^T \left[ \frac{\partial (-P)}{\partial \hat{p}_{ij}} \right] \xi_k \quad (\text{by (D.10),(D.17)}) \\
&= \begin{cases} -\xi_k^T \left[ \hat{P}(\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] \xi_k, & i = j \\ -\xi_k^T \left\{ \hat{P} \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \right. \\ \quad \left. + \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \hat{P} \right\} \xi_k, & i \neq j \end{cases} \quad (\text{D.26})
\end{aligned}$$

$$\frac{\partial \lambda_k}{\partial k_{ij}} = 0 \quad (\text{D.27})$$

**Ellipsoidal State Constraint**

$$\begin{aligned}
\frac{\partial [\Gamma(P^{-1})\Gamma^T]}{\partial \hat{p}_{ij}} &= \Gamma \left[ -P^{-1} \left( \frac{\partial P}{\partial \hat{p}_{ij}} \right) P^{-1} \right] \Gamma^T \quad (\text{by (D.10),(D.15)}) \\
&= \begin{cases} -\Gamma (\hat{P}\hat{P})^{-1} \left[ \hat{P}(\mathbf{EL})_{ij}^{\hat{P}} \right. \\ \quad \left. + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] (\hat{P}\hat{P})^{-1} \Gamma^T, & i = j \\ -\Gamma (\hat{P}\hat{P})^{-1} \left\{ \hat{P} \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \right. \\ \quad \left. + \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \hat{P} \right\} (\hat{P}\hat{P})^{-1} \Gamma^T, & i \neq j \end{cases} \quad (\text{D.28})
\end{aligned}$$

$$\frac{\partial [\Gamma(P^{-1})\Gamma^T]}{\partial k_{ij}} = 0 \quad (\text{D.29})$$

**Semi-Ellipsoidal (Reduced-Dimension) State Constraint** Let  $W = \text{null}\{\Gamma(A - BK)\}$ ,  $\Gamma^R = \Gamma W$ ,  $P^R = W^T P W$ , where  $\Gamma(A - BK) \neq 0$ .

$$\begin{aligned}
\frac{\partial \left\{ \Gamma^R \left[ (P^R)^{-1} \right] (\Gamma^R)^T \right\}}{\partial \hat{p}_{ij}} &= \Gamma^R \left[ - (P^R)^{-1} \left( \frac{\partial P^R}{\partial \hat{p}_{ij}} \right) (P^R)^{-1} \right] (\Gamma^R)^T \quad (\text{by (D.10),(D.15)}) \\
&= -\Gamma^R (P^R)^{-1} W^T \left( \frac{\partial P}{\partial \hat{p}_{ij}} \right) W (P^R)^{-1} (\Gamma^R)^T \quad (\text{by (D.10)}) \\
&= \begin{cases} -\Gamma W (W^T \hat{P} \hat{P} W)^{-1} W^T \left[ \hat{P}(\mathbf{EL})_{ij}^{\hat{P}} \right. \\ \quad \left. + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] (W^T \hat{P} \hat{P} W)^{-1} W^T \Gamma^T, & i = j \\ -\Gamma W (W^T \hat{P} \hat{P} W)^{-1} W^T \left\{ \hat{P} \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \right. \\ \quad \left. + \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \hat{P} \right\} (W^T \hat{P} \hat{P} W)^{-1} W^T \Gamma^T, & i \neq j \end{cases} \quad (\text{D.30})
\end{aligned}$$

Before deriving the partial derivative with respect to the state feedback gain parameters, recall that the derivative of  $W$  is computed numerically, as in (D.31), where  $\epsilon$  is a small constant. (Care must be taken to ensure that the ordering of the eigenvalues and eigenvectors is consistent when computing the new set of null vectors.)

$$\frac{\partial W}{\partial k_{ij}} \approx \frac{W(K + \epsilon(\mathbf{E}\mathbf{L})_{ij}^K) - W(K)}{\epsilon} \quad (\text{D.31})$$

To simplify the derivation to follow, the derivatives of  $\Gamma^R$  and  $P^R$  with respect to  $k_{ij}$  are first presented in (D.32), (D.33), respectively.

$$\frac{\partial \Gamma^R}{k_{ij}} = \Gamma \frac{\partial W}{\partial k_{ij}} \quad (\text{by (D.9)}) \quad (\text{D.32})$$

$$\frac{\partial P^R}{k_{ij}} = \left( \frac{\partial W}{\partial k_{ij}} \right)^T P W + W^T P \left( \frac{\partial W}{\partial k_{ij}} \right) \quad (\text{by (D.11)}) \quad (\text{D.33})$$

The derivative of the reduced-dimension state constraint function with respect to  $k_{ij}$  is given in (D.34).

$$\begin{aligned} \frac{\partial \left\{ \Gamma^R \left[ (P^R)^{-1} \right] (\Gamma^R)^T \right\}}{\partial k_{ij}} &= \left( \frac{\partial \Gamma^R}{\partial k_{ij}} \right) (P^R)^{-1} (\Gamma^R)^T \\ &\quad + \Gamma^R \left[ \frac{\partial (P^R)^{-1}}{\partial k_{ij}} \right] (\Gamma^R)^T \quad (\text{by (D.8)}) \\ &\quad + \Gamma^R (P^R)^{-1} \left( \frac{\partial \Gamma^R}{\partial k_{ij}} \right)^T \\ &= \Gamma \left( \frac{\partial W}{\partial k_{ij}} \right) (P^R)^{-1} (\Gamma^R)^T \\ &\quad + \Gamma^R \left[ - (P^R)^{-1} \left( \frac{\partial P^R}{\partial k_{ij}} \right) (P^R)^{-1} \right] (\Gamma^R)^T \quad (\text{by (D.15)}) \\ &\quad + \Gamma^R (P^R)^{-1} \left( \frac{\partial W^T}{\partial k_{ij}} \right) \Gamma^T \\ &= \Gamma \left( \frac{\partial W}{\partial k_{ij}} \right) (W^T \hat{P} \hat{P} W)^{-1} W^T \Gamma^T \\ &\quad - \Gamma W \left\{ (W^T \hat{P} \hat{P} W)^{-1} \left[ \left( \frac{\partial W}{\partial k_{ij}} \right)^T \hat{P} \hat{P} W + \right. \right. \\ &\quad \left. \left. W^T \hat{P} \hat{P} \left( \frac{\partial W}{\partial k_{ij}} \right) \right] (W^T \hat{P} \hat{P} W)^{-1} \right\} W^T \Gamma^T \\ &\quad + W \Gamma (W^T \hat{P} \hat{P} W)^{-1} \left( \frac{\partial W}{\partial k_{ij}} \right)^T \Gamma^T \quad (\text{D.34}) \end{aligned}$$

### Control Constraint

$$\frac{\partial \left[ (K P^{-1} K^T) / \bar{u}^2 \right]}{\partial \hat{p}_{ij}} = (K / \bar{u}) \left[ -P^{-1} \left( \frac{\partial P}{\partial \hat{p}_{ij}} \right) P^{-1} \right] (K / \bar{u})^T \quad (\text{by (D.10), (D.15)})$$

$$= \begin{cases} -(K/\bar{u}) (\hat{P}\hat{P})^{-1} \left[ \hat{P} (\mathbf{EL})_{ij}^{\hat{P}} \right. \\ \quad \left. + (\mathbf{EL})_{ij}^{\hat{P}} \hat{P} \right] (\hat{P}\hat{P})^{-1} (K/\bar{u})^T, & i = j \\ -(K/\bar{u}) (\hat{P}\hat{P})^{-1} \left\{ \hat{P} \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \right. \\ \quad \left. + \left[ (\mathbf{EL})_{ij}^{\hat{P}} + (\mathbf{EL})_{ji}^{\hat{P}} \right] \hat{P} \right\} (\hat{P}\hat{P})^{-1} (K/\bar{u})^T, & i \neq j \end{cases} \quad (\text{D.35})$$

$$\begin{aligned} \frac{\partial [(KP^{-1}K^T)/\bar{u}^2]}{\partial k_{ij}} &= \left(\frac{1}{\bar{u}}\right)^2 \left[ \left(\frac{\partial K}{\partial k_{ij}}\right) P^{-1}K^T + KP^{-1} \left(\frac{\partial K^T}{\partial k_{ij}}\right) \right] \quad (\text{by (D.11)}) \\ &= \left(\frac{1}{\bar{u}}\right)^2 \left[ (\mathbf{EL})_{ij}^K (\hat{P}\hat{P})^{-1} K^T \right. \\ &\quad \left. + K (\hat{P}\hat{P})^{-1} (\mathbf{EL})_{ji}^{K^T} \right] \quad (\text{by (D.6),(D.7)}) \end{aligned} \quad (\text{D.36})$$

## Appendix E

# Overlapping Ellipsoid Constraint Derivation for Variable Structure Control

In Chapter 5, a sufficient condition for controlled invariance of a semi-ellipsoidal set is derived using state feedback control. Another option is to use a separate control law for each surface. In this appendix, the control is assumed to be constant (i.e., either  $\bar{u}$  or  $\underline{u}$ ) on each state constraint surface forming a portion of the semi-ellipsoidal set boundary (this implies a bias in the equation defining  $\partial\dot{G}_i$ , as in Figure E.1). In a sense, this defines the limiting case by identifying the “maximum” region on each  $\partial\dot{G}_i$  for which  $\frac{d}{dt}(\Gamma_i x) \leq 0$ . However, invariance of the semi-ellipsoidal set is still an unresolved issue with this form of control. The inequality derivation is similar to that of Chapter 5, but involves a modified definition of the derivative of the state constraint function.

The development of  $\partial\dot{G}_i$  proceeds as follows.

$$\begin{aligned}\Gamma_i \dot{x} &\leq 0 \\ \Gamma_i A x + \Gamma_i B u^* &\leq 0 \\ &\downarrow \\ \partial\dot{G}_i &= \{W_i v + W_i^0 | v \in R^{(n-1)}\} \end{aligned} \tag{E.1}$$

where

$$W_i = (\Gamma_i A)^\perp$$

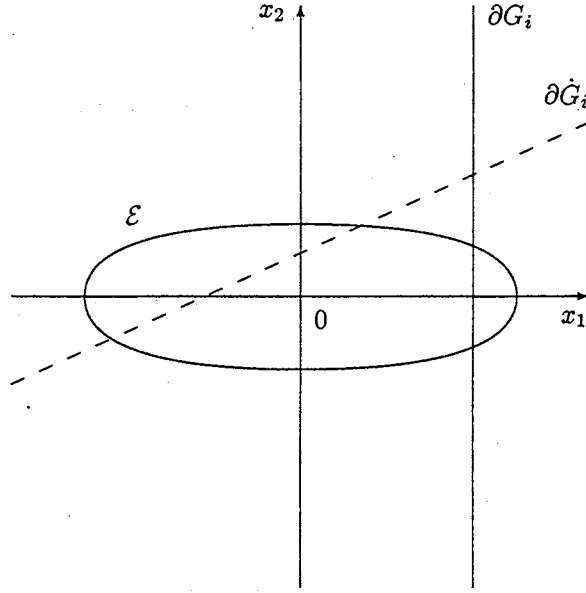


Figure E.1: Illustration of Biased Constraint Derivative Boundary.

$$\begin{aligned}
 W_i^0 &= -(\Gamma_i A)^T [(\Gamma_i A)(\Gamma_i A)^T]^{-1} (\Gamma_i B u^*) \\
 (\Gamma_i A)^\perp &= \text{null} \{ \Gamma_i A \} \\
 u^* &= \begin{cases} \underline{u}, \Gamma_i B > 0 \ (\underline{u} < 0) \\ \bar{u}, \Gamma_i B < 0 \ (\bar{u} > 0) \end{cases}
 \end{aligned}$$

(Note that  $W_i$  and  $W_i^0$  are constant in  $P$ . Also, note that it is assumed  $\Gamma_i A x \neq 0$ . The trivial case,  $\Gamma_i A x = 0, \Gamma_i B \neq 0$ , implies that all points on  $\partial G$  satisfy (E.1) for  $u = u^*$ , so that no constraint on the region of overlap is needed. The case  $\Gamma_i A x = 0, \Gamma_i B = 0$  implies that  $\Gamma_i$  is an uncontrollable mode of the system, contradicting controllability assumptions.)

In reduced dimensions, the set of points satisfying both  $\mathcal{E}$  and  $\partial \dot{G}$  is given in (E.2).

$$\mathcal{E}^R = \left\{ v \mid (W_i v + W_i^0)^T P (W_i v + W_i^0) \leq 1 \right\} \quad (\text{E.2})$$

To use the previous results, this relation must be transformed to one of the form

$$\mathcal{E}^R = \{ z \mid z^T P^R z \leq 1 \} \quad (\text{E.3})$$

using some linear state transformation, (E.4).

$$v = C_1 z + C_0 \quad (C_1 \text{ invertible}) \quad (\text{E.4})$$

## E.1 Derivation of Linear Transformation

For simplicity, define the following constant matrices,

$$\begin{aligned} M_2 &= W_i^T P W_i \text{ (} M_2 \text{ positive definite, symmetric)} \\ M_1 &= (W_i^0)^T P W_i \\ M_0 &= (W_i^0)^T P W_i^0 \end{aligned}$$

such that

$$(W_i v + W_i^0)^T P (W_i v + W_i^0) = v^T M_2 v + M_1 v + v^T M_1^T + M_0 \quad (\text{E.5})$$

Substituting  $z$  for  $v$  in (E.2) yields (E.6).

$$\begin{aligned} z^T C_1^T M_2 C_1 z + (C_0^T M_2 C_1 + M_1 C_1) z \\ + z^T (C_0^T M_2 C_1 + M_1 C_1)^T + (C_0^T M_2 C_0 + M_1 C_0 + C_0^T M_1^T + M_0) \leq 1 \end{aligned} \quad (\text{E.6})$$

To obtain the desired form, (E.3),  $C_0$  and  $C_1$  must be chosen such that the coefficient of  $z$  (and  $z^T$ ) is zero.

$$\begin{aligned} C_0^T M_2 C_1 + M_1 C_1 &= 0 \\ C_0^T M_2 + M_1 &= 0 \\ &\Downarrow \\ C_0 &= -M_2^{-1} M_1^T \end{aligned} \quad (\text{E.7})$$

This choice for  $C_0$  reduces (E.6) to (E.8).

$$z^T C_1^T M_2 C_1 z \leq 1 - M_0 + M_1 M_2^{-1} M_1^T \quad (\text{E.8})$$

Any number of basis sets, which define the structure of  $C_1$ , could be chosen for this transformation.

For simplicity, choose the scaled identity matrix, (E.9).

$$C_1 = I / \sqrt{1 - M_0 + M_1 M_2^{-1} M_1^T} \quad (\text{E.9})$$

These choices for  $C_1$  and  $C_0$  yield the desired form, (E.3), where  $P_i^R$  is defined as in (E.10).

$$P_i^R = (W_i^T P W_i) / f_i(P) \quad (\text{E.10})$$

and where  $f_i$  is a scalar quantity defined as follows:

$$f_i(P) = 1 - (W_i^0)^T P W_i^0 + (W_i^0)^T P W_i (W_i^T P W_i)^{-1} W_i^T P W_i^0 \quad (\text{E.11})$$



Redefining  $\partial\dot{G}_i$  in terms of  $z$  gives representation shown in (E.12).

$$\partial\dot{G}_i = \left\{ \left[ W_i / f(P)^{1/2} \right] z + \left[ I - W_i (W_i^T P W_i)^{-1} W_i^T P \right] W_i^0 \mid z \in R^{(n-1)} \right\} \quad (\text{E.12})$$

The set of points satisfying both (E.12) and (5.1) is found by substituting into the state constraint:

$$\begin{aligned} \Gamma_i x &\leq 1 \\ \Downarrow \\ \left[ \Gamma_i W_i / f_i(P)^{1/2} \right] z &\leq 1 - \Gamma_i \left[ I - W_i (W_i^T P W_i)^{-1} W_i^T P \right] W_i^0 \end{aligned}$$

This relation can be manipulated into the desired form,

$$\partial G_i^R = \{ z \mid \Gamma_i^R z \leq 1 \}$$

via the following definition:

$$\Gamma_i^R = \Gamma_i W_i / \left[ f_i(P)^{1/2} g_i(P) \right] \quad (\text{E.13})$$

where  $g_i$  is a scalar quantity defined as

$$g_i(P) = 1 - \Gamma_i W_i^0 + \Gamma_i W_i (W_i^T P W_i)^{-1} W_i^T P W_i^0 \quad (\text{E.14})$$

Finally, using the previous result, (3.11), the inequality constraint for overlap of the ellipsoid and state constraint, (E.15), is obtained.

$$\begin{aligned} \Gamma_i^R (P_i^R)^{-1} (\Gamma_i^R)^T &\leq 1 \\ \Downarrow \\ \left( \Gamma_i W_i / \left[ f(P)^{1/2} g(P) \right] \right) \left[ (W_i^T P W_i) / f(P) \right]^{-1} \left( \Gamma_i W_i / \left[ f(P)^{1/2} g(P) \right] \right)^T &\leq 1 \\ \Downarrow \\ \left[ (\Gamma_i W_i) (W_i^T P W_i)^{-1} (\Gamma_i W_i)^T \right] / [g_i(P)]^2 &\leq 1 \\ \Downarrow \\ \left[ (\Gamma_i W_i) (W_i^T P W_i)^{-1} (\Gamma_i W_i)^T \right] / \left[ 1 - \Gamma_i W_i^0 + \Gamma_i W_i (W_i^T P W_i)^{-1} W_i^T P W_i^0 \right]^2 &\leq 1 \quad (\text{E.15}) \end{aligned}$$

## E.2 Analytic Gradient of Constraint

As a preliminary step to computing the gradient of the modified constraint, (E.15), the derivative of the function  $g_i(P)$  is first presented. To avoid confusion with the parameter indexing subscripts,

the subscript  $i$  is dropped from the state constraint and related matrices  $(\Gamma, W, W^0)$ .

$$\begin{aligned}
\frac{\partial [g(P)]}{\partial \hat{p}_{ij}} &= \Gamma W \frac{\partial [(W^T P W)^{-1}]}{\partial \hat{p}_{ij}} W^T P W^0 \\
&\quad + \Gamma W (W^T P W)^{-1} W^T \frac{\partial P}{\partial \hat{p}_{ij}} W^0 \quad (\text{by (D.8),(D.9)}) \\
&= -\Gamma W (W^T P W)^{-1} \frac{\partial (W^T P W)}{\partial \hat{p}_{ij}} (W^T P W)^{-1} W^T P W^0 \\
&\quad + \Gamma W (W^T P W)^{-1} W^T \frac{\partial P}{\partial \hat{p}_{ij}} W^0 \quad (\text{by (D.15)}) \\
&= -\Gamma W (W^T P W)^{-1} W^T \frac{\partial P}{\partial \hat{p}_{ij}} W (W^T P W)^{-1} W^T P W^0 \\
&\quad + \Gamma W (W^T P W)^{-1} W^T \frac{\partial P}{\partial \hat{p}_{ij}} W^0 \quad (\text{by (D.10)}) \\
&= \Gamma W (W^T P W)^{-1} W^T \frac{\partial P}{\partial \hat{p}_{ij}} \left[ I - W (W^T P W)^{-1} W^T P \right] W^0 \\
&= \begin{cases} \Gamma W (W^T P W)^{-1} W^T \left[ \hat{P} (\mathbf{E}\mathbf{L})_{ij}^{\hat{P}} + (\mathbf{E}\mathbf{L})_{ij}^{\hat{P}} \hat{P} \right] \\ \quad \cdot \left[ I - W (W^T P W)^{-1} W^T P \right] W^0 & i = j \\ \Gamma W (W^T P W)^{-1} W^T \left\{ \hat{P} \left[ (\mathbf{E}\mathbf{L})_{ij}^{\hat{P}} + (\mathbf{E}\mathbf{L})_{ji}^{\hat{P}} \right] \right. \\ \quad \left. + \left[ (\mathbf{E}\mathbf{L})_{ij}^{\hat{P}} + (\mathbf{E}\mathbf{L})_{ji}^{\hat{P}} \right] \hat{P} \right\} \\ \quad \cdot \left[ I - W (W^T P W)^{-1} W^T P \right] W^0 & i \neq j \end{cases} \quad (\text{by (D.19)}) \quad (\text{E.16})
\end{aligned}$$

The derivative of the modified state constraint is now presented, where substitution of (E.14), (D.19), and (E.16) should be made into (E.17) where appropriate.

$$\begin{aligned}
&\frac{\partial}{\partial \hat{p}_{ij}} \left\{ \left[ (\Gamma W) (W^T P W)^{-1} (\Gamma W)^T \right] / [g(P)]^2 \right\} \\
&= [g(P)]^{-2} \frac{\partial \left[ (\Gamma W) (W^T P W)^{-1} (\Gamma W)^T \right]}{\partial \hat{p}_{ij}} \\
&\quad + \left[ (\Gamma W) (W^T P W)^{-1} (\Gamma W)^T \right] \frac{\partial \{ [g(P)]^{-2} \}}{\partial \hat{p}_{ij}} \quad (\text{by (D.8)}) \\
&= [g(P)]^{-2} (\Gamma W) \frac{\partial [(W^T P W)^{-1}]}{\partial \hat{p}_{ij}} (\Gamma W)^T \\
&\quad - 2 [g(P)]^{-3} \left[ (\Gamma W) (W^T P W)^{-1} (\Gamma W)^T \right] \frac{\partial [g(P)]}{\partial \hat{p}_{ij}} \quad (\text{by (D.10)}) \\
&= -[g(P)]^{-2} (\Gamma W) \left[ (W^T P W)^{-1} \frac{\partial (W^T P W)}{\partial \hat{p}_{ij}} (W^T P W)^{-1} \right] (\Gamma W)^T \\
&\quad - 2 [g(P)]^{-3} \left[ (\Gamma W) (W^T P W)^{-1} (\Gamma W)^T \right] \frac{\partial [g(P)]}{\partial \hat{p}_{ij}} \quad (\text{by (D.15)}) \\
&= -[g(P)]^{-2} (\Gamma W) \left[ (W^T P W)^{-1} W^T \frac{\partial P}{\partial \hat{p}_{ij}} W (W^T P W)^{-1} \right] (\Gamma W)^T \\
&\quad - 2 [g(P)]^{-3} \left[ (\Gamma W) (W^T P W)^{-1} (\Gamma W)^T \right] \frac{\partial [g(P)]}{\partial \hat{p}_{ij}} \quad (\text{by (D.10)}) \quad (\text{E.17})
\end{aligned}$$

$$\frac{\partial}{\partial k_{ij}} \left\{ \left[ (\Gamma W) (W^T P W)^{-1} (\Gamma W)^T \right] / [g(P)]^2 \right\} = 0 \quad (\text{E.18})$$

## Appendix F

# Matlab Code for Ellipsoidal Sets

### F.1 Function Files for Computing Recoverable Ellipsoidal Set

#### F.1.1 Optimization Routine

```
function [P,Kv,U,X]=recover(A,B,GAMMA,UMAX,Q0,PO,K0v);
%
% RECOVER finds the largest recoverable ellipsoid for a linear system with
% constrained states and inputs.
%
% Parameter definitions:
%
% [A,B]: state-space description of linear system
% GAMMA: state constraints of the form GAMMA*x<=1
% UMAX: maximum allowable control (assumed symmetric)
% Q0: specifies decay rate of the Lyapunov function
% PO: initial guess for the ellipsoid matrix (x'Px<=1)
% K0v: initial guess for state feedback (must stabilize [A,B])
%
% P: the optimal ellipsoid
% Kv: the corresponding state feedback gain matrix
% U: the maximum control effort on the boundary of P
% X: the final parameter search vector (elements of P, Kv)
%
% Usage:
%
% [P,Kv,U,X]=recover(A,B,GAMMA,UMAX,Q0,PO,K0v) finds the largest
% recoverable ellipsoid, P, and corresponding state-feedback
% matrix, Kv, for the system [A,B] and the constraints, GAMMA,
% UMAX. Definitions of Q0, PO, and K0v are optional. If not
% provided, Q0 is assumed to be zero and PO is initialized using
% an LQR-based approach. If K0v is not specified, or if the given
% K0v does not stabilize [A,B], it is chosen (arbitrarily) using
% the LQR technique.
%
% Notice:
%
% This algorithm is based on the dissertation "Ellipsoidal and
% Semi-Ellipsoidal Controlled Invariant Sets for Constrained
```

```

%      Linear Systems" by Brian O'Dell, Oklahoma State University, 1999.
%

% Start counter for run-time
tic

% DEFINE SIMULATION CONSTANTS
fig_handle=1;    % Set the figure handle
ON=1;
OFF=0;
PARTIAL=0.5;
beta=0.95;      % Decrease factor for size of P; MUST BE LESS THAN 1.0

[n,m]=size(B);  % 'n' is number of states, 'm' is number of inputs
pe=(n^2-n)/2+n; % Defines number of ellipse parameters
pg=n*m;         % Defines number of state feedback gain parameters
p=pe+pg;        % Defines the total number of parameters.

PET=1;          % 'Percent Error Tolerance' for terminating searches

plotting=PARTIAL; % Turn plotting 'on/off'
plot_pts=150;    % Number of points to use in plotting

% DEFINE SEARCH PARAMETERS (for search OPTIONS, type 'help foptions')
max_passes=20;
min_iters=30*p;
max_iters=70*p; % Maximum number of search iterations per cycle (nom 70*p)
weight=5e1;     % Weighting coefficient for constraint vector (nom 50)
pd_weight=1e3;  % Additional (multiplicative) weight for pos. def. const.
options(1)=1;   % Turns off display & suppresses warnings
options(6)=1;
options(14)=max_iters; % Set the maximum number of iterations per pass
options(16)=1e-10;
options(17)=1e-8;

% INITIALIZE SEARCH DATA VECTORS
Fstart=[];      % Vector of cost function values before each pass
Gstart=[];      % Matrix of constraint function values before each pass
Fstop=[];       % Vector of cost function values after each pass
Gstop=[];       % Matrix of constraint function values after each pass
search_log=[];  % Matrix of parameter values after each pass

% CHECK VALIDITY OF INPUT DEFINITIONS
N=nargin;
if N<4
    error('Not enough input arguments.')
```

```

elseif N==6
    KOv=[];
end

% CHECK SIZE OF A,B
if size(A,1)~=size(A,2)
    error('A is non-square.')
elseif size(A,1)~=n
    error('A and B must have same number of rows.')
end

% CHECK CONSTRAINT SPACE SIZE
if rank(GAMMA)<n
    error('State constraints do not form closed set.')
end

% CHECK UMAX
if min(UMAX)<=0
    error('Control constraints must be positive.')
end

% CHECK Q0
if isempty(Q0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing Q0 matrix:')
    disp(' ')
    Q0=zeros(size(A))
elseif max(max(abs(Q0-Q0')))>0 % Check for symmetry
    error('Q0 is not symmetric.')
elseif min(eig(Q0))<0 % Check for positive definiteness
    error('Q0 is not positive definite.')
end

% USE LQR SOLUTION FOR ANY NECESSARY INITIALIZATIONS
poles=[1:n]; % Define (arbitrary) positive pole locations
k0=place(A,B,poles); % Compute state-feedback gain
R=eye(m); % Arbitrary pos. def. weighting matrix for LQR
[temp,p20]=lqr(A-B*k0,B,Q0,R); % Compute stabilizing controller and ellipse
dk=inv(R)*B'*p20; % Define stabilizing controller

% CHECK KOv
if isempty(KOv) % Initialize with LQR if KOv not given
    % Display initialization message
    disp(' ')
    disp('Initializing KOv matrix:')
    disp(' ')
    KOv=k0+dk/2 % Define marginally stabilizing controller
end

```

```

    Kv=dk/4                % Define gain such that (K0v+Kv) is stabilizing
                          % (in positive time)
else
    if (size(K0v,1)~=m)|(size(K0v,2)~=n) % Check size of K0v
        error('K0v must be same size as B transpose.')
    end
    if max(real(eig(A-B*K0v)))>0 % Check stability of A-B*K0v
        error('K0v does not stabilize A.')
    end
end

% CHECK P0
if isempty(P0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing P0 matrix:')
    disp(' ')
    con=[(K0v+Kv)/UMAX;GAMMA]; % Define matrix of all bounds
    p20=p20*max(diag((con*inv(p20)*con'))); % Normalize ellipsoid to touch one
                                          % of the bounds.
    P0=real(sqrtm(p20)) % Compute P0 from p20
    P20=P0'*P0
elseif max(max(abs(P0-P0')))>0 % Check for symmetry
    error('P0 is not symmetric.')
elseif min(eig(P0))<0 % Check for positive definiteness
    error('P0 is not positive definite.')
end

% ASSIGN P0,K0v TO ELEMENTS OF SEARCH SPACE VECTOR
X=pk2x(P0,Kv,n,m); % Parametrize search with P0, the square root of the
                  % ellipsoid matrix, to minimize search errors with
                  % positive definiteness of ellipse. (P=P0*P0)

if plotting==ON
    % Plot initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    ellipse(P20,[],plot_pts);grid on;axis square;
    title('INITIAL CONDITION')
    drawnow;

    % Plot trajectories of initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    recover_p(X,A,B,K0v,plot_pts);
    title('TRAJECTORIES OF INITIAL CONDITION')
    drawnow;
end

save recover.mat X

pvol=det(inv(P20));

```

```

save recover_best.mat X pvol

% =====

% OPTIMIZE SOLUTION

options(14)=max_iters; % Set the maximum number of iterations to max_iters
pass=1;                % Reset the counter
control_const=0N;     % Turn the control constraint 'on'

change=100;
options(10)=max_iters;
while (pass<=max_passes)&((abs(change)>PET)|(options(10)>(0.8*options(14))))
    % Loop until change in ellipsoid size < 2%

    disp(' ')
    disp(sprintf(' ===== OPTIMIZING: CHANGE = %6.2f %% ===== ',...
        change))
    disp(' ')

    % Increase search iterations if necessary
    if round(pass/5)*5==pass
        options(14)=min([min_iters,round(options(14)/1.1)]);
    end

    % Log start-of-pass search constraint vector and cost function
    [fstart,g]=recover_fg(X,A,B,K0v,GAMMA,UMAX,Q0,control_const,...
        weight,pd_weight);
    Gstart=[Gstart g];
    Fstart=[Fstart fstart];

    % Scale ellipse dow by factor of beta to start the next pass off the
    % constraints
    X(1:pe,1)=X(1:pe,1)/sqrt(beta);

    % Perform search.
    options(9)=OFF;
    [X,options]=constr('recover_fg',X,options,[],[],'recover_dfg',...
        A,B,K0v,GAMMA,UMAX,Q0,control_const,weight,pd_weight);

    % Log end-of-pass search constraint vector and cost function
    [fstop,g,U,constraints]=recover_fg(X,A,B,K0v,GAMMA,UMAX,Q0,...
        control_const,weight,pd_weight);
    Gstop=[Gstop g];
    Fstop=[Fstop fstop];

    % Compute change in cost function
    change=100*(exp(fstart)-exp(fstop))/exp(fstart);

    % Log parameter space,control value, and number of iterations
    search_log=[search_log X];

    % Parameter Assignments

```



```

P=x2pk(X,n,m);
P2=P'*P;

% Plot ellipsoid in state space
if plotting==0N
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    ellipse(P2,[],plot_pts);grid on;axis square;
    title(sprintf('OPTIMIZING: CHANGE = %1.4g',change));
    drawnow
end

save recover.mat X U

if det(inv(P2))>pvol
    pvol=det(inv(P2));
    save recover_best.mat X P2 Kv pvol
end

pass=pass+1;

end

% =====

% DISPLAY TERMINATION CRITERIA

disp(' ')
disp(sprintf('Search terminated on change in cost function of %0.4g %%.',...
    change))
disp(' ')

% =====

% CLOSING TASKS

% Plot trajectories
if plotting>=PARTIAL
    figure(fig_handle);clf;drawnow;
    recover_p(X,A,B,K0v,plot_pts);
    title('RECOVERABLE ELLIPSOID')
end

% Compute composite gain.
Kv=Kv+K0v;

% Define output ellipsoid matrix
P=P2;

% Compute maximum gain on ellipse, U

```

```
[f,g,U,constraints]=recover_fg(X,A,B,KOv,GAMMA,UMAX,QO,...  
    control_const,weight,pd_weight);
```

```
% Clean up hard drive  
delete recover.mat  
delete recover_best.mat
```

```
% Terminate counter and display elapsed time  
toc
```

```
% ===== END OF FILE: RECOVER.M =====
```

## F.1.2 Cost Function and Constraints

```
function [f,g,Uv,constraints]=recover_fg(X,A,B,Kov,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight);
% RECOVER_FG is the cost function and constraint routine for use with RECOVER.

% parametrizes in terms of sqrt(P) and Kv
% modifies definition of (at end) and pos. def. inequalities

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
[p]=max(size(X));
[c]=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(det(P2));

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Invariance constraint
Acl=A-B*(Kv+Kov);
temp=P2*Acl+Acl'*P2+Q; % Write as negative definite form
[v,d]=eig(temp);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

% Positive definiteness constraints.
[v,d]=eig(-P2*pd_weight); % Write as negative definite form
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Positive Definiteness');
end

% State constraints
for i=1:c;
    g=[g; GAMMA(i,:)*inv(P2)*GAMMA(i,:)'-1];
    constraints=char(constraints,['State Constraint ' num2str(i)]);
end;
```

```
% Control constraints
if control_const==1
    Uv=sqrt((Kv+K0v)*inv(P2)*(Kv+K0v)');
    g=[g; (Uv*Uv)/(UMAX*UMAX)-1];    % Normalize to 1
    constraints=char(constraints,'Control');
end

% ===== END OF FILE: RECOVER_FG.M =====
```

### F.1.3 Derivatives of Cost Function and Constraints

```
function [df_dX,dg_dX]=recover_dfg(X,A,B,K0v,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight);
% RECOVER_DFG is the derivative cost function and constraint routine
%   for use with RECOVER.

% parametrizes in terms of sqrt(P) and Kv
% modifies definition of (at end) and pos. def. inequalities

% Empirical weights on constraints (experimented w/weighting pos.def. 1e3)

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
p=max(size(X));
c=size(GAMMA,1);
pg=n*m; % number of state feedback gain parameters
pe=p-pg; % number of ellipsoid parameters

% PARAMETER ASSIGNMENTS
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(det(P2));

Pi=inv(P);

df_dP=[];
xindex=0;
for i=1:n
    for j=i:n
        if i==j
            df_dP=[df_dP;2*Pi(i,j)];
        else
            df_dP=[df_dP;4*Pi(i,j)];
        end
    end
end
df_dKv=zeros(size(Kv))';
df_dX=[df_dP;df_dKv];

% =====

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear constraint array
dg_dX=[]; % clear constraint derivative array

% Invariance constraint
Acl=A-B*(Kv+K0v);
```

```

IC=P2*Acl+Acl'*P2+Q;
[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
Vi=v(:,sort_index);
g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*Acl+Acl'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKv=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Kv));
        EL(i,j)=1;
        dAcl_dKv_ij=-B*EL;
        dIC_dKv_ij=P2*(dAcl_dKv_ij)+(dAcl_dKv_ij)'*P2;
        dDi_dKv=[dDi_dKv; diag(Vi'*(dIC_dKv_ij)*Vi)'];
    end
end
dDi_dX=[dDi_dP; dDi_dKv];
dg_dX=[dg_dX dDi_dX];

% Positive definiteness of P2 constraint.
PDC=-P2*pd_weight;
[v,d]=eig(PDC);
[Dpd,sort_index]=esort(diag(d));
Vpd=v(:,sort_index);
g=[g;Dpd];      % Write as 'less than' constraint

dDpd_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dPDC_dP_ij=-1*(EL'*P+P*EL)*pd_weight;
        dDpd_dP=[dDpd_dP; diag(Vpd'*(dPDC_dP_ij)*Vpd)'];
    end
end
dDpd_dKv=zeros(pg,n); % 'n' eigenvalues

dDpd_dX=[dDpd_dP;dDpd_dKv];
dg_dX=[dg_dX dDpd_dX];

% State constraints
for k=1:c;

```

```

SC=GAMMA(k,:)*inv(P2)*GAMMA(k,:)'-1;
g=[g;SC];

dSC_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P));
        EL(i,j)=1;EL(j,i)=1;
        dSC_dP_ij=GAMMA(k,:)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*GAMMA(k,:)' ;
        dSC_dP=[dSC_dP; dSC_dP_ij];
    end
end
dSC_dKv=zeros(pg,1);
dSC_dX=[dSC_dP;dSC_dKv];
dg_dX=[dg_dX dSC_dX];
end;

% Control constraints
if control_const==1
CC=(1/UMAX^2)*((Kv+K0v)*inv(P2)*(Kv+K0v)')'-1;
g=[g;CC];

dCC_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P));
        EL(i,j)=1;EL(j,i)=1;
        dCC_dP_ij=(1/UMAX^2)*((Kv+K0v)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))...
            *(Kv+K0v)');
        dCC_dP=[dCC_dP; dCC_dP_ij];
    end
end
dCC_dKv=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Kv));
        EL(i,j)=1;
        dCC_dKv_ij=(1/UMAX^2)*((EL)*inv(P2)*(Kv+K0v)'+(Kv+K0v)*inv(P2)*(EL)');
        dCC_dKv=[dCC_dKv; dCC_dKv_ij];
    end
end
dCC_dX=[dCC_dP; dCC_dKv];
dg_dX=[dg_dX dCC_dX];
end

% ===== END OF FILE: RECOVER_DFG.M =====

```

### F.1.4 Plotting Routine

```
function [U,P,Kv]=recover_p(X,A,B,K0v,plot_pts);
% RECOVER_P is the trajectory plotting routine for use with RECOVER

% Plotting options
N=5; % plot trajectory from every Nth data point on ellipsoid

if nargin<4
    error('Not enough input arguments.')
end

% Message to screen
disp(' ')
disp('+++++')
disp('Plotting recovering trajectories')
disp('+++++')
disp(' ')

% Parameter Assignments (extract P,K's from X)
[n,m]=size(B);
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

Kv=Kv+K0v; % Construct composite gain

if n==2 % 2-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2]=ellipse(P2,[],plot_pts);

    % Compute corresponding control effort
    ue=Kv*[x1;x2];

    % Compute maximum control effort on boundary
    U=max(ue);

    % Plot boundary vs. control
    plot(x1,x2,'k');grid on;axis square;hold on

    % Compute reaching and recovering state-space descriptions
    syse=ss(A-B*Kv,B,eye(n),zeros(n,m));

    % Plot trajectories for every Nth point on the ellipsoid boundary
    for i=1:N:max(size(x1));
        [y,t,x]=initial(syse,[x1(i),x2(i)],[0:.01:10]);
        plot(x(:,1),x(:,2),'r')
    end

    xlabel('x1'),ylabel('x2'),zlabel('u')
elseif n==3 % 3-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2,x3]=ellipse(P2,[],plot_pts);
```



```

% Compute corresponding control effort
for i=1:size(x1,1)
    for j=1:size(x1,2)
        ue(i,j)=Kv*[x1(i,j);x2(i,j);x3(i,j)];
    end
end

% Compute maximum control effort on boundary
U=max(max(abs(ue)));

% Plot boundary vs. control
surf(x1,x2,x3,ue/U);grid on;axis square;hold on

% Compute reaching and recovering state-space descriptions
syse=ss(A-B*Kv,B,eye(n),zeros(n,m));

% Plot trajectories for every Nth point on the ellipsoid boundary
hold on
for i=1:N^2:size(x1,1);
    for j=1:N^2:size(x1,2)
        [y,t,x]=initial(syse,[x1(i,j),x2(i,j),x3(i,j)], [0:.01:10]);
        plot3(x(:,1),x(:,2),x(:,3),'r')
    end
end
drawnow
grid on
colormap('copper')
lighting phong
light('Position',[1 -1 5])
h=findobj('Type','surface');
set(h,'FaceLighting','phong',...
    'FaceColor','interp',...
    'EdgeColor',[.4 .4 .4],...
    'BackFaceLighting','reverselit',...
    'AmbientStrength',1,...
    'DiffuseStrength',1);
shading interp
xlabel('x1'),ylabel('x2'),zlabel('x3')

end

% ===== END OF FILE: RECOVER_P.M =====

```

## F.2 Function Files for Computing Reachable Ellipsoidal Set

### F.2.1 Optimization Routine

```
function [P,Ke,U,X]=reach(A,B,GAMMA,UMAX,Q0,P0,K0e);
%
% REACH finds the largest reachable ellipsoid for a linear system with
% constrained states and input.
%
% Parameter definitions:
%
% [A,B]: state-space description of linear system
% GAMMA: state constraints of the form GAMMA*x<=1
% UMAX: maximum allowable control (assumed symmetric)
% Q0: specifies decay rate of the Lyapunov function
% P0: initial guess for the ellipsoid matrix (x'Px<=1)
% K0e: initial guess for state feedback (must stabilize [-A,-B])
%
% P: the optimal ellipsoid
% Ke: the corresponding state feedback gain matrix
% U: the maximum control effort on the boundary of P
% X: the final parameter search vector (elements of P, Ke)
%
% Usage:
%
% [P,Ke,U,X]=reach(A,B,GAMMA,UMAX,Q0,P0,K0e) finds the largest
% reachable ellipsoid, P, and corresponding state-feedback
% matrix, Ke, for the system [A,B] and the constraints, GAMMA,
% UMAX. Definitions of Q0, P0, and K0e are optional. If not
% provided, Q0 is assumed to be zero and P0 is initialized using
% an LQR-based approach. If K0e is not specified, or if the given
% K0e does not stabilize [A,B], it is chosen (arbitrarily) using
% the LQR technique.
%
% Notice:
%
% This algorithm is based on the dissertation "Ellipsoidal and
% Semi-Ellipsoidal Controlled Invariant Sets for Constrained
% Linear Systems" by Brian O'Dell, Oklahoma State University, 1999.
%
% Start counter for run-time
tic

% DEFINE SIMULATION CONSTANTS
fig_handle=1; % Set the figure handle
ON=1;
OFF=0;
PARTIAL=0.5;
beta=0.95; % Decrease factor for size of P; MUST BE LESS THAN 1.0

[n,m]=size(B); % 'n' is number of states, 'm' is number of inputs
pe=(n^2-n)/2+n; % Defines number of ellipse parameters
pg=n*m; % Defines number of state feedback gain parameters
```

```

p=pe+pg;          % Defines the total number of parameters.

PET=1;           % 'Percent Error Tolerance' for terminating searches

plotting=PARTIAL; % Turn plotting 'on'/'off'
plot_pts=150;    % Number of points to use in plotting

% DEFINE SEARCH PARAMETERS (for search OPTIONS, type 'help foptions')
max_passes=20;
min_iters=30*p;
max_iters=70*p; % Maximum number of search iterations per cycle (nom 70*p)
weight=5e1;     % Weighting coefficient for constraint vector (nom 50)
pd_weight=1e3; % Additional (multiplicative) weight for pos. def. const.
options(1)=1;   % Turns off display & suppresses warnings
options(6)=1;
options(14)=max_iters; % Set the maximum number of iterations per pass
options(16)=1e-10;
options(17)=1e-8;

% INITIALIZE SEARCH DATA VECTORS
Fstart=[];      % Vector of cost function values before each pass
Gstart=[];      % Matrix of constraint function values before each pass
Fstop=[];       % Vector of cost function values after each pass
Gstop=[];       % Matrix of constraint function values after each pass
search_log=[];  % Matrix of parameter values after each pass

% CHECK VALIDITY OF INPUT DEFINITIONS
N=nargin;
if N<4
    error('Not enough input arguments.')
elseif N==4
    Q0=[];P0=[];K0e=[];
elseif N==5
    P0=[];K0e=[];
elseif N==6
    K0e=[];
end

% CHECK SIZE OF A,B
if size(A,1)~=size(A,2)
    error('A is non-square.')
elseif size(A,1)~=n
    error('A and B must have same number of rows.')
end

% CHECK CONSTRAINT SPACE SIZE
if rank(GAMMA)<n
    error('State constraints do not form closed set.')
end

```

```

% CHECK UMAX
if min(UMAX)<=0
    error('Control constraints must be positive.')
end

% CHECK Q0
if isempty(Q0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing Q0 matrix:')
    disp(' ')
    Q0=zeros(size(A))
elseif max(max(abs(Q0-Q0')))>0 % Check for symmetry
    error('Q0 is not symmetric.')
elseif min(eig(Q0))<0 % Check for positive definiteness
    error('Q0 is not positive definite.')
end

% USE LQR SOLUTION FOR ANY NECESSARY INITIALIZATIONS
poles=[1:n]; % Define (arbitrary) positive pole locations
k0=place(A,B,poles); % Compute state-feedback gain
R=eye(m); % Arbitrary pos. def. weighting matrix for LQR
[temp,p20]=lqr(A-B*k0,B,Q0,R); % Compute stabilizing controller and ellipse
dk=inv(R)*B'*p20; % Define stabilizing controller

% CHECK K0e
if isempty(K0e) % Initialize with LQR if K0e not given
    % Display initialization message
    disp(' ')
    disp('Initializing K0e matrix:')
    disp(' ')
    K0e=k0+dk/2 % Define marginally stabilizing controller
    Ke=-dk/4 % Define gain such that (K0e+Ke) is stabilizing
    % (in positive time)
else
    if (size(K0e,1)~=m)|(size(K0e,2)~=n) % Check size of K0e
        error('K0e must be same size as B transpose.')
    end
    if max(real(eig(A-B*K0e)))>0 % Check stability of A-B*K0e
        error('K0e does not stabilize A.')
    end
end

% CHECK P0
if isempty(P0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing P0 matrix:')
    disp(' ')

```

```

con=[(K0e+Ke)/UMAX;GAMMA];           % Define matrix of all bounds
p20=p20*max(diag((con*inv(p20)*con'))); % Normalize ellipsoid to touch one
                                         % of the bounds.
P0=real(sqrtm(p20))                   % Compute P0 from p20
P20=P0'*P0
elseif max(max(abs(P0-P0')))>0 % Check for symmetry
    error('P0 is not symmetric.')
elseif min(eig(P0))<0          % Check for positive definiteness
    error('P0 is not positive definite.')
end

% ASSIGN P0,K0e TO ELEMENTS OF SEARCH SPACE VECTOR
X=pk2x(P0,Ke,n,m); % Parametrize search with P0, the square root of the
                  % ellipsoid matrix, to minimize search errors with
                  % positive definiteness of ellipse. (P=P0*P0)

if plotting==ON
    % Plot initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    ellipse(P20,[],plot_pts);grid on;axis square;
    title('INITIAL CONDITION')
    drawnow;

    % Plot trajectories of initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    reach_p(X,A,B,K0e,plot_pts);
    title('TRAJECTORIES OF INITIAL CONDITION')
    drawnow;
end

save reach.mat X

pvol=det(inv(P20));

save reach_best.mat X pvol

% =====

% OPTIMIZE SOLUTION

options(14)=max_iters; % Set the maximum number of iterations to max_iters
pass=1;                % Reset the counter
control_const=ON;      % Turn the control constraint 'on'

change=100;
options(10)=max_iters;
while (pass<=max_passes)&((abs(change)>PET)|(options(10)>(0.8*options(14))))
    % Loop until change in ellipsoid size < 2%

    disp(' ')

```

```

disp(sprintf(' ===== OPTIMIZING: CHANGE = %6.2f %% ===== ',...
change))
disp(' ')

% Increase search iterations if necessary
if round(pass/5)*5==pass
    options(14)=min([min_iters,round(options(14)/1.1)]);
end

% Log start-of-pass search constraint vector and cost function
[fstart,g]=reach_fg(X,A,B,KOe,GAMMA,UMAX,Q0,control_const,...
weight,pd_weight);
Gstart=[Gstart g];
Fstart=[Fstart fstart];

% Scale ellipse dow by factor of beta to start the next pass off the
% constraints
X(1:pe,1)=X(1:pe,1)/sqrt(beta);

% Perform search.
options(9)=OFF;
[X,options]=constr('reach_fg',X,options,[],[],'reach_dfg',...
A,B,KOe,GAMMA,UMAX,Q0,control_const,weight,pd_weight);

% Log end-of-pass search constraint vector and cost function
[fstop,g,U,constraints]=reach_fg(X,A,B,KOe,GAMMA,UMAX,Q0,...
control_const,weight,pd_weight);
Gstop=[Gstop g];
Fstop=[Fstop fstop];

% Compute change in cost function
change=100*(exp(fstart)-exp(fstop))/exp(fstart);

% Log parameter space,control value, and number of iterations
search_log=[search_log X];

% Parameter Assignments
P=x2pk(X,n,m);
P2=P'*P;

% Plot ellipsoid in state space
if plotting==ON
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    ellipse(P2,[],plot_pts);grid on;axis square;
    title(sprintf('OPTIMIZING: CHANGE = %1.4g',change));
    drawnow
end

save reach.mat X U

if det(inv(P2))>pvol
    pvol=det(inv(P2));
    save reach_best.mat X P2 Ke pvol

```

```

    end

    pass=pass+1;

end

% =====

% DISPLAY TERMINATION CRITERIA

disp(' ')
disp(sprintf('Search terminated on change in cost function of %0.4g %%.',...
    change))
disp(' ')

% =====

% CLOSING TASKS

% Plot trajectories
if plotting>=PARTIAL
    figure(fig_handle);clf;drawnow;
    reach_p(X,A,B,KOe,plot_pts);
    title('REACHABLE ELLIPSOID')
end

% Compute composite gain.
Ke=Ke+KOe;

% Define output ellipsoid matrix
P=P2;

% Compute maximum gain on ellipse, U
[f,g,U,constraints]=reach_fg(X,A,B,KOe,GAMMA,UMAX,Q0,...
    control_const,weight,pd_weight);

% Clean up hard drive
delete reach.mat
delete reach_best.mat

% Terminate counter and display elapsed time
toc

% ===== END OF FILE: REACH.M =====

```

## F.2.2 Cost Function and Constraints

```
function [f,g,Ue,constraints]=reach_fg(X,A,B,KOe,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight);
% REACH_FG is the cost function and constraint routine for use with REACH.

% parametrizes in terms of sqrt(P) and Ke
% modifies definition of (at end) and pos. def. inequalities

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
[p]=max(size(X));
[c]=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Ke]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(det(P2));

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Invariance constraint
Acl=A-B*(Ke+KOe);
temp=P2*(-Acl)+(-Acl)'+P2+Q; % Write as negative definite form
[v,d]=eig(temp);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

% Positive definiteness constraints.
[v,d]=eig(-P2*pd_weight); % Write as negative definite form
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Positive Definiteness');
end

% State constraints
for i=1:c;
    g=[g; GAMMA(i,:)*inv(P2)*GAMMA(i,:)'-1];
    constraints=char(constraints,['State Constraint ' num2str(i)]);
end;
```



```
% Control constraints
if control_const==1
    Ue=sqrt((Ke+K0e)*inv(P2)*(Ke+K0e)');
    g=[g; (Ue*Ue)/(UMAX*UMAX)-1]; % Normalize to 1
    constraints=char(constraints,'Control');
end
```

```
% ===== END OF FILE: REACH_FG.M =====
```

### F.2.3 Derivatives of Cost Function and Constraints

```

function [df_dX,dg_dX]=reach_dfg(X,A,B,K0e,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight);
% REACH_DFG is the derivative cost function and constraint routine
%   for use with REACH.

% 1) parametrizes in terms of sqrt(P) and Ke
% 2) optional weights on constraints

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
p=max(size(X));
c=size(GAMMA,1);
pg=n*m; % number of state feedback gain parameters
pe=p-pg; % number of ellipsoid parameters

% PARAMETER ASSIGNMENTS
[P,Ke]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(det(P2));

Pi=inv(P);

df_dP=[];
xindex=0;
for i=1:n
    for j=i:n
        if i==j
            df_dP=[df_dP;2*Pi(i,j)];
        else
            df_dP=[df_dP;4*Pi(i,j)];
        end
    end
end
df_dKe=zeros(size(Ke))';
df_dX=[df_dP;df_dKe];

% =====

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear constraint array
dg_dX=[]; % clear constraint derivative array

% Invariance constraint
Acl=A-B*(Ke+K0e);
IC=P2*(-Acl)+(-Acl)'*P2+Q;

```

```

[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
Vi=v(:,sort_index);
g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*(-Acl)+(-Acl)'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKe=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Ke));
        EL(i,j)=1;
        dAcl_dKe_ij=-B*EL;
        dIC_dKe_ij=P2*(-dAcl_dKe_ij)+(-dAcl_dKe_ij)'*P2;
        dDi_dKe=[dDi_dKe; diag(Vi'*(dIC_dKe_ij)*Vi)'];
    end
end
dDi_dX=[dDi_dP; dDi_dKe];
dg_dX=[dg_dX dDi_dX];

% Positive definiteness of P2 constraint.
PDC=-P2*pd_weight;
[v,d]=eig(PDC);
[Dpd,sort_index]=esort(diag(d));
Vpd=v(:,sort_index);
g=[g;Dpd];      % Write as 'less than' constraint

dDpd_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dPDC_dP_ij=-1*(EL'*P+P*EL)*pd_weight;
        dDpd_dP=[dDpd_dP; diag(Vpd'*(dPDC_dP_ij)*Vpd)'];
    end
end
dDpd_dKe=zeros(pg,n); % 'n' eigenvalues

dDpd_dX=[dDpd_dP;dDpd_dKe];
dg_dX=[dg_dX dDpd_dX];

% State constraints
for k=1:c;
    SC=GAMMA(k,:)*inv(P2)*GAMMA(k,:)'-1;
end

```

```

g=[g;SC];

dSC_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P));
        EL(i,j)=1;EL(j,i)=1;
        dSC_dP_ij=GAMMA(k,)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*GAMMA(k,)'';
        dSC_dP=[dSC_dP; dSC_dP_ij];
    end
end
dSC_dKe=zeros(pg,1);
dSC_dX=[dSC_dP;dSC_dKe];
dg_dX=[dg_dX dSC_dX];
end;

% Control constraints
if control_const==1
    CC=(1/UMAX^2)*((Ke+K0e)*inv(P2)*(Ke+K0e)')-1;
    g=[g;CC];

    dCC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P));
            EL(i,j)=1;EL(j,i)=1;
            dCC_dP_ij=(1/UMAX^2)*((Ke+K0e)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))...
                *(Ke+K0e)');
            dCC_dP=[dCC_dP; dCC_dP_ij];
        end
    end
    dCC_dKe=[];
    for i=1:m
        for j=1:n
            EL=zeros(size(Ke));
            EL(i,j)=1;
            dCC_dKe_ij=(1/UMAX^2)*((EL)*inv(P2)*(Ke+K0e)'+(Ke+K0e)*inv(P2)*(EL)');
            dCC_dKe=[dCC_dKe; dCC_dKe_ij];
        end
    end
    dCC_dX=[dCC_dP; dCC_dKe];
    dg_dX=[dg_dX dCC_dX];
end

% ===== END OF FILE: REACH_DFG.M =====

```

## F.2.4 Plotting Routine

```
function [U,P,Kv]=reach_p(X,A,B,K0v,plot_pts);
% REACH_P is the trajectory plotting routine for use with REACH

% Plotting options
N=5; % plot trajectory from every Nth data point on ellipsoid

if nargin<4
    error('Not enough input arguments.')
```

end

```
% Message to screen
disp(' ')
disp('+++++')
disp('Plotting reaching trajectories')
disp('+++++')
disp(' ')

% Parameter Assignments (extract P,K's from X)
[n,m]=size(B);
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

Kv=Kv+K0v; % Construct composite gain

if n==2 % 2-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2]=ellipse(P2,[],plot_pts);

    % Compute corresponding control effort
    ue=Kv*[x1;x2];

    % Compute maximum control effort on boundary
    U=max(ue);

    % Plot boundary vs. control
    plot(x1,x2,'k');grid on;axis square;hold on

    % Compute reaching and reaching state-space descriptions
    syse=ss(-(A-B*Kv),B,eye(n),zeros(n,m));

    % Plot trajectories for every Nth point on the ellipsoid boundary
    for i=1:N:max(size(x1));
        [y,t,x]=initial(syse,[x1(i),x2(i)],[0:.01:10]);
        plot(x(:,1),x(:,2),'b')
    end

    xlabel('x1'),ylabel('x2'),zlabel('u')
elseif n==3 % 3-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2,x3]=ellipse(P2,[],plot_pts);
```

```

% Compute corresponding control effort
for i=1:size(x1,1)
    for j=1:size(x1,2)
        ue(i,j)=Kv*[x1(i,j);x2(i,j);x3(i,j)];
    end
end

% Compute maximum control effort on boundary
U=max(max(abs(ue)));

% Plot boundary vs. control
surf(x1,x2,x3,ue/U);grid on;axis square;hold on

% Compute reaching and reaching state-space descriptions
syse=ss(-(A-B*Kv),B,eye(n),zeros(n,m));

% Plot trajectories for every Nth point on the ellipsoid boundary
hold on
for i=1:N^2:size(x1,1);
    for j=1:N^2:size(x1,2)
        [y,t,x]=initial(syse,[x1(i,j),x2(i,j),x3(i,j)], [0:.01:10]);
        plot3(x(:,1),x(:,2),x(:,3),'b')
    end
end
end
drawnow
grid on
colormap('copper')
lighting phong
light('Position',[1 -1 5])
h=findobj('Type','surface');
set(h,'FaceLighting','phong',...
    'FaceColor','interp',...
    'EdgeColor',[.4 .4 .4],...
    'BackFaceLighting','reverselit',...
    'AmbientStrength',1,...
    'DiffuseStrength',1);
shading interp
xlabel('x1'),ylabel('x2'),zlabel('x3')

end

% ===== END OF FILE: REACH_P.M =====

```

## F.3 Function Files for Computing Controllable Ellipsoidal Set

### F.3.1 Optimization Routine

```
function [P,Kv,Ke,U,X]=control(A,B,GAMMA,UMAX,Q0,P0,K0v,K0e);
%
% CONTROL finds the largest controllable ellipsoid for a linear system with
% constrained states and input.
%
% Parameter definitions:
%
%   [A,B]: state-space description of linear system
%   GAMMA: state constraints of the form GAMMA*x<=1
%   UMAX: maximum allowable control (assumed symmetric)
%   Q0: specifies decay rate of the Lyapunov function
%   P0: initial guess for the ellipsoid matrix (x'Px<=1)
%   K0v: initial guess for recovering state feedback matrix
%   K0e: initial guess for reaching state feedback matrix
%
%   P: the optimal ellipsoid
%   Kv: the corresponding recovering state feedback matrix
%   Ke: the corresponding reaching state feedback matrix
%   U: the maximum control effort on the boundary of P
%   X: the final parameter search vector (elements of P, K)
%
% Usage:
%
%   [P,Kv,Ke,U,X]=control(A,B,GAMMA,UMAX,Q0,P0,K0v,K0e) finds the
%   largest ellipsoid, P, which is both reachable and recoverable,
%   and corresponding state-feedback matrices, Ke,Kv, for the
%   system [A,B] and the constraints, GAMMA, UMAX. Definitions of
%   Q0, P0, and K0's are optional. If not provided, Q0 is assumed
%   to be zero and P0 is initialized using an LQR-based approach.
%   If K0e is not specified, or if the given K0e does not stabilize
%   [-A,-B], it is chosen (arbitrarily) using the LQR technique.
%   Similarly, if K0v is not specified, or if the given K0v does
%   not stabilize [A,B], it is chosen (arbitrarily) using the LQR
%   technique.
%
% Notice:
%
%   This algorithm is based on the dissertation "Ellipsoidal and
%   Semi-Ellipsoidal Controlled Invariant Sets for Constrained
%   Linear Systems" by Brian O'Dell, Oklahoma State University, 1999.
%
% Start counter for run-time
tic

% DEFINE SIMULATION CONSTANTS
fig_handle=1; % Set the figure handle
ON=1;
OFF=0;
```

```

PARTIAL=0.5;
beta=0.95;      % Decrease factor for size of P; MUST BE LESS THAN 1.0

[n,m]=size(B); % 'n' is number of states, 'm' is number of inputs
pe=(n^2-n)/2+n; % Defines number of ellipse parameters
pg=n*m;        % Defines number of state feedback gain parameters
p=pe+pg;       % Defines the total number of parameters.

PET=1;         % 'Percent Error Tolerance' for terminating searches
flag=1;        % Defines as controllable ellipsoid

plotting=PARTIAL; % Turn plotting 'on'/'off'
plot_pts=150;    % Number of points to use in plotting

% DEFINE SEARCH PARAMETERS (for search OPTIONS, type 'help foptions')
max_passes=20;
min_iters=50*p;
max_iters=70*p; % Maximum number of search iterations per cycle (nom 70*p)
weight=5e1;     % Weighting coefficient for constraint vector (nom 50)
pd_weight=1e3;  % Additional (multiplicative) weight for pos. def. const.
options(1)=1;   % Turns off display & suppresses warnings
options(6)=1;
options(14)=max_iters; % Set the maximum number of iterations per pass
options(16)=1e-10;
options(17)=1e-8;

% INITIALIZE SEARCH DATA VECTORS
Fstart=[];      % Vector of cost function values before each pass
Gstart=[];      % Matrix of constraint function values before each pass
Fstop=[];       % Vector of cost function values after each pass
Gstop=[];       % Matrix of constraint function values after each pass
search_log=[];  % Matrix of parameter values after each pass

% CHECK VALIDITY OF INPUT DEFINITIONS
N=nargin;
if N<4
    error('Not enough input arguments.')
elseif N==4
    QO=[];PO=[];K0v=[];K0e=[];
elseif N==5
    PO=[];K0v=[];K0e=[];
elseif N==6
    K0v=[];K0e=[];
elseif N==7
    K0e=[];
end

% CHECK SIZE OF A,B
if size(A,1)~=size(A,2)
    error('A is non-square.')

```



```

elseif size(A,1)~=n
    error('A and B must have same number of rows.')
end

% CHECK CONSTRAINT SPACE SIZE
if rank(GAMMA)<n
    error('State constraints do not form closed set.')
end

% CHECK UMAX
if min(UMAX)<=0
    error('Control constraints must be positive.')
end

% CHECK Q0
if isempty(Q0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing Q0 matrix:')
    disp(' ')
    Q0=zeros(size(A))
elseif max(max(abs(Q0-Q0')))>0 % Check for symmetry
    error('Q0 is not symmetric.')
elseif min(eig(Q0))<0 % Check for positive definiteness
    error('Q0 is not positive definite.')
end

% USE LQR SOLUTION FOR ANY NECESSARY INITIALIZATIONS
poles=[1:n]; % Define (arbitrary) positive pole locations
k0=place(A,B,poles); % Compute state-feedback gain
R=eye(m); % Arbitrary pos. def. weighting matrix for LQR
[temp,p20]=lqr(A-B*k0,B,Q0,R); % Compute stabilizing controller and ellipse
dk=inv(R)*B'*p20; % Define stabilizing controller

% CHECK KOv
if isempty(KOv) % Initialize with LQR if KOv not given
    % Display initialization message
    disp(' ')
    disp('Initializing KOv matrix:')
    disp(' ')
    KOv=k0+dk/2 % Define marginally stabilizing controller
    Kv=dk/4 % Define gain such that (KOv+Kv) is stabilizing
    % (in positive time)
else
    if (size(KOv,1)~=m)|(size(KOv,2)~=n) % Check size of KOv
        error('KOv must be same size as B transpose.')
    end
    if max(real(eig(A-B*KOv)))>0 % Check stability of A-B*KOv
        error('KOv does not stabilize A.')
    end
end

```

```

end
end

% CHECK K0e
if isempty(K0e) % Initialize with LQR if K0v not given
    % Display initialization message
    disp(' ')
    disp('Initializing K0e matrix:')
    disp(' ')
    K0e=k0+dk/2 % Define marginally stabilizing controller
    Ke=-dk/4 % Define gain such that (K0e+Ke) is stabilizing
    % (in negative time)
else
    if (size(K0e,1)~=m)|(size(K0e,2)~=n) % Check size of K0e
        error('K0e must be same size as B transpose.')
    end
    if max(real(eig((-A)-(-B)*K0e)))>0 % Check stability of A-B*K0e
        error('K0e does not stabilize A.')
    end
end

% CHECK P0
if isempty(P0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing P0 matrix:')
    disp(' ')
    con=[(K0v+Kv)/UMAX;(K0e+Ke)/UMAX;GAMMA]; % Define matrix of all bounds
    p20=p20*max(diag((con*inv(p20)*con'))); % Normalize ellipsoid to touch one
    % of the bounds.
    P0=real(sqrtm(p20)) % Compute P0 from p20
    P20=P0'*P0
elseif max(max(abs(P0-P0')))>0 % Check for symmetry
    error('P0 is not symmetric.')
elseif min(eig(P0))<0 % Check for positive definiteness
    error('P0 is not positive definite.')
end

% ASSIGN INITIAL CONDITIONS TO ELEMENTS OF SEARCH SPACE VECTOR
X=pk2x(P0,Kv,Ke,n,m); % Parametrize search with P0, the square root of the
% ellipsoid matrix, to minimize search errors with
% positive definiteness of ellipse.

if plotting==ON
    % Plot initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    ellipse(P20,[],plot_pts);grid on;axis square;
    title('INITIAL CONDITION')
    drawnow;

```

```

    % Plot trajectories of initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    control_p(X,A,B,KOv,KOe,plot_pts);
    title('TRAJECTORIES OF INITIAL CONDITION')
    drawnow;
end

save control.mat X

pvol=det(inv(P20));

save control_best.mat X pvol

% =====

% OPTIMIZE SOLUTION

options(14)=max_iters; % Set the maximum number of iterations to max_iters
pass=1;                % Reset the counter
control_const=ON;     % Turn the control constraint 'on'

change=100;
options(10)=max_iters;
while (pass<=max_passes)&((abs(change)>PET)|(options(10)>(0.8*options(14))))
    % Loop until change in ellipsoid size < 2%

    disp(' ')
    disp(sprintf(' ===== OPTIMIZING: CHANGE = %6.2f %% ===== ',...
        change))
    disp(' ')

    % Increase search iterations if necessary
    if round(pass/5)*5==pass
        options(14)=min([min_iters,round(options(14)/1.1)]);
    end

    % Log start-of-pass search constraint vector and cost function
    [fstart,g]=control_fg(X,A,B,KOv,KOe,GAMMA,UMAX,Q0,control_const,...
        weight,pd_weight);
    Gstart=[Gstart g];
    Fstart=[Fstart fstart];

    % Scale ellipse dow by factor of beta to start the next pass off the
    % constraints
    X(1:pe,1)=X(1:pe,1)/sqrt(beta);

    % Perform search.
    options(9)=OFF;
    [X,options]=constr('control_fg',X,options,[],[],'control_dfg',...
        A,B,KOv,KOe,GAMMA,UMAX,Q0,control_const,weight,pd_weight);

    % Log end-of-pass search constraint vector and cost function

```

```

[fstop,g,U,constraints]=control_fg(X,A,B,K0v,K0e,GAMMA,UMAX,Q0,...
    control_const,weight,pd_weight);
Gstop=[Gstop g];
Fstop=[Fstop fstop];

% Compute change in cost function
change=100*(exp(fstart)-exp(fstop))/exp(fstart);

% Log parameter space, control value, and number of iterations
search_log=[search_log X];

% Parameter Assignments
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% Plot ellipsoid in state space
if plotting==ON
    figure(fig_handle);clf;drawnow;
    fig_handle=fig_handle+1;
    ellipse(P2,[],plot_pts);grid on;axis square;
    title(sprintf('OPTIMIZING: CHANGE = %1.4g',change));
    drawnow
end

save control.mat X U

if det(inv(P2))>pvol
    pvol=det(inv(P2));
    save control_best.mat X P2 Ke pvol
end

pass=pass+1;

end

% =====

% DISPLAY TERMINATION CRITERIA

disp(' ')
disp(sprintf('Search terminated on change in cost function of %0.4g %%.',...
    change))
disp(' ')

% =====

% CLOSING TASKS

% Plot trajectories

```

```

if plotting>=PARTIAL
    figure(fig_handle);clf;drawnow;
    control_p(X,A,B,K0v,K0e,plot_pts);
end

% Compute composite gain.
Kv=Kv+K0v;
Ke=Ke+K0e;

% Define output ellipsoid matrix
P=P2;

% Compute maximum gain on ellipse, U
[f,g,U,constraints]=control_fg(X,A,B,K0v,K0e,GAMMA,UMAX,Q0,...
    control_const,weight,pd_weight);

% Clean up hard drive
delete control.mat
delete control_best.mat

% Terminate counter and display elapsed time
toc

% ===== END OF FILE: CONTROL.M =====

```

### F.3.2 Cost Function and Constraints

```
function [f,g,Uc,constraints]=control_fg(X,A,B,K0v,K0e,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight);
% CONTROL_FG is the cost function and constraint routine for use with CONTROL.

% parametrizes in terms of sqrt(P) and Kv, Ke
% modifies definition of (at end) and pos. def. inequalities

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
[p]=max(size(X));
[c]=size(GAMMA,1);
flag=1;

% PARAMETER ASSIGNMENTS
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(det(P2));

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Invariance constraint
Acl=A-B*(Kv+K0v);
temp=P2*Acl+Acl'*P2+Q; % Write as negative definite form
[v,d]=eig(temp);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

Acl=A-B*(Ke+K0e);
temp=P2*(-Acl)+(-Acl)'+P2+Q; % Write as negative definite form
[v,d]=eig(temp);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

% Positive definiteness constraints.
[v,d]=eig(-P2*pd_weight); % Write as negative definite form
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
```

```

g=[g;D];      % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Positive Definiteness');
end

% State constraints
for i=1:c;
    g=[g; GAMMA(i,:)*inv(P2)*GAMMA(i,:)'-1];
    constraints=char(constraints,['State Constraint ' num2str(i)]);
end;

% Control constraints
if control_const==1
    Uv=sqrt((Kv+K0v)*inv(P2)*(Kv+K0v)');
    g=[g; (Uv*Uv)/(UMAX*UMAX)-1];      % Normalize to 1
    constraints=char(constraints,'Control');
    Ue=sqrt((Ke+K0e)*inv(P2)*(Ke+K0e)');
    g=[g; (Ue*Ue)/(UMAX*UMAX)-1];      % Normalize to 1
    constraints=char(constraints,'Control');
end

Uc=max(Uv,Ue);

% ===== END OF FILE: CONTROL_FG.M =====

```

### F.3.3 Derivatives of Cost Function and Constraints

```

function [df_dX,dg_dX]=control_dfg(X,A,B,KOv,KOe,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight);
% CONTROL_DFG is the derivative cost function and constraint routine
%   for use with CONTROL.

% 1) parametrizes in terms of sqrt(P) and Ke
% 2) optional weights on constraints

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
p=max(size(X));
c=size(GAMMA,1);
pg=n*m; % number of state feedback gain parameters
pe=p-pg; % number of ellipsoid parameters
flag=1;

% PARAMETER ASSIGNMENTS
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(det(P2));

Pi=inv(P);

df_dP=[];
xindex=0;
for i=1:n
    for j=i:n
        if i==j
            df_dP=[df_dP;2*Pi(i,j)];
        else
            df_dP=[df_dP;4*Pi(i,j)];
        end
    end
end
df_dKv=zeros(size(Kv))';
df_dKe=zeros(size(Ke))';
df_dX=[df_dP;df_dKv;df_dKe];

% =====

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear constraint array
dg_dX=[]; % clear constraint derivative array

% Invariance constraint (recoverable/positive time)

```



```

Acl=A-B*(Kv+K0v);
IC=P2*Acl+Acl'*P2+Q;
[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
Vi=v(:,sort_index);
g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*Acl+Acl'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKv=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Kv));
        EL(i,j)=1;
        dAcl_dKv_ij=-B*EL;
        dIC_dKv_ij=P2*(dAcl_dKv_ij)+(dAcl_dKv_ij)'*P2;
        dDi_dKv=[dDi_dKv; diag(Vi'*(dIC_dKv_ij)*Vi)'];
    end
end
dDi_dKe=zeros(pg,n); % 'n' eigenvalues

dDi_dX=[dDi_dP; dDi_dKv; dDi_dKe];
dg_dX=[dg_dX dDi_dX];

% Invariance constraint (reachability/negative time)
Acl=A-B*(Ke+K0e);
IC=P2*(-Acl)+(-Acl)'*P2+Q;
[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
Vi=v(:,sort_index);
g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*(-Acl)+(-Acl)'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKv=zeros(pg,n);
dDi_dKe=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Ke));

```

```

        EL(i,j)=1;
        dAcl_dKe_ij=-B*EL;
        dIC_dKe_ij=P2*(-dAcl_dKe_ij)+(-dAcl_dKe_ij)'+P2;
        dDi_dKe=[dDi_dKe; diag(Vi'*(dIC_dKe_ij)*Vi)'];
    end
end
dDi_dX=[dDi_dP; dDi_dKv; dDi_dKe];
dg_dX=[dg_dX dDi_dX];

% Positive definiteness of P constraints.
PDC=-P2*pd_weight;
[v,d]=eig(PDC);
[Dpd,sort_index]=esort(diag(d));
Vpd=v(:,sort_index);
g=[g;Dpd]; % Write as 'less than' constraint

dDpd_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P));
        EL(i,j)=1;EL(j,i)=1;
        dPDC_dP_ij=-1*(EL'*P+P*EL)*pd_weight;
        dDpd_dP=[dDpd_dP; diag(Vpd'*(dPDC_dP_ij)*Vpd)'];
    end
end
dDpd_dKv=zeros(pg,n); % 'n' eigenvalues
dDpd_dKe=zeros(pg,n);

dDpd_dX=[dDpd_dP;dDpd_dKv;dDpd_dKe];
dg_dX=[dg_dX dDpd_dX];

% State constraints
for k=1:c
    SC=GAMMA(k,:)*inv(P2)*GAMMA(k,:)'-1;
    g=[g;SC];

    dSC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P));
            EL(i,j)=1;EL(j,i)=1;
            dSC_dP_ij=GAMMA(k,:)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*GAMMA(k,:)';
            dSC_dP=[dSC_dP; dSC_dP_ij];
        end
    end
    dSC_dKv=zeros(pg,1);
    dSC_dKe=zeros(pg,1);
    dSC_dX=[dSC_dP;dSC_dKv;dSC_dKe];
    dg_dX=[dg_dX dSC_dX];
end
end

```

```

% Control constraints
if control_const==1
    % Recoverable control
    CC=(1/UMAX^2)*((Kv+K0v)*inv(P2)*(Kv+K0v)')-1;
    g=[g;CC];

    dCC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P2));
            EL(i,j)=1;EL(j,i)=1;
            dCC_dP_ij=(1/UMAX^2)*((Kv+K0v)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*...
                (Kv+K0v)');
            dCC_dP=[dCC_dP; dCC_dP_ij];
        end
    end
    dCC_dKv=[];
    for i=1:m
        for j=1:n
            EL=zeros(size(Kv));
            EL(i,j)=1;
            dCC_dKv_ij=(1/UMAX^2)*((EL)*inv(P2)*(Kv+K0v)'+(Kv+K0v)*inv(P2)*(EL)');
            dCC_dKv=[dCC_dKv; dCC_dKv_ij];
        end
    end
    dCC_dKe=zeros(pg,1);
    dCC_dX=[dCC_dP; dCC_dKv; dCC_dKe];
    dg_dX=[dg_dX dCC_dX];

    % Reachable control
    CC=(1/UMAX^2)*((Ke+K0e)*inv(P2)*(Ke+K0e)')-1;
    g=[g;CC];

    dCC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P2));
            EL(i,j)=1;EL(j,i)=1;
            dCC_dP_ij=(1/UMAX^2)*((Ke+K0e)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*...
                (Ke+K0e)');
            dCC_dP=[dCC_dP; dCC_dP_ij];
        end
    end
    dCC_dKv=zeros(pg,1);
    dCC_dKe=[];
    for i=1:m
        for j=1:n
            EL=zeros(size(Ke));
            EL(i,j)=1;
            dCC_dKe_ij=(1/UMAX^2)*((EL)*inv(P2)*(Ke+K0e)'+(Ke+K0e)*inv(P2)*(EL)');
            dCC_dKe=[dCC_dKe; dCC_dKe_ij];
        end
    end
    dCC_dX=[dCC_dP; dCC_dKv; dCC_dKe];

```

```
dg_dX=[dg_dX dCC_dX];  
end
```

```
% ===== END OF FILE: CONTROL_DFG.M =====
```

### F.3.4 Plotting Routine

```
function [U,P,Kv,Ke]=control_p(X,A,B,K0v,K0e,plot_pts);
% CONTROL_P is the trajectory plotting routine for use with CONTROL

% Plotting options
N=8; % plot trajectory from every Nth data point on ellipsoid
flag=1;

if nargin<4
    error('Not enough input arguments.')
end

% Message to screen
disp(' ')
disp('+++++')
disp('Plotting trajectories')
disp('+++++')
disp(' ')

% Parameter Assignments (extract P,K's from X)
[n,m]=size(B);
p=max(size(X));

[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

Kv=Kv+K0v; % Define composite gains
Ke=Ke+K0e;

if n==2 % 2-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2]=ellipse(P2,[],plot_pts);

    % Compute corresponding control effort
    ue=Kv*[x1;x2];

    % Compute maximum control effort on boundary
    U=max(ue);

    % Plot boundary vs. control
    plot(x1,x2,'k');grid on;axis square;hold on

    % Compute reaching and recovering state-space descriptions
    sysv=ss(A-B*Kv,B,eye(n),zeros(n,m));
    syse=ss(-A+B*Ke,B,eye(n),zeros(n,m));

    % Plot trajectories for every Nth point on the ellipsoid boundary
    for i=1:ceil(sqrt(N)):max(size(x1));
        [y,t,x]=initial(sysv,[x1(i),x2(i)],[0:.01:10]);
        plot(x(:,1),x(:,2),'r')
        [y,t,x]=initial(syse,[x1(i),x2(i)],[0:.01:10]);
    end
end
```

```

    plot(x(:,1),x(:,2),'b')
end

xlabel('x1'),ylabel('x2')
elseif n==3 % 3-D ELLIPSOID
% Compute ellipsoid boundary
[x1,x2,x3]=ellipse(P2,[],plot_pts);

% Compute corresponding control effort
for i=1:size(x1,1)
    for j=1:size(x1,2)
        uv(i,j)=Kv*[x1(i,j);x2(i,j);x3(i,j)];
        ue(i,j)=Ke*[x1(i,j);x2(i,j);x3(i,j)];
    end
end

% Compute maximum control effort on boundary
U=max(max([abs(uv);abs(ue)]));

% Plot boundary vs. control
subplot(121)
surf(x1,x2,x3,uv/U);grid on;axis square;hold on
subplot(122)
surf(x1,x2,x3,ue/U);grid on;axis square;hold on

% Compute reaching and recovering state-space descriptions
sysv=ss(A-B*Kv,B,eye(n),zeros(n,m));
syse=ss(-A+B*Ke,B,eye(n),zeros(n,m));

% Plot trajectories for every Nth point on the ellipsoid boundary
subplot(121)
hold on
for i=1:N^2:size(x1,1);
    for j=1:N^2:size(x1,2)
        [y,t,x]=initial(sysv,[x1(i,j),x2(i,j),x3(i,j)], [0:.01:10]);
        plot3(x(:,1),x(:,2),x(:,3),'r')
    end
end
end
drawnow
grid on
colormap('copper')
lighting phong
light('Position',[1 -1 5])
h=findobj('Type','surface');
set(h,'FaceLighting','phong',...
    'FaceColor','interp',...
    'EdgeColor',[.4 .4 .4],...
    'BackFaceLighting','reverselit',...
    'AmbientStrength',1,...
    'DiffuseStrength',1);
shading interp
xlabel('x1'),ylabel('x2'),zlabel('x3')
title('Recovering Trajectories')

```

```

subplot(122)
hold on
for i=1:N^2:size(x1,1);
    for j=1:N^2:size(x1,2)
        [y,t,x]=initial(syse,[x1(i,j),x2(i,j),x3(i,j)], [0:.01:10]);
        plot3(x(:,1),x(:,2),x(:,3),'b')
    end
end
drawnow
grid on
colormap('copper')
lighting phong
light('Position',[1 -1 5])
h=findobj('Type','surface');
set(h,'FaceLighting','phong',...
    'FaceColor','interp',...
    'EdgeColor',[.4 .4 .4],...
    'BackFaceLighting','reverselit',...
    'AmbientStrength',1,...
    'DiffuseStrength',1);
shading interp
xlabel('x1'),ylabel('x2'),zlabel('x3')
title('Reaching Trajectories')

end

```

```

% ===== END OF FILE: CONTROL_P.M =====

```

## F.4 Miscellaneous Files

### F.4.1 Plotting Point Generator for Ellipsoidal Set

```
function [xx,yy,zz] = ellipse(P,X0,n,linetype)
% ELLIPSE Generates plotting points for 2-D/3-D ellipsoid.
% [X,Y,Z] = ELLIPSE(P,X0,n) generates the unit ellipsoid
%
% (x-X0)'*P*(x-X0)=1
%
% For P matrix (3x3), ELLIPSE generates three (n+1)x(n+1)
% matrices so that SURF(X,Y,Z) produces the 3-D ellipsoid,
% For P matrix (2x2), ELLIPSE generates two (n+1)x(1)
% vectors so that PLOT(X,Y) produces the 2-D ellipsoid.
%
% The arguments X0 and n are optional. Default values are
% the origin for X0 and 40 points for n.
%
% ELLIPSE(P,X0,n) without any return variables graphs the
% ellipse using SURFACE/PLOT.
%
% Original code: SPHERE.M
% Clay M. Thompson 4-24-91, CBM 8-21-92.
% Copyright (c) 1984-98 by The MathWorks, Inc.
% $Revision: 5.3 $ $Date: 1997/11/21 23:46:48 $
%
% Modified code: ELLIPSE.M
% Brian D. O'Dell 11-19-98

if nargin == 0, error('Must define ellipsoid matrix, P.');
```

```
end

% CHECK VALIDITY OF P
if size(P,1)~=size(P,2)
    error('P must be square.')
```

```
end
if min(eig(P))<=0
    error('P must be positive definite.')
```

```
end

% COMPUTE NUMBER OF STATES
states=size(P,1);
if (states~=2)&(states~=3)
    error('P must be a 2x2 or 3x3 matrix.')
```

```
end

% CHECK FOR OPTIONAL ARGUMENTS
if nargin==1
    X0=[];
    n=40;
elseif nargin==2
    n=40;
end

% SET X0 TO ORIGIN IF NOT OTHERWISE DEFINED
```



```

if isempty(X0)
    X0=zeros(states,1);
end

if states==2 % 2-D ELLIPSE
    % -pi <= theta <= pi is a row vector

    theta = (-n:2:n)/n*pi;

    sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;

    x0 = cos(theta); % Define points for a unit circle
    y0 = sintheta;

    for i=1:(n+1) % Loop through the data points
        temp=[x0(i);y0(i)]; % Create a vector for the data point
        alpha=sqrt(temp'*P*temp); % Compute the scaling factor for unit ellipse
        x(i)=x0(i)/alpha+X0(1); % Scale the data points
        y(i)=y0(i)/alpha+X0(2);
    end

    if nargout == 0
        plot(x,y)
        xlabel('x1'),ylabel('x2')
    else
        xx = x; yy = y;
    end
end

else % 3-D ELLIPSE
    % If plotting, display full ellipse; for trajectories, generate quadrant
    if nargout==0
        % -pi <= theta <= pi is a row vector.
        % -pi/2 <= phi <= pi/2 is a column vector.

        theta = (-n:2:n)/n*pi;
        phi = (-n:2:n)'/n*pi/2;
        cosphi = cos(phi); cosphi(1) = 0; cosphi(n+1) = 0;
        sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;
    else
        % -pi/2 <= theta <= pi/2 is a row vector.
        % -pi/2 <= phi <= 0 is a column vector.

        theta = (-n:2:n)/n*pi/2;
        phi = (-2*n:2:0)'/(2*n)*pi/2;
        cosphi = cos(phi);
        sintheta = sin(theta);
    end

    x0 = cosphi*cos(theta); % Define points for a unit sphere
    y0 = cosphi*sintheta;
    z0 = sin(phi)*ones(1,n+1);

    for i=1:max(size(theta)) % Loop through the data points
        for j=1:max(size(phi))

```

```

        temp=[x0(i,j);y0(i,j);z0(i,j)]; % Create a vector for the data
        alpha=sqrt(temp'*P*temp); % Scaling factor for unit ellipse
        x(i,j)=x0(i,j)/alpha+X0(1); % Scale data points
        y(i,j)=y0(i,j)/alpha+X0(2);
        z(i,j)=z0(i,j)/alpha+X0(3);
    end
end

if nargout == 0 % Plot if no output
    disp(' ')
    disp('+++++')
    disp('Plotting ellipse')
    disp('+++++')
    disp(' ')

    surf(x,y,z)
    grid on
    colormap('copper')
    lighting phong
    light('Position',[1 -1 5])
    h=findobj('Type','surface');
    set(h,'FaceLighting','phong',...
        'FaceColor','interp',...
        'EdgeColor',[.4 .4 .4],...
        'BackFaceLighting','reverselit',...
        'AmbientStrength',1,...
        'DiffuseStrength',1);
    shading interp
    xlabel('x1'),ylabel('x2'),zlabel('x3')
else
    xx = x; yy = y; zz = z;
end
end

% ===== END OF FILE: ELLIPSE.M =====

```

## F.4.2 Search Parameter/Matrix Parameter Mapping Routines

### Ellipsoid Matrix and Control Gains to Search Vector

```
function [X]=pk2x(Pi,K1,K2,n,m)
% PK2X assigns elements of the ellipse matrix's inverse, Pi, and the
% state-feedback matrix, K, to elements of the search vector, X.

if nargin==4
    m=n;
    n=K2;

    X=[];
    xindex=0;
    for i=1:n
        for j=i:n
            xindex=xindex+1;
            X(xindex,1)=Pi(i,j);
        end
    end
    for i=1:m
        for j=1:n
            xindex=xindex+1;
            X(xindex,1)=K1(i,j);
        end
    end
elseif nargin==5
    X=[];
    xindex=0;
    for i=1:n
        for j=i:n
            xindex=xindex+1;
            X(xindex,1)=Pi(i,j);
        end
    end
    for i=1:m
        for j=1:n
            xindex=xindex+1;
            X(xindex,1)=K1(i,j);
        end
    end
    for i=1:m
        for j=1:n
            xindex=xindex+1;
            X(xindex,1)=K2(i,j);
        end
    end
end

% ===== END OF FILE: PK2X.M =====
```

### Search Vector to Ellipsoid Matrix and Control Gains

```
function [Pi,K1,K2]=x2pk(X,n,m,flag)
```

```
% X2PK assigns elements of the search vector, X, to elements of the ellipse  
% matrix's inverse, Pi, and to the state-feedback gain, K.
```

```
if nargin==3  
    flag=0;  
end  
  
Pi=[];  
xindex=0;  
for i=1:n  
    for j=i:n  
        xindex=xindex+1;  
        Pi(i,j)=X(xindex,1);  
        Pi(j,i)=X(xindex,1);  
    end  
end  
K1=[];  
for i=1:m  
    for j=1:n  
        xindex=xindex+1;  
        K1(i,j)=X(xindex,1);  
    end  
end  
if flag==1  
    K2=[];  
    for i=1:m  
        for j=1:n  
            xindex=xindex+1;  
            K2(i,j)=X(xindex,1);  
        end  
    end  
else  
    K2=[];  
end
```

```
% ===== END OF FILE: X2PK.M =====
```

## Appendix G

# Matlab Code for Semi-Ellipsoidal Sets

### G.1 Function Files for Computing Recoverable Semi-Ellipsoidal Set

#### G.1.1 Optimization Routine

```
function [P,Kv,U,X,Fstop,Gstop]=recover(A,B,GAMMA,UMAX,Q0,PO,K0v);
%
% RECOVER finds the largest recoverable set for a linear system with
% constrained states and inputs, where the subset is constructed from
% the intersection of the state constraints and the computed ellipsoid.
%
% Parameter definitions:
%
%   [A,B]: state-space description of linear system
%   GAMMA: state constraints of the form  $GAMMA*x \leq 1$ 
%   UMAX: maximum allowable control (assumed symmetric)
%   Q0: specifies decay rate of the Lyapunov function
%   PO: initial guess for the ellipsoid matrix ( $x'Px \leq 1$ )
%   K0v: initial guess for state feedback (must stabilize [A,B])
%
%   P: the optimal ellipsoid
%   Kv: the corresponding state feedback gain matrix
%   U: the maximum control effort (via state feedback) in subset
%   X: the final parameter search vector (elements of P, Kv)
%
% Usage:
%
%   [P,Kv,U,X]=recover(A,B,GAMMA,UMAX,Q0,PO,K0v) finds the largest
%   recoverable ellipsoid, P, and corresponding state-feedback
%   matrix, Kv, for the system [A,B] and the constraints, GAMMA,
%   UMAX. Definitions of Q0, PO, and K0v are optional. If not
%   provided, Q0 is assumed to be zero and PO is initialized using
%   an LQR algorithm. If K0v is not specified, or if the given
%   K0v does not stabilize [A,B], it is chosen (arbitrarily) using
%   the LQR technique.
%
% Notice:
```

```

%
% This algorithm is based on the dissertation "Ellipsoidal and
% Semi-Ellipsoidal Controlled Invariant Sets for Constrained
% Linear Systems" by Brian O'Dell, Oklahoma State University, 1999.
%

% Start counter for run-time
tic

% DEFINE SIMULATION CONSTANTS
fig_handle=1; % Set the figure handle
ON=1; % Switch ON
PARTIAL=0.5; % Switch HALF-ON
OFF=0; % Switch OFF
beta=0.95; % Decrease factor for size of P; MUST BE LESS THAN 1.0
tol=1e-5; % Error tolerance for terminating search
gradient_tol=1e-2; % Minimum gradient for maximal control search
PET=1; % 'Percent Error Tolerance' for terminating searches

% DEFINE GENERAL PARAMETERS
plotting=PARTIAL; % Turn plotting 'on/off'; PARTIAL plots only final result
plot_pts=150; % Number of points to use in plotting
grad_check=OFF; % Checks analytical gradients against numerical estimates
search_output=ON; % Displays intermediate search results

% DEFINE MATRIX DIMENSIONING CONSTANTS
[n,m]=size(B); % 'n' is number of states, 'm' is number of inputs
c=size(GAMMA,1); % Defines the number of state constraints
pe=(n^2-n)/2+n; % Defines number of ellipse parameters
pg=n*m; % Defines number of state feedback gain parameters
p=pe+pg; % Defines the total number of parameters.

% DEFINE SEARCH PARAMETERS (for search OPTIONS, type 'help foptions')
max_passes=20; % Maximum number of search cycles
min_iters=40*p; % Minimum number of search iterations per cycle
max_iters=40*p; % Maximum number of search iterations per cycle
weight=1e0; % Weighting coefficient for constraint vector
pd_weight=1e0; % Additional (multiplicative) weight for pos. def. const.
inv_weight=1e0; % Additional (multiplicative) weight for invariance const.
options(1)=ON; % Displays intermediate search results
options(14)=max_iters; % Set the maximum number of iterations per pass

% INITIALIZE SEARCH DATA VECTORS
Fstart=[]; % Vector of cost function values before each pass
Gstart=[]; % Matrix of constraint function values before each pass
Fstop=[]; % Vector of cost function values after each pass
Gstop=[]; % Matrix of constraint function values after each pass
search_log=[]; % Matrix of parameter values after each pass

% CHECK VALIDITY OF INPUT DEFINITIONS
N=nargin;
if N<4

```

```

    error('Not enough input arguments.')
```

```

elseif N==4
    Q0=[];P0=[];K0v=[];
elseif N==5
    P0=[];K0v=[];
elseif N==6
    K0v=[];
end

% CHECK SIZE OF A,B
if size(A,1)~=size(A,2)
    error('A is non-square.')
```

```

elseif size(A,1)~=n
    error('A and B must have same number of rows.')
```

```

end

% CHECK CONSTRAINT SPACE SIZE
if size(GAMMA,2)~=n
    error('GAMMA defined with different number of states than A.')
```

```

end

% CHECK UMAX
if min(UMAX)<=0
    error('Control constraints must be positive.')
```

```

end

% CHECK Q0
if isempty(Q0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing Q0 matrix:')
    disp(' ')
    Q0=zeros(size(A))
elseif max(max(abs(Q0-Q0')))>0 % Check for symmetry
    error('Q0 is not symmetric.')
```

```

elseif min(eig(Q0))<0 % Check for positive definiteness
    error('Q0 is not positive definite.')
```

```

end

% USE LQR SOLUTION FOR ANY NECESSARY INITIALIZATIONS
poles=[1:n]; % Define (arbitrary) positive pole locations
k0=place(A,B,poles); % Compute state-feedback gain
R=eye(m); % Arbitrary pos. def. weighting matrix for LQR
[temp,p20]=lqr(A-B*k0,B,Q0,R); % Compute stabilizing controller and ellipse
dk=inv(R)*B'*p20; % Define stabilizing controller

% CHECK K0v
if isempty(K0v) % Initialize with LQR if K0v not given
```

```

% Display initialization message
disp(' ')
disp('Initializing K0v matrix:')
disp(' ')
K0v=k0+dk/2           % Define marginally stabilizing controller
Kv=dk/4              % Define gain such that (K0v+Kv) is stabilizing
                    % (in positive time)
else
    if (size(K0v,1)~=m)|(size(K0v,2)~=n) % Check size of K0v
        error('K0v must be same size as B transpose.')
    end
    if max(real(eig(A-B*K0v)))>0 % Check stability of A-B*K0v
        error('K0v does not stabilize A.')
    end
end
end

% CHECK P0
if isempty(P0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing P0 matrix:')
    disp(' ')
    P0=real(sqrtm(p20)) % Compute P0 from p20
    P20=P0'*P0
elseif max(max(abs(P0-P0')))>0 % Check for symmetry
    error('P0 is not symmetric.')
elseif min(eig(P0))<0 % Check for positive definiteness
    error('P0 is not positive definite.')
end

% ASSIGN P0,K0v TO ELEMENTS OF SEARCH SPACE VECTOR
X=pk2x(P0,Kv,n,m); % Parametrize search with P0, the square root of the
                  % ellipsoid matrix, to minimize search errors with
                  % positive definiteness of ellipse. (P=P0*P0)

if plotting==0N
    % Plot initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    ellipse(P20,[],plot_pts);grid on;axis square;
    title('INITIAL CONDITION')
    drawnow;
    fig_handle=fig_handle+1;

    % Plot trajectories of initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    recover_p(X,A,B,K0v);
    title('TRAJECTORIES OF INITIAL CONDITION')
    drawnow;
    fig_handle=fig_handle+1;
end

pvol=det(inv(P20));

```



```

save recover_best.mat X

% =====

% OPTIMIZE SOLUTION

U=0; % Initialize maximum observed control to zero
UMAX_hat=UMAX; % Initialize pseudo-maximum control to true maximum
control_iters=0; % Initialize counter for control iteration passes

while (U<beta*UMAX)|(U>(UMAX+tol)) % Loop while observed control outside tol.

    disp(' ')
    disp(sprintf(' ===== SEARCHING: UMAX_hat = %6.2f ===== ',...
        UMAX_hat))
    disp(' ')

    options(14)=max_iters; % Set the maximum number of iterations to max_iters
    pass=1; % Reset the counter
    control_const=ON; % Turn the control constraint 'on'

    change=100; % Initialize change in ellipsoid size to 100%
    options(10)=max_iters; % Initialize number of passes to maximum allowable

    while (pass<=max_passes)&((abs(change)>PET)|(options(10)>(beta*options(14))))
        % Loop until change in ellipsoid size < 2%

        disp(' ')
        disp(sprintf(' ===== OPTIMIZING: CHANGE = %6.2f %% ===== ',...
            change))
        disp(' ')

        % Log start-of-pass search constraint vector and cost function
        [fstart,g,U,constraints]=recover_fg(X,A,B,KOv,GAMMA,UMAX_hat,Q0,...
            control_const,weight,pd_weight,inv_weight);
        Gstart=[Gstart g];
        Fstart=[Fstart fstart];
        Pstart=x2pk(X,n,m);
        P2start=Pstart'*Pstart;

        % Scale ellipse down by factor of beta to start the next pass off the
        % constraints
        X(1:pe,1)=X(1:pe,1)/sqrt(beta);

        % Perform search.
        options(9)=OFF;
        [X,options]=constr('recover_fg',X,options,[],[],'recover_dfg',...
            A,B,KOv,GAMMA,UMAX_hat,Q0,...
            control_const,weight,pd_weight,inv_weight);

        % Log end-of-pass search constraint vector and cost function
        [fstop,g,U,constraints]=recover_fg(X,A,B,KOv,GAMMA,UMAX_hat,Q0,...

```

```

    control_const,weight,pd_weight,inv_weight);
Gstop=[Gstop g];
Fstop=[Fstop fstop];

% Parameter Assignments
Pstop=x2pk(X,n,m);
P2stop=Pstop'*Pstop;

% Compute change in cost function
change=100*(trace(P2start)-trace(P2stop))/trace(P2start);

% Log parameter space
search_log=[search_log X];

% Parameter Assignments
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

% Plot ellipsoid in state space
if plotting==ON
    figure(fig_handle);hold off;clf;drawnow;
    ellipse(P2,[],plot_pts);grid on;axis square;hold on
    title(sprintf('OPTIMIZING: CHANGE = %1.4g',change));
    drawnow
    fig_handle=fig_handle+1;
end

save recover.mat X U

% Check for improvement
if det(inv(P2))>pvol
    pvol=det(inv(P2));
    save recover_best.mat X P2 Kv pvol
end

pass=pass+1;

end

% DISPLAY TERMINATION CRITERIA

disp(' ')
disp(sprintf('Search terminated on change in cost function of %0.4g %%.',...
    change))
disp(' ')

% SEARCH FOR MAXIMUM OBSERVED CONTROL ON RESTRICTED STATE-SPACE
disp(' ')
disp(sprintf(' ===== SEARCHING FOR MAXIMUM OBSERVED CONTROL ===== '))
disp(' ')
if control_iters==0
    % Find point which maximizes control on restricted state-space
    umax_options(1)=-1; % Don't display intermediate results

```

```

x0=rand(n,1); % Generate random initial condition for search
x0=constr('recover_u_fg',x0,umax_options,[],[],'recover_u_dfg',K0v,...
    GAMMA,X,n,m);

% Compute maximum control effort
[fumax,gumax,U]=recover_u_fg(x0,K0v,GAMMA,X,n,m);

% Display maximal control on restricted region
U

% Store previous search results
U_old=U;
UMAX_hat_old=UMAX_hat;

% Compute a new estimate for the UMAX_hat that will achieve U=UMAX.
UMAX_hat=UMAX_hat*(beta^(sign(U-UMAX)));

if (U>=beta*UMAX)&(U<=(UMAX+tol))
    %
    disp(' ')
    disp(cat(2,'Search terminated: Maximum observed control ',...
        'within tolerance of UMAX.'))
    disp(' ')
end
else
% Use previous search result for initial condition
x0=constr('recover_u_fg',x0,umax_options,[],[],'recover_u_dfg',...
    K0v,GAMMA,X,n,m);

% Compute maximum control effort
[fumax,gumax,U]=recover_u_fg(x0,K0v,GAMMA,X,n,m);

% Display maximal control on restricted region
U

% Estimate new value of UMAX_hat using secant search
gradient=(U_old-U)/(UMAX_hat_old-UMAX_hat);
UMAX_hat_tmp=UMAX_hat-(U-UMAX)/gradient;

% Store previous search results
U_old=U;
UMAX_hat_old=UMAX_hat;

UMAX_hat=UMAX_hat_tmp;

% Check to see if improvement feasible
if (U>=beta*UMAX)&(U<=(UMAX+tol))
    %
    disp(' ')
    disp(['Search terminated: Maximum observed control ',...
        'within tolerance of UMAX.'])
    disp(' ')
elseif gradient<gradient_tol
    disp(' ')

```

```

disp(['Search terminated: No increase expected in maximum ',...
      'observed control.'])
disp(' ')

% Redefine U to artificially terminate loop
U=UMAX;
end

end

% Advance counter
control_iters=control_iters+1;

end

% =====

% CLOSING TASKS

% Plot final result
if plotting>=PARTIAL

disp(' ')
disp('+++++')
disp('Computing plot points.')
disp('+++++')
disp(' ')

figure(fig_handle);clf;drawnow;
if n==2
    recover_p(X,A,B,K0v,GAMMA,plot_pts);
elseif n==3
    subplot(121)
    semiellipse(P2,GAMMA,[],plot_pts),hold on
    subplot(122)
    recover_p(X,A,B,K0v,GAMMA,plot_pts);
end
fig_handle=fig_handle+1;
end

% Compute composite gain.
Kv=Kv+K0v;

% Define output ellipsoid matrix
P=P2;

% Compute maximum gain on ellipse, U
[f,g,U,constraints]=recover_fg(X,A,B,K0v,GAMMA,UMAX_hat,Q0,...
    control_const,weight,pd_weight,inv_weight);
Gstop=[Gstop g];
Fstop=[Fstop f];

```

```
% Clean up hard drive
delete recover.mat
delete recover_best.mat
```

```
% Terminate counter and display elapsed time
toc
```

```
% ===== END OF FILE: RECOVER.M =====
```

## G.1.2 Cost Function and Constraints

```
function [f,g,Uv,constraints]=recover_fg(X,A,B,K0v,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight,inv_weight);
% RECOVER_FG is the cost function and constraint routine for use with RECOVER.

% parametrizes in terms of sqrt(P) and Kv

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
[p]=max(size(X));
[c]=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(trace(P2));

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Invariance constraint
Acl=A-B*(Kv+K0v);
IC=P2*Acl+Acl'*P2+Q; % Write as negative definite form
[v,d]=eig(IC);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D*inv_weight]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

% Positive definiteness constraints.
[v,d]=eig(-P2*pd_weight); % Write as negative definite form
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Positive Definiteness');
end

% State constraints (See dissertation for details)
for i=1:c;
    Gamma=GAMMA(i,:);
    w=null(Gamma*(A-B*(Kv+K0v)));
```

```

if size(w,2)==n % Gamma*(A-B*(Kv+K0v)) is zero; entire state-space valid
    g=[g;0];
    Gamma=GAMMA(i,:),A,B,Kv,K0v,Gamma*(A-B*(Kv+K0v))
    w=null(Gamma*(A-B*(Kv+K0v)))
    pause

else
    g=[g; (Gamma*w)*inv(w'*P2*w)*(Gamma*w)'-1];
end

constraints=char(constraints,['State Constraint ' num2str(i)]);
end;

% Control constraints
if control_const==1
    Uv=sqrt((Kv+K0v)*inv(P2)*(Kv+K0v)');
    g=[g; (Uv*Uv)/(UMAX*UMAX)-1]; % Normalize to 1
    constraints=char(constraints,'Control');
end

% ===== END OF FILE: REACH_FG.M =====

```

### G.1.3 Derivatives of Cost Function and Constraints

```
function [df_dX,dg_dX]=recover_dfg(X,A,B,K0v,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight,inv_weight);
% RECOVER_DFG is the derivative cost function and constraint routine
%   for use with RECOVER.

% parametrizes in terms of sqrt(P) and Kv

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
p=max(size(X));
c=size(GAMMA,1);
pg=n*m; % number of state feedback gain parameters
pe=p-pg; % number of ellipsoid parameters

% PARAMETER ASSIGNMENTS
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(trace(P2));

df_dP=[];
xindex=0;
for i=1:n
    for j=i:n
        if i==j
            df_dP=[df_dP;2*P(i,j)/trace(P2)];
        else
            df_dP=[df_dP;4*P(i,j)/trace(P2)];
        end
    end
end
df_dKv=zeros(size(Kv))';
df_dKe=zeros(size(Ke))';
df_dX=[df_dP;df_dKv;df_dKe];

% =====

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear constraint array
dg_dX=[]; % clear constraint derivative array

% Invariance constraint
Acl=A-B*(Kv+K0v);
IC=P2*Acl+Acl'*P2+Q;
[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
```



```

Vi=v(:,sort_index);
g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*Acl+Acl'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKv=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Kv));
        EL(i,j)=1;
        dAcl_dKv_ij=-B*EL;
        dIC_dKv_ij=P2*(dAcl_dKv_ij)+(dAcl_dKv_ij)'*P2;
        dDi_dKv=[dDi_dKv; diag(Vi'*(dIC_dKv_ij)*Vi)'];
    end
end

dDi_dX=[dDi_dP; dDi_dKv]*inv_weight;
dg_dX=[dg_dX dDi_dX];

% Positive definiteness of P constraints.
PDC=-P2*pd_weight;
[v,d]=eig(PDC);
[Dpd,sort_index]=esort(diag(d));
Vpd=v(:,sort_index);
g=[g;Dpd];      % Write as 'less than' constraint

dDpd_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P));
        EL(i,j)=1;EL(j,i)=1;
        dPDC_dP_ij=-1*(EL'*P+P*EL)*pd_weight;
        dDpd_dP=[dDpd_dP; diag(Vpd'*(dPDC_dP_ij)*Vpd)'];
    end
end
dDpd_dKv=zeros(pg,n); % 'n' eigenvalues

dDpd_dX=[dDpd_dP;dDpd_dKv];
dg_dX=[dg_dX dDpd_dX];

% State constraints
for k=1:c;
    Gamma=GAMMA(k,:);
    w=null(Gamma*(A-B*(Kv+K0v)));
end

```

```

if size(w,2)==n % Gamma*(A-B*(Kv+K0v)) is zero; entire state-space valid
    SC=0;

    g=[g;SC];

    dSC_dP=zeros(pe,1);
    dSC_dKv=zeros(pg,1);
    dSC_dX=[dSC_dP;dSC_dKv];
    dg_dX=[dg_dX dSC_dX];
else
    SC=(Gamma*w)*inv(w'*P2*w)*(Gamma*w)'-1;

    g=[g;SC];

    dSC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P2));
            EL(i,j)=1;EL(j,i)=1;

            dinv_dP_ij=-inv(w'*P2*w)*(w'*(P*EL+EL*P)*w)*inv(w'*P2*w);
            dSC_dP_ij=(Gamma*w)*dinv_dP_ij*(Gamma*w)';

            dSC_dP=[dSC_dP; dSC_dP_ij];
        end
    end
    dSC_dKv=[];
    for i=1:m
        for j=1:n
            % Use numerical approximation
            EL=zeros(size(Kv));
            EL(i,j)=1;
            deltaKv_ij=EL*1e-5;
            deltaw=null(Gamma*(A-B*(Kv+K0v+deltaKv_ij)))-w;
            gradw=deltaw/1e-5;
            dinv_dKv_ij=-inv(w'*P2*w)*(gradw'*P2*w+w'*P2*gradw)*inv(w'*P2*w);
            dSC_dKv_ij=(Gamma*gradw)*inv(w'*P2*w)*(Gamma*w)'+...
                (Gamma*w)*dinv_dKv_ij*(Gamma*w)'+...
                (Gamma*w)*inv(w'*P2*w)*(Gamma*gradw)';
            dSC_dKv=[dSC_dKv; dSC_dKv_ij];
        end
    end
    dSC_dX=[dSC_dP;dSC_dKv];
    dg_dX=[dg_dX dSC_dX];
end
end

% Control constraints
if control_const==1
    CC=(1/UMAX2)*((Kv+K0v)*inv(P2)*(Kv+K0v)')-1;
    g=[g;CC];

    dCC_dP=[];
    for i=1:n

```

```

    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dCC_dP_ij=(1/UMAX^2)*((Kv+K0v)*(-inv(P2))*(EL'*P+P*EL)*inv(P2))*...
            (Kv+K0v)');
        dCC_dP=[dCC_dP; dCC_dP_ij];
    end
end
dCC_dKv=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Kv));
        EL(i,j)=1;
        dCC_dKv_ij=(1/UMAX^2)*((EL)*inv(P2)*(Kv+K0v)'+(Kv+K0v)*inv(P2)*(EL)');
        dCC_dKv=[dCC_dKv; dCC_dKv_ij];
    end
end
dCC_dX=[dCC_dP; dCC_dKv];
dg_dX=[dg_dX dCC_dX];
end

```

```

% ===== END OF FILE: REACH_DFG.M =====

```

## G.1.4 Cost Function and Constraints for Finding $\tilde{u}$

```
function [f,g,U]=recover_u_fg(x0,K0v,GAMMA,X,n,m);
% RECOVER_U_FG is the cost function and constraint routine for use with
% overlapping recoverable ellipsoids to find maximal control on restricted
% state-space.

% SYSTEM SIZE DEFINITION
[c]=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Kv]=x2pk(X,n,m);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=-x0'*(Kv+K0v)'*(Kv+K0v)*x0;

U=abs(-(Kv+K0v)*x0);

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Ellipsoid constraint
EC=x0'*P2*x0-1; % Write as negative definite form
g=[g;EC];
constraints=char(constraints,'Ellipsoid Constraint');

% State constraints
for i=1:c
    for j=1:2
        SC=(-1^j)*GAMMA(i,:)*x0-1; % Consider both plus/minus GAMMA
        g=[g;SC];
        constraints=char(constraints,['State Constraint ' num2str(i)]);
    end
end;

% ===== END OF FILE: RECOVER_U_FG.M =====
```

### G.1.5 Derivatives of Cost Function and Constraints for Finding $\tilde{u}$

```
function [df_dx0,dg_dx0]=recover_u_dfg(x0,K0v,GAMMA,X,n,m);
% RECOVER_U_DFG is the derivative cost function and constraint routine for
% use with overlapping recoverable ellipsoids to find maximal control
% on restricted state-space.

% SYSTEM SIZE DEFINITION
c=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Kv]=x2pk(X,n,m);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=-x0'*(Kv+K0v)'*(Kv+K0v)*x0;

df_dx0=-2*(Kv+K0v)'*(Kv+K0v)*x0;

% =====

% CONSTRAINTS (Must be of form <= 0)
dg_dx0=[]; % clear constraint derivative array

% Ellipsoid constraint
EC=x0'*P2*x0-1; % Write as negative definite form
dEC_dx0=2*P2*x0;
dg_dx0=[dg_dx0 dEC_dx0];

% State constraints
for i=1:c
    for j=1:2
        dSC_dx0=(-1^j)*GAMMA(i,:)'; % Consider both plus/minus GAMMA
        dg_dx0=[dg_dx0 dSC_dx0];
    end
end

% ===== END OF FILE: CONTROL_U_DFG.M =====
```

## G.1.6 Plotting Routine

```

function [U,P,Kv]=recover_p(X,A,B,K0v,GAMMA,plot_pts);
% RECOVER_P is the trajectory plotting routine for use with RECOVER

% Plotting options
N=max((floor(plot_pts/20)^2),5); % plot trajectory from every Nth data
                                % point on the semi-ellipse with a
                                % maximum of 20 pts

if nargin<4
    error('Not enough input arguments.')
end

% Message to screen
disp(' ')
disp('+++++')
disp('Plotting recovering trajectories')
disp('+++++')
disp(' ')

% Parameter Assignments (extract P,K's from X)
[n,m]=size(B);
[P,Kv]=x2pk(X,n,m);
P2=P'*P;

Kv=Kv+K0v; % Construct composite gain

if n==2 % 2-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2]=semiellipse(P2,GAMMA,[],plot_pts);

    % Compute corresponding control effort
    ue=Kv*[x1;x2];

    % Compute maximum control effort on boundary
    U=max(ue);

    % Plot boundary vs. control
    plot(x1,x2,'k');grid on;axis square;hold on

    % Compute reaching and recovering state-space descriptions
    syse=ss(A-B*Kv,B,eye(n),zeros(n,m));

    % Plot trajectories for every Nth point on the ellipsoid boundary
    for i=1:ceil(sqrt(N)):max(size(x1));
        [y,t,x]=initial(syse,[x1(i),x2(i)],[0:.01:10]);
        plot(x(:,1),x(:,2),'r')
    end

    xlabel('x1'),ylabel('x2'),zlabel('u')
elseif n==3 % 3-D ELLIPSOID
    % Compute ellipsoid boundary

```

```

[x1,x2,x3]=semiellipse(P2,GAMMA,[],plot_pts);

% Compute corresponding control effort
for i=1:size(x1,1)
    for j=1:size(x1,2)
        ue(i,j)=Kv*[x1(i,j);x2(i,j);x3(i,j)];
    end
end

% Compute maximum control effort on boundary
U=max(max(abs(ue)));

% Plot boundary vs. control
surf(x1,x2,x3,ue/U);grid on;axis square;hold on

% Compute reaching and recovering state-space descriptions
syse=ss(A-B*Kv,B,eye(n),zeros(n,m));

% Plot trajectories for every Nth point on the ellipsoid boundary
hold on
for i=1:N:size(x1,1);
    for j=1:N:size(x1,2)
        [y,t,x]=initial(syse,[x1(i,j),x2(i,j),x3(i,j)], [0:.01:10]);
        plot3(x(:,1),x(:,2),x(:,3),'k')
    end
end
end
drawnow
grid on
colormap('copper')
lighting phong
light('Position',[1 -1 5])
h=findobj('Type','surface');
set(h,'FaceLighting','phong',...
    'FaceColor','interp',...
    'EdgeColor',[.4 .4 .4],...
    'BackFaceLighting','reverselit',...
    'AmbientStrength',1,...
    'DiffuseStrength',1);
shading interp
xlabel('x1'),ylabel('x2'),zlabel('x3')

end

% ===== END OF FILE: RECOVER_P.M =====

```

## G.2 Function Files for Computing Reachable Semi-Ellipsoidal Set

### G.2.1 Optimization Routine

```
function [P,Ke,U,X,Fstop,Gstop]=reach(A,B,GAMMA,UMAX,Q0,P0,K0e);
%
% REACH finds the largest reachable set for a linear system with
% constrained states and inputs, where the subset is constructed from
% the intersection of the state constraints and the computed ellipsoid.
%
% Parameter definitions:
%
% [A,B]: state-space description of linear system
% GAMMA: state constraints of the form GAMMA(i,:)*x<=1
% UMAX: maximum allowable control (assumed symmetric)
% Q0: specifies decay rate of the Lyapunov function
% P0: initial guess for the ellipsoid matrix (x'Px<=1)
% K0e: initial guess for state feedback (must stabilize [A,B])
%
% P: the optimal ellipsoid
% Ke: the corresponding state feedback gain matrix
% U: the maximum control effort on the boundary of P
% X: the final parameter search vector (elements of P, Ke)
%
% Usage:
%
% [P,Ke,U,X]=reach(A,B,GAMMA,UMAX,Q0,P0,K0e) finds the largest
% reachable ellipsoid, P, and corresponding state-feedback
% matrix, Ke, for the system [A,B] and the constraints, GAMMA,
% UMAX. Definitions of Q0, P0, and K0e are optional. If not
% provided, Q0 is assumed to be zero and P0 is initialized using
% an LQR algorithm. If K0e is not specified, or if the given
% K0e does not stabilize [A,B], it is chosen (arbitrarily) using
% the LQR technique.
%
% Notice:
%
% This algorithm is based on the dissertation "Ellipsoidal and
% Semi-Ellipsoidal Controlled Invariant Sets for Constrained
% Linear Systems" by Brian O'Dell, Oklahoma State University, 1999.
%
% Start counter for run-time
tic

% DEFINE SIMULATION CONSTANTS
fig_handle=1; % Set the figure handle
ON=1; % Switch ON
PARTIAL=0.5; % Switch HALF-ON
OFF=0; % Switch OFF
beta=0.97; % Decrease factor for size of P; MUST BE LESS THAN 1.0
tol=1e-5; % Error tolerance for terminating search
```



```

gradient_tol=1e-2; % Minimum gradient for maximal control search
PET=2.5;          % 'Percent Error Tolerance' for terminating searches

% DEFINE GENERAL PARAMETERS
plotting=PARTIAL; % Turn plotting 'on/off'; PARTIAL plots only final result
plot_pts=150;     % Number of points to use in plotting
grad_check=OFF;   % Checks analytical gradients against numerical estimates
search_output=ON; % Displays intermediate search results

% DEFINE MATRIX DIMENTIONING CONSTANTS
[n,m]=size(B);   % 'n' is number of states, 'm' is number of inputs
c=size(GAMMA,1); % Defines the number of state constraints
pe=(n^2-n)/2+n;  % Defines number of ellipse parameters
pg=n*m;          % Defines number of state feedback gain parameters
p=pe+pg;         % Defines the total number of parameters.

% DEFINE SEARCH PARAMETERS (for search OPTIONS, type 'help foptions')
max_passes=30;   % Maximum number of search cycles
min_iters=30*p;  % Minimum number of search iterations per cycle
max_iters=60*p;  % Maximum number of search iterations per cycle
weight=1e0;      % Weighting coefficient for constraint vector
pd_weight=1e0;   % Additional (multiplicative) weight for pos. def. const.
inv_weight=1e0;  % Additional (multiplicative) weight for invariance const.
options(1)=ON;   % Displays intermediate search results
options(14)=max_iters; % Set the maximum number of iterations per pass

% INITIALIZE SEARCH DATA VECTORS
Fstart=[];       % Vector of cost function values before each pass
Gstart=[];       % Matrix of constraint function values before each pass
Fstop=[];        % Vector of cost function values after each pass
Gstop=[];        % Matrix of constraint function values after each pass
search_log=[];   % Matrix of parameter values after each pass

% CHECK VALIDITY OF INPUT DEFINITIONS
N=nargin;
if N<4
    error('Not enough input arguments.')
```

```

    elseif N==4
        Q0=[];P0=[];K0e=[];
    elseif N==5
        P0=[];K0e=[];
    elseif N==6
        K0e=[];
    end

% CHECK SIZE OF A,B
if size(A,1)~=size(A,2)
    error('A is non-square.')
```

```

    elseif size(A,1)~=n
        error('A and B must have same number of rows.')
```

```

    end

```

```

% CHECK CONSTRAINT SPACE SIZE
if size(GAMMA,2)~=n
    error('GAMMA defined with different number of states than A.')
end

% CHECK UMAX
if min(UMAX)<=0
    error('Control constraints must be positive.')
end

% CHECK Q0
if isempty(Q0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing Q0 matrix:')
    disp(' ')
    Q0=zeros(size(A))
elseif max(max(abs(Q0-Q0')))>0 % Check for symmetry
    error('Q0 is not symmetric.')
elseif min(eig(Q0))<0 % Check for positive definiteness
    error('Q0 is not positive definite.')
end

% USE LQR SOLUTION FOR ANY NECESSARY INITIALIZATIONS
poles=[1:n]; % Define (arbitrary) positive pole locations
k0=place(A,B,poles); % Compute state-feedback gain
R=eye(m); % Arbitrary pos. def. weighting matrix for LQR
[temp,p20]=lqr(A-B*k0,B,Q0,R); % Compute stabilizing controller and ellipse
dk=inv(R)*B'*p20; % Define stabilizing controller

% CHECK KOe
if isempty(KOe) % Initialize with LQR if KOe not given
    % Display initialization message
    disp(' ')
    disp('Initializing KOe matrix:')
    disp(' ')
    KOe=k0+dk/2 % Define marginally stabilizing controller
    Ke=-dk/4 % Define gain such that (KOe+Ke) is stabilizing
    % (in negative time)
else
    if (size(KOe,1)~=m)|(size(KOe,2)~=n) % Check size of KOe
        error('KOe must be same size as B transpose.')
    end
    if max(real(eig(A-B*KOe)))>0 % Check stability of A-B*KOe
        error('KOe does not stabilize A.')
    end
end
end

% CHECK PO

```

```

if isempty(P0) % Check for proper initialization
% Display initialization message
disp(' ')
disp('Initializing P0 matrix:')
disp(' ')
P0=real(sqrtm(p20)) % Compute P0 from p20
P20=P0'*P0
elseif max(max(abs(P0-P0')))>0 % Check for symmetry
error('P0 is not symmetric.')
elseif min(eig(P0))<0 % Check for positive definiteness
error('P0 is not positive definite.')
end

% ASSIGN P0,K0e TO ELEMENTS OF SEARCH SPACE VECTOR
X=pk2x(P0,Ke,n,m); % Parametrize search with P0, the square root of the
% ellipsoid matrix, to minimize search errors with
% positive definiteness of ellipse.

if plotting==ON
% Plot initial condition ellipsoid
figure(fig_handle);clf;drawnow;
ellipse(P20,[],plot_pts);grid on;axis square;
title('INITIAL CONDITION')
drawnow;

% Plot trajectories of initial condition ellipsoid
figure(fig_handle+1);clf;drawnow;
reach_p(X,A,B,K0e);
title('TRAJECTORIES OF INITIAL CONDITION')
drawnow;
end

pvol=det(inv(P20));

save reach_best.mat X

% =====

% OPTIMIZE SOLUTION

U=0; % Initialize maximum observed control to zero
UMAX_hat=UMAX; % Initialize pseudo-maximum control to true maximum
control_iters=0; % Initialize counter for control iteration passes

while (U<beta*UMAX)|(U>(UMAX+tol)) % Loop while observed control outside tol.

disp(' ')
disp(sprintf(' ===== SEARCHING: UMAX_hat = %6.2f ===== ',...
UMAX_hat))
disp(' ')

```

```

options(14)=max_iters; % Set the maximum number of iterations to max_iters
pass=1;                % Reset the counter
control_const=ON;      % Turn the control constraint 'on'

change=100;            % Initialize change in ellipsoid size to 100%
options(10)=max_iters; % Initialize number of passes to maximum allowable

while (pass<=max_passes)&((abs(change)>PET)|(options(10)>(beta*options(14))))
    % Loop until change in ellipsoid size < 2%

    disp(' ')
    disp(sprintf(' ===== OPTIMIZING: CHANGE = %6.2f %% ===== ',...
        change))
    disp(' ')

    % Log start-of-pass search constraint vector and cost function
    [fstart,g,U,constraints]=reach_fg(X,A,B,KOe,GAMMA,UMAX_hat,Q0,...
        control_const,weight,pd_weight,inv_weight);
    Gstart=[Gstart g];
    Fstart=[Fstart fstart];
    Pstart=x2pk(X,n,m);
    P2start=Pstart'*Pstart;

    % Scale ellipse down by factor of beta to start the next pass off the
    % constraints
    X(1:pe,1)=X(1:pe,1)/sqrt(beta);

    % Perform search.
    options(9)=OFF;
    [X,options]=constr('reach_fg',X,options,[],[],'reach_dfg',...
        A,B,KOe,GAMMA,UMAX_hat,Q0,...
        control_const,weight,pd_weight,inv_weight);

    % Log end-of-pass search constraint vector and cost function
    [fstop,g,U,constraints]=reach_fg(X,A,B,KOe,GAMMA,UMAX_hat,Q0,...
        control_const,weight,pd_weight,inv_weight);
    Gstop=[Gstop g];
    Fstop=[Fstop fstop];

    % Parameter Assignments
    Pstop=x2pk(X,n,m);
    P2stop=Pstop'*Pstop;

    % Compute change in cost function
    change=100*(trace(P2start)-trace(P2stop))/trace(P2start);

    % Log parameter space
    search_log=[search_log X];

    % Parameter Assignments
    [P,Ke]=x2pk(X,n,m);
    P2=P'*P;

    % Plot ellipsoid in state space

```

```

if plotting==ON
    figure(fig_handle+2);hold off;clf;drawnow;
    ellipse(P2, [],plot_pts);grid on;axis square;hold on
    title(sprintf('OPTIMIZING: CHANGE = %1.4g',change));
    drawnow
end

save reach.mat X U

% Check for improvement
if det(inv(P2))>pvol
    pvol=det(inv(P2));
    save reach_best.mat X P2 Ke pvol
end

pass=pass+1;

end

% DISPLAY TERMINATION CRITERIA

disp(' ')
disp(sprintf('Search terminated on change in cost function of %0.4g %%.',...
    change))
disp(' ')

% SEARCH FOR MAXIMUM OBSERVED CONTROL ON RESTRICTED STATE-SPACE
disp(' ')
disp(sprintf(' ===== SEARCHING FOR MAXIMUM OBSERVED CONTROL ===== '))
disp(' ')
if control_iters==0
    % Find point which maximizes control on restricted state-space
    umax_options(1)=-1; % Don't display intermediate results

    x0=rand(n,1); % Generate random initial condition for search
    x0=constr('reach_u_fg',x0,umax_options,[],[],'reach_u_dfg',K0e,...
        GAMMA,X,n,m);

    % Compute maximum control effort
    [fumax,gumax,U]=reach_u_fg(x0,K0e,GAMMA,X,n,m);

    % Display maximal control on restricted region
    U

    % Store previous search results
    U_old=U;
    UMAX_hat_old=UMAX_hat;

    % Compute a new estimate for the UMAX_hat that will achieve U=UMAX.
    UMAX_hat=UMAX_hat*(beta^(sign(U-UMAX)));

    if (U>=beta*UMAX)&(U<=(UMAX+tol))
        %
        disp(' ')
    end
end

```

```

        disp(cat(2,'Search terminated: Maximum observed control ',...
            'within tolerance of UMAX.'))
        disp(' ')
    end
else
    % Use previous search result for initial condition
    x0=constr('reach_u_fg',x0,umax_options,[],[],'reach_u_dfg',...
        KOe,GAMMA,X,n,m);

    % Compute maximum control effort
    [fumax,gumax,U]=reach_u_fg(x0,KOe,GAMMA,X,n,m);

    % Display maximal control on restricted region
    U

    % Estimate new value of UMAX_hat using secant search
    gradient=(U_old-U)/(UMAX_hat_old-UMAX_hat);
    UMAX_hat_tmp=UMAX_hat-(U-UMAX)/gradient;

    % Store previous search results
    U_old=U;
    UMAX_hat_old=UMAX_hat;

    UMAX_hat=UMAX_hat_tmp;

    % Check to see if improvement feasible
    if (U>=beta*UMAX)&(U<=(UMAX+tol))
        %
        disp(' ')
        disp(['Search terminated: Maximum observed control ',...
            'within tolerance of UMAX.'])
        disp(' ')
    elseif gradient<gradient_tol
        disp(' ')
        disp(['Search terminated: No increase expected in maximum ',...
            'observed control.'])
        disp(' ')

        % Redefine U to artificially terminate loop
        U=UMAX;
    end

end

end

% Advance counter
control_iters=control_iters+1;

end

% =====

% CLOSING TASKS

```

```

% Plot final result
if plotting>=PARTIAL

    disp(' ')
    disp('+++++')
    disp('Computing plot points.')
    disp('+++++')
    disp(' ')

    figure(fig_handle);clf;drawnow;
    if n==2
        reach_p(X,A,B,KOe,GAMMA,plot_pts);
    elseif n==3
        subplot(121)
        semiellipse(P2,GAMMA,[],plot_pts),hold on
        subplot(122)
        reach_p(X,A,B,KOe,GAMMA,plot_pts);
    end
end

% Compute composite gain.
Ke=Ke+KOe;

% Define output ellipsoid matrix
P=P2;

% Compute maximum gain on ellipse, U
[f,g,U,constraints]=reach_fg(X,A,B,KOe,GAMMA,UMAX_hat,Q0,...
    control_const,weight,pd_weight,inv_weight);
Gstop=[Gstop g];g,constraints
Fstop=[Fstop f];

% Clean up hard drive
delete reach.mat
delete reach_best.mat

% Terminate counter and display elapsed time
toc

% ===== END OF FILE: REACH.M =====

```

## G.2.2 Cost Function and Constraints

```

function [f,g,Ue,constraints]=reach_fg(X,A,B,K0e,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight,inv_weight);
% REACH_FG is the cost function and constraint routine for use with REACH.

% parametrizes in terms of sqrt(P) and Ke

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
[p]=max(size(X));
[c]=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Ke]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(trace(P2));

% CONSTRAINTS (Must be of form <= 0)
g=[];          % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Invariance constraint
Acl=A-B*(Ke+K0e);
IC=P2*(-Acl)+(-Acl)'+P2+Q; % Write as negative definite form
[v,d]=eig(IC);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D*inv_weight];      % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

% Positive definiteness constraints.
[v,d]=eig(-P2*pd_weight); % Write as negative definite form
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Positive Definiteness');
end

% State constraints (See dissertation for details)
for i=1:c;
    Gamma=GAMMA(i,:);
    w=null(Gamma*(-(A-B*(Ke+K0e))));

    g=[g; (Gamma*w)*inv(w'*P2*w)*(Gamma*w) '-1];

    constraints=char(constraints,['State Constraint ' num2str(i)]);
end;

```



```
% Control constraints
if control_const==1
    Ue=sqrt((Ke+K0e)*inv(P2)*(Ke+K0e)');
    g=[g; (Ue*Ue)/(UMAX*UMAX)-1]; % Normalize to 1
    constraints=char(constraints,'Control');
end
```

```
% ===== END OF FILE: REACH_FG.M =====
```

### G.2.3 Derivatives of Cost Function and Constraints

```

function [df_dX,dg_dX]=reach_dfg(X,A,B,KOe,GAMMA,UMAX,Q,...
    control_const,weight,pd_weight,inv_weight);
% REACH_DFG is the derivative cost function and constraint routine
%   for use with REACH.

% parametrizes in terms of sqrt(P) and Ke

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
p=max(size(X));
c=size(GAMMA,1);
pg=n*m; % number of state feedback gain parameters
pe=p-pg; % number of ellipsoid parameters

% PARAMETER ASSIGNMENTS
[P,Ke]=x2pk(X,n,m);
P2=P'*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(trace(P2));

df_dP=[];
xindex=0;
for i=1:n
    for j=i:n
        if i==j
            df_dP=[df_dP;2*P(i,j)/trace(P2)];
        else
            df_dP=[df_dP;4*P(i,j)/trace(P2)];
        end
    end
end
df_dKe=zeros(size(Ke))';
df_dX=[df_dP;df_dKe];

% =====

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear constraint array
dg_dX=[]; % clear constraint derivative array

% Invariance constraint
Acl=A-B*(Ke+KOe);
IC=P2*(-Acl)+(-Acl)'+P2+Q;
[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
Vi=v(:,sort_index);

```

```

g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*(-Acl)+(-Acl)'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKe=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Ke));
        EL(i,j)=1;
        dAcl_dKe_ij=-B*EL;
        dIC_dKe_ij=P2*(-dAcl_dKe_ij)+(-dAcl_dKe_ij)'*P2;
        dDi_dKe=[dDi_dKe; diag(Vi'*(dIC_dKe_ij)*Vi)'];
    end
end

dDi_dX=[dDi_dP; dDi_dKe]*inv_weight;
dg_dX=[dg_dX dDi_dX];

% Positive definiteness of P constraints.
PDC=-P2*pd_weight;
[v,d]=eig(PDC);
[Dpd,sort_index]=esort(diag(d));
Vpd=v(:,sort_index);
g=[g;Dpd];      % Write as 'less than' constraint

dDpd_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P));
        EL(i,j)=1;EL(j,i)=1;
        dPDC_dP_ij=-1*(EL'*P+P*EL)*pd_weight;
        dDpd_dP=[dDpd_dP; diag(Vpd'*(dPDC_dP_ij)*Vpd)'];
    end
end
dDpd_dKe=zeros(pg,n); % 'n' eigenvalues

dDpd_dX=[dDpd_dP;dDpd_dKe];
dg_dX=[dg_dX dDpd_dX];

% State constraints
for k=1:c;
    Gamma=GAMMA(k,:);
    w=null(Gamma*(-(A-B*(Ke+K0e))));

    SC=(Gamma*w)*inv(w'*P2*w)*(Gamma*w)'-1;
end

```

```

g=[g;SC];

dSC_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;

        dinv_dP_ij=-inv(w'*P2*w)*(w'*(P*EL+EL*P)*w)*inv(w'*P2*w);
        dSC_dP_ij=(Gamma*w)*dinv_dP_ij*(Gamma*w)';

        dSC_dP=[dSC_dP; dSC_dP_ij];
    end
end
dSC_dKe=[];
for i=1:m
    for j=1:n
        % Use numerical approximation
        EL=zeros(size(Ke));
        EL(i,j)=1;
        deltaKe_ij=EL*1e-5;
        deltaw=null(Gamma*(-(A-B*(Ke+K0e+deltaKe_ij))))-w;
        gradw=deltaw/1e-5;
        dinv_dKe_ij=-inv(w'*P2*w)*(gradw'*P2*w+w'*P2*gradw)*inv(w'*P2*w);
        dSC_dKe_ij=(Gamma*gradw)*inv(w'*P2*w)*(Gamma*w)'+...
            (Gamma*w)*dinv_dKe_ij*(Gamma*w)'+...
            (Gamma*w)*inv(w'*P2*w)*(Gamma*gradw)';
        dSC_dKe=[dSC_dKe; dSC_dKe_ij];
    end
end
dSC_dX=[dSC_dP;dSC_dKe];
dg_dX=[dg_dX dSC_dX];
end

% Control constraints
if control_const==1
    CC=(1/UMAX^2)*((Ke+K0e)*inv(P2)*(Ke+K0e)')-1;
    g=[g;CC];

    dCC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P2));
            EL(i,j)=1;EL(j,i)=1;
            dCC_dP_ij=(1/UMAX^2)*((Ke+K0e)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*...
                (Ke+K0e)');
            dCC_dP=[dCC_dP; dCC_dP_ij];
        end
    end
    dCC_dKe=[];
    for i=1:m
        for j=1:n
            EL=zeros(size(Ke));
            EL(i,j)=1;

```

```
        dCC_dKe_ij=(1/UMAX^2)*((EL)*inv(P2)*(Ke+K0e)'+(Ke+K0e)*inv(P2)*(EL)');
        dCC_dKe=[dCC_dKe; dCC_dKe_ij];
    end
end
dCC_dX=[dCC_dP; dCC_dKe];
dg_dX=[dg_dX dCC_dX];
end
```

```
% ===== END OF FILE: REACH_DFG.M =====
```

## G.2.4 Cost Function and Constraints for Finding $\tilde{u}$

```
function [f,g,U]=reach_u_fg(x0,K0e,GAMMA,X,n,m);
% REACH_U_FG is the cost function and constraint routine for use with
% overlapping reachable ellipsoids to find maximal control on restricted
% state-space.

% SYSTEM SIZE DEFINITION
[c]=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Ke]=x2pk(X,n,m);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=-x0'*(Ke+K0e)'*(Ke+K0e)*x0;

U=abs(-(Ke+K0e)*x0);

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Ellipsoid constraint
EC=x0'*P2*x0-1; % Write as negative definite form
g=[g;EC];
constraints=char(constraints,'Ellipsoid Constraint');

% State constraints
for i=1:c
    for j=1:2
        SC=(-1^j)*GAMMA(i,:)*x0-1; % Consider both plus/minus GAMMA
        g=[g;SC];
        constraints=char(constraints,['State Constraint ' num2str(i)]);
    end
end;

% ===== END OF FILE: REACH_U_FG.M =====
```

## G.2.5 Derivatives of Cost Function and Constraints for Finding $\tilde{u}$

```

function [df_dx0,dg_dx0]=reach_u_dfg(x0,K0e,GAMMA,X,n,m);
% REACH_U_DFG is the derivative cost function and constraint routine for
% use with overlapping reachable ellipsoids to find maximal control
% on restricted state-space.

% SYSTEM SIZE DEFINITION
c=size(GAMMA,1);

% PARAMETER ASSIGNMENTS
[P,Ke]=x2pk(X,n,m);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=-x0'*(Ke+K0e)'*(Ke+K0e)*x0;

df_dx0=-2*(Ke+K0e)'*(Ke+K0e)*x0;

% =====

% CONSTRAINTS (Must be of form <= 0)
dg_dx0=[]; % clear constraint derivative array

% Ellipsoid constraint
EC=x0'*P2*x0-1; % Write as negative definite form
dEC_dx0=2*P2*x0;
dg_dx0=[dg_dx0 dEC_dx0];

% State constraints
for i=1:c
    for j=1:2
        dSC_dx0=(-1^j)*GAMMA(i,:)'; % Consider both plus/minus GAMMA
        dg_dx0=[dg_dx0 dSC_dx0];
    end
end

% ===== END OF FILE: REACH_U_DFG.M =====

```

## G.2.6 Plotting Routine

```
function [U,P,Ke]=reach_p(X,A,B,K0e,GAMMA,plot_pts);
% REACH_P is the trajectory plotting routine for use with REACH

% Plotting options
N=max(floor((plot_pts/20)^2),5); % plot trajectory from every Nth data
                                % point on the semi-ellipse with a
                                % maximum of 20 pts

if nargin<4
    error('Not enough input arguments.')
end

% Message to screen
disp(' ')
disp('+++++')
disp('Plotting reaching trajectories')
disp('+++++')
disp(' ')

% Parameter Assignments (extract P,K's from X)
[n,m]=size(B);
[P,Ke]=x2pk(X,n,m);
P2=P'*P;

Ke=Ke+K0e; % Construct composite gain

if n==2 % 2-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2]=semiellipse(P2,GAMMA,[],plot_pts);

    % Compute corresponding control effort
    ue=Ke*[x1;x2];

    % Compute maximum control effort on boundary
    U=max(ue);

    % Plot boundary vs. control
    plot(x1,x2,'k');grid on;axis square;hold on

    % Compute reaching state-space description (neg.time)
    syse=ss(-(A-B*Ke),-B,eye(n),zeros(n,m));

    % Plot trajectories for every Nth point on the ellipsoid boundary
    for i=1:ceil(sqrt(N)):max(size(x1));
        [y,t,x]=initial(syse,[x1(i),x2(i)],[0:.01:10]);
        plot(x(:,1),x(:,2),'r')
    end

    colormap('gray')

    xlabel('x1'),ylabel('x2'),zlabel('u')
```



```

elseif n==3 % 3-D ELLIPSOID
    % Compute ellipsoid boundary
    [x1,x2,x3]=semiellipse(P2,GAMMA,[],plot_pts);

    % Compute corresponding control effort
    for i=1:size(x1,1)
        for j=1:size(x1,2)
            ue(i,j)=Ke*[x1(i,j);x2(i,j);x3(i,j)];
        end
    end

    % Compute maximum control effort on boundary
    U=max(max(abs(ue)));

    % Plot boundary vs. control
    surf(x1,x2,x3,ue/U);grid on;axis square;hold on

    % Compute reaching and recovering state-space descriptions
    syse=ss(-(A-B*Ke),-B,eye(n),zeros(n,m));

    % Plot trajectories for every Nth point on the ellipsoid boundary
    hold on
    for i=1:N:size(x1,1);
        for j=1:N:size(x1,2)
            [y,t,x]=initial(syse,[x1(i,j),x2(i,j),x3(i,j)], [0:.01:10]);
            plot3(x(:,1),x(:,2),x(:,3),'r')
        end
    end
    end
drawnow
grid on
colormap('copper')
lighting phong
light('Position',[1 -1 5])
h=findobj('Type','surface');
set(h,'FaceLighting','phong',...
    'FaceColor','interp',...
    'EdgeColor',[.4 .4 .4],...
    'BackFaceLighting','reverselit',...
    'AmbientStrength',1,...
    'DiffuseStrength',1);
shading interp
xlabel('x1'),ylabel('x2'),zlabel('x3')

end

% ===== END OF FILE: REACH_P.M =====

```

## G.3 Function Files for Computing Controllable Semi-Ellipsoidal Set

### G.3.1 Optimization Routine

```
function [P,Kv,Ke,U,X]=control(A,B,GAMMA,UMAX,Q0,PO,KOv,KOe);
%
% CONTROL finds the largest controllable subset for a linear system with
% constrained states and inputs, where the subset is constructed from
% the intersection of the state constraints and the computed ellipsoid.
%
% Parameter definitions:
%
% [A,B]: state-space description of linear system
% GAMMA: state constraints of the form GAMMA(i,:)*x<=1
% UMAX: maximum allowable control (assumed symmetric)
% Q0: specifies decay rate of the Lyapunov function
% PO: initial guess for the ellipsoid matrix (x'Px<=1)
% KOv: initial guess for recovering state feedback matrix
% KOe: initial guess for reaching state feedback matrix
%
% P: the optimal ellipsoid
% Kv: the corresponding recovering state feedback matrix
% Ke: the corresponding reaching state feedback matrix
% U: the maximum control effort (via state feedback) on subset
% X: the final parameter search vector (elements of P, K)
%
% Usage:
%
% [P,Kv,Ke,U,X]=control(A,B,GAMMA,UMAX,Q0,PO,KOv,KOe) finds the
% largest ellipsoid, P, which is both reachable and recoverable,
% and corresponding state-feedback matrices, Ke,Kv, for the
% system [A,B] and the constraints, GAMMA, UMAX. Definitions of
% Q0, PO, and KO's are optional. If not provided, Q0 is assumed
% to be zero and PO is initialized using an LQR algorithm. If
% KOe is not specified, or if the given KOe does not stabilize
% [-A,-B], it is chosen (arbitrarily) using an LQR technique.
% Similarly, if KOv is not specified, or if the given KOv does
% not stabilize [A,B], it is chosen (arbitrarily) using an LQR
% technique.
%
% Notice:
%
% This algorithm is based on the dissertation "Ellipsoidal and
% Semi-Ellipsoidal Controlled Invariant Subsets for Constrained
% Linear Systems" by Brian O'Dell, Oklahoma State University, 1999.
%
% Start counter for run-time
tic

% DEFINE SIMULATION CONSTANTS
fig_handle=1; % Set the figure handle
ON=1; % Switch ON
```

```

PARTIAL=0.5;      % Switch HALF-ON
OFF=0;           % Switch OFF
beta=0.95;       % Decrease factor for size of P; MUST BE LESS THAN 1.0
tol=1e-5;        % Error tolerance for terminating search
gradient_tol=1e-5; % Minimum gradient for maximal control search
PET=1;           % 'Percent Error Tolerance' for terminating searches
flag=1;          % Defines as controllable ellipsoid

% DEFINE GENERAL PARAMETERS
plotting=PARTIAL; % Turn plotting 'on/off'; PARTIAL plots only final result
plot_pts=150;     % Number of points to use in plotting
grad_check=OFF;   % Checks analytical gradients against numerical estimates
search_output=ON; % Displays intermediate search results

% DEFINE MATRIX DIMENTIONING CONSTANTS
[n,m]=size(B);   % 'n' is number of states, 'm' is number of inputs
c=size(GAMMA,1); % Defines the number of state constraints
pe=(n^2-n)/2+n;  % Defines number of ellipse parameters
pg=2*n*m;        % Defines number of state feedback gain parameters
p=pe+pg;         % Defines the total number of parameters.

% DEFINE SEARCH PARAMETERS (for search OPTIONS, type 'help foptions')
max_passes=20;   % Maximum number of search cycles
min_iters=30*p;  % Minimum number of search iterations per cycle
max_iters=60*p;  % Maximum number of search iterations per cycle
weight=1e0;      % Weighting coefficient for constraint vector
pd_weight=1e0;   % Additional (multiplicative) weight for pos. def. const.
inv_weight=1e0;  % Additional (multiplicative) weight for invariance const.
options(1)=ON;   % Displays intermediate search results
options(14)=max_iters; % Set the maximum number of iterations per pass

% INITIALIZE SEARCH DATA VECTORS
Fstart=[];       % Vector of cost function values before each pass
Gstart=[];       % Matrix of constraint function values before each pass
Fstop=[];        % Vector of cost function values after each pass
Gstop=[];        % Matrix of constraint function values after each pass
search_log=[];   % Matrix of parameter values after each pass

% CHECK VALIDITY OF INPUT DEFINITIONS
N=nargin;
if N<4
    error('Not enough input arguments.')
elseif N==4
    Q0=[];P0=[];K0e=[];K0v=[];
elseif N==5
    P0=[];K0e=[];K0v=[];
elseif N==6
    K0e=[];K0v=[];
elseif N==7
    K0v=[];
end

```

```

% CHECK SIZE OF A,B
if size(A,1)~=size(A,2)
    error('A is non-square.')
elseif size(A,1)~=n
    error('A and B must have same number of rows.')
end

% CHECK CONSTRAINT SPACE SIZE
if size(GAMMA,2)~=n
    error('GAMMA defined with different number of states than A.')
end

% CHECK UMAX
if min(UMAX)<=0
    error('Control constraints must be positive.')
end

% CHECK Q0
if isempty(Q0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing Q0 matrix:')
    disp(' ')
    Q0=zeros(size(A))
elseif max(max(abs(Q0-Q0')))>0 % Check for symmetry
    error('Q0 is not symmetric.')
elseif min(eig(Q0))<0 % Check for positive definiteness
    error('Q0 is not positive definite.')
end

% USE LQR SOLUTION FOR ANY NECESSARY INITIALIZATIONS
poles=[1:n]; % Define (arbitrary) positive pole locations
k0=place(A,B,poles); % Compute state-feedback gain
R=eye(m); % Arbitrary pos. def. weighting matrix for LQR
[temp,p20]=lqr(A-B*k0,B,Q0,R); % Compute stabilizing controller and ellipse
dk=inv(R)*B'*p20; % Define stabilizing controller

% CHECK K0v
if isempty(K0v) % Initialize with LQR if K0v not given
    % Display initialization message
    disp(' ')
    disp('Initializing K0v matrix:')
    disp(' ')
    K0v=k0+dk/2 % Define marginally stabilizing controller
    Kv=dk/4 % Define gain such that (K0v+Kv) is stabilizing
    % (in positive time)
else
    if (size(K0v,1)~=m)|(size(K0v,2)~=n) % Check size of K0v
        error('K0v must be same size as B transpose.')
    end
end

```

```

end
if max(real(eig(A-B*K0v)))>0 % Check stability of A-B*K0v
    error('K0v does not stabilize A.')
end
end

% CHECK K0e
if isempty(K0e) % Initialize with LQR if K0v not given
    % Display initialization message
    disp(' ')
    disp('Initializing K0e matrix:')
    disp(' ')
    K0e=k0+dk/2 % Define marginally stabilizing controller
    Ke=-dk/4 % Define gain such that (K0e+Ke) is stabilizing
    % (in negative time)
else
    if (size(K0e,1)~=m)|(size(K0e,2)~=n) % Check size of K0e
        error('K0e must be same size as B transpose.')
    end
    if max(real(eig((-A)-(-B)*K0e)))>0 % Check stability of A-B*K0e
        error('K0e does not stabilize A.')
    end
end
end

% CHECK P0
if isempty(P0) % Check for proper initialization
    % Display initialization message
    disp(' ')
    disp('Initializing P0 matrix:')
    disp(' ')
    P0=real(sqrtm(p20)) % Compute P0 from p20
    P20=P0'*P0
elseif max(max(abs(P0-P0')))>0 % Check for symmetry
    error('P0 is not symmetric.')
elseif min(eig(P0))<0 % Check for positive definiteness
    error('P0 is not positive definite.')
end
end

% ASSIGN INITIAL CONDITIONS TO ELEMENTS OF SEARCH SPACE VECTOR
X=pk2x(P0,Kv,Ke,n,m); % Parametrize search with P0, the square root of the
% ellipsoid matrix, to minimize search errors with
% positive definiteness of ellipse.

if plotting==ON
    % Plot initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    ellipse(P20,[],plot_pts);grid on;axis square;
    title('INITIAL CONDITION')
    drawnow;
    fig_handle=fig_handle+1;
end

```

```

    % Plot trajectories of initial condition ellipsoid
    figure(fig_handle);clf;drawnow;
    control_p(X,A,B,KOv,KOe);
    title('TRAJECTORIES OF INITIAL CONDITION')
    drawnow;
    fig_handle=fig_handle+1;
end

pvol=det(inv(P20));

save control_best.mat X pvol

% =====

% OPTIMIZE SOLUTION

U=0; % Initialize maximum observed control to zero
UMAX_hat=UMAX; % Initialize pseudo-maximum control to true maximum
control_iters=0; % Initialize counter for control iteration passes

while (U<beta*UMAX)|(U>(UMAX+tol)) % Loop while observed control outside tol.

    disp(' ')
    disp(sprintf(' ===== SEARCHING: UMAX_hat = %6.2f ===== ',...
        UMAX_hat))
    disp(' ')

    options(14)=max_iters; % Set the maximum number of iterations
    pass=1; % Reset the counter
    control_const=ON; % Turn the control constraint 'on'

    change=100; % Initialize change in ellipsoid size to 100%
    options(10)=max_iters; % Initialize number of passes to maximum allowable

    while (pass<=max_passes)&((abs(change)>PET)|(options(10)>(0.8*options(14))))
        % Loop until change in ellipsoid size < 2%

        disp(' ')
        disp(sprintf(' ===== OPTIMIZING: CHANGE = %6.2f %% ===== ',...
            change))
        disp(' ')

        % Log start-of-pass search constraint vector and cost function
        [fstart,g,U,constraints]=control_fg(X,A,B,KOv,KOe,GAMMA,UMAX,QO,...
            control_const,weight,pd_weight,inv_weight);
        Gstart=[Gstart g];
        Fstart=[Fstart fstart];
        Pstart=x2pk(X,n,m);
        P2start=Pstart'*Pstart;

        % Scale ellipse down by factor of beta to start the next pass off the
        % constraints

```

```

X(1:pe,1)=X(1:pe,1)/sqrt(beta); % Since search parameters define P, not P2

% Perform search.
options(9)=OFF;
[X,options]=constr('control_fg',X,options,[],[],'control_dfg',...
    A,B,KOv,KOe,GAMMA,UMAX,QO,...
    control_const,weight,pd_weight,inv_weight);

% Log end-of-pass search constraint vector and cost function
[fstop,g,U,constraints]=control_fg(X,A,B,KOv,KOe,GAMMA,UMAX,...
    QO,control_const,weight,pd_weight,inv_weight);
Gstop=[Gstop g];
Fstop=[Fstop fstop];

% Parameter Assignments
Pstop=x2pk(X,n,m);
P2stop=Pstop'*Pstop;

% Compute change in cost function
change=100*(trace(P2start)-trace(P2stop))/trace(P2start);

% Log parameter space,control value, and number of iterations
search_log=[search_log X];

% Parameter Assignments
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% Plot ellipsoid in state space
if plotting==ON
    figure(fig_handle);clf;drawnow;
    ellipse(P2,[],plot_pts);grid on;axis square;
    title(sprintf('OPTIMIZING: CHANGE = %1.4g',change));
    drawnow
end

save control.mat X U

% Check for improvement
if det(inv(P2))>pvol
    pvol=det(inv(P2));
    save control_best.mat X P2 Kv pvol
end

pass=pass+1;

end

% DISPLAY TERMINATION CRITERIA

disp(' ')
disp(sprintf('Search terminated on change in cost function of %0.4g %%.',...
    change))
disp(' ')

```

```

% SEARCH FOR MAXIMUM OBSERVED CONTROL ON RESTRICTED STATE-SPACE
disp(' ')
disp(sprintf(' ===== SEARCHING FOR MAXIMUM OBSERVED CONTROL ===== '))
disp(' ')
if control_iters==0
    % Find point which maximizes control on restricted state-space
    umax_options(1)=-1; % Don't display intermediate results

    x0=rand(n,1); % Generate random initial condition for search
    x0=constr('control_u_fg',x0,umax_options,[],[],'control_u_dfg',K0v,K0e,...
        GAMMA,X,n,m);

    % Compute maximum control effort
    [fumax,gumax,U]=control_u_fg(x0,K0v,K0e,GAMMA,X,n,m);

    % Display maximal control on restricted region
    U

    % Store previous search results
    U_old=U;
    UMAX_hat_old=UMAX_hat;

    % Compute a new estimate for the UMAX_hat that will achieve U=UMAX.
    UMAX_hat=UMAX_hat*(beta^(sign(U-UMAX)));

    if (U>=beta*UMAX)&(U<=(UMAX+tol))
        %
        disp(' ')
        disp(cat(2,'Search terminated: Maximum observed control ',...
            'within tolerance of UMAX.'))
        disp(' ')
    end
else
    % Use previous search result for initial condition
    x0=constr('control_u_fg',x0,umax_options,[],[],'control_u_dfg',...
        K0v,K0e,GAMMA,X,n,m);

    % Compute maximum control effort
    [fumax,gumax,U]=control_u_fg(x0,K0v,K0e,GAMMA,X,n,m);

    % Display maximal control on restricted region
    U

    % Estimate new value of UMAX_hat using secant search
    gradient=((U_old-UMAX)-(U-UMAX))/(UMAX_hat_old-UMAX_hat);
    UMAX_hat_tmp=UMAX_hat-(U-UMAX)/gradient;

    % Store previous search results
    U_old=U;
    UMAX_hat_old=UMAX_hat;

    UMAX_hat=UMAX_hat_tmp;

    % Check to see if improvement feasible

```



```

if (U>=beta*UMAX)&(U<=(UMAX+tol))
%
disp(' ')
disp(cat(2,'Search terminated: Maximum observed control ',...
'within tolerance of UMAX.'))
disp(' ')
elseif abs(gradient)<gradient_tol
disp(' ')
disp(cat(2,'Search terminated: No increase expected in maximum ',...
'observed control.'))
disp(' ')

% Redefine U to artificially terminate loop
U=UMAX;
end

end

% Advance counter
control_iters=control_iters+1;

end

% =====

% CLOSING TASKS

% Plot trajectories
% Plot final result
if plotting>=PARTIAL

disp(' ')
disp('+++++')
disp('Computing plot points.')
disp('+++++')
disp(' ')

figure(fig_handle);clf;drawnow;
semiellipse(P2,GAMMA,[],plot_pts),hold on
fig_handle=fig_handle+1;

figure(fig_handle);clf;drawnow;
control_p(X,A,B,KOv,KOe,GAMMA,plot_pts);
end

% Compute composite gain.
Kv=Kv+KOv;
Ke=Ke+KOe;

% Define output ellipsoid matrix
P=P2;

```

```
% Compute maximum gain on ellipse, U
[f,g,U,constraints]=control_fg(X,A,B,KOv,KOe,GAMMA,UMAX,...
    Q0,control_const,weight,pd_weight,inv_weight);
Gstop=[Gstop g];
Fstop=[Fstop f];

% Clean up hard drive
delete control.mat
delete control_best.mat

% Terminate counter and display elapsed time
toc

% ===== END OF FILE: CONTROL.M =====
```

### G.3.2 Cost Function and Constraints

```
function [f,g,Uc,constraints]=control_fg(X,A,B,K0v,K0e,GAMMA,UMAX,...
    Q,control_const,weight,pd_weight,inv_weight);
% CONTROL_FG is the cost function and constraint routine for use with CONTROL.

% parametrizes in terms of sqrt(P) and Kv
% modifies definition of (at end) and pos. def. inequalities

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
[p]=max(size(X));
[c]=size(GAMMA,1);

flag=1; % Defines as controlable subspace

% PARAMETER ASSIGNMENTS
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(trace(P2));

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Invariance constraint
Acl=A-B*(Kv+K0v);
IC=P2*Acl+Acl'*P2+Q; % Write as negative definite form
[v,d]=eig(IC);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D*inv_weight]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

Acl=A-B*(Ke+K0e);
IC=P2*(-Acl)+(-Acl)'*P2+Q; % Write as negative definite form
[v,d]=eig(IC);
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D*inv_weight]; % Write as 'less than' constraint
for i=1:n
    constraints=char(constraints,'Invariance');
end

% Positive definiteness constraints.
[v,d]=eig(-P2*pd_weight); % Write as negative definite form
[D,sort_index]=esort(diag(d));
V=v(:,sort_index);
g=[g;D]; % Write as 'less than' constraint
for i=1:n
```

```

    constraints=char(constraints,'Positive Definiteness');
end

% State constraints
for i=1:c;
    Gamma=GAMMA(i,:);

    % Check to see if constraint is control dependent
    if max(max(Gamma*B))==0 % Not dependent
        g=[g; Gamma*inv(P2)*Gamma'-1]; % Note: not sign/time depend.
        constraints=char(constraints,['State Constraint ' num2str(i)]);
    else
        w=null(Gamma*(A-B*(Kv+K0v)));
        g=[g; (Gamma*w)*inv(w'*P2*w)*(Gamma*w)'-1];
        constraints=char(constraints,['Recovering State Constraint 'num2str(i)]);
        w=null(Gamma*(-(A-B*(Ke+K0e))));
        g=[g; (Gamma*w)*inv(w'*P2*w)*(Gamma*w)'-1];
        constraints=char(constraints,['Reaching State Constraint 'num2str(i)]);
    end
end;

% Control constraints
if control_const==1
    Uv=sqrt((Kv+K0v)*inv(P2)*(Kv+K0v)');
    g=[g; (Uv*Uv)/(UMAX*UMAX)-1]; % Normalize to 1
    constraints=char(constraints,'Control');
    Ue=sqrt((Ke+K0e)*inv(P2)*(Ke+K0e)');
    g=[g; (Ue*Ue)/(UMAX*UMAX)-1]; % Normalize to 1
    constraints=char(constraints,'Control');
end

Uc=max(Uv,Ue);

% ===== END OF FILE: CONTROL_FG.M =====

```

### G.3.3 Derivatives of Cost Function and Constraints

```
function [df_dX,dg_dX]=control_dfg(X,A,B,K0v,K0e,GAMMA,UMAX,...
    Q,control_const,weight,pd_weight,inv_weight);
% CONTROL_DFG is the derivative cost function and constraint routine
%   for use with CONTROL.

% parametrizes in terms of sqrt(P) and Kv
% modifies definition of (at end) and pos. def. inequalities

% Empirical weights on constraints (experimented w/weighting pos.def. 1e3)

% SYSTEM SIZE DEFINITION
[n,m]=size(B);
p=max(size(X));
c=size(GAMMA,1);
pg=n*m; % number of state feedback gain parameters (per gain matrix)
pe=p-2*pg; % number of ellipsoid parameters

flag=1; % Defines as controllable subspace

% PARAMETER ASSIGNMENTS
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=log(trace(P2));

df_dP=[];
xindex=0;
for i=1:n
    for j=i:n
        if i==j
            df_dP=[df_dP;2*P(i,j)/trace(P2)];
        else
            df_dP=[df_dP;4*P(i,j)/trace(P2)];
        end
    end
end
df_dKv=zeros(size(Kv))';
df_dKe=zeros(size(Ke))';
df_dX=[df_dP;df_dKv;df_dKe];

% =====

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear constraint array
dg_dX=[]; % clear constraint derivative array

% Invariance constraint (recoverable/positive time)
Acl=A-B*(Kv+K0v);
```

```

IC=P2*Acl+Acl'*P2+Q;
[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
Vi=v(:,sort_index);
g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*Acl+Acl'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKv=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Kv));
        EL(i,j)=1;
        dAcl_dKv_ij=-B*EL;
        dIC_dKv_ij=P2*(dAcl_dKv_ij)+(dAcl_dKv_ij)'*P2;
        dDi_dKv=[dDi_dKv; diag(Vi'*(dIC_dKv_ij)*Vi)'];
    end
end
dDi_dKe=zeros(pg,n); % 'n' eigenvalues

dDi_dX=[dDi_dP; dDi_dKv; dDi_dKe]*inv_weight;
dg_dX=[dg_dX dDi_dX];

% Invariance constraint (reachability/negative time)
Acl=-(A-B*(Ke+K0e));
IC=P2*Acl+Acl'*P2+Q;
[v,d]=eig(IC);
[Di,sort_index]=esort(diag(d));
Vi=v(:,sort_index);
g=[g;Di];      % Write as 'less than' constraint

dDi_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dIC_dP_ij=(EL'*P+P*EL)*Acl+Acl'*(EL'*P+P*EL);
        dDi_dP=[dDi_dP; diag(Vi'*(dIC_dP_ij)*Vi)'];
    end
end
dDi_dKv=zeros(pg,n);
dDi_dKe=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Ke));
        EL(i,j)=1;
        dAcl_dKe_ij=-(-B)*EL;

```

```

        dIC_dKe_ij=P2*(dAcl_dKe_ij)+(dAcl_dKe_ij)'*P2;
        dDi_dKe=[dDi_dKe; diag(Vi'*(dIC_dKe_ij)*Vi)'];
    end
end
dDi_dX=[dDi_dP; dDi_dKv; dDi_dKe]*inv_weight;
dg_dX=[dg_dX dDi_dX];

% Positive definiteness of P constraints.
PDC=-P2*pd_weight;
[v,d]=eig(PDC);
[Dpd,sort_index]=esort(diag(d));
Vpd=v(:,sort_index);
g=[g;Dpd]; % Write as 'less than' constraint

dDpd_dP=[];
for i=1:n
    for j=i:n
        EL=zeros(size(P));
        EL(i,j)=1;EL(j,i)=1;
        dPDC_dP_ij=-1*(EL'*P+P*EL)*pd_weight;
        dDpd_dP=[dDpd_dP; diag(Vpd'*(dPDC_dP_ij)*Vpd)'];
    end
end
dDpd_dKv=zeros(pg,n); % 'n' eigenvalues
dDpd_dKe=zeros(pg,n);

dDpd_dX=[dDpd_dP;dDpd_dKv;dDpd_dKe];
dg_dX=[dg_dX dDpd_dX];

% State constraints
for k=1:c;
    Gamma=GAMMA(k,:);

    % Check to see if constraint is control dependent
    if max(max(Gamma*B))==0 % Not dependent
        SC=Gamma*inv(P2)*Gamma'-1;
        g=[g; SC]; % Note: not sign/time depend.

        dSC_dP=[];
        for i=1:n
            for j=i:n
                EL=zeros(size(P2));
                EL(i,j)=1;EL(j,i)=1;
                dSC_dP_ij=Gamma*(-inv(P2)*(P*EL+EL*P)*inv(P2))*Gamma';
            end
            dSC_dP=[dSC_dP; dSC_dP_ij];
        end
        end
        dSC_dKv=zeros(pg,1);
        dSC_dKe=zeros(pg,1);
        dSC_dX=[dSC_dP;dSC_dKv;dSC_dKe];
        dg_dX=[dg_dX dSC_dX];

```

```

else
    w=null(Gamma*(A-B*(Kv+K0v)));
    SC=(Gamma*w)*inv(w'*P2*w)*(Gamma*w)''-1;
    g=[g; SC];
    dSC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P2));
            EL(i,j)=1;EL(j,i)=1;

            dinv_dP_ij=-inv(w'*P2*w)*(w'*(P*EL+EL*P)*w)*inv(w'*P2*w);
            dSC_dP_ij=(Gamma*w)*dinv_dP_ij*(Gamma*w)';

            dSC_dP=[dSC_dP; dSC_dP_ij];
        end
    end
    dSC_dKv=[];
    for i=1:m
        for j=1:n
            % Use numerical approximation
            EL=zeros(size(Kv));
            EL(i,j)=1;
            deltaKv_ij=EL*1e-5;
            deltaw=null(Gamma*(-(A-B*(Kv+K0v+deltaKv_ij))))-w;
            gradw=deltaw/1e-5;
            dinv_dKv_ij=-inv(w'*P2*w)*(gradw'*P2*w+w'*P2*gradw)*inv(w'*P2*w);
            dSC_dKv_ij=(Gamma*gradw)*inv(w'*P2*w)*(Gamma*w)'+...
                (Gamma*w)*dinv_dKv_ij*(Gamma*w)'+...
                (Gamma*w)*inv(w'*P2*w)*(Gamma*gradw)';
            dSC_dKv=[dSC_dKv; dSC_dKv_ij];
        end
    end
    dSC_dKe=zeros(pg,1);
    dSC_dX=[dSC_dP;dSC_dKv;dSC_dKe];
    dg_dX=[dg_dX dSC_dX];

    w=null(Gamma*(-(A-B*(Ke+K0e))));
    SC=(Gamma*w)*inv(w'*P2*w)*(Gamma*w)''-1;
    g=[g; SC];
    dSC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P2));
            EL(i,j)=1;EL(j,i)=1;

            dinv_dP_ij=-inv(w'*P2*w)*(w'*(P*EL+EL*P)*w)*inv(w'*P2*w);
            dSC_dP_ij=(Gamma*w)*dinv_dP_ij*(Gamma*w)';

            dSC_dP=[dSC_dP; dSC_dP_ij];
        end
    end
    dSC_dKv=zeros(pg,1);
    dSC_dKe=[];

```



```

    for i=1:m
        for j=1:n
            % Use numerical approximation
            EL=zeros(size(Ke));
            EL(i,j)=1;
            deltaKe_ij=EL*1e-5;
            deltaw=null(Gamma*(-(A-B*(Ke+K0e+deltaKe_ij))))-w;
            gradw=deltaw/1e-5;
            dinv_dKe_ij=-inv(w'*P2*w)*(gradw'*P2*w+w'*P2*gradw)*inv(w'*P2*w);
            dSC_dKe_ij=(Gamma*gradw)*inv(w'*P2*w)*(Gamma*w)'+...
                (Gamma*w)*dinv_dKe_ij*(Gamma*w)'+...
                (Gamma*w)*inv(w'*P2*w)*(Gamma*gradw)';
            dSC_dKe=[dSC_dKe; dSC_dKe_ij];
        end
    end
    dSC_dX=[dSC_dP;dSC_dKv;dSC_dKe];
    dg_dX=[dg_dX dSC_dX];
end

% Control constraints
if control_const==1
    % Recoverable control
    CC=(1/UMAX^2)*((Kv+K0v)*inv(P2)*(Kv+K0v)')-1;
    g=[g;CC];

    dCC_dP=[];
    for i=1:n
        for j=i:n
            EL=zeros(size(P2));
            EL(i,j)=1;EL(j,i)=1;
            dCC_dP_ij=(1/UMAX^2)*((Kv+K0v)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*...
                (Kv+K0v)');
            dCC_dP=[dCC_dP; dCC_dP_ij];
        end
    end
    dCC_dKv=[];
    for i=1:m
        for j=1:n
            EL=zeros(size(Kv));
            EL(i,j)=1;
            dCC_dKv_ij=(1/UMAX^2)*((EL)*inv(P2)*(Kv+K0v)'+(Kv+K0v)*inv(P2)*(EL)');
            dCC_dKv=[dCC_dKv; dCC_dKv_ij];
        end
    end
    dCC_dKe=zeros(pg,1);
    dCC_dX=[dCC_dP; dCC_dKv; dCC_dKe];
    dg_dX=[dg_dX dCC_dX];

    % Reachable control
    CC=(1/UMAX^2)*((Ke+K0e)*inv(P2)*(Ke+K0e)')-1;
    g=[g;CC];

    dCC_dP=[];

```

```

for i=1:n
    for j=i:n
        EL=zeros(size(P2));
        EL(i,j)=1;EL(j,i)=1;
        dCC_dP_ij=(1/UMAX^2)*((Ke+K0e)*(-inv(P2)*(EL'*P+P*EL)*inv(P2))*...
            (Ke+K0e)');
        dCC_dP=[dCC_dP; dCC_dP_ij];
    end
end
dCC_dKv=zeros(pg,1);
dCC_dKe=[];
for i=1:m
    for j=1:n
        EL=zeros(size(Ke));
        EL(i,j)=1;
        dCC_dKe_ij=(1/UMAX^2)*((EL)*inv(P2)*(Ke+K0e)' + (Ke+K0e)*inv(P2)*(EL)');
        dCC_dKe=[dCC_dKe; dCC_dKe_ij];
    end
end
dCC_dX=[dCC_dP; dCC_dKv; dCC_dKe];
dg_dX=[dg_dX dCC_dX];
end

```

% ===== END OF FILE: CONTROL\_DFG.M =====

### G.3.4 Cost Function and Constraints for Finding $\tilde{u}$

```

function [f,g,U]=control_u_fg(x0,K0v,K0e,GAMMA,X,n,m);
% CONTROL_U_FG is the cost function and constraint routine for use with
% overlapping controllable ellipsoids to find maximal control on restricted
% state-space.

% SYSTEM SIZE DEFINITION
[c]=size(GAMMA,1);

flag=1; % Defines as controllable subspace

% PARAMETER ASSIGNMENTS
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=-[x0'*(Kv+K0v)'+(Kv+K0v)*x0;x0'*(Ke+K0e)'+(Ke+K0e)*x0];
[f,f_index]=min(f);

smallest_index=zeros(2,1);
smallest_index(f_index(1))=1;

U=abs(-smallest_index'*[Kv+K0v;Ke+K0e]*x0);

% CONSTRAINTS (Must be of form <= 0)
g=[]; % clear array
constraints=[]; % constraints assignments (below) don't work with older Matlab

% Ellipsoid constraint
EC=x0'*P2*x0-1; % Write as negative definite form
g=[g;EC];
constraints=char(constraints,'Ellipsoid Constraint');

% State constraints
for i=1:c
    for j=1:2
        SC=(-1^j)*GAMMA(i,:)*x0-1; % Consider both plus/minus GAMMA
        g=[g;SC];
        constraints=char(constraints,['State Constraint ' num2str(i)]);
    end
end;

% ===== END OF FILE: CONTROL_U_FG.M =====

```

### G.3.5 Derivatives of Cost Function and Constraints for Finding $\tilde{u}$

```

function [df_dx0,dg_dx0]=control_u_dfg(x0,K0v,K0e,GAMMA,X,n,m);
% CONTROL_U_DFG is the derivative cost function and constraint routine for
% use with overlapping controllable ellipsoids to find maximal control
% on restricted state-space.

% SYSTEM SIZE DEFINITION
c=size(GAMMA,1);
flag=1;

% PARAMETER ASSIGNMENTS
[P,Kv,Ke]=x2pk(X,n,m,flag);
P2=P*P;

% OBJECTIVE FUNCTION (Minimization)
f=-[x0'*(Kv+K0v)'*(Kv+K0v)*x0;x0'*(Ke+K0e)'*(Ke+K0e)*x0];
[f,f_index]=min(f);

smallest_index=zeros(2,1);
smallest_index(f_index(1))=1;

df_dx0=-[2*(Kv+K0v)'*(Kv+K0v)*x0 2*(Ke+K0e)'*(Ke+K0e)*x0]*smallest_index;

% =====

% CONSTRAINTS (Must be of form <= 0)
dg_dx0=[]; % clear constraint derivative array

% Ellipsoid constraint
EC=x0'*P2*x0-1; % Write as negative definite form
dEC_dx0=2*P2*x0;
dg_dx0=[dg_dx0 dEC_dx0];

% State constraints
for i=1:c
    for j=1:2
        dSC_dx0=(-1^j)*GAMMA(i,:); % Consider both plus/minus GAMMA
        dg_dx0=[dg_dx0 dSC_dx0];
    end
end

% ===== END OF FILE: CONTROL_U_DFG.M =====

```

### G.3.6 Plotting Routine

```
function [U,P,Kv,Ke]=control_p(X,A,B,K0v,K0e,GAMMA,plot_pts);
% CONTROL_P is the trajectory plotting routine for use with CONTROL

flag=1;

% Plotting options
N=max(floor((plot_pts/20)^2),5); % plot trajectory from every Nth data
                                % point on the semi-ellipse with a
                                % maximum of 20 pts

if nargin<4
    error('Not enough input arguments.')
```

```

    [y,t,x]=initial(sysv,[x1(i),x2(i)],[0:.01:10]);
    plot(x(:,1),x(:,2),'r')
end

xlabel('x1'),ylabel('x2')
title('Recovering Trajectories')

% Plot boundary for reachable trajectories
subplot(122)
plot(x1,x2,'k');grid on;axis square;hold on

% Compute reaching and recovering state-space descriptions
syse=ss(-A+B*Ke,B,eye(n),zeros(n,m));

% Plot trajectories for every Nth point on the ellipsoid boundary
for i=1:ceil(sqrt(N)):max(size(x1));
    [y,t,x]=initial(syse,[x1(i),x2(i)],[0:.01:10]);
    plot(x(:,1),x(:,2),'b')
end

xlabel('x1'),ylabel('x2')
title('Reaching Trajectories')
elseif n==3 % 3-D ELLIPSOID
% Compute ellipsoid boundary
[x1,x2,x3]=semiellipse(P2,GAMMA,[],plot_pts);

% Compute corresponding control effort
for i=1:size(x1,1)
    for j=1:size(x1,2)
        uv(i,j)=Kv*[x1(i,j);x2(i,j);x3(i,j)];
        ue(i,j)=Ke*[x1(i,j);x2(i,j);x3(i,j)];
    end
end

% Compute maximum control effort on boundary
U=max(max([abs(uv);abs(ue)]));

% Plot boundary vs. control
subplot(121)
surf(x1,x2,x3,uv/U);grid on;axis square;hold on
subplot(122)
surf(x1,x2,x3,ue/U);grid on;axis square;hold on

% Compute reaching and recovering state-space descriptions
sysv=ss(A-B*Kv,B,eye(n),zeros(n,m));
syse=ss(-A+B*Ke,B,eye(n),zeros(n,m));

% Plot trajectories for every Nth point on the ellipsoid boundary
subplot(121)
hold on
for i=1:N:size(x1,1);
    for j=1:N:size(x1,2)
        [y,t,x]=initial(sysv,[x1(i,j),x2(i,j),x3(i,j)],[0:.01:10]);
        plot3(x(:,1),x(:,2),x(:,3),'r')
    end
end

```

```

        end
    end
    drawnow
    grid on
    colormap('copper')
    lighting phong
%   light('Position',[5 -5 2])
    light('Position',[0 0 5])
    h=findobj('Type','surface');
    set(h,'FaceLighting','phong',...
        'FaceColor','interp',...
        'EdgeColor',[.4 .4 .4],...
        'BackFaceLighting','reverselit',...
        'AmbientStrength',1,...
        'DiffuseStrength',1);
    shading interp
    xlabel('x1'),ylabel('x2'),zlabel('x3')
    title('Recovering Trajectories')

    subplot(122)
    hold on
    for i=1:N:size(x1,1);
        for j=1:N:size(x1,2)
            [y,t,x]=initial(syse,[x1(i,j),x2(i,j),x3(i,j)], [0:.01:10]);
            plot3(x(:,1),x(:,2),x(:,3),'b')
        end
    end
    drawnow
    grid on
    colormap('copper')
    lighting phong
%   light('Position',[5 -5 2])
    light('Position',[0 0 5])
    h=findobj('Type','surface');
    set(h,'FaceLighting','phong',...
        'FaceColor','interp',...
        'EdgeColor',[.4 .4 .4],...
        'BackFaceLighting','reverselit',...
        'AmbientStrength',1,...
        'DiffuseStrength',1);
    shading interp
    xlabel('x1'),ylabel('x2'),zlabel('x3')
    title('Reaching Trajectories')

end

% ===== END OF FILE: CONTROL_P.M =====

```

## G.4 Miscellaneous Files

### G.4.1 Plotting Point Generator for Semi-Ellipsoidal Set

```
function [xx,yy,zz] = semiellipse(P,GAMMA,X0,n)
% SEMIELLIPSE Generates the semi-ellipsoidal set.
% [X,Y,Z] = SEMIELLIPSE(P,GAMMA,X0,n) generates the unit
% ellipsoid
%
%  $(x-X0)'*P*(x-X0)=1$ 
%
% then scales these values so that they fit within the
% linear constraint
%
%  $GAMMA*x \leq 1$ 
%
% For P matrix (3x3), SEMIELLIPSE generates three (n+1)x(n+1)
% matrices so that SURF(X,Y,Z) produces the 3-D set,
% For P matrix (2x2), SEMIELLIPSE generates two (n+1)x(1)
% vectors so that PLOT(X,Y) produces the 2-D set.
%
% The arguments X0 and n are optional. Default values are
% the origin for X0 and 40 points for n.
%
% SEMIELLIPSE(P,X0,n) without any return variables graphs the
% ellipse using SURFACE/PLOT.
%
% Original code: SPHERE.M
% Clay M. Thompson 4-24-91, CBM 8-21-92.
% Copyright (c) 1984-98 by The MathWorks, Inc.
% $Revision: 5.3 $ $Date: 1997/11/21 23:46:48 $
%
% Modified code: SEMIELLIPSE.M
% Brian D. O'Dell 4-19-98

if nargin == 0, error('Must define ellipsoid matrix, P.');
```

```
end

% CHECK VALIDITY OF P
if size(P,1)~=size(P,2)
    error('P must be square.')
```

```
end
if min(eig(P))<=0
    error('P must be positive definite.')
```

```
end

% COMPUTE NUMBER OF STATES
states=size(P,1);
if (states~=2)&(states~=3)
    error('P must be a 2x2 or 3x3 matrix.')
```

```
end

% CHECK SIZE OF GAMMA
if size(P,1)~=size(GAMMA,2)
    error('P and GAMMA must have same number of columns (states).')
```



```

end

% CHECK FOR OPTIONAL ARGUMENTS
if nargin==2
    n=40;
end

if states==2 % 2-D ELLIPSE
    % -pi <= theta <= pi is a row vector

    theta = (-n:2:n)/n*pi;

    sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;

    x0 = cos(theta); % Define points for a unit circle
    y0 = sintheta;

    for i=1:(n+1) % Loop through the data points
        temp=[x0(i);y0(i)]; % Create a vector for the data point
        alpha=sqrt(temp'*P*temp); % Compute the scaling factor for unit ellipse
        x(i)=x0(i)/alpha; % Scale the data points
        y(i)=y0(i)/alpha;
        overlap_scale=max([1;abs(GAMMA*[x(i);y(i)])]);
        x(i)=x(i)/overlap_scale;
        y(i)=y(i)/overlap_scale;
    end

    if nargout == 0
        plot(x,y)
        xlabel('x1'),ylabel('x2')
    else
        xx = x; yy = y;
    end
end

else % 3-D ELLIPSE
    % If plotting, display full ellipse; for trajectories, generate quadrant
    if nargout==0
        % -pi <= theta <= pi is a row vector.
        % -pi/2 <= phi <= pi/2 is a column vector.

        theta = (-n:2:n)/n*pi;
        phi = (-n:2:n)'/n*pi/2;
        cosphi = cos(phi); cosphi(1) = 0; cosphi(n+1) = 0;
        sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;
    else
        % -pi/2 <= theta <= pi/2 is a row vector.
        % -pi/2 <= phi <= 0 is a column vector.

        theta = (-n:2:n)/n*pi/2;
        phi = (-2*n:2:0)'/((2*n)*pi/2);
        cosphi = cos(phi);
        sintheta = sin(theta);
    end
end

```

```

x0 = cosphi*cos(theta);           % Define points for a unit sphere
y0 = cosphi*sintheta;
z0 = sin(phi)*ones(1,n+1);

for i=1:max(size(theta))           % Loop through the data points
    for j=1:max(size(phi))
        temp=[x0(i,j);y0(i,j);z0(i,j)]; % Create a vector for the data
        alpha=sqrt(temp'*P*temp);      % Scaling factor for unit ellipse
        x(i,j)=x0(i,j)/alpha;         % Scale data points
        y(i,j)=y0(i,j)/alpha;
        z(i,j)=z0(i,j)/alpha;
        overlap_scale=max([1;abs(GAMMA*[x(i,j);y(i,j);z(i,j)])]);
        x(i,j)=x(i,j)/overlap_scale;
        y(i,j)=y(i,j)/overlap_scale;
        z(i,j)=z(i,j)/overlap_scale;
    end
end

if nargin == 0                     % Plot if no output
    disp(' ')
    disp('+++++')
    disp('Plotting semi-ellipsoidal set')
    disp('+++++')
    disp(' ')

    surf(x,y,z)
    grid on
    colormap('copper')
    lighting phong
    light('Position',[5 -5 2])
    h=findobj('Type','surface');
    set(h,'FaceLighting','phong',...
        'FaceColor','interp',...
        'EdgeColor',[.4 .4 .4],...
        'BackFaceLighting','reverselit',...
        'AmbientStrength',1,...
        'DiffuseStrength',1);
    shading interp
    xlabel('x1'),ylabel('x2'),zlabel('x3')
else
    xx = x; yy = y; zz = z;
end
end

% ===== END OF FILE: SEMIELLIPSE.M =====

```

## G.4.2 Search Parameter/Matrix Parameter Mapping Routines

### Ellipsoid Matrix and Control Gains to Search Vector

```
function [X]=pk2x(P,K1,K2,n,m)
% PK2X assigns elements of the ellipse matrix, P, and the
% state-feedback matrix, K, to elements of the search vector, X.
% If 5 arguments are provided, it assumes two state-feedback
% matrices are provided.
```

```
if nargin==4
    m=n;
    n=K2;

    X=[];
    xindex=0;
    for i=1:n
        for j=i:n
            xindex=xindex+1;
            X(xindex,1)=P(i,j);
        end
    end
    for i=1:m
        for j=1:n
            xindex=xindex+1;
            X(xindex,1)=K1(i,j);
        end
    end
elseif nargin==5
    X=[];
    xindex=0;
    for i=1:n
        for j=i:n
            xindex=xindex+1;
            X(xindex,1)=P(i,j);
        end
    end
    for i=1:m
        for j=1:n
            xindex=xindex+1;
            X(xindex,1)=K1(i,j);
        end
    end
    for i=1:m
        for j=1:n
            xindex=xindex+1;
            X(xindex,1)=K2(i,j);
        end
    end
end
end
```

```
% ===== END OF FILE: PK2X.M =====
```

## Search Vector to Ellipsoid Matrix and Control Gains

```
function [P,K1,K2]=x2pk(X,n,m,flag)
% X2PK assigns elements of the search vector, X, to elements of the ellipse
% matrix, P, and to the state-feedback gain(s), K1 (and K2, if flag=1).

if nargin==3
    flag=0;
end

P=[];
xindex=0;
for i=1:n
    for j=i:n
        xindex=xindex+1;
        P(i,j)=X(xindex,1);
        P(j,i)=X(xindex,1);
    end
end
K1=[];
for i=1:m
    for j=1:n
        xindex=xindex+1;
        K1(i,j)=X(xindex,1);
    end
end
if flag==1
    K2=[];
    for i=1:m
        for j=1:n
            xindex=xindex+1;
            K2(i,j)=X(xindex,1);
        end
    end
else
    K2=[];
end

% ===== END OF FILE: X2PK.M =====
```

VITA

Brian Dwayne O'Dell

Candidate for the Degree of

Doctor of Philosophy

Thesis: ELLIPSOIDAL AND SEMI-ELLIPSOIDAL CONTROLLED INVARIANT SETS FOR  
CONSTRAINED LINEAR SYSTEMS

Major Field: Mechanical Engineering

Biographical:

Personal: Born in Fort Smith, Arkansas, on August 30, 1971, the son of Danny and Melanie O'Dell.

Education: Graduated from Roland High School, Roland, Oklahoma, in May 1989; received Bachelor of Science degree in Mechanical Engineering and a Minor in Mathematics from Oklahoma State University, Stillwater, Oklahoma, in May 1993; received Master of Science degree in Mechanical Engineering from Oklahoma State University, Stillwater, Oklahoma, in December 1994. Completed the requirements for the Doctor of Philosophy degree with a major in Mechanical Engineering at Oklahoma State University in July, 1999.

Experience: Raised on a farm near Vian, Oklahoma; employed as an engineering intern, AES Shady Point, Inc., Panama, Oklahoma, summer 1991; employed as a teaching assistant for undergraduate measurements and instrumentation course, Oklahoma State University, Department of Mechanical & Aerospace Engineering, Fall 1994-Fall 1995 semesters; employed as a teaching assistant for graduate instrumentation course, Oklahoma State University, Department of Mechanical & Aerospace Engineering, Summer 1995 and Summer 1996 semesters; employed as lecturer for undergraduate numerical methods course, Oklahoma State University, Department of Mechanical & Aerospace Engineering, Spring 1996 semester; employed as a research assistant, Oklahoma State University, Department of Mechanical & Aerospace Engineering, Advanced Controls Laboratory, September 1996-present; employed as a system administrator, Oklahoma State University, Department of Mechanical & Aerospace Engineering, Advanced Controls Laboratory and Bioinformatics Laboratory, April 1999-present.

Professional Memberships: American Society of Mechanical Engineers.