AALBORG UNIVERSITY

Linear Matrix Inequality for Robust Control

LMI based Vibration Control

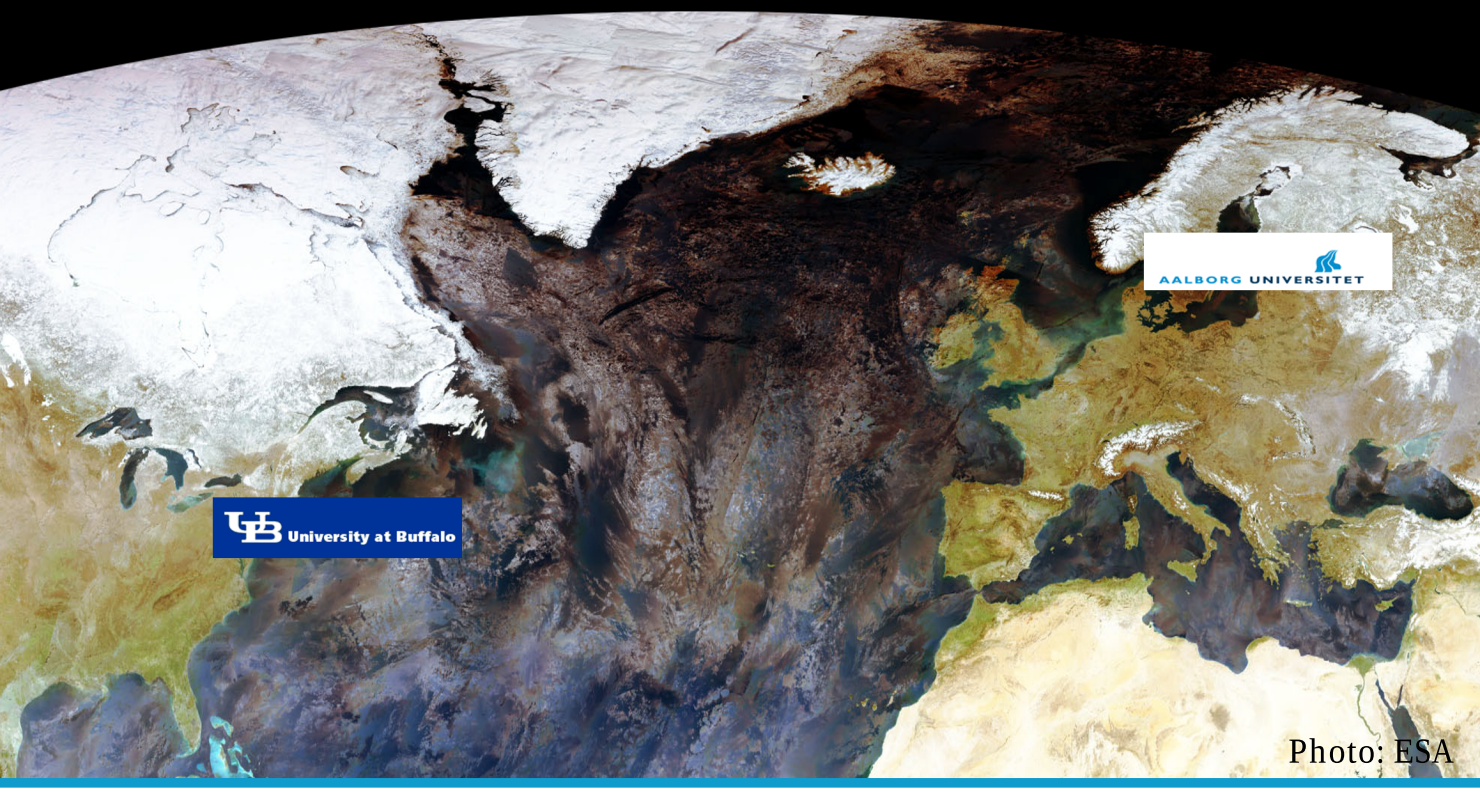


Photo: ESA

9th semester - by Rasmus Hviid Knudsen

2005

Intelligent Autonomous Systems

Aalborg University

Institute of Electronic Systems

Fredrik Bajers Vej 7 ▪ DK-9220 Aalborg Øst Telephone +45 96 35 87 00

Title: A research in Linear Matrix Inequality and Robust Control

Theme: Study Abroad at University at Buffalo

The State University of New York

Project period: August 28th 2005 - January 5th 2006

Project group:
938h

By:
Rasmus Hviid Knudsen

Supervisors:
Tarunraj Singh
Jakob Stoustrup

Publications: 5
Pages: 70
Finished: Jan 5th 2006

Abstract

This research is on LMI based robust control. Two general methods is derived a state feedback LMI formulation and a shaped input feed forward LMI formulation. The two design methods result in a general and efficient way of implementing robust control problems.

The state feedback design is applied to a 2 mass spring dash pot system which is very interesting in vibration control. Compared with LQR the LMI based robust state feedback controller has significant better performance in stabilizing a system with uncertain parameters.

In the feedforward approach bounded control and robustness to a uncertain parameter was archived. Two methods for robustness ware developed with different performance. Optimization was applied with respect to time and corresponds to the optimal bang bang control. However deflection control was not archived properly.

Two general LMI design methods is developed with interesting performance and can be used in many applications. The next step is to expand this approach to different systems and not necessary for vibration control. A step was taken to apply this to a satellite attitude control however the system was not reliable for the attempted robust control. The implementation ware very easy to set up and worked without problems.

Preface

This report serves as documentation for my Project in the academic year 2005/2006. A dream comes true. In this year I had the amazing opportunity to take my first study abroad. Therefore this work is done in my journey to the United States of America where I was working for Professor Dr. Singh.

In the field of control research this project is focused on formulating desired properties for robust control systems in form of Linear Matrix Inequalities. The goal is to define a robust control design as a LMI problem and then use the open source MATLAB toolbox Yalmip to solve the optimal solution. Yalmip supports different algorithms and offer an easy and convenient way of formulating the problem.

The project is made at my 9^{th} semester in the specialization Intelligent Autonomous Systems at Aalborg University. For this research I worked for my advisor Professor Dr. Singh. I worked in his lab at the Mechanical and Aerospace Department at New York State University at Buffalo. At Aalborg University I was happy to have Professor Jakob Stoustrup as my second advisor for the project.

Throughout the report figures, tables and equations are numbered consecutively according to the chapters. Citation numbers are referring to the bibliography at page 60, e.g. [Scherer & Weiland, page 10] is referring to page 10 in *Linear Matrix Inequalities in Control* by Carsten Scherer and Siep Weiland. References is inserted when used or in the end of a section if used in the hole section.

Simulation files, documentation and other relevant information are available from the enclosed CD-ROM.

The $\succ$ notation is used for matrix inequalities and the $>$ notation is used for scalar inequalities.

New York state university at Buffalo, Jan 5 2005

Rasmus Hviid Knudsen

Table of Contents

1 Introduction

The desire to control more complex systems and design of intelligent autonomous systems maintain an ongoing evolving in control theories and control design. In this area robust control has proven to be efficient in designing controllers which can operate systems in a wider area than by classical controllers.

Research in Robust control design powered by development of power full Linear Matrix Inequalities (LMI) algorithms has appeared to be a very efficient way to formulate controller performance and design. Instead of obtaining an analytical solution the problem is formulated into a LMI problem where the desired system performances are rewritten into LMI constraints. The development of efficient LMI solvers and algorithms improves this area. These algorithms will be used in this project through the open source MATLAB toolbox Yalmip. The intent of the first part of this introduction is to clarify some of the LMI principles. The second part introduces the Yalmip program and describes some of the benefit provided by this software.

1.1 Linear Matrix Inequality introduction

Lets define a Linear Matrix Inequality to get started with LMIs. The general LMI expression is defined as:

$$F(x) = F_0 + x_1F_1 + \ldots + x_nF_n < 0 \tag{1.1}$$

where F(x) is an affine function of the real vector $x = [x_1, x_2, \ldots, x_m]^T$ the decisions variables and $F_0, F_1, \ldots, F_n$ are real symmetric matrices, i.e. $F_i = F_i^T$. The inequality < 0 in equation 1.1 means negative definite which means $u^T F(x)u < 0$ for all $u \in \mathbb{R}^m$. Alternative this means that the smallest eigenvalue of F(x) is less then zero.

The linear matrix inequality defines a convex constraint on x which means that the solution of equation 1.1 is convex. In figure 1.1 a convex and a non convex solution area are shown. For a convex solution all solutions connected with a straight line must be within the object.

This means that if a optimization problem is convex and can be formulate as a LMI problem then there exits very efficient algorithms to solve the problem for the optimal and feasible solution. For real world applications one can not relay on a single model of a system because uncertainties always will exist between the model and the real world. In order to handle these problems robust controllers will be considered later on in this project. In the next section LMI in control design will be discussed.

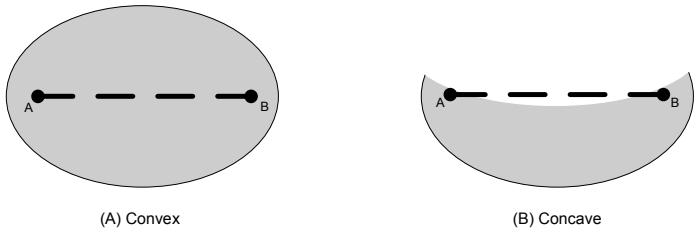


Figure 1.1: Illustration of a convex solution area and a non convex solution area.

1.1.1 LMI in Control Design

In these problems the Lyapunov equation for stability has proven to be very useful. Consider the system given by equation 1.2

$$\dot{x} = Ax \tag{1.2}$$

where $A \in \mathbb{R}^{n\times n}$. By Lyapunov's Theorem this system is asymptotic stable if and only if there exist $P \succ 0$ and $\mathbb{S}^n$ i.e P is an $(n \times n)$ symmetric positive definite matrix such that:

$$A^T P + PA \prec 0 \tag{1.3}$$

For this simple system the Schur complement can be used in order to formulate this as a pure LMI see e.g. appendix A for a description of the Schur complement. Asymptotic stability of the system 1.2 is now equivalent to feasibility of the LMI.

$$\begin{bmatrix} P & 0 \\ 0 & -A^T P - PA \end{bmatrix} \succ 0 \tag{1.4}$$

Using the form given in equation 1.4 we are ready to look at LMI for control.

1.1.2 State Feedback

To design a state feedback for a state-space model, the theory derived for the system in equation 1.4 are used. For a state-space system of the form:

$$\dot{x} = Ax + Bu \tag{1.5}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ i.e. a system of n-dimensional state-space and m-dimensional input space. The design of a stabilizing control law $u = Fx$ with $F \in \mathbb{R}^{m\times n}$ for this system relay on the Lyapunov equation 1.4. Find a matrix $P \succ 0$ and a matrix $K \in \mathbb{R}^{m\times n}$ such that

$$(A + BK)^T P + P(A + BK) \prec 0 \tag{1.6}$$

where K is the state feedback.
Since this equation have products of unknowns here P and K, the equation is not a LMI. A LMI formulation of the equation can be obtained through simplification, by introducing new variables $X = P^{-1}$ and $Y = KP^{-1}$

$$XA^T + AX + Y^T B^T + BY \prec 0 \tag{1.7}$$

The LMI in equation 1.7 is obtained by pre and post multiplication of P^{-1} and the state feedback is given by $K = YX^{-1}$. [Scherer & Weiland]

However the above theory only gives a feasible state feedback since optimization is not considered here. Optimization in order to obtain special performance for selected states is considered later on.

1.1.3 Closed Loop Feedback

Lets now consider the problem of designing a stabilizing output feedback for the system given in figure 1.2.

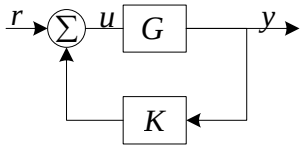


Figure 1.2: Illustration of a simple feedback system.

Here again only the a feasible solution is analyzed.

Let G in figure 1.2 be a state-space model of the form (A,B,C,D) where $D = 0$ and the output feedback K written as an entire state-pace model (A_k,B_k,C_k,D_k). Hence the closed loop system A_{cl} can be defined as follows:

$$A_{cl} = \begin{bmatrix} A + BD_kC & BC_k \\ B_kC & A_k \end{bmatrix} \tag{1.8}$$

then by defining:

$$\mathcal{A} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} 0 & B \\ 0 & 0 \end{bmatrix}, \quad \mathcal{C} = \begin{bmatrix} 0 & I \\ C & 0 \end{bmatrix} \tag{1.9}$$

and the matrix of controller parameters as:

$$K = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix} \tag{1.10}$$

a convenient notation for state-space system A_{cl} can now be derived from the above definitions:

$$A_{cl} = \mathcal{A} + \mathcal{B}K\mathcal{C} \tag{1.11}$$

The Lyapunov equation 1.3 on page 8 for system stability are again used in this design approach. The system with the feedback design will be stable if there exits a matrix $P \succ 0$ such that:

$$(\mathcal{A} + \mathcal{B}K\mathcal{C})^T P + P(\mathcal{A} + \mathcal{B}K\mathcal{C}) \prec 0 \tag{1.12}$$

The design problem is now to find a matrix $P \in \mathcal{S}^{n\times n}$ and K such that equation 1.12 is satisfied. However the equation in 1.12 is still not convex because there is products of P and K, therefore the Elimination Lemma is used on the problem see appendix C for the Elimination Lemma. By defining $N = P\mathcal{B}$, $M = \mathcal{C}^T$, $K = X$ and $H = P\mathcal{A} + \mathcal{A}^T P$ the inequality from equation 1.12 becomes:

$$NXM^T + MX^T N^T + H \prec 0 \tag{1.13}$$

Then by further manipulations and another Elimination lemma described in [Ulf Jönsson] there exits the following Stabilization Theorem for the problem.

The system (A,B,C,D) can be stabilized by a n_K-dim controller where n_k are the states of the controller, if and only if there exits $X, Y \in S_+^{n\times n}$ where $n = n_G$ equals the number of states in the plant G(s) such that:

$$B_\perp(AY + YA^T) \prec 0 (C^T)_\perp(XA + A^T X)((C^T)_\perp)^T \prec 0 \tag{1.14}$$

$$\begin{bmatrix} X & I \\ I & Y \end{bmatrix} \succeq 0 \text{ and rank } \begin{bmatrix} X & I \\ I & Y \end{bmatrix} \preceq n_G + n_K \tag{1.15}$$

where $B_\perp$ is the orthogonal complement of the matrix B. For $n_K \geq n_G$ the solution will be convex because the rank constraint can be removed. So for systems that satisfy $n_K \geq n_G$ there exist by the Stabilization Theorem a controller that will stabilize the system. For a system where there exits an output feedback, the final step is to find the relationship between the feedback K and the matrices X and Y. By Schur complement the matrix P in equation 1.12 is:

$$P = \begin{bmatrix} X & X_2 \\ X_2^T & I \end{bmatrix} \succ 0 \tag{1.16}$$

where $X - Y^{-1} = X_2 X_2^T$.

From this construction of P, the LMI in equation 1.13 can be built and solved for K. Then with the feedback gain K the closed-loop control system can be calculated using equation 1.11.

1.2 Yalmip

The main interest in software will be on the free open source LMI solver **Yalmip** developed by Dr. Johan Löfberg. Yalmip works as a MATLAB toolbox, therefore the goal is to achieve experience in using this software toolbox and develop programs for easy calculation of optimization problems.

Yalmip offers a easy and advantageous way for coding LMI formulated optimization problems. Yalmip support convex linear, quadratic, second order and semidefinite programming. [Yal]

Lets formulate a simple LMI problem and program it in Yalmip to see the convention.

To create variables in Yalmip simply call *sdpvar*:

$$x = \text{sdpvar}(n, m, 'field', 'type') \tag{1.17}$$

this will create a matrix x of variables. The matrix is of height n and width m. The input *'field'* can be set to real or complex depending on the users need. The input *'type'* determines the type of the matrix e.g. 'full' or 'symmetric'. Hence in order to create a scalar simply type $x = \text{sdpvar}(1,1)$. Constraints on the system are also defined easily here the Yalmip function *set* is used.

$$\text{LMI}_{\text{constraint}} = set(X, 'tag') \tag{1.18}$$

where X defines the constraint and *'tag'* are used if the user want to index the constraints. For multiple constraints the constraints can just be added together (e.g. the constraint (con) given in equation 1.4 on page 8 can be defined as:

$$\begin{aligned} \text{con} &= \text{set}(\text{P} > 0) \\ \text{con} &= \text{con} + \text{set}(\text{A}^{\text{T}}\text{P} + \text{PA} < 0) \end{aligned} \tag{1.19}$$

Programming in Yalmip becomes very simple since Yalmip uses the general MATLAB notation. So the constraint defined in equation 1.4 on page 8 could also be defined as:

$$\text{con} = \text{set}([\text{P}\ 0; 0\ -\text{A}^{\text{T}}\text{P} - \text{AP}] > 0) \tag{1.20}$$

This notation has proven to be a very convenient way for describing constraints compared to the MATLAB LMI toolbox. The Yalmip solver function ends this introduction to Yalmip. The solvespd are defined as:

$$\text{diagnostics} = \text{solvespd}(con, X, ops) \tag{1.21}$$

where *con* describe the constraints, X describes the spdvar object (the variables) and is also used to give commands e.g. *sum(X)* will minimize the object function X. The field *ops* let

the user set options using the *sdpsettings* function e.g. ops can be used to specify a specific algorithm for the solver. The function *double(X)* is used to take out the results returned by solvesdp.

For a program example see appendix B on page 63. [Yal]

Motivation

2

In this project the main research will be in archiving knowledge in using LMI methods for formulation and designing of robust. After deriving the theories for a LMI formulations of controller design problems, the methods will be implemented in MATLAB by use of Yalmip. The implemented methods will afterwards be applied for different systems to test the technology and archive results. For verification of the designed robust controller a classical controller will be designed.

2.1 State Feedback Formulation

The first problem for research will be: Find a general design method for robust controllers with Linear Quadratic Regulator (LQR) performance. Here the introduction and literature is a great motivation for the further research in a LMI formulation for controller design. If the method proves to be efficient and good results is archived then the LMI formulation will be implemented in a more complex system.

- LMI formulation for Feedback design
- LMI formulation of Robust controller with LQR performance
- Implementation and tests

The criteria are to develop a LMI formulation to construct a controller which is able to stabilize a system for a variety of uncertainties in the system. Here the two mass spring dash pot system will be considered since this is a very reliable system for vibration control. The criteria are satisfied if the designed robust controller can stabilize a system with uncertainties with a significant better performance than with a LQR controller applied.

2.2 Feed Forward Formulation

Another interesting approach for LMI formulation is highlighted by [Thomas Conord, page 41] here the LMI tool is used as an optimization solver. In this research area a LMI method for feedforward design is developed and used. The method is based on shaping the input of a system such that the residual energy at the end of the maneuver is minimal.

- LMI method for feedforward design

- Shaped input design
- Bounded control input
- Optimization (steps and time)
- Deflection Constraint
- Robustness

In this research the focus is more on controlling a system maneuver than stabilizing the system. However vibration control will stile be considered since the two mass spring dash pot system is used. Here the goal is to develop a controller with robust performance to a uncertain parameter. A system maneuver will be defined and the criteria are then to control this maneuver and keep the system stable. Here different constraints will be applied in the LMI design such as bounded control and deflection constrain. Also optimization will be considered with respect to time and amount of input steps. Criteria: The system must be robust to a uncertain parameter, the system must be optimized with respect to time for the maneuver and amount of steps. Possible reach the optimal bang bang controller. The system must be able to control deflection in the system.

2.3 Research Limitations

Since time is definitely a element or dimension which this research is opposed to there have to be limitations in the area of research and range of details for which each topic can be studied. Therefore the range of complexity in the models which the research technologies is opposed to is limited to simple systems which however still is very reliable especially for vibration control. However the discussion about applying the technology to more complex system will be taken. Also the range of details which an area can be studied is limited, the idea here is more to come up with some very interesting research areas and show there possibilities. Models are simplified; values and parameter will in common be pretended measurable.

State Feedback Controller 3

This chapter serves to use the theories and definitions stated in the introduction to actually design controllers. In the introduction the theory and definition was only concerned about designing controllers for system stability. However this chapter goes further bye introducing controller with performances. The first part of controller performance will deal with robust controllers having Linear Quadratic Regulator (LQR) performances.

3.1 Controller Performance Definition

From the theories stated in the introduction it is known that a feasible state feedback controller can be designed by the LMI method using the Lyapunov stability criteria. This was introduced in section 1.1.2 on page 8 as: Find a matrix $P \succ 0$ and a matrix $K \in \mathbb{R}^{m \times n}$ such that

$$(A+BK)^T P + P(A+BK) \prec 0 \tag{3.1}$$

where K is the state feedback. By manipulation this was formulated as a real LMI equation and the equation for the feasible feedback was found. However as stated in section 1.1.2 on page 8 the above equation only gives a feasible feedback for stabilizing the system. In many problems the system is opposed to certain requirements and therefore performance must be defined and included in the LMI formulation.

3.1.1 Quadratic Performance Definition

In optimal control the performance for the controller is archived by minimizing a performance index. The general performance index and design structure for quadratic stability (LQR) is given as.

$$Min \quad J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{3.2}$$

where Q and R are positive-definite matrixes. Since the negative feedback control law is given as $u(t) = -Kx(t)$ equation 3.2 can be written as:

$$Min \quad J = \int_0^\infty x^T (Q + K^T R K) x \, dt \tag{3.3}$$

this LQR quadratic stability problem can be linked to the Lyapunov stability problem stated in equation 3.1 on the previous page by use of a quadratic candidate Lyapunov function:

$$V(t) = x(t)^T P x(t) \tag{3.4}$$

where P is a positive symmetric matrix $P = P^T \succ 0$ such that the derivative of the Lyapunov function becomes:

$$\dot{V} = \frac{dV(x(t))}{dt} \leq 0 \tag{3.5}$$

Then by the Lyapunov stability criteria: If the quadratic Lyapunov function in equation 3.4 is always positive definite and the derivative in equation 3.5 is always negative definite then the system will be global asymptotical stable at the equilibrium point here zero. Now from [Ogata, 920] it is known that:

$$x^T(Q + K^T R K)x \preceq -\frac{d}{dt}(x^T P x) \tag{3.6}$$

then be taking the integral from zero to infinity on both sides of the inequality and assuming that the closed-loop system is stable:

$$\int_0^\infty x^T(Q + K^T R K)x \, dt \leq \int_0^\infty -\frac{d}{dt}(x^T P x) dt \tag{3.7}$$

this brings back the cost function with a upper bound:

$$\int_0^\infty x^T(Q + K^T R K)x \, dt \leq x(0)^T P x(0) \tag{3.8}$$

Also it can be shown by the following manipulation of equation 3.6 that it comes very close to the original Lyapunov equation.

$$\frac{d}{dt}(x^T P x) + x^T(Q + K^T R K)x \preceq 0 \tag{3.9}$$

by taking the derivative and using the fact that $\dot{x} = Ax + BKx = (A + BK)x$:

$$x^T[(A + BK)^T P + P(A + BK) + (Q + K^T R K)]x \preceq 0 \tag{3.10}$$

then from equation 3.3 on the preceding page it is known that the above equation must be true for all $x(t)$ meaning the system goes to steady state, therefore:

$$(A + BK)^T P + P(A + BK) + (Q + K^T R K) \preceq 0 \tag{3.11}$$

Using the the above information a formulation for calculating controllers with LQR performance using the LMI approach can be described.

3.1.2 Equivalence between LMI formulation and LQR

From the Lyapunov development, it is known that there exists a unique optimal solution $P \succ 0$ and K that minimizes the quadratic cost J in problem 3.3. So the problem can now be reformulated as follows

$$\begin{aligned} \underset{P\succ 0,K}{Min} \quad & x(0)^T P x(0) \\ & such\ that \\ & (A+BK)^T P + P(A+BK) + Q + K^T R K \preceq 0 \\ & for\ all\ x(0) \end{aligned} \tag{3.12}$$

so by minimizing $x(0)^T P x(0)$ in order to find the feedback gain K the solution converge to a LQR solution. In other words the feedback gain K minimizes the quadratic cost defined in equation 3.2 on page 15. Hereby a connection between the solution of the problem 3.2 and 3.12 is shown. This formulation will now be used to set up a proper LMI design with performance.

3.2 LMI formulation for Quadratic Design

In formulating problem 3.12 as a proper LMI equation the first problem encountered is the cost function which depends on the unknown matrix P and therefore dos not correspond directly to a LMI formulation. Like described earlier the cost problem can be reformulated by the Schur complement. From [Gregory J.] we have:

$$Min\ \gamma\ such\ that \qquad x(0)^T P x(0) < \gamma \tag{3.13}$$

$$which\ is\ equivalent\ to \qquad \begin{pmatrix} \gamma & x(0)^T \\ x(0) & P^{-1} \end{pmatrix} \succ 0$$

Since there are products of P and K in equation 3.12 this is still not a proper LMI formulation. Therefore the transformation already used in the introduction is recalled. Pre and post multiply equation 3.11 on the facing page with P^{-1} additionally create the variables $X = P^{-1}$ and $Y = KP^{-1}$. Equation 3.11 becomes:

$$XA^T + AX + Y^T B^T + BY + XQX + Y^T R Y \preceq 0 \tag{3.14}$$

since X is symmetric this can be rewritten into:

$$XA^T + AX + Y^T B^T + BY + \begin{bmatrix} X \\ Y \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} \preceq 0 \tag{3.15}$$

by these approaches the problem formulation becomes:

$$\underset{P\succ 0,K}{Min} \quad \gamma \quad such\,that \tag{3.16}$$

$$\begin{pmatrix} \gamma & x(0)^T \\ x(0) & X \end{pmatrix} \succ 0$$

$$XA^T + AX + Y^T B^T + BY + \begin{bmatrix} X \\ Y \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} \preceq 0$$

$$for\,all\,x(0)$$

still a problem remains with the formulation in 3.16. The cost function J has to be minimized for all $x(0)$ at the same time, which corresponds to a non general solutions and a heavy computational job since it corresponds to infinity inequalities depending on the sampling of the unit circle. Different approaches are possible, however to obtain a general solution another simplification is used. In this approach, the cost is not anymore $x(0)^T Px(0)$ but an upper bound of it which is:

$$max(eig(P))x(0)^T x(0) \tag{3.17}$$

and it is known that for all $x(0)$:

$$J \leq x(0)^T Px(0) \leq max(eig(P))x(0)^T x(0) \tag{3.18}$$

so by minimizing 3.17, the LQR cost J is decreased but it is not proven that it will reach its minimum since:

$$min(x(0)^T Px(0)) \neq min(max(eig(P)x(0)^T x(0)) \tag{3.19}$$

instead of having a minimizing problem the formulation now express a minmax problem which corresponds to the worst case design. Therefore the solution will not converge exactly to the LQR solution but the solution which minimize the cost for the worst direction of the state.

3.2.1 Final LMI formulation

The LMI formulation can now be described in a convenient way, since the initial state $x(0)$ can be removed from the cost. The cost function which still has to be minimized is now $max(eig(P))$ or $min(eig(X))$ which has to be maximized. So from 3.16 and by using Schur

complement the problem formulation now becomes:

$$\begin{aligned} \underset{X\succ 0,Y}{Max} \quad & \gamma \\ such\ that \quad & \gamma I \prec X \end{aligned} \tag{3.20}$$

$$\begin{pmatrix} -Q^{-1} & 0 & X \\ 0 & -R^T & Y \\ X^T & Y^T & AX + XA^T + BY + Y^T B^T \end{pmatrix} \prec 0$$

Now a very efficient method is developed which dos not depend on a initial state or initial guess for any unknown. The gain feedback is given with the same relationship as in the introduction $K = YX^{-1}$. From this development and formulation it is easy to write a proper LMI design for robust control, here the main inequality has to be repeated for all the different plants making the system robust for uncertainties. This formulation is shown here:

$$\begin{aligned} \underset{X\succ 0,Y}{Max} \quad & \gamma I \\ such\ that \quad & \gamma I \prec X \end{aligned} \tag{3.21}$$

$$\begin{pmatrix} -Q^{-1} & 0 & X \\ 0 & -R^T & Y \\ X^T & Y^T & A_i X + XA_i^T + B_i Y + Y^T B_i^T \end{pmatrix} \prec 0$$

$for\ all\ i$

where the scalar γ is multiplied with the identity matrix to make the design general. The $i's$ corresponds to all the different systems which the robust controller must be able to control. This means that for every time a parameter can change it will correspond to a different system which must be considered in the controller design. Using the benefit of LMI algorithms this formulation of the problem becomes convenient and computationally inexpensive and by the above formulation robustness can be studied. Formulation 3.21 will be used in the following approach of this method to a 2-mass spring system. [Thomas Conord]

3.3 Two-mass Spring Dash pot System

In this section two goals will be archived, first of all the LMI method will be illustrated on the system and second the computational development for calculating the solution will be showed and compared for Yalmip and the MATLAB LMI toolbox. Also the results will be verified with the MATLAB LQR toolbox.

3.3.1 The Two-mass Spring Dash pot System Model

The problem discussed here is the uncertain floating oscillator:

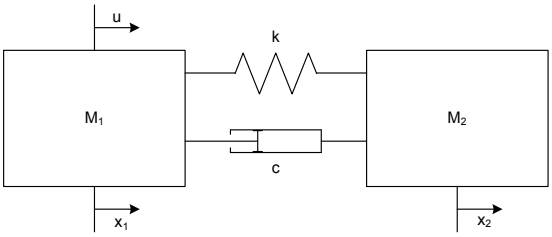


Figure 3.1: Illustration of the 2-mass spring system model.

In order to develop this system for a robust controller the parameters m_1, m_2, k and c are assumed to be uncertain. The domain of uncertainty is defined to be $\pm 30\%$ of their nominal values. The nominal values are given here as $m_{1_0} = 1$, $m_{2_0} = 1$, $k_0 = 1$ and $c_0 = 0.1$. As said in the introduction the system has to be convex, therefore the bounds from the uncertain parameters has to be covered entirely by the convex hull. Since there are no particular rule for constructing the convex hull, it depends on the number of uncertain variables and there appearance in the state-space matrices with different expressions. This can be seen by describing the system by a state-space model.
The system differential equations is given by the Newton dynamics principle.

$$\begin{aligned} m_1\ddot{x}_1 &= -k(x_1 - x_2) - c(\dot{x}_1 - \dot{x}_2) + u \\ m_2\ddot{x}_2 &= -k(x_2 - x_1) - c(\dot{x}_2 - \dot{x}_1) \end{aligned} \tag{3.22}$$

From the dynamic equations (describing the acceleration of the two masses) the state-space model can be derived.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{3.23}$$

writing out the state-space model:

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k}{m_1} & \frac{k}{m_1} & -\frac{c}{m_1} & \frac{c}{m_1} \\ \frac{k}{m_2} & -\frac{k}{m_2} & \frac{c}{m_2} & -\frac{c}{m_2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m_1} \\ 0 \end{bmatrix} \begin{bmatrix} Kx \end{bmatrix} \tag{3.24}$$

$$y = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

the two weight matrices Q and R corresponding to the LQR design parameters are taken arbitrarily and for generality as identity matrices. For a more specific design the designer can adjust the design by changing the parameters in the two weight matrices, where the parameters in the Q matrix define the weight of the system states and the parameters in the R matrix define the weight of the input signal.
Depending on how the uncertainties are handled for the LMI algorithm the design can be adjusted between robustness and performance. Since the goal is to show the benefit of the LMI method for robust controller design the following simulations have been done for a large convex hull compared to the amount of parameters in the system. The convex hull has been defined such that all different expressions of the uncertainties appearing in the state-space model are considered as independent uncertain parameters. This means that the system contains 16 vertices, which here will be a $\pm 30\%$ uncertainty for the 4 parameters, m_1, m_2, m_1 and m_2 which gives $2^4 = 16$ possible combinations.
Defining a smaller convex hull will result in a controller with better performance, but in this example the goal is to show the effective robustness of the feedback design with the LMI method. Therefore this relative wide convex hull is used in order to get a significant difference between the nominal design using LQR and the robust design by the LMI method.

3.4 Robust Controller Design

In this section the design and implementation of the LMI formulation from equation 3.21 on page 19 is discussed and illustrated. In this particular case the implementation and illustration is based on the 2 mass spring dash pot system, however the design method is very general and would be easy to implement to other systems. In this design case it is assumed that a state space model of the system is available or can be derived and also it is assumed that all parameters in the system is measurable, however with uncertainties.

3.4.1 Design Parameters

In this design method the design parameters is the LQR weight matrices and the uncertain parameters in the system and there amount of uncertainty. If the parameter which is expected to have uncertainties is known then the designer has to decide or measure the amount of uncertainty that can be expected for the parameters. These considerations and measurement corresponds to the design parameters for the robust controller, these will be enclosed as constraints to the LMI algorithm and thereby give the controller robust performance.

3.4.2 Uncertainties

Using the formulation in 3.21 and Yalmip corresponds to a nice and almost strait forward design. However certain steps in the design becomes interesting for a closer look. The first step for consideration will be the definition and implementation of the uncertainties into the design.

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{\Delta k}{\Delta m_1} & \frac{\Delta k}{\Delta m_1} & -\frac{\Delta c}{\Delta m_1} & \frac{\Delta c}{\Delta m_1} \\ \frac{\Delta k}{\Delta m_2} & -\frac{\Delta k}{\Delta m_2} & \frac{\Delta c}{\Delta m_2} & -\frac{\Delta c}{\Delta m_2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\Delta m_1} \\ 0 \end{bmatrix} [Kx] \tag{3.25}$$

since it is pretended that the system parameters are uncertain by an amount of $\pm 30\%$ the deltas in equation 3.25 represent that the parameter are changing over time. Since there is 16 combinations as mentioned there will also be 16 corresponding state space models for the system. This means that the $i's$ in the LMI formulation in equation 3.21 equals $1:16$, so when the robust controller is calculated the LMI algorithm will be exposed to one constraint for each different system meaning every time a parameter is changing. By adding all these constraints and the general ones from the final LMI formulation the LMI algorithm has all the needed information to calculate a robust controller which will have the described performance for all the uncertainties in the system.

$$\begin{aligned} \underset{X\succ 0,Y}{Max} \quad & \gamma I \\ such\ that \quad & \gamma I \prec X \\ & \left[\begin{pmatrix} -Q^{-1} & 0 & X \\ 0 & -R^T & Y \\ X^T & Y^T & A_{i=1:16}X + XA_{i=1:16}^T + B_{i=1:16}Y + Y^T B_{i=1:16}^T \end{pmatrix} \right]_{i=1:16} \prec 0 \\ for\ all\ i \end{aligned} \tag{3.26}$$

a nice way to construct the uncertainty structure is to make a binary string which describes all the combinations of uncertainties. After setting up the above scheme eventually all the constraints can be added to the system. For further description of the implementation see appendix B on page 63.

3.4.3 Nominal Design Results

As mentioned in the design formulation the LMI calculated controller is designed with LQR performance, therefore the MATLAB LQR toolbox is used to calculate a reference controller using the standard Riccati equations for solving the controller gain. Using the MATLAB LQR command the following controller gain is archived:

$$LQR_{gain} = (1.4458 \quad -0.0316 \quad 1.9345 \quad 1.1874) \tag{3.27}$$

The corresponding eigenvalues for the closed loop system with gain feedback is:

- $-0.3873 \pm 1.3587i$
- $-0.6799 \pm 0.4961i$

Similarly the LMI routine has been checked by only calculating the nominal feedback for the nominal system. Using the MATLAB LMI toolbox return the following feedback gain:

$$LMI_{gain} = (1.4463 \quad -0.0320 \quad 1.9375 \quad 1.1881) \tag{3.28}$$

with this gain the the corresponding eigenvalues for the closed loop system becomes:

- $-0.3873 \pm 1.3579i$
- $-0.6815 \pm 0.4948i$

which is hardly different from the LQR results. However a higher cost must be expected.

Then finally the nominal feedback gain is calculated using the Yalmip toolbox:

$$LMI(Yal)_{gain} = (1.4458 \quad -0.0316 \quad 1.9345 \quad 1.1874) \tag{3.29}$$

as can be seen from the results the gain calculated by the LQR method and the LMI method using Yalmip correspond to exactly the same gain, however using long format in MATLAB would show difference at the fifth exponent. Also the corresponding eigenvalues are almost identical. From these results it can be concluded that a LMI method for designing a controller with LQR performance is archived.

3.4.4 Robust Design Results

The robust controller is designed adding constraints for the hole convex hull. This means the gain is calculated for the 16 vertices. Control gain using Yalmip.

$$LMI(Yal)_{gain} = (8.9432 \quad -5.8254 \quad 6.3666 \quad 4.8151) \tag{3.30}$$

the robust gain have the following eigenvalues for the closed loop system

- -4.3717
- $-0.8915 + 0.9679i$
- $-0.8915 - 0.9679i$
- -0.4119

Using the LMI toolbox in MATLAB

$$LMI(Matlab)_{gain} = (8.9598 \quad -5.8341 \quad 6.3845 \quad 4.8254) \tag{3.31}$$

and have the following eigenvalues for the closed loop system

- -4.3957
- -0.8887 + 0.9688i
- -0.8887 - 0.9688i
- -0.4114

Thereby the two methods Yalmip and MATLAB LMI corresponds to very similar results. The traditional LQ controller is limited to the nominal design; hence the nominal LQR gain is also used in the following test with uncertainties. Since the results using Yalmip compared to the results using the LMI toolbox in MATLAB are very similar only Yalmip will be used in the following work.

3.4.5 Controller Gain

After setting up all constraints for the design, the LMI algorithm is executed and the robust gain is calculated. From section 3.2.1 on page 18 it is known that the feedback gain can be calculated as:

$$K = YX^{-1} \tag{3.32}$$

from equation 3.32 the feedback gain for the closed loop system is calculated so now this gain is implemented to the system and the system is ready for simulations.

3.5 Test Case

The 2-mass spring-dash pot system represents a vibrating system with respect to a given input. To test the designed controller the system will be given a impulse input to start the vibration in the system. The controller performance will then be evaluated depending on it's ability to bring the system to rest again with respect to time.

3.5.1 Nominal System

The system without uncertainties is simulated with a impulse input and only the nominal parameters. Also the cost function is calculated in order to verify the performance between robust and nominal controller design. Equation 3.33

$$J = trace(CPC^T) \tag{3.33}$$

is used to calculate the cost function for a LMI and LQR designed controller. Equation 3.33 is the sum of the diagonal elements of the matrix CPC^T where the matrix P is given in the LMI formulation as the inverse of the matrix X. $P = X^{-1}$.

3.5.2 Robust System

In the system with uncertainties the system is simulated for all 16 vertices corresponding to 16 different systems as defined in the LMI formulation. The test case can be defined by the following table:

	Design	Uncertainty
1	LQR	$\pm 30\%$
2	Yalmip	$\pm 30\%$
3	LQR	$\pm 15\%$
4	Yalmip	$\pm 15\%$

Here also the cost function will be calculated for the robust controllers. And in order to compare the two design methods Yalmip and LMI toolbox the computational time for the robust feedback gain is also calculated using the *tic toc* command in matlab.

3.6 Simulations

To verify the theory and the performance obtained with the calculated controllers the 2-mass spring dash pot system is simulated with different controllers inserted. To insert a feedback controller the system is changed to a closed loop system where the state matrix becomes:

$$A_{cl} = A + B(-K) \tag{3.34}$$

since it's a 2 mass spring dash pot system a impulse simulation is a very advantageous way to show performance and the reaction of the system. The system is simulated for both the nominal system without uncertainties and for the system with uncertainties. To verify the controllers the system is simulated for both the LQR controller and for the LMI designed robust controller.

3.6.1 Results for Nominal System

In the following simulations the nominal system is simulated with the three feedback gains from the nominal design.

As can bee seen from figure 3.2 the three controller design is very similar since there is almost no difference in the impulse plot. The quadratic cost for the Yalmip controller is: $J_{Yalmip(LMI)} = 2.6418$ and for the MATLAB(LMI): $J_{Matlab(LMI)} = 2.6473$ which is very close to the MATLAB LQR cost: $J_{Matlab(LQR)} = 2.6418$. The three results is very similar however with a small benefit for the Yalmip and LQR design. The difference in the quadratic cost function was somehow expected. In the simplification of the system formulation it was mentioned that the result might not reach the minimum for the quadratic cost because the LMI formulation correspond

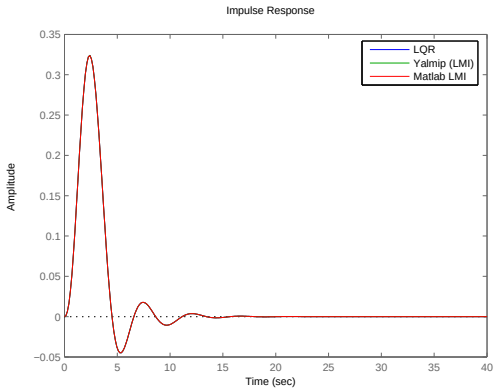


Figure 3.2: Impulse response for the nominal system with all three controllers.

to a minmax formulation and thereby a worst case design. However compared with the LQR design the two methods actually correspond to the minimum cost.

3.6.2 Results for Robust Control

Now the uncertainties will be taken into consideration. In figure 3.3 a plot of the impulse response from the closed loop system with the LQR found in equation 3.27 is shown:

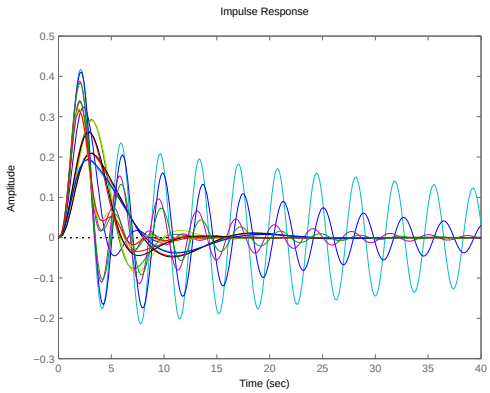


Figure 3.3: Impulse response for the convex hull with LQR design.

From the plot it is obviously that the nominal feedback design with a LQR lack for uncertain parameters in the convex hull. In this worst case situation where the system is exposed to the hole convex hull with uncertainty $\pm 30\%$ the LQR design proves to lack in robustness while the settling time expands above 40 seconds. Therefore a robust controller is designed using the formulation in equation 3.21.

with this robust controller the equivalent simulation of the impulse response for all vertices and with uncertainty $\pm 30\%$ are showed here:

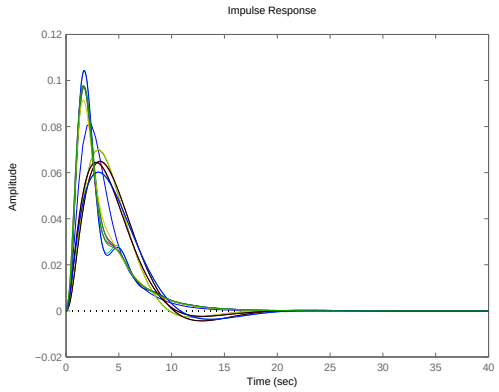


Figure 3.4: Impulse response for the convex hull with a robust controller using the Yalmip LMI method.

From the impulse response plot in figure 3.4 it can be seen that the LMI approach results in a really efficient robust controller. As can be seen all the exposed vertices is controlled very well and there is almost no oscillations. The system is stabilized after 20 seconds and the amplitude is lower then with the LQR approach. In figure 3.5 a comparison of the robust controller and the LQR design is shown:

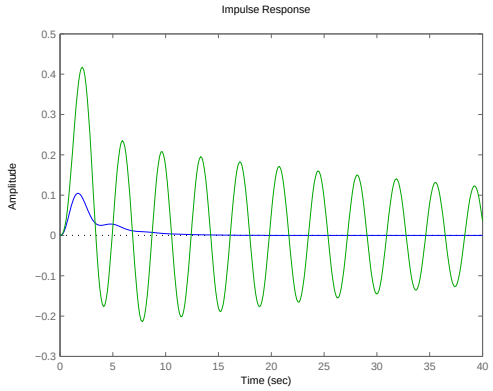


Figure 3.5: Here the two methods is compared for the worst vertices.

The nominal feedback design cannot control these vertices very well actually the system is not stabilized after more then 180 seconds. The worst vertices happens to appear when the parameter $\frac{k}{m_1}$ and $\frac{k}{m_2}$ has there maximum value and the other parameters $\frac{c}{m_1}$ and $\frac{c}{m_2}$ has there minimum values. However the robust controller stabilizes the system at almost 15 seconds. Since this combination has such a huge impact other simulations will be done where the amount of uncertainty is less. The quadratic cost function for the robust controller: $J_{Matlab(LMI)} = 22.8715$ and $J_{Yalmip} = 21.7028$ which is very similar. However here again the Sedumi powered Yalmip LMI approach benefit in having the lowest cost function. A possible reason can be a difference in the definition of the convex hull in the two design methods. More significant is the difference compared with the cost for the nominal design, as expected the robust controller corresponds to a higher cost. As described in the test case the compu-

tational time is also simulated in order to compare performance the two design methods. The computational time using Sedumi in Yalmip is $2.2859s$ where the computational time for the MATLAB lmi toolbox is $4.3839s$ which corresponds to a good performance benefit for the Yalmip method.

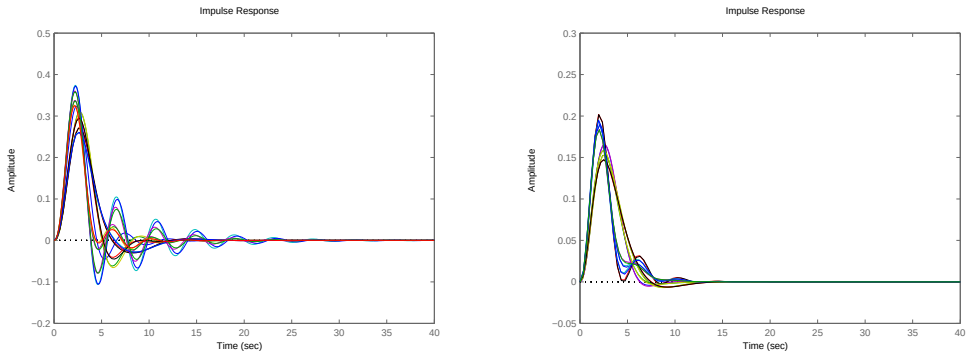
3.6.3 Simulation with less Uncertainty

The above results give a good example for the benefit that can be derived by having a robust controller for a system with uncertainties. However to take a more realistic example the same system and the same controllers are simulated with a system where the uncertainties are set to $\pm 15\%$ of there nominal values. The system is build and simulated as described in 3.4 with the new amount for the uncertainties. The feedback gain calculated using Yalmip becomes:

$$LMI(Yalmip)_{gain} = (3.3757 \quad -1.3609 \quad 3.4025 \quad 2.1994) \tag{3.35}$$

where the cost is $J_{Yalmip(LMI)} = 6.3431$.

First a impulse responds for the closed loop system with the LQR inserted.



(a) Impulse responds for the LQR design with ± 15 parameter uncertainty.

(b) Impulse responds for the LMI design with ± 15 parameter uncertainty.

Figure 3.6: Simulation with ± 15 parameter uncertainty for both LQR and LMI controller design.

From figure 3.6(a) the true robustness which is embedded in the LQR design comes to function and it is seen that the LQR design really controls the system with the damped uncertainties. However the settling time is still slower then for the system with the robust controller as can be seen from figure 3.6(b). Again the system is simulated for the worst vertices:

still the robust controller benefit from the LQR design by no oscillation and a faster settling time. However one has to consider the higher cost for a robust controller so the designer has to make a decision between a robust controller which has a high effect for uncertainties in the system and a LQR design with less robustness but a lower cost.

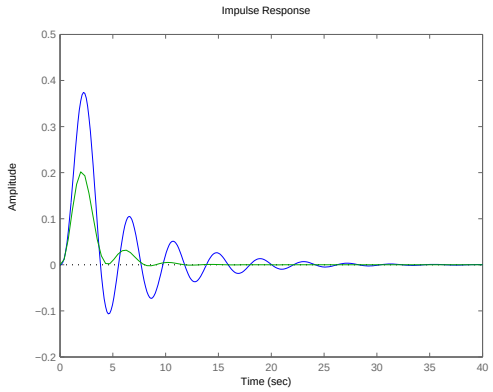


Figure 3.7: Comparing the two methods for the worst vertices.

3.7 Conclusion

The goal of developing a LMI method to calculate a controller with LQR performance was archived. A worst case design reducing the standard quadratic cost in equation 3.3 has been accomplished. In the design of a controller for the nominal system the LMI method can not be guarantied to be the optimal solution. However in the above simulation the cost for the nominal system is almost identical for all three solutions with the lowest cost by the LQR and Yalmip method.

When dealing with a uncertain system the LMI approach presents a easy and efficient way of defining and code the task. Moreover this approach is not depending on any initial guess or initial state and is not computational expensive. The LMI approach results in significant better performance than by the LQR controller. The system with uncertainties is stabilized within a short time period compared to the LQR performance. Even with fewer uncertainties the LMI method result in significant better performance for stabilizing the system.

However the designer has to take into consideration that the cost for a robust controller is higher that for the nominal design. This relay on the specific design criteria and how robust the system shall be, it is however easy by this method to change the amount of robustness.

In comparing the Sedumi powered Yalmip solution to the MATLAB toolbox there is benefit in using Yalmip. The implementation and design appear to be strait forward in Yalmip where the implementation using the LMI toolbox appear more strictly and not really strait forward. Also the Yalmip approach happened to have a better algorithm in Sedumi since a lower cost is archived and the computational time is significant lower.

Feedforward design

4

To extent the research in Linear Matrix Inequalities and its application in system and control design this chapter will explore different LMI approach. In the previous chapter a LMI method for robust feedback design was developed and implemented to a system. In this chapter a LMI formulation for feedforward design will be derived and implemented to a system. In feedforward design the LMI tool is used as an optimization solver, this means that the goal here is to derive a LMI formulation for optimization problems. This chapter will be focusing explicit on feedforward design, therefore only feedforward design will be considered for problem solutions.

4.1 Feedforward LMI formulation

In feedforward design the input to a system is shaped such that the residual energy at the end of the maneuver is minimal. The feedforward technique can be applied to any kind of linear time invariant (LTI) systems. This dos not depend on the system being stable or unstable either if the system is a open or closed loop system. Assume the system is on the general state-space model form:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= C(x) + Du(t) \end{aligned} \tag{4.1}$$

The system input can be shaped by inserting a time delay filter after the reference input such that the Laplace transform of the input becomes:

$$u(s) = \frac{1}{s}(A_0 + A_1 e^{-sT_1} + A_2 e^{-sT_2} + \cdots + A_N e^{-sT_N}) \tag{4.2}$$

this is illustrated in figure 4.1

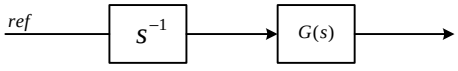


Figure 4.1: Illustration of feedforward design with shaped input.

The design variables in this approach becomes the amplitude of the incrimination A_i and the time delays T_i where $i = 0 : N$ and N is the final incrimination. Where A_i is relative to the control input to the system. This means that the optimal solutions correspond to the definition of these variables. However the first challenge for consideration is to formulate

the problem as a LMI and since the time delay do not appear linearly in the input, the time delay have to be fixed. Where $T_i = i \cdot T_s$ and T_s is a given sampling time.

4.1.1 Problem Formulation

The design purpose based on the above introduction is to minimize the residual energy of the system at the final time T_N with respect to the coefficients $A_0, A_1, \cdots, A_N$:

$$\underset{A_0,A_1,\cdots A_n}{Min} \quad J = (x(T_N) - x_f)^T Q(x(T_N) - xf) \tag{4.3}$$

here $x(T_N) - xf = \Delta x$ where x_f is the desired final state and $x(T_N)$ is the actual state of the system at the desired final time $T_N = N \cdot Ts$, Δx is the state difference. The weighting matrix represents some potential or kinematic energies in the system depending on the nature of the system. Constraint for the system can be applied by forcing the variables A_i to satisfy different conditions. These conditions can be $[A_i > 0]$ for all $i's$ which makes the input strictly increasing or $[1- < A_i < 1]$ for all i which will force the effective value of the control input $u(t)$ between -1 and 1 for all time t.

LMI formulation

Now the problem comes to transform the above optimization problem into a LMI formulation. The first step is to convert the quadratic cost into a LMI using the Schur complement. The problem can be defined as minimize the quadratic cost J by minimizing a scalar γ since $J < \gamma$. Now the optimization problem 4.3 can be defined as:

$$\begin{aligned} &\underset{A_0,A_1,\cdots A_N}{Min} \quad \gamma \\ &subject\ to \quad \gamma - \Delta x^T Q \Delta x \succ 0 \end{aligned} \tag{4.4}$$

which by using the Schur complement is equivalent to:

$$\begin{aligned} &\underset{A_0,A_1,\cdots A_N}{Min} \quad \gamma \\ &subject\ to \quad \begin{bmatrix} \gamma & \Delta x^T \\ \Delta x & Q^{-1} \end{bmatrix} \succ 0 \end{aligned} \tag{4.5}$$

this equivalence is however only valid if and only if $Q \succ 0$ meaning that Q has to be a non singular matrix. Equation 4.5 represents a LMI as long as Δx is an affine function of the

magnitude of the input increments A_i. The superposition principle can be used to confirm that this is a LMI formulation since the system is linear. Superposition:

For a linear homogeneous ordinary differential equation, if $y_1(x)$ and $y_2(x)$ are solutions, then so is $y_1(x)+ y_2(x)$. [Mathworld]

Here it means that the response of a linear system to a linear combination of different inputs is the same linear combination of the responses to each different input. Therefore by the superposition principle $x(T_N)$ is a linear combination of the magnitudes A_i and $\Delta x = x(T_N) - x_f$ is an affine function of these magnitudes. It also constitute that problem 4.5 is a LMI where the magnitudes A_i are the variables. However Δx has to be expressed analytically in function of these parameters.

State variance Expression

In the following the direct decomposition of the final state will be used in order to express the deflection in the system Δx. The approach starts from the integral expression of the state-space model:

$$\begin{aligned} \Delta x &= x(T_N) - x_f \\ &= e^{AT_N}x(0) + \int_0^{T_N} e^{A(T_N-t)}Bu(t)dt - x_f \end{aligned} \tag{4.6}$$

this occurs if the initial state is zero. Let $x(0) = 0$, and $\tilde{A}_i = A_0 + \cdots + A_i = u(t)$ where $t_i \in [i \cdot T_s; (i+1) \cdot T_s]$ for all i between 0 and N where N represents the number of input steps. The new magnitudes $\tilde{A}_i$, which represents the effective value of the input on each interval, is a easier way to express the boundary control of the input. With the previous expression the constraint has to be defined for the sum of every single magnitude A, now the same constraint can be applied simply by:

$$-1 < \tilde{A}_i < 1 \quad for\ all\ i. \tag{4.7}$$

Now by summarizing all integrals for each interval equation 4.6 becomes:

$$\Delta x = \sum_{k=0}^{N-1} \left(\int_{k \cdot T_s}^{(k+1)T_s} e^{A(T_N-t)} B\tilde{A}_k \, dt \right) - x_f \tag{4.8}$$

remind that $T_i = i \cdot T_s$ for all i between 0 and N then by changing the variable t into $\tau = t - k \cdot T_s$ equation 4.8 is equivalent to:

$$\Delta x = \sum_{k=0}^{N-1} \left(\int_0^{T_s} e^{A((N-k)T_s-\tau)} B \, d\tau \right) \tilde{A}_k - x_f \tag{4.9}$$

final by using the fact that $\tilde{A}_k$ is a scaler:

$$\Delta x = \left(\sum_{k=0}^{N-1} \tilde{A}_k e^{A(N-k-1)T_s} \right) \left(\int_0^{T_s} e^{A(T_s-\tau)} B \, d\tau \right) - x_f \tag{4.10}$$

Now the magnitudes of the input appear as a affine function in the expression of Δx. Since the transformation which transforms A_i into $\tilde{A}_i$ is linear the expression of the final state is also a linear combination of the new magnitudes $\tilde{A}_i$. Thereby problem 4.5 on page 31 is still a LMI for all the $\tilde{A}_i$ parameters.

4.2 Design and Implementation

In this section the design and implementation of the LMI formulation for feedforward design will be described for a state-space system.

4.2.1 2-mass Spring Dash pot System

The 2-mass spring dash pot system appear to be a very nice model for illustration of the feedforward approach. The 2-mass spring system has already been explored in the feedback approach, however in the feedback approach the goal was to stabilize a system, the goal in the feedforward design approach is to control a system maneuver by minimizing the residual energy and maintain constraints.

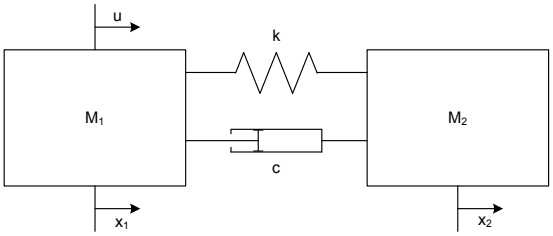


Figure 4.2: Illustration of the 2-mass spring system model.

Since the goal is to minimize the residual energy for a given maneuver of the system the system description is reformulated with respect to energy. The system differential equations is given by Newton dynamics principles:

$$\begin{aligned} m_1\ddot{x}_1 &= -k(x_1 - x_2) - c(\dot{x}_1 - \dot{x}_2) + u \\ m_2\ddot{x}_2 &= -k(x_2 - x_1) - c(\dot{x}_2 - \dot{x}_1) \end{aligned} \tag{4.11}$$

the system equation in 4.11 is equivalent to the form:

$$M\ddot{\tilde{x}} + \beta\dot{\tilde{x}} + K\tilde{x} = \tilde{B}u \tag{4.12}$$

where M, β and K are given as:

$$M = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix} \quad \beta = \begin{pmatrix} c & -c \\ -c & c \end{pmatrix} \quad K = \begin{pmatrix} k & -k \\ -k & k \end{pmatrix} \tag{4.13}$$

and $\tilde{x}$ and $\tilde{B}$ as:

$$\tilde{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \tilde{B} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{4.14}$$

from the system description in equation 4.13 and 4.14 the state-space description for the system can be found:

$$A = \begin{pmatrix} 0 & I_{2:2} \\ -M^{-1}K & -M^{-1}\beta \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ \tilde{B} \end{pmatrix}$$

$$C = (0\,0\,0\,0) \qquad D = (0)$$

the state vector x is $(x_1, x_2, \dot{x}_1, \dot{x}_2)^T$. Now the weight matrix can be defined with respect to the energy in the system and the energy in system maneuver. The weight matrix Q will now by the definition of the residual energy and is therefore constructed as:

$$Q = \begin{pmatrix} K & 0 \\ 0 & M \end{pmatrix} \tag{4.15}$$

where K corresponds to the potential energy here the position of the system and M corresponds to the kinetic energy of the system the velocity of the system. This can be further described by recalling the residual energy function:

$$x^T Q x \tag{4.16}$$

by writing out equation 4.16:

$$(x_1\ x_2\ \dot{x}_1\ \dot{x}_2)^T\, Q \begin{pmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} \tag{4.17}$$

which can be rewritten as:

$$(x_1\ x_2)\,K\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (\dot{x}_1\ \dot{x}_2)\,M\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} \tag{4.18}$$

by equation 4.16 this is approximately equal to $\frac{1}{2}K[x_1+x_2]^2 + \frac{1}{2}M[\dot{x}_1+\dot{x}_2]^2$ which represent the elastic potential energy and kinetic energy of the system.

Non-singular Weight matrix

By equation 4.5 on page 31 the weight matrix Q has to be non-singular however the defined weight matrix is a singular matrix. The weight matrix can be singular and invertible by adding another potential energy, here the rigid body part of the system which will be the potential energy of m_1 with respect to the origin. This corresponds to add the potential energy description $m_1x_1^2$ to the residual energy $J = \Delta x^T Q \Delta x$. Thereby the new weight matrix becomes:

$$Q_{new} = Q + \begin{pmatrix} m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ & & 0_{2\times 4} & \end{pmatrix} \tag{4.19}$$

In the simulation of the system the system parameters will be set equal to one ($m_1 = m_2 = k = 1$) and the damping coefficient $c = 0.1$.

4.3 Feedforward controller Design

In the following section the implementation of problem formulation 4.5 on page 31 will be described. The variable $[\gamma > 0]$ is implemented by constructing a scalar variable and insert the constraints. Calculation of Δx is a greater challenge: First of all the variables $\tilde{A}_i$ is constructed as a vector of N variables corresponding to the number of control steps which is a design criteria. The final state condition x_f is given by the simulation definition. The system will be simulated from the initial states, the origin, $x_i = [0\ 0\ 0\ 0]^T$ to there final state $x_f = [1\ 1\ 0\ 0]^T$. The remaining right hand side of equation 4.10 on page 33 the integral can be calculated by use of the Van Loan Identity:

For any square matrix A and any matrix B of the same number of rows, if P is the square matrix defined as follows:

$$P = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \tag{4.20}$$

then

$$e^{PT_s} = \begin{bmatrix} e^{AT_s} & \int_0^{T_s} e^{A(T_s-\tau)}Bd\tau \\ 0 & I \end{bmatrix} \tag{4.21}$$

now the integral can be calculated by taking E as the corresponding columns of B in P in the new matrix e^{PT_s}. The left hand side of equation 4.10 called the ϕ matrix:

$$\sum_{k=0}^{N-1} \tilde{A}_k e^{A(N-k-1)T_s} \tag{4.22}$$

is actually strait forward to calculate by the MATLAB command *expm*. Now by summarizing matrix ϕ for all $\tilde{A}_i$'s the explanation for Δx is derived:

$$\Delta x = \phi \cdot E - x_f \tag{4.23}$$

Now that both γ and Δx is defined the actually constraint for the overall residual energy can be defined. Remember that the objective is to minimize the residual energy at the final time T_N with respect to the coefficients $\tilde{A}_i$'s. Where T_N equals the final time T_f for the final step. This was done by minimizing γ such that $J < \gamma$. Using Yalmip the constraint

$$\begin{bmatrix} \gamma & \Delta x^T \\ \Delta x & Q^{-1} \end{bmatrix} \succ 0 \tag{4.24}$$

is given by writing it directly as it appear. Remanning is the calculations for the optimal step inputs by solving for the minimal γ. However more constraints can be applied to the system depending on the design and performance criteria. In the following section the system has been simulated with a constrain on $\tilde{A}_i$'s which is equivalent to bounded control signal.

4.4 Optimal Final Time

By using a bisection algorithm on the feedforward design a search can be done to find the optimal final time T_f for the maneuver in the feasible area. A bisection algorithm for this system works by evaluating the residual energy returned from the LMI algorithm corresponding to γ, by evaluating different maneuver time defined by $T_s = T_f/N$. If the returned $\gamma \leq 0$ the system will reach the desired position and therefore the time is feasible. A illustration is shown in figure 4.3.

where $T_f = T_s \cdot N$. Figure 4.3 illustrates the implemented bisection algorithm. The bisection algorithm is initialized by evaluating the resulting residual energy for different T_s here X_L for the lower part X_U for the upper part and X_M is given as $\left[X_M = \frac{1}{2}(X_L + X_U)\right]$. Then by evaluating the resulting residual energy γ, the bisection algorithm decide the direction for the next evaluating point and by X_M the step size. By initialize the bisection algorithm with a lover final time X_L in the infeasible area and a upper final time X_U in the feasible area the bisection algorithm will converge to the optimal final time which in this case is the minimum

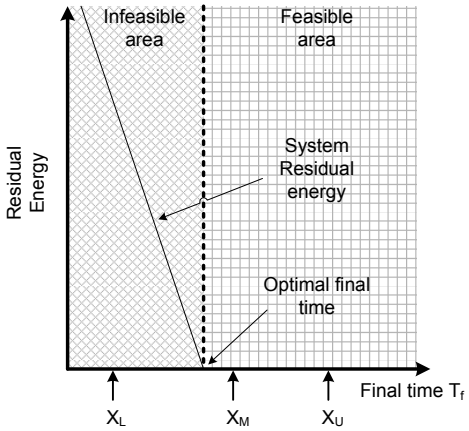


Figure 4.3: Bisection illustration.

time where the system still reach the desired position. The rate of precession is based on some converge definitions made by the designer. In order to use the bisection for final time optimization the residual energy as a function of different T_s has been plotted.

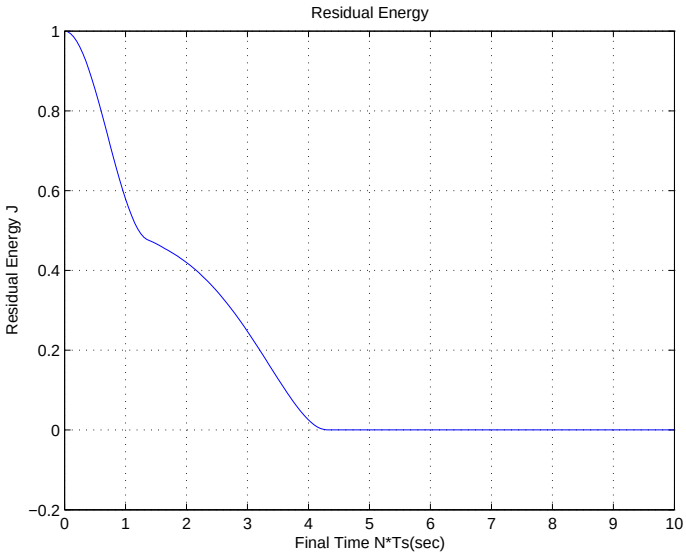


Figure 4.4: Simulation of the residual energy in function of the final time for $N = 5$.

Figure 4.4 shows the residual energy in function of the final time $T_f = N \cdot T_s$ for 5 step input. From the simulation it can be seen that the infeasible area appears for $T_f < 4.3s$ so the optimal value for T_s is expected to be around $T_s = \frac{T_f}{N}$ which is $T_s = \frac{4.3}{5} = 0.86s$. It can also be observed that the residual energy converge to one which was expected with respect to the defined weight matrix however the shape of the rasing curve is more difficult to explain. A reason for the break at 0.5 could correspond to the LMI algorithm since it has been observed that in the infeasible area the LMI algorithm happens to run into numerical problems, this however wont affect the bisection algorithm in function since the value still corresponds to the infeasible area. From the plot of the residual energy the bisection algorithm can be designed for

the system and reasonable evaluation points can be selected. [Bisection]

4.5 Deflection Constraint

Dealing with performance consideration another interesting problem to consider would be to minimize the deflection between the two masses. This means that the distance between the two masse will be stable without oscillations. In LMI formulation this performance can be added to the system by an additional constraint on the deflection between the two masses in the system. In the above problem formulation the difference between the states has already been described by Δx see equation 4.8 on page 32. However here the deflection for consideration is only between state one and two in the system and in difference from the above expression this deflection is considered with respect to each step and not to the final sate. The expression for $\Delta x_{1,2} = x_1 - x_2$ becomes.

$$\Delta x_{1,2} = \sum_{k=0}^{N-1} \left(\int_0^{T_s} e^{A(T_N - t)} B dt \right) \tilde{A}_k \tag{4.25}$$

where T_N is the time at each step $0 : N-1$ this can be expressed as $T_N = T_s(N-1) - T_s(k-1)$. It is known from equation 4.3 on page 35 that the integral can be calculated using the Van Loan Identity:

$$P = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \tag{4.26}$$

then

$$e^{PT_N} = \begin{bmatrix} e^{AT_N} & \int_0^{T_s} e^{A(T_N - t)} B dt \\ 0 & I \end{bmatrix} \tag{4.27}$$

the integra can now easily be found be taking it out from the above equation. Now $\Delta x_{1,2}$ for each step input is calculated by:

$$\Delta x_{1,2} = \Delta x \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} \tag{4.28}$$

in this approach linear algebra is used to build up the summarization of all $\Delta x_{1,2}$. Since the final constrain will be defined by bounding the deflection $\Delta x_{1,2}$ between a defined value, a matrix is build up summarizing all $\Delta x_{1,2}(1 : N)$. This can be illustrated by:

$$\underbrace{\begin{bmatrix} \Delta x_1 & 0 & 0 & 0 \\ \Delta x_1 & \Delta x_2 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 \\ \Delta x_1 & \Delta x_2 & \cdots & \Delta x_N \end{bmatrix}}_{M\Delta x} \left[\tilde{A}_k\right]_{N\times 1} < [def]_{N\times 1} \tag{4.29}$$

where $M\Delta x$ is a $N \times N$ matrix and $\tilde{A}_k$ is the vector of N variables corresponding to the number of steps. The def vector is a $N \times 1$ vector which contains the value for the deflection definition. By including the constraints by equation 4.29 for $\pm M\Delta x$ in the LMI constrains, this will force the deflection between the two masses to be within a certain area defined by def.

4.6 Robust Control

Robust control design is a major goal in this research therefore the LMI formulation developed above will now be subject for adding robust performance to the design. In this design only one parameter will be considered uncertain. A illustration of a system maneuver with respect to a uncertain parameter can be seen in figure 4.5.

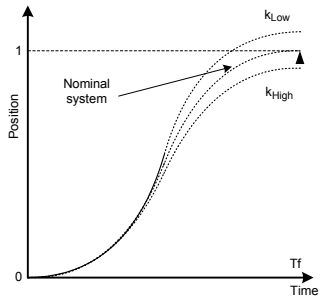


Figure 4.5: Illustration of the system maneuver for different values of a uncertain parameter.

In the illustration the uncertain parameter is the spring constant k which for a value below the nominal value causes a overshoot in the system maneuver if the final time is constant. In the case of a spring constant above the nominal value the system can not archive the correct position within the final time T_f. Since k is uncertain the system need robust performance in order to still archive the desired position within a given final time.

4.6.1 MinMax Solution

Similarly to the robust control in the feedback design a system parameter will here be chosen to be uncertain. Here the spring constant is chosen to be uncertain with a 30% area around the nominal value. This means that $k \in [0.7; 1.3]$ since the nominal value for k is one. The method is very similar to the feedback system. Every different value of k corresponds to a different system which again correspond to new constraints in the LMI design. Thereby the

robust design becomes simple, just add constraints for every new value of k to the original design for the nominal system. In the following simulations 13 values of k has been taken homogeneously in the uncertainty area 0.7 to 1.3 and for every value of k the corresponding new constraints is added to the LMI design. For implementation this means that equation 4.30 and 4.31 is calculates for $i = 1:13$ and added as constraints to the LMI algorithm.

$$\Delta x = \left(\sum_{k=0}^{N-1} \tilde{A}_k e^{A_i(N-k-1)T_s} \right) \left(\int_0^{T_s} e^{A_i(T_s-\tau)} B_i \, d\tau \right) - x_f \tag{4.30}$$

by the definition of the weight matrix in 4.19 Q is changing for every value of k therefore the new constraint for every value of k corresponds to add equation 4.31 for $i = 1:13$.

$$\begin{bmatrix} \gamma & \Delta x_i^T \\ \Delta x_i & Q_i^{-1} \end{bmatrix} \succ 0 \tag{4.31}$$

This will correspond to a robust design with respect to the uncertain range of the spring constant k.

Optimal time Consideration

A goal here will also be to find the optimal final time as described above, however this has proven to be impossible which will be further described in the simulation section 4.8.4 on page 52. However to give an idea of the problem the residual energy for the system with a uncertain parameter k is plotted in figure 4.6.

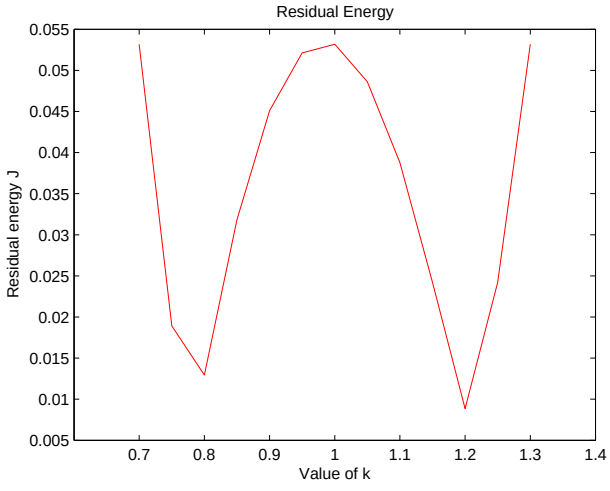


Figure 4.6: Residual energy simulation for $N = 10$ and $T_s = 0.6s$.

Since the Bisection algorithm calculate the optimal final time by finding the corresponding satisfactory minimal residual energy it is obviously that the bisection algorithm will fail for this system. Therefore the following system is developed.

4.6.2 Augmented Robust Design

In the augmented solution the state space model in equation 4.15 on page 34 is augmented with four new states. The four new states describe the system with the derivative of the uncertain parameter k. The new model states becomes:

$$x = \begin{bmatrix} x \\ \frac{dx}{dk} \end{bmatrix} \tag{4.32}$$

where $\frac{dx}{dk}$ equals the derivative of $\dot{x} = Ax + Bu$:

$$\frac{d\dot{x}}{dx} = \frac{dA}{dk}x + A\frac{dx}{dk} + \frac{dB}{dk}u + B\frac{du}{dk} \tag{4.33}$$

to illustrate the idea of adding the new states to the system see figure 4.7.

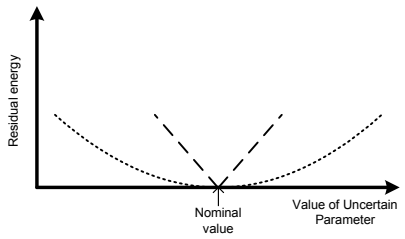


Figure 4.7: Illustration of the Augmented system.

taking the derivative of the system with respect to the uncertain parameter k and adding this to the system model corresponds to adding robust performance with respect to the uncertain parameter. This is possible if the new states are set equal to zero at the final time. The shape of the residual energy of the system will change as illustrated in figure 4.7. The residual energy will be low in a wider area of the uncertain parameter compared to the nominal design. Thereby a more robust design is archived.

Augmented Model

The new state space model of the system with the derivative states implemented becomes:

$$\begin{pmatrix} \dot{x} \\ \frac{d\dot{x}}{dk} \end{pmatrix} = \begin{bmatrix} A & 0 \\ \frac{dA}{dk} & A \end{bmatrix} \begin{pmatrix} x \\ \frac{dx}{dk} \end{pmatrix} + \begin{bmatrix} B \\ \frac{dB}{dk} \end{bmatrix} u \tag{4.34}$$

Now by setting the new final state to $x_f = [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$ will set the new state $\frac{d\dot{x}}{dk}$ to zero at the final time. Hence the sensitivity to the uncertain parameter is driven to zero at the final time. This approach will add robust performance to the time delay filter design.

4.7 Test Case

A simple and illustrative maneuver for this system could be to calculate the control input for a maneuver from the zero positions of the two masses m_1 and m_2 to position one as described above by x_f. Different constraints and optimizations problems can be discovered for this maneuver e.g. time optimizations for the maneuver or deflection optimization. The two problems can also be combined by applying multiple constraints on the system. The following test cases are defined for the simulation of the nominal system, robust min max and augmented system.

4.7.1 Nominal System

The first challenge will be to calculate the optimal step input to the system in order to move the two masses from position zero to position one. However the design allows manipulation of different parameters such as final time and amount of steps. Therefore the following test case for the nominal system design:

	$\mathbf{T_s}$	**N**
1	2	5
2	1	5
3	Optimal	5
4	Optimal	20

As described in the above test case the nominal system will first be simulated for a arbitrary values of N and T_s where T_s is the design parameter for the final time as described earlier. If the system for decreasing final time still can archive the desired final state then the bisection algorithm in 4.4 on page 36 will be used in order to find the optimal final time. The last design parameter is the amount of steps. Therefore the final test is done for $N = 20$ to see if this will cause performance improvement.

4.7.2 System with Deflection Constraint

To show the new performance of the system after implementing the deflection constraint to the LMI formulation a test case is developed. The system maneuver will be the same as described above, however here a deflection constraint is defined for the system with respect to the distance between the two masses. Test case for the 2 mass spring system with deflection constrain.

	T_s	N	def
1	2	10	0.04
2	2	10	0.03
3	$\sim$ Optimal	101	0.09
4	$\sim$ Optimal	101	$\sim$ Optimal
5	$\sim$ Optimal	101	∞

The above simulation scheme is constructed to show the benefit of adding deflection constraint. In the first two simulations the deflection is forced down from 0.04 to 0.03. The next simulations shows the same system for 101 input steps. In the deflection simulation the distance between the two masses is simulated together with the normal simulation of the system maneuver. The distance is simply calculated by simulating the system and take out $\Delta x_{1,2}$ for each step. Using *lsim* in MATLAB makes it very easy to take out the desired states.

4.7.3 Robust System

For the robust design there is derived 2 methods the min max solution and the augmented solution. Because of the optimal final time consideration in the min max solution the augmented system will be simulated first. Test case for robust augmented system:

	T_s	N
1	1	10
2	Optimal	10
3	Optimal	20

The simulation of the augmented robust system is based on the knowledge from simulating the nominal system. Finding the optimal final time by the augmented leads to the test case for the min max solution. Test case for the robust min max solution:

	T_s	N
1	1	10
2	Optimal	10
3	Optimal	20

However the above simulation will only show if the robust design can handle a system with uncertainty.

4.7.4 Comparing Robustness

To show the robust performance for the different control design the residual energy will be simulated with respect to the hole range of the uncertain parameter. The residual energy is calculated as:

$$J = x(T_f)_i^T Q_i x(T_f)_i \tag{4.35}$$

where x is at the final time. Taking the square root of equation 4.35 gives the final residual energy. The robust performance will be compared by simulating the residual energy for the nominal system, the min max solution and the augmented solution with respect to the entire area of the uncertain parameter. The simulation will be performed for optimal time calculated from the augmented system and for 10 and 20 steps.

4.7.5 Comparing MATLAB LMI toolbox vs. Yalmip and Sedumi

The simulation in this section is developed to compare the results corresponding to use MATLAB's LMI toolbox or the Sedumi powered Yalmip solution. This project is based on the great work done by Thomas Conord [Thomas Conord], however in this project the LMI formulation is derived and implemented using Yalmip as described in the introduction. A comparison of the performance by the two methods is done by the following test scheme. Compare test scheme:

	$\mathbf{T_s}$	**N**
1	Optimal	10
2	Optimal	20

Since the results from the two methods appear to be very similar the above test scheme is only considered for the Robust Min Max solution. The optimal time for the simulation is taken from Thomas report. For the test scheme the system maneuver and the residual energy is simulated. Also the computational time for the two methods is calculated using the *tic toc* MATLAB command. The simulation from the LMI toolbox is done by the code lend by Thomas after verification.

4.8 Simulations

In the following the 2 mass-spring dash pot system has been simulated with the control signal bounded $[-1 < u(t) < 1]$. The shaped input feedforward controller is implemented and simulated for the maneuver defined in section 4.2. For the first simulation the number of input steps is set to $N = 5$ and the sampling time is set to $T_s = 2s$. The LMI algorithm calculate the optimal control values between -1 and 1 of the steps of input on each interval corresponding to achieve the minimum residual energy at the final time here $T_f = N \cdot T_s = 10s$. The following plot shows the responses of the two states x_1 and x_2 with the optimal shaped input.

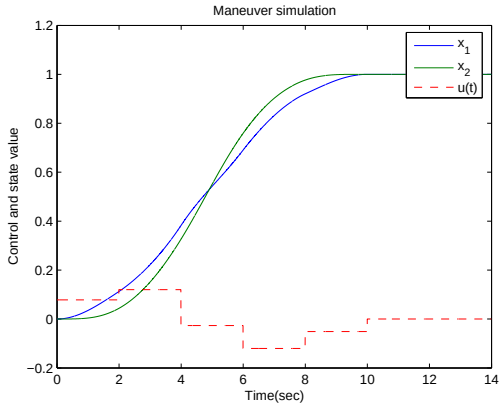


Figure 4.8: Time response of the system for the optimal shaped input with $N = 5$ and $T_s = 2s$.

From the simulation in figure 4.8 it is clear that the designed feedforward controller is efficient to control the motion of the system in the defined maneuver here from the origin to the final position $x_f = [1\,1;0\,0]$. The simulations show a stable motion of the system and a stable system after the maneuver. However the simulations also indicate that not the entire control range is used meaning that depending on the design criteria there are space for optimization. The next simulation illustrates this further:

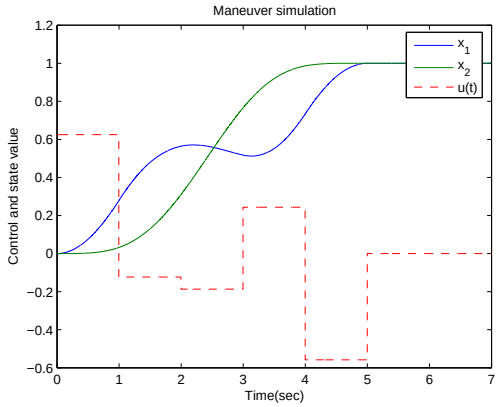


Figure 4.9: Time response of the system for the optimal shaped input with $N = 5$ and $T_s = 1s$.

the simulation in figure 4.9 clearly shows that maneuver time is decreased to $T_f = N \cdot T_s = 5s$ because T_s is set to 1 second. For further illustration of the optimization problem lets look at the next simulation.

Here clearly the entire control area are used, however the time $T_f = N \cdot T_s = 2.5s$ shows to corresponds to a infeasible solution the system can not reach the defined final state and appear to be unstable. Based on these observations it becomes interesting to search for the optimal time period for the maneuver in the feasible area.

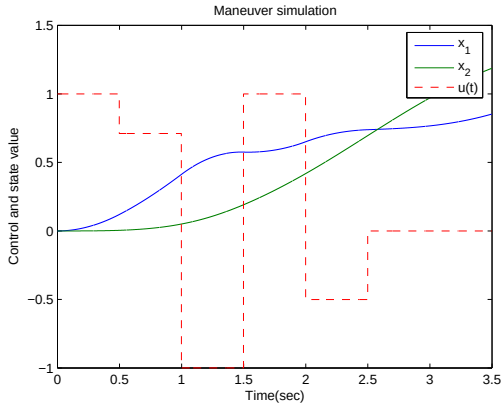
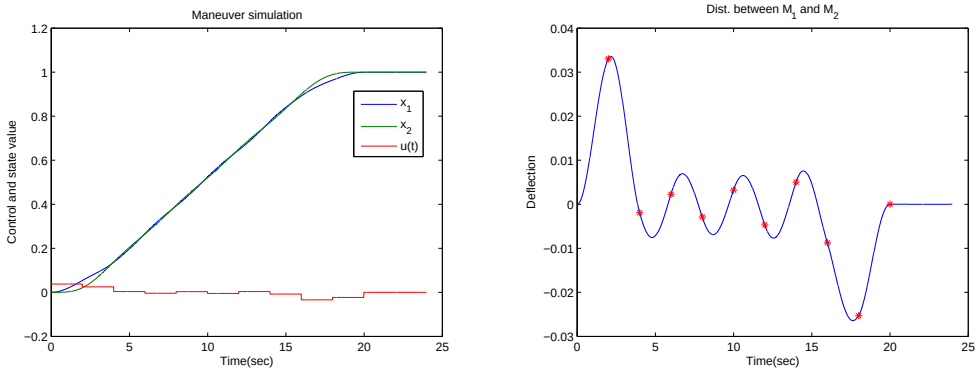


Figure 4.10: Time response of the system for the optimal shaped input with $N = 5$ and $T_s = 0.5s$.

4.8.1 Deflection Constraint included

First simulation only serves to show the reaction of adding additional constrain here the deflection constrain. In figure 4.11 the system is simulated for $T_s = 2s$ and $N = 10$ the deflection is set to $def = 0.04$.



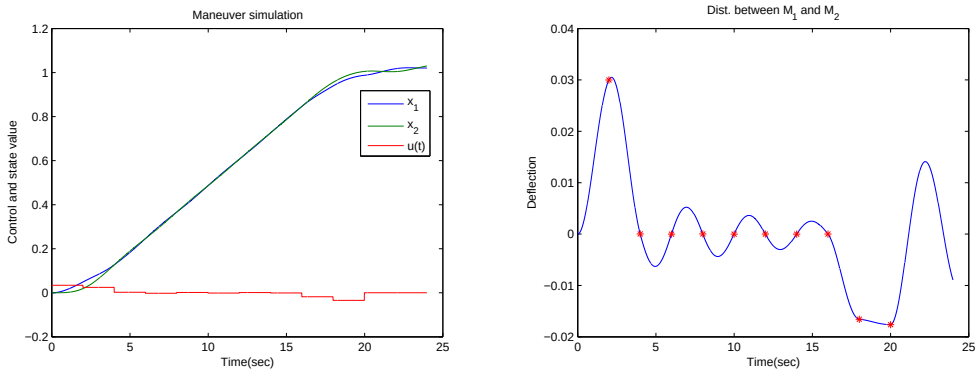
(a) System maneuver.

(b) Deflection simulation.

Figure 4.11: Plot of the system maneuver and deflection for the system with deflection constraint. $N = 10$, $T_f = 20s$ and $def = 0.04$

Observing figure 4.11 leads to strength the deflection constraint to see if it can be controlled. In the deflection simulation the stars points out the deflection at each step $1 : N$. From the deflection simulation it is chosen to set $def = 0.03$. The new system is simulated in figure 4.12.

From the simulation i appear that the constraint works and is capable to add new performance in the system. However the simulation time appear to be very long and the control input range is hardly used looking at figure 4.12(a). To further show the problem the amount of step input is increased to $N = 101$. To simulate the system very close to the optimal time the residual energy with respect to T_s is simulated.



(a) System maneuver.

(b) Deflection simulation.

Figure 4.12: Plot of the system maneuver and deflection for the system with deflection constraint. $N = 10$, $T_f = 20s$ and $def = 0.03$.

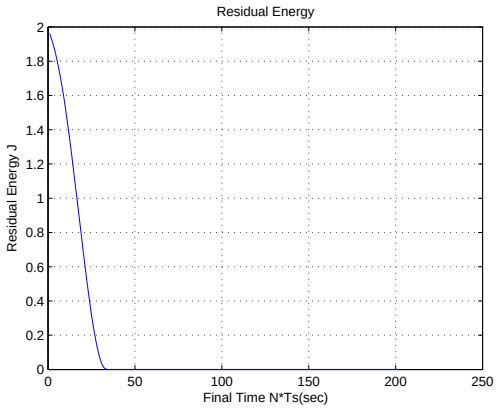


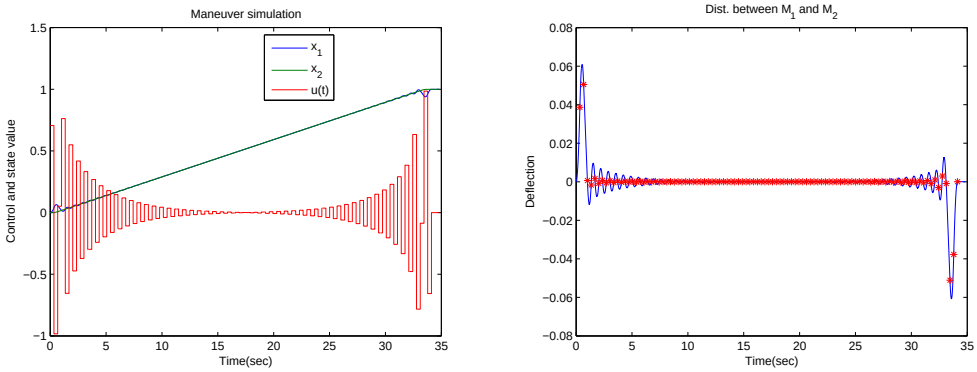
Figure 4.13: Residual energy vs. T_f. Deflection $def = 0.09$ and $N = 101$.

Form the residual energy simulation in figure 4.13 the optimal time is around $T_f = 30s$ hence the simulation time for the following simulation is taken close but above the optimal time to allow more constraint in the deflection. First simulation is done for $def = 0.09$ and the $T_f = \sim Optimal = 34.14s$

From the above simulation it appears especially from figure 4.14 that either the simulation time can be decreased or the deflection can be increased. Hence the the deflection in the next simulation will be $def = 0.059$ which should be reasonable from figure 4.14(b).

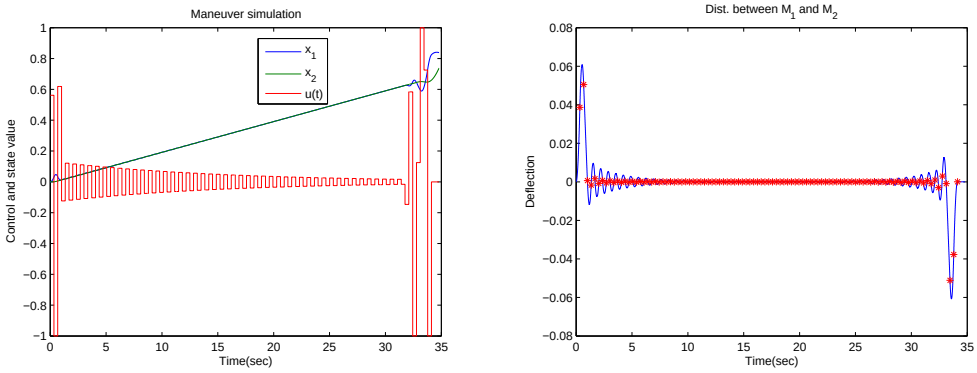
To show that there really is more interesting work to be accomplices before this performance criteria can be implemented a simulation for the system with the same final time but without the deflection constraint implemented.

Comparing figure 4.15 with 4.16 truly shows that there is some problems in the LMI definition of the deflection constraint. The only benefit of implementing the constraint is that it gives a small control of the deflection, however as can be seen from figure 4.16 the performance without the constraint is much better. The reason for this is still for consideration; different



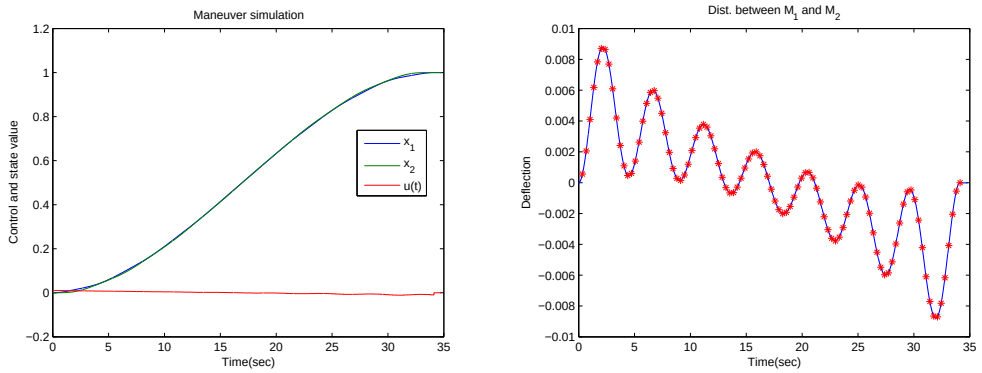
(a) System maneuver. (b) Deflection simulation.

Figure 4.14: Plot of the system maneuver and deflection for the system with deflection constraint. $N = 101$, $T_f = 34.14s$ and $def = 0.09$



(a) System maneuver. (b) Deflection simulation.

Figure 4.15: Plot of the system maneuver and deflection for the system with deflection constraint. $N = 101$, $T_f = 34.14s$ and $def = 0.059$.



(a) System maneuver.

(b) Deflection simulation.

Figure 4.16: Plot of the system maneuver and deflection for the system with deflection constraint. $N = 101$, $T_f = 34.14s$ and $def = \infty$.

approach has been developed and implemented however without luck. Another problem was numerical problems in the LMI algorithm. Changing the LMI algorithm solved this problem see below. The work is however still included since it would be interesting to archive this performance so hopefully this work can serve as a introduction.

For all simulation with deflection constrain implemented the LMI algorithm has been changed from *Sedumi* to *sdpt3* [SDPT3]. The reason for the change is that *Sedumi* for these calculations happens to run into numerical problems very often. The *sdpt3* algorithm however proves to be more stable in that sense.

4.8.2 Optimal Final Time

Using the Bisection algorithm with $X_L = 0.1$ and $X_U = 1$ for the system and fixed 5 steps the bisection algorithm converge to the optimal final time as $T_f = 5 \cdot 0.8669 = 4.3345s$. The next figure shows the simulation for $T_s = 0.8669s$.

A method to determine the time optimal control for a given number of control input steps is now developed clearly it is seen from the graph in figure 4.17 that the entire control area $[-1 < u(t) < 1]$ is used. The time responds corresponds to the bang-bang solution given by the time optimal control theory [Thomas Conord, p71]. Now that a method is developed and proved to be functioning by simulation lets increase the number of input steps to see if this can improve the time range. Increasing the fixed step input from 5 to 20 steps results in the following plot.

From the graph in figure 4.18 it is clearly to see that the system is very near it's optimal value corresponding to the constraints and performance criteria. Increasing the number of input steps dos not decrease the time range particularly, however this result was expected since the limited control area was already used. It is seen that only 8 significant steps is used and the final time is decreased to $T_f = 20 \cdot 0.2116s = 4.2320s$ an improvement of 2.4%. Increasing the number of steps further clearly wound not result in great benefit.

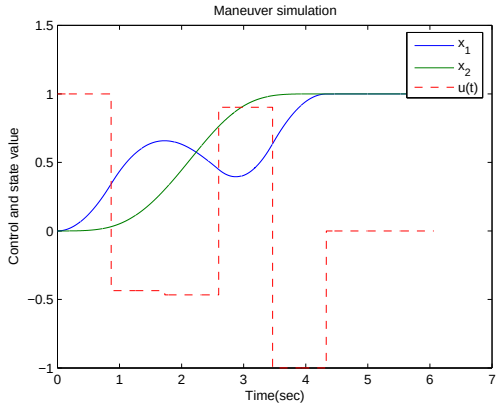


Figure 4.17: Time response of the system for the optimal shaped input with $N = 5$ and optimal time $T_s = 0.8669s$.

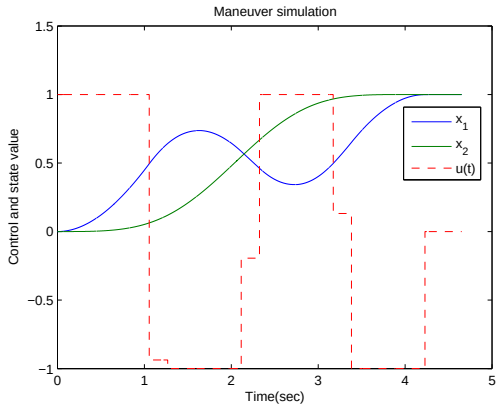


Figure 4.18: Time response of the system for the optimal shaped input with $N = 20$ and optimal time $T_s = 0.2116s$.

4.8.3 Simulation with Augmented system

The first simulation is for illustration of the augmented approach this means that the standard simulation is considered, however the spring constant is uncertain as described in the robust simulation scheme. Figure 4.19 shows the first simulation with $T_s = 1s$ and $N = 10$:

As can be seen from figure 4.19 the relative long simulation time $T_s = 1s$ result in almost similar maneuver comparing to the nominal system. However the plot shows the different system maneuvers corresponding to the uncertain parameter k. In order to find the optimal time for the next simulation the residual energy is simulated as shown in figure 4.20.

From the simulation of the residual energy the upper and lower parameters for the Bisection algorithm can be found. The optimal final time must be between 5 and 7 seconds. Hence the upper value for the bisection algorithm is set to $X_U = 0.7$ and lower to $X_L = 0.5$. By the bisection algorithm the optimal final time is found to be $T_f = 10s \cdot 0.5946s = 5.946s$. See figure 4.21.

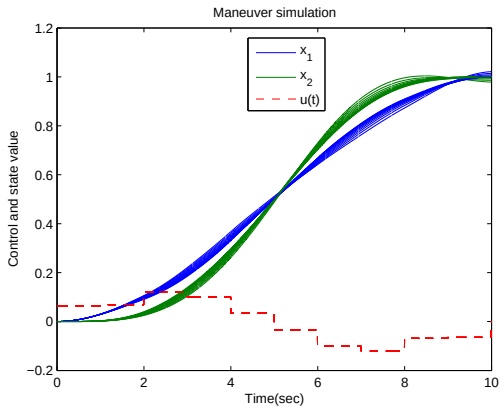


Figure 4.19: System maneuver simulation for $T_s = 1s$ and $N = 10$.

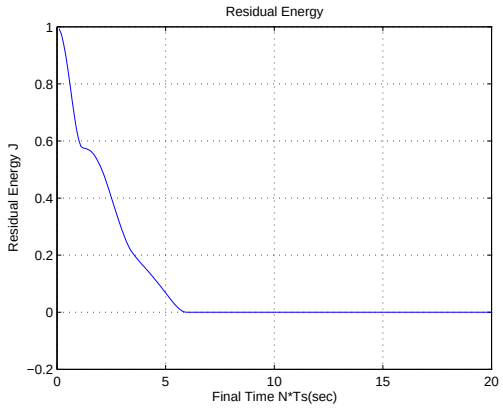
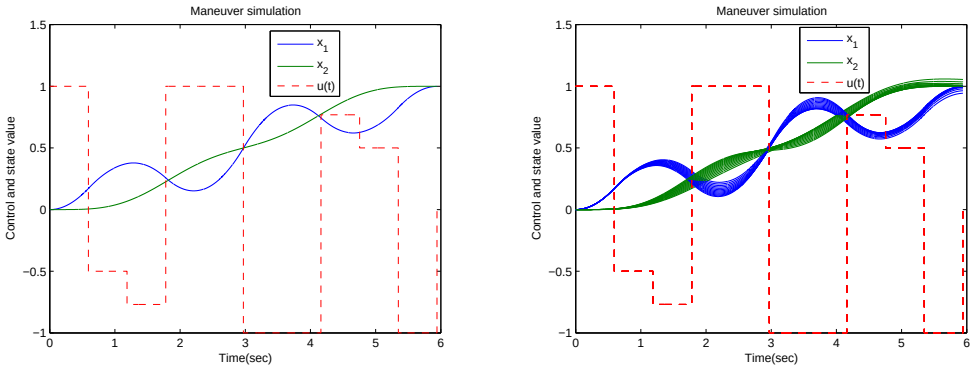


Figure 4.20: Residual energy for the Augmented system with $N = 10$.

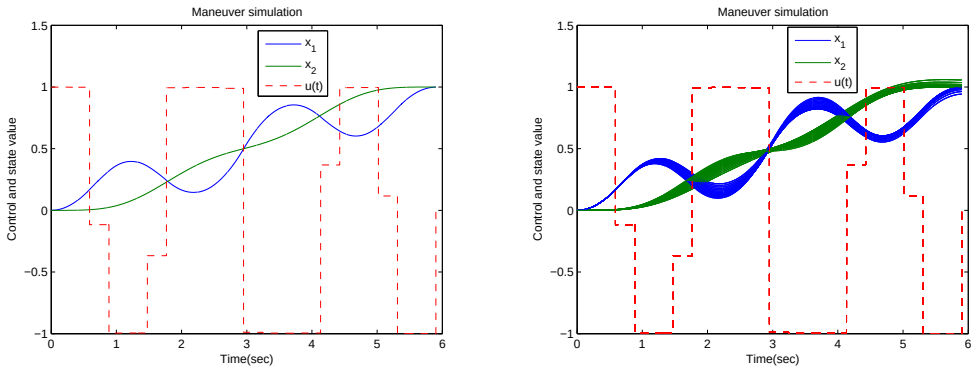


(a) Robust control with nominal system.

(b) Robust control with uncertain system.

Figure 4.21: Plot of the system maneuver for the nominal system and for the system with uncertain parameters. $N = 10$ $T_f = 5.946s$

Figure 4.21 shows the augmented control of the system maneuver with optimal time $T_f = 5.946s$ and $N = 10$. In 4.21(a) the system is simulated for the nominal value of the uncertain parameter; however the control input is calculated for the uncertain system. The control proves to be bring the system to the desired position within the final time. The control is also very close to the optimal control (bang bang control). In figure 4.21(b) the same control is applied but here the system is simulated for the defined range of uncertainty parameters. The controller proves to maneuver the system to the desired position for all values of the uncertain parameter. Like in the simulation for the nominal system it would be interesting to see the benefit of more steps in the control input. Therefore the number of steps has been increased to 20 in the next simulation. The optimal time is calculated using the bisection algorithm.



(a) Robust control with nominal system.

(b) Robust control with uncertain system.

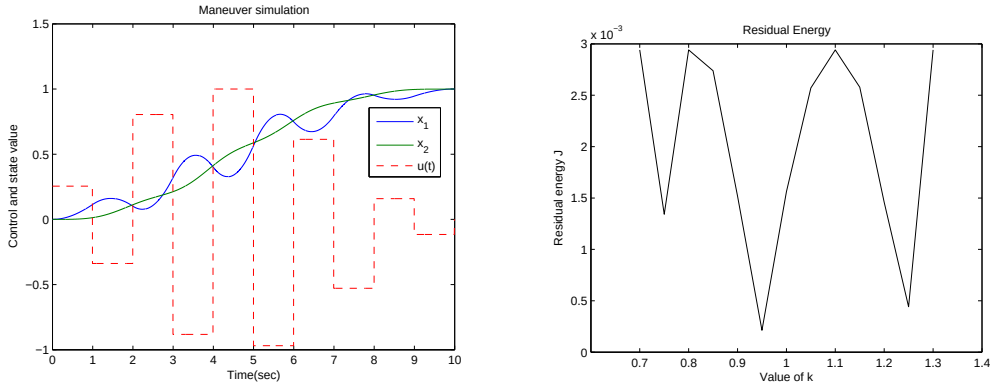
Figure 4.22: Plot of the system maneuver for the nominal system and for the system with uncertain parameters. $N = 20$ $T_f = 5.906s$

The simulation in figure 4.22 There is almost no benefit in time however the control comes very close to a bang bang control meaning that the this solution is very close to the limit.

4.8.4 Simulation with Min Max solution

For the robust design the same maneuver has been simulated as above and the system is simulated for the nominal system however adding the new constraints correspond to a different control input and thereby a different resulting maneuver of the system. The first simulation is a simulation of the system with $N = 10$ and $T_s = 1s$ as described in the test case:

The simulation in graph 4.23(a) shows the nominal system simulated with the control input corresponding to the robust design. The simulation also shows that it is possible to improve the maneuver time where the control input is expected to get close to the optimal control input. Running a bisection e.g. from a love value of $X_L = 0.5$ to a upper value of $X_U = 0.7$ selected with respect to the results from the previous simulations corresponds to bisection result of $T_s = 1s$. This result can from figure 4.23(a) however not be the optimal solution. The plot in figure 4.23(b) shows the residual energy at the final time for $N = 10$ and $T_s = 1s$. The simulation shows that a bisection algorithm is not feasible to find the optimal maneuver time since there are local minimums. Selecting a arbitrarily optimal time $T_s = 0.6s$ corresponds to the following results. Figure 4.24 is a graph of the maneuver of the system and figure 4.6 shows the corresponding residual energy.



(a) Time response of the robust system for the optimal shaped input with $N = 10$ and $T_s = 1s$.

(b) Residual energy simulation for $N = 10$ and $T_s = 1s$.

Figure 4.23: Time response and residual energy simulation.

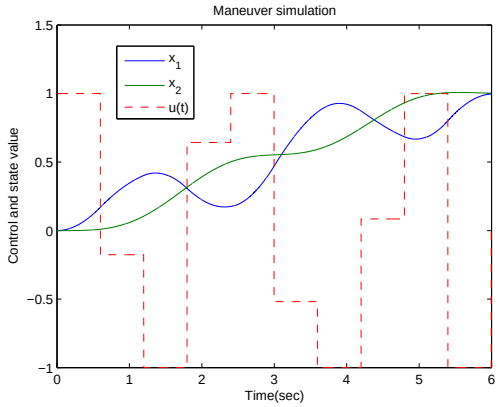


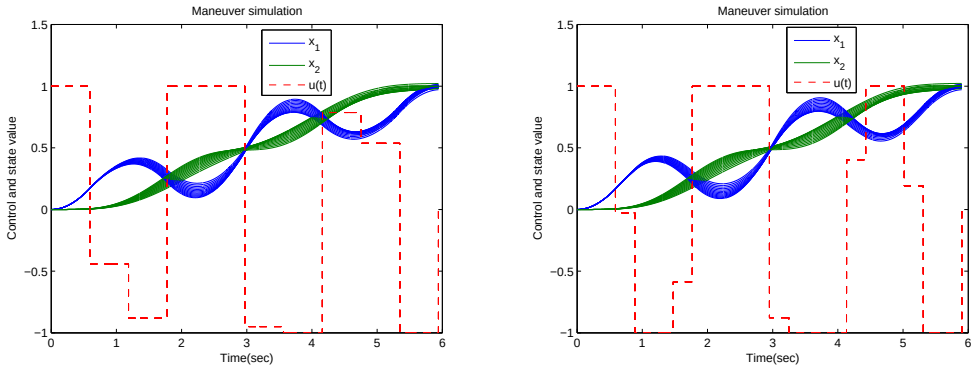
Figure 4.24: System maneuver for robust design for $N = 10$ and $T_s = 0.6s$.

Figure 4.24 shows that this time is very close to the optimal maneuver time since the control input is very close to the optimal bang bang control. This time is also close to the optimal time for the above simulation of the augmented system, hence the bisection results for the augmented system will be used as the optimal time.

From figure 4.25 showing the system maneuver for 10 and 20 steps, it is seen that the result is very similar to the augmented system. Again the controller proves to be robust to the defined uncertainty in the system and expanding the steps to 20 corresponds to reaching the limit with a bang bang solution. However from the residual energy and theory the two solutions can not be similar therefore it remain to show the differences.

4.8.5 Comparing robustness

As described in the test case the corresponding residual energy has been simulated for the three different solutions in order to compare performance. This serves to better illustrate the

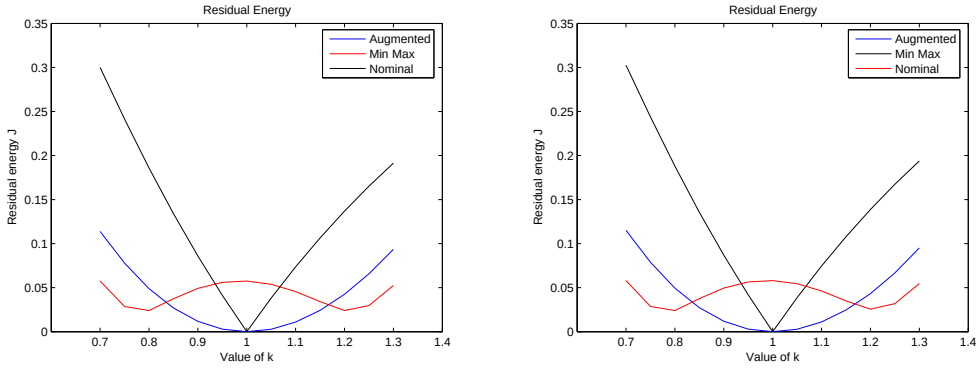


(a) Robust control with min max solution: Uncertain system ($T_f = 5.946s$ N = 10).

(b) Robust control with min max solution: Uncertain system ($T_f = 5.906s$ N = 20).

Figure 4.25: Plot of the system maneuver for 10 and 20 steps with the min max solution and optimal time.

robustness corresponding to the different methods. The nominal, the augmented solution and the Min Max solution. The simulation is done for the optimal time calculated by the augmented solution and the residual energy at the final time is simulated for 10 and 20 steps. See figure 4.26.



(a) Robust control with min max solution: Uncertain system ($T_f = 5.946$ N = 10).

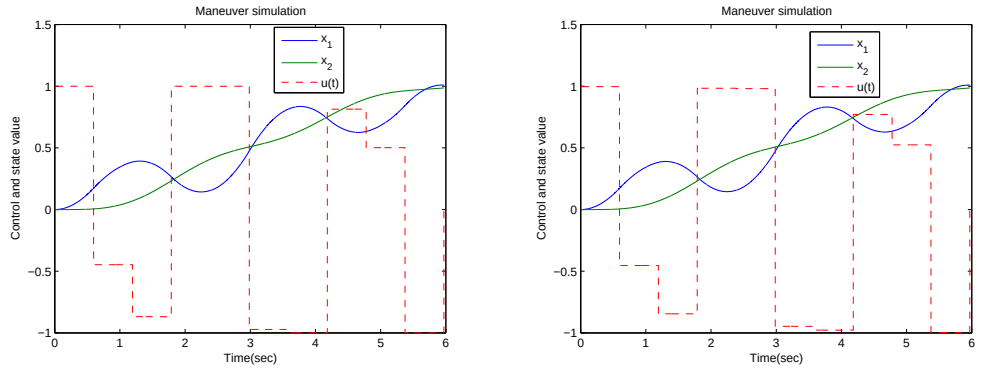
(b) Robust control with min max solution: Uncertain system ($T_f = 5.906$ N = 20).

Figure 4.26: Plot of the system maneuver for 10 and 20 steps with the min max solution and optimal time.

The augmented and min max solution truly benefit compered to the nominal design in Robustness. The nominal system is very specific designed for the nominal value of the uncertainty, where the two robust design have better performance over the entire rage of uncertainty. As expected from the design of the augmented system the best performance is around the nominal system moreover the augmented system keep the residual energy down in a curve. As shown in the above simulation the min max solution results in different performance. The residual energy is high around the nominal system; however over the entire uncertain area the residual energy is low for the min max solution. A combination of the two controllers would be the best solution e.g. a switch between the two controllers.

4.8.6 Comparing Sedumi powered Yalmip vs. MATLAB LMI toolbox

Since one of the goal for this project was to research the performance for different LMI algorithms and LMI toolbox for MATLAB the following simulation serves to compare the results corresponding to use the LMI toolbox in MATLAB or the *Sedumi* powered Yalmip toolbox. The test scheme described above is used for the simulations. First the optimal time simulation for $N = 10$.

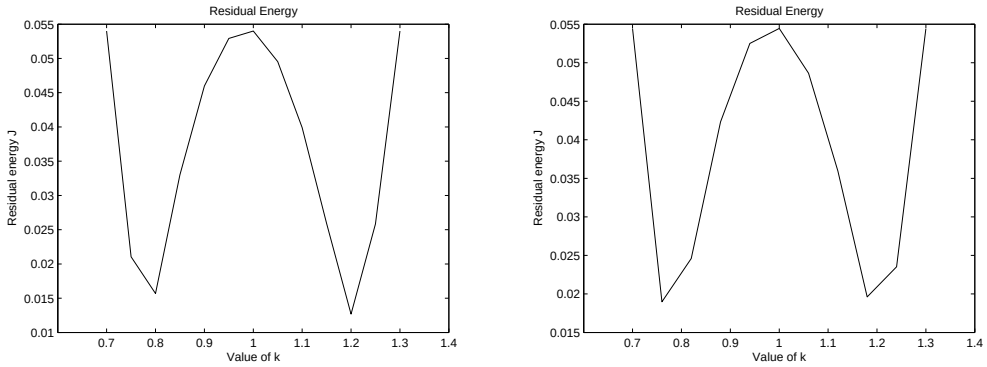


(a) System maneuver for Yalmip solution.

(b) System maneuver for MATLAB LMI toolbox solution.

Figure 4.27: Plot of the system maneuver for $N = 10$ and optimal final time $T_f = 5.975s$.

As can be seen from figure 4.27 the results is almost identical. The final residual energy or error meaning the optimal value for γ is $\gamma_{Yalmip(LMI)} = 0.0025$ and for the LMI toolbox $\gamma_{Matlab(LMI)} = 0.0026$. However the computational time for the Sedumi powered Yalmip solution is $T_{comp.} = 1.8395s$ where the time for the LMI toolbox is $T_{comp.} = 150.2476s$ which is a great improvement in performance for the Yalmip solution. The corresponding residual energy appear also similar see figure 4.28.

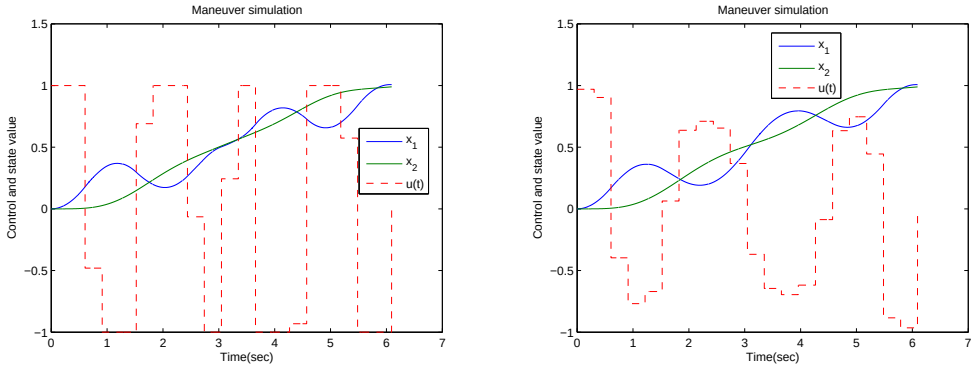


(a) Residual energy for Yalmip solution.

(b) Residual energy for MATLAB LMI toolbox solution.

Figure 4.28: Plot of the residual energy for $N = 10$ and optimal final time $T_f = 5.975s$.

The corresponding residual energy is also almost identical. In more interesting observation is archived with the next simulation for $N = 20$ and the optimal final time $T_f = 6.097s$.



(a) System maneuver for Yalmip solution.

(b) System maneuver for MATLAB LMI toolbox solution.

Figure 4.29: Plot of the system maneuver for $N = 20$ and optimal final time $T_f = 6.097s$.

In figure 4.29 the Yalmip solution result in a different control input closer to the optimal bang bang solution compared to the MATLAB LMI toolbox control input. The reason for this difference can be that the Sedumi LMI algorithm was able to reach a better solution. The final residual energy for the system or the maneuver error is for Yalmip $\gamma_{Yalmip(LMI)} = 0.0029$ where the LMI toolbox gives a $\gamma_{Matlab(LMI)} = 0.0030$ which is almost identical. The computational time for the two solutions is $T_{comp.} = 2.2916s$ for the Sedumi algorithm and $T_{comp.} = 150.2476s$ for the MATLAB LMI toolbox.

4.9 Conclusion

The feedforward design approach has been discovered in many different areas in the above simulation and results. A LMI based design method is derived which allow easy implementation of design problems first for the nominal system where the LMI formulation corresponds to the optimal shaped input filter. The bisection algorithm proved to be a powerful method to archive the optimal time giving a defined number of input steps. Using the bisection algorithm and bounding the input signal between ± 1 corresponds to a solution very close to the optimal bang bang controller.

Different constraints ware implemented, e.g. the bounded control input. Adding additional constraint such as deflection constraint has been discovered however without much luck. In this area more work has do be done. Different approach has to be implemented. Since the used approach appear to give control of the deflection but fare from optimal, it appear to be positive to find a solution.

In robustness consideration different approach was implemented with different performance. Robustness to uncertainty in a system can be archived with very nice results. The bisection algorithm was also applied in the robust design and corresponds to the optimal bang bang control. Depending on the criteria the designer has to choose between the augmented solu-

tion or the minmax solution. The augmented system results in high robustness close to the nominal system where the minmax solution have a more general distribution of robustness. However the minmax solution lack in performance around the nominal system compared to the nominal and augmented solution. Moreover it was not possible to use the bisection algorithm for optimization in the minmax approach. A combination of the two solutions would be the best solution.

Comparing the Yalmip results powered by *Sedumi* with the results from the MATLAB LMI toolbox appear to give similar results. However the implementation of a LMI formulation is very easy and strait forward in Yalmip where the LMI toolbox has limitations by the design. Yalmip and Sedumi proved to have significant better performance when it comes to computational time in the test results up to more than hundred seconds.

5 Conclusion

A LMI formulation was first developed for robust feedback design with linear quadratic controller performance. The designed controller was implemented on a 2 mass spring dash pot system in order to stabilize the system composed to a impulse input. A convex hull was defined corresponding to different uncertainties in the system. The designed and implemented controller appears to be a elegant way to design a robust controller. Compared in robustness the LMI designed controller had a significant better performance than the linear quadratic controller. However the corresponding cost for the robust controller ware as expected higher than for a controller without robustness. For the nominal system the cost for the LMI and LQR approach was the same. A general LMI method for state feedback control is developed with significant better performance in robustness. The simulated system is somehow simple however highly reliable in vibration control.

In feedforward design a LMI formulation was derived to design a optimal shaped input filter. Again the controller was applied to the 2 masse spring dash pot system. Optimization was archived by using a bisection algorithm, this result in a design very close to the optimal bang bang control. In robustness two different approaches was applied with different performance in result. Robustness to an uncertain parameter was archived with better performance compared to the nominal design. Depending on the criteria to the system the designer must choose between the two methods. The augmented method corresponds to have the best robustness performance near the nominal system where the minmax solution have a more general distributed robust performance, however with less robustness near the nominal system compared to the other solutions. Adding additional constrains such as deflection between the two masses was unsuccessful, this area is still in consideration. The deflection can be controlled but fare from optimal.

Yalmip appear to be a very easy and strait forward toolbox for solving LMI's compared to implementation in MATLAB LMI toolbox. Powered by Sedumi Yalmip is also a very powerful algorithm to calculate the LMI solution. Sedumi however can lack in performance, in this research the problem was numerical. Yalmip however offer a very easy interface to change the algorithm and a change to *sdpt3* solved the problem.

5.1 Outlook

Through this project some powerful methods for designing state feedback controllers with LQR and Robust performance is developed. Also different methods to calculate a shaped input feedforward controller has been developed and used. This research defined a very interesting background in LMI based robust control and optimization and therefore there are many possible applications and possible expansion of the research. Some very nice results is archived and Yalmip proves to be a very powerful tool to develop and implement LMI

design in.

However one of the approaches in the feedforward design was without succeeding. The attempt to include constraint on the deflection between the two masses did not come out as expected. The fist step to archive this performance will be to completely understand why the applied method didn't work as expected. One interesting approach to this could be to look if the approach is conservative. Understanding the problem hopefully reach to the ability to develop a method and implement the performance of controlling the deflection.

This research is especially useful in designing controllers for vibrating systems which is a area of great research at Buffalo University e.g. in the area of buildings. Vibrating system appears in many applications for control, so the system studied here can be used in many areas. The problem formulation however is made very general and also implemented in a general way therefore it becomes interesting to apply this methods to other and possible more advanced systems. The application for the feedback design is great since the design method for the state feedback design formulation is very close to designing a LQR controller. The LQR approach is widely used and therefore these approaches which support an easy way of including robust performance is a great expansion.

Also this study has only been theoretical so a very interesting challenge is to apply this method to a real application and study the performance. Since the systems in this research has been linearized to state space form there will be some difference and more unknown parameters. Here the robustness will be very interesting, this apply to both the methods.

5.1.1 AAUSAT-II Approach

In the research it was tried to implement the robust state feedback to the AAUSAT-II satellite model since I had experience from a past semester with that model. The AAUSAT-II satellite will have solar panels onboard which will be deployed after some time in space, this will change the inertia matrix of satellite and therefore a robust controller with respect to the inertia matrix would be interesting. To simplify the model three states corresponding to momentum wheels in the satellite was taken out of the model. Then the simplified linearized state space model becomes:

$$A = \begin{bmatrix} -S_\omega & \frac{1}{2}1_{3\times3} \\ 0_{3\times3} & I^{-1}(S_{I\omega} - S_\omega I) \end{bmatrix} \tag{5.1}$$

where $1_{3\times3}$ is a three times three identity matrix and S_ω is defined as:

$$S_\omega = \begin{bmatrix} 0 & \bar{\omega}_3 & \bar{\omega}_2 \\ \bar{\omega}_3 & 0 & \bar{\omega}_1 \\ \bar{\omega}_2 & \bar{\omega}_1 & 0 \end{bmatrix} \tag{5.2}$$

and $\bar{\omega}$ which is the angular rotation of the satellite is:

$$\bar{\omega} = [0\,0\,0] \tag{5.3}$$

zero because the goal is to stabilize and control the attitude of the satellite. The inertia matrix is defined as:

$$I = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{bmatrix} \tag{5.4}$$

where x, y and z is the inertia value for the corresponding axis. So changing parameters in the inertia matrix wont affect the state matrix A because everything goes out by the linearizing and simplification. However it will still affect the input matrix B:

$$B = \begin{bmatrix} 0_{3\times 3} \\ -I^{-1} \end{bmatrix} \tag{5.5}$$

Since the inertia matrix only affect the input matrix the amount of uncertainty in the inertia matrix has to bee a unreasonable value before the robust controller will have positive affect also seen by simulations. Therefore I had to cancel this approach. However I had a good experience in system knowledge and the general method was easy to use. Another way to deal with the uncertain inertia matrix due to the solar panel could be to use a adaptive controller which would adjust the controller due to the change in the satellite dynamic and kinematics. At University at Buffalo I had experience in working with Applied Nonlinear Control. My final project was not in adaptive control but in attitude control of a rigid body satellite using a backstepping approach. This is somehow out of the scope here and is therefore only included as an appendix, however it still show a way to archive a controller which is stable in a wide area. See appendix D on page 70

Bibliography

[Sedumi] Advanced Optimization Lab at McMaster University. *Sedumi version 1.1 Self-Dual-Minimization – Optimization solver*, 2005. http://sedumi.mcmaster.ca/.

[SDPT3] R. H. Tutuncu C K. C. Toh and M. J. Todd. *SDPT3 version 3.02 – a MATLAB software for semidefinite-quadratic-linear programming*, 2002. http://www.math.nus.edu.sg/~mattohkc/sdpt3.html.

[Thomas Conord] Thomas Conord. *Linear Matrix Inequality based Robust Control Synthesis*, 2005. Thesis. University of New York at Buffalo.

[Yal] Dr. Johan Löfberg. *Yalmip Homepage*, 2005. http://control.ee.ethz.ch/~joloef/yalmip.php.

[Ogata] Katsuhiko Ogata. *Modern Control Engineering 3rd. ed.* Prentice-Hall, Inc., 1997. ISBN 0-13-227307-1.

[Ulf Jönsson] Ulf Jönsson Royal Institute of Technology Sweden. *Control Synthesis Using LMIs*, 2001.

[Bisection] Tarunraj Singh. *Linear Programming*, 2005.

[Scherer & Weiland] Carsten Scherer and Siep Weiland. *Lecture Notes DISC Course on Linear Matrix Inequalities in Control*, 1999.

[Gregory J.] Gregory J. Toussaint. *Nonlinear Control Seminar*, 2000. http://decision.csl.uiuc.edu/nl-grad-sem/sem_papers/sem7Feb00.ps.gz.

[Mathworld] wolfram. *Superposition Principle*, December 2005. http://mathworld.wolfram.com/SuperpositionPrinciple.html.

Schur complement

A

Theorem: The symmetric matrix:

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{A.1}$$

By Schur Complement this matrix is negative definite if and only if:

$$M_{11} \prec 0 \quad \text{and} \quad M_{22} - M_{21} M_{11}^{-1} M_{12} \prec 0 \tag{A.2}$$

or if and only if

$$M_{22} \prec 0 \quad \text{and} \quad M_{11} - M_{12} M_{22}^{-1} M_{21} \prec 0 \tag{A.3}$$

as can be seen from the second inequality in A.2 and A.3 these are nonlinear inequalities. Using this result , it is obviously that a nonlinear inequality by Schur complement can be converted to a linear matrix inequality.

Yalmip Example

B

In this appendix a example of implementing a LMI formulation in Yalmip will bee derived also this appendix will introduce some of the algorithms which Yalmip can use and how they are implemented. In order to show how to implement a developed LMI formulation and archive results this appendix will go through the implementation of the robust state feedback design defined in 3.4 on page 21. The trivial steps such as converting to a state space model will be skipped since this should be known. Still this appendix is very basic. The focus here will be how to go from a model definition to design and calculate the feedback gain using Yalmip.

B.1 Nominal State Feedback Design

First some useful data is needed. Number of states which equals the number of rows in the state matrix and the number of inputs which equals the number of columns in input matrix. Then define γ:

$$\gamma = \text{sdpvar}(1,1) \tag{B.1}$$

now γ is defined as a real value scalar variable. The symmetric P matrix from the design formulation is connected to the X matrix and has the sice of the number of states here 4. So $X = X_{4\times4}$ symmetric matrix of variables:

$$X = sdpvar(4,4,'symmetric') \tag{B.2}$$

The Y matrix is more connected to the feedback gain remind that $K = YX^{-1}$ therefore the Y matrix is connected to the input matrix and has the sice of inputs times states. So in this case $Y = Y_{1\times4}$ vector of variables.

$$Y = sdpvar(1,4) \tag{B.3}$$

Now all the variables is defined and we are ready to design the LMI formulation for the controller. This actually correspond to implement the definition in equation 3.26 on page 22. First we implement the constraint for the scalar variable γ:

$$lmi = set(gamma \cdot eye(4) < X); \tag{B.4}$$

where *lmi* is used as the variable containing all constraints. Since X is a 4×4 matrix the identity matrix of sice states is multiplied with γ. Then we include the constraint that $X \succ 0$.

$$lmi = lmi + set(X > 0); \tag{B.5}$$

Now we only need to define the matrix in equation 3.26 then the system is defined for the nominal system:

$$lmi = lmi + set([-inv(Q)zeros(4,1)X; zeros(1,4) - inv(R)Y; X'Y'A \cdot X + X \cdot A' + B \cdot Y + Y' \cdot B'] < 0); \tag{B.6}$$

As seen above this is also very simple. Here the zeros is defined to fill out the matrix however using variables for number of states and number of inputs this can be made very general see e.g. the code in the end of this appendix. Now we are ready to include robustness in the design.

B.2 Robust State Feedback Design

In the robust design the great matrix in equation 3.26 is included as a constraint for every one of the 16 vertices. See equation 3.25 on page 22. I will not go completely through that step since it would be to tedious here a explanation is more convenient. To construct the convex hull defined in 3.3.1 on page 19 a binary vector is very convenient. Having a binary vector describing all possible combinations of the vertices will help the implementation. Then all the constraints is implemented by running a loop for every vertices here 16 times in the loop all values of the uncertain parameters is set with respect to the binary vector. In the end of the loop the actually constraint is includes here as:

$$lmi = lmi + set([-inv(Q)zeros(4,1)X; zeros(1,4) - inv(R)Y; X'Y'A \cdot X + X \cdot A' + B \cdot Y + Y' \cdot B'] < 0); \tag{B.7}$$

the above equation is executed for every vertices because the matrix A and B will change every time a parameter is changing here 16 times. Now we can calculate for the optimal γ and thereby get the desired variables in X and Y. However in the design formulation and simplification we had to say maximize γ therefore we solve for γ with a negative sign to get the optimal value corresponding to the lowest cost for the controller.

$$solvesdp(lmi, sum(-gamma)); \tag{B.8}$$

the above equation will solve for the optimal γ with respect to all the constraints defined in the *lmi* variable. Since the feedback gain is given as $K = -Y \cdot X^{-1}$ we have to take out X and Y from the results.

$$Xopti = double(X); \tag{B.9}$$

where $Xopti$ is the optimal value for the variables in the symmetric matrix X. The optimal values for in vector Y is archived in the same way:

$$Yopti = double(Y); \tag{B.10}$$

Now we can simply calculate the optimal robust feedback gain by:

$$K = -Yopti \cdot inv(Xopti); \tag{B.11}$$

Also if one like to calculate the cost for the controller by use of the matrix $P = X^{-1}$ so:

$$P = inv(Xopti); \tag{B.12}$$

What is back in the program is to insert the feedback gain in the system as a negative or positive feedback gain and simulate the system. The actual MATLAB code can be found on the CD and the code described here is inserted in the end of this appendix.

B.2.1 LMI solvers

In Yalmip it is very easy to change the LMI algorithm/solver. Different solvers have different performance in computational time or can deal with different mathematical problems. I have only been using [Sedumi] and [SDPT3]. *Sedumi* is a add-on for MATLAB, which lets you solve optimization problems with linear, quadratic and semidefiniteness constraints. Complex valued data and variables is possible in Sedumi. In all it solve large scale optimization problems efficiently. The *SDPT3* solver almost support the same problems semidefinite quadratic linear programming, however in the new version the problem data must be real.

From my experience *Sedumi* is very powerful especially in computational time, however I experienced some numerical problems from time to time. Most offend in the infeasible area of the solution so not a real problem, however in the deflection section 4.8.1 on page 46 I had to change the solver from *Sedumi* to *SDPT*3 to get reasonable results. To find and change the solver is easy and explained at the Yalmip home page [Yal]. In the *solvesdp* command include a option as:

$$solvesdp(lmi, sum(-gamma), options); \tag{B.13}$$

and then set the solver by the option as:

$$options = sdpsettings('solver','sedumi'); \tag{B.14}$$

now just change *sedumi* to the solver you like and follow the instruction on the Yalmip home page. Go to the Yalmip manual and find the side: Interfaced solvers.

B.3 Matlab Code

Here the actually MATLAB code to calculate the robust feedback gain is included:

```
%Yalmip function for calculating the feedback gain K

function [YALtime Pyal Kyal] = yalmip2masss(A,B,C,D,Q,R,N)

% Number of states:
ns = length(A(:,1));

% Number of inputs:
ni = length(B(1,:));

%gamma = scalar
gamma = sdpvar(1,1);

%Variables X 4x4
X = sdpvar(ns,ns,'symmetric');

%Variables Y 1x4
Y = sdpvar(ni,ns);

%Constraints
lmi = set(gamma*eye(ns) < X);
lmi = lmi + set(X > 0);

%Constraints for A0
lmi = lmi + set([-inv(Q) zeros(ns,ni) X ;
                 zeros(ni,ns) -inv(R) Y ;
                 X' Y' A*X+X*A'+B*Y+Y'*B'] < 0);

%Make a binary serie for the 2^N uncertainties
bina = binary(N);

%Amount of uncertainty in percent
%Constraints for Ai
Vmax = 15;%Positive
Vmin = 15;%Negative

Vmax = (Vmax/100)+1;
Vmin = 1-(Vmin/100);

%Constants
m1(1) = 1; m2(1) = 1; k(1) = 1; c(1) = 0.1;


for i=1:2\^2*N-1
    %k/m1
    if bina(i,4) == 1
        un1(i) = (k(1)*Vmax)/(m1(1)*Vmin);
```

```
    else un1(i) = (k(1)*Vmin)/(m1(1)*Vmax);
    end
    %k/m2
    if bina(i,3) == 1
        un2(i) = (k(1)*Vmax)/(m2(1)*Vmin);
    else un2(i) = (k(1)*Vmin)/(m2(1)*Vmax);
    end
    %c/m1
     if bina(i,2) == 1
        un4(i) = (c(1)*Vmax)/(m1(1)*Vmin);
    else un4(i) = (c(1)*Vmin)/(m1(1)*Vmax);
     end
     %c/m2
     if bina(i,1) == 1
        un5(i) = (c(1)*Vmax)/(m2(1)*Vmin);
    else un5(i) = (c(1)*Vmin)/(m2(1)*Vmax);
     end

A = [   0      0        1      0 ;
        0      0        0      1 ;
     -(un1(i)) (un1(i)) -(un4(i)) (un4(i));
      un2(i)   -(un2(i)) un5(i)  -(un5(i))];

%Construct the corresponding Bi matrix
B = [0 0 1 0]'* [ones(1,N*2)*1/(m1*Vmax) ones(1,N*2)*1/(m1*Vmin)];
B = B(:,i);
%
C = [0 1 0 0];
%
D = [0];

%Include the corresponding constraints
lmi = lmi + set([-inv(Q) zeros(ns,ni) X ;
                 zeros(ni,ns) -inv(R) Y ;
                 X' Y' A*X+X*A'+B*Y+Y'*B'] < 0);

end

%Define the algorithm
options = sdpsettings('solver','sedumi');

%Solve for the variables
tic
solvesdp(lmi,sum(-gamma),options);
YALtime = toc
%solvesos(lmi,sum(-gamma));

%Take out the calculated variables for X
Xopti = double(X);

%Take out the calculated variables for Y
Yopti = double(Y);
```

```
%Calculate the feedback gain
Kyal = -Yopti*inv(Xopti);

%Get the P matrix in order to calculate the cost
Pyal = inv(Xopti); Pyal = Xopti\^-1;
```

Elimination Lemma

Definition 1. An orthogonal complement $N_\perp$ is any matrix of maximal rank such that $N_\perp N = 0$ and $N_\perp N_\perp^T > 0$.

Elimination Lemma 1 *Let $M \in \mathbf{R}^{n\times k}$, $N \in \mathbf{R}^{n\times m}$, and $H = H^T \in \mathbf{R}^{n\times n}$. The following statements are equivalent*

(I) There exists $X \in \mathbf{R}^{m\times k}$ such that

$$NXM^T + MX^TN^T + H < 0 \tag{C.1}$$

(II) The following two conditions hold

$$N_\perp H N_\perp^T < 0 \quad or \quad NN^T > 0 \tag{C.2}$$
$$M_\perp H M_\perp^T < 0 \quad or \quad MM^T > 0 \tag{C.3}$$

Nonlinear Control Project

D

In this appendix the final project in my applied nonlinear control class is inserted.