ISSN 0280–5316
ISRN LUTFD2/TFRT--5585--SE

Robustness Analysis using Omola with Application to Power Networks

Michael Lantz

Department of Automatic Control
Lund Institute of Technology
October 1997

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
October 1997

Document Number
ISRN LUTFD2/TFRT--5585--SE

Author(s)
Michael Lantz

Supervisor
Anders Rantzer, Sven Erik Mattsson,
Magnus Akke (Sydkraft)

Sponsoring organisation

Title and subtitle
Robustness Analysis using Omola with Application to Power Networks

Abstract

Mathematical tools exist for robustness analysis on a model with uncertainty. Modern modeling tools do not provide an automated way of transforming a differential-algebraic model into a model ready for robustness analysis in Matlab.

This thesis presents a way of doing the transformation. The emphasis lies on usage of robustness tools using the modeling language Omola but other modeling tools are briefly discussed. Some concepts of model reduction combined with this transformation are discussed.

Models are developed in Omola and analyzed using the robustness routines in the mu-toolbox, using the described transformation. Particular effort has been put in the analysis of a 16 generator model of the Nordic power system using the mu-toolbox.

Key words
Robustness Analysis, Differential-algebraic model, Uncertain parameters, Power systems

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
60

Recipient's notes

Security classification

Contents

Acknowledgements

First of all I would like to thank my main supervisor Anders Rantzer for his support during the thesis project. I would also like to thank Sven Erik Mattsson for all his help with modeling in Omola and how to model power systems. I am also very grateful to Lennart Andersson who was a major help during the weeks when Anders was in the USA. I would also like to thank Tomas Schönthal for immediately taking the time to change the OmSim code when I asked him to.

I would also like to take the opportunity to thank Olof Samuelsson for reading and commenting on the thesis report. My thanks also to Magnus Akke and Daniel Karlsson of Swedish power supplier Sydkraft for providing us with interesting feedback on our visit.

1. Introduction

The thesis focuses on the usage of modern modeling tools for robustness analysis. This chapter gives an introduction to the thesis and formulates the objective.

1.1 Objective

Nature is extremely complex in itself and the contributions of mankind to our world makes it even more so. Science has realized during the past century that a complete understanding and modelling of our world will be impossible. One way of dealing with this impossibility of creating a complete model is to model the system as an uncertain model, where the system has some distinguishing characteristics, and where the unmodeled features are introduced as uncertainties. These models with uncertainties can be analyzed in a rigorous way, making it possible to draw conclusions of the systems even though a rather rough model with a lot of uncertainties are considered. During the past 20 years the theory of analysis and modelling of uncertain systems, robustness analysis, has progressed quickly. See *e.g.* [13]. However, few large models have been analyzed since the modeling tools of today do not support the use of robustness analysis. Mathematical software, *e.g.* the μ-toolbox in Matlab [4], provides powerful numerical algorithms for analyzing uncertain models. The μ-toolbox has a major drawback as it demands the model written on a certain form where the uncertainties are extracted, the diagonal perturbation form, which is not the normal way of setting up a model. The μ-toolbox use computation-intensive algorithms which did not allow this project just ten years ago, but with the computer power of today even large models should be analyzable.

The main objective of the thesis project was to implement routines which make it possible to analyze a model, constructed in the modeling language Omola, with the μ-toolbox in Matlab. A large example was selected in order to show the possibilities with this transformation. A linearized power system model with 16 generators representing the nordic power system has been developed and this model has been analyzed using the routines in the μ-toolbox.

1.2 Outline of the thesis

The topic of robustness analysis is relatively theoretical for a Master thesis and during the project a lot of time has been spent on theoretical studies of robustness analysis and mathematical concepts. As the reader may not be familiar with the theory of μ-analysis an introduction to the subject is presented in Appendix A. The presentation in Appendix A contains only aspects of robustness analysis which are used in the thesis.

Chapter 2 describes a transformation from a very general form of system model containing eleven model matrices into the diagonal perturbation form which is presented in Appendix A. In Chapter 2 some variations of the transformation procedure such as minimal realizations, model reduction and positive uncertainties, are also presented. After this some different modeling

tools and their impact on the previous transformation procedure are discussed in Chapter 3. The emphasis lies on the use of Omola and the modeling and simulation environment OmSim. A smaller machanical example is considered in Chapter 4, and it is analyzed thoroughly using the discussed routines. Finally, in Chapter 5, the large Nordic Power System example is analyzed in a way that should demonstrate the power of the developed routines. We see that numerical difficulties arise when analyzing a problem of this size. In Appendix B the syntax for the developed matlab routines is presented in order to simplify the use of the routines. In Appendix C, an example analysis procedure using Omola is presented to simplify the usage of the routines.

2. Transformation from a Differential-Algebraic Form to the Diagonal Perturbation Form

2.1 Introduction

The aim of this chapter is to present a way of transforming an ordinary dynamic system, possibly including algebraic conditions, with uncertain parameters in the appropriate matrices, to the standard Diagonal Perturbation Form. Readers unfamiliar with this approach to robustness analysis are recommended to read Appendix A, which contains a brief introduction to the subject.

A transformation of the type presented here is often useful when the model is given by a modeling tool like the modeling language Omola and when it contains several uncertain parameters. In order to use the μ-toolbox in Matlab,the system must be written on the diagonal perturbation form. The work transforming an arbitrary system into this form is rather tedious, thus making the need for some automatic way of doing this obvious. A suggestion is given in this chapter. This is done in a matlab function and the syntax for this function is found in Appendix B, among the other Matlab routines.

2.2 Problem formulation

A general form of a linear time-invariant dynamic system with inputs u, outputs y, dynamic states x and algebraic variables z is

$$\begin{aligned} E\dot{x} &= Ax + Bu + Fz \\ Gz &= Hx + Ju \\ y &= Cx + Du + K\dot{x} + Lz \end{aligned} \tag{2.1}$$

with eleven matrices, possibly with uncertain parameters in them, describing the system. By algebraic variables we mean variables which do not appear differentiated in the model description. The first equation given in (2.1) is the same as in an ordinary state space description with the addition of a dependence on the algebraic variables in Fz and an E-matrix before the derivatives, thus introducing a more easily interpreted model of a dynamic system. The middle equation given in (2.1) describes the algebraic conditions connecting z and x with the possible influence of inputs u also. The last equation provides the outputs y which can be dependent of $\dot{x}$ and z besides the dynamic state x and u. This step is also taken to enhance physical interpretation of the different model matrices. This many matrices will also provide us with extra freedom when we transform the system into the diagonal perturbation

form. The uncertainties in the model will appear as uncertain parameters in the different model matrices.

It should be noted that this system is a multi-input-multi-output system, a MIMO system, indicating that the B- and J-matrices are not necessarily column vectors, the C-, K and L-matrices are not necessarily row vectors and the D-matrix is not necessarily a scalar. Naturally, z and x indicate vectors of state variables, algebraic and differential respectively.

The following section decsribes a conversion of a system with algebraic constrints given by (2.1) into the form

$$\begin{cases} \begin{bmatrix} \dot{x} \\ v \\ y \end{bmatrix} = M \begin{bmatrix} x \\ w \\ u \end{bmatrix} \\ w = \Delta v \\ \Delta = \text{diag}(\delta_i I_{p_i}), i = 1, 2, \ldots, n \end{cases}$$

where n is the number of perturbed parameters and p_i is the size of the identity matrix corresponding to the ith perturbation. In other words the system can be seen as a block diagram. This is shown in Figure 2.1.

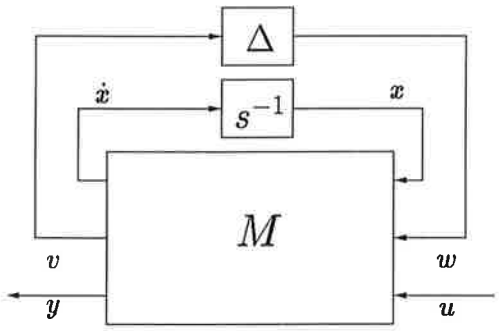

Figure 2.1 Linear dynamic system with extracted uncertainties.

A generalized method for transforming a system into the diagonal perturbation form is very complex and in this thesis some necessary simplifications of the prerequisites are done. We therefore assume the following:

- every uncertain parameter appears linearly in the system matrices
- every uncertain parameter is real
- the nominal matrices E and G in system (2.1) are invertible

If these criteria are fulfilled the transformation is rather straightforward as seen in the next section.

2.3 A single perturbed parameter in the system

In this section we assume that there is only one perturbed, uncertain parameter in the system. Assuming this, it is possible to write system (2.1) with one

perturbed real parameter as

$$\begin{aligned}(E_0+\delta E_1)\dot{x} &= (A_0+\delta A_1)x + (B_0+\delta B_1)u + (F_0+\delta F_1)z \\ (G_0+\delta G_1)z &= (H_0+\delta H_1)x + (J_0+\delta J_1)u \\ y &= (C_0+\delta C_1)x + (D_0+\delta D_1)u + (K_0+\delta K_1)\dot{x} + (L_0+\delta L_1)z\end{aligned} \tag{2.2}$$

where

$$\delta \in [-1,+1].$$

We assume that the uncertain parameter corresponding with the uncertainty δ appears in all the system matrices. In practice, this is usually not the case, but to get a more general approach we assume uncertainties in all the model matrices.

The goal is now to rewrite this system to the diagonal perturbation form, thus extracting the uncertainty in the uncertain parameter, *i.e.* δ, and get a system on the form described in Figure 2.2.

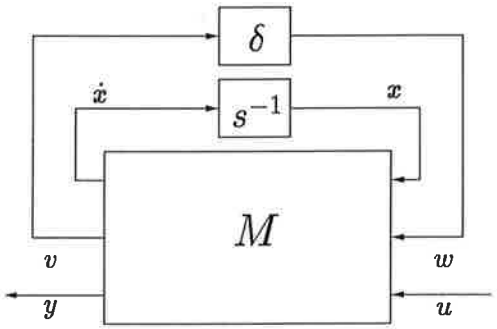

Figure 2.2 Linear dynamic system with one uncertain parameter.

This is possible by rewriting system (2.2). Let us first examine the inputs and outputs of the system (2.2) and rewrite the system using a block matrix approach. To get a simpler analysis let us assume that the matrices E_0 and G_0 are the corresponding identity matrices. We can do this as we required these matrices to be invertible in the initial assumptions, and this assumption will not change anything in the discussion below. By removing E_0 and G_0 we get

$$\begin{aligned}(I+\delta E_1)\dot{x} &= (A_0+\delta A_1)x + (B_0+\delta B_1)u + (F_0+\delta F_1)z \\ (I+\delta G_1)z &= (H_0+\delta H_1)x + (J_0+\delta J_1)u \\ y &= (C_0+\delta C_1)x + (D_0+\delta D_1)u + (K_0+\delta K_1)\dot{x} + (L_0+\delta L_1)z\end{aligned}$$

which can be rewritten as

$$\begin{bmatrix}\dot{x}\\ z\\ y\end{bmatrix} = P\begin{bmatrix}\dot{x}\\ z\\ x\\ u\end{bmatrix} + \delta R\begin{bmatrix}\dot{x}\\ z\\ x\\ u\end{bmatrix} \tag{2.3}$$

where the matrices P and R are defined by

$$P = \begin{bmatrix}0 & F_0 & A_0 & B_0\\ 0 & 0 & H_0 & J_0\\ K_0 & L_0 & C_0 & D_0\end{bmatrix}$$

and

$$R = \begin{bmatrix} -E_1 & F_1 & A_1 & B_1 \\ 0 & -G_1 & H_1 & J_1 \\ K_1 & L_1 & C_1 & D_1 \end{bmatrix}.$$

In a physical application the perturbed parameter does not often appear in all the elements of the eleven system matrices. This implies that many elements in the submatrices of R are zero. The mathematical interpretation of this is that the matrix R will most probably not have full rank and it is possible to express R in a simpler way. The tool for doing this is Singular Value Decomposition, SVD, which is described in any book on matrix theory, *e.g.* [12].

The SVD makes a factorization of an arbitrary matrix R, of size $n \times m$, as

$$R = U\Sigma V^H.$$

The matrix Σ is diagonal, with the singular values in diminishing order on the diagonal. The matrices U and V are unitary matrices and the H denotes hermitian conjugate, which equals the normal transponate in the case with real matrices. The matrix R has precisely $r = \text{rank}(R)$ number of singular values that are separated from zero. As the matrix R most often loses rank, a rank factorization SVD will suit us well. If we partition the matrices U and V according to the non-zero singular values we get

$$R = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^H = U_1 \Sigma_r V_1^H. \tag{2.4}$$

Here we can easily combine Σ_r with either one of U_1 or V_1^H giving the full rank factorization as

$$R = R_1 R_2$$

$$R_1 \in \mathbf{R}^{n \times r}$$

$$R_2 \in \mathbf{R}^{r \times m}$$

This gives us possibilities to reduce the order of the system. This reduction is necessary to do when we have system with many perturbed parameters and many states and algebraic conditions. Suppose an example with 100 different algebraical variables, 30 different states, 5 outputs and 10 perturbed parameters. As will be seen in later chapters it is not difficult to imagine a problem of this size. A system of this size will give us an R-matrix with 135 rows and even more columns. As will be seen later on we introduce a R-matrix for each perturbation and these will be brought together in a single matrix which can be transformed to the M-matrix of Figure 2.2. Without the factorization this intermediate matrix will be of a size larger than 1300×1300 which is nearly impossible to analyze with existing computer power. However, if each R-matrix instead has $\text{rank}(R) = 2$ the intermediate matrix will be of size we can handle numerically. This idea will be seen more clearly while we proceed with the calculation of M.

With the rank factorization described above we can rewrite (2.3) as

$$\begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} = P \begin{bmatrix} \dot{x} \\ z \\ x \\ u \end{bmatrix} + R_1 \delta R_2 \begin{bmatrix} \dot{x} \\ z \\ x \\ u \end{bmatrix}.$$

The reason for the rank factorization will now be obvious if we define variables v and w as we had in Figure 2.2 as

$$v = R_2 \begin{bmatrix} \dot{x} \\ z \\ x \\ u \end{bmatrix}$$

$$w = \delta v.$$

We see that because of the factorization the size of the vectors v and w will be exactly r =rank(R). By defining v and w it is possible to rearrange (2.3) and extract the uncertainty δ. Let us add v to the outputs and w to the inputs, hence

$$\begin{bmatrix} v \\ \begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & R_2 \\ R_1 & P \end{bmatrix} \begin{bmatrix} w \\ \begin{bmatrix} \dot{x} \\ z \\ x \\ u \end{bmatrix} \end{bmatrix}.$$

We see that the uncertainties has been extracted and that the system is closely related with the one described in Figure 2.2. The problem is now that the outputs and inputs to this system do not correspond with the outputs and inputs to the matrix M which is desirable in the diagonal perturbation form. We need to remove the $\dot{x}$ and the z from the inputs and to remove the z from the outputs. We will solve this by block partitioning P and R and then close the loop corresponding to the undesired inputs and outputs. Let us partition the submatrices as

$$R_1 = \begin{bmatrix} R_{1,\dot{x}} \\ R_{1,z} \\ R_{1,y} \end{bmatrix} \qquad R_2 = \begin{bmatrix} R_{2,\dot{x}z} & R_{2,xu} \end{bmatrix}$$

$$P = \begin{bmatrix} P_{\dot{x},\dot{x}z} & P_{\dot{x},xu} \\ P_{z,\dot{x}z} & P_{z,xu} \\ P_{y,\dot{x}z} & P_{y,xu} \end{bmatrix},$$

which gives us a description of the system as

$$\begin{bmatrix} v \\ \dot{x} \\ z \\ \dot{x} \\ y \end{bmatrix} = \begin{bmatrix} 0 & R_{2,\dot{x}z} & R_{2,xu} \\ R_{1,\dot{x}} & P_{\dot{x},\dot{x}z} & P_{\dot{x},xu} \\ R_{1,z} & P_{z,\dot{x}z} & P_{z,xu} \\ R_{1,\dot{x}} & P_{\dot{x},\dot{x}z} & P_{\dot{x},xu} \\ R_{1,y} & P_{y,\dot{x}z} & P_{y,xu} \end{bmatrix} \begin{bmatrix} w \\ \begin{bmatrix} \dot{x} \\ z \end{bmatrix} \\ \begin{bmatrix} x \\ u \end{bmatrix} \end{bmatrix}$$

It is now possible to close the loop corresponding to the middle $\dot{x}$ and z-pair thus removing these variables from the system and revealing the desired matrix M. The closing of the loop yields

$$M = G_{22} + G_{21}(I - G_{11})^{-1}G_{12},$$

where the different matrices G_{ij}, i and j is 1 or 2, are defined by

$$G_{11} = \begin{bmatrix} P_{\dot{x},\dot{x}z} \\ P_{z,\dot{x}z} \end{bmatrix} \; G_{12} = \begin{bmatrix} R_{1,\dot{x}} & P_{\dot{x},xu} \\ R_{1,z} & P_{z,xu} \end{bmatrix}$$

$$G_{21} = \begin{bmatrix} R_{2,\dot{x}z} \\ P_{\dot{x},\dot{x}z} \\ P_{y,\dot{x}z} \end{bmatrix} G_{22} = \begin{bmatrix} 0 & R_{2,xu} \\ R_{1,\dot{x}} & P_{\dot{x},xu} \\ R_{1,y} & P_{y,xu} \end{bmatrix}.$$

The invertibility of the matrix $I - G_{11}$ is guaranteed as we know from (2.3) that $P_{z,\dot{x}z}$ is identically zero, *i.e.* the algebraic variables z are independent of the $\dot{x}$ and z, which is quite natural.

The calculations which calculates M from a set of eleven parameter equations are straightforward if done with mathematical software like *e.g.* matlab. If some of the matrices in the model are identically equal to zero, for example if we have an ordinary state space model without algebraic equations, the operations done here will of course work just as fine.

The matrix M will be referred to as the SYSTEM matrix with capital letters in the matlab way in the rest of the thesis. This matrix can be used to do a robustness analysis of an arbitrary system given on the form of (2.1). The derivation of the SYSTEM matrix presented in this section is however only done in the case of one uncertain parameter. We need to be able to handle an arbitrary number of perturbed parameters. The operations will not be more difficult but the need for computer tools to do it will become even more obvious.

2.4 An arbitrary number of perturbed parameters

The previous section involved a systematic approach of extracting the SYSTEM matrix when we have only one perturbed parameter in the system. When we have n perturbed parameters the problem is approached in a similar manner with the addition of more perturbations δ_i and associated matrices. We get

$$\begin{aligned}
(I + \sum_{i=1}^{n} \delta_i E_i)\dot{x} &= (A_0 + \sum_{i=1}^{n} \delta_i A_i)x + (B_0 + \sum_{i=1}^{n} \delta_i B_i)u + (F_0 + \sum_{i=1}^{n} \delta_i F_i)z \\
(I + \sum_{i=1}^{n} \delta_i G_i)\dot{x} &= (H_0 + \sum_{i=1}^{n} \delta_i H_i)x + (J_0 + \sum_{i=1}^{n} \delta_i J_i)u \\
y &= (C_0 + \sum_{i=1}^{n} \delta_i C_i)x + (D_0 + \sum_{i=1}^{n} \delta_i D_i)u + \\
&\quad + (K_0 + \sum_{i=1}^{n} \delta_i K_i)\dot{x} + (L_0 + \sum_{i=1}^{n} \delta_i L_i)z.
\end{aligned}$$

Again, we assume the matrices E_0 and G_0 to be unity, which simplifies the analysis. It follows easily, in the same way as was done in the previous section, *i.e.* by introducing a new matrix P corresponding to the nominal system and matrices R_i corresponding to the different perturbations. Naturally, R_i corresponds to the ith pertubation. We can write the general matrix form as

$$\begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} = P \begin{bmatrix} \dot{x} \\ z \\ x \\ u \end{bmatrix} + \sum_{i=1}^{n} \delta_i R_i \begin{bmatrix} \dot{x} \\ z \\ x \\ u \end{bmatrix}. \tag{2.5}$$

We will of course do the rank factorization here by assuming $R_i = R_{i,1} R_{i,2}$ and by introducing new variables v_i and w_i as

$$v_i = R_{i,2} \begin{bmatrix} \dot{x} \\ z \\ x \\ u \end{bmatrix}$$
$$w_i = \delta_i v_i.$$

It is now possible to set up the structure of a matrix which has the uncertainties δ_i of the system extracted in the way we did in the previous section. This will give us the system as

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \begin{bmatrix} \dot{x} \\ z \end{bmatrix} \\ y \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & R_{1,2} \\ 0 & 0 & \dots & R_{2,2} \\ \vdots & \vdots & \ddots & \vdots \\ R_{1,1} & R_{2,1} & \dots & P \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \begin{bmatrix} \dot{x} \\ z \end{bmatrix} \\ x \\ u \end{bmatrix}$$
$$w_i = \delta_i v_i$$

The next step is to visualize the loop corresponding to the $\left[\begin{smallmatrix} \dot{x} \\ z \end{smallmatrix}\right]$-part of the inputs and outputs like we did in the previous section, and then to close this loop. These operations will yield a SYSTEM matrix M with a corresponding Δ-matrix containing the different perturbations of the uncertain model and with a structure as

$$\Delta = \begin{bmatrix} \delta_1 I_{r_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \delta_n I_{r_n} \end{bmatrix}$$

where we have n uncertain parameters and

$$r_i = \text{rank}(R_i) \qquad\qquad i = 1, 2, 3 \dots n. \tag{2.6}$$

The total size of the SYSTEM matrix will still be large if the number of uncertain parameters is large. Some measures of reducing the system can be taken without doing an actual approximation of the system. We will deal with some ways in the following sections.

2.5 Minimal realization

In an arbitrary model rewritten to a SYSTEM matrix it is quite possible that some of the states in the nominal resulting model is either unobservable, uncontrollable or both. Such states should be removed from the model, *i.e.* we make a minimal realization of the nominal model. Let us define a partitioned version of M as

$$\begin{bmatrix} \dot{x} \\ \begin{bmatrix} v \\ y \end{bmatrix} \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} x \\ \begin{bmatrix} w \\ u \end{bmatrix} \end{bmatrix}.$$

According to a Kalman decomposition described in *e.g.* [15] it is possible to get a minimal realization of the system by only using the states corresponding to the subspace that is both observable and controllable. By using a transformation of variables we transform the M-matrix and get a new system which has the structure

$$\begin{bmatrix} \dot{\hat{x}} \\ \begin{bmatrix} v \\ y \end{bmatrix} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \hat{x} \\ \begin{bmatrix} w \\ u \end{bmatrix} \end{bmatrix}, \tag{2.7}$$

where the submatrices A, B, C are given by

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 \\ 0 & A_{22} & 0 & 0 \\ A_{31} & A_{32} & A_{33} & A_{34} \\ 0 & A_{42} & 0 & A_{44} \end{bmatrix}$$

$$B = \begin{bmatrix} B_1 \\ 0 \\ B_3 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} C_1 & C_2 & 0 & 0 \end{bmatrix}.$$

The new state vector $\hat{x}$ is

$$\hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix}$$

where $\hat{x}_1$ denotes the observable and controllable subspace, $\hat{x}_2$ denotes the observable but uncontrollable subspace, $\hat{x}_3$ denotes the controllable but unobservable subspace and $\hat{x}_4$ denotes the uncontrollable and unobservable subspace. By observable and uncontrollable subspace we mean a subspace othogonal to the unobservable and controllable subspace, respectively.

The system (2.7) has the same input-output behaviour as the system

$$\begin{bmatrix} \dot{\hat{x}}_1 \\ \begin{bmatrix} v \\ y \end{bmatrix} \end{bmatrix} = \begin{bmatrix} A_{11} & B_1 \\ C_1 & D \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \begin{bmatrix} w \\ u \end{bmatrix} \end{bmatrix},$$

which is why this form is a minimal realization of the system.

2.6 Model reduction

When introducing uncertainties in a differential-algebraic system as (2.1) the complexity of the system quickly becomes large. The rank factorization using SVD which was described earlier may still leave the model large and time-consuming to analyze and control.

In many cases it is possible to reduce the complexity of the model and still maintain a model that contains all the dominant features of the original system. For example, it might be possible to remove one of the uncertainties of the system if it does not affect the stability much.

A simple way of reducing the system has been implemented in the Matlab routines. When doing a singular value decomposition it is necessary to introduce a level of tolerance due to numerical considerations. Singular values which are less than this tolerance will be considered as zero and the information and the uncertainties corresponding to these small singular values will be omitted from the model. As the singular value is small the impact of this reduction on the input-output behaviour of the system will also be small in some sense. This implies that we should be able to use this tolerance as a reduction tool. Other more elaborate reduction algorithms exist, such as *e.g.* multidimensional balanced realization, but the drawback with this is the computationally intense algorithms associated with it, which makes the reduction procedure slow if the model which are to be reduced is large. The strength with the reduction method presented here is that it is fast. The rank factorization must still be done, so we lose no time comparing to an ordinary transformation. The main weakness is that we do not have an error bound of the input-output gain as we have in other methods. We will however see that the μ-routines will provide us with a rough comparison method.

When doing a singular value composition we can decompose an arbitrary matrix R as

$$R = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}^H = U_1 S_r V_1^H,$$

where S_r contains the singular values of A that are separated from zero. If we assume the matrix R as

$$A = \begin{bmatrix} U_{11} & U_{12} & U_2 \end{bmatrix} \begin{bmatrix} S_{r,1} & 0 & 0 \\ 0 & S_{r,2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{11} \\ V_{12} \\ V_2 \end{bmatrix}^H, \tag{2.8}$$

where the singular values in $S_{r,2}$ are small, *i.e.* $\sigma(S_{r,2}) <$ tolerance and we decide to consider them as zero. It is now possible to get the approximation $\hat{A}$ of the original A as

$$\hat{A} = U_{11} S_{r,1} V_{11}.$$

If we use this reduction when doing the SVD described in equation (2.4), we get a smaller system than we get when we include all the singular values separated from zero.

Future studies of this way of doing a model reduction will have to consider a way of making a more quantitative estimate of the error which appears if we reduce the model. Without an error bound the model reduction will be limited.

2.7 Using positive uncertainty sets

In section 2.3 we used the uncertainty set

$$\delta \in [-1, +1]. \tag{2.9}$$

Sometimes we do not wish for a symmetric interval of the perturbation. Consider for example a system with some kind of spring. It might be interesting to see how large the spring constant can be without the system becoming unstable. In this case we obviously need a perturbation that is always positive, *i.e.*

$$\hat{\delta} \in [0, +1].$$

The initiated reader may suggest that this can be solved by shifting the designated parameter interval and thus find the largest possible spring constant. This is of course possible, but as the μ-routines demand a nominally stable model, many shifting steps may be necessary. By introducing positive parameter sets you only need one modification. The problem with the positive approach is that the matlab routines for analyzing robustness, *i.e.* the μ-toolbox assumes an uncertainty set as given by (2.9). We must, in order to solve problems with a positive uncertainty set, make a transformation of the positive set to a symmetric set. This can be done by introducing

$$\hat{\delta} = \delta^2,$$

where δ and $\hat{\delta}$ are defined above. This introduces a nonlinearity in the system, which can be taken care of using a loop transformation which introduces a second loop corresponding to the multiplication of δ with itself. For simplicity let us assume a system on state-space form with one uncertain parameter corresponding to the positive uncertainty set given above. This will correspond to a system with an uncertainty δ defined by (2.9),

$$\begin{aligned}(E_0 + \delta^2 E_1)\dot{x} &= (A_0 + \delta^2 A_1)x + (B_0 + \delta^2 B_1)u \\ y &= (C_0 + \delta^2 C_1)x + (D_0 + \delta^2 D_1)u,\end{aligned}$$

which gives by assuming that E_0 is the identity matrix

$$\begin{bmatrix}\dot{x} \\ y\end{bmatrix} = P \begin{bmatrix}\dot{x} \\ x \\ u\end{bmatrix} + \delta^2 R \begin{bmatrix}\dot{x} \\ x \\ u\end{bmatrix}$$

if we define P and R as in previous chapters. If we introduce new inputs and

outputs and do the usual rank factorization of R we get

$$\begin{aligned} v_1 &= R_2 \begin{bmatrix} \dot{x} \\ x \\ u \end{bmatrix} \\ w_1 &= \delta v_1 \\ v_2 &= w_1 \\ w_2 &= \delta v_2. \end{aligned}$$

By introducing an extra variable pair we remove the nonlinearity introduced by δ^2. However, this has a major drawback. As we introduce an extra loop to take care of the square, we make the resulting SYSTEM matrix correspondingly large. As the routines for μ-analysis are computationally intense this makes the analysis slower. In the above case the system with extracted uncertainties will become

$$\begin{bmatrix} v_1 \\ v_2 \\ \begin{bmatrix} \dot{x} \\ y \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & R_2 \\ I & 0 & 0 \\ 0 & R_1 & P \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \begin{bmatrix} \dot{x} \\ x \\ u \end{bmatrix} \end{bmatrix}.$$

The transformation from this form to the SYSTEM matrix is straightforward and done in the same way we did before.

This has now provided us with means to analyze an uncertainty belonging to a positive set. Note that when analyzing a system defined in this way the μ-toolbox does not know that positive uncertainties are examined. We must be aware of this when we draw conclusions from the data presented by matlab. In order to get the corresponding $\hat{\delta}$ we must take the square of the δ presented by matlab. The perturbation thus received can then be analyzed and we might answer questions of the kind we wanted to. It is however not recommended to use this approach except in small models as the computation time in large problems quickly will become too large to handle.

3. Using Modeling Tools to provide Differential - Algebraic Forms

3.1 Introduction

As we have seen so far in the thesis, it is possible to rewrite a rather general model describing a set of differential-algebraic system equations into the diagonal perturbation form which is analyzable using the standard robustness tools. In a realistic case the size of the model described in (2.1) can be both large and complex, which makes the need of some kind of modeling tool obvious. In this chapter we discuss some different modeling tools, *i.e.* Omola and Simulink and their impact on the previous discussion and the steps which need to be taken if the user wants to use yet another modeling tool.

3.2 Using Omola as a modeling tool

The modeling environment OmSim and the modeling language Omola is often used fo examination of a dynamic system with algebraic constraints. It is possible to detect such things as nominal stability during simulations with different parameter values in the model. Robust stability is, however, not possible to analyze directly in the OmSim environment. This motivates the previously discussed transformation.

The matrix generation of OmSim

The OmSim environment is capable of creating some Matlab m-files, corresponding to the analyzed model. The original model has to be linear which of course is a major drawback and something which has to be changed in the future. Three files are created by OmSim.

Matrices.m This first file is a Matlab script which creates the eight matrices corresponding to the model

$$\begin{aligned} [\,\hat{E}_1 \quad \hat{E}_2\,] \begin{bmatrix} \dot{x} \\ z \end{bmatrix} &= \hat{A}x + \hat{B}u \\ y &= [\,\hat{E}_{1y} \quad \hat{E}_{2y}\,] \begin{bmatrix} \dot{x} \\ z \end{bmatrix} + \hat{C}x + \hat{D}u, \end{aligned} \tag{3.1}$$

which is a modified state-space model. In the m-file the matrices contain the different parameters of the analyzed Omola model on symbolic form. If we want to run this file we need to have the parameters defined in the current matlab workspace. This symbolic representation is the key that enables us to do the generation of the SYSTEM matrix with extracted uncertainties. We are able to extract the matrices corresponding to the different perturbations

δ_i by changing the parameter values and rerunning the file matrices. Let us consider an example. Suppose we have a matrix A with an uncertain parameter k. The parameter k is supposed to be uncertain in a specific interval, given by $k = k_0 + \delta k_1, \delta \in [-1, 1]$. If we assume that k appears linearly in A we are able to extract the model matrices we need for the transformation in the previous Chapter as

$$A_0 = A(k_0)$$

$$A_1 = A(k_0 + k_1) - A(k_0).$$

The same procedure is done for all the model matrices and all the perturbations.

Parfile.m The Matlab script Parfile presents all the parameters which were defined as parameters in Omola as a matlab scalar. The values these parameters had in the Omola model are given them upon calling of this function. As we need to be able to address the different parameters without having to define them separately we have to extract the parameter names from the file parfile in an automatic way. This is done with the use of the matlab stringhandling tools. The values of the parameters presented in parfile are a great help as the parameter values are inputs to the transformation functions presented in Appendix B.

Variables.txt This text file presents data of the model, such as the structure of the current model and the names of the inputs, outputs, state variables and algebraic variables that are used. The current algorithm in the OmSim code that creates these files sets all the terminals in the analyzed Omola model, which are defined as input terminals as inputs in the system. Note that only inputs in the top level of the Omola model are considered as inputs to the system. This is not a major hindrance but introduces some extra defining and connection of input terminals. Note also that all the terminals of the Omola model are considered as outputs, which implies that the user must choose which outputs that are interesting in the current application. The numbers of these outputs, which can be seen in the textfile, are inputs to the transformation functions in Appendix B.

Transformation to the desired form

We see that the difference between (3.1) and (2.1) is not big. Basically, all we have to do is separate the equations corresponding to the algebraic variables.

If we partition the matrices in (3.1) as

$$\begin{bmatrix} \hat{E}_{1,x} & \hat{E}_{2,x} \\ \hat{E}_{1,z} & \hat{E}_{2,z} \end{bmatrix} \begin{bmatrix} \dot{x} \\ z \end{bmatrix} = \begin{bmatrix} \hat{A}_x \\ \hat{A}_z \end{bmatrix} x + \begin{bmatrix} \hat{B}_x \\ \hat{B}_z \end{bmatrix} u$$

$$y = \begin{bmatrix} \hat{E}_{1y} & \hat{E}_{2y} \end{bmatrix} \begin{bmatrix} \dot{x} \\ z \end{bmatrix} + \hat{C}x + \hat{D}u,$$

it is easy to see that the partition matrices in this model directly gives the matrices in (2.1) with the exception of the matrix $\hat{E}_{1,z}$. The matrix $\hat{E}_{1,z}$ is identically equal to zero, which is a prerequisite of the analysis in chapter 2.

We now get the eleven model matrices of (2.1) as

$$
\begin{aligned}
A &= \hat{A}_x \\
B &= \hat{B}_x \\
C &= \hat{C} \\
D &= \hat{D} \\
E &= \hat{E}_{1,x} \\
F &= -\hat{E}_{2,x} \\
G &= \hat{E}_{2,z} \\
H &= \hat{A}_z \\
J &= \hat{B}_z \\
K &= \hat{E}_{1y} \\
L &= \hat{E}_{2y}.
\end{aligned}
$$

The purpose of the thesis was to be able to transform from Omola models so the model given in (2.1) was based on the model presented from Omola. This is of course why the correspondance between the two models is so simple.

Omola is a powerful modeling tool and if all the possibilities of the described model matrices are used, many robustness problems can be analyzed using Omola, even though they are complex in nature.

3.3 Using Simulink as a modeling tool

Simulink is a modeling tool that has become very widely used during the last few years, as it is connected with the mathematical package Matlab. Simulink offers a vast library of different modeling tools, and an easily grasped graphical user interface, which makes it a powerful tool in modeling and simulation of systems. One of the weaknesses of Simulink as with most other modeling tools is that it does not contain any facilities for analyzing uncertain systems.

In this section we will see a way of using Simulink for defining the model, and then transforming this model to the diagonal perturbation form as presented earlier in the thesis. Simulink does however not provide the same versatility in the model as Omola does with eleven model matrices. This drawback invokes a major limit of the usage of Simulink for robustness purposes.

The transformation

Each model object in Simulink has parameters connected to it. It is possible to define these parameters explicitly in Simulink by writing their values directly in the object. It is, however, also possible to give the parameter a name which represents the name of a variable which Simulink gets from the current matlab workspace during simulation and other Simulink operations.

Moreover, a Matlab function which transforms the current Simulink model into a linear state-space form has been provided by the software supplier, which makes the transformation easy, when we have described the various uncertain parameters in the model. This function has however some major drawbacks comparing to the corresponding linear matrices which was used in the Omola case. The Simulink function does not support algebraic constraints which leads us to exclude the z variables from the model or at least the uncertainties corresponding with the matrices F,G,H,J and L of (2.1). Furthermore the E-

and K-matrices are not supported which transforms the model to a standard state space model. Note that since Simulink only presents four model matrices the possibility of a nonlinear appearance of an uncertain parameter is higher. Simulink uses the model

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

This simplification makes the model less flexible, and many of the models analyzable with an Omola model is not possible to analyze with a Simulink model. Still, despite these obstacles, simple uncertain models can be analyzed using Simulink, and the matlab functions presented in Appendix B provides the means to do this.

3.4 Other modeling tools

Omola and Simulink are but two in the vast area of modeling tools. Naturally we might need to use other modeling tools than these two. Direct use of these modeling tools is not possible with the current implementation but if a model as in (2.1) is presented in a matlab function and this function is provided as input argument to the transform function developed, a transformation can be performed for a general case. It is also possible to ensure that a modeling tool automatically generates the two m-files Matrices.m and Parfile.m that was discussed above. Then, the model is analyzable in the same way as an Omola model is.

More detailed information of this procedure is provided in Appendix B among the different Matlab routines.

4. A Mechanical Example

4.1 Introduction

In this section we analyze an example model defined in Omola, using the transformations we have described earlier and matlab tools for robustness analysis. This section illustrates a complete analysis procedure which will further enhance the understanding of robustness concepts. All the modeling and transformation has been done with Omola and the developed routines. The analysis is then done using the μ-toolbox in Matlab.

The objective with this section has been to verify the robustness claims of a particular controller design presented in an article.

4.2 The model

The example is a benchmark problem taken from [8], and concerns robustness analysis on a system consisting of two mobile carts interconnected with a spring. This system is described by the differential equations

$$\begin{aligned} m_1\ddot{x}_1 &= -k(x_1 - x_2) + d_1 + u \\ m_2\ddot{x}_2 &= k(x_1 - x_2) + d_2 \end{aligned} \tag{4.1}$$

where the two masses m_1 and m_2 and the spring constant k, is the parameters of the system. The inputs to the system are the control signal u and the two exogenous signals d_1 and d_2, which are disturbances on the two carts. The variables x_1 and x_2 are the positions of the two carts respectively, and the output, the position to be controlled, is x_2, the position of the second cart. Note that the difficulty is that we want to control the position of the second cart while exerting a control signal on the first cart.

We create this 4th order model in OmSim using Omola. The automatic generation of the matlab m-files results in a system like (3.1).

4.3 The controller

In [8] a controller is created using the Coefficient Diagram Method. This design method results in a continous RST-controller, where the different polynomials are

$$\begin{aligned} R(s) &= s^4 + 50.50s^3 + 626.55s^2 + 1419.8s + 1269.5 \\ S(s) &= 215.04s^3 - 418.66s^2 + 701.30s + 108.90 \\ T(s) &= 108.90. \end{aligned} \tag{4.2}$$

The controller should be able to control the output x_2, and thus keep a reference value of zero. The control signal operates on the first cart and the

controlled signal is the second cart, which makes the controlling objectives rather difficult. The controller must be able to fulfill three basic robustness goals. These are

- The closed-loop system must achieve robust stability for $0.5 \leq k \leq 2.0$, while $m_1 = m_2 = 1.0$
- The closed-loop system must achieve robust stability for simultaneous uncertainties $0.7 \leq k, m_1, m_2 \leq 1.3$
- The closed-loop system must have reasonable robust performance when exposed to high-frequency disturbances

In the original benchmark problem additional performance specifications were presented. These specifications do not concern robustness analysis and are therefore excluded from the discussion here.

The controller represented by (4.2) should be able to fulfill all these robustness goals. We are able to verify this by using the results presented previously in the thesis.

4.4 Robustness analysis

Let us now do the actual robustness analysis of the example. First, we create the model using Omola and OmSim. Then, by using the matlab functions described in appendix B, we transform this system into a matlab SYSTEM matrix M, where the uncertainties are extracted, as described in chapter 1. The system is then interconnected with the controller, which is called K, to form the closed-loop system, as given below in Figure 4.1. This system is then analyzed with the help of μ-analysis, as described in Appendix A.

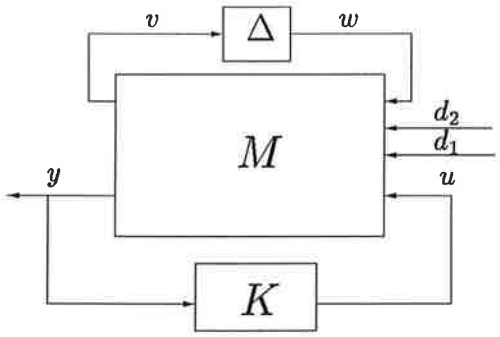

Figure 4.1 Control of an uncertain system.

Robust stability - Spring uncertainty

Let us begin with the first robustness restriction, *i.e.* robust stability when varying the spring constant, $0.5 \leq k \leq 2.0$, and letting the masses be unchanged. This imposes a restriction on k as

$$k = k_{nom} + \delta k_\delta$$

where $\delta \in [-1, +1]$, $k_{nom} = 1.25$ and $k_\delta = 0.75$. This is a condition which is recognized from the earlier discussion, and it is now easy to use matlab

to create an interconnected closed-loop system on the diagonal perturbation form. This interconnected system is then analyzed with μ-analysis in Matlab. The main advantage of using these values of k_{nom} and k_δ is that we easily see that the robust criterion is fulfilled if $\mu < 1$.

However, when we analyze the system, we discover that the algorithms that matlab uses when calculating the lower bound of the μ-value do not converge when the perturbations are made up of only real blocks. We take care of this problem in the way described in [4], *i.e.* by introducing a very small complex perturbation which is added to each real perturbation. This does not affect the μ-value very much but makes the algorithms converge. Let us introduce these small complex perturbations as

$$\delta_i = \delta_{i,R} + 0.01\delta_{i,C} \tag{4.3}$$

where $\delta_{i,C}$ is a complex perturbation with $|\delta_{i,C}| \leq 1$, and $\delta_{i,R} \in [-1, +1]$ as usual. This means that we introduce a complex addition that is 1% of the real perturbation. The μ-values are plotted against frequency in Figure 4.2. The two lines in the plot corresponds to the upper and lower bound of the μ-value.

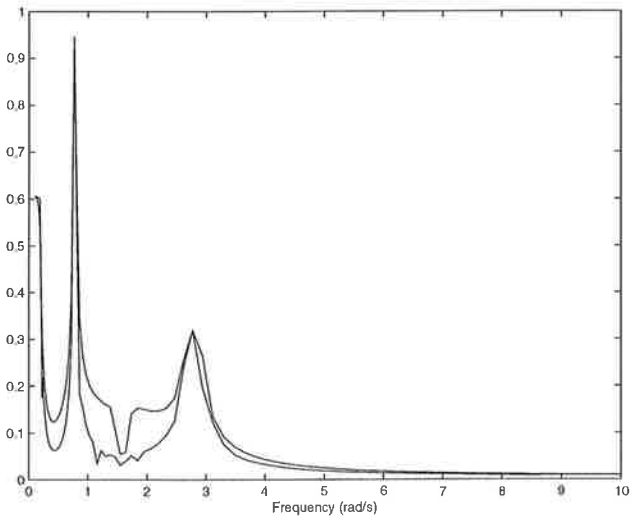

Figure 4.2 Robust stability μ-plot, $0.5 \leq k \leq 2.0$.

As we can see in Figure 4.2 the first stability criterion is fulfilled. The maximum μ-value is 0.95 which is below the critical value of 1. Without further analysis we can deduct that the system is stable for $k \in [0.46, 2.04]$. If we examine the actual δ that causes the maximum μ presented by matlab, we see that it is the lower bound of k that corresponds to the maximum μ-value. Naturally, it is interesting to find out the maximum k-value that still gives stability in the system. This can be found out by modifying k_{nom} and k_δ but we will examine this upper bound on k using positive uncertainty sets as described in chapter 1. Let us say that we have

$$k = k_{nom} + \hat{\delta} k_\delta, \qquad \hat{\delta} \in [0, +1].$$

where k_{nom} should be a value of k for which the system is stable, *e.g.* $k_{nom} = 2$. If we impose this positive uncertainty set and *e.g.* $k_\delta = 1$ we can examine the

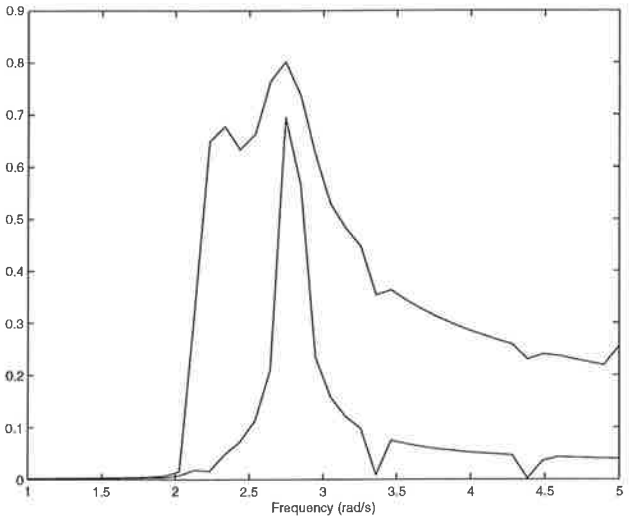

Figure 4.3 Robust stability μ-plot, positive uncertainty set.

upper critical value of k in an easy way. A μ-plot of this is shown in Figure 4.3.

We see that the convergence of the lower bound is not very good, but the maximum μ-value can be seen as $0.7 \leq \mu \leq 0.8$ which implies a critical perturbation $\hat{\delta}$ constrained by $1.56 \leq \hat{\delta}_{max} \leq 2.04$, *i.e.* $k_{max} \in [3.56, 4.04]$.

Robust Stability - Uncertainties in spring and masses

Next we analyze the second robustness restriction, *i.e.* robust stability when varying all the parameters, $0.7 \leq k, m_1, m_2 \leq 1.3$. In this case we cannot extract exact measures on the different parameter sets that guarantees stability, as they all contain uncertainties that may appear independently of each other. We can, however, examine if the robustness criterion given is fulfilled.

In the same way as was done with the first robustness restriction we see that we get restrictions on the three parameters as

$$\begin{aligned} k &= k_{nom} + \delta_1 k_\delta \\ m_1 &= m_{1,nom} + \delta_2 m_{1,\delta} \\ m_2 &= m_{2,nom} + \delta_3 m_{2,\delta} \end{aligned}$$

where $\delta_i \in [-1, +1]$, $k_{nom} = m_{1,nom} = m_{2,nom} = 1.0$ and $k_\delta = m_{1,\delta} = m_{2,\delta} = 0.3$. We use, in the same way as before, matlab to create a interconnected closed-loop system on the diagonal perturbation form.

We have only real perturbations, so we introduce the same small complex addition to the perturbations, *i.e.* the one given in (4.3). This time we have three uncertain parameters which should be able to vary independently of each other. This naturally makes the resulting matrices larger and thus the μ-calculations slower.

The μ-values are plotted against frequency in Figure 4.4. We see that the lower bound diverges much from the upper bound. This is a result of the difficulties of making the lower bound algorithm converge when the perturbations are real. The small complex part that has been added makes the algorithm converge but the result is still far from the correct one. Let us concentrate on the upper bound.

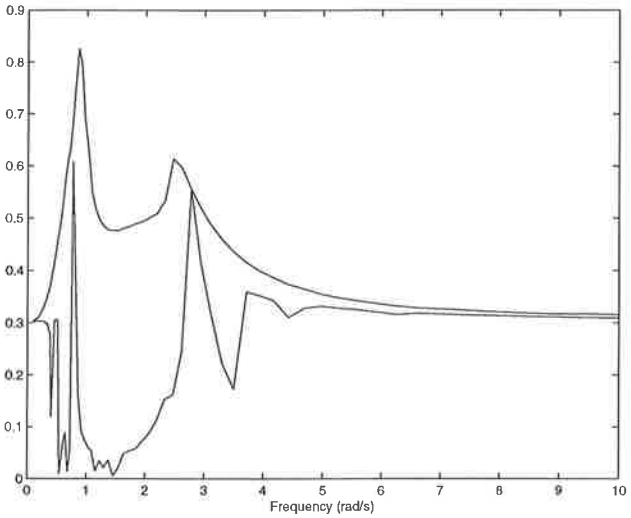

Figure 4.4 Robust stability μ-plot, $0.7 \leq k, m_1, m_2 \leq 1.3$.

The second stability criterion is obviously also fulfilled. The maximum μ-value is less than 0.82 which is below the critical stability boundary of 1. This implies that the closed-loop system is stable when varying the different parameters independently within the given restrictions.

Robust performance

Finally we analyze the robust performance of the system. Even if the system is robustly stable, it may not reduce the effects of the input disturbances d_1 and d_2 on the output. This might prove disastrous when running the real process. Therefore it is vital to examine this with a robust performance test.

This involves yet another step as we need to augment the structure of the Δ-matrix with a complex block corresponding to the inputs and the outputs of the system. This is described in Appendix A. In this case when we only have one output y and two inputs d_1 and d_2, the complex augmentation is a complex perturbation block Δ_C. The size of this complex block is a 2×1 matrix.

The μ-values are plotted against frequency in Figure 4.5 for the two different cases of uncertain parameters.

Note that in this case we do not have any convergence problems as the perturbations already contain complex elements, *i.e.* the performance block which we augmented in order to inspect robust performance. Thus, the small complex addition is in this case unnecessary, and the upper and lower bound is very close to each other, indicating a good convergence.

As we can see the closed-loop system does not remain stable for all values of the uncertain parameters. The μ-values become very large at low frequencies, which implies that the system is sensible to low frequency disturbances in the exogenous inputs d_1 and d_2. However, at frequencies above 3 rad/s the μ is significantly below 1, which gives us good high frequency disturbance rejection at all the uncertainties. The analyzed problem, a cart- and wagon problem, is typically a problem with only high-frequency disturbances so the third robustness condition is fulfilled. But, one must be careful if the experiment is conducted in the vicinity of some machines that run on low frequencies, as

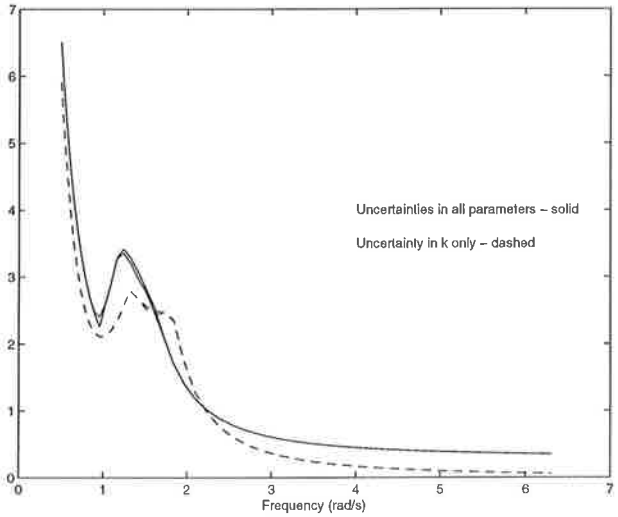

Figure 4.5 Robust performance μ-plot.

this might destroy the performance of the system.

We can conclude that the design presented in [8] fulfills the robust specifications it is said to fulfill. The system performs well under rather large uncertainties in the different parameters, and this CDM-designed controller is a good example of robust control.

We also saw that the analysis of robust stability was more difficult than the robust performance analysis. Convergence difficulties appear. In practice the separate check of robust stability is often excluded from the analysis and only the robust performance check is done. The robust performance check shows immediately, for a specified parameter set, if the model is robustly stable or not. This is due to the theorem (A.8).

5. A Power System Example

5.1 Introduction

The motivation of introducing computer tools for transforming an arbitrary model written with a modeling tool is that we are then able to analyze very big and complex systems. Perhaps the largest and most complex system made by man is the power network. Today, with the powernets of many countries connected into a single large system with thousands of generators and powerlines connecting them, there are problems of a magnitude impossible to analyze just a few years ago. With the rapid development of computer power it is now possible to analyze larger systems than was possible earlier. The standard approach for analyzing power systems has been to simulate an advanced model for different sets of parameters, in some kind of simulation software.

This chapter deals with the concept of making a robustness analysis of a linear power system model, presented in [1]. The model describes the Nordic powernet system greatly simplified, with 16 generators and connections between them. The model is large enough to put strains on the computer power and to illustrate the problems associated with analysis of a large problem.

The robustness analysis consists of verification of robust stability for a given range of variations in the load description and the line description. The worst case parameters have been extracted during the analysis and the performance for these parameters has been compared with the worst gain possible. We also examine a case where we have disconnected a power line and we also do a simple reduction of some of the uncertain parameters.

First, a short presentation of how a power net model works is presented. A linearized model is also presented. Then the model of the nordic power system is presented, and last a robustness analysis of the model is presented. This analysis is made in order to illustrate the possibilities of the computer tools, and it will not cover all aspects of a full robustness analysis of this complex model.

5.2 Description of a power system model

A power system model relies on the theory of synchronous generators and electric components. This theory will be omitted from the discussion here and the basic differential and algebraic equations will be described but not commented in detail. The model description is based on [1], [9] and [10].

A power system model consists basically of four different components

- generators, *i.e.* power plants
- a power grid consisting of transmission lines between the generators
- consumption loads
- controllers, which stabilizes and improves the prestanda

The power system should be able to provide power to the consumers at all times. This power should not vary in voltage and frequency too much. If some kind of variation occurs the system should be well damped in order to attenuate disturbances.

A generator

The generator can be modelled by three first-order differential equations and a set of algebraic equations. The reader should note that no effort has been made to explain the logic behind the generator equations. The interested reader should pursue this issue in a basic book on synchronous generators or for example [9]. The first differential equation, the swing equation, is

$$\frac{2H}{\omega_0}\frac{d\omega}{dt} + \frac{D}{\omega_0}\omega = P_m - P_e \tag{5.1}$$

where H is the inertia time constant associated with the generator in question and D is the damping associated with this generator. P_m is the mechanical power of the plant and this will be used as input to the system. P_e is the electrical power produced by the generator and ω_0 is the synchronous frequency of the system, in the nordic system 50 Hz or 100π rad/s. The variable ω is the actual frequency in the current generator.

By using the so called Park's transformation it is possible to deduct the second differential equation of the generator. The motivation for using Park's transformation is mainly the fact that the model becomes time invariant which greatly simplifies the model and the analysis. The equation is given by

$$T'_{d0}\frac{E'_q}{dt} = U_f - E'_q - (X_d - X'_d)I_d \tag{5.2}$$

where T'_{d0}, X_d, X'_d are generator constant parameters, and the variable E'_q is a voltage in the coordinate system introduced by Park's transformation and I_d is a current in the system. The variable U_f is the field voltage of the generator. This can be used to control voltage and reactive power of the system, which is often used to improve the system prestanda.

Furthermore, some algebraic equations are added to complete the description of the generator. The electrical power P_e is described as

$$P_e = E'_q I_q + (X_q - X'_d)I_d I_q \tag{5.3}$$

which is a nonlinear equation where X_q and X'_d are constant plant parameters.

The voltages V_d and V_q of the generator are given by

$$\begin{aligned} V_d &= X_q I_q \\ V_q &= E'_q - X'_d I_d. \end{aligned} \tag{5.4}$$

The description of the generator is however not yet complete. The Park's transformation which has given the equations above provides us with the local currents I_d and I_q and the local voltages V_d and V_q. The voltages seen from the rest of the powernet, the global voltages, depends on the local voltages and the current difference in angles, δ, between the current generator and the angle of the global system. The issue of choosing an angle of the global system has been widely discussed, but a common choice is to introduce the

global reference angular frequency ω_r as a weighted average of the angular frequencies of all the generators. We get the global currents and voltages as

$$\begin{aligned} V_{d,g} &= \frac{V_n}{V_B}(V_d \cos\delta - V_q \sin\delta) \\ V_{q,g} &= \frac{V_n}{V_B}(V_d \sin\delta + V_q \cos\delta) \\ I_{d,g} &= \frac{I_n}{I_B}(I_d \cos\delta - I_q \sin\delta) \\ I_{q,g} &= \frac{I_n}{I_B}(I_d \sin\delta + I_q \cos\delta) \end{aligned} \tag{5.5}$$

where δ is given by the simple differential equation, the third differential equation of the generator,

$$\frac{d\delta}{dt} = \omega - \omega_r.$$

The different constant parameters V_n, I_n, V_B and I_B are scaling parameters which are specified for each generator. Notice that a lot of nonlinearities exists in the equations (5.3) and (5.5), which implies that we have to linearize the model before we can use it for robustness purposes in the way we described previously in the thesis.

The power grid and the loads

The power grid serves the purpose of describing how the different generators influence each other. Basically the power grid, or network, consists of power lines between the generators. These lines are most easily modelled as a resistance and an inductance in series. The network is then a set of impedances connecting the different generators and loads. The dependence between the different generators is then easily solved using Kirchhoff's formulas.

The loads are also modelled as impedances. However, as the line impedance typically should transmit all the power without losses, the reactance is larger than the resistance. In the power loads the resistances are much larger than the reactances. In the ideal case we do not want any reactance at all as this introduces phase shiftings in the load voltages. In practice it is impossible to have loads without any reactances, but they have to be small to avoid unpleasant effects in the system. Normally such reactances are compensated by capacitors or by using the field voltage control of the generators. The loads are considered to be the ends of the system which is why the load impedances are connected to the ground, *i.e.* one end of the loads are at zero voltage.

Controllers

A powernet system consisting of only generators, lines and loads does not have to be stable at all. Some kind of controller is introduced in order to stabilize the system and improve the performance. Traditionally this controlling of the powernet introduces control of each single generator. This control is performed via the field voltage U_f and/or the mechanical power P_m. In recent years efforts have been made to control a powernet using methods for MIMO-control, thus placing the poles of the entire powernet system, and not trying to control the separate generators. Examples of this can be found in *e.g.* [1]. The linearized model provided by the routines presented in this thesis is ideal for making

some kind of MIMO control design. This has however not been done in the thesis project. Instead, focus has been on the analysis of the powernet models. A simple dynamic controller defined by the control law

$$U_f = U_{f0} + \frac{K_v(V_{ref} - V_t)}{s+2}$$

has been used, where V_{ref} is the desired value of V_t, which is defined by

$$V_t = \sqrt{V_d^2 + V_q^2}. \tag{5.6}$$

The parameter U_{f0} is the value of U_f which is desired in stationarity. K_v is a controller parameter which makes it possible to tune the controller. In the analysis a value $K_v = 30$ is used.

In the analyzed model, all control of the electrical power P_e has been omitted. This is not entirely realistic but provides us with more interesting analysis results as seen below.

A linearized model

As mentioned before, the transformation to the diagonal perturbation form requires a linear system. In order to produce the eleven model matrices given in (2.1) we need to linearize the model. Nonlinearities appears in some of the generator equations, (5.3), (5.5), (5.6) which can be linearized as

$$\Delta P_e = I_{q0}(X_q - X'_d)\Delta I_d + I_{q0}\Delta E'_q + (E'_{q0} + I_{d0}(X_q - X'_d))\Delta I_q, \tag{5.7}$$

$$\begin{aligned}
\Delta V_{d,g} &= \frac{V_n}{V_B}(\cos\delta_0\Delta V_d - \sin\delta_0\Delta V_q - (V_{d0}\sin\delta_0 + V_{q0}\cos\delta_0)\Delta\delta) \\
\Delta V_{q,g} &= \frac{V_n}{V_B}(\sin\delta_0\Delta V_d + \cos\delta_0\Delta V_q + (V_{d0}\cos\delta_0 - V_{q0}\sin\delta_0)\Delta\delta) \\
\Delta I_{d,g} &= \frac{I_n}{I_B}(\cos\delta_0\Delta I_d - \sin\delta_0\Delta I_q - (I_{d0}\sin\delta_0 + I_{q0}\cos\delta_0)\Delta\delta) \\
\Delta I_{q,g} &= \frac{I_n}{I_B}(\sin\delta_0\Delta I_d + \cos\delta_0\Delta I_q + (I_{d0}\cos\delta_0 - I_{q0}\sin\delta_0)\Delta\delta)
\end{aligned} \tag{5.8}$$

and

$$\Delta V_t = \frac{V_{d0}}{\sqrt{V_{d0}^2 + V_{q0}^2}}\Delta V_{d0} + \frac{V_{q0}}{\sqrt{V_{d0}^2 + V_{q0}^2}}\Delta V_{q0}. \tag{5.9}$$

In the equations (5.7), (5.8) and (5.9) Δ signifies the difference between the actual variable value and the linearization point. For example in the case V_t we have

$$\Delta V_t = V_t - V_{t0},$$

where the added index 0 in the equations denotes the linearization point.

All other model equations used in our powernet model are linear in the original structure which implies that they can be used in the linearized model if we replace the original differential or algebraic variables with the deviations from the operating point.

We now have model equations which we can use to build a complex Omola model from which we are able to extract the eleven model matrices of (2.1) containing the model parameters on symbolic form.

In order to use the developed model we need to find a linearization point of the system. This is extremely hard to do by explicit calculation from the model equations. A very good linearization point is possible to get by simulating the nonlinear system using *e.g* OmSim. When the nonlinear system reaches stationarity it is possible to extract the values of all model variables from Omsim. These values can then be used in the linearized model description above as a linearization point.

This completes the description of the model equations. No Omola code has been provided in the thesis report, but the interested reader is recommended to read [9] or [10] which give detailed descriptions of Omola code for powernets.

5.3 The Nordel powernet model

The Nordel Powernet as presented by [1] has been used to illustrate a typical powernet. This model describes the Nordel power system which connects the powernets of the nordic countries Sweden, Norway, Denmark and Finland. The model consists of 16 generators and a total of 20 different transmission lines which connects these 16 generators. Power loads are introduced at 16 different places in immediate connection to the generators. This model is a major simplification of the actual power system which consists of hundreds of different generators and perhaps thousands of different transmission lines and at least as many different loads. The 16 generator model contains however the basic structure of the Nordel powernet and is a fairly realistic model of an extremely complex system. A picture of the system is given in Figure 5.1.

As we have seen in the description of the model equations three differential states are introduced in the system for each generator. Moreover, as we have a dynamic controller with one additional state for each generator, we assume that the complete model will have $4 \cdot 16 = 64$ different states. We choose the mechanical powers P_m of the different generators as inputs, which implies that our model has 16 inputs. It has also 16 outputs as we choose the electrical power P_e from the generators as output. It is of course possible to choose other variables as output, but P_e is a common choice. With this choice of outputs one state will be unobservable in the output. The unobservable output corresponds to the absolute angles in the system, which is quite uninteresting for we use only the differences between angles. Naturally the model is transformed and the unobservable state can be removed from the model according to the discussion in section 2.6. This will leave us with 63 states, 16 inputs and 16 outputs in the model we have chosen to analyze. The need for computer tools in order to do this analysis is obvious.

The concept of this thesis has been to analyze the effects of uncertainties in different parameters which appears linearly in the eleven model equations (2.1). In the 16 generator Nordel model we have very many different parameters, more than 500 different ones, even though some of them are not uncertain at all. Consider for example the scaling factors V_n, V_B, I_n and I_B which appeared in (5.8) but are set by the user when doing the model. Some of the parameters may be uncertain but cannot be analyzed as they appear in a nonlinear way in the model equations. Consider for example the linearization

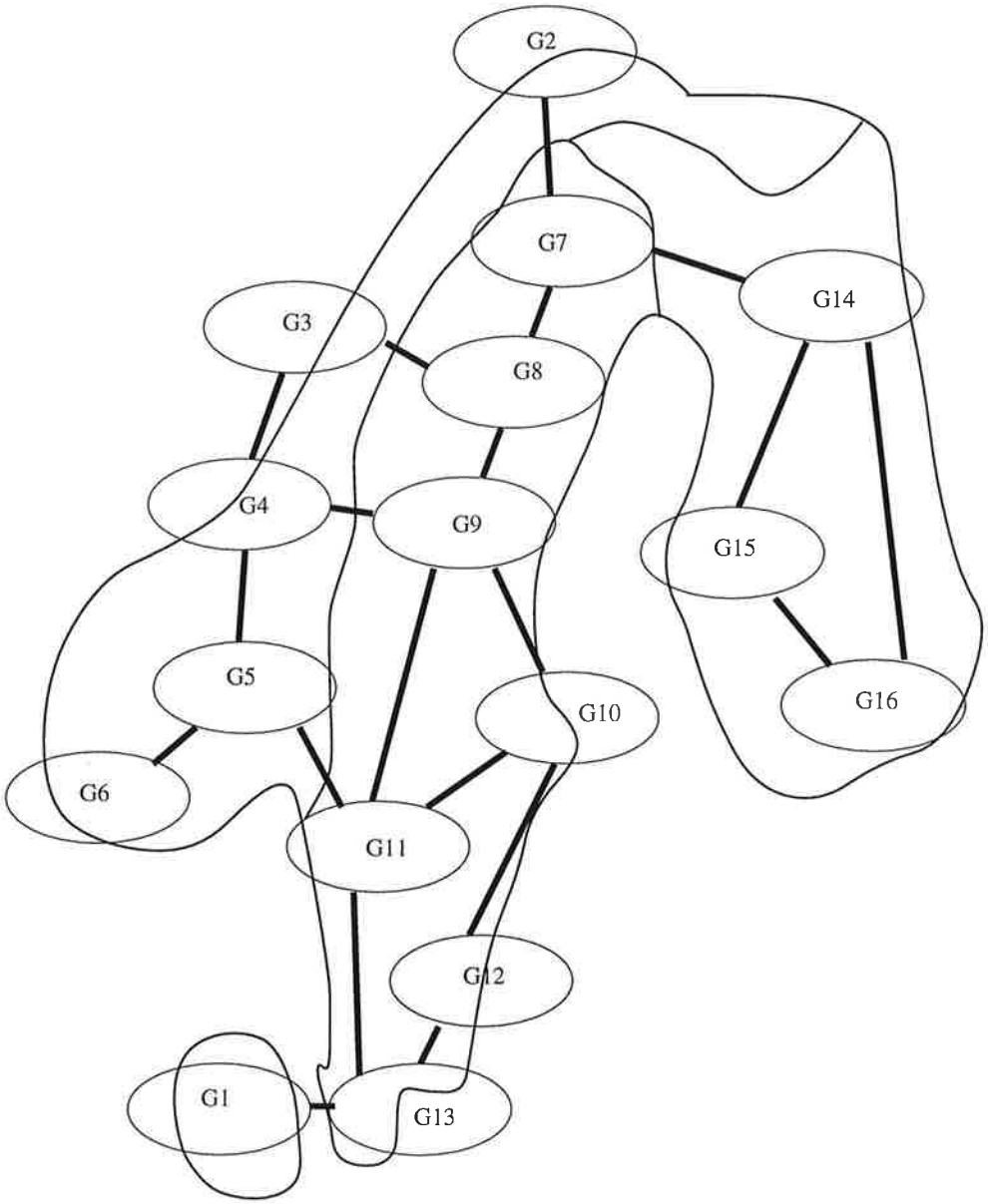

Figure 5.1 The Nordel power system as a 16 generator model

point with a total of 6 new linearization parameters, $E'_{q0}, \delta_0, V_{d0}, V_{q0}, I_{d0}, I_{q0}$, per generator, totalling 96 parameters for the entire model. If the linearization point is not certain, then an uncertainty appears in the linearization parameters. This uncertainty is not possible to analyze with the present implementation as these parameters appear in a nonlinear way. See equations (5.8) and (5.9). However, some of the most interesting uncertainties are still possible to analyze. The most apparent uncertainties in a power system model are of course the loads. The loads vary in time and are impossible to predict exactly. In our model we consider the loads as impedances with a large real part. Of course uncertainty appears in both the resistance and the reactance.

Let us consider a load as

$$Z_{load} = R_l + iX_l.$$

If we like different uncertainty scaling factors in the real and imaginary part we introduce uncertainties in the load as

$$Z_{load} = R_l + \delta_1 R_{l,\delta} + i(X_l + \delta_2 X_{l,\delta}) \tag{5.10}$$

where

$$\delta_1, \delta_2 \in [-1, +1].$$

This introduces 32 uncertain parameters in the system and their effect can be analyzed. The real and imaginary part of the loads appears in a similar way in the model matrices, which suggests that some kind of simplification of the uncertainty structure would be appropriate. This is something that will have to be examined in the future.

The loads are obvious sources of uncertainties but if this is the only source of uncertainty the model will still perform rather robustly, *i.e.* the input-output behaviour does not degrade very much even under relatively large uncertainties. Another source of uncertainty is the power lines. We model these as impedances with small resistances and larger reactances. These connections over long distances typically vary in both reactance and resistance due to varying external conditions. A natural way of dealing with these variations is to model them as uncertainties in the resistances and reactances in the same way as was done with the load impedances. Hence, we consider an uncertain transmission line as

$$Z_{line} = R_l + \delta_1 R_{l,\delta} + i(X_l + \delta_2 X_{l,\delta}) \tag{5.11}$$

where we again consider $\delta_i \in [-1, +1], i = 1, 2$.

We have, as mentioned previously, 20 different lines. This will in the case with real parameters introduce 40 new uncertain parameters, adding up to a total of 72 parameters. This number of uncertain parameters is highly unpractical to analyze with the computer power available and only some of the lines are considered uncertain in the following analysis.

5.4 Robustness analysis

A robustness analysis of a problem as large as this, is very time-consuming and a complete analysis is outside the scope of a master's thesis. The objective with this section is instead to show how the robustness routines in the μ-toolbox and the transformation routines described can be used to analyze a complex problem. The system parameters are relatively realistic but a simpler controller structure than normal has been used to simplify the analysis. Furthermore, computation power has been limited during the analysis, which has led to the exclusion of some of the line uncertainties in the analysis. The computation time might still extend to hours in some cases with the current choice of uncertainties and frequencies.

First we analyze the system to see if robust stability is provided by certain sets of parameters. This is the most basic feature that must be provided by a dynamic system. Then we extract the worst-case parameters for the discussed parameter set, and examine how large gain the model will have for this gain. Then we analyze the effect on the robustness issues if we disconnect one of the transmission lines. Last we see if we can reduce the model to a simpler model, and we use the μ-value to roughly compare the approximated model with the original one.

Robust stability

The first thing we must analyze in the system is if the model of the system still is stable when we introduce uncertainties of varying sizes in the model. How to choose these uncertainties is not easy and a thorough understanding of power networks is a prerequisite. In this case we have decided to focus on the demonstration of the possibilities with the developed Matlab routines, which is why the controller structure is simple. This simplicity makes the system rather sensible to parameter changes which is exactly what we are looking for in order to demonstrate the routines.

Let us assume that the different parameters may deviate 7% from their nominal values. It is not hard to imagine that this is a small uncertainty in the model. We assume uncertain parameters in all the loads and in 6 of the lines. This introduces 44 different parameters if we consider the real and imaginary part of the impedances separately. A robust stability analysis yields the results presented in Figure 5.2. Only the upper bound of the μ-value is displayed, because of bad convergence of the lower bound.

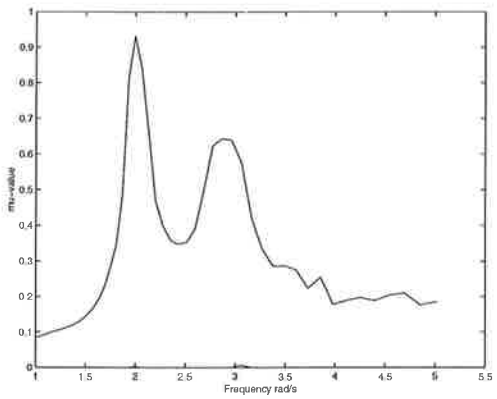

Figure 5.2 Robust stability using real uncertainties. 7% deviation.

We see that the system is robustly stable for a 7% deviation, but it cannot handle deviations that are very much larger.

The time for computing a robust stability check like this extends to about 30 minutes on a fast computer when about 80 frequencies are analyzed. Naturally this is very much if we would like to change the parameter sets in order to get a more thorough analysis. We would like to shorten the computation time, and some kind of optimization of the numerical algorithms used in the μ-toolbox would help us. We will however see in the next section that the real computation time appears as we introduce iterations in order to extract the worst case parameters.

The worst parameters

We now have discovered that our power system was robustly stable when exposed to parameter perturbations of 7% of the nominal values of the parameters. We would also like to have an estimate of the parameters which gives us the worst performance. As the μ of the robust stability test is close to 1, the stability boundary for the chosen parameter set, this hints that there might be some serious performance degradation for some parameters. We would like to know these parameters so we can try to avoid them.

A Matlab routine which extracts the worst parameters has been developed. This routine performs iterations of the kind described in section A.6, and requires a very long time to extract an estimate. Computation times of 2 or 3 hours is not unusual in this large model even in a small frequency interval. The iteration algorithm might be more optimized, but, clearly, this type of analyses requires a lot of computer power. The worst case parameters for the 7% deviation example was extracted, and the gain, *i.e.* the norm of the transfer function was calculated for the original system and the worst-case system and the result can be seen in Figure 5.3.

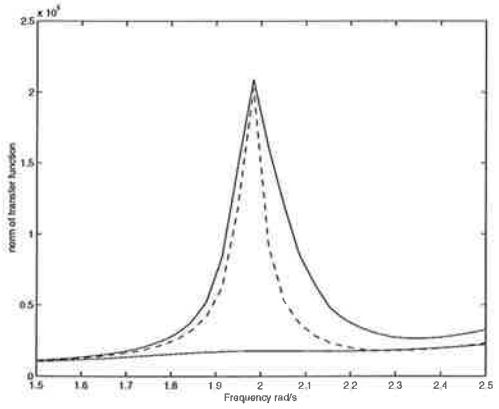

Figure 5.3 Input-output gain of 16 generator model. Nominal system gain - lower solid line. Worst case parameter gain - dashed line. Maximum gain possible - upper solid line.

As we suspected the gain becomes very much higher than for the original system in a frequency range around 2 rad/s. The upper bound of the gain which we got from the function call implies that the function was fairly successful at finding the worst parameter combination in the frequency range. The choice of the interval $[1.5, 2.5]$ rad/s is due to the fact that in this area the robust stability μ was highest. Note that there might be other degradations in other frequency areas, but we have chosen to examine this one.

The worst-case parameter values can be plotted in a diagram with the original parameter values. As we have impedances it is more useful to plot the parameters in the complex plane putting the resistance on the x-axis and the reactance on the y-axis. The model analyzed contains 22 different impedances, *i.e.* 44 real parameters, so we choose to show the principle by plotting five of the impedances, the ones belonging to the loads at generators G1, G7, G14, G15 and G16, in Figure 5.4.

When we decide on a specific parameter uncertainty, we decide how large the box around the nominal value is going to be. The μ-toolbox is able to decide where the worst parameter lies in this box and we can extract these

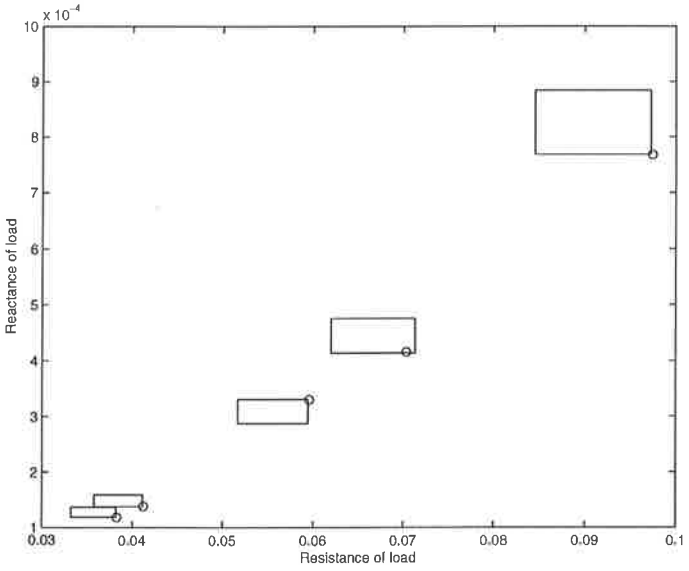

Figure 5.4 Five of the loads shown. The rectangles indicate the possible perturbation area, and the circles indicate the worst-case parameters.

parameters with the use of the toolbox.

It is interesting to see that the system is more sensitive the larger the loads in the system. According to the Swedish power supplier Sydkraft this is what to expect. They have concluded this result out of experience. Nevertheless, a more thorough method of extracting the worst parameters is now available.

Disconnection of a line

Robustness analysis is quite good at handling uncertainties of different kinds but if an uncertainty is of the kind that a parameter value becomes infinitely large it is not so easy to analyze. In our power system model we have such a case when *e.g.* a tree falls on a transmission line and the line gets cut off. Such an uncertainty must be analyzed by a change in the model.

Let us for example consider a case where we disconnect the line between generators G5 and G11 in the model shown in Figure 5.1. This is in the Oslo-Gothenburg area and the example is not unrealistic as power lines in this area have suffered damages and been disconnected in recent years with power oscillations as a result. We assume the same parameter uncertainties as we have done in the previous examples, *i.e.* 7% deviation from the original parameter values. Consider the robust stability μ-plot in Figure 5.5.

As we see the system does not fulfill the basic robust stability criterion $\mu < 1$ anymore. Clearly the possible incident of trees falling on the power lines must be avoided or somehow taken care of to guarantee good behaviour at all times for the power system.

Model reduction

As we have seen computation time is often very large when using the robustness routines of the μ-toolbox on large models. Earlier in the thesis we discussed the need for some kind of model reduction. A reduction will allow faster computations and might allow us to use simpler model in our pursuit of system understanding.

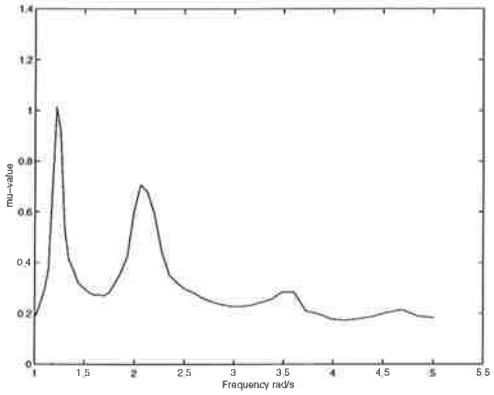

Figure 5.5 Robust stability μ-plot of the model with one disconnected line. 7% deviation

We have discussed two ways of doing model reductions. The first and most theoretically advanced, the one using the multidimensional balanced realization uses computation intense solving of linear matrix inequalities and a try of making a reduction of our model did not yield any result in 15 hours of computation, so this technique is highly unpractical with large models.

The other way of making reductions involved the reduction of singular values less than a certain tolerance during the building of the Diagonal Perturbation Form. This reduction method does not take longer time than an ordinary building of the model, which is about 10 minutes for a model of the size of the power system model analyzed here. A tolerance of 0.01, *i.e.* the singular values less than 0.01 were considered to be zero gives us a reduced model where 22 of the parameters has been removed from the model. These excluded parameters consists of all the reactances from the loads and all the resistances from the power lines. Consider Figure 5.6.

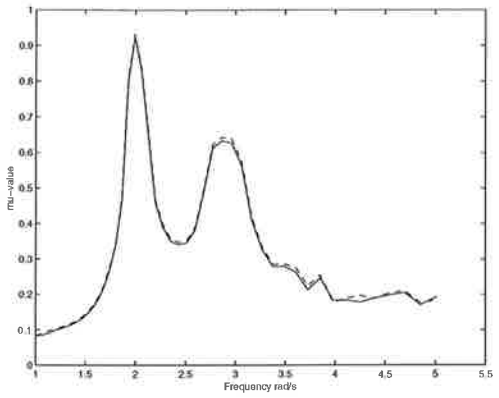

Figure 5.6 Comparison between stability μ of reduced model (solid) and original model (dashed). 7% deviation.

In this figure we see that the μ-plots for the two models are almost identical even though 22 of the parameters have been excluded from the model corresponding to the dashed line. The interpretation of this is that the model behaves almost as a model where the uncertainties of the power lines lies only in the reactance and the uncertainty of the power loads lies only in the resis-

tance. This is not surprising as we used a percentage of the nominal value to characterize the uncertainties. The resistance of the lines and the reactances of the loads are small comparing to their corresponding reactance or resistance.

Nevertheless, the ability of making a reduction like this is valuable as we might reduce the computation time when we analyze larger models, where we might use uncertainties in only some of the parameters and still get a realistic model.

6. Conclusions

6.1 Results

The objective of this thesis has been to present a method for transformation of a general differential-algebraic model into the diagonal perturbation form ready for robustness analysis with the μ-toolbox. During the transformation process the SYSTEM matrix will be as small as possible as both rank factorizations and the concept of minimal realizations are used. The modelling can be done in both Omola and Simulink but the Simulink modelling does not have the flexibility of the Omola modelling.

The routines are able to perform model reduction according to a simple algorithm. This reduction method is to remove some of the singular values close to zero during the rank factorization. This provides a fast way of doing a model reduction. However, no simple error bound estimation has been developed, which limits the usage.

Two models have been examined. First, a mechanical example taken from a benchmark problem of robust controller design. The design in the paper has been analyzed using the developed routines and the claims made in the article have been verified.

The second example that has been analyzed is a large model of the nordic power network. This model consists of 16 inputs, 16 outputs, 63 states and more than 500 parameters. Uncertainties in the loads and the transmission lines have been considered, and some aspects of robustness analysis have been examined. An estimate of the worst-case parameters have been extracted from the system using μ-analysis. Furthermore the effect of a removal of the transmission line between Gothenburg and Oslo, due to an accident has shown us that robustness analysis might be important to do in order to identify critical and sensitive points in a large model.

It should be noted that the transformation presented here might find uses in areas of automatic control far from robustness analysis. With no uncertainties in the model the routines easily presents a linear SYSTEM matrix of an Omola model which might be used for designing controllers or analyzing other features of the model than robustness.

All in all, the computer tools developed should find much use in the engineering field of today. Robustness analysis has not reached far outside the university field, but with the use of rather simple modeling tools like OmSim for robustness analysis, the usage in industry might rise.

6.2 Future work

Future work which uses the results in this thesis and develops them further are easy to see. First of all we need an automatic way of dealing with unmodeled dynamics and of analyzing nonlinearities as an uncertainty in a model. Direct usage of robustness routines in a simulation environment with a menu-driven user interface would be the ideal approach. Another weakness with the routines

is the demand for linear appearance of the parameters in the model matrices. Polynomial appearance might be dealt with using a loop transformation in the same way as we did with the introduction of positive uncertainties, but other nonlinear appearances must be dealt with in another way.

Furthermore, more examination of the computation time in the robustness routines will have to be done. The 16 generator power model presented here is by no means the largest model possible and the μ-toolbox needs to be optimized to be of use in the analysis of larger systems.

Yet another topic for further studies is to analyze a more realistic power system model and design a good robust controller for such a system. Further examination of the possibilities of making model approximations of power system models might also require pursuit.

A. An Introduction to μ-Analysis

A.1 Introduction

The goal of this chapter is to present the basic concepts of μ-analysis, which is one of the ways of examining the effects of uncertainties in a system. It is a powerful tool and many types of robustness problems can be analyzed with μ-analysis. The deeper theory and proofs of the theorems are omitted, but by reading and understanding the parts presented here, the rest of the thesis will be easier to follow.

The introduction presented here is based mainly on [4], [6] and [13].

A.2 Robustness analysis - the basic idea

In controller design and system analysis, we often rely on a model of the system. Every model of a physical system is, however, to some degree imperfect. The control designer without knowledge of robustness theory might be tempted to trust his model while designing, and later find that his design will cause instabilities in the closed-loop system. These instabilities may not show up during simulation as a simulation model is an exact model, and assumes no uncertainties. The effects when running the design on a real process, *e.g.* a nuclear plant, might be disastrous.

One way to deal with this problem is to assume that there exists uncertainties in the model. These uncertainties can be analyzed in a rigorous and mathematical way, and thus the designer will be able to successfully control the system without having a perfect model of it. The design should be able to achieve reasonable performance in the presence of modeling errors, *i.e.* it must be stable and the performance must not degrade to a high degree when exposed to different disturbances.

Some typical examples of uncertainties include unmodeled dynamics, linearizations of nonlinearities and unknown system parameters. In this thesis we mainly examine the latter. It is easy to imagine that it is quite impossible to know or measure the values of all our system parameters, such as values of electrical components, masses, moments of inertia *etc.*

An uncertain parameter P in the model can be written as

$$P = P_0 + \Delta_P,$$

where P_0 is the nominal value of the parameter and Δ_P represents an unknown perturbation of the nominal value. This perturbation is constrained to an uncertainty set which represents the possible values of the perturbation. It is, as will be seen later, more comfortable to let all the different uncertainties belong to the same uncertainty set. This will easily be accomplished by introducing a scaling factor to each perturbation, thus

$$P = P_0 + \delta_P P_\delta.$$

Now, the uncertainty set that δ_P belongs to can be of any magnitude as P_δ scales the uncertainty to the one corresponding to the perturbed parameter. The most obvious choice of uncertainty set is when the magnitude is unity, *i.e.*, if the parameter is real,

$$\delta_P \in [-1, +1]. \tag{A.1}$$

Naturally, if the parameter involves uncertainties in phase as well as in magnitude, a complex uncertainty set will have to be used. This does not change any of the discussion in this appendix, so let us assume that the uncertainty set is described by (A.1).

A.3 The diagonal perturbation form

By introducing uncertainties as done in the section above, we have a way of analyzing the effects of these uncertainties on the system. However, as the parameters and the corresponding uncertainties may appear in the system in many different places, it is necessary to introduce a way of separating the uncertainties from the rest of the system. The model obtained by "extracting the δ-s" is called the diagonal perturbation form.

Let us say that we have a linear dynamic system represented by a transfer function matrix G like the one in Figure A.1. In this system we have n

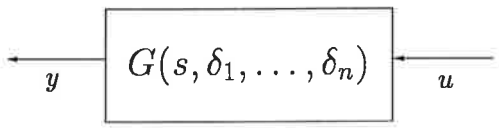

Figure A.1 Linear dynamic system with uncertainties.

uncertain parameters, *i.e.* n different δ-s representing the uncertainties. Each δ is constrained to the uncertainty set described in (A.1). If we extract these uncertainties to a $n \times n$ diagonal matrix, we get the diagonal perturbation form. The structure of this matrix which we will call the Δ-matrix is

$$\Delta = \mathrm{diag}(\delta_1, \ldots, \delta_n) = \begin{bmatrix} \delta_1 & 0 & \ldots & 0 \\ 0 & \delta_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & \ldots & \delta_n \end{bmatrix}. \tag{A.2}$$

It is now possible to rewrite the transfer function matrix, given in Figure A.1, on block-diagram form. The system will correspond to the system in Figure A.2. The transfer function matrix $M(s)$ is the transfer function matrix "seen" by the Δ-matrix and the system inputs. We have introduced n new outputs and n new inputs to M, called v and w respectively, corresponding to the different uncertainties in Δ.

This is basically the diagonal perturbation form. However, in this definition we view the system from an input-output point of view, thus implying that the matrices consist of transfer functions. This means that we have rational functions of the transform variable s as elements in the matrices. This will be

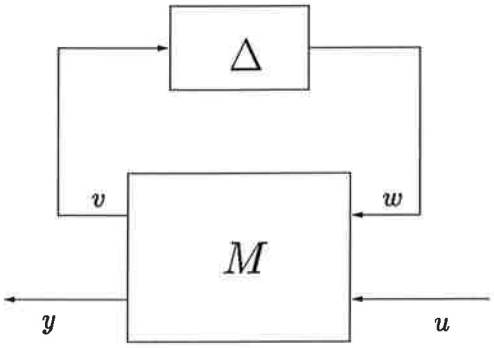

Figure A.2 The diagonal perturbation form.

a major obstacle when analyzing a system with computer software that does not handle symbolic calculations. In order to deal with this it is common to consider the matrices as a state-space realization of the transfer functions. The dynamics, *i.e.* the s-part, is then extracted in the same way as the uncertainties making the system analyzable.

A.4 Robust stability

In order to analyze the robust stability of the system, let us examine the system without the system inputs u and the system outputs y. We assume naturally that the system fulfills nominal stability, *i.e.* stability when $\Delta = 0$. If we partition the M-matrix, given in Figure A.2, as

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

we see that for stability analysis of the system the matrix M_{11} is critical. This matrix corresponds to the transfer functions connecting the n inputs and outputs to the Δ-loop in Figure A.2. Let us therefore reduce the system to a system, consisting of the internal dynamics and the uncertainties, *i.e.* as in Figure A.3.

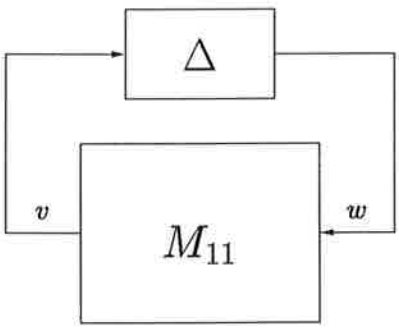

Figure A.3 Closed loop for analyzing robust stability.

The loop equations of the system pictured in Figure A.3 are

$$\begin{aligned} v &= M_{11}w \\ w &= \Delta v, \end{aligned}$$

which gives

$$(I - \Delta M_{11})w = 0.$$

It is easy to see that the characteristic equation is given by

$$\det(I - \Delta M_{11}) = 0. \tag{A.3}$$

A condition for robust stability can be deducted from (A.3). We see that as long as equation (A.3) is not fulfilled, the only possible solutions to the loop equations of Figure A.3 are the trivial $w = v = 0$, which implies stability. We have to examine if (A.3) is fulfilled for some Δ at some frequency. Unfortunately, we do not know the uncertainty Δ specifically and thus we need a more thorough method of examining the system.

Intuitively, the smaller the perturbation, the more likely it is that the system is stable, since it approaches the nominal, stable system. We clearly need some way of measuring the size of the perturbation matrix Δ. This is provided by the largest singular value, $\bar{\sigma}$, of Δ. It is now possible to define the Structured Singular Value, or the μ-value as it is often called. It is given by

$$\mu_{\Delta}(M_{11}) = \frac{1}{\min\{\bar{\sigma}(\Delta) : \det(I - M_{11}\Delta) = 0\}}. \tag{A.4}$$

In words, the μ-value is the inverse of the smallest perturbation that causes instability according to equation (A.3). The size of the perturbation is measured with the largest singular value, $\bar{\sigma}$. Note that the μ-value is frequency-dependent since M_{11} is frequency-dependent.

We started out with the assumption that our system must be able to manage perturbations with a maximum magnitude of 1 without losing stability, in order to fulfill the specifications. This implies that if the μ-value is larger than 1 at any one frequency, the system is not robustly stable. This is the main result that is used in μ-analysis. Robust stability is provided if

$$\mu_{\Delta}(\omega) < 1, \forall \omega \in [0, \infty[. \tag{A.5}$$

This might appear to be just as difficult to calculate and analyze as the first stability criterion which was deducted from (A.3). However, some good numerical algorithmsexist, which calculates an upper and a lower bound on μ. This makes a feasible way of analyzing our system.

A.5 Robust performance

Let us now introduce the concept of inputs and outputs to the model,*i.e.* we examine the model as it was given in Figure A.2. It is easy to imagine that the concept of robust stability discussed in the previous section may not be enough for analysis of a system. When the system is exposed to different exogenous disturbances, the performance might be degraded and the behaviour will be bad. This performance loss might make the system useless even if it fulfills the robust stability criterion we discussed earlier. This leads us to introduce the concept of robust performance.

Before we approach the performance problem, let us introduce the concept of linear fractional transformations, LFT. An LFT is basically a matrix

transformation which determines the relationship between the input u and the output y given in Figure A.2. The transformation can be done easily if we start out from the relationships

$$\begin{aligned} v &= M_{11}w + M_{12}u \\ y &= M_{21}w + M_{22}u \\ w &= \Delta v. \end{aligned} \tag{A.6}$$

When rewriting v given by (A.6) we get

$$v = M_{11}\Delta v + M_{12}u = (I - M_{11}\Delta)^{-1}M_{12}u$$

and by substituting this expression for v in the other equations in (A.6) we get

$$y = (M_{21}\Delta(I - M_{11}\Delta)^{-1}M_{12} + M_{22})u$$

which gives the desired relationship between the input u and the output y. We now define the upper linear fractional transform $F_u(M, \Delta)$ as

$$F_u(M, \Delta) = M_{21}\Delta(I - M_{11}\Delta)^{-1}M_{12} + M_{22}. \tag{A.7}$$

The matrix norm of the LFT can be interpreted as the gain from the inputs to the outputs and if this gain is larger than 1, a disturbance input will not be attenuated and the performance is unacceptable. This can be analyzed in a rigorous way since the concept of μ is closely related to the largest singular value, *i.e.* the so called induced 2-norm.

In order to examine robust performance, some additional steps need to be taken compared to the stability analysis. Let us assume that the system has q_1 inputs and q_2 outputs. Then we introduce an additional perturbation, Δ_F, which we use to close the open loop. The system can be seen in Figure A.4.

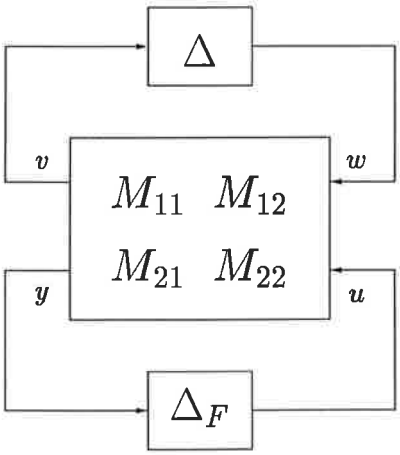

Figure A.4 Closed loop for analyzing robust performance.

The fictitious uncertainty block, Δ_F, is a full complex uncertainty block, *i.e.*

$$\Delta_F \in C^{q_1 \times q_2}.$$

This makes the problem virtually the same as the one we had when we considered robust stability. The μ-values can be calculated in the same way as before, by using (A.4),with the difference that we now have another augmented perturbation matrix, Δ_P, given by

$$\Delta_P = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta_F \end{bmatrix}.$$

We naturally also use the complete transfer function matrix M during the analysis, and thus we get the different μ-values for the robust performance test. In Figure A.5 we see the structure of the performance test when it is written as a stability test.

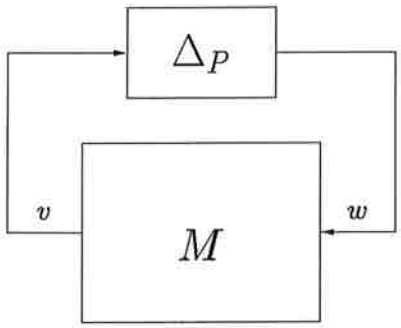

Figure A.5 Closed loop for analyzing robust performance.

In the same way as before we draw our conclusions of the robustness from these μ-values. These conclusions are based on the equivalence

$$\mu_{\Delta_P} < 1 \Longleftrightarrow \begin{cases} \mu_\Delta(M_{11}) < 1 \\ \| F_u(M, \Delta) \| < 1 \end{cases} \tag{A.8}$$

The equivalence given in (A.8) implies that if the μ-value given from the system in Figure A.5 is less than 1, we can draw the conclusions that the system given by Figure A.2 is robustly stable and that it fulfills the performance criterion to attenuate disturbances. The system is then said to perform robustly.

A.6 Finding the worst case parameters

Even if a model is robustly stable it can still perform so badly that it is useless. This concept is measured by the robust performance test presented in the previous section. However, it might be interesting to find the parameter set corresponding to the case of worst performance of the system. By worst performance we mean the highest gain from the inputs to the outputs. This can be done via clever use of the μ-toolbox and the robust performance theory.

First, and most important of all, it is vital to know that the standard calculations of the lower bound of μ also provides the perturbation Δ which causes this μ-value. By using (A.8) we see that if the system is robustly stable and the μ-value is less than 1 then we can draw the conclusion that the gain from input to output, the linear fractional transform, is less than 1. The critical

step is to introduce a weight on the input or the output which gives a μ-value of 1. Then we can draw conclusions of the behaviour of the performance of the system. For example we know that the system is robustly stable for all $\| \Delta \| < 1$, fulfilling the desired structure of Δ. We also know that the gain from the inputs to the outputs, given by the norm of the upper LFT, is less than one. Consider Figure A.6.

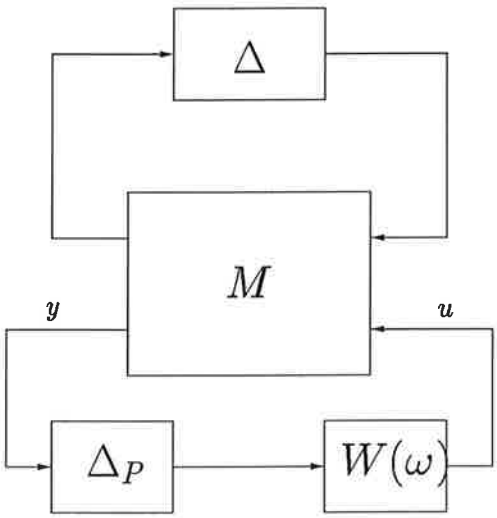

Figure A.6 System with performance block Δ_P and iteration weight $W(\omega)$.

By adjusting the weight $W(\omega)$ we are able to scale the importance of Δ_P at different frequencies and thus make it possible to draw conclusions of how the system reacts to all the interesting combinations of perturbations. By incorporating the iteration weight in the matrix M and putting the normal Δ and the performance perturbations Δ_P together we get the system as in Figure A.7.

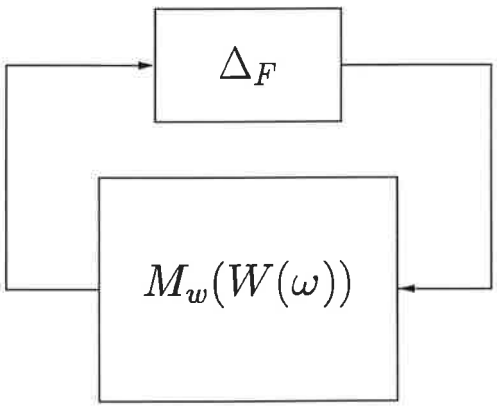

Figure A.7 System depending on $W(\omega)$ with perturbation block.

Conclusions can now be drawn, using the theorem (A.8). We see that

$$\mu(M_w(W(\omega)), \Delta_F) < 1 \Longrightarrow W(\omega) \| F_u(M, \Delta) \| < 1, \qquad \text{(A.9)}$$

where the right hand side can be used to get an upper bound on the gain from

the inputs to the outputs as

$$\| F_u(M, \Delta) \| < W(\omega)^{-1}.$$

If we use (A.9) with different weights W we should be able to, if the original system is robustly stable, get an upper bound of μ just about one. The inverse of the iteration weight which gave that μ-value gives an estimate of the worst-case gain. From the lower bound of μ we get a set of perturbations which give the worst behaviour, *i.e.* the largest gain.

By using these worst parameters we can easily calculate the gain the system will have with these parameters. Naturally the maximum gain is less than the maximum gain implied by the inverse of the iteration weight. This is due to the difference between the lower and upper bound in the μ-computations and the fact that the iteration sequence never reaches exactly $\mu = 1$. This will however provide us with a relatively easy way of finding an estimate of the worst case parameters.

B. Matlab routines

Introduction

This appendix describes the significant features and usage of the matlab functions that implements the transformations and reductions that have been described in the thesis. The different functions are described in alphabetical order. Note that information of the various function is also available through the on-line Help facility. It is also vital to note that the some of the functions use the subfunction facility of Matlab-5.0, so Matlab of earlier versions will not be appropriate.

DEINITPARS

Purpose

To clear the workspace of all the global variables introduced by INITPARS.

Synopsis

```
deinitpars;
```

Note that it requires a string `parnames` in the matlab workspace before usage. This string should contain the parameter names of the model.

Example

See the function SIMULINK2SS.

INITPARS

Purpose

To prepare the matlab workspace for the use of SIMULINK2SS by making the parameters global.

Synopsis

```
initpars;
```

Note that it requires a string `parnames` in the matlab workspace before usage. This string should contain the parameter names of the model.

Example

See the function SIMULINK2SS.

OMOLA2DAE

Purpose

Make matlab state-space matrices from the M-files presented by the Omsim environment, from the values of the different parameters in the Omsim model.

Synopsis

`[A,B,C,D,E,F,G,H,J,K,L]=omsim2ss(pars,ynbr)`

Description

Input The vector `pars` contains the values of the different parameters which are in the Omsim model. The order of these parameters must be the same as the order presented in the M-file `parfile.m` which is presented by OmSim. The vector `ynbr` gives the numbers of the outputs which the user wants in the matlab SYSTEM matrix. The outputs to choose among can be seen in the text file variables.txt, which is presented by OmSim.

Output The matrices `A,B,C,D,E,F,G,H,J,K,L` are the model matrices of a differential-algebraic model as given by (2.1), *i.e.*

```
E*x'=A*x+B*u+F*z
G*z=H*x+J*u
y=C*x+D*u+K*x'+L*z
```

Example

Suppose that we have created a model in Omola with two parameters `par1` and `par2`. If they appear in `parfile.m` in the order `par2,par1` and you would like the parameter values to be `par1=1` and `par2=2` the matlab commands are

```
>>modelpars=[2 1];
>>[A,B,C,D,E,F,G,H,J,K,L]=omsim2ss(modelpars);
```

where the model matrices given are the ones corresponding to the parameters in `modelpars`.

SIMULINK2SS

Purpose

Make matlab state-space matrices from a Simulink model, from the values of the different parameters appearing by name in the model.

Synopsis

```
[A,B,C,D,E]=simulink2ss(pars, model,parnames)
```

Description

Input The vector `pars` contains the values of the different parameters which are in the Simulink model. The string `model` is the name of the Simulink model analyzed, and the string `parnames` contains the names of the parameters in the model separated with blanks.

Output The matrices `A,B,C,D,E` are the state-space matrices of the current model, given by

```
E*x'=A*x+B*u
y=C*x+D*u
```

Note that, since Simulink is limited, the matrix `E` is the identity matrix.

Example

Suppose that we have created a model `myModel` in Simulink with two parameters `par1` and `par2`. In order to use SIMULINK2SS we nee to make the parameters global with INITPARS. First define the string `parnames`

```
>>parnames='par1 par2';
```

Then make them global with

```
>>initpars;
```

Suppose that we want the parameter values to be `par1=1` and `par2=2`. Then we get the model matrices as

```
>>modelpars=[1 2];
>>[A,B,C,D,E]=simulink2ss(modelpars,parnames,'myModel');
```

When finished with this model you should do

```
>>deinitpars;
```

in order to remove the global variables.

DAE2DPF

Purpose

Transform a model given as eleven model matrices in a differential-algebraic equation system to the diagonal perturbation form.

Synopsis

`[M,block,reduced]=dae2dpf(f,pars,dpars,option,in1,in2,in3)`

Description

Input The string `f` is the name of another matlab function with the synopsis `[A,B,C,D,E]=f(pars)` or `[A,B,C,D,E,F,G,H,J,K,L]=f(pars)`, *i.e.* `f` is a function which gives the model matrices from a set of parameters. The nominal parameters are given in `pars`. The perturbations which we want to examine are in `dpars`, *i.e.* the actual parameters can vary in a certain set given by deltapars. The function can handle either functions which produces a set of state-space equations or a case where all the eleven model matrices described in the thesis are given.

The string option can handle three cases:

'n': normal (default)	In this case no `input` should be given, except because of the case with Simulink or Omola.
'pos': positive	In this case some of the parameters have a positive perturbation [0,1]. The vector `in1` gives the numbers of these parameters.
'r': reduction	A reduction during rank factorization has taken place. `In1` is is the level of reduction. Default is 0.1;

If the function `f` is Omola2dae, an input corresponding to the outputs from the Omola model has to be given. All inputs corresponding to an option are given before.

If the function `f` is Simulink2ss, two additional inputs corresponding to the Simulink modelname and the Simulink parameter names has to be given. The inputs are strings. All inputs corresponding to an option are given before.

Output The matrix `M` is a SYSTEM matrix which describes the system on the diagonal perturbation form. The matrix `block` is the structure of the perturbation matrix. In the case described in this thesis repeated real blocks are used, which is denoted in the ith row of `block` as `[-r 0]` where `r` is the size of the ith perturbation block. The vector `reduced` contains the numbers of those uncertainties which are removed entirely from the model as a result of the reduction. If no reduction has taken place, it is zero.

Examples

Suppose that the model matrices are given by the M-file `'mySys.m'`. The system depends on 2 parameters and their nominal values are given in `np` and their maximum perturbation from the nominal values are given in `dp`. Let us say that $Parameter1 \in [1, 2]$ and $Parameter2 \in [0.51]$. The matrices `M` and `block` in the diagonal perturbation formulation are then given by the commands

```
>>np=[1.5 0.75];
>>dp=[0.5 0.25];
>>[M,block]=dae2dpf('mySys',np,dp);
```

Let us now assume that for some reason the user would like to examine the case where the second parameter has a nominal value of 1 and a positive uncertainty set, *i.e.* the actual parameter value is never below the nominal value. We also assume that there is now perturbation in the first parameter. The commands are

```
>>np=[1.5 1];
>>dp=[0 1];
>>pospar=2;
>>[M,block]=dae2dpf('mySys',np,dp,'pos',pospar);
```

Let us now suppose that we would like to create the same system as we did in the first example, but we would like to reduce the resulting system to a smaller one if that is possible without changing the properties too much. We set the reduction level to 1. The reduced M-matrix of the diagonal perturbation formulation is presented in `Mr` and the structure of its corresponding perturbation matrix in `blockr`. If one of the parameters has been excluded from the model the function presents a message to the user that this particular parameter has been removed. `Redpar` will also contain the number of the parameter.

```
>>l=1;
>>np=[1.5 0.75];
>>dp=[0.5 0.25];
>>[Mr,blockr,Redpar]=ss2dpf('mySys',np,dp,'r',l);
```

For example, if the second parameter has been reduced, reduced contains 2.

If we do the same analysis procedure as above using Omola we have to give some more inputs. We define the 6th and 7th terminal of the Omola model as outputs. The outputs are seen in 'variables.txt'. It is also vital to remember that no default value of `option` now exists. The instruction sequence is

```
>>% Normal case
>>np=[1.5 0.75];
>>dp=[0.5 0.25];
>>outputs=[6 7];
>>[M,block]=dae2dpf('omola2dae',np,dp,'n',outputs);
>>% Positive parameters
>>np=[1.5 1];
>>dp=[0 1];
>>pospar=2;
>>[M,block]=dae2dpf('omola2dae',np,dp,'pos',pospar,outputs);
% Reduction during matrix generation
>>l=1;
>>np=[1.5 0.75];
>>dp=[0.5 0.25];
>>[Mr,blockr,Redpar]=ss2dpf('omola2dae',np,dp,'r',l,outputs);
```

Finally, in order to avoid mistakes we show the same instruction sequence using a Simulink model. The model analyzed are called `'myM'` and the parameter names are called `parnames`. It is assumed that the parameter names are made global according to the analysis procedure given under Simulink2ss.

```
>>f='simulink2ss';
>>% Normal case
>>np=[1.5 0.75];
>>dp=[0.5 0.25];
>>[M,block]=dae2dpf(f,np,dp,'n','myM',parnames);
>>% Positive parameters
>>np=[1.5 1];
>>dp=[0 1];
>>pospar=2;
>>[M,block]=dae2dpf(f,np,dp,'pos',pospar,'myM',parnames);
% Reduction during matrix generation
>>l=1;
>>np=[1.5 0.75];
>>dp=[0.5 0.25];
>>[Mr,blockr,Redpar]=ss2dpf(f,np,dp,'r',l,myM',parnames);
```

WORSTPOINT

Purpose

Extract the worst parameters, *i.e.* the parameters that brings on the worst input-output behaviour of the system. The current implementation supports the use of Omola only.

Synopsis

`[wpars,bnd]=worstpoint(np,dp,ynbr,w,startW,tol)`

Description

Input The inputs consists of the nominal parameters `np` and the perturbation of the uncertain parameters `dp` as described in the other functions. The input `ynbr` contains the numbers of the outputs from Omola, see Omola2dae. The frequencies for which perturbation are checked are `w` and the startestimate of the iteration weight is `startW`. The default value is 1, but faster computations will be appropriate if a more correct value is given. The input `tol` signifies how far from the correct μ-value the iterations will go. The default is 1e-2. It is assumed that the current Omola model is robustly stable in the designated parameter area. The function produces an error message if it is not. Convergence problems of the μ-routines may also cause an error message.

Output The outputs are `wpars` with the worst parameter combination. The output `bnd` corresponds to an upper bound of the gain. If the worst parameters are entered in the model the gain corresponding to this parameter set is always less than this bound, if no numerical problems have arisen.

Example

If we have a robustly stable Omola model represented by the three matrices 'Matrices.m', 'Parfile.m' and 'variables.txt' in the current Matlab path, we can extract the worst parameter combination of this model. If we assume that the interesting frequency area lies between 1 and 3 rad/s and if we assume that the numbers of the outputs in the Omola model are defined in `Omnbrs` the instructions are

```
>>w=linspace(1,3,30);
>>worst=worstpoint(np,dp,Omnbrs,w)
```

The nominal parameters are defined in `np` and the perturbed are in `dp` just as in the other functions. Note that this iteration procedure may last a couple of hours if the model is large enough. Messages are displayed during the iteration and the user can see how fast the iterations are progressing.

However, some speedups can be made if information of the model behaviour is known. If we for example know that the gain of the model is at least above 1000, then we know that the weight W at the most should be 1/1000, which can be used as input. Furthermore, the tolerance of the iterations, *i.e.* how close to μ=1 the iterations will bring the performance μ, can be larger. This will worsen the estimate of the worst point but it will speed up the function call greatly as smaller steps in W are taken the closer to μ=1 we come. Instructions for this are

```
>>West=1/1000;
>>tol=0.1;
>>worst=worstpoint(np,dp,Omnbrs,omega,West,tol);
```

This will make the function call faster in our case. The tolerance is high which makes the results less accurate.

C. The analysis procedure using Omola and Matlab

This appendix presents an example of a typical robustness analysis of a model developed in Omola using the Matlab routines presented in this thesis and the μ-toolbox. The aim is to simplify for the user and to further enhance the understanding of the functions.

Defining an Omola model

Before starting the Omsim environment, the linearizer which produces the matlab M-files needs to be activated. This is done by writing

```
>setenv OMSIM_LINEARIZER_ON 1
```

in the X-term window in which Omsim will be started. Note that this procedure will make it impossible to use OmSim for actual simulation. This is a drawback, but is solved by starting two OmSim. Start Omsim. Next, define a model of the system using Omola. Readers unfamiliar with Omola and Omsim are recommended to read *e.g.* [3], which contains a minor tutorial.

The Matlab M-files are automatically created when Omsim starts a simulator of the model. The files 'matrices.m', 'parfile.m' and 'variables.txt' have been created in the directory which OmSim was started in. We are now ready to perform the actual analysis of the model.

Analyzing in Matlab

Let us assume that we want to examine if our model is stable for a certain set of parameters defined by `npars` and `dpars`. We check in variables.txt to see which outputs are vital. These are defined in `outputs`.First we transform the system to the diagonal perturbation formulation.

```
>>[M,block]=ss2dpf('omola2dae',nompars,dpars,'n',outputs);
```

Remember to have the M-files created by Omsim in your current Matlab path. We now have a SYSTEM matrix which describes the model and the perturbation. In order to use the `mu`-command we need to make a frequency response out of this `M`. Let us say that we now that the interesting frequencies are between 1 rad/s and 100 rad/s.

```
>>omega=logspace(0,2,50);
>>[Mresp]=frsp(M,omega);
```

When we only examine the stability we do not need to concern ourselves with the parts of `M` that corresonds to the inputs and output. Thus, we select a subsystem corresponding to the M_{11} matrix in Appendix A.

```
>>M11=sel(Mresp,1:sum(abs(block(:,1))),1:sum(abs(block(:,1))));
```

Now, we are able to do the actual calculation of μ. As it is a problem with real parameters there might be problems with convergence of the μ-calculations, but this might be taken care of by introducing a small complex perturbation or by examining robust performance instead. This is described in [4].

```
>>bnds=mu(M11,block);
>>vplot(bnds)
```

In the plot we are now able to see the *mu*-values for 50 frequencies between 1 rad/s and 100 rad/s. If the μ is less than 1 for all frequencies this implies that the system is stable for the parameter values defined by `nompars` and `deltapars`.

In a case where we are interested in robust performance for the same SYSTEM matrix, *i.e.* for the same parameter set, we do not need to extract the M_{11} matrix. Suppose we have 2 inputs and 3 outputs. The instruction sequence will then be

```
>>perfblock=[3 2];
>>newblock=[block;perfblock];
>>bnds=mu(Mresp,newblock);
>>vplot(bnds)
```

This will give us a performance analysis of the Omola model.

D. References

[1] Akke,M. *Power System Stabilizers in Multimachine Systems.* Licentiate Thesis, Automatic Control,LTH. 1989

[2] Andersson, L. *Comparison and Simplification of Uncertain Models.* Licentiate Thesis, Automatic Control,LTH. 1997.

[3] Andersson, M. *OmSim and Omola. Tutorial and User's Manual.* Automatic Control, LTH. 1995.

[4] Balas, G, Doyle, J, Glover, K, Packard, A, Smith, R. *μ-Analysis and Synthesis Toolbox.* Mathworks 1993.

[5] Gahinet, P, Nemirovski, A, Laub, A, Chilali, M. *LMI Control Toolbox.* Mathworks 1995.

[6] Holohan, A. "A Tutorial on μ-analysis". Robust and Adaptive Control Workshop. University of Dublin. 1994

[7] Johansson, R *System Modeling and Identification.* Prentice-Hall 1993.

[8] Lo, J and Pang, G. "Solution to a Benchmark Control Problem using the CDM Design Method."

[9] Mattsson, S-E. "Object-Oriented Modelling of Hybrid Techical Systems", Lecture Notes,Automatic Control, LTH. 1996

[10] Mattsson, S-E. "Modelling of Power Systems in Omola for Transient Stability Studies", Automatic Control, LTH. 1992

[11] Pärt-Enander, E, Isaksson, P, Melin, B, Sjöberg, A. *Användarhandledning för MATLAB 4.2.* Uppsala Universitet 1995.

[12] Spanne S. *Matristeori.* KFS 1995.

[13] Zhou, K, Doyle, J and Glover, K. *Robust and Optimal Control.* Prentice-Hall 1994.

[14] Åström, KJ, Wittenmark, B. *Computer-Controlled Systems.* Prentice-Hall 1990.

[15] Åström KJ. *Reglerteori.* Almqvist & Wiksell 1968.