Student Thesis IST-77

Neocybernetics in traditional control applications

Adaptive rotor vibration damping in electrical machines

cand. kyb. Heiko Patz

Supervisor: Prof. Dr. Tech. Heikki Hyötyniemi
Prof. Dr.-Ing. Frank Allgöwer

University of Stuttgart
Institute for System Theory and Automatic Control
Prof. Dr.–Ing. F. Allgöwer

August 2007

Preface

The thesis at hand was written at Control Engineering Laboratory of the Helsinki University of Technology.

Special thanks to my instructor Prof. Dr. Tech. Heikki Hyötyniemi who helped me in all theoretical question in spite of sickness. His support is remarkable and his ideas helped me to broaden my cybernetic horizont. I wish him the best for his health.
I would like to mention Dr. Tech. Kai Zenger, too. He helped me a lot by correcting and questioning my thesis.
Thanks to Prof. Dr.-Ing. Frank Allgöwer for the friendly organizational support.

I would also like to thank my room mates Juha, Maurizio and Anssi who always were open for my questions.

I praise my Lord Jesus Christ for giving creativity and support in all respects which made this wonderful time in Finland possible.

Hereby I, Heiko Patz, declare that I made the thesis at hand self-dependent and only with the help of the mentioned references.

Espoo, August 9th 2007

Contents

List of Figures

Chapter 1

Introduction

Generally cybernetics is no good way to avoid complexity in live. But anyway a lot of efforts are driven to find the simplest way as possible to handle complex systems. Despite of all efforts however the world still offers plenty of amazing systems which are still too complex to control them. In this thesis a theory is utilized which attacks complexity which very simple ideas. The method is completely data based and acts locally (single signal) in that way that global structure (whole system) is found. In this sense it has got a *bottom up perspective*. This theory is called *neocybernetics*.
The neocybernetics theory then is tested on realistic linear models of a cage induction electrical machine. Different kind of SISO and MIMO controllers are applied, simulated and their results are presented. It is shown how the theory works for this concrete case. Finally all important problems are discussed and explained.

Chapter 2

About neocybernetics

Neocybernetics is a new approach to handle complexity. Therefore it uses well known mathematical tools (mainly linear algebra and Principal Component Analysis), but it is the way of thinking and interpreting complex systems which makes the difference.

While cybernetics generally tries to find abstract descriptions of complex systems one can say that neocybernetics tries to describe the *emergence* of new cybernetic structure. This means that it aims to describe what happens when changing to another level of abstraction (e.g. small scale view on molecules - large scale view on pressure, temperature). On this kind of border new structure appeares, which is called emergence in a neocybernetic context.

A whole overview to the neocybernetic theory and detailed exploitation of the neocybernetic ideas are given in [1].

2.1. Basic ideas

To have a closer look at the basic principles of neocybernetic modeling, there are three areas that have to be discussed: the dynamic balances in systems, the high-dimensionality and the linearity pursuit of modeling.

2.1.1. Balances

In the process of modeling emergence one is mainly interested in the final balance of a system rather than the process which leads to that balance. Therefore it is assumed that the systems are stable. Real systems are always stable in this sense that they reach a final balance. If they were not stable, they would consume infinite energy which can only happen in theory. (That does not mean that the reached balance is optimal or good.) This stability is caused by feedbacks. The feedback structure is not interesting as long as it maintains stability. Therefore we can assume stability and study the *emerging pattern* which are formed when reaching the steady state.

2.1.2. High dimensionality

In order to deal with complexity one is automatically faced with high dimensionality. In neocybernetic models environment data is captured in *high dimensional data vectors*. In this context one acts on the assumption that environment data is highly redundant. This has to be kept in mind when evaluating neocybernetics, because the assumption does not hold for the system presented in this thesis. The actuator rotor system as considered here is rather low dimensional (not more than order 20) and the inputs (environment) are nearly independend of each other - no redundance. But it is an example how the method works, and one should notice that the applied principles would also work in high dimensional problems which are typical e.g. in biological systems [1].

2.1.3. Distributedness

Distributedness is another kind of complexity. Most cybernetic approaches use concentrated models assuming that the whole information can be handled in one central unit. Neocybernetics uses the idea of many distributed agents (subsystems) which are connected but act locally in the sense that they use only locally available data (outputs of directly connected subsystems). Global structure is seen as a result which will emerge if the agents act in an appropriate way. All methods presented implement this idea. This kind of *emergent*

pattern can be seen for example in 2.3.2.

2.1.4. Linearity

Neocybernetic systems are entirely linear. In the current trend of emphasizing nonlinear methods this looks like a too severe restriction. But there are good reasons for choosing linear models:
As neocybernetics searches for the *essence* of a system, passing different levels of abstraction, one is interested in scalability. This can only be offered by linear systems. Furthermore, considering only the balances of a system relaxes the assumption of linear systems, because even nonlinear systems can have a linear input-output relation in steady state.
And of course, there exist powerfull tools to deal with high dimensionality as mentioned above. Some more aspects are given in [1]. It turns out that linear models are the preferred choice to handle neocybernetic systems. Therefore in this thesis only linear tools and models are used.

2.2. Properties and assumptions of neocybernetic systems

In this section the basic discussed above are formulated in a more concrete way. It is actually a summary of Sec. 3.1 of [1].

2.2.1. Input output pattern / existence of balance

Assume a linear system with a n-dimensional state x and some input vector w is given by

$$\frac{dx}{dt} = -Ax + Bw \tag{2.1}$$

As the system is assumed to be stable (A positive definite) it will reach a steady state $\dot{x} = 0$ (for $t \to \infty$)

$$0 = \quad -Ax + Bw \tag{2.2}$$

$$\Leftrightarrow Ax = \quad Bw \tag{2.3}$$

Note that (2.1) can also be seen as a reduced model of an originally higher order one. In this case the state x is just the output $x = y_{ext} = C_{ext}x_{ext}$ of the higher order system $\dot{x}_{ext} = A_{ext}x_{ext} + B_{ext}w$. In this sense x can be the state or the output of a system. For the sake of constistent notation in this thesis x is always used to denote the currently available part of the state.
Now a symmetric A is assumed. If it is not symmetric one can continue by multiplying (2.3) with A^T. If (2.3) is integrated by x

$$\mathcal{J}(x,u) = \frac{1}{2}x^T Ax - x^T Bw \tag{2.4}$$

is obtained. This equation can be intepreted as a *cost criterion* which is minimized by the system. This leads to an interesting observation when it is compared to the next section.

2.2.2. Elasticity of systems

All neocybernetic systems are identical to (or even can be defined by) elastical systems. To explain this an example is used: Study a spring which is streched by a length s using an external force F. For this spring internally and externally stored (positively defined) energies are given as:

- Due to the potential field:

$$W_{ext} = \int_0^s Fds = Fs$$

- Due to the internal tensions:

$$W_{int} = \int_0^s ksds = \frac{1}{2}ks^2$$

  (k denotes the spring constant).

If this example is now extended to a set of n springs affected by m forces, a closed matrix formulation for the potential energies can be

found. Therefore all interaction factors can be collected in matrices A and B. Then the potential energies for the whole system look like

$$W_{int}(s) = \frac{1}{2} \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix}^T A \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} \tag{2.5}$$

and

$$W_{ext}(s, F) = \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix}^T B \begin{pmatrix} F_1 \\ \vdots \\ F_m \end{pmatrix} \tag{2.6}$$

Note that A must be symmetric and positive definite ro represent stable structures sustaining external stresses. If the forces are seen as inputs and if the distances are seen as states the cost crition (2.4) becomes the difference of the given energies. Hence the set of springs seen as a damped dynamic system like (2.1) ends up in the minimum of the difference of potential energies. This aligns with the *Principle of minimum potential (deformation) energy* [6]. This idea of elastic mechanical systems is generalized for neocybernetic systems: No matter what a model physically represents its inputs are considered as gerneralized forces u causing some deformation x. And if the external stress is removed the system will go back to orginial states. This leads to a general *assumption* which is assumed for all neocybernetic system:

> *If the input of a system increases the value of the state decreases.*

Hence an applied negative feedback to this kind of system always remains stable.

2.3. Neocybernetic control

In this section it is shown how a controller can be introduced to the system defined in (2.1). Neocybernetics always considers closed loop systems because real systems are always connected to the environement, and thus the environment always implements a feedback (See

Sec. 3.2.1 in [1]). Following this idea the system (2.1), assumed stable, can be thought to be stabilized through an environmental feedback. Hence the system is seen as a result of an underlying structure which consists of an open loop system

$$\frac{dx}{dt} = -\tilde{A}x + \tilde{B}u \tag{2.7}$$

whose m-dimensional input is given by the state feedback

$$u = \varphi(x - w) \tag{2.8}$$

If (2.8) is substituted in (2.7) we obtain

$$\frac{dx}{dt} = -(\tilde{A} - \tilde{B}\varphi)x - \tilde{B}\varphi w \tag{2.9}$$

One can easily verify that this result is equal to (2.1) if the simplifications $A = \tilde{A} - \tilde{B}\varphi$ and $B = -\tilde{B}\varphi$ are used.
Obviously w is the reference value which x is compared to. This input is of minor importance to the basic result if it is slow compared to the system dynamics. As it is always zero in the systems used for simulations (meaning that vibrations are controlled to zero) it is generally assumed to be zero later on. From the neocybernetic perspective the open loop system (2.7) never exists separateley. But for the sake of applying the neocybernetic ideas to technical systems, which are typically open loop (like (2.7)) until an controller is applied (like in (2.9)), system and controller is formulated seperately now.

2.3.1. Adaptive controller

Neocybernetic controllers are a kind of mirror image of the steady state of the system which is to be controlled. Therefore the mappings φ and ϕ^T (inverse mapping of φ) are iteratively designed to control the system. The controller is introduced here rather intuitivly but in Sec. 2.3.2 it is shown which interesting results it leads to.
In (2.8) φ is introduced as just any state feedback which controls the (sub-) system (2.7). The following equation shows how this feedback is constructed:

$$\varphi := B \cdot \mathrm{E}\{ux^T\} \tag{2.10}$$

u and x are measurement data. Therefore the expectation operator $\mathrm{E}\{\cdot\}$ in a neocybnetic context is defined as the *mean value of any stationary, periodic funktion* $f(t)$ for a certain time intervall T.

$$\mathrm{E}\{f(t)\} = \frac{1}{T} \int_{t_0}^{t_0+T} f(t)dt \tag{2.11}$$

Of course x and u are not generally periodic. $\mathrm{E}\{ux^T\}$ mean that the periodic part of x and u (after reaching a stationary state) is used. It might be better to write $\mathrm{E}\{\bar{u}\bar{x}^T\}$ but in accordance to [1] the bars are left out.
(2.10) constructs a statical model depending on measurement data. If desired the other direction (a mapping from u to $x - w$) can be modelled, too.

$$(x - w) = \phi^T u \tag{2.12}$$

where

$$\phi^T := Q \cdot \mathrm{E}\{xu^T\} \tag{2.13}$$

Here B and Q are diagonal matrices whose diagonal elements b_i and q_i are arbitrary parameters. Both definitions hold only for the *closed loop system*. Therefore it does not make any sense to use both definitions at the same time: It would be an algebraic loop unrelated to the system.
Note that these formulars cannot be calculated explicitly: To determine the expectation values, x and u for a time period are needed, but to measure them (in closed loop), φ or ϕ^T is needed. The whole learning process is *highly iterative*. This means an adaptation procedure has to be run. The implementation of the adaptation law is shown in Sec. 4.1.1.

How can these definitions be motivated?

To give some intuition, have a closer look at $\mathrm{E}\{ux^T\}$: The more an input u_j correlates with an output x_i the larger will be their corresponding feedback factor in φ. Note that an elastic system is assumed now. Hence the higher (negative) feedback leads to a lower state. Therefore it is evident that a balance of the adaptation is reached.

Furthermore note that the adaptation is a generalization of the *Hebbian Learning Method* [5] which is used for adapting neural networks. This method is also discussed in a neocybernetic background (see [2]).

How about stability of the proposed adaptation law?

As shown before there is positive feedback in the adaptation law working against the (assumed) negative feedback of the controlled system. Hence it is nontrivial to assume that the adaptation converges. Now simulations (see Ch. 4) show that the adaptation at least for the systems used in this thesis behave well under certain conditions. Thus it is probable that there exist a generell proof for the convergence of this adaptation process, but it has not been discovered yet.

2.3.2. Principal subspace

If convergence is assumed some properties of the proposed adaptation can be derived to give motivation to this method. Using (2.10) and (2.8) ($w = 0$) the adaptation law can be formulated as

$$\varphi^{(n+1)} = B \cdot \mathrm{E}\{ux^T\} = B\varphi^{(n)} \cdot \mathrm{E}\{xx^T\} \tag{2.14}$$

After adaptation there must hold $\varphi^{(n+1)} = \varphi^{(n)} =: \varphi$ therefore (2.14) becomes

$$\begin{aligned} \varphi = &\quad B\varphi \cdot \mathrm{E}\{xx^T\} \\ \Leftrightarrow B^{-1}\varphi = &\quad \varphi\mathrm{E}\{xx^T\} \end{aligned} \tag{2.15}$$

where symmetry of $\mathrm{E}\{xx^T\} = \mathrm{E}\{xx^T\}^T$ was used. Now one can see that in a fully adapted system the non-zero elements b_i of the diagonal matrix B are the *eigenvalues of the output covariance matrix* $\mathrm{E}\{xx^T\}$. Furthermore the rows of the resulting feedback matrix φ are some *eigenvectors* of $\mathrm{E}\{xx^T\}$. In [1] Sec. 3.2. and in [5] it is proven that these eigenvectors are even *the most significant eigenvectors* in that sense that the corresponding eigenvalues b_i are the m largest ones of $\mathrm{E}\{xx^T\}$.
This means, that the introduced adaptation algorithm implements PCA in that sense that the feedback φ selects the m most significant

principal components from the output. This result will also be proven by simulations later on.
Of course this is trivial for $m \geq n$. For this case, (2.13) can be used as adaptation law

$$(\phi^T)^{(n+1)} = Q \cdot \mathrm{E}\{xu^T\} = Q(\phi^T)^{(n)} \cdot \mathrm{E}\{uu^T\} \tag{2.16}$$

A fully adapted system leads to $(\phi^T)^{(n+1)} = (\phi^T)^{(n)} =: \phi^T$.

$$\begin{aligned} \phi^T = &\quad Q\phi^T \cdot \mathrm{E}\{uu^T\} \\ \Leftrightarrow Q^{-1}\phi^T = &\quad \phi^T \mathrm{E}\{uu^T\} \end{aligned} \tag{2.17}$$

Thus it can be seen that an identical result than in (2.15) can be obtained for the mapping ϕ^T which is constituted by the eigenvectors of $\mathrm{E}\{uu^T\}$. And Q^{-1} includes the eigenvalues of $\mathrm{E}\{uu^T\}$. As the system considered in this thesis always has less inputs than outputs ($n \geq m$) this result was only presented for the sake of completeness.

Chapter 3

Introduction to the system

The system, which is considered in this thesis, describes a force actuator in a cage induction electrical machine. We are faced with the problem that unbalanced mass excitation forces cause radial rotor vibrations. These vibrations lead to such problems as noise, increased bearing wear or rotordynamic instability. There is a running project at the Conrol Engineering Laboratoy of Helsinki University of Technology whose main goal is to damp these low frequency vibrations with higher harmonic components. For a project description, see [9] or [3].

3.1. Configuration overview of the real system

The test machine, which is being contructed to the laboratory constists of an industrial motor whose resonance frequency is intentionally reduced to about 50Hz. A schematic view of the motor is given in Fig. 3.1. The rotor shaft was extended and displacement sensors and extra bearings were installed. The original bearings of the machine are used only as safety bearings.
In the rotor three additional actuator windings are mounted. They can be used to apply radial forces on the rotor constituting the input signal which is produced by the vibration controller.

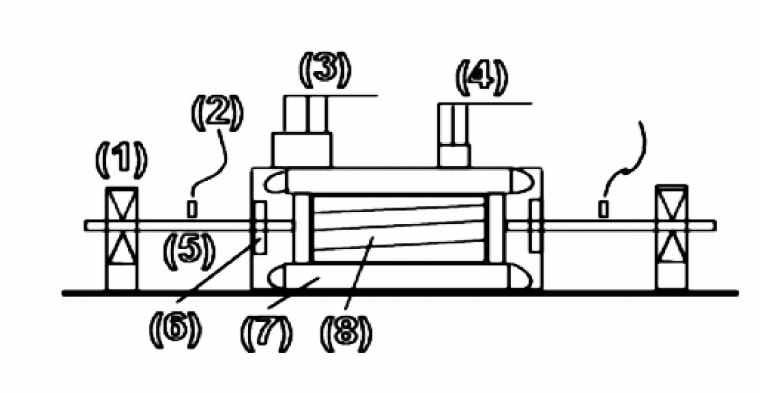

Figure 3.1.: Schematic view of the test machine. (1) bearing, (2) displacement sensor, (3) main winding supply, (4) control winding supply, (5) rotor shaft, (6) safety bearing, (7) stator core, (8) rotor core

3.2. Models

Several mathematical models were developed for the process. A simplified one is shown in (3.2) and (3.2) constisting of two parts: An electrical part (state i) and a mechanical part (states ξ and η). The electrical part describes a three dimensional first order connection between the actuator input voltages (one for each winding) and the resulting forces on the rotor. The mechanical part is a high dimensional second order oscillation equality, described by modal coordinates η on given actuator forces. Each coordinate describes the excitation of one real mode shape. Not all modes can be seen by the two measurment points, whereby not all elements of η are visible at the output (see description of Φ_μ below). Please refer to [3] for a detailed derivation of the model and a listing of the used assumptions.

$$\frac{d}{dt}\begin{pmatrix}\xi\\ \eta\\ i\end{pmatrix} = \begin{bmatrix}-2\Omega\Xi & -\Omega^2 & \Phi_r^T c C_e m\\ I & 0 & 0\\ 0 & 0 & A_e m\end{bmatrix}\begin{pmatrix}\xi\\ \eta\\ i\end{pmatrix} \tag{3.1}$$
$$+ \quad \begin{bmatrix}0\\ 0\\ B_e m\end{bmatrix} U + \begin{bmatrix}\Phi_r^T c\\ 0\\ 0\end{bmatrix} f_e x$$
$$u = \begin{bmatrix}0 & \Phi_\mu & 0\end{bmatrix}\begin{pmatrix}\xi\\ \eta\\ i\end{pmatrix} \tag{3.2}$$

where $A_e m$ and $B_e m$ describe the dynamics of the actuator (electrical part). Φ_μ and Φ_{rc} denote the submatrices of Φ containing the transversal displacement degrees of fredom at the rotor displacement sensor locations (see Fig. 3.1) and rotor center, respectively. The other matrices are given by

$$\Xi = \quad \operatorname{diag}\{\zeta_1 \quad ... \quad \zeta_n\} \tag{3.3}$$
$$\Omega = \quad 2\pi \cdot \operatorname{diag}\{f_1 \quad ... \quad f_n\} \tag{3.4}$$
$$\Phi = \quad \{\phi^{(1)} \quad ... \quad \phi^{(n)}\} \tag{3.5}$$

where f_k denotes the k'th eigenfrequency of the machine, $\phi^{(k)}$ the corrsponding mass-normalized mode shape and ζ_k the equivalent viscous damping coefficient.

3.3. Comments about the models in the neocybernetic context

The model presented in this chapter is one of the most complex ones developed for the system. It was implemented in Comsol Multiphysics® (see [8]) and constitutes the standard model, which all other models are compared to. The problem with the Comsol Multiphysics® model is the simulation speed. It takes hours of time to simulate a few seconds of working machine. Neocybernetics was already presented

as a highly iterative method, which needs hundreds of simulations. Therefore very fast models are needed. There are several linear models provided for control engineering issues. They have been identified from the origianl model. Following linear models have been used to abtain the results of this thesis:

- *SISO 6 states model.* This model considers vibration in x direction and provides only one input for the same direction. 2 states are used to model the disturbance force (sine wave close to the rotor resonance frequency), actuator and rotor are modelled as one PT1 system each. Input is actuator voltage and output is the displacement.
- *MIMO 10 states model.* For this model just the states of actuator and rotor have been doubled. Two actuator voltages control two displacements in x and y directions. There are still 2 states used to model the disturbance.
- *MIMO 18 states model.* Final model also used for developing the traditional controllers (LQR, pole placement, convergent control). 4 states represent the disturbance forces in x and y direction, 10 states are used for the actuator and 4 for the rotor (still two PT1 systems).

All models can be (and were) simulated in Matlab Simulink®. Satisfactory accurancy is only provided by the third (18 states) model, but since accurancy is not of fundamental importance for showing the principles of neocybernetics, most calculations have been done with the first two models due to speed and stiffness problems with the third one.

Chapter 4

Neocybernetic first level control

4.1. Simulation Description

For all simulations and calculations the Matlab/Simulink environment was used. The models are given as linear state space models in Matlab.

4.1.1. Typical Simulation Procedure

In order to run an adaptation an initial guess for the controller has to be made. This initial value is very arbitrative. Of course it must not destabilize the system but it also decides if there is a solution for the adaptation as the controller cannot change signs arbitrarily (see next sections). The probability of success increases the closer the inital controller is to the final one. E.g. for a series of adaptations the resulting controller of an adaptation was used as inital controller for the next adaptation respectively.
Another important parameter for successful adaptation is the filter factor F shown in Fig. 4.1. This is a low pass filter which slows down the change of φ to avoid an overshoot of the feedback, which could lead to instable system behaviour (see 4.3)
Now, if the desired output eigenvalues are given in B one can start

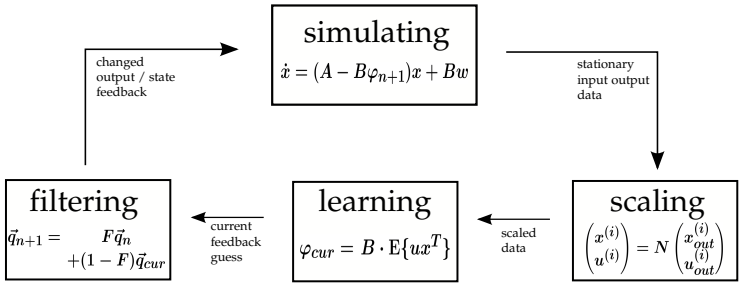

Figure 4.1.: Structure of first level adaptation process.

the simulation which continues until a stationary state is reached. As shown in Fig. 4.1 the data is scaled by some diagonal matrix N to obtain comparable values of all outputs. Now the learning step is taken and the resulting guess for the output feedback is filtered according to filter factor F ($F \in [0;1]$). The new feedback is applied to the simulator and the next simulation run can be started. The adaptation ends when the elements of φ do not change more than a small limit value r_{lim}. Hence, after adaptation it holds:

$$\|\varphi - B \cdot \mathrm{E}ux^T\| < r_{term} \tag{4.1}$$

Note that this terminating condition doesn't nessecarily mean that φ and B implement PCA. It just means that the adaption has become very slow. This can happen for example if the system is out of control (subcybernetic case) because of too small values of B or if there is no solution for the adaptation. In the following sections those possibilities are considered.

4.1.2. Implementation

The whole adaptation process was implemented with three nested scripts as shown in Fig. 4.2. The inner script runs the Simulink model with a given Controller K and calculates the input output covariances $E\left\{ux^T\right\}$ and $E\left\{xx^T\right\}$. See A for an example of the simulink file.
In a second step, the middle script implements the adaptation loop.

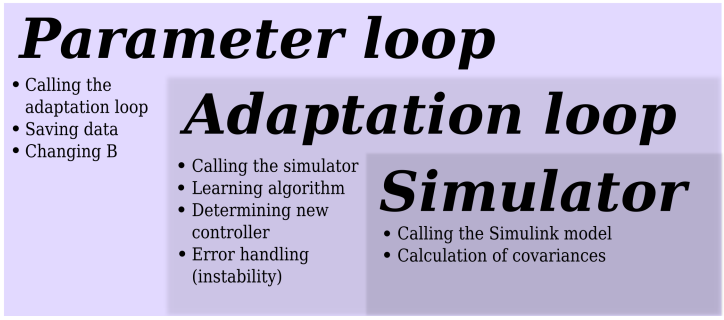

Figure 4.2.: Illustration of implemented Matlab scripts

This means that after calling the simulation script a learning step is performed and the new estimation for the feedback φ is determined. The outer script is used to perform series of adaptions. E.g. a series of different signal eigenvalue matrices B (see 2.3.2) whose results can be seen later on e.g. in Fig 4.4. The script just start the middle script and stores the adaptation results.

4.2. Simulation results

During the project several different feedback structures were tested.

4.2.1. SISO P Controller

In the first phase only the SISO system as described in Sec. 3.3 is used. The simulator is configured for a constant feedback $u = K_P \cdot x$ and a initial guess of a stable K_P is set. Now the proposed learning law is applied. That means:

$$K_P = \varphi = B \cdot E\left\{ux^T\right\} \tag{4.2}$$

Fig. 4.3 shows a typical adaptation process: The feedback mapping $\varphi = B \cdot E\left\{ux^T\right\}$, which is used as controller in (4.2), converges depending on B. Consequently the control error converges, too. Thus, a stable behavior of the adaptation algorithm can be achieved. The

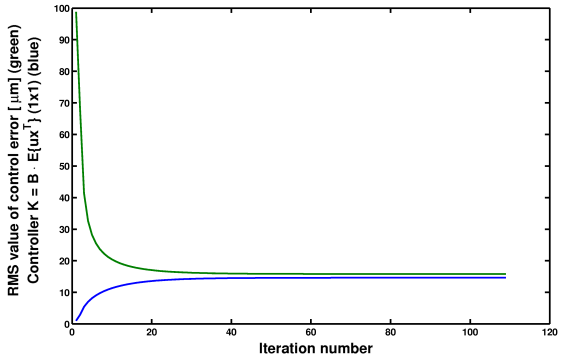

Figure 4.3.: A typical adaption process. In this case $B = 4000$ and a weak initial controller were selected.

resulting controller reduces vibration to $\sqrt{\frac{1}{B}} = 15.8\mu m$ (RMS). That is 15.65% of the open loop value ($101\mu m$ RMS). But is there a better P controller? Can the best P controller be found?

Therefore you should have a look at the matrix B. As shown in Sec. 2.3.2 B^{-1} contains the eigenvalues of $E\left\{xx^T\right\}$. In order to achieve better control results the overall variation of the output x should be reduced. Therefore higher values for B should be applied. Thus a series of adaptations was started with increasing values for B.

The result is shown in Fig. 4.4 and Fig. 4.5. Clearly one can observe a decreasing control error (which is the same as $E\left\{xx^T\right\}$ for the SISO case) when B grows. To understand why, just have a look at Fig. 4.5. Using the control law introduced in (4.2) it is easy to verify that K_P constantly grows (note the scaling). Thus a better control result in the SISO case is applied just by increasing the feedback factor.

Even more can be seen from Fig. 4.5. $E\left\{ux^T\right\}$ obviously seems to "jump" at a specific value of B. Before that the input output covariance is (nearly) zero and the output covariance is constantly high. This point is the lower limit of B at which the system gets out of control as the required signal eigenvalue B^{-1} is higher than that of the open loop system ($\lambda^{-1} = 97.7196$). Because of (2.14) the sign of

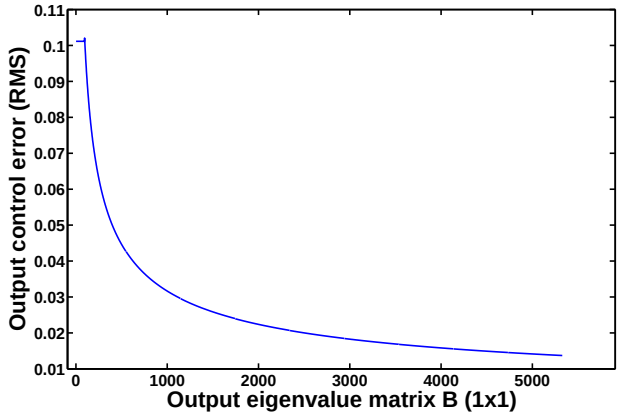

Figure 4.4.: SISO adaptation series showing the control error against B.

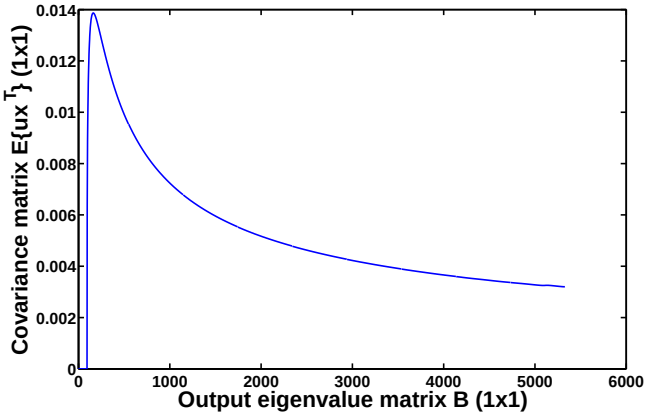

Figure 4.5.: SISO Adaptation series showing $\mathrm{E}\{ux^T\}$ against B.

φ cannot change during SISO adaptation which would lead to higher

output than in open loop. Thus the adaption algorithm can only try to raise the output variance by minimizing the feedback. Only because of numerical reasons there are values of $E\left\{ux^T\right\}$ which are visibly greater than zero in that interval.
The right end of Fig. 4.5 was not selected by chance. Actually it is the upper limit of B, which could not be exceeded without obtaining a unstable system behaviour during adaptation. This issue is discussed further in Sec. 4.3.

4.2.2. MIMO P Controller

After successfully controlling the SISO system, the second step was taken: The SISO model was exchanged by the more realistic MIMO model. For this experiment the 10 states model as described in Sec. 3.3 was used. As it is a two in two out model a simple proportional output feedback is given by:

$$u = Kx = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} x \tag{4.3}$$

The same adaptation law than given in (4.2) $K = \varphi = BE\left\{ux^T\right\}$ was used whereby $B = b \cdot I$ was always choosen such that its diagonal elements are (approximately) the same. They never should be equal because if there are equal b_i, (2.15) can hold without determining *all* eigenvalues of $E\left\{xx^T\right\}$. In this case all equal b_i could determine one eigenvalue together. The corresponding eigenvectors given by φ would become linearly dependent and the rest of the eigenvalues of $E\left\{xx^T\right\}$ would be totally free. Thus in general it is not possible to drive two or more output eigenvalues to exactly the same value. They always have to be slightly different. In this case a difference of 2 was seleceted but for simlification it is treated as one value b later on.
Fig. 4.6 shows the results of the MIMO controller. Obviously there is also a lower limit of b. But actually there are two areas of dramatical changes. This can be explained by different output eigenvalues of the open loop system. As described in the foregoing section these open loop eigenvectors determine the borders at which the adaptation starts to take control of the output signal's eivenvalues. In this case $\lambda_1 = 5.96 \cdot 10^{-5}$ and $\lambda_2 = 5.05 \cdot 10^{-3}$ are the numerically calculated

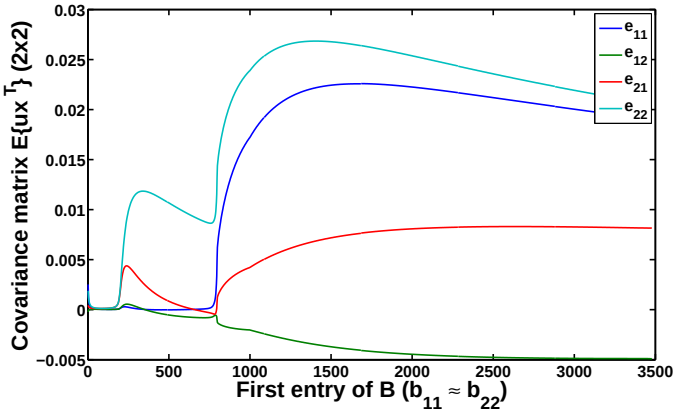

Figure 4.6.: MIMO Adaptation series showing E$\{ux^T\}$ against B.

eigenvalues of $E\left\{xx^T\right\}$ in the open loop system. The corresponding b_i are $\lambda_1^{-1} = 1.68 \cdot 10^4$ and $\lambda_2^{-1} = 1.98 \cdot 10^2$. Hence the transistions in Fig. 4.6 happen when the first eigenvalue can be controlled (left transition at λ_2^{-1}) and when both eigenvalues can be controlled. (The second transistion is not at λ_1^{-1} because it is no open loop system at this point any more). Furthermore in MIMO case the non diagonal elements of φ can even change sign, which can lead to higher outputs than in open loop and to smoother transisition than in the first order case (see also B).
On the right side of Fig. 4.6 the same stability probelm occours than in the SISO case. The calculation stopped at this upper unstable location because a further increase of b failed: The system always became unstable during adaptation. See section 4.3 below for a discussion of that problem.

4.2.3. State feedback

In the same way as in the foregoing section a MIMO state feedback was tested. Regarding the disturbance as input, 8 states of the 10 states model were feedbacked trying to control 2 eigenvalues of

$\mathrm{E}\{xx^T\}$. The rows of φ sucessfully adapted close to the most significant eigenvectors. *So PCA could be shown.* But the adaptation never really converged. It just got slowly instead and finally lead to an unstable system. The problem is the same as described in Sec. 4.3.4.

4.3. Stability and adaptation problems

As seen in the foregoing section there exist upper bounds of the matrix B which the neocybernetic theory doesn't explain. And there are more problems during adaptation which need to be investigated in order to offer a reliable framework. Actually there occurred three types of limitations during the simulation of first level control.

4.3.1. Controller overshoot

The first type of limitation which is also the easiest to explain is the already mentioned upper bound of B. B contains the (inverted) requested eigenvalues of $\mathrm{E}\{xx^T\}$. If B increases the controller has to reduce the output signal. In SISO case this can only happen by increasing the gain of the controller. But naturally the system is not stable for all feeback gains.

4.7 shows root locus diagrams for an applied negative (left) and

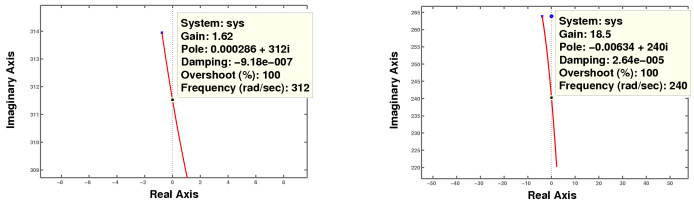

Figure 4.7.: Sections of root locus diagrams of the system considered in Sec. 4.2.1. On the left side negative feedback, on the right side positive feedback were used

positive (right) feedback. For both cases the limit gain is shown at

which the first system eigenvalue crosses the imaginary axis. Hence the adaptation just touches the stability border of the P controller. The given positive maximum gain is the one reached in 4.5 ($\varphi = B \cdot \mathrm{E}\{ux^T\} = 5331 \cdot 3.202 \cdot 10^{-3} = 17.07$).
Basically the same thing happens in MIMO case or for state feedback controllers. When the expectations given in B become too high the adaptation leaves smoothly the stable area in parameter space. The adaptation law proposes controllers which demonstrably destabilize the system.
Remark: Compared to the SISO case there is one difference. The adaptively found P controller is usually unique, and thus it is the best P controller obtaining the requested control result. In the other cases there are plenty of possible controllers which lead to the desired signal eivenvalues. Hence if the neocybernetic adaptation creates a controller overshoot, it means only that there exists no stable structured controller in the sense of neocybernetic PCA structure. As shown in Sec. 4.3.4 stable controllers may exist even when the adaptation algorithm does not find a solution.

4.3.2. Instability caused by adaptation

This problem occured mainly when operating close to the stability border of the system but also when the initial values and filter factor were not selected properly. What happens is that adaptation fails although there exists a neocybernetic solution for the given control problem. These failures are caused by the undefined way in control parameter space the adaptation takes while searching for the solution. As the adaptation does not take stability issues into account the unstable areas can be touched during adaptation.
This problem could always be solved by reducing the adaptation speed (increasing of the filter factor F, see Fig. 4.1) or by selecting inital controllers closer to the searched one. For example, in Fig. 4.4, Fig. 4.5 and Fig. 4.6 all adaptations were started by taking the previous result as the inital value.

4.3.3. Highly differing output eigenvalues

Up to now all signal eigenvalues which were supposed to be controlled (if there was more than one) were in the same range. The presented MIMO system has got very symmetrical scalings for the two outputs (x and y displacements). Furthermore they are quite independent and thus $\mathrm{E}\{xx^T\}$ is close to diagonal form and has two eigenvalues close to each other. Consequently B always contained values in the same range. Then problems with highly differing state eigenvalues occured during second level control simulations (see Ch. 5). Because equal output eigenvalues are a special case, this problem is presented here.
For this experiment one output signal was scaled differently. Then B and the inital controller were scaled appropriately and the adaptation was started. Nearly all experiments led to unstable systems. Even when taking the rescaled results of an adaptation (which nessecarily already start close to the expected result) the problem remained. Only initial controllers fullfilling the adaptation law with very high precision (5 decimal powers and more) could successfully adapt. The problem already occurred with eigenvalues differing by only 1 decimal power.
Without changes on the system or the control structure itself the adaptation failed. What actually happens is that both vectors of φ adapt to the same most significant eigenvector and then the higher b_i (which should correspond with the lower eigenvalue) controls the higher eigenvalue. This lead to an controller overshoot and the system collapses.
This problem is discussed in more detail in Ch. 6.

4.3.4. Accuracy problem

A general problem of the MIMO simulations was the accuracy of the solution. All results presented in Sec. 4.2.2 were obtained by using a relatively high limit values r_{lim} (see (4.1)). Adaptation with higher accuracy could not be finished sucessfully.
Fig. 4.8 shows the typical behaviour of φ for an appropriate but arbitrary initial feedback and a small r_{lim}. After a quick adaptation close to the principial output components the feedback changes slowly but

continously until the system gets unstable. Note that the control result (output eigenvalues) is achieved pretty fast as the corresponding control error diagram Fig. 4.9 shows.
The changes in Fig. 4.8 are normally so slow and it takes so many iterations until the system gets unstable that the problem at first was not found. But the effect could be increased by applying a slightly different feedback structure:
At first the adaptation was run as described in Sec. 4.2.2. Then a matrix operation was applied at the input of the system such that there holds

$$u = M\hat{u} \tag{4.4}$$

M is an invertible $m \times m$ matrix which mixes the inputs $\hat{u}_i$ in similar parts. For example:

$$M = \begin{pmatrix} 0.6 & 0.4 \\ 0.45 & 0.55 \end{pmatrix} \tag{4.5}$$

To find a feeback matrix $\hat{\varphi}$ which produces same results than the (fully adapted) feedback φ there must hold

$$u = \quad \varphi x = M\hat{\varphi}x \tag{4.6}$$

$$\Rightarrow \hat{\varphi} = \quad M^{-1}\varphi \tag{4.7}$$

The new feedback was applied and sucessfully tested. But when the adaptation was started with this new structure and $\hat{\varphi}$ as initial feedback it failed even with the normally used r_{lim}. The process ran for this initial feedback and all other tested initial feedbacks into the unstable area of feedback parameter space. Fig. 4.8 and Fig 4.9 show simulation results of this mixed input structure.
Finally with the results of Ch. 6 this behaviour could be shown for all MIMO simulations of this chapter.

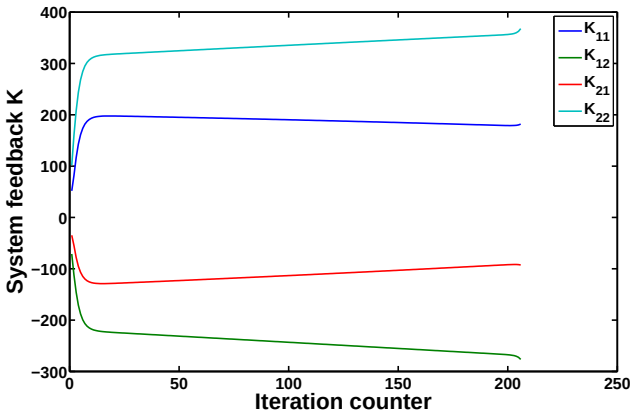

Figure 4.8.: Feedback $\hat{\varphi}$ during adaptation using the mixed input structure.

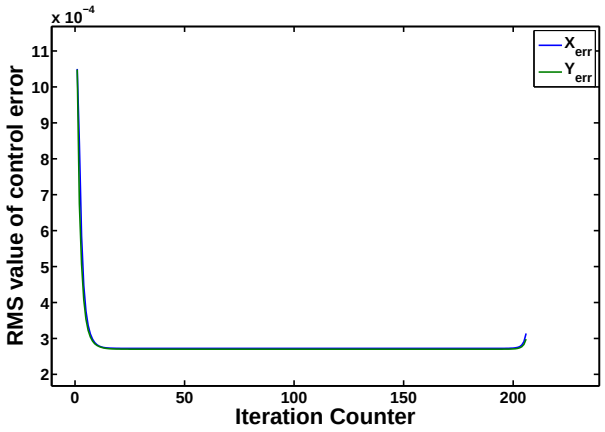

Figure 4.9.: Control error diagram corresponding to 4.8. It also represents the inverted output signal eigenvalues because $\mathrm{E}\{xx^T\}$ is close to diagonal form

Chapter 5

Second level control

Up to now neocybernetics was used to directly create a controller out of simulation data. No system knowledge was used, and thus a complete model-free controller could be obtained by this method if real measurement data was used instead of simulations. But the controller structures are still very simple, and consequently the obtained controllers have always been of very limited quality. Now one possible solution to this problem is tested. The idea is to combine neocybernetic modelling with a traditional controller. The goal of model-free working is abandoned by using a complete state feedback obtained from a LQ controller design. Neocybernetics is used to calculate the Q and R matrices of the LQ controller in order to gain a more structured method for tuning it and to attack the general robustness problem of LQ control (see for example [7]).

5.1. Adaptation process

The adaptation procedure presented in Sec. 4.1.1 is now being changed. Still the system output is used as data source. But now the Q and R matrices are seen as the "input" of the simulated system. In Fig. 5.1 one can see that the filter output now is a vector $\vec{q}$ which represents the diagonal elements of the Q matrix.
While the normalization step remains the same the learning step and the filter look different now. That is explained in the next section.

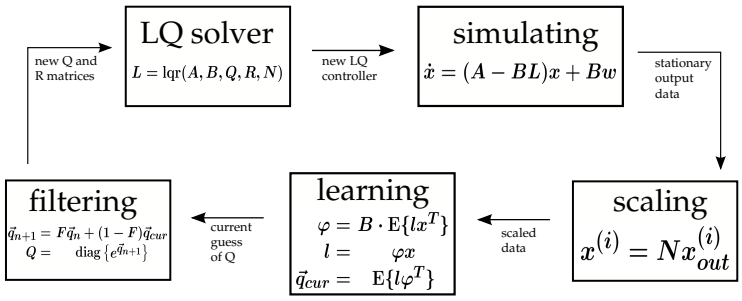

Figure 5.1.: Structure of second level adaptation process.

After new values for Q have been found a traditional LQ control design is performed using the matlab lqr solver. The resulting controller C is applied to the system and the adaptation can go on until the terminating condition

$$\|\vec{q} - \vec{q}_{cur}\| < r_{term} \tag{5.1}$$

holds.

5.2. Latent subspace

Reduction to significant information

For this adaptation the whole state of the system is used. Depending on the system structure several different states might be highly correlated. This leads to very dominant principal components of $\mathrm{E}\{xx^T\}$ which mean large differences between the eigenvalues. Hence it is reasonable to use only few principal components. Therefore a so called latent vector l (see pg. 52 in [1]) representing the level of the used principal components is introduced here. Its dimension s can be chosen between 1 and n. The adaptation law becomes

$$\varphi = B \cdot \mathrm{E}\{lx^T\} \tag{5.2}$$

where φ is a $s \times n$ matrix now, mapping from x-space to latent l-space.

Taking control of parameter space

But there are still n elements in $\vec{q}$ which are supposed to be determinend by l. Another mapping "back" to n dimensions is needed. Therefor φ^T is proposed to form this mapping from s-dimensional latent space to the n-dimensional LQ parameter space always following the intution: The higher a state x_i the higher its corresponding q_i (negative feedback of the system assumed).
One is faced another problem when searching for an appropriate mapping: In the first level case the output u of the controller was time depending. Thus the mapping φ (in second level case it would be $\varphi^T\varphi$) could be directly applied to the system. Now the learning step must result in some time independent, non-negative $\vec{q}$. First idea was to use RMS values $\bar{x}$ instead of x. But then $\mathrm{E}\{xx^T\} = \bar{x}\bar{x}^T$ would always have only one non-zero eigenvalue, because

$$\bar{x}\bar{x}^T = \begin{bmatrix} \bar{x}_1\bar{x}^T \\ \vdots \\ \bar{x}_n\bar{x}^T \end{bmatrix}$$

constists of n linearly dependent row vectors. Hence the information loss would be to big.
To fix this problem the whole learning step is calculated with time depending variables. Just before the results are used for Q the expectation values are taken. As this still can lead to negative elements of $\vec{q}$ the exponential function is used to map negative values to small (positive) ones.
Now the whole learning step using (5.2) can be summarized to

$$l = \quad \varphi x \tag{5.3}$$
$$\vec{q}_{cur} = \quad \mathrm{E}\{l\varphi^T\} \tag{5.4}$$

The current guess $\vec{q}_{cur}$ is low pass filtered like in first level control and the filtered parameter vector mapped to a diagonal matric Q:

$$\vec{q} = \quad F\vec{q} + (1-F)\vec{q}_{cur} \tag{5.5}$$
$$Q = \quad \mathrm{diag}\left\{e^{\vec{q}}\right\} \tag{5.6}$$

Alternative learning step

The above presented learning step is of course not compulsory. For example we can choose ϕ directly for $\vec{q}_{cur}$:

$$\vec{q}_{cur} = \sum_{i=0}^{s} \vec{\varphi}_i \tag{5.7}$$

where $\vec{\varphi}_i$ are the row vectors of φ. The idea is the same than before: The higher x correlates with some latent variable l, the higher gets their connection φ, and in consequence the corresponding element of Q increases which leads to an negative feedback.
This method gets rid of the problem of time independence of Q. Furthermore if $\varphi = B\varphi \mathrm{E}\{xx^T\}$ is used instead of (5.2) l never has to be calculated explicitely and no postdata of the previous step is used for the next step. This increases the stability of the algoritm.
Both presented methods were tested successfully.

5.3. Simulation results

For simulation speed and stability reasons the LQR simulations were done on the 6 states SISO system. As 2 states only model the disturbance force only 4 states and the corresponding subsystem were used for the LQ design. The implementation is identical to 4.2 whereas of course the second level control formulas has been implemented in the scripts.

5.3.1. One dimensional latent subspace

Beginning with an one dimensional subspace ($s = 1$) the second level adaptations were tested. A typical adaptation process is shown in Fig. 5.2. Q stabilizes at a reasonable value controlling the most significant output eigenvalue to the given B. The range of B is also (very) limited: There is a lower limit determined by the output eigenvalue on a completely zeroed Q. The upper bound of B is reached if the values of Q and R differ too much and the optimization problem cannot be solved any more. In the next section it is explained why this upper bound is reached quite quickly for $s = 1$. Of course there

is also the technical limitation like for example an input saturation which does not allow arbitraily high inputs. In this context it is also very import to see that the neocybernetic controller only searches for the PCA structure which does generally not optimize for the technical limitations.

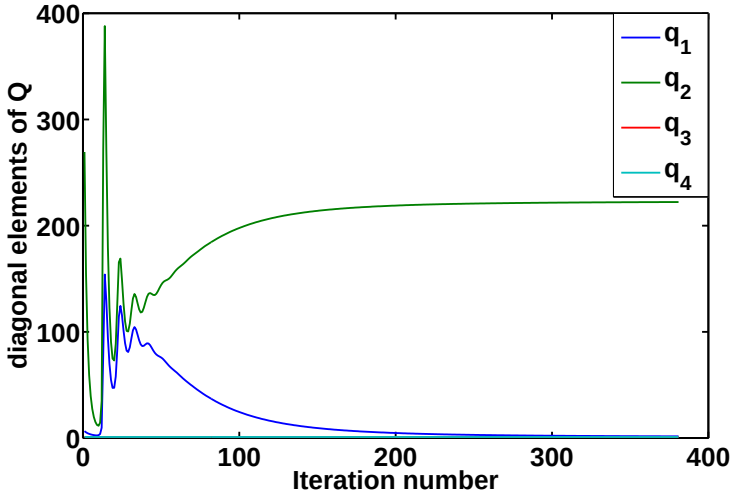

Figure 5.2.: Adaptation of Q matrix using a one dimensional latent subspace. Only the displacement (x_2) is emphasized, all other q_i are close to zero

5.3.2. Problems of second level control

When having a second look how $Q(\varphi)$ is calculated in the foregoing section it is easy to see that diag$\{Q\}$ is basically a linear combination of $\vec{\varphi}_i$ which are the eigenvectors of E$\{xx^T\}$. In the case of a one dimensonal latent subspace it is only one scaled eigenvector. All simulations revealed that the eigenvectors keep pretty much the same if Q is only scaled. Thus the ratios between the entries of Q are somehow fixed. If the output is too high just the whole Q is raised.

But a higher Q just emphasizes all states more than the inputs. This has got a very little effect on the output but demands high efforts on the inputs. As a result Q gets easily too high and adaptation fails even without good controller results.
To move the entries of Q in a clearly different way one needs a higher dimensional latent subspace. But as it turns out in the next chapter (Thm. 6.2) there is no way for the 6 states model to increase s. All simulations confirmed this: When trying a two dimensional latent subspace all $\vec{\varphi}_i$ always adapted only to the most significant eigenvector. When they were forced to adapt to the first two eigenvectors still one $\vec{\varphi}_i$ converged to zero and lost any kind of control.

Chapter 6

Further theoretical considerations

In the last chapters it turned out that the presented controller structure (Sec. 2.3) can be successfully, used but still there remain plenty of unexplained problems. Therefore some further theoretical steps are presented in this chapter to give intuition or even provable explanation to most of the problems.

6.1. Convergence of adaptation law

It was noted in Sec. 2.3 that the adaptation law used has no proven properties concerning stability and the existence of solutions. Now some intuition is given how this adaptation law finds a balance for the output eigenvalues and how it implements PCA.
At first it is only assumed that there exists a feeback φ such that (2.8) holds:

$$u = \varphi x \tag{6.1}$$

So the adaptation law (2.10) can be reformulated like in (2.14). For one row vector φ_i of the matrix φ this looks like

$$\varphi_i^{(n+1)} = b_i \cdot \mathrm{E}\{ux^T\} = b_i \varphi_i^{(n)} \cdot \mathrm{E}\{xx^T\} \tag{6.2}$$

As $\mathrm{E}\{xx^T\}$ is symmetric it can be diagonalized and the diagonalization

$$\mathrm{E}\{xx^T\} = VDV^T = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} \operatorname{diag}\begin{Bmatrix} \delta_1 & \dots & \delta_n \end{Bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix} \tag{6.3}$$

is defined. Using (6.3) and reformulating (6.2) we obtain

$$\varphi_i^{(n+1)} = b_i(\varphi_i^{(n)}V)(DV^T) \tag{6.4}$$

Another reformulation according to the brackets used in (6.4) shows that the adaptation law (2.10) can be seen as linear combination of the eigenvectors of $\mathrm{E}\{xx^T\}$ for each feedback vector φ_i.

$$\varphi_i^{(n+1)} = \sum_{k=1}^{n} < \varphi_i^{(n)}, v_k > b_i\delta_k v_k^T \tag{6.5}$$

This formulation now gives intuition how the adaptation law generally works:

1. At each iteration step each φ_i is newly constructed as a linear combination of the orthogonal v_k. For all φ_i the same rule is used and there is only indirect influence of the other φ_i.

2. The v_k are emphasized by three factors. At first this v_k which is closest to φ_i gets hightest priority through $< \varphi_i^{(n)}, v_k >$. Thus φ_i adaptes very quickly to one of the v_k whereby the others are multiplied with 0 (because of orthogonality of v_k). If v_k has equal direction to φ_i the sclar product results exactly in the length of φ_i because v_k is normalized.

3. Furthermore each v_k is multiplied by $b_i\delta_k$. After adapting to one v_k this is only interesting for this found eigenvector. $b_i\delta_k$ will stretch φ_i if δ_k is greater than $\frac{1}{b_i}$. The longer φ_i leads to an higher input u and the negative feedback of the system hopefully abates the output and δ_k will go down. The inverse dynamic will happen if δ_k is less than $\frac{1}{b_i}$. Hence the adaptation stabilize at $\delta_k = \frac{1}{b_i}$.

This result allows a refinement of the assumption for the adaptation formulated in Ch. 2. For a successfull adaptation process the following assumptions must hold:

- *Negative feedback*: The considered system must implement a negative feedback behaviour for a feedback $\varphi_i \propto v_k$ and its corresponding output eigenvalue δ_k
- *Sufficient good initial feedback*: The initial guess $\varphi^{(0)}$ of the feedback matrix must contain rows $\varphi_i^{(0)}$ which are close enough to one v_k respectively. Identical $\varphi_i^{(0)}$ for example adapt together to the most significant v_k.
- *Different b_i or δ_k*: b_i and δ_k must not differ too much. If they do lower significant φ_i will leave their eigendirection and adapt to a higer significant eigenvector because $b_i\delta_k$ becomes to high for the higher significant terms. The upper bound depends on
  - Accuracy of $\varphi^{(0)}$ (Second assumption)
  - Length of φ_i
  - Iteration speed (Filter Faktor F, See 4.1.1)
  - Measurement noise of $\mathrm{E}\{xx^T\}$

6.2. Existence of solutions

In this section one first step on the way to a *general proof of convergence and existence of the neocybernetic adaptation* is done for the special case of a linear homogenous system. All models used for this thesis are of this kind and so this special case will be enough to explain problems which appeared during the simulations presented in the last chapters.

Lemma 6.1 *Consider a linear time invariant homogen system*

$$\dot{x} = Ax \tag{6.6}$$

whose eigenvalues $\lambda_i \neq 0$ statisfy $\Re(\lambda_i) \leq 0$. Let the first k eigenvalues be purely imaginary ($\Re(\lambda_i) = 0$) and let v_i be the corresponding

eigenvectors to λ_i. All v_i are assumed to be linearly independent. The general solution of the system can be written like

$$x(t) = x_{osc} + x_{asym} = \sum_{i=1}^{k} c_i e^{\lambda_i t} v_i + x_{asym}(t) \tag{6.7}$$

where c_i are determined by the initial value problem $x(0) = x_0$. Then the system state covariance matrix $E\{xx^T\}$ has maximum k nonzero eigenvalues and they are pairwise dependent. Furthermore $E\{xx^T\}$ is given by

$$E\{xx^T\} = \sum_{i=1,3,..}^{k-1} c_i c_{(i+1)} v_i v_{(i+1)}^T \tag{6.8}$$

Proof: The system is only considered in steady state. Therefore all transients are vanished ($x_{asym} = 0$). Let us assume that the steady state is reached at time t_1. Then the stationary solution looks like

$$\bar{x} = x_{osc} = \sum_{i=1}^{k} c_i e^{\lambda_i t} v_i \tag{6.9}$$

Reformulated in real numbers and token into account that $\Re(\lambda_i) = 0$ we obtain

$$\bar{x} = \sum_{i=1,3,..}^{k-1} 2d_i(\cos\beta_i t a_i - \sin\beta_i t b_i) - 2e_i(\sin\beta_i t a_i + \cos\beta_i t b_i) \tag{6.10}$$

where $d_i \pm ie_i = c_{i/i+1}$, $\alpha_i \pm i\beta_i = \lambda_{i/i+1}$ and $a_i \pm ib_i = v_{i/i+1}$.
$\mathrm{E}\{xx^T\}$ can now be calculated as mean value for a time intervall T:

$$\mathrm{E}\{xx^T\} = \frac{1}{T} \int_{t_1}^{t_1+T} x(t)x(t)^T dt \tag{6.11}$$

If (6.10) is substituted in (6.11) all resulting integrals are of the following types:

$$\int \sin^2 \beta_i dt = -\frac{1}{2\beta_i} \sin \beta_i \cos \beta_i + \frac{1}{2} t \tag{6.12}$$

$$\int \cos^2 \beta_i dt = \frac{1}{2\beta_i} \sin \beta_i \cos \beta_i + \frac{1}{2} t \tag{6.13}$$

$$\int \sin \beta_i \cos \beta_i dt = -\frac{1}{2\beta_i} \cos^2 \beta_i t \tag{6.14}$$

$$\int \sin \beta_i \cos \beta_j dt = -\frac{\beta_i}{\beta_i^2 - \beta_j^2} (\cos \beta_i t \cos \beta_j t + \frac{\beta_j}{\beta_i} \sin \beta_i t \sin \beta_j t) \tag{6.15}$$

$$\int \sin \beta_i \sin \beta_j dt = \frac{\beta_i}{\beta_i^2 - \beta_j^2} (-\cos \beta_i t \sin \beta_j t + \frac{\beta_j}{\beta_i} \sin \beta_i t \cos \beta_j t) \tag{6.16}$$

$$\int \cos \beta_i \cos \beta_j dt = \frac{\beta_i}{\beta_i^2 - \beta_j^2} (\sin \beta_i t \cos \beta_j t - \frac{\beta_j}{\beta_i} \cos \beta_i t \sin \beta_j t) \tag{6.17}$$

$i, j \in \{1, 3, \ldots, k-1\}, i \neq j$

All sine and cosine terms change highly by small changes of T (if T is selected properly high). And as t_1 and T are selected freely these terms constitute the measurement noise of $\mathrm{E}\{xx^T\}$. But as they are bounded for all t this noise will vanish for high T. Here a sufficiently high T is assumed and all sine and cosine terms are left out.
Then only the two linear terms in (6.12) and (6.13) remain and (6.11) looks like

$$\mathrm{E}\{\bar{x}\bar{x}^T\} = \frac{1}{T} \sum_{i=1,3,..}^{k-1} \frac{1}{2} T \left[\left((2d_i)^2 + (-2e_i)^2 \right) (a_i a_i^T + b_i b_i^T) \right] \tag{6.18}$$

From this equation it can be seen, that $\mathrm{E}\{\bar{x}\bar{x}^T\}$ consists of a combination of k matrices each of them constituted by a squared vector. This can be reformulated as a singular value decomposition which shows that $\mathrm{E}\{\bar{x}\bar{x}^T\}$ can never have more eigenvalues than k. Furthermore the matrices are pairwise multiplied by the same value

$\frac{1}{2}\big((2d_i)^2 + (-2e_i)^2\big)$. As a_i and b_i are dependend because of

$$\|c_i\|^2 = \|a_i\|^2 + \|b_i\|^2 \tag{6.19}$$

only $\frac{k}{2}$ eigenvalues can change freely.
Now T is cancled out and the equation is reformulated using the definitinos of d_i,e_i,a_i and b_i

$$\mathrm{E}\{\bar{x}\bar{x}^T\} = \sum_{i=1,3,..}^{(} k-1) c_i c_{(i+1)} v_i v_{(i+1)}^T \tag{6.20}$$

is obtained which is identical to (6.8) if the bars are left out like it has generally been done earlier also.

□

Lemma 6.1 has got *direct consequences for first and second level control*. Only nonzero output eigenvalues can be controlled by the neocybernetic adaptation. Thus the lemma limits the dimension of latent subspace which can be used to $s \leq \frac{k}{2}$. This statement is included in the following two theorems which concern the general existence of neocybernetic solutions for the considered systems.

Theorem 6.1 *Given a linear closed loop system $\dot{x} = Ax+bu$ with the feedback $u = \varphi x$ and initial value x_0 under the assumptions of Lemma 6.1. Let the k purely imaginary eigenvalues be* not *controllable.*
Then the neocybernetic first level adaptation law $\varphi = B \cdot E\{ux^T\}$ has got a solution if and only if the following equation system has got a solution

$$(A + b\varphi)v_i = \quad \lambda_i v_i \quad i \in [1,k] \tag{6.21}$$

$$\sum_{i=1}^{k} c_i v_i + x_{asym}(0) = \quad x_0 \tag{6.22}$$

$$B\varphi \sum_{i=1,3,..}^{k} c_i c_{(i+1)} v_i v_{(i+1)}^T = \quad \varphi \tag{6.23}$$

If this equation system only results in destabilizing φ's than the real adaptation does not have a solution.

Proof: (6.23) is the adaptation law itself using the result of Lemma 6.1: (6.21) denotes the eigenvector equation of the closed loop system and (6.22) denotes the initival value problem of the general solution.

□

In the same way a theorem for the second level control can be formulated by including the *Matrix Riccati Equation* to the system. By this theorem the "alternative method" for second level adaptation like it is presented in Sec. 5.2 is used.

Theorem 6.2 *Given a linear system $\dot{x} = Ax + bu$ with the feedback $u = -R^{-1}b^T Px$ and initial value x_0 under the assumptions of Lemma 6.1. Let the k purely imaginary eigenvalues be* not *controllable. Furthermore let R be the given $m \times m$ input weighting matrix. And let there be a $s \times n$ matrix φ which determines the LQ weighting matrix Q such that*

$$Q(\varphi) = diag\left\{exp(\sum_{i=0}^{s} \vec{\varphi}_i)\right\} \tag{6.24}$$

and

$$\varphi = \begin{pmatrix} \vec{\varphi}_1 \\ \vdots \\ \vec{\varphi}_s \end{pmatrix} \tag{6.25}$$

Then the neocybernetic second level adaptation law $\varphi = B \cdot \varphi E\{xx^T\}$ has got a solution if and only if the following equation system has got a solution

$$(A - bR^{-1}b^T P)v_i = \lambda_i v_i \quad i \in [1, k] \tag{6.26}$$

$$\sum_{i=1}^{k} c_i v_i + x_{asym}(0) = x_0 \tag{6.27}$$

$$B\varphi \sum_{i=1,3,..}^{k} c_i c_{(i+1)} v_i v_{(i+1)}^T = \varphi \tag{6.28}$$

$$-PA - A^T P + PbR^{-1}b^T P = Q(\varphi) \tag{6.29}$$

Proof: (6.28) is the adaptation law itself using the result of Lemma 6.1: (6.26) denotes the eigenvector equation of the closed loop system

and (6.27) denotes the initival value problem of the general solution. (6.29) is the algebraic Riccati equation solving the linear quadratic optimization problem for the controller.

□

The theorems above consider only the homogenous case, which is important for the models of this thesis. If inhomogenous systems are used which are fed by external signals the situation can change completely. Test simulations showed that the output covariance matrix of an asymptotically stable linear system can have two linearly independ eigenvectors for each sine wave which is included in the input. This allows full rank output covariances even for SIMO systems. But the input has to be fast enough as the next lemma shows.

Lemma 6.2 *Consider an asymptotically stable linear system of dimension n with an periodic m-dimensional input u ($m < n$).*

$$\dot{x} = Ax + Bu \tag{6.30}$$

Let A be invertible and let the asymptotical dynamics of A be so fast ($\Re(\lambda_i) << 0$) that transient dynamics are neglectable. Therefore the system is always in steady state.

$$0 = Ax + Bu \tag{6.31}$$

Then the state covariance matrix is given by

$$E\{xx^T\} = A^{-1}B \cdot E\{uu^T\} \cdot (A^{-1}B)^T \tag{6.32}$$

Hence $E\{xx^T\}$ has maximum rank m.

Proof: As (6.31) holds and as A is invertible there holds

$$x = -A^{-1}Bu \tag{6.33}$$

$\mathrm{E}\{xx^T\}$ is defined as the mean value of xx^T.

$$\mathrm{E}\{xx^T\} = \frac{1}{T} \int_{t_1}^{t_1+T} x(t)x(t)^T dt \tag{6.34}$$

Substituting (6.33) into (6.34) gives

$$\begin{aligned} \mathrm{E}\{xx^T\} = & \quad \tfrac{1}{T}\int_{t_1}^{t_1+T} A^{-1}Buu^T(A^{-1}B)^T dt \\ = & \quad A^{-1}B \cdot \tfrac{1}{T}\int_{t_1}^{t_1+T} uu^T dt \cdot (A^{-1}B)^T \\ = & \quad A^{-1}B \cdot \mathrm{E}\{uu^T\} \cdot (A^{-1}B)^T \end{aligned} \tag{6.35}$$

Because of $m < n$, $\mathrm{E}\{uu^T\}$ limits the rank of $\mathrm{E}\{xx^T\}$.

□

Consequences of the presented theorems

Theorem 6.1 explains the problems which appeared during MIMO adaptations in Sec. 4.2.2. There the 10 states MIMO model was used. This model has *only one pair of purely imaginary eigenvalues* modeling the disturbance. Hence only one eigenvalue can be controlled freely and no final solution could be found. After a quite fast adaptation the iteration became very slow (explained in Sec. 4.3.4). Using Lemma 6.1 the simulator could be replaced by eigenvector calculations increasing the simulations speed enormously. It turned out that the slow changes describe an oscillation as shown in Fig. 6.1.
When the 18 states model including 4 purely imaginary states was used 2 output eigenvalues could be controlled successfully.
It also needs to be mentioned that the equation system (6.21) - (6.23) can also be solved using for example any kind of *Newton solver* instead of the adaptation law. In the case shown in Fig. 6.1 the Newton solver just failed to solve the problem which gives some better information about the situation. In the solveable case both solver found the same solution whereby the neocybernetic adaptation law was faster. So it has not been figured out yet which way of solving the problem works better but in any case Thms. 6.1 and 6.2 reduce the highly iterative adaptation using simulations to a *completely offline optimization process*.

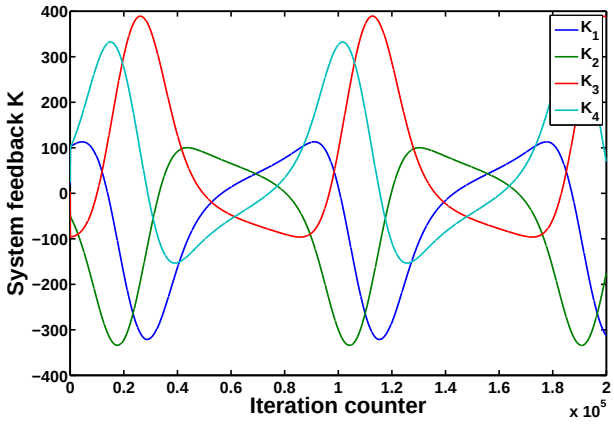

Figure 6.1.: Feedback $\hat{\varphi}$ during adaptation using the mixed input structure and Lemma 6.1 for determining $\mathrm{E}\{xx^T\}$. For most parts of the diagram the system is unstable.

Chapter 7

Discussion and Evaluation

7.1. Summary

The thesis in hand describes one "theoretical iteration step" for the so called neocybernetic theory.
At first the new way of thinking is introduced. Regarding natural systems stability is no longer the main goal but the general assumption because natural systems are (almost) always stable. From this perspective complex, distributed, high order systems are tackled by one very simple idea: Connect each input independently of other inputs with the outputs dependening on its correlation to them. Then it is shown that these simple local connections lead to global structure (PCA).
In the next chapters these ideas are applied to a typical technical system. It is shown that the basic principles work and simple (adaptive) controllers can be found. Because of very evident limitations of P controllers the method is then extended to second level control: The system itself is controlled by a traditional LQR but the LQR is controlled with neocybernetic principles. Although this idea seems to fit perfectly to the basic assumption of negative feedback the situation turned out to be even more difficult.
So the last chapter tries to provide the desire for explanations. The adaptation law and the considered type of system are examinend in a mathematical way and some restrictions are found and proven.

7.2. Critical view of neocybernetics

Neocybernetic controllers have got several properties which can cause problems to technical applications. Most of the properties are directly connected to the basic assumptions and to the properties of neocybernetics. Hence it is not fair to critisize the whole theory because of these problems. But they are mentioned here as *weak points of neocybernetics in a technical context.*

unhandled stability: Stability is generally assumed for neocybernetic systems and thus all stablity issues are ignored by the adaptation and by the resulting controllers. But stability is a very basic question for technical systems. Natural systems are truly always stable (in a global perspective), technical systems may not be stable (or the stablelizing effect is neglected), and instablities sometimes mean the bad malfunction of the system. Of course all adaptation runs can be done savely in the simulator, but this weak point remains as adaptation can lead to instablities even if there exists a stable controller with the same controller structure.

unneeded structure: Neocybertetic controllers always implement a PCA structure. It forces the controller to use the principal components. This limitation reduces the solution space dramatically, although it is not necessarily important. Especially if stablity gets lost the price is too high.

ignored transient dynamics: Neocybernetics per definition ignores alle transient dynamics. It does not care about how long it takes to reach the steady state. This means that very basic questions about controller perfomance are left open, so that neocybernetics can be used only if these questions do not matter.

lack of information: The last problem also depends on the previous one: If only the stationary state is considerd a lot of information about the system is not used. As shown in this thesis a homogenous linear system has got only as many output covariance eigenvalues than system eigenvalues on the imaginary

axis. An asymptotic stable linear homogenous system even has no nonzero output eigenvalues. Thus the neocybernetic control suffers from a lack of information which constricts the variation. This can be seen at second level control: In the example here it was either a 1 dimensional latent subspace controlling 6 $(n + m - k)$ free parameters or a 2 dimensional subspace controlling 16 free parameter. Not that this situation can change completely if external signals feed the system (inhomogenous case).

7.3. Outlook

Although there appeared more problems and limitations than prosperities for the vibration control of the considered electrical machines or homogenous linear systems in general it also confirms that *the basic neocybernetic principals do work*. It seems that neocybernetic controllers are obviously not very suitable for linear homogenous technical problems but where do they fit better to? All mentioned critical points can vanish if a naturally (e.g. nonlinear) stable system is considered whose transients are out of interest and which contains a lot of redundant statistical information. It was already mentioned that inhomogenous linear systems can have a full rank output covariance matrix depending on the inputs. [1] introduces neocybernetics on a biological background. Also commercial, social or networked systems are conceivable.
The basic feature of local acting and global emergent structure was not needed for this thesis. How about systems which are to complex for centralized analyses?

Appendix A

Implementations in Matlab

Matlab script: Runs adaptation series. Implementation of "Parameter Loop" in Fig. 4.2

```
% CalcBSeries
% Calculates a series of adaptions with different B
% author: Heiko Patz

% initial values
saveB = 0;          % array of all used B
saveEux=0;          % array of all Eux
saveExx=0;          % array of all Exx
saveCE=0;           % array of all control errors

step = 2;           % Stepsize between adaptations
BVerified = saveB(length(saveB));
B = 0;

while (step > 1e-3),
        B=BVerified+step;
        disp(sprintf('Trying B=%f',B));

        saveK=K;            % Backup of K
        RunAdaptation;      % Call Adaptation script

        % Did system converge?
        if (phi==phi_old),  % 'no'
                step=0.5*step;
                K=saveK;
                disp(sprintf('adaption failed. Trying step size= %f',step));
        else                % 'yes'
                saveExx=[saveExx;xx_mean];
                saveEux=[saveEux;covUX];
                saveCE=[saveCE;ControlError];
                saveB=[saveB;B];
                BVerified=B;
                disp(sprintf('adaptation finished. B verified at %f',B));
        end
end             %of while
```

Matlab script: Runs an adaptation loop. Implementation of "Adaptation Loop" in Fig. 4.2

```
% Iteration program for adaptation of controller to the vibration model
% author: Heiko Patz

% Find input output dimensions:
[n in] = size(StateSpaceSystem.b);
[out n] = size(OutputMatrix);

% Set initial values
% ****************************************************************
if (InitialRun==1)
        % set a stable Initial Controller:
        K=[1.719291e+001,-2.028678e+000;4.203311e+000,2.391040e+001];
        phi = K;
        virgin = 0;             % equals one in first run for initialisation
        forget=0.8;             % Filter factor F
        % Initialize array for adaptation diagrams
        phis = zeros(200000,numel(K)+2+in);
        IterationCount = 1; % counter reset
end
% ****************************************************************

phi_old = 24*K;       % just different from phi!
% Mapping matrix phi conferged ??
while sum(sum(abs((phi-phi_old)))) > 5e-3,

        % run the simulationen
        SimulateVibrationModel;

        % Was system stable?
        if (StabInd > -1)
                phi_old = phi;
                phi = forget*phi+ (1-forget) * B*CurCovUX;
        %unstable system -> abort adaptation
        else
                display('Instable system behaviour detected...');
                K=0;
                % stop emergin loop:
```

```
        break;
    end

    if (DisplayOn==1)
        % How good does phi fit to EV(CurCovXX) ?
        [V D]=eigs(xx_mean);
        V=[V(:,2)';V(:,1)'];
        % Calculate Minium squared Error between phi and V
        F=diag([(phi(1,:)*phi(1,:)')^-1*phi(1,:)*V(1,:)',
                (phi(2,:)*phi(2,:)')^-1*phi(2,:)*V(2,:)']);
        phi_err=diag((F*phi-V)*(F*phi-V)');
        display(sprintf('phi = [%s]',sprintf('%e,',phi')));
        msg=sprintf('B =%f ControlErr = [%s] phi_err = [%s] D = [%f %f]',B(1,1),
                sprintf('%f ',ControlError),sprintf('%f ',phi_err),D(1,1),D(2,2));
        % If simulation is replaced by calculation stability is indicated here
        if (StabInd==1)
            msg=sprintf('%s (stable)',msg);
        end
        if (StabInd==0)
            msg=sprintf('%s (unstable)',msg);
        end
    end
    disp(sprintf('%s Iteration: %d',msg,IterationCount)); end
    % Save adaptation parameters
    phis(IterationCount,:) = [K(1,:),K(2,:),ControlError,D(1,1),D(2,2)];
    IterationCount=IterationCount+1;
end         % of while
```

Matlab script: Runs the Simulator or calculates the simulation results. Implementation of "Simulator" in Fig. 4.2 for the 10 states MIMO model

```
% SimulateVibrationModel.m:
% runs the simulation and calculates covariance matrices of x and u
% author: Heiko Patz

% Model parameters
% Select two states out of 10
OutputMatrix = [[0,3.686801955454,0,0;0,0,0,3.738608167152],zeros(2,6)];
buffer = 20000;      % number of values to be kept from last
                     % values of x and u
OmegaN = 49.5;       % nominal frequency of machine in rad/s

% Apply new feedback faktor
K = phi;

if (ReplaceSimulator==1)
        % Calculate Simulation result
        [V D] =eig(StateSpaceSystem.a+StateSpaceSystem.b*K*OutputMatrix);
        %Initial values for 10 state model: x0=[0,0,0,0,0,0,0,0,42,0];
        c=V^-1*[0,0,0,0,0,0,0,0,42,0]';
        CurCovXX=(c(9)*c(10)*(V(:,9)*V(:,10).'+V(:,10)*V(:,9).'));
        % Eliminate imaginary numerical leftovers:
        CurCovXX=real(CurCovXX);
        CurCovXX=OutputMatrix*CurCovXX*OutputMatrix';
        CurCovUX=phi*CurCovXX;
        if (real(D)<=0)
                StabInd=ones(1,10);          % stable system
        else
                StabInd=zeros(1,10);         % unstable system
        end
else
        % Simulate until steady state
        sim ('VibrationModel2D');            % run Simulink model
        x=x*OutputMatrix';
        % Calculate some kind of derivative between first half of x and second
        % one:
        StabInd = sum(x(1:0.5*buffer,:).*x(1:0.5*buffer,:));
```

```
        StabInd = StabInd - sum(x(0.5*buffer:buffer,:).*x(0.5*buffer:buffer,:));
        % Calculate means of uxT and xxT products for steady state period
        CurCovUX = 1/length(x) * u'*x;
        CurCovXX = 1/length(x) * x'*x;
end

% Calculate squared err of displacement
ControlError=sqrt([CurCovXX(2,2),CurCovXX(4,4)]);
```

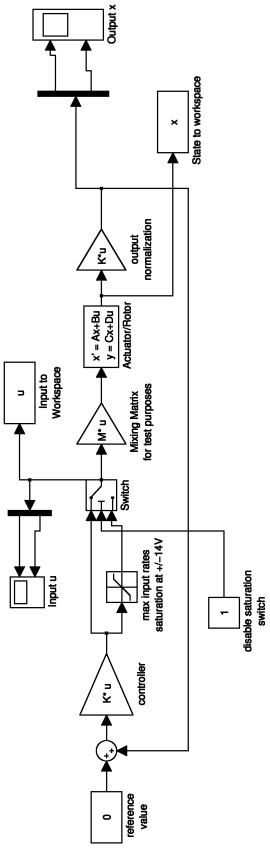

Figure A.1.: Implementation sample for a 2-dimensional MIMO model in Simulink

Appendix B

Additional figures

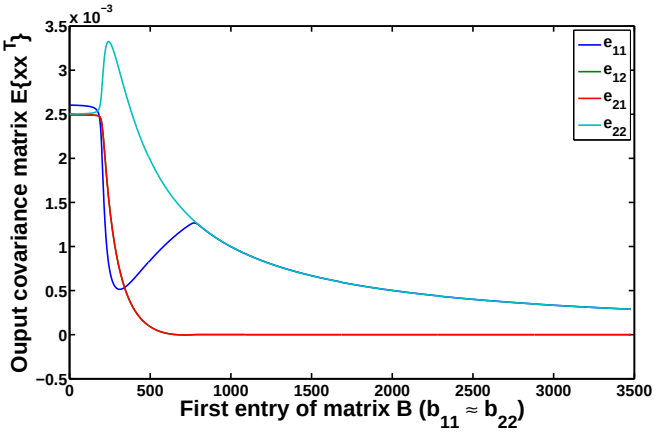

Figure B.1.: MIMO adaptation series: Output covariance matrix for a set of $B \approx b \cdot I$.

Bibliography

[1] H. Hyötyniemi : Neocybernetics in biological systems. Helsinki University of Technology, Control Engineering Laboratory, Report 151, 2006.

[2] Dipl. kyb. Michael Sailer: Distributed Control of a Deformable System - Analysis of a Neocybernetic Framework, pg 3-45, Espoo, Finnland, March 2006

[3] A. Laiho, A. Burakov, K. Tammi, K. Zenger, A. Arkkio: Active control of radial rotor vibration in cage induction electrical machines by a built-in force actuator, Finland: Helsinki University of Technology, Control Engineering Laboratory, IDETC/CIE 2007 (accepted for publication)

[4] A. Basilevsky: Statistical Factor Analysis and Related Methods. John Wiley & Sons, New York, 1994

[5] H. Hyötyniemi: Hebbian Neuron Grids: System Theoretic Approach. Technical report, Helsinki University of Technology, Control Engineering Laboratory,Report 144, 2004. http://www.control.hut.fi/hyotyniemi/publications/04_report144.htm

[6] R.D. Cook, D.S. Malkus, M.E. Pseha and R.J. Witt: Concepts and Applications of Finite Element Analysis. Wiley & sons, 2001 (4th edition)

[7] Torkel Glad and Lennart Ljung: Control Theory, Multivariable and nonlinear Methods, 2000

[8] Comsol Multiphysics: Modelling and simulation software package, http://www.femlab.com/products/multiphysics/

[9] Helsinki University of Technology, Automation Control Laboratory: Homepage of ACRVEM project, 2007, http://www.control.hut.fi/Research/Acrvem/