Lehrstuhl für Mathematische Systemtheorie
Universität Stuttgart

Projektarbeit

im Bachelorstudiengang
Simulation Technology

Iterative Learning Control

Erstgutachter/in: Prof. Dr. Carsten W. Scherer
Zweitgutachter/in: ....

vorgelegt von

Name
Straße
PLZ + Ort
E-mail

Studienfach
Fachsemester
Matrikelnummer
Abgabetermin: XX.XX.201X

Contents

List of Figures

List of Tables

Abkürzungsverzeichnis

O.B.d.A ...

Symbolverzeichnis

$G(s)$	Übertragungsfunktion eines LTI Systems
(A, B, C, D)	Realization des Systems im Zustandsraum

Chapter 1

Basics

<Einfuehrung>

We consider a discrete time iteration

$$\begin{array}{rcl} x(t+1) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) + Du(t), \end{array} \tag{1.1}$$

$$x(0) = x_0, \quad t \geq 0, \tag{1.2}$$

with error

$$e(t) = r(t) - y(t) \tag{1.3}$$

at time $t = 0, 1, 2, \ldots$.

Here $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times l}$, $C \in \mathbb{R}^{m \times n}$, and $D \in \mathbb{R}^{m \times l}$ are real matrices, $r(t) \in \mathbb{R}^m$ is a reference signal at time t. For notation simplification, we write (A, B, C, D) and mean it to be the discrete-time control system (1.1).

We aim to find a "good" input signal $u(t)$, such that the reference $r(t)$ is tracked possibly well for all $t \geq 0$. With other words, it means, that the error norm $||e(\cdot)||$ must stay small in some $||\cdot||$, e.g. the L_2-norm. We can achieve it by finding a stabilizing controller – such that our system state goes not to infinity over time t. To design it, we must first clearly define what stability means. Since let us first formulate the common definitions and entertain ideas of a stabilizing controller design.

1.1 Definitions

Definition 1 *A discrete time dynamical system*

$$x(t+1) = Ax(t) \tag{1.4}$$

for $t \geq 0$ is said to be asymptotically stable, if for all initial condition $x(0) \in \mathbb{R}^n$

$$\lim_{t\to\infty} x(t) = 0 \tag{1.5}$$

is satisfied. It is said to be stable, if

$$\limsup_{t\to\infty} ||x(t)|| < \infty \tag{1.6}$$

for all initial condition $x(0) \in \mathbb{R}^n$. We call the matrix A (asymptotically) stable, if the dynamical system (1.4) *is (asymptotically) stable.*

In the same way we want to define (asymptotic) stabilizability and detectability.

Definition 2 *A a discrete time system (A, B, C, D) or the pair (A, B) is (asymptotically) stabilizable, if there exists a matrix F, such that $A - BF$ is (asymptotically) stable. The system or the pair (A, C) is detectable, if (A^T, C^T) is assymptotically stabilizable.*

1.2 Stabilizing Controller based on Separation Principle

We can find a stabilizing controller for (1.1) using feedback control. First let us consider the case $y(\cdot) = x(\cdot)$.
Then the system (1.1) becomes stable, if we choose $u(t) = -Fx(t)$ with some matrix F, such that $(A - BF)$ is stable.
Clearly, if the system is (asymptotically) stabilizable, we can find such matrix F.
But what is a "good" choice for F? One option is the so-called Linear-Quadratic-Regulator (LQR), which minimizes the cost function

$$V(u(\cdot)) = \sum_{t=0}^{\infty} \left((x(t))^T Q(x(t)) + u(t)^T R u(t) \right), \tag{1.7}$$

overall stabilizing piecewise constant input signals $u(\cdot) \in \mathbb{R}^m$, and with $x(\cdot)$ satisfying (1.1) (with $C = I$ and $D = 0$). $Q \succ 0$ and $R \succ 0$ are here some weighting matrices, and allow us to weight elements in input and output signals. For example, one of the inputs can impact the system as another, or one output must be driven faster.
The solution of this optimal control problem can be found by choosing

$$F = (R + B^T P B)^{-1} B^T P A. \tag{1.8}$$

where P is a unique symmetric positive definite matrix, the stabilizing solution of the

discrete time algebraic Riccati equation

$$A^TA - A^TPB(R + B^TPB)^{-1}B^TPA + Q - P = 0. \tag{1.9}$$

Stabilizing solution means, that from (1.8) resulting matrix $(A - BF)$ is stable, and hence this solution stabilizes our system.
For the case $y(\cdot) \neq x(\cdot)$, we can additionally construct an observer to approximate the signal $x(\cdot)$. The controller we get is said to be based on *separation principle*. It requires the pair (A, C) to be asymptotically detectable which allows us to find a matrix J, such that $A - JC$ is asymptotically stable.
We set

$$u = -F\hat{x}, \text{ with } \hat{x} \text{ given via dynamical system} \tag{1.10}$$

$$\hat{x}(t+1) = A\hat{x}(t) + Bu(t) + J(y - \hat{y}(t)), \tag{1.11}$$
$$\hat{y}(t) = C\hat{x}(t) + Du(t), \tag{1.12}$$
$$\hat{x}(0) = x_0, \ t = 0, 1, 2, \ldots, N. \tag{1.13}$$

With this observer system the estimation error can be calculated as

$$\tilde{x}(t+1) = x(t+1) - \hat{x}(t+1) = (A - JC)\tilde{x}(t) \tag{1.14}$$

for $t \geq 0$ and hence $\hat{x}(\cdot)$ converges to $x(\cdot)$ at least asymptotically.
We set M as a right inverse of the matrix

$$D + (C - DF)(I - A + BF)^{-1}B \tag{1.15}$$

Then the choice

$$u = -F\hat{x} + Mr, \tag{1.16}$$

solves tracking issue for constant $r(\cdot)$. It still can be used for piecewise constant $r(\cdot)$, if the signal does not change too fast.

Chapter 2

ILC Algorithms

<Einfuerung>

Let us imagine that we need to process the same action multiple times. For example, a robot manipulator must put some objects in a box with high accuracy, while the objects to put are always located in the same place. If we already have some input sequence, which solves this issue, we can use the same data for the next iteration and get the same precision. The other possibility is to "learn" from the previous iteration and try to enhance the exactness, while the tracking objective $r(\cdot)$ stays the same over all iterations.

More formally, we consider the system (1.1) over a *finite time horizon* $t = 0, 1, 2, \ldots, N$, $N \in \mathbb{N}$. As we discussed in the previous chapter, we are interested in good tracking of the signal $r(\cdot)$. At first glance, the stability concept seems to be not necessary for $N < \infty$. Still, in this thesis, it is the requirement we presuppose on considered system. The reason is that also the discrete system over finite horizon can produce vast signals for large N, if the system is not stable. Especially, it means, that A has at least one eigenvalue larger that one, which implies very large norm of the matrices A^t for $t \gg 1$. This makes difficult the numerical calculations, since the solution of the difference equation (1.1) includes it.

Moreover, for our purposes, for stabilizable systems we can assume stability without loss of generality.

If the system is stabilizable, we can find such a matrix F, such that $(A - BF)$ is stable. We define $u(t) = -Fx(t) + \tilde{u}(t)$, $t = 0, 1, 2, \ldots, N$ with a new input signal $\tilde{u}(t) \in \mathbb{R}^l$. Then the system (1.1) results in

$$\begin{aligned} x(t+1) &= (A - BF)x(t) + B\tilde{u}(t), \\ y(t) &= (C - DF)x(t) + D\tilde{u}(t),\ t = 0, 1, 2, \ldots, N. \end{aligned} \tag{2.1}$$

We can set

$$\begin{array}{lll} \tilde{A} & = (A - BF), & \tilde{B} \quad = B \\ \tilde{C} & = (C - DF), & \tilde{B} \quad = D. \end{array} \tag{2.2}$$

The shifted system $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$ is stable and we still can improve the input signal by considering $\tilde{u}(\cdot)$.

Example 3 *Let us consider the system (A, B, C, D) with*

$$A = \begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix}, B = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, C = \begin{pmatrix} 0 & 1 \end{pmatrix}, D = 2. \tag{2.3}$$

This system is controllable and observable, and hence we can calculate the matrices F and J, such that

$$(A - BF) \text{ and } (A^T - C^T J) \text{ are stable.} \tag{2.4}$$

With MATLAB we calculate the solutions f discrete algebraic Riccati equation (for $Q = I_m$, $R = I_l$) and get the shifting matrices

$$F = \begin{pmatrix} 1.9282 & 1.2679 \end{pmatrix} \text{ and } J = \begin{pmatrix} 1.8304 & 4.6385 \end{pmatrix}. \tag{2.5}$$

Then we re-describe our system as

$$\begin{array}{rcl} x(t+1) & = & \begin{pmatrix} 0.0718 & -0.2679 \\ 0.1436 & 0.4641 \end{pmatrix} x(t) + \begin{pmatrix} 1 \\ 2 \end{pmatrix} \tilde{u}(t) \\ y(t) & = & \begin{pmatrix} -3.8564 & -1.5359 \end{pmatrix} x(t) + 2\tilde{u}(t), \qquad t = 0, 1, 2, \ldots, N. \end{array} \tag{2.6}$$

We take as a reference $r(\dot{})$ the pulse signal and set $N = 20$.
With controller based on separation principle (here shortly: LQR), we get the tracking illustrated in Figure 2.1.

As one can see, the tracking in Example 3 is not perfect at all. However, we can extract a bunch of information from this controlled system. For example, we get the input and output sequences

$$\{u(0), u(1), u(2) \ldots, u(N)\} \subset \mathbb{R}^l \tag{2.7}$$

$$\{y(0), y(1), y(2), \ldots, y(N)\} \subset \mathbb{R}^m. \tag{2.8}$$

This sequences are limited due to the finite time horizon. It seems to be an advantage.

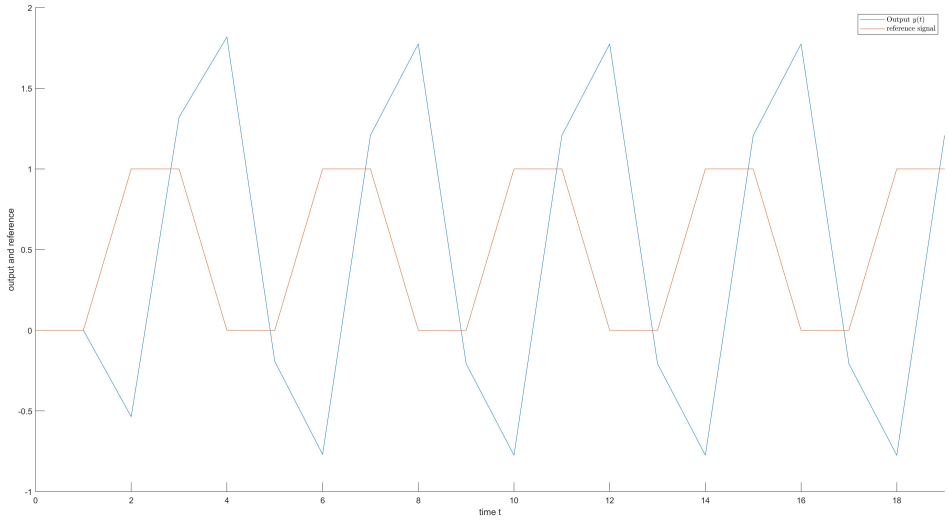

Figure 2.1: Tracking with controller based on separation principle for system (2.6)

Indeed, if we put the signals we got from simulation together, we get a large, but finite dimensional vector. That means, the standard tools from linear algebra can be applied.

As a matter of fact, we can express our whole simulated system as a linear equation. We call such vectors *Supervectors*, and the whole system is said to be "lifted".

2.1 Supervector description

For the sequences (2.7) and (2.8) we define the supervectors

$$u = \begin{pmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(N) \end{pmatrix} \in \mathbb{R}^{l(N+1)}, \; y = \begin{pmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N) \end{pmatrix} \in \mathbb{R}^{m(N+1)}. \tag{2.9}$$

Respectively, we determine supervectors $r \in \mathbb{R}^{m(N+1)}$ and $e \in \mathbb{R}^{m(N+1)}$.

For system (1.1) the relation between u and y, and the error e can be written in a linear matrix form

$$y = Gu + d, \tag{2.10}$$

$$e = r - y. \tag{2.11}$$

The matrix G represents here the state-space model and is given via

$$G = G(A,B,C,D) = \\ = \begin{pmatrix} D & 0 & \cdots & 0 & 0 & 0 \\ CB & D & \cdots & 0 & 0 & 0 \\ CAB & CB & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ CA^{N-2}B & CA^{N-3}B & \dots & CB & D & 0 \\ CA^{N-1}B & CA^{N-2}B & \dots & CAB & CB & D \end{pmatrix} \in \mathbb{R}^{m(N+1)\times l(N+1)}. \tag{2.12}$$

The vector d depends on the initial condition x_0:

$$d = d(C,A,x_0) = \begin{pmatrix} Cx_0 \\ CAx_0 \\ CA^2x_0 \\ \vdots \\ CA^Nx_0 \end{pmatrix} \in \mathbb{R}^{m(N+1)}. \tag{2.13}$$

The system (2.10) looks pretty simple. If we know the signal r we want to track, we can find a perfect solution by choosing

$$u_\infty = G^{-1}r - d. \tag{2.14}$$

However, this choice does not look greatly credible.
For example, if our system is uncertain and is described by $\tilde{G}$, but modeled by G, we get

$$e = r - y = (I - \tilde{G}G^{-1})r. \tag{2.15}$$

This error might be non-zero, even for a little model perturbation. If we can reformulate this idea in a more robust way, we may get a good and simple applicable algorithm, which we call the Inverse Model Algorithm.

2.2 Inverse Model Algorithms

Algorithm 1 *Let the matrix D in (1.1) have an inverse. Then the matrix $G = G(A,B,C,D)$ is non-singular as well, and the Inverse Model Algorithm (IA) is given via input update law*

$$\begin{aligned} u_{k+1} &= u_k + \beta G^{-1}e_k, \\ u_0 &\in \mathbb{R}^{l(N+1)}, \end{aligned} \tag{2.16}$$

with error evolution

$$\begin{aligned} e_{k+1} &= (1-\beta)e_k,\ k \geq 0, \\ e_0 &= r - Gu_0 - d. \end{aligned} \tag{2.17}$$

In particular, $(e_k)_{k\geq 0}$ *converges to zero for* $k \to \infty$ *for any initial error* $e_0 \in \mathbb{R}^{m(N+1)}$ *if and only if*

$$0 < \beta < 2.$$

Proof:
Firstly, the matrix G has lower triangular structure, with the matrix D on its diagonal. Hence, $\det(G) = \det(D)^{N+1}$, which is non-zero if and only if D has full rank. Taking the limit over relation (2.17) results in

$$\lim_{k\to\infty} e_{k+1} = \lim_{k\to\infty} (1-\beta)e_k = \lim_{k\to\infty} (1-\beta)^k e_0. \tag{$*1$}$$

Choosing $0 < \beta < 2$ yields the proof. ■

For β close to 0 or 2, we get slower convergence and better robustness. For $\beta = 1$ we get convergence in one iteration, but this choice might be highly non-robust [1], pp 149, 152-155.

Example 4 *In the system* (2.6) *the matrix* $D = 2$ *is invertible, and we can apply IA. Result for* $N = 20$ *and* $\beta = .1$ *is illustrated in Figure* **??**.

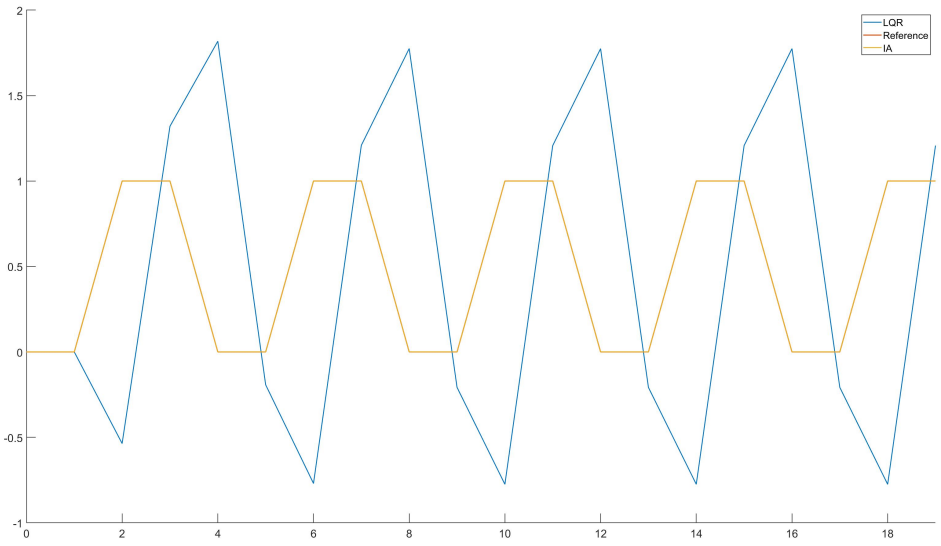

Figure 2.2: Tracking with LQR Controller (without ILC Algorithm), and tracking with Right Inverse Model Algorithm for the system (**??**)

But if we try to increase the time horizon N, for example to 50 steps, we can see a wrong system behavior (Figure 2.3).

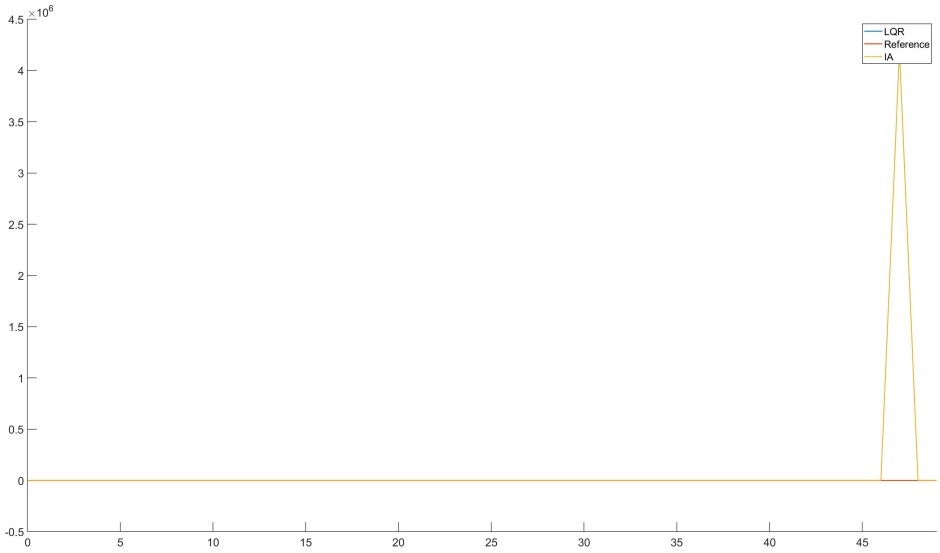

Figure 2.3: Application of the result of RIA for 2.6 for $N = 50$

The reason might be the condition number of the matrix G, which equals 4.5041e+16. We see: even for a triangular matrix with non-zero eigenvalues, the matrix inversion can be not numerically adjustable.

The Inverse Model Algorithm sets up very strong requirements: the matrix G should be non-singular, and its condition number must be small enough we can compute a numerically stable inverse. We can modify this algorithm by taking the Moore-Penrose inverse. This is the so-called pseudo inverse, which is uniquely defined and can be computed for all matrices [4].

Definition 5 *For a matrix $G \in \mathbb{R}^{m(N+1)\times l(N+1)}$ a psedo inverse is defined by matrix $G^+ \in \mathbb{R}^{l(N+1)\times m(N+1)}$, satisfying the criteria:*

(a) $GG^+G = G$

(b) $G^+GG^+ = G^+$

(c) $(GG^+)^* = GG^+$

(d) $(G^+G)^* = G^+G$

Algorithm 2 *The Pseudo Inverse Model Algorithm (PIA) is given via input update law*

$$\begin{aligned} u_{k+1} &= u_k + \beta G^+ e_k, \\ u_0 &\in \mathbb{R}^{l(N+1)}, \end{aligned} \tag{2.18}$$

with error evolution

$$\begin{aligned} e_{k+1} &= (1 - \beta G G^{+}) e_k, \; k \geq 0, \\ e_0 &= r - G u_0 - d. \end{aligned} \tag{2.19}$$

Monotonic convergence to

$$e_\infty = \lim_{k \to \infty} e_k = P_{\ker[G^+]} e_0, \tag{2.20}$$

is guaranteed if

$$0 < \beta < 2$$

$P_{\ker[G^+]}$ *denotes the positive orthogonal projection operator onto* $\ker[G^+]$. *In particular, the zero convergence is attainable, if and only if the tracking signal* r *is feasible.*

Proof:
Proof wird hier stehen wenn es so passt ■

Example 6 *We calculae a solution for* (2.6) *with PIA, and illustrate it in Figure 2.4. This time the tracking signal and system output fit perfectly despite the large condition number.*

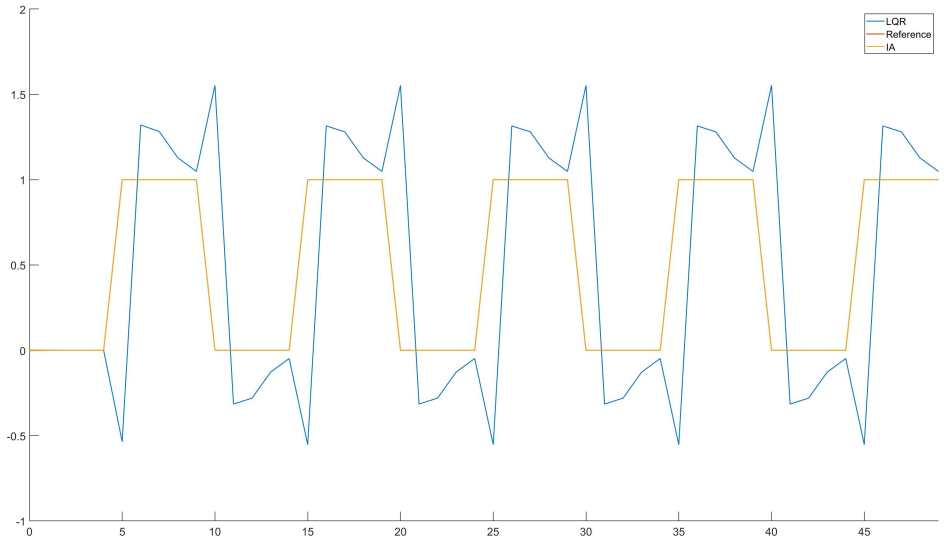

Figure 2.4: Reference signal tracking with LQR and PIA

We can replace the use of pseudo inverse by taking a left or right inverse, if it exists. However, it only makes sense if the calculating afford or numerical stability are better for left or right inverse calculation.

The pseudo inverse has its advantage to equal the right/left inverse, if it exists. Moreover, for right/left inverse we need to restrict the rank requirements, as it exists not for every matrix. The algorithm can be applied for any matrix G.
(Pseudo) Inverse Algorithm belongs to the class of so-called Unit Memory Algorithms – these are the algorithms, where we assume to 0"remember" the last trial.

2.3 Unit Memory Algorithm

We are interested in the processes, which are executed repetitively. Let $k = 0, 1, 2, \ldots$ be the number of completed iterations. Then we get a system

$$\begin{aligned} y_k &= Gu_k + d, \\ e_k &= r - y_k, \end{aligned} \tag{2.21}$$

$u_0 \in \mathbb{R}^{l(N+1)},\ k = 0, 1, 2, \ldots$. This is again a discrete-time system, with time increment k.
To achieve a good tracking at trial k we need to ensure that the norm $||e_k||$ is small. The tracking behavior improves for consecutive iterations if the sequence $||e_k||$ is monotonically decreasing, i.e.,

$$||e_{k+1}|| < ||e_k|| \text{ for all } k \geq 0. \tag{2.22}$$

Note that in this case the sequence $||e_k||$ is guaranteed to converge. If the sequence even converges to zero we say that perfect tracking is achieved asymptotically.
We have already seen, that it is not always possible to achieve zero convergence. In Algorithm 2 the perfect tracking is possible, if the reference signal r is feasible. In common the convergence properties can also depend on the choice u_0.
We assume linear dependency of u_{k+1} upon u_k and e_k for $k \geq 0$, and define the following algorithm using feedback control.

Algorithm 3 *The Unit Memory Algorithm is given via input update law*

$$\begin{aligned} u_{k+1} &= u_k + Ke_k, \\ u_0 &\in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.23}$$

The error dynamic is then given via

$$e_{k+1} = (I - GK)e_k,\ k \geq 0, \tag{2.24}$$

$$e_0 = r - Gu_0 - d. \tag{2.25}$$

$K \in \mathbb{R}^{l(N+1)\times m(N+1)}$ *is called* ***learning matrix****.*

To ensure the stability of the algorithm, it is enough to consider the iteration process

$$e_{k+1} = (I - GK)e_k = (I - GK)^k e_0 := L^k e_0,\ k \geq 0. \tag{2.26}$$

That it is a discrete time dynamical system over k. We can reformulate our goal as follows: find some matrix $K \in \mathbb{R}^{l(N+1)\times m(N+1)}$, which renders (2.26) stable.
We can also rewrite our system as

$$\begin{pmatrix} e_{k+1} \\ e_k \end{pmatrix} = \left(\begin{array}{c|c} I & -G \\ \hline 0 & K \end{array}\right) \begin{pmatrix} e_k \\ v_k \end{pmatrix} \tag{2.27}$$

Then our problem is reduced to finding a stabilizing controller K with

$$v_k = Ke_k. \tag{2.28}$$

We know how to work with such systems – for example, LQR controller provides a good solution. This is a reliable method, but since the matrix G can have large dimension, it also can be costly. Often we can achieve good results with much more simple methods. For example, we can reduce the number of parameters to be chosen.
In the Algorithms 1 and 2 we chose the fixed matrix K_0 as the (pseudo) inverse G, and left the parameter β to decide on. So, we reduced the number of degrees of freedom to 1, and still get satisfying results.
However, as we have seen in Example 4, the calculation of the inverse matrix is not the most reliable method. The pseudo inverse refers better results, but is also much more expensive.
An algorithm, which does not require the model inversion, might be here of use.

2.4 Gradient Algorithms

We can begin to define a new algorithm with the requirements on it. We want this algorithm to be monotonic convergent, so we can easily decide on termination criterion. The direct calculation provides

$$||e_{k+1}|| < ||e_k|| \text{ for all } k > 0, \tag{2.29}$$

in some norm $||\cdot||$ in $\mathbb{R}^{m(N+1)}$.
We chose some positive definite weighting matrices $Q(t)$ and $R(t)$, $t = 0, 1, 2, \ldots, N$, and define the scalar products

$$\langle y, z\rangle_Q = \sum_{t=0}^{N} y(t)^T Q(t) z(t),\ \langle u, v\rangle_R = \sum_{t=0}^{N} u(t)^T R(t) v(t), \tag{2.30}$$

with $y(t), z(t) \in \mathbb{R}^m$, $u(t), v(t) \in \mathbb{R}^l$ for $t = 0, 1, 2, \ldots, N$.

We denote with $||\cdot||_Q$ and $||\cdot||_R$ the associated norms, and set

$$G^\star = R^{-1}G^TQ \text{ for all } G \in \mathbb{R}^{m(N+1)\times l(N+1)},$$
$$K^\star = Q^{-1}K^TR \text{ for all } K \in \mathbb{R}^{l(N+1)\times m(N+1)},$$

where

$$Q := \text{diag}(Q(0), Q(1), Q(2), \ldots, Q(N)) \text{ and } R := \text{diag}(R(0), R(1), R(2), \ldots, R(N)). \tag{2.31}$$

In the same way, we can define

$$L^\star = Q^{-1}L^TQ \text{ for all } L \in \mathbb{R}^{m(N+1)\times m(N+1)} \text{ and} \tag{2.32}$$
$$M^\star = R^{-1}M^TR \text{ for all } M \in \mathbb{R}^{l(N+1)\times l(N+1)}. \tag{2.33}$$

Recall, that for $R = I$ and $Q = I$ the matrices $G^\star$ and $K^\star$ are just the conjugate transpose. The notation from above can be seen as conjugate transpose in the vector space with scalar products (2.30).

Applying the weighted norms on (2.29) provides

$$\begin{aligned} ||e_{k+1}||_Q^2 &= ||e_k - GKe_k||_Q^2 = ||e_k||_Q - 2\langle e_k, GKe_k\rangle_Q^2 + ||GKe_k||_Q^2 \\ &\leq ||e_k||_Q^2, \text{ if } ||GKe_k||_Q^2 < 2\langle e_k, GKe_k\rangle_Q. \end{aligned} \tag{2.34}$$

The last inequality is equivalent to

$$\frac{1}{2}e_k^TK^TG^TQGKe_k < e_k^TQGKe_k, \tag{2.35}$$

If we set $K = \beta G^\star$ it results in

$$\beta^2(GG^\star)^TQ(GG^\star) \prec \beta QGG^\star + \beta(QGG^\star)^T. \tag{2.36}$$

Since the matrix Q is positive definite, we can find a matrix $Q^{1/2}$, such that $Q = Q^{1/2}Q^{1/2}$. We define the positive semi-definite, symmetric matrix

$$H := Q^{1/2}GG^\star Q^{-1/2} = Q^{1/2}GR^{-1}G^TQ^{1/2}. \tag{2.37}$$

This is a congruence transformation over a symmetric matrix, and hence (2.36) is equivalent to

$$\beta^2H^TH \prec \beta(H + H^T) \Leftrightarrow \beta H^2 \prec 2. \tag{2.38}$$

This is satisfied, if $\beta \in (0, 2/\lambda_{\max}(H))$.

This choice of K determines the *Steepest Descent Algorithm*

2.4.1 Steepest Descent Algorithm

Algorithm 4 *The Steepest Descent Algorithm is characterized by choosing $K = \beta G^\star$, where $\beta > 0$ is a real scalar gain. The iterative law is given via*

$$\begin{aligned} u_{k+1} &= u_k + \beta G^\star e_k, \\ e_{k+1} &= (I - \beta G G^\star) e_k, \ k \geq 0, \\ u_0 &\in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.39}$$

Monotonic convergence to

$$e_\infty = \lim_{k \to \infty} e_k = P_{\ker[G^\star]} e_0, \tag{2.40}$$

is guaranteed if

$$0 < \beta < \frac{2}{\lambda_{\max}(H)}$$

In particular, the zero convergence is attainable, if $e_0 \in \mathrm{im}[GG^\star]$.

Proof:
We consider the matrix $L := (I - \beta G G^\star)$. Since $0 < \beta < 2/\lambda_{\max}(H)$, it holds the relation

$$(1 - \beta ||G^\star|| \cdot ||G||) I \preceq (I - \beta G G^\star), \tag{$*1$}$$

$$\Downarrow \tag{$*2$}$$

$$\underbrace{(2 - \beta ||G^\star|| \cdot ||G||)}_{>0} I \preceq (I + L). \tag{$*3$}$$

Hence the matrix $I + L$ has an inverse.

We take an arbitrary $e_0 \in \mathrm{im}[GG^\star]$. Then $e_0 \in \mathrm{im}[I - L]$, so it must exist some $w_0 \in \mathbb{R}^{m(N+1)}$, such that

$$e_0 = (I - L) w_0 \text{ and } w_0 = (I + L) w_1 \tag{$*4$}$$

for some $w_1 \in \mathbb{R}^{m(N+1)}$.

As next, since $L^\star = L$, for any $p \in \mathbb{N}$ it holds:

$$\begin{aligned}\sum_{k=0}^{p} ||e_k||^2 &= \sum_{k=0}^{p} \langle L^k e_0, L^k, e_0\rangle_Q = \sum_{k=0}^{p} \langle e_0, L^{2k} e_0\rangle_Q = \sum_{k=0}^{p} \langle e_0, L^{2k}(I-L^2) w_1\rangle_Q \\ &= \langle e_0, \left[\left(\sum_{k=0}^{p} L^{2k}\right)(I-L^2)\right] w_1\rangle_Q = \langle e_0, \left(I - L^{2(p+1)}\right) w_1\rangle \\ &\leq ||e_0||_Q ||w_1||_Q \left(1 + ||L||^{2(p+1)}\right) \leq 2||e_0||_Q ||w_1||_Q.\end{aligned} \tag{$*5$}$$

This yields the convergence of the series, if we let $p \to \infty$: $(||e_k||^2)_k$ is a null sequence, what proofs

$$\lim_{k\to\infty} e_k = 0. \tag{$*6$}$$

Now, let $e_0 \in \mathbb{R}^{m(N+1)}$ be arbitrary. Since $\mathbb{R}^{m(N+1)} = \ker[I-L] \oplus \mathrm{im}[I-L]$, we can write e_0 as

$$e_0 = e_{\ker} + e_{\mathrm{im}}, \tag{$*7$}$$

where $e_{\ker} \in \ker[I-L]$ and $e_{\mathrm{im}} \in \mathrm{im}[I-L]$ are uniquely defined.
Hence

$$e_k = L^k e_{\mathrm{im}} + e_{\ker}, \tag{$*8$}$$

and as $L^k e_{\mathrm{im}}$ converges to 0 for $k \to \infty$. This shows (2.40), as

$$\lim_{k\to\infty} e_k = e_{\ker} = P_{\ker[GG^\star]} e_0 = P_{\ker[G^\star]} e_0. \tag{$*9$}$$

■

An interesting observation is, that via this algorithms got input signal u_∞ is the unique solution of the (minimum norm) optimization problem

$$u_\infty = \arg\min_{u\in\mathbb{R}^{l(N+1)}} \{||u-u_0||_R^2 : \text{ subject to } r = Gu + d\}. \tag{2.41}$$

One consequence is that the choice of u_0 has more significance than the simple intuition. A good choice will influence convergence rates beneficially. More generally, the limit u_∞ is the closest input to u_0 in the norm, and since choosing $u_0 = 0$ leads to minimum energy solution u_∞ [1] p. 168.

Choice of the Weighting Matrices

With weighting matrices Q and R we can impact the behavior of the algorithm. For example, with the choice of the matrix Q we can weight some elements of inputs and outputs signals, depending on their importance in measuring accuracies and conver-

gence rates, or accent the relative importance of different time intervals in measuring tracking accuracy and required convergence rates. For rapidly convergence in the initial parts of the time interval, we can set $Q(t) = \varepsilon^{2t}\tilde{Q}$ with some time independent $\tilde{Q}$ and $\varepsilon \in (0, 1)$.
The choice of $R(\cdot)$ may arise out of the real need to converge to a minimum input energy solution. Again, using the ε-weighed $R(t) = \varepsilon^{2t}\tilde{R}$ with some fixed $\tilde{R}$, we can accent the input signals at some times $0 \leq t_1 \leq t \leq t_2 \leq N$. This can be used if we want to limit the control action at some time intervals, or if we need to reflect the physical units used.

Example 7 *For the system* (2.6) *we choose* $N = 50$, $\beta = 0.0026$ *and* $R = I$, $Q = I$. *The reference tracking is illustrated in Figure 2.5. This time we have indeed the perfect following for the signal* r.

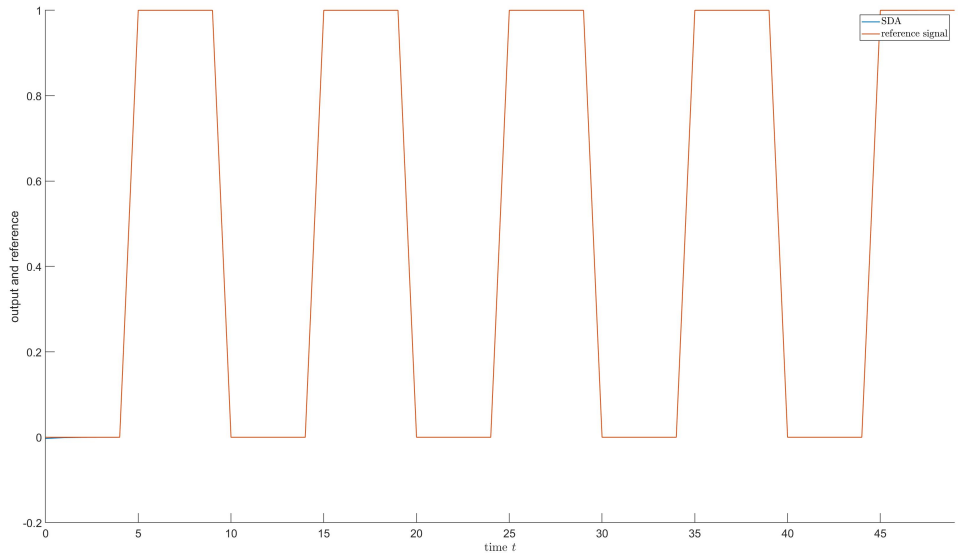

Figure 2.5: Reference signal tracking with SDA for the system (2.6)

By taking a closer look, we can find out, that the matching is not perfect at the beginning. Indeed, with MATLAB calculated error $e_\infty = 0.0029$[1].
The convergence in the first time steps was too slow. Let us choose TODO: add weighted matrices (it does not look better?)

2.4.2 Suppression of eigenvalues

In the previous algorithms we assumed the learning matrix K to be constant for all iterations. We can presume another controller structure and set variable K_k for $k \geq 0$.

[1] Of course, we did not calculate the infinity error with MATLAB. Here the last calculated error value is meant.

The simplest modification way is to set an iteration varying gain β for each step: $K_k = \beta_k K_0$, $k \geq 0$ (K_0 is here a fixed matrix).
Let us consider the eigenvalues of the matrix $GG^\star$ and choose the gains β_k, $k \geq 0$ according to them.

Theorem 8 *Assume, that the matrix $GG^\star$ has q non-zero ordered eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_q$. Set $\beta_k = \frac{1}{\lambda_k}$ for $k = 0, 1, \ldots, q$ and consider the update law*

$$\begin{aligned} u_{k+1} &= u_k + \beta_k G^\star e_k, \\ e_{k+1} &= (I - \beta_{k+1} GG^\star) e_k = \prod_{l=0}^{k+1} (I - \beta_l GG^\star) e_0, \ k \geq 0, \\ u_0 &\in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.42}$$

Then the error sequence $(e_k)_{k \geq 0}$ converges in a finite number of iterations.

Proof:
Set $\tilde{N} := m(N+1) - 1$. With spectral theorem, there exists a basis of the eigenvectors $\{v_0, v_1, \ldots, v_{\tilde{N}}\}$ with corresponding eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_q, 0, \ldots 0$. Then we can write e_0 as

$$e_0 = \sum_{p=0}^{\tilde{N}} \gamma_p v_p \tag{$*1$}$$

for some uniquely defined $\gamma_p \in \mathbb{R}$, $p = 0, 1, \ldots, \tilde{N}$.
For $k \in \mathbb{N}$ it follows

$$e_k = \left(\prod_{j=0}^{k} (I - \beta_j GG^\star) \right) e_0 = \sum_{p=0}^{\tilde{N}} \gamma_p \left(\prod_{j=0}^{k} (I - \beta_j \lambda_p I) \right) v_p. \tag{$*2$}$$

For the first iteration the error becomes

$$e_1 = \sum_{p=0}^{\tilde{N}} \gamma_p (I - \beta_1 \lambda_p I) v_p = \sum_{p=1}^{\tilde{N}} \gamma_p (I - \beta_1 \lambda_p I) v_p. \tag{$*3$}$$

The component v_0 is eliminated from the error. Assuming, that for $k \geq 1$ the first k components are eliminated, we get

$$e_{k+1} = \sum_{p=0}^{\tilde{N}} \gamma_p \left(\prod_{j=0}^{k} (I - \beta_j \lambda_p I) \right) v_p = \sum_{p=k+1}^{\tilde{N}} \gamma_p \left(\prod_{j=0}^{k} (I - \beta_j \lambda_p I) \right), \tag{$*4$}$$

and hence the first $k+1$ components $v_0, v_1, \ldots, v_k$ are eliminated. By induction, the iteration process terminates after, at most, $m(N+1) - q - 1$ iterations, as all non-zero

eigenvalues have been covered and hence all corresponding eigenvectors are eliminated. ■

Although conceptually interesting this algorithm is not well applicable for the real problems. One can see, that the small non-zero eigenvalues of $GG^\star$ will lead to very large values of β_k and hence we get extremely large transient variations in error norm. Also computing the eigenvalues can be numerically questionable. Moreover, to achieve the monotonic convergence we need to consider only the eigenvalues $\lambda > \frac{1}{2}\lambda_{\max}(H)$. If our model is inaccurate, and some eigenvalues are uncertain, it has directly impact on the algorithm result.
That all makes this algorithm not solid. Still, we can use the idea, to apply the eigenvalues of $GG^\star$ – but not directly. We can try to "pick" the compatible eigenvalues.

Algorithm 5 *Choose a finite number $N_p + 1$ of points $p_0, p_1, \ldots$ spread over the half-open interval $(\frac{1}{2}\lambda_{\max}(H), \lambda_{\max}(H)]$. The Gradient Algorithm with Suppression of Eigenvalues (SoE Algorithm) is defined via choosing of the iteration-depending control law $K_k = \beta_k G^\star e_k$, where*

$$\begin{aligned} \beta_k &= \frac{1}{p_k} \text{ for } k = 0, 1, \ldots, N_p, \\ \beta_k &= \beta \text{ for } k > N_p, \end{aligned} \tag{2.43}$$

with $\beta \in (0, \frac{2}{\sigma^2_{\max}})$. The iteration law is given via

$$\begin{aligned} u_{k+1} &= u_k + \beta_k G^\star e_k, \ k \geq 0 \\ e_k &= \left[\prod_{l=0}^{k}(I - \beta_l GG^\star)\right] e_0, \ k = 0, 1, \ldots, N_p, \\ e_k &= (I - \beta GG^\star)^{k-N_p} e_{N_p}, \ k > N_p, \\ e_0 &= r - Gu_0 - d, \ u_0 \in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.44}$$

This algorithm has the same convergence properties as Algorithm **??**, but potentially better convergence rates due to the eigenvalue suppression. Intuitively, the approach will increase the convergence speed in the first $N_p + 1$ iterations, if N_p is large enough for good approximation of the interval $(\frac{1}{2}\lambda_{\max}(H), \lambda_{\max}(H)]$.

Example 9 *We apply the SoE Algorithm on the system* (2.6)*. The error evolution we get is illustrated in Figure 2.6. The new algorithm needs only 148 iteration, while classical SDA terminates by 382 iterations. We also get a much better improvement $||e_0/e_\infty||$, while e_∞ is meant to be the error in the last iteration.*
In fact, for $Q = I_m$, $R = I_l$, for 36 of 51 eigenvalues λ of $GG^\star$ holds

$$\lambda > \frac{1}{2}\lambda_{\max}(H) = 37.9048\,. \tag{2.45}$$

This explained the better performance of the algorithm, and better convergence rate.

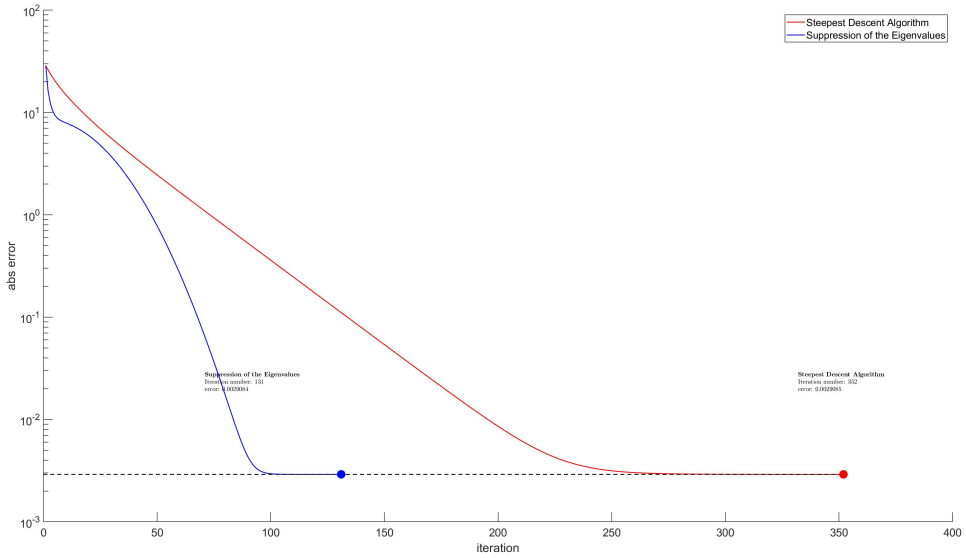

Figure 2.6: Error evolution for SDA and SoE for system (2.6)

2.5 Feedback Design

With the previous algorithms we can achieve a better tracking for repetitively executed processes. But they do not accent the special features of the model. For example, we can not manage the non-minimum-phase zeros, which can impact the convergence rate [1]. If we consider the plant structure and construct a feedback controller, and then convert it into a steepest descent-like algorithm, we possibly can achieve better performance and robustness properties.

Let us denote with $G(z)$ the transfer matrix in variable z of the system (A, B, C, D), and with G the supervector matrix.

More precisely, we consider a forward path compensator $K_c(z)$ in a unity negative feedback system for (1.1). The design criteria for $K_c(z)$ include the closed-loop stability and the ability to track, albeit approximately, the reference signal r. With this compensator, depending on the model, we can remedy the plant properties such as oscillation, loop interaction, or the effects of non-minimum-phase zeros.

We denote the complementary sensitivity of the resulting closed loop with $T(z)$ and its sensitivity with $S(z)$:

$$T(z) = (I + G(z)K_c(z))^{-1}G(z)K_c(z) \text{ and } S(z) = I - T(z) = (I + G(z)K_c(z))^{-1}, \tag{2.46}$$

and define the controller

$$K_0(z) = K_c(z)S(z). \tag{2.47}$$

Then the compensated Steepest Descent Algorithm is given as follows.

Algorithm 6 *Write the transfer functions in supervector description as K_0, K_c, T and S. The compensated Steepest Descent Algorithm is is characterized by choosing $K = \beta K_0$, $\beta \in \mathbb{R}$. The iterative law is given via*

$$\begin{aligned} u_{k+1} &= u_k + \beta K_0 T^* e_k, \\ e_{k+1} &= (I - \beta T T^*) e_k, \; k \geq 0, \\ u_0 &\in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.48}$$

Monotonic convergence to

$$e_\infty = \lim_{k \to \infty} e_k = P_{\ker[T^*]} e_0, \tag{2.49}$$

is guaranteed if

$$0 < \beta < \frac{2}{\lambda_{\max}(TT^*)}.$$

T^* stays here for conjugate transpose of the matrix T.

Proof:
The proof is identical to this in Algorithm 4 if consider the system

$$\tilde{y} = T\tilde{u} + \tilde{d}. \tag{$*1$}$$

■

The effect of the operator TT^* can be seen by considering the closed-loop relation

$$y = Tr. \tag{2.50}$$

We get a perfect tracking, if $T = I$, which is not achievable by feedback control. Still, we can assume a good tracking if $||T|| \approx 1$.
Therefore if K_c provides excellent feedback control of G, then rapid convergence could be attained with a choice of gain $\beta \approx 1$ [1].

Chapter 3

Application

The listed algorithms have their theoretical benefits, but they might not be good for real issues. The numeric obstacles require the trade-offs between robustness, stability, and calculation time. The computer memory has limited capacity; systems discretization adds the plant errors. We considered the examples do not have a physical reference system. Now, we try to apply the algorithms on the more "real" plant and see the problems that occur.

Example 10 *We consider a dynamical system, which describes a flexible satellite. As known from [2] the continuous-time transfer function has the form*

$$G(s) = \frac{.036(s + 25.28)}{s^2(s^2 + .0396s + 1)}. \tag{3.1}$$

As input we choose the torque on first mass, and as output is set the position θ_2.
Since we are working with discrete time systems, we discretize it for sample time $T_s = 10^{-3}$ and get the state space description

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cccc|c} 1.000 & 0.005 & 0.000 & 0.000 & 1.047 \cdot 10^{-13} \\ 0 & 1.000 & 0.001 & 0.000 & 8.333 \cdot 10^{-11} \\ 0 & 0 & 1.000 & 0.001 & 2.5 \cdot 10^{-7} \\ 0 & 0 & -0.001 & 1.000 & 5 \cdot 10^{-4} \\ \hline 0.362 & 0.072 & 0 & 0 & 0 \end{array}\right). \tag{3.2}$$

In this example, the matrix D is zero, what makes our algorithms not applicable. Moreover, the D-part's neglecting is typical: the system output must mostly image the system state and not the known input we send.
To see more precisely, the input update rule has the form

$$y(t) = Cx(t) = C(Ax(t) + Bu(t)) \text{ for } t = 0, 1, 2, \ldots, N. \tag{3.3}$$

With other words, the input $u(t)$ begins to impact the output by first iteration, if the

matrix CB is not a zero-matrix. That means, that the *relative degree* of the system equals 1. We define it mathematically precise:

Definition 11 *The relative degree of the system (A, B, C, D) is 0 if $D \neq 0$. For $D = 0$ it is the smallest integer k for which $CA^{k-1}B \neq 0$.*

Let us denote it with k^*. Then the new system we consider can be written as

$$x(t+1) = Ax(t) + Bu(t), \tag{3.4}$$

$$y(t) = Cx(t) + CA^{k^*}Bu(t), \text{ for } t = k^*, k^*+1, \dots N. \tag{3.5}$$

We need to put the first iterations away and define a new initial condition $x(k^*)$. The start input condition is then $u(k^*)$.
For new system all the requirements we considered are satisfied. However, we do not get a better solution for $t = 0, 1, \dots, k^\star - 1$.
Example 0(continued...)
For our system the relative degree is 1, as

$$CB = 6.038 \cdot 10^{-12} \neq 0. \tag{3.6}$$

We calculate the solution with LQR. Result is illustrated in Figure 3.1.

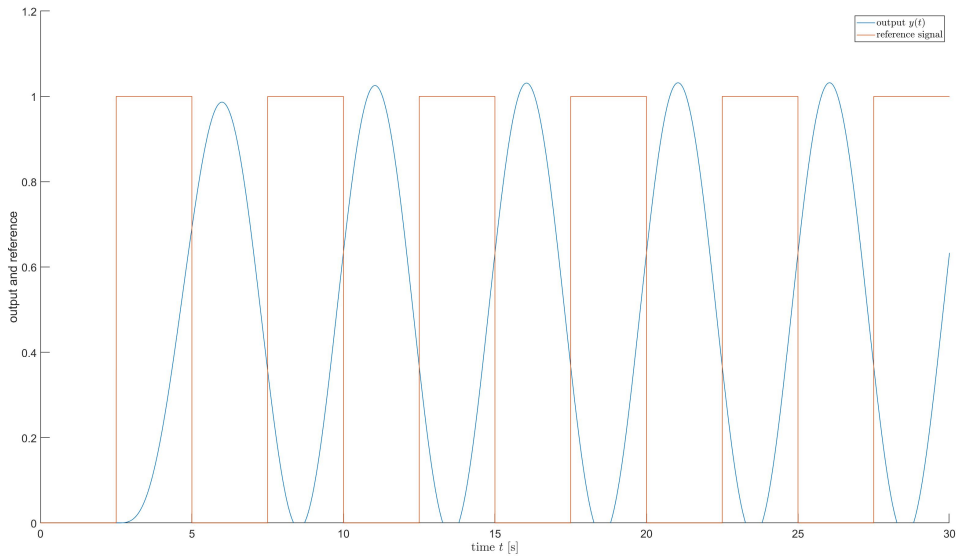

Figure 3.1: LQR Solution for the system (3.2)

It looks like something we can improve. However, for $Ts = 10^{-3}s$ we get $N = 10^3$ time steps even for one iteration second. The matrix G has 10^9 elements and can not easily be handled.
In example **??** the Inverse Model Algorithm is not applicable even for $N = 40$. That means, we need a trade off between the time horizon and robustness. However, for

real problems the sample times are mostly small, and hence even for one second of the simulation we get huge number for time horizon. Already for SISO system the matrix G will have N^2 elements. Our algorithms become unusable for casual applications since they need vast masses of memory.

We solve this problem by application the algorithms iterative, for smaller number of time steps. We only need to adapt the initial input condition and the reference signal: we split them in N_{max}/N parts, and take each iteration the next one. Here $N_{\max}$ is the number of time steps in total, and N is the number of time steps we put into the algorithms .

The choice of N is than another design decision to be made. It must be not too large, since a used algorithm can diverge, if it is too small, we probably will need more iterations in total.

Example 1 *Let us use the Steepest descent algorithm, and choose put each time* $N = 100$ *elements into the algorithm. The result is depicted in the Figure 3.2 We can see, that steepest descent algorithm converges very fast to tracking value, and even for 20 iterations we get perfect tracking.*

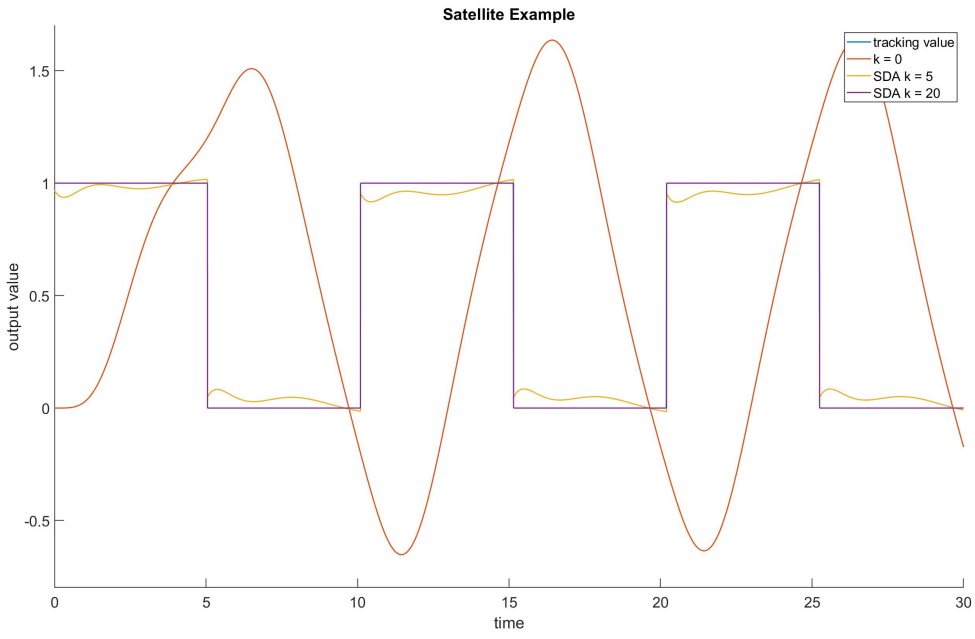

Figure 3.2: LQR Solution for the system (3.2)

Chapter 4

Robustness of the ILC Algorithms

If our model is not precisely known a chosen ILC algorithm might not provide the desired result. Therefore it is useful to consider how robust the algorithms are. In this thesis, we are going to consider the parametric uncertainty. That means, we do not know the true value of a real parameter (or the parameters) but know some range in which this parameter should lie.

We can normalize such uncertainties without loss of generality, as any uncertainty $\delta \in [a, b] \subset \mathbb{R}$ can be expressed as

$$\delta = \delta_0 + w\hat{\delta} \tag{4.1}$$

with $\hat{\delta} \in [-1, 1]$, nominal value $\delta_0 \in \mathbb{R}$ and weight $w \in \mathbb{R}$.

In order to guarantee the convergence of the algorithms despite such parametric uncertainties it is required that (2.26) stays stable for all possible value of parameters.

Let us consider, how the uncertainty looks in the lifted system like and deliberate over the method to solve the robustness issue.

<Einfuehrung>

4.1 Uncertain Lifted System

Let us consider a system (1.1) with uncertainty $\hat{\Delta}$ from the set

$$\hat{\mathbf{\Delta}} = \left\{ \begin{pmatrix} \delta_1 I_{p_1} & & \\ & \ddots & \\ & & \delta_\tau I_{p_\tau} \end{pmatrix} \;\middle|\; \delta_l \in [-1,\ 1], p_l \in \mathbb{N} \text{ for } l = 1, 2, \ldots, \tau. \right\}. \tag{4.2}$$

$\tau \in \mathbb{N}$ is the number of the uncertain parameter.

Each uncertain system $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \left(\hat{\Delta}\right)$ we can rewrite as a feedback interconnection

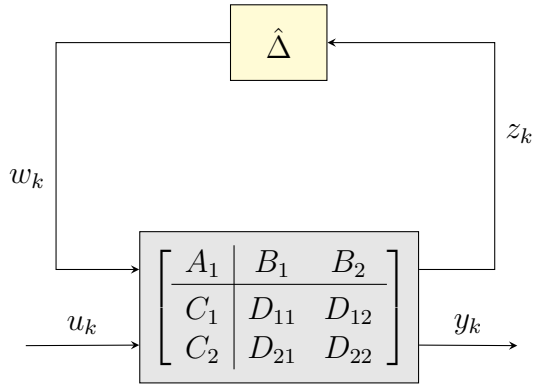

Figure 4.1: Uncertain system as feedback interconnection of LFR and the uncertainty $\hat{\Delta}$

of the system $\left[\begin{array}{c|cc} A_1 & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$ and $\hat{\Delta}$. Then the original system can be find as star product (or linear fractional transformation, LFT)

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \hat{\Delta} \star \left[\begin{array}{c|cc} A_1 & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \tag{4.3}$$

We illustrate it in the Figure 4.1. We speak here about the "pulling out" the uncertainty, or the *linear fractional representation* (LFR).

With LFR we rewrite the system (1.1) as

$$\begin{aligned} x(t+1) &= A_1 x(t) + B_1 u(t) + B_2 w(t), \\ y(t) &= C_1 x(t) + D_{11} u(t) + D_{12} w(t), \\ z(t) &= C_2 x(t) + D_{21} u(t) + D_{22} w(t), \\ w(t) &= \hat{\Delta} z(t), \\ x(0) &\in \mathbb{R}^n, t = 0, 1, 2, \ldots, N, \end{aligned} \tag{4.4}$$

with error $e(\cdot) = r(\cdot) - y(\cdot)$.

The corresponding lifted system has the form

$$y = G_1 u + G_2 w + d_1, \tag{4.5}$$

$$z = G_3 u + G_4 w + d_2, \tag{4.6}$$

$$e = r - y, \tag{4.7}$$

$$w = \Delta z, \tag{4.8}$$

with the matrices

$$G_1 = \begin{pmatrix} D_{11} & 0 & \cdots & 0 & 0 & 0 \\ C_1B_1 & D_{11} & \cdots & 0 & 0 & 0 \\ C_1A_1B_1 & C_1B_1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ C_1A_1^{N-2}B_1 & C_1A_1^{N-3}B_1 & \dots & C_1B_1 & D_{11} & 0 \\ C_1A_1^{N-1}B_1 & C_1A_1^{N-2}B_1 & \dots & C_1A_1B_1 & C_1B_1 & D_{11} \end{pmatrix}, \tag{4.9}$$

$$G_2 = \begin{pmatrix} D_{12} & 0 & \cdots & 0 & 0 & 0 \\ C_1B_2 & D_{12} & \cdots & 0 & 0 & 0 \\ C_1A_1B_2 & C_1B_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ C_1A_1^{N-2}B_2 & C_1A_1^{N-3}B_2 & \dots & C_1B_2 & D_{12} & 0 \\ C_1A_1^{N-1}B_2 & C_1A_1^{N-2}B_2 & \dots & C_1A_1B_2 & C_1B_2 & D_{12} \end{pmatrix}, \tag{4.10}$$

$$G_3 = \begin{pmatrix} D_{21} & 0 & \cdots & 0 & 0 & 0 \\ C_2B_1 & D_{21} & \cdots & 0 & 0 & 0 \\ C_2A_1B_1 & C_2B_1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ C_2A_1^{N-2}B_1 & C_2A_1^{N-3}B_1 & \dots & C_2B_1 & D_{21} & 0 \\ C_2A_1^{N-1}B_1 & C_2A_1^{N-2}B_1 & \dots & C_2A_1B_1 & C_2B_1 & D_{21} \end{pmatrix}, \tag{4.11}$$

$$G_4 = \begin{pmatrix} D_{22} & 0 & \cdots & 0 & 0 & 0 \\ C_2B_2 & D_{22} & \cdots & 0 & 0 & 0 \\ C_2A_1B_2 & C_2B_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ C_2A_1^{N-2}B_2 & C_2A_1^{N-3}B_2 & \dots & C_2B_2 & D_{22} & 0 \\ C_2A_1^{N-1}B_2 & C_2A_1^{N-2}B_2 & \dots & C_2A_1B_2 & C_2B_2 & D_{22} \end{pmatrix}, \tag{4.12}$$

and vectors

$$d_1 = \begin{pmatrix} C_1 x_0 \\ C_1 A_1 x_0 \\ C_1 A_1^2 x_0 \\ \vdots \\ C_1 A_1^N x_0 \end{pmatrix}, \tag{4.13}$$

$$d_2 = \begin{pmatrix} C_2 x_0 \\ C_2 A_1 x_0 \\ C_2 A_1^2 x_0 \\ \vdots \\ C_2 A_1^N x_0 \end{pmatrix}. \tag{4.14}$$

Δ is here a block diagonal matrix

$$\Delta = \mathrm{diag}(\underbrace{\hat{\Delta}, \hat{\Delta}, \ldots, \hat{\Delta}}_{(N+1) \text{ times}}) \tag{4.15}$$

and we define

$$\mathbf{\Delta} = \{\mathrm{diag}(\hat{\Delta}, \hat{\Delta}, \ldots, \hat{\Delta}) |\ \hat{\Delta} \in \hat{\mathbf{\Delta}}\}. \tag{4.16}$$

Using LFT, we get

$$y = G(\Delta)u + d(\Delta) := \left[G_1 + G_2\Delta(I - G_4\Delta)^{-1}G_3\right] u + \left[G_2\Delta(I - G_4\Delta)^{-1}d_2 + d_1\right], \tag{4.17}$$

while the inverse of the matrix $(I - G_4\Delta)$ is assumed to exist. It is always the case if the norm of the matrix G_4 is (strictly) bounded by 1 [2].

With ILC algorithms we get the input update rule

$$u_{k+1} = u_k + Ke_k, \tag{4.18}$$

and hence

$$e_{k+1} = L(\Delta)e_k = (I - G(\Delta)K)e_k. \tag{4.19}$$

If we pull out the uncertainty from $L(\Delta)$, we get

$$\left(\begin{array}{c|c} e_{k+1} \\ \hline z_k \end{array}\right) = \left(\begin{array}{c|c} \mathcal{A} & \mathcal{B} \\ \hline \mathcal{C} & \mathcal{D} \end{array}\right) \left(\begin{array}{c|c} e_k \\ \hline w_k \end{array}\right) := \left(\begin{array}{c|c} I - G_1 & -G_2 \\ \hline G_3 K & G_4 \end{array}\right) \left(\begin{array}{c|c} e_k \\ \hline w_k \end{array}\right), \tag{4.20}$$
$$w_k = \Delta z_k,$$

with the uncertain system defined through the uncertain matrix

$$L(\Delta) = \Delta \star \left(\begin{array}{c|c} \mathcal{A} & \mathcal{B} \\ \hline \mathcal{C} & \mathcal{D} \end{array}\right) = \mathcal{A} + \mathcal{B}\Delta(I - \mathcal{D}\Delta)^{-1}\mathcal{C}. \tag{4.21}$$

We are interested in ensuring of the stability for the system (4.19).
The most obviously way is to ensure $||L(\Delta)|| < 1$ for all uncertainties $\Delta \in \mathbf{\Delta}$. However, this criteria might be too restrictive. We illustrate it on the following example.

Example 12 *Let us consider the system*

$$e_{k+1} = L(\alpha)e_k = \begin{pmatrix} .1 & \alpha \\ 0 & .1 \end{pmatrix} e_k \tag{4.22}$$

with uncertain parameter $\alpha \in \mathbb{R}$. Then the norm $||L(\alpha)||$ is proportional to $|\alpha|$.
For example, with the spectral norm, we can ensure stability only for $|\alpha| < 1$.
On the other hand, one can directly see, that the system is stable for all $\alpha \in \mathbb{R}$, since the eigenvalues of the matrix equal .1.

This example motivates to devise another method for ensuring the stability of the uncertain systems. Much more elegant results we can reach by considering stability using Lyapunov techniques.

4.2 Full-Block S-Procedure

First, let us consider the uncertainty Δ without taking into account its structure. Using the Lyapunov LMI we formulate the following Theorem.

Theorem 13 *Suppose that there exists some positive definite matrix X satisfying*

$$\begin{pmatrix} I \\ L(\Delta) \end{pmatrix}^T \begin{pmatrix} -X & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} I \\ L(\Delta) \end{pmatrix} = L(\Delta)XL(\Delta) - X \preceq 0 \text{ for all } \Delta \in \mathbf{\Delta}. \tag{4.23}$$

Then the discrete dynamical system (4.19) is stable for all $\Delta \in \mathbf{\Delta}$ and for all $e_0 \in \mathbb{R}^{m(N+1)}$. It is asymptotically stable if the inequality is strict.

Proof:
First, let us prove the Theorem for the strict LMI. Then we can find some $\gamma \in (0,1)$, such that

$$\begin{pmatrix} I \\ L(\Delta) \end{pmatrix} \begin{pmatrix} -\gamma X & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} I \\ L(\Delta) \end{pmatrix} = L(\Delta)^T X L(\Delta) - \gamma X \prec 0 \text{ for all } \Delta \in \mathbf{\Delta}. \qquad (*1)$$

Moreover, since $X \succ 0$, there exist some $\alpha, \beta > 0$, such that

$$\alpha I \prec X \prec \beta I. \qquad (*2)$$

Set $\eta_k := e_k^T X e_k$. Then we have

$$\eta_{k+1} = e_{k+1}^T X e_{k+1} = e_k^T L(\Delta)^T X L(\Delta) e_k \leq \gamma e_k X e_k = \gamma \eta_k \text{ for all } k \geq 0. \qquad (*3)$$

From $(*2)$ we get

$$\alpha ||e_k||^2 \leq \eta_k \leq \beta ||e_k||^2 \text{ for all } k \geq 0, \qquad (*4)$$

and with $(*3)$ it follows

$$||e_k||^2 \leq \frac{1}{\alpha}\eta_k \leq \frac{\gamma^k}{\alpha}\eta(0) \leq \frac{\beta}{\alpha}\gamma^k ||e_0||^2. \qquad (*5)$$

Since $\gamma \in (0,\ 1)$, $||e_k||^2 \to 0$ for $k \to \infty$, and taking the square root yields the claim.
For not strict case, we set in $(*3)$ and $(*5)$ $\gamma = 1$, and replace in $(*1)$ "$\prec$" with "$\preceq$". Then $||e_k||^2$ (and hence $||e_k||$) is bounded over all $k \geq 0$ and hence the system (4.19) is stable.

■

In this Theorem, we still have the uncertainty in our LMI problem. To get rid of it, we fix the matrix $X \succ 0$ and try to find a multiplier $P = P^T$, which fulfills

$$\begin{pmatrix} I \\ \Delta \end{pmatrix}^T P \begin{pmatrix} I \\ \Delta \end{pmatrix} \succeq 0, \qquad (4.24)$$

and, additionally,

$$\begin{pmatrix} I & 0 \\ \mathcal{A} & \mathcal{B} \end{pmatrix}^T \begin{pmatrix} -X & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} I & 0 \\ \mathcal{A} & \mathcal{B} \end{pmatrix} + \begin{pmatrix} \mathcal{C} & \mathcal{D} \\ 0 & I \end{pmatrix}^T P \begin{pmatrix} \mathcal{C} & \mathcal{D} \\ 0 & I \end{pmatrix} \preceq 0. \qquad (4.25)$$

If we can find such a matrix P, the stability of the system (4.19) can be proven.
We formulate it as a Theorem.

Theorem 14 *Let $\mathbb{P}$ be a subset of the symmetric matrices, and $X \succ 0$ be fixed. Then the system* (4.19) *is stable if there exists a matrix $P = P^T \in \mathbb{P}$, such that* (4.24) *and* (4.25) *are fulfilled. If the inequality* (4.25) *is strict, system becomes asymptotic stable.*

Proof:
Let $P = P^T \in \mathbb{P}$ be some matrix, which fulfills the relations (4.24) and (4.25).
We multiply (4.25) with

$$H = \begin{pmatrix} I \\ \Delta(I - \mathcal{D}\Delta)^{-1}C \end{pmatrix} \tag{$*1$}$$

right and its transpose left. Then the first term results in

$$H^T \begin{pmatrix} I & 0 \\ \mathcal{A} & \mathcal{B} \end{pmatrix}^T \begin{pmatrix} -X & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} I & 0 \\ \mathcal{A} & \mathcal{B} \end{pmatrix} H = L(\Delta)^T X L(\Delta) - X. \tag{$*2$}$$

Hence, if we can prove that the second term multiplied with H right and its transpose left is positive semi-definite, this matrix P will ensure the monotonic convergence of the algorithms. Multiplying H yields to

$$H^T \begin{pmatrix} \mathcal{C} & \mathcal{D} \\ 0 & I \end{pmatrix}^T P \begin{pmatrix} \mathcal{C} & \mathcal{D} \\ 0 & I \end{pmatrix} H = (I - \Delta\mathcal{D})^{-T} \underbrace{\begin{pmatrix} I \\ \Delta \end{pmatrix}^T P \begin{pmatrix} I \\ \Delta \end{pmatrix}}_{\succeq 0} (I - \Delta\mathcal{D}) \succeq 0,$$

what proofs the first statement. If the inequality (4.25) is strict, the LMI (4.23) is also strict and we achieve the asymptotic stability. ■

So, the robustness issue can be reduced to a certain LMI over a multiplier set $\mathbb{P}$.
The choosing of the set $\mathbb{P}$ is here the cornerstone. It must be large enough, to be able to find such multiplier P, if the system is stable, but also adequate small, such that the searching action is not too costly.
Let us begin with a single uncertain parameter δ. Then the uncertainty $\Delta = \delta I$ with $\delta \in [-1,\ 1]$. A possible choice for multiplier set is

$$\mathbb{P} = \{ \begin{pmatrix} S & R \\ R^T & -S \end{pmatrix} \mid S \succeq 0,\ R + R^T = 0 \}. \tag{4.26}$$

There is

$$\begin{pmatrix} I \\ \delta I \end{pmatrix}^T \begin{pmatrix} S & R \\ R^T & -S \end{pmatrix} \begin{pmatrix} I \\ \delta I \end{pmatrix} = \underbrace{(1 - \delta^2)}_{\geq 0} S + \delta \underbrace{(R + R^T)}_{=0} \succeq 0, \tag{4.27}$$

and hence the requirements on the set $\mathbb{P}$ for the Theorem (14) are fulfilled.

For $\tau \in \mathbb{N}$ uncertain real parameters the uncertainty has the form

$$\Delta = \operatorname{diag}(\underbrace{\hat{\Delta}_\tau, \hat{\Delta}_\tau, \ldots, \hat{\Delta}_\tau}_{(N+1) \text{ times}}) \tag{4.28}$$

with

$$\hat{\Delta}_\tau = \begin{pmatrix} \delta_1 I_{p_1} & & & \\ & \delta_2 I_{p_2} & & \\ & & \ddots & \\ & & & \delta_\tau I_{p_\tau} \end{pmatrix}. \tag{4.29}$$

We choose a permutation matrix $\mathcal{P}$, such that

$$\Delta_{\text{perm}} = \mathcal{P}\Delta = \begin{pmatrix} \delta_1 I_{(p_1 \cdot N)} & & & \\ & \delta_2 I_{(p_2 \cdot N)} & & \\ & & \ddots & \\ & & & \delta_\tau I_{(p_\tau \cdot N)} \end{pmatrix}. \tag{4.30}$$

If we choose the multiplier set

$$\mathbb{P}_\tau = \left\{ \begin{pmatrix} I_{\tau(N+1)} \\ \mathcal{P} \end{pmatrix}^T \begin{pmatrix} S & R \\ R^T & -S \end{pmatrix} \begin{pmatrix} I_{\tau(N+1)} \\ \mathcal{P} \end{pmatrix} \middle| \begin{array}{c} S = \operatorname{diag}(S_1, S_2, \ldots, S_\tau) \succeq 0, \\ R = \operatorname{diag}(R_1, R_2, \ldots, R_\tau), R + R^T = 0 \end{array} \right\}, \tag{4.31}$$

then for $P_\tau \in \mathbb{P}_\tau$, the following relation is satisfied

$$\begin{pmatrix} I \\ \Delta \end{pmatrix}^T P_\tau \begin{pmatrix} I \\ \Delta \end{pmatrix} = \begin{pmatrix} I \\ \Delta_{\text{perm}} \end{pmatrix}^T \begin{pmatrix} S & R \\ R^T & -S \end{pmatrix} \begin{pmatrix} I \\ \Delta_{\text{perm}} \end{pmatrix} \tag{4.32}$$

$$= \begin{pmatrix} (1-\delta_1^2)S_1 & & & \\ & (1-\delta_2^2)S_2 & & \\ & & \ddots & \\ & & & (1-\delta_\tau^2)S_\tau \end{pmatrix}. \tag{4.33}$$

Go back to the example 12. Let $\alpha \in [0, 2]$ be an uncertainty with the nomimal value $\alpha_0 = 1$. With linear fractional transformation the system

$$e_{k+1} = \begin{pmatrix} .1 & \alpha \\ 0 & .1 \end{pmatrix} e_k \tag{4.34}$$

results in

$$\begin{pmatrix} e_{k+1} \\ z_k \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & .1 & 1 \\ 0 & 0 & .1 \end{pmatrix} \begin{pmatrix} e_k \\ w \end{pmatrix}. \tag{4.35}$$

With the choice

$$X = \mathrm{diag}(2, .5) \tag{4.36}$$

we can compute with MATLAB the solution

$$P = \mathrm{diag}(4.61, -4.61), \tag{4.37}$$

which renders the eigenvalues of the left side term in the relation (4.25) to

$$\lambda_1 = -2.35,\ \lambda_2 = -1.21 \text{ and } \lambda_3 = -.488. \tag{4.38}$$

Hence, according to the Theorem 14, the system is stable for all $\alpha \in [0, 2]$.

Example 15 *In the chapter 2 we said, that the choice of β might impact the robustness properties of the Inverse Model Algorithms. Let us illustrate it on the system* (2.3)*, but with uncertain parameter $a \in [\underline{a},\ \overline{a}]$ and nominal value $a_0 = 4$:*

$$A = \begin{pmatrix} 2 & 1 \\ a & 3 \end{pmatrix} \tag{4.39}$$

For $N = 23$ and $\beta = .1$ our system still stays robust for $\overline{a} = -\underline{a} = 2$. If we set $\beta = .4$, we can not ensure the robustness using Theorem 14 even for $\overline{a} = -\underline{a} = .1$.
However, the ability to find the adequate matrices X and P depends on the time horizon N. For $N = 20$, we get a much more robust system, for which the stability can be proven for $\overline{a} = -\underline{a} = 5$.

Hier können auch Schlußfolgerungen präsentiert und ein Ausblick gegeben werden.

Bibliography

[1] David H. Owens. *Iterative Learning Control*. Springer, 2016.

[2] Carsten W. Scherer. *Lecture Notes Robust Control*.

[3] Carsten W. Scherer. *Linear Control Theory*.

[4] Carsten W. Scherer. *Lineare Algebra and Analytische Geometrie*.

.1 Beweise

.1.1 Beweis 1

$$1 + 1 = 2 \tag{40}$$

Eidesstattliche Erklärung

Ich erkläre, dass ich meine Bachelor-Arbeit [*Titel der Arbeit*] selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe und dass ich alle Stellen, die ich wörtlich oder sinngemäß aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner Prüfungsbehörde vorgelegen.
Stuttgart, den XX.XX.2013

(Name des Kandidaten)

Appendix A

Zusammenfassung und Ausblick