Lehrstuhl für Mathematische Systemtheorie
Universität Stuttgart

BACHELORARBEIT

im Bachelorstudiengang
Mathematik

TITEL DER ARBEIT

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

Recap

Topics to be considered:

(a) Vector norms

(b) GP

(c) direct sum

Add Monotonic convergence!
By theorem **??**, the reight inverse of $G = G(A, B, C, D)$ exists if and only if the matrix D has full row rank.
The choice of β is the only decision to make in this algorithm. For β close to 0 or 2, we get slower convergence and better robustness. For $\beta = 1$ we get convergence in one iteration, but as discussed above, this choice might be not robust at all. evtl kommen Robustheit Beispiele.
If there exists (only) a left inverse of G, we can also calculate a better choice of input u. But in this algorithm we do not have guarantee of zero-convergence.

Chapter 2

Basics

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.

2.1 Supervector description

More formally, let us consider a discrete time iteration

$$\begin{aligned} x(t+1) &= Ax(t) + \tilde{B}r(t) + Bu(t) && t = 0,\, 1,\, 2,\, \ldots,\, N-1 \\ y(t) &= Cx(t) + \tilde{D}r(t) + Du(t), && t = 0,\, 1,\, 2,\, \ldots,\, N \\ e(t) &= r(t) - y(t), && t = 0,\, 1,\, 2,\, \ldots,\, N \\ x(0) &= x_0, \end{aligned} \tag{2.1}$$

over a finite time horizon ($N \in \mathbb{N}$).

Here $A \in \mathbb{R}^{n\times n}$, $\tilde{B} \in \mathbb{R}^{n\times m}$, $B \in \mathbb{R}^{n\times l}$, $C \in \mathbb{R}^{m\times n}$, $\tilde{D} \in \mathbb{R}^{n\times m}$, and $D \in \mathbb{R}^{m\times l}$ are real matrices, $r(t) \in \mathbb{R}^m$ is a reference signal at time t. The error between the reference signal and the measurement output $y(\cdot)$ is denoted with $e(\cdot)$ and represents the performance channel of the plant.

Let us assume, that we already have a stabilizing controller for the system (2.1). After the controlled system run through we get some input and output sequences $\{u(0),\, u(1),\, u(2)\, \ldots,\, u(N)\}$, $\{y(0),\, y(1),\, y(2),\, \ldots,\, y(N)\}$, respectively.

The relation between them can be written in linear matrix form

$$y = Gu + d, \tag{2.2}$$

if we use the *supervector* description

$$u = \begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(N) \end{bmatrix} \in \mathbb{R}^{l(N+1)},\ y \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} \in \mathbb{R}^{m(N+1)}. \tag{2.3}$$

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

$$G = G(A, B, C, D) = \\ = \begin{bmatrix} D & 0 & \cdots & 0 & 0 & 0 \\ CB & D & \cdots & 0 & 0 & 0 \\ CAB & CB & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ CA^{N-2}B & CA^{N-3}B & \dots & CB & D & 0 \\ CA^{N-1}B & CA^{N-2}B & \dots & CAB & CB & D \end{bmatrix} \in \mathbb{R}^{m(N+1)\times l(N+1)}, \tag{2.4}$$

while the vector d depends on the initial condition x_0:

$$d = d(C, A, x_0) = \begin{bmatrix} Cx_0 \\ CAx_0 \\ CA^2x_0 \\ \vdots \\ CA^Nx_0 \end{bmatrix} \in \mathbb{R}^{m(N+1)}. \tag{2.5}$$

In the same way we get the supervector description for the sequences $\{e(0),\ e(1),\ e(2),\ \dots,\ e(N)\}$ and $\{r(0),\ r(1),\ r(2),\ \dots,\ r(N)\}$ and denote them with e, r, respectively.

The error e is given just as

$$e = r - y. \tag{2.6}$$

2.2 Unit Memory Algorithm

As said above, we are interested in the processes, which are executed repetitively. Let $k = 0,\ 1,\ 2,\ \dots$ be the number of completed iterations.

Our goal is to find a "perfect" signal u_∞, such that $r = Gu_\infty + d$. Let us assume linear dependence of u_{k+1} on the previous signals u_k, e_k and y_k. Denoting v_k as the control

input, we get a plant

$$\begin{aligned} u_{k+1} &= u_k + v_k, \\ y_k &= Gu_k + d, \\ e_k &= r - y_k, \\ u_0 &\in \mathbb{R}^{l(N+1)},\ k = 0,\ 1,\ 2, \dots . \end{aligned} \tag{2.7}$$

One can see that this is again a discrete-time system, with time increment k: u defines here internal state of the plant, y is measurement output and v is control input, e is a controlled variable and represents the performance channel. The signals r and d can be seen as generalized disturbance. If we can stabilize the system, the error will converge to zero with increasing k – our problem is reduced to finding a stabilizing controller. Assuming feedback control, we conclude *Unit Memory Algorithm*:

Algorithm 1 *The Unit Memory Algorithm is given via iteration law defined by choosing*

$$v_k = K(r - Gu_k - d) = Ke_k,\ k \geq 0, \tag{2.8}$$

as control input in the plant (2.7). $K \in \mathbb{R}^{l(N+1)\times m(N+1)}$ *is called* ***learning matrix***.

For the closed-loop system we get

$$\begin{aligned} u_{k+1} &= u_k + Ke_k,\ k \geq 0 \\ e_k &= (I - GK)e_{k-1} = (I - GK)^{k-1}e_0 =: L^{k-1}e_0,\ k \geq 1, \\ e_0 &= r - Gu_0 - d,\ u_0 \in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.9}$$

This leads to plain but essential result:

Theorem 1 *(a) The closed-loop 2.9 is stable if and only if the matrix L satisfies*

$$\rho(L) < 1, \tag{2.10}$$

(b) If $\rho(L) = 1$, then $\lim_{k\to\infty} L^k = \hat{L}$ for some $\hat{L} \in \mathbb{R}^{m(N+1)\times m(N+1)}$ and the sequence $(e_k)_{k\geq 0}$ is monotonically decreasing and converges to $e_\infty = \hat{L}e_0$, if and only if for L all Jordan Blocks of the eigenvalue 1 have dimension 1. If in additional $e_0 \in \ker(\hat{L})$, then monotone convergence to 0 is guaranteed.

Proof:

Hier wird es proof geben

■

We will see, that it is often not possible, to achieve $\rho(L) < 1$. Still, the convergence to 0 is attainable. Since e_0 depends on u_0 and r, the design problem here is to find adequate

target r – usually that means, that it must be feasible – and a good start input value u_0. Furthermore, the matrix K is still a design issue in Unit Memory Algorithm, since it allows us to regulate the spectrum of L.

2.2.1 Inverse Model Algorithm

Let us come back to the relation (2.2). If the matrix G is invertible, the choice $K = G^{-1}$ leads to

$$e_{k+1} = (I - GG^{-1})e_k = 0 \text{ for all } k \geq 0,$$

and the resulting input signal for "perfect tracking" is given by

$$u_\infty = G^{-1}(r - d).$$

This result does not look very credible and may be explained by the following two reasons. First, if the model is not accurate, the resulting output y can differ from the true value, significantly. Second, computing the inverse numerically might lead to errors.
A more plausible design is to choose $K = \beta G^{-1}$ for some real constant β, which allows a trade-off between the robustness and the convergence rate. Also, we can ease the conditions by replacing the inverse matrix G^{-1} with a right or left inverse.

Algorithm 2 *Let the matrix D in* (2.1) *have full row rank. Then the matrix $G = G(A, B, C, D)$ has a right inverse G_R:*

$$GG_R = I.$$

The Right Inverse Model Algorithm is characterized by choosing $K = \beta G_R$, where β is a real scalar, called "learning gain". The feedback interconnection (2.9) *has the form*

$$\begin{aligned} u_{k+1} &= u_k + \beta G_R e_k,\ k \geq 0 \\ e_k &= (1-\beta)e_{k-1} = (1-\beta)^{k-1}e_0,\ k \geq 1, \\ e_0 &= r - Gu_0 - d,\ u_0 \in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.11}$$

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:
TODO: full row rank (D) $\Leftrightarrow$ full row rank (G)

The convergence to zero is guaranteed by 1. ■

The choice of β is the only decision to make in this algorithm. For β close to 0 or 2, we get slower convergence and better robustness. For $\beta = 1$ we get convergence in one iteration, but as discussed above, this choice might be not robust at all.

If there exists (only) a left inverse of G, we can also calculate a better choice of input u. But in this algorithm, we do not have a guarantee of zero-convergence.

Algorithm 3 *Let the matrix D in* (2.1) *have full column rank. Then the matrix $G = G(A, B, C, D)$ has a left inverse G_L:*

$$G_L G = I.$$

The Left Inverse Model Algorithm is characterized by choosing $K = \beta G_L$, where β is a real scalar, called "learning gain". The feedback interconnection (2.9) *has the form*

$$\begin{aligned} u_{k+1} &= u_k + \beta G_L e_k,\ k \geq 0 \\ e_k &= (I - \beta G G_L) e_{k-1} = (I + \left[(1-\beta)^{k-1} - 1\right] G G_L) e_0,\ k \geq 1, \\ e_0 &= r - G u_0 - d,\ u_0 \in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.12}$$

Monotonic convergence to

$$e_\infty = \lim_{k \to \infty} e_k = (I - G G_L) e_0 \tag{2.13}$$

for $k \to \infty$ is guaranteed if and only if

$$0 < \beta < 2.$$

Proof:
TODO: full column rank (D) $\Leftrightarrow$ full column rank (G)

Using the relation $(GG_L)^2 = GG_LGG_L = GG_L$ the error evolution formula (3.8) can be proven by using mathematical induction:

for $k = 0$ it follows:

$$e_1 = (I - \beta G G_L) e_0 = (I + G G_L - G G_L - \beta G G_L) e_0 = (I + [(1-\beta) + 1] G G_L) e_0.$$

For $k \in \mathbb{N}$ it follows:

$$\begin{aligned} e_{k+1} &= (I - \beta G G_L) e_k = (I - \beta G G_L)(I + \left[(1-\beta)^{k-1} - 1\right] G G_L) e_0 = \\ &= \left(I - \beta G G_L + (1-\beta)^{k-1} G G_L - G G_L - \beta G G_L \left[(1-\beta)^{k-1} - 1\right] G G_L\right) e_0 = \\ &= \left(I - (\beta - (1-\beta)^{k-1} + \beta \left[(1-\beta)^{k-1} - 1\right]) G G_L\right) e_0 \\ &= \left(I - \left(\beta - (1-\beta)^{k-1} + 1 + \beta(1-\beta)^{k-1} - \beta\right) G G_L\right) e_0 \\ &= (I + \left[(1-\beta)^k - 1\right] G G_L) e_0 \end{aligned}$$

To prove (3.9) recall that

$$\mathbb{R}^{m(N+1)} = \mathrm{im}(G) \oplus \ker(G_L).$$

Hence e_0 can be written as

$$e_0 = G w_0 + v_0,$$

where $v_0 = (I - G G_L) e_0 \in \ker(G_L)$ is uniquely defined and $w_0 = G_L e_0 \in \mathbb{R}^{l(N+1)}$. It follows:

$$e_{k+1} = (1 + \left[(1-\beta)^k - 1\right] G G_L)^k e_0 = (1-\beta)^k G G_L e_0 + v_0.$$

Then for $\beta \in (0, 2)$ the error e_k converges to v_0 for $k \to \infty$. ■

The choice of β is identical to this in Algorithm 5.

2.2.2 Gradient Algorithms

One of the benefits of the previous algorithms is monotonic convergence. Still, the computation of the inverse matrix can be a hard issue for example due to numerical inaccuracy.

Let us contemplate this problem from another side, and try to guarantee monotonous decreasing error norm by direct calculation.

The goal is to achieve

$$||e_{k+1}|| < ||e_k|| \text{ for all } k > 0, \tag{2.14}$$

in some norm $||\cdot||$ in $\mathbb{R}^{m(N+1)}$.

By adding a zero to e_{k+1} we get

$$\begin{aligned} ||e_{k+1}||^2 &= ||e_k + (e_{k+1} - e_k)||^2 = \\ &= ||e_k||^2 + 2\langle e_k, e_{k+1} - e_k\rangle + ||e_{k+1} - e_k||^2 \\ &= ||e_k||^2 + 2\langle e_k, (I - GK)e_k - e_k\rangle + ||(I - GK)e_k - e_k||^2 \\ &= ||e_k||^2 - 2\langle e_k, GKe_k\rangle + ||GKe_k||^2 \\ &= ||e_k||^2 - 2\langle G^*e_k, Ke_k\rangle + ||GKe_k||^2 \\ &< ||e_k||^2, \text{ if } ||GKe_k||^2 < 2\langle G^*e_k, Ke_k\rangle. \end{aligned} \tag{2.15}$$

By using of the 2nd norm, the last inequality is equivalent to

$$\frac{1}{2}e_k K^*G^*GKe_k < e_k^*GKe_k, \tag{2.16}$$

which is satisfied if

$$\frac{1}{2}K^*G^*GK - GK \prec 0. \tag{2.17}$$

If we set $K = \beta G^*$ for some scalar $\beta > 0$, the last matrix inequality becomes

$$\frac{1}{2}\beta(GG^*)^2 - GG^* \prec 0. \tag{2.18}$$

This is fulfilled, if we choose $\beta \in (0, \beta^*)$, with $\beta^* = \frac{2}{\sigma_{\max}(G)^2}$, where $\sigma_{\max}(G)$ denotes the maximum singular value of the matrix G, since then

$$\frac{1}{2}\beta(GG^*)^2 - G^*G \prec \frac{1}{\sigma_{\max}(G)^2}(G^*G)^2 - GG^* \preceq 0. \tag{2.19}$$

The last inequality if satisfied, because

$$\begin{aligned} &\lambda_{\max}\left(\frac{1}{\sigma_{\max}(G)^2}(GG^*)^2\right) = \frac{1}{\sigma_{\max}(G)^2}\lambda_{\max}(GG^*GG^*) \\ &= \frac{1}{\sigma_{\max}(G)^2}\lambda_{\max}(GG^*)^2 = \frac{\lambda_{\max}(GG^*)^2}{\lambda_{\max}(GG^*)} = \lambda_{\max}(GG^*). \end{aligned} \tag{2.20}$$

This choice of K determines the *Steepest Descent Algorithm*

Algorithm 4 *The Steepest Descent Algorithm is characterized by choosing $K = \beta G^*$, where $\beta > 0$ is a real scalar gain. Feedback interconnection* (2.9) *has the form*

$$\begin{aligned} u_{k+1} &= u_k + \beta G^* e_k, \ k \geq 0 \\ e_k &= (I - \beta GG^*)e_{k-1} = (I - \beta GG^*)^{k-1}e_0, \ k \geq 0, \\ e_0 &= r - Gu_0 - d, \ u_0 \in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.21}$$

Monotonic convergence to

$$e_\infty = \lim_{k\to\infty} e_k = P_{\ker[G^*]} e_0, \tag{2.22}$$

is guaranteed if

$$0 < \beta < \frac{2}{\sigma_{\max}(G)^2}.$$

Here $P_{\ker[G^*]}$ *denotes the positive orthogonal projection operator onto* $\ker[G^*]$*, In particular, the zero convergence is attainable, if* $e_0 \in \overline{\mathrm{im}[GG^*]}$.

Proof:

■

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

$$u_\infty = \arg \min_{u\in\mathbb{R}^l(N+1)} \{||u - u_0||^2 : \text{ subject to } r = Gu + d\}. \tag{2.23}$$

One consequence is that the choice of u_0 has more significance that the simple intuition that 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_∞.

Wird weiter nicht benoetingt, soll ich das stehen lassen (+proof)?

2.2.3 Suppression of eigenvalues

In the considered algorithms we assumed the learning matrix K to be constant for all iterations. If we set variable K_k for $k \geq 1$, we can achieve better convergence rates.

The simplest modification way is to set inconstant gain β for each iteration: $K_k = \beta_k K$, $k \geq 1$.

Let us consider the eigenvalues of the matrix GG^* and choose the gains β_k, $k \geq 0$ according to them.

Theorem 2 *Assume, that the matrix* GG^* *has* q *non-zero ordered eigenvalues* $\lambda_1, \lambda_2, \ldots, \lambda_q$. *Set* $\beta_k = \frac{1}{\lambda_k}$ *for* $k = 1, 2, \ldots, q$ *and consider the control input law:*

$$v_k = \beta_k G^* e_k. \tag{2.24}$$

Then iteration law is given via the closed loop interconnection

$$\begin{aligned} u_{k+1} &= u_k + \beta G^* e_k,\ k \geq 0 \\ e_k &= (I - \beta_k G G^*) e_{k-1} = (I - \beta_{k-1} G G^*)^{k-1} e_0,\ k \geq 0, \\ e_0 &= r - G u_0 - d,\ u_0 \in \mathbb{R}^{l(N+1)}, \end{aligned} \tag{2.25}$$

and the error sequence $(e_k)_{k\geq 1}$ converges to the limit e_∞ in a finite number of iterations.

Proof:
content... ■

Although conceptually interesting this algorithm is not well applicable for the real problems. One can see, that the small non-zero eigenvalues of GG^* will lead to very large values of β_k and hence we get extremely large transient variations in error norm. The model errors can make this problem intolerable, when elimination of eigenvector components will not be achieved in any iteration and/ or may be re-introduced in later iterations.
Also computing the eigenvalues can be numerically questionable (while $\sigma_{\max}(G)$ can be approximated via frequency domain **?**). To achieve the monotonic convergence, we need to consider only the eigenvalues $\lambda > \frac{1}{\sigma_{\max}(G)^2}$.
Because of that we introduce the following algorithm, where we try to "pick" the compatible eigenvalues of GG^*.

Algorithm 5 *Choose a finite number N_p of points $p_1, p_2, \ldots$ spread over the half-open interval $(\frac{1}{2}\sigma_{\max}(G)^2, \sigma_{\max}(G)^2]$. The Gradient Algorithm with Suppression of Eigenvalues is defined via choosing of the iteration-depending control law $v_k = \beta_k G^* e_k$, where*

$$\begin{aligned} \beta_k &= \frac{1}{p_k} \text{ for } k = 1, 2, \ldots, N_p, \\ \beta_k &= \beta \text{ for } k > N_p, \end{aligned} \tag{2.26}$$

with $\beta \in (0, \frac{2}{\sigma_{\max}(G)^2})$. The closed loop interconnection (2.9) has the form:

$$\begin{aligned} u_{k+1} &= u_k + \beta_k G^* e_k,\ k \geq 0 \\ e_k &= \left[\prod_{j=1}^{k} (I - \beta_j G G^*)\right] e_0,\ k = 1, 2, \ldots, N_p, \\ e_k &= (I - \beta G G^*)^{k-N_p} e_{N_p},\ k > N_p, \\ e_0 &= r - G u_0 - d,\ u_0 \in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{2.27}$$

This algorithm has the same convergence properties as Algorithm 4, but potentially better convergence rates due to the eigenvalue suppression. Intuitively, the approach will increase the convergence speed in the first N_p iterations , if N_p is large enough for good covering of the interval $(\frac{1}{2}\sigma_{\max}(G)^2, \sigma_{\max}(G)^2]$.

Chapter 3

Unit Memory Algorithm

We consider the mathematical model of the form given in (2.2).
For the reference signal r let $y_\infty = r$ be perfect tracking requirement, with input-output relation $y_\infty = Gu_\infty + d$.
The aim is to improve the actual tracking – that means, to reduce the norm of the error e until desired accuracy ε:

$$\begin{aligned} &\text{for any } \varepsilon > 0 \text{ find some } y \in \mathbb{R}^{m(N+1)} \text{ such that } ||e|| = ||r - y|| < \varepsilon \\ &\text{for some norm } ||\cdot|| \text{ in } \mathbb{R}^{m(N+1)}. \end{aligned} \tag{3.1}$$

Not every algorithm can ensure the convergence of the error to 0. In this case, we try just to reduce the error to some $e_\infty \in \mathbb{R}^{m(N+1)}$, such that $||e_\infty|| < ||e||$.

Algorithm 6 *A **Unit Memory Algorithm** is given by the update low*

$$u_{k+1} = u_k + K_0 e_k, \qquad \textit{(The Input Update Rule)} \tag{3.2}$$

$$\textit{with } y_k = Gu_k + d,\ k \geq 0 \qquad \textit{(Plant Dynamics).} \tag{3.3}$$

The initial choice of input signal to be given as $u_0 \in \mathbb{R}^{m(N+1)}$ (e.g. the signal we got from controller).
The error evolution is given by the equation

$$e_{k+1} = (I - GK_0)e_k = (I - GK_0)^k e_0 =: L^k e_0,\ k \geq 0, \tag{3.4}$$

with initial error $e_0 = r - Gu_0 - d$.

With formula (3.4) the convergence requirement (3.1) can be achieved by ensuring $||L|| < 1$. Indeed, it is enough to calculate the maximal absolute value of the matrix L. This illustrates the following theorem.
For the m-input m-output systems convergency test can be reduced to the estimation of the maximal absolute value of the matrix D:

Theorem 3 *For an $m-$input $m-$output system (A, B, C, D) let L be given as in (3.4). Then the spectrum of L is precisely the set of the eigenvalues of D and hence $\rho(L) = \rho(D)$. Especially, the sequence from theorem 1 converges to 0 if and only if $\rho(D) < 1$.*

Theorem 4 *Norm equivalency? If in the recall chapter: with Q and R.*

3.1 Inverse Model Algorithms

For the given relation $y = Gu + d$ the perfect tracking is achieved if there exists some u_∞ with $r = Gu_\infty + d$. If G is invertable, u_∞ can be calculated as $u_\infty = G^{-1}(r - d)$. This motivates to choose K_0 in (3.2) as $K_0 = G^{-1}$.
The idea of the inverse model can be also used if the matrix G is singular, since it can still have right or left inverse.

3.1.1 Right Inverse Model Algorithm

Algorithm 7 *Let the matrix G have a right inverse G_R:*

$$GG_R = I.$$

Set $K_0 = \beta G_R$, where β is a real scalar "learning gain". Then the input update rule is given by

$$u_{k+1} = u_k + G_R e_k,\ k \geq 0, \tag{3.5}$$

and the error evolution satisfies

$$e_{k+1} = (1 - \beta)e_k = (1 - \beta)^k e_0,\ k \geq 0. \tag{3.6}$$

In particular, the error e_k converges to zero for any initial error e_0 if and only if

$$0 < \beta < 2.$$

The existence of a right inverse is guaranteed by theorem **??** if and only if $\mathrm{im}(D) = m$ – with other words, the matrix D must have full row rank.
The convergence rate of the algorithm 5 depends upon the chosen gain β. For $\beta = 1$ we get convergence in one iteration – since fast, this choice is not robust at all. Even little perturbations in the model or numerical inaccuracy can lead to false result. -¿ Ref chapter ????.

3.1.2 Left Inverse Model Algorithm

If the matrix G still have a left inverse, one can also improve the tracking – but this time it is not always possible to achieve convergence to zero.

Algorithm 8 *Let the matrix G have a left inverse G_L:*

$$G_L G = I.$$

Set $K_0 = \beta G_L$, where β is a real scalar "learning gain". Then the input update rule is given by

$$u_{k+1} = u_k + G_L e_k,\ k \geq 0, \tag{3.7}$$

and the error evolution satisfies

$$e_{k+1} = (I - \beta G G_L)e_k = (I + \left[(1-\beta)^k - 1\right] G G_L)e_0,\ k \geq 0. \tag{3.8}$$

Monotonic convergence to

$$e_\infty = \lim_{k\to\infty} e_k = (I - G G_L)e_0 \tag{3.9}$$

is guaranteed if and only if

$$0 < \beta < 2.$$

Proof:
Using the relation $(GG_L)^2 = GG_LGG_L = GG_L$ the error evolution formula (3.8) be proven by using mathematical induction:
for $k = 0$ it follows:

$$e_1 = (I - \beta G G_L)e_0 = (I + GG_L - GG_L - \beta GG_L)e_0 = (I + \left[(1-\beta) + 1\right] GGL).$$

For $k \in \mathbb{N}$ it follows:

$$\begin{aligned}
e_{k+1} &= (I - \beta G G_l)e_k = (I - \beta G G_L)(I + \left[(1-\beta)^{k-1} - 1\right] G G_L)e_0 = \\
&= \left(I - \beta G G_L + (1-\beta)^{k-1} G G_L - G G_L - \beta G G_L \left[(1-\beta)^{k-1} - 1\right] G G_L\right) e_0 = \\
&= \left(I - (\beta - (1-\beta)^{k-1} + \beta \left[(1-\beta)^{k-1} - 1\right]) G G_L\right) e_0 \\
&= \left(I - \left(\beta - (1-\beta)^{k-1} + 1 + \beta(1-\beta)^{k-1} - \beta\right) G G_L\right) e_0 \\
&= (I + \left[(1-\beta)^k - 1\right] G G_L)e_0
\end{aligned}$$

To prove (3.9) recall that

$$\mathbb{R}^{m(N+1)} = \mathrm{im}(G) \oplus \ker(G_L).$$

Since e_0 can be written as

$$e_0 = Gw_0 + v_0,$$

where $v_0 = (I - GG_L)e_0 \in \ker(G_L)$ is uniquely defined and $w_0 = G_L e_0 \in \mathbb{R}^{l(N+1)}$, it follows

$$e_{k+1} = (1 + \left[(1-\beta)^k - 1\right] GG_L)e_0 = (1-\beta)^k GG_L e_0 + v_0.$$

Then the error e_k converges to v_0 for $k \to \infty$. ■

As it has been seen above, the monotonic convergence to zero will only occur if $v_0 = 0$ – this befalls if $e_0 \in \mathrm{im}(G)$. As $e_0 = r - Gu_0 - d$, the last condition becomes $r - d \in \mathrm{im}(G)$. This simply means that there exist a control input $u \in \mathbb{R}^{l(N+1)}$, such that the desired value r is tracked exactly. Clearly, if there does not exist such an input, the algorithm just converges to the smallest possible error.

3.1.3 Choice of learning gain β

The benefit of the algorithms 5 and 6 is intuitive understanding and need to adapt only one control parameter β.
In simply words, for β close to 0 or 2, we get slow convergence with good robustness properties. For β close to unity we have rapidly convergence, but with reduced degree of robustness. For $\beta = 1$ we get convergence in one iteration.

$$A^T P A - P + Q = 0 \tag{3.10}$$

Chapter 4

Questions

P without uncertainty:

$$\left[\begin{array}{c} u_{k+1} \\ \hline y_k \\ \hline e_k \end{array}\right] = \left[\begin{array}{c|c|cc} I & I & 0 & 0 \\ \hline G & 0 & 0 & I \\ \hline -G & 0 & I & -I \end{array}\right] \left[\begin{array}{c} u_k \\ \hline v_k \\ \hline r \\ d \end{array}\right] \tag{4.1}$$

$$v_k = Ke_k = K(r - Gu_k - d) = -KGu_k + Kr - Kd \tag{4.2}$$

$P \star K$ with uncertainty:

$$\left[\begin{array}{c} u_{k+1} \\ \hline e_k \\ \hline z_k \end{array}\right] = \left[\begin{array}{c|c|c} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ \hline C_2 & D_{21} & D_{22} \end{array}\right] \left[\begin{array}{c} u_k \\ \hline r \\ d \\ \hline w_k \end{array}\right], \tag{4.3}$$

$$w_k = \Delta(\delta) z_k. \tag{4.4}$$

Matrix seen by uncertainty: $M = D_{22}$.

Exercise: $(I - D_{22}\Delta)^{-1}$ exists and is **stable** if there exists a matrix $P = P^T$, such that

$$\begin{bmatrix} I \\ \Delta(\delta) \end{bmatrix}^T P \begin{bmatrix} I \\ \Delta(\delta) \end{bmatrix} \succeq 0, \text{ and } \begin{bmatrix} D_{22} \\ I \end{bmatrix}^T P \begin{bmatrix} D_{22} \\ I \end{bmatrix} \prec 0. \tag{*}$$

For real parametric uncertainty $\Delta(\delta) = \delta I$, $\delta \in [-1, 1]$,

$$\mathbb{P} = \{P = \begin{bmatrix} Q & S \\ S^T & -Q \end{bmatrix} : Q \preceq 0, S = -S^T\}, . \tag{4.5}$$

For $\Delta = \begin{bmatrix} \delta_1 I & & & \\ & \delta_2 I & & \\ & & \ddots & \\ & & & \delta_\tau I \end{bmatrix}$:

$$\mathbb{P} = \left\{ P = \left[\begin{array}{c|ccc} P_0 & & & \\ \hline & P_1 & & \\ & & P_2 & \\ & & & \ddots \\ & & & & P_\tau \end{array}\right] \middle| \; P_i \succeq 0 \right\} \tag{4.6}$$

4.1 From last session:

$$e_{k+1} = L(\delta) e_k. \tag{4.7}$$

Need:

$$||L(\delta)|| < 1 \Leftrightarrow L(\delta)^T L(\delta) < I \Leftrightarrow \begin{bmatrix} I \\ L(\delta) \end{bmatrix}^T \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I \\ L(\delta) \end{bmatrix} < 0 \; \forall \delta \in [-1,\, 1]. \tag{$\circ$}$$

$$\begin{bmatrix} e_{k+1} \\ z_k \end{bmatrix} = \left[\begin{array}{c|c} \mathcal{A} & \mathcal{B} \\ \hline \mathcal{C} & \mathcal{D} \end{array}\right] \begin{bmatrix} e_k \\ w_k \end{bmatrix}$$

$$L(\delta) = \Delta(\delta) \star \left[\begin{array}{c|c} \mathcal{A} & \mathcal{B} \\ \hline \mathcal{C} & \mathcal{D} \end{array}\right]$$

$$(\circ) \Leftarrow \exists P = P^T : \begin{bmatrix} I & 0 \\ \mathcal{A} & \mathcal{B} \end{bmatrix}^T \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathcal{A} & \mathcal{B} \end{bmatrix} + \begin{bmatrix} \mathcal{C} & \mathcal{D} \\ 0 & I \end{bmatrix}^T P \begin{bmatrix} \mathcal{C} & \mathcal{D} \\ 0 & I \end{bmatrix} \prec 0, \tag{4.8}$$

$$\begin{bmatrix} I \\ \Delta(\delta) \end{bmatrix}^T P \begin{bmatrix} I \\ \Delta(\delta) \end{bmatrix} \succeq 0 \; \forall \delta \in [-1,\, 1].$$

For $P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}, P_1 = P_1^T, P_2 = P_2^T$

$$(4.8) \Leftrightarrow \begin{bmatrix} -I + \mathcal{A}^T\mathcal{A} & \mathcal{A}^T\mathcal{B} \\ \mathcal{B}^T\mathcal{A} & \mathcal{B}^T\mathcal{B} \end{bmatrix} + \begin{bmatrix} \mathcal{C}^T P_1 \mathcal{C} & \mathcal{C}^T P_1 \mathcal{D} \\ \mathcal{D}^T P_1 \mathcal{C} & \mathcal{D}^T P_1 \mathcal{D} + P_2 \end{bmatrix} \prec 0, \begin{bmatrix} I \\ \Delta(\delta) \end{bmatrix}^T P \begin{bmatrix} I \\ \Delta(\delta) \end{bmatrix} \succeq 0 \; \forall \delta \in [-1,\, 1].$$

Discrete Lyapunov inequation:

$$L^T X L - X \prec 0 \Leftrightarrow \begin{bmatrix} I \\ L(\delta) \end{bmatrix}^T \begin{bmatrix} -X & 0 \\ 0 & X \end{bmatrix} \begin{bmatrix} I \\ L(\delta) \end{bmatrix} \prec 0, \text{ for } X \succ 0,$$

$$\Leftrightarrow \mathcal{A}^T X \mathcal{A} + \mathcal{C}^T (I - \mathcal{D}\Delta(\delta))^{-T} \mathcal{B}^T \Delta(\delta) X \mathcal{A} + \mathcal{A}^T X \mathcal{B} \Delta(\delta) (I - \mathcal{D}\Delta)^{-1} \mathcal{C} +$$

$$+ \mathcal{C}^T (I - \mathcal{D}\Delta(\delta))^{-T} \mathcal{B}^T \Delta(\delta) X \mathcal{B} \Delta(\delta) (I - \mathcal{D}\Delta)^{-1} \mathcal{C} + \mathcal{C}^T (I - \mathcal{D}) - X \prec 0$$

$$L(\delta)^T L(\delta) - I = \mathcal{A}^T \mathcal{A} + \mathcal{C}^T (I - \mathcal{D}\Delta(\delta))^{-T} \mathcal{B}^T \Delta(\delta) \mathcal{A} + \mathcal{A}^T \mathcal{B} \Delta(\delta) (I - \mathcal{D}\Delta)^{-1} \mathcal{C} +$$

$$+ \mathcal{C}^T (I - \mathcal{D}\Delta(\delta))^{-T} \mathcal{B}^T \Delta(\delta) \mathcal{B} \Delta(\delta) (I - \mathcal{D}\Delta)^{-1} \mathcal{C} + \mathcal{C}^T (I - \mathcal{D}) - I \overset{!}{\prec} 0.$$

Chapter 5

Robustness of the ILC Algorithms

Undependend on the algorithm we use, we have in common the closed loop

Chapter 6

Zusammenfassung und Ausblick

Hier können auch Schlußfolgerungen präsentiert und ein Ausblick gegeben werden.

Bibliography

David H. Owens. Iterative Learning Control, pp. 110–111, 20016.

.1 Beweise

.1.1 Beweis 1

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

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)