SimTech Chair for Mathematical Systems Theory
University of Stuttgart

Project Thesis

in Bachelor Degree
Simulation Technology

Iterative Learning Control

by Tatiana Strelnikova

Examiner: Prof. Dr. Carsten W. Scherer
Examination Date: 20.06.2021

Lehrstuhl für Mathematische Systemtheorie
Universität Stuttgart

Projektarbeit

im Bachelorstudiengang
Simulation Technology

Iterative Learning Control

Prüfer: Prof. Dr. Carsten W. Scherer

vorgelegt von

Name	Studienfach
Straße	Fachsemester
PLZ + Ort	Matrikelnummer
E-mail	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 3

ILC Algorithms

Imagine, that some action needs to be performed multiple times. For example, a robot manipulator must put objects in a box with high accuracy, while the objects to put are always located in the same place. It takes the same path again and again. If an input sequence for this issue solving already exists, it can be used for the next iteration and delivers the same precision. The other possibility is to "learn" from the previous iteration and try to enhance the exactness.

First, assume the system (2.1) to have stable matrix A. There is no less of generality, if the system is stabilizable. In this case, it is always possible to find such a matrix F, such that in the system

$$\begin{pmatrix} x(t+1) \\ y(t) \end{pmatrix} = \begin{pmatrix} A - BF & B \\ C - DF & D \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} \tag{3.1}$$

the matrix $A - BF$ is stable. The new input is here given by $v := u - BFx$. Depending on controller, there are other ways to transform the system, such a new input is defined, which can be adjusted using ILC algorithms. An example for a system controlled by a controller based on separation principle is presented below.

Example 5 *Consider a stabilizable and detectable system (A, B, C, D) with input u and output y. Extend the input u to $u + \tilde{u}$, where u is controlled with controller based on separation principle $u(t) = -F\hat{x}(t) + Mr(t)$, $t \geq 0$, and $\tilde{u}$ is initially zero. Then the system can be rewritten as*

$$x(t+1) = (A - BF)x(t) + B(F\tilde{x}(t) + \tilde{u}(t) + Mr(t)), \tag{3.2}$$

$$y(t) = (C - DF)x(t) + D(F\tilde{x}(t) + \tilde{u}(t) + Mr(t)), \quad t \geq 0. \tag{3.3}$$

This system is stable. Defining $v(t) := F\tilde{x}(t) + \tilde{u}(t) + Mr(t)$, $t \geq 0$, this system can be *used for ILC algorithms, while the initial input is*

$$v_0 = \begin{pmatrix} Mr(0) \\ F\tilde{x}(1) + Mr(1) \\ F(A-JC)x(1) + Mr(2) \\ \vdots \\ F(A-JC)^{N-2}x(1) + Mr(N) \end{pmatrix}. \tag{3.4}$$

Example 6 *Consider the system (A, B, C, D) with*

$$A = \begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix}, B = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, C = \begin{pmatrix} 0 & 1 \end{pmatrix}, D = 2. \tag{3.5}$$

This system is asymptotically stabilizable and detectable. Use the same notation as in Chapter 2.2, and choose the weights $Q = 3$ and $R = 2$. Then

$$F = \begin{pmatrix} 1.9674 & 1.3042 \end{pmatrix} \text{ and } J = \begin{pmatrix} 1.8705 & 4.7321 \end{pmatrix} \tag{3.6}$$

can be used for the controller based on separation principle.
Then the system is modified to

$$\begin{aligned} x(t+1) &= \begin{pmatrix} 0.0326 & -0.3042 \\ 0.0652 & 0.3916 \end{pmatrix} x(t) + \begin{pmatrix} 1 \\ 2 \end{pmatrix} v(t) \\ y(t) &= \begin{pmatrix} -3.9348 & -1.6084 \end{pmatrix} x(t) + 2v(t), \qquad t = 0, 1, 2, \ldots, N \end{aligned} \tag{3.7}$$

with a new input signal $v(\cdot)$.

3.1 Inverse Model Algorithms

Consider again the equation (2.7). The tracking problem here seems to be straightforward. If the reference signal r is known, a perfect solution can be find by choosing

$$u_\infty = G^{-1}r - d. \tag{3.8}$$

However, this choice does not look greatly credible. For example, if a measurement are disturbed by noise, the output signal becomes

$$y + \varepsilon w = Gu_\infty + d + \varepsilon w = r + \varepsilon w \neq r, \tag{3.9}$$

for some non-zero vector $w \in \mathbb{R}^{m(N+1)}$ and a small $\varepsilon > 0$. Also for uncertain models this method might not be applicable, as it can be difficult to calculate the inverse. This idea can be reformulated in a more robust way, and result in a good and simple applicable algorithm, which is called the *Inverse Model Algorithm.*

Algorithm 7 *Let the matrix D in* (2.1) *be square (i.e. $m = l$) and invertable. 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*

$$\begin{aligned} u_{k+1} &= u_k + \beta G^{-1} e_k, \\ u_0 &\in \mathbb{R}^{m(N+1)}. \end{aligned} \tag{3.10}$$

The error evolution follows

$$\begin{aligned} e_{k+1} &= (1-\beta) e_k,\ k \geq 0, \\ e_0 &= r - Gu_0 - d. \end{aligned} \tag{3.11}$$

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:
Firstly, the matrix $G \in \mathbb{R}^{(N+1)m\times(N+1)m}$ is also square and has the 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 is invertable. Taking the limit over relation (3.11) 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. ■

The choice of β close to 0 or 2 yields slower convergence and better robustness. For $\beta = 1$ the algorithm converges in one iteration, but this choice might be non-robust [**ILC**], .
An example of applying of this algorithm on the system 3.7 is presented below.

Example 8 *In the system* (3.7) *the matrix $D = 2$ is invertible, and IA can be applied. Choose $\beta = 0.1$. Convergence to 0 with tolerance 10^{-8} is achieved after 74 iterations.The result is illustrated in Figure 3.1.*
However, by trying to increase the length of the time horizon N, for example to 50 steps, the system behaviour is wrong (Figure 3.2).
The reason is the condition number of the matrix G, which equals $1.353\,10^{16}$. This shows: even for a triangular matrix with non-zero eigenvalues, the numerically disturbances can lead to an unexpected algorithm behaviour.

The Inverse Model Algorithm sets up very strong requirements: the matrix G should be non-singular, and its condition number must be small enough, such that a an inverse

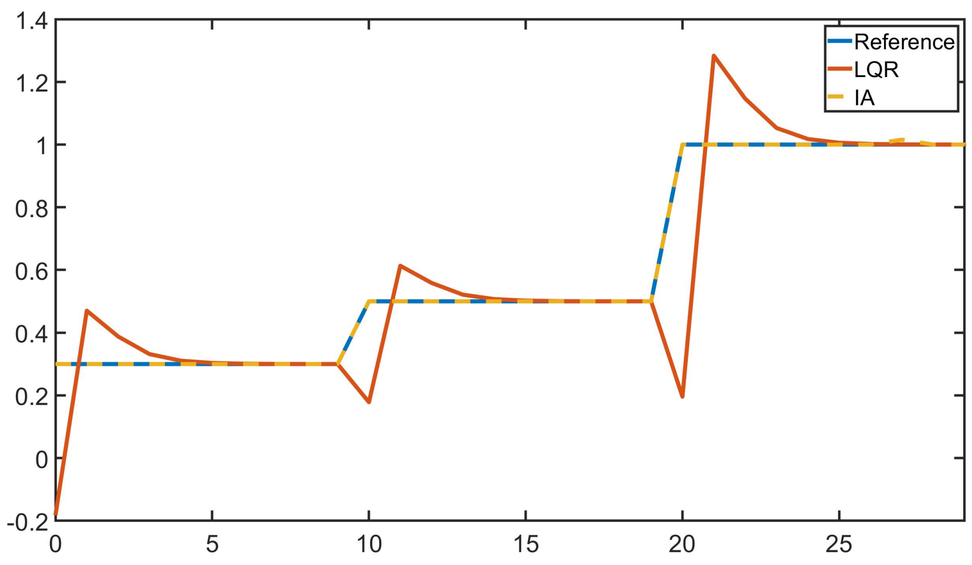

Figure 3.1: Tracking with LQR Controller (without ILC algorithm), and tracking with IA for the system (3.5) for $N = 30$

can be correct computed by numerical methods. That motivates not to use the inverse of the matrix, but a pseudo-inverse. This is a uniquely defined matrix, which can be computed for any matrix, even not for a square one.

Definition 9 *For a matrix $G \in \mathbb{R}^{m(N+1)\times l(N+1)}$ a psedo inverse is defined by a matrix $G^+ \in \mathbb{R}^{l(N+1)\times m(N+1)}$, satisfying the criteria:*

(a) $GG^+G = G$,

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

(c) $(GG^+)^T = GG^+$,

(d) $(G^+G)^T = G^+G$.

Algorithm 10 *The Pseudo Inverse Model Algorithm (PIA) is given via input update law*

$$\begin{aligned} u_{k+1} &= u_k + \beta G^+ e_k, \\ u_0 &\in \mathbb{R}^{l(N+1)}, \end{aligned} \tag{3.12}$$

with the error evolution

$$\begin{aligned} e_{k+1} &= (I - \beta GG^+)e_k,\ k \geq 0, \\ e_0 &= r - Gu_0 - d. \end{aligned} \tag{3.13}$$

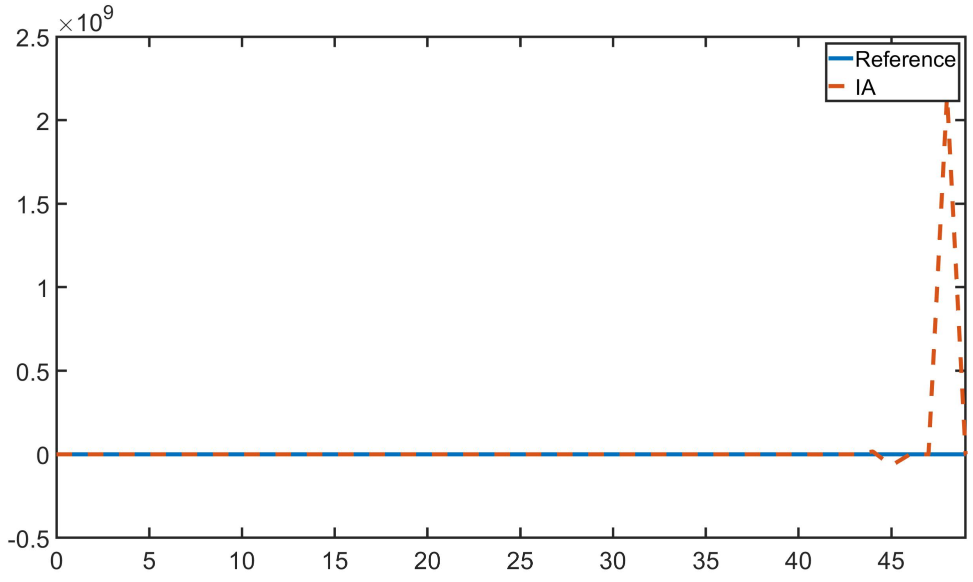

Figure 3.2: Tracking with IA for the system (3.5) for $N = 50$

Monotonic convergence to

$$e_\infty = \lim_{k\to\infty} e_k = P_{\ker[G^+]} e_0, \tag{3.14}$$

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. That is, there exists an input* u_∞*, such that* $r = Gu_\infty + d$*.*

Proof:
Because of the pseudo inverse definition, it holds $(GG^+)^2 = GG^+GG^+ = GG^+$. Then the error evolution formula can be proven by mathematical induction over k.

For $k = 1$ it follows:

$$e_1 = (I - \beta GG^+)e_0 = (I + GG^+ - GG^+ - \beta GG^+)e_0 = (I + [(1-\beta) + 1]\, GG^+)e_0.$$

For $k \in \mathbb{N}$

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

To prove (3.14) recall that

$$\mathbb{R}^{m(N+1)} = \operatorname{im}(G) \oplus \ker(G^+),$$

The symbol $\oplus$ denotes here the direct sum of two vector spaces. Hence, e_0 can be written as

$$e_0 = Gw_0 + v_0,$$

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

$$e_k = (1 + \left[(1-\beta)^{k-1} - 1\right] GG^+)e_0 = (1-\beta)^{k-1}Gw_0 + v_0.$$

For $\beta \in (0, 2)$ the error e_k converges to v_0 for $k \to \infty$.
Further, it is clear, that if the error e_k converges to zero, the problem is feasible with $u_\infty = \lim_{k\to\infty} u_k$. On the other hand, if the problem is feasible, it holds

$$e_k = r - y_k = G(u_\infty - u_k), \qquad (*1)$$

and hence by using the property (a) from Definition 9

$$\begin{aligned} e_{k+1} &= (I - \beta GG^+)e_k = (I - \beta GG^+)G(u_\infty - u_k) && (*2a) \\ &= (G - \beta GG^+G)(u_\infty - u_k) = (1-\beta)G(u_\infty - u_k) && (*2b) \\ &= (1-\beta)e_k = (1-\beta)^k e_0 \to 0 \text{ as } k \to \infty, && (*2c) \end{aligned}$$

because of choice of β. ■

Example 11 *Calculate a solution for* (3.5) *with PIA, and illustrate it in Figure 3.3. This time the tracking signal and system output fit well despite the large condition number. However, a deviation can be observed at the beginning, which causes the bad invertability of the matrix G.*

Owens in his book [**ILC**] chooses the left and right inverse matrix instead of the pseudo

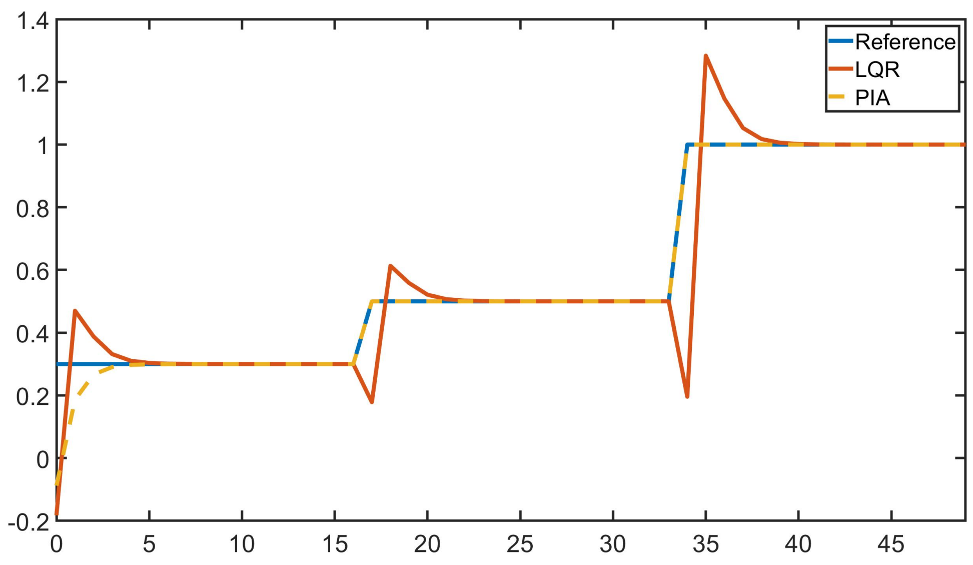

Figure 3.3: Reference signal tracking with LQR and PIA for the sysytem (3.5) for $N = 50$

inverse. The pseudo inverse has its advantage to equal the right/left inverse, if it exists. Moreover, for right/left inverse does not exist for any matrix. The algorithm in the above presented formulation can be applied for any matrix G and can be linked to the feasibility of the tracking value r.

The Algorithm 7 is a special case of PIA. However, numerically the use of the inverse matrix might be more desirable as it requires less calculation effort, if the condition number is not large.

The (Pseudo) Inverse Algorithm belongs to the class of so-called Unit Memory Algorithms. These are the algorithms which can "remember" the last trial.

3.2 Unit Memory Algorithms

Of interest are the processes, which are executed repetitively. Let $k = 0, 1, 2, \ldots$ be the number of completed iterations. Then for the k-th trial the system, it holds

$$\begin{aligned} y_k &= Gu_k + d, \\ e_k &= r - y_k, \end{aligned} \tag{3.15}$$

with some initial input $u_0 \in \mathbb{R}^{l(N+1)}$.

Each pass k the tracking can be improved, if the update law is chosen properly. Assuming the linear dependency of u_{k+1} upon u_k and e_k for $k \geq 0$, the following algorithms results by using the feedback control.

Algorithm 12 *The Unit Memory Algorithm is given via the input update law*

$$\begin{aligned} u_{k+1} &= u_k + Ke_k, \\ u_0 &\in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{3.16}$$

The error dynamic is then given via

$$e_{k+1} = (I - GK)e_k, \; k \geq 0, \tag{3.17}$$

$$e_0 = r - Gu_0 - d. \tag{3.18}$$

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

To achieve a good tracking at trial k the norm $||e_k||$ is must be small. The tracking behaviour improves for consecutive iterations if the sequence $||e_k||$ is monotonically decreasing, i.e.,

$$||e_{k+1}|| < ||e_k|| \text{ for all } k \geq 0. \tag{3.19}$$

Note that in this case the sequence $||e_k||$ is guaranteed to converge. If the sequence even converges to zero, the perfect tracking is achieved asymptotically.

To the small error, it is enough to consider the iteration process

$$e_{k+1} = (I - GK)e_k = (I - GK)^k e_0 := L^k e_0, \; k \geq 0. \tag{3.20}$$

This is a discrete time dynamical system over k. Then the goal can be reformulated as follows: find some matrix $K \in \mathbb{R}^{l(N+1)\times m(N+1)}$, which renders (3.20) stable.

Rewriting the system as

$$\begin{pmatrix} e_{k+1} \\ e_k \end{pmatrix} = \left(\begin{array}{c|c} I & -G \\ \hline I & 0 \end{array}\right) \begin{pmatrix} e_k \\ v_k \end{pmatrix}, \tag{3.21}$$

our problem reduces to finding a stabilizing controller K with

$$v_k = Ke_k. \tag{3.22}$$

At the beginning of this thesis one way to calculate such a controller was presented: the controller based on separation principle. This is a reliable method, but since the matrix G can have a large dimension, it also can be costly. Often good results are achievable with more simple methods. For example, the number of parameters to be chosen can be reduced. In the presented Algorithms 7 and 10, the number of parameters to be chosen was reduced to 1, and these algorithms still show acceptable results. With notation from Algorithm 12, the Algorithms 7 and 10 can be written as

$$K = \beta K_0 \tag{3.23}$$

with a fixed matrix K_0 as the (pseudo) inverse of G, and left the parameter β to decide on.

However, the calculation of the inverse matrix can be not very reliable method. The pseudo inverse delivers better results, but is also much more expensive. An algorithm, which does not require the model inversion, might be of use here.

3.3 Gradient Algorithms

The aim of the ILC algorithms is to make the error $||e_k||$ small in some norm $||\cdot||$ in $\mathbb{R}^{m(N+1)}$. It is advantages, even to have a monotonic convergence, which means the (3.19) must hold. In the above algorithms, the straightforward way was used: some algorithm was chosen based on the system, and the error norm was shown to converge to some $e_\infty \in \mathbb{R}^{m(N+1)}$. Now, derivate an algorithm from the other side, and begin with the consideration of the error norm in the first place.

Choose the norm by defining the weighting matrices $Q(t)$ and $R(t)$, $t = 0, 1, 2, \ldots, N$. Let the norms $||\cdot||_Q$, $||\cdot||_R$ to be defined associated with scalar products

$$\langle y, z\rangle_Q = \sum_{t=0}^{N} y(t)^T Q(t) z(t), \;\; \langle u, v\rangle_R = \sum_{t=0}^{N} u(t)^T R(t) v(t), \tag{3.24}$$

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

$$G^\star = R^{-1} G^T Q \text{ for all } G \in \mathbb{R}^{m(N+1)\times l(N+1)}, \tag{3.25}$$

where

$$\begin{aligned} Q &:= \operatorname{diag}(Q(0), Q(1), Q(2), \ldots, Q(N)) \text{ and} \\ R &:= \operatorname{diag}(R(0), R(1), R(2), \ldots, R(N)). \end{aligned} \tag{3.26}$$

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 (3.24).

Applying the weighted norms on (3.19) provides

$$\begin{aligned} ||e_{k+1}||_Q^2 = ||e_k - GKe_k||_Q^2 &= ||e_k||_Q - 2\langle e_k, GKe_k\rangle_Q + ||GKe_k||_Q^2 \\ &< ||e_k||_Q^2, \text{ if } ||GKe_k||_Q^2 < 2\langle e_k, GKe_k\rangle_Q. \end{aligned} \tag{3.27}$$

The last inequality is equivalent to

$$e_k^T K^T G^T Q G K e_k < 2 e_k^T Q G K e_k. \tag{3.28}$$

If $K = \beta G^\star$, this is implicated by

$$\beta^2 (GG^\star)^T Q (GG^\star) \prec \beta Q G G^\star + \beta (Q G G^\star)^T. \tag{3.29}$$

Since the matrix Q is positive definite, there exists a positive definite matrix $Q^{1/2}$, such that $Q = Q^{1/2} Q^{1/2}$.

Write $Q^{-1/2}$ for $\left(Q^{1/2}\right)^{-1}$, and define a matrix

$$H := Q^{1/2} G G^\star Q^{-1/2}. \tag{3.30}$$

The matrix H has the same eigenvalues as $GG^\star$, as for its characteristic polynomial holds

$$\begin{aligned} \det(I - \lambda H) &= \det\left(I - \lambda Q^{1/2} G G^\star Q^{-1/2}\right) \\ &= \det\left(Q^{1/2}\right) \det\left(I - \lambda G G^\star\right) \det\left(Q^{1/2}\right)^{-1} \\ &= \det\left(I - \lambda G G^\star\right) \text{ for any } \lambda \in \mathbb{R}. \end{aligned} \tag{3.31}$$

Rewrite (3.29) as

$$\beta^2 H^T H \prec \beta (H + H^T). \tag{3.32}$$

This relation is fulfilled if

$$0 < \beta < \frac{2}{\lambda_{\max}(H)}, \tag{3.33}$$

where $\lambda_{\max}(H)$ is the greatest eigenvalue of the matrix H in its absolute value.

This deliberation inspires the *Steepest Descent Algorithm*.

3.3.1 Steepest Descent Algorithm

Algorithm 13 *The Steepest Descent Algorithm is characterized by choosing* $K = \beta G^\star$*, where* $\beta > 0$ *is a real scalar gain. The input update law is given via*

$$\begin{aligned} u_{k+1} &= u_k + \beta G^\star e_k, \\ u_0 &\in \mathbb{R}^{l(N+1)}, \end{aligned} \tag{3.34}$$

and error evolution results in

$$e_{k+1} = (I - \beta GG^\star)e_k,\ k \geq 0. \tag{3.35}$$

Monotonic convergence to

$$e_\infty = \lim_{k\to\infty} e_k = P_{\ker[G^\star]}e_0, \tag{3.36}$$

is guaranteed if

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

Convergence to zero is assured in the case that $e_0 \in \mathrm{im}[GG^\star]$ *holds.*

Proof:

To prove the statement remember that, as $GG^\star$ is a symmetric matrix. This implies a bunch of useful properties. Firstly, there exit the unique vectors $e_{\mathrm{im}} \in \mathrm{im}(GG^\star)$, $e_{\mathrm{ker}} \in \ker(GG^\star)$, such that

$$e_0 = e_{\mathrm{im}} + e_{\mathrm{ker}}. \tag{$*1$}$$

Substituting e_0 in error evolution by this expression results in

$$e_{k+1} = (I - \beta GG^\star)e_k = (I - \beta GG^\star)^k e_{\mathrm{im}} + e_{\mathrm{ker}}. \tag{$*2$}$$

Furthermore, there exist an invertable matrix $S \in \mathbb{R}^{m(N+1)\times m(N+1)}$ and a diagonal matrix $\Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_{m(N+1)} \end{pmatrix}$, such that $(I - \beta GG^\star) = S^{-1}\Lambda S$. The values λ_i, $i = 1, 2, \ldots, \lambda_{m(N+1)}$ are the eigenvalues of the matrix $(I - GG^\star)$.

Then $(*2)$ becomes

$$e_{k+1} = (I - \beta S^{-1}\Lambda S)^k\, e_{\text{im}} + e_{\text{ker}} = S^{-1}(I-\beta\Lambda)^k S\, e_{\text{im}} + e_{\text{ker}} \tag{$*$3a}$$

$$= S^{-1}\begin{pmatrix} 1-\beta\lambda_1 & & & \\ & 1-\beta\lambda_2 & & \\ & & \ddots & \\ & & & 1-\beta\lambda_{m(N+1)} \end{pmatrix}^k S\, e_{\text{im}} + e_{\text{ker}} \tag{$*$3b}$$

$$= S^{-1}\begin{pmatrix} (1-\beta\lambda_1)^k & & & \\ & (1-\beta\lambda_2)^k & & \\ & & \ddots & \\ & & & (1-\beta\lambda_{m(N+1)})^k \end{pmatrix} S\, e_{\text{im}} + e_{\text{ker}}. \tag{$*$3c}$$

Therefore, the $(I-\beta GG^{\star})^k$ converges to 0 (and hence e_k to e_{im}) as $k \to \infty$, if and only if

$$|1-\beta\lambda_i| < 1 \text{ for all } i = 1, 2, \ldots, m(N+1). \tag{$*$4}$$

By using (3.31), the last statement becomes

$$|1-\beta\lambda_{\max}(H)| < 1. \tag{$*$5}$$

This yields the proof. ■

An interesting observation is, that provided by this algorithm signal u_∞ is the unique solution of the (minimum norm) optimization problem

$$u_\infty = \arg \min_{u\in\mathbb{R}^l(N+1)} \{||u-u_0||_R^2 : \text{ subject to } r = Gu + d\}. \tag{3.37}$$

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 the minimum energy solution u_∞ [**ILC**] p. 168.

Choice of the Weighting Matrices

The choice of the weighting matrices Q and R provides the additional degrees of freedom. It impacts the behaviour of the algorithm. For example, with the choice of the matrix Q, some elements of input and output signals can be weighted, depending on their importance in measuring accuracies and convergence rates, or accent the relative importance of different time intervals in measuring tracking accuracy and required convergence rates. For example, the rapid convergence in the initial parts of the time interval provides the choice $Q(t) = \varepsilon^{2t}\tilde{Q}$ 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. Using the ε-weighed $R(t) = \varepsilon^{2t}\tilde{R}$ with some fixed $\tilde{R}$ accents the input signals at some times $0 \le t_1 \le t \le t_2 \le N$. It can be used if we want to limit the control action in some time intervals, or if to reflect the physical units used.

Example 14 *In the Example 11 the perfect tracking at the first steps could not be provided. This can be improved by using an adequate weight Q and SDA.*
The choice

$$Q = \operatorname{diag}\begin{pmatrix}10^{-2} & 1 & 1 & 1 & 10^{-2} & \dots & 10^{-2}\end{pmatrix}. \tag{3.38}$$

speeds up the convergence at the second, third and fourth time step.
The result is illustrated in Figure 3.4.

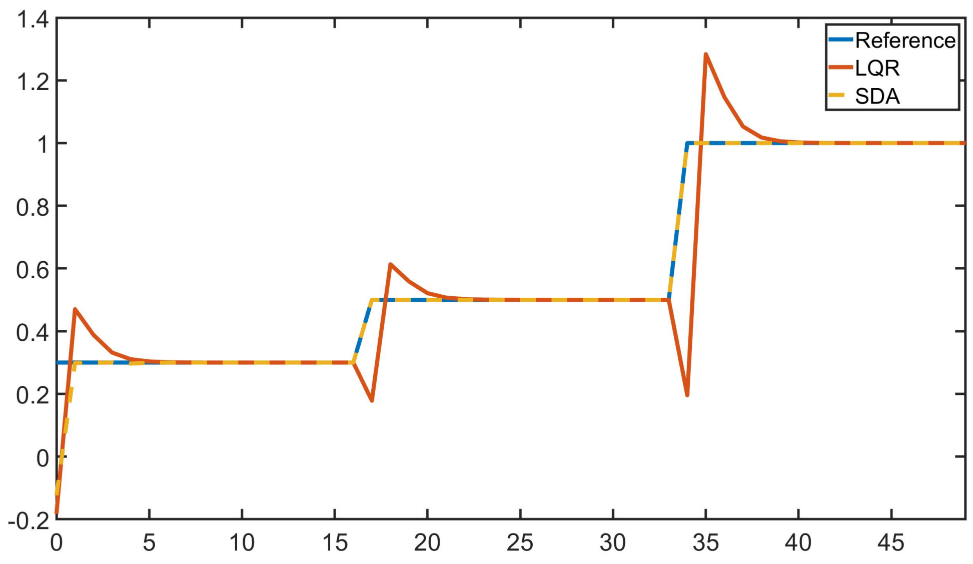

Figure 3.4: Reference signal tracking with SP and SDA for the system (3.5)

By choosing the weight R we the control action becomes regulated.The limitation of the possible input signals, it also leads to a rapid convergence. For example, for $R = I$ the tolerance of 10^{-8} is achieved after 8124 iterations, while with

$$R = \operatorname{diag}\begin{pmatrix}1 & 1 & 1 & 1 & 0.1 & \dots & 0.1\end{pmatrix} \tag{3.39}$$

only 2569 trials are required.

3.3.2 Suppression of eigenvalues

In the previous algorithms the learning matrix K was constant over all iterations. Another way is to choose the variable K_k for $k \geq 0$.

The simplest modification way is to set an iteration varying the gain β for each step: $K_k = \beta_k K_0$, $k \geq 0$ for a fixed matrix K_0. An appropriate choice for β_k seems to be the eigenvalues of the matrix $GG^\star$. This motivates the following theorem.

Theorem 15 *Assume, that the matrix $GG^\star$ has $q+1$ non-zero ordered eigenvalues $\lambda_0, \lambda_1, \dots, \lambda_q$. Set $\beta_k = \frac{1}{\lambda_k}$ for $k = 0, 1, \dots, q$ and consider the update law*

$$\begin{aligned} u_{k+1} &= u_k + \beta_k G^\star e_k, \ k \geq 0 \\ u_0 &\in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{3.40}$$

with error evolution

$$e_{k+1} = (I - \beta_{k+1} GG^\star) e_k = \prod_{l=0}^{k+1} (I - \beta_l GG^\star) e_0, \ k \geq 0. \tag{3.41}$$

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 the spectral theorem, there exists a basis of the eigenvectors $\{v_0, v_1, \dots, v_{\tilde{N}}\}$ with corresponding eigenvalues $\lambda_0, \lambda_1, \dots, \lambda_q, 0, \dots 0$. Then e_0 can be written as

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

with uniquely defined $\gamma_p \in \mathbb{R}$, $p = 0, 1, \dots, \tilde{N}$.
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. If for $k \geq 1$ the first k components were eliminated, the $(k+1)$-th error becomes

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

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 this algorithm is conceptually interesting, it is not suitable for real-world problems. The small non-zero eigenvalues of $GG^\star$ will potentially lead to very large values of β_k and which follow extremely large transient variations in the error norm. Also computing the eigenvalues can be numerically difficult or deliver not precise results, and hence the eliminating of the eigenvectors is not guaranteed. Moreover, to achieve the monotonic convergence only the eigenvalues $\lambda > \frac{1}{2}\lambda_{\max}(H)$ are an option. Furthermore, the model inaccuracy or uncertain eigenvalues have direct impact on the algorithm result.
That all makes this algorithm not solid. Still, the idea to apply the eigenvalues of $GG^\star$ still can be used – but not directly, but by "picking" the compatible eigenvalues.

Algorithm 16 *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 (SoEA) is defined via choosing of the iteration-depending control law $K_k = \beta_k G^\star e_k$, where*

$$\begin{aligned} \beta_k &= \frac{1}{p_k} \text{ for } k = 0, 1, \ldots, N_p, \\ \beta_k &= \beta \text{ for } k > N_p, \end{aligned} \qquad (3.42)$$

with $\beta \in (0, \frac{2}{\lambda_{\max}})$.
Then the input update law becomes

$$u_{k+1} = u_k + \beta_k G^\star e_k, \; k \geq 0, \qquad (3.43)$$

$$u_0 \in \mathbb{R}^{l(N+1)}, \qquad (3.44)$$

and error evolution results in

$$\begin{aligned} e_k &= \left[\prod_{l=0}^{k} (I - \beta_l GG^\star) \right] e_0, \; k = 0, 1, \ldots, N_p, \\ e_k &= (I - \beta GG^\star)^{k - N_p} e_{N_p}, \; k > N_p. \end{aligned} \qquad (3.45)$$

This algorithm has the same convergence properties as Algorithm 13, 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 17 *Apply SoEA on the system* (3.5) *with weights $Q = I$, $R = I$. The error evolution we get is illustrated in Figure 3.5.*
The new algorithm needs only 121 iterations, while for the not adjusted SDA 730 trials are necessary. Indeed, for 31 of 51 eigenvalues λ of $GG^\star$ holds

$$\lambda > \frac{1}{2}\lambda_{\max}(H) = 461.5\,. \tag{3.46}$$

Also the influence of the weighting matrices on the convergence speed is here demonstrated. Since no better performance at the first time steps is required, the algorithm converges with much less iterations.

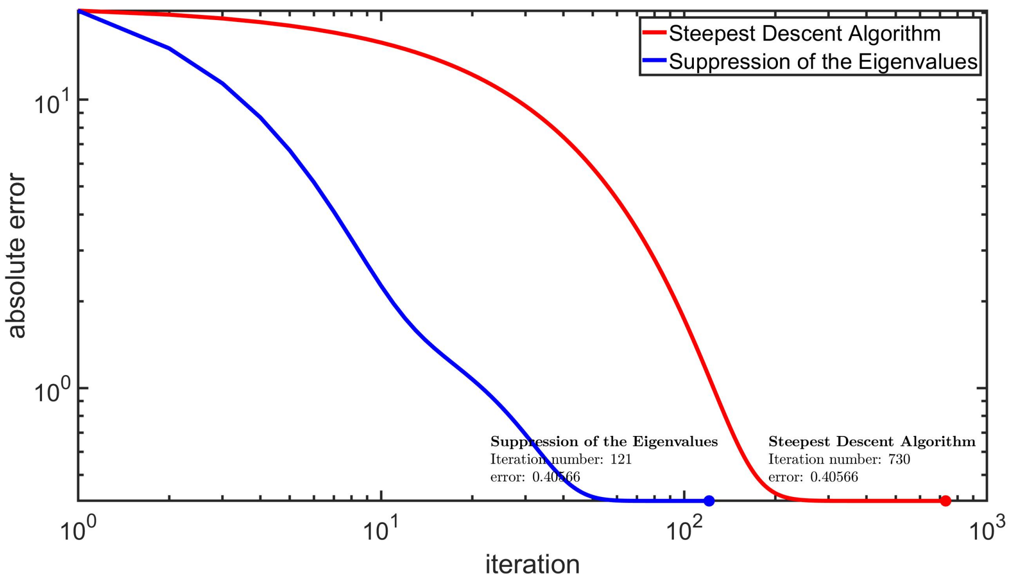

Figure 3.5: Error evolution for SDA and SoEA for system (3.5)

3.4 Feedback Design

The previous algorithms may achieve a better tracking for repetitively executed processes. But they do not accent the special features of the model. For example, the non-minimum-phase zeros are not managed, but they can impact the convergence rate [**ILC**]. The consideration of the plant structure and contraction of a feedback controller, further converted into a steepest descent-like algorithm can possibly achieve better performance and robustness properties.
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, consider a forward path compensator $K_c(z)$ in a unity negative feedback system for (2.1). 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, the plant properties such as oscillation, loop interaction, or the effects of non-minimum-phase zeros can be remedied .

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

$$T(z) = (I + G(z)K_c(z))^{-1}G(z)K_c(z) \text{ and } S(z) = I - T(z) = (I + G(z)K_c(z))^{-1}, \tag{3.47}$$

and define the controller

$$K_0(z) = K_c(z)S(z). \tag{3.48}$$

Then the compensated Steepest Descent Algorithm is given as follows.

Algorithm 18 *Write the transfer functions in supervector description as K_0, K_c, T and S. The compensated Steepest Descent Algorithm is characterized by choosing $K = \beta K_0$, $\beta \in \mathbb{R}$. The iterative law is given via*

$$\begin{aligned} u_{k+1} &= u_k + \beta K_0 T^* e_k, \\ e_{k+1} &= (I - \beta T T^*)e_k, \ k \geq 0, \\ u_0 &\in \mathbb{R}^{l(N+1)}. \end{aligned} \tag{3.49}$$

Monotonic convergence to

$$e_\infty = \lim_{k\to\infty} e_k = P_{\ker[T^*]}e_0, \tag{3.50}$$

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 that of Algorithm 13 if consider the system

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

■

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

$$y = Tr. \tag{3.51}$$

A perfect tracking is ensured if $T = I$ – which is not achievable by feedback control. But still, a good tracking can be expected if $||T|| \approx 1$. Therefore if K_c provides excellent feedback control of G, then rapid convergence could be attained with a choice of the gain $\beta \approx 1$ [**ILC**].

Hier können auch Schlußfolgerungen präsentiert und ein Ausblick gegeben werden.

.1 Beweise

.1.1 Beweis 1

$$1 + 1 = 2 \tag{15}$$

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