Design of PI Controller by Minimization of IAE

Andreas, Helena; Åström, Karl Johan

1997

Document Version:
Publisher's PDF, also known as Version of record

ISSN 0280–5316
ISRN LUTFD2/TFRT--7565--SE

Design of PI Controller by Minimization of IAE

Helena Andreas
Karl Johan Åström

Department of Automatic Control
Lund Institute of Technology
September 1997

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

Document name
REPORT

Date of issue
September 1997

Document Number
ISRN LUTFD2/TFRT--7565--SE

Author(s)
Helena Andreas and Karl Johan Åström

Supervisors

Sponsoring organisation

Title and subtitle
Design of PI Controller by Minimization of IAE.

Abstract

This paper explores the possibilities of designing PI controller simply by minimizing the integrated absolute error for a load disturbance on the process input. It is shown that this may lead to controller that are very sensitive to parameter variations. The method is therefore not useful unless constraints on the robustness are introduced. The method is also very demanding computationally. If a constraint on the robustness is introduced the method is not very different from minimization of the integrated error which is much less demanding computationally.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
18

Recipient's notes

Security classification

Design of PI Controller by Minimization of IAE*

H. Andreas K. J. Åström

Department of Automatic Control
Lund Institute of Technology, Box 118, Lund, Sweden
Fax +46-46-138118, email: helena@control.lth.se

Abstract

This paper explores the possibilities of designing PI controller simply by minimizing the integrated absolute error for a load disturbance on the process input. It is shown that this may lead to controller that are very sensitive to parameter variations. The method is therefore not useful unless constraints on the robustness are introduced. The method is also very demanding computationally. If a constraint on the robustness is introduced the method is not very different from minimization of the integrated error which is much less demanding computationally.

1. Introduction

The PI(D) controller is the main algorithm used in industrial control. The usefulness of the controller is improved substantially by features for automatic tuning and adaptation, see [1]. This has led to an increased interest in methods for finding proper controller parameters. When designing the controller it is important to consider

- Load disturbances
- Measurement noise
- Sensitivity to process variations

[1]This work has been supported by the Swedish Research Council for Engineering Science under contract 95-759

- Set point response
- Process information required
- Simplicity of the method

In most applications the primary purpose of the controller is to reduce the effects of load disturbances. A classic method for tuning controllers is to minimize IAE, which is the integrated absolute error cause by a unit step load disturbance at the process input, see [2]. The main drawback with this is that the calculations are complicated. For this reason it has been suggested to instead minimize the integrated error with a constraint on the sensitivity. In several cases this gives the result which are similar to those obtained by minimizing IAE, particularly if the constraining value of the maximum sensitivity is chosen sufficiently small, see [3].

The purpose of this report is to investigate whether minimization of the IAE gives good PI controllers without additional constraints. It is also investigated if computationally efficient method for the minimization procedure can be found. A scheme for computing the gradient and the Jacobian of the loss function is used. This method can actually be applied to any loss function. The method is similar to the ones used in maximum likelihood system identification, see [4].

Conclusions

We found that minimization of the IAE is not sufficient for good control performance. A sensitivity constraint must be used to moderate the controller. The numerical procedure used involves extensive computations and is therefore not a good practical option. Furthermore, the optima are usually flat and convergence is consequently very slow.

2. The Problem

Consider a the closed loop system shown in Figure 1 with a process and a controller. There are load disturbances that drive the process away from it desired state, measurement noise that corrupts the information from the process and command signal changes. The controller has parameters θ which should be adjusted to minimize a performance criterion J for a particular set of disturbances. In the figure the inputs are denoted y_{sp}, l and n, and the interesting signals, are u, e and y. The loss function is a scalar function of the signals. For the IAE it

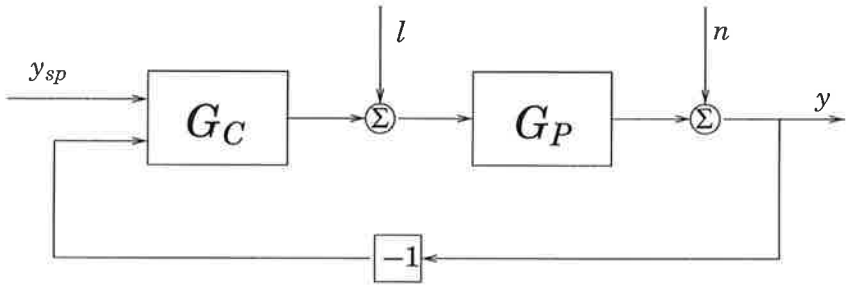

Figure 1 Block diagram of a simple closed loop system.

becomes simply

$$J(\theta) = \int_0^\infty |e((\theta,t),t)|dt$$

where θ represents the PID-parameters to be determined. In general we will assume that y_{sp}, l and n are fixed and that the loss function is

$$J(\theta) = \int_0^\infty g(x(\theta),t)dt \tag{1}$$

where $x(\theta,t) = [x_1(\theta,t)\ x_2(\theta,t)\ \ldots\ x_n(\theta,t)]^T$ are the states of the system. The problem is thus to minimize the function J. We will try to do this by an approximate Newton method. For this purpose we need to calculate the gradient and the (approximate) Jacobian matrix.

3. Optimization

The method for solving the equation $J'(\theta) = 0$ consists of expanding the function in a Taylor series and ignoring the derivatives of order higher than two. The idea to do this was taken from [5]. Solving for θ gives the iterative algorithm

$$\theta_{i+1} = \theta_i - \gamma_i J'(\theta_i)\left[J''(\theta_i)\right]^{-1} \tag{2}$$

where γ_i is used to change the iteration step length. Obviously, we have to find the gradient and the Jacobian matrix before going any further.

The Gradient

The gradient is found by straightforward derivation of the loss function

$$J'(\theta) = \int_0^\infty (\frac{\delta g}{\delta x})^T (\frac{\delta x}{\delta \theta})dt \tag{3}$$

Consequently, the time derivative of the gradient is

$$\frac{dJ'(\theta)}{dt} = (\frac{\delta g}{\delta x})^T(\frac{\delta x}{\delta \theta}) \tag{4}$$

The Jacobian

Yet another derivation with respect to θ yields the Jacobian matrix

$$J''(\theta) = \begin{bmatrix} \int_0^\infty \frac{\delta^2 J}{\delta \theta_1^2} dt & \int_0^\infty \frac{\delta^2 J}{\delta \theta_1 \delta \theta_2} dt & \dots & \int_0^\infty \frac{\delta^2 J}{\delta \theta_1 \delta \theta_m} dt \\ \int_0^\infty \frac{\delta^2 J}{\delta \theta_2 \delta \theta_1} dt & \int_0^\infty \frac{\delta^2 J}{\delta \theta_2^2} dt & \dots & \int_0^\infty \frac{\delta^2 J}{\delta \theta_2 \delta \theta_m} dt \\ \vdots & \vdots & \ddots & \vdots \\ \int_0^\infty \frac{\delta^2 J}{\delta \theta_m \delta \theta_1} dt & \int_0^\infty \frac{\delta^2 J}{\delta \theta_m \delta \theta_2} dt & \dots & \int_0^\infty \frac{\delta^2 J}{\delta \theta_m^2} dt \end{bmatrix} \tag{5}$$

By neglecting the second order derivatives of x the computations are significantly simplified

$$J''(\theta) = \int_0^\infty (\frac{\delta g}{\delta x})^T \frac{\delta^2}{\delta \theta^2} + (\frac{\delta x}{\delta \theta})^T \frac{\delta^2 g}{\delta x^2} (\frac{\delta x}{\delta \theta}) dt \approx \int_0^\infty (\frac{\delta x}{\delta \theta})^T \frac{\delta^2 g}{\delta x^2} (\frac{\delta x}{\delta \theta}) dt \tag{6}$$

To evaluate this we need to compute the sensitivity derivatives $\frac{\delta x}{\delta \theta}$, $n * m$ in number. In general the sensitivity derivatives can be found through

$$\begin{aligned} \frac{dx}{dt} &= f(x(\theta, t), \theta) \\ \frac{d}{dt}(\frac{\delta x}{\delta \theta}) = \frac{\delta}{\delta \theta}(\frac{dx}{dt}) &= f_x \frac{\delta x}{\delta \theta} + f_\theta \end{aligned} \tag{7}$$

If f_x and f_θ are known, this is a differential equation on the common form $\frac{dz}{dt} = f_x z + f_\theta$ with $z = \frac{\delta x}{\delta \theta}$ and can be solved.

The IAE case

The special case of the IAE, that is the loss function of (2) gives

$$\frac{\delta J}{\delta \theta} = \int_0^\infty sgn(e) \frac{\delta e}{\delta \theta} dt = -\int_0^\infty sgn(e) \frac{\delta y}{\delta \theta} \tag{8}$$

because $e = y_{sp} - y$ and, as y_{sp} is constant, $\frac{\delta e}{\delta \theta} = -\frac{\delta y}{\delta \theta}$. Proceeding in the same fashion as before gives

$$\frac{\delta^2 J}{\delta \theta^2} = -\int_0^\infty sgn(e) \frac{\delta^2 y}{\delta \theta^2} dt + \int_0^\infty 2\delta(e) (\frac{\delta y}{\delta \theta})^T (\frac{\delta y}{\delta \theta}) dt \tag{9}$$

The second order derivatives are again neglected

$$\frac{\delta^2 J}{\delta\theta^2} \approx \int_0^\infty 2\delta(e)(\frac{\delta y}{\delta\theta})^T(\frac{\delta y}{\delta\theta})dt = 2\sum_{e(t)=0}(\frac{\delta y}{\delta\theta})^T(\frac{\delta y}{\delta\theta}) \tag{10}$$

Alas, closer scrutiny of these equations reveals that the gradient is a discontinuous function when e approaches zero, and the Jacobian is therefore singular near the desired (zero) value of e.

We tried approximating the function $|e|$ with $\frac{e^2}{\varepsilon+|e|}$ when calculating derivatives in the hope of getting a non-singular Jacobian matrix, but this did not improve matters much. (Besides, calculation of all the elements of the Jacobian increases the computational time substantially.) Finally, we settled for the simpler iteration algorithm

$$\theta_{i+1} = \theta_i - \gamma_i J'(\theta_i) \tag{11}$$

where γ_i is a diagonal matrix used to change the iteration step length. This algorithm converges rather more slowly than we had hoped for, due to (among other things) the small step length that turned out to be necessary.

4. Application to a PI-controlled process

In General

Consider a linear system without time delay

$$\begin{aligned} \frac{dx_p}{dt} &= A_p x_p + B_p(u+l) \\ y &= C_p x_p + n \end{aligned} \tag{12}$$

controlled by a PI-controller described by

$$U = k\left[(bY_{sp} - Y)\right] + \frac{k_i}{s}(Y_{sp} - Y) \tag{13}$$

We prefer this description to the one with $\frac{k}{T_i}$ as the integration factor k_i can be determined more independently of k. To obtain a state-space form for the controller introduction of state variables for the integration (x_{n+1}) is necessary.

$$\frac{dx_{n+1}}{dt} = k_i(y_{sp} - y) \tag{14}$$

The controller signal thus becomes

$$u = k\left[-C_p x_p + b y_{sp} - n\right] + x_{n+1} \tag{15}$$

Inserting this in (12) yields the system on state-space form

$$\begin{aligned} \frac{dx}{dt} &= \begin{bmatrix} \hat{A}_p & B_p \\ -k_i C_p & 0 \end{bmatrix} \begin{bmatrix} x_p \\ x_{n+1} \end{bmatrix} + \begin{bmatrix} \hat{B}_1 \\ k_i \end{bmatrix} y_{sp} + \begin{bmatrix} \hat{B}_2 \\ 0 \end{bmatrix} l + \begin{bmatrix} \hat{B}_3 \\ -k_i \end{bmatrix} n \\ y &= \begin{bmatrix} C_p & 0 \end{bmatrix} x + n \end{aligned} \tag{16}$$

with $x = [x_p \; x_{n+1}]^T$ the state vector and

$$\begin{aligned} \hat{A}_p &= A_p - k B_p C_p \\ \hat{B}_1 &= k b B_p \\ \hat{B}_2 &= B_p \\ \hat{B}_3 &= -k B_p \end{aligned}$$

The controller parameters to be determined are $\theta = [k \; k_i]$. b is set to be constant ($b = 1$).

Off-Line Optimization

Applying the concept of (7) gives $f_x = A$ and $f_\theta = \left[\frac{\delta f}{\delta k} \; \frac{\delta f}{\delta k_i}\right]$. After some computations

$$f_\theta = \begin{bmatrix} -B_p C_p x_p + b B_p y_{sp} - B_p n & 0 \\ 0 & -C_p x_p + y_{sp} - n \end{bmatrix} \tag{17}$$

Armed with these equations we can find the sensitivity derivatives

$$\frac{\delta x}{\delta \theta} = \begin{bmatrix} \frac{\delta x_p}{\delta k} & \frac{\delta x_p}{\delta k_i} \\ \frac{\delta x_{n+1}}{\delta k} & \frac{\delta x_{n+1}}{\delta k_i} \end{bmatrix} \tag{18}$$

We can always find a state space realization of the process with $C_p = [1 0 \ldots 0]$, which means that $y = x_1$. Consequently, $\frac{\delta y}{\delta \theta} = \frac{\delta x_1}{\delta \theta}$ which is the first row of the sensitivity derivative matrix (18). Calculation of (8) is now straightforward and θ_{opt} can be found. As stated above, quite a few iterations are necessary. Up to 150 iterations is not uncommon for complete convergence, although the loss function normally decreases very little after 30 iterations or so.

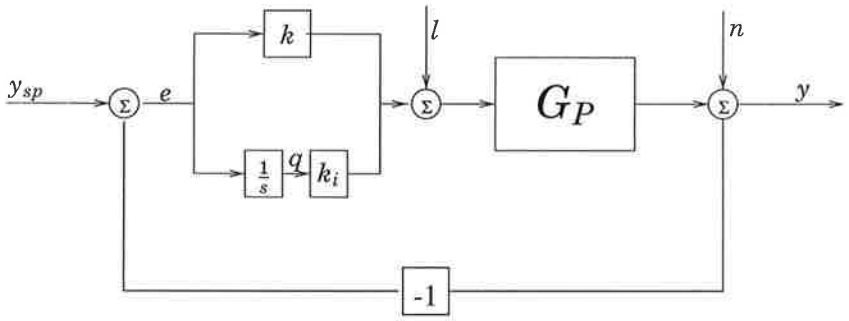

Figure 2 Block diagram for the system with the controller separated into parts.

On-Line Optimization

For on-line tuning, we cannot assume all states to be measurable and must therefore try to compute the sensitivity derivatives through the states we can measure, as in [5]. The system (see Figure 2) is

$$Y = \frac{G_p}{1+G_pG_c}L + \frac{G_pG_c}{1+G_pG_c}Y_{sp} = G_LL + G_{sp}Y_{sp} \tag{19}$$

where $G_p(s)$ is the process transfer function and $G_c(s) = k + \frac{k_i}{s}$ is the controller transfer function. After a step change in set point $G_{sp} \to 1$ as $t \to \infty$. This means that all experiments performed when the set point has been stable for a long time confront the system $Y = G_LL + Y_{sp}$. The sensitivity derivatives are

$$\frac{\delta E}{\delta \theta} = -\frac{\delta Y}{\delta \theta} = -\frac{\delta Y}{\delta G_c}\frac{\delta G_c}{\delta \theta} = \frac{G_p^2}{(1+G_pG_c)^2}L\frac{\delta G_c}{\delta \theta} = [G_L^2L \quad \frac{1}{s}G_L^2L] \tag{20}$$

It seems the sensitivity derivative $\frac{\delta E}{\delta k}$ can be got from sending the load disturbance throughout the system of G_L twice. We examine if this is possible to do while the system is running. The following two experiments are performed

1. Set Y_{sp} constant, $L^{(1)}$ = a step load disturbance. Measure the output $Y^{(1)}$.
2. Set Y_{sp} constant, $L^{(2)} = Y^{(1)} - Y_{sp}$. Measure $Y^{(2)}$ and $Q^{(2)}$.

Between each experiment the system should be allowed to settle down with zero load disturbance. An example is given in Figure 3. Ingeniously, we see that

$$\frac{\delta E}{\delta k} = G_L(Y^{(1)} - Y_{sp}) = G_LL^{(2)} = Y^{(2)} - Y_{sp} \tag{21}$$

$$\frac{\delta E}{\delta k_i} = \frac{1}{s}G_LL^{(2)} = -Q^{(2)} \tag{22}$$

as Q (see Figure 2) measured in step 2 is

$$Q^{(2)} = \frac{1}{s}(Y_{sp} - Y^{(2)}) = \frac{1}{s}Y_{sp} - \frac{1}{s}(G_LL^{(2)} + Y_{sp}) = -\frac{1}{s}G_LL^{(2)} \tag{23}$$

Calculation of (8) is performed as before and the same θ_{opt} as before is found.

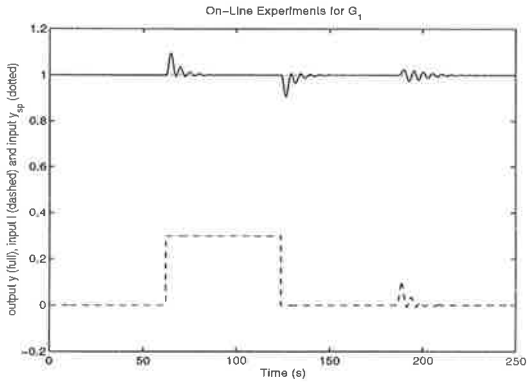

Figure 3 Experiment for system G_1. First the system is allowed to settle from a possible set point step. Second, a step load disturbance is introduced and the output measured. The load disturbance is again set to zero as the system settles down. Finally, the output $y^{(1)}$ is applied as input $l^{(2)}$ and $y^{(2)}$ and $q^{(2)}$ measured.

5. Examples

So we have laboriously found controller parameters that minimize the IAE. But can we be sure that the other aspects of control performance are satisfactory? In this section we examine the achieved response to a set point step, the control signal behavior, the sensitivity to modeling errors and the influence of process noise, especially in comparison to the results when minimizing the integrated error with a constraint on the sensitivity.

The most prominent feature of the method is that the controller parameters that minimize the IAE give high controller amplification. This means that the responses, although fast, tend to oscillate. (See [6] that states that the IAE is minimal when the natural frequency of the system has a maximum.) Furthermore the sensitivity to modeling errors is greater than our reasonable constraints in the IE method allow.

As examples, we have chosen four often encountered systems that are relatively easy to control.

$$G_1 = \frac{1}{(1+s)^3} \qquad G_2 = \frac{1}{(s+1)(1+0.2s)(1+0.04s)(1+0.008s)}$$
$$G_3 = \frac{1}{(s+1)^4} \qquad G_4 = \frac{1}{s(1+s)^2} \tag{24}$$

Process	M_S	k	k_i	*IAE*
$G_1(s)$	**3.5**	**3.37**	**0.90**	**0.37**
	2.0	1.22	0.69	0.57
	1.8	1.06	0.58	0.60
	1.6	0.86	0.46	0.68
	1.4	0.63	0.32	0.94
$G_2(s)$	**4.4**	**12.5**	**15.0**	**0.022**
	2.1	4.13	7.94	0.047
	1.8	3.47	5.60	0.056
	1.6	2.74	4.09	0.074
	1.4	1.93	2.60	0.12
$G_3(s)$	**2.6**	**1.65**	**0.40**	**0.84**
	2.0	0.77	0.38	1.09
	1.8	0.68	0.33	1.11
	1.6	0.57	0.26	1.20
	1.4	0.43	0.19	1.58
$G_4(s)$	**5.0**	**0.98**	**0.100**	**3.38**
	2.0	0.33	0.042	7.58
	1.8	0.29	0.032	9.38
	1.6	0.23	0.021	14.1
	1.4	0.17	0.012	25.2

Table 1 Properties of controllers obtained by minimizing IAE (bold) and IE.

Closed Loop System Response

Table 1 shows the controller parameters obtained by minimizing the IE with the constraints of M_S = 1.4, 1.6, 1.8 and 2.0[2] and the IAE. System G_1 is stable without any particularly fast dynamics whereas system G_2 has one extremely fast pole. System G_3 is similar to G_1 but of a higher order and system G_4 contains an integrator. Figure 4 shows that the closed loop response with the controller parameters from the IAE give a faster and more oscillating response to set point and load disturbance changes. (This naturally leads to a large and oscillating

[2] M_S is the maximum value of the sensitivity function $S(i\omega) = \frac{1}{1+L}$ where L is the transfer function of the open loop system.

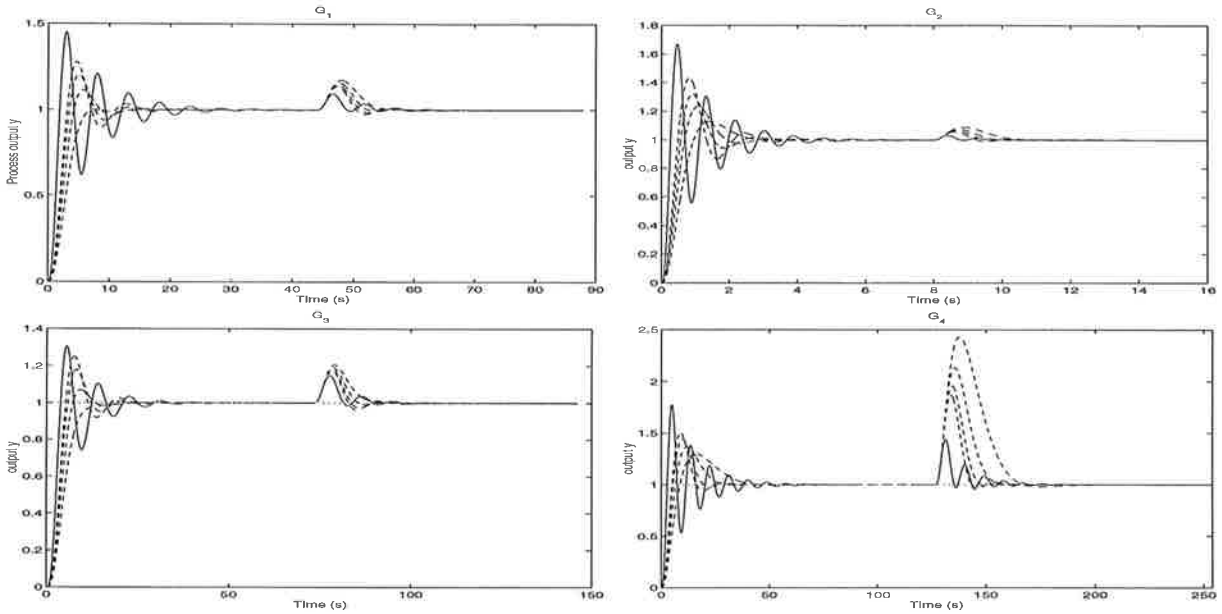

Figure 4 Comparison between the PI controller determined by minimization of the IAE (full) and the ones determined by minimization of the IE with a constraint on the sensitivity (dashed). The weaker the constraint the faster the response. The diagram shows a unit step response followed by a load disturbance of 0.3.

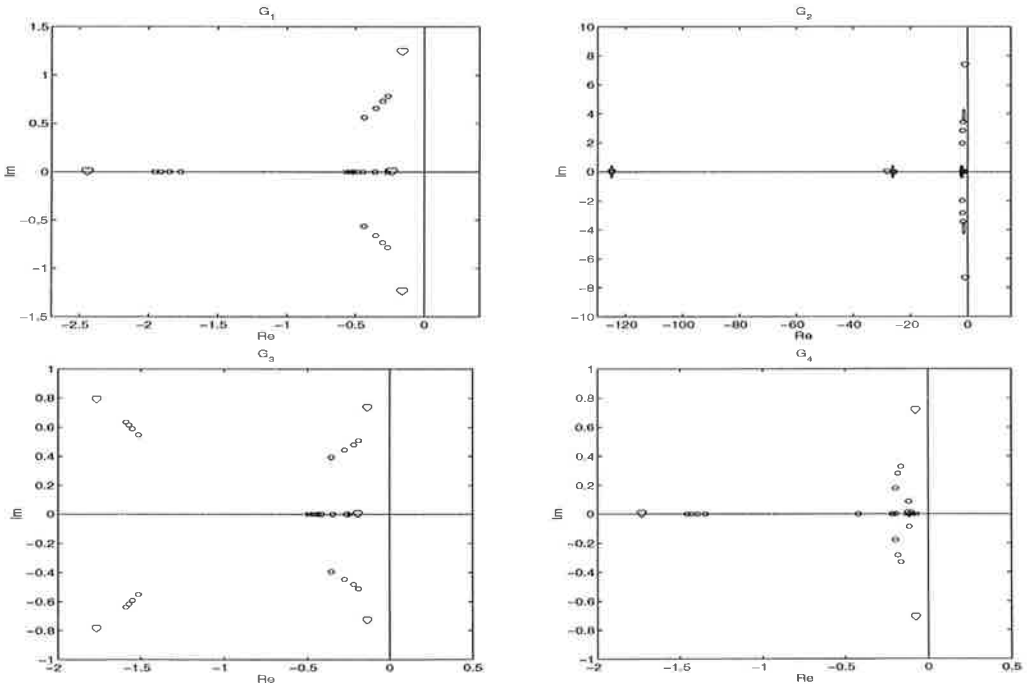

Figure 5 Pole zero plots for the different closed loop systems with different controller parameters. The heart-shapes indicate the poles of the IAE system and the rings the IE systems.

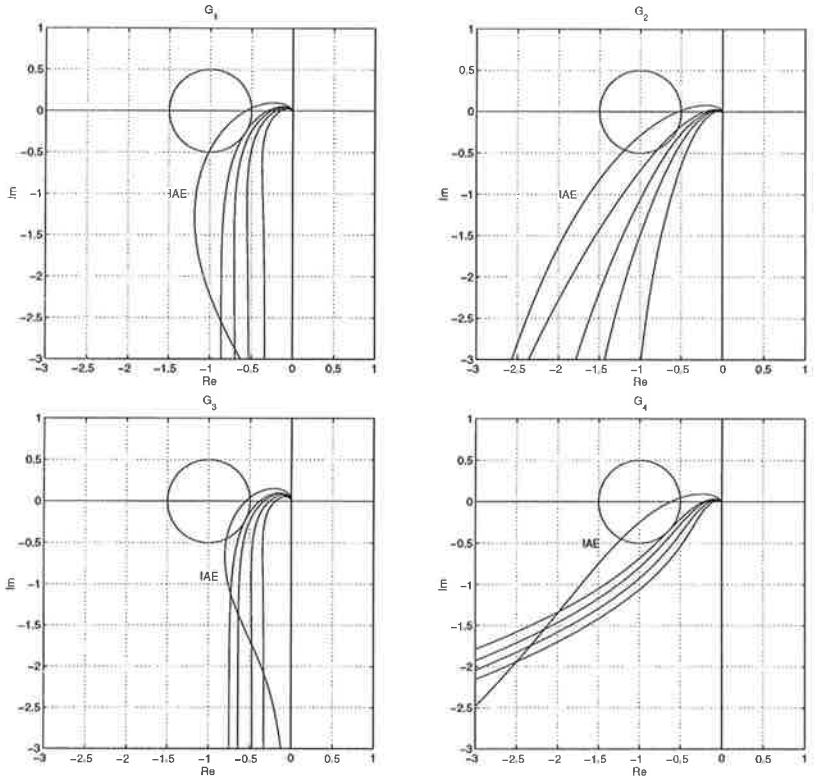

Figure 6 Nyquist plots for the different systems and different controllers. Decreasing sensitivity from left to right. The constraint $M_S = 2$ has been drawn as a circle.

control signal.) Although the reaction is faster for the IAE it takes longer for the error to become zero. This is confirmed by the pole zero plots in Figure 5. The increased oscillatory behavior is especially noticeable as the poles with increasing M_S move closer to the imaginary axis. Notice the big difference between the fastest IE poles and the IAE poles.

Sensitivity to Modeling Errors

The sensitivity to modeling errors is decidedly greater for the IAE than for the IE; obviously the sensitivity for the IE is never greater than the constraints allow. Figure 6 makes it clear that in all IAE cases the sensitivity (M_S is the inverse of the radius of the circle centered in -1 that tangents the Nyquist curve) is considerably larger than 2, the weakest constraint the IE has been determined for. The exact M_S values are given in Table 1. Exploring this yet further, Bode plots of the sensitivity functions in Figure 7 show larger sensitivity peaks for the IAE than for the IE. This is expected, as M_S is the height of the sensitivity peak.

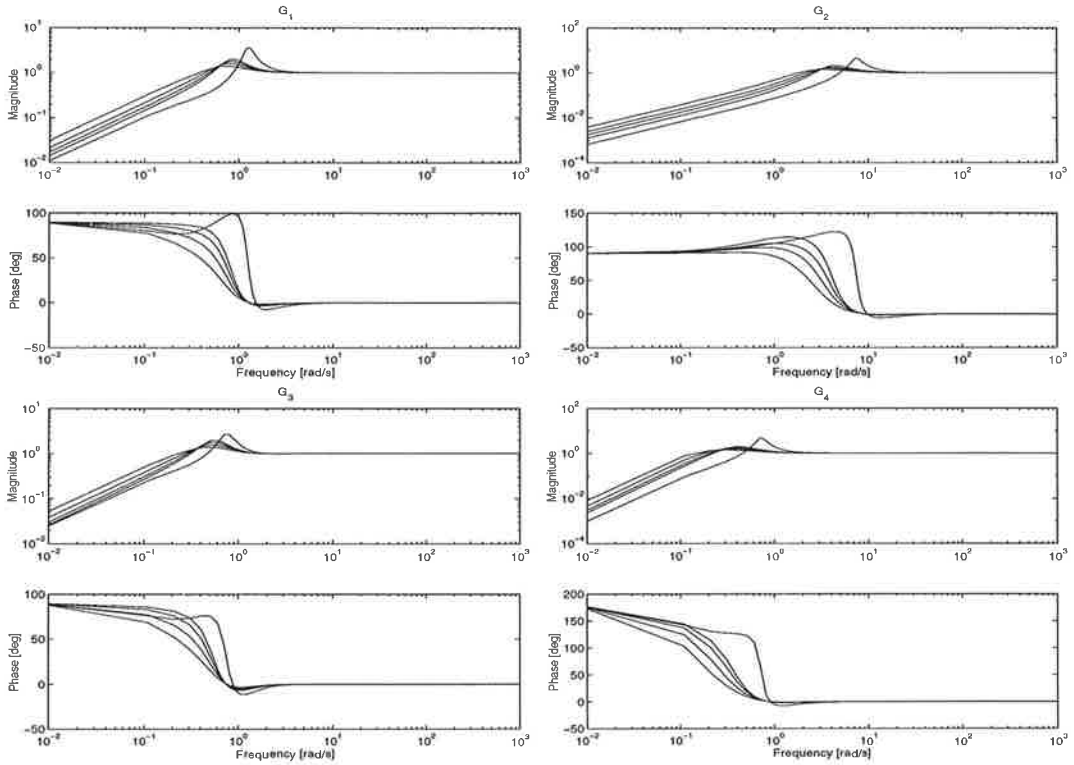

Figure 7 Bode plots for the different systems and different controllers. The resonance peaks decrease with the sensitivity constraint.

Noise Influence

Process noise naturally increases J quite a bit for all the controllers, but the difference in output between the IAE method controller and the controller obtained by the IE method with the hardest constraint on sensitivity ($M_S = 1.4$) is surprisingly small. The control signal, on the other hand, is strongly affected. In a practical application, a control signal such as the one for IAE in Figure 8 is unacceptable.

Improvement possibilities

As seen in Figure 9 our method of minimization can often pinpoint the k_i that minimizes the IAE quite well, but a relatively wide range of k-values give a loss function very close to the minimal one. This implies that thorough knowledge of the system can improve controller performance, if we can settle for an almost-but-not-quite minimal loss function J given by a smaller k. Unfortunately this only marginally improves controller performance in most cases.

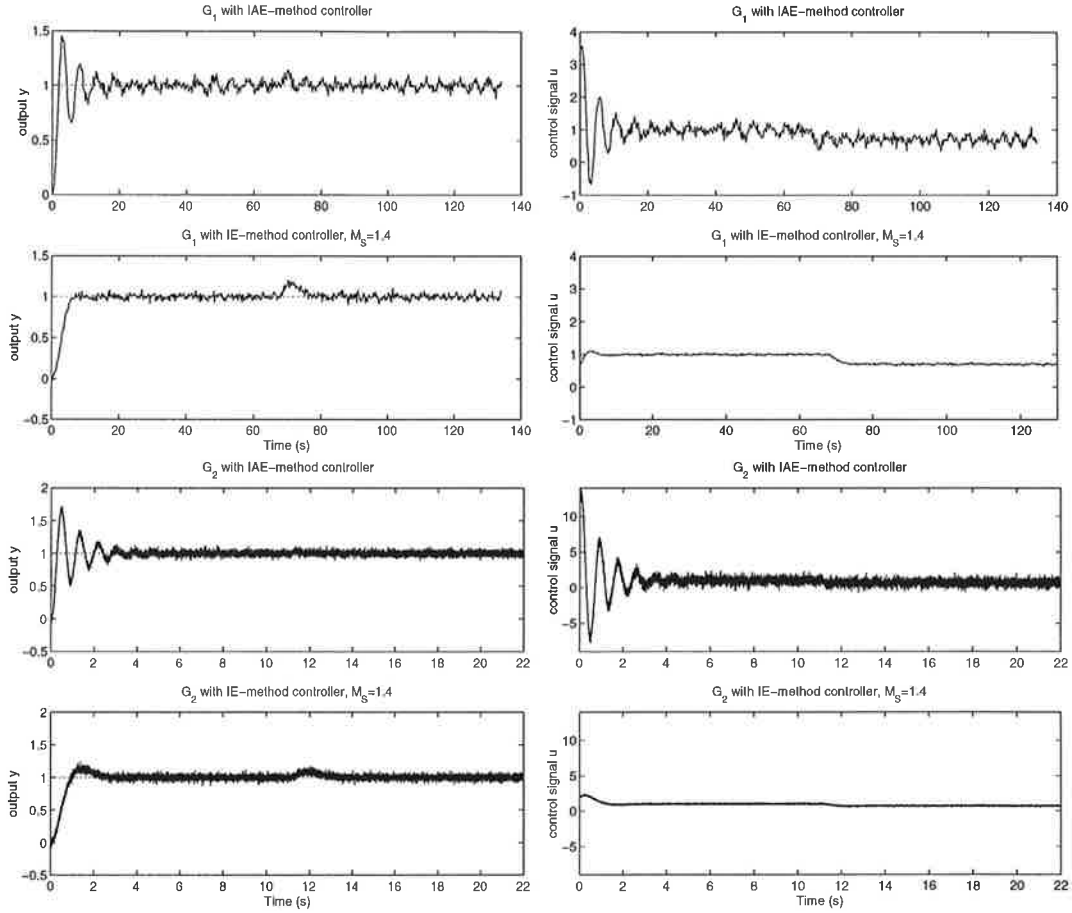

Figure 8 Comparison of process noise in the output and the control signal for systems G_1 and G_2 for the different controllers

6. Conclusions

Unfortunately, minimization of IAE has some serious drawbacks. We conclude that

1. It is not sufficient to minimize IAE. A sensitivity constraint must be used to get acceptable control performance.
2. A good numerical method is hard to find.
    - The computations necessary for the method are complicated and extensive.
    - The optima are usually quite flat which leads to slow convergence.

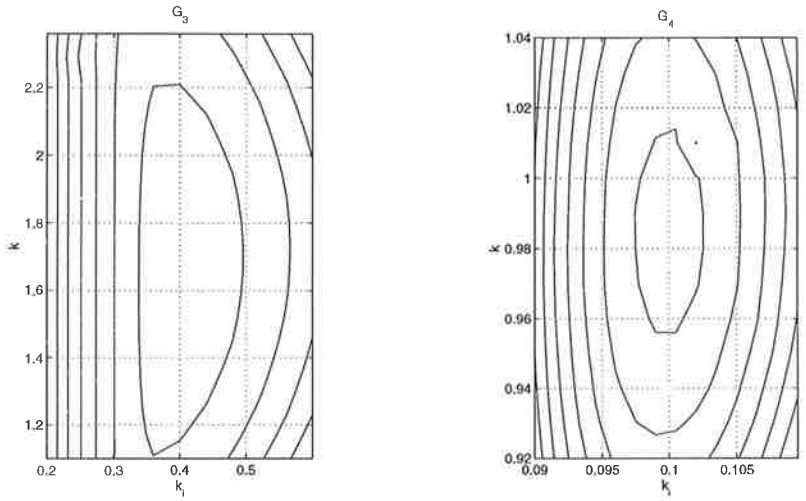

Figure 9 Level curves for the functions G_3 and G_4

7. References

[1] Karl Johan Åström and Tore Hägglund. *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, Research Triangle Park, NC, second edition, 1995.

[2] F. G. Shinskey. *Process-Control Systems. Application, Design, and Tuning*. McGraw-Hill, New York, third edition, 1988.

[3] Karl Johan Åström, Hélène Panagopoulos and Tore Hägglund. *Design of PI Controllers*.

[4] Karl Johan Åström and Torsten Bohlin. Numerical identification of linear dynamic systems from normal operating records. In *Proc. IFAC Conference on Self-Adaptive Control Systems*, Teddington, UK, 1965.

[5] H. Hjalmarsson, S. Gunnarsson and M. Gevèrs A Convergent Iterative Restricted Complexity Control Design Scheme From *33rd Conference on Decision and Control*, December 1994.

[6] F. Leonard An IAE Optimum Control Tuned by One Parameter. From *European Control Conference*, July 1997.

A. Matlab Programs

allofit.m

```
SetInputs; %Sets the inputs and defines the process and cont. pars
Jres=0;
Jprimres=[0 0];
thetares=[0 0];
NbrOfIterations=XX;
j=0;
while j< NbrOfIterations
j=j+1;
[A,B,C,D,SYS,t_end,T,u]=MakeSys(Ap,Bp,Cp,Dp,theta,b,deltaT);
[x_p,x_n1,y]=SimSys(T,u,x0,SYS,m);

ts_end=length(T);            % Last INDEX for u (last TIME is 2*t_end)
ts_0=ceil(0.5*length(T));    % Starting INDEX for calc. of the sens. ders,
                             % ie when the step load disturbance occurs
t_0=ts_0*deltaT;             % Starting TIME for calc. of the sens. ders
tspan=T(ts_0:ts_end);        % From start to stop time after load dist.

[Z1,Z2] = FindSD(Ap,Bp,Cp,m,theta(1),theta(2),b,x_p(ts_0:ts_end,:),x_n1(ts_0:ts_end),
                           u(ts_0:ts_end,1),u(ts_0:ts_end,3),tspan);
J=quad('dJ',t_0,2*t_end-deltaT,[],[],u,y,tspan,ts_0,ts_end);
Jres(j)=J
Jprim1=quad('dJprim',t_0,2*t_end-deltaT,[],[],u,y,Z1,tspan,ts_0,ts_end);
Jprim2=quad('dJprim',t_0,2*t_end-deltaT,[],[],u,y,Z2,tspan,ts_0,ts_end);
Jprim=[Jprim1 Jprim2]
Jprimres(j,:)=Jprim;
gamma_i=[0.1 0;0 0.1];
theta=theta-Jprim*gamma_i
    % theta is now the newest set of controller parameters
thetares(j,:)=theta;
end
U=k*(-(Cp*x_p')'+b*u(:,1)-u(:,3))+x_n1;
```

SetInputs.m

```
% This file is used for setting the system inputs y_sp, l and n.
% T is the time vector for which the inputs are set. If initial values for the
% states are wanted to be non-zero, they can be stated in x0 (a column vector)
% It is also used for defining the process itself and the (initial) controller
% parameters.
Ap=[-2 1 0;-1 0 1;0 0 0];       % The process (invented)
Bp=[0;0;1];
Cp=[1 0 0];
Dp=[0 0 1];                     % Addition of the noise term n

p=size(Bp); m=p(1);             % Size of the process matrices
x0=[0;0;0;0];                   % Initial values for the states
k=0.17; ki=0.012; theta=[k ki]; % Initial controller parameters
b=1;
deltaT=0.2;                     % The space between moments of measurement
```

MakeSys.m

```
function [A,B,C,D,SYS,t_end,T,u]=MakeSys(Ap,Bp,Cp,Dp,theta,b,deltaT);
% Creates the system matrices from the process matrices and the controller
% parameters in theta=[k Ti Td]. For simulation purposes, the system is
% descibed as SYS.
k=theta(1);                    % OBS! Lokala variabler !!!!!
ki=theta(2);
Aphatt=Ap-k*Bp*Cp;
B1hatt=k*b*Bp;
B2hatt=Bp;
B3hatt=-k*Bp;

A=[Aphatt Bp ; -ki*Cp 0 ];
B=[B1hatt B2hatt B3hatt; ki 0 -ki];
C=[Cp 0];
D=Dp;
SYS=ss(A,B,C,D);

lambda=-max(real(eig(A)));     % The slowest processpole determines
t_end=round(10/lambda);        % the integration time (~= Inf) for good
                               % enough stabilization
T=0:deltaT:2*t_end;            % Setting the inputs
y_sp=ones(1,length(T));        % Constant set point
l=[zeros(1,floor(0.5*length(T))) 0.3*ones(1,ceil(0.5*length(T)))];
                               % Step load disturbance after a time T
n=zeros(1,length(T));          % Constant zero noise
%n=0.03*randn(length(T),1)';   % White noise with the amplitude 0.03
u=[y_sp' l' n'];
```

SimSys.m

```
function [x_p,x_n1,y]=SimSys(T,u,x0,SYS,m);
% Simulates the system described by SYS with the inputs y_sp, l and n and
% gives y(t) and x(t). m is the size of the system matrix.
[Y,T,X]=lsim(SYS,u,T,x0);
x_p=X(:,1:m);
x_n1=X(:,m+1);
y=Y;
```

FindSD.m

```
function [Z1,Z2]= FindSD(Ap,Bp,Cp,m,k,ki,b,x_p,x_n1,y_sp,n,tspan);
% Returns the sensitivity derivatives with respect to theta for the
% system described by the matrices Ap,Bp,Cp and controlled with a
% PIDcontroller with the parameters theta=[k Ti Td]. N and b are considered
% predetermined controller parameters.
[T1,Z1]=ode45('firstcol',tspan,[zeros(1,m)';0],[],Ap,Bp,Cp,m,k,ki,b,x_p,x_n1,n,y_sp,tspan);
[T2,Z2]=ode45('middlecol',tspan,[zeros(1,m)';0],[],Ap,Bp,Cp,m,k,ki,b,x_p,x_n1,n,y_sp,tspan);
```

firstcol.m

```
function dz = firstcol(t,z,flag,Ap,Bp,Cp,m,k,ki,b,x_p,x_n1,n,y_sp,tspan);
% Subfunction that describes the sensitivity derivatives with respect to k
Aphatt=Ap-k*Bp*Cp;
dz=[Aphatt*z(1:m)+Bp*z(m+1)-Bp*Cp*interp1(tspan,x_p,t)'+
                          b*Bp*interp1(tspan,y_sp,t)-Bp*interp1(tspan,n,t);
-ki*Cp*z(1:m)];
```

middlecol.m

```
function dz = middlecol(t,z,flag,Ap,Bp,Cp,m,k,ki,b,x_p,x_n1,n,y_sp,tspan);
% Subfunction that describes the sensitivity derivatives with respect to ki
Aphatt=Ap-k*Bp*Cp;
dz=[Aphatt*z(1:m)+Bp*z(m+1);
-ki*Cp*z(1:m)-Cp*interp1(tspan,x_p,t)'+interp1(tspan,y_sp,t)
                                                    -interp1(tspan,n,t)];
```

dJ.m

```
function f=dJ(t,u,y,tspan,ts_0,ts_end)
% The integrand of the loss function |e|=|y_sp-y|
f=abs(interp1(tspan,u(ts_0:ts_end,1),t)-interp1(tspan,y(ts_0:ts_end),t));
```

dJprim.m

```
function f=dJprim1(t,u,y,Z,tspan,ts_0,ts_end);
% Description of the J'-function to be integrated (a row vector)
% -sgn(e)*delta x/delta theta_i
f=-sign(interp1(tspan,u(ts_0:ts_end,1),t)-interp1(tspan,y(ts_0:ts_end),t)).*
                                                    interp1(tspan,Z(:,1),t) ;
```