

SGA Lernmodul Nr. 3
Lineare Regelsysteme mit der Student Edition von Matlab
A. H. Glattfelder und W. Schaufelberger
Institut für Automatik, ETHZ Oktober 1995
Gekürzte, frei verfügbare Version März 1998

Inhaltsverzeichnis

1	Vorwort	2
2	Einführung	3
3	Grundlegende Darstellungen	4
3.1	Übertragungsfunktionen	4
3.2	Zustandsraumdarstellung	5
4	Blockdiagramme	5
5	Wechsel der Darstellung	6
6	Simulation und Optimierung	10
6.1	Simulation	10
6.2	Optimierung	15
7	Klassische Regelungstechnik	17
8	Moderne Regelungstechnik	23

1 Vorwort

In diesem Lernmodul soll gezeigt werden, wie Regelungstechnik mit der Student Edition von Matlab (Version 4) betrieben werden kann. Dabei werden Kenntnisse von Matlab vorausgesetzt, wie sie durch das Handbuch der Student Edition vermittelt werden. Im folgenden soll vor allem gezeigt werden, wie:

- Regelsysteme mit Matrizen (Zustandsraum) und Vektoren (Übertragungsfunktionen) beschrieben werden können
- Systemantworten durch Simulation berechnet werden können
- Resultate graphisch dargestellt werden können
- die typischen Berechnungen der klassischen Regelungstechnik (Bodediagramm, Nyquistdiagramm, Wurzelortskurven) durchgeführt werden können
- die Berechnung von Zustandsreglern durch Polfestlegung oder quadratische Optimierung (Riccati) erfolgen kann
- die symbolische Rechnung in der Regelungstechnik eingesetzt werden kann

Im Prinzip entstehen auf diese Weise sehr einfache kleine Programme, die aus dem vorliegenden Lernmodul leicht übernommen werden können. Leider muss hier festgehalten werden, dass die Student Edition nicht auf allen Maschinen einwandfrei läuft. So sind bei den Macintosh und Windows Systemen besondere Anforderungen zu erfüllen (Prozessor, FPU). Wir hoffen, dass bald durchgängig laufende Versionen erhältlich sein werden.

Die guten Help Funktionen unterstützen die Arbeit mit Matlab sehr. Es ist jederzeit leicht möglich, Information über einen Befehl oder über eine Funktion vom System zu erhalten.

2 Einführung

In diesem Lernmodul soll gezeigt werden, wie zahlreiche Grundaufgaben der Regelungstechnik mit der Student Edition gelöst werden können. Zunächst sind die wichtigsten Zusammenhänge noch einmal zusammengestellt.

Ausgangspunkt ist die Zustandsraumdarstellung (ZRD, state space Darstellung) für lineare Systeme:

$$\begin{aligned}\dot{\underline{x}}(t) &= A\underline{x}(t) + \underline{b}u(t); & \underline{x}(0) &= \underline{x}_0 \\ y(t) &= \underline{c}'\underline{x}(t) + du(t)\end{aligned}$$

mit der Lösung

$$y(t) = \underline{c}'e^{At}\underline{x}_0 + \int_0^t \underline{c}'e^{A(t-\tau)}\underline{b}u(\tau) d\tau + du(t)$$

und

$$e^{At} = \sum_{i=0}^{\infty} \frac{A^i t^i}{i!}$$

Die Zustandsgleichungen können laplaciert werden:

$$\begin{aligned}s\underline{X}(s) - \underline{x}_0 &= A\underline{X}(s) + \underline{b}U(s) \\ Y(s) &= \underline{c}'\underline{X}(s) + dU(s)\end{aligned}$$

und daraus folgt

$$Y(s) = \underline{c}'(sI - A)^{-1}\underline{x}_0 + \underline{c}'(sI - A)^{-1}\underline{b}U(s) + dU(s)$$

wobei $G(s) = \underline{c}'(sI - A)^{-1}\underline{b} + d$ als Übertragungsfunktion (Transfer function) bezeichnet wird.

Es gilt

$$G(s) = \frac{\underline{c}'\text{adj}(sI - A)\underline{b}}{\det(sI - A)} + d = \frac{z(s)}{n(s)}$$

wobei $n(s)$ ein Polynom n -ter Ordnung und $z(s)$ ein Polynom vom Grad höchstens n ist. Oft gilt $d = 0$ und damit ist der Grad von $z(s)$ höchstens $n - 1$. Das Nennerpolynom $n(\cdot)$ ist gerade das charakteristische Polynom der Matrix A . Damit gilt ebenfalls

$$G(s) = \frac{z(s)}{n(s)} = \frac{b_ms^m + b_{m-1}s^{m-1} + \dots + b_1s + b_0}{s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0}$$

sowie

$$G(s) = \frac{z(s)}{n(s)} = k \frac{(s - s_1)(s - s_2)(s - s_3) \dots (s - s_m)}{(s - p_1)(s - p_2)(s - p_3) \dots (s - p_n)}$$

mit den Nullstellen s_i und den Polen p_i (zero-pole Darstellung). Diese können reell oder konjugiert komplex und einfach oder mehrfach sein.

3 Grundlegende Darstellungen

Matlab kennt als Datenstruktur die komplexe Matrix. Diese eignet sich sehr gut für die Darstellung aller in der Regelungstechnik verwendeten Beschreibungen, insbesondere diejenige mit Übertragungsfunktionen und diejenige im Zustandsraum.

3.1 Übertragungsfunktionen

Die Darstellung besteht aus einem Zähler- und einem Nennerpolynom:

$$G(s) = \frac{z(s)}{n(s)} = \frac{b_ms^m + b_{m-1}s^{m-1} + \dots + b_1s + b_0}{s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0}$$

sowie

$$G(s) = \frac{z(s)}{n(s)} = k \frac{(s - s_1)(s - s_2)(s - s_3) \dots (s - s_m)}{(s - p_1)(s - p_2)(s - p_3) \dots (s - p_n)}$$

Dafür bietet sich in Matlab die Beschreibung mit Polynomen an. Polynome sind nach fallenden Potenzen geordnete Vektoren, d. h. für

$$P(s) = b_ms^m + b_{m-1}s^{m-1} + \dots + b_1s + b_0$$

verwendet man einen Vektor mit

```
bm=...; bmm1=...; ... b1=...;b0=...;  
p=[bm bmm1 ... b1 b0]
```

wobei den Parametern b_i vor der Benützung im Vektor Werte zugewiesen werden müssen. Nach Abmachung wird eine Übertragungsfunktion als aus zwei Polynomen bestehend angesehen. Funktionen werden deshalb im allgemeinen wie folgt definiert:

```
[num,den]=berechneIrgendEineFunktion(num1,den1)
```

Als Beispiel dient die Umwandlung aus der Übertragungsfunktion in die Pol- Nullstellendarstellung:

```
[z,p,k]=tf2zp(num,den)
```

die als dritten Parameter noch eine Verstärkung enthält. Für das Arbeiten mit Polynomen können die folgenden Befehle verwendet werden:

conv	- Convolution and polynomial multiplication.
poly	- Roots to polynomial conversion.
residue	- Partial fraction expansion.
roots	- Polynomial roots.

3.2 Zustandsraumdarstellung

Die Zustandsraumdarstellung

$$\begin{aligned}\dot{\underline{x}}(t) &= A\underline{x}(t) + \underline{b}u(t); & \underline{x}(0) &= \underline{x}_0 \\ y(t) &= \underline{c}'\underline{x}(t) + du(t)\end{aligned}$$

verlangt nach vier Matrizen A,b,c,d. Diese werden in Matlab einfach entsprechend bezeichnet und führen so direkt auf die Systembeschreibung. Mit a11=.; a12=.; ... ann=.; b1=.; etc.

```
A=[a11 a12 .. a1n;a21 a22 ... a2n;... ; an1 an2 ... ann];  
b=[b1; b2; ... bn]; % Kolonnenvektor  
c=[c1 c2 ... cn]; % Zeilenvektor  
d=0;
```

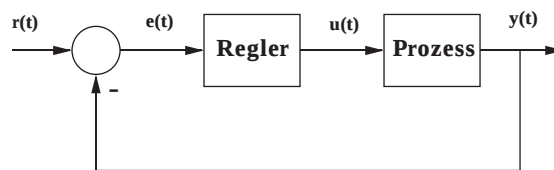
4 Blockdiagramme

Für die Verknüpfung von Systemen, wie sie zum Beispiel bei der Darstellung mit Blockdiagrammen auftreten, sind spezielle Befehle vorhanden:

append	- Append system dynamics.
cloop	- Close loops of system.
feedback	- Feedback system connection.
series	- Series system connection.
ssselect	- Select subsystem from larger system.

Mit append können zwei Systeme zu einem zusammengefasst werden, mit ssselect kann ein Teilsystem isoliert werden.

Für das Standardblockdiagramm mit Regler und Prozess (Figur 1) ergibt sich dabei die folgende Berechnung des geschlossenen Kreises :



Figur 1: Grundregelkreis

```
roots=[-1 -1 -1]; % Pole der Strecke  
nProc=[1]; dProc=poly(roots); % Zaehler (n) und Nenner (d) der Strecke  
Kp=1; Ki=1; % PI-Regler mit entsprechenden Verstaerkungen  
nCon=[Kp Ki]; dCon=[1 0]; % Reglerpolynome  
[nOpen dOpen]=series(nCon, dCon, nProc, dProc) % der offene Kreis  
[nCloop, dCloop]=cloop(nOpen, dOpen, -1) % der geschlossene Kreis
```

mit den Resultaten

nOpen = 0 0 0 1 1

dOpen = 1 3 3 1 0

nCloop = 0 0 0 1 1

dCloop = 1 3 3 2 1

Mit diesen Polynomen können nun weitere Operationen ausgeführt werden, wie zum Beispiel die Berechnung von Stoss- oder Schrittantworten etc.

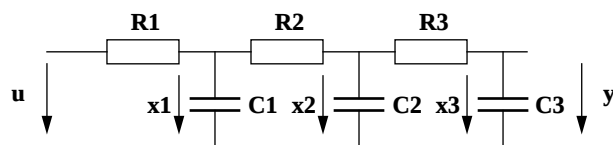
5 Wechsel der Darstellung

Der Wechsel in der Systemdarstellung wird durch die folgenden Befehle gut unterstützt.

Linear system transformations:

ss2tf	- State-space to transfer function conversion.
ss2zp	- State-space to zero-pole conversion.
tf2ss	- Transfer function to state-space conversion.
tf2zp	- Transfer function to zero-pole conversion.
zp2ss	- Zero-pole to state-space conversion.
zp2tf	- Zero-pole to transfer function conversion.

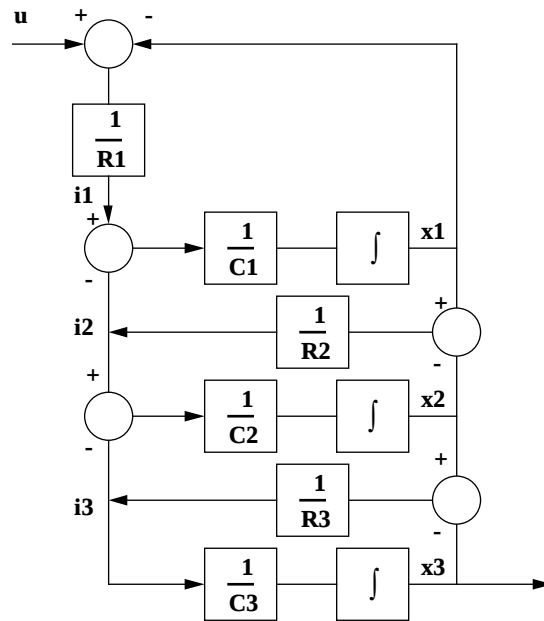
Die Benützung wird an dem folgenden kleinen Beispiel erläutert (Figur 2):



Figur 2: RC Kreis

Aus dem Schema kann sofort ein Blockdiagramm (Bild 3) mit rückwirkungsfreien Blöcken angegeben werden:

Aus dieser Darstellung sollen weitere interessierende Darstellungen ermittelt werden. Zunächst interessiert die Lage der Eigenwerte:



Figur 3: RC BlockDiagramm

Programme und Resultate	
Programme	Resultate
<pre>% Die physikalische Beschreibung R1= 1e5; R2=1e5; R3=1e5; C1= 1e-5; C2=1e-5; C3=1e-5; % aus dem Diagramm A=[-(1/R1+1/R2)/C1 1/R2/C1 0; 1/R2/C2 -(1/R2+1/R3)/C2 1/R3/C2; 0 1/R3/C3 -1/R3/C3] b=[1/R1/C1 ; 0 ; 0] c=[0 0 1] d=0;</pre>	<pre>% ergibt die Matrizen A = -2 1 0 1 -2 1 0 1 -1 b = 1 0 0 c = 0 0 1 d = 0</pre>
<pre>% Die Eigenwerte eig(A)</pre>	<pre>% liegen bei ei = -1.5550 -0.1981 -3.2470</pre>

Das System soll aus dieser Darstellung in die Übertragungsfunktion und in die Pol- Null-

stellendarstellung transformiert werden.

Programme und Resultate	
Programme	Resultate
<pre>% Uebertragungsfunktion [num, den]=ss2tf(A,b,c,d) % Pol- Nulstellendarstellung [z,p,k]=ss2zp(A,b,c,d)</pre>	<pre>% man erhaelt num = 0 0 -0.0000 1.0000 den = 1.0000 5.0000 6.0000 1.0000 z = [] p = -1.5550 -0.1981 -3.2470 k = 1</pre>

Weniger gut funktioniert die Transformation aus der Übertragungsfunktion in die Pol-Nullstellendarstellung:

Programme und Resultate	
Programme	Resultate
<pre>[num, den]=ss2tf(A,b,c,d); [z,p,k]=tf2zp(num, den)</pre>	<pre>z = 1.1259e+15 p = -3.2470 -1.5550 -0.1981 k = -8.8818e-16</pre>

Die Nullstelle bei -10^{15} weist auf numerische Probleme hin und wird durch die Verstärkung von 10^{-16} kompensiert. In solchen Fällen muss die Aufgabe auf eine andere Weise mit einem anderen Weg gelöst werden.

Zur Kontrolle erfolgt noch die Rücktransformation aus der Übertragungsfunktion in den Zustandsraum.

Programme und Resultate	
Programme	Resultate
<pre>% Ruecktransformation [A,b,c,d]=tf2ss(num,den)</pre>	<pre>% dabei ergibt sich eine neue % Darstellung A = -5.0000 -6.0000 -1.0000 1.0000 0 0 0 1.0000 0 b = 1 0 0 c = 0 -0.0000 1.0000 d = 0</pre>

Als einfaches weiteres Beispiel soll die Serieschaltung von drei Tiefpässen mit Zeitkonstante 1 berechnet werden.

Programme und Resultate	
Programme	Resultate
<pre>% Tiefpaesse num=1 den=poly([-1 -1 -1]) [A,b,c,d]=tf2ss(num,den)</pre>	<pre>% ergibt die Darstellung num = 1 den = 1 3 3 1 A = -3 -3 -1 1 0 0 0 1 0 b = 1 0 0 c = 0 0 1 d = 0</pre>

und die Transformation in die Pol- Nullstellendarstellung ergibt:

Programme und Resultate	
Programme	Resultate
<code>[z,p,k]=ss2zp(A,b,c,d)</code>	<pre>% ergibt die erwartete Darstellung, % waehrend der Weg ueber % die Uebertragungsfunktion % auch hier nicht funktioniert z = [] p = -1.0000 -1.0000 + 0.0000i -1.0000 - 0.0000i k = 1</pre>

In genau analoger Weise können die weiteren Transformationen zwischen den Darstellungen benutzt werden.

6 Simulation und Optimierung

Die Student Edition stellt einige Funktionen für die Simulation und für die Optimierung zur Verfügung. Diese sollen hier vorgestellt und an kleinen Beispielen eingesetzt werden.

6.1 Simulation

Unter Simulation versteht man die Berechnung der Lösungen von Differential- oder Differenzengleichungen, wobei die ersteren hier im Vordergrund stehen. Es soll also gezeigt werden, wie Gleichungen von der Form

$$\dot{\underline{x}}(t) = f(\underline{x}(t), t); \quad \underline{x}(0) = \underline{x}_0$$

numerisch integriert und die Lösungen dargestellt werden können. Für allgemein nichtlineare Gleichungen sind die Befehle

`ode23 ode45 - Numerical integration`

und für lineare Systeme zusätzlich der Befehl `lsim` mit den beiden Aufrufarten

```
lsim      - Continuous simulation and plot to arbitrary inputs.
[y,x] = lsim(A,b,c,d,u,t,xo)
[y,x] = lsim(num,den,u,t)
```

vorhanden. Dabei ist u der Eingangsvektor, t der Zeitvektor und x_0 der Vektor der Anfangsbedingungen. Bei `lsim` wird ein äquivalentes zeitdiskretes System für die Integration benutzt, daher muss t äquidistant sein.

Die beiden Befehle `ode23` und `ode45` (ode für ordinary differential equations) sind allgemeiner und sollen hier vor allem besprochen werden. Sie verlangen einen Aufruf von der Form:

```
[t,x]=ode23('funkt', t0, tf, x0)
```

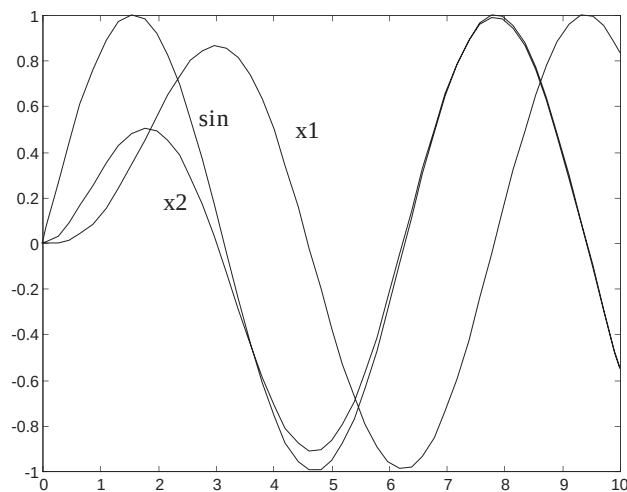
wobei t den Vektor der Zeitwerte, x den Zustand, t_0 die Anfangszeit, t_f die Endzeit und x_0 den Anfangszustand beschreiben. 'funkt' ist der Name einer m-Funktion, die die Gleichungen beschreibt. Diese hat den festen Aufbau $\dot{x} = \text{funkt}(t, x)$ mit t als skalarem Zeitwert. Als Ausgangswerte erhält man t und x , also den ganzen Zustand. Man kann demnach ein System 2. Ordnung mit einem sinusförmigen Eingangssignal wie folgt beschreiben und simulieren. Die Resultate sind in Bild 4 dargestellt.

Skript für den Aufruf und die Darstellung:

```
t0=0;tf=10;x0=[0;0];
[t,x]=ode45('SystemOrd2',t0,tf,x0);
plot(t,x); hold on;
plot(t,sin(t)); hold off;
```

m-File mit den Gleichungen:

```
function xdot = SystemOrd2(t,x);
% System zweiter Ordnung mit sinusfoermiger Anregung
% File SystemOrd2.m
xdot =[x(2); -x(1)-x(2)+sin(t)];
```



Figur 4: Antwort eines Systems 2. Ordnung auf eine sinusförmige Anregung

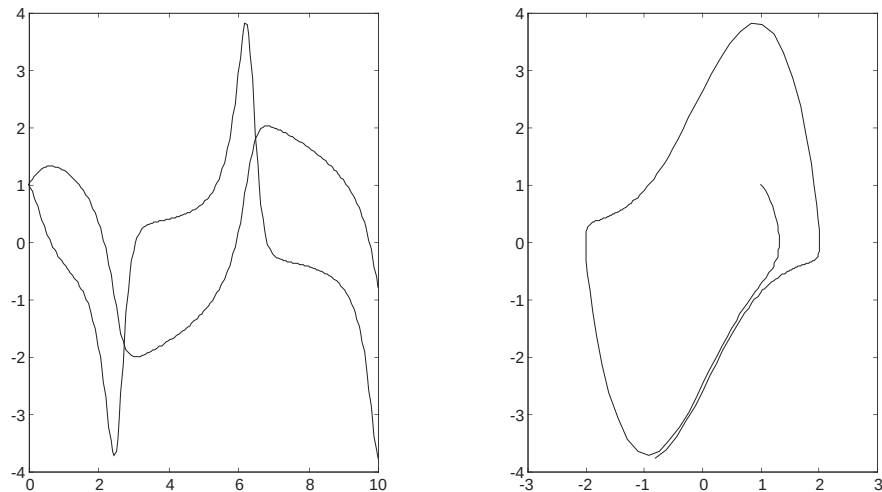
Für viele Fälle (Identifikation, Optimierung) ist es nützlich, einen Algorithmus mit fester Schrittweite zur Verfügung zu haben. Unter dem Kapitel Entwicklung von Matlabroutinen wird ein einfacher Runge-Kutta Algorithmus bereitgestellt, der hier nur eingesetzt wird. Der Aufruf wird in Analogie zu `ode23` und `ode45` gestaltet. Für die Lösung der Van der Pol Gleichung ergibt sich der folgende Aufruf

```
[t,x]=rk4('VanDerPol',0,10,0.05,[1; 1]);
subplot(1,2,1);
plot(t',x','-w');
subplot(1,2,2);
x=x';
plot(x(:,1),x(:,2),'-w');
```

mit der Routine VanDerPol:

```
function xdot=VanDerPol(t,x);
xdot= [x(2); 2*(1-x(1)^2)*x(2)-x(1)];
```

und dem Resultat im Bild 5

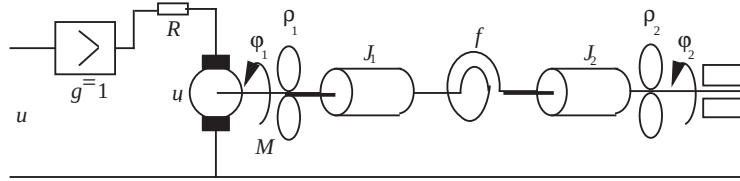


Figur 5: Lösung der Van der Pol Gleichung im Zeitbereich und in der Phasenebene

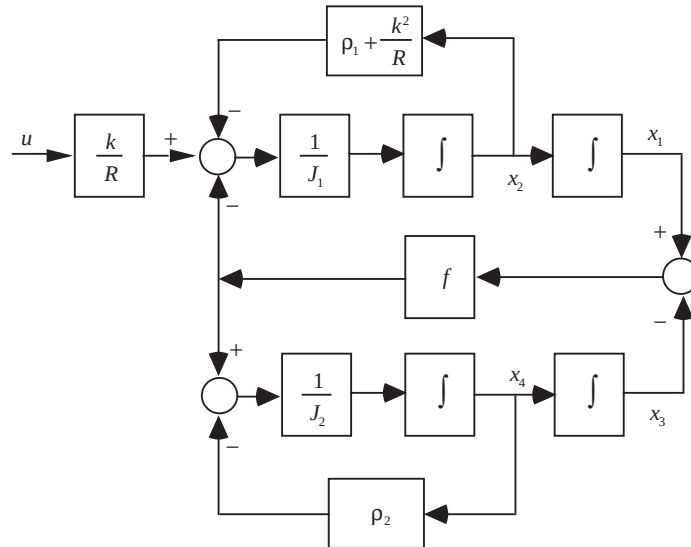
Da die Form der Beschreibung der Differentialgleichungen durch die aufzurufende Funktion fest vorgegeben ist, stellt sich die Frage der Übergabe von Parametern in die Differentialgleichung hinein. Dazu stehen zwei Techniken zur Verfügung. Die Parameter können als global deklariert werden und stehen dann auch in den Differentialgleichungen zur Verfügung, oder die Parameter werden als zusätzliche Zustände eingeführt und durch entsprechende Differentialgleichungen $\dot{p}_j = 0$ beschrieben. Die Werte können in diesen Fällen durch die Anfangsbedingungen übertragen werden. Als weiteres Beispiel soll hier ein mechanisches Servosystem nach [Jia und Schaufelberger 1995] und den Bildern 6 und 7 simuliert werden.

Für die Beschreibung des Systems erhält man etwa:

```
function xd=DCServo(t,x);
% Modell eines Servosystems mit 4 mechanischen Zuständen
% und der Federkonstante als Parameter/Zustand
```



Figur 6: Schema des Servosystems



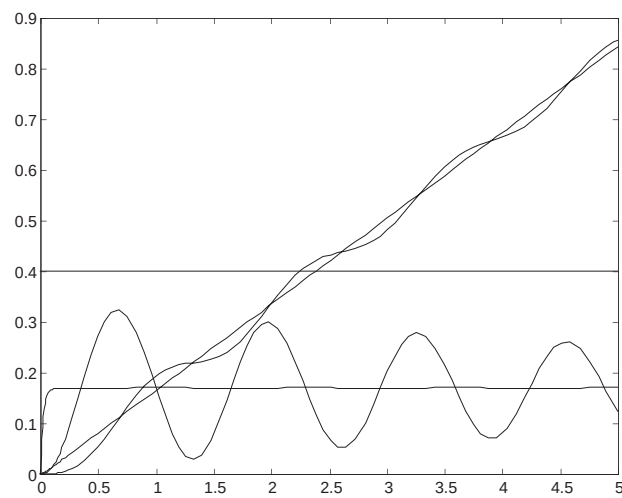
Figur 7: Blockdiagramm des Servosystems mit Zustandsgrößen

```
% physikalische Konstante
k = 0.06*60;
J1 = 0.287;
rho1 = 4.6;
J2 = 0.017;
rho2 = 0.004;
R = 1.8;
v = 0.16; % Messverstaerker
f=x(5); %f = 0.4;
% Schrittantwort
u=1;
% die Differentialgleichungen
x1d=x(2);
x2d=-f/J1*x(1) -(rho1/J1 + k*k/(R*J1))*x(2) + f/J1*x(3) + k/R/J1*u;
x3d=x(4);
x4d=f/J2*x(1) - f/J2*x(3) - rho2/J2*x(4);
x5d=0; % Federkonstante
xd=[x1d; x2d; x3d; x4d; x5d];
```

und für den Aufruf

```
% Simulation des DCServo Systems  
% mit der Federkonstante  
f=0.4;  
x0=[0; 0; 0; 0; f];  
t0=0; tf=5;  
[t,x]=ode45('DCServo',t0,tf,x0);  
plot(t,x,'-w');
```

Die fünf Zustandsgrößen zeigt Bild 8.



Figur 8: Verlauf der Zustandsgrößen

6.2 Optimierung

Für die mehrdimensionale Optimierung stellt die Student Edition die Funktion `fmins` mit dem Aufruf

```
fmins('f',x0)
```

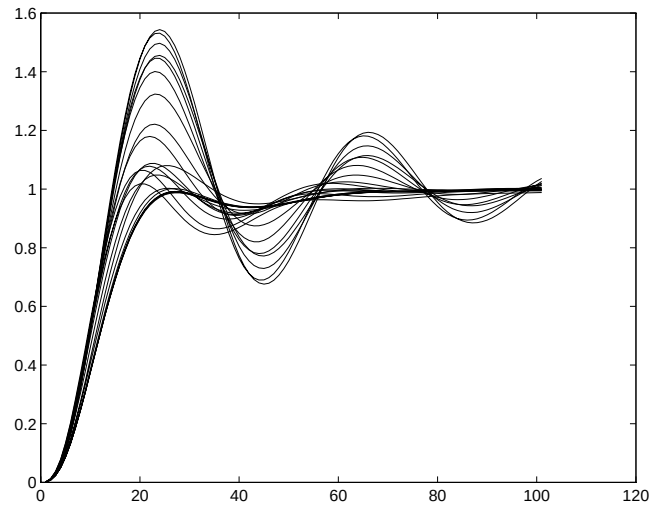
wobei `f` die zu optimierende Funktion darstellt. Damit soll ein PI-Regler optimiert werden.

```
% optimizing a classical two term controller
global qmin popt
hold off;
qmin=10000;
popt=[];
p=[]; p(1,1)=1; p(2,1)=1;
D=eye(2);
epsi=0.000001;
% choose either
fmins('twotermapi',p)
% or
% po=DavidonFP('twotermapi', p, D, epsi);
% where DavidonFP is not in the Student Edition, but can be implemented after Bazaari
```

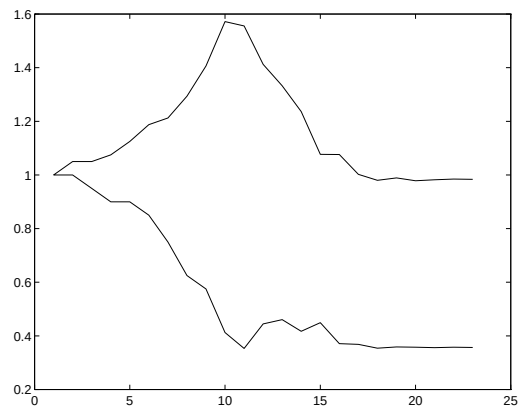
Dabei wird die folgende Zielfunktion verwendet: $q = \int_0^T (y^2 + u^2) dt + \exp(-10p)' \exp(-10p)$ wobei der letzte Term für positive Koeffizienten sorgen soll.

```
function q = twotermapi(p);
% performance index for two term controller optimization
% on a third order plant
global qmin popt
kp=p(1); ki=p(2);
A = [-1 1 0 0; 0 -1 1 0; -kp 0 -1 1; -ki 0 0 0];
b = [ 0; 0; kp; ki];
c = [ 1, 0, 0, 0; -kp 0 0 1];
d=[0;kp];
x0=[ 0; 0; 0; 0];
t0=0; tf=20; dt= 0.2;
u=ones((tf-t0)/dt+1,1);
y=rk4Abcd(A,b,c,d,x0,u,t0,tf,dt);
[n1,n2]=size(y);
ym=ones(n1,n2);
% plot(y'); hold on; plot(ym');
e=(y-ym)*(y-ym)';
q=e(1,1)+e(2,2)+exp(-10*p)'*exp(-10*p);
if q < qmin, qmin=q, plot(y(1,:),'-w'); popt=[popt, p]; hold on; end;
```

Die Resultate sind in den Bildern 9 und 10 dargestellt.:



Figur 9: Schrittantworten während der Optimierung



Figur 10: Konvergenz der Parameter (oben: k_p , unten: k_i)

7 Klassische Regelungstechnik

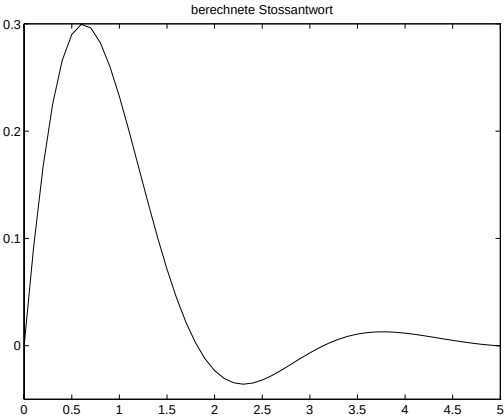
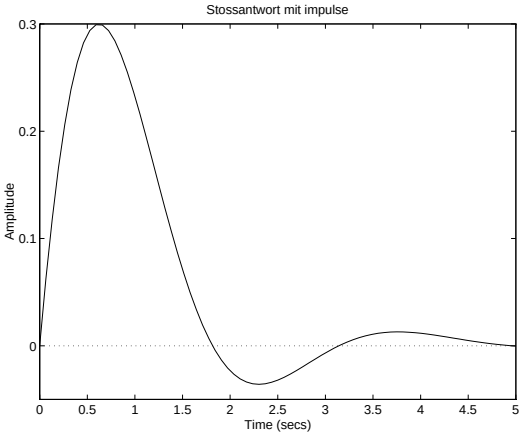
Die Student Edition von Matlab bietet ausgezeichnete Möglichkeiten für die Berechnung der an Systemen auftretenden Signale. Die wichtigsten Befehle sind:

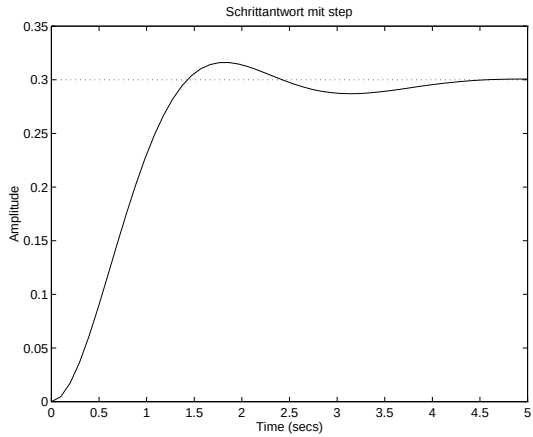
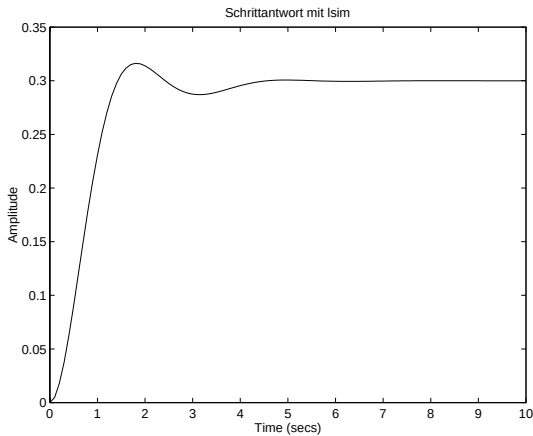
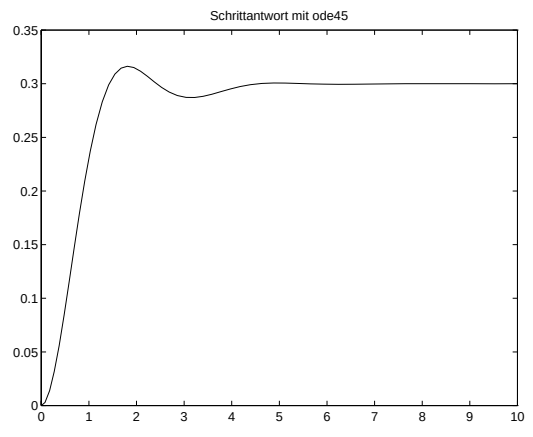
Time response:

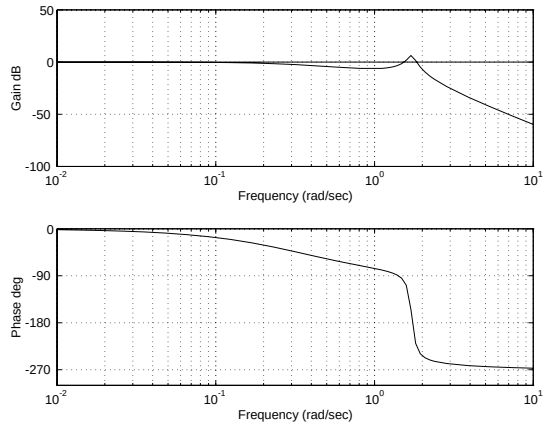
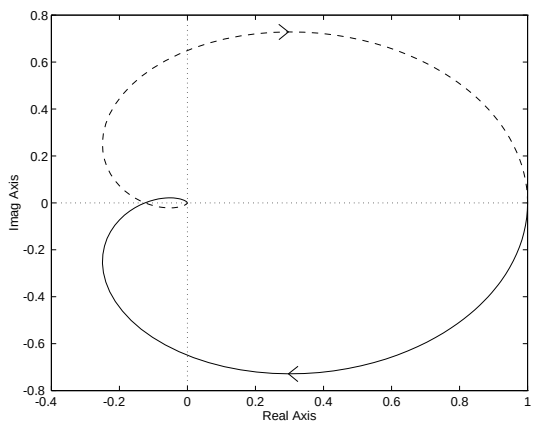
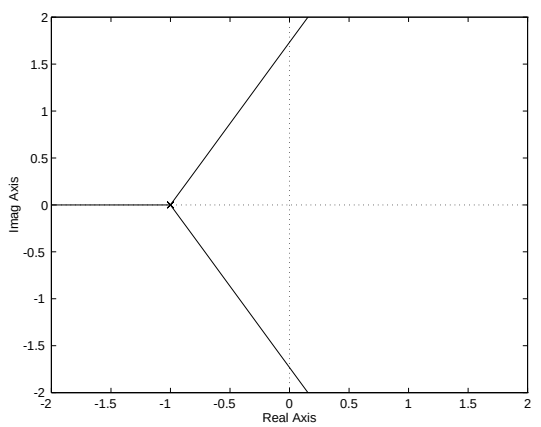
<code>impulse</code>	- Impulse response of continuous-time linear systems.
<code>lsim</code>	- Continuous simulation to arbitrary inputs.
<code>step</code>	- Step response.
<code>ode23</code> <code>ode45</code>	- Numerical integration

`lsim`, `ode23` und `ode45` wurden schon im letzten Abschnitt besprochen und benützt. Neu kommen spezielle Funktionen für die Berechnung der Schritt- und der Stossantwort hinzu. Beide kennen verschiedene Eingabeformen mit `num` und `den` oder `A`, `b`, `c`, `d` als Systembeschreibung. Die Resultate werden entweder direkt gezeichnet oder in Variablen zur Verfügung gestellt. Detaillierte Auskunft erhält man mit `Help`.

Die Stoss- und die Schrittantwort, d. h. die Reaktion eines Systems auf eine stossförmige oder schrittförmige Anregung können mit Hilfe dieser Time Response Funktionen einfach berechnet werden. Zunächst werden Matrizen mit den gewünschten Eigenschaften bereitgestellt, anschliessend werden die Stoss- und die Schrittantwort auf verschiedene Weise berechnet.

Programme und Resultate	
Programme	Resultate
<pre> % Matrizen mit den gewuenschten % Eigenschaften bereitstellen clear global Ag bg cg dg pole=[-1; -1+2*i; -1-2*i]; % nenner=poly(pole); nullst=[-1.5]; % zaehler=poly(nullst); k=1; [A,b,c,d] = zp2ss(nullst, pole, k); </pre>	<pre> % Die Resultate A = -1.0000 0 0 0.5000 -2.0000 -2.2361 0 2.2361 0 b = 1 1 0 c = 0 0 0.4472 d = 0 </pre>
<pre> % Impulsantwort berechnen % mit Hilfe der Exponentialmatrix T=0:0.1:5; y=[]; for t=T, y=[y, c*expm(A*t)*b]; end; plot(T,y); title('berechnete Stossantwort'); pause; </pre>	
<pre> % Benuetzen der Funktion impulse impulse(A,b,c,0); title('Stossantwort mit impulse'); pause; </pre>	

Programme und Resultate	
Programme	Resultate
<pre> % Benutzen der Funktion step step(A,b,c,0); title('Schrittantwort mit step'); pause; </pre>	
<pre> % Benutzen der Funktion lsim t=0:0.1:10; u=ones(size(t)); lsim(A,b,c,d,u,t); title('Schrittantwort mit lsim'); pause; </pre>	
<pre> % Benutzen von ode23 und ode45 Ag=A; bg=b; cg=c; dg=d; t0=0; tf=10; x0=[0 0 0]'; [t,x]=ode45('stepsim',t0,tf,x0); y=c*x'; plot(t,y); title('Schrittantwort mit ode45'); pause; % function xDot= stepsim(t,x); % braucht ein eigenes File global Ag bg cg dg xDot=Ag*x + bg*1; return </pre>	

Programme und Resultate	
Programme	Resultate
<pre> % System 3. Ordnung num=1 den=[1 0.5 3 1] bode(num,den) </pre>	
<pre> % 3 Tiefpaesse num=1 den=[1 3 3 1] nyquist(num,den) % Hinweis: mit % axis square erhaelt man eine % quadratische Darstellung </pre>	
<pre> % 3 Tiefpaesse num=1 den=[1 3 3 1] rlocus(num,den) </pre>	

Als Beispiel für die Anwendung der klassischen Verfahren der Regelungstechnik soll der Entwurf für eine Strecke mit zweifach integrierendem Verhalten $G(s) = \frac{1}{s^2}$ kurz diskutiert werden. Der entsprechende Entwurf eines lead-lag Reglers ist in [Glattfelder und Schaufelberger 1995]

S. 141 ff enthalten und wird hier nicht wiederholt. Für den Regler wird die folgende Übertragungsfunktion erhalten: $G_R(s) = k_p \frac{1+sT_v}{1+sT_f} \frac{1+sT_n}{sT_n}$ Die konkreten Werte können dem folgenden Listing entnommen werden:

```
% lead lag design nach der Vorlesung Lineare Regelsysteme
% fuer einen Doppelintegrator

zu=1;
nu = [1 0 0];

Tmax = 1/(pi);
alpha = 10;
Tv = Tmax*sqrt(alpha);
kp = 1/((Tmax)^2*sqrt(alpha));
Tn = (alpha/2)*Tmax;

zlead = [Tv 1];
nlead = [Tv/alpha 1];
zlag = [Tn 1];
nlag = [Tn 0];

[zR,nR]=series((kp*zlead),nlead,zlag,nlag);
[zop,nop]=series(zu,nu,zR,nR);

om=logspace(-1,2,100);

[mlead,plead,om]=bode(zlead,nlead,om);
[mlag,plag,om]=bode(zlag,nlag,om);
[mop,pop,om]=bode(zop,nop,om);

subplot(2,1,1);
loglog(om,mlead,'-w',om,mlag,'--w',om,mop,'-w');
line([1e-1,1e+1],[1.,1.]);
axis([1e-1,1e+1,1e-1,1e+1]); grid;
title('lead-lag Entwurf: Bodediagramm');
subplot(2,1,2);
semilogx(om,plead,'-w',om,plag,'--w',om,pop,'-w');
axis([1e-1,1e+1,-270, 90]);
line([1e-1,1e+1],[-180,-180]); grid;
pause;

clf;

[relead,imlead,om]=nyquist(zlead,nlead,om);
[relag,imlag,om]=nyquist(zlag,nlag,om);
[reop,imop,om]=nyquist(zop,nop,om);

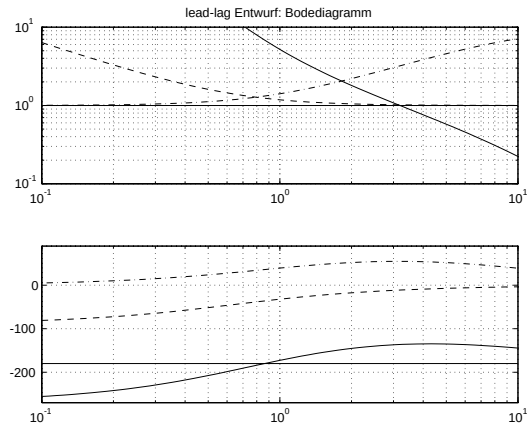
plot(relead,imlead,'w'); hold on;
```

```

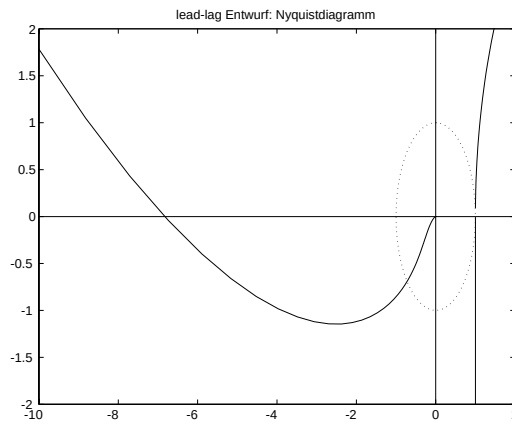
line([-10,2],[0,0]); line([0,0],[-2,2]);
t=0:0.1:6.3; plot(sin(t),cos(t),'w:');
plot(relag,imlag,'w');
plot(reop,imop,'w'); axis([-10,2,-2,2]); hold off;
title('lead-lag Entwurf: Nyquistdiagramm');

```

Die folgenden beiden Bilder 11 und 12 zeigen die Reultate.



Figur 11: Bodediagramme zum lead-lag Entwurf



Figur 12: Nyquistdiagramme zum lead-lag Entwurf, mit dem Diagramm des offenen Kreises und denjenigen des lead und des lag Elementes

8 Moderne Regelungstechnik

Für die Lösung von Aufgaben aus dem Bereich der Zustandsregelungen stehen die folgenden Funktionen zur Verfügung:

Gain selection.

- acker - SISO pole placement.
- lqe - Linear-quadratic estimator design.
- lqr - Linear-quadratic regulator design.
- place - Pole placement.

Damit können die üblichen Aufgaben aus dem Bereich der Polfestlegung und der quadratischen Optimierung gelöst werden. Als Beispiel wird hier der Entwurf einer Regelung mit Beobachter nach dem sogenannten LQG/LTR Verfahren ausgeführt. Die Technik ist in [Anderson und Moore 1990] und [Guzzella und Chapuis 1993] dargestellt. Im folgenden File können beim Entwurf r und ρ verändert werden.

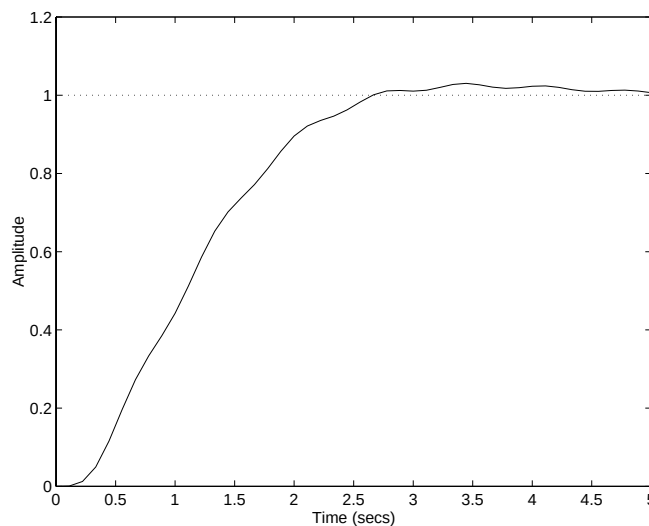
```
% robust design of servo system
% in physical variables from identification
hold off
% Beschreibung fuer unser Laborsystem
A=[ 0 1 0 0
    -42 -1.5 42 0
     0 0 0 1
     42 0 -42 0];
b=[0 3.8*pi 0 0]';
c=[0 0 1/pi 0];
d=0;
% robust design
% of controller
r=1;
Q=c'*c; % Anderson
% Q=[0.5 0 0 0;0 0 0 0;0 0 1 0;0 0 0 0]; % physical arguments
[k,S,E]=lqr(A, b, Q, r);
k
E
step(A-b*k, b, c, 0, 1); pause;
%
axis([-2 1 -1.5 1.5]);
w=logspace(-1,2,400);
% nyquist(A, b, k, 0, 1, w);
% or
[num,den]=ss2tf(A,b,k,0,1);
nyquist(num,den,w); hold on;
t=0:0.1:6.3; plot(-1+sin(t),cos(t),'w:');
axis([-2 1 -1.5 1.5]); hold off; pause;
% and of observer
rho=2.0e3;
Q=b*b'*(1+rho);
```

```

[h,S,E]=lqr(A', c', Q, 1);
h
E
step(A-h'*c, b, c, 0, 1); pause;
Ac=[A, b*k; 0*eye(4), A-b*k-h'*c];
bc=[0 0 0 0 h]';
cc=[c 0 0 0 0];
w=logspace(-1,2,100);
axis([-2 1 -1.5 1.5]);
nyquist(Ac, bc, cc, 0, 1, w); hold on;
t=0:0.1:6.3; plot(-1+sin(t),cos(t),'w:');
axis([-2 1 -1.5 1.5]); hold off; pause;
% or
[num1,den1]=ss2tf(A,b,c,0,1);
[num2,den2]=ss2tf(A-b*k-h'*c,h',k,0,1);
num=conv(num1,num2);
den=conv(den1,den2);
% nyquist(num,den,w);

```

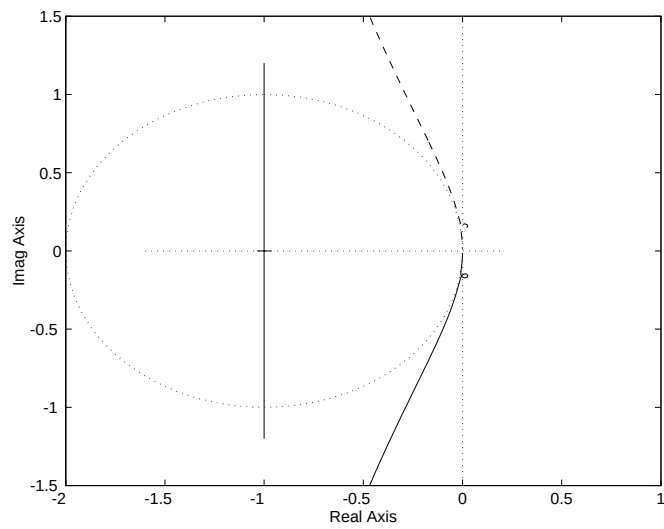
Die Entwürfe werden durch verschiedene Plots veranschaulicht. Es wird ein relativ langsamer Entwurf gezeigt ($r=1$, $\rho=2000$). Die Resultate sind in den Figuren 13, 14, 15, und 16 dargestellt.



Figur 13: Schrittantwort des geregelten Systems

Die numerischen Resultate sind

```
r=1; rho=2.0e3; % :
```



Figur 14: Nyquistdiagramm des geregelten Systems ohne Beobachter

$k = \quad 0.2554 \quad 0.1164 \quad 0.0629 \quad 0.1134$

$E =$

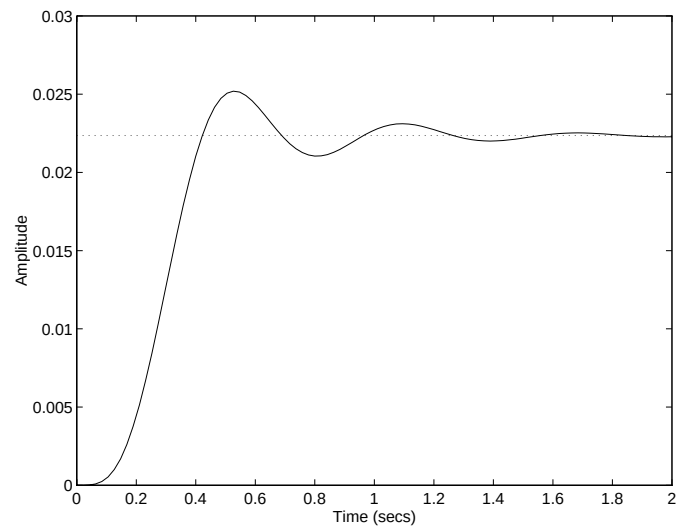
$-1.0582 + 0.8897i$
 $-1.0582 - 0.8897i$
 $-0.3865 + 9.1298i$
 $-0.3865 - 9.1298i$

$h = \quad 68.1236 \quad 4.6087 \quad 51.8106 \quad 427.2253$

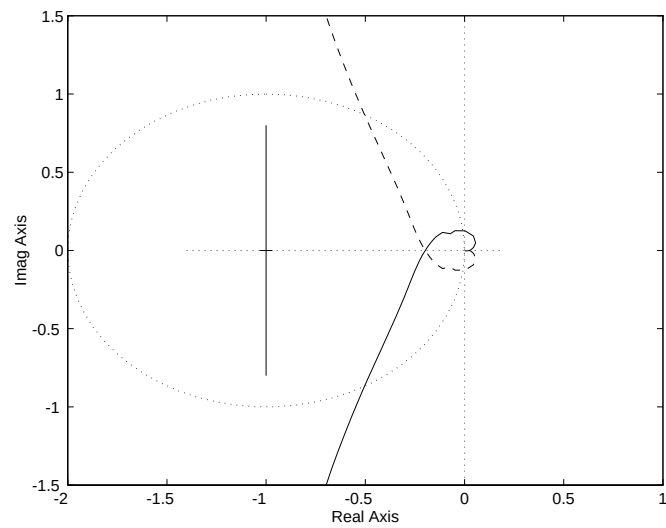
$E =$

$-6.5139 + 4.0644i$
 $-6.5139 - 4.0644i$
 $-2.4820 + 10.7213i$
 $-2.4820 - 10.7213i$

Für eine ausführliche Diskussion siehe [Jia und Schaufelberger 1995], wo auch Resultate von Laborversuchen dargestellt sind.



Figur 15: 'Schrittantwort' des Beobachters



Figur 16: 'Nyquistdiagramm' des Systems mit Beobachter

Literaturverzeichnis

- [Anderson und Moore 1990] Anderson B. D. O. and J. B. Moore (1990): *Optimal Control*. Prentice-Hall, Englewood Cliffs, N.J.
- [Bazaraa, Serali und Shetti 1993] Bazaraa M. S., H. D. Serali and C. M. Shetti (1993): *Nonlinear Programming, Theory and Algorithms*. Wiley-Interscience.
- [Char] Char B. W., K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, St. M. Watt: *Maple V, Language Reference Manual, Maple V Library Reference Manual*. Springer.
- [Glattfelder et al 1991] Glattfelder A. H., Z. H. Jiang , W. Schaufelberger (1991): *Tools for the Design of Single Loop Control Systems with Actuator Saturation applied to Hydropower Control*. IFAC Design of Control Systems, Zürich.
- [Glattfelder und Schaufelberger 1995] Glattfelder A. H. und W. Schaufelberger (1995): *Lineare Regelsysteme*. Institut für Automatik, ETHZ
- [Guzzella und Chapuis 1993] Guzzella L. und J. Chapuis (1993): *Auslegung und Implementation von praxisgerechten Zustandsreglern*, Kurs für Fachleute aus der Industrie. Institut für Automatik der ETH.
- [Jia und Schaufelberger 1995] Jia L. and W. Schaufelberger (1995): *Software for Control Engineering Education*. vdf, ETH-Zürich.
- [Linnemann 1993] Linnemann A. (1993): *Numerische Methoden für lineare Regelsysteme*. BI-Wiss.-Verlag.
- [Ludyk 1990] Ludyk G. (1990); *CAE von dynamischen Systemen, Analyse, Simulation, Entwurf von Regelsystemen*. Springer.
- [Mathworks] *The Student Edition of Matlab, Version 4, User's Guide*. Mathworks Inc. 1995
- [Page 1981] Page C. C. (1981): *Properties of Numerical Algorithms Related to Computing Controllability*. IEEE Trans. AC, Vol. AC-26, No. 1, 130-138.
- [Wolfram] Wolfram St.: *Mathematica, A System for Doing Mathematics by Computer*. Addison-Wesley.