Analysis and Design of Control Systems using CTRL-C

Wittenmark, Björn

1984

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

CODEN:LUTFD2/(TFRT-7272)/1-022/(1984)

ANALYSIS AND DESIGN OF CONTROL SYSTEMS USING CTRL-C

BJÖRN WITTENMARK

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY
JUNE 1984

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 725
S 220 07 Lund 7 Sweden

Document name
Report

Date of issue
June 1984

Document number
CODEN:LUTFD2/(TFRT-7272)/1-022/(1984)

Author(s)
Björn Wittenmark

Supervisor

Sponsoring organization

Title and subtitle

Analysis and design of control systems using CTRL-C.

Abstract

CTRL-C is a language for computer-aided of multivariable control systems and is an extension of MATLAB. In CTRL-C it is possible to create new user defined functions. This facility is used to create an improved operator communication for analysis and design of SISO systems. The intension is that the functions will be used in introductory courses in automatic control.
The report gives a short description of 20 user defined functions that can be used together with the standard CTRL-C functions.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
22

Recipient's notes

Security classification

1. INTRODUCTION

It is very important that the education in automatic control can be exemplified and illustrated using both simple examples and realistic examples. It is thus very important that a good design package is available for computer aided design of control systems. In the graduate education at the PhD-level at the Department of Automatic Control at Lund Institute of Technology we have used the package SYNPAC, see Aström (1983). SYNPAC is designed for synthesis based on state space models. It includes for instance possibilities to solve linear quadratic control problems. Also classical design problems can be solved using SYNPAC. One main drawback with the package is a quite complicated way to describe and define systems. One way to overcome this has been demonstrated in Hagander and Wittenmark (1983), where a collection of macros is given. These can be used by the unexperienced user for analysis and synthesis of control systems. This report describes another solution to the same problem. In the report the language CTRL-C is used, see the references. CTRL-C is made for computer-aided design of multivariable control systems and is an extension of MATLAB, see Moler (1980). The user interface in MATLAB and CTRL-C is good and easy to learn and use. One drawback is, however, that there is no other data structures than matrices. This has for instance the consequence that all system matrices of a state space system have to be given in the commands. Also the plotting command is quite difficult to use for the unexperienced user. In CTRL-C as in SYNPAC it is, however, possible to create macros or functions that can be used as new commands. This gives a good possibility to make a better user interface. A new set of commands has been written that can be used for analysis and design of single input single output (SISO) systems. The new functions can be collected into a library that will automatically be available when logging into CTRL-C.

The report is organized in the following way: Section 2 gives some design considerations and the limitations of the approach. Section 3 contains a description of the new functions and some examples are given in Section 4. The experience of using CTRL-C is given in Section 5. References are given in Section 6. The functions are listed in Appendix.

2. DESIGN CONSIDERATIONS

The intention with this project is to create an environment which makes it easy to solve control problems of different complexity. Special considerations are given to help unexperienced users. The price of making the system easier to use is of course a sacrifice in flexibility.

There are a couple of drawbacks with CTRL-C. First the only data structure is matrices. Second the plotting commands have a quite complex syntax. The new functions will make it easier to handle these difficulties. To make the syntax simpler only SISO systems are considered. There are several ways in which a control system can be represented. CTRL-C allows transformation between state space and transfer function forms and vice versa. There is, however, no way to connect the polynomials of the transfer function to a system description of state space form. It is then necessary to work with only one set of system descriptions. The basic system description is chosen to be the state space form

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

The system matrices A, B, C, and D can be concatenated into a system description matrix

$$S = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

This matrix will be used as the basic data structure. Since only SISO systems are considered there is no ambiguity how to split S into the matrices A, B, C, and D. Each time the transfer function is desired it is recomputed from the state space form. This will guarantee that only one description of a system is available at each instant of time.

In general the standard CTRL-C commands use the matrices A, B, C, and D. This implies that more must be typed and remembered than if a single name can be used. By making a 'preprocessor' to the standard commands it is possible to use the system description matrix instead.

To overcome the problem with the plotting command special functions have been written which automatically draw Bode and Nyquist diagrams, step, impulse and ramp responses. Further functions have been written which make it possible to draw several curves in the same diagram.

3. FUNCTION DESCRIPTIONS

User defined functions

It is easy to write new functions or macros in the CTRL-C language. A user defined function has the following structure

```
//[output1,output2,...]=function_name(input1,input2,...)
.
.   function body
.
```

Only the first four letters and/or digits of the name of the function is used for the identification of the function. If no output is generated from the function one can write

```
//[ ]= ...
```

The function can be called in different ways which are given in the manual or described in the HELP file of CTRL-C.

Summary of functions

The following functions are described:

System transformations

CASC	Cascades two systems
CLOSE	Closed loop system through output feedback
CONC	Concatenation of matrices into a system
SPLIT	Splits a system into matrices
TFSS	Transforms from transfer function to state space form
SSTF	Transforms from state space to transfer function form
STFB	Closed loop system through state feedback

Analysis

BOPL	Plots Bode diagram
NYPL	Plots Nyquist diagram
STPL	Plots step response
IPPL	Plots impulse response
RAPL	Plots ramp response
X0PL	Plots initial value response
ROLC	Plots a root locus

Synthesis

LAG	Creates lag network
LEAD	Creates lead network
PLEIG	Determines state feedback vector
PID	Creates a PID controller

Plotting

NWPL	Makes a new plot in the previous diagram
B2PL	Plots two Bode curves in the same diagram

Syntax of the functions

The syntax of each function is briefly described. The functions are given in alphabetic order.

[ ] = B2PL(fr1,fr2)

Plots two Bode diagrams in the same diagram. fr1 and fr2 are assumed to be generated by the function BOPL and contain three columns with frequency, magnitude and phase. (The frequencies do not need to be the same for fr1 and fr2.)

[fr] = BOPL(syst,ds,df)

Creates and plots the Bode diagram of the system syst in the frequency range (10**ds,10**df) rad/s. The output fr contains three columns with frequency, magnitude and phase.

[new] = CASC(first,second)

Cascades the two systems first and second giving the system new.

[csyst] = CLOSE(syst)

The system syst is closed with -1 from the output giving the system csyst.

[syst] = CONC(A,B,C,D)

Makes a concatenation of the matrices A, B, C, and D giving the system matrix

$$\text{syst} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

[yr] = IPPL(syst,int,del)

Calculates and plots the impulse response of the system syst over the time interval (0, int) with the time step del. yr will contain two columns with the time and the output.

[syst] = LAG(M,a)

Generates a lag network syst that corresponds to the transfer function

$$G(s) = M \frac{s + a}{Ms + a}$$

[syst] = LEAD(N,b)

Generates a lead network syst that corresponds to the transfer function

$$G(s) = \frac{N(s + b)}{s + bN}$$

[ ] = NWPL(y)

Makes a plot of the second column agains the first column of y in the same diagram as the last plot.

[fr] = NYPL(syst,ds,df)

Calculates and plots the Nyquist curve of the system syst in the frequency range (10**ds,10**df) rad/s. The output fr contains three columns with frequency, real part and imaginary part.

[syst]=PID(K,Ti,Td,N)

Creates the system syst corresponding to a PID controller with the transfer function

$$G(s) = K \left[1 + \frac{1}{s*Ti} + \frac{s*Td}{1 + s*Td/N} \right]$$

If Td ≤ 0 then a PI controller is obtained.

[L] = PLEIG(syst,e)

Calculates the state feedback vector L such that the eigenvalues of the system syst closed with u = -L*x is placed in the locations specified by the vector e.

[yr] = RAPL(syst,int,del)

Calculates and plots the ramp response of the system syst over the time interval (0, int) with the time step del. yr will contain two columns with time and output.

[r] = ROLC(syst,ks,n,inc)

Computes and plots the root locus of the system syst. I.e. the roots of the equation

$$A(s) + K\ B(s) = 0$$

where B(s)/A(s) is the transfer function of syst. ks is the starting value of the gain K. n roots are calculated with the increment of inc in the gain. The open loop poles and zeros are also marked in the plot.

[A,B,C,D] = SPLIT(syst)

Splits the system syst into the A, B, C, and D matrices. It is assumed that syst represents a SISO system.

[B,A] = SSTF(syst)

Generates the transfer function polynomials

$$G(s) = \frac{B(s)}{A(s)}$$

of the state space system syst.

[csyst] = STFB(syst,L,M)

Gives the closed loop system of syst and the control law u = -L*x+M*uc, where uc is the reference value.

[yr] = STPL(syst,int,del)

Calculates and plots the step response of the system syst over the time interval (0, int) with the time step del. yr will contain two columns with time and output.

[syst] = TFSS(B,A)

Generates a controllable canonical form state space description of the transfer function

$$G(s) = \frac{B(s)}{A(s)}$$

B and A must be row vectors.

[xr] = X0PL(syst,x0,int,del)

Calculates and plots the states of the system syst when the initial value of the state is x0 and when the input is zero. The time interval is (0, int) and the time step is del. xr will contain nx+1 columns with the time and the states. nx is the order of the system.

4. EXAMPLES

The use of CTRL-C and the new functions is illustrated by three examples.

Example 1

Consider the system

$$\dot{x} = \begin{bmatrix} -0.0197 & 0 \\ 0.0178 & -0.0129 \end{bmatrix} x + \begin{bmatrix} 0.0263 \\ 0 \end{bmatrix} u$$

$$y = [0 \quad 1]\, x$$

This can represent a two tank flow system where the states are the levels in

the tanks.

Determine a PI controller such that the compensated system has a cross over frequency 0.025 rad/s and a phase margin of 50 degrees. Also determine the poles and zeros of the closed loop system.

The prompt sign in CTRL-C is [>. If a command is followed by a semicolon then the result of the command is not displayed. Comments are written after //. The following session will solve Example 1:

```
[> a=[-0.0197 0; 0.0178 -0.0129];
[> b=[0.0263; 0];
[> c=[0 1];
[> d=0;
[> sys=conc(a,b,c,d)

SYS    =

  -0.0197     0.0000     0.0263
   0.0178    -0.0129     0.0000
   0.0000     1.0000     0.0000

   //
   // sys is the system description matrix in Example 1
   //
[> fr=bopl(sys,-4,-1);
   //
   // The Bode plot is shown in Fig. 4.1
   // fr contains frequency magnitude and phase
   //
[> fr(80,:)

ANS    =

   0.0248     0.5296 -113.9957

   //
   // The gain is thus 0.53 and the phase -114 at the
   // frequency 0.025. A PI controller with K=1.85 and
   // Ti=160 will give a system that fulfills the
   // specifications
   //
[> comp=pid(1.85,160,0,10);
[> csys=casc(comp,sys);
[> ccsy=close(csys);
[> bopl(csys,-4,-1);
   //
   // The Bode plot of the compensated system is shown
   // in Fig. 4.2 and the specifications are fulfilled
   //
```

```
[> stpl(ccsy,500,5);
   //
   // The step response of the closed loop system is
   // shown in Fig. 4.3.
   //
[> [a,b,c,d]=split(ccsy);
[> pole=eig(a)

POLE  =

  -0.0135 + 0.0281i
  -0.0056 - 0.0000i
  -0.0135 - 0.0281i

[>zero=tzero(a,b,c,d)

ZERO  =

  -0.0063
```

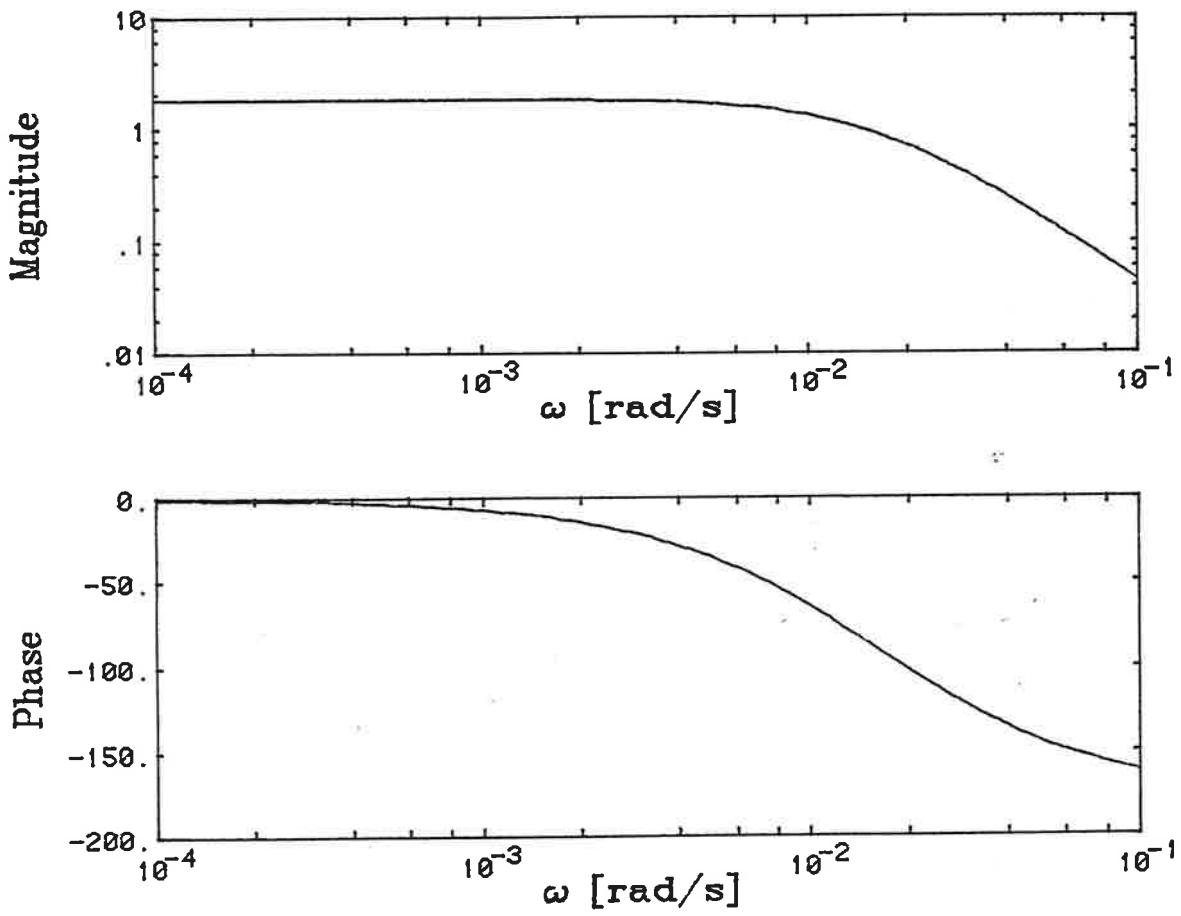

Fig. 4.1 Bode plot of the open loop system in Example 1.

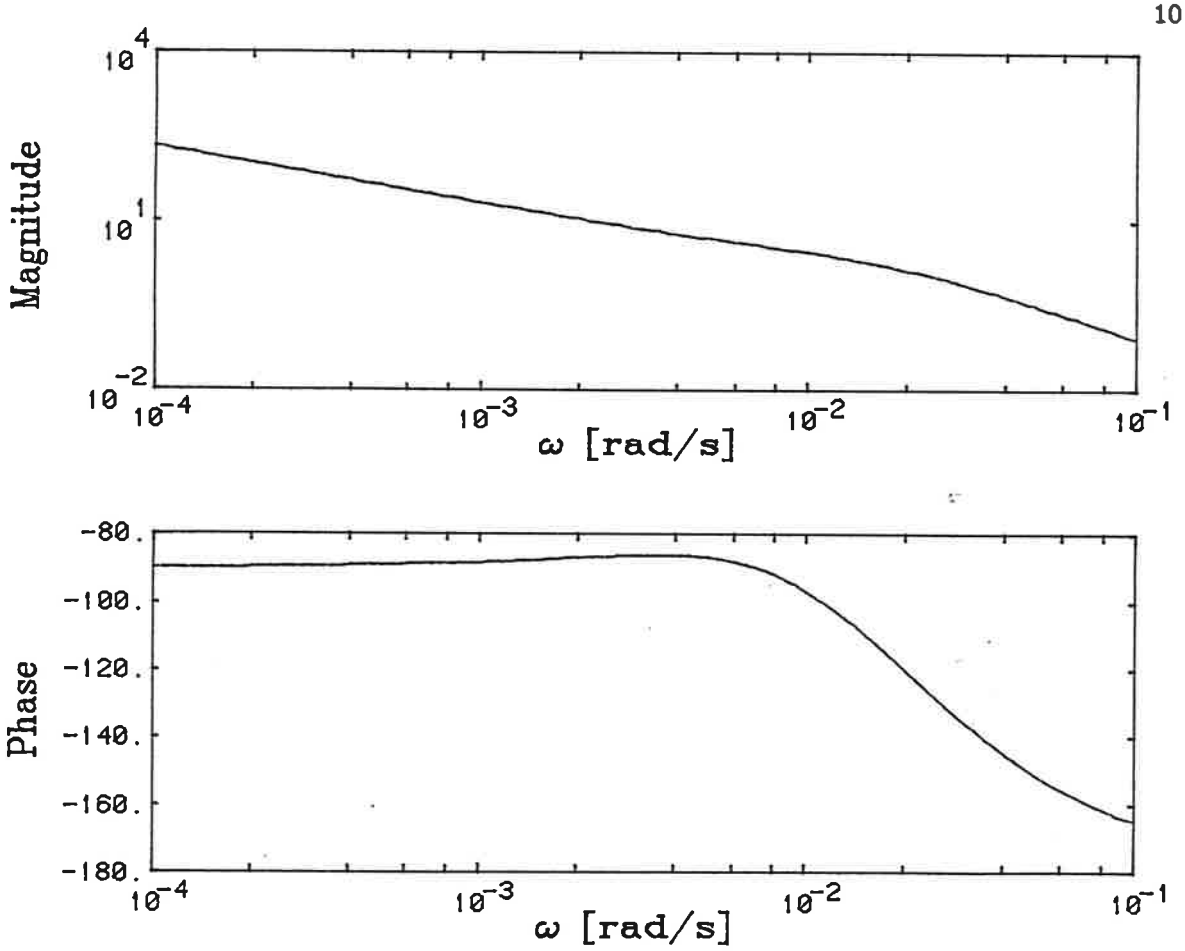

Fig. 4.2 Bode plot of the compensated open loop system in Example 1.

Example 2

Consider the system in Example 1. Determine a state feedback controller

$$u = -Lx + Mu_c$$

such that the characteristic polynomial of the closed loop system is

$$s^2 + 0.0434s + 0.000961$$

Further the steady state gain of the closed loop system should be unity.

The closed loop system has the transfer function

$$G_c(s) = C(sI-A+BL)^{-1}BM$$

Thus we get

$$M=1/[C(-A+BL)^{-1}B]$$

We can solve the problem with the following session:

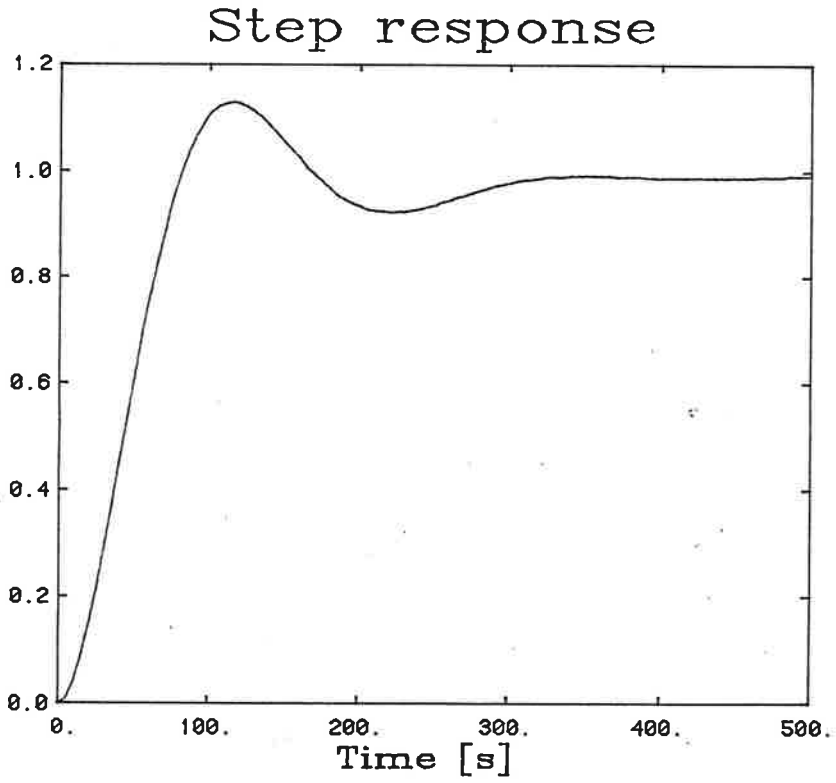

Fig. 4.3 Step response of the closed loop system in Example 1.

```
[> cpol=[1 0.0434 0.000961];
[> e=roots(cpol);
[> l=pleig(sys,e)

L      =

   0.4106     1.2124

[> [a,b,c,d]=split(sys);
[> m=1/(c*inv(-a+b*l)*b)

M      =

   2.0528

[> csys=stfb(sys,l,m);
[> stpl(csys,500,5);
   //
   // The step response of the closed loop system is
   // given in Fig. 4.4.
   //
[> [b,a]=sstf(csys);
[> long
```

```
[> a

A       =

 1.000000000000000   0.043400000000000   0.000961000000000
```

Example 3

Plot the root locus of a system with the transfer function

$$G(s) = \frac{s^3 + 7s^2 + 17s + 15}{s^3 + 2s^2 + 2s}$$

We get the following session:

```
[> a=[1 2 2 0];
[> b=[1 7 17 15];
[> roots(a)
```

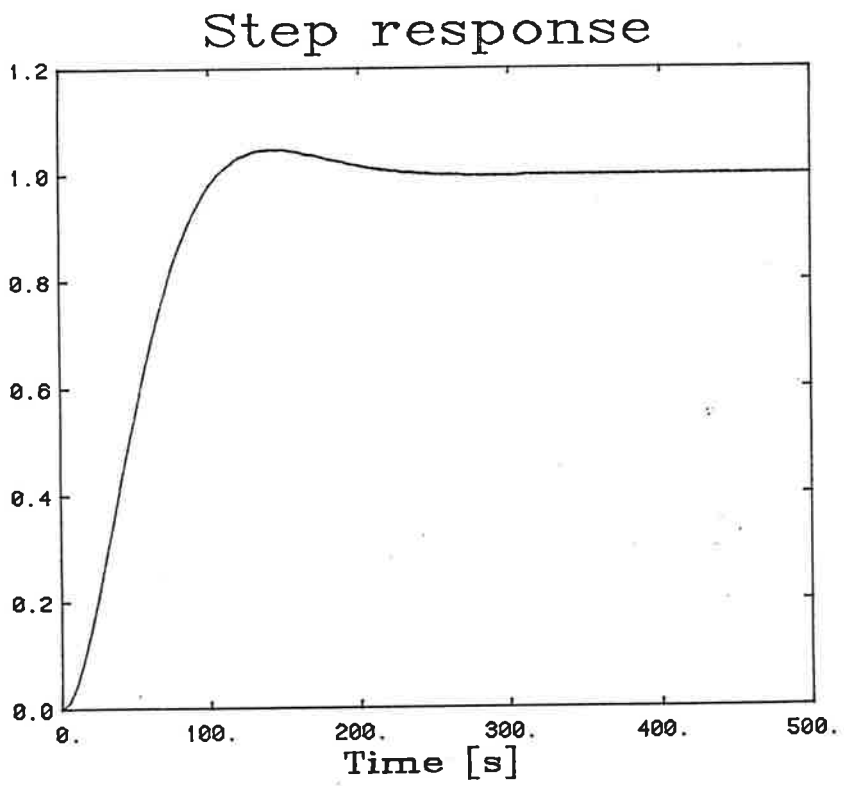

Fig.4.4 Step response of the closed loop system in Example 2.

```
ANS  =

  -1.0000 + 1.0000i
  -1.0000 - 1.0000i
   0.0000 + 0.0000i

[> roots(b)

ANS  =

  -2.0000 + 1.0000i
  -3.0000 - 0.0000i
  -2.0000 - 1.0000i

[> sys=tfss(b,a);
[> rolc(sys,0,50,0.5);
   //
   // The root locus is shown in Fig. 4.5
```

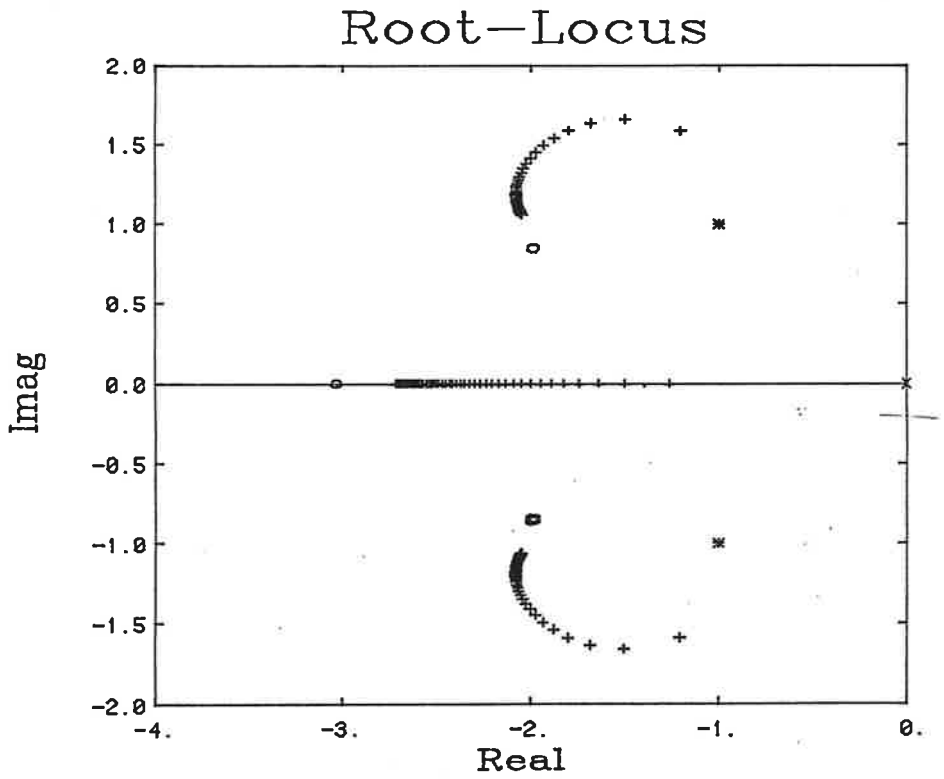

Fig. 4.5 Root locus of the system in Example 3. Each cross corresponds to one value of the gain. The open loop poles (x) and the open loop zeros (o) are also shown.

5. EXPERIENCE

The work has shown that CTRL-C is a good package to work with. It contains most of the basic routines that one would like to have in a package for analysis and design of control systems. The possibility to write user defined functions makes it easy to adapt the language to different kinds of users as has been demonstrated in this report.

There are some drawbacks of CTRL-C that limits its use:

* The only data structure is matrices. It is desirable that other data structures are available, for instance descriptions of systems.
* For documentation it is necessary that hard-copies can be made which is not the case for the moment.
* The available design methods should be extended to include polynomial design methods.
* It should be possible to handle time delays when plotting Bode and Nyquist diagrams.
* The plotting commands should be more flexible.

In summary CTRL-C has a good potential value both for educational and research purposes.

6. REFERENCES

Aström K.J. (1983): Computer aided modelling, analysis and design of control systems - A perspective, Control Systems Magazine, 3, No 2, May 1983

CTRL-C Manual (1983), Systems Control Technology, Palo Alto, CA

Hagander P, Wittenmark B. (1983): Anpac - Analysis and synthesis of continuous time systems, Report Department of Automatic Control, Lund Institute of Technology, Lund, Sweden

Moler C.B. (1980): MATLAB User's Guide, Report, Department of Computer Science, University of New Mexico

APPENDIX

```
//[ ]=b2pl(fr1,fr2)
//
//Plots two Bode diagrams in the same diagram.
//fr1 and fr2 are assumed to have been generated
//by BOPL, i.e. fr=[frequency,magnitude,phase].
//
eras;
window('211');
plot(fr1(:,1)',fr1(:,2)','loglog',fr2(:,1)',fr2(:,2)','loglog');
ylabel('Magnitude',' llllllll');
xlabel('w [rad/s]','g  lll l ');
window('212');
plot(fr1(:,1)',fr1(:,3)','logx',fr2(:,1)',fr2(:,3)','logx');
ylabel('Phase',' llll');
xlabel('w [rad/s]','g  lll l ');


//[fr]=bopl(s,ds,df)
//
//Plots the Bode diagram of the system
//s=(a,b,c,d) in the frequency range [10**ds,10**df]
//fr contains frequency, magnitude and phase.
//
[a,b,c,d]=split(s);
w=logspace(ds,df);
[ma,ph]=bode(a,b,c,d,1,w);
eras;
window('211');
plot(w,ma,'loglog');
ylabel('Magnitude',' llllllll');
xlabel('w [rad/s]','g  lll l ');
window('212');
plot(w,ph,'logx');
ylabel('Phase',' llll');
xlabel('w [rad/s]','g  lll l ');
fr=[w' ma' ph'];


//[new]=casc(first,second)
//
//Cascades the two systems first and second
//giving the system new
//
[a1,b1,c1,d1]=split(first);
[a2,b2,c2,d2]=split(second);
[a,b,c,d]=seri(a1,b1,c1,d1,a2,b2,c2,d2);
new=conc(a,b,c,d);
```

```
// [csyst]=close(syst)
//
//The system syst is closed with -1 from the
//output giving the system csyst
//
[a,b,c,d]=split(syst);
ac=a-b*c;
cc=c-d*c;
csyst=conc(ac,b,cc,d);


//[syst]=conc(a,b,c,d)
//
//Makes a concatenation of the matices a, b, c, and d
//into the system matrix syst.
//
syst=[a,b;c,d];


//[yr]=ippl(syst,int,del)
//
//Makes a plot of the impulse response
//of the system syst
//del = time increment
//int = inteval of the simulation
//
[a,b,c,d]=split(syst);
t=[0:del:int];
y=impu(a,b,c,1,t);
eras;
plot(t,y);
title('Impulse response',' llllll llllllll');
yr=[t',y'];


//[syst]=lag(m,a)
//
//Genarates a lag network syst with the transfer function
//  G(s)=M*(s+a)/(Ms+a)
//
a1=-a/m;
b1=a-a/m;
c1=1;
d1=1;
[syst]=conc(a1,b1,c1,d1);


//[syst]=lead(N,b)
//
//Creates a lead network syst with the transfer function
//  G(s)= N*(s+b)/(s+bN)
//
a1=-b*N;
b1=N*(b-b*N);
c1=1;
d1=N;
[syst]=conc(a1,b1,c1,d1);
```

```
//[ ]=nwpl(y)
//
//Makes a plot of the second column against the first
//colum of y in the same diagram as the last plot.
//
a=plot('peek');
plot(a,'scale');
t=y(:,1)';
y1=y(:,2)';
plot(t,y1);
plot('scale');


//[fr]=nypl(syst,ds,df)
//
//Calculates and plots the Nyquist curve of the
//system syst in the frequency range (10**ds,10**df).
//fr will contain frequency, real and imaginary parts.
//
[a,b,c,d]=split(syst);
w=logspace(ds,df);
[r,i]=nyqu(a,b,c,d,1,w);
eras;
plot(r,i);
fr=[w' r' i'];


//[syst]=pid(K,Ti,Td,N)
//
//Creates the system syst corresponding to a PID controller
//with the transfer function
//     G(s) = K*[1 + 1/(Ti*s) + Td*s/(1 + Td*s/N)]
//If Td<=0 then a PI controller is obtained.
//
if Td<=0, a=0; b=K/Ti; c=1; d=K;..
else a=[0 0;0 -N/Td]; b=[K/Ti;-K*N*N/Td];c=[1 1];d=K*(1+N);
syst=conc(a,b,c,d);


//[L]=pleig(syst,e)
//
//Calculates the state feedback vector L such that
//the eigenvalues of the system syst closed with
//u=-L*x is placed in the locations specified by e
//
[a,b,c,d]=split(syst);
[L]=place(a,b,e);
```

```
//[yr]=rapl(syst,int,del)
//
//Calculates and plots the ramp response of
//the system syst.
//del = time increment
//int = time interval
//
[a,b,c,d]=split(syst);
t=0:del:int;
y=ramp(a,b,c,d,1,t);
eras;
plot(t,y);
title('Ramp response',' lll llllllll');
yr=[t' y'];


//[r]=rolc(syst,ks,n,inc)
//
//Computes the root loci of the system syst=(a,b,c,d)
//  ks = starting value of gain
//  n  = number of points
//  inc= increment in the gain
//
eps=10e-6;
k=ks;
[a,b,c,d]=split(syst);
[nx,mx]=size(a);
for i=1:n, r(i,:)=eig(a-b*c*k/(1+d*k))'; k=k+inc;
pole=eig(a);
[bpol,apol]=sstf(syst);
for  i=1:nx, if abs(bpol(i))>eps, ind(i)=i; else ind(i)=nx+1;
eras;
if min(ind)>nx,..
   plot(real(r(:)),imag(r(:)),'point=0',..
        real(pole),imag(pole),'point=1');..
else..
   zero=tzero(a,b,c,d);..
   plot(real(r(:)),imag(r(:)),'point=0',..
        real(pole),imag(pole),'point=1',..
        real(zero),imag(zero),'point=9');
title('root-locus',' lll  llll');
xlabel('real',' lll'), ylabel('imag',' lll');


//[a,b,c,d]=split(syst)
//
//Splits the system matrix syst into the
//a, b, c, and d matrices.
//It is assumed that syst is a SISO system.
//
[m,n]=size(syst);
a=syst(1:n-1,1:n-1);
b=syst(1:n-1,n);
c=syst(n,1:n-1);
d=syst(n,n);
```

```
//[B,A]=sstf(syst)
//
//Generates the polynomials of the transfer function
//     G(s) = B(s)/A(s)
//of the system syst.
//
[a1,b1,c1,d1]=split(syst);
[B,A]=ss2tf(a1,b1,c1,d1,1);


//[csyst]=stfb(syst,L,M)
//
//Gives the closed loop system of syst=[a,b,c,d]
//and the control law u=-L*x+M*uc
//
[a,b,c,d]=split(syst);
ac=a-b*L;
bc=b*M;
cc=c-d*L;
dc=d*M;
[csyst]=conc(ac,bc,cc,dc);


// [ystep]=stpl(syst,int,del)
//
// Plots step response of the system s=(a,b,c,d)
// del=time increment
// int=interval of the simulation
//
[a,b,c,d]=split(syst);
t=0:del:int;
y=step(a,b,c,d,1,t);
eras;
plot(t,y);
title('Step response',' lll llllllll');
xlabel('Time [s]',' lll  l ');
ystep=[t' y'];


//[syst]=tfss(B,A)
//
//Generates a controller canonical form state space
//description of the transfer function
//     G(s) = B(s)/A(s)
//B and A must be row vectors
//
[a,b,c,d]=tf2ss(B,A);
syst=conc(a,b,c,d);
```

```
//[xr]=x0pl(syst,x0,int,del)
//
//Calculates and plots the states of the system
//syst when the initial value of the state is x0
//and when the input is zero.
//del = time increment
//int = time interval
//
[a,b,c,d]=split(syst);
t=0:del:int;
[y,x]=impu(a,x0,c,1,t);
eras;
plot(t,x);
xr=[t' x'];
```