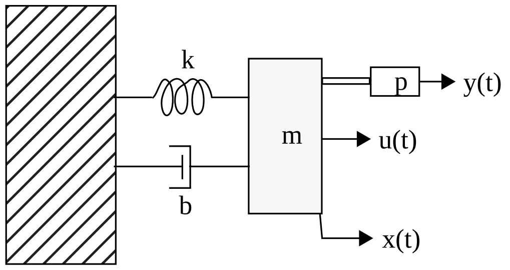
MATLAB TRAINING SESSION VI

BASIC CONTROL THEORY

System Modelling:

Linear and control system analysis and design begins with models of real systems. These models, which are mathematical representations of such things as machinery, electrical circuits, and chemical processes are used to study the dynamic response of real systems. The mathematical techniques used by MATLAB to design and analyze these systems assume processes that are physically realizable, linear, and time-invariant (LTI). Time varying or nonlinear systems either cannot be analyzed or must be approximated by LTI representations.

MATLAB uses models in the form of transfer functions or state-space equations. Either model form can be expressed in continuous-time (analog) or discrete-time (digital) forms. Transfer functions can be expressed as the ratio of two polynomials or in zero-pole-gain form. State-space system models are particularly well suited to MATLAB, since they a matrix based expression.
EXAMPLE



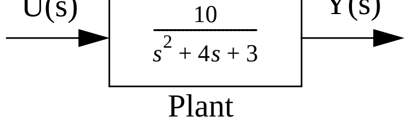
For the Transfer Function Representation of the system:

$$m\ddot{x} + b\dot{x} + kx = u(t) \qquad y(t) = px(t)$$

$$(ms^2 + bs + k)x(s) = u(s) \qquad \frac{x(s)}{u(s)} = \frac{1}{ms^2 + bs + k}$$

$$y(s) = px(s) \qquad \frac{y(s)}{x(s)} = p \qquad then \qquad \frac{y(s)}{u(s)} = \frac{p}{ms^2 + bs + k}$$

If m = 1, b = 4, k = 3, and p = 10; Then the transfer function system model is



Which can be modelled in MATLAB as:
`num = [10]; den = [1 4 3];` or as `z = []; p = [-3;-1]; k = 10;`

For the State-Space Representation of the system:

$$\textit{Let} \quad x_1 = x \quad x_2 = \dot{x}$$

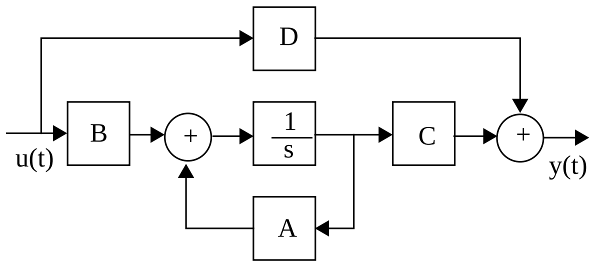
$$\textit{Then} \quad m\dot{x}_2 + bx_2 + kx_1 = u(t) \quad y(t) = px_1$$

$$\textit{Or} \quad \dot{x}_1 = x_2 \quad \dot{x}_2 = -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}u(t)$$

$$\dot{x} = Ax + Bu \qquad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y = Cx + Du \qquad y(t) = \begin{bmatrix} 10 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u(t)$$

The state-space system model is:



Which can be modelled in MATLAB as:

```
A = [0 1;-3 -4]; B = [0;1]; C = [10 0]; D = 0;
```

Model Conversion:

MATLAB has a number of functions that make it easy to convert from one model form to another and to convert continuous-time systems to discrete-time systems. These conversion functions are summarized below and more information can be obtained by typing `help` ‘function_name’ at the MATLAB command prompt:

• `c2d`	Continuous state-space to discrete state-space conversion
• `residue`	Partial fraction expansion conversion
• `ss2tf`	State-space to transfer function conversion
• `ss2zp`	State-space to zero-pole-gain conversion
• `tf2ss`	Transfer function to state-space conversion
• `tf2zp`	Transfer function to zero-pole-gain conversion
• `zp2ss`	Zero-pole-gain to state-space conversion
• `zp2tf`	Zero-pole-gain to transfer function conversion

Design and Analysis Functions:

MATLAB has several functions that are useful for designing and analyzing linear systems. These functions can be used in both continuous and discrete time systems. The basic design and analysis functions a summarized below:

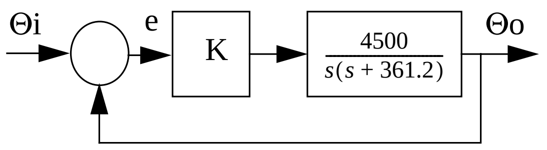
• `rlocus`	Evans root-locus plot
• `rlocfind`	Locating the system gain from pole locations indicated be the mouse
• `nyquist`	Nyquist frequency response plot
• `bode`	Log-magnitude and phase vs. frequency response plot

• `place`	Pole placement state feedback gain determination (A-BK)
• `lqr`	Linear-quadratic regulator determination for optimal control
• `lqe`	Linear-quadratic estimator determination for optimal control
• `step`	Unit step time response plot
• `lsim`	Time response simulation for generic input functions

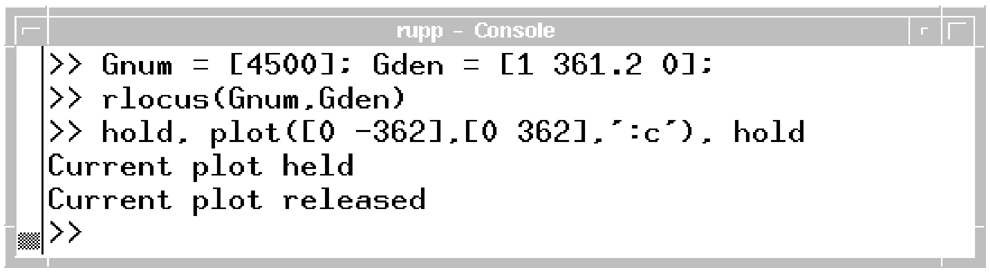
Design and Analysis Examples:

Suppose we are given the transfer function of position-control system, and since the electrical time constant is much smaller than the mechanical time constant we can approximate the open-loop system as:

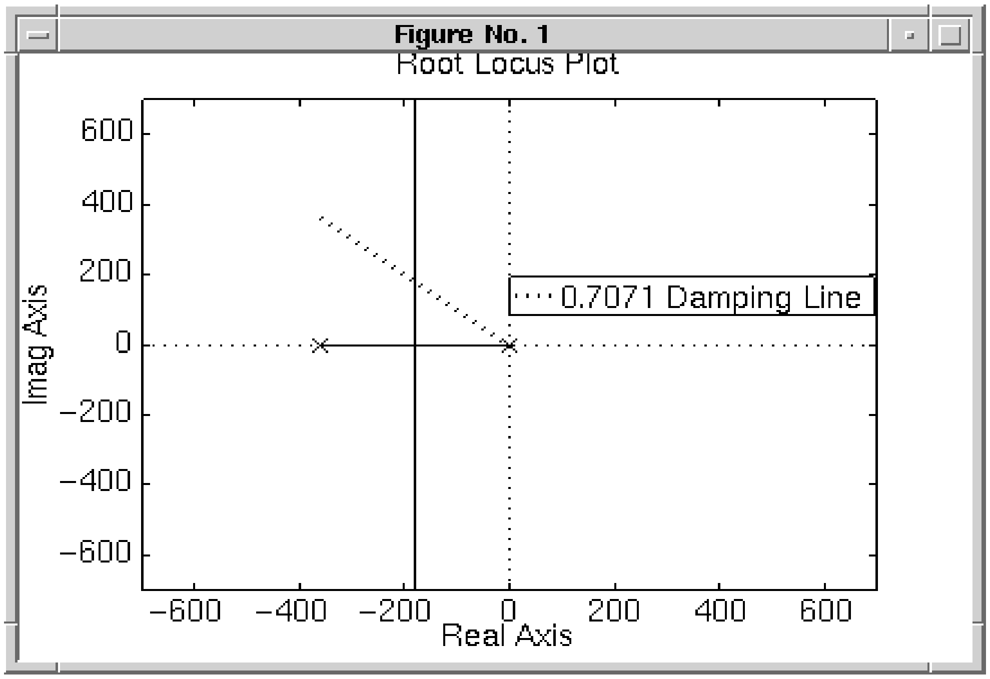
$$\frac{\Theta_o(s)}{\Theta_i(s)} = \frac{4500K}{s(s+361.2)}$$



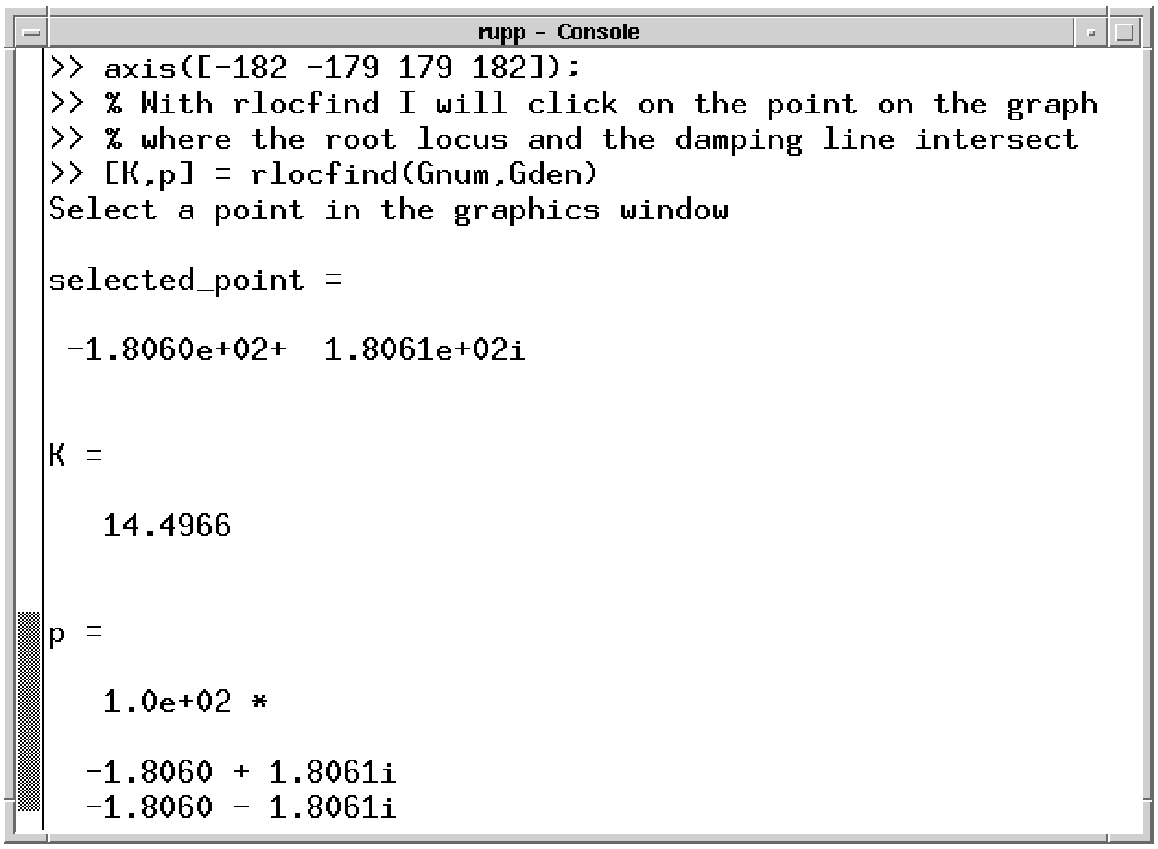
The free integrator in the denominator will ensure that there will be zero steady-state error to a step command. It is desired that the closed loop system has a damping ratio of 0.7071. We can find the value of the gain K that will achieve this damping by looking at the roots of 1 + KG(s) as K varies using the `root-locus` method. Lets use MATLAB to accomplish the design.



```
>> Gnum = [4500]; Gden = [1 361.2 0];
>> rlocus(Gnum,Gden)
>> hold, plot([0 -362],[0 362],':c'), hold
Current plot held
Current plot released
>>
```

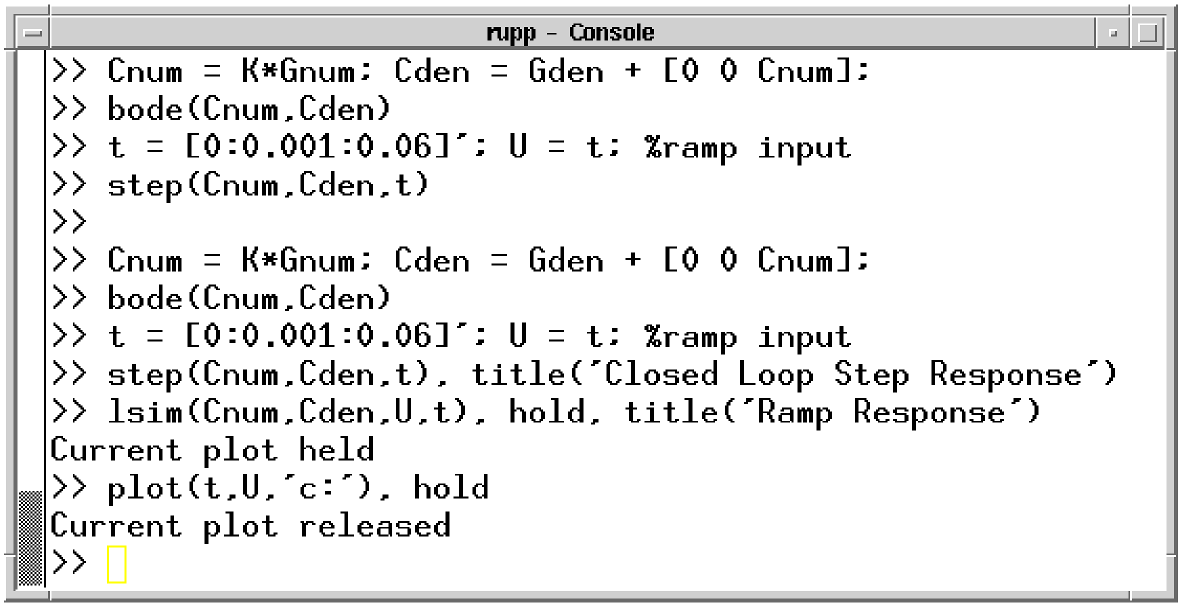


Select a good viewing window to focus on the intersection of the root-locus plot and the 0.7071 damping line (i.e. the slope of the line m = $\cos^{-1}(x) = 45$).



```
>> axis([-182 -179 179 182]);
>> % With rlocfind I will click on the point on the graph
>> % where the root locus and the damping line intersect
>> [K,p] = rlocfind(Gnum,Gden)
Select a point in the graphics window

selected_point =

 -1.8060e+02+  1.8061e+02i


K =

   14.4966


p =

   1.0e+02 *

  -1.8060 + 1.8061i
  -1.8060 - 1.8061i
```

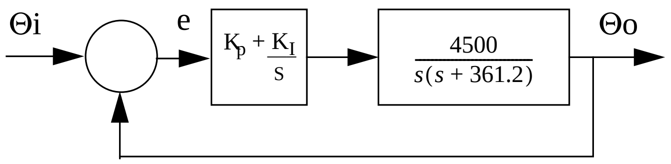
Which gives the desired gain to be approximately 14.5. Lets close the loop and test the system. Remember $G_{cl}(s) = G_{num}(s)/(G_{den}(s)+G_{num}(s))$



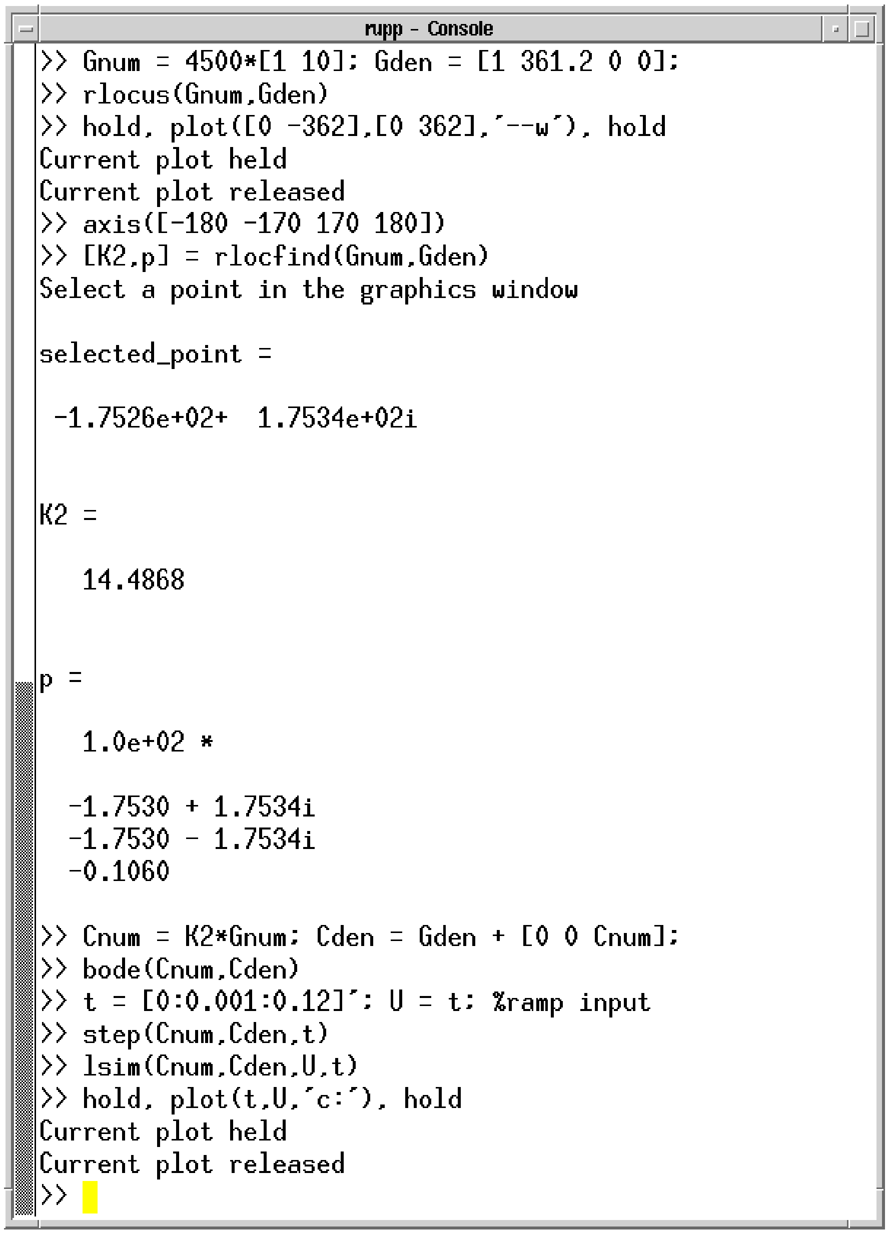
```
>> Cnum = K*Gnum; Cden = Gden + [0 0 Cnum];
>> bode(Cnum,Cden)
>> t = [0:0.001:0.06]'; U = t; %ramp input
>> step(Cnum,Cden,t)
>>
>> Cnum = K*Gnum; Cden = Gden + [0 0 Cnum];
>> bode(Cnum,Cden)
>> t = [0:0.001:0.06]'; U = t; %ramp input
>> step(Cnum,Cden,t), title('Closed Loop Step Response')
>> lsim(Cnum,Cden,U,t), hold, title('Ramp Response')
Current plot held
>> plot(t,U,'c:'), hold
Current plot released
>>
```

Now, suppose we need zero steady state error for ramp inputs. We need to add integral action to the controller. The model be comes:

$$\frac{\Theta_o(s)}{\Theta_e(s)} = \frac{4500K_p(s + K_I/K_p)}{s^2(s + 361.2)}$$



Applying Routh's test to the system indicates that the system is stable for $0 < K_I < 361.2K_p$. This means that the zero of G(s) at $s = -K_I/K_p$ cannot be placed too far into the left hand plane without destabilizing the system. A viable method for designing a PI controller is to place the zero near the origin and away from the most significant pole of G(s) at s = -361.2. If the zero is too close to the origin we will generate a slow pole and hence large settling time and too far will produce a zero that causes large overshoot. A good compromise is to put the zero at $s = -K_I/K_p$ at -10. Then the design and analysis is similar to the session above.


rupp - Console

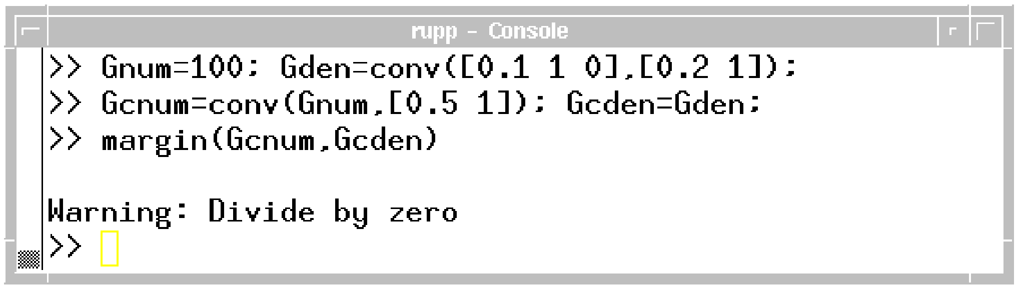
```
>> Gnum = 4500*[1 10]; Gden = [1 361.2 0 0];
>> rlocus(Gnum,Gden)
>> hold, plot([0 -362],[0 362],'--w'), hold
Current plot held
Current plot released
>> axis([-180 -170 170 180])
>> [K2,p] = rlocfind(Gnum,Gden)
Select a point in the graphics window

selected_point =

 -1.7526e+02+  1.7534e+02i


K2 =

   14.4868


p =

   1.0e+02 *

  -1.7530 + 1.7534i
  -1.7530 - 1.7534i
  -0.1060

>> Cnum = K2*Gnum; Cden = Gden + [0 0 Cnum];
>> bode(Cnum,Cden)
>> t = [0:0.001:0.12]'; U = t; %ramp input
>> step(Cnum,Cden,t)
>> lsim(Cnum,Cden,U,t)
>> hold, plot(t,U,'c:'), hold
Current plot held
Current plot released
>>
```

Another approach to control design is the frequency response method. Lets us design a PID controller for the following servo-system:

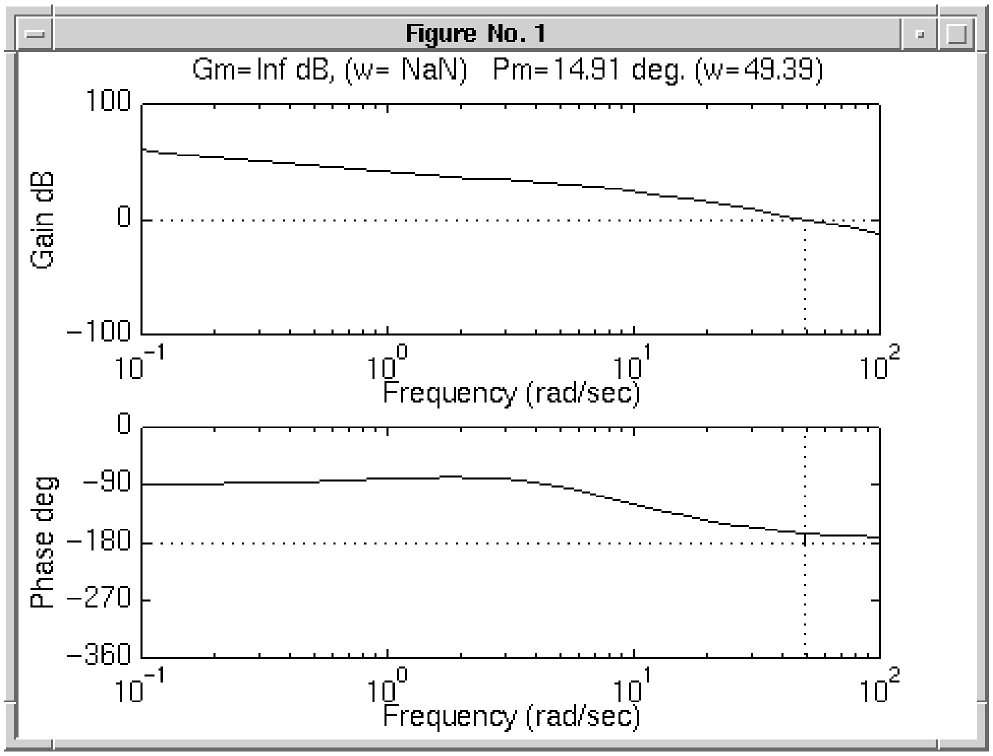
$$G(s) = \frac{100}{s(0.1s+1)(0.2s+1)}$$

$$G_c(s) = K_P + K_D s + \frac{K_I}{s} = (1 + K_{D1}s)\left(K_{P2} + \frac{K_{I2}}{s}\right)$$

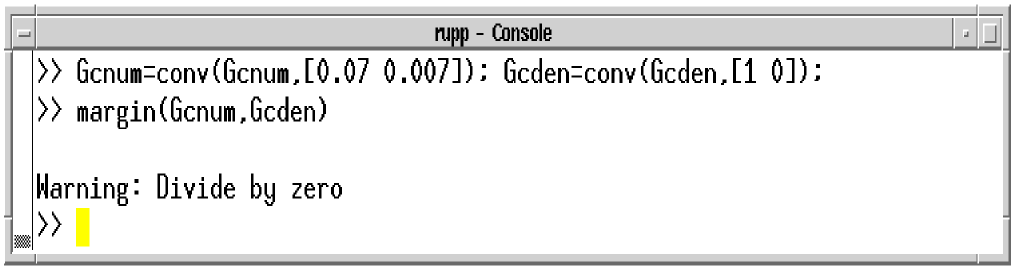
Without compensation the gain margin is -16.5 dB so the loop cannot be closed before compensation. The performance specifications state that the ramp error constant K_v is to be infinite and the phase margin should be greater than 45°. We shall first design the PD portion of the controller. After plotting many bode diagrams it is found that the best compromise between the phase margin and the open-loop bandwidth of the system is when K_{D1} = 0.5. So lets augment the system $G_c(s)$ with (1+0.5s) and find the new phase margin.



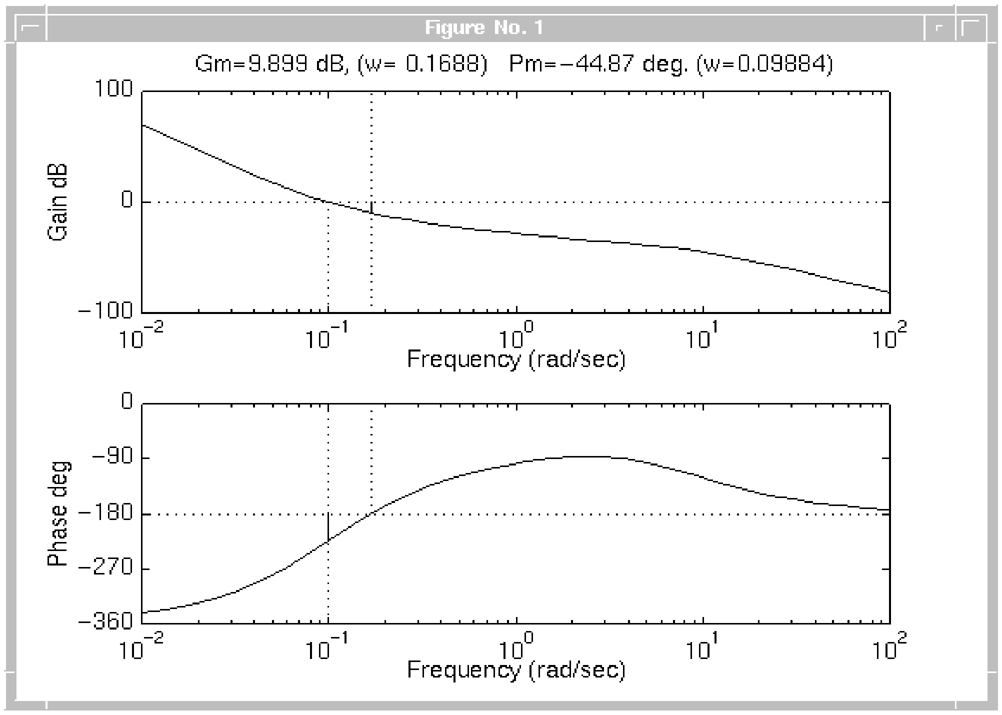
```
>> Gnum=100; Gden=conv([0.1 1 0],[0.2 1]);
>> Gcnum=conv(Gnum,[0.5 1]); Gcden=Gden;
>> margin(Gcnum,Gcden)

Warning: Divide by zero
>> 
```

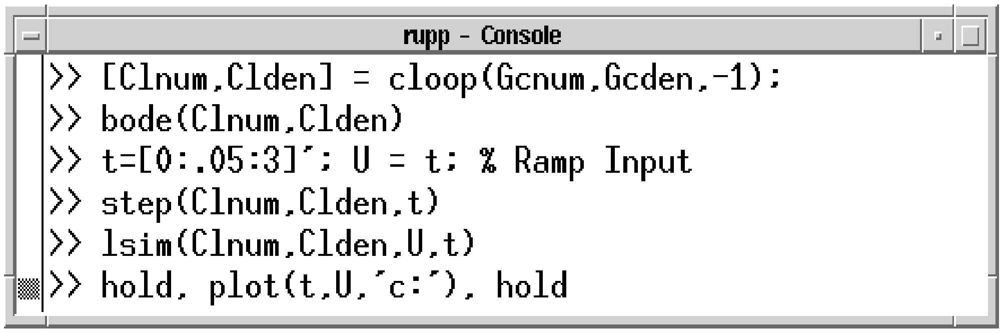


We see that the PD compensated open-loop system has a new phase margin of 14.9°. To realize a phase margin of 45°, without affecting the phase curve, the gain-crossover frequency should be approximately 13 rad/sec, at which point the magnitude curve has a gain of 20 dB. Thus, the PI controller must provide an attenuation of -20 dB at the same time it introduces another free integrator that will give us an infinite ramp error constant. Lets add a -3 dB factor of safety to the attenuation, since the phase curve will inevitably be degraded by the phase of the PI controller. The value of K_{P2} is determined as 10^(-23/20) = 0.07 and K_{I2} is chosen one decade below the required corner frequency (13/10)*$K_{P2} \approx 0.007$. Thus, the parameters of the PID controller are determined to be $K_P = 0.0735$, $K_D = 0.035$, and $K_I = 0.007$.


rupp - Console

```
>> Gcnum=conv(Gcnum,[0.07 0.007]); Gcden=conv(Gcden,[1 0]);
>> margin(Gcnum,Gcden)

Warning: Divide by zero
>>
```




rupp - Console

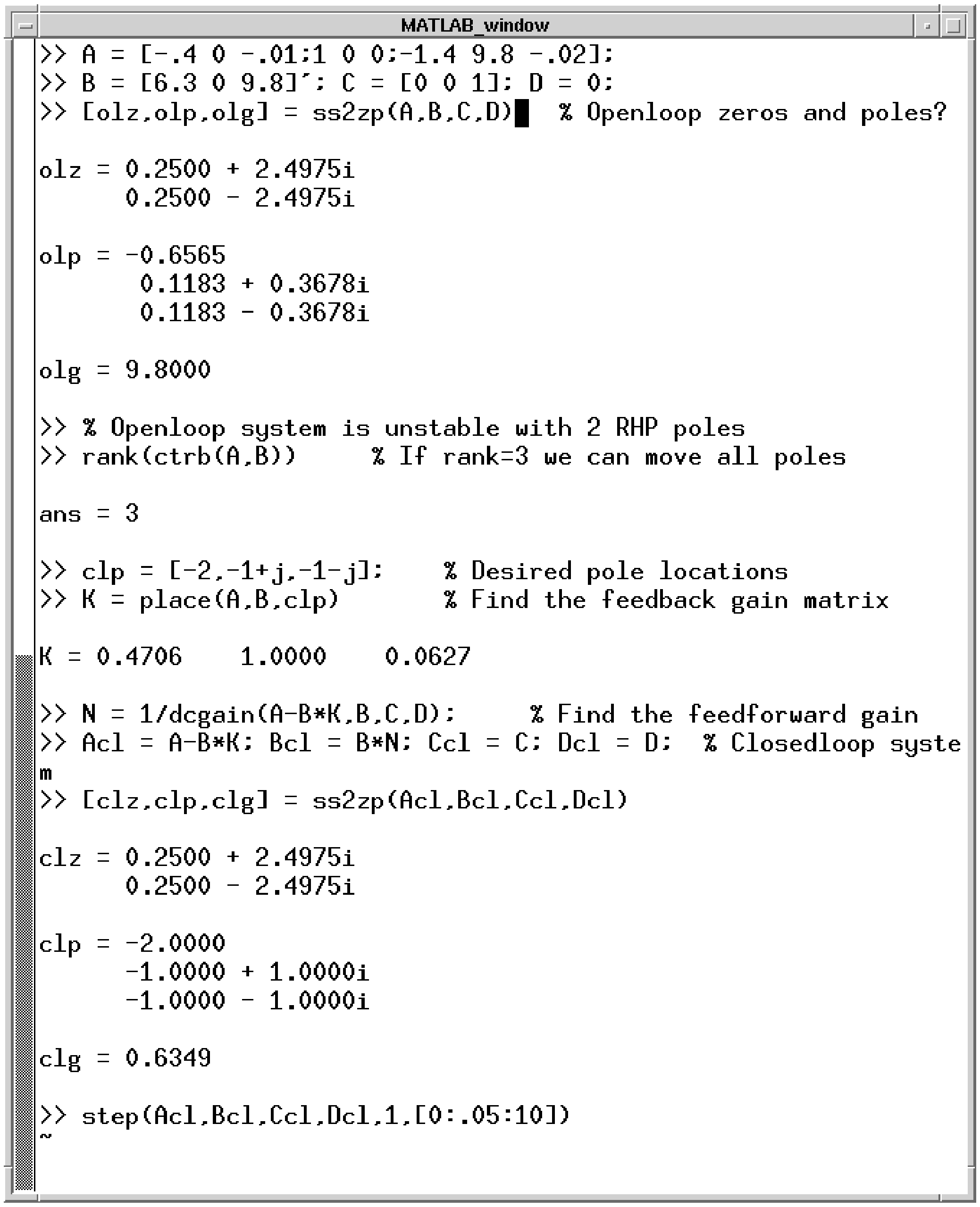
```
>> [Clnum,Clden] = cloop(Gcnum,Gcden,-1);
>> bode(Clnum,Clden)
>> t=[0:.05:3]'; U = t; % Ramp Input
>> step(Clnum,Clden,t)
>> lsim(Clnum,Clden,U,t)
>> hold, plot(t,U,'c:'), hold
```

State Space Design:

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

State Feedback (Pole Placement) Regulator Design:
The task is to find the state feedback gain matrix K (u = -Kx + Nv) to move the open loop poles to stabilize and or prescribe the relative degree of stability of the closed loop system. The ability to assign the location of the closed loop poles is dependent on the controllability or stabilizability of the pair (A,B). If the uncontrollable modes of the system are stable we can use state feedback to stabilize the system. Lets look at:

$$\dot{x} = \begin{bmatrix} -0.4 & 0 & -0.01 \\ 1 & 0 & 0 \\ -1.4 & 9.8 & -0.02 \end{bmatrix} x + \begin{bmatrix} 6.3 \\ 0 \\ 9.8 \end{bmatrix} u \qquad y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x$$

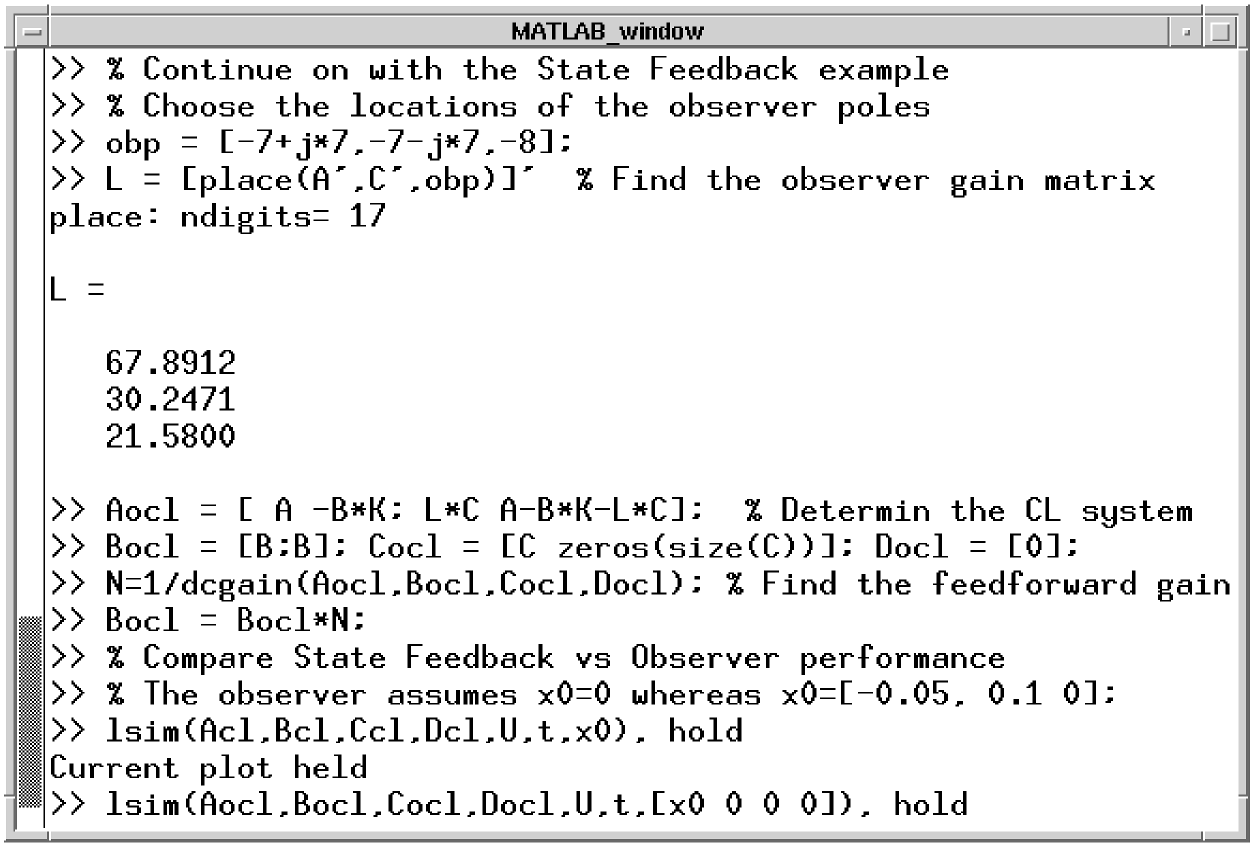


```
>> A = [-.4 0 -.01;1 0 0;-1.4 9.8 -.02];
>> B = [6.3 0 9.8]'; C = [0 0 1]; D = 0;
>> [olz,olp,olg] = ss2zp(A,B,C,D)█  % Openloop zeros and poles?

olz = 0.2500 + 2.4975i
      0.2500 - 2.4975i

olp = -0.6565
       0.1183 + 0.3678i
       0.1183 - 0.3678i

olg = 9.8000

>> % Openloop system is unstable with 2 RHP poles
>> rank(ctrb(A,B))     % If rank=3 we can move all poles

ans = 3

>> clp = [-2,-1+j,-1-j];    % Desired pole locations
>> K = place(A,B,clp)       % Find the feedback gain matrix

K = 0.4706    1.0000    0.0627

>> N = 1/dcgain(A-B*K,B,C,D);     % Find the feedforward gain
>> Acl = A-B*K; Bcl = B*N; Ccl = C; Dcl = D;  % Closedloop syste
m
>> [clz,clp,clg] = ss2zp(Acl,Bcl,Ccl,Dcl)

clz = 0.2500 + 2.4975i
      0.2500 - 2.4975i

clp = -2.0000
      -1.0000 + 1.0000i
      -1.0000 - 1.0000i

clg = 0.6349

>> step(Acl,Bcl,Ccl,Dcl,1,[0:.05:10])
~
```

Observer Design:
In a typical system with many states, it is not possible or practical to sense all the states and feed them back. Any practical compensator must rely on measured output and inputs for feedback. Even though we do not have access to all the system states can can estimate them:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x} - Du) \qquad \hat{x}(0) = 0$$

If the pair (A,C) is observable then we can design an observer to asymptotically estimate all the states. Lets continue with the previous state feedback example:


MATLAB_window

```
>> % Continue on with the State Feedback example
>> % Choose the locations of the observer poles
>> obp = [-7+j*7,-7-j*7,-8];
>> L = [place(A',C',obp)]'  % Find the observer gain matrix
place: ndigits= 17

L =

   67.8912
   30.2471
   21.5800

>> Aocl = [ A -B*K; L*C A-B*K-L*C];  % Determin the CL system
>> Bocl = [B;B]; Cocl = [C zeros(size(C))]; Docl = [0];
>> N=1/dcgain(Aocl,Bocl,Cocl,Docl); % Find the feedforward gain
>> Bocl = Bocl*N;
>> % Compare State Feedback vs Observer performance
>> % The observer assumes x0=0 whereas x0=[-0.05, 0.1 0];
>> lsim(Acl,Bcl,Ccl,Dcl,U,t,x0), hold
Current plot held
>> lsim(Aocl,Bocl,Cocl,Docl,U,t,[x0 0 0 0]), hold
```

Linear Quadratic Regulators:
Suppose we want to design an optimal regulator to control a system that consists of an inverted pendulum on a movable base. The purpose of the regulator will be to move the base such that the pendulum remains vertical, much like balancing a stick on your finger. The four states of the system are the position and velocity of the base and the angular position and velocity of the pendulum. The state-space equations and the control law for the system is:

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

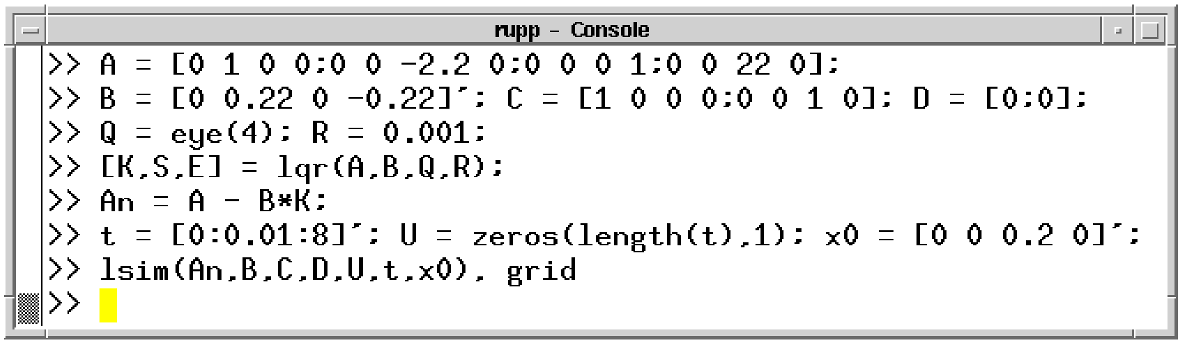
The Cost Function is:

$$J[u] = \int_0^\infty (x^T Q x + u^T R u)dt$$

Use lqr to calculate the feedback gain constant K assuming:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -2.2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 22 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0.22 \\ 0 \\ -0.22 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 0.001 \end{bmatrix}$$

The MATLAB statements are:



```
rupp - Console
>> A = [0 1 0 0;0 0 -2.2 0;0 0 0 1;0 0 22 0];
>> B = [0 0.22 0 -0.22]'; C = [1 0 0 0;0 0 1 0]; D = [0;0];
>> Q = eye(4); R = 0.001;
>> [K,S,E] = lqr(A,B,Q,R);
>> An = A - B*K;
>> t = [0:0.01:8]'; U = zeros(length(t),1); x0 = [0 0 0.2 0]';
>> lsim(An,B,C,D,U,t,x0), grid
>> 
```

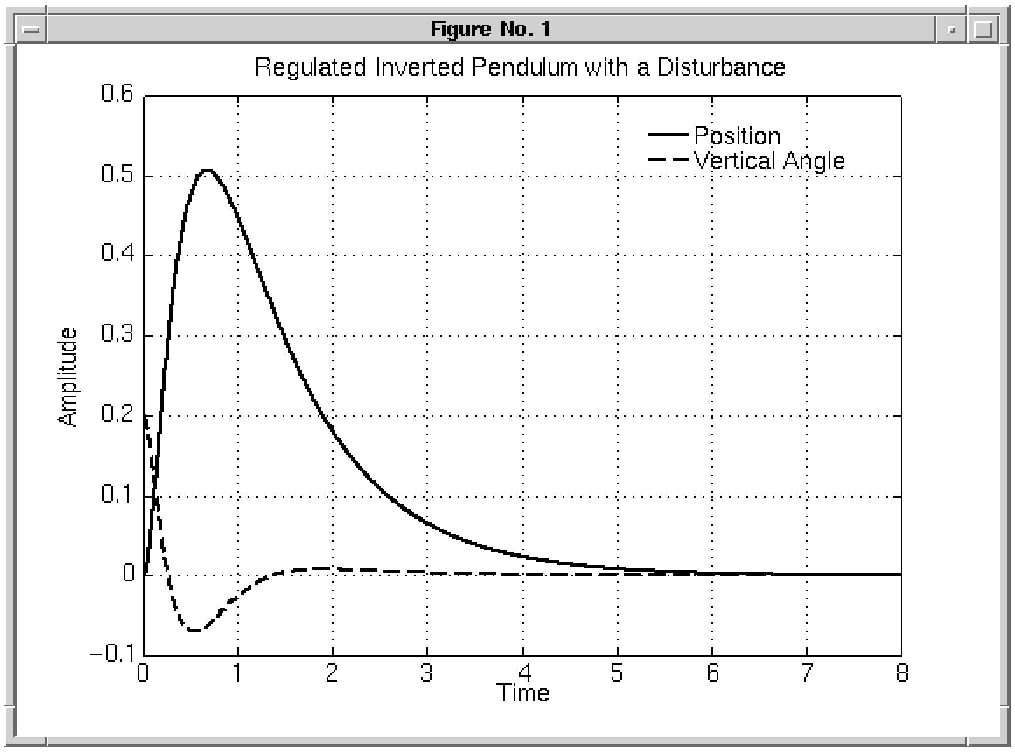


Figure No. 1
Regulated Inverted Pendulum with a Disturbance
Position
Vertical Angle
Amplitude
Time