AUTOMATIC CONTROL ENGINEERING

HINCHEY

CONTROLS NOTES

FEEDBACK CONTROL CONCEPT

ZIEGLER NICHOLS GAINS

CONTROL SYSTEM SIMULATION

CONTROL SYSTEM STABILITY

CONTROL SYSTEM PERFORMANCE

SWITCHING CONTROLLERS

DIGITAL CONTROL

STATE SPACE CONTROL

SAMPLE TESTS

AUTOMATIC CONTROL ENGINEERING

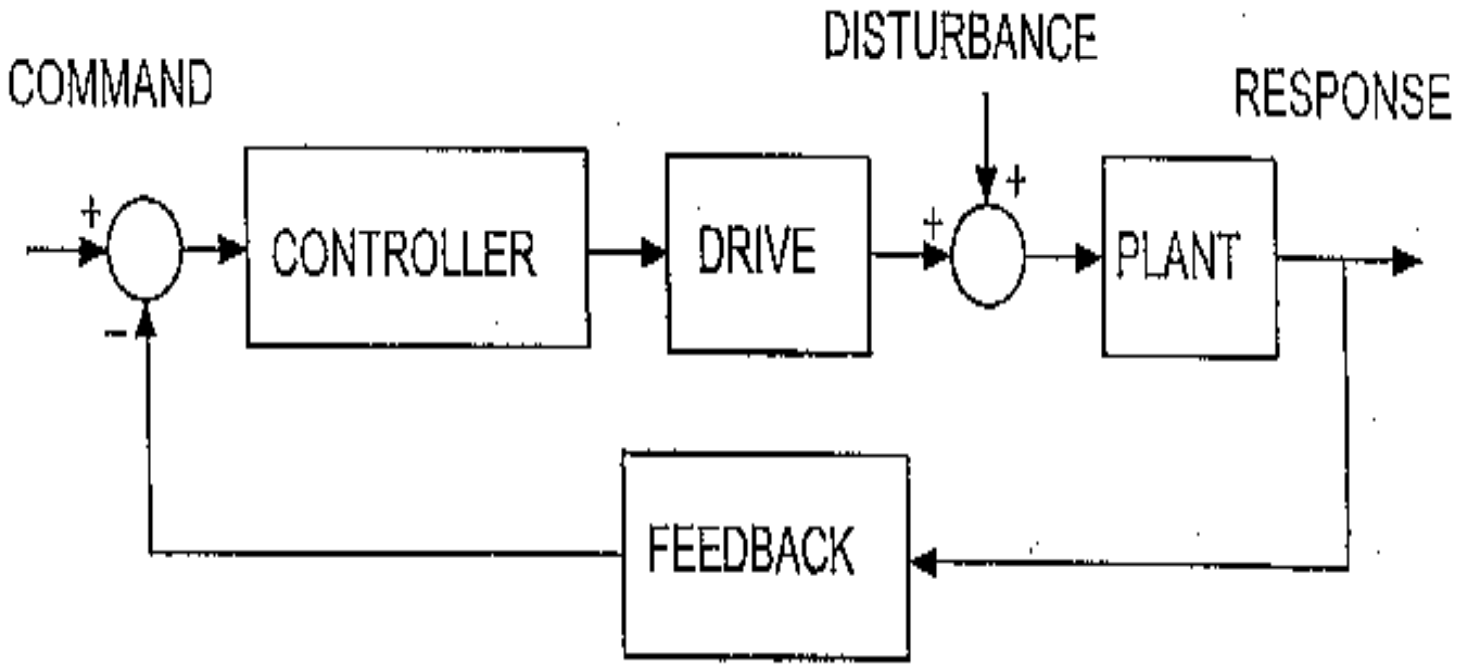
FEEDBACK CONTROL CONCEPT

The sketch on the next page shows a typical feedback or error driven control system. What has to be controlled is generally referred to as the plant. What the plant is doing is known as its response. What it should be doing is known as the command. The plant receives a control signal from a drive and a disturbance signal from the surroundings. The goal is to pick a controller that can make the response follow closely command signals but reject disturbances. The controller acts on an error signal: this is command minus some measure of the response. This is why it is usually called error driven control. Two types of error driven control are PID and Switching. PID stands for proportional integral derivative. Proportional generates a signal which is proportional to error. Integral generates a signal which is proportional to the integral of the error. Derivative generates a signal which is proportional to the rate of change of error. Switching generally gives out signals with constant levels.

AUTONOMOUS UNDERWATER VEHICLE DEPTH CONTROL

To illustrate some error driven control strategies we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. According to Newton's Second Law of Motion, the equation governing its up and down motion is:

$$M \, d^2R/dt^2 = B + D - W$$


COMMAND
DISTURBANCE
RESPONSE
+
+
+
−
CONTROLLER
DRIVE
PLANT
FEEDBACK

where R is the depth of the auv, M is its overall mass, B is the control force from the propulsion system, D is a disturbance load caused for example by sudden weight changes and W is a drag load. Drag load has two components: wake drag and wall drag:

$$W = X\, dR/dt\, |dR/dt| + Y\, dR/dt$$

where X and Y account mainly for the size and shape of the auv.

A simple model of the propulsion system is:

$$J\, dB/dt + I\, B = Q$$

where Q is the control signal. There are two basic types of propulsion systems that could be used to move the auv up and down. One is an air/water ballast tank. In this case, the control signal Q would produce a change in buoyancy and J would account for the fact that this is caused by a flow: I would be zero. If J was very large, the control force B would build up very slowly. The other type of propulsion system uses motor driven propellors to generate B. Usually, for protection, these would be located inside a duct. In this case, I would account for the size and shape of the blades and duct, while J would account for things like rotor inertia. Again, if J was very large, the control force B would build up very slowly. One could determine J and I experimentally.

The PID error driven strategy lets the control signal Q be:

$$Q = K_P E + K_I \int E d\tau + K_D \, dE/dt$$

where $E = C - R$ is the depth error and K_P K_I K_D are gains: C is the command depth. Usually, gains are constants. However, they can be made a function of the state of the system or its surroundings. In this case, control is said to be adaptive.

Imagine the auv is at the water surface and it suddenly commanded to go to some constant command depth C. Assume that there is a disturbance with a constant level D acting downward. Also assume the auv is using motor driven propellors for propulsion.

Proportional by itself would cause the propellors to spin in such a way that the auv would move towards the command depth. The amount of spin would be proportional to depth error. When the auv reaches the command depth, the proportional control signal would be zero. If the auv was held at the command depth, its propellors would stop spinning. The disturbance would cause the auv to stop below the command depth. This offset would be such that the propellors generate just enough upward force to balance the downward disturbance. The offset would be DI/K_P. When D is known, something called feedforward compensation can be used to get rid of the offset. Basically, we measure D and subtract ID from Q in the drive equation. When motions settle down, the drive gives out an extra signal minus D which cancels D. But we must know D. Another way to get rid of the offset is to give the auv a false

command C^*. If the false command C^* was set at $[C-DI/K_P]$, the auv would end up at C. It would hang below C^* by DI/K_P and thus end up at C. If the gain K_P was very large, offsets such as DI/K_P would probably be tolerable. However, large gain would generate very large Q when the depth is well away from the command depth. Very large Q could burn out drives. To avoid this, a limit is usually put on the magnitude of Q. In this case, the control is referred to as proportional with saturation. If the disturbance was greater than the saturation limits, then control would be impossible.

Integral by itself would cause the propellors to spin in such a way that the auv would move towards the command depth. The amount of spin would be proportional to the integral of depth error. As the auv moves towards the command depth, the propellors would spin faster and faster. Obviously, this would cause the auv to overshoot the command depth. Because of these overshoots, integral cannot be used alone. The good thing about integral is, if the system is stable, it gives zero offsets. If the auv was held with positive depth error, the integral control signal would get bigger and bigger. This is known as integral windup. If it was released after a long time, it would take a very large integrated negative error to cancel out the windup due to integrated positive error. A simple way to avoid integral windup is to activate integral only within a band surrounding the command depth. All we need is for the band to be wide enough for proportional to get the auv within the band so that integral can then home it into the command depth.

Derivative like integral cannot be used alone. Assume that the command C is a constant, and let the auv be stopped far away from the command depth. In this case, dE/dt would be zero. So, the controller would not generate a force to move the auv to the command depth. Derivative mimics drag load and helps motions settle down. It generates a control signal which opposes motion. Something called rate feedback could also be used to help make motions settle down. The controller would act on depth error E minus a constant times the depth rate dR/dt. Substitution into the governing equations shows that rate feedback mimics drag. Note that derivative could be used to make the auv move at a constant speed: dC/dt is made a constant. Drag and the dR/dt part of dE/dt would tend to limit speed.

With all three components of PID acting together, as soon as the auv passes through the command depth, proportional would tend to counteract integral. Also, proportional would get the auv closer to the command depth faster, so it would limit integral windup. Derivative would help counteract overshoots. The auv would home in quickly on the command depth with minimal overshoots. So, we get the good characteristics of all three controllers.

There are many types of switching control. They often have trouble with overshoots. Basic relay switching is the simplest. It would try to make the propellors rotate at a constant speed: the direction of rotation would depend on the sign of depth error. Relay with deadband would allow the auv to drift once it gets

inside a band surrounding the command depth. The propulsion device would be shut down and drag load would cause the auv to slow down. Relay with hysteresis would reverse the direction of control before the auv gets to the command depth. In this case, the propulsion device would act as a brake. A bias signal could be added to counteract disturbances.

Propulsion system dynamics would cause the control force to lag the control signal. The amount of lag depends on how large J is relative to I. Consider the case where proportional control is acting alone and the error is initially positive. For a slowly reacting propulsion system, positive error would cause a positive control force to gradually build up. As it builds up, this force would move the auv towards the command depth. However, when the auv gets to the command depth, because of lag, the control force would still be positive, and this would cause overshoot. In some cases, these overshoots would settle down. In other cases, they would not settle down but would limit because of wake drag.

Control signals for an auv would be generated within a computer control loop. The loop period must be much smaller than the basic period of auv motion: otherwise severe overshoots could develop. If the auv was controlled remotely by a computer onboard a ship, the time taken for the depth signal to travel from the auv to the ship and the time taken for the drive signal to travel back from the ship to the auv could cause overshoots, because the auv would be responding to past error not present error.

SUBSEA ROBOT SPRING/DASHPOT PID ANALOGY

Equations governing subsea robot depth motion are:

$$M\ d^2R/dt^2 + X\ dR/dt|dR/dt| + Y\ dR/dt = B + D$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K_P(C-R) + K_I\int(C-R)d\tau + K_D(dC/dt-dR/dt)$$

Let the drive be a propellor in a duct driven by a DC motor. For most of what follows, we will assume that the drive is fast acting, so that J is approximately zero. In this case,

$$B = K_P/I\ (C-R) + K_I/I\ \int(C-R)d\tau + K_D/I\ (dC/dt-dR/dt)$$

$$B = \mathbf{K}_P\ (C-R) + \mathbf{K}_I\ \int(C-R)d\tau + \mathbf{K}_D\ (dC/dt-dR/dt)$$

We will also assume that the robot is initially at one depth and it is suddenly commanded to go to another depth. When proportional control is acting alone, the control force B is a linear function of depth error. This pulls the robot towards the command depth. As the robot approaches the command depth, the propellor slows down. Note that a spring with its ends attached to the robot and the command depth would move the robot the same way. Because the drive is spring like, disturbances D cause the robot to settle down away from the command depth. When integral control is acting alone, the control force B gradually builds up and pulls the robot towards the command depth. As the robot approaches the command depth, the propellor goes faster and faster. This causes the robot to

overshoot the command depth. As soon as it overshoots, the control force starts to decrease: meaning the propellor starts to slow down. It takes time for the control force to go to zero. Beyond this point, the control force changes sign and acts initially like a brake and causes the robot to stop and then start back towards the command depth. Again, when it reaches the command depth, it overshoots it. These overshoots do not settle down. If they did, B would equal minus D and R would equal C. One could replace the integral drive with a spring with one end attached to the robot and the other end free to move. Initially the free end moves towards the command depth. This causes the spring to stretch and pull the robot towards the command depth. The spring stretching mimics the integration of error. The spring keeps stretching until the robot overshoots the command depth. Then, it gradually slackens. It takes time for the spring to totally slacken so it pulls the robot beyond the command depth. When the spring is totally slack, the free end starts back towards the command depth. In this case, the spring acts initially like a brake and causes the robot to stop and then start back towards the command depth. With proportional and integral acting together it is possible for the robot to settle at the command depth. Proportional suppresses the overshoots caused by integral and integral gets rid of offsets. Derivative control is not spring like. The equation for B shows that it instead mimics a dashpot. When the drive is slow acting, control actions are not instantaneous. This can cause severe overshoots.

CAR/DRIVER PID ANALOGY

Imagine a car at position A on a straight road that is suddenly commanded to go to position B on the same road. A proportional driver would suddenly depress the gas peddle down to some level. This would cause the car to gradually pick up speed. As the car moves towards B, the driver would depress the gas peddle less and less. The amount of depression would be a linear function of position error or distance between B and the car position. When the car reaches B, peddle depression would be zero. Because of its momentum, the car would overshoot B. As soon as it does so, the driver would suddenly put the car into reverse and depress the gas peddle an amount again dependent on position error. This would cause the car to gradually come to a stop and reverse direction back towards B. If there was no wind and the road was horizontal, wake drag and drive friction would gradually make the car come to rest at B. Otherwise, it would come to rest away from B. An integral driver starting at A would gradually depress the gas peddle based on the integral of position error. This would move the car towards B but at a faster and faster speed. When the car reaches B, peddle depression would be maximum. Obviously, the car would overshoot B. As soon as it does so, the driver would gradually depress the gas peddle less and less. Basically, the position error would now be negative, and

integrated error would gradually decrease. When it reaches zero, the peddle depression would also be zero, and the driver would suddenly put the car into reverse and gradually depress the gas peddle again based on the integral of position error. This would cause the car to gradually come to a stop and reverse direction back towards B. When the car reaches B, it would again overshoot. The car would never settle at B but would oscillate back and forth at an amplitude dependent on wake drag and drive friction. The mean position error would be zero, even when there was wind or the road was not horizontal. A proportional plus integral driver could make the car settle at B, even when there was wind or the road was not horizontal. The proportional part would bring the car close to B before the integral part could build up too much signal. The integral part would then home the car into B. Whereas the proportional plus integral driver would work only the gas peddle, a proportional plus integral plus derivative driver would also use the brake. The derivative part would apply the brake an amount based on speed. This would help control overshoots if they are a problem. Driver reaction time could cause severe overshoots. Its control is based on past error not present error.

ZIEGLER NICHOLS GAINS

Ziegler and Nichols, through a series of experiments on simple systems, developed criteria for picking gains in a controller that would give good tracking performance. For a system that can be made unstable with proportional acting alone, the procedure they recommend is as follows. With proportional acting alone, increase its gain until the system becomes borderline stable. Let the borderline gain be $\mathbf{K_P}$: let its period be $\mathbf{T_P}$. According to Ziegler and Nichols, reasonable PID gains are:

$$K_P = 0.6*\mathbf{K_P} \qquad K_I = K_P/T_I \qquad K_D = K_P*T_D$$

$$T_I = 0.5*\mathbf{T_P} \qquad T_D = 0.125*\mathbf{T_P}$$

When only proportional and integral are acting, they recommend the following PI gains:

$$K_P = 0.45*\mathbf{K_P} \qquad K_I = K_P/T_I$$

$$T_I = 0.83*\mathbf{T_P}$$

When only proportional is acting, they recommend:

$$K_P = 0.5*\mathbf{K_P}$$

AUTONOMOUS UNDERWATER VEHICLE

ZIEGLER NICHOLS GAINS

To illustrate a procedure for getting Ziegler Nichols gains, we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. According to Newton's Second Law of Motion, the equation governing the up and down motion of the auv is:

$$M\ d^2R/dt^2 = B + D - W$$

where R is the depth of the auv, M is its overall mass, B is the control force from the propulsion system, D is a disturbance load caused for example by sudden weight changes and W is a drag load consisting of wake drag and wall drag:

$$W = X\ dR/dt\ |dR/dt| + Y\ dR/dt$$

where X and Y account for the size and shape of the auv. Here we linearize the drag to get:

$$W = N\ dR/dt$$

A simple model of the propulsion system is:

$$J\ dB/dt + I\ B = Q$$

where Q is the control signal: J and I are drive constants.

The PID error driven strategy lets the control signal Q be:

$$Q = K_P E + K_I \int E d\tau + K_D \, dE/dt$$

where $E = C - R$ is the depth error and K_P K_I K_D are the controller gains: C is the command depth.

To get Ziegler Nichols gains, we start by assuming only proportional is active. Manipulation of the governing equations gives:

$$J \, [\, M \, d^3R/dt^3 + N \, d^2R/dt^2 - dD/dt]$$

$$+ \; I \, [\, M \, d^2R/dt^2 + N \, dR/dt - D \,] = K_P \, C - K_P \, R$$

We then assume that C and D are both constants and that the auv is undergoing a limit cycle oscillation for which

$$R = R_o + \Delta R \, \mathrm{Sin} \, [\omega t]$$

Substitution into the modified drive equation gives

$$- J \; M \; \omega^3 \, \Delta R \, \mathrm{Cos}[\omega t] - J \; N \; \omega^2 \, \Delta R \, \mathrm{Sin}[\omega t]$$

$$- I \; M \; \omega^2 \, \Delta R \, \mathrm{Sin}[\omega t] + I \; N \; \omega \; \Delta R \, \mathrm{Cos}[\omega t]$$

$$- I \; D_o = \mathbf{K_P} \; C_o - \mathbf{K_P} \; R_o - \mathbf{K_P} \; \Delta R \, \mathrm{Sin}[\omega t]$$

This equation is of the form:

$$i \, \mathrm{Sin}[\omega t] + j \, \mathrm{Cos}[\omega t] + k = 0$$

Mathematics requires that i=0 j=0 k=0:

$$- J N \omega^2 - I M \omega^2 + \mathbf{K_P} = 0$$

$$- J M \omega^3 + I N \omega = 0$$

$$+ I D_o + \mathbf{K_P} C_o - \mathbf{K_P} R_o = 0$$

Manipulation of these equations gives

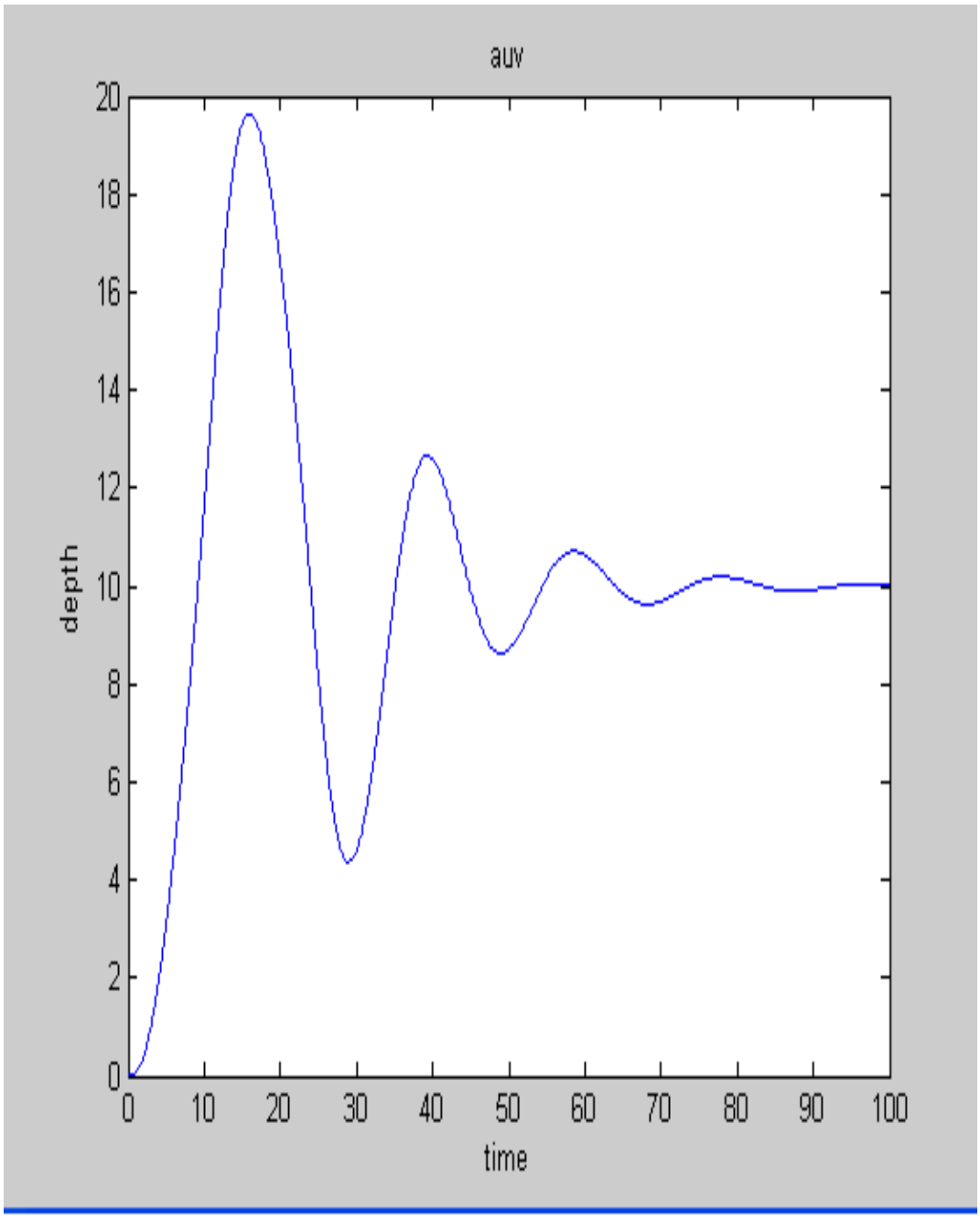
$$R_o = C_o + I D_o / \mathbf{K_P}$$

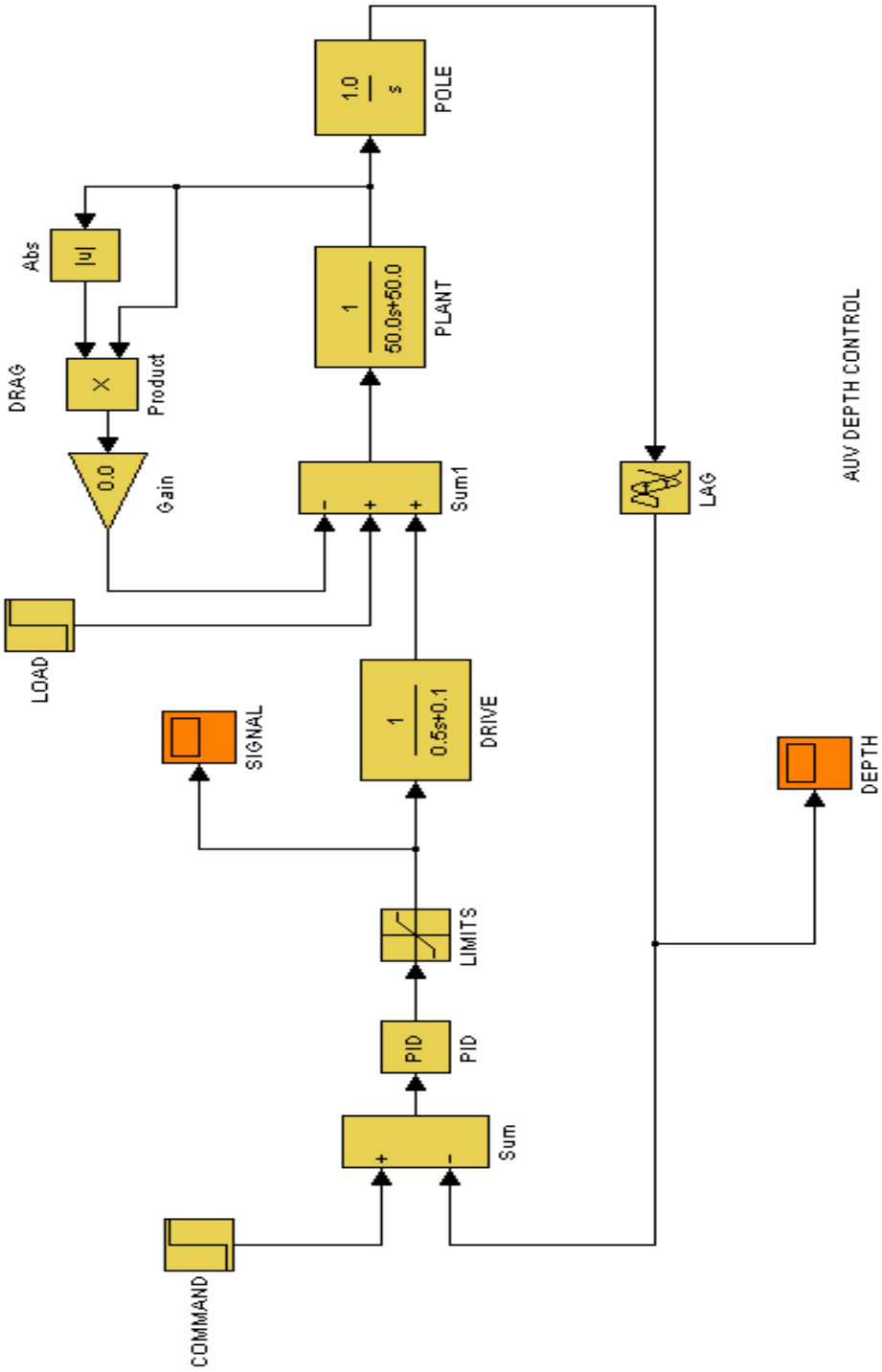
$$\omega^2 = [I N] / [J M]$$

$$\mathbf{K_P} = [J N + I M] \omega^2$$
$$= [J N + I M] [I N] / [J M]$$

For the illustration we let : M=50 N=50 J=0.5 I=0.1. The above equations give ω=0.447, $\mathbf{K_P}$=6 and $\mathbf{T_P}$=14. Substitution into the Ziegler Nichols gains equations gives: K_P = 3.6; K_I = 0.54; K_D = 6.3. An m code for the auv is given below. This is followed by a Ziegler Nichols response generated by the code. A SIMULINK Block diagram follows the m code response. It gives basically the same response as the code.

```
% AUV DEPTH CONTROL
  rold=0.0;uold=0.0;bold=0.0;
  told=0.0;m=50.0;load=0.0;
  wake=0.0;wall=50.0;sum=0.0;
  j=0.5;i=0.1;wrong=0.0;
  gp=6.0;gi=0.0;gd=0.0;
  gp=3.6;gi=0.54;gd=6.3;
  delt=0.01;target=10.0;
  for k=1:10000
  error=target-rold;
  rate=(error-wrong)/delt;
  control=gp*error;
  control=control+gi*sum;
  control=control+gd*rate;
  if(control>+12.0) ...
   control=+12.0;end;
  if(control<-12.0) ...
   control=-12.0;end;
  sum=sum+delt*error;
  drag=wake*uold*abs(uold);
  drag=drag+wall*uold;
  abe=bold+load-drag;
  xyz=control-bold*i;
  rnew=rold+delt*uold;
  unew=uold+delt*abe/m;
  bnew=bold+delt*xyz/j;
  tnew=k*delt;wrong=error;
  rold=rnew;uold=unew;
  bold=bnew;told=tnew;
  r(k)=rnew;t(k)=tnew;
  end;     plot(t,r)
  xlabel('time')
  ylabel('depth')
  title('auv')
'
```


auv
depth
time
0
2
4
6
8
10
12
14
16
18
20
0
10
20
30
40
50
60
70
80
90
100


AUV DEPTH CONTROL
1.0
s
POLE
1
50.0s+50.0
PLANT
Abs
|u|
DRAG
Product
0.0
Gain
Sum1
LAG
LOAD
SIGNAL
1
0.5s+0.1
DRIVE
DEPTH
LIMITS
PID
PID
Sum
COMMAND

PIPE FLOW SETUP

ZIEGLER NICHOLS GAINS

To illustrate a procedure for getting Ziegler Nichols gains, we will consider the task of controlling the temperature of the air flowing down the pipe in the lab pipe flow setup. Basically the setup consists of a fan which draws air from atmosphere and sends it down a pipe. A heater just downstream of the fan is used to heat the air. It receives a signal from a controller. The temperature of the air at the pipe exit is measured by a thermistor. The governing equations are:

$$X\ dR/dt + Y\ R = H + D$$

$$A\ dH/dt + B\ H = Z\ Q$$

$$Q = K_P\ E + K_I \int E d\tau + K_D\ dE/dt$$

$$E = C - \mathbf{R}$$

where R is the temperature of the air at the heater, **R** is the temperature of the air at the sensor, C is the command temperature, E is the temperature error, Q is the control signal, H is the heat generated by the heater, D is a disturbance heat and K_P K_I K_D are the controller gains. Note that **R** is what R was T seconds back in time: T is the time it takes for the air to travel down the pipe.

To get Ziegler Nichols gains, we start by assuming only proportional is active. Manipulation of the governing equations gives:

$$A\left[X \frac{d^2R}{dt^2} + Y \frac{dR}{dt} - \frac{dD}{dt}\right]$$

$$+ B\left(X \frac{dR}{dt} + Y R - D\right) = Z K_P C - Z K_P \mathbf{R}$$

We then assume that C and D are both constants and that the setup is undergoing a limit cycle oscillation for which

$$R = R_o + \Delta R \sin[\omega t] \qquad \mathbf{R} = R_o + \Delta R \sin[\omega(t-T)]$$

Substitution into the modified drive equation gives

$$- A X \omega^2 \Delta R \sin[\omega t] + A Y \omega \Delta R \cos[\omega t]$$

$$+ B X \omega \Delta R \cos[\omega t] + B Y R_o + B Y \Delta R \sin[\omega t]$$

$$- B D_o = Z \mathbf{K_P} C_o - Z \mathbf{K_P} R_o - Z \mathbf{K_P} \Delta R \sin[\omega(t-T)]$$

A trigonometric identity gives

$$\sin[\omega(t-T)] = \sin[\omega t] \cos[\omega T] - \cos[\omega t] \sin[\omega T]$$

Substitution into the modified drive equation gives an equation of the form

$$i \sin[\omega t] + j \cos[\omega t] + k = 0$$

Setting i=0 and j=0 and k=0 gives

$$- A\ X\ \omega^2 \ + \ B\ Y \ + \ Z\ \ \mathbf{K_P}\ \mathrm{Cos}[\omega T] = \ 0$$

$$A\ Y\ \omega \ + \ B\ X\ \omega \ - \ Z\ \mathbf{K_P}\ \mathrm{Sin}[\omega T] = \ 0$$

$$B\ Y\ R_o \ - \ B\ D_o \ - \ Z\ \mathbf{K_P}\ C_o \ + \ Z\ \mathbf{K_P}\ R_o \ = \ 0$$

Manipulation of the first two equations gives

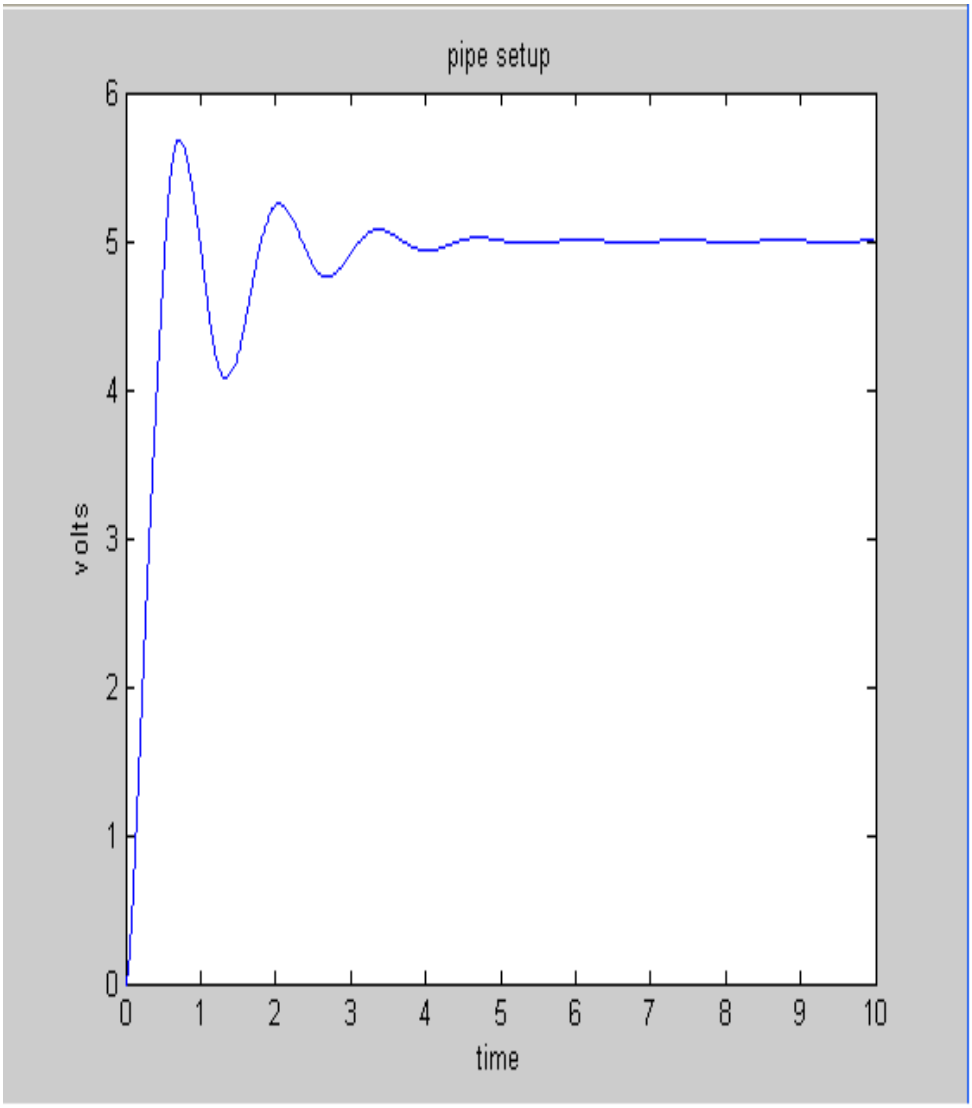
$$\mathbf{K_P} = [A\ X\ \omega^2 - B\ Y] \ / \ [Z\ \mathrm{Cos}[\omega T]]$$

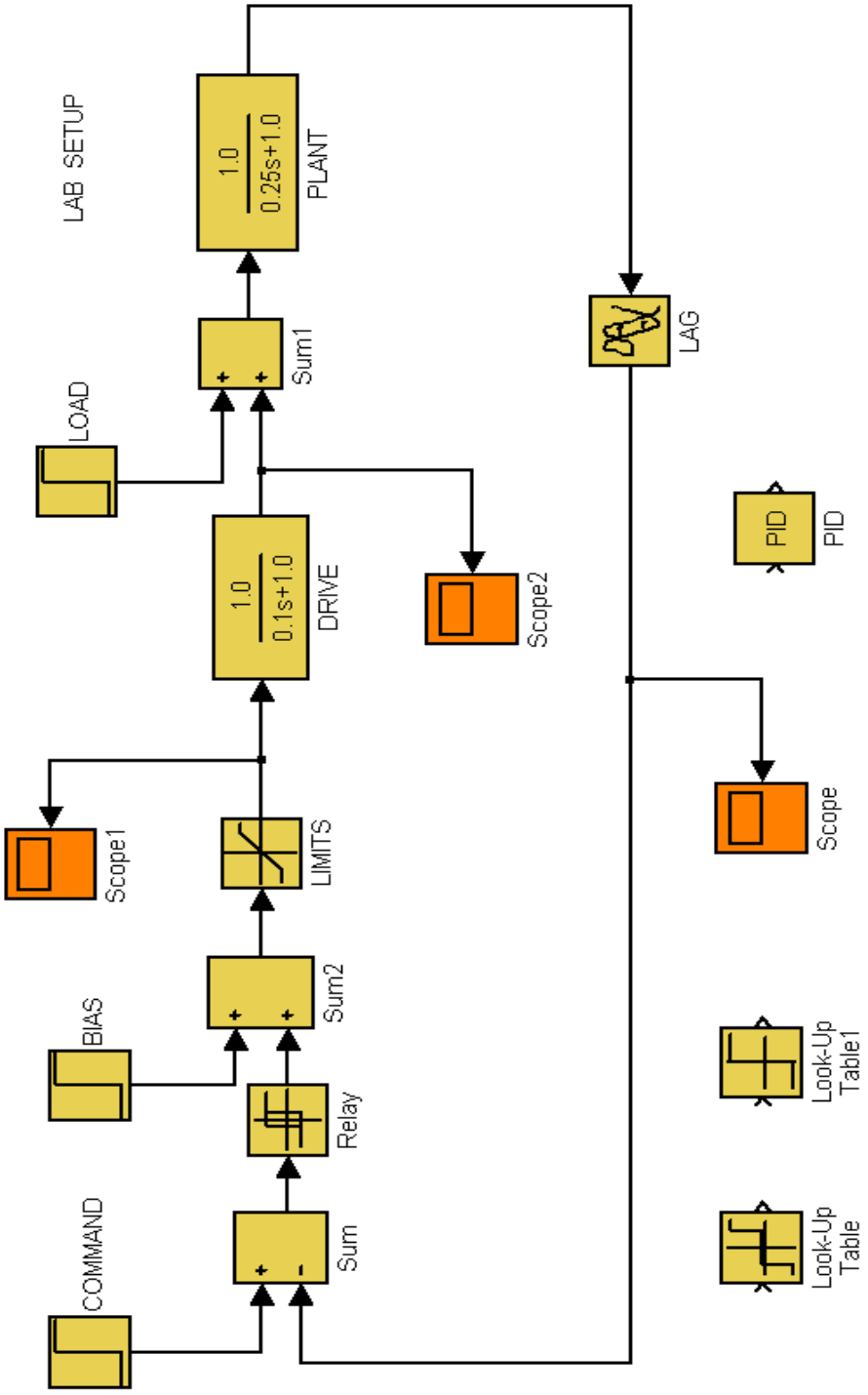
$$\mathbf{K_P} = [A\ Y\ \omega + B\ X\ \omega] \ / \ [Z\ \mathrm{Sin}[\omega T]]$$

$$\mathrm{Sin}[\omega T]/\mathrm{Cos}[\omega T] = \mathrm{Tan}[\omega T]$$
$$= [A\ Y\ \omega + B\ X\ \omega] \ / \ [A\ X\ \omega^2 - B\ Y]$$

The last equation gives ω. Once ω is known we can then solve for $\mathbf{K_P}$. For the illustration, we let: X=0.25 Y=1.0 A=0.1 B=1.0 Z=1.0 T=0.5. The above equations give ω=3.97, $\mathbf{K_P}$=1.5 and $\mathbf{T_P}$=1.58. Substitution into the Ziegler Nichols gains equations gives: K_P=0.9; K_I=1.2; K_D=0.17. An m code for the setup is given below. This is followed by a Ziegler Nichols response. A SIMULINK Block diagram follows the response. It gives basically the same response as the code.

```
% PIPE FLOW TEMPERATURE CONTROL
  ROLD=0.0;HOLD=0.0;SENSOR=ROLD;
  TARGET=5.0;LOAD=0.0;DUMP=10.0;
  X=0.25;Y=1.0;A=0.1;B=1.0;Z=1.0;
  WRONG=TARGET-SENSOR;SUM=0.0;
  NIT=10000;MIT=500;TIME=0.0;
  GP=1.5;GI=0.0;GD=0.0;
  GP=0.9;GI=1.2;GD=0.17;
  DELT=0.001;
  for IT=1:NIT
  TIME=TIME+DELT;
  if(IT>MIT) ...
   SENSOR=R(IT-MIT); end;
  ERROR=TARGET-SENSOR;
  RATE=(ERROR-WRONG)/DELT;
  CONTROL=GP*ERROR;
  CONTROL=CONTROL+GI*SUM;
  CONTROL=CONTROL+GD*RATE;
  SUM=SUM+DELT*ERROR;
  if(CONTROL>DUMP) ...
   CONTROL=DUMP; end;
  if(CONTROL<0.0) ...
   CONTROL=0.0; end;
  ABC=Z*CONTROL-B*HOLD;
  XYZ=HOLD+LOAD-Y*ROLD;
  HNEW=HOLD+DELT*ABC/A;
  RNEW=ROLD+DELT*XYZ/X;
  T(IT)=TIME;R(IT)=RNEW;
  ROLD=RNEW;HOLD=HNEW;
  WRONG=ERROR;
  end; plot(T,R)
  xlabel('time')
  ylabel('volts')
  title('pipe setup')
```


pipe setup
volts
time
0
1
2
3
4
5
6
7
8
9
10


LAB SETUP
1.0
0.25s+1.0
PLANT
LAG
Sum1
LOAD
1.0
0.1s+1.0
DRIVE
Scope2
PID
PID
Scope1
LIMITS
Scope
Sum2
BIAS
Look-Up
Table1
Relay
Look-Up
Table
COMMAND
Sum

COMPUTER SIMULATION OF CONTROL SYSTEMS

PREAMBLE

Simulation allows one to study the behavior of a system before it is actually constructed. This can serve as an aid to system design. Simulations are inexpensive and easy to put together. They can handle all sorts of phenomena. These include transport lag and computer loop rate phenomena. Simulations can also handle multiple strong nonlinearities. They are often used as a check on more conventional analysis. However, simulations are like experiments. For complex systems, it is hard to make sense of responses. Before digital computers were developed, systems were simulated using analog electronics. When digital computers became common place, simulations made use of time stepping procedures. Basically, these follow local slopes or rates step by step in time. Special software packages based on these procedures have been developed. Probably, the popular package is SIMULINK under MATLAB.

AUTONOMOUS UNDERWATER VEHICLE

TIME STEPPING SIMULATION

To illustrate time stepping we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. The governing equations are:

$$M\ d^2R/dt^2 = B + D - W$$

$$W = X\ dR/dt\ |dR/dt| + Y\ dR/dt$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K_P\ E + K_I \int E d\tau + K_D\ dE/dt$$

$$E = C - R$$

where R is the depth of the auv, M is its overall mass, B is the control force from the propulsion system, D is a disturbance load caused for example by sudden weight changes, W is a drag load consisting of wake drag and wall drag, E is the depth error, C is the command depth, M X Y J I are process constants and K_P K_I K_D are the controller gains.

Manipulation of the governing equations gives

$$dR/dt = U$$

$$dU/dt = (B + D - W) / M$$

$$W = X\ U\ |U| + Y\ U$$

$$dB/dt = (Q - I\ B) / J$$

$$Q = K_P\ E + K_I \int E d\tau + K_D\ dE/dt$$

$$E = C - R$$

Application of time stepping gives

$$R_{NEW} = R_{OLD} + \Delta t * U_{OLD}$$

$$U_{NEW} = U_{OLD} + \Delta t * (B_{OLD} + D_{OLD} - W_{OLD}) / M$$

$$W_{OLD} = X\ U_{OLD}\ |U_{OLD}| + Y\ U_{OLD}$$

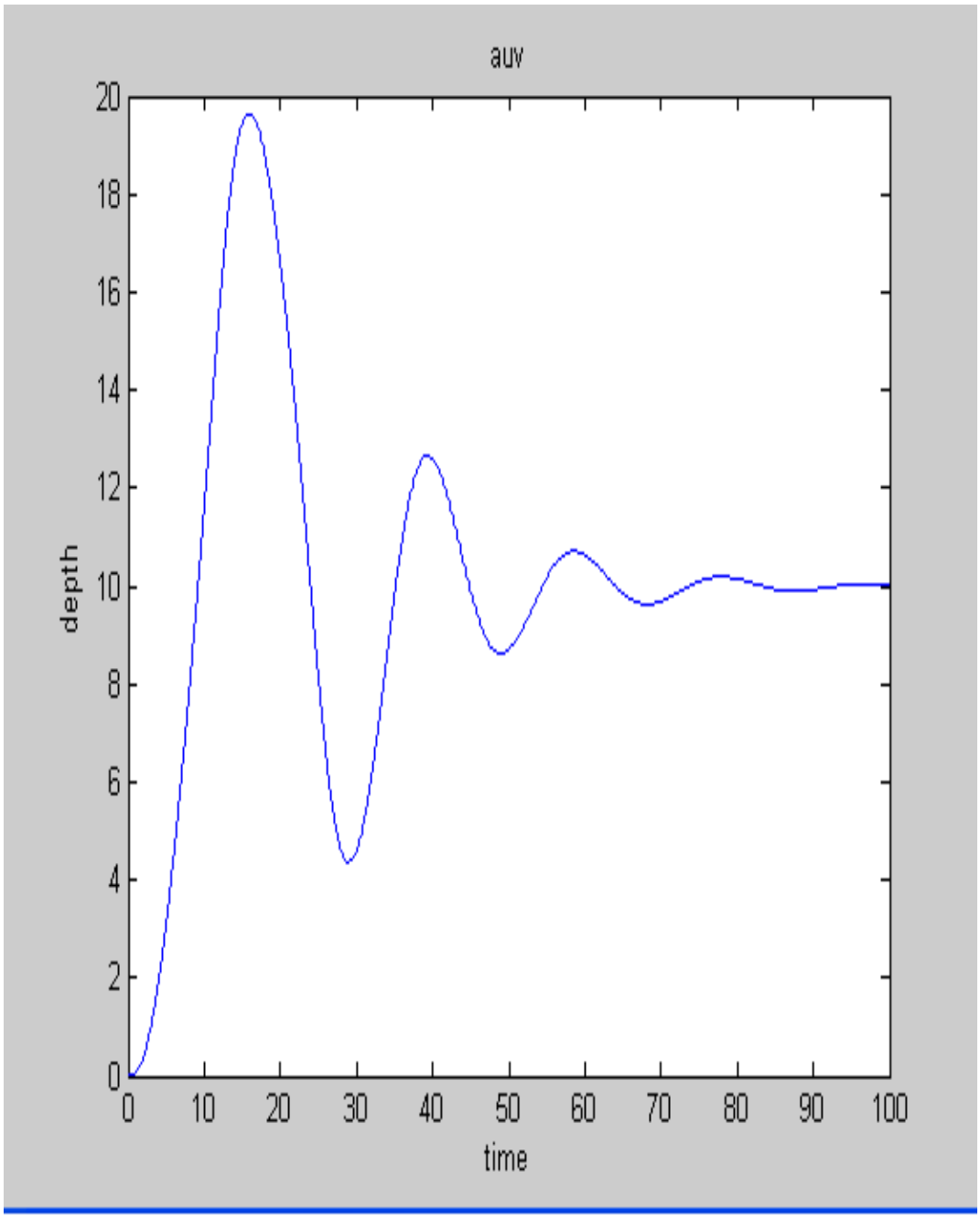
$$B_{NEW} = B_{OLD} + \Delta t * (Q_{OLD} - I\ B_{OLD}) / J$$

$$Q_{OLD} = K_P\ E_{OLD} + K_I\ \Sigma\ E_{OLD}\ \Delta t + K_D\ \Delta E_{OLD}/\Delta t$$

$$E_{OLD} = C_{OLD} - R_{OLD}$$

An m code for the auv is given below. This is followed by a Ziegler Nichols response generated by the code.

```
% AUV DEPTH CONTROL
  rold=0.0;uold=0.0;bold=0.0;
  told=0.0;m=50.0;load=0.0;
  wake=0.0;wall=50.0;sum=0.0;
  j=0.5;i=0.1;wrong=0.0;
  gp=6.0;gi=0.0;gd=0.0;
  gp=3.6;gi=0.54;gd=6.3;
  delt=0.01;target=10.0;
  for k=1:10000
  error=target-rold;
  rate=(error-wrong)/delt;
  control=gp*error;
  control=control+gi*sum;
  control=control+gd*rate;
  if(control>+12.0) ...
   control=+12.0;end;
  if(control<-12.0) ...
   control=-12.0;end;
  sum=sum+delt*error;
  drag=wake*uold*abs(uold);
  drag=drag+wall*uold;
  abe=bold+load-drag;
  xyz=control-bold*i;
  rnew=rold+delt*uold;
  unew=uold+delt*abe/m;
  bnew=bold+delt*xyz/j;
  tnew=k*delt;wrong=error;
  rold=rnew;uold=unew;
  bold=bnew;told=tnew;
  r(k)=rnew;t(k)=tnew;
  end;     plot(t,r)
  xlabel('time')
  ylabel('depth')
  title('auv')
'
```


auv
depth
time
0
2
4
6
8
10
12
14
16
18
20
0
10
20
30
40
50
60
70
80
90
100

PIPE FLOW SETUP

TIME STEPPING SIMULATION

To illustrate time stepping we will consider the task of controlling the temperature of air flowing down a pipe. The setup is shown on the next page. The governing equations are:

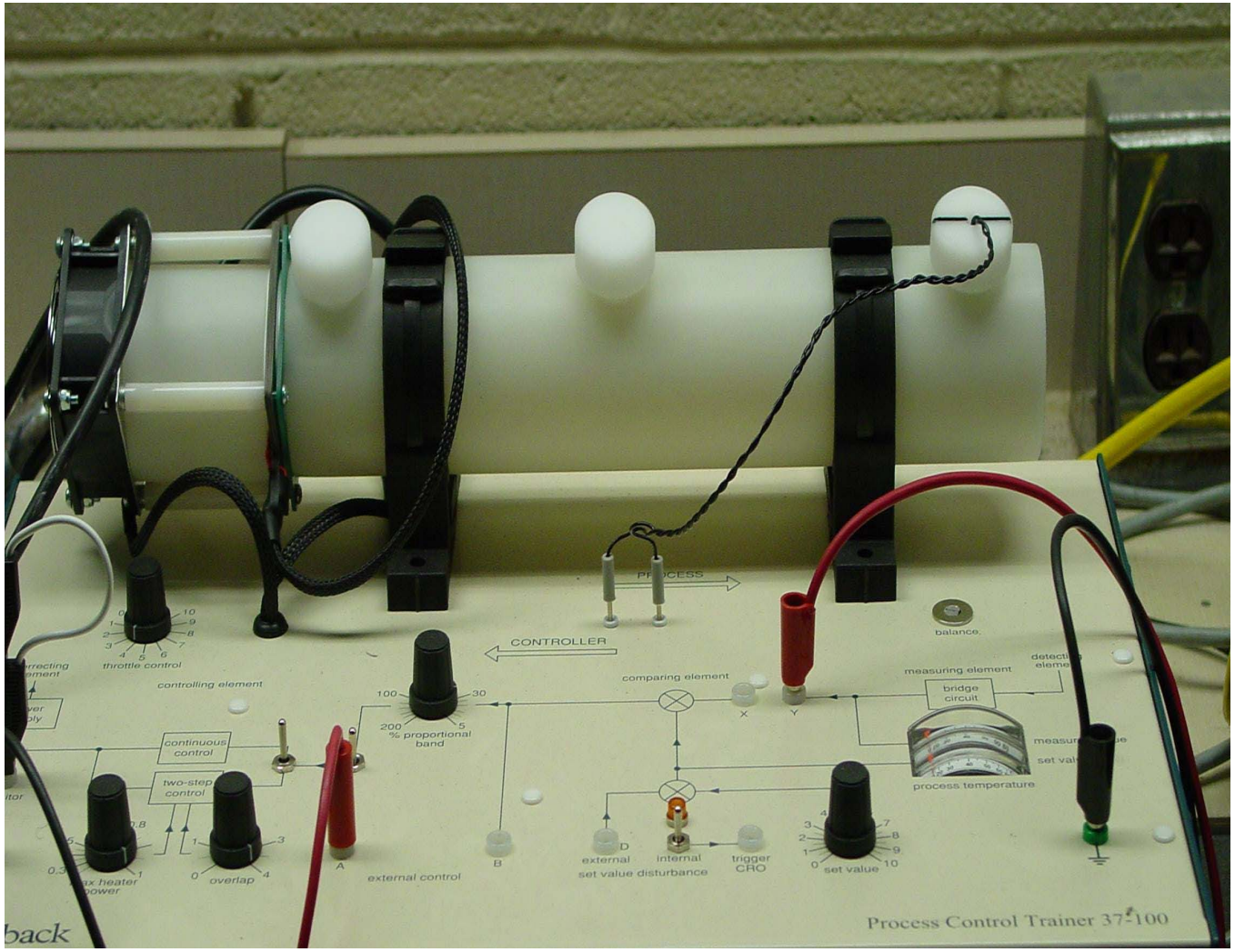
$$X\ dR/dt + Y\ R = H + D$$

$$A\ dH/dt + B\ H = Z\ Q$$

$$Q = K_P\ E + K_I \int E d\tau + K_D\ dE/dt$$

$$E = C - \mathbf{R}$$

where R is the temperature of the air at the heater, **R** is the temperature of the air at the sensor, C is the command temperature, E is the temperature error, Q is the control signal, H is the heat generated by the heater, D is a disturbance heat (plus or minus), X Y A B Z are process constants and K_P K_I K_D are the controller gains. Note that **R** is what R was T seconds back in time: T is the time it takes for the air to travel down the pipe.


CONTROLLER
balance
throttle control
controlling element
comparing element
measuring element
bridge circuit
continuous control
two-step control
% proportional band
process temperature
external set value
internal disturbance
trigger CRO
set value
overlap
external control
Process Control Trainer 37-100

Manipulation of the governing equations gives

$$dR/dt = (H + D - Y\ R) / X$$

$$dH/dt = (Z\ Q - B\ H) / A$$

$$Q = K_P\ E + K_I \int E d\tau + K_D\ dE/dt$$

$$E = C - \mathbf{R}$$

Application of time stepping gives

$$R_{NEW} = R_{OLD} + \Delta t * (H_{OLD} + D_{OLD} - Y\ R_{OLD}) / X$$

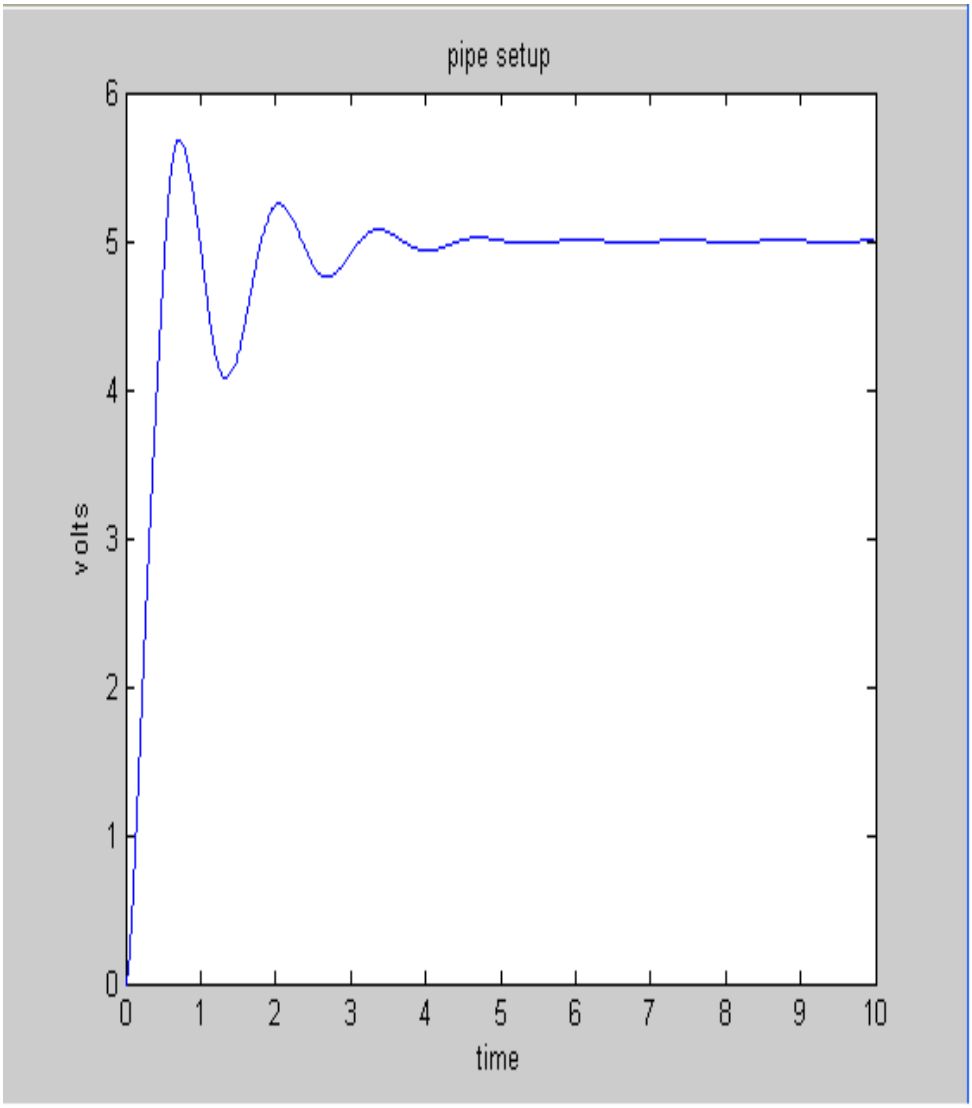
$$H_{NEW} = H_{OLD} + \Delta t * (Z\ Q_{OLD} - B\ H_{OLD}) / A$$

$$Q_{OLD} = K_P\ E_{OLD} + K_I\ \Sigma\ E_{OLD}\ \Delta t + K_D\ \Delta E_{OLD}/\Delta t$$

$$E_{OLD} = C_{OLD} - \mathbf{R}_{OLD}$$

An m code for the setup is given below. This is followed by a Ziegler Nichols response generated by the code.

```
% PIPE FLOW TEMPERATURE CONTROL
  ROLD=0.0;HOLD=0.0;SENSOR=ROLD;
  TARGET=5.0;LOAD=0.0;DUMP=10.0;
  X=0.25;Y=1.0;A=0.1;B=1.0;Z=1.0;
  WRONG=TARGET-SENSOR;SUM=0.0;
  NIT=10000;MIT=500;TIME=0.0;
  GP=1.5;GI=0.0;GD=0.0;
  GP=0.9;GI=1.2;GD=0.17;
  DELT=0.001;
  for IT=1:NIT
  TIME=TIME+DELT;
  if(IT>MIT) ...
   SENSOR=R(IT-MIT); end;
  ERROR=TARGET-SENSOR;
  RATE=(ERROR-WRONG)/DELT;
  CONTROL=GP*ERROR;
  CONTROL=CONTROL+GI*SUM;
  CONTROL=CONTROL+GD*RATE;
  SUM=SUM+DELT*ERROR;
  if(CONTROL>DUMP) ...
   CONTROL=DUMP; end;
  if(CONTROL<0.0) ...
   CONTROL=0.0; end;
  ABC=Z*CONTROL-B*HOLD;
  XYZ=HOLD+LOAD-Y*ROLD;
  HNEW=HOLD+DELT*ABC/A;
  RNEW=ROLD+DELT*XYZ/X;
  T(IT)=TIME;R(IT)=RNEW;
  ROLD=RNEW;HOLD=HNEW;
  WRONG=ERROR;
  end; plot(T,R)
  xlabel('time')
  ylabel('volts')
  title('pipe setup')
```


pipe setup
volts
time
0
1
2
3
4
5
6
7
8
9
10

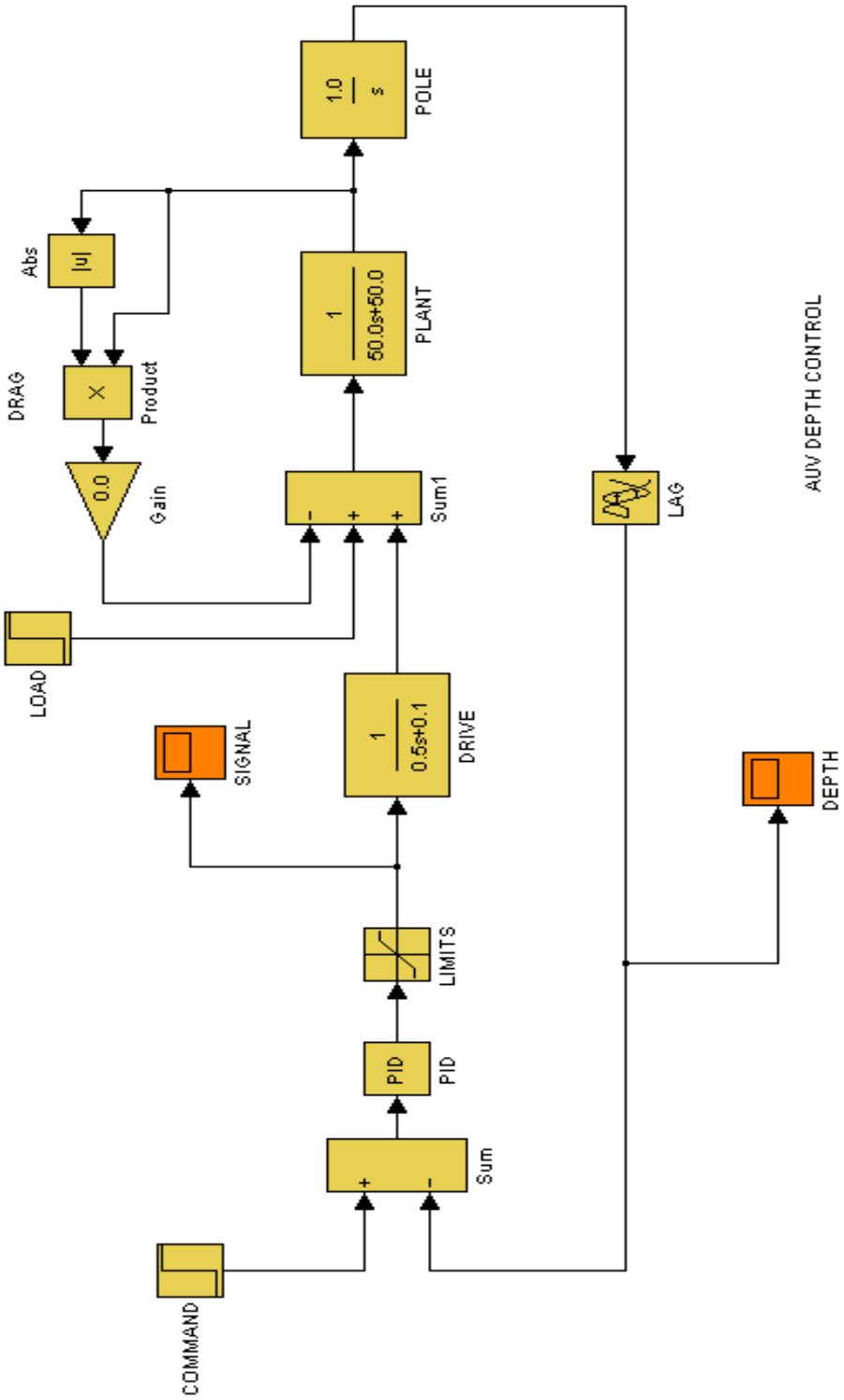
SIMULINK CONTROL SYSTEM SIMULATION

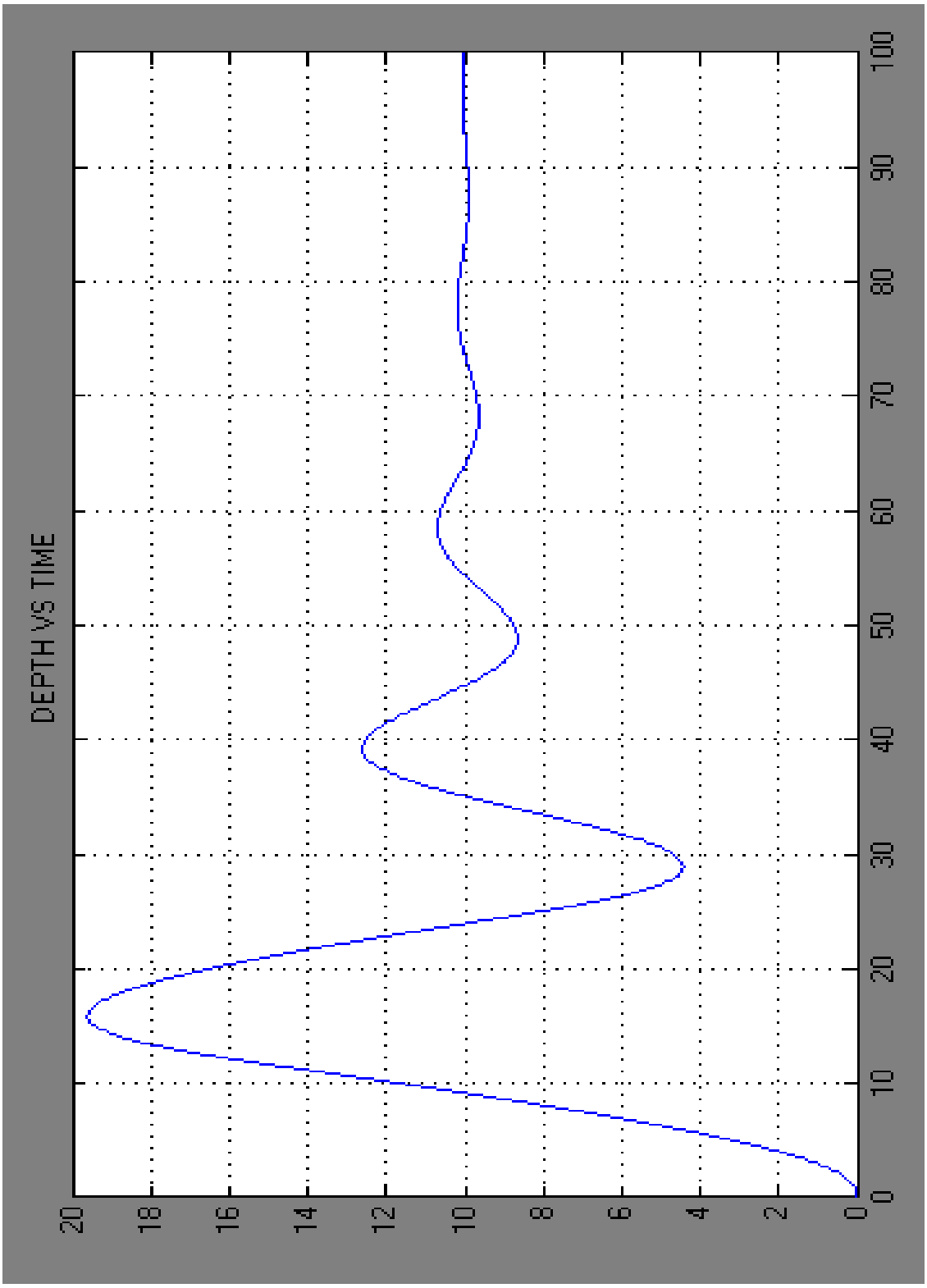
SIMULINK makes use of a block diagram representation of the system. One activates SIMULINK by typing SIMULINK and pressing enter in the main MATLAB window. Blocks are formed by picking blocks from groups of blocks in the main SIMULINK window. The group labeled SOURCES contains blocks that could be used for commands and disturbances. The group labeled SINKS contains blocks that could be used for display of responses. The group labeled CONTINUOUS contains many common transfer functions and state space blocks. The group labeled DISCRETE contains blocks that could be used to mimic loop rate phenomena. The group labeled MATH contains blocks for things like summation junctions and gains. The group labeled NONLINEAR contains various types of nonlinearities and switching controllers. Many of the switching controllers can be formed using LOOK UP TABLE under the group of blocks labeled FUNCTIONS & TABLES. The PID controller can be found under ADDITIONAL LINEAR under SIMULINK EXTRAS under BLOCK SETS & TOOL BOXES.

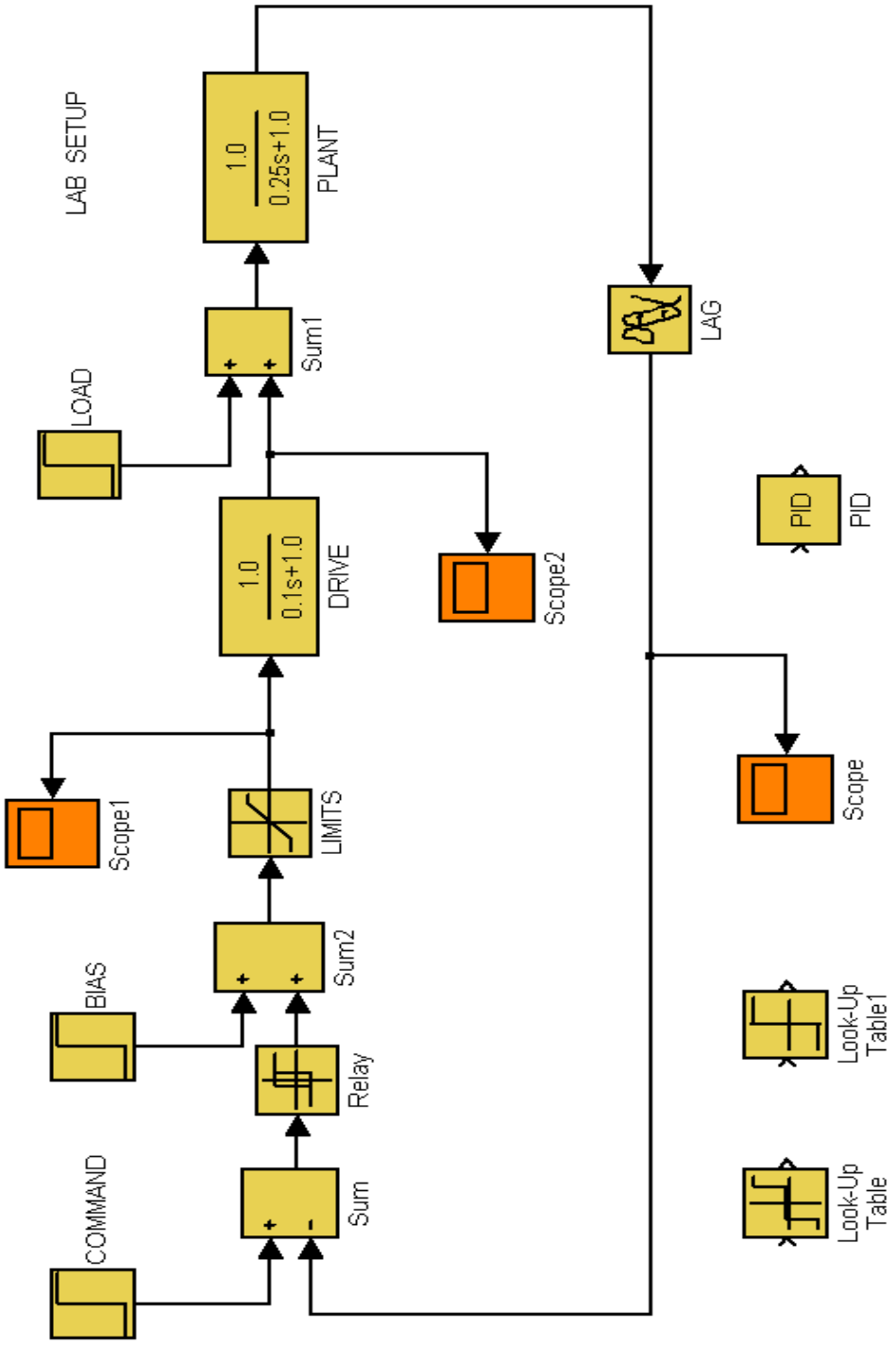
Block diagram construction makes extensive use of the click and drag functions of the left and right buttons of the mouse. To illustrate the construction, imagine you have an empty MINE window open on the screen. From the SIMULINK window, double left click on the SOURCES icon. Then, from its window, left click on the STEP block and drag it to the MINE window. All other blocks can be moved this way. You can also use COPY and PASTE. To move blocks around in the MINE window, just left click and drag them. You can also use CUT and PASTE. To join blocks with lines, you again use left click and drag. To create break lines, you use right click on the break point and drag. To change parameters, double left click on the block to activate a block menu.

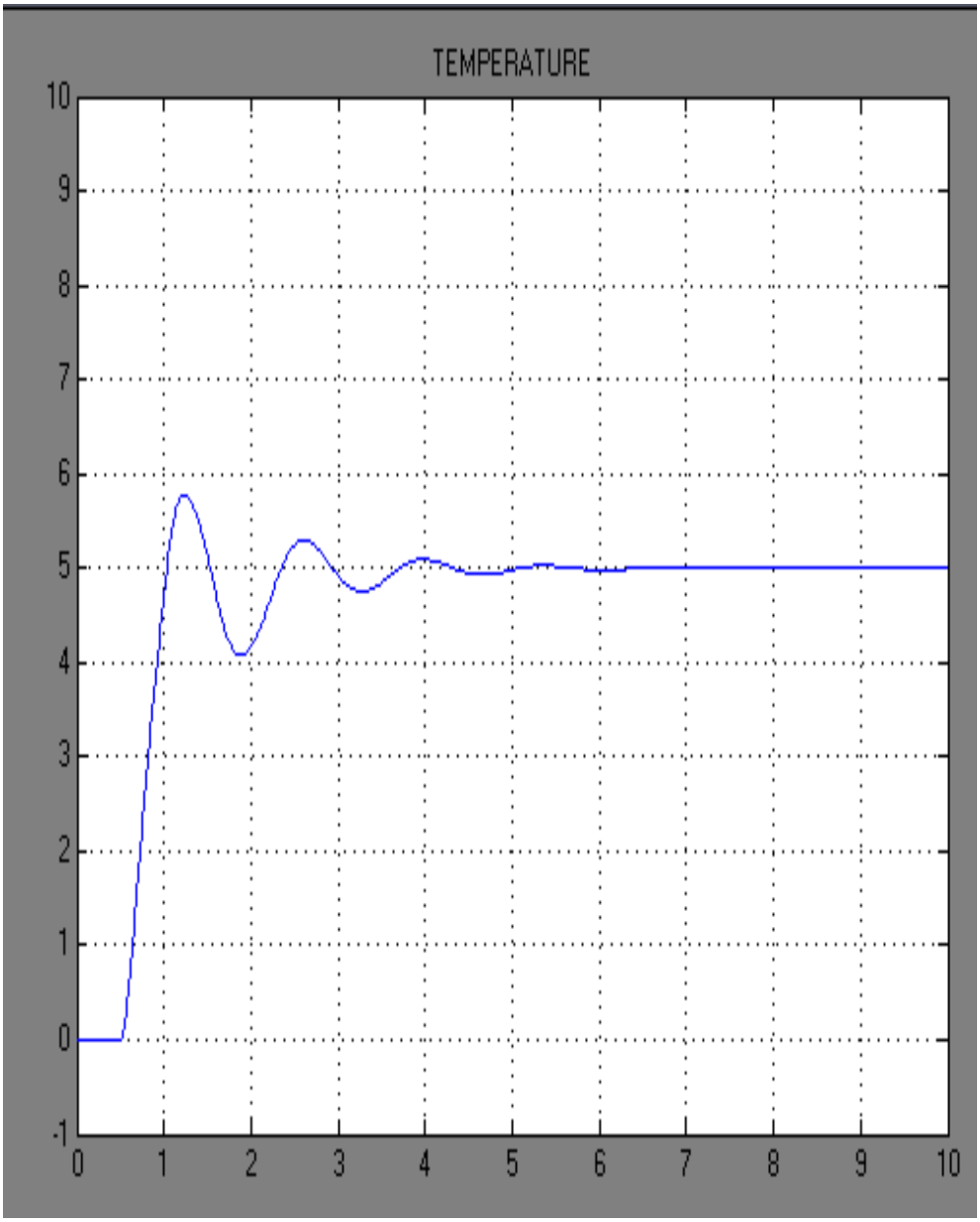
To run a simulation, first pick PARAMETERS under SIMULATION to set things like ODE integration scheme. Then, pick START under SIMULATION to run the simulation.

SIMULINK block diagrams for AUV Depth Control and Pipe Flow Temperature Control are attached. Also attached are Ziegler Nichols responses of each system to a step in command.


AUV DEPTH CONTROL
1.0
s
POLE
1
50.0s+50.0
PLANT
Abs
|u|
DRAG
Product
0.0
Gain
Sum1
LAG
LOAD
SIGNAL
1
0.5s+0.1
DRIVE
DEPTH
LIMITS
PID
PID
Sum
COMMAND


DEPTH VS TIME
0
2
4
6
8
10
12
14
16
18
20
0
10
20
30
40
50
60
70
80
90
100


LAB SETUP
1.0
0.25s+1.0
PLANT
Sum1
LOAD
LAG
1.0
0.1s+1.0
DRIVE
Scope2
PID
PID
Scope1
LIMITS
Scope
Sum2
BIAS
Look-Up
Table1
Relay
Sum
COMMAND
Look-Up
Table


TEMPERATURE
10
9
8
7
6
5
4
3
2
1
0
-1
0
1
2
3
4
5
6
7
8
9
10

EXPERIMENTAL METHODS

MATLAB TUTORIAL

The equations governing the attitude of the Apollo rocket relative to the vertical are:

$$J\ d^2R/dt^2 - I\ R = B - G + D$$

$$G = M\ H \qquad H = N\ dR/dt$$

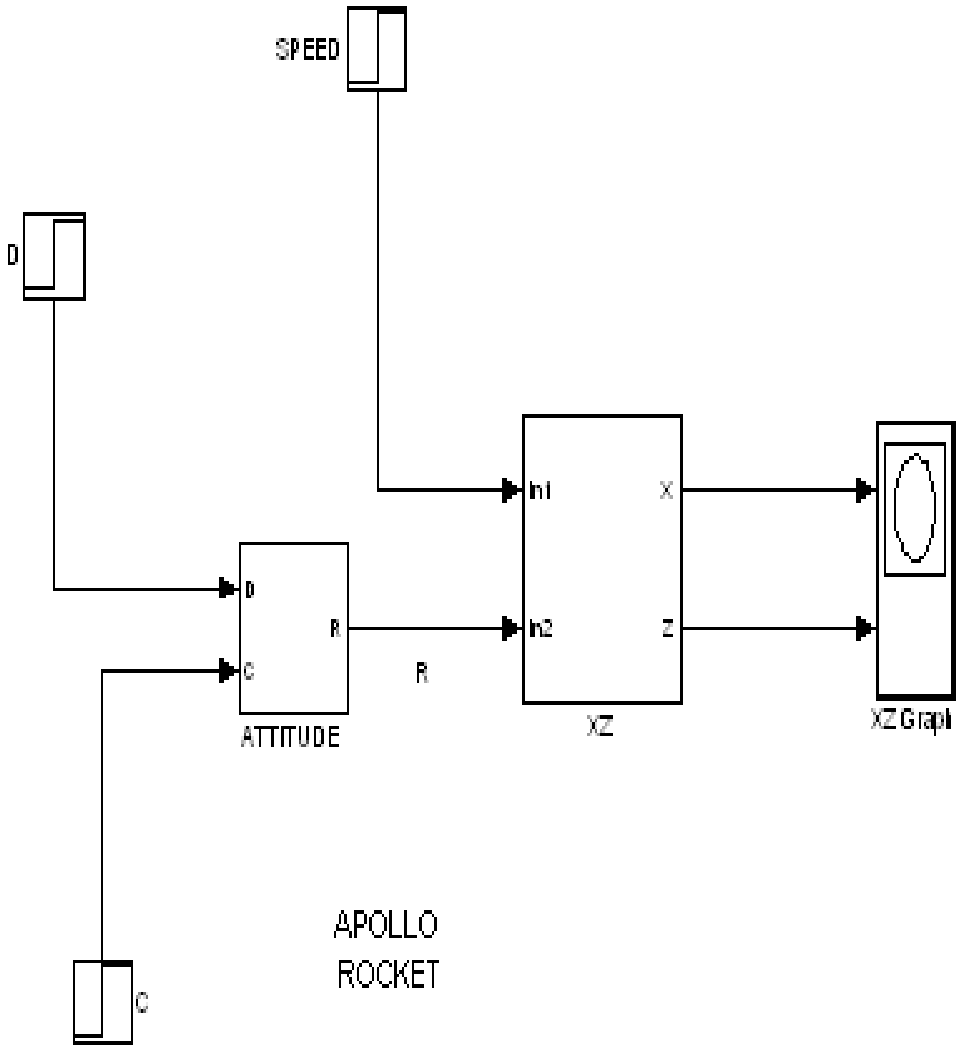
$$B = P\ Q \qquad Q = K\ (C-R)$$

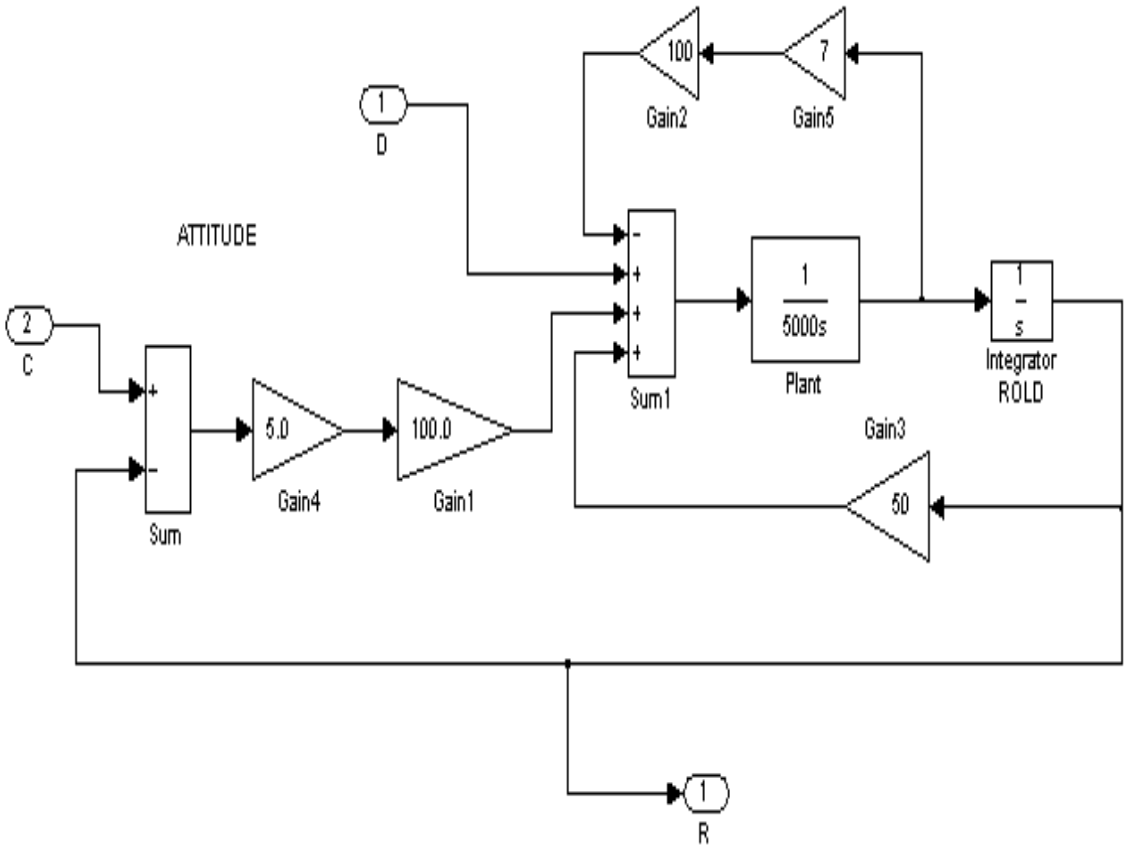
where R is the actual attitude of the rocket, C is the command or target attitude, B and G are control torques, D is a disturbance torque and J I M N P are plant and drive and controller constants.

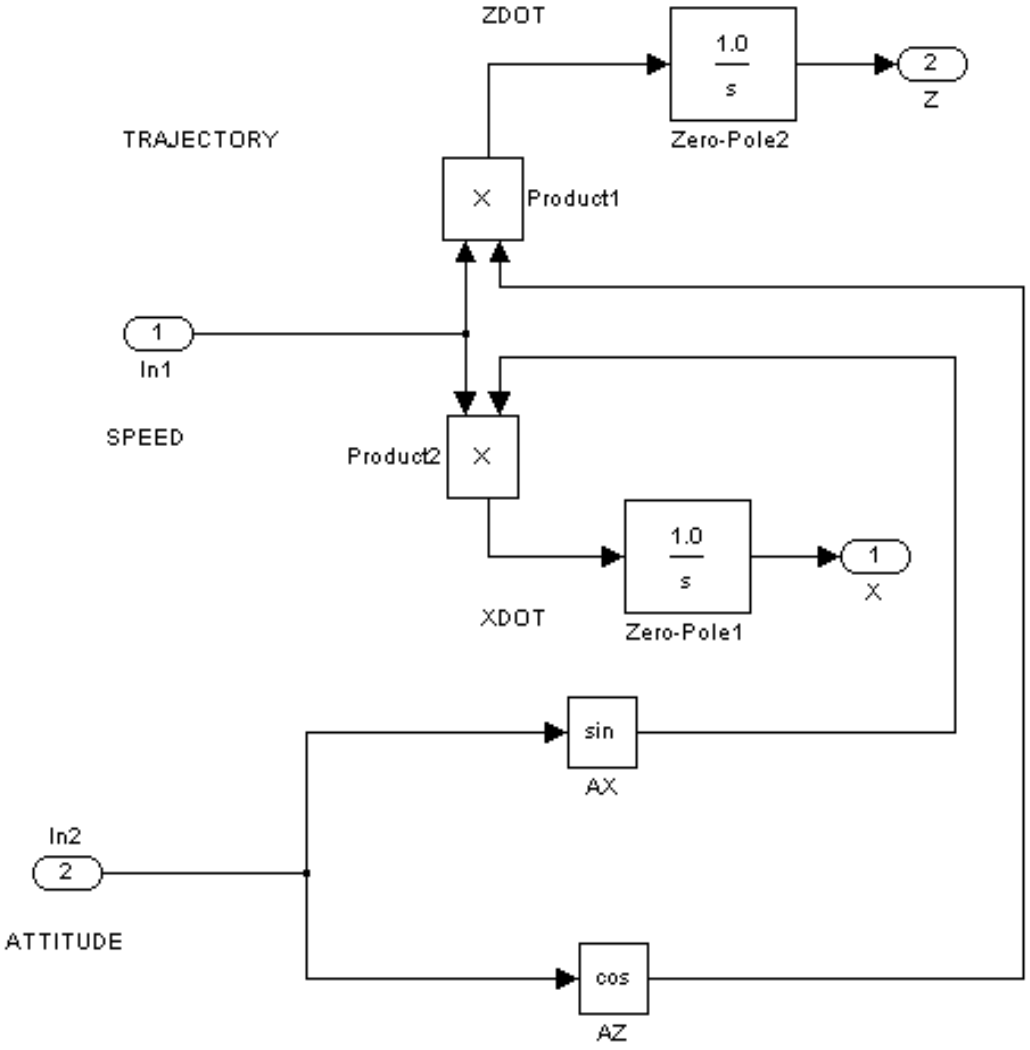
J=5000 I=50 M=100 N=7 P=100

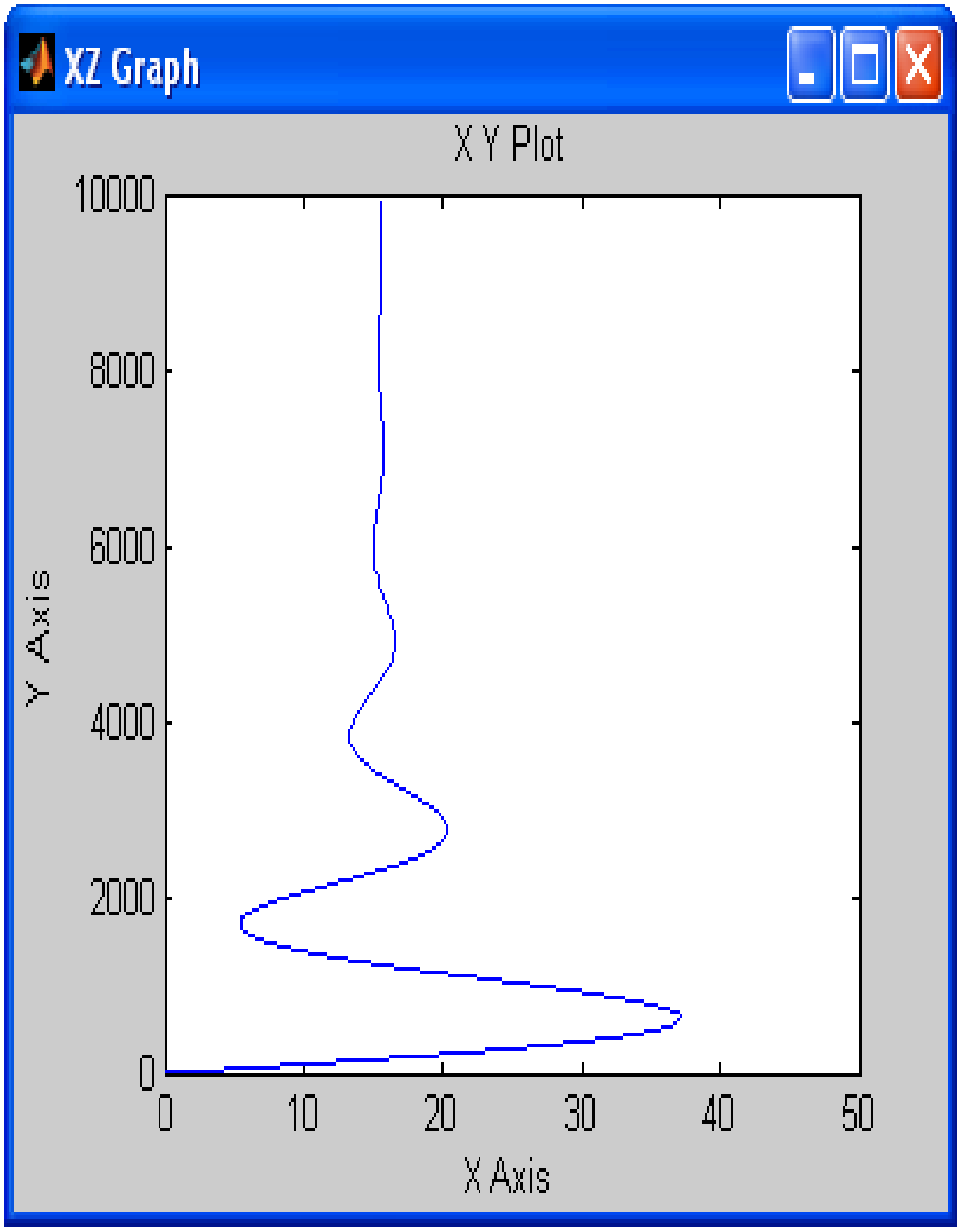
Determine **K** for borderline stable operation of the rocket. Develop a simulation template for the rocket. Write an m code based on this template. Use this to confirm the borderline gain **K**. Develop a SIMULINK block diagram for the rocket. Use this to confirm the borderline gain **K**. Add statements to the code and blocks to the block diagram to get the horizontal versus vertical trajectory of the rocket. For this let the speed of the rocket be S=100.

```
% APOLLO ROCKET
% simulation data
  clear all
  J=5000.0;I=50.0;M=100.0;N=7.0;
  P=100.0;COMMAND=0.0;GAIN=5.0;
  ROLD=0.1;UOLD=0.0;LOAD=0.0;
  XOLD=0.0;YOLD=0.0;TIME=0.0;
  DELT=0.01;SPEED=100.0;
% simulation loop
  for K=1:10000
  TIME=TIME+DELT;
  RNEW=ROLD+DELT*UOLD;
  SIGNAL=GAIN*(COMMAND-ROLD);
  B=P*SIGNAL;H=N*UOLD;G=M*H;
  RATE=(B-G+LOAD+I*ROLD)/J;
  UNEW=UOLD+DELT*RATE;
  XDOT=SPEED*sin(RNEW);
  YDOT=SPEED*cos(RNEW);
  XNEW=XOLD+DELT*XDOT;
  YNEW=YOLD+DELT*YDOT;
  ROLD=RNEW;UOLD=UNEW;
  XOLD=XNEW;YOLD=YNEW;
  R(K)=RNEW;U(K)=UNEW;
  X(K)=XNEW;Y(K)=YNEW;
  T(K)=TIME;
  end
% responses
  plot(T,R)
  plot(X,Y)
  title('APOLLO')
```


SPEED
D
In1
X
In2
Z
D
R
C
R
ATTITUDE
XZ
XZ Graph
APOLLO
ROCKET
C


ATTITUDE
1
D
2
C
100
Gain2
7
Gain5
Sum1
1
5000s
Plant
1
s
Integrator
ROLD
Gain3
50
Sum
5.0
Gain4
100.0
Gain1
1
R


ZDOT
1.0
s
2
Z
Zero-Pole2
TRAJECTORY
Product1
1
In1
SPEED
Product2
1.0
s
1
X
XDOT
Zero-Pole1
sin
AX
In2
2
ATTITUDE
cos
AZ


XZ Graph
X Y Plot
10000
8000
6000
4000
2000
0
Y Axis
0
10
20
30
40
50
X Axis

MATLAB CONTROLS OVERVIEW

One can get responses of control systems to various inputs by first forming the transfer function connecting input to output. Typically this is a ratio of two polynomials like:

$$TF = \frac{a\,S^2 + b\,S + c}{n\,S^4 + m\,S^3 + p\,S^2 + q\,S + r}$$

In matlab we represent these polynomials with arrays of coefficients in descending order:

```
>    num=[a b c];
>    den=[n m p q r];
```

We then form the transfer function as follows:

```
>    sys=tf(num,den);
```

We can also form the transfer functions for various parts of the system using the tf function and then use the series parallel and feedback functions to get the overall sys. Once the sys function is obtained we can get impulse and step responses as follows:

```
>    impulse(sys)
>    step(sys)
```

We can get the frequency response magnitude ratio MR and phase shift Φ for a system using the bode function:

```
>    bode(sys)
```

We can also get frequency response data using T(jω). For example if we set ω we can get data as follows:

```
>    S=complex(0.0,ω);
>    num=a*S^2+b*S+c;
>    den=n*S^4+m*S^3+p*S^2+q*S+r;
>    tf=num/den
```

This gives a complex number: P+Qi. Manipulation gives:

$$MR = \sqrt{[P^2+Q^2]} \qquad \Phi=\tan^{-1} Q/P.$$

To get responses using Partial Fraction Expansion PFE and Inverse Laplace Transformation ILT we can use the convolution function conv to form the numerator and the denominator of R(S). The residue function can then be used to do PFE. This gives the roots of R(S) together with its residues:

```
>    num=conv([a b],[n m]);
>    den=conv([x y z],[u v w]);
>    [r,p,k]=residue(num,den)
```

One can then use ILT to get R(t).

We can get the roots of a characteristics equation using the roots function. It could also be used to construct Root Locus Plots:

```
>     z=[0.1:0.1:10.0];
>     for k=1:length(z)
      q=[x y a*z(k)+b u v w];
      p(:,k)=roots(q);
>     end
>     plot(real(p),imag(p))
>     grid
```

When a parameter such as a gain can be isolated from GH, one can get Root Locus Plots using the rlocus function as follows:

```
>     num=[a b];
>     den=[x y z];
>     gh=tf(num,den);
>     rlocus(gh)
```

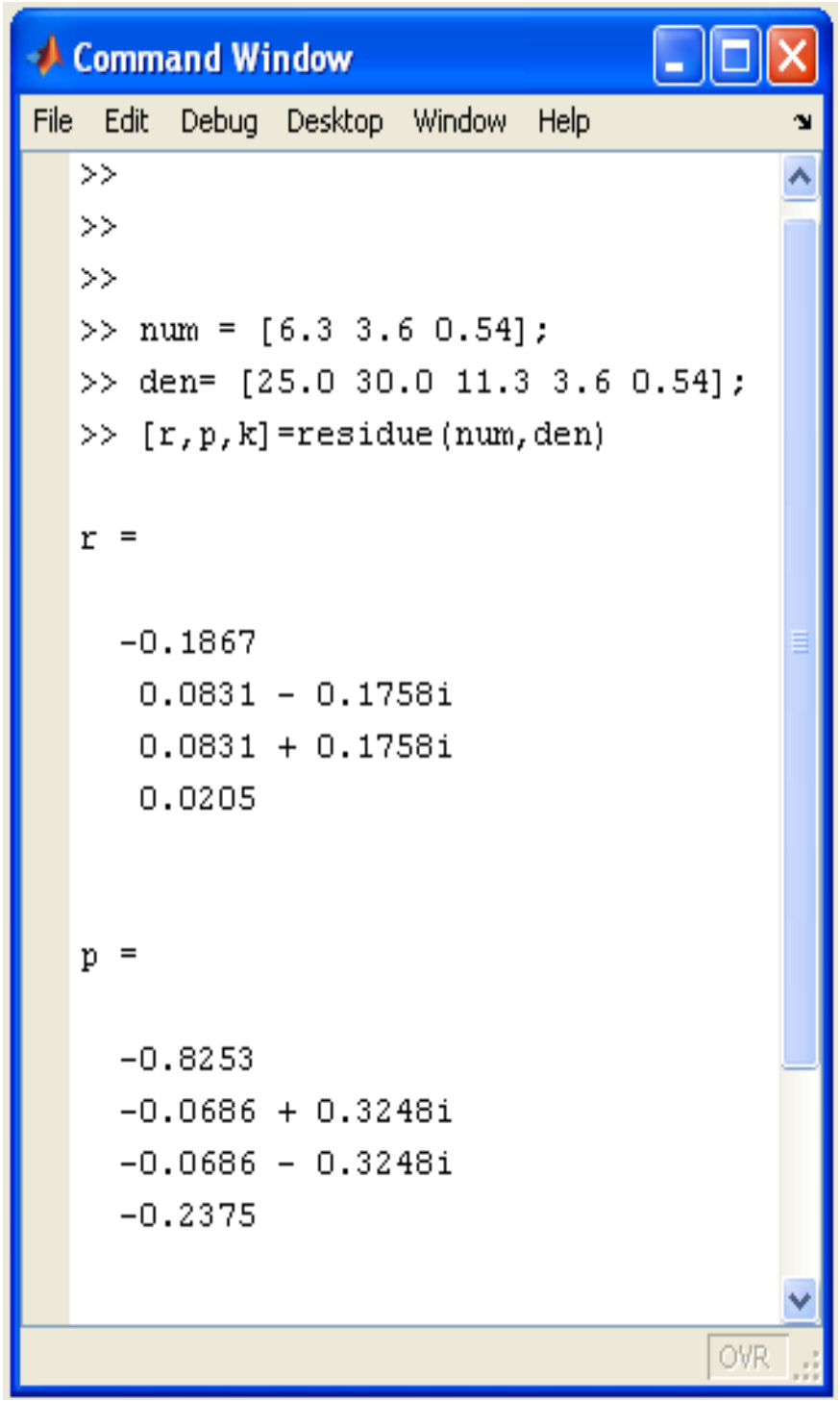
One can get GH Plots and Stability Margins as follows:

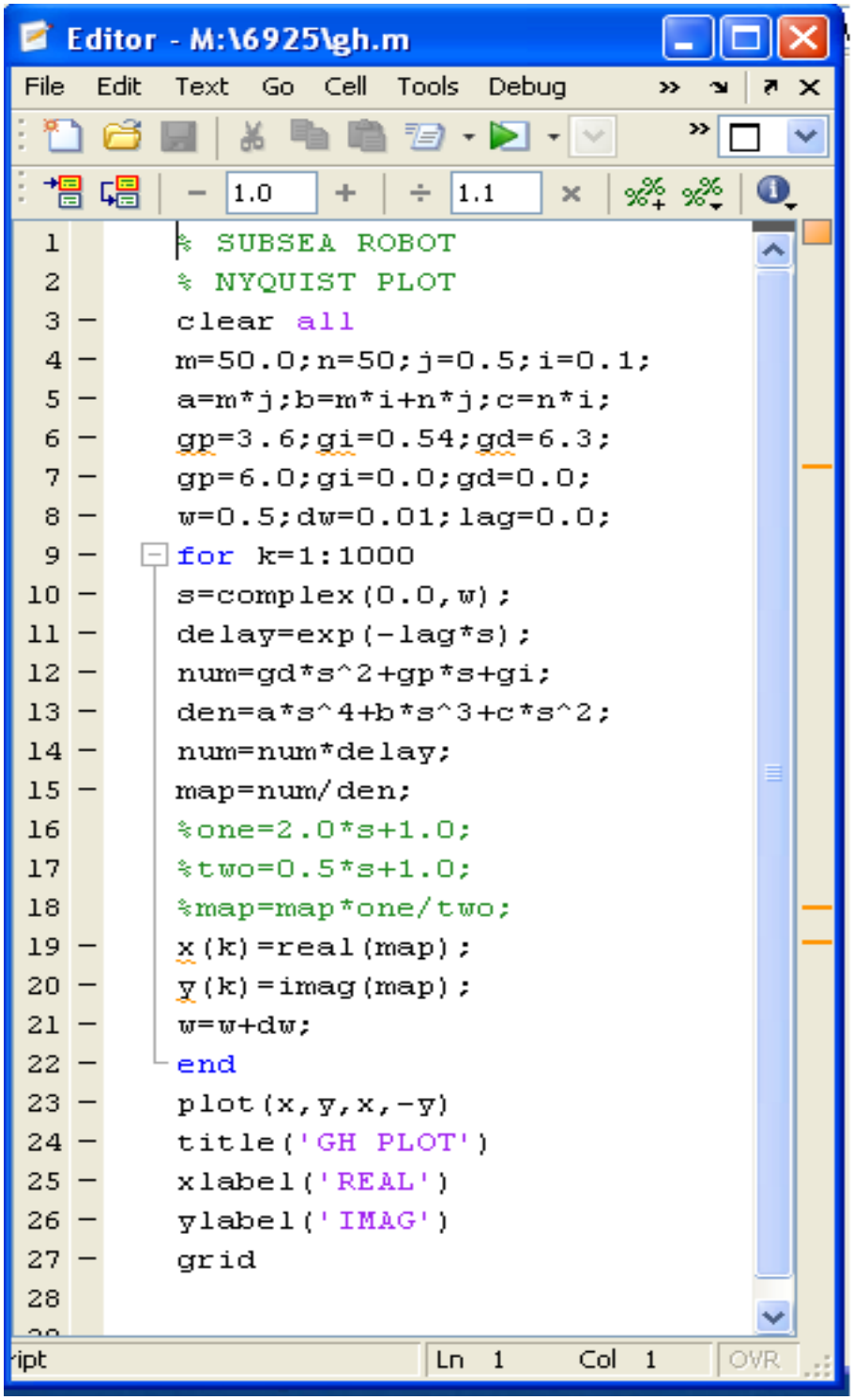
```
>     nyquist(gh)
>     bode(gh)
>     [mag,phase,w]=bode(gh);
>     [GM,PM,WG,WP]=margin(mag,phase,w)
```

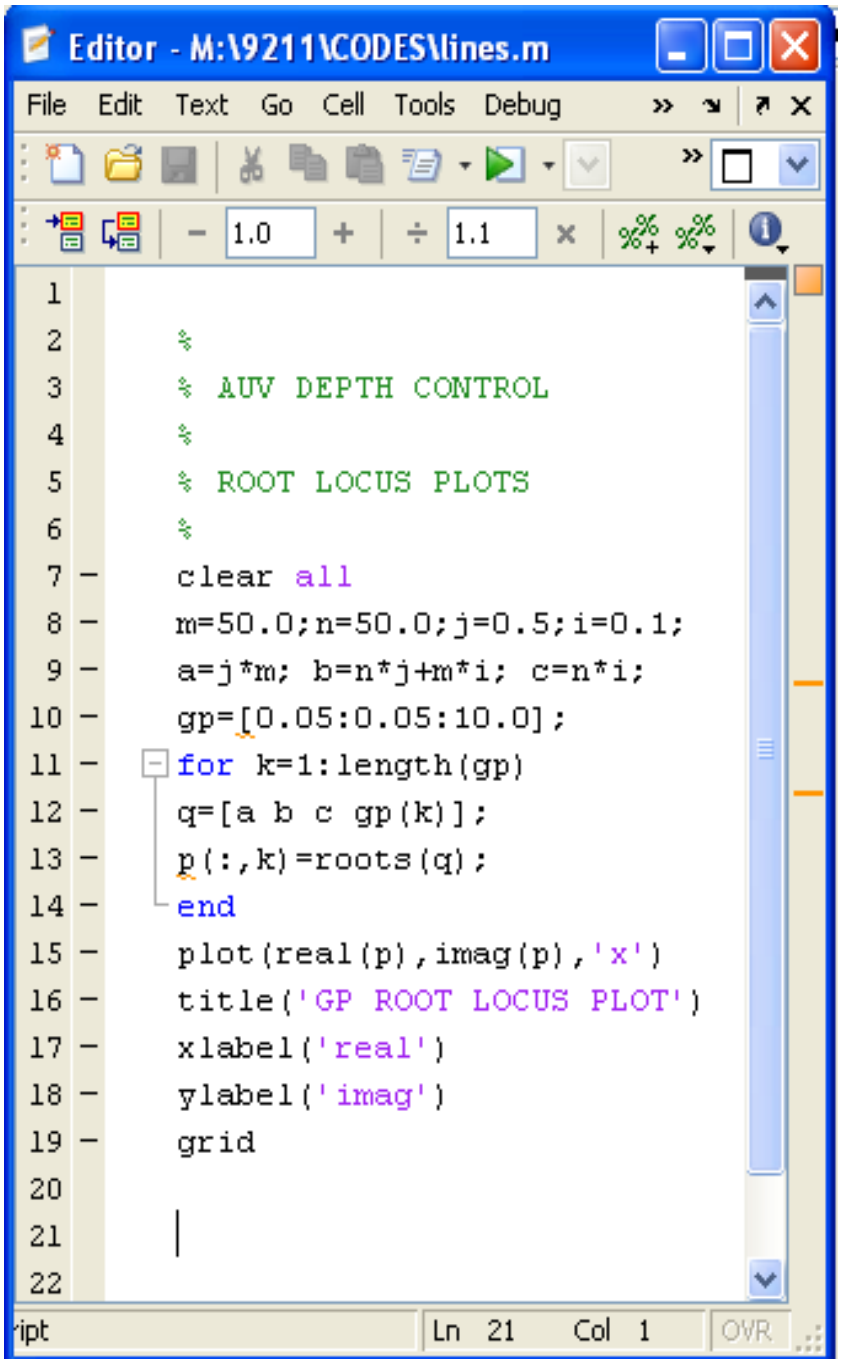
If a system has a time delay one can approximate it as a ratio of two polynomials using the Pade Approximant as follows:

```
>     [a,b]=pade(T,p);
```

The time delay is T and the order of the approximant is p. The numerator of the approximant is a and its denominator is b. One can use tf(a,b) to get an approximate transfer function for the delay. An exact transfer function for a delay is: exp(-T*S).


Command Window
File Edit Debug Desktop Window Help
>>
>>
>>
>> num = [6.3 3.6 0.54];
>> den= [25.0 30.0 11.3 3.6 0.54];
>> [r,p,k]=residue(num,den)
r =
-0.1867
0.0831 - 0.1758i
0.0831 + 0.1758i
0.0205
p =
-0.8253
-0.0686 + 0.3248i
-0.0686 - 0.3248i
-0.2375
OVR


Editor - M:\6925\gh.m
File Edit Text Go Cell Tools Debug
1.0
1.1
% SUBSEA ROBOT
% NYQUIST PLOT
clear all
m=50.0;n=50;j=0.5;i=0.1;
a=m*j;b=m*i+n*j;c=n*i;
gp=3.6;gi=0.54;gd=6.3;
gp=6.0;gi=0.0;gd=0.0;
w=0.5;dw=0.01;lag=0.0;
for k=1:1000
s=complex(0.0,w);
delay=exp(-lag*s);
num=gd*s^2+gp*s+gi;
den=a*s^4+b*s^3+c*s^2;
num=num*delay;
map=num/den;
%one=2.0*s+1.0;
%two=0.5*s+1.0;
%map=map*one/two;
x(k)=real(map);
y(k)=imag(map);
w=w+dw;
end
plot(x,y,x,-y)
title('GH PLOT')
xlabel('REAL')
ylabel('IMAG')
grid
Ln 1 Col 1
OVR


Editor - M:\9211\CODES\lines.m
File Edit Text Go Cell Tools Debug
1.0
1.1
%
% AUV DEPTH CONTROL
%
% ROOT LOCUS PLOTS
%
clear all
m=50.0;n=50.0;j=0.5;i=0.1;
a=j*m; b=n*j+m*i; c=n*i;
gp=[0.05:0.05:10.0];
for k=1:length(gp)
q=[a b c gp(k)];
p(:,k)=roots(q);
end
plot(real(p),imag(p),'x')
title('GP ROOT LOCUS PLOT')
xlabel('real')
ylabel('imag')
grid
Ln 21 Col 1
OVR

REVIEW OF LAPLACE TRANSFORMATION

Laplace Transformation converts ordinary differential equations or ODEs into algebraic equations or AEs. Manipulation of the AEs followed by Inverse Laplace Transformation gives responses back in time. Manipulation of the AEs also gives the system transfer functions or TFs. Most control theories are based on TFs.

The Laplace Transform Integral is:

$$F(S) = \mathfrak{I}\,[f(t)] = \int_{o}^{\infty} f(t)\, e^{-St}\, dt \quad .$$

Usually, mechanical engineers do not have to evaluate this integral. All of the important cases have already been worked out.

Some Laplace Transform (LT) pairs used to reduce ODEs to AEs are:

$$\mathfrak{I}\,[df/dt] = S\,F(S) - f(0) \qquad \mathfrak{I} \int f d\tau = F(S) / S$$

$$\mathfrak{I}\,[d^2f/dt^2] = S^2\,F(S) - S\,f(0) - df(0)/dt$$

$$\mathfrak{I}\,[d^3f/dt^3] = S^3\,F(S) - S^2\,f(0) - S\,df(0)/dt - d^2f(0)/dt^2$$

Usually, initial condition terms are set to zero for control because, in most cases, a system starts from some rest state.

Manipulation of algebraic equations often gives factors of the form: $\Gamma/(S-\lambda)$. Inverse Transformation gives back in time: $\Gamma e^{+\lambda t}$.

Typical commands/disturbances into control systems include: a step with height A / a pulse with height A and short duration T / a sine or cosine wave with amplitude A and frequency ω / a linear ramp in time with slope A. Laplace Transform pairs for these are:

$$\mathfrak{I}(\text{Step with Height A}) = A/S$$

$$\mathfrak{I}(\text{Short Duration Pulse}) = AT$$

$$\mathfrak{I}(\text{Sine Wave}) = A\omega/(S^2+\omega^2)$$

$$\mathfrak{I}(\text{Cosine Wave}) = AS/(S^2+\omega^2)$$

$$\mathfrak{I}(\text{Linear Ramp}) = A/S^2 \quad .$$

Control systems often have time delays or transport lags inherent in them. These can seriously degrade performance. When a signal is delayed in time by T seconds, Laplace Transformation gives:

$$\mathfrak{I}\ (f[t-T]) = e^{-ST}\ F(S) \ .$$

The Final Value Theorem states that

$$\lim_{t\to\infty} f(t) = \lim_{S\to 0} S\ F(S) \quad .$$

This can be used to get the final state of stable systems subjected to step commands or step disturbances. Ideally for a step command the final state should be equal to the command while for a step disturbance the final state should be zero. The Final Value Theorem gives unrealistic results when systems are unstable.

COMPLEX NUMBERS AND COMPLEX PLANES

There are two ways to represent complex numbers. These are shown schematically in Figure 1:

Cartesian $z = x + y\,j$ Polar $z = r\angle\theta = re^{j\theta}$.

Manipulations give:

$r\,e^{+j\theta} = r\,\mathrm{Cos}\theta + j\,r\,\mathrm{Sin}\theta$ $r\,e^{-j\theta} = r\,\mathrm{Cos}\theta - j\,r\,\mathrm{Sin}\theta$

$\mathrm{Sin}\theta = (e^{+j\theta} - e^{-j\theta})/\,2j$ $\mathrm{Cos}\theta = (e^{+j\theta} + e^{-j\theta})/\,2$.

When adding or subtracting complex numbers, it is easier to use the Cartesian representation. Take two complex numbers z_1 and z_2:

$z_1 = x_1 + y_1\,j$ $z_2 = x_2 + y_2\,j$

where x_1 y_1 x_2 y_2 are known. One gets:

$z_1 + z_2 = (x_1 + x_2) + (y_1 + y_2)\,j$

$z_1 - z_2 = (x_1 - x_2) + (y_1 - y_2)\,j$.

When multiplying or dividing complex numbers, it is best to use the Polar representation. Take two complex numbers z_1 and z_2:

$$z_1 = r_1 \angle \theta_1 \qquad z_2 = r_2 \angle \theta_2$$

where $r_1\ \theta_1\ r_2\ \theta_2$ are known. One gets:

$$z_1\ z_2 = r_1\ r_2\ \angle\ \theta_1 + \theta_2\ =$$

$$(x_1\ x_2 - y_1\ y_2) + (x_1\ y_2 + x_2\ y_1)\ j$$

$$z_1\ /\ z_2 = r_1\ /\ r_2 \angle\ \theta_1 - \theta_2\ =$$

$$[\ (\ x_1\ x_2 + y_1\ y_2\ )\ +\ (\ x_2\ y_1 - x_1\ y_2\ )\ j\ ]\ /\ [\ x_2\ x_2 + y_2\ y_2\ ]\ .$$

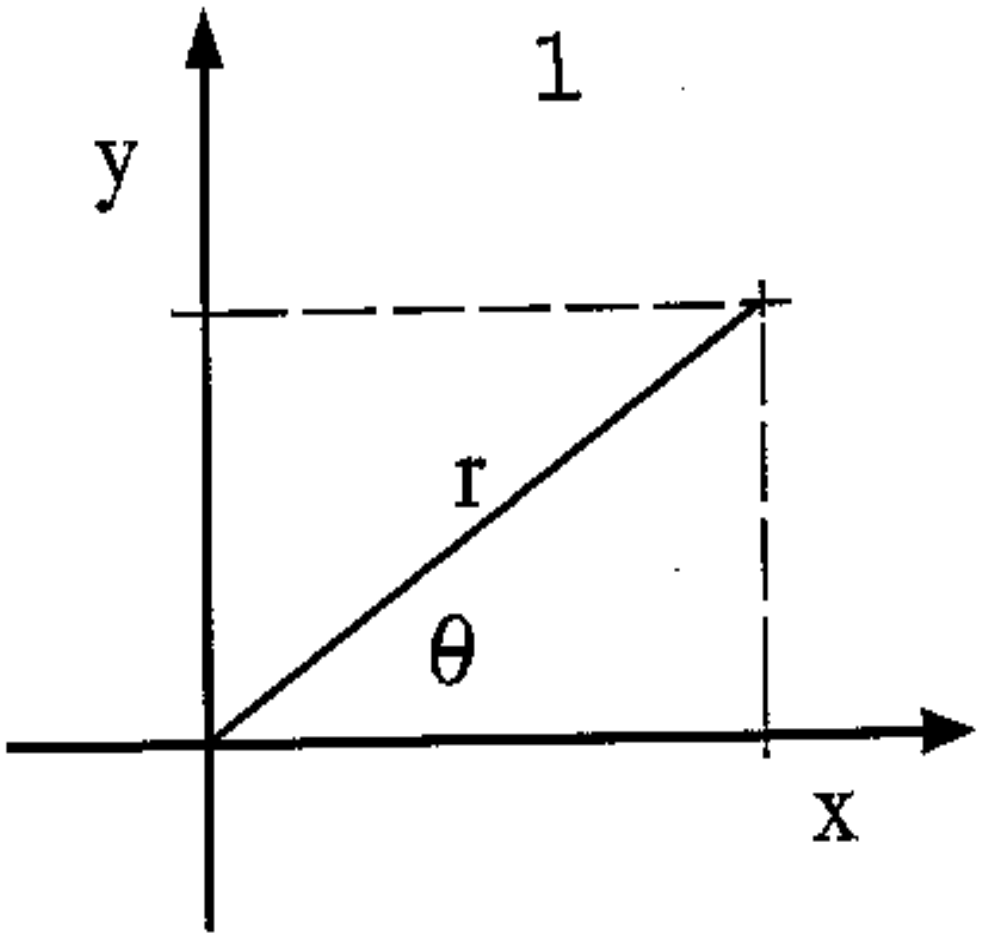
The Nyquist Procedure for checking stability of feedback control systems maps a closed contour in one complex plane known as the S plane to another complex plane known as the GH plane. A function of S known as the system GH function is the mapping function. As an illustration consider a system with the GH function:

$$GH = 2\ /\ [\ S\ (\ S^2 + S + 1)\ ]\quad .$$

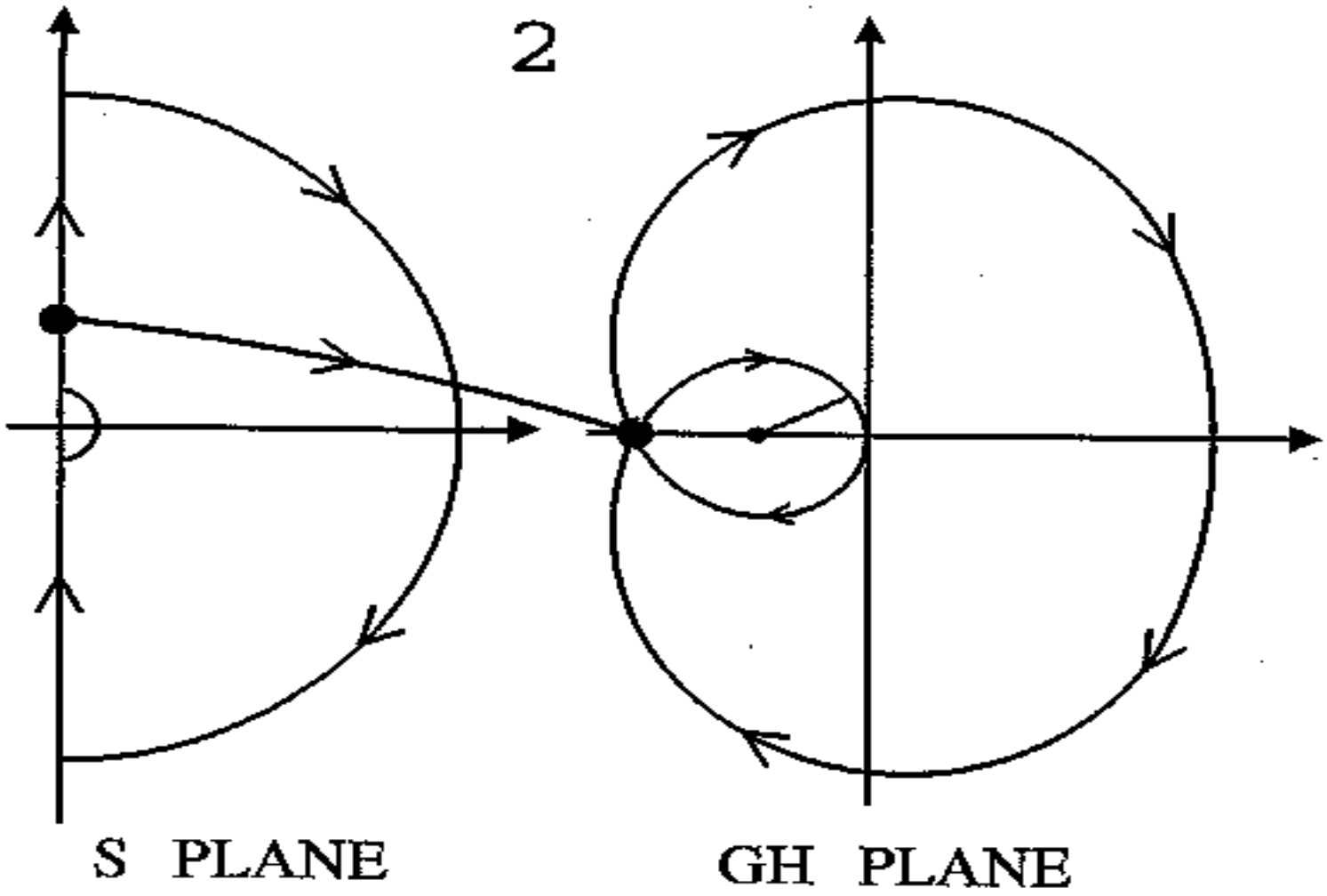
The contour in the S plane that is mapped to the GH plane surrounds the entire right half of the S plane. It is shown in Figure 2. To illustrate the mapping, consider the point S = +j on the S plane contour. Substitution into the GH function gives:

$$GH = 2/[j(j^2+j+1)] = 2/[j(-1+j+1)] = 2/[j(j)] = -2\quad .$$

This point and the complete contour in the GH plane are shown in Figure 2. The GH plot shows that the system is unstable!



$$x + yj = r\angle\theta = re^{j\theta}$$

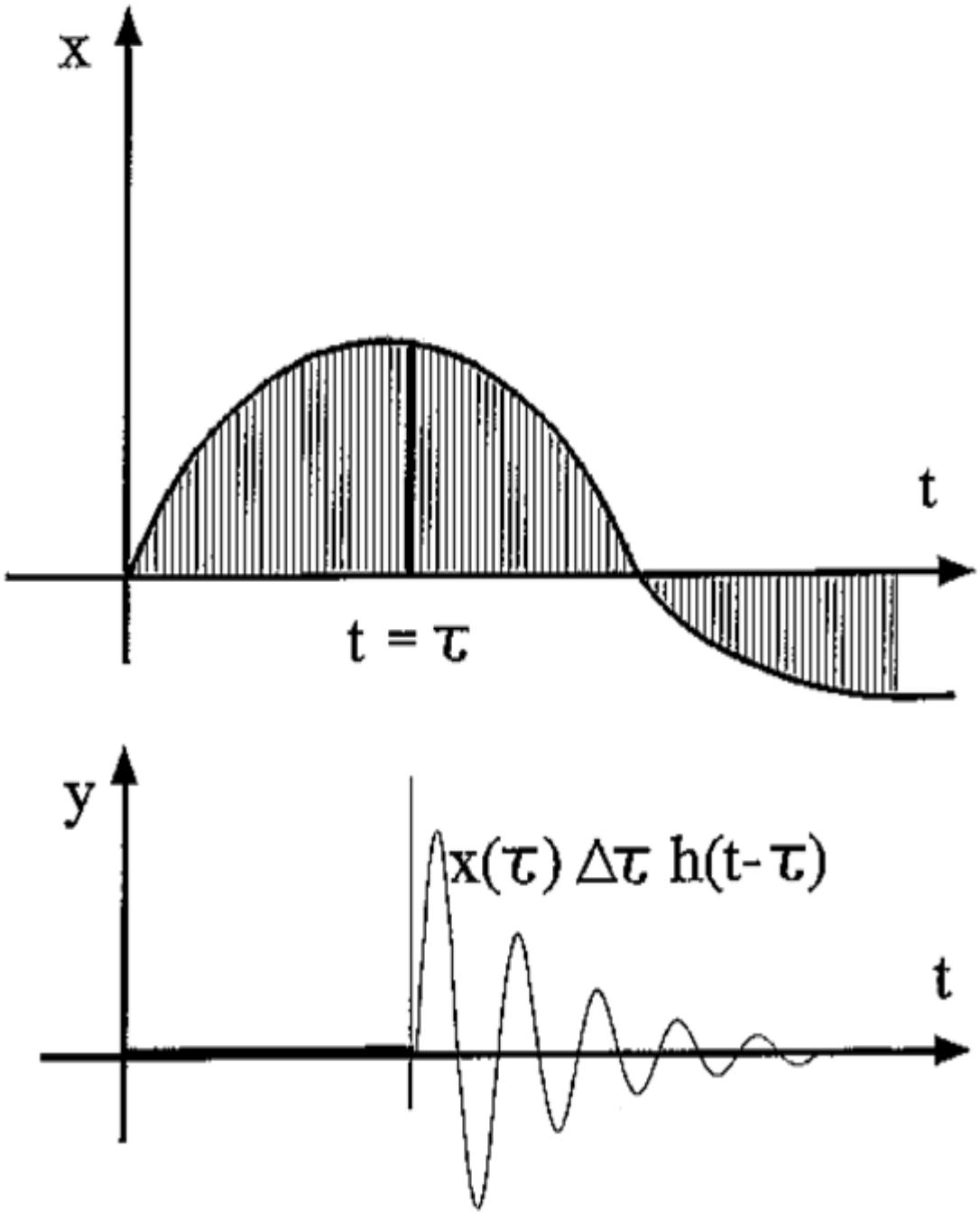

2
S PLANE
GH PLANE

SYSTEM TRANSFER FUNCTIONS

The response of a system to a unit impulse input at time t = 0 is usually denoted by h(t). Its Laplace Transform **H**(S) is known as a Unit Impulse Response Function. It turns out that: TF(S) = **H**(S) where TF(S) is a system transfer function. For a system with input x and output y, this is: TF(S) = Y(S)/X(S).

Why is TF(S) = **H**(S)? In other words: What is so special about the unit impulse as an input? As shown in the sketch below, any input x(t) into a system can be broken down into a sequence or train of pulses. Superposition of the pulses generates a staircase like approximation to x(t). In the limit as the pulse duration tends to zero, this becomes exact. The pulse that comes on at time t = τ has strength x(τ) and duration $\Delta\tau$: thus its area is x(τ)$\Delta\tau$. In the limit as $\Delta\tau$ tends to zero, this pulse becomes basically a scaled version of the unit impulse. The scaling factor is infinitesimal and is the area x(τ)$\Delta\tau$. The response of the system due to a unit impulse input at t = τ is h(t-τ). The response due to an impulse with area x(τ)$\Delta\tau$ at t = τ is x(τ)$\Delta\tau$ h(t-τ). The response at some point in time due to all of the impulses up to that time is:

$$y(t) = \sum_{n=0}^{N} x(n\Delta\tau)\Delta\tau\ h(t-n\Delta\tau)$$


x
t
t = τ
y
x(τ) Δτ h(t-τ)
t

where N is the total number of impulses and n denotes a specific impulse. In the limit as $\Delta\tau$ tends to zero, N tends to infinity, and the summation becomes the Convolution Integral:

$$y(t) = \int_{o}^{t} x(\tau)\, h(t-\tau)\, d\tau \quad .$$

Laplace Transformation of this integral gives:

$$Y(S) = X(S)\ \mathbf{H}(S) = X(S)\ TF(S) \quad .$$

Impulses do not exist in reality. In other words, there is no such thing as an infinite strength, infinitesimal duration, signal. But strong, short duration, pulses do exist, and they often mimic the unit impulse. Why consider the unit impulse as an input? An impulse jars a system to some state and the motion thereafter is pure transient. If transients grow, the system is unstable: if transients decay, the system is stable. Manipulation of **H**(S) gives: **H**(S) = **N**(S)/**D**(S) where **N**(S) and **D**(S) are polynomials. PFE applied to **N**(S)/**D**(S) gives factors of the form: **N**(S)/**D**(S) = $\Sigma\ \Gamma/[S-\lambda]$. ILT applied to these factors gives: $h(t) = \Sigma\ \Gamma\ e^{+\lambda t}$. Each λ is a value of S which satisfies the characteristic equation **D**(S)=0. For stable operation, each root λ must be a negative real number or a complex number with a negative real part.

To get the response of a system to an input, one starts with:

$$Y(S) = TF(S)\ X(S)$$

Partial Fraction Expansion or PFE gives

$$Y(S) = \Sigma\ A/[S-a]$$

Inverse Laplace Transformation or ILT gives

$$y(t) = \Sigma\ A\ e^{+at}$$

Manipulation of this then gives the response in terms of exponential and trigonometric functions in time.

The MATLAB residue function can be used to simplify PFE. The input into this would be the numerator and denominator of Y(S). The output would be the roots "a" and the residues "A".

```
>    num=[a b c];
>    den=[x y z u v w];
>    [r,p,k]=residue(num,den)
```

Here r indicates residues and p indicates roots or poles.

AUTONOMOUS UNDERWATER VEHICLE

DEPTH CONTROL RESPONSES

To illustrate application of the Laplace Transformation procedure we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. According to Newton's Second Law of Motion, the equation governing the up and down motion of the auv is:

$$M\ d^2R/dt^2 \ = \ B \ + \ D \ - \ W$$

where R is the depth of the auv, M is its overall mass, B is the control force from the propulsion system, D is a disturbance load caused for example by sudden weight changes and W is a drag load consisting of wake drag and wall drag:

$$W \ = \ X\ dR/dt\ |dR/dt| \ + \ Y\ dR/dt$$

where X and Y account for the size and shape of the auv.

A simple model of the propulsion system is:

$$J\ dB/dt \ + \ I\ B \ = \ Q$$

where Q is the control signal: J and I are drive constants.

The PID error driven strategy lets the control signal Q be:

$$Q \ = \ K_P\ E \ + \ K_I \int E d\tau \ + \ K_D\ dE/dt$$

where E = C - R is the depth error and K_P K_I K_D are the controller gains: C is the command depth.

Laplace Transformation of the governing equations gives

$$(M\ S^2 + N\ S)\ R = B + D$$

$$(J\ S + I)\ B = Q$$

$$Q = (K_P + K_I/S + K_D\ S)\ E$$

$$E = C - R$$

Algebraic manipulation gives

$$\frac{R}{C} = \frac{K_D\ S^2 + K_P\ S + K_I}{MJ\ S^4 + (NJ+MI)\ S^3 + (NI+K_D)\ S^2 + K_P\ S + K_I}$$

$$\frac{R}{D} = \frac{J\ S^2 + I\ S}{MJ\ S^4 + (NJ+MI)\ S^3 + (NI+K_D)\ S^2 + K_P\ S + K_I}$$

To give a numerical example we will let the parameters be:

M = 50.0 N = 50.0. J = 0.5 I = 0.1

One can show that the Ziegler Nichols gains are:

$K_P = 3.6$ $K_I = 0.54$ $K_D = 6.3$

Substitution gives

$$\frac{R}{C} = \frac{6.3\ S^2 + 3.6\ S + 0.54}{25.0\ S^4 + 30.0\ S^3 + 11.3\ S^2 + 3.6\ S + 0.54}$$

$$\frac{R}{D} = \frac{0.5\ S^2 + 0.1\ S}{25.0\ S^4 + 30.0\ S^3 + 11.3\ S^2 + 3.6\ S + 0.54}$$

For the command case we will work through 4 cases: unit impulse; unit step; unit sine; unit ramp. For a unit impulse C(S)=1. In this case

$$R = \frac{6.3\ S^2 + 3.6\ S + 0.54}{25.0\ S^4 + 30.0\ S^3 + 11.3\ S^2 + 3.6\ S + 0.54}$$

Dividing through top and bottom by 25.0 gives

$$R = \frac{0.252\ S^2 + 0.144\ S + 0.0216}{S^4 + 1.2\ S^3 + 0.452\ S^2 + 0.144\ S + 0.0216}$$

One can put this in the factored form

$$R = \frac{0.252\ S^2 + 0.144\ S + 0.0216}{(S-a)\ (S-b)\ (S-v)\ (S-w)}$$

$$= \frac{A}{(S-a)} + \frac{B}{(S-b)} + \frac{V}{(S-v)} + \frac{W}{(S-w)}$$

Partial Fraction Expansion followed by Inverse Laplace Transformation gives

$$A\,e^{+at} + B\,e^{+bt} + (X+Yj)\,e^{+(x+yj)t} + (X-Yj)\,e^{+(x-yj)t}$$

where

$$v = x + yj \qquad w = x - yj$$
$$V = X + Yj \qquad W = X - Yj$$

Manipulation gives

$$A\,e^{+at} + B\,e^{+bt} + 2X\,e^{+xt}\,\mathrm{Cos}[yt] - 2Y\,e^{+xt}\,\mathrm{Sin}[yt]$$

The MATLAB residue function gives

$$a = -0.8253 \qquad b = -0.2375$$
$$x = -0.0686 \qquad y = +0.3248$$
$$A = -0.1867 \qquad B = +0.0205$$
$$X = +0.0831 \qquad Y = -0.1758$$

For a step case $C(S) = C_o/S$ where C_o is the height of the step. In this case one gets

$$R = \frac{0.252\,S^2 + 0.144\,S + 0.0216}{(S-a)\,(S-b)\,(S-v)\,(S-w)}\,\frac{C_o}{S}$$

$$\frac{A}{(S-a)} + \frac{B}{(S-b)} + \frac{V}{(S-v)} + \frac{W}{(S-w)} + \frac{Z}{S}$$

Partial Fraction Expansion followed by Inverse Laplace Transformation gives

$$A\, e^{+at} + B\, e^{+bt} + Z + 2X\, e^{+xt}\, \mathrm{Cos}[yt] - 2Y\, e^{+xt}\, \mathrm{Sin}[yt]$$

The MATLAB residue function gives when $C_o = 1.0$:

$$a = -0.8253 \quad b = -0.2375$$
$$x = -0.0686 \quad y = +0.3248$$
$$Z = 1.0$$
$$A = +0.2262 \quad B = -0.0863$$
$$X = -0.5700 \quad Y = -0.1355$$

For a sine case one gets

$$R = \frac{0.252\, S^2 + 0.144\, S + 0.0216}{(S-a)\,(S-b)\,(S-v)\,(S-w)} \; \frac{C_o\, \omega}{(S^2+\omega^2)}$$

$$\frac{A}{(S-a)} + \frac{B}{(S-b)} + \frac{V}{(S-v)} + \frac{W}{(S-w)} + \frac{N}{(S-j\omega)} + \frac{M}{(S+j\omega)}$$

where $N=G+Hj$ and $M=G-Hj$. Partial Fraction Expansion followed by Inverse Laplace Transformation gives

$$A\, e^{+at} + B\, e^{+bt} + 2X\, e^{+xt}\, \mathrm{Cos}[yt] - 2Y\, e^{+xt}\, \mathrm{Sin}[yt] + 2G\, \mathrm{Cos}[\omega t] - 2H\, \mathrm{Sin}[\omega t]$$

The MATLAB residue function gives when C_o = 1.0 and ω = 0.3:

a = -0.8253 b = -0.2375
x = -0.0686 y = +0.3248
A = -0.0726 B = +0.0420
X = +0.9906 Y = +0.7993
G = -0.9753 H = -1.0085

For a ramp case one gets

$$R = \frac{0.252\,S^2 + 0.144\,S + 0.0216}{(S-a)\,(S-b)\,(S-v)\,(S-w)}\,\frac{C_o}{S^2}$$

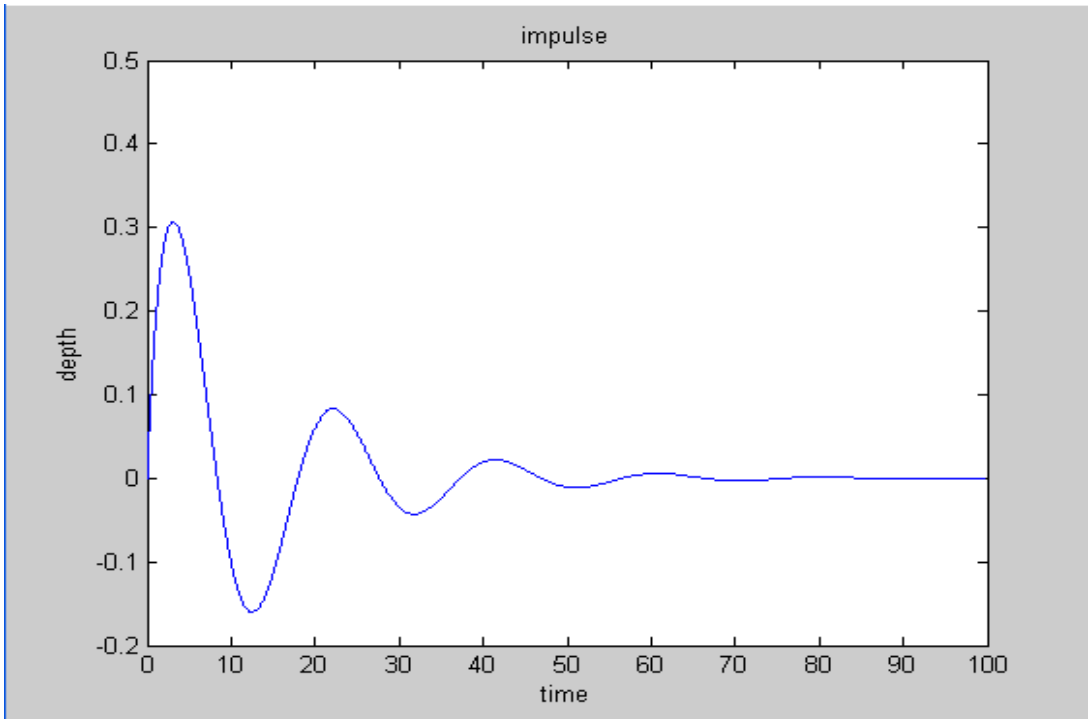
$$\frac{A}{(S-a)} + \frac{B}{(S-b)} + \frac{V}{(S-v)} + \frac{W}{(S-w)} + \frac{N}{S} + \frac{M}{S^2}$$

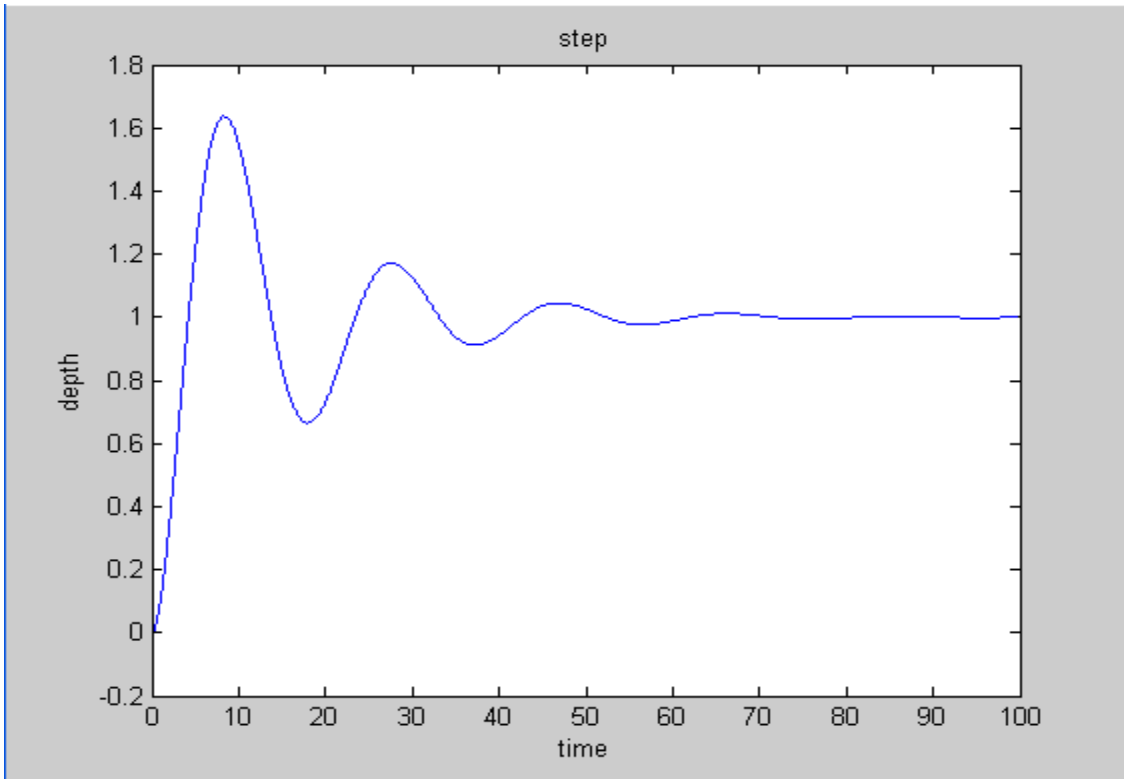
Partial Fraction Expansion followed by Inverse Laplace Transformation gives

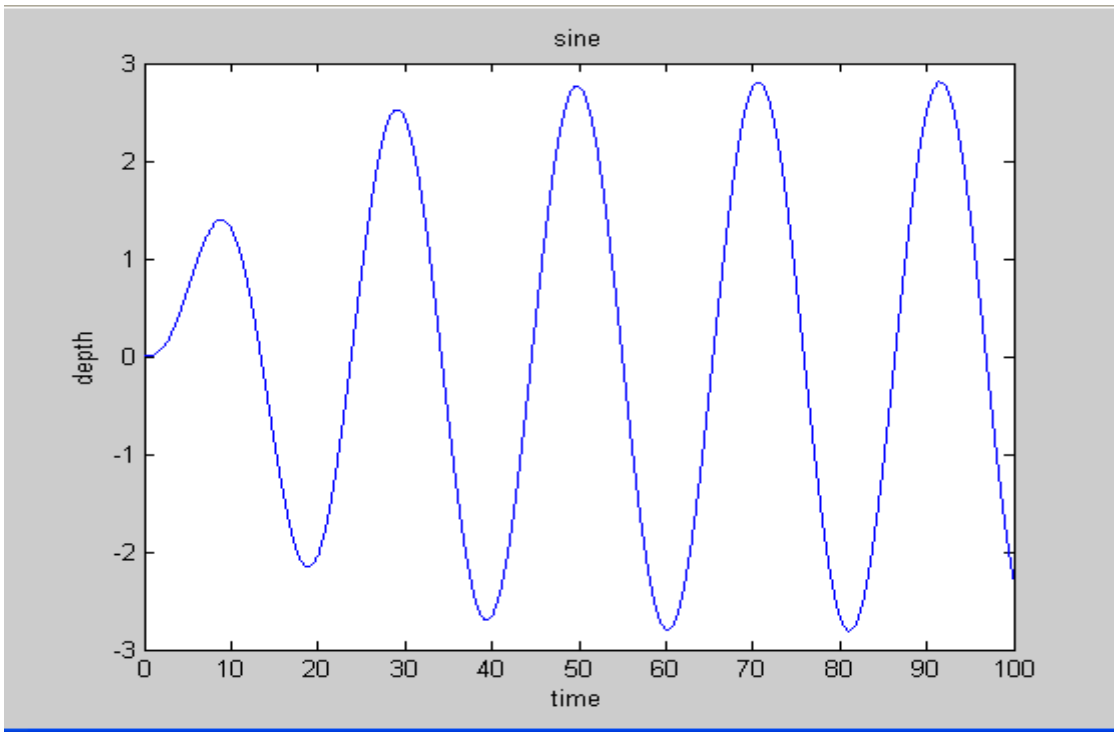
$$A\,e^{+at} + B\,e^{+bt} + N + Mt$$
$$+ 2X\,e^{+xt}\,\mathrm{Cos}[yt] - 2Y\,e^{+xt}\,\mathrm{Sin}[yt]$$

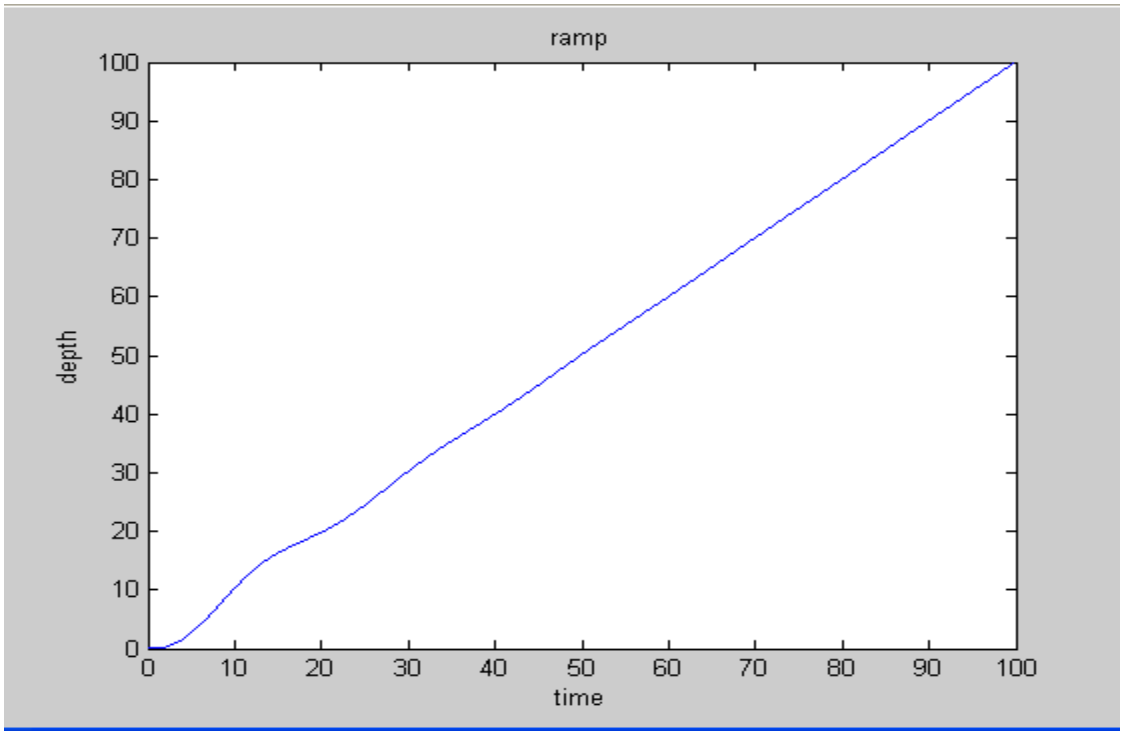
The MATLAB residue function gives when C_o = 1.0:

a = -0.8253 b = -0.2375
x = -0.0686 y = +0.3248
A = -0.2740 B = +0.3631
X = -0.0445 Y = +1.7643
N = 0.0 M = +1.0


impulse
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
depth
0
10
20
30
40
50
60
70
80
90
100
time


step
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
depth
0
10
20
30
40
50
60
70
80
90
100
time


sine
depth
time
3
2
1
0
-1
-2
-3
0
10
20
30
40
50
60
70
80
90
100


ramp
depth
time
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100

PIPE FLOW SETUP

TEMPERATURE CONTROL RESPONSES

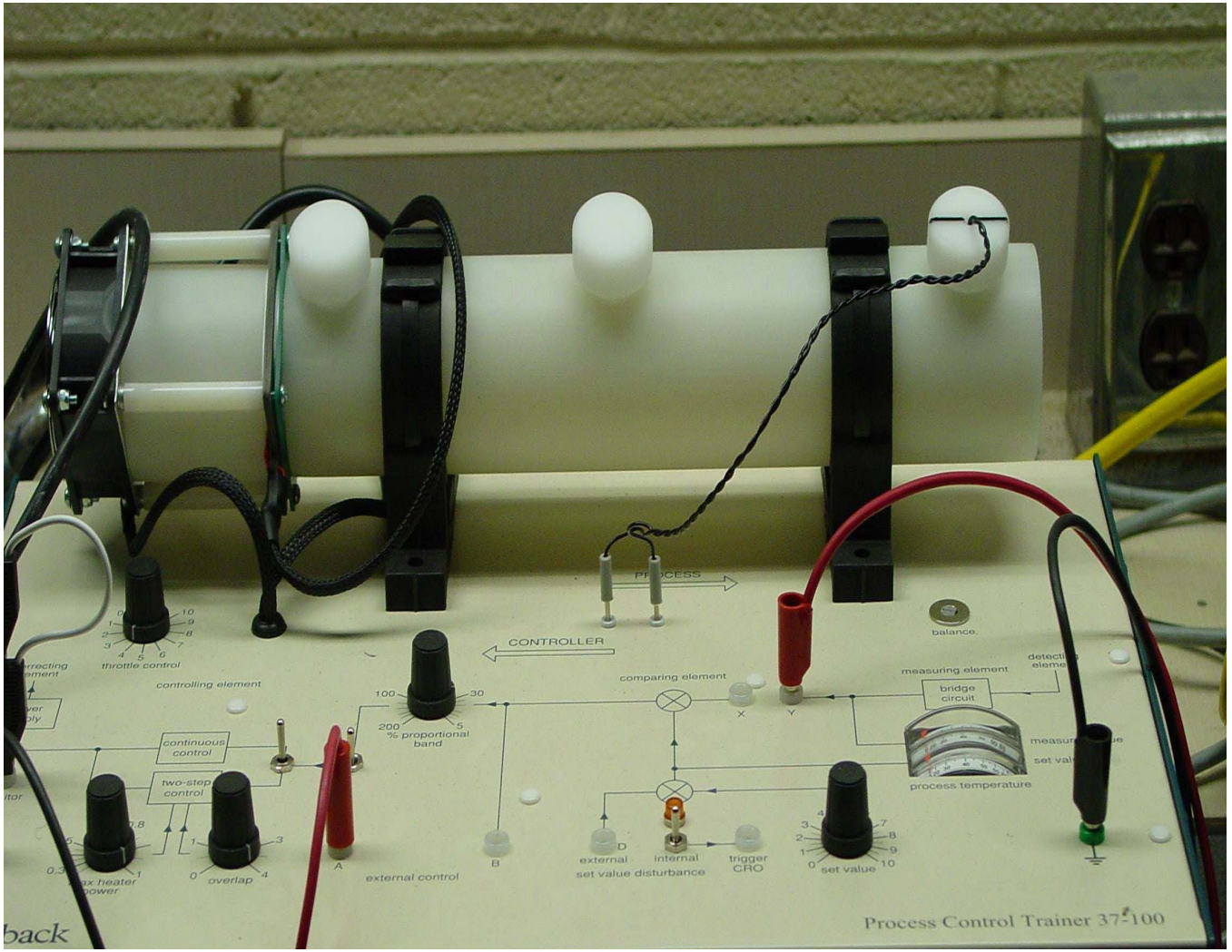
To illustrate application of the Laplace Transformation procedure we will consider the task of controlling the temperature of air flowing down a pipe. The setup is shown on the next page. Basically it consists of a fan which draws air from atmosphere and sends it down a pipe. A heater just downstream of the fan is used to heat the air. It receives a signal from a controller. The temperature of the air is measured by a thermistor. The governing equations are:

$$X \, dR/dt + Y \, R = H + D$$

$$H = Z \, Q \qquad Q = K_P \, E$$

$$E = C - \mathbf{R}$$

where R is the temperature of the air at the heater, **R** is the temperature of the air at the sensor, C is the command temperature, E is the temperature error, Q is the control signal, H is the heat generated by the heater, D is a disturbance heat and K_P is the controller gain. **R** is what R was T seconds back in time, where T is the time it takes for the air to travel down the pipe.


CONTROLLER
throttle control
controlling element
continuous control
two-step control
proportional band
external control
comparing element
measuring element
bridge circuit
balance
process temperature
external set value
internal disturbance
trigger CRO
set value
overlap
Process Control Trainer 37-100

Laplace Transformation of the governing equations gives

$$(X\ S + Y)\ R = H + D$$

$$H = Z\ Q \qquad Q = K_P\ E$$

$$E = C - \mathbf{R} \qquad \mathbf{R} = e^{-TS}\ R$$

We approximate the time lag as follows:

$$e^{-TS} = (1 - T/2\ S) / (1 + T/2\ S)$$

To give a numerical example we will let the parameters be:

$$X = 0.25 \qquad Y = 1.0 \qquad Z = 1.0$$

Substitution into the governing equations gives

$$(0.25\ S + 1.0)\ R = D +$$

$$K_P\ [\ C - (1 - T/2\ S) / (1 + T/2\ S)\ R\ ]$$

Algebraic manipulation gives

$$(0.25\ S + 1.0)\ (1 + T/2\ S)\ R =$$

$$(1 + T/2\ S)\ D + K_P\ (1 + T/2\ S)\ C$$

$$- K_P\ (1 - T/2\ S)\ R$$

More manipulation gives

$$[\ 0.25\ T/2\ S^2\ +\ (0.25\ +\ T/2\ -\ K_P\ T/2)\ S\ +\ (1.0\ +\ K_P)\ ]\ R$$

$$=\quad (1\ +\ T/2\ S)\ D\quad +\quad K_P\ (1\ +\ T/2\ S)\ C$$

A typical time lag is 0.5 seconds. In this case we get

$$[\ 0.0625\ S^2\ +\ (0.5\ -\ 0.25\ K_P)\ S\ +\ (1.0\ +\ K_P)\ ]\ R$$

$$=\quad (1.0\ +\ 0.25\ S)\ D\ +\ K_P\ (1.0\ +\ 0.25\ S)\ C$$

Setting C equal to zero gives

$$\frac{R}{D} = \frac{1.0\ +\ 0.25\ S}{0.0625\ S^2\ +\ (0.5\ -\ 0.25\ K_P)\ S\ +\ (1.0\ +\ K_P)}$$

Setting D equal to zero gives

$$\frac{R}{C} = \frac{K_P\ (1.0\ +\ 0.25\ S)}{0.0625\ S^2\ +\ (0.5\ -\ 0.25\ K_P)\ S\ +\ (1.0\ +\ K_P)}$$

The characteristic equation for the setup is:

$$0.0625\ S^2\ +\ (0.5\ -\ 0.25\ K_P)\ S\ +\ (1.0\ +\ K_P)\ =\ 0$$

This has the form of a mass on a spring and a dashpot:

$$m\ S^2\ +\ c\ S\ +\ k\ =\ 0$$

The mass m is 0.0625: the dashpot c is (0.5 - 0.25 K_P): the spring k is (1.0 + K_P). The equation shows that the damping is zero when K_P is equal to 2. This is a borderline gain. The oscillation frequency is:

$$\omega = \sqrt{[k/m]} = \sqrt{[(1.0 + \mathbf{K_P}) / 0.0625]} = 6.92$$

This gives a borderline period $\mathbf{T_P}$ equal to 0.91.

For the case where K_P is equal to half $\mathbf{K_P}$

$$\frac{R}{C} = \frac{1.0 + 0.25\ S}{0.0625\ S^2 + 0.25\ S + 2.0}$$

For a unit impulse command one gets

$$R = \frac{1.0 + 0.25\ S}{0.0625\ S^2 + 0.25\ S + 2.0}$$

Partial Fraction Expansion gives

$$R = \frac{A + Bj}{S - (a + bj)} + \frac{A - Bj}{S - (a - bj)}$$

Inverse Laplace Transformation gives

$$(A+Bj)\ e^{+(a+bj)t} + (A-Bj)\ e^{+(a-bj)t}$$

Manipulation gives

$$+ 2A\ e^{+at}\ Cos[bt] - 2B\ e^{+at}\ Sin[bt]$$

The MATLAB residue function gives

$$A = +2.0 \quad B = -0.76$$

$$a = -2.0 \quad b = +5.3$$

For a unit step command one gets

$$R = \frac{1.0 + 0.25\,S}{0.0625\,S^2 + 0.25\,S + 2.0} \cdot \frac{1}{S}$$

Partial Fraction Expansion gives

$$R = \frac{M + Nj}{S - (m + nj)} + \frac{M - Nj}{S - (m - nj)} + \frac{P}{S}$$

Inverse Laplace Transformation gives

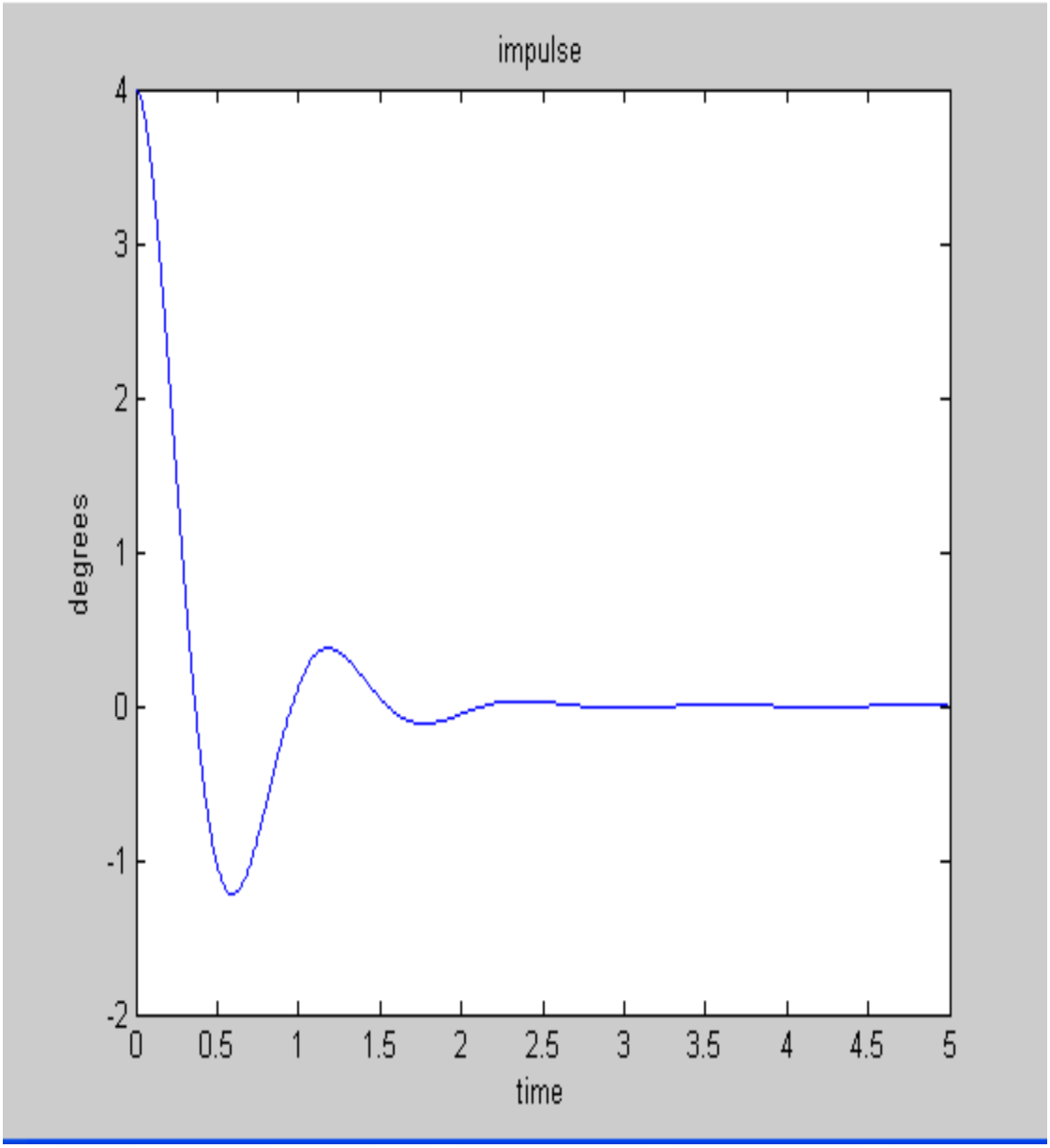
$$+ 2M\,e^{+mt}\,\mathrm{Cos}[nt] - 2N\,e^{+mt}\,\mathrm{Sin}[nt] + P$$

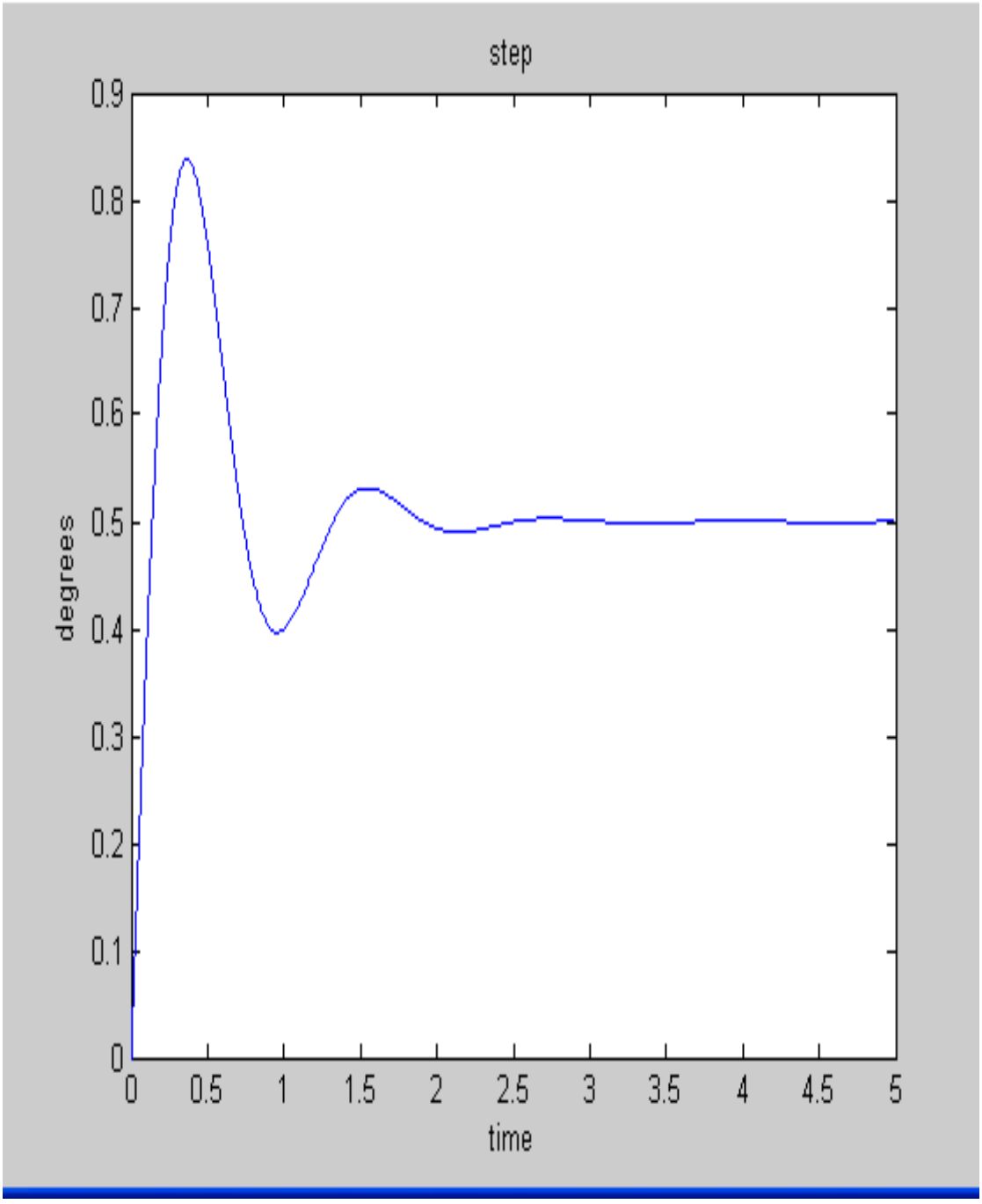
The MATLAB residue function gives

$$M = -0.25 \quad N = -0.28$$

$$m = -2.0 \quad n = +5.3$$

$$P = 0.5$$


impulse
degrees
time
4
3
2
1
0
-1
-2
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


step
degrees
time
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

CONTROL SYSTEM STABILITY

CHARACTERISTIC EQUATION: The overall transfer function for a feedback control system is: TF = G / [1+GH] . The G and H functions can be put into the form:

$$G(S) = A(S) / B(S) \qquad H(S) = X(S) / Y(S)$$

where A B X Y are polynomials. Substitution into the TF gives:

$$TF = A/B \; / \; [1 + A/B \; X/Y] \; = AY / [BY + AX] \; .$$

The transfer function can also be reduced to a ratio of two polynomials **N**(s) and **D**(s). In terms of these polynomials the characteristic equation is: **D**(S) = 0. Thus the characteristic equation in terms of A B X Y is: AX + BY = 0 .

The GH function is: GH = A/B X/Y = AX/BY = N/D. So the characteristic equation in terms of the GH function is:

$$N + D = 0 \; .$$

Note that the characteristic equations for the subsytems are all contained in D(S)=0. Often D(S) is in factored form: so simple inspection tells if the subsystems are stable or unstable. This is not the case for the overall system because, even though both N(S) and D(S) may be in factored form, adding them destroys this.

ROOT LOCUS PLOTS : As some parameter of a system is varied, each root of its characteristic equation moves around in the S plane and traces out a path known as a Root Locus. The Root Locus Method is systematic set of sketching rules based on the GH function for finding approximate location of these paths. Numerical schemes for finding roots of polynomials can now be used to find Root Locus paths exactly. So the Root Locus Method is obsolete. However the paths themselves are very important because they show system parameter values corresponding to the onset of instability. Root Locus Plots for some simple systems are given in Figure 1. To generate each plot, the parameter K was varied from 0 to ∞.

ROUTH-HURWITZ CRITERIA : These criteria infer stability information directly from the coefficients in the characteristic equation. The method is based on the theorem of residues. It is rarely derived from first principles in controls text books. It shows that, when some of the coefficients of the characteristic equation are positive and some are negative, the system is unstable. It also shows that a zero coefficient implies that the best a system can be is borderline stable. As a bare minimum, for stable operation of a system, all of the coefficients must be nonzero and all must have the same sign. Consider the cubic characteristic equation:

$$A\ S^3 + B\ S^2 + C\ S + D = 0$$

where A is positive. For this case, Routh-Hurwitz shows that, for stable operation, all coefficients must be positive, and they must also produce a positive value when substituted into the test function $X=BC-AD$. Consider the quartic characteristic equation:

$$A\ S^4 + B\ S^3 + C\ S^2 + D\ S + E = 0$$

where A is positive. For this case, Routh-Hurwitz shows that, for stable operation, all coefficients must be positive, and they must also produce positive values when substituted into the test functions $X=BC-AD$ and $Y=DX-B^2E$. Consider the quintic equation:

$$A\ S^5 + B\ S^4 + C\ S^3 + D\ S^2 + E\ S + F = 0$$

where A is positive. For this case, Routh-Hurwitz shows that, for stable operation, all coefficients must be positive, and they must also produce positive values when substituted into the test functions $X=BC-AD$ $Z=BE-AF$ and $Y=(DX-BZ)Z-X^2F$.

NYQUIST : A Nyquist Plot is a closed contour in the GH plane (or the 1+GH plane). It is obtained by mapping a closed contour in the S plane to the GH plane (or the 1+GH plane) using GH (or 1+GH) as a mapping function. The closed contour in the S plane surrounds the entire right half or unstable half of the S plane. A typical

mapping is shown in Figure 2. Stability is inferred from the plot in the GH plane (or the 1+GH plane). Development of the Nyquist Concept is based on the 1+GH function:

$$1 + GH = 1 + \frac{N}{D} = \frac{N + D}{D} \quad .$$

The overall characteristic function is N+D: the subsystems characteristic function is D. The roots of N+D=0 are called the zeros of the 1+GH function while the roots of D=0 are called the poles of the 1+GH function. Zeros are roots of the overall characteristic equation while poles are roots of the subsystem characteristic equations. At a zero $|1+GH|=0$ while at a pole $|1+GH|=\infty$. One can construct a 3D image of |1+GH| by taking the S plane as a horizontal plane and plotting |1+GH| vertically. At a zero the image would touch the S plane. At a pole its height above the S plane would be infinite. The plot could be used to determine the stability of the system and its subsystems.

One could factor 1+GH to get its zeros Z and poles P:

$$1 + GH = \frac{K\ (S-Z_1)\ (S-Z_2)\ :::::\ (S-Z_n)}{(S-P_1)\ (S-P_2)\ :::::\ (S-P_m)} \quad .$$

In the S plane, each (S-Z) or (S-P) factor is basically a vector with radius r and angle θ: $r\angle\theta$. A typical vector is shown in Figure 3. What happens to these vectors as the tip of the S vector moves once in a clockwise sense around the contour which surrounds the entire right half of the S plane? As shown in Figure 4,

vectors inside rotate clockwise 360° while vectors outside only nod up and down. What are the implications of this for the 1+GH function? Consider the Polar Form of 1+GH:

$$K\ [\Pi r_Z \angle \Sigma\theta_Z]\ /\ [\Pi r_P \angle \Sigma\theta_P]\ =\ R \angle \Theta$$

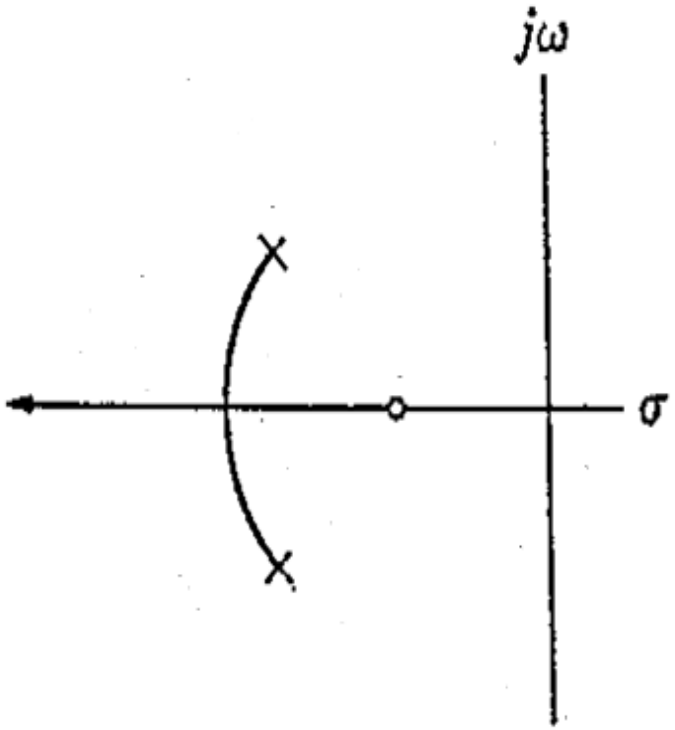
where Π indicates product and Σ indicates sum. Zeros inside cause clockwise rotations of 1+GH: poles inside cause counterclockwise rotations of 1+GH. Only zeros and poles inside cause such rotations: zeros and poles outside only cause 1+GH to nod up and down. If clockwise rotations are considered positive and counterclockwise rotations are considered negative, then the net clockwise rotations of 1+GH must be: $N = N_Z - N_P$ where N_Z is the number of zeros in the unstable half of the S plane while N_P is the number of poles there. For stable operation, N_Z must be zero. When N_Z is positive, the system is unstable. Inspection of D gives N_P. Inspection of the 1+GH plot gives N. Substitution into $N_Z = N + N_P$ gives N_Z. When a vector is drawn from the origin of the 1+GH plane to the 1+GH plot, N is the net number of times that this vector rotates clockwise when its tip moves along the plot.

The minus one point on the real axis in the GH plane corresponds to the origin in the 1+GH plane. This implies that a rotation of the GH vector drawn from the minus one point in the GH plane is equivalent to a rotation of the 1+GH vector drawn from the origin in the 1+GH plane. So one can get N from inspection of the GH plot or the 1+GH plot. This is illustrated in Figure 5.

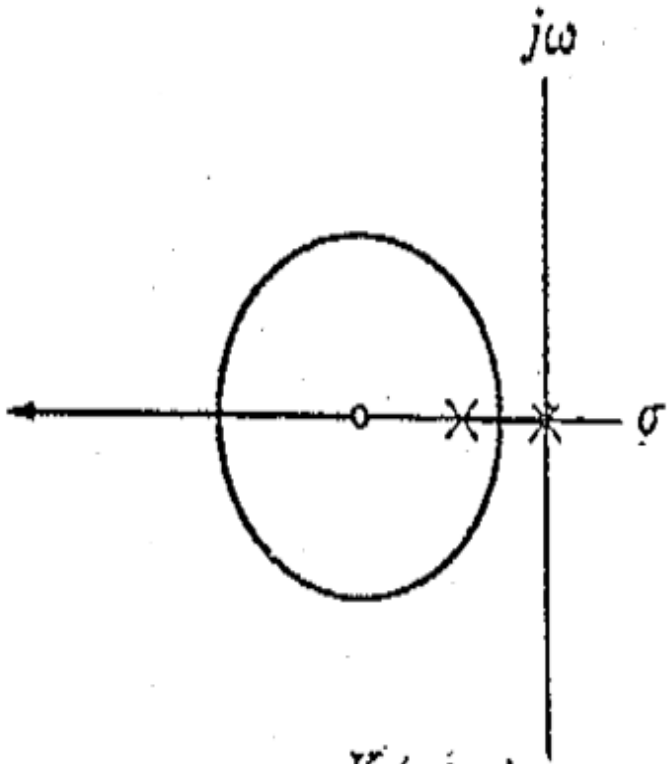
The basic Nyquist contour in the S plane consists of the imaginary axis and an infinite radius semicircle. This contour surrounds the entire right half or unstable half of the S plane. Sometimes there are poles of GH on the imaginary axis in the S plane. They are usually located at the origin. At a pole GH is infinite. To avoid this, the contour is indented locally with an infinitesimal radius counterclockwise semicircle centered on the pole.

To construct a GH plot, each section of the Nyquist contour is mapped separately. The infinite radius semicircle usually maps to the origin in the GH plane. An infinitesimal radius semicircle always maps to an infinite radius semicircle in the GH plane. Each pole on the imaginary axis produces one semicircle in the GH plane. The imaginary axis in the S plane can be mapped point by point to the GH plane. The negative imaginary axis portion is a mirror image of the positive imaginary axis portion. The location of these portions relative to the minus one point is usually critical. One can get a rough sketch of these portions by first fixing the small and large ω end points. One then examines the GH function to see if it is possible to make it purely real or purely imaginary. Purely real means there is a real axis crossover while purely imaginary means there is an imaginary axis crossover. With known end points and crossovers, one can quickly sketch the plot.

Root Locus Plots

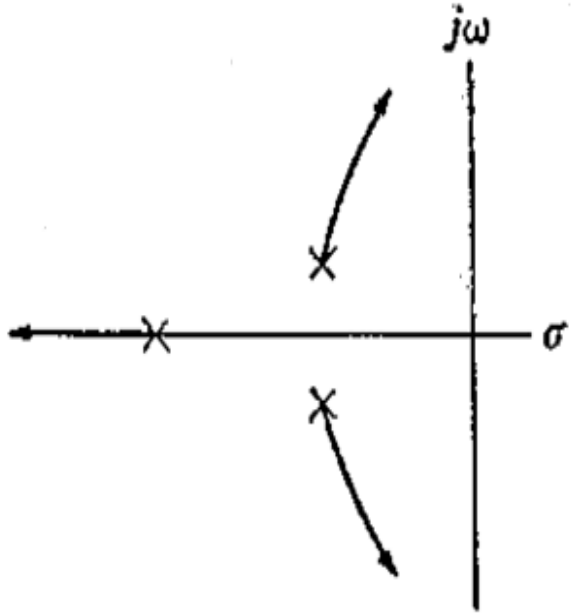


$$G(s)H(s) = \frac{K(s+a)}{(s+\alpha+j\beta)(s+\alpha-j\beta)}$$

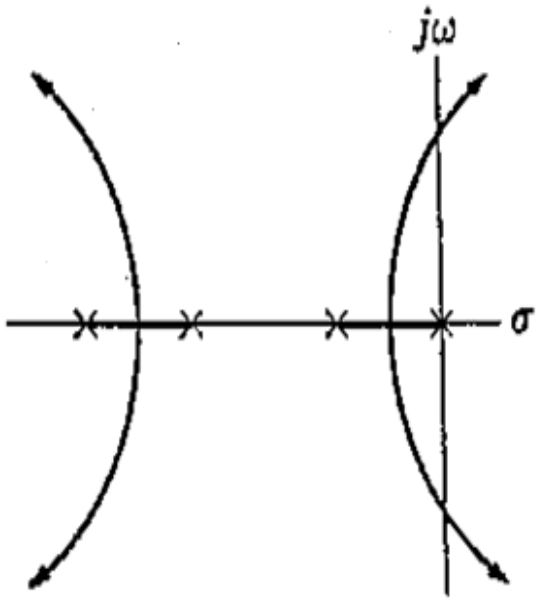


$$G(s)H(s) = \frac{K(s+a)}{s(s+b)}$$

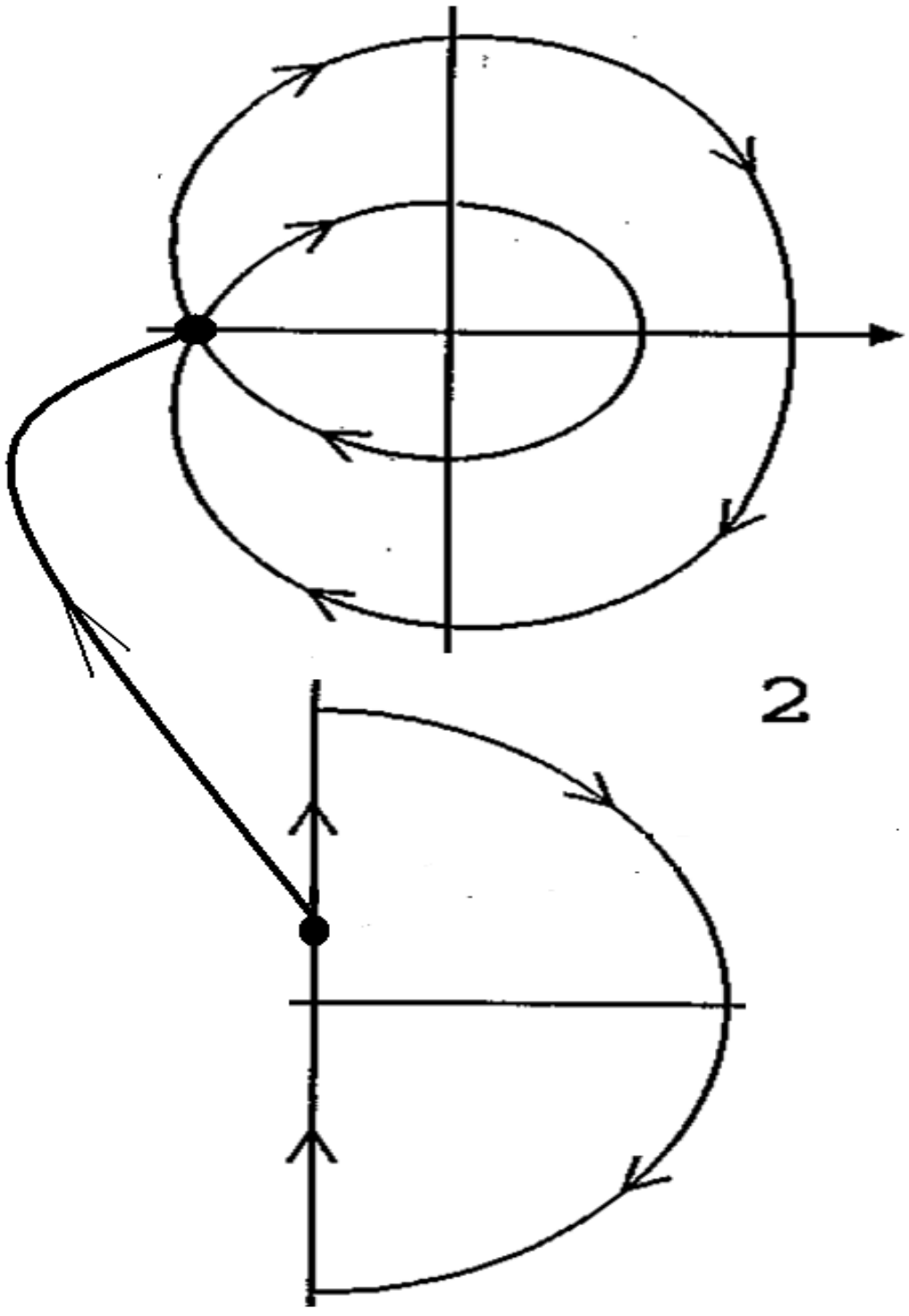
Root Locus Plots

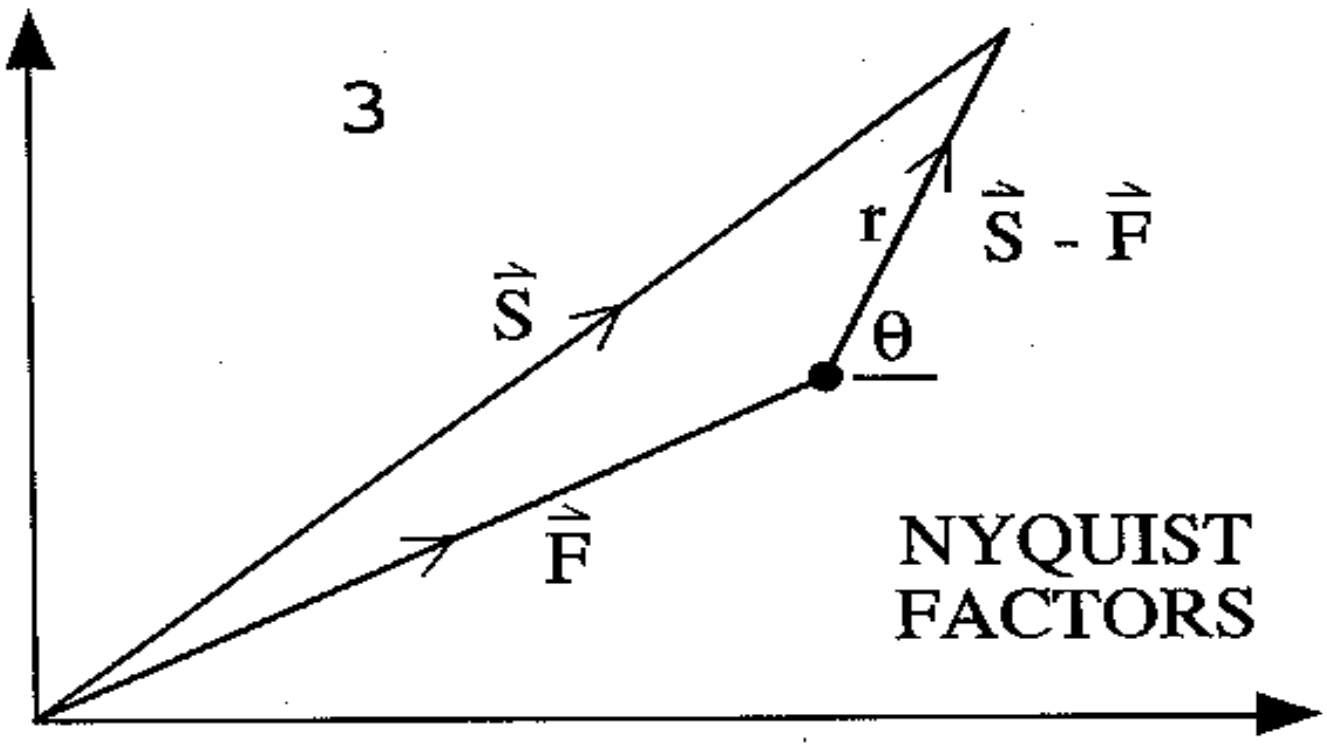


$$G(s)H(s) = \frac{K}{(s+a)(s+\alpha+j\beta)(s+\alpha-j\beta)}$$



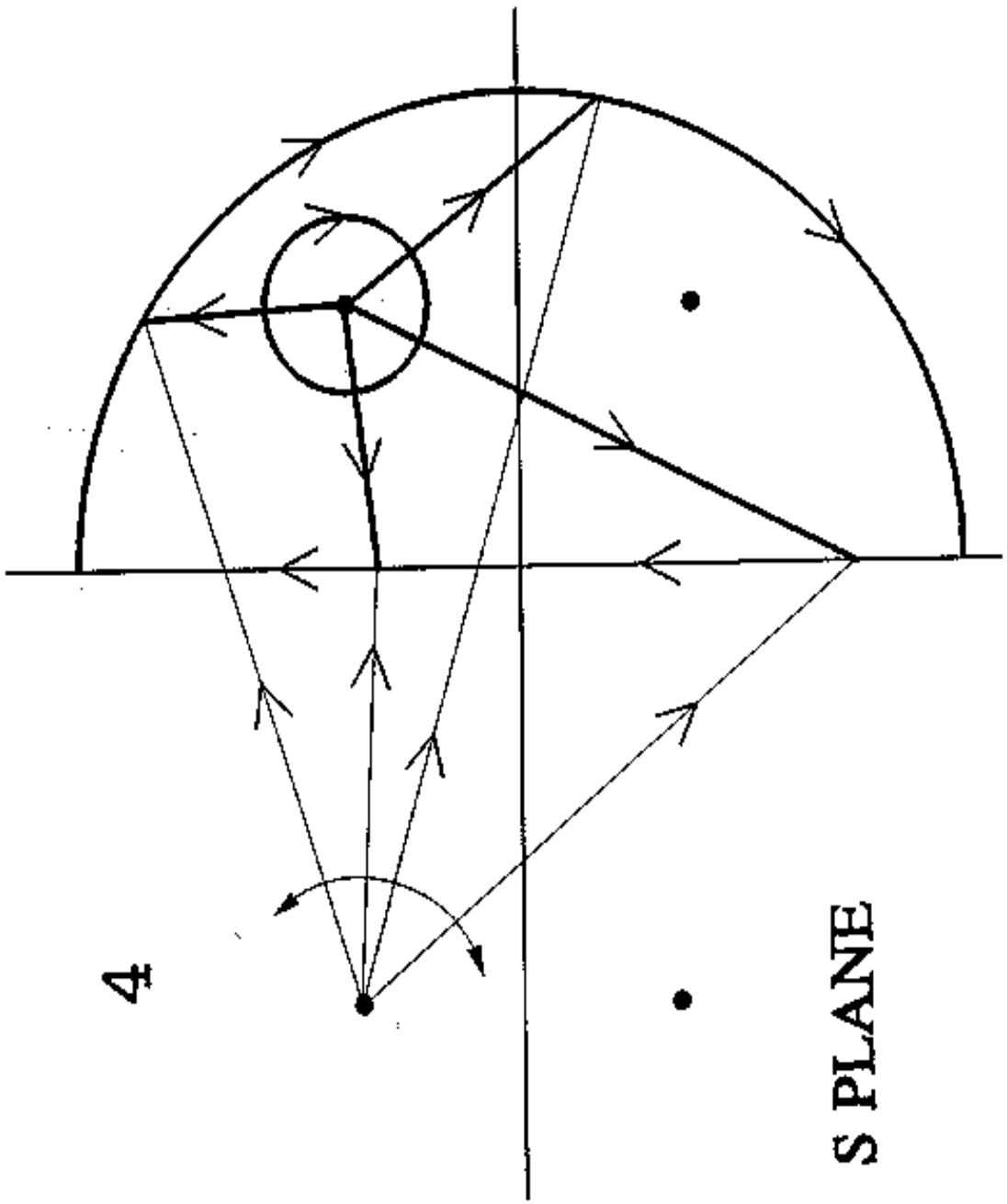
$$G(s)H(s) = \frac{K}{s(s+a)(s+b)(s+c)}$$

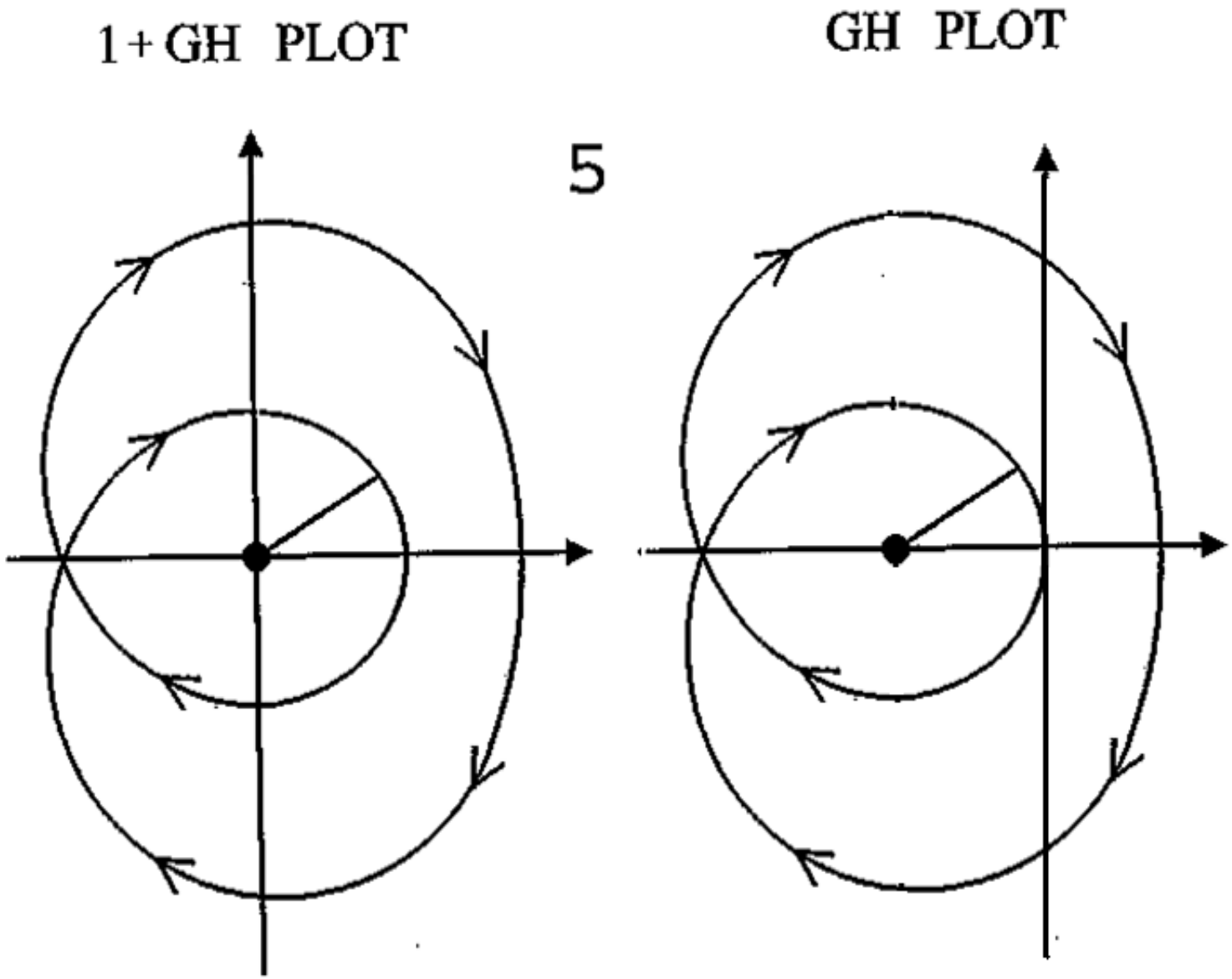

2



$S = A+Bj \qquad F = a+bj$

$S - F = (A-a) + (B-b)j = r \angle\theta$


4
S PLANE



5

AUTONOMOUS UNDERWATER VEHICLE

NYQUIST APPLICATION

To illustrate application of the Nyquist Procedure we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. The equations governing the motion of the auv are:

$$M\ d^2R/dt^2 + N\ dR/dt = B + D$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K_P\ E \qquad E = C - R$$

Laplace Transformation of the governing equations gives

$$(M\ S^2 + N\ S)\ R = B + D$$

$$(J\ S + I)\ B = Q$$

$$Q = K_P\ E \qquad E = C - R$$

The GH function for the auv is:

$$K_P\ /\ [\ (M\ S^2 + N\ S)\ (J\ S + I)\ ]$$

To give a numerical example we will let the parameters be:

$$M = 50.0 \quad N = 50.0 \quad J = 0.5 \quad I = 0.1$$

In this case the GH function reduces to

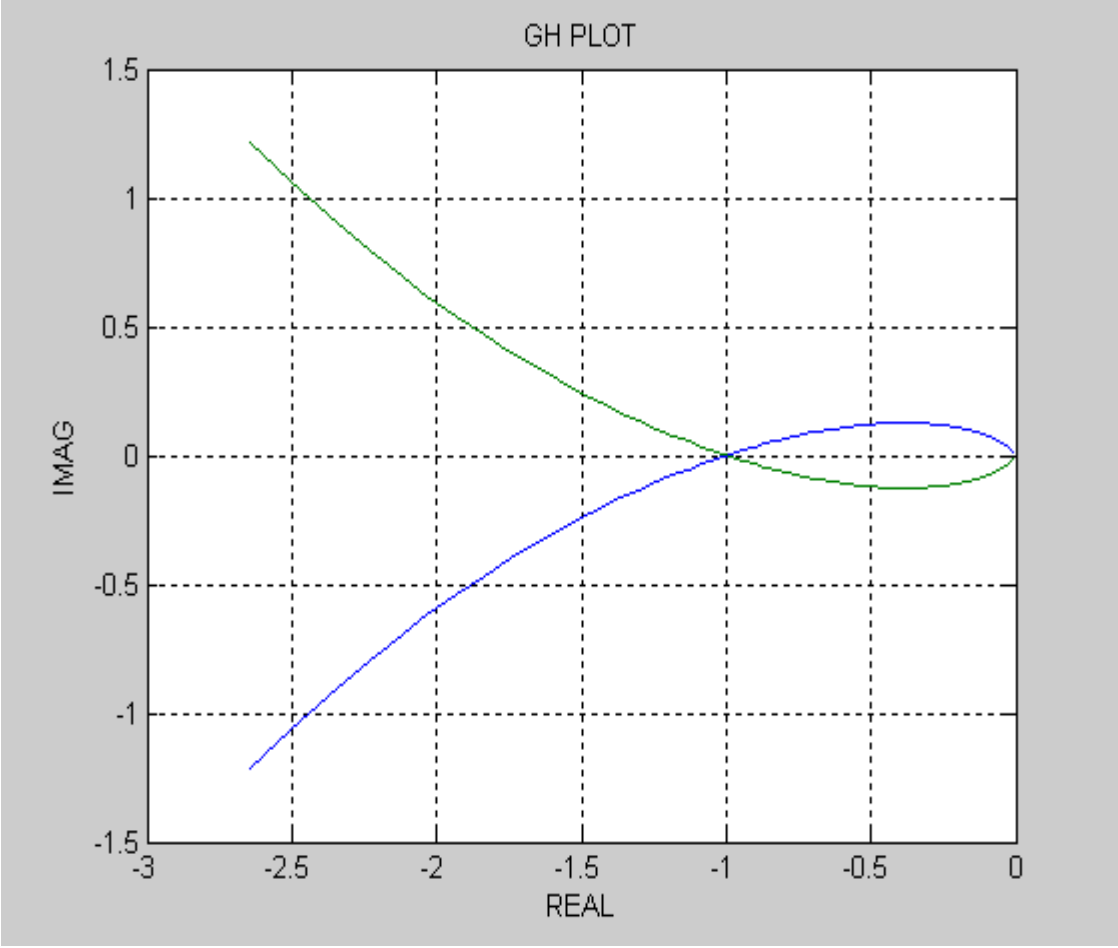
$$GH = K_P\ /\ [\ (50.0\ S^2 + 50.0\ S)\ (0.5\ S + 0.1)\ ]$$

$$= K_P\ /\ [25.0\ S^3 + 30.0\ S^2 + 5.0\ S]$$

Letting S=jω this can be written as:

$$GH = K_P / [-25.0\ \omega^3 j - 30.0\ \omega^2 + 5.0\ \omega j]$$

As ω tends to 0 GH tends to -∞j while as ω tends to ∞ it tends to +0j. There is a real axis crossover when $\omega^2=5/25=1/5$. With this ω^2 the term in square brackets reduces to -30/5 or -6. This implies that the borderline stable gain $\mathbf{K_P}$ which makes the crossover GH=-1 is 6. The matlab GH plot for the borderline case is given below.



PIPE FLOW SETUP

NYQUIST APPLICATION

To illustrate application of the Nyquist Procedure we will consider the task of controlling the temperature of air flowing down a pipe. The governing equations are:

$$X \frac{dR}{dt} + Y R = H + D$$

$$H = Z Q \qquad Q = K_P E$$

$$E = C - \mathbf{R}$$

Laplace Transformation of the governing equations gives

$$(X S + Y) R = H + D$$

$$H = Z Q \qquad Q = K_P E$$

$$E = C - \mathbf{R} \qquad \mathbf{R} = e^{-TS} R$$

The GH function for the setup is:

$$K_P Z e^{[-TS]} / (X S + Y)$$

To give a numerical example we will let the parameters be:

$$X = 0.25 \quad Y = 1.0 \quad Z = 1.0 \quad T = 0.5$$

In this case the GH function reduces to:

$$GH = K_P e^{[-0.5S]} / (0.25 S + 1.0)$$

Letting $S=j\omega$ this can be written as:

$$GH = K_P\,[\cos(0.5\omega) - j\,\sin(0.5\omega)] \,/\, (0.25\,j\omega + 1.0)$$

$$= K_P\,[P + Q\,j] \,/\, W$$

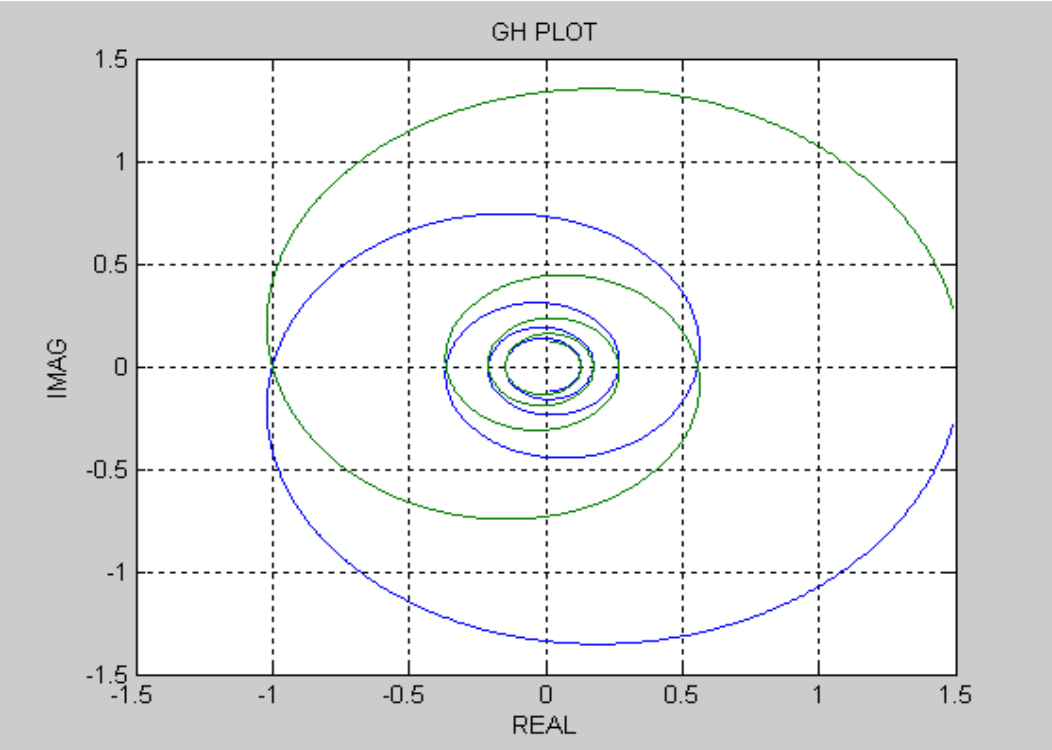
where

$$P = -\,0.25\,\omega\,\sin(0.5\omega) + \cos(0.5\omega)$$

$$Q = -\,0.25\,\omega\,\cos(0.5\omega) - \sin(0.5\omega)$$

$$W = (0.25\omega)^2 + 1.0$$

As ω tends to 0 GH tends to K_P while as ω tends to ∞ it tends to 0. Real axis crossovers occur when Q is equal to 0. Iteration shows that the first crossover occurs when ω=4.58. This gives P/W=-0.66. This implies that the borderline stable gain $\mathbf{K_P}$ which makes the crossover GH=-1 is 1.52. The matlab GH plot for the borderline case is given below.



NYQUIST PROCEDURE

The Nyquist procedure is based on the 1+GH function:

$$1 + GH = 1 + N/D = (N+D)/D \quad .$$

The overall characteristic function is N+D: the subsystems characteristic function is D. The roots of N+D=0 are called the zeros of the 1+GH function while the roots of D=0 are called the poles of the 1+GH function. Zeros Z are roots of the overall characteristic equation while poles P are roots of the subsystem characteristic equations. Manipulation of 1+GH gives:

$$1 + GH = \frac{K\ (S-Z_1)\ (S-Z_2)\ :::::\ (S-Z_n)}{(S-P_1)\ (S-P_2)\ :::::\ (S-P_m)}$$

$$= K\ [\Pi r_Z \angle \Sigma\theta_Z]\ /\ [\Pi r_P \angle \Sigma\theta_P]$$

$$= R \angle \Theta \quad .$$

So 1+GH is basically a vector with radius R and angle Θ. One can plot this in a 1+GH plane. Let us surround the entire unstable half of the S plane with a clockwise contour. When the tip of the S vector moves clockwise around this contour, zeros inside it cause clockwise rotations of 1+GH while poles inside it cause counterclockwise rotations. Only zeros and poles inside cause such rotations: zeros and poles outside only cause 1+GH to nod up and down. One can imagine the 1+GH function to be a clock like

mechanism: unstable zeros cause its hand to rotate clockwise while unstable poles cause its hand to rotate counterclockwise: stable zeros and poles only cause its hand to swing back and forth.

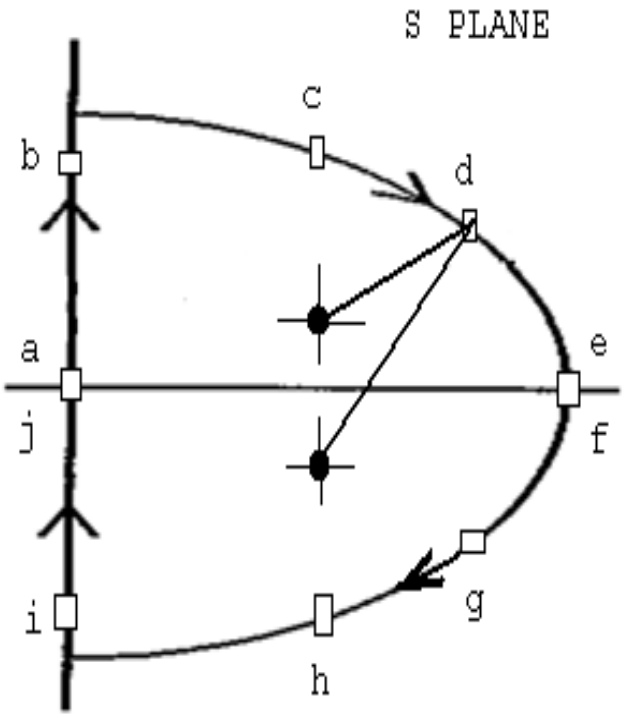
As an illustration, consider the case, shown two pages over, where there two unstable zeros in the right half of the S plane and all other zeros and poles are far into the left half of the S plane. Now surround the unstable zeros by a clockwise contour as shown on the top of the page. When we map points on this contour to the 1+GH plane, we get the contour shown on the bottom of the page. Note that no attempt has been made to get exact lengths in the 1+GH sketch: the focus is on getting the angles approximately correct. When we draw a vector with radius R and angle Θ to the contour in the 1+GH plane and count the number of times it rotates clockwise as we move around the contour in the S plane, we get two clockwise rotations. These rotations are caused by the unstable zeros. Nyquist allows us to determine the number of unstable zeros without having to find their exact locations.

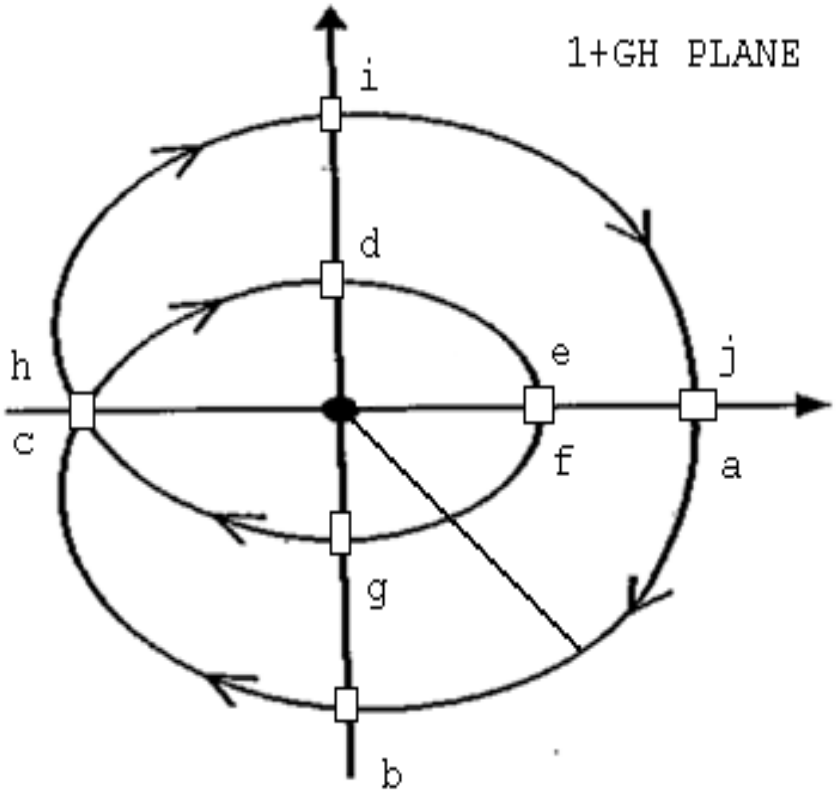
In a GH plane, the vector with radius R and angle Θ is drawn from the minus one point on the negative real axis. If clockwise rotations are considered positive and counterclockwise rotations are considered negative, then the net clockwise rotations of GH must be: $N=N_Z-N_P$ where N_Z is the number of unstable zeros and N_P is the number of unstable poles. For stable operation, N_Z must be zero. When N_Z is positive, the system is unstable. Inspection of D gives N_P. Inspection of the GH plot gives N. Then substitution into $N_Z=N+N_P$ gives N_Z. For a stable system, the nearness of a GH Plot

to the minus one point is a measure of the degree of stability of the system. There are two stability margins: gain margin and phase margin. These can be used for design.

Consider the case where only proportional control is being used and the GH plot passes through the minus one point in the GH plane. If GH=-1 then 1+GH=0. This implies that at this point S=Z: it is a root of the overall characteristic equation. But along the GH plot S=±jω. So Z=±jω. So there is a complex conjugate pair of roots of the overall characteristic equation on the imaginary axis in the S plane. This means the system is borderline stable and the gain is **K**. The frequency of the borderline oscillation is **ω**. This means the borderline period is **T**=2Π/**ω**. These borderline gain and period allow us to calculate Ziegler Nichols gains.

A GH plot is basically a polar open loop frequency response plot. When GH=-1, a command sine wave produces a response which has the same magnitude as the command but is 180° out of phase. If the command was suddenly removed and the loop was suddenly closed, the negative of the response would take the place of the command and keep the system oscillating. If the gain was bigger than **K**, the command would produce a response bigger than itself. When this takes over, it would produce growing or unstable oscillations. If the gain was smaller than **K**, the command would produce a response smaller than itself. When this takes over, it would produce decaying or stable oscillations.


S PLANE
c
b
d
a
e
j
f
g
i
h


1+GH PLANE
i
d
h
e
j
c
f
a
g
b

ROOT LOCUS CONCEPT

For a feedback control system 1+GH=0 when S is a root of the overall characteristic equation for the system. This implies that when S is a root GH=-1. So at a root the magnitude of GH is 1 and its angle is plus or minus 180^o. The Root Locus Concept is based on these magnitude and phase requirements on GH. Consider a system with the following GH function:

$$GH = K \times [(S-v) (S-w)] / [(S-a) (S-b) (S-c)]$$

The Root Locus Concept can be used to find the paths traced out by the overall roots as some parameter is varied. It can also be used to find the K corresponding to borderline stable operation. The procedure has two stages. In stage I, an S point on the imaginary axis is picked and the angles to it from the zeros and poles of GH are measured. This is shown in the sketch on the next page. Those angles must total plus or minus 180^o. Zero angles are added: pole angles are subtracted. The S point is moved by trial and error until the angle requirement is met. In stage II, the lengths of the vectors from the zeros and poles to the final S point are measured. The magnitude requirement gives:

$$\mathbf{K} [X \; V \; W] / [A \; B \; C] = 1$$

This gives

$$\mathbf{K} = [A \; B \; C] / [X \; V \; W]$$

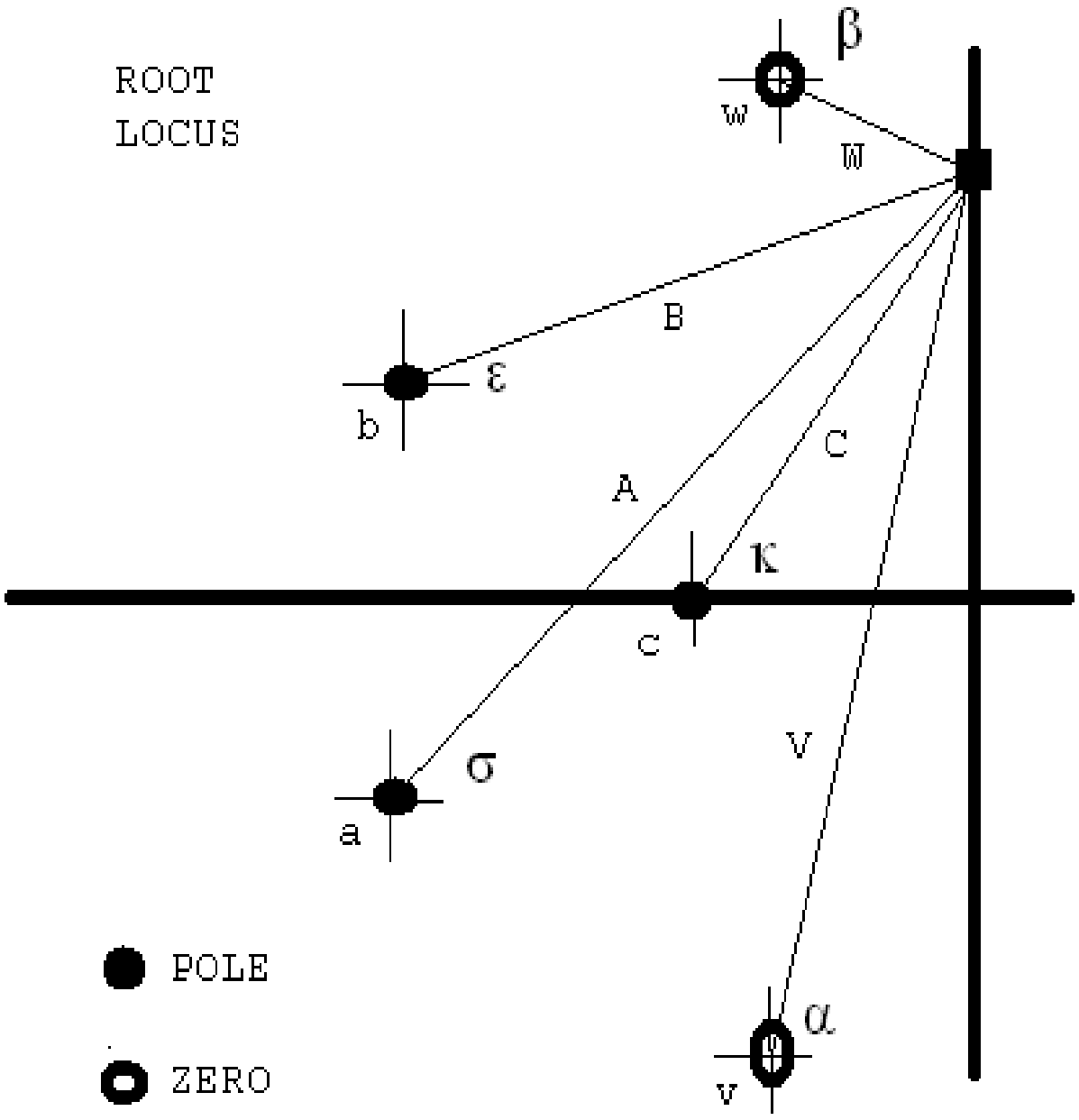

ROOT
LOCUS
β
w
W
B
ε
b
C
A
κ
c
σ
a
V
α
v
POLE
ZERO

ILLUSTRATION : AUV DEPTH CONTROL

The governing equations are:

$$M\ d^2R/dt^2 + N\ dR/dt = B + D$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K_p\ E \qquad E = C - R$$

The GH Function is:

$$GH = K_p / [\ (M\ S^2 + N\ S)\ (J\ S + I)\ ]$$

$$= [K_p/MJ] / [\ S\ (S - [-N/M])\ (S - [-I/J])\ ]$$

Letting M=50.0 N=50.0 J=0.5 I=0.1 gives:

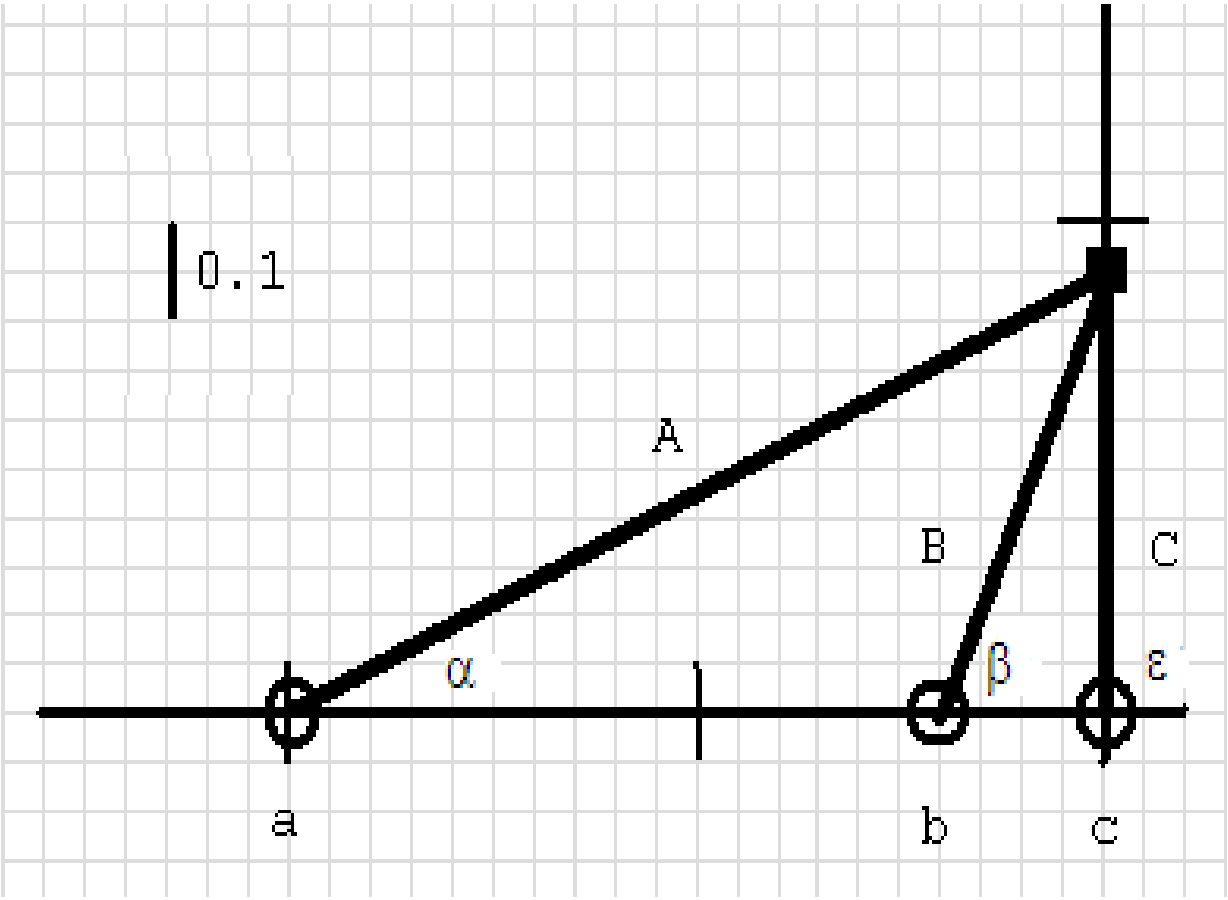
$$GH = [K_p * 1/25] / [\ S\ (S - [-1])\ (S - [-0.2])\ ]$$

$$= K_p * X / [\ (S-a)\ (S-b)\ S-c)\ ]$$

The S plane sketch for this case, after the trial and error adjustment to get the angle requirement satisfied, is shown on the next page. The magnitude requirement gives:

$$K_p * X / [\ A * B * C] = 1$$

$$K_p = [\ A * B * C] / X = 6$$



α=25 β=65 ε=90

A=1.09 B=0.49 C=0.45

AUTONOMOUS UNDERWATER VEHICLE

ROOT LOCUS APPLICATION

As an application of the root locus procedure, we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. According to Newton's Second Law of Motion, the equation governing the up and down motion of the auv is:

$$M\ d^2R/dt^2 = B + D - W$$

$$W = X\ dR/dt\ |dR/dt| + Y\ dR/dt = N\ dR/dt$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K_P\ E + K_I \int E d\tau + K_D\ dE/dt$$

$$E = C - R$$

where R is the depth of the auv, M is its overall mass, B is the control force from the propulsion system, D is a disturbance load caused for example by sudden weight changes, W is a drag load consisting of wake drag and wall drag, X and Y account for the size and shape of the auv, Q is the control signal, J and I are drive constants, E is the depth error, K_P K_I K_D are the controller gains and C is the command depth.

Laplace Transformation of the governing equations gives

$$(M\ S^2 + N\ S)\ R = B + D$$

$$(J\ S + I)\ B = Q$$

$$Q = (K_P + K_I/S + K_D\ S)\ E$$

$$E = C - R$$

Algebraic manipulation gives

$$\frac{R}{C} = \frac{K_D\ S^2 + K_P\ S + K_I}{MJ\ S^4 + (NJ+MI)\ S^3 + (NI+K_D)\ S^2 + K_P\ S + K_I}$$

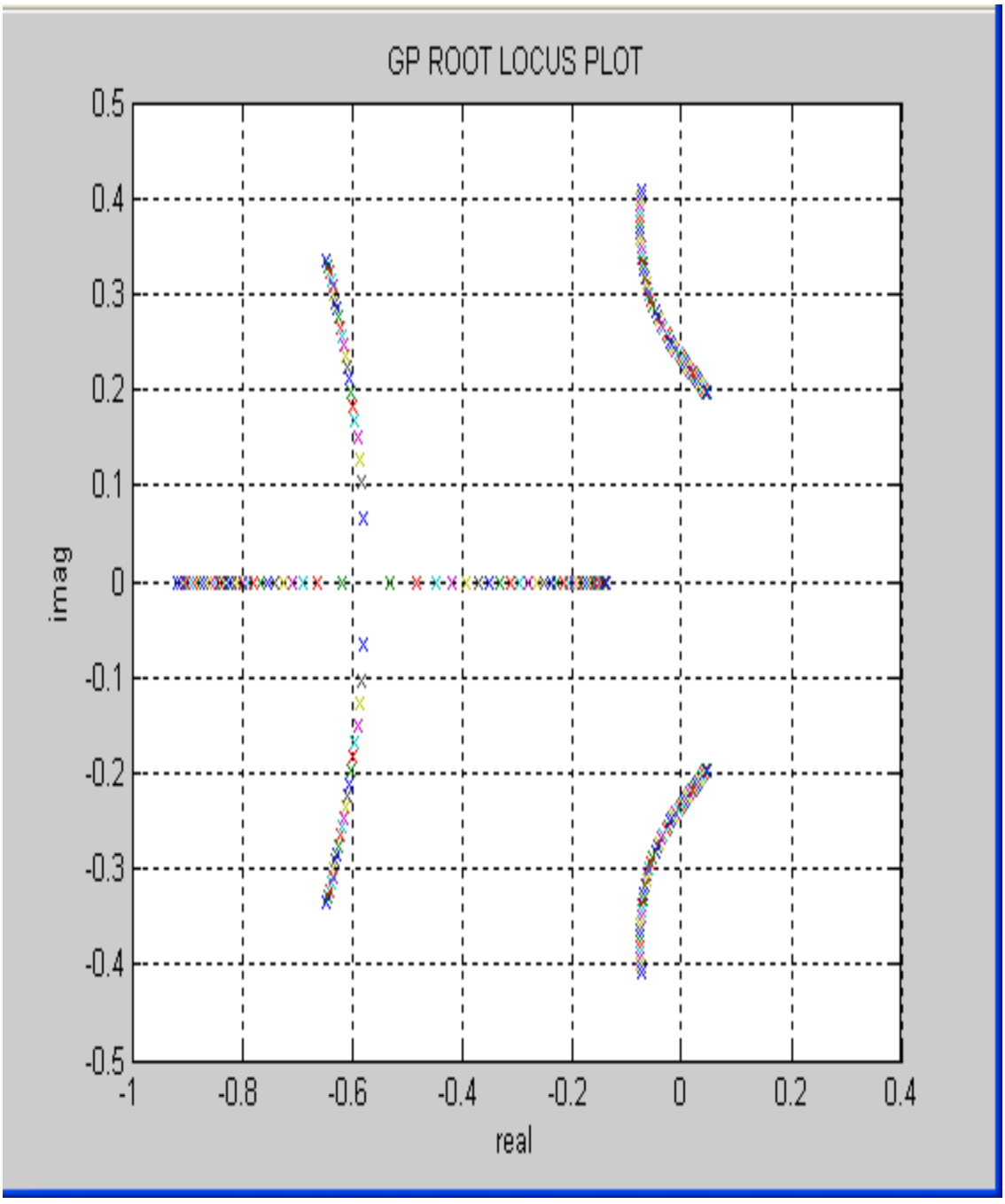
$$\frac{R}{D} = \frac{J\ S^2 + I\ S}{MJ\ S^4 + (NJ+MI)\ S^3 + (NI+K_D)\ S^2 + K_P\ S + K_I}$$

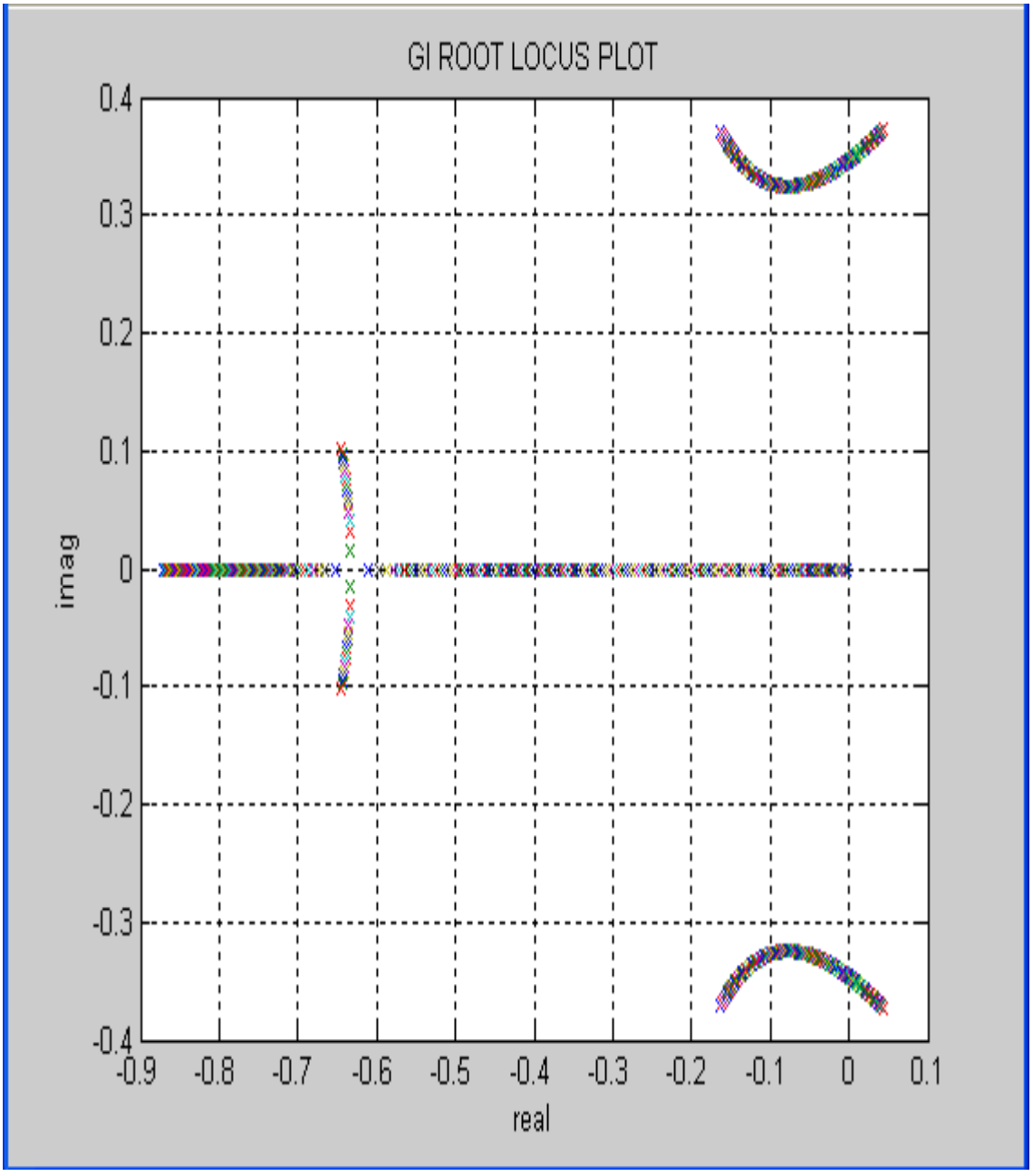
The characteristic equation is

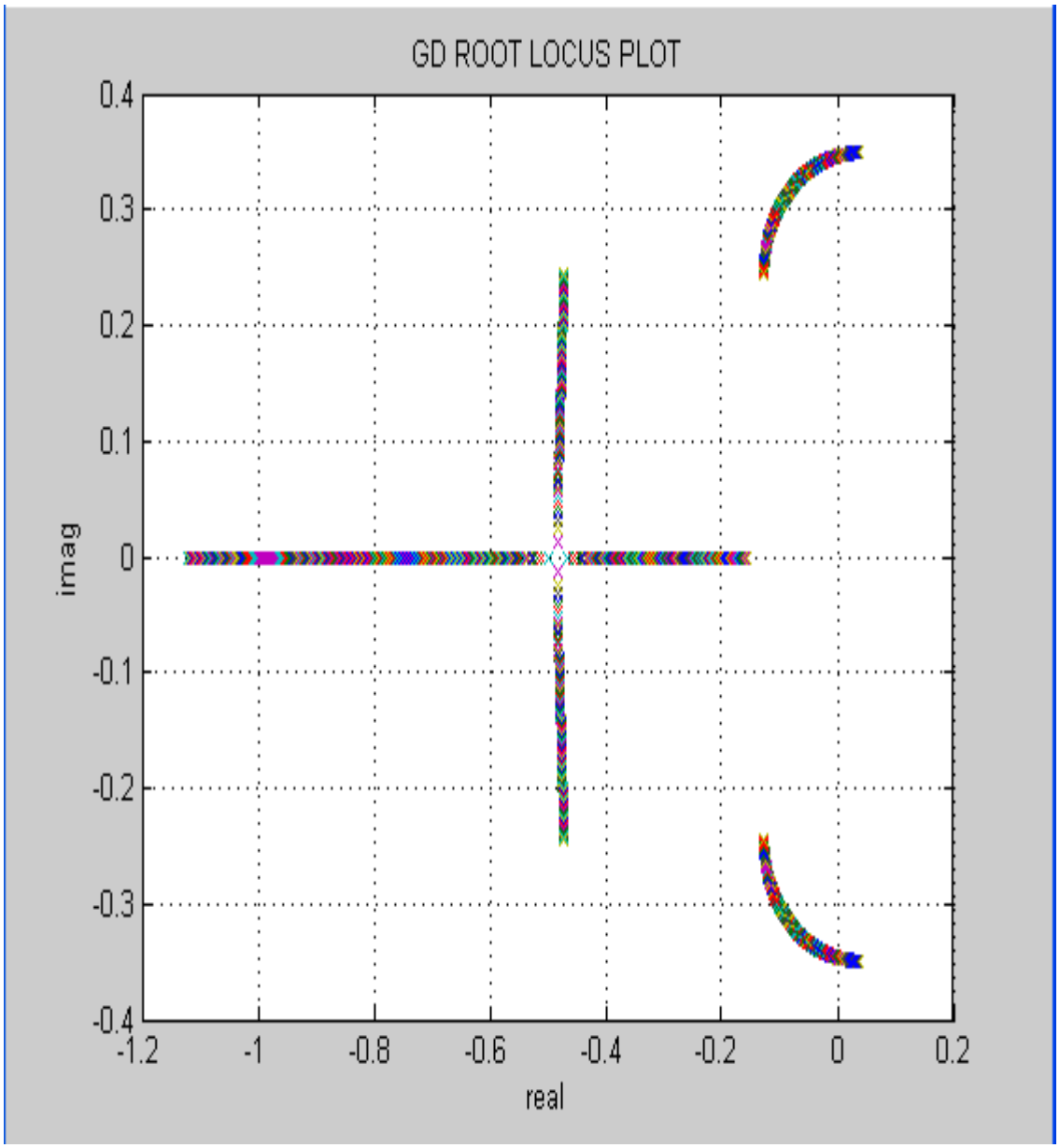
$$MJ\ S^4 + (NJ + MI)\ S^3 + (NI + K_D)\ S^2 + K_P\ S + K_I = 0$$

The code on the next page can be used to get the root locus plot for any parameter of the auv. It is presently set up to study the influence of K_D. A series of root locus plots generated by the code follows the code.

```
% SUBSEA ROBOT
% ROOT LOCUS PLOTS
clear all
m=50.0;n=50.0;
j=0.5;i=0.1;
a=j*m; b=n*j+m*i;
gp=3.6; gi=0.54;
gd=[0.01:0.01:10.0];
for k=1:length(gd)
c=n*i+gd(k);
q=[a b c gp gi];
p(:,k)=roots(q);
end
plot(real(p),imag(p),'x')
title('GD ROOT LOCUS PLOT')
xlabel('real')
ylabel('imag')
grid
```


GP ROOT LOCUS PLOT
imag
real
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4


GI ROOT LOCUS PLOT
imag
real
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.9
-0.8
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1


GD ROOT LOCUS PLOT
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
imag
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
real

PIPE FLOW SETUP

ROOT LOCUS APPLICATION

As an application of the root locus procedure, we will consider the task of controlling the temperature of air flowing down a pipe. The governing equations are:

$$X\ dR/dt + Y\ R = H + D$$

$$A\ dH/dt + B\ H = Z\ Q$$

$$Q = K_P\ E \qquad E = C - \mathbf{R}$$

where R is the temperature of the air at the heater, **R** is the temperature of the air at the sensor, C is the command temperature, E is the temperature error, Q is the control signal, H is the heat generated by the heater, D is a disturbance heat and K_P is the controller gain. **R** is what R was T seconds back in time, where T is the time it takes for the air to travel down the pipe.

Laplace Transformation of the governing equations gives

$$(X\ S + Y)\ R = H + D$$

$$(A\ S + B)\ H = Z\ Q$$

$$Q = K_P\ E \qquad E = C - \mathbf{R}$$

$$\mathbf{R} = e^{-TS}\ R$$

We approximate the time lag as follows:

$$e^{-TS} = (1 - T/2\ S) / (1 + T/2\ S)$$

Algebraic manipulation gives

$$\frac{R}{C} = \frac{K_P\ Z\ (1 + T/2\ S)}{(A\ S + B)\ (X\ S + Y)\ (1 + T/2\ S) + K_P\ Z\ (1 - T/2\ S)}$$

$$\frac{R}{D} = \frac{(A\ S + B)\ (1 + T/2\ S)}{(A\ S + B)\ (X\ S + Y)\ (1 + T/2\ S) + K_P\ Z\ (1 - T/2\ S)}$$

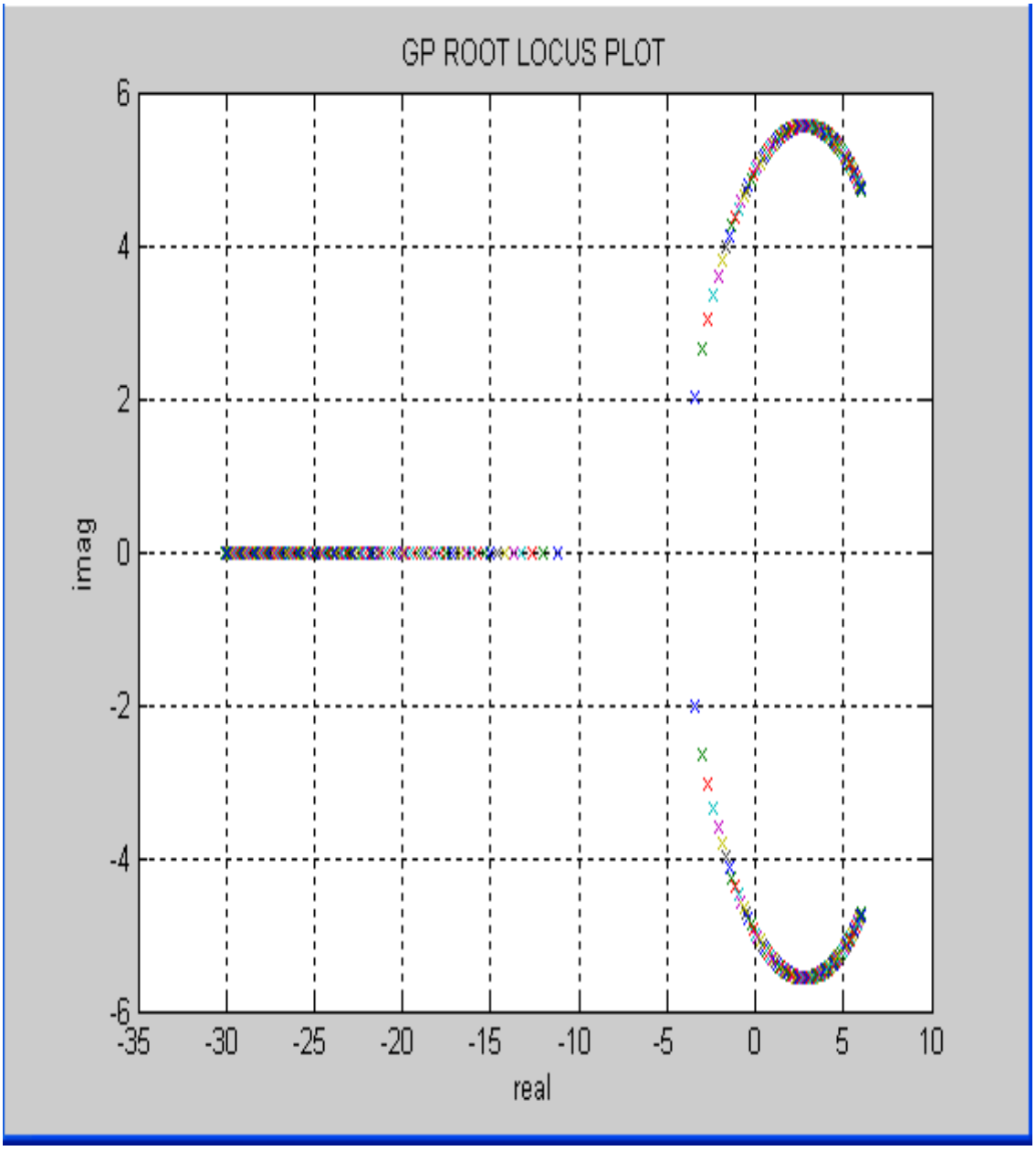
The characteristic equation is

$$T/2\ AX\ S^3 + [T/2\ (BX + AY) + AX]\ S^2$$

$$+ [T/2\ BY + BX + AY - K_P\ Z\ T/2]\ S + K_P\ Z + BY = 0$$

The code on the next page can be used to get the root locus plot for any parameter of the pipe flow setup. It is presently set up to study the influence of K_P. A root locus plot generated by the code follows the code.

```
% PIPE FLOW SETUP
% ROOT LOCUS PLOTS
clear all
A=0.1; B=1.0;
X=0.25; Y=1.0;
Z=1.0; T=0.5;
U=T/2*(A*X);
V=T/2*(B*X+A*Y)+A*X;
W=T/2*B*Y+B*X+A*Y;
G=Z*T/2; H=B*Y;
GP=[0.1:0.1:10.0];
for k=1:length(GP)
q=[U V W-GP(k)*G GP(k)*Z+H];
p(:,k)=roots(q);
end
plot(real(p),imag(p),'x')
title('GP ROOT LOCUS PLOT')
xlabel('real')
ylabel('imag')
grid
```


GP ROOT LOCUS PLOT
imag
real
6
4
2
0
-2
-4
-6
-35
-30
-25
-20
-15
-10
-5
0
5
10

SYSTEM PERFORMANCE

Often the dynamics of a control system is dominated by a pair of roots just to the left of the imaginary axis in the S plane. All other roots are much further to the left and have transients which die away very quickly. In such cases, the system behaves basically like a mass on a spring and a dashpot. The sketch on the next page shows such a system. Its governing equation is:

$$m\ d^2R/dt^2 = c\ (dC/dt - dR/dt) + k\ (C - R)$$

Laplace Transformation gives the transfer function

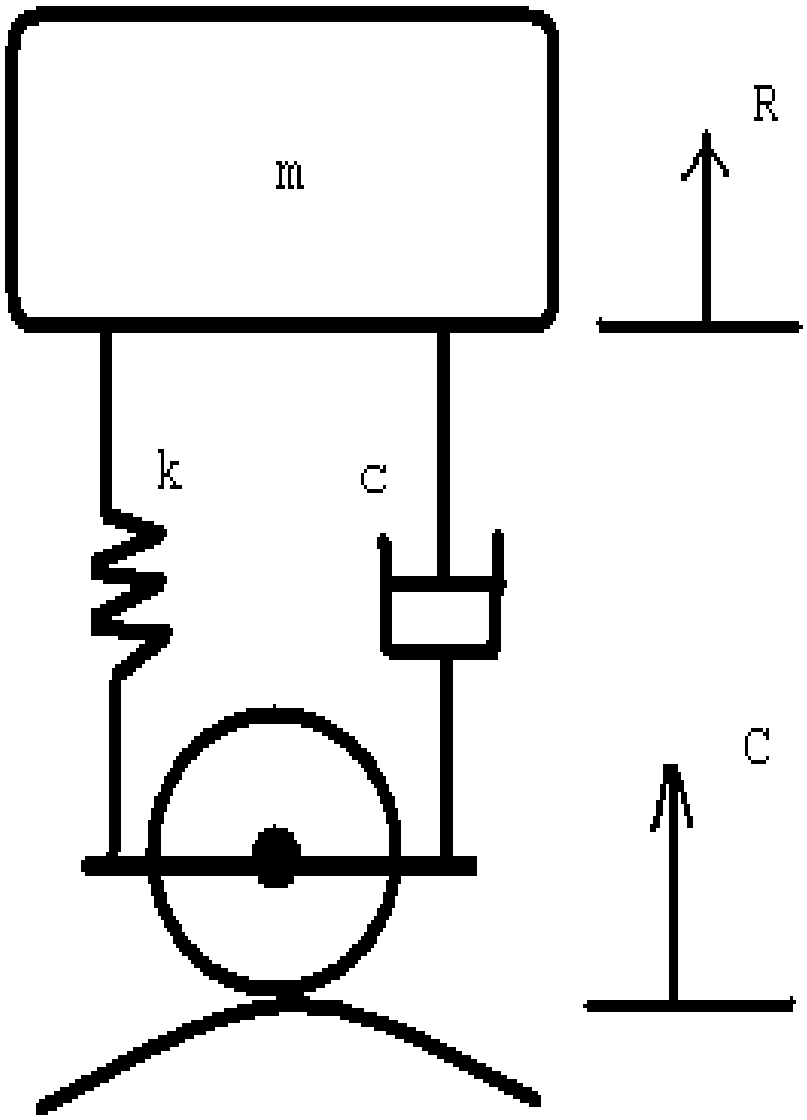
$$\frac{R}{C} = TF = \frac{c\ S + k}{m\ S^2 + c\ S + k}$$

Manipulation gives

$$\frac{R}{C} = TF = \frac{c/m\ S + k/m}{S^2 + c/m\ S + k/m}$$

Vibration theory allows one to rewrite this as

$$\frac{R}{C} = TF = \frac{2\zeta\omega_n\ S + \omega_n\ \omega_n}{S^2 + 2\zeta\omega_n\ S + \omega_n\ \omega_n}$$


m
R
k
c
c

where ω_n is the system undamped natural frequency and ζ is its damping factor. These are by definition

$$\omega_n = \sqrt{[k/m]} \qquad \zeta = c/c_c \qquad C_c = 2\sqrt{[k*m]}$$

The roots of the system characteristic equation are

$$x + yj \qquad x - yj$$

$$x = -\zeta\omega_n \qquad y = \omega_d$$

where ω_d is the damped natural frequency given by

$$\omega_d = \omega_n \sqrt{[1 - \zeta^2]}$$

For an impulse input, the response has the form

$$X\, e^{+xt}\, \mathrm{Cos}[yt] + Y\, e^{+xt}\, \mathrm{Sin}[yt]$$

while for a step input, it has the form

$$\mathbf{X}\, e^{+xt}\, \mathrm{Cos}[yt] + \mathbf{Y}\, e^{+xt}\, \mathrm{Sin}[yt] + \mathbf{Z}$$

In a response, overshoots must be kept to a tolerable level. This is dependent mainly on the damping factor ζ. The angle that a root vector makes with x axis is:

$$\Theta = \tan^{-1}[y/x] = \tan^{-1}[\omega_d/[\zeta\omega_n]]$$

$$= \tan^{-1}[\omega_n\sqrt{[1-\zeta^2]}/[\zeta\omega_n]] = \tan^{-1}[\sqrt{[1-\zeta^2]}/\zeta]$$

This shows that radial lines drawn out from the origin are lines of constant ζ. The vertical axis corresponds to ζ equal to zero or no damping, while the horizontal axis corresponds to ζ equal to unity or critical damping. Experience shows that ζ should be at least 0.5, which corresponds to an angle Θ of 60°. So roots should lie inside a wedge shaped region in the S plane.

Speed of response is another important requirement of a system. This is dependent mainly on the natural frequency ω_n. The radius r of a circle drawn from the origin in the S plane is:

$$r^2 = x^2 + y^2$$

$$= [\zeta\omega_n]^2 + [\omega_d]^2$$

$$= [\zeta\omega_n]^2 + [\omega_n\sqrt{[1-\zeta^2]}]^2$$

$$= \zeta^2[\omega_n]^2 + [1-\zeta^2][\omega_n]^2 = [\omega_n]^2$$

This implies that r is equal to ω_n. So, to get a desired speed of response, roots must be outside a semi circle with radius ω_n.

Thus, to keep overshoots tolerable and the speed of response adequate, the roots must be in the composite wedge/circle region shown in the sketch on the next page.

The characteristic equation for an autonomous underwater vehicle or auv with a proportional controller is

$$S^3 + [NJ+MI]/[MJ]\ S^2 + [NI]/[MJ]\ S + [K_P]/[MJ] = 0$$

A general cubic characteristic equation follows from

$$(S - \lambda_1)\ \ (S - \lambda_2)\ \ (S - \lambda_3)\ \ = \ 0$$

Expansion gives

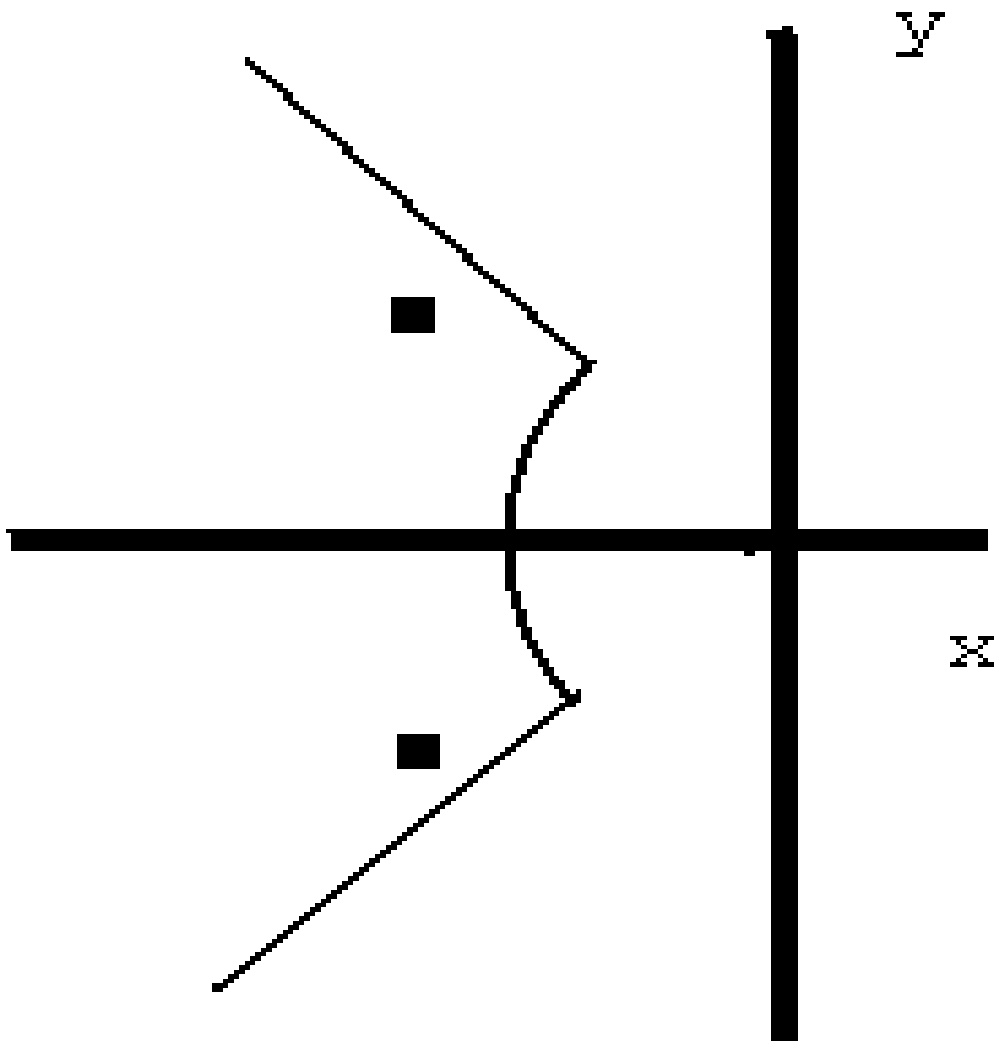
$$S^3 + a\ S^2 + b\ S + c = 0$$

$$a \ = \ - [\lambda_1 \ + \ \lambda_2 \ + \ \lambda_3] \ = \ [NJ+MI]/[MJ]$$

$$b \ = \ + [\lambda_1\ \lambda_2 \ + \ \lambda_1\ \lambda_3 \ + \ \lambda_2\ \lambda_3] \ = \ [NI]/[MJ]$$

$$c \ = \ - [\lambda_1\ \lambda_2\ \lambda_3] \ = \ [K_P]/[MJ]$$

To get good performance, we put the λ roots in the composite region in the S plane. Then the last three equations can be solved for the corresponding system parameter values.


S PLANE
y
x

NONLINEAR PHENOMENA

Linear theory predicts that, when an unstable system is disturbed from a rest state, the transients which develop grow indefinitely. For example, when transients are oscillatory, the oscillation amplitude tends to ∞ as time tends to ∞. In reality, infinite amplitudes are never observed. Sometimes large amplitudes cause the system to break down. Often nonlinearities limit amplitudes to some finite level before breakdown can occur. These finite amplitude oscillations are known as limit cycles. Sometimes limit cycle amplitudes are very small: in this case, system is often considered to be practically stable. Nonlinearities can also cause systems which are stable in a linear sense to be practically unstable.

When a system has strong multiple nonlinearities, simulation is the only option. When a system has only one strong nonlinearity, such as a switching controller, one can use its Describing Function DF. In some texts, the letter N is used to denote it instead of DF. The DF replaces the nonlinear controller.

When a system with a nonlinear controller is undergoing a limit cycle, its behavior resembles a borderline stable linear system: no growth or decay. The controller seems to be able to adjust

its gain to make the system borderline stable. The describing function DF for a nonlinear controller approximates this adjustable gain. To get DF, the system is assumed to be undergoing a limit cycle and to be nonforced. Also the signal fedback to the controller is taken to be a pure sinusoid. This is usually a good assumption because the linear elements which follow the controller generally act as a low pass filter: they let only the fundamental component out of the controller get back to the controller. When the input into the nonlinear controller is:

$$I_N = E_o \, \text{Sin}\omega t$$

its output is generally of the form:

$$O_N = O_B + O_S \, \text{Sin}\omega t + O_C \, \text{Cos}\omega t + \text{Higher Harmonics} \quad .$$

With the same input:

$$I_{DF} = E_o \, \text{Sin}\omega t$$

the describing function gives out:

$$O_{DF} = O_B + O_S \, \text{Sin}\omega t + O_C \, \text{Cos}\omega t \quad .$$

So a describing function analysis ignores higher harmonics. This is appropriate because they are filtered away anyhow. For most control situations, the bias term O_B is zero.

When a system is undergoing a limit cycle, its linear elements are forced sinusoidally by the limit cycle. In this case, each transfer function reduces to the form:

$$O/I = TF = A + Bj$$

where

$$I = \text{Sin}\omega t \qquad O = A\ \text{Sin}\omega t + B\ \text{Cos}\omega t\ .$$

By analogy, the DF for a nonlinear controller is:

$$O_{DF} / I_{DF} = DF = O_S/E_o + O_C/E_o\ j$$

where

$$I_{DF} = E_o\ \text{Sin}\omega t \qquad O_{DF} = O_S\ \text{Sin}\omega t + O_C\ \text{Cos}\omega t\ .$$

DF is essentially an amplitude dependent gain. Each E_o gives a GH Plot. Application of Nyquist theory in each case shows if E_o grows or decays. A limit cycle exists when the minus one point is on the GH Plot. When DF can be isolated from GH it is customary to divide minus one and GH by DF. In this case a limit cycle exists when the minus 1/DF curve intersects the GH/DF or G^*H Plot. Application of Nyquist theory shows if the limit cycle is stable or unstable.

As an illustration of the development of a describing function, consider the ideal relay controller. When it has a sinusoidal input, its output is a square wave. A Fourier Series analysis of a square wave gives the components:

$$O_S = 2/T \int_0^T Q(t)\ \mathrm{Sin}\omega t\ dt = 4Q_o/\pi$$

$$O_C = 2/T \int_0^T Q(t)\ \mathrm{Cos}\omega t\ dt = 0$$

$$O_B = 2/T \int_0^T Q(t)\ dt = 0$$

So the fundamental output is:

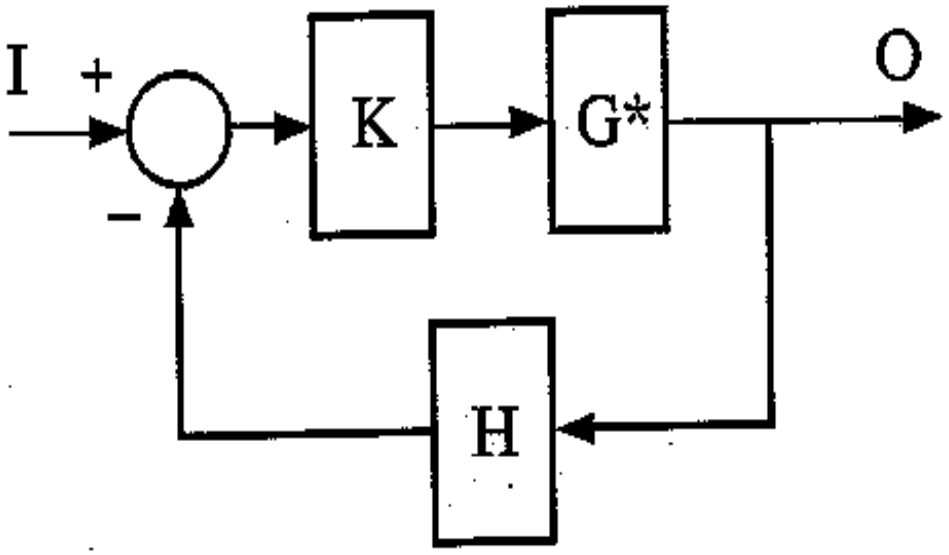
$$O_{DF} = [4Q_o/\pi]\ \mathrm{Sin}\omega t$$

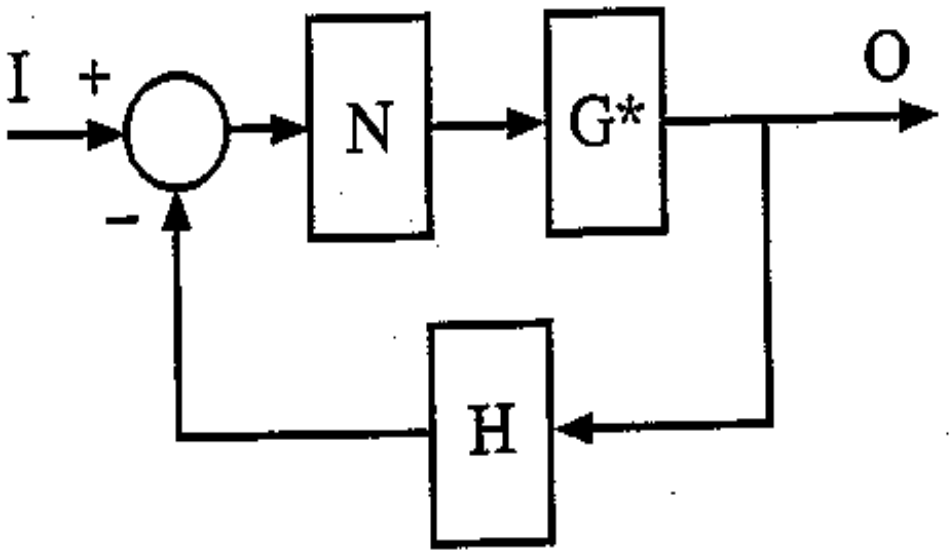
The input is

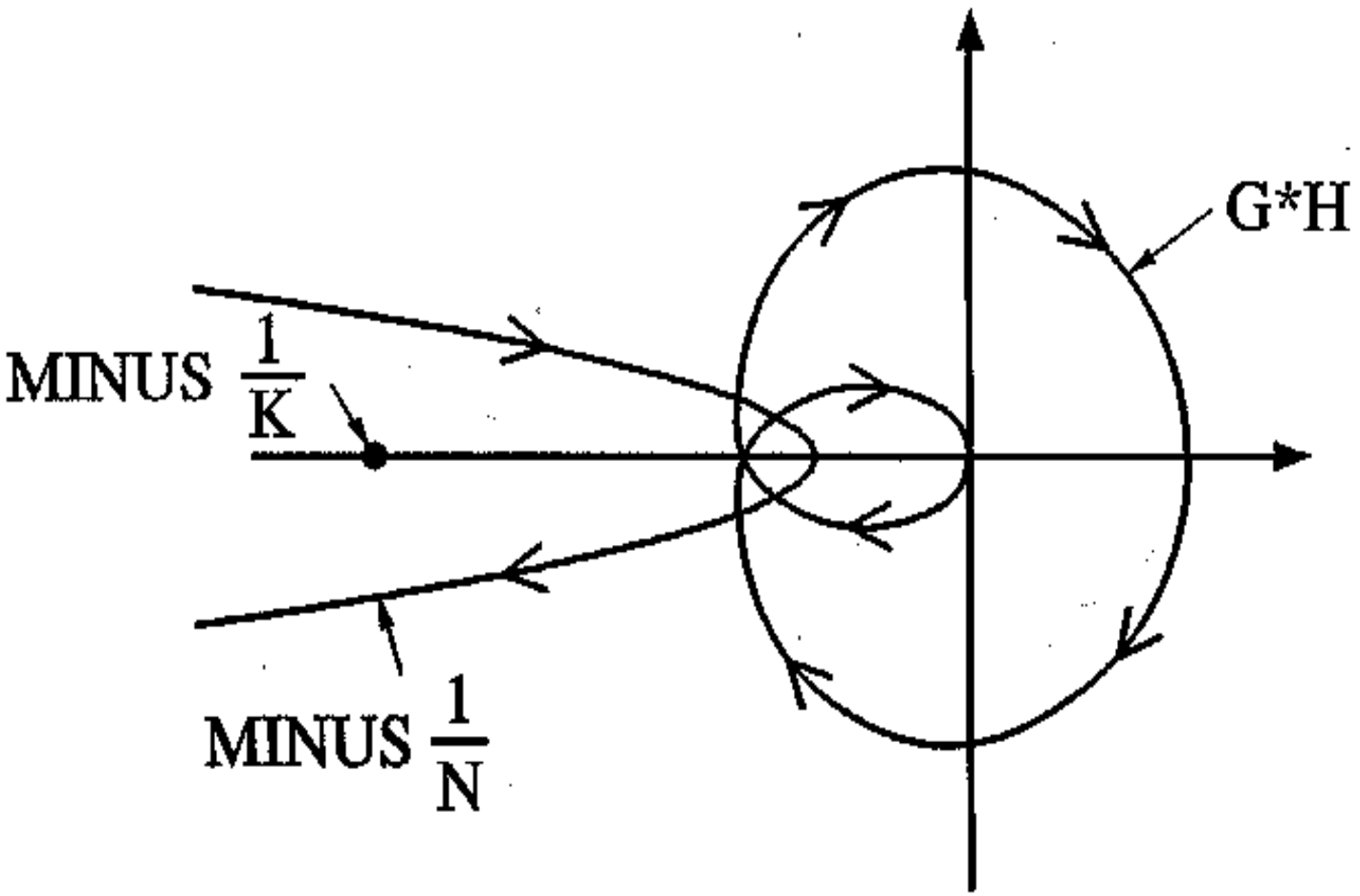
$$I_{DF} = E_o\ \mathrm{Sin}\omega t$$

So the Describing Function is

$$DF = [4Q_o]/[\pi E_o]$$


I
+
−
K
G*
O
H


I
+
−
N
G*
O
H


G*H
MINUS 1/K
MINUS 1/N

RELAY CONTROLLERS

AUTONOMOUS UNDERWATER VEHICLE

To illustrate nonlinear phenomena, we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. The schematic of the system is shown on the next page. Relay controllers resemble the proportional controller. For the proportional controller case, the governing equations for the auv are:

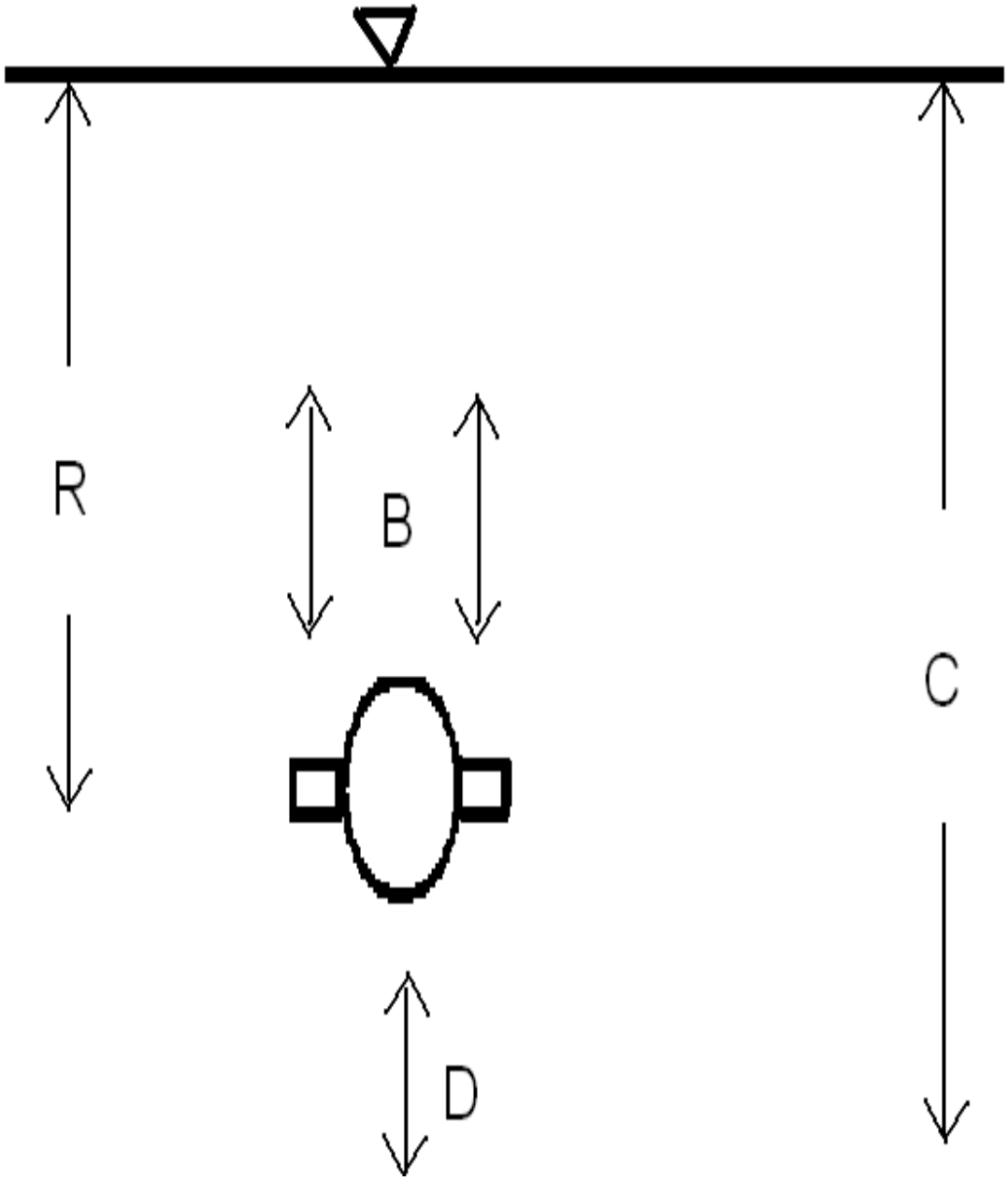
$$M\ d^2R/dt^2 = B + D - W$$

$$W = X\ dR/dt\ |dR/dt| + Y\ dR/dt$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K_P\ E \qquad E = C - R$$

where R is the depth of the auv, M is its overall mass, B is the control force from the propulsion system, D is a disturbance load caused for example by sudden weight changes, W is a drag load consisting of wake drag and wall drag, E is the depth error, C is the command depth, M X Y J I are process constants and K_P is the controller gain.


R
B
C
D

Linearization allows us to write W as:

$$W = N\, dR/dt$$

To give a numerical example we will let the parameters be:

$$M = 50.0 \quad N = 50.0$$

$$J = 0.5 \quad I = 0.1$$

Theory shows that the borderline proportional gain $\mathbf{K_P}$ for the auv is 6 and the borderline period $\mathbf{T_P}$ is 14.

The describing function for an ideal relay controller is:

$$DF = [4\, Q_o] / [\pi\, E_o]$$

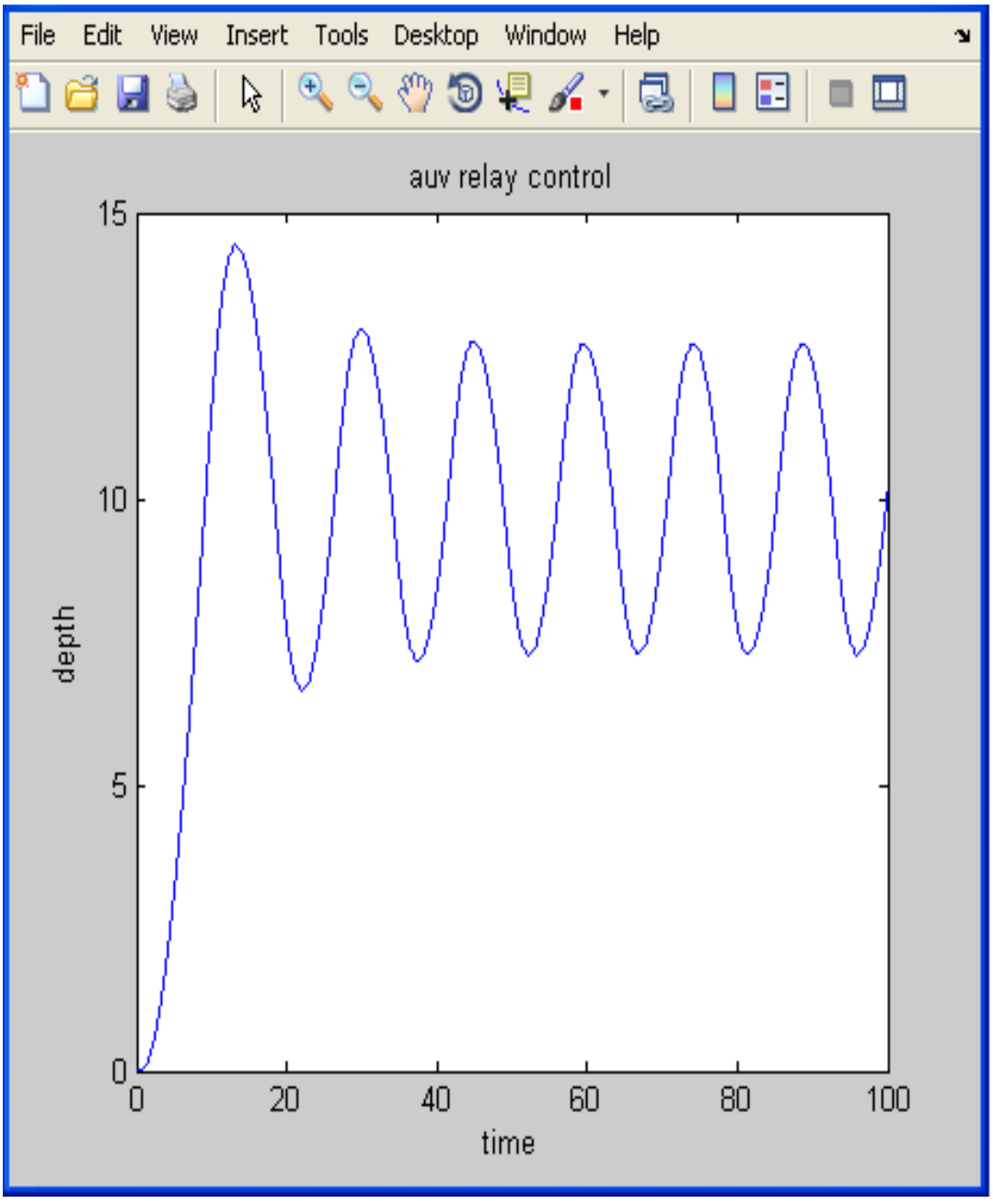
At a limit cycle this is equal to the borderline proportional gain $\mathbf{K_P}$. Setting **DF** equal to $\mathbf{K_P}$ gives:

$$\mathbf{E_o} = [4\, Q_o] / [\pi\, \mathbf{DF}] = [4Q_o] / [\pi\, \mathbf{K_P}]$$

The saturation limit for the controller is 12. Substitution into the amplitude equation gives $\mathbf{E_o}$ equal to 2.5.

An m code for the auv for the ideal relay controller case is given below. This is followed by a response generated by the code. As can be seen, it agrees with DF predictions.

```
% AUTONOMOUS UNDERWATER VEHICLE
% RELAY DEPTH CONTROLLERS
  clear all
  rold=0.0;uold=0.0;bold=0.0;
  told=0.0;m=50.0;load=0.0;
  wake=0.0;wall=50.0;
  jump=12.0;band=0.0;
  j=0.5;i=0.1;
  delt=0.01;
  target=10.0;
  for k=1:10000
  control=0.0;
  error=target-rold;
  if(error>+band) ...
  control=+jump;end;
  if(error<-band) ...
  control=-jump;end;
  drag=wake*uold*abs(uold);
  drag=drag+wall*uold;
  abe=bold+load-drag;
  xyz=control-bold*i;
  rnew=rold+delt*uold;
  unew=uold+delt*abe/m;
  bnew=bold+delt*xyz/j;
  tnew=k*delt;
  rold=rnew;uold=unew;
  bold=bnew;told=tnew;
  r(k)=rnew;t(k)=tnew;
  end; plot(t,r)
  xlabel('time')
  ylabel('depth')
  title('auv relay control')
```


File Edit View Insert Tools Desktop Window Help
auv relay control
15
10
5
0
depth
0
20
40
60
80
100
time

PIPE FLOW SETUP

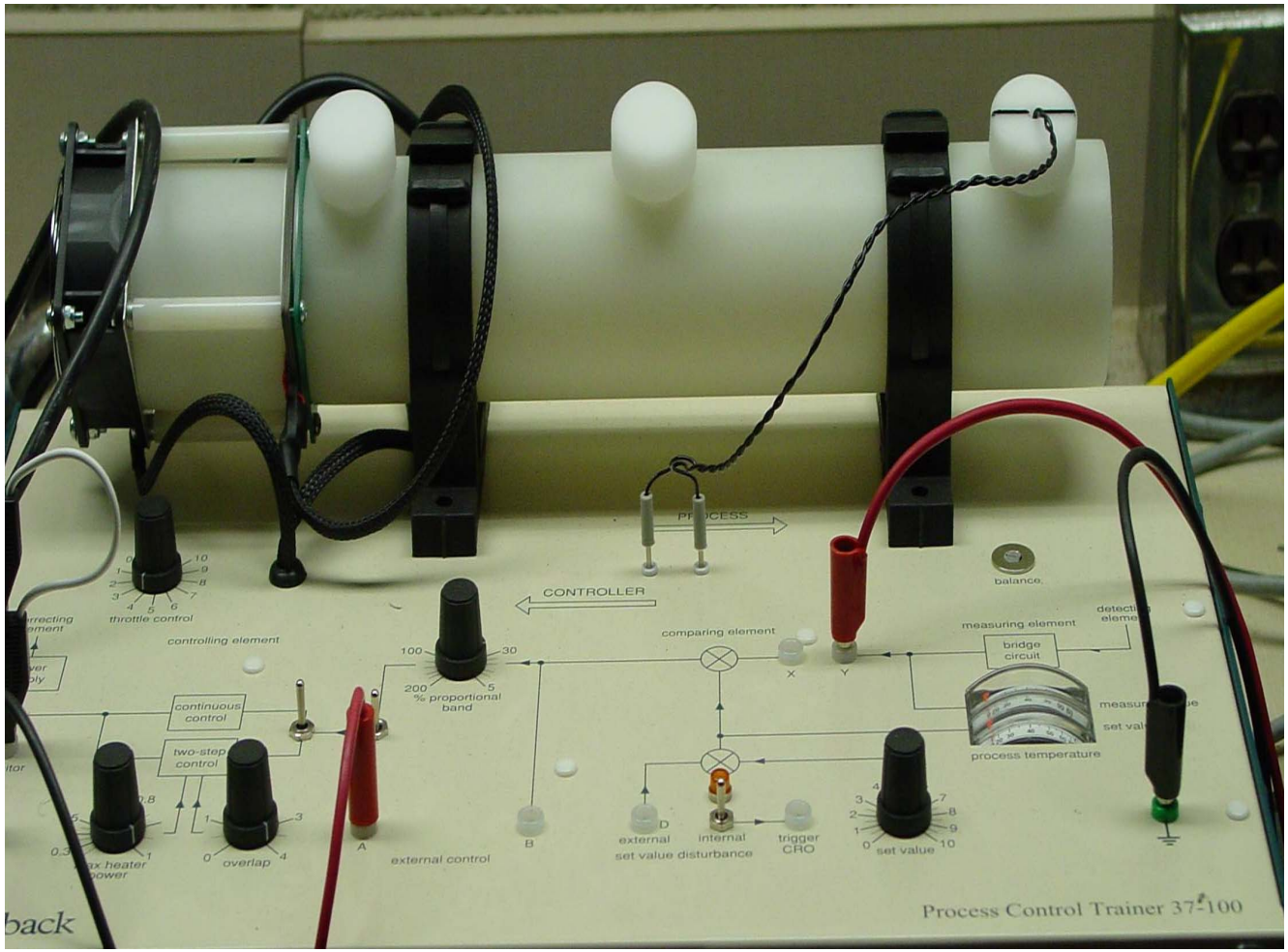
To illustrate nonlinear phenomena, we will consider the task of controlling the temperature of air flowing down a pipe. The setup is shown on the next page. Relay controllers resemble the proportional controller. For the proportional controller case, the governing equations for the setup are:

$$X\ dR/dt + Y\ R = H + D$$

$$A\ dH/dt + B\ H = Z\ Q$$

$$Q = K_P\ E \qquad E = C - \mathbf{R}$$

where R is the temperature of the air at the heater, **R** is the temperature of the air at the sensor, C is the command temperature, E is the temperature error, Q is the control signal, H is the heat generated by the heater, D is a disturbance heat (plus or minus), X Y A B Z are process constants and K_P is the controller gain. Note that **R** is what R was T seconds back in time: T is the time it takes for the air to travel down the pipe.


CONTROLLER
balance
throttle control
controlling element
comparing element
measuring element
bridge circuit
continuous control
two-step control
% proportional band
external control
external set value
disturbance
internal
trigger CRO
set value
process temperature
overlap
A
B
X
Y
Process Control Trainer 37-100

To give a numerical example we will let the parameters be:

$$X = 0.25 \quad Y = 1.0$$

$$A = 0.1 \quad B = 1.0$$

$$Z = 1.0 \quad T = 0.5$$

Theory shows that the borderline proportional gain $\mathbf{K_P}$ for the setup is 1.5 and the borderline period $\mathbf{T_P}$ is .

The describing function for an ideal relay controller is:

$$DF = [4\ Q_o] / [\pi\, E_o]$$

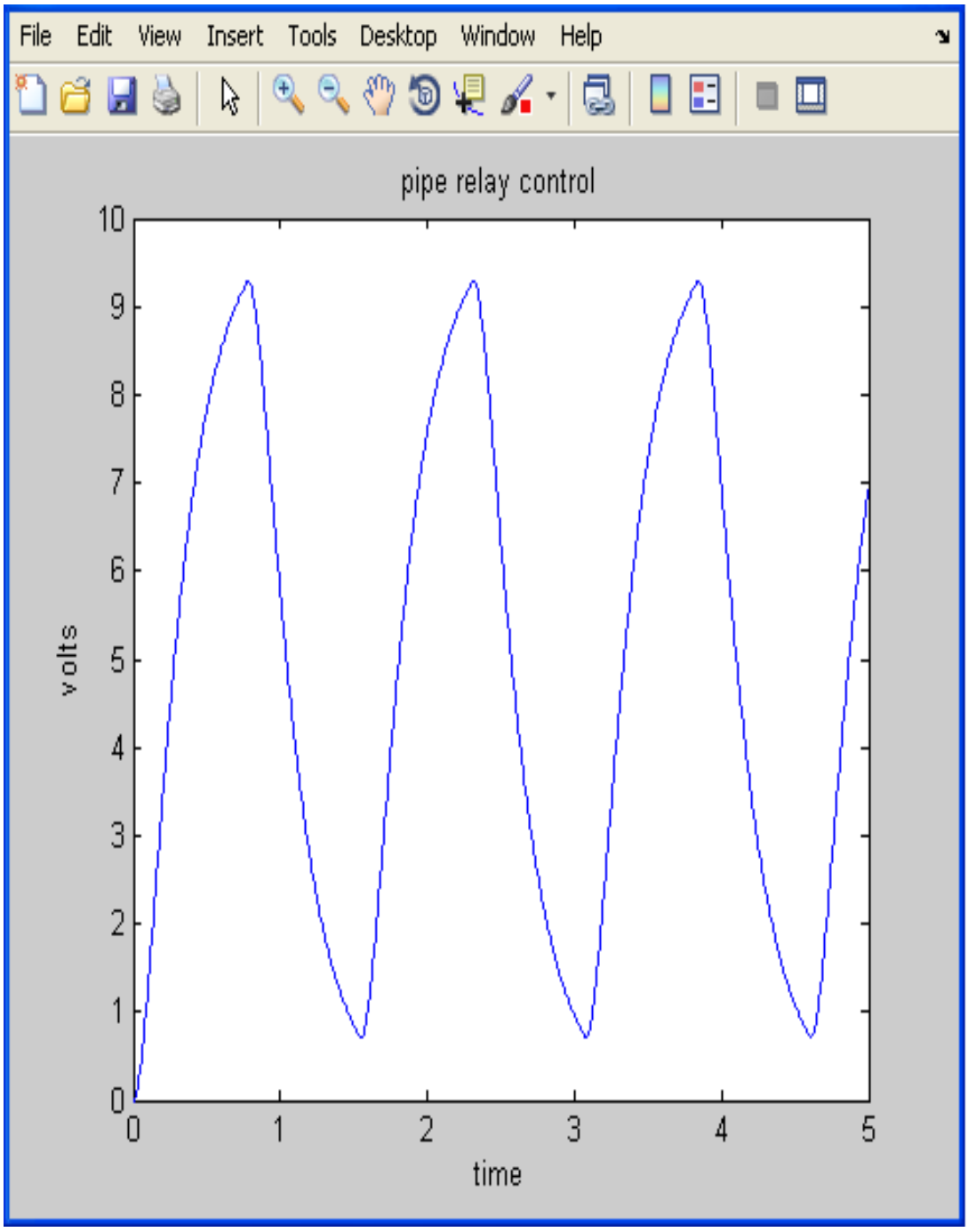
At a limit cycle this is equal to the borderline proportional gain $\mathbf{K_P}$. Setting **DF** equal to $\mathbf{K_P}$ gives:

$$\mathbf{E_o} = [4\ Q_o] / [\pi\, \mathbf{DF}] = [4Q_o] / [\pi\, \mathbf{K_P}]$$

The saturation limit for the controller is 5. Substitution into the amplitude equation gives $\mathbf{E_o}$ equal to 4.2.

An m code for the setup for the ideal relay controller case is given below. This is followed by a response generated by the code. As can be seen, it agrees with DF predictions.

```
% PIPE FLOW SETUP
% RELAY ONTROLLERS
ROLD=0.0;HOLD=0.0;SENSOR=ROLD;
TARGET=5.0;LOAD=0.0;DUMP=10.0;
X=0.25;Y=1.0;A=0.1;B=1.0;Z=1.0;
NIT=1000;MIT=100;TIME=0.0;
BIAS=5.0;JUMP=5.0;BAND=0.0;
DELT=0.005;
for IT=1:NIT
TIME=TIME+DELT;
if(IT>MIT) ...
 SENSOR=R(IT-MIT); end;
ERROR=TARGET-SENSOR;
CONTROL=BIAS;
if(ERROR>+BAND) ...
  CONTROL=BIAS+JUMP;end;
if(ERROR<-BAND) ...
  CONTROL=BIAS-JUMP;end;
ABC=Z*CONTROL-B*HOLD;
XYZ=HOLD+LOAD-Y*ROLD;
HNEW=HOLD+DELT*ABC/A;
RNEW=ROLD+DELT*XYZ/X;
T(IT)=TIME;R(IT)=RNEW;
ROLD=RNEW;HOLD=HNEW;
end; plot(T,R)
xlabel('time')
ylabel('volts')
title('pipe relay control')
```


File Edit View Insert Tools Desktop Window Help
pipe relay control
volts
time
0
1
2
3
4
5
6
7
8
9
10

DIGITAL CONTROL

In almost every control system today, a computer samples the state of the system and takes corrective action within a control loop. The rate at which it does this can cause performance to degrade. One can study the phenomena using new from old time stepping or one can study it using Z transforms.

As an illustration of the Z transform method, we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. Its governing equations are:

$$M\ d^2R/dt^2 + N\ dR/dt = B + D - W$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K\ E \qquad E = C - R$$

Laplace Transformation of the governing equations gives

$$(M\ S^2 + N\ S)\ R = B + D$$

$$(J\ S + I)\ B = Q$$

$$Q = K\ E \qquad E = C - R$$

For the command case, the overall transfer function is

$$\frac{R}{C} = \frac{K}{MJ\ S^3 + (NJ+MI)\ S^2 + NI\ S + K}$$

Let the parameters be:

$$M = 50.0 \quad N = 50.0.$$

$$J = 0.5 \quad I = 0.1 \quad K = 3$$

Substitution gives

$$\frac{R}{C} = \frac{3}{25\ S^3 + 30\ S^2 + 5\ S + 3}$$

The forward path transfer function G(S) is

$$\frac{K}{MJ\ S^3 + (NJ+MI)\ S^2 + NI\ S}$$

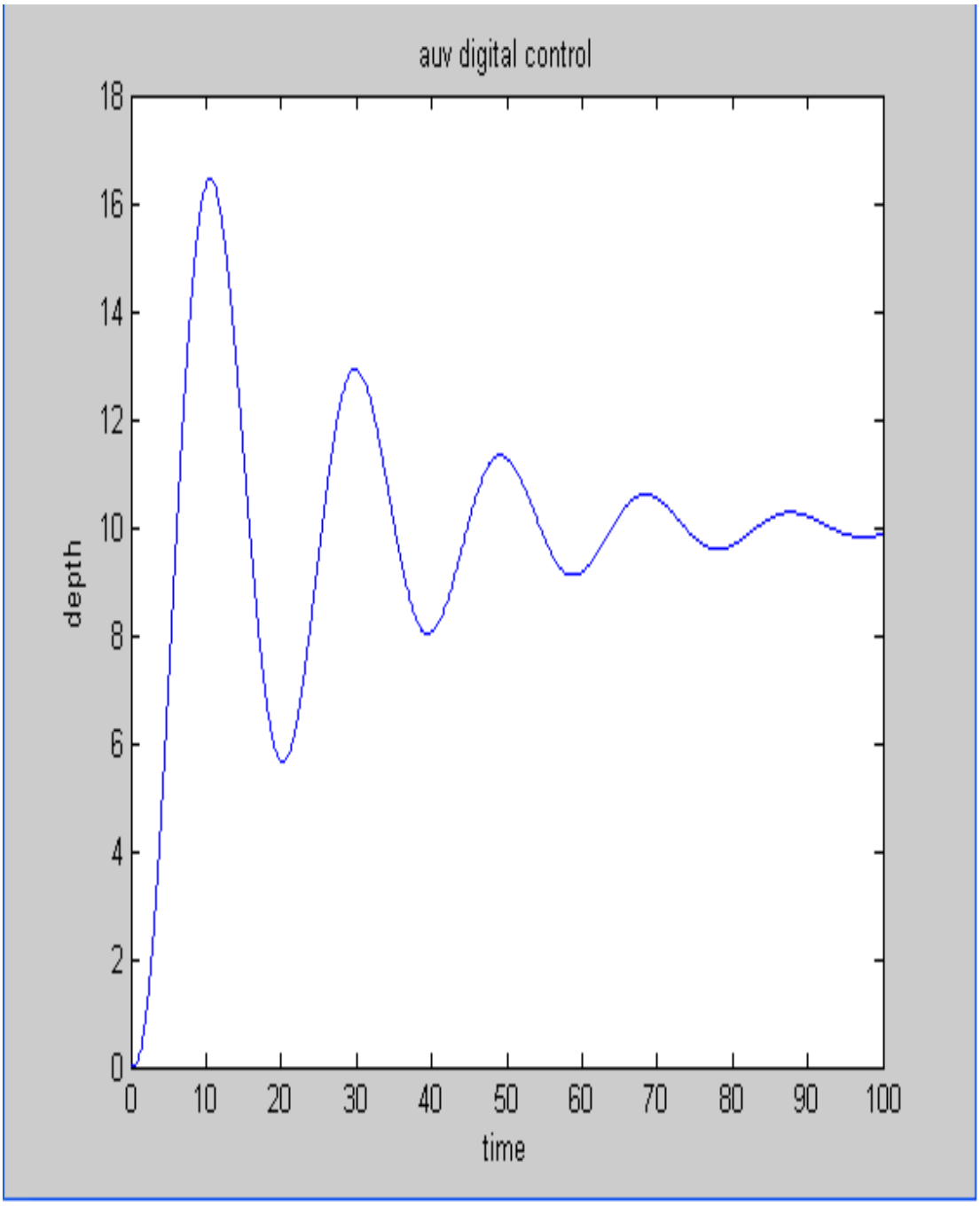
$$= \frac{3}{25\ S^3 + 30\ S^2 + 5\ S}$$

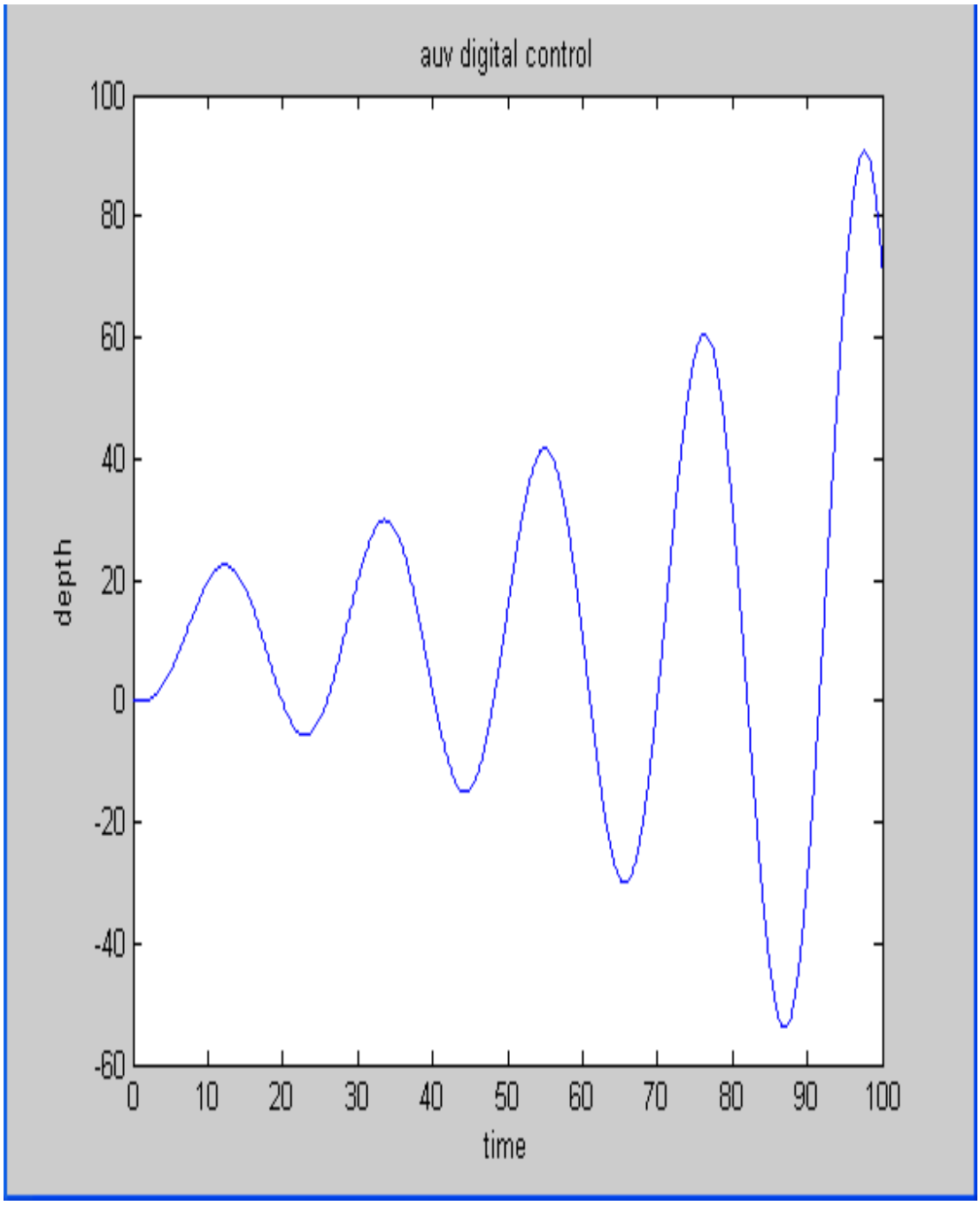
The matlab c2d function can be used to convert the continuous transfer function G(S) to the discrete transfer function G(Z). The system has unity feedback. Its transfer function H(Z) is 1/Z. The overall transfer function is:

$$\frac{G(Z)}{1 + G(Z)\ H(Z)}$$

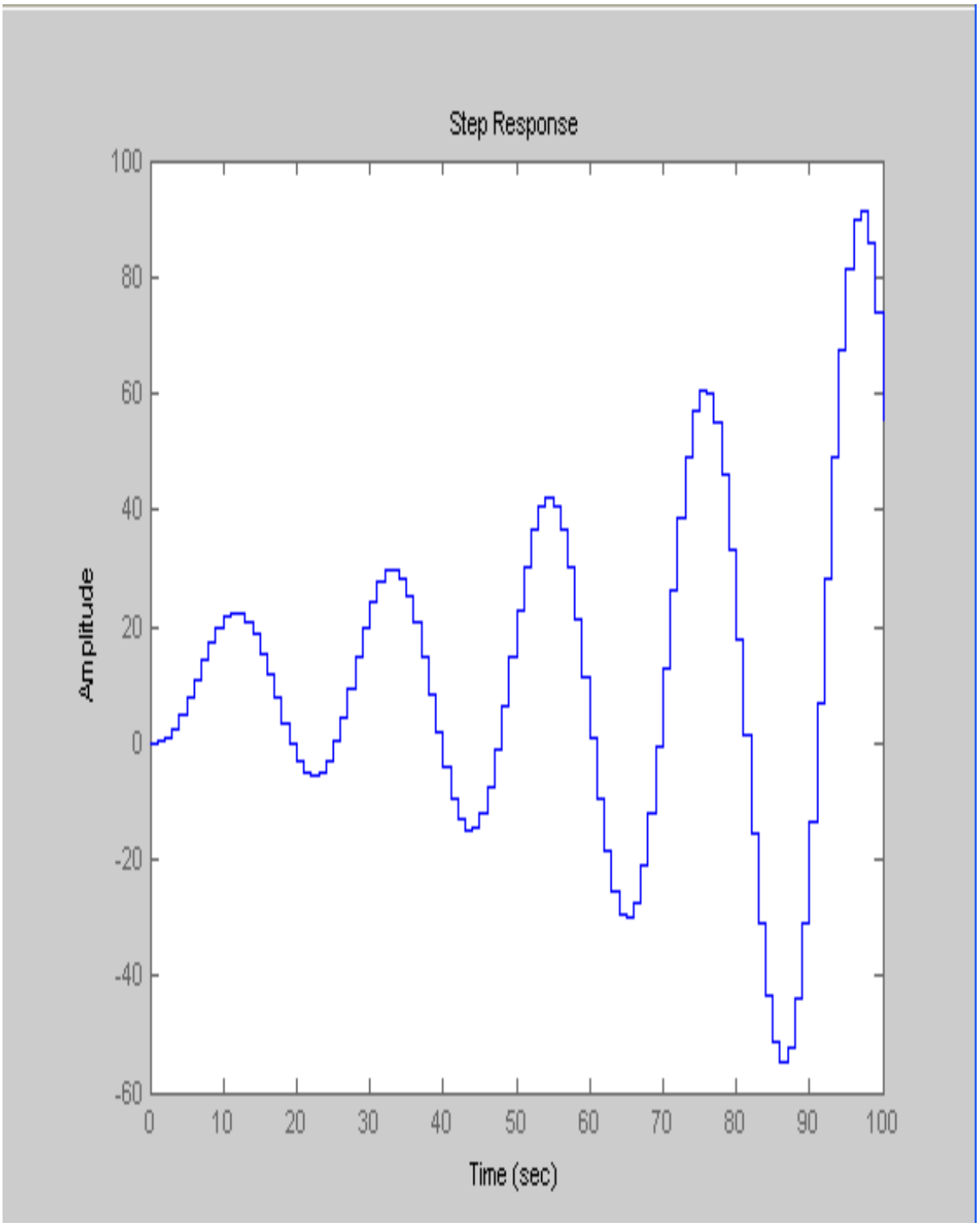
To examine the influence of digital phenomena on the performance of the system, we consider its response to a step command with a height of 10. A simulation m code for the system is given on the next page. The response of the continuous system follows it. The response of the discrete system for the case where the loop period is 1 follows the continuous response. As can be seen, the system has become unstable. Some matlab script which makes use of Z transforms follows the simulation responses. Its response for the loop period equal to 1 case agrees with the simulation. A SIMULINK block diagram for the discrete system follows the script response. Its response agrees with the simulation and script responses.

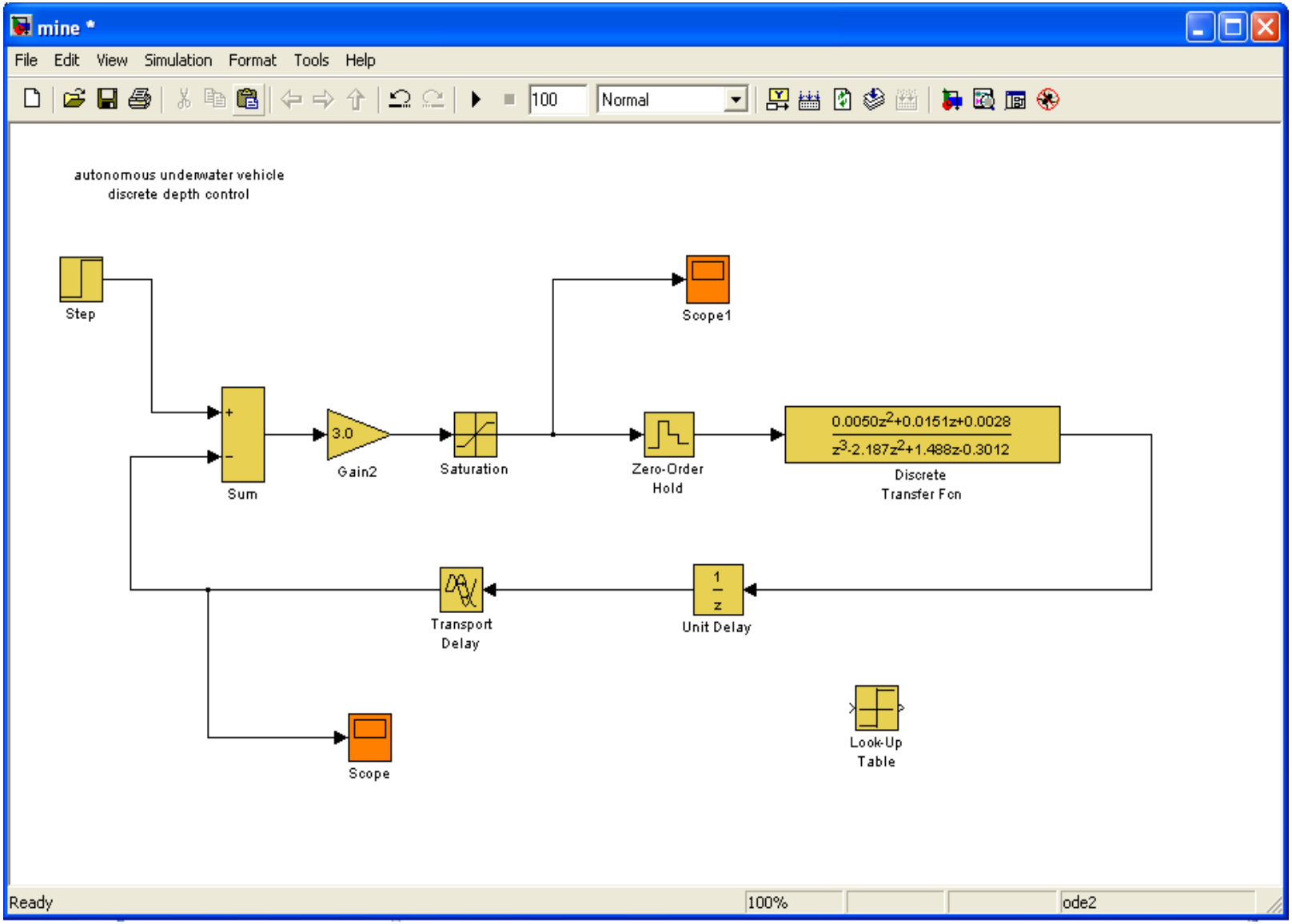
```
% AUTONOMOUS UNDERWATER VEHICLE
% SIMULATION OF DIGITAL CONTROL
clear all
rold=0.0;uold=0.0;bold=0.0;
told=0.0;m=50.0;load=0.0;
wake=0.0;wall=50.0;sum=0.0;
j=0.5;i=0.1;wrong=0.0;
gp=3.6;gi=0.54;gd=6.3;
gp=3.0;gi=0.0;gd=0.0;
jit=0;mit=100;nit=10000;
delt=0.01;step=delt*mit;
sensor=rold;signal=0.0;
target=10.0;
for it=1:nit
tnew=it*delt;
jit=jit+1;
if(jit==1) ...
 sensor=rold; end;
if(jit==mit)
error=target-sensor;
rate=(error-wrong)/step;
control=gp*error;
control=control+gi*sum;
control=control+gd*rate;
if(control>+1111111.0) ...
 control=+1111111.0;end;
if(control<-1111111.0) ...
 control=-1111111.0;end;
signal=control;
sum=sum+step*error;
wrong=error;
jit=0;end;
drag=wake*uold*abs(uold);
drag=drag+wall*uold;
abe=bold+load-drag;
xyz=signal-bold*i;
rnew=rold+delt*uold;
unew=uold+delt*abe/m;
bnew=bold+delt*xyz/j;
rold=rnew;uold=unew;
bold=bnew;told=tnew;
r(it)=rnew;u(it)=unew;
b(it)=bnew;t(it)=tnew;
end; plot(t,r)
xlabel('time')
ylabel('depth')
title('auv digital control')
```

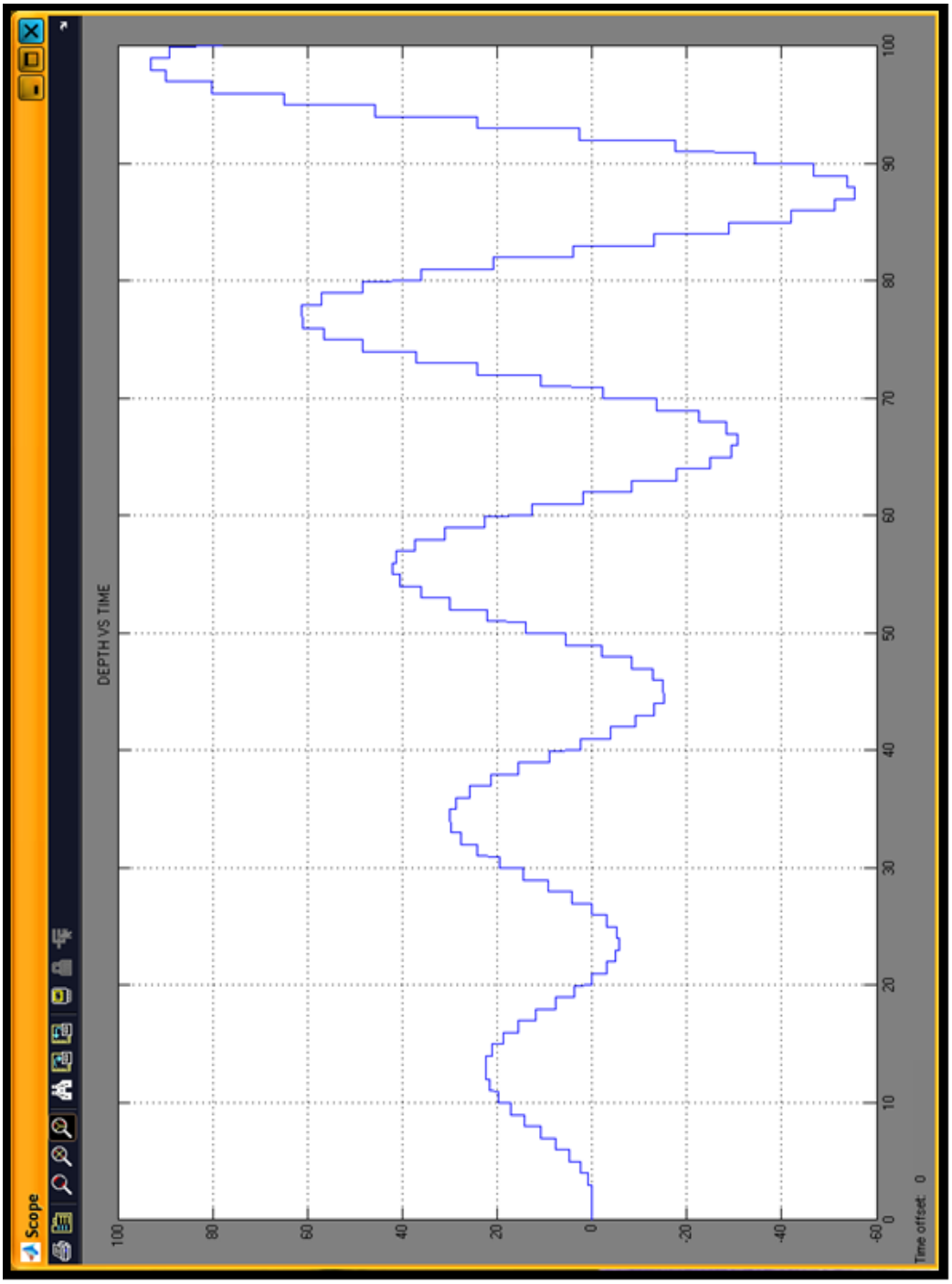

auv digital control
depth
time
0
2
4
6
8
10
12
14
16
18
0
10
20
30
40
50
60
70
80
90
100


auv digital control
depth
time
100
80
60
40
20
0
-20
-40
-60
0
10
20
30
40
50
60
70
80
90
100

```
>>
>> % auv depth control
>>
>> z=tf('z',1); dh=1/z;
>>
>> g=tf([3],[25 30 5 0]);
>>
>> dg=c2d(g,1,'zoh');
>>
>> sys=feedback(dg,dh)

Transfer function:
     0.01506 z^3 + 0.04541 z^2 + 0.008277 z
-----------------------------------------------
z^4 - 2.187 z^3 + 1.503 z^2 - 0.2558 z + 0.008277

Sampling time: 1
>>
>> step(10*sys,100)
>>
```


Step Response
Amplitude
Time (sec)
100
80
60
40
20
0
-20
-40
-60
0
10
20
30
40
50
60
70
80
90
100


mine *
File Edit View Simulation Format Tools Help
100
Normal
autonomous underwater vehicle
discrete depth control
Step
Scope1
Sum
3.0
Gain2
Saturation
Zero-Order
Hold
0.0050z2+0.0151z+0.0028
z3-2.187z2+1.488z-0.3012
Discrete
Transfer Fcn
Transport
Delay
1
z
Unit Delay
Scope
Look-Up
Table
Ready
100%
ode2


Scope
DEPTH VS TIME
100
80
60
40
20
0
-20
-40
-60
0
10
20
30
40
50
60
70
80
90
100
Time offset: 0

SAMPLING AND CONSTRUCTION OF SIGNALS

Today, most systems are controlled by digital computers. Within a computer control loop, sampling of sensor signals is usually done first followed by calculation of control signals followed by control signal construction using ZOH. The time it takes to move once through the control loop is known as the loop period. This is usually much less than the basic system period. When the two periods are comparable, performance is usually very poor.

One can use digital simulation or time stepping to study the performance of such a system. One can also use pulse transfer functions. For this, one must first focus in on the points in time where sensors are sampled because control signals are based solely on state of system at these instants. Mathematically sampling is done by multiplying each signal of interest by a train of unit impulses. The area of each impulse generated is a signal level at a sampling instant. For example, sampling the unit impulse response function h(t) gives:

$$h^*(t) = \Sigma \; h(nT) \; \delta(t-nT) \quad .$$

where T is the sampling period. Laplace Transformation of this followed by manipulation gives the pulse transfer function H(Z):

$$H(Z) = \Sigma \; h(nT) \; Z^{-n} \qquad \text{where} \quad Z=e^{ST} \quad .$$

The response of a system to a unit impulse is of the form:

$$h(t) = \Sigma\ \Gamma\ e^{\lambda t} \quad .$$

Laplace Transformation of this gives:

$$H(S) = \Sigma\ \Gamma\ /\ (S-\lambda) \quad .$$

Z Transformation gives:

$$H(Z) = \Sigma\ \Gamma\ /\ (1 - e^{\lambda T}\ Z^{-1}) \quad .$$

This gives the roots $Z = e^{\lambda T}$. Substitution into this shows that stable region in S plane maps to the inside of a unit circle in Z plane. If roots fall outside unit circle, the system is unstable.

Often manipulation gives H(Z) as a ratio of two polynomials:

$$H(Z) = N(Z)\ /\ D(Z)$$

where D(Z)=0 is the system characteristic equation. Long division of D(Z) into N(Z) can be used to reduce H(Z) to the basic form:

$$H(Z) = \Sigma\ h(nT)\ Z^{-n} \quad .$$

Inspection of this tells if the system is stable or unstable.

STATE SPACE CONTROL

State space control works with the ordinary differential equations and algebraic equations governing the state of the system. It puts these equations in matrix form. Manipulation of the matrix equations gives something called the State Transition Matrix. Setting the determinant of this matrix to zero gives the characteristic equation for the system. State space control tries to control all of the states of the system. For this, instead of just a single gain, it uses a gain matrix. To control all states, all states must be sensed. For some systems, this may be impossible. In these cases, the governing equations are used to estimate the states. State space control is computationally expensive and may be inappropriate for pic controlled systems.

As an illustration of state space control, we will consider the task of controlling the submergence depth of a small autonomous underwater vehicle or auv. Its governing equations are:

$$M\ d^2R/dt^2 + N\ dR/dt = B + D$$

$$J\ dB/dt + I\ B = Q$$

$$Q = K\ (C - R)$$

Manipulation of these equations gives:

$$dR/dt = U$$

$$dU/dt = B/M + D/M - N/M\ U$$

$$dB/dt = K\ (C-R)/J - I/J\ B$$

When C and D are zero, one gets the matrix equation:

$$\begin{vmatrix} dR/dt \\ dU/dt \\ dB/dt \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 \\ 0 & -N/M & +1/M \\ -K/J & 0 & -I/J \end{vmatrix} \begin{vmatrix} R \\ U \\ B \end{vmatrix}$$

The response following an impulse will have the form:

$$R = R_o\ e^{\lambda t} \qquad U = U_o\ e^{\lambda t} \qquad B = B_o\ e^{\lambda t}$$

Substitution into the matrix equation gives

$$\begin{vmatrix} \lambda & -1 & 0 \\ 0 & \lambda+N/M & -1/M \\ +K/J & 0 & \lambda+I/J \end{vmatrix} \begin{vmatrix} R_o \\ U_o \\ B_o \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix}$$

Setting the matrix determinant to zero gives:

$$\lambda\ [\lambda+N/M]\ [\lambda+I/J] \ + \ [K/J]\ [1/M] \ = \ 0$$

Manipulation gives:

$$MJ\ \lambda^3 \ + \ [MI+NJ]\ \lambda^2 \ + NI\ \lambda \ + \ K \ = \ 0$$

This is the overall system characteristic equation. It is basically the same as that obtained from classical control:

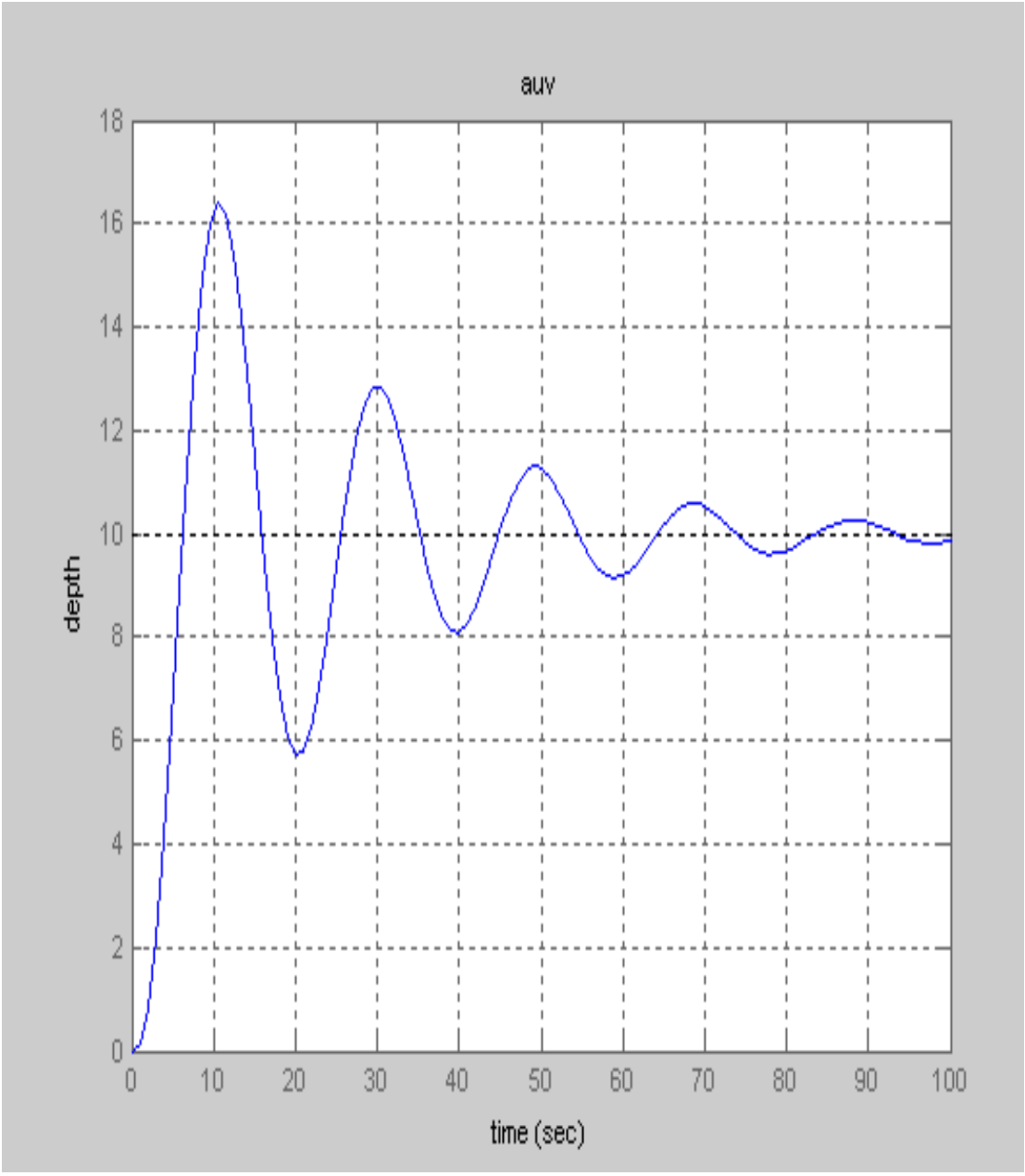
$$MJ\ S^3 \ + \ [MI+NJ]\ S^2 \ + NI\ S \ + \ K \ = \ 0$$

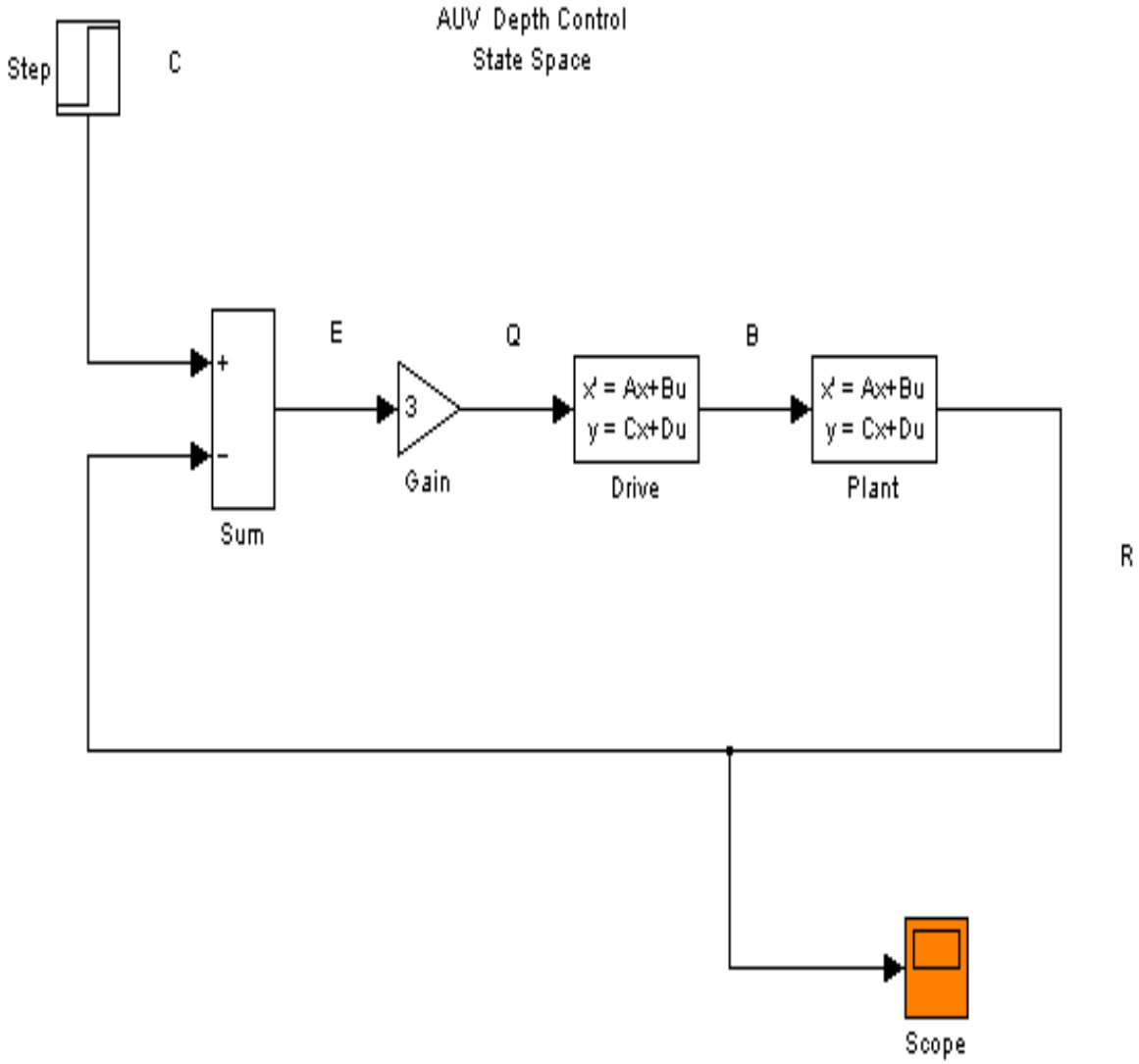
State space control uses matrix manipulation to get many other system properties. Details are beyond the scope of this note.

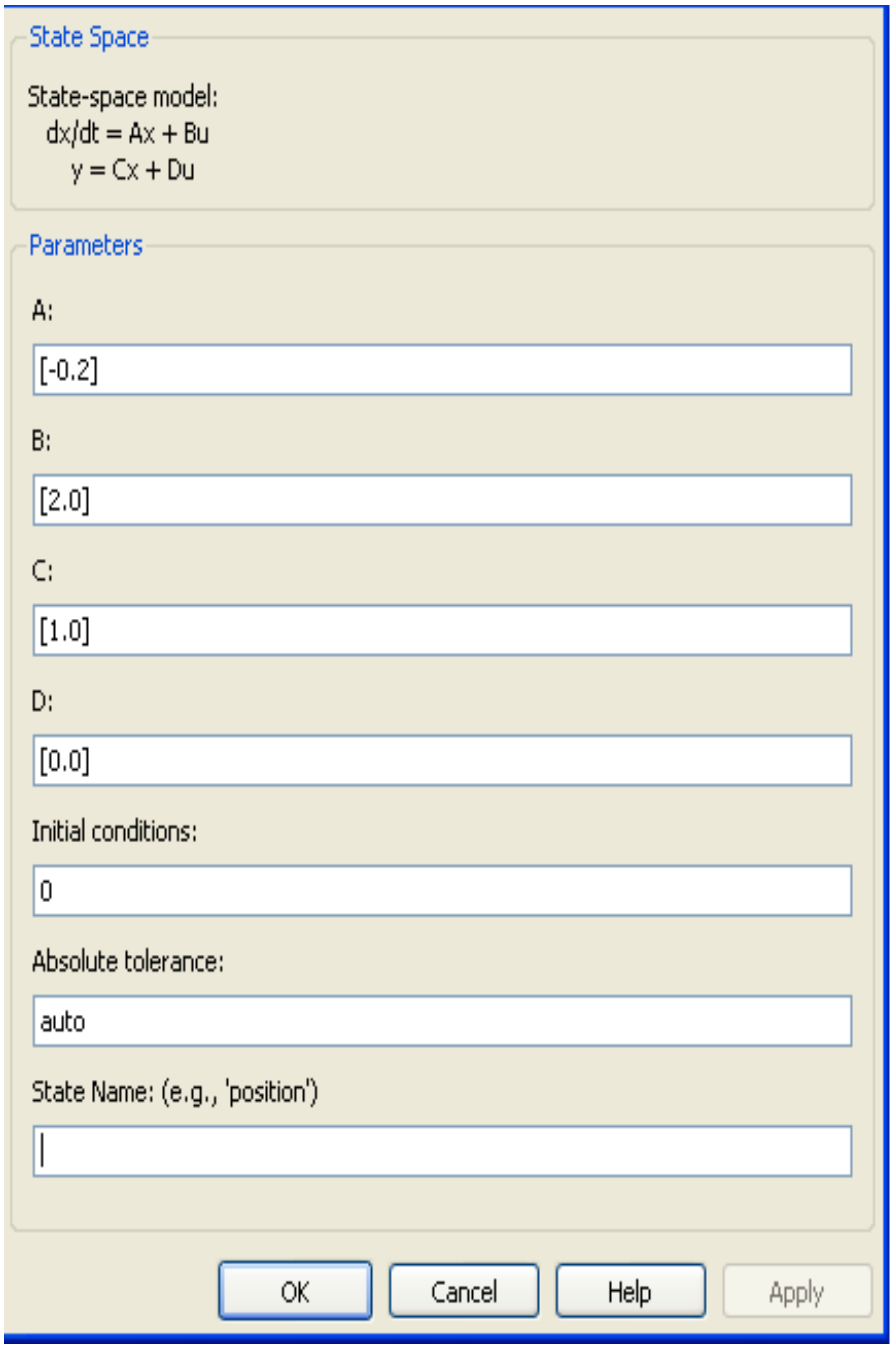
Matlab has a function SS which converts classical control transfer functions to state space matrix form. The m code on the next page shows an application of this for auv depth control. The step response produced by the code follows it and is identical to that obtained by classical control. This is not surprising because they are both based on the same equations.

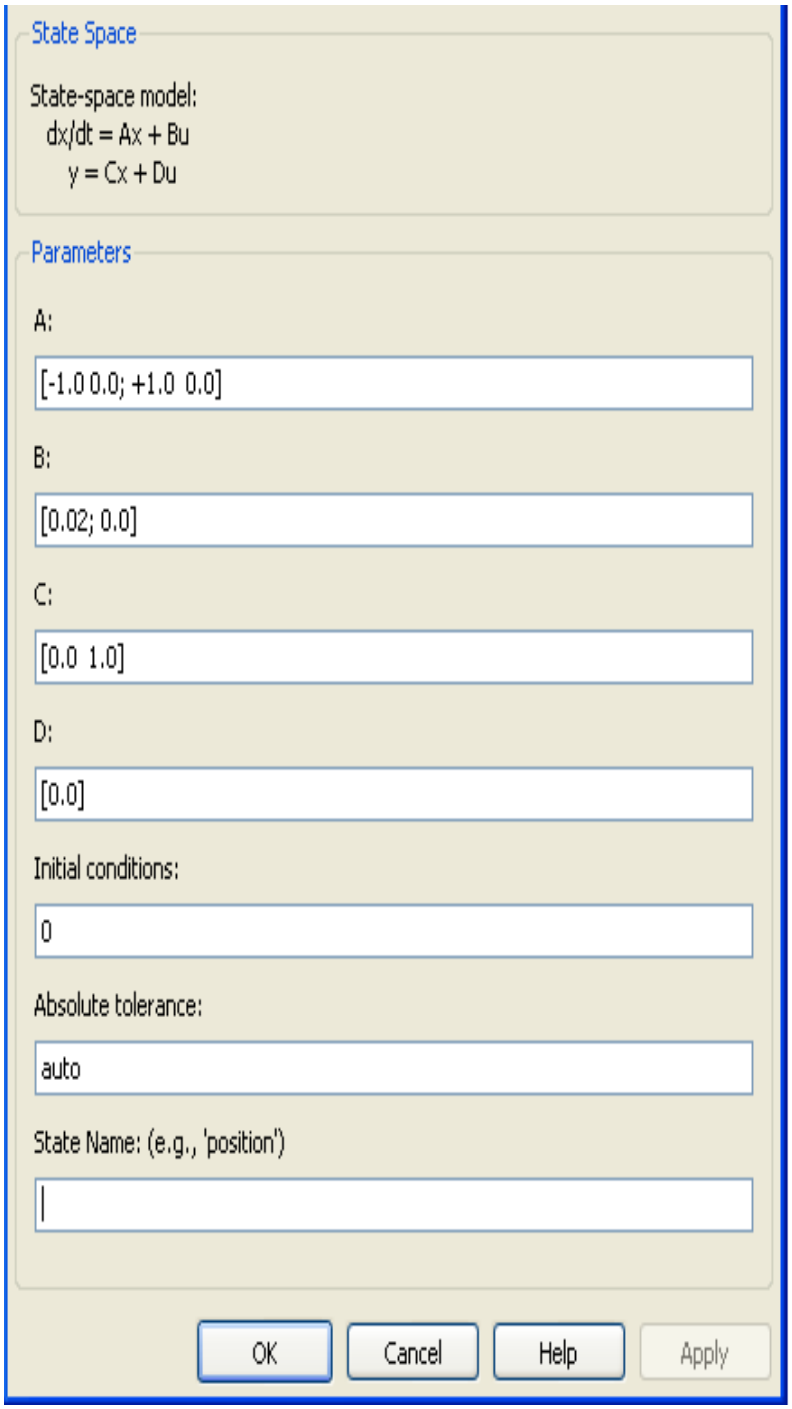
Simulink has state space blocks. A state space block diagram for the auv depth control case is shown immediately after the SS code and its response. The drive and plant state space menus are also shown. In them, x indicates the states, u indicates the inputs and y indicates the outputs. The step response produced by the state space block diagram is basically the same as that produced by the m code. Again, this is not surprising.

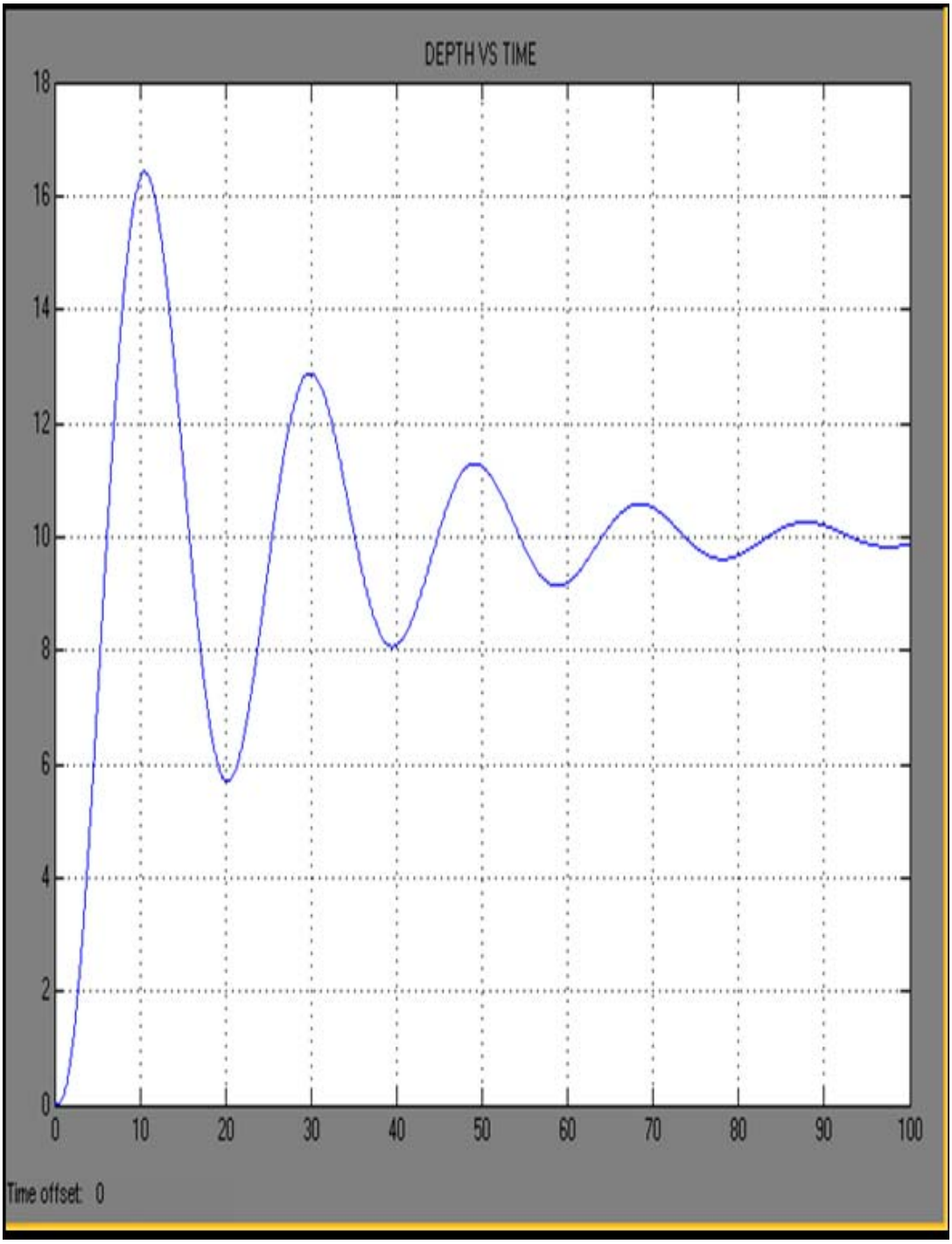
```
% AUV DEPTH CONTROL

% STATE SPACE MODEL

clear all
% gain
K=3.0;
% drive
num=[1];
den=[0.5 0.1];
d=tf(num,den);
sd=ss(d);
% plant
num=[1];
den=[50.0 50.0 0];
p=tf(num,den);
sp=ss(p);
% forward path
sg=K*series(sd,sp);
% G/[1+GH]
sys=feedback(sg,1);
% step response
step(10*sys,100)
title('auv')
ylabel('depth')
xlabel('time')
grid
```


auv
depth
time (sec)
0
2
4
6
8
10
12
14
16
18
0
10
20
30
40
50
60
70
80
90
100


AUV Depth Control
State Space
Step
C
E
Q
B
+
-
3
x' = Ax+Bu
y = Cx+Du
x' = Ax+Bu
y = Cx+Du
Sum
Gain
Drive
Plant
R
Scope


State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
[-0.2]
B:
[2.0]
C:
[1.0]
D:
[0.0]
Initial conditions:
0
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply


State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
[-1.0 0.0; +1.0 0.0]
B:
[0.02; 0.0]
C:
[0.0 1.0]
D:
[0.0]
Initial conditions:
0
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply


DEPTH VS TIME
18
16
14
12
10
8
6
4
2
0
0
10
20
30
40
50
60
70
80
90
100
Time offset: 0