AUTOMATIC CONTROL ENGINEERING

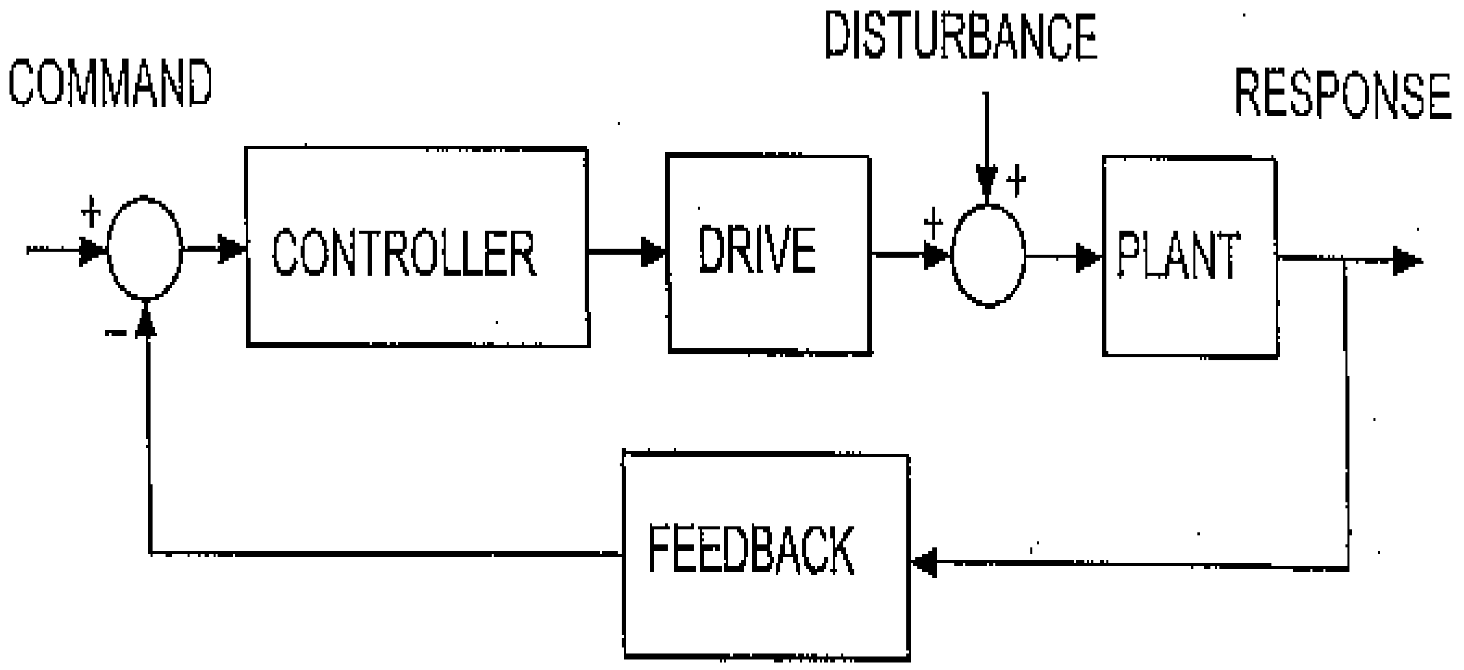
FEEDBACK CONTROL CONCEPT

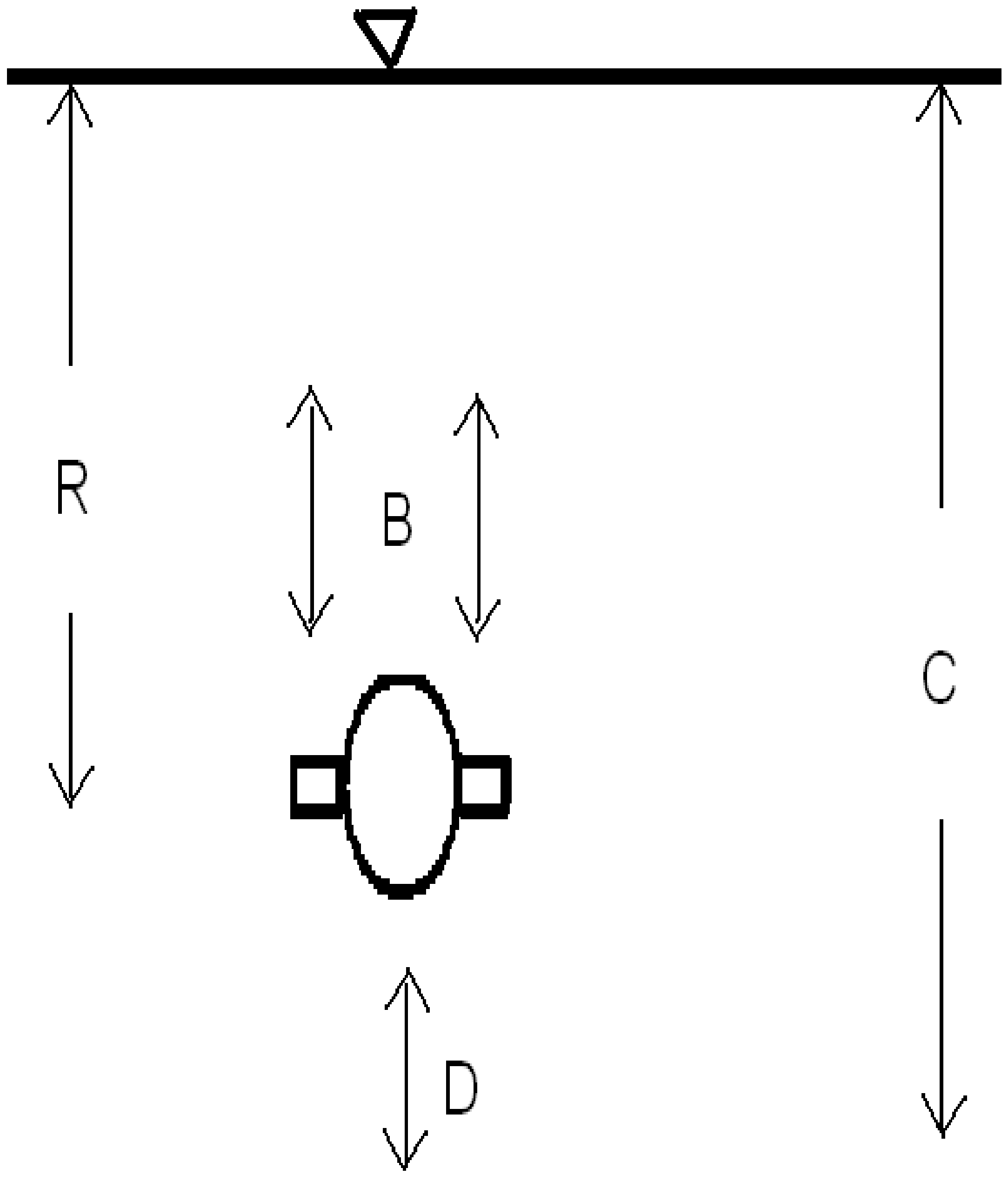
The sketch on the next page shows a typical feedback or error driven control system. What has to be controlled is generally referred to as the plant. What the plant is doing is known as its response. What it should be doing is known as the command. The plant receives a control signal from a drive and 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
+
−
CONTROLLER
DRIVE
DISTURBANCE
+
+
PLANT
RESPONSE
FEEDBACK


R
B
C
D

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

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

Substitution into the modified drive equation gives

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

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

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

This equation is of the form:

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

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

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

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

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

Manipulation of these equations gives

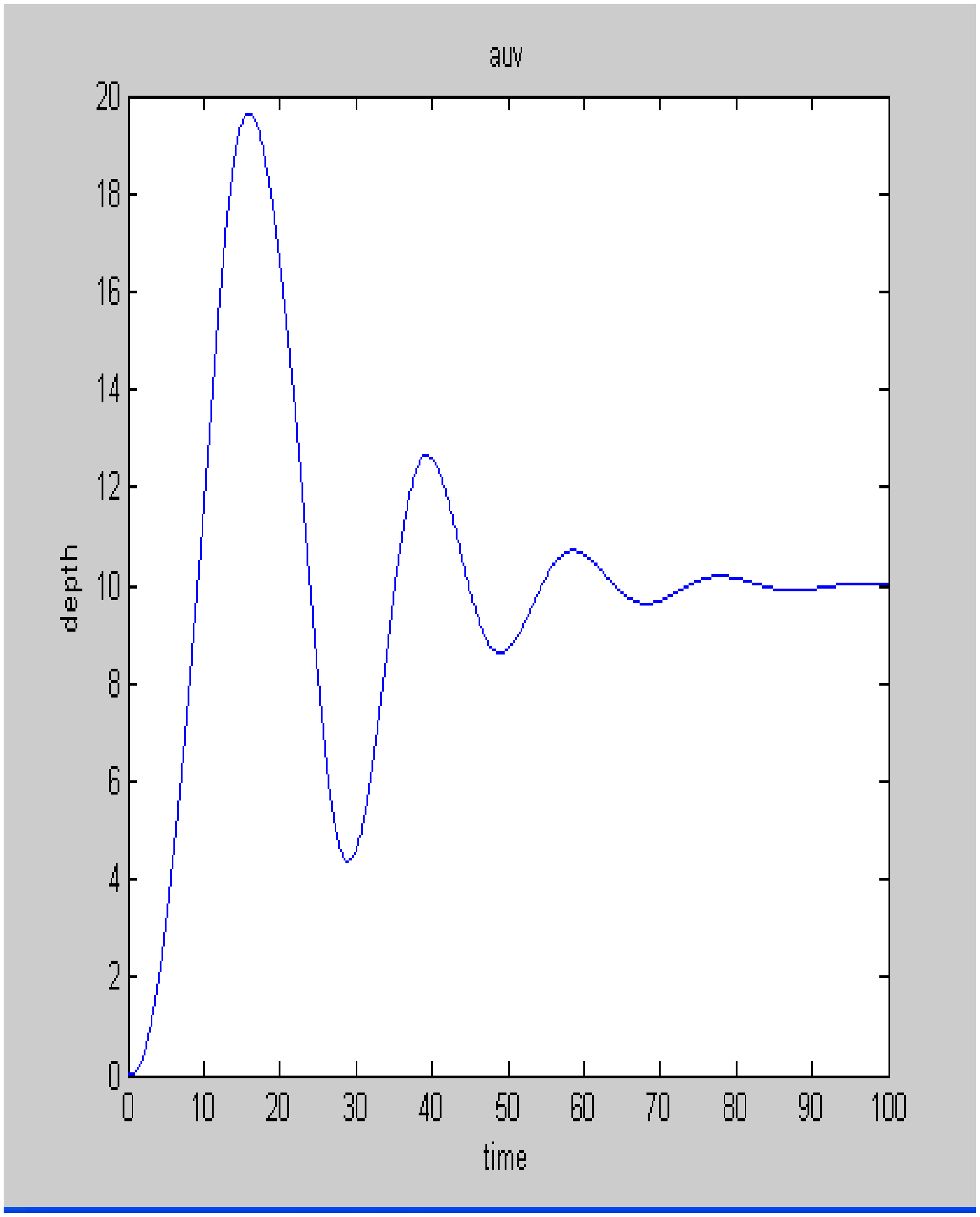
$$R_o = C_o + I D_o / \mathbf{K_P}$$

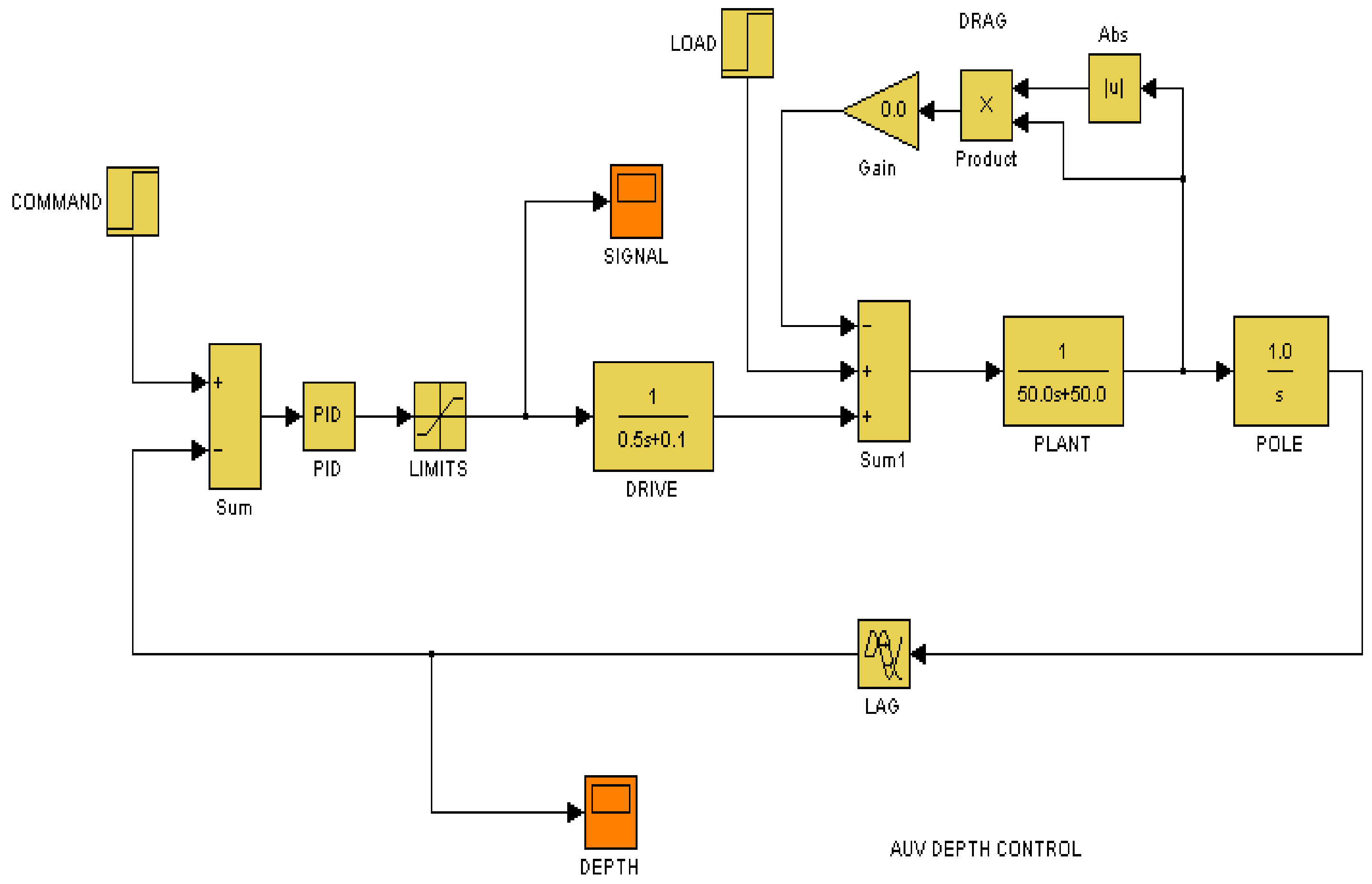
$$\boldsymbol{\omega}^2 = [I N] / [J M]$$

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

For the illustration we let : M=50 N=50 J=0.5 I=0.1. The above equations give $\boldsymbol{\omega}$=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


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

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:

$$A\ [\ X\ d^2R/dt^2\ +\ Y\ dR/dt\ -\ dD/dt]$$

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

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

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

Substitution into the modified drive equation gives

$$-\ A\ X\ \boldsymbol{\omega}^2\ \Delta R\ \mathrm{Sin}[\boldsymbol{\omega} t]\ +\ A\ Y\ \boldsymbol{\omega}\ \Delta R\ \mathrm{Cos}[\boldsymbol{\omega} t]$$

$$+\ B\ X\ \boldsymbol{\omega}\ \Delta R\ \mathrm{Cos}[\boldsymbol{\omega} t]\ +\ B\ Y\ R_o\ +\ B\ Y\ \Delta R\ \mathrm{Sin}[\boldsymbol{\omega} t]$$

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

A trigonometric identity gives

$$\mathrm{Sin}[\boldsymbol{\omega}(t-T)]\ =\ \mathrm{Sin}[\boldsymbol{\omega} t]\ \mathrm{Cos}[\boldsymbol{\omega} T]\ -\ \mathrm{Cos}[\boldsymbol{\omega} t]\ \mathrm{Sin}[\boldsymbol{\omega} T]$$

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

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

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

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

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

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

Manipulation of the first two equations gives

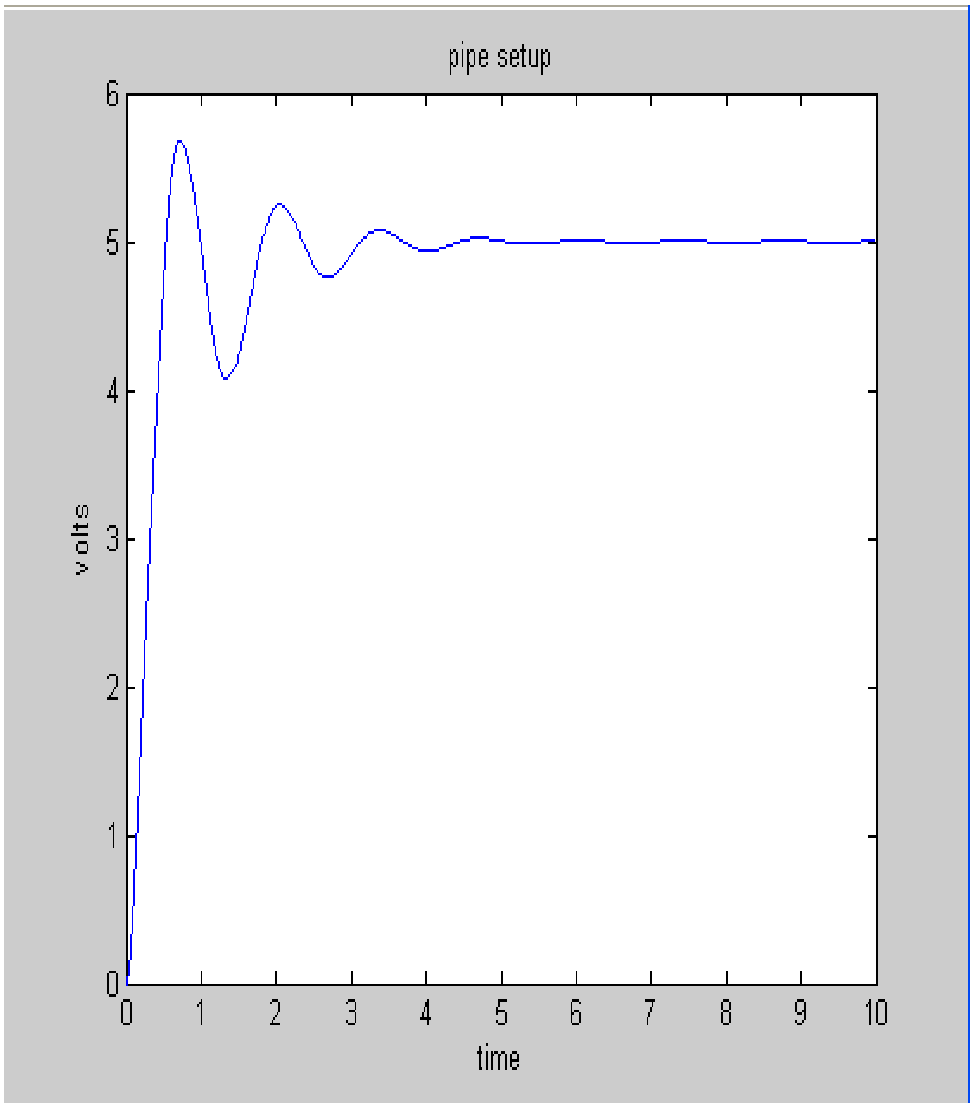
$$\mathbf{K_P} = [A X \boldsymbol{\omega}^2 - B Y] / [Z \operatorname{Cos}[\boldsymbol{\omega}T]]$$

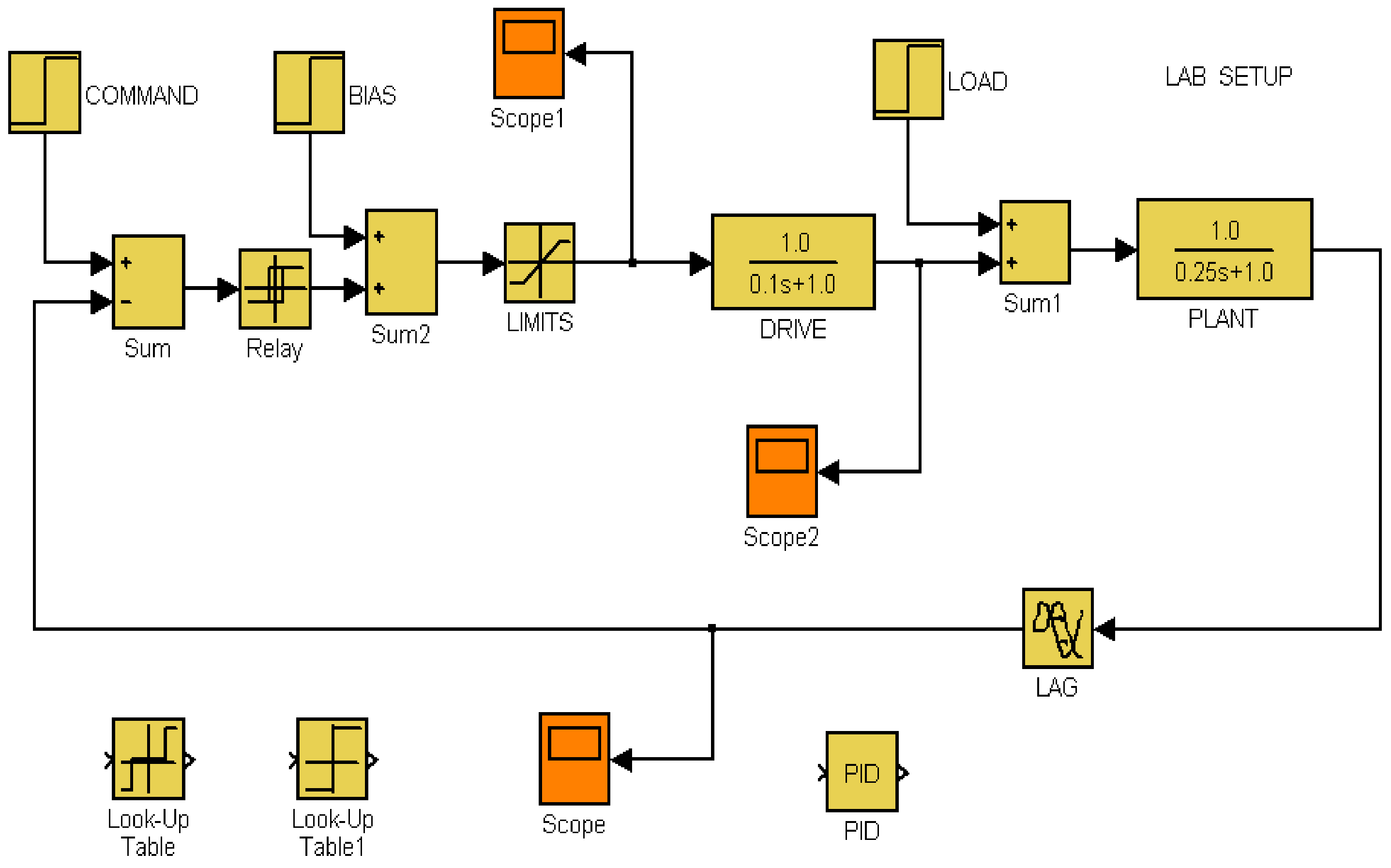
$$\mathbf{K_P} = [A Y \boldsymbol{\omega} + B X \boldsymbol{\omega}] / [Z \operatorname{Sin}[\boldsymbol{\omega}T]]$$

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

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.

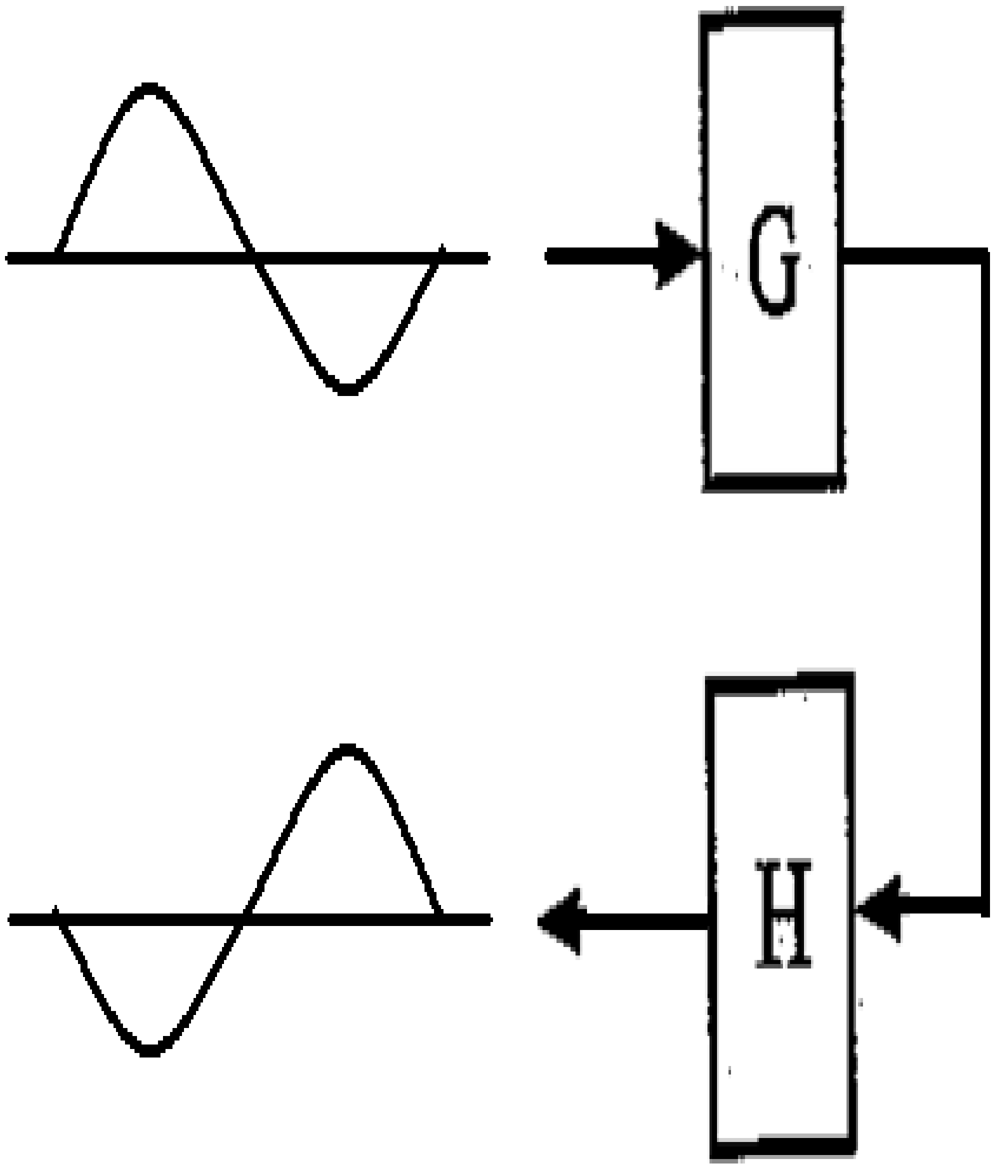
```
% PIPE FLOW TEMPERATURE CONTROL
  ROLD=0.0;HOLD=0.0;SENSOR=ROLD;
  TARGET=5.0;LOAD=0.0;DUMP=10.0;
  X=0.25;Y=1.0;A=0.1;B=1.0;Z=1.0;
  WRONG=TARGET-SENSOR;SUM=0.0;
  NIT=10000;MIT=500;TIME=0.0;
  GP=1.5;GI=0.0;GD=0.0;
  GP=0.9;GI=1.2;GD=0.17;
  DELT=0.001;
  for IT=1:NIT
  TIME=TIME+DELT;
  if(IT>MIT) ...
   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')
```

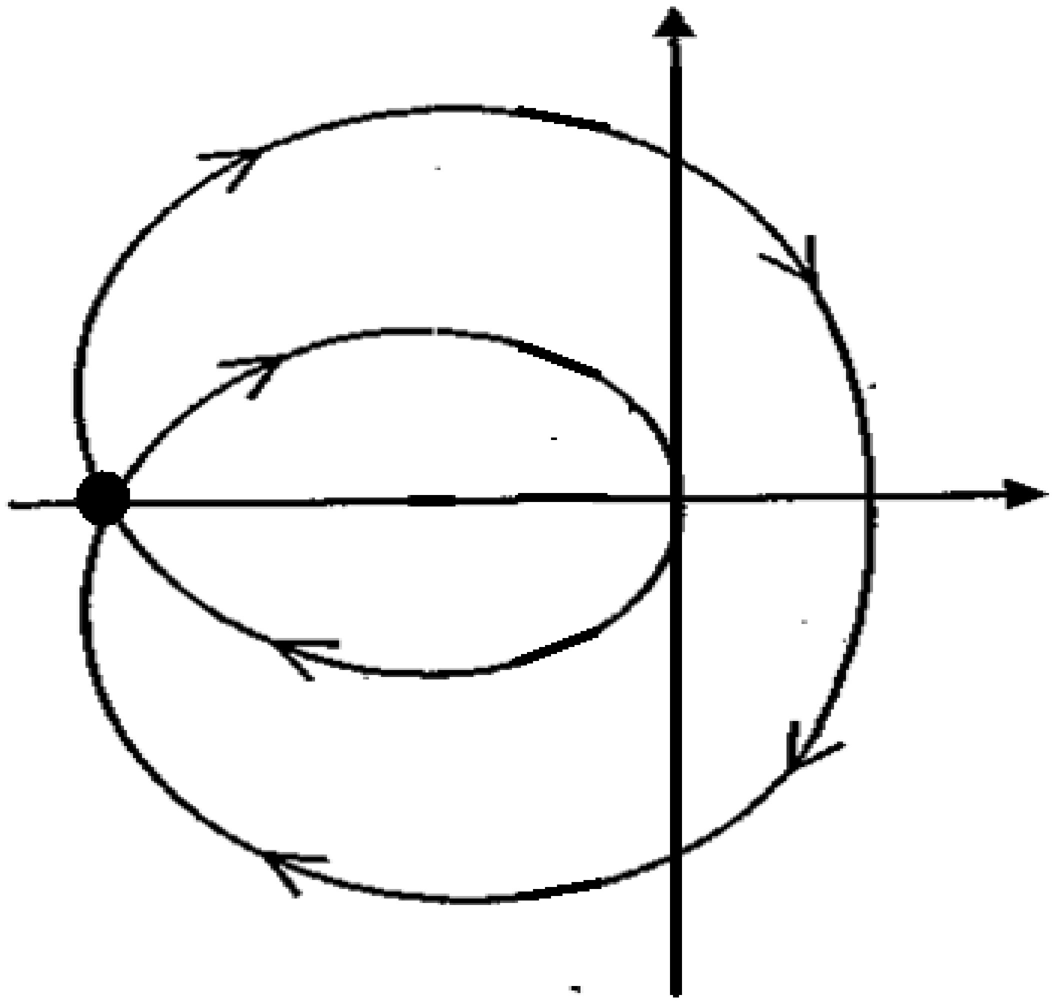

pipe setup
volts
time
0
1
2
3
4
5
6
7
8
9
10


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

GH NYQUIST PLOT

A GH plot is basically a polar open loop frequency response plot. Consider the case where only proportional control is being used. 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. The system would be borderline stable with gain **K**. 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.


G
H



GH PLOT

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:

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

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:

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

$$\Im\,[d^2f/dt^2] = S^2\ F(S) - S\ f(0) - df(0)/dt$$

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

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:

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

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

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

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

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

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:

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

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.

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

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

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

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

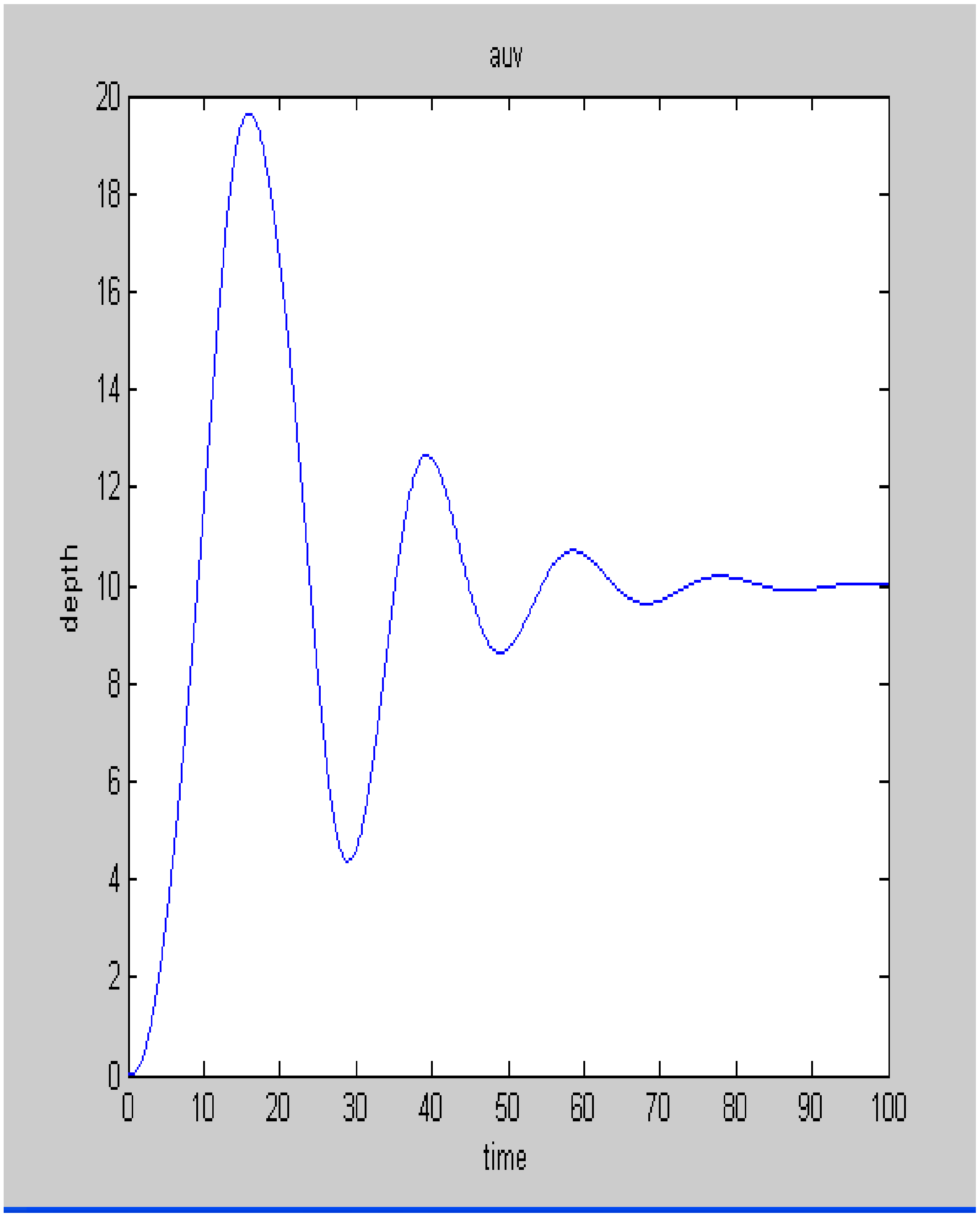
$$B_{NEW} = B_{OLD} + \Delta t * (Q_{OLD} - I\ B_{OLD}) / J$$

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

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

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:

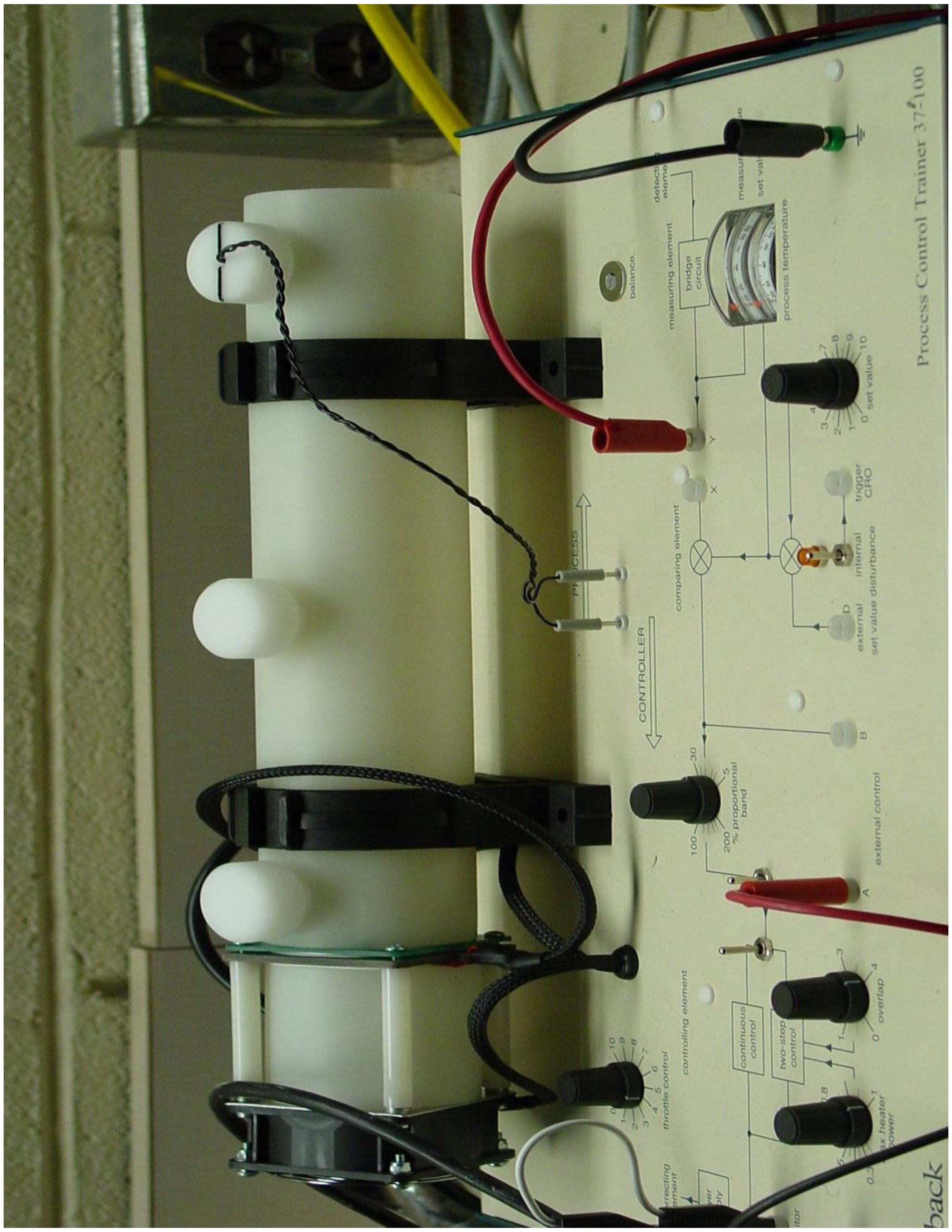
$$X\, dR/dt + Y\, R = H + D$$

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

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

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

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.


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

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

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

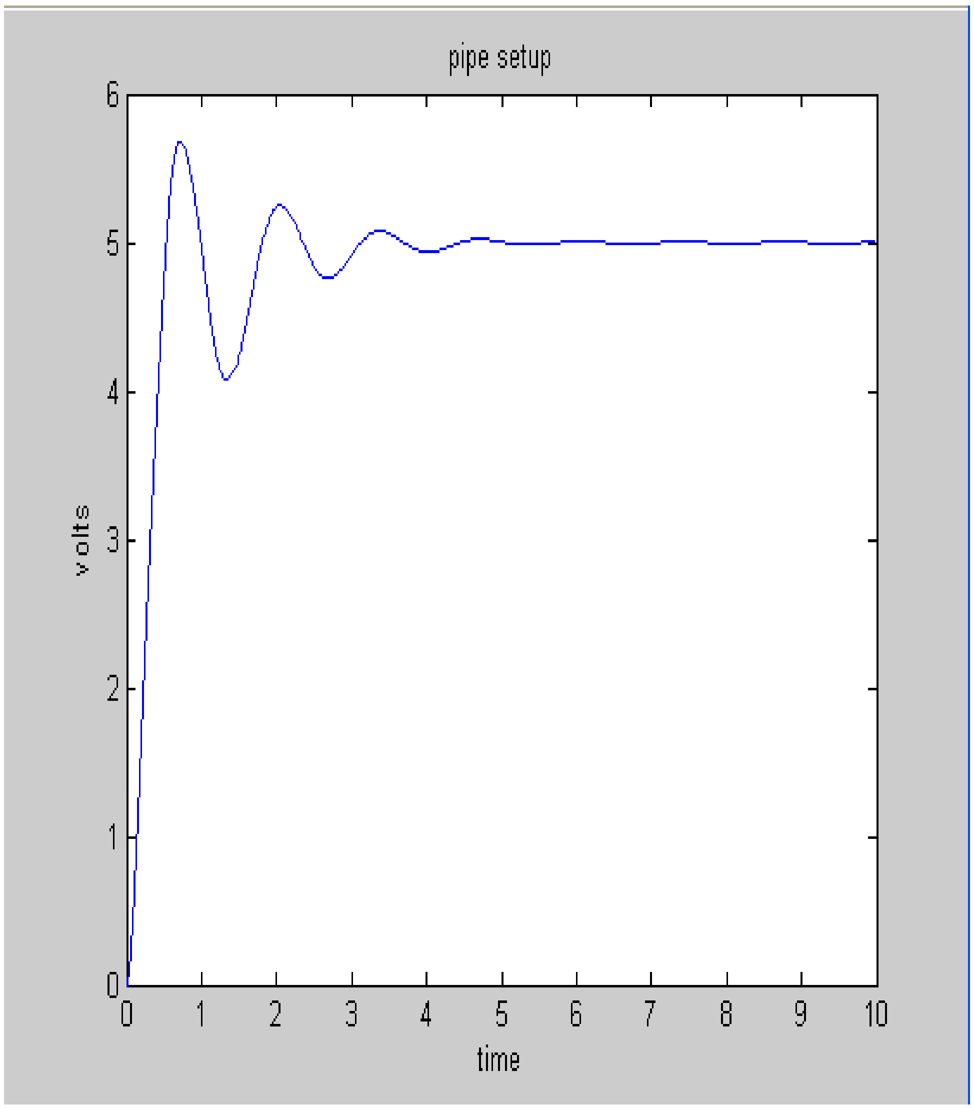
$$H_{NEW} = H_{OLD} + \Delta t * (Z Q_{OLD} - B H_{OLD}) / A$$

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

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

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
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.

AUTONOMOUS UNDERWATER VEHICLE

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

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

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

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

$$E = C - R$$

Laplace transformation gives

$$[\, M\, S^2 + N\, S\,]\, R = B + D$$

$$[\, J\, S + I\,]\, B = Q$$

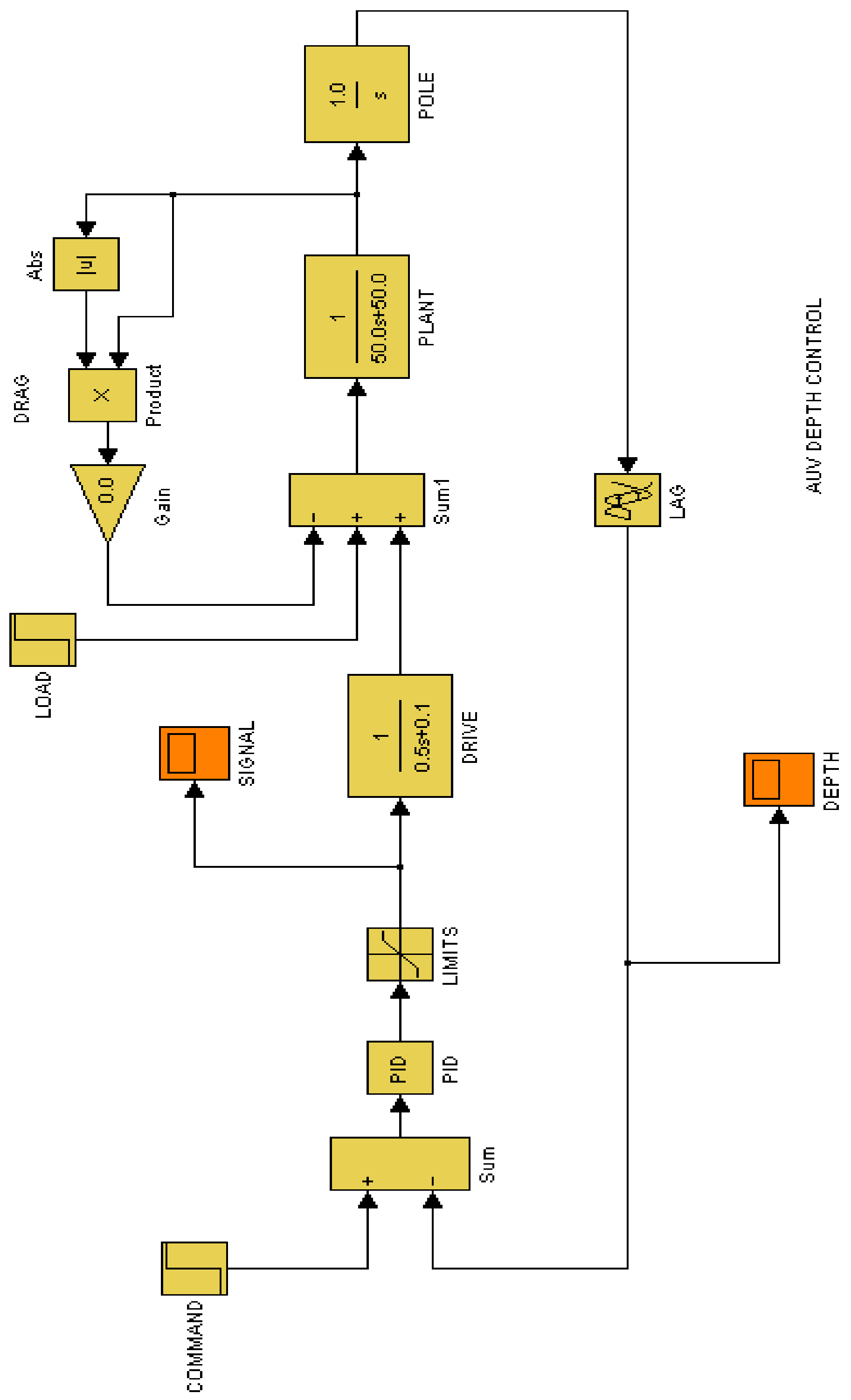
$$Q = [K_P + K_I/S + K_D\, S\,]\, E$$

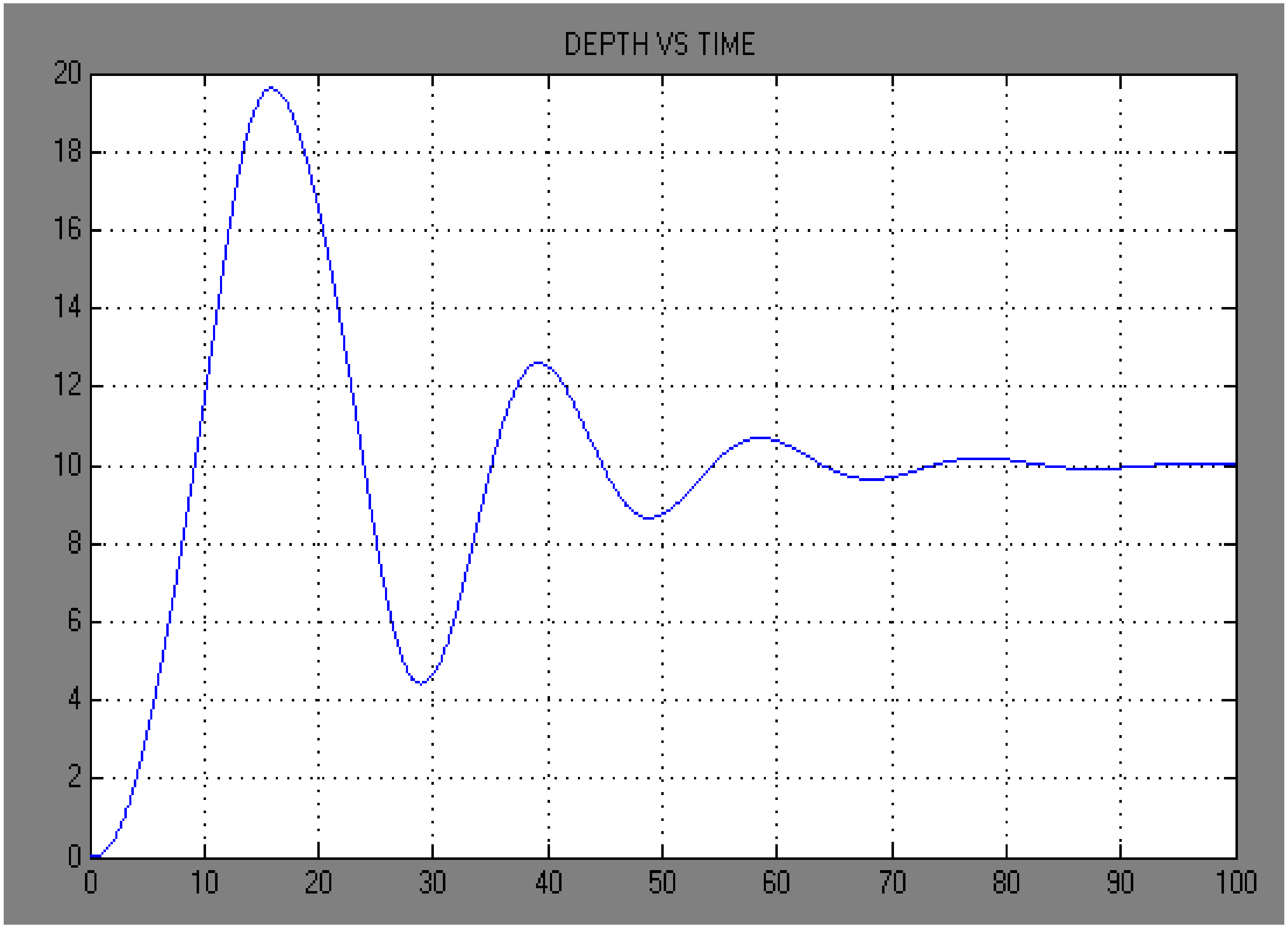
Transfer functions are

$$R \,/\, [\, B + D\,] = 1 \,/\, [\, M\, S^2 + N\, S\,]$$

$$B \,/\, Q = 1 \,/\, [\, J\, S + I\,]$$

$$Q \,/\, E = [K_P + K_I/S + K_D\, S\,]$$


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


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

PIPE FLOW SETUP

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

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

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

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

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

Laplace transformation gives

$$[\ X\ S + Y\ ]\ R = H + D$$

$$[\ A\ S + B\ ]\ H = Z\ Q$$

$$Q = [K_P + K_I/S + K_D\ S\ ]\ E$$

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

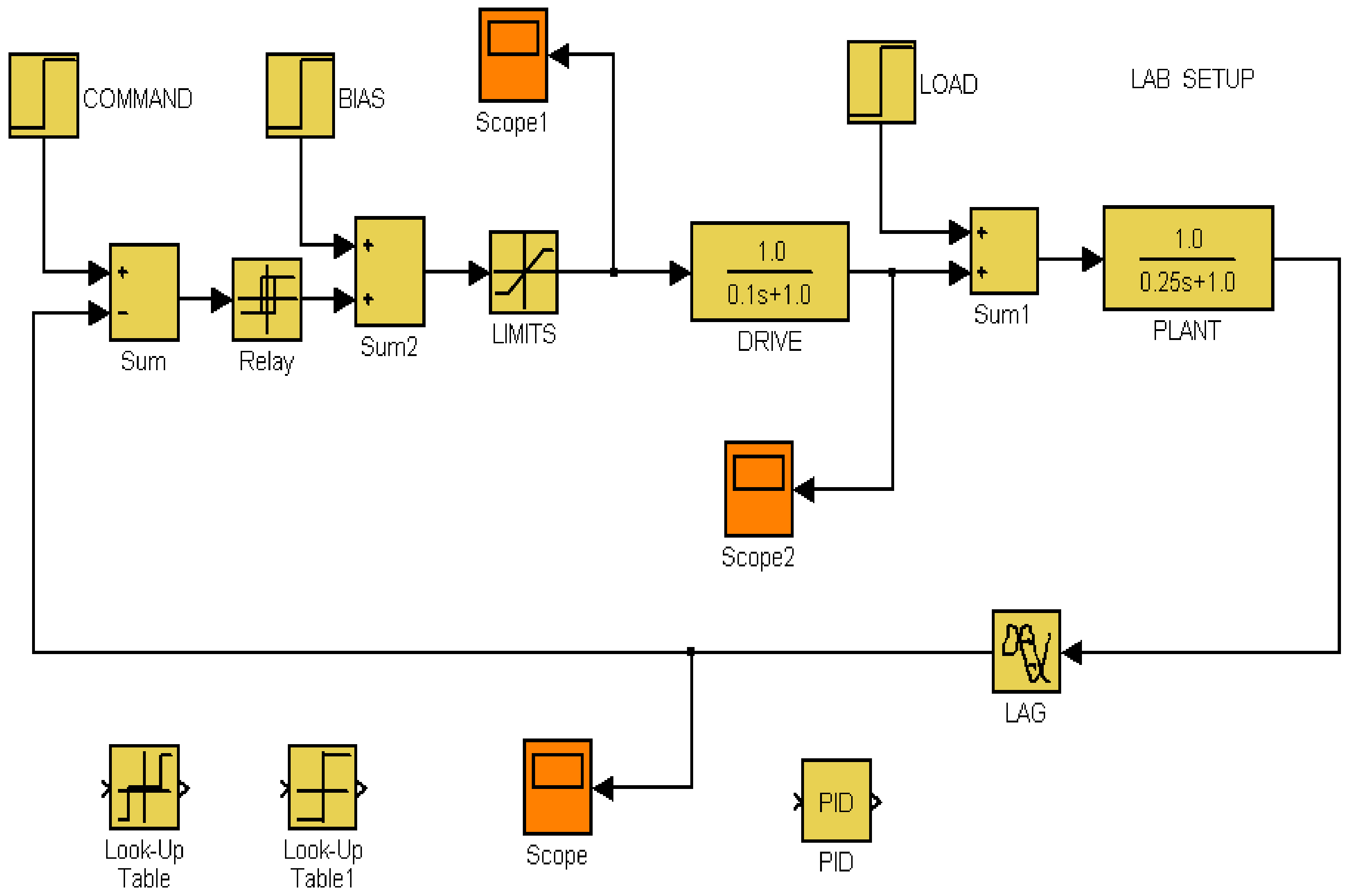
Transfer functions are

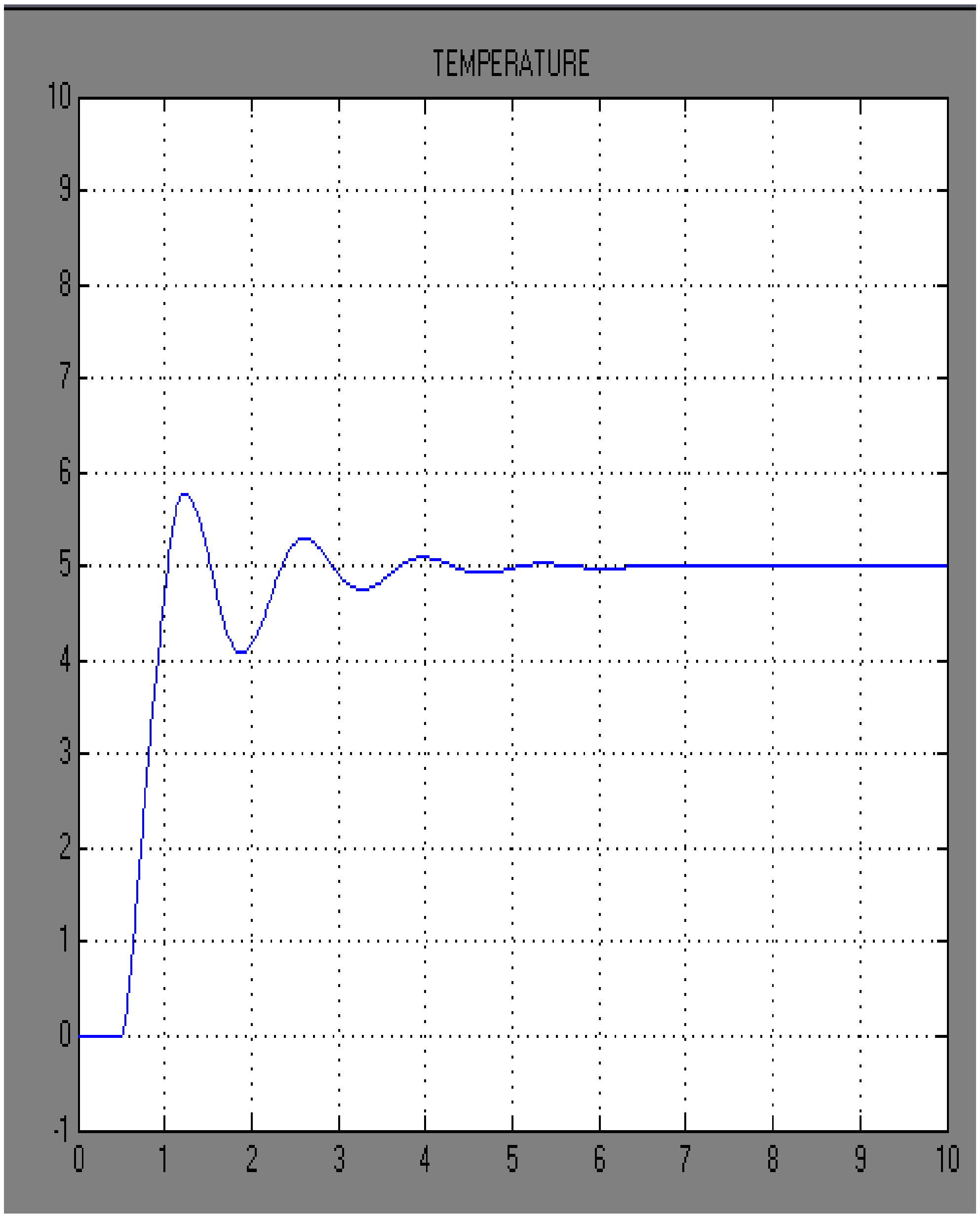
$$R\ /\ [\ H + D\ ] = 1\ /\ [\ X\ S + Y\ ]$$

$$H\ /\ Q = Z\ /\ [\ A\ S + B\ ]$$

$$Q\ /\ E = [K_P + K_I/S + K_D\ S\ ]$$

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


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


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

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:

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

its output is generally of the form:

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

With the same input:

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

the describing function gives out:

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

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

$$I_{DF} = E_o \sin\omega t \qquad O_{DF} = O_S \sin\omega t + O_C \cos\omega t \ .$$

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:

$$O_S = 2/T \int_0^T Q(t) \sin\omega t \, dt = 4Q_o/\pi$$

$$O_C = 2/T \int_0^T Q(t) \cos\omega t \, dt = 0$$

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

So the fundamental output is:

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

The input is

$$I_{DF} = E_o \sin\omega t$$

So the Describing Function is

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

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:

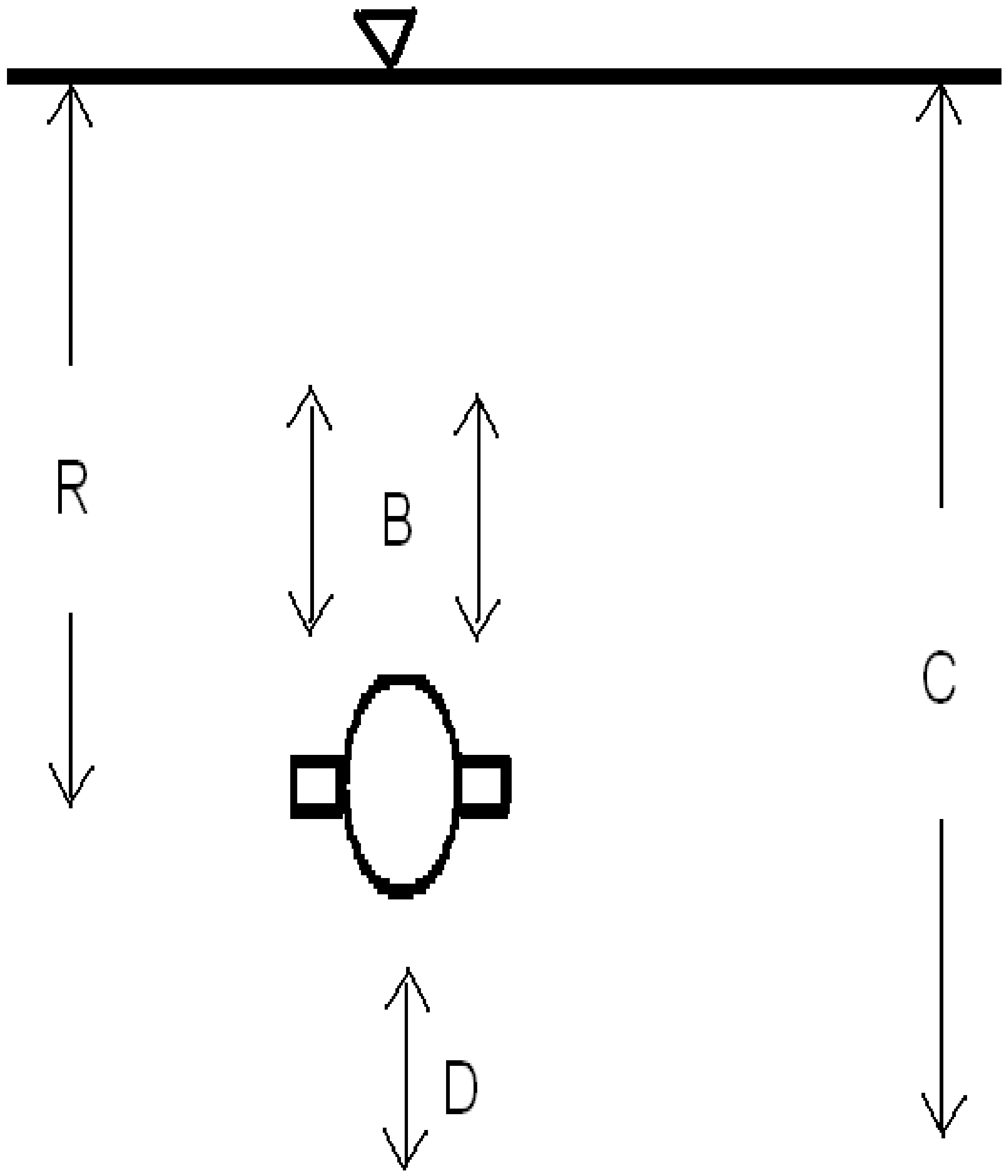
$$M\, d^2R/dt^2 = B + D - W$$

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

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

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

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:

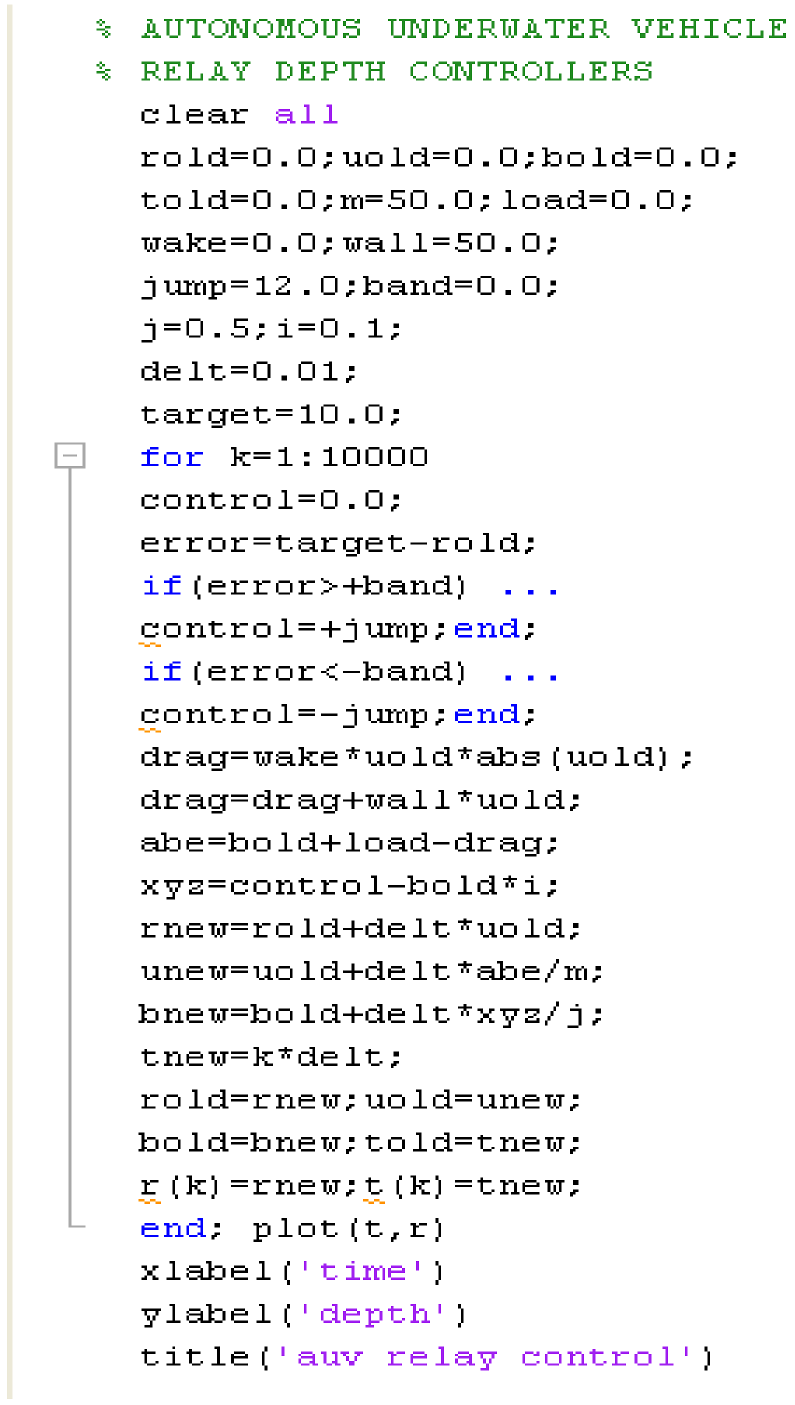
$$DF = [4\ Q_o] / [\pi E_o]$$

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

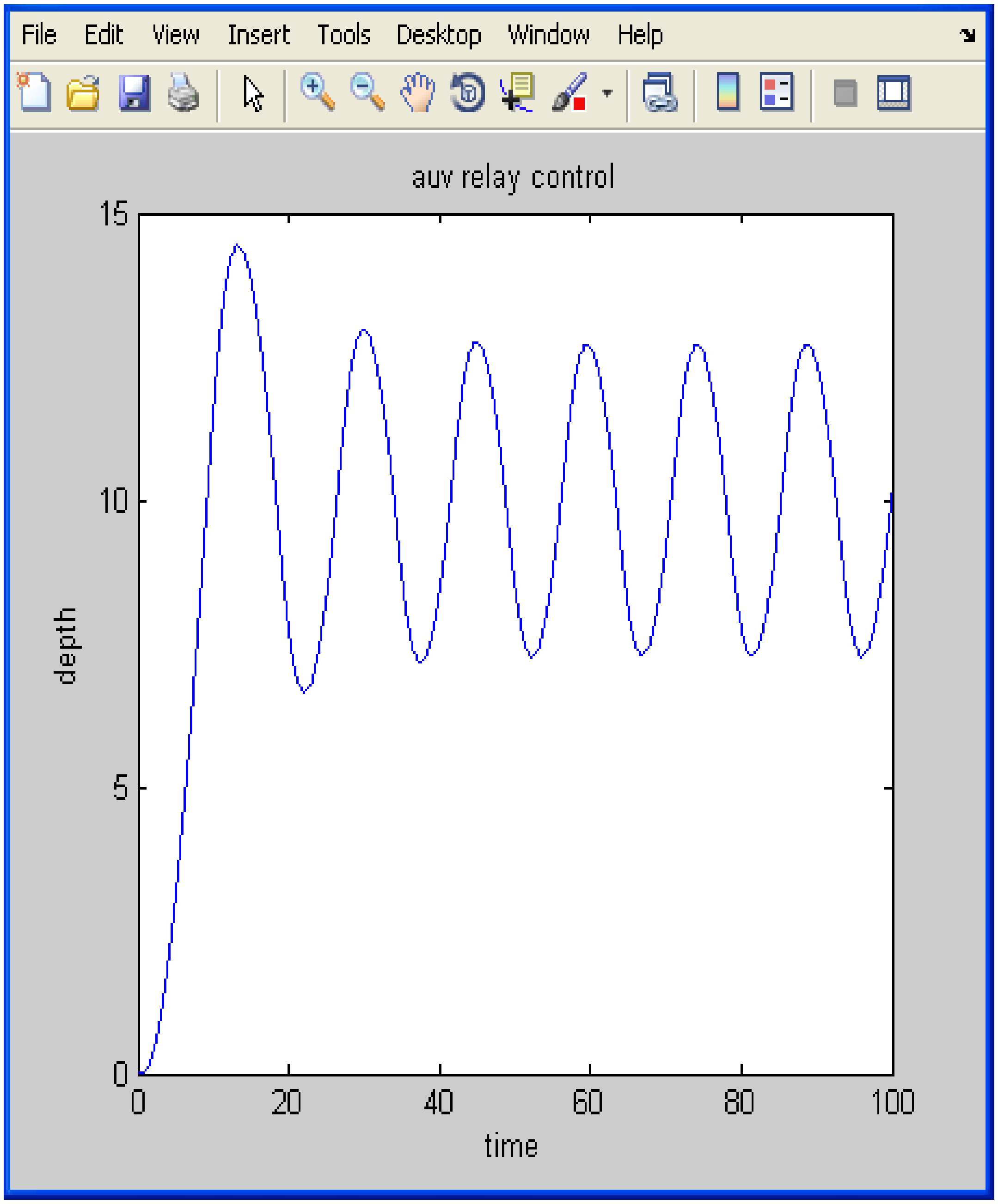
$$\mathbf{E_o} = [4\ Q_o] / [\pi\, \mathbf{DF}] = [4Q_o] / [\pi\, \mathbf{K_P}]$$

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
depth
time
0
5
10
15
0
20
40
60
80
100

PIPE FLOW SETUP

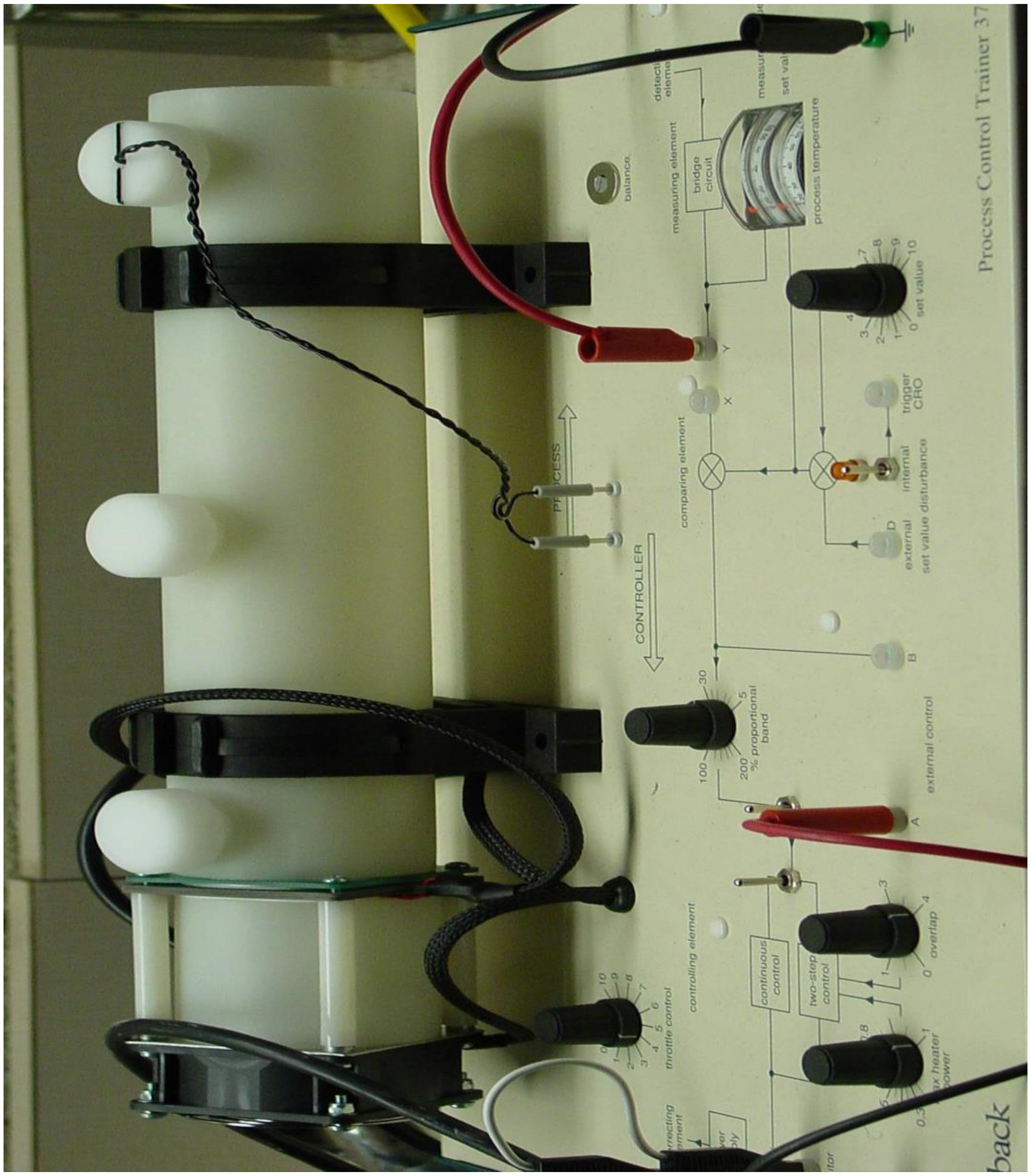
To illustrate nonlinear phenomena, we will consider the task of controlling the temperature of air flowing down a pipe. The setup is shown on the next page. Relay controllers resemble the proportional controller. For the proportional controller case, the governing equations for the setup are:

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

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

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

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.


Process Control Trainer 37
balance
measuring element
bridge circuit
process temperature
detecting element
measured value
set value
set value
trigger CRO
internal set value disturbance
external set value disturbance
comparing element
PROCESS
CONTROLLER
% proportional band
external control
A
B
D
X
Y
controlling element
throttle control
continuous control
two-step control
overlap
max heater power
correcting element
back

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 1.58.

The describing function for an ideal relay controller is:

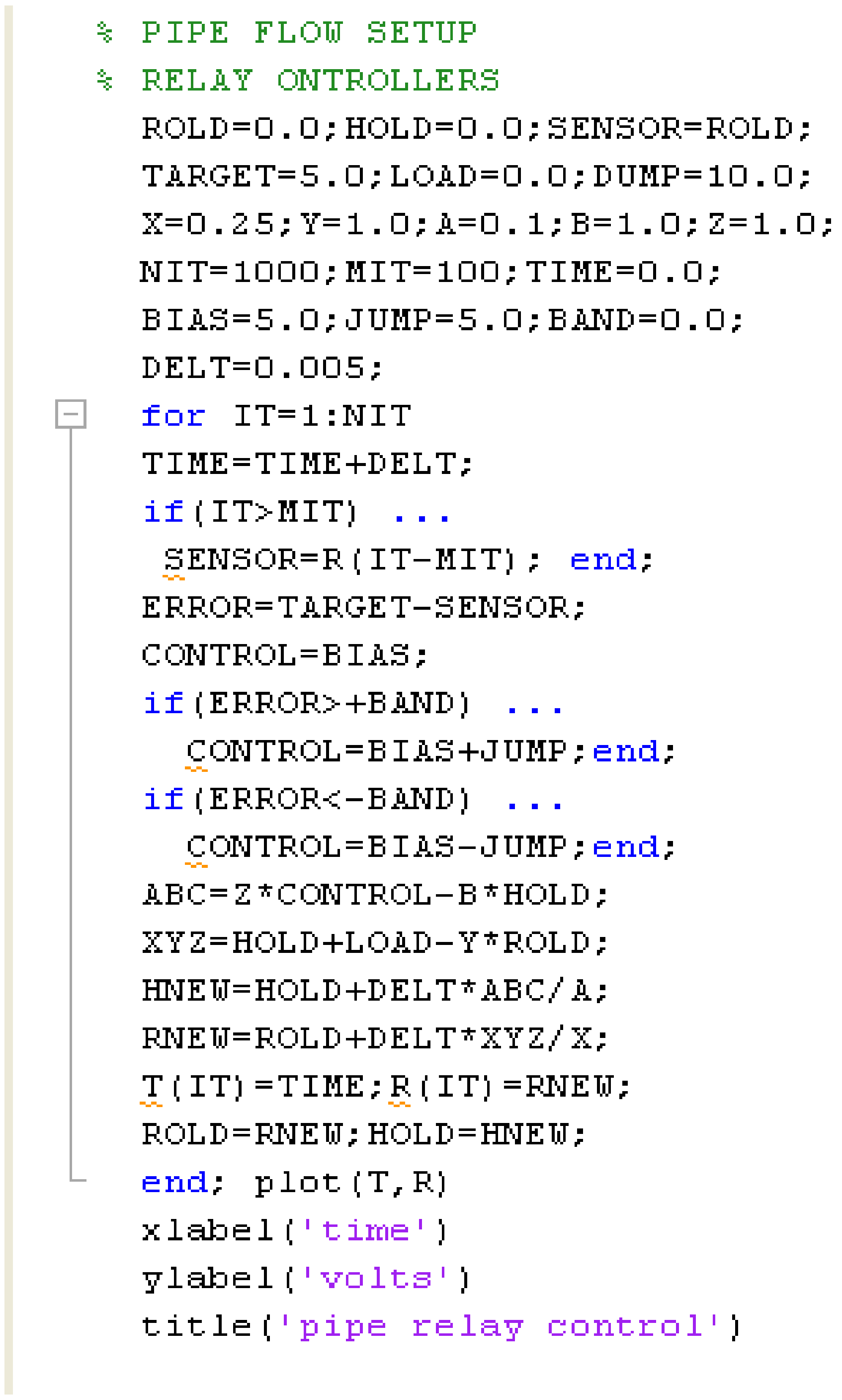
$$DF = [4\ Q_o] / [\pi E_o]$$

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

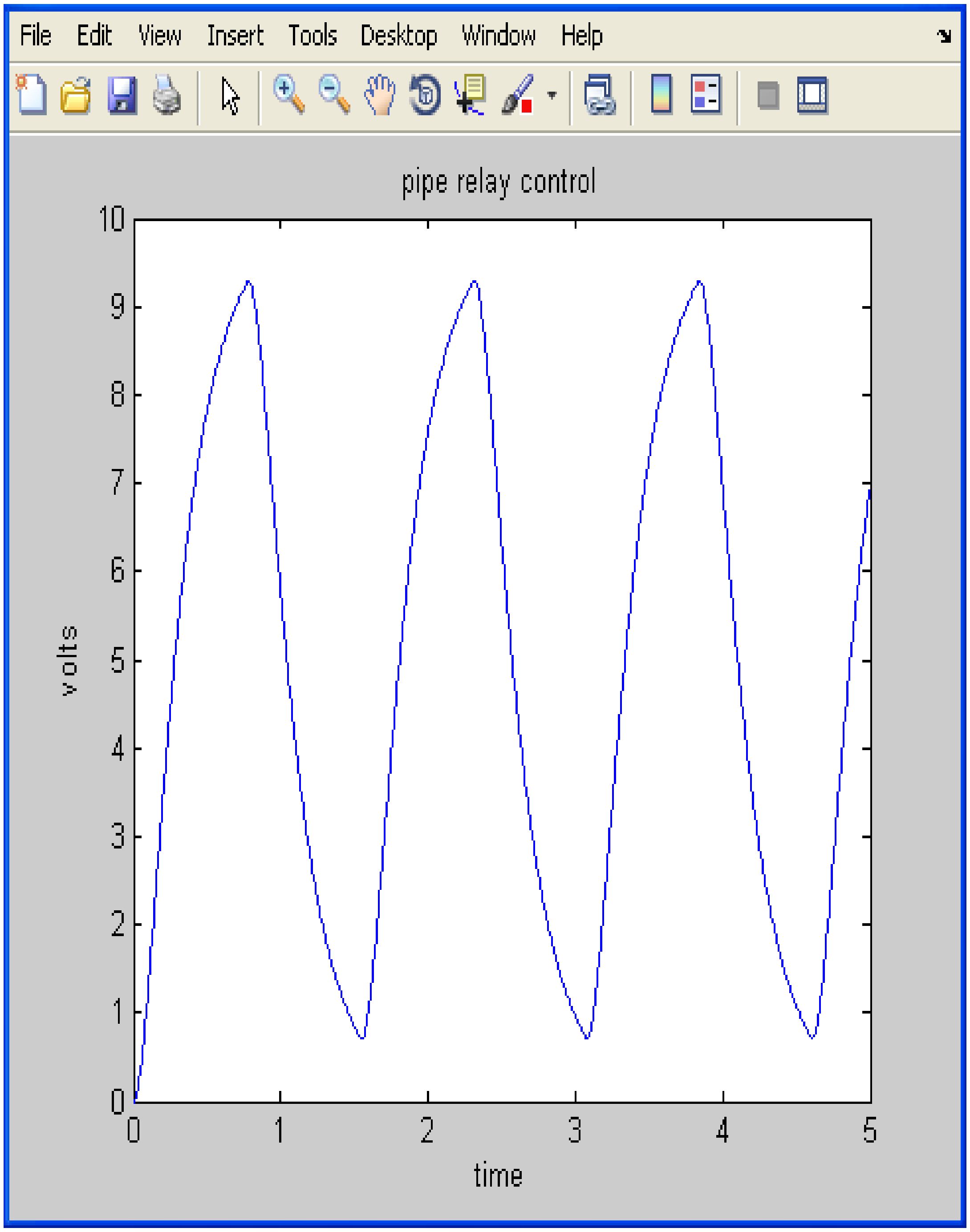
$$\mathbf{E_o} = [4\ Q_o] / [\pi\,\mathbf{DF}] = [4Q_o] / [\pi\,\mathbf{K_P}]$$

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
0
1
2
3
4
5