Course Subject : Digital Control System

Course Code : EKK234

Departement : Electrical Engineering

Semester : V

Week : 3

Time alocation : 100 minutes

Competence : Students understand the basic terms and concepts of digital control modelling, able to analyze the performance and design simple digital control systems using Matlab.

Sub Competence : Students understand concept modelling of digital control system.

Competency Achievement Indicators :

1. The ability to model of digital control system plant.
2. The ability to simulate performance of digital control system.
3. LEARNING OBJECTIVES :

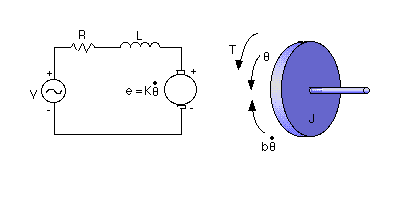
a. Student know basic concept modelling of digital control system

b. Student can analysis the transfer function of digital control plant and response system of digital control system using MATLAB

1. Lecture Material :

## Physical setup and system equations

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in the following figure:



For this example, we will assume the following values for the physical parameters. These values were derived by experiment from an actual motor in Carnegie Mellon's undergraduate controls lab.

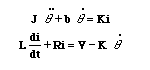
\* moment of inertia of the rotor (J) = 0.01 kg.m^2/s^2  
\* damping ratio of the mechanical system (b) = 0.1 Nms  
\* electromotive force constant (K=Ke=Kt) = 0.01 Nm/Amp  
\* electric resistance (R) = 1 ohm   
\* electric inductance (L) = 0.5 H  
\* input (V): Source Voltage  
\* output (theta): position of shaft  
\* The rotor and shaft are assumed to be rigid

The motor torque, **T**, is related to the armature current, **i**, by a constant factor **Kt**. The back emf, **e**, is related to the rotational velocity by the following equations:

Mfo1

In SI units (which we will use), **Kt** (armature constant) is equal to **Ke** (motor constant).

From the figure above we can write the following equations based on Newton's law combined with Kirchhoff's law:



### 1. Transfer Function

Using Laplace Transforms, the above modeling equations can be expressed in terms of s.

Mfo3

Mfo4

By eliminating I(s) we can get the following open-loop transfer function, where the rotational speed is the output and the voltage is the input.

Mfo9

We can represent the above transfer function into Matlab by defining the numerator and denominator matrices as follows:

Mfo6  
Mfo7

Create a new [m-file](http://www.engin.umich.edu/group/ctm/extras/mfile.html) and enter the following commands:

J=0.01;

b=0.1;

K=0.01;

R=1;

L=0.5;

num=K;

den=[(J\*L) ((J\*R)+(L\*b)) ((b\*R)+K^2)];

**Continuous to Discrete Conversion**

The first step in designing a discrete control system is to convert the continuous transfer function to a discrete transfer function. Matlab command [c2dm](http://www.engin.umich.edu/group/ctm/digital/digital.html#conv) will do this for you. The c2dm command requires the following four arguments: the numerator polynomial (num), the denominator polynomial (den), the sampling time (Ts) and the type of hold circuit. In this example, the hold we will use is the [zero-order hold](http://www.engin.umich.edu/group/ctm/digital/digital.html#zoh) ('zoh').

From the design requirement, let the sampling time, **Ts** equal to 0.12 seconds, which is 1/10 the time constant of a system with a settling time of 2 seconds. Let's create a new [m-file](http://www.engin.umich.edu/group/ctm/extras/mfile.html) and enter the following commands:

R=1;

L=0.5;

Kt=0.01;

J=0.01;

b=0.1;

num = Kt;

den = [(J\*L) (J\*R)+(L\*b) (R\*b)+(Kt^2)];

Ts = 0.12;

[numz,denz] = c2dm(num,den,Ts,'zoh')

Running this m-file should return the following:

*numz =*

*0 0.0092 0.0057*

*denz =*

*1.0000 -1.0877 0.2369*

From these matrices, the discrete transfer function can be written as:

discrete_trans

First, we would like to see what the closed-loop response of the system looks like without any control. If you see the *numz* matrices shown above, it has one extra zero in the front, we have to get rid of it before closing the loop with the Matlab cloop command. Add the following code into the end of your m-file:

numz = [numz(2) numz(3)];

[numz\_cl,denz\_cl] = cloop(numz,denz);

After you have done this, let's see how the closed-loop step response looks like. The dstep command will generate the vector of discrete output signals and stairs command will connect these signals.

Discrete step response

To plot a step response of a discrete system, we will use two separate Matlab functions, dstep and stairs. The dstep will be used to obtain N number of output sample points, where N is supplied by an user. The stairs plots a stairstep graph of supplied vectors, namely the time vector [t] and the amplitude vector [x].

Recall that the continuous-time transfer function for a PID controller is:

Con_PID

Equivalently, the c2dm command in Matlab will help you to convert the continuous-time PID compensator to discrete-time PID compensator by using the "tustin" method in this case. The "tustin" method will use bilinear approximation to convert to discrete time of the derivative. According to the [PID Design Method for the DC Motor](http://www.engin.umich.edu/group/ctm/examples/motor/PID2.html) page, **Kp** = 100, **Ki** = 200 and **Kd** = 10 are satisfied the design requirement. We will use all of these gains in this example. Now add the following Matlab commands to your previous m-file and rerun it in Matlab window.

% Discrete PID controller with bilinear approximation

Kp = 100;

Ki = 200;

Kd = 10;

[dencz,numcz]=c2dm([1 0],[Kd Kp Ki],Ts,'tustin');

Note that the numerator and denominator in c2dm were reversed above. The reason is that the PID transfer function is not proper. Matlab will not allow this. By switching the numerator and denominator the c2dm command can be fooled into giving the right answer. Let's see if the performance of the closed-loop response with the PID compensator satisfies the design requirements. Now add the following code to the end of your m-file and rerun it. You should get the following close-loop stairstep response.

numaz = conv(numz,numcz);

denaz = conv(denz,dencz);

[numaz\_cl,denaz\_cl] = cloop(numaz,denaz);

[x2] = dstep(numaz\_cl,denaz\_cl,101);

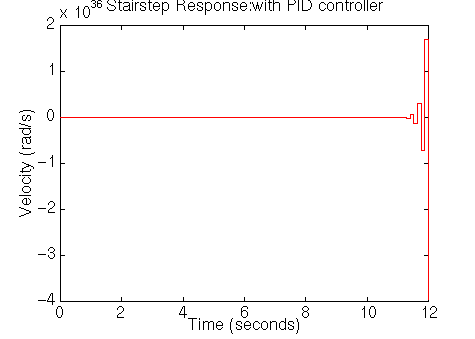
t=0:0.12:12;

stairs(t,x2)

xlabel('Time (seconds)')

ylabel('Velocity (rad/s)')

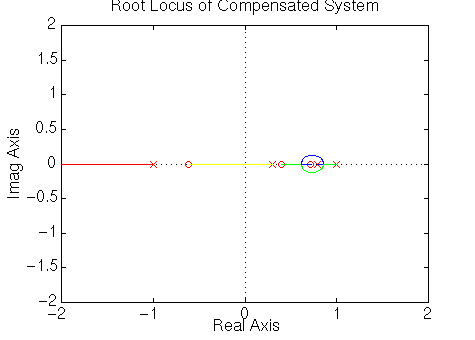
title('Stairstep Response:with PID controller')



As you can see from the above plot, the closed-loop response of the system is unstable. Therefore there must be something wrong with compensated system. So we should take a look at root locus of the compensated system. Let's add the following Matlab command into the end of your m-file and rerun it.

rlocus(numaz,denaz)

title('Root Locus of Compensated System')



From this root-locus plot, we see that the denominator of the PID controller has a pole at -1 in the z-plane. We know that if a pole of a system is outside the unit circle, the system will be unstable. This compensated system will always be unstable for any positive gain because there are an even number of poles and zeroes to the right of the pole at -1. Therefore that pole will always move to the left and outside the unit circle. The pole at -1 comes from the compensator, and we can change its location by changing the compensator design. We choose it to cancel the zero at -0.62. This will make the system stable for at least some gains. Furthermore we can choose an appropriate gain from the root locus plot to satisfy the design requirements using rlocfind.

Enter the following Matlab code to your m-file.

dencz = conv([1 -1],[1.6 1])

numaz = conv(numz,numcz);

denaz = conv(denz,dencz);

rlocus(numaz,denaz)

title('Root Locus of Compensated System');

[K,poles] = rlocfind(numaz,denaz)

[numaz\_cl,denaz\_cl] = cloop(K\*numaz,denaz);

[x3] = dstep(numaz\_cl,denaz\_cl,101);

t=0:0.12:12;

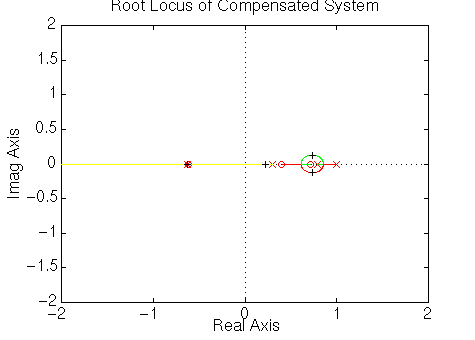
stairs(t,x3)

xlabel('Time (seconds)')

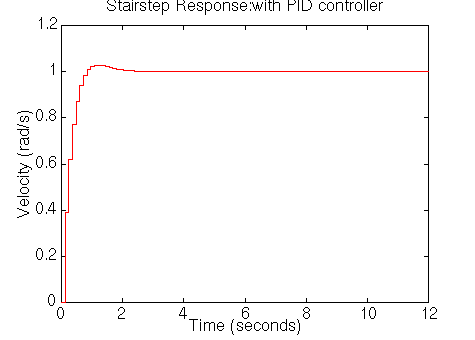
ylabel('Velocity (rad/s)')

title('Stairstep Response:with PID controller')

The new dencz will have a pole at -0.625 instead of -1, which almost cancels the zero of uncompensated system. In the Matlab window, you should see the command asking you to select the point on the root-locus plot. You should click on the plot as the following:



Then Matlab will return the appropriate gain and the corresponding compensated poles, and it will plot the closed-loop compensated response as follows.



The plot shows that the settling time is less than 2 seconds and the percent overshoot is around 3%. In addition, the steady state error is zero. Also, the gain, K, from root locus is 0.2425 which is reasonable. Therefore this response satisfies all of the design requirements.

1. Learning Method

Learning method that use in this course is discussion and simulation using MATLAB.

1. Reference
2. Digital Control Of Dynamic System, Third Edition, 2006, Gene F Franklin, Addison Wesley.
3. Digital Control Engineering Analysis and Design, 2009, M Sam Fadali,Elsevier.
4. Evaluation

From the [Modeling: a DC Motor](http://www.engin.umich.edu/group/ctm/examples/motor/motor.html), the open-loop transfer function for DC motor's speed was derived as:

1. OPL_trans

Where:

* electrical resistance (R) = 1 ohm
* electrical inductance (L) = 0.5 H
* electromotive force constant (Ke=Kt) = 0.01 Nm/Amp
* moment of inertia of the rotor (J) = 0.01 kg\*m^2/s^2
* damping ratio of the mechanical system (b) = 0.1 Nms
* input (V): Source Voltage
* output (theta dot): Rotating speed
* The rotor and shaft are assumed to be rigid

Using PID controller with parameter as follow : **Kp** = 100, **Ki** = 200 and **Kd** = 10, speed of DC motor can be controlled.

Try plot the discrete transfer function of the plant (motor DC ) and transfer function of discrete PID also plot the step response of that DC motor speed control system and state the stability of that system.