Sample Solutions to Homework Assignement 3

National University of Singapore

Department of Electrical and Computer Engineering

EE3304 Digital Control Systems

Home Assignment III

(Semester II, 2002-2003)

Name: Zhang Xiaoqiang

Matric No.: U00-5668A-11

Email: eng00176@nus.edu.sg

Lecturer: A/P. Ben M. Chen

The plant for this assignment can be expressed in *s* domain as the following

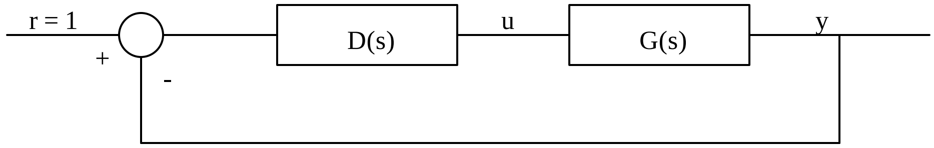

Fig.1: System in continuous time domain

This can be translated into z domain by using state space approach

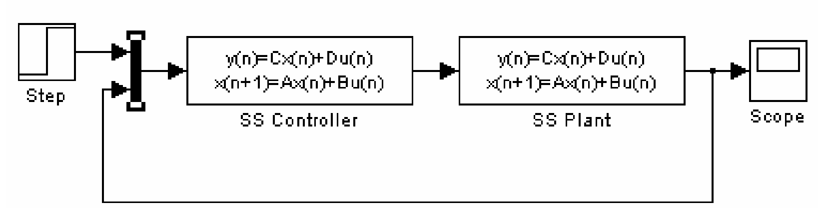

Fig. 2: System in discrete time domain

The design specifications for this assignment are given as

$$\begin{cases} M_p\% < 5\% \\ t_s < 2ms \end{cases}$$

(where the input is a step reference of 1 micrometer)

Generally speaking, to design a controller by using estimator, the procedure is quite systematic. In sum, six steps are required.

1. Step One

Determine the desired closed-loop pole locations from the given specifications.

The settling time is less than 2 milliseconds, and the maximum overshoot is less than 5%. Read from the relationship diagram of Mp% vs. ?, we get

$$z = 0.7$$

Hence,

$$t_s \cong \frac{4.6}{zw_n}$$

$$\Rightarrow w_n \cong \frac{4.6}{zt_s} = \frac{4.6}{0.7 \times 1.5 \times 10^{-3}} = 4381$$

So the poles in *s* domain are located at

$$P_{S(1,2)} = -3066.7 \pm j2399.5$$

The corresponding poles in the z domain are

$$P_{Z(1,2)} = 0.7148 \pm j0.1749$$

2. Step Two

Find the state feedback gain ***F*** such that the eigenvalues of ***A-BF*** coincides with the desired pole locations obtained in the first step.

Discretize the plant $G(s) = \dfrac{6\times10^7}{s^2}$ by zero-order-hold, and we can get

$$G(z) = \frac{0.3z + 0.3}{z^2 - 2z + 1}$$

Convert this transfer function into state space representation,

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases}$$

where,

$$A = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 0.3 & 0.3 \end{bmatrix}$$
$$D = [0]$$

Define the desired poles in z domain as a column vector

$$PDesired = \begin{bmatrix} 0.7148 + j0.1749 \\ 0.7148 - j0.1749 \end{bmatrix}$$

In MatLab, there is one function called ***acker()***, which is short for Pole Placement Gain Selection Using Ackermann's Formula, can calculate the feedback gain matrix ***K*** such that the single input system $x(k+1) = Ax(k) + Bu(k)$ with a feedback law of $u(k) = -Kx(k)$ has closed loop poles at the values specified in vector ***P***. This is same as $P = eig(A - B*K)$.

Hence,

$$F = acker(A, B, PDesired)$$

This gives,

$$F = \begin{bmatrix} 0.5704 & -0.4585 \end{bmatrix}$$

3. Step Three

Find the constant gain ***J*** such that $[(C-DF)(I-A+BF)^{-1}B+D]J=1$

This can be translated into an equation as

$$J = inv((C-D*F)*inv(eye(2)-A+B*F)*B+D)$$

Hence,

$$J = 0.1865$$

4. Step Four

Find the estimator gain ***K*** such that the eigenvalues of ***A-KC*** are in pre-specified locations (or place all at 0 otherwise to yield a deadbeat estimator).

Use the same method as what we have done in step two. Define the pre-specified eigenvalues as a column vector,

$$PEstimator = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

So,

$$K = acker(A', B', PEstimator)'$$

This gives,

$$K = \begin{bmatrix} 4.1667 \\ 2.5000 \end{bmatrix}$$

5. Step Five

Compute the parameters of the controller: $\hat{A} = A - KC - BF + KDF$,

$\hat{B} = [(B-KD)J \quad K]$, $\hat{C} = -F$, $\hat{D} = [J \quad 0]$.

These can be easily calculated by using MatLab, we can get

$$\hat{A} = \begin{bmatrix} 0.1796 & -1.7915 \\ 0.2500 & -0.7500 \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} 0.1865 & 4.1667 \\ 0 & 2.5000 \end{bmatrix}$$

$$\hat{C} = \begin{bmatrix} -0.5704 & 0.4585 \end{bmatrix}, \quad \hat{D} = \begin{bmatrix} 0.1865 & 0 \end{bmatrix}$$

6. Step Six

Construct the controller in state space form,

$$\begin{cases} \hat{x}(k+1) = \hat{A}\hat{x}(k) + \hat{B}\begin{bmatrix} r(k) \\ y(k) \end{bmatrix} \\ \hat{y}(k) = \hat{C}\hat{x}(k) + \hat{D}\begin{bmatrix} r(k) \\ y(k) \end{bmatrix} \end{cases}$$

Verification in discrete-time domain and continuous-time domain

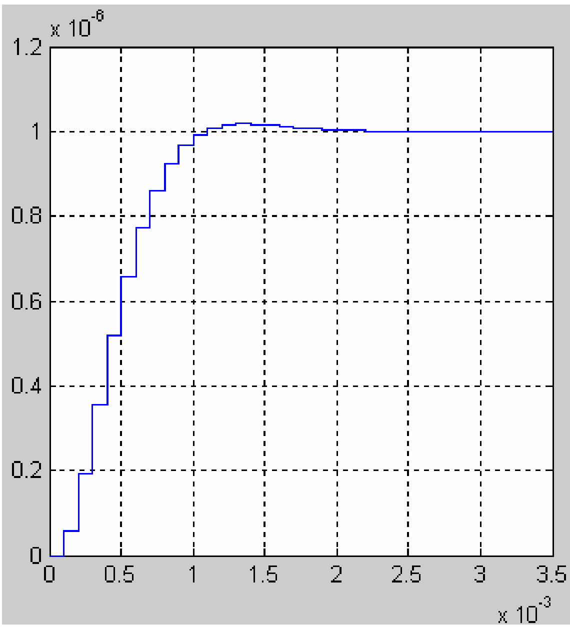

Fig. 3: Step response in discrete-time domain

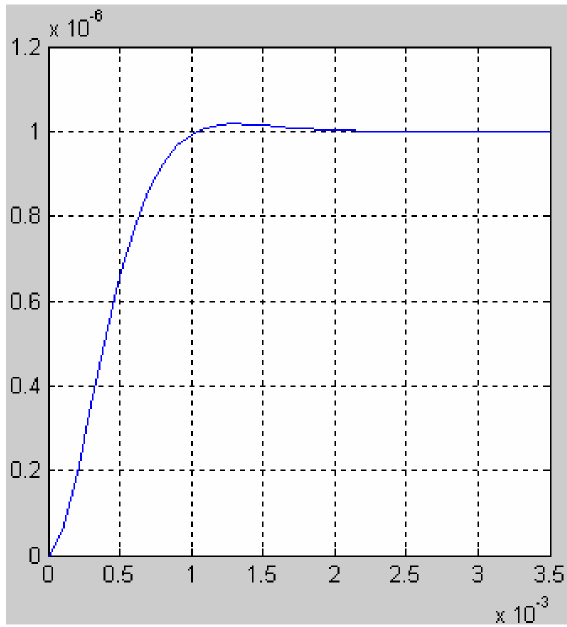

Fig. 4: Step response in continuous-time domain

From the above simulation with MatLab,

$$\begin{cases} M_p\% \cong 4\% < 5\% \\ t_s = 1.7ms < 2ms \end{cases}$$

Hence, the controller can meet the design specification.

Comparisons among the three linear controller design methods

	Advantages	**Disadvantages**
PID Controller	• Simple • Cheap • The PID controller provides quick acting corrective control of most process variables. Adding integral control to a proportional controller will eliminate the steady state error, but will increase overshoot and settling time. But by adding derivative control, the overshoot and settling time can be reduced.	• Slow • PID controller is not used for highly noisy control variables like flow control, because the derivative response will amplify the random fluctuations in the system.
PID compensator	The same as PID controller (They are the same.)	• The same as PID controller • Require more computation
SS approach	• Systematic • Accurate	• More expensive to construct • Required powerful PC and well designed software package to do the heavy computation

Appendix : M-File

```
%Home assignment 3 for EE3304
clear all;
close all;
%Define the plant in continuous time domain
CNum = [6E7];
CDen = [1 0 0];
%Define system variables
STime = 1.5E-3;
SpFreq = 1E4;
DRatio = 0.7;
NFreq = 4.6/(DRatio*STime);
CTfPole1 = -DRatio*NFreq + NFreq*(1-DRatio)^(1/2)*i;
CTfPole2 = -DRatio*NFreq - NFreq*(1-DRatio)^(1/2)*i;
DTfPole1 = exp((1/SpFreq)*CTfPole1);
DTfPole2 = exp((1/SpFreq)*CTfPole2);
%Define the Plane in State Space
C_G = tf(CNum, CDen);
D_G = c2d(C_G, 1/SpFreq, 'zoh' );
[DNum, DDen] = tfdata(D_G, 'v');
[A, B, C, D] = tf2ss(DNum, DDen);
SS_G = ss(A, B, C, D);
%Conpute feedback gain (F) and constant gain (J)
PDesired = [DTfPole1; DTfPole2];
F = acker(A, B, PDesired );
J = inv((C-D*F)*inv(eye(2)-A+B*F)*B+D);
%Compute estimator gain (K)
PEstimator = [0; 0];
K = acker(A', C', PEstimator)';
%Computer State Sapce controller parameters
SS_A = A - K*C-B*F;
SS_B = [(B-K*D)*J K];
SS_C = -F;
SS_D = [J, 0];
%Construst SS controller
SS_Cont = ss(SS_A, SS_B, SS_C, SS_D);
```