SIGNALS AND SYSTEMS I

Computer Assignment 7

Implementing State Systems

Linear time-invariant (LTI) systems can be described by the state space equations

$$\boldsymbol{q}(n+1) = \boldsymbol{A}\boldsymbol{q}(n) + \boldsymbol{B}\boldsymbol{x}(n)$$

$$\boldsymbol{y}(n) = \boldsymbol{C}\boldsymbol{q}(n) + \boldsymbol{D}\boldsymbol{x}(n)$$

which are a set of first-order difference equations. State systems can also be implemented using MATLAB's *for* loop constructs. For example, the first 10 output samples of the state equations

$$\begin{bmatrix} q_1(n+1) \\ q_2(n+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} q_1(n) \\ q_2(n) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} x(n)$$

$$y(n) = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} q_1(n) \\ q_2(n) \end{bmatrix} + dx(n)$$

where $x(n) = u(n)$ can be calculated as

```
x=ones(11,1);
A=[a11 a12; a21 a22];
B=[b1; b2];
C=[c1 c2];
D = d;
q = zeros(2,1)
for n=2:11
     y(n)=C*q + D*x(n);
     q = A*q + B*x(n);
end
y = y(2:11);
```

Outputs of the state equations can also be calculated using built-in MATLAB functions. Before using these functions, the state system must be defined. MATLAB's built-in function, **ss**, can be used to define a state space system. For example, the discrete state system

$$\boldsymbol{q}(n+1) = \boldsymbol{A}\boldsymbol{q}(n) + \boldsymbol{B}\boldsymbol{x}(n)$$

$$\boldsymbol{y}(n) = \boldsymbol{C}\boldsymbol{q}(n) + \boldsymbol{D}\boldsymbol{x}(n)$$

is defined as

sys = ss(A,B,C,D,1)

Controls Toolbox

Although MATLAB's controls toolbox allows discrete systems to be modeled as transfer functions in rational and factored representations, the state space representation is the principal system representation in MATLAB's controls toolbox. For example, many of MATLAB 's functions which accept rational, factored and partial fraction expansion system representations as inputs use a state space representation of the system internally. Regardless of the system representation, MATLAB's controls toolbox requires that the model be defined.

MATLAB's controls toolbox assumes that a system's transfer function or system function, $H(z)$, has the form

$$H(z) = \frac{b(1)z^N + b(2)z^{N-1} + \cdots + b(N+1)}{a(1)z^N + a(2)z^{N-1} + \cdots + a(N+1)}$$

The *eqtflength* function can be used to convert a transfer function in the signal processing toolbox form to a transfer function in the controls toolbox form. A discrete rational transfer function can then be defined as

```
sys_tf = tf(b,a,-1)
```

A factored system representation can be defined as

```
sys_zpk = zpk(z,p,k,-1)
```

and a state space model of the form,

$$\mathbf{q}(n+1) = \mathbf{A}\mathbf{q}(n) + \mathbf{B}\mathbf{x}(n)$$
$$\mathbf{y}(n) = \mathbf{C}\mathbf{q}(n) + \mathbf{D}\mathbf{x}(n)$$

can be defined as

```
sys = ss(A,B,C,D,-1)
```

State Space System Analysis

After the system has been defined, the *freqresp*, *pole*, *zero*, *dcgain*, *impulse*, and *step* functions can be used to determine the system's frequency response, poles, zeros, gain, impulse response and step response, respectively.

For example,

```
[h,n] = impulse(sys,10)
```

calculates the vector, *h*, which contains the first 11 samples of the system's impulse response and the vector *n* which contains the corresponding sample numbers. Similarly,

[s,n] = step(sys,10)

calculates the vector, *s*, which contains the first 11 samples of the system's step response and the vector, *n*, which contains the corresponding sample numbers for the elements in *s*. The **lsim** function calculates the system's response to a user defined input signal, *x*(*n*). For example

[y,n] = lsim(sys,x)

calculates the system's first *N* output samples in response to the input signal, *x*, where *N* is the length of the input signal, *x*. The **ss**, **impulse**, **step**, and **lsim** functions are part of MATLAB's control toolbox which should be installed on the college systems.

Systems can also be combined using addition and multiplication. Addition performs a parallel interconnection, and multiplication performs a series interconnection. For example,

```
sys1 = tf([1 0],[1 0.5],-1);
sys2 = tf([1 0],[1 0.25],-1);
sys3 = sys1*sys2
```

generate the system function,

$$H_3(z) = H_1(z) \cdot H_2(z) = \frac{1}{1+\frac{1}{2}z^{-1}} \cdot \frac{1}{1+\frac{1}{4}z^{-1}} = \frac{1}{1+\frac{3}{4}z^{-1}+\frac{1}{8}z^{-2}}$$

Exercises

For Exercises 1-11, use the discrete system described by the differential equation,

$$y[n] - 2r \cdot \cos(\omega_0) \cdot y[n-1] + r^2 \cdot y[n-2] = x[n] - r \cdot \cos(\omega_0)\, x[n-1]$$

where *x*(*n*) is the system's input, *y*(*n*) is the system's output,

$r = 0.9$ and $\omega_0 = \pi/4$ rad/sample.

1. Draw (and label)

   a Direct Form I block diagram of the system
   b Direct Form II block diagram of the system

Indicate the number of delays (memory registers) required to implement each block diagram.

2. Generate the state space equations for both the DFI and DFII implementations in matrix form.

3. Using the *tf2ss* and *eqtflength* functions, generate a state space model from the transfer function model that you developed in Exercise 1 (the numerator and denominator).

Compare the result to your state space model that you derived in Exercise 2 (the Matlab result should match your hand solution)

4. Using the *ss* function (don't forget tf=-1 for discrete systems), the *freqresp* function (Controls Toolbox included with student license), and the *squeeze* function, calculate the frequency response, $H(\omega)$, for $0 \leq \omega \leq 2\pi$. Plot $|H(\omega)|$ in dB and the phase of $H(\omega)$ in degrees using the *plot*, *title*, *xlabel*, *ylabel, subplot* and ***log10*** functions. (You should generate 2 plots on 1 page.)
5. Using the *zero*, *pole* and *dcgain* functions, determine the system's poles, zeros and gain.
6. Using the *pzmap* function, generate a pole zero plot of $H(z)$.
7. Using the *impulse* function, generate the system's impulse response, $h(n)$ for $0 \leq n \leq 50$. Plot $h(n)$ for $0 \leq n \leq 50$ using the *stem*, *title*, *xlabel*, and *ylabel* functions. Show that the results using the DFI and DFII are the same.
8. Using the *step* function, generate the system's step response, $s(n)$ for $0 \leq n \leq 50$. Plot $s(n)$ for $0 \leq n \leq 50$ using the *stem*, *title*, *xlabel*, and *ylabel* functions. Show that the results using the DFI and DFII are the same.
9. Generate a system model for the step function and multiply it with your system model. Using the resulting model and the *impulse* function, generate the system's step response, $s(n)$ for $0 \leq n \leq 50$. Plot $s(n)$ for $0 \leq n \leq 50$ using the *stem*, *title*, *xlabel*, and *ylabel* functions.
10. Using MATLAB's built-in functions to calculate the system output for $0 \leq n \leq 50$ with input $x(n) = \cos(0.2\pi n)u(n) + \cos(0.7\pi n)\,u(n)$. Show that the results using the DFI and DFII are the same.
11. In your own words, how would you be able to use this assignment to help you with future coursework in Digital Signal Processing? What are the bare minimum variables (or things to set up) to prepare in MATLAB so that you can analyze any discrete system? Note this exercise should serve as your own guideline on how to setup discrete systems.