SIGNALS AND SYSTEMS I
Computer Assignment 2

Lumped linear time-invariant discrete and digital systems are often implemented using linear constant coefficient difference equations. In MATLAB, difference equations can be implemented using one of MATLAB's two loop constructs, *for* loops and *while* loops, or using MATLAB's built-in **filter** function. If the system is also a state system, then if can also be implemented using a *for* loop or a *while* loop, or using built-in MATLAB functions.

Implementing Difference Equations

MATLAB's *for* loop construct is similar to the *for* loop construct in *C* and the *do* loop construct in FORTRAN. MATLAB's *for* loop construct repeats a set of commands between a *for* command and its corresponding *end* command a fixed, predetermined number of times. The *for* loop's syntax is

```
for var = start : increment (default = 1) : end,
   statements
end
```

For example,

```
for n = -2:2,
   x(n+3) = n;
end
x
```

generates the signal

$$x = [-2 \quad -1 \quad 0 \quad 1 \quad 2],$$

and

```
for n = 4:-2:-4,
   y(-n/2+3) = n;
end
y
```

generates the signal

$$y = [4 \quad 2 \quad 0 \quad -2 \quad -4].$$

The semicolon at the end of the statements, `x(n+3) = n;` and `y(-n/2+3) = n;` , suppresses the display of the values of $x(n+3)$ and $y(-n/2+3)$, respectively.

To implement a difference equation using a *for* loop, the length of the output must be known a priori. A single output of the difference equation is then calculated for each iteration of the loop. For example, the first 10 output samples of the causal difference equation

$$y(n) + a_1 y(n-1) = b_0 x(n) + b_1 (n-1)$$

where

$$x(n) = u(n)$$

can be calculated as follows:

```
x = ones (11,1);
y(1) = 0;
for n = 2:11,
    y(n) = -a1*y(n-1) + b0*x(n) +b1*x(n-1);
end
y = y(2:11)
```

If the difference equation is noncausal then the loop is operated in reverse. For example, the first 10 output samples of the noncausal difference equation

$$y(n) + a_1 y(n-1) = b_0 x(n) + b_1 (n-1)$$

where

$$x(n) = u(-n)$$

can be implemented as follows:

```
x = ones (11,1);
y(11) = 0;
for n = 11:2,
    y(n-1) = -y(n)/a1 + b0*x(n)/a1 +b1*x(n-1)/a1;
end
y = y(1:10)
```

MATLAB's *while* loop repeats a set of command between a *while* command and its corresponding *end* command until a given logical condition is false. The *while* loop's syntax is

```
while condition,
      statements
end
```

For example,

```
n = 0;
while n < 5,
      x(n+1) = n;
      n = n + 1;
end
x
```

Because of its construct, a *while* loop can implement a difference equation until a particular event, such as an end of data or a button is pressed, occurs. A *while* loop can also implement a difference equation much like a *for* loop does. For example, the first 10 output samples of the causal difference equation

$$y(n) + a_1 y(n-1) = b_0 x(n) + b_1 (n-1)$$

where

$$x(n) = u(n)$$

can be implemented as follows:

```
x = ones (11,1);
y(1) = 0;
n = 1;
while n <= 11,
   y(n) = -a1*y(n-1) + b0*x(n) +b1*x(n-1);
   n = n + 1;
end
y = y(2:11)
```

Difference equations can also be implemented using MATLAB's built-in **filter** function. The syntax for MATLAB's built-in **filter** function can be obtained by typing

```
help filter
```

at a MATLAB command prompt. The **filter** function is part of MATLAB's signal processing toolbox which should be installed on the college systems. If MATLAB reports back that

```
filter.m  not  found
```

MATLAB's signal processing toolbox has not been installed. Please E-mail the system administrator at staff@egr.unlv.edu and inform him of this problem. They might inform you that you need to use the college UNIX systems.

Exercises

For Exercises 1-3, use the discrete system described by the difference equation,

$$a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + a_3 y[n-3] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + b_3 x[n-3]$$

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

$a_0 = 1$	$a_1 = -0.18902544$	$a_2 = 0.71974192$	$a_3 = -0.15739157$
$b_0 = 0.25445939$	$b_1 = 0.43220307$	$b_2 = 0.43220307$	$b_3 = 0.25445939$

1. Draw (either using Simulink, a vector-based drawing application, or hand-drawn)
   a) a Direct Form I block diagram of the system.
   b) a Direct Form II block diagram of the system.
   c) a transposed Direct Form II block diagram of the system.

   Indicate the number of delays (memory registers) required to implement each block diagram.
2. Using a *for* loop or a *while* loop, write programs that implement each of the block diagrams that you drew in Exercise 1. Indicate the number of delays (memory registers) that each of your difference equation implementations use. Using all three programs, calculate the system's first 51 outputs when the system's input, $x(n)$, is
   a) $x(n) = \delta(n)$.
   b) $x(n) = u(n)$.
   c) $x(n) = \cos(0.05\pi n)u(n)$
   d) $x(n) = \cos(0.05\pi n)u(n) + \sin(0.6\pi n)u(n)$

   Plot the inputs and outputs (DFI, DFII, and TDFII) using the **stem**, **title** and **subplot** functions. (You should generate 16 plots on 4 pages, that is, four plots per page.). For part d, explain in your own words what is happening to the input/output as it goes through your program.
3. Using MATLAB's built-in **filter** function, calculate the system's first 51 outputs when the system's input, $x(n)$, is
   a) $x(n) = \delta(n)$.
   b) $x(n) = u(n)$.
   c) $x(n) = \cos(0.05\pi n)u(n)$
   d) $x(n) = \cos(0.05\pi n)u(n) + \sin(0.6\pi n)u(n)$

   Plot the inputs and outputs using the **stem**, **title** and **subplot** functions. (You should generate 8 plots on 4 pages, that is, two plots per page.) Note, Exercise 3's output should look like Exercise 2.

Implementing State Systems

Because linear time-invariant state systems can be described by the state equations,

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

which are a set of first order difference equations, state systems can also be implemented using MATLAB's *for* loop and *do* 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 follows:

```
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 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 system. For example, the discrete state system

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

is defined as

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

After the system has been defined, the **impulse** function, the **step** function and the **lsim** function can be used to calculate the system's impulse response, step response and response to a user defined input signal, 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 for the elements in *h*. 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. If MATLAB reports back that

function.m not found

MATLAB's control toolbox has not been installed. Please E-mail the system administrator at staff@egr.unlv.edu and inform him of this problem. He might inform you that you need to use the college UNIX systems.

Exercises

4. Generate state equations for the Direct Form II and the Transposed Direct Form II block diagrams that you drew in Exercise 1. Redraw these two block diagrams with the appropriate state equations.

5. Using a *for* loop or a *while* loop, write programs that implement each of the block diagrams that you drew in Exercise 4. Using both programs, calculate the system's first 51 outputs when the system's input, $x(n)$, is

   a) $x(n) = \delta(n)$.

   b) $x(n) = u(n)$.

   c) $x(n) = \cos(0.05\pi n)u(n)$

   d) $x(n) = \cos(0.05\pi n)u(n) + \sin(0.6\pi n)u(n)$

   Plot the inputs and outputs (DFII and TDFII) using the **stem**, **title** and **subplot** functions. (You should generate 12 plots on 4 pages, that is, three plots per page.) Note that the outputs will look similar to Exercise 2.

6. Using MATLAB's built-in functions (defined on pages 5 and 6) and one of your state space models, calculate the system's first 51 outputs when the system's input, $x(n)$, is

   a) $x(n) = \delta(n)$.

   b) $x(n) = u(n)$.

   c) $x(n) = \cos(0.05\pi n)u(n)$

   d) $x(n) = \cos(0.05\pi n)u(n) + \sin(0.6\pi n)u(n)$

   Plot the inputs and outputs using the **stem**, **title** and **subplot** functions. (You should generate 8 plots on 2 pages, that is, two plots per page.)