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. He 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,

$$y(n)+a_1y(n-1)+a_2y(n-2)+a_3y(n-3)=b_0x(n)+b_1(n-1)+b_2(n-2)+b_3(n-3)$$

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
$b_0=0.25445939$	$b_2=0.43220307$	$b_2=0.43220307$	$b_3=0.25445939$

1. Draw a Direct Form I block diagram of the system (see Chapter 10.8 for more details).

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

2. Using a *for* loop or a *while* loop, write programs that implement the DFI the block diagram that you drew in Exercise 1. 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) $x(n) = \cos(0.2\pi n)u(n)$
   d) $x(n) = \cos(0.2\pi n)u(n) + \cos(0.7\pi n)u(n)$

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

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) $x(n) = \cos(0.2\pi n)u(n)$
   d) $x(n) = \cos(0.2\pi n)u(n) + \cos(0.7\pi n)u(n)$

   Plot the inputs and outputs using the **stem**, **title** and **subplot** functions. (You should generate 8 plots on 2 pages, that is, four plots per page.) Compare the output between part 2. and 3. to ensure they are the same.

Convolution

In general, a lumped linear time invariant system can be described by a linear *N*th order constant coefficient difference equation of the form

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$$

where $x(n)$ is the system's input and $y(n)$ is the system's output. If the system is nonrecursive (the output is not a function of the output at other times), then the difference equation can be written as

$$y(n) = \sum_{k=0}^{M} b_k x(n-k)\,.$$

This nonrecursive system's impulse response, $h(n)$, can be determined by letting $x(n) = \delta(n)$ which implies that

$$h(n) = \sum_{k=0}^{M} b_k \delta(n-k) = b_n\,.$$

As a result, difference equations of nonrecursive systems are typically written as

$$y(n) = \sum_{k=0}^{M} h(k) x(n-k)$$

which is the convolution sum for a system with a finite impulse response (FIR) of length M+1.

Because nonrecursive systems do not have feedback, Direct Form I and Direct Form II implementations of nonrecursive systems are identical. Often, Direct Form implementations of convolution are implemented on array processors or digital signal processors (DSPs). These processor are typically optimized to perform vector matrix operations. As a result, convolution is often expressed as

$$y(n) = \sum_{k=0}^{M} h(k) x(n-k) = \mathbf{h}^T \mathbf{x}(n)$$

where

$$\mathbf{h}^T = \begin{bmatrix} h(0) & h(1) & \cdots & h(M) \end{bmatrix} \text{ and } \mathbf{x}(n) = \begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-M) \end{bmatrix}^T$$

Exercises

For Exercises 4, use the discrete system described by the finite impulse response, $h(n)$, where

$$h(0) = h(8) = \text{-}0.07568267 \qquad h(2) = h(6) = 0.09354893 \qquad h(4) = 0.4$$
$$h(1) = h(7) = \text{-}0.06236595 \qquad h(3) = h(5) = 0.30273069$$

4. Using MATLAB's built-in **conv** 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) $x(n) = \cos(0.2\pi n)u(n)$
   d) $x(n) = \cos(0.2\pi n)u(n) + \cos(0.7\pi n)u(n)$

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

Signal & Systems – I | Matlab Exercise – II

Paritosh Parmar

Do the following:

1. Read an audio file (I have emailed you an audio file)
2. Plot the audio envelope and look for the frequency of the signal
3. Play the audio clip
4. Modify the signal so that the new audio file plays at twice the original speed. Please create a new signal for this one and every other exercise.
5. Repeat the previous step, only this time the audio should play at half the original speed
6. Using convolution, modify the audio file such that it is twice as loud; show/listen to the resultant signal
7. Using convolution, add some noise to the original audio file; show/listen to the resultant signal. Try to plot the over each other for better comparison.
8. Smoothen out the original signal. Hint: moving average helps in doing so. Further hint: smoothening system should do the averaging over a small window
9. Bonus: try to remove the noise that was added in 7.