

Computer Exercises for Digital Signal Processing

References:

1. *Computer Based Exercises for Signal Processing Using MATLAB 5*, C. S. Burrus et al., Prentice-Hall, 1998. ISBN: 0-13-789009-5
2. *Digital Signal Processing Using MATLAB*, V. K. Ingle and J. G. Proakis, Brooks/Cole, 2000. ISBN: 0-534-37174-4
3. *Digital Signal Processing: A Computer-Based Approach*, S. K. Mitra, McGraw-Hill, 2001, ISBN: 0-07-232105-9
4. *Digital Signal Processing Laboratory Using MATLAB*, S. K. Mitra, McGraw-Hill, 1999, ISBN: 0-07-232246-2
5. *Digital Signal Processing with Examples in MATLAB*, S. Stearns, CRC Press, 2002, ISBN: 0-84-93109-11

Background:

This provides some general outline and instructions for several sets of computer exercises (assignments) with associated demonstration examples that will be handed out periodically during the course of the semester. These computer exercises for digital signal processing was developed to be used together with the textbook and as a supplement to the theory presented in the course lectures and the homework material. The primary goal of using software tools is to enhance your understanding of the fundamental concepts, theory and applications of digital signal processing.

The demonstration examples are designed using MATLAB[#]. MATLAB is a high-performance numeric computation, data analysis and visualization software. Because of the enormous popularity MATLAB has achieved recently and also due to the fact that it is available for both personal computers (Windows) and workstations (Unix), it has been adopted for this course. The student version of MATLAB is also available and can be purchased directly from the publishers (www.mathworks.com) or from the USF bookstores.

This is intended to encourage you to optimize your learning by further experimenting with these computer exercises, above and beyond what is given. Please feel free to provide any feedback including comments, corrections, additions or improvements you might like to see in the future.

General Instructions:

For all of the computer problems, provide the following:

1. Goals or objectives of the assignment
2. Any analytical equations or derivations, where necessary
3. The software code with comments and flow chart of the algorithm, where necessary
4. The input parameters and the output results (Wherever possible in graphical plots or concise tables and try to avoid printouts of long listings or input/output vectors)
5. Interpretation of the results with discussion or conclusion

[#] It is preferred that you use MATLAB for these computer assignments but one can use any other available software tools to solve the digital signal processing problems.

EEL 6502 - DIGITAL SIGNAL PROCESSING I
Computer Assignment #1 (Due Date: See Course Homepage)

I. SIGNALS

Background Reading: Oppenheim & Schaffer: Sections 2.0 & 2.1.

Overview: MATLAB can be used to generate and represent basic signals such as the unit impulse, exponentials of the form $a^n u[n]$, sine waves and their generalizations to complex exponentials.

Demonstration Examples:

(a) The following code will create 41 points of a discrete-time sinusoid:

$$x(n) = \sin(n)$$

```
index = 0:40;           % vector of time indices
sine = sin(index);
stem(index,sine);      % plot the sine function
```

Note that index (1) and sine (1) refer to $n=0$ in MATLAB. Try the following changes to the code with $\text{sine} = \sin(\text{index}/4)$ and $\text{sine} = \sin(\text{index}/2 + 1)$ and observe their effects.

(b) The following function will generate a discrete-time exponential signal of the form:

$$x(n) = a^n u(n)$$

A user-created function can be stored as an M-file; create a file with extension *.m* (e.g., *genexp.m* or *example.m*) and enter the following code. To invoke the function, assign values to the scalar parameters a , n_0 and L and then issue the command:

```
>> y = genexp(a, n0, L) or >> y = example(a, n0, L)
```

```
function y = genexp(a,n0,L)
% a is an input scalar giving the ratio between the terms, n0 is the starting index
% L is the length of the signal to be generated and y is the output signal.
if (L<=0)
    error('GENEXP: length not positive')
end
n = n0 + [1:L]' - 1;           % vector of indices
y = a.^n;                     % compute the function
stem(n,y)                     % plot the function
end
```

Try to use the function to plot $x(n) = (0.9)^n$ over the range $n = 0:40$

- (c) To plot the real and imaginary parts of a discrete-time signal such as $\mathbf{x}(n) = e^{jn/3}$ we can use the subplot function as follows.

```

index = 0:25;
xx = exp(j*index/3);
subplot(211)
stem(index, real(xx))
title('Real Part'), xlabel('Index (n)')
subplot(212)
stem(index, imag(xx))
title('Imaginary Part'), xlabel('Index (n)')

```

Try plotting the real part versus the imaginary part by using *plot (real(xx), imag(xx))*

PROBLEM 1.

- (a) Use Euler's Identity

$$\mathbf{x}(n) = (z_0)^n = r^n e^{j\theta n} = r^n (\cos \theta n + j \sin \theta n)$$

to generate a complex exponential with $z_0 = .9 \angle 45$. Plot the real and imaginary parts of $x(n)$ over the range $0 \leq n \leq 20$. Notice that the angle of z_0 controls the frequency of the sinusoids. Experiment with different angles.

- (b) Plot the real part versus the imaginary part of the signal in part (a).
- (c) Euler's identity can be modified by multiplying with a complex constant, allowing scaling of the amplitude and phase of the sinusoids.

$$G.z_0^n = A e^{j\phi} r^n e^{j\theta n} = A r^n [\cos(\theta n + \phi) + j \sin(\theta n + \phi)]$$

Generate and plot each of the following sequences. You may convert the sinusoids to complex notation and then create the signal vector using the function *exp*. If the signal is purely real, then taking the real part of the complex signal can generate it. The horizontal axes should extend over the range indicated and should be labeled accordingly.

$$\begin{aligned} x_1[n] &= 3 \cos(n\pi/8) + j4 \sin(n\pi/8) & 0 \leq n \leq 48 \\ x_2[n] &= 1.1^n \cos(n\pi/10 + \pi/4) & 0 \leq n \leq 48 \end{aligned}$$

II. DIFFERENCE EQUATIONS & FREQUENCY RESPONSE

Background Reading: Oppenheim & Schaffer: Sections 2.2 through 2.9.

Overview: An IIR (infinite impulse response) filter can be expressed as a difference equation:

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

We can represent the difference equation by two vectors, one for the coefficients b_l and one for the coefficients a_k . The MATLAB function $y = \text{filter}(b,a,x)$ implements a digital filter defined by the a and b coefficient vectors to filter the data stored in x . If x is the unit impulse, then y will be the impulse response $h(n)$.

If the transfer function corresponding to the difference equation is rational, the function, freqz , can be used to find its frequency response. The command $[H,W] = \text{freqz}(b,a,N,'whole')$ will evaluate the frequency response of the filter at N points, equally spaced in radian frequency around the unit circle. If you do not use the 'whole' option, freqz will only use the upper half of the unit circle (0 to π), which is sufficient for filters with real coefficients.

Demonstration Examples:

(a) Generate vectors b and a for the difference equation,

$$y[n] + 0.9 y[n-2] = 0.3 x[n] + 0.6 x[n-1] + 0.3 x[n-2]$$

Create a unit impulse vector, imp , of length 50. Generate the first 50 points of the impulse response of this filter. Use stem to plot the response versus time.

```
function y=de(b,a,L)
index = 0:(L-1); % generate an impulse
imp = zeros(L,1); % vector imp with all zero elements
imp(1) = 1; % imp(n=0) = 1
x = imp;
y = filter(b,a,x);
stem(index,y)
end
```

To use the function, first input a , b and L and then call the function as follows:

```
>> a = [1 0 0.9]; % coefficients from the difference equation
>> b = [0.3 0.6 0.3];
>> L = 51;
>> y = diffa(b,a,L) % the function statement should be y=de(b,a,L) without that
% the values generated by the function will not be saved
```

- (b) For the same filter, evaluate the frequency response at 512 points by using the command,
 $\gg [H,W] = \text{freqz}(b,a,N,'whole')$
 Make plots of the magnitude and phase responses with 512 frequency samples around the entire unit circle. You may use $\text{plot}(W, \text{abs}(H))$ or $\text{plot}(W, \text{angle}(H))$ or $\text{plot}(W, \text{angle}(H)*180/\pi)$. Also try $[H,F] = \text{freqz}(b,a,N,Fs)$
- (c) A transfer function can be specified in MATLAB by two vectors, one containing the coefficients of the numerator and the other containing the coefficients of the denominator. The zeros and poles can be obtained from the transfer function, by using the function tf2zp . A zero-pole plot can then be obtained by using the function zplane . (Type help function-name to get information on these functions or lookfor subject to search matlab functions on a specific topic)

```
 $\gg [z, p, k] = \text{tf2zp}(b,a);$  % b and a are the coefficients from the difference eqn.
 $\gg \text{zplane}(z, p)$ 
```

PROBLEM 2.

- (a) Use the filter function to generate and plot the impulse response $h[n]$ of the following difference equation. Plot $h(n)$ in the range of $0 \leq n \leq 100$.

$$y[n] - 1.6 \cos(\pi/8)y[n-1] + 0.64 y[n-2] = x[n] + 0.5 x[n-1]$$
- (b) Determine the impulse response analytically and confirm your results.
- (c) Verify that the transfer function of this system is rational (hint: compute the frequency response $H(e^{j\omega})$ from the difference equation).
- (d) Use the function freqz to evaluate the frequency response and draw the magnitude and phase responses, with 512 frequency samples around the entire unit circle. Draw the pole-zero plot.
- (e) Specify the type of filter defined by this difference equation: high-pass, low-pass, all-pass, band-pass or band-stop.
- (f) For the following difference equations, determine the frequency response and state what type of filter it defines.
- $y[n] + 0.13y[n-1] + 0.52y[n-2] + 0.3y[n-3] = 0.16x[n] - 0.48x[n-1] + 0.48x[n-2] - 0.16x[n-3]$
 - $y[n] + 0.268y[n-2] = 0.634x[n] + 0.634x[n-2]$
 - $y[n] - 0.268y[n-2] = 0.634x[n] - 0.634x[n-2]$
 - $4y[n] - 2y[n-1] + y[n-2] = x[n] - 2x[n-1] + 4x[n-2]$

III. FOURIER TRANSFORM: DTFT

Background Reading: Oppenheim & Schaffer: Sections 2.5 through 2.9.

Overview: The Fourier representation of a signal via the forward and inverse DTFT is a key part of signal analysis. The analysis and synthesis equations are respectively,

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}$$
$$x(n) = \frac{1}{2\pi} \int_{\pi}^{-\pi} X(e^{j\omega}) e^{j\omega n} d\omega$$

The frequency response of an LTI system is the DTFT of its impulse response.

Demonstration Examples:

- (a) Pulses are the easiest examples of using the DTFT. The MATLAB function will suffice for a signal with infinite length. For a signal with finite length, we can write the following function which is essentially a layer that calls *fft* (*h,N*).

```
function [H,W] = dtft(h,N)
% This function calculates DTFT at N equally spaced frequencies
% Usage H = dtft(h,N)
% h : finite-length input vector, whose length is L
% N : number of frequencies for evaluation over [ -pi, pi )
%     ==> constraint : N >= L
%
% H : DTFT values (complex)
% W : (2nd output) vector of frequencies where DTFT is computed
%
N = fix(N);
L = length(h); h = h(:);           %<-- for vectors ONLY !!!
if (N < L)
    error(' DTFT : # data samples cannot exceed # freq samples')
end
W = ( 2* pi / N ) * [ 0 : (N-1) ]';
mid = ceil( N / 2 ) + 1;
W(mid : N) = W( mid : N ) - 2 * pi;   % <-- move [pi, 2pi) to [pi, 0)
W = fftshift(W);
H = fftshift( fft ( h, N ) );        % <-- move negative freq components
```

Since the DTFT is periodic, the region from $\omega = \pi$ to 2π is actually the negative frequency region, so the transform values just need to be reordered. This is accomplished by the MATLAB function *fftshift*, which exchanges the upper and lower halves a vector.

- (i) For a **unit-pulse** of length $L=5$, evoke the function *dtft* given above and plot the magnitude and phase response for different values of N (say $N=32, 64, 128$, etc.)
- (ii) For a signal $\mathbf{x}(n) = (0.88 e^{j2\pi/5})^n$, the function in the above example can be used to compute the dtft and then a two-panel subplot to display the magnitude and phase response can be obtained as follows:

```
% --- example of calculating and plotting a DTFT
%
format compact, subplot (111)
a = 0.88 * exp (sqrt(-1) * 2 * pi / 5 ) ;
nn = 0:40; xn = a.^nn;
[X, W] = dtft ( xn, 128); % the matlab file containing the function dtft
subplot (211), plot ( W / 2 / pi, abs(X) );
grid, title ( ' MAGNITUDE RESPONSE' )
xlabel ( ' NORMALIZED FREQUENCY' ), ylabel ( ' | H(w) | ' )
subplot (212), plot( W / 2 / pi, 180 / pi * angle(X) );
grid, xlabel( ' NORMALIZED FREQUENCY' ), ylabel ( ' DEGREES ' )
title( ' PHASE RESPONSE ' )
```

PROBLEM 3.

Suppose that a rectangular pulse $r(n)$ is defined by

$$r(n) = \begin{cases} 1 & 0 \leq n \leq L-1 \\ 0 & \text{elsewhere} \end{cases}$$

- (a) Show that the DTFT of $r(n)$ is given by the formula,

$$R(e^{j\omega n}) = \frac{\sin(\omega L / 2)}{\sin(\omega / 2)} e^{-j\omega(L-1)/2}$$

The first term in this transform has a special form called the aliased sinc (*asinc*) function.

- (b) Use the function *dtft* to evaluate the DTFT of a 12-point pulse. Make a plot of the DTFT versus ω over the range $\omega = -\pi$ to π .
- (c) Plot the magnitude of the DTFT (See *abs* in MATLAB). To make the plot appear smooth, choose a number of frequency samples that is 5 to 10 times the pulse length. Experiment with different numbers of frequency samples. Be careful to label the frequency axis correctly for the variable ω when plotting. Check the zero locations and the peak height.
- (d) Notice that the zeros of the *asinc* function are at regularly spaced locations. Repeat the DTFT calculation and plot the magnitude for an odd length pulse, say 15-point pulse.