CALIFORNIA STATE UNIVERSITY **LONG BEACH**
COLLEGE OF ENGINEERING
COMPUTER ENGINEERING COMPUTER SCIENCE DEPARTMENT

# Digital Signal Processing

Project-2

Due: April 17, 2019

Instructor: Haney Williams

The labs and projects materials are taken from Dr. Thomas Johnson

# Part 1: Fourier Series.

Given a reasonably well behaved function x(t), Jean Baptiste Fourier long ago found the equivalent infinite series sinusoidal representation for the function and the resultant error of using only a finite number of terms.

1. ***The Complex Fourier Series Coefficients***

   Suppose you are given a simple periodic square wave: f(t) = 1/2 on t∈[0,1/2) and f(t) = −1/2 on t∈[1/2,1)

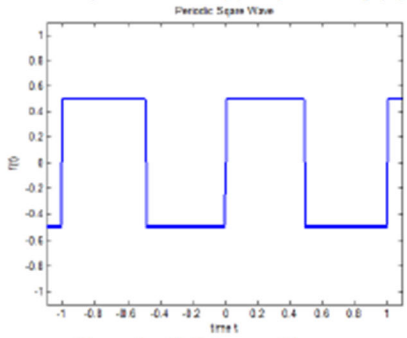



Figure 1 – Unit Square Wave

a) After reading the pdf file, find the expression for the complex Fourier coefficients $c_n$ for the square wave signal f(t) from these two formulas. g(t) is the reconstruction of f(t) using the coefficients $c_n$:

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-j\frac{2\pi nt}{T}}dt \qquad g(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi nt}{T}}$$

Note that $c_0$ is just the average value of the function taken over its period.
Once found, reconstruction of the waveform $g_N(t)$ using the complex coefficients $c_n$ is done in Matlab by:

```
Wn=exp( 1j*2*pi/T *  t'*n ) ;      % T is period, t is time vector, n is coefficient index vector.
W_n=conj(Wn);                      % The conjugate of Wn
gN=c0 + Wn*cn .' + W_n*c_n .';  % Careful not to conjugate transpose the row vector cn
```

b) Using the coefficients $c_n$ , plot the square wave in blue and its 10 term (n= ± 1, ± 3 ± 5, ±7, ±9) representation g(t) in red on the same plot using the "hold on" command. Note: In Matlab, a unit square wave of period T=1 over interval −1.1 to 1.1 can be constructed in simple fashion by the commands:

```
T=1;  t = -1.1:0.01:1.1;  f = 0.5*sign(sin(2*pi*t/T));  plot(t,f);
axis([-1.1, 1.1, -1.1, 1.1]);  title('Periodic Square Wave');  xlabel('time t');  ylabel('f(t)');
```

c) Plot the absolute value of the error between the points in f(t) and its estimate $g_N(t)$. Also calculate the rmse between the two and put it in the title of the plot using Matlab's *sprintf* command like so:

```
rmse = sqrt(sum(abs(f-gN).^2)/length(f));  plot(t,rmse);
str = sprintf ('RMS Error ||x(t)−gN(t)|| with RMSE= %6.4f ', rmse);
title(str);  xlabel('time t');  ylabel('Root Mean Square Error');
```

d) Determine how many coefficients are required for the rmse to fall just below a value of 0.075.

2. *Rate of Convergence*

Repeat the process of (1), but this time use a periodic triangular waveform defined by x=t on t∈[0,1] and x=2 – t on t∈[1,2]. You can make use of the integrals in the pdf for the saw–tooth wave to find the $c_n$.

a) For which signal does the Fourier series converge more rapidly, the periodic square wave signal or the periodic triangular signal? On what grounds could you base your answer? Justify your response by finding the number of coefficients of each that is required to have the rmse fall just below the rmse threshold of 0.05.

3. *Non–Integrable Functions*

Suppose you have a periodic waveform on t∈[0,5] such as:

$$x(t)= t^2 (5-|t|)e^{(-1+\cos(\sin(\pi t)))}/(2+e^{-|t-2.5|})$$

In this case the analytic integration formula for the Fourier series coefficients is difficult or impossible to find. When this occurs, the coefficients can be obtained numerically. Remember that an integral can be approximated by a sum (area under the curve):

$$\int_0^T f(t)dt \approx \Delta T \sum_m f(m\Delta T)$$

For example the Fourier coefficients

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-jn\omega_0 t}\,dt$$

can be approximated by a sum

$$c_n \approx \frac{\Delta T}{T}\sum_{m=0}^{M-1} f(m\Delta T)e^{-jn\omega_0 m(\Delta T)}$$

Using numerical integration Matlab program outlined in the pdf, find the n=±10 coefficients $c_n$ in the complex FS expansion of x(t). Plot the functions and error as done in (1). Also find the number of terms required to make the rmse fall just below the 0.075 threshold value.

4. *The Time-Shift Property*

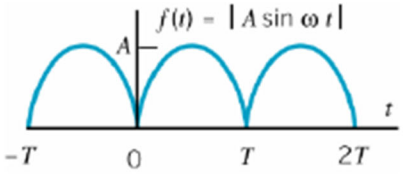
The time-shift property states that if a periodic signal is shifted in time, then the Fourier coefficients of that signal are shifted in frequency according to the following rule:

$$\textit{If } x(t) \leftrightarrow c_n \quad \textit{then} \quad x(t-t_o) \leftrightarrow \exp(-j\,n\,\omega_o\,t_o)\,c_n$$

Verify this property numerically using Matlab, by letting g(t) = f (t–0.25) of the square wave of (1). Then find the Fourier coefficients of g(t) by first finding the Fourier coefficients of f (t) of problem (1) and modifying those coefficients according to the rule. Use the new coefficients to synthesize the periodic signal $g_N(t)$ with N=20 by summing the complex exponentials. Confirm the time-shift property by a 3x1 subplot of the results of f(t), g(t) and its FS approximation $g_N(t)$ and noting their similarity, and then better still, finding the root mean squared error between g(t) and $g_N(t)$ and putting it in the title of the subplot of $g_N(t)$.

5. *FS for Periodic Trigonometric Function*

Find the Fourier series coefficients for the periodic signal $\boldsymbol{f(t) = |\cos(\pi\, t)|}$ , t∈[–1/2,1/2).
Repeat parts (1a) and (1b) only. To help you out a bit, look at this expansion:




Full wave rectified sine wave: $\omega_0 = \frac{2\pi}{T}$

$$f(t) = \frac{2A}{\pi} - \frac{4A}{\pi}\sum_{n=1}^{\infty}\frac{\cos(n\,\omega_0 t)}{4n^2-1}$$

## Part 2: The Filter Function

The filter function computes the output of a causal LTI system for a given input sequence when the system is specified by a linear constant-coefficient difference equation of the form

$$\sum_{k=0}^{K} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]$$

where x[n] is the system input sequence and y[n] is the system output sequence. If x is a Matlab vector of the input sequence x[n] on the interval $n_x \le n \le n_x + N_x - 1$ and the vectors a and b are the coefficients $a_k$ and $b_m$, then y=filter(b,a,x) returns the output sequence y[n] of the LTI system over the same interval as the input sequence x[n]. Remember that the first index of a Matlab vector is 1 and not 0. The function filter assumes all values of a sequence at indicies not explicitly defined are zero, especially at any indicies less than 1, e.g. x(0)=x(-1)=y(-1)=0.

1. Let x=[1,2,3,4] and find the output for the following LTI difference equations:

   (a) y[n]=0.5x[n]+x[n-1]+2x{n-2]
   (b) y[n]=0.8y[n-1]+2x[n]
   (c) y[n]-0.8y[n-1]=2x[n-1]

The function filter can also be used to do discrete-time convolution provided that the only non-zero coefficient on the LHS of the difference equation is $a_0$=1. This defines an FIR (finite impulse response) filter. In this case the equation is

$$y[n] = \sum_{m=0}^{M} b[m]\, x[n-m] = \sum_{m=-\infty}^{\infty} b[m]\, x[n-m]$$

where b[m] is now the impulse response of the FIR LTI filter.

2. Let x[n]=(-1).^n over interval $0 \le n \le 5$ and let y[n]=x[n]+0.5x[n-1]-2x[n-2]+0.75x[n-3].
   (a) Find the vector b for the LTI equation for y[n]. Set h=b and use the filter function y=filter(h,1,x) to find y[n] over the interval ny=0:5. Plot the results using stem(ny,y). Note: vector a is just the vector [1].
   (b) If ***filter*** is to generate the same output as ***conv***, then the input x[n] to the filter function must be of length 9 (=6+4-1), not 5 as before. Append 3 zeros to x to make x1[n] of length 9, and find y1=filter(h,1,x1). Compare this result to the result ***yconv=conv(h,x).*** Is the difference ***y1-yconv*** identically zero at all points?
   (c) The filter function can implement a non-causal impulse response. Let h(n)=n for $0 \le n \le 5$. Generate the impulse response h1[n]=h[n+5] over $-5 \le n \le 0$. Let x[n]=(-1).^n over $0 \le n \le 5$. The output y1[n] using h1[n] can be shown to be just an advanced version of the output y[n] using h[n], namely y1[n]=y[n+5]. Stem plot using subplot the impulse responses h[n] and h1[n] and the outputs of y[n] and y1[n] to verify this result.

## Part 3: The Echo Cancellation

Oftentimes recorded signals are corrupted by noises, such as echoes. This exercise attempts to create an echo and then to remove the echo by signal processing.

1. Load the file ***mtlb*** into Matlab by executing the command: ***load mtlb.mat***. Now enter "who" and see what variables are present. You should see the variable ***mtlb*** and ***Fs***. Set x=mtlb and play the signal variable x by execution the command "sound(x,Fs)". You should hear the word "matlab." The echo signal y[n] which is represented by the vector y, is of the form $y[n]=x[n] + \alpha\, x[n-D]$, where x[n] is the uncorrupted speech signal which has been delayed by D samples and added back in with its amplitude multiplied by the gain factor $\alpha < 1$. This is a reasonable model of an echo reflecting off an absorbing barrier such as a wall. Here assume the value of $\alpha = 0.9$. Find the length of x, and append that many zeroes to the vector x to increase the length of the sound vector to allow some room for the echo. Let the length of x be Nx and the number of delay samples be D=2750. Define the filter vectors ***a*** and ***b*** from the difference equation and generate the echo signal using ***y=filter(b,a,x).*** Play the echo signal back using ***sound(y,Fs)*** to hear the echo. Subplot on a 3x1 figure(1) the signals x and y.

2. Since the echo signal can be represented by a linear system of the form mentioned, determine the impulse response of this echo system. Store the impulse response in a vector ***he*** for $0 \le n \le Nx$.

3. Consider an echo removal system described by the LSI difference equation $z[n] + \alpha\, z[n-D] = y[n]$, where y[n] is the input and z[n] is the output which has the echo removed. Show that this equation is indeed an inverse of the first equation by deriving the overall difference equation relating z[n] to x[n]. Is z[n]=x[n] a valid solution to the overall difference equation?

4. The inverse filter echo removal system of (3) will have an infinite impulse response. Assuming the previous values of D and $\alpha$, compute the impulse response using ***filter*** with an impulsive input d[n] given by the vector ***d***=[1,50000]. Store this approximation to the infinite impulse response in the vector ***her***.

5. Implement the echo removal system using ***w=filter(1,a,y)*** where ***a*** is the appropriate coefficient vector derived from the equation in (3). Plot the output using plot as the third graph in figure(1). Also listen to the output using ***sound(w)***. The echo should not be present.

6. Calculate the overall impulse response of the cascaded echo system (equation in (2)) and the echo removal system (equation in (3)), by convolving **he** with ***her*** and store the results in ***hoa.*** In figure(2), plot on a 3x1 subplot the vectors ***he***, ***her*** and ***hoa***. The resulting plot of **hoa** is not a unit impulse as you would expect. Why is this the case?

# Part 4: The DTFT

The DTFT of a discrete-time signal x[n] is given by the DTFT analysis equation

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

and the synthesis equation

$$x[n] = \frac{1}{2\pi}\int_{\omega_0}^{\omega_0+2\pi} X(e^{j\omega})\, e^{j\omega n}\, d\omega$$

where the interval in integration can be any interval of length $2\pi$.
The DTFT is periodic with period $2\pi$. If x[n] is infinite in length, it must be truncated to a finite number of samples since only a finite length signal can be represented as vectors in Matlab. $X(e^{j\omega})$ is continuous in $\omega$ but can be represented in Matlab by a sequence of closely placed samples which can be plotted and will appear as a continuous function. For computational efficiency, the optimum set of frequency samples is the set of equal-spaced points in the interval $0 \le \omega < 2\pi$ given by $\omega_k = 2\pi k/N$ for k=0,1,2...,N-1. For an x[n] nonzero only on $0 \le n \le M-1$, the frequency samples are

$$X(e^{j\omega}) = \sum_{n=0}^{M-1} x[n]e^{-j2\pi kn/N} \quad for\ k = 0,1,2,\ldots.,N-1$$

The usefulness of the DTFT is that it represents the complex eigenvalues for the exponential eigenfunctions of the LSI constant coefficient difference equations. If the sequence x[n] is the impulse response h[n] then $X(e^{j\omega})$ is just the frequency response function $H(e^{j\omega})$. Suffice it to say that if an LSI system has an input of $\exp(j\omega_0 n)$ then the output of the system is the product $H(e^{j\omega_0})\exp(j\omega_0 n)$. This multiplication by a complex quantity acts on only the magnitude and phase of the input in generating the system output. You can read from the plot of the continuous function $H(e^{j\omega})$ how each particular frequency is affected. You do not need to re-compute the output for every frequency of interest. This is a time-saver.

1. Create a vector h[n]=u[n]-u[n-11] where u[n] is the unit step function. For N=100, use the DTFT equation to find $H(e^{j\omega})$. Plot the results in figure(1). Remember to plot the magnitude and phase (in degrees) of a complex quantity, normally using a 2x1 subplot with the frequency axis in units of $\omega/\pi$.

2. The DTFT equation can be accomplished more easily by using the matrix capabilities of Mablab. Define a vector k=[0:N] where N is the number of frequency samples required over $[0,2\pi]$ and a vector n=[n1:n2] where n1 is the initial sample index of the sequence x[n] and n2 is the final sample index. Then in Matlab the expression X=x*(exp(-1j*2*pi/N)).^(n' * k) gives the vector representing the continuous function $X(e^{j\omega})$ over the interval $[0,2\pi]$ with w = k*2*pi/N. Find the DTFT of $x(n) = e^{-0.2|n|}$ over n=-10:10 with N=1000 using this vector approach to finding $X(e^{j\omega})$ and plotting the results in figure(2).

3. Repeat (2) for the impulse response sequence $h[n]= (0.81)^n$ [u(n)-u(n-101)] and plot the DTFT of h[n] in figure(3). If such a system has an sinusoidal input of x[n]= cos(0.2πn), what would be the output based on the plot of $H(e^{j\omega})$? What is the output's delay in terms of number of samples?

## Part 5: The Sampling and Reconstruction

1. Consider an analog signal $x_a(t) = \cos(20\pi t)$ over $0 \le t \le 1$ seconds. It is sampled at intervals of Ts = 0.01 and 0.05 and 0.1 sec to obtain three discrete signals $x_1(n)$, $x_2(n)$ and $x_3(n)$ respectively.

   (a) For each Ts, plot the resulting $x_k(n)$ for k=1,2,3 using a 3x1 stem subplot.

   (b) Reconstruct the analog signal $y_c(t)$ from the samples x(n) using the ***zero-order hold*** interpolation. See page 90. Using subplot, plot $y_c(t)$ in each case. Determine the frequency of $y_c(t)$ from the your plots, ignoring end effects. An analog post-filter would be used to smooth the corners of the first staircase signal to yield a sine-like waveform.

   (c) Reconstruct the analog signal $y_c(t)$ from the samples x(n) using the ***first-order hold*** interpolation. See page 91. Using subplot, plot $y_c(t)$ in each case. The plotting function connects adjacent points with a straight line. Determine the frequency of $y_c(t)$ from the your plots, ignoring end effects. An analog post-filter would be used to smooth the corners of the straight-line plots of the signal to yield a smoother sine-like waveform.

   (d) Reconstruct the analog signal $y_c(t)$ from the samples x(n) using the ***cubic spline*** interpolation. See page 96. Using subplot, plot $y_c(t)$ in each case. Determine the frequency of $y_c(t)$ from the your plots, ignoring end effects.

   (e) Reconstruct the analog signal $y_s(t)$ from each set of samples $x_k(n)$ using the ***sinc*** interpolation ($\Delta t$=0.001). See page 93 of the text. Using a 3x1 subplot, plot $y_s(t)$ in each case. Determine the frequency of $y_s(t)$ from your plots, ignoring end effects.

   (f) Comment on the results.