EE 274: Tutorials for Discrete Fourier Transform

Rhandley D. Cajote

The Discrete Fourier Transform

The *Discrete Fourier Transform* (DFT) gives the frequency samples obtained by evaluating the *Fourier Transform X(ω)* at a set of N (equally spaced) discrete frequencies. The Fourier Transform of any finite-duration sequence of length L is

$$X(\omega) = \sum_{n=0}^{L-1} x(n)e^{-j\omega}, \text{ for } 0 \leq \omega \leq 2\pi \tag{1}$$

Sampling *X(ω)* at equally spaced frequencies $\omega_k = \frac{2\pi k}{N}, \text{ for } k = 0, 1, 2, ..., N-1$

Where N $\geq L$, the resulting samples are

$$X(k) = X\left(\frac{2\pi k}{N}\right) = \sum_{n=0}^{N-1} e^{-j2\pi kn/N} \tag{2}$$

The relation in equation (2) is called the DFT of *x(n)*. For convenience the limit of the summation is increased from L-1 to N-1. This is still valid since *x(n)* is zero for $n \geq L$.

The inverse of DFT (Inverse Discrete Fourier Transform) allows the recovery of the sequence *x(n)* from the frequency samples *X(k)*. The relation is expressed by:

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k)e^{j2\pi kn/N} \tag{3}$$

The nature of the DFT implies periodicity in the time-domain. From the results in (3), x(n) is non-zero for $0 \leq n \leq L-1$ and it is periodic every N samples.

The signal x(n) is created by taking N samples of the original signal x(t) over some finite duration of time, D. The signal X(k) is the DFT of the analog time signal, x(t). The previous equation shows it can be computed by taking the N samples of x(t), multiplying each sample by a weighted exponential function, then summing the results. The DFT will have N values spaced apart in the frequency domain by 1/D.

1.1 Direct computation of the DFT

The DFT can be computed by direct application of equation (1). It requires computing for the matrix of complex sinusoids, the columns represent the frequency components, the rows the n-samples. To illustrate, let us apply equation (1) to solve for the DFT of a discrete time signal given below, plot the magnitude and phase of the DFT.

$$x(n) = 5cos\left(\frac{7\pi n}{16} + \frac{\pi}{4}\right) + 3sin\left(\frac{7\pi n}{8} + \frac{\pi}{18}\right) \quad \text{for } n = 0,1,2,\ldots,N-1$$

Last modified. Sept. 20, 2020

Calculate the DFT using N = 100 sample points, the code is:

```
N = 100;
k = [0:N-1];
n = [0:N-1];
xn = 5*cos(7*pi.*n/16+pi/4) + 3*sin(7*pi.*n/8+pi/18);
w = 2*pi.*k/N;
Xw = xn*exp(-i*n'*w);
figure(1); subplot(2,1,1), plot(w,abs(Xw));
subplot(2,1,2), plot(w, angle(Xw));
```

This also demonstrates that the DFT can be computed as a linear transformation. The plot of the magnitude response shows two peaks, because the signal is a sum of two sinusoids. The x-axis shows the normalized frequency from 0 to 2π. In MATLAB the DFT is also computed using the **fft()** function.

1.2. Demonstration of the DFT using the fft() function

The function *fft()* calculates the DFT of a given signal. The function allows you to specify the number of frequency samples to compute. By increasing the number of frequency samples the frequency resolution of the signal can be increased. If the numbers of frequency samples are multiples of powers of two (512, 1024, 2048, etc.), the DFT will be computed using the FFT (Fast Fourier Transform) algorithm. The example below will demonstrate how to compute for the DFT using fft().

Generate at least five periods of three sinusoids at the following frequencies freq = [220 440 880] with amplitudes of 1, 1.5 and 0.75 respectively and a sampling rate of 8000 Hz. Add the three sinusoids and plot the resulting waveform.

```
fs = 8000;
freq = [220 440 880];
amp = [1 1.5 0.75];
ph = [0 0 0];
t = [0:1/fs:5/min(freq)];

% generate the sinusoid
y
=((amp'*ones(1,length(t))).*sin((2*pi*freq'*t)+ph'*ones(1,length(t))))';
y_sum = sum(y');
figure(2); plot(y_sum);
```

Calculate the Fourier Transform of the signal by using the fft() command in Matlab, with 4096 frequency points. Generate the frequency axis from 0 to fs. Plot the magnitude and phase response. What property of the DFT can be derived from the plot? What are the frequencies of the sinusoids?

```
% Use the fft routine to determine frequency components of the signal
Nfft = 4096;
y_fft = fft(y_sum,Nfft);

% generate the frequency axis
```

```
w = linspace(0, fs, Nfft);

% Plot the magnitude and the phase response,
mag = abs(y_fft); ph1 = angle(y_fft);
figure(3);
subplot(2,1,1), plot(w, mag);
subplot(2,1,2), plot(w, ph1);
```

The fft() calculates the fourier transform of the signal from 0 to fs. It is sometimes convenient to view the magnitude and phase response of a signal with frequency range from –fs/2 to fs/2. To do this, we have to use the fftshift() function to shift the frequency response and generate a new frequency axis ranging from –fs/2 to fs/2. What does the function fftshift() do?

```
% Using a different perspective, plotting from -fs/2 to fs/2
w2 = linspace(-fs/2, fs/2, Nfft);
figure(4); subplot(2,1,1), plot(w2,fftshift(mag));
subplot(2,1,2), plot(w2,fftshift(ph1));
```

Since we assume that the signal does not have frequency components above fs/2, we are interested mostly in the DFT from 0 to fs/2. The spectral information from fs/2 to fs is just a mirror image of 0 to fs/2. We can truncate the fourier transform of the signal to show only the magnitude and phase response from 0 to fs/2.

```
w3 = linspace(0, fs/2, Nfft/2);
y_fft2 = y_fft(1:length(y_fft)/2);
figure(5); subplot(2,1,1), plot(w3, abs(y_fft2));
subplot(2,1,2), plot(w3, angle(y_fft2));
```

Important: The magnitude of the Fourier Transform gives you information about the frequency components of a given signal and their relative amplitudes. To properly interpret the output of the fft(), always remember that the frequency range of the transform is from 0 to f_s, where f_s is the sampling rate. Due to the symmetric nature of the DFT, usually only the spectral information from 0 to f_s /2 is of main interest.

1.3 On Frequency resolution of the DFT

Since the frequency range is always from 0 to f_s, and the number of frequency samples can be determined by specifying the length N of the DFT. These two parameters give you the frequency resolution of DFT, given by the relation

$$f_{\text{bin}} = f_s / N$$

where f_{bin} is the smallest frequency that the DFT can resolve. To increase the frequency resolution, you have to increase N which increases the computation time.

A. Resolving two signals that are close in frequency

EE 274: Tutorials for Discrete Fourier Transform

Rhandley D. Cajote

1. Generate two sinusoids at f1 = 100 Hz with an amplitude of 1.2 and at f2 = 150 Hz with an amplitude of 0.8. The length of the sinusoids should be five periods of the lowest frequency (which is 100 Hz) and the sampling rate is 4000 Hz. Plot the sum of the sinusoids in time.

The following Matlab commands will do this

```
fs = 4000;
freq = [100 150];
amp = [1.2 0.8];
ph = [0 0];
t = [0:1/fs:5/min(freq)];

% generate the sinusoid
y =
((amp'*ones(1,length(t))).*sin((2*pi*freq'*t)+ph'*ones(1,length(t))))'
;
figure(6); plot(sum(y'));
```

2. Calculate the DFT of the first 100 samples of the signal using *fft*. Use stem instead of plot to show the magnitude and phase response. Use the *axis* to zoom in the frequency range: 0 to 200 Hz.

```
ty_sum = sum(y');
y_sum = ty_sum(1:100);

Nfft = 100;
y_fft = fft(y_sum, Nfft);

% generate the frequency axis
w = linspace(0,fs,Nfft);

% Plot the magnitude and the phase response
mag = abs(y_fft);
ph1 = angle(y_fft);
figure(7); subplot(2,1,1), stem(w, mag, '^');
axis ([0, 200, 0, 140]);
% use axis to zoom into plot to determine frequencies of sinusoids
title('The magnitude response from 0 to fs');
axis;  % turn autoscaling back on
subplot(2,1,2), stem(w,ph1, '^');
title('The phase response from 0 to fs');
```

a) What is the frequency resolution of our DFT?
b) Can you tell from the plot of the magnitude response the number of sinusoids in the signal?
c) From the plot, can you determine the frequency of the sinusoids?

Although the magnitude plot of the DFT indicates that there are two sinusoids because there are two peaks. The frequencies of the signals cannot be resolved. This is because the number of frequency samples (100) and the sampling rate (4000 Hz) gives us a resolution of f_{bin} = 4000/100 = 40 Hz. The magnitude of the DFT gives us the frequency response at multiples of 40 Hz in the frequency axis. This will mislead us into thinking that the sinusoids are at 80 Hz and at 160 Hz. The frequencies of 100 and 150 Hz are resolved to the closest frequencies that the DFT can discriminate which is 80 and 160 Hz. This observation is also known as "picket fencing', where the magnitude response is resolved depending on the frequency resolution. For this case the frequency resolution is poor and cannot resolve the actual frequencies of the sinusoids.

Last modified. Sept. 20, 2020

To increase the frequency resolution to 10 Hz, we must have at least 400 frequency samples, so that $f_{bin} = 4000/400 = 10$ Hz.

3. Change the number of frequency samples to 400 in order to to increase the frequency resolution of the DFT to 10 Hz. Compute and plot the magnitude response.

a) Can we now correctly determine the frequencies of the sinusoids?
b) What is the effect of increasing the number of frequency samples on the accuracy of the DFT?

4. This time, reduce the length of the signal to two periods of the lowest frequency. Compute and plot the DFT of the signal with N = 100 and N = 400, and a sampling rate of 4000 Hz. Can the DFT resolve the frequencies of the two signals? Why or why not?

B. Effects of zero padding and signal length on the DFT

1. Generate two periods of a sinusoid with amplitude of 1, frequency 100 Hz and a sampling rate of 400 Hz. Compute the DFT with N = 100 frequency samples and plot the magnitude response from 0 to $f_s/2$.

```
fs = 4000;
freq = 1000;
amp = 1;
ph = 0;
t = [0:1/fs:2/freq];

% generate the sinusoid
y = amp*sin((2*pi*freq*t)+ph)';

y_sum = y;
Nfft = 100;
y_fft = fft(y_sum, Nfft);

% generate the frequency axis
w = linspace(0, fs/2, Nfft/2);
y_fft = y_fft(1:length(y_fft)/2);

% Plot the magnitude response,
mag = abs(y_fft);
figure(11); plot(w, mag, '^');
title('The magnitude response from 0 to fs/2');
```

a) Why is the plot of the frequency response similar to a sinc function?

b) What is the effect of varying the number of frequency samples on the magnitude response?

c) What is the effect on the magnitude response by increasing the number of samples (also increasing the length of the signal) from 2 periods to 5 periods?

d) What is the effect of increasing the signal length on the width of the main lobe of the magnitude response?

2. In computing for the *fft* of the signal, compute for the DFT without specifying the number of FFT points. Plot the magnitude of the DFT without specifying the frequency axis.

a) What is the effect on the magnitude response when N is not specified?

b) To increase the number of signal samples, the easiest way is to pad the signal with 100 zeros at the end. Plot the magnitude response. What is the effect on the DFT when the signal is padded with zeros?

c) The more difficult way is to add more observations of the signal. Remove the zeros that you padded from b). Now increase the number of signal samples by increasing the number of periods of the signal from 2 to 5. Compute the DFT and plot the magnitude. What is the effect of increasing the number of periods on the magnitude plot? How does this compare with zero padding?

C. For Graduate students, you must also do the following experiments.

This part of the exercise is taken from "Matlab Exercise to Explain Discrete Fourier Transforms" by Prof. Kathleen A.K. Ossman, PhD. University of Cincinnati with some modifications on the Matlab code.

The DFT, which can be computed from a finite number of samples of x(t), is approximately equal to the sampled spectrum of x(t) scaled by a constant,1/ts. The accuracy of the DFT depends on two factors: the sampling rate chosen for x(t) and the duration of time over which x(t) is sampled.

- If x(t) is not sampled sufficiently fast, aliasing will occur. The DFT will not be the same as the original spectrum
- If x(t) is not sampled for a long enough duration, the frequency resolution of the DFT will be poor.

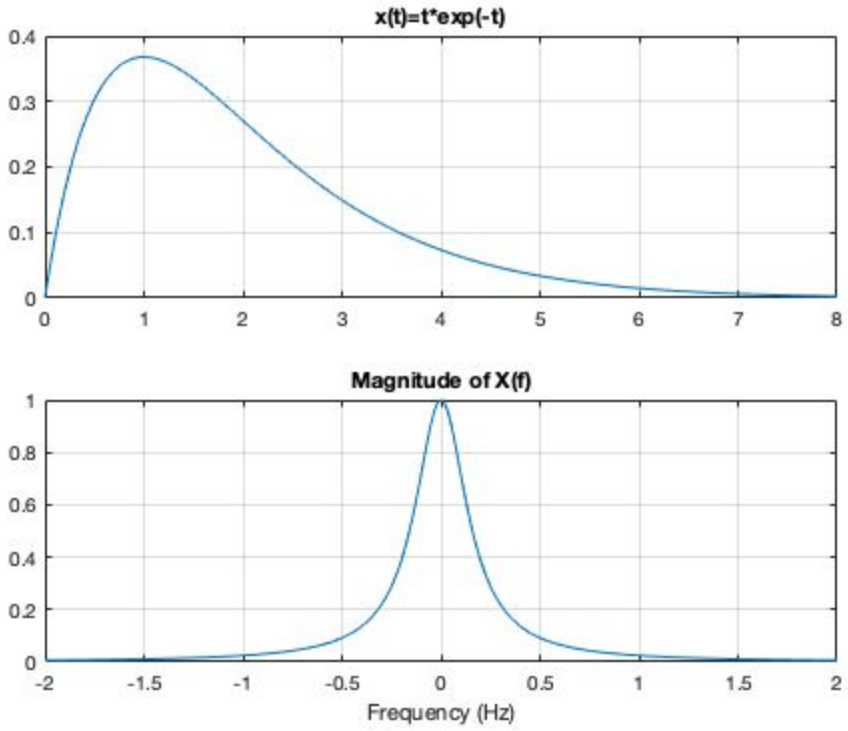
In this tutorial you will investigate the effect of the sampling rate and duration on the DFT.
The time signal chose is the exponential function given by:

$$x(t) = te^{-t} \Leftrightarrow X(f) = 1/(1+j2\pi f)^2$$

The signal and its spectrum are plotted in Matlab as

```
%% plot of t e^(-t) and spectrum X(f)
clear;
```

```
close all;

t=0:0.01:8;     % use a high enough sample rate to make it look continuous
x=t.*exp(-t);
subplot(2,1,1); plot(t,x); title('x(t)=t*exp(-t)');
grid on;
f=-2:0.01:2;
xf=1./(1+2*pi*j*f).^2;
subplot(2,1,2); plot(f,abs(xf)); title('Magnitude of X(f)'); grid on;
xlabel('Frequency (Hz)');
```



In order to compute a DFT, the time signal x(t) must be sampled over some duration of time. According to the Nyquist Theorem, the sampling frequency should be at least twice the bandwidth of the signal. As seen from the plot of the spectrum, the spectral components above 1 Hz are very small. In order to see the effects of sampling rate, two different sampling frequencies are chosen: 2 Hz and 16 Hz. The duration of time over which the signal is sampled is chosen to be 8 seconds. The table below shows some of the key parameters that you will investigate:

Sampling Frequency, Sf	**Duration, D**	**Number of sample, N**	**Resolution, fo**	**Frequency Range**
2 Hz	2 seconds	16	⅛ Hz	-2 Hz to 2 Hz
16 Hz	4 seconds	128	⅛ Hz	-8 Hz to 8 Hz

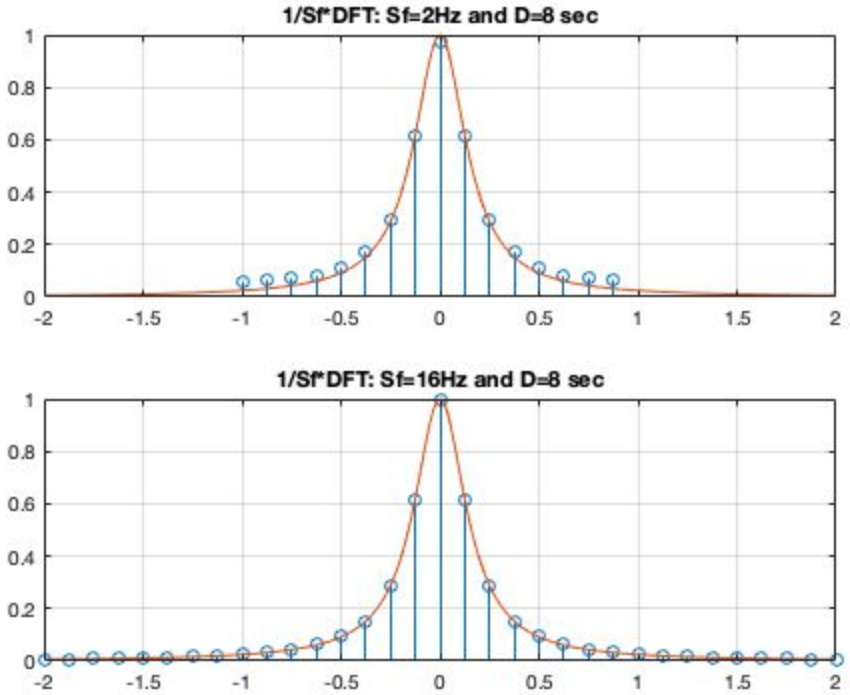
An increase in sampling frequency or duration will increase the size of the DFT which in turn increases computational complexity. The following figures illustrate the effect of increasing the sampling rate on the accuracy of the DFT. At the lower sampling rate of 2 Hz, the accuracy of the DFT begins to degrade around 0.5 Hz due to aliasing. A sampling rate of 16 Hz results in a very accurate sampled spectrum. The DFT in the second case actually ranges from –8 to 8 Hz but was truncated to 2 Hz to allow a nice comparison with the lower sampling rate over the frequency range of interest.

This is demonstrated in the following code.

```
%% Sample the signal and plot the spectrum
% (a) Sf= 2 Hz, D=8 sec, N=16 samples
Sf=2;       % sampling period is 2s
D=8;        % duration in seconds
N=16;       % Number of DFT samples

tn=0: 1/Sf: (N-1)*1/Sf;   % discrete-time samples
xn=tn.*exp(-tn);
ya=fft(xn,N);
ya=fftshift(ya);

fo = 1/D;     %Spectral Resolution;
fa = -(N/2)*fo:fo:(N/2-1)*fo ;
figure; subplot(2,1,1); stem(fa,1/Sf*abs(ya));
title('1/Sf*DFT: Sf=2Hz and D=8 sec'); grid on; hold on;
plot(f,abs(xf)); hold off;
% some aliasing is expected here, why?

% (b) Sf=16 Hz; D= 8 sec; N=128 samples
Sf=16;
D=8;
N=128;
tn = 0:1/Sf: (N-1)*1/Sf;
xn=tn.*exp(-tn);
yb=fft(xn,N); yb=fftshift(yb);

fo=1/D; %Spectral Resolution;
fb=-(N/2)*fo:fo:(N/2-1)*fo;

% Truncate the frequency range to 2 Hz.
% Find the entry in the frequency vector corresponding to -2 and +2 Hz:
f1= (-2-(-N/2*fo))/fo + 1; % (Desired - Start)/Increment + 1
f2=(2 - (-N/2*fo))/fo + 1;
subplot(2,1,2); stem(fb(f1:f2), 1/Sf*abs(yb(f1:f2)));
title('1/Sf*DFT: Sf=16Hz and D=8 sec'); grid on; hold on;
plot(f,abs(xf)); hold off;
```

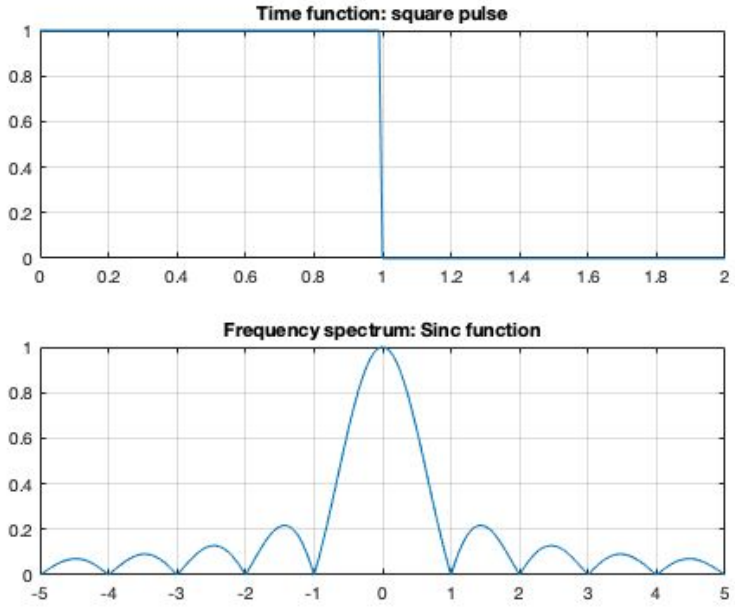
EE 274: Tutorials for Discrete Fourier Transform

Rhandley D. Cajote



To investigate the effect of time duration D on the DFT, the time signal is chosen to be a pulse of amplitude 1 with a duration of 1 second. The spectrum of this signal is a sinc function as shown.

$$x(t) = \text{rect}(t - 1/2) \Leftrightarrow X(f) = e^{-j\square f}\text{sinc}(f)$$




```
%% Example 2 on rectangular pulse
clear;
```

Last modified. Sept. 20, 2020

```
close all;
P = 2;          % duration of pulse is 2 seconds
W = 2;          % width is W/2 seconds
t=0:0.01:P;
R = rectpuls(t,W);

% plot the theoretical spectrum of square wave
f = -5:0.01:5;
% Theoretical DFT of a rectangular pulse
XR = exp(-j*pi.*f).*sinc(f);

figure; subplot(2,1,1), plot(t,R);
title("Time function: square pulse");
grid on;
subplot(2,1,2), plot(f,abs(XR));
title("Frequency spectrum: Sinc function");
grid on;
```

In order to compute a DFT, the time signal x(t) must be sampled over some duration of time. It is necessary to choose a sampling rate, Sf, and a duration of time, D. Several choices are explored in this example. According to the sampling theorem, the sampling rate should be at least twice the bandwidth of the signal. This particular signal is not band-limited, but we will assume an effective bandwidth of 4 Hz which would require a sampling rate of at least 8 Hz. The sampling rate is initially chosen to be 8 Hz then increased to 16 Hz. Since the signal will not be pre-filtered in this example, some aliasing will occur. The duration of time, D, is initially chosen to be 1 sec then increased to 2 sec and 4 sec to illustrate the effect on resolution. The effects of varying sampling rate and duration on the accuracy of the DFT are shown in the following figures.

EE 274: Tutorials for Discrete Fourier Transform

Rhandley D. Cajote

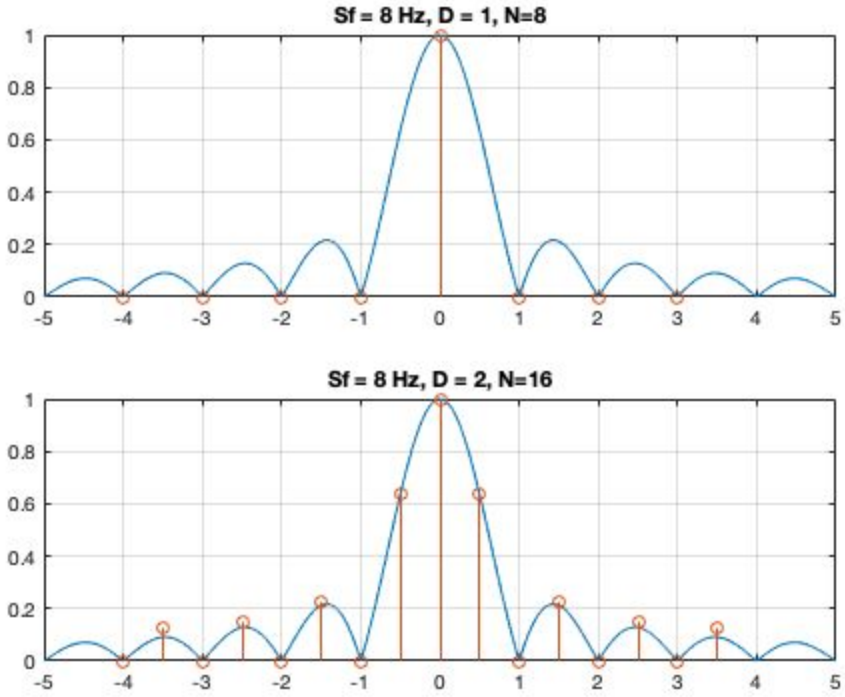


Figure A

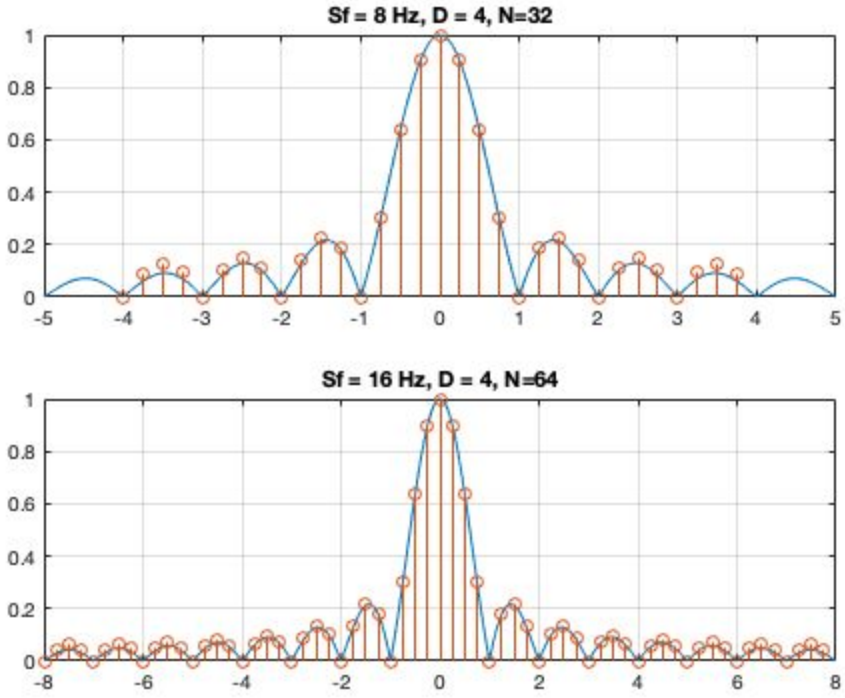


Figure B

Last modified. Sept. 20, 2020

The figures clearly illustrate the effect of duration, D, on the resolution or spectral spacing of the DFT. Clearly, a duration of 1 second does not yield a good sampled version of the spectrum of x(t). As D is increased, the resolution improves. The discrepancies between the DFT and the original signal spectrum illustrate the effect of aliasing. A higher sampling rate reduces the aliasing effect and produces a closer match between the DFT and the signal spectrum.

Your task for this part is to write the Matlab code that will generate the above figures Figure A and Figure B. The code is very similar to the ones given in the example, so you don't need to write the code from scratch.