Fast Fourier Transform and MATLAB Implementation

by
Wanjun Huang
for
Dr. Duncan L. MacFarlane

Signals

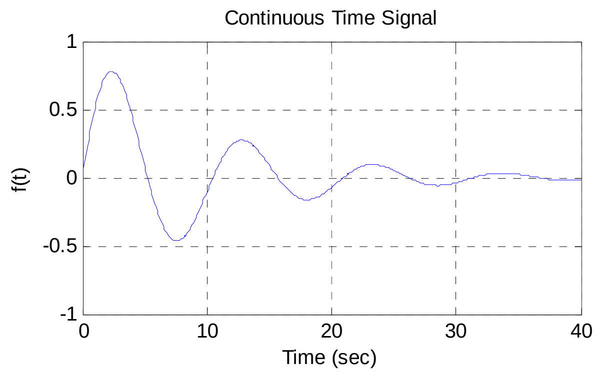
In the fields of communications, signal processing, and in electrical engineering more generally, a signal is any time-varying or spatial-varying quantity

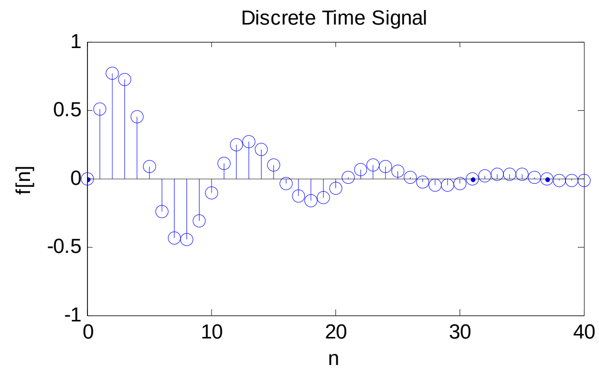
This variable(quantity) changes in time

- Speech or audio signal: A sound amplitude that varies in time
- Temperature readings at different hours of a day
- Stock price changes over days
- Etc.

Signals can be classified by continues-time signal and discrete-time signal:

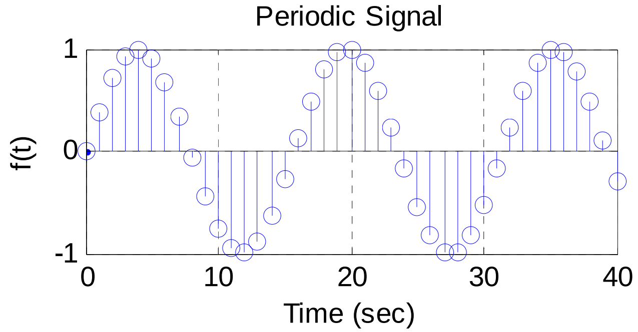
- A discrete signal or discrete-time signal is a time series, perhaps a signal that has been sampled from a continuous-time signal
- A digital signal is a discrete-time signal that takes on only a discrete set of values

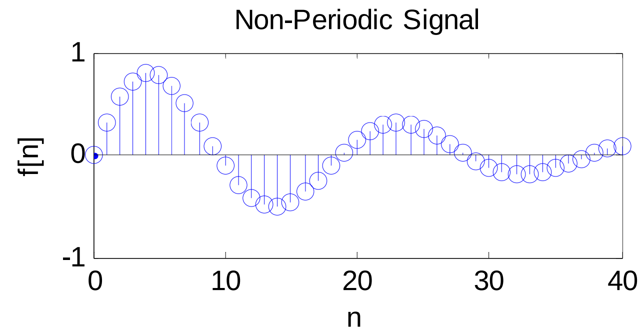




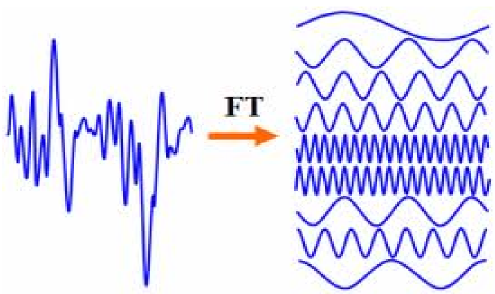
Periodic Signal

periodic signal and non-periodic signal:





- Period *T*: The minimum interval on which a signal repeats
- Fundamental frequency: $f_0=1/T$
- Harmonic frequencies: kf_0
- Any periodic signal can be approximated by a sum of many sinusoids at harmonic frequencies of the signal(kf_0) with appropriate amplitude and phase

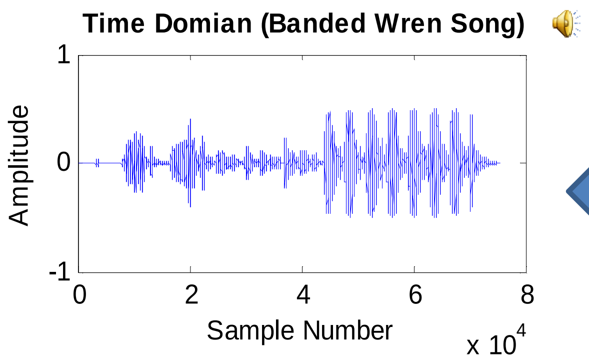


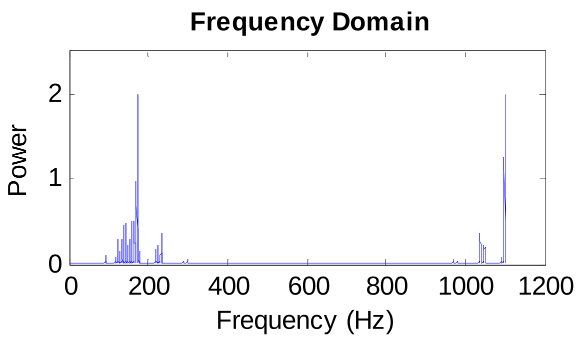
- Instead of using sinusoid signals, mathematically, we can use the complex exponential functions with both positive and negative harmonic frequencies

Euler formula: $\exp(j\omega t) = \sin(\omega t) + j\cos(\omega t)$

Time-Frequency Analysis

- A signal has one or more frequencies in it, and can be viewed from two different standpoints: *Time domain* and *Frequency domain*





Time-domain figure: how a signal changes over time

Frequency-domain figure: how much of the signal lies within each given frequency band over a range of frequencies

Why frequency domain analysis?

- To decompose a complex signal into simpler parts to facilitate analysis
- Differential and difference equations and convolution operations in the time domain become algebraic operations in the frequency domain
- Fast Algorithm (FFT)

Fourier Transform

We can go between the time domain and the frequency domain by using a tool called ***Fourier transform***

- A Fourier transform converts a signal in the time domain to the frequency domain(spectrum)
- An inverse Fourier transform converts the frequency domain components back into the original time domain signal

Continuous-Time Fourier Transform:

$$F(j\omega) = \int_{-\infty}^{+\infty} f(t) e^{-j\omega t} dt \qquad f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(j\omega) e^{j\omega t} d\omega$$

Discrete-Time Fourier Transform(DTFT):

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

Fourier Representation For Four Types of Signals

The signal with different time-domain characteristics has different frequency-domain characteristics

1. Continues-time periodic signal ---> discrete non-periodic spectrum
2. Continues-time non-periodic signal ---> continues non-periodic spectrum
3. Discrete non-periodic signal ---> continues periodic spectrum
4. Discrete periodic signal ---> discrete periodic spectrum

The last transformation between time-domain and frequency is most useful

The reason that discrete is associated with both time-domain and frequency domain is because computers can only take finite discrete time signals

Periodic Sequence

A periodic sequence with period *N* is defined as:

$$\tilde{x}(n) = \tilde{x}(n + kN)$$, where *k* is integer

For example: $W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$ (it is called *Twiddle Factor*)

Properties: Periodic $W_N^{kn} = W_N^{(k+N)n} = W_N^{k(n+N)}$

Symmetric $W_N^{-kn} = (W_N^{kn})^* = W_N^{(N-k)n} = W_N^{k(N-n)}$

Orthogonal $\sum_{k=0}^{N-1} W_N^{kn} = \begin{cases} N & n = rN \\ 0 & other \end{cases}$

For a periodic sequence $\tilde{x}(n)$ with period *N*, only *N* samples are independent. So that *N* sample in one period is enough to represent the whole sequence

Discrete Fourier Series(DFS)

Periodic signals may be expanded into a series of sine and cosine functions

$$\tilde{X}(k) = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{kn}$$

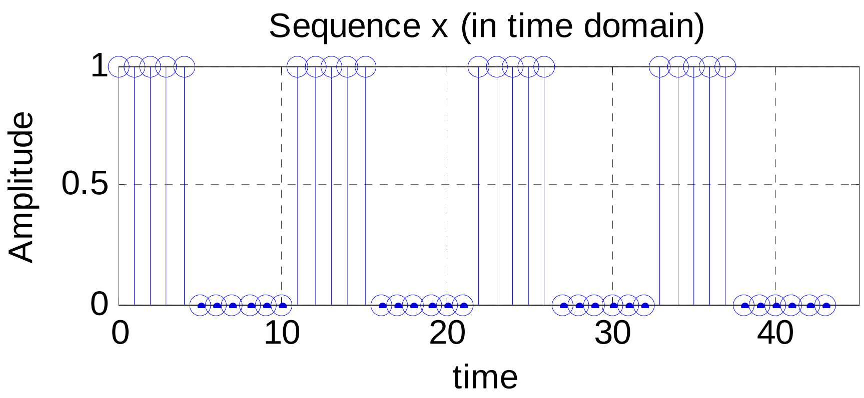
$$\tilde{x}(n) = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{X}(k) W_N^{-kn}$$

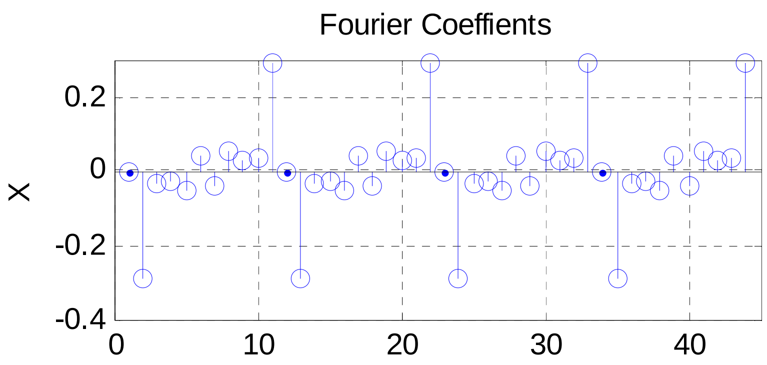
$$\tilde{X}(k) = DFS\ (\tilde{x}(n))$$

$$\tilde{x}(n) = IDFS\ (\tilde{X}(k))$$

$\tilde{X}(k)$ is still a periodic sequence with period *N* in frequency domain

The Fourier series for the discrete-time periodic wave shown below:





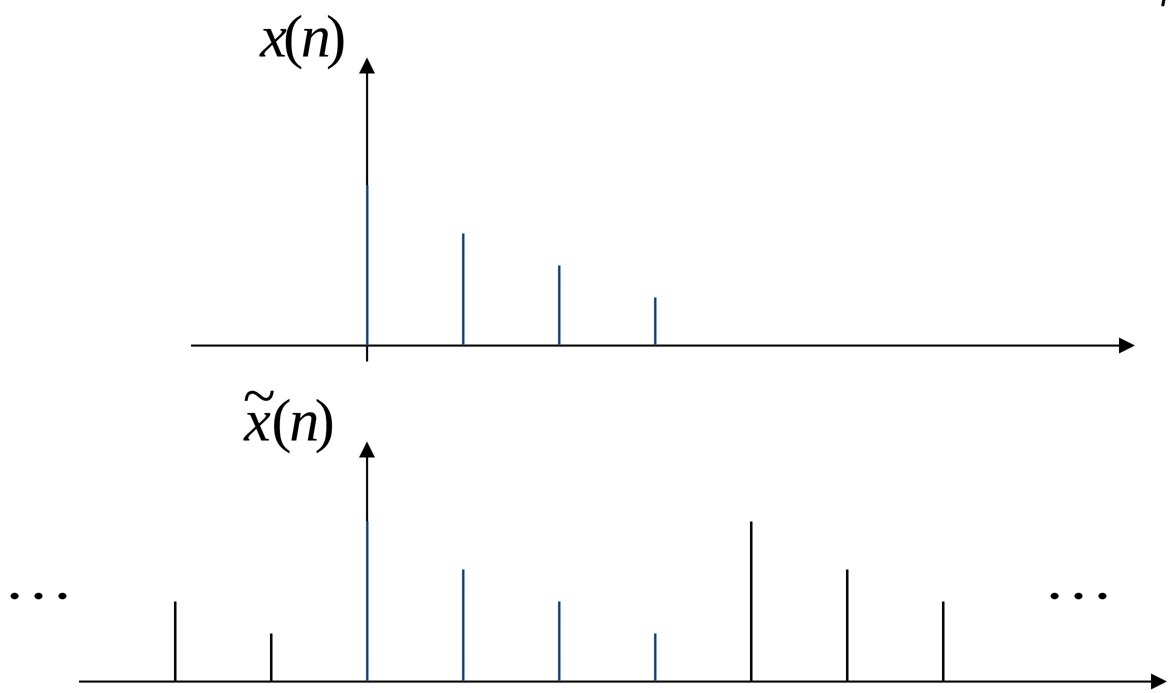
Finite Length Sequence

Real lift signal is generally a finite length sequence

$$x(n) = \begin{cases} x(n) & 0 \le n \le N-1 \\ 0 & others \end{cases}$$

If we periodic extend it by the period *N*, then

$$\tilde{x}(n) = \sum_{r=-\infty}^{\infty} x(n+rN)$$



Relationship Between Finite Length Sequence and Periodic Sequence

A periodic sequence is the periodic extension of a finite length sequence

$$\tilde{x}(n) = \sum_{m=-\infty}^{\infty} x(n + rN) = x((n))_N$$

A finite length sequence is the principal value interval of the periodic sequence

$$x(n) = \tilde{x}(n) R_N(n) \qquad \text{Where} \qquad R_N(n) = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & others \end{cases}$$

So that:

$$x(n) = \tilde{x}(n) R_N(n) = IDFS[\tilde{X}(k)] R_N(n)$$

$$X(k) = \tilde{X}(k) R_N(k) = DFS[\tilde{x}(n)] R_N(n)$$

Discrete Fourier Transform(DFT)

- Using the Fourier series representation we have Discrete Fourier Transform(DFT) for finite length signal
- DFT can convert time-domain discrete signal into frequency-domain discrete spectrum

Assume that we have a signal $\{x[n]\}_{n=0}^{N-1}$. Then the DFT of the signal is a sequence $X[k]$ for $k = 0, \cdots, N-1$

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi jnk/N}$$

The Inverse Discrete Fourier Transform(IDFT):

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{2\pi jnk/N}, n = 0, 2, \cdots, N-1.$$

Note that because MATLAB cannot use a zero or negative indices, the index starts from 1 in MATLAB

DFT Example

The DFT is widely used in the fields of spectral analysis, acoustics, medical imaging, and telecommunications.

For example:

$$x[n] = [2 \;\; 4 \;\; -1 \;\; 6], \quad N = 4, \; (n = 0,1,2,3)$$

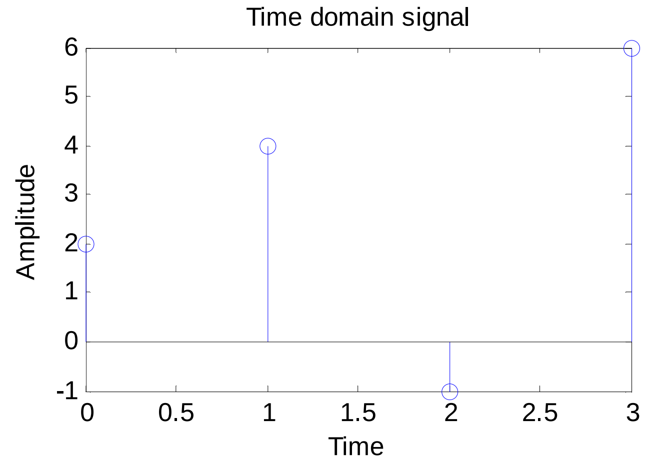
$$X[k] = \sum_{n=0}^{3} x[n] e^{-j\frac{\pi}{2}nk} = \sum_{n=0}^{3} x[n](-j)^{nk}$$

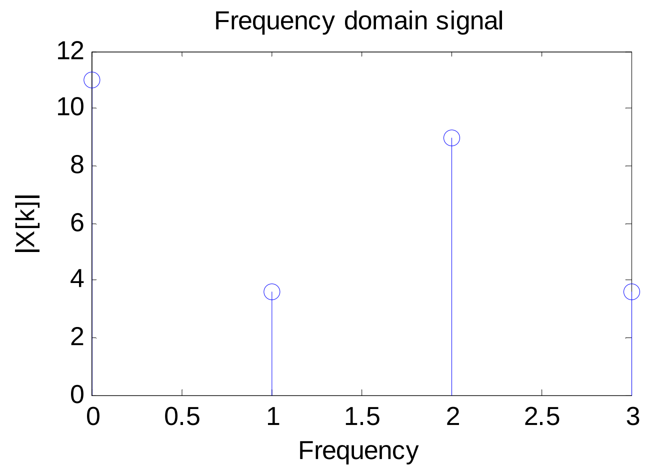
$$X[0] = 2 + 4 + (-1) + 6 = 11$$

$$X[1] = 2 + (-4j) + 1 + 6j = 3 + 2j$$

$$X[2] = 2 + (-4) + (-1) - 6 = -9$$

$$X[3] = 2 + (4j) + 1 - 6j = 3 - 2j$$





Fast Fourier Transform(FFT)

- The Fast Fourier Transform does not refer to a new or different type of Fourier transform. It refers to a very efficient algorithm for computing the DFT
- The time taken to evaluate a DFT on a computer depends principally on the number of multiplications involved. DFT needs N^2 multiplications. FFT only needs $N\log_2(N)$
- The central insight which leads to this algorithm is the realization that a discrete Fourier transform of a sequence of N points can be written in terms of two discrete Fourier transforms of length $N/2$
- Thus if N is a power of two, it is possible to recursively apply this decomposition until we are left with discrete Fourier transforms of single points

Fast Fourier Transform(cont.)

Re-writing $X[k]=\sum_{n=0}^{N-1} x[n]e^{-2\pi jnk/N}$ as $X[k]=\sum_{n=0}^{N-1} x[n]W_N^{nk}$

It is easy to realize that the same values of W_N^{nk} are calculated many times as the computation proceeds

Using the symmetric property of the *twiddle factor*, we can save lots of computations

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk} = \sum_{\substack{n=0\\ even\ n}}^{N-1} x(n)W_N^{kn} + \sum_{\substack{n=0\\ odd\ n}}^{N-1} x(n)W_N^{kn} \\
&= \sum_{r=0}^{N/2-1} x(2r)W_N^{2kr} + \sum_{r=0}^{N/2-1} x(2r+1)W_N^{k(2r+1)} \\
&= \sum_{r=0}^{N/2-1} x_1(r)W_{N/2}^{kr} + W_N^{k}\sum_{r=0}^{N/2-1} x_2(r)W_{N/2}^{kr} \\
&= X_1(k) + W_N^{k} X_2(k)
\end{aligned}
$$

Thus the *N*-point DFT can be obtained from two *N*/2-point transforms, one on even input data, and one on odd input data.

Introduction for MATLAB

MATLAB is a numerical computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, and implementation of algorithms

Getting help

You can get help by typing the commands `help` or `lookfor` at the >> prompt, e.g.

```
>> help fft
```

Arithmetic operators

Symbol	Operation	Example
+	Addition	3.1 + 9
-	Subtraction	6.2 – 5
*	Multiplication	2 * 3
/	Division	5 / 2
^	Power	3^2

Data Representations in MATLAB

Variables: Variables are defined as the assignment operator "=" . The syntax of variable assignment is

```
variable name = a value (or an expression)
```

For example,

```
>>  x = 5
x =
        5
>>  y = [3*7, pi/3];    % pi is  π  in MATLAB
```

Vectors/Matrices: MATLAB can create and manipulate arrays of 1 (vectors), 2 (matrices), or more dimensions

row vectors: a = [1, 2, 3, 4] is a 1X4 matrix

column vectors: b = [5; 6; 7; 8; 9] is a 5X1 matrix, e.g.

```
>> A = [1 2 3; 7 8 9; 4 5 6]
A  =   1 2 3
       7 8 9
       4 5 6
```

Mathematical Functions in MATLAB

MATLAB offers many predefined mathematical functions for technical computing, e.g.

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`angle(x)`	Phase angle
`exp(x)`	Exponential	`conj(x)`	Complex conjugate
`sqrt(x)`	Square root	`log(x)`	Natural logarithm

Colon operator (:)

Suppose we want to enter a vector *x* consisting of points (0,0.1,0.2,0.3,…,5). We can use the command

```
>> x = 0:0.1:5;
```

Most of the work you will do in MATLAB will be stored in files called *scripts*, or m-files, containing sequences of MATLAB commands to be executed over and over again

Basic plotting in MATLAB

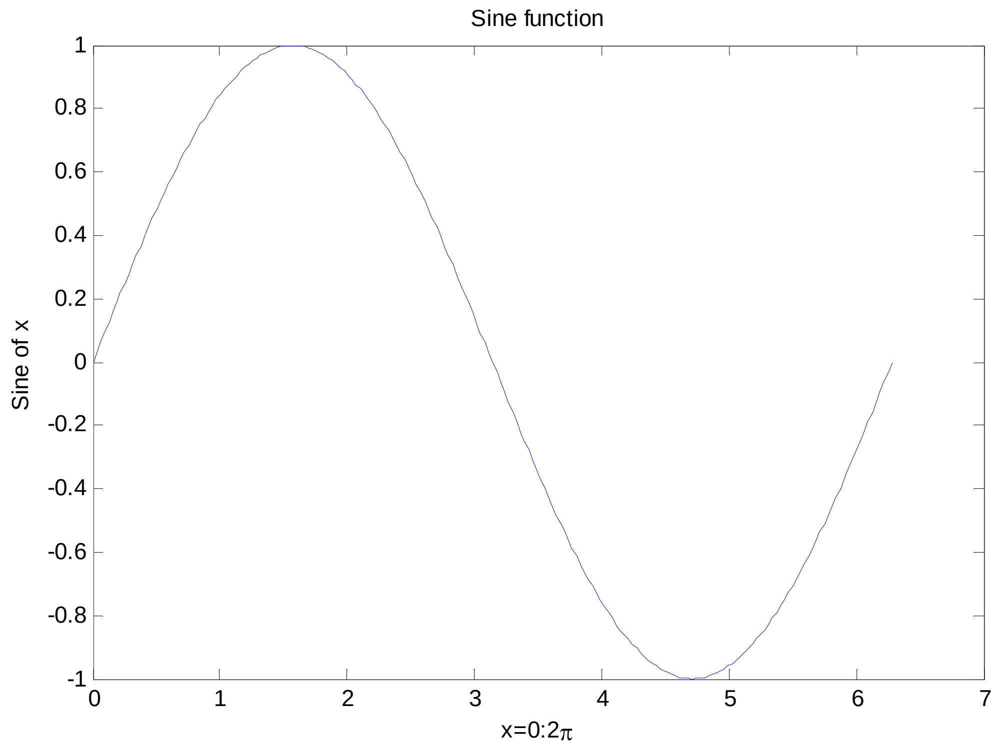
MATLAB has an excellent set of graphic tools. Plotting a given data set or the results of computation is possible with very few commands

The MATLAB command to plot a graph is plot(x,y), e.g.

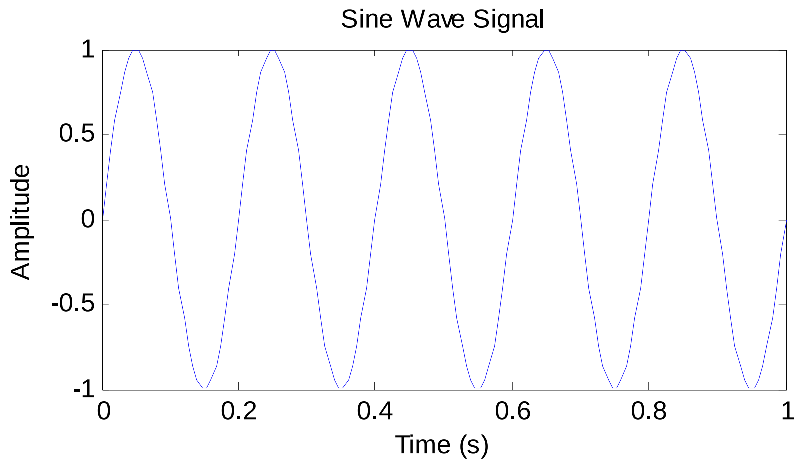
```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
```

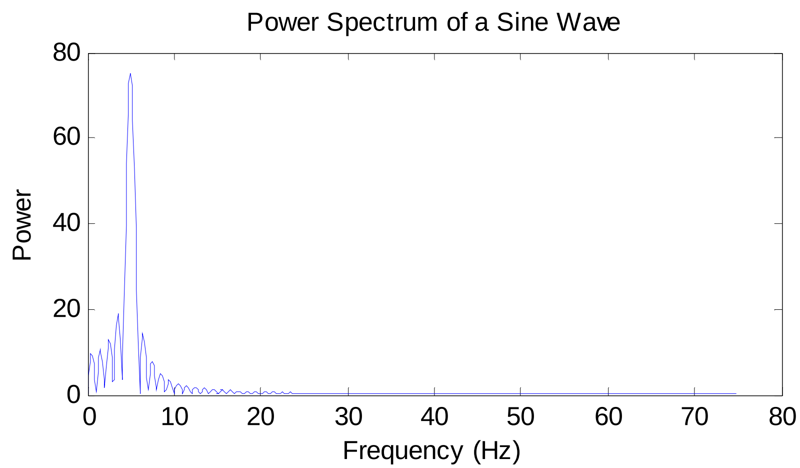
MATLAB enables you to add axis Labels and titles, e.g.

```
>> xlabel('x=0:2\pi');
>> ylabel('Sine of x');
>> tile('Sine function')
```



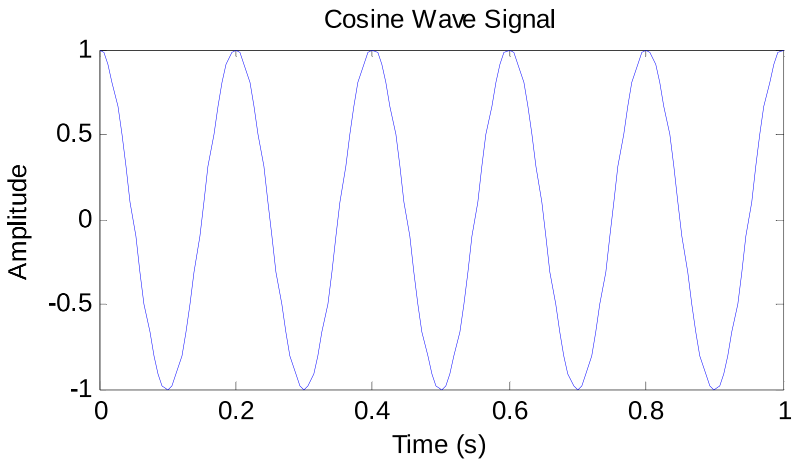
Example 1: Sine Wave

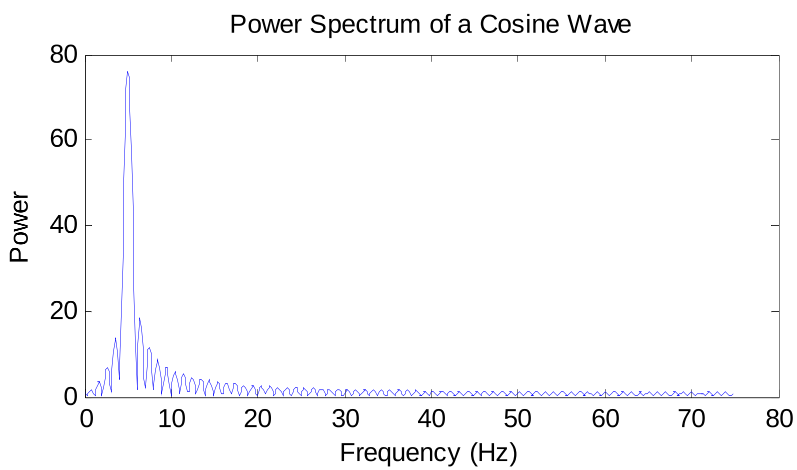





```
Fs = 150; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second
f = 5; % Create a sine wave of f Hz.
x = sin(2*pi*t*f);
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that length(X)
is equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Sine Wave Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Power');
```

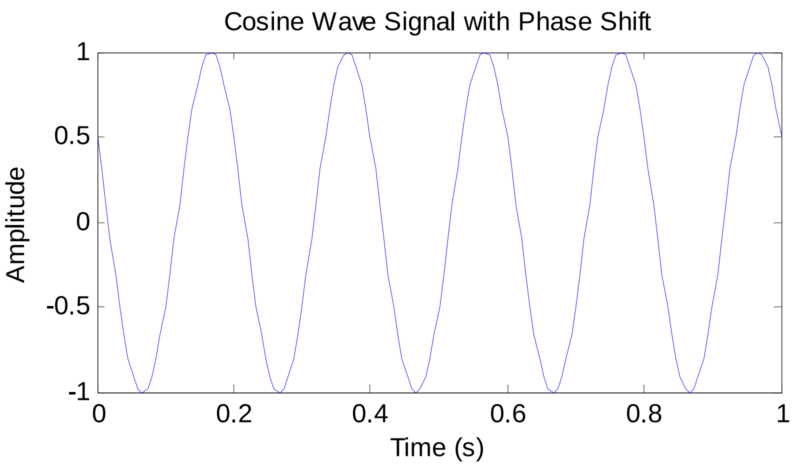
Example 2: Cosine Wave

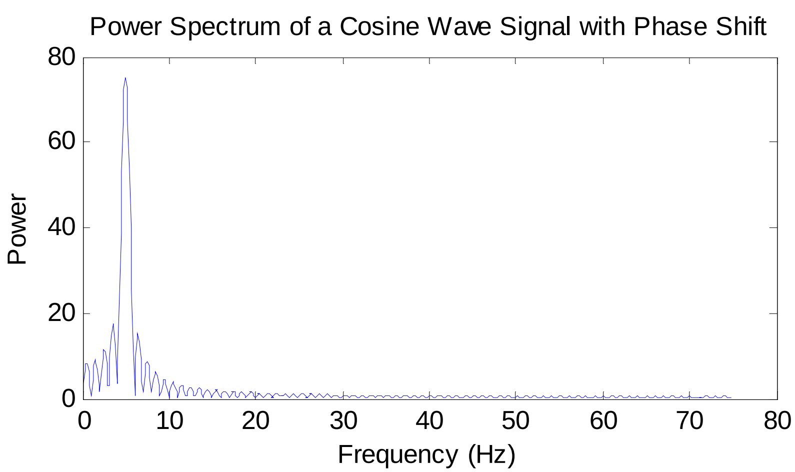





```
Fs = 150; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second
f = 5; % Create a sine wave of f Hz.
x = cos(2*pi*t*f);
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that length(X) is
equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Sine Wave Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Power');
```

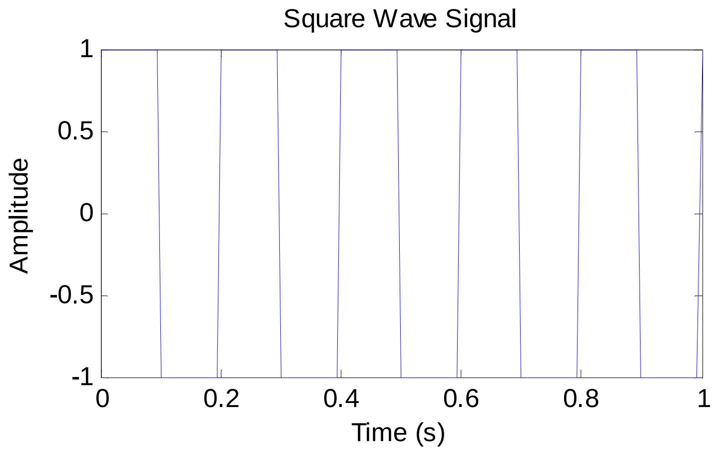
Example 3: Cosine Wave with Phase Shift

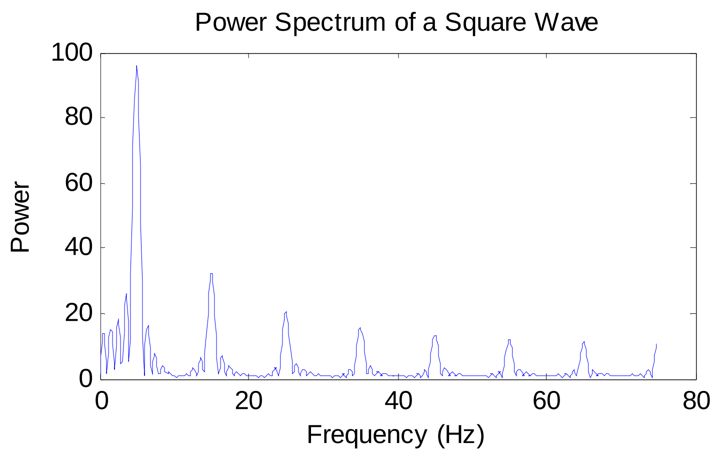





```
Fs = 150; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second
f = 5; % Create a sine wave of f Hz.
pha = 1/3*pi; % phase shift
x = cos(2*pi*t*f + pha);
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that length(X) is
equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Sine Wave Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Power');
```

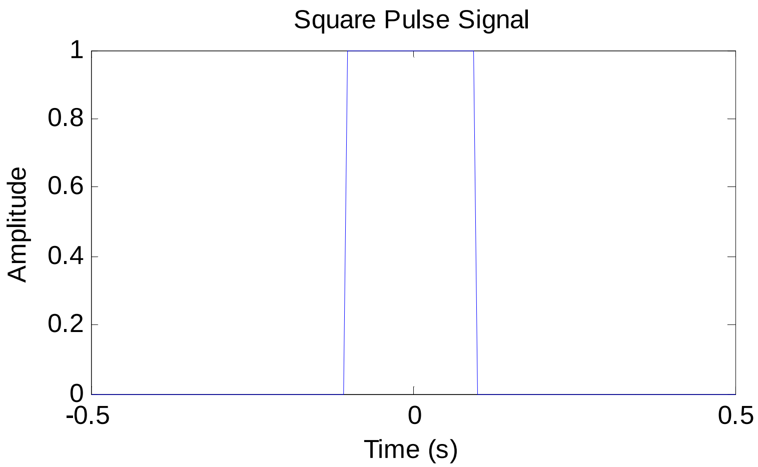
Example 4: Square Wave

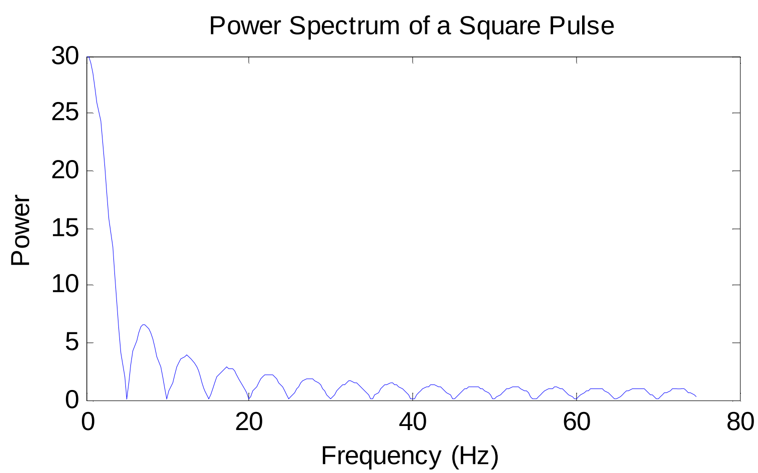





```
Fs = 150; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second
f = 5; % Create a sine wave of f Hz.
x = square(2*pi*t*f);
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that length(X) is
equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Square Wave Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Square Wave');
xlabel('Frequency (Hz)');
ylabel('Power');
```

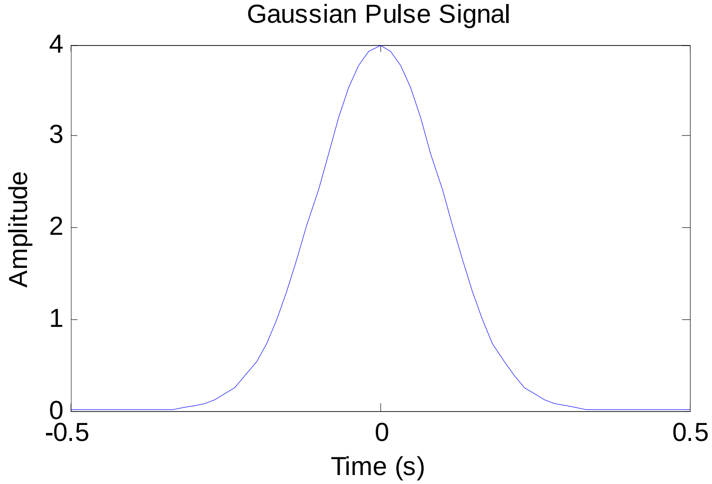
Example 5: Square Pulse

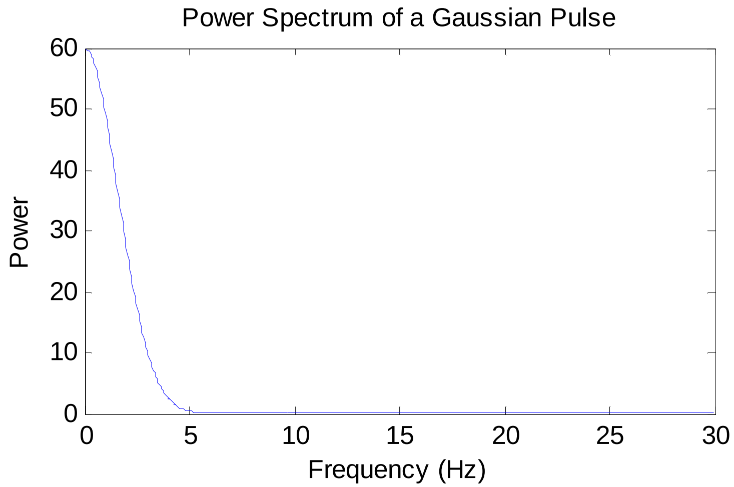





```
Fs = 150; % Sampling frequency
t = -0.5:1/Fs:0.5; % Time vector of 1 second
w = .2; % width of rectangle
x = rectpuls(t, w); % Generate Square Pulse
nfft = 512; % Length of FFT
% Take fft, padding with zeros so that length(X) is
equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Square Pulse Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Square Pulse');
xlabel('Frequency (Hz)');
ylabel('Power');
```

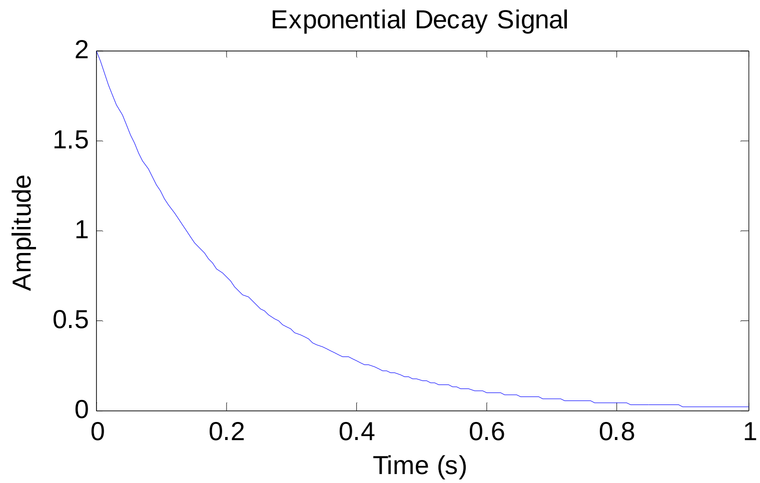
Example 6: Gaussian Pulse

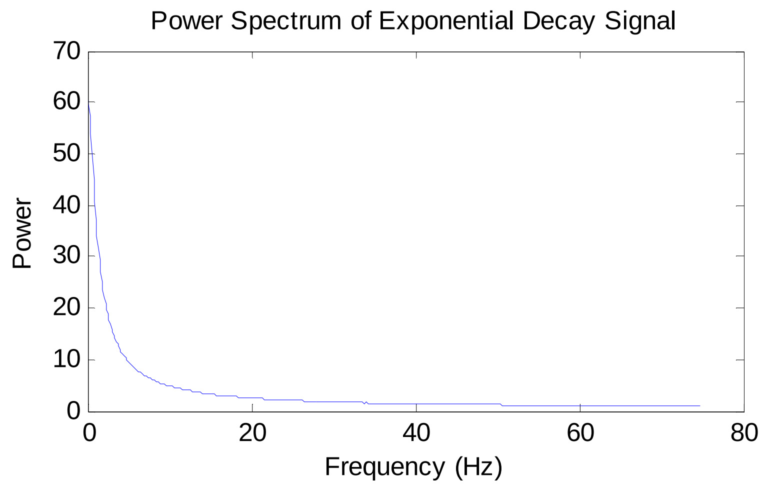





```
Fs = 60; % Sampling frequency
t = -.5:1/Fs:.5;
x = 1/(sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that
length(X) is equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% This is an evenly spaced frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Gaussian Pulse Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Gaussian Pulse');
xlabel('Frequency (Hz)');
ylabel('Power');
```

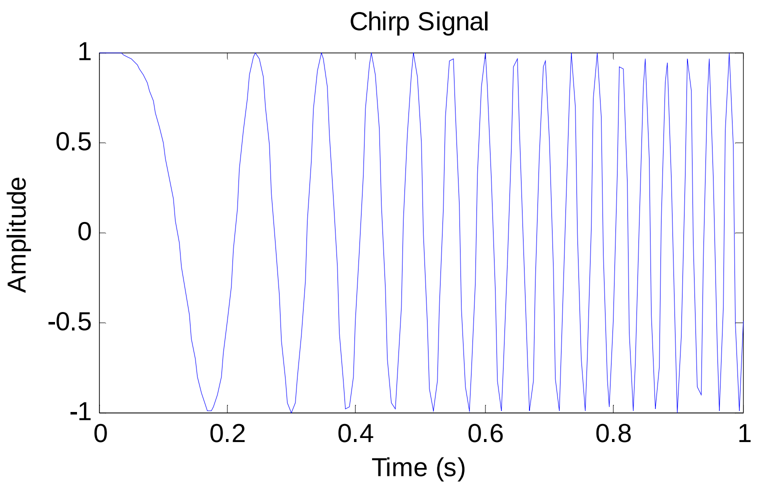
Example 7: Exponential Decay

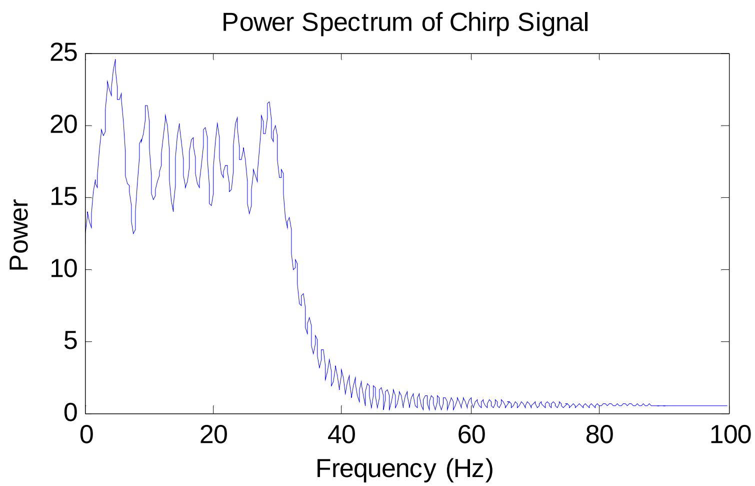





```
Fs = 150; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second
x = 2*exp(-5*t);
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that
length(X) is equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second
half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% This is an evenly spaced frequency
vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Exponential Decay Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of Exponential
Decay Signal');
xlabel('Frequency (Hz)');
ylabel('Power');
```

Example 8: Chirp Signal






```
Fs = 200; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second
x = chirp(t,0,1,Fs/6);
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that
length(X) is equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% This is an evenly spaced frequency
vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Chirp Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of Chirp Signal');
xlabel('Frequency (Hz)');
ylabel('Power');
```