Plot FFT using Matlab – FFT of sine wave & cosine wave

July 16, 2014 by Mathuranathan

Key focus: Learn how to plot FFT of sine wave and cosine wave using Matlab. Understand FFTshift. Plot one-sided, double-sided and normalized spectrum.

Introduction

Numerous texts are available to explain the basics of Discrete Fourier Transform and its very efficient implementation – Fast Fourier Transform (FFT). Often we are confronted with the need to generate simple, standard signals (sine, cosine, Gaussian pulse, squarewave, isolated rectangular pulse, exponential decay, chirp signal) for simulation purpose. I intend to show (in a series of articles) how these basic signals can be generated in Matlab and how to represent them in frequency domain using FFT. If you are inclined towards Python programming, visit here.

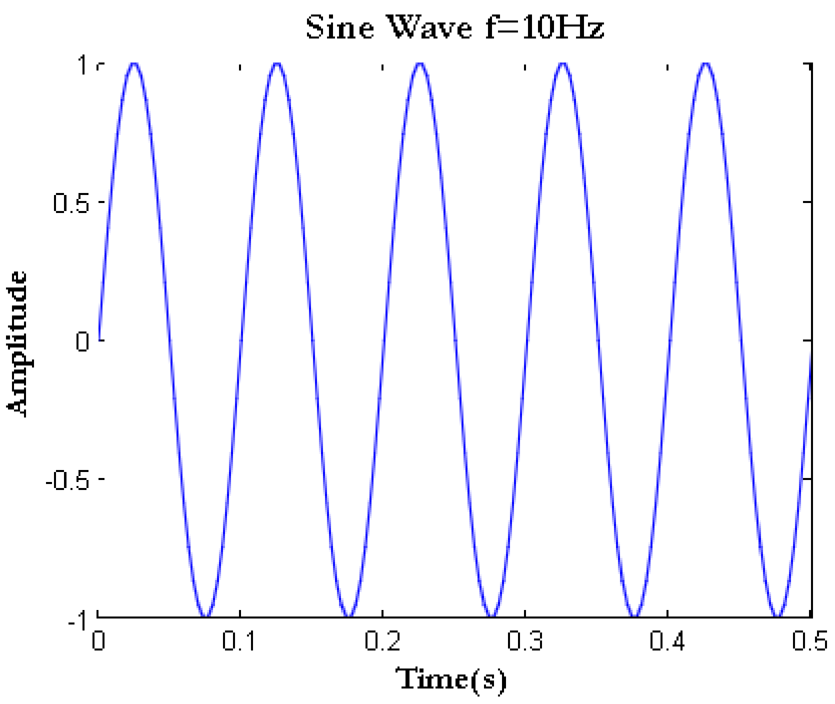
Sine Wave

In order to generate a sine wave in Matlab, the first step is to fix the frequency f of the sine wave. For example, I intend to generate a ***f=10 Hz*** sine wave whose minimum and maximum amplitudes are $-1V$ and $+1V$ respectively. Now that you have determined the frequency of the sinewave, the next step is to determine the sampling rate. Matlab is a software that processes everything in digital. In order to generate/plot a smooth sine wave, the sampling rate must be far higher than the prescribed minimum required sampling rate which is at least twice the frequency f – as per Nyquist Shannon Theorem. A oversampling factor of 30 is chosen here – this is to plot a smooth continuous-like sine wave (If this is not the requirement, reduce the oversampling factor to desired level). Thus the sampling rate becomes $f_s = 30 \times f = 300Hz$. If a phase shift is desired for the sine wave, specify it too.

```
f=10; %frequency of sine wave
overSampRate=30; %oversampling rate
fs=overSampRate*f; %sampling frequency
phase = 1/3*pi; %desired phase shift in radians
nCyl = 5; %to generate five cycles of sine wave

t=0:1/fs:nCyl*1/f; %time base

x=sin(2*pi*f*t+phase); %replace with cos if a cosine wave is desired
plot(t,x);
title(['Sine Wave f=', num2str(f), 'Hz']);
xlabel('Time(s)');
ylabel('Amplitude');
```



Representing in Frequency Domain

Representing the given signal in frequency domain is done via Fast Fourier Transform (FFT) which implements Discrete Fourier Transform (DFT) in an efficient manner. Usually, power spectrum is desired for analysis in frequency domain. In a power spectrum, power of each frequency component of the given signal is plotted against their respective frequency. The command $FFT(x, N)$ computes the N-point DFT. The number of points – N – in the DFT computation is taken as power of (2) for facilitating efficient computation with FFT. A value of $N = 1024$ is chosen here. It can also be chosen as next power of 2 of the length of the signal.

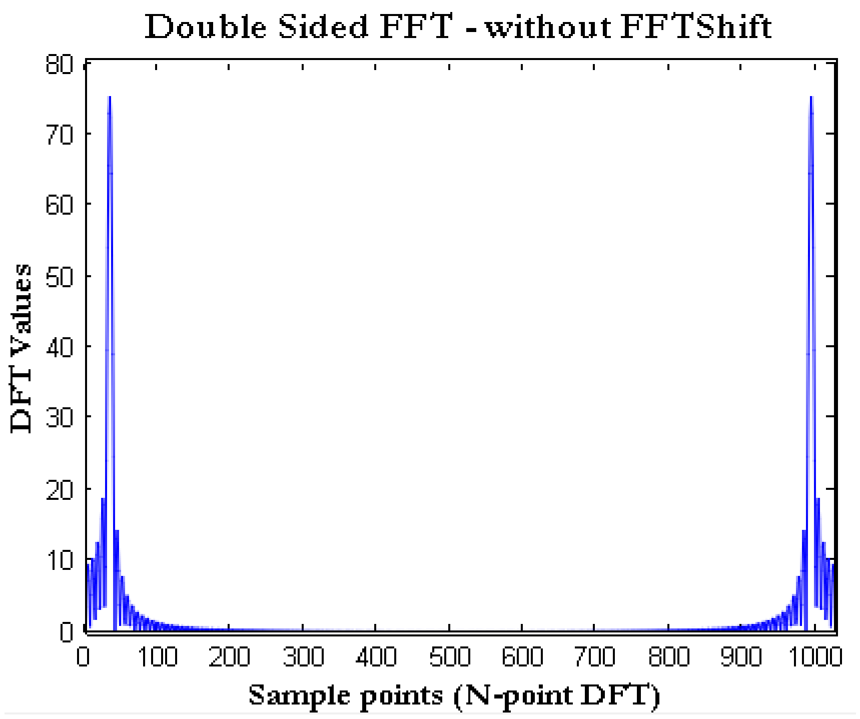
Different representations of FFT:

Since FFT is just a numeric computation of N-point DFT, there are many ways to plot the result.

1. Plotting raw values of DFT:

The x-axis runs from 0 to $N-1$ – representing N sample values. Since the DFT values are complex, the magnitude of the DFT $abs(X)$ is plotted on the y-axis. From this plot we cannot identify the frequency of the sinusoid that was generated.

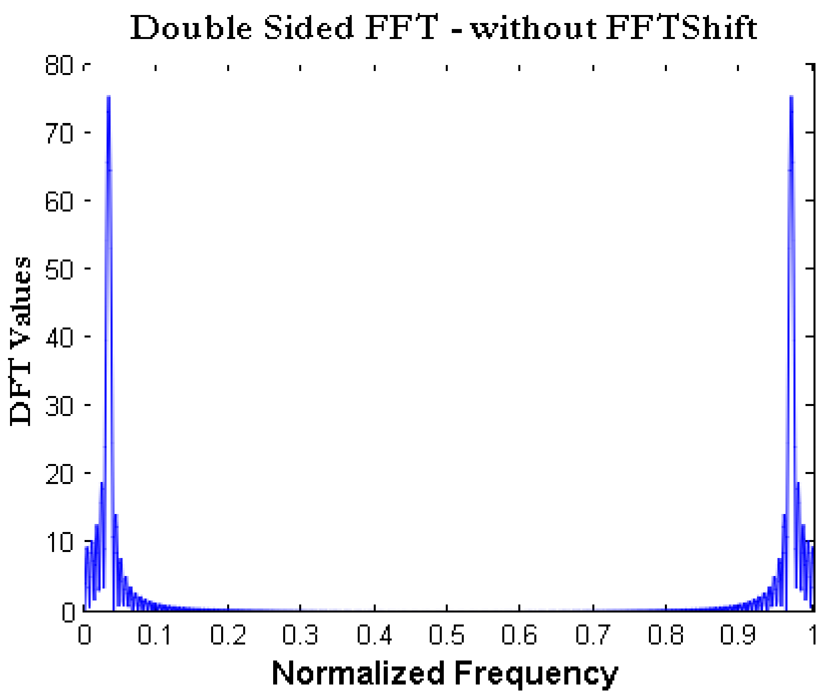
```
NFFT=1024; %NFFT-point DFT
X=fft(x,NFFT); %compute DFT using FFT
nVals=0:NFFT-1; %DFT Sample points
plot(nVals,abs(X));
title('Double Sided FFT - without FFTShift');
xlabel('Sample points (N-point DFT)')
ylabel('DFT Values');
```



2. FFT plot – plotting raw values against Normalized Frequency axis:

In the next version of plot, the frequency axis (x-axis) is normalized to unity. Just divide the sample index on the x-axis by the length N of the FFT. This normalizes the x-axis with respect to the sampling rate f_s. Still, we cannot figure out the frequency of the sinusoid from the plot.

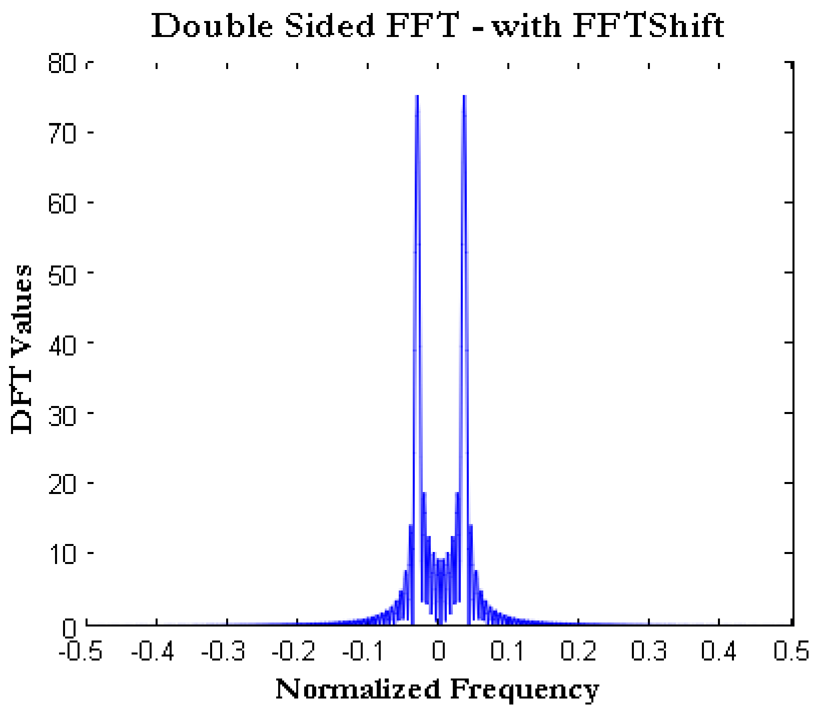
```
NFFT=1024; %NFFT-point DFT
X=fft(x,NFFT); %compute DFT using FFT
nVals=(0:NFFT-1)/NFFT; %Normalized DFT Sample points
plot(nVals,abs(X));
title('Double Sided FFT - without FFTShift');
xlabel('Normalized Frequency')
ylabel('DFT Values');
```



3. FFT plot – plotting raw values against normalized frequency (positive & negative frequencies):

As you know, in the frequency domain, the values take up both positive and negative frequency axis. In order to plot the DFT values on a frequency axis with both positive and negative values, the DFT value at sample index 0 has to be centered at the middle of the array. This is done by using $FFTshift$ function in Matlab. The x-axis runs from -0.5 to 0.5 where the end points are the normalized 'folding frequencies' with respect to the sampling rate f_s.

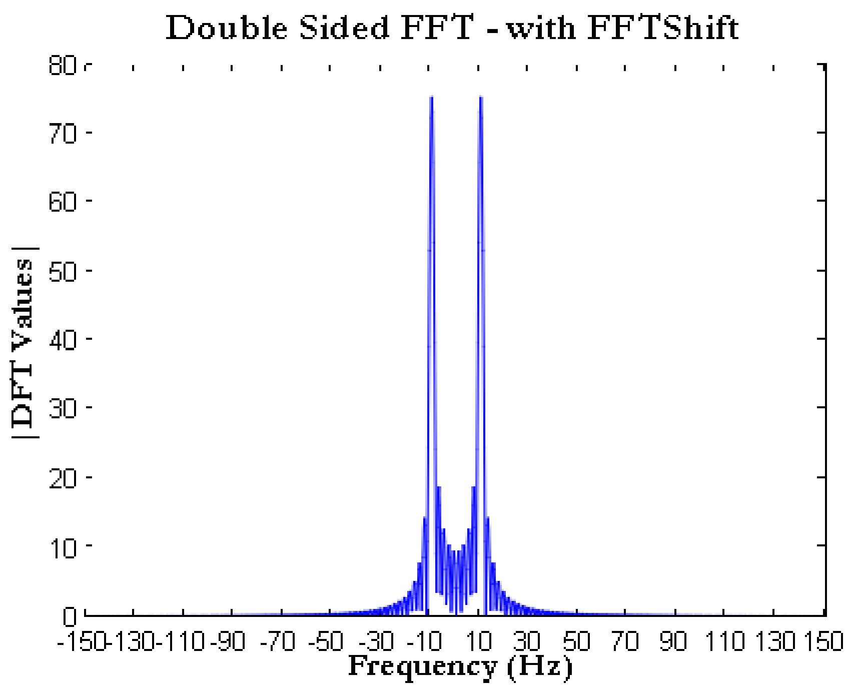
```
NFFT=1024; %NFFT-point DFT
X=fftshift(fft(x,NFFT)); %compute DFT using FFT
fVals=(-NFFT/2:NFFT/2-1)/NFFT; %DFT Sample points
plot(fVals,abs(X));
title('Double Sided FFT - with FFTShift');
xlabel('Normalized Frequency')
ylabel('DFT Values');
```



4. FFT plot – Absolute frequency on the x-axis Vs Magnitude on Y-axis:

Here, the normalized frequency axis is just multiplied by the sampling rate. From the plot below we can ascertain that the absolute value of FFT peaks at $10Hz$ and $-10Hz$. Thus the frequency of the generated sinusoid is $10Hz$. The small side-lobes next to the peak values at $10Hz$ and $-10Hz$ are due to spectral leakage.

```
NFFT=1024;
X=fftshift(fft(x,NFFT));
fVals=fs*(-NFFT/2:NFFT/2-1)/NFFT;
plot(fVals,abs(X),'b');
title('Double Sided FFT - with FFTShift');
xlabel('Frequency (Hz)')
ylabel('|DFT Values|');
```



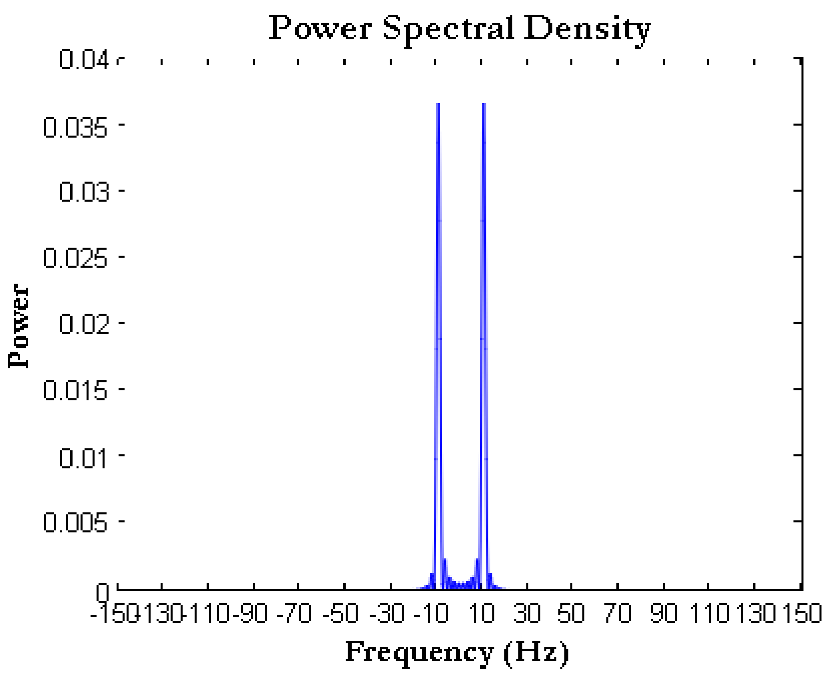
5. Power Spectrum – Absolute frequency on the x-axis Vs Power on Y-axis:

The following is the most important representation of FFT. It plots the power of each frequency component on the y-axis and the frequency on the x-axis. The power can be plotted in linear scale or in log scale. The power of each frequency component is calculated as

$P_x(f) = X(f)X^*(f)$

Where $X(f)$ is the frequency domain representation of the signal $x(t)$. In Matlab, the power has to be calculated with proper scaling terms (since the length of the signal and transform length of FFT may differ from case to case).

```
NFFT=1024;
L=length(x);
X=fftshift(fft(x,NFFT));
Px=X.*conj(X)/(NFFT*L); %Power of each freq components
fVals=fs*(-NFFT/2:NFFT/2-1)/NFFT;
plot(fVals,Px,'b');
title('Power Spectral Density');
xlabel('Frequency (Hz)')
ylabel('Power');
```



If you wish to verify the total power of the signal from time domain and frequency domain plots, follow this link.
Plotting the power spectral density (PSD) plot with y-axis on log scale, produces the most encountered type of PSD plot in signal processing.

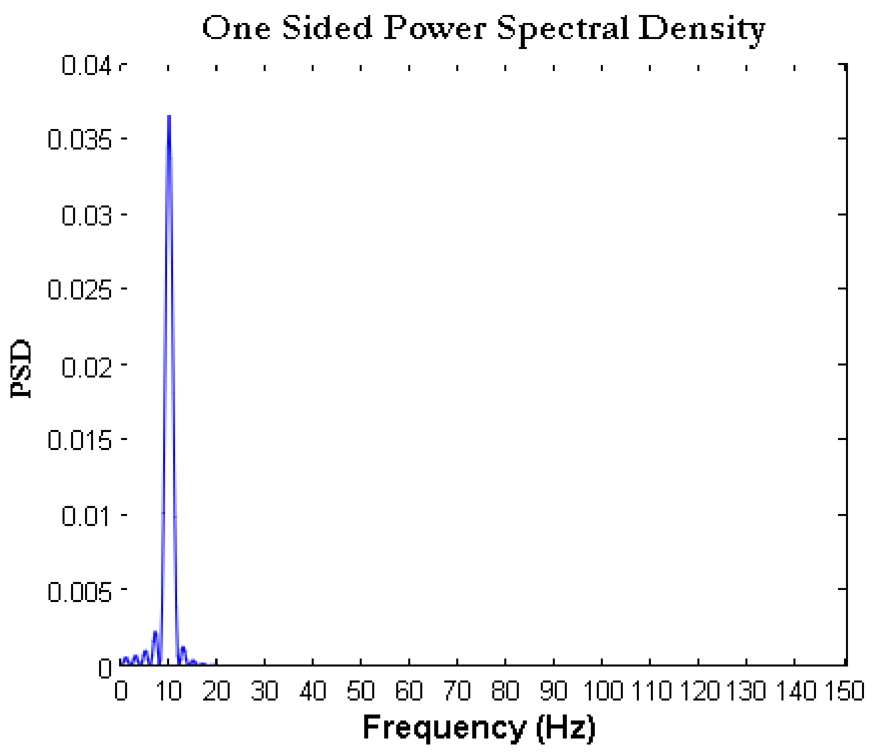
```
NFFT=1024;
L=length(x);
X=fftshift(fft(x,NFFT));
Px=X.*conj(X)/(NFFT*L); %Power of each freq components
fVals=fs*(-NFFT/2:NFFT/2-1)/NFFT;
plot(fVals,10*log10(Px),'b');
title('Power Spectral Density');
xlabel('Frequency (Hz)')
ylabel('Power');
```

6. Power Spectrum – One-Sided frequencies

In this type of plot, the negative frequency part of x-axis is omitted. Only the FFT values corresponding to 0 to $N/2$ sample points of N-point DFT are plotted. Correspondingly, the normalized frequency axis runs between 0 to 0.5. The absolute frequency (x-axis) runs from 0 to $f_s/2$.

```
L=length(x);
NFFT=1024;
X=fft(x,NFFT);
Px=X.*conj(X)/(NFFT*L); %Power of each freq components
fVals=fs*(0:NFFT/2-1)/NFFT;
```

```
plot(fVals,Px(1:NFFT/2),'b','LineSmoothing','on','LineWidth',1);
title('One Sided Power Spectral Density');
xlabel('Frequency (Hz)')
ylabel('PSD');
```



Rate this article: (**117** votes, average: **4.67** out of 5)

For further reading

[1] Power spectral density – MIT opencourse ware

Topics in this chapter

Books by the author

Latest Articles, Matlab Codes, Signal Processing, Tips & Tricks
FFT, Fourier Analysis, Fourier transform, Matlab Code, Power spectral Density, PSD, Tips & Tricks

10 thoughts on “Plot FFT using Matlab – FFT of sine wave & cosine wave”

Vamsi Krishna Reddy Anyam
March 7, 2022 at 7:48 pm

Thank you for such an elaborate explanation.

Reply

Muna Shehan
March 31, 2017 at 10:08 am

https://uploads.disquscdn.com/images/d7f7499d564689bb015bdbbb0b86a039eff7d59c75d957cb5453db12b75e4543.jpg Hi all; this article for a sine wave signal. what can i do if the signal is still sine wave with five cycles where each cycle amplitude is less than the previous cycle until the wave is decay at a specific time, and if the x value is represented by a vector which contain the data (I mean x is not represented by equation). My question how i can implemented fft for my signal.
Thanks in advance

Reply

Mathuranathan
March 31, 2017 at 3:17 pm

Just assign the signal contents to the variable x and the call the FFT routine as mentioned in the post
X=fft(x,NFFT); %x is your signal,

you can use any of the different representations to plot the FFT output.

Reply

Mohand
May 27, 2016 at 6:19 pm

Hi everybody,

I need to do fft to obtain the frequency content of my signal and calculate the energy to obtain the transmission coefficient, but I have a lot of problems since I don't know exatly what is the correct way to obtain everything with a correct scale and units.

I have an output signal (voltage vs time : size(u)=(100000,2)) obtained from excitation: pulse (pulse width = 100 µs); on the oscilloscope : duration is 10 ms and 10 MSa/s of sampling rate.

for the frequencies, I expect to get a fundamental at around 8 KHz and the harmonics (2nd and third).

this is a very smal part of my signal :
t(s) t(volts)
3.99000000E-02 1.3574E-02
3.99001000E-02 1.4299E-02
3.99002000E-02 1.3969E-02
3.99003000E-02 1.3508E-02
3.99004000E-02 1.3706E-02

My questions are:

I need the values of the frequencies pics (fundamental, 2nd and 3rd) and calculate the energy of power in order to get the transmission coefficient of the signal.

1 – how should I proceed to get the frequencies in a correct scale? (plot of all the Spectrum in linear scale – Spectrum in dB scale)

2 – is it necessary to include a window in the fft ? which is the better one in that case?

P.S: I tried following the method explained in this website; here is my matlab code!!!!

```
u=load('u10.tsv');
tau = 0.000000099 ;

Nfft=length(u);
w=blackman(Nfft);
u_windowed_temp=u;
u_windowed=u.*w;

Y1 = fft(u_windowed,Nfft);
%freq1(1)=10^-10;
freq1(2:Nfft/2)=(1:Nfft/2-1)/(Nfft*tau);
plot(freq1/1000,abs(Y1(1:length(freq1))))
```

thank's a lot!

Reply

Yamuna

December 25, 2015 at 1:20 pm

Hi Sir,

In the above example when I gave the number of cycles as 10, then it is giving a spectrum which has a higher amplitude. This seems so strange to me because the number of cycles is just a visualisation parameter and that should not chance the appearance of the spectrum.

```
nCyl = 10; %to generate ten cycles of sine wave
```

Kindly explain why this is happening.

Reply

Mathuranathan
December 26, 2015 at 9:02 pm

I was plotting the NON-normalized magnitude spectrum. The amplitude is influenced the number of samples taken to compute DFT. For normalized magnitude spectrum please check the following post.
https://www.gaussianwaves.com/2015/11/interpreting-fft-results-obtaining-magnitude-and-phase-information

Reply

ArrozConCostra
August 9, 2015 at 9:08 pm

Thanks very much, very explanatory article

There's an error on point 3. It should read
X=fftshift(x,NFFT); %compute DFT using FFT

Rgds

Reply

Mathuranathan
December 26, 2015 at 9:42 pm

Thanks for pointing it out

Reply

Antonis Tsiflikiotis
June 4, 2015 at 8:03 pm

Hi!
Can you plean explain me why in Psd:
Px=X.*conj(X)/(NFFT*L)

you divide by (NFFT*L) and not L^2?

Reply

ArrozConCostra
August 9, 2015 at 9:15 pm

I would ask why he does not divide only by L, as this is the usual scaling factor when calculating FFT

Reply