Computational Data analysis of Fourıer Transformatıon by Numerical experiments(Numerical CODE)

Tadesse Lamessa
College of Natural scince and Computational scince
Department of Applied Mathematics

Woalita sodo university,

sodo Ethiopia,P.O.BOX .138

Abstract
The Fourier series (FS) and the Discrete Fourier Transform (DFT) should be thought of as playing similar roles for periodic signals in either continuous time (FS) or discrete time (DFT). Both analyze signals into amplitude, phases, and frequencies of complex exponentials; both synthesize signals by linearly combining complex exponentials with appropriate amplitude, phase, and frequency. Finally, both transforms have aspects that are extremely important to remember and other aspects that are important, but can be adjusted as necessary. As we work through some of the details, we'll identify these very important and the not so important aspects. Frequency analysis is one of the key issues in the IEEE Society. Using computers in numerical calculations means moving into a non-physical, synthetic environment. Numerically, discrete or fast Fourier transformations (DFT or FFT) are used to obtain the frequency contents of a time signal and these are totally different than mathematical definition of the Fourier transform. This article simple reviews DFT and FFT with characteristic examples.

***Phrase**: Numerically, frequency, Fourier series ,numerical code*

1. Introduction

The analysis of real world signals is a fundamental problem for many engineers and scientists, especially for electrical engineers since almost every real world signal is changed into electrical signals by means of transducers, e.g., accelerometers in mechanical engineering, EEG electrodes and blood pressure probes in biomedical engineering, seismic transducers in Earth Sciences, antennas in electromagnetics, and microphones in communication engineering, etc.

Traditional way of observing and analyzing signals is to view them in time domain. Baron Jean Baptiste Fourier [1], more than a century ago, showed that any waveform that exists in the real world can be represented (i.e., generated) by adding up sine waves. Since then, we have been able to build (break down) our real world time signal in terms of (into) these sine waves. It is shown that the combination of sine waves is unique; any real world signal can be represented by only one combination of sine waves [2].

The Fourier transform (FT) has been widely used in circuit analysis and synthesis, from filter design to signal processing, image reconstruction, stochastic modeling to non-destructive measurements. The FT has also been widely used in electromagnetics from antenna theory to radiowave propagation modeling, radar cross-section prediction to multi-sensor system system design. For example, the split-step parabolic equation method (which is nothing but the beam propagation method in optics) has been in use more than decades and is based on sequential FT operations between the spatial and wavenumber domains. Two and three dimensional propagation problems with non-flat realistic terrain profiles and inhomogeneous atmospheric variations above have been solved with this method successfully [3-5].

2. Fourier Transformation

The principle of a transform in engineering is to find a different representation of a signal under investigation. The FT is the most important transform widely used in electrical and computer (EC) engineering.

2.1 Fourier transform

The transformation from the time domain to the frequency domain (and back again) is based on the Fourier transform and its inverse, which are defined as

$$S(\omega) = \int_{-\infty}^{\infty} s(t) e^{-j2\pi f t} dt \tag{1a}$$

$$s(t) = \int_{-\infty}^{\infty} S(f) e^{j2\pi f t} df \tag{1b}$$

Here, *s(t), S(ω), and f* are the time signal, the frequency signal and the frequency, respectively, and $j = \sqrt{-1}$. We, the physicists and engineers, sometimes prefer to write the transform in terms of angular frequency $\omega = 2\pi f$, *as*

$$S(\omega) = \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt \tag{2a}$$

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{j\omega t} d\omega \tag{2b}$$

which, however, destroys the symmetry. To restore the symmetry of the transforms, the convention

$$S(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt \tag{3a}$$

$$s(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} S(\omega) e^{j\omega t} d\omega \tag{3b}$$

is sometimes used. The FT is valid for real or complex signals, and, in general, is a complex function of ω (or *f*).

The FT is valid for both periodic and non-periodic time signals that satisfy certain minimum conditions. Almost all real world signals easily satisfy these requirements (It should be noted that the Fourier series is a special case of the FT). Mathematically,

- FT is defined for continuous time signals.
- In order to do frequency analysis, the time signal must be observed infinitely.

Under these conditions, the FT defined above yields frequency behavior of a time signal at every frequency, with zero frequency resolution. Some functions and their FT are listed in Table 1.

***Table 1:** Some functions and their Fourier transforms*

Time domain	Fourier domain
Rectangular window	Sinc function
Sinc function	Rectangular window
Constant function	Dirac Delta function
Dirac Delta function	Constant function
Dirac comb (Dirac train)	Dirac comb (Dirac train)
Cosine function	Two, real, even Delta function
Sine function	Two, imaginary, odd Delta function
Exp function -$\{j\exp\{j\omega t\}\}$	One , positive, real Delta function
Gauss function	Gauss function

2.2 Discrete Fourier transforms (DFT)

In the second lecture we covered the Fourier transform of continuous functions but when we work with digital data, functions are sampled at discrete points which we will assume are uniformly spaced (i.e. at a time interval of Δt for time series or Δx for spatial data).

Discrete Fourier Transform

The Fourier series of a periodic function can be written in terms of complex exponentials as

$$y(t)=\sum_{k=-\infty}^{\infty} c_k \exp\left(\frac{ik2\pi t}{T}\right) \tag{4-1}$$

with

$$c_k=\frac{1}{T}\int_{-T/2}^{T/2} y(t)\exp\left(\frac{-ik2\pi t}{T}\right)dt \tag{4-2}$$

To construct the discrete Fourier Transform, we will replace y(t) by a discrete representation $y_j = 0, 1, 2, \ldots, N-2, N-1$ with the sample interval Δt related to the period T by $N\Delta t = T$. Equation (4-2) becomes a discrete Fourier transform (DFT)

$$c_k=\frac{1}{N\Delta t}\sum_{j=0}^{N-1} y_j\exp\left(-ik\frac{2\pi}{N\Delta t}j\Delta t\right)\Delta t=\frac{1}{N}\sum_{j=0}^{N-1} y_j\exp\left(-i\frac{2\pi}{N}jk\right) \tag{4-3}$$

Now consider c_{k+N}

$$\begin{aligned} c_{k+N} &= \frac{1}{N}\sum_{j=0}^{N-1} y_j\exp\left(-i(k+N)\frac{2\pi}{N}j\right)\\ &= \frac{1}{N}\sum_{j=0}^{N-1} y_j\exp\left(-ik\frac{2\pi}{N}j\right)\exp\left(-iN\frac{2\pi}{N}j\right)\\ &= \frac{1}{N}\sum_{j=0}^{N-1} y_j\exp\left(-i\frac{2\pi}{N}jk\right)\exp(-i2\pi j)\\ &= \frac{1}{N}\sum_{j=0}^{N-1} y_j\exp\left(-i\frac{2\pi}{N}jk\right)=c_k \end{aligned} \tag{4-4}$$

where we have used the fact that $\exp(-i2\pi j) = 1$ for integer j. Thus, c_k is completely defined by specifying N values – it just repeats itself for values outside this range of indices.

In *MATLAB* and in most other implementations of the DFT, the components are specified for

$$c_k, \quad k = 0, 1, \ldots. N-2, N-1 \tag{4-5}$$

If we specified the Fourier transform based on its derivation from a Fourier series and our understanding that the components at negative and positive frequencies define the phase while summing to yield a real time series, we would specify the components

$$\begin{aligned} c_k, \quad k &= -\left(\frac{N}{2}-1\right), \ldots. -1, 0, 1, \ldots. N/2 \quad \text{for } N \text{ even} \\ k &= -\frac{(N-1)}{2}, \ldots. -1, 0, 1, \ldots. \frac{(N-1)}{2} \quad \text{for } N \text{ odd} \end{aligned} \tag{4-6}$$

Since c_k repeats every N values, the ranges defined by equations (4-5) and (4-6) are completely equivalent although the alternate ordering can be confusing at first. To change from one convention to the other we only need to reorder the components on the DFT. In *MATLAB* the function fftshift does this reordering for you.

Inverse Discrete Fourier Transform

We might guess from inspection of equation (4-1) that the inverse discrete Fourier transform is given by

$$y_j = \sum_{k=0}^{N-1} c_k \exp\left(i\frac{2\pi}{N} jk\right) \tag{4-7}$$

To show that this works we can substitute for c_k using equation (4-3)

$$y_j = \sum_{k=0}^{N-1} \left[\frac{1}{N}\sum_{l=0}^{N-1} y_l \exp\left(-i\frac{2\pi}{N} lk\right)\right] \exp\left(i\frac{2\pi}{N} jk\right) = \frac{1}{N}\sum_{l=0}^{N-1} y_l \sum_{k=0}^{N-1} \exp\left(-i\frac{2\pi}{N}(l-j)k\right) \tag{4-8}$$

Now the sum over k is a geometric progression of the form

$$S = 1 + r + r^2 \ldots + r^{N-1} \tag{4-9}$$

with

$$r = \exp\left(-i\frac{2\pi}{N}(l-j)\right) \tag{4-10}$$

If $l = j$ we can see that the exponent term, r is unity and the sum $S = N$. If $l \neq j$ we can sum the geometric expression by first multiplying equation (4-9) by r

$$rS = r + r^2 \ldots + r^{N-1} + r^N \tag{4-11}$$

Subtracting equation (4-11) from equation (4-9) yields

$$(1-r)S = \left(1-r^N\right) \tag{4-12}$$

which gives the following expression for the sum

$$S = \frac{\left(1-r^N\right)}{(1-r)} \tag{4-13}$$

If we substitute equation (4-10) into (4-13) the numerator is

$$1 - \exp\left(-i2\pi(l-j)\right) = 0 \tag{4-14}$$

So all the $l \neq j$ terms sum to zero and we can see that equation (4-7) is correct.

Fourier Transform Pairs

If we replace c_k with Y_k in equations (4-3) and (4-7) we get a Discrete Fourier transform pair in the conventional notation

$$Y_k = \frac{1}{N}\sum_{j=0}^{N-1} y_j \exp\left(-\frac{i2\pi jk}{N}\right)$$
$$y_j = \sum_{k=0}^{N-1} Y_k \exp\left(\frac{i2\pi jk}{N}\right) \qquad (4\text{-}15)$$

Now as for the continuous Fourier Transform, there is an ambiguity in terms of the multiplying term in front of the inverse and discrete Fourier transform. In *MATLAB* the DFT pair is defined by

$$Y_k = \sum_{j=0}^{N-1} y_j \exp\left(-\frac{i2\pi jk}{N}\right)$$
$$y_j = \frac{1}{N}\sum_{k=0}^{N-1} Y_k \exp\left(\frac{i2\pi jk}{N}\right) \qquad \textit{MATLAB}\ (4\text{-}16)$$

Thus, compared with equation (5-15) the terms Y_k are N times as big and the inverse DFT requires the introduction of a $1/N$ term. One could also write a symmetric form

$$Y_k = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} y_j \exp\left(-\frac{2\pi ikj}{N}\right)$$
$$y_j = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} Y_k \exp\left(\frac{2\pi ikj}{N}\right) \qquad (4\text{-}17)$$

but I am not aware of anyone who uses this convention.

To compute the Fourier transform numerically on a computer, discretization plus numerical integration are required. This is an approximation of the true (i.e., mathematical), analytically-defined FT in a synthetic (digital) environment, and is called discrete Fourier transformation (DFT). There are three difficulties with the numerical computation of the FT:

- *Discretization* (introduces periodicity in both the time and the frequency domains)
- *Numerical integration* (introduces numerical error, approximation)
- *Finite time duration* (introduces maximum frequency and resolution limitations)

The DFT of a continuous time signal sampled over the period of T, with a sampling rate of Δt can be given as

$$S(m\Delta f) = \frac{T}{N}\sum_{n=0}^{N-1} s(n\Delta t)e^{-j2\pi m\Delta f\, n\Delta t} \qquad (4\text{-}18)$$

where $\Delta f=1/T$, and, is valid at frequencies up to $f_{max} = 1/(2\Delta t)$. Table 2 lists a simple Matlab m-file that computes (5) for a time record $s(t)$ of two sinusoids whose frequencies and amplitudes are user-specified. The record length and sampling time interval are also supplied by the user and DFT of this record is calculated inside a simple integration loop.

Table 2: *A Matlab module for DFT calculations*

```
%-------------------------------------------------------------------------------
-------
%   Program  : DFT.m
%     Purpose  : To calculate DFT of a time signal of two
sinusoids
% ------------------------------------------------------------------------------
------
%  Get the input parameters
    fr1=input('Frequency of the first sinusoid [Hz] = ? ');
    a1=input('Amplitude of the first sinusoid [ ] = ? ');
    fr2=input('Frequency of the second sinusoid [Hz] = ? ');
    a2=input('Amplitude of the first sinusoid [ ] = ? ');
    T=input ('Time record length [s] = ? ');
    dt=input ('sampling time interval  [s] = ? ');
    fmax=input('maximum frequency for DFT [Hz] = ? ');
    df=input('frequency sampling interval for DFT [Hz] = ? ');

    N=T/dt;  w1=2*pi*fr1;    w2=2*pi*fr2;    M=fmax/df;
%  Build the input time series
    for k=1:N
      st(k)=a1*sin(w1*dt*k)+a2*sin(w2*dt*k);
    end
%  Apply the DFT with M points
    for k=1:M
         Sf(k)=complex(0,0);
         for n=1:N
             Sf(k)=Sf(k)+st(n)*exp(-i*2*pi*n*dt*k*df);
         end
         Sf(k)=Sf(k)*dt/2*pi;
    end
%  Prepare the frequency samples
    for k=1:M
          F(k)=(k-1)*df;
    end
%  Plot the output
    plot(F,abs(Sf));
    title('The DFT of the Sum of two Sinusoids')
    xlabel('Frequency [Hz]');   ylabel('Amplitude')
%----------------------------- End of DFT.m ----------------------------
             -------
```

2.3 Fast Fourier Transform (FFT)

From the expression for the discrete Fourier transform shown in equation (4-16), it is clear that calculating each term for a real time series requires N multiplications of a real number and a complex exponential or $2N$ multiplications. Now all, N terms require $2N^2$ operations which is reduced to N^2 when we remember the symmetry properties of the Fourier transform.

However, it turns out that the Fourier transform can be computed in only $N \log N$ operations when the number of samples is a power of 2 (i.e., $N = 2^M$ where M is an integer). We will not cover the details, but we can explain it briefly as follows. The forward transform of equation (4-16) can be written as two series over the odd and even terms to yield

$$\begin{aligned} Y_k &= \sum_{j=0}^{N-1} y_j \exp\left(-\frac{2\pi i k j}{N}\right) \\ &= \sum_{j=0}^{N/2-1} y_{2j} \exp\left(-\frac{2\pi i k 2j}{N}\right) + \sum_{j=0}^{N/2-1} y_{2j+1} \exp\left(-\frac{2\pi i k(2j+1)}{N}\right) \\ &= \sum_{j=0}^{N/2-1} y_{2j} \exp\left(-\frac{2\pi i k j}{N/2}\right) + \exp\left(-\frac{2\pi i k}{N}\right) \sum_{j=0}^{N/2-1} y_{2j+1} \exp\left(-\frac{2\pi i k j}{N/2}\right) \\ &= P_k + \exp\left(-\frac{2\pi i k}{N}\right) Q_k \end{aligned} \tag{4-19}$$

We have expressed Y_k in terms of a sum of two Fourier transforms P_k and Q_k, each for a time series of $N/2$ samples. The reason why this reduces the number of calculations is that because the periodicity of the Fourier transform we can write

$$P_k = P_{k+\frac{N}{2}} \tag{4-20}$$

so the calculation of all the terms in P and Q requires only half the number of calculations required for all the terms in G. Now if N is a power of 2 we can keep on subdividing the Fourier series until we are left with series of two-point sequences. The book keeping becomes fairly complex to express, but the net result is that the number of operations is reduced to $N \log N$, an enormous computational saving for long time series. Without the FFT, Fourier transforms would be much less prevalent in computational data analysis.

The Fast Fourier transform (FFT) is an algorithm for computing DFT, before which the DFT required excessive amount of computation time, particularly when high number of samples (N) was required. The FFT forces one further assumption, that N is an integer multiple of 2. This allows certain symmetries to occur reducing the number of calculations (especially multiplications) which have to be done.

To write an FFT routine is not as simple as DFT routine, but there are many internet addresses where one can supply FFT subroutines (including source codes) in different programming languages, from Fortran to C++. Therefore, the reader does not need to go into details, rather to include them in their codes by simply using "include" statements or “call” commands. In Matlab, the calling command is *fft(s,N)* for the FFT and *ifft(s,N)* for the inverse FFT, where s is the recorded N-element time array. A simple m-file is included in Table 3, which computes and plots FFT of a sine-modulated Gaussian function whose parameters are supplied by the user.

Table 3: *A Matlab module for FFT calculations*

```
%------------------------------------------------------------------------------------
----
%   Program  : FFT.m
%   Purpose  : To calculate FFT of a sine modulated Gauss function
% ----------------------------------------------------------------------------------
-----
% Get the input parameters
```

```
   N=input ('N = ? ');
   fr=input('Modulation frequency [Hz] = ? ');
   dt=input ('time step, dt  [s] = ? ');
   w=2*pi*fr;

% Build the input time series
   for k=1:N
       X(k)=sin(w*dt*k)*exp(-(dt*N/(10*dt*k))^2);
   end
% Apply the FFT
   X2=fft(X,N);
   X2=fftshift(X2);               % swaps the left and right halves
   X2=abs(X2);
% Prepare the frequency samples
   fmax=1/dt;    df=1/(N*dt);
   for k=1:N
       F(k)=-fmax/2+(k-1)*df;
   end
% Plot the output
   plot(F,X2);
   title('The FFT of a Time Signal')
   xlabel('Frequency [Hz]');  ylabel('Amplitude')
%------------------------------ End of FFT.m ---------------------------------
            -----
```

2.4 Aliasing effect, Spectral Leakage and Scalloping Loss

As stated above, performing FT in a discrete environment introduces artificial effects. These are called aliasing effects, spectral leakage and scalloping loss.

It should be kept in mind when dealing with discrete FT that:

- Multiplication in the time domain corresponds to a convolution in the frequency domain.
- The FT of an impulse train in the time domain is also an impulse train in the frequency domain with the frequency samples separated by $T_0 = 1/f_0$.
- The narrower the distance between impulses (T_0) in the time domain the wider the distance between impulses (f_0) in the frequency domain (and vice versa).
- The sampling rate must be greater than twice the highest frequency of the time record, i.e., $\Delta t \geq 1/(2f_{\max})$ (Nyquist sampling criterion).
- Since *time – bandwidth* product is constant, narrow transients in the time domain possess wide bandwidths in the frequency domain.
- In the limit, the frequency spectrum of an impulse is constant and covers the whole frequency domain (that's why an impulse response of a system is enough to find out the response of any arbitrary input).

If the sampling rate in the time domain is lower than the Nyquist rate *aliasing* occurs. Two signals are said to alias if the difference of their frequencies falls in the frequency range of interest, which is always generated in the process of sampling (aliasing is not always bad; it is called mixing or heterodyning in analog electronics, and is commonly used in tuning radios and TV channels). It should be noted that, although obeying Nyquist sampling criterion is sufficient to avoid aliasing, it does not give high quality display in time domain record. If a sinusoid existing in the time signal not bin-centered (i.e., if its frequency is not equal to any of the frequency samples) in the frequency domain *spectral leakage* occurs. In addition, there is a reduction in coherent gain if the frequency of the sinusoid differs in value from the frequency samples, which is termed *scalloping loss*.

2.5 Windowing and Window Functions

Using a finite-length discrete signal in the time domain in FT calculations is to apply a rectangular window to the infinite-length signal [6]. This does not cause a problem with the transient signals which are time-bounded inside this window. But, what happens if a continuous time signal like a sine wave is of interest? If the length of the window (i.e., the time record of the signal) contains an integral number of cycles of the time signal, then, periodicity introduced by discretization makes the windowed signal exactly same as the original. In this case, the time signal is said to be periodic in the time record. On the other hand, there is a difficulty if the time signal is not periodic in the time record, especially at the edges of the record (i.e., window). If the DFT or FFT could be made to ignore the ends and concentrate on the middle of the time record, it is expected to get much closer to the correct signal spectrum in the frequency domain. This may be achieved by a multiplication by a function that is zero at the ends of the time record and large in the middle. This is known as *windowing*.

It should be realized that, the time record is tempered and perfect results shouldn't be expected. For example, windowing reduces spectral leakage but does not totally eliminate it. It should also be noted that, windowing is introduced to force the time record to be zero at the ends; therefore transient signals which occur (starts and ends) inside this window do not require a window. They are called *self-windowed* signals, and examples are impulses, shock responses, noise bursts, sine bursts, etc.

Other window functions (as opposed to the natural rectangular window which has the narrowest mainlobe, but the highest sidelobe level) are used to obtain a compromise between a narrow main lobe (for high resolution) and low sidelobes (for low spectral leakages). High frequency resolution provides accurate estimation of the frequency of an existing sinusoid and results in the separation of two sinusoids that are closely spaced in the frequency domain. Low spectral leakage improves the detectability of a weak sinusoid in the presence of a strong one that is not bin-centered. A few examples of common window functions are (*n = 0, 1,2, ... N-1*)):

Rectangular:
$$W(n) = 1 \tag{5a}$$

Hanning:
$$W(n) = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N}\right) \tag{5b}$$

Hamming:
$$W(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right) \tag{5c}$$

All of these window functions act as a filter with a very rounded top. If a sinusoid in the time record is centered in the filter then it will be displayed accurately. Otherwise, the filter shape (i.e., the window) will attenuate the amplitude of the sinusoid (scalloping loss) by up to a few dB (15-20 %) when it falls midway between two consecutive discrete frequency samples. The solution of this problem is to choose a flat-top window function;

Flat-top:
$$W(n) = 0.2810639 - 0.5208972\cos\left(\frac{2\pi n}{N}\right) + 0.1980399\cos\left(\frac{4\pi n}{N}\right) \tag{6}$$

which reduces the amplitude loss to a value less than 0.1 dB (1 %). However, this accuracy improvement does not come without its price; it widens the mainlobe in the frequency domain response (i.e., a small degradation in frequency resolution). It should be remembered that there is always a pay off between accuracy and resolution when applying a window function.

3. Basic Discretization DFT and FFT Requirements

Mathematically defined Fourier transformation can be used to calculate the FT of a function at any frequency. There is no maximum frequency, or frequency resolution limit, since these are numerical FT restrictions. The maximum frequency in DFT or FFT depends on the sampling interval, and the frequency resolution is determined by the signal record length. That is N samples of a time signal recorded during a finite duration of T with a sampling period of Δt ($N=T/\Delta t$) can be transformed into N samples in the frequency domain between $-f_{max}$ and $+f_{max}$ according to

$$f_{\max}=\frac{1}{2\Delta t},\quad \Delta f=\frac{1}{T}. \tag{7}$$

Since sampling interval and signal record lengths are finite in numerical computations in the computers maximum frequency and the resolution are also finite. This means

- any frequency component f_c beyond $+f_{\max}$ can not be observed in its actual frequency; instead it enters from left because of rotational symmetry and periodicity and appears at $-f_{\max}+f_D$ where $f_D=f_c-f_{\max}$.
- Similarly, any frequency component $-f_c$ beyond $-f_{\max}$ can not be observed in its actual frequency; instead it enters from left because of rotational symmetry and periodicity and appears at $f_{\max}-f_D$ where $f_D=|f_c-f_{\max}|$.

It should be noted that, the Matlab code in Table 2 directly integrates (4) numerically, so any number of frequency samples ($n\times\Delta f$) can be used to plot a frequency spectrum. Unfortunately, (7) still holds for the DFT. In other words, for example if one wants to discriminate two sinusoids with 50Hz and 55Hz in the frequency domain the frequency resolution Δf must be much less than their difference (5Hz).

4. Tests with DFT and FFT

A few examples are presented to review your knowledge and train yourself. Scenarios and plots obtained via either DFT or FFT routines are given below. Check the parameters of each scenario carefully and commend on the results in the frequency domain.

Question:
At which frequencies do you observe the signal if *Δt=10ms and T=1s for*

(a) s(t)=sin(100πt)+0.5 sin(70πt),*
*(b) s(t)=sin(32πt)+ 0.5*sin(86πt),*
*(c) s(t)=sin(90πt)+ 0.5*exp(j104πt),*
*(d) s(t)=exp(-j120πt)+ 0.5*sin(206πt)?*

The answers are given below but not in order. Which plot belongs to which case, why? Modify the Matlab code given in Table 3 and do the tests numerically.

What happens if Δt=10ms and T=0.5s in question 1?
Do it yourself and see what happens.

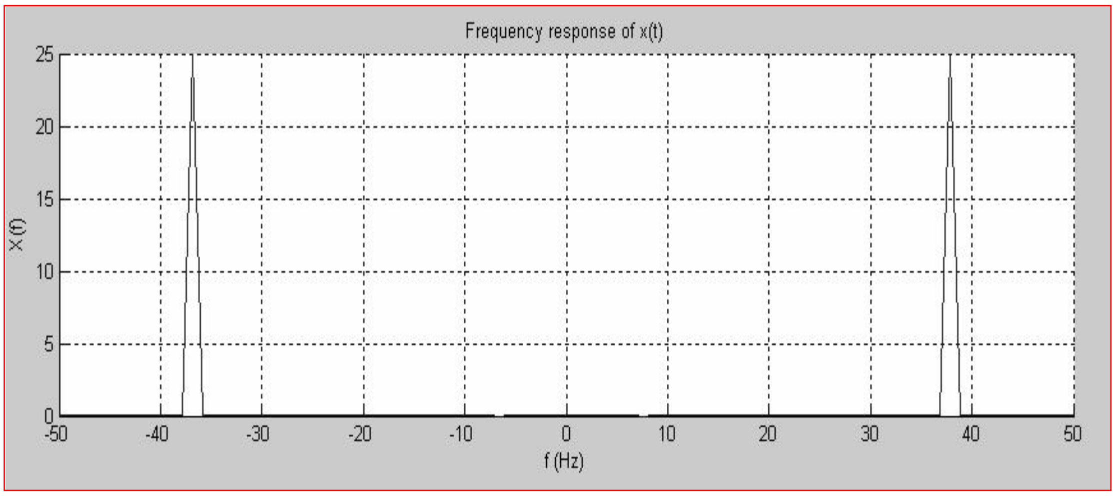

***Figure 2a*:**

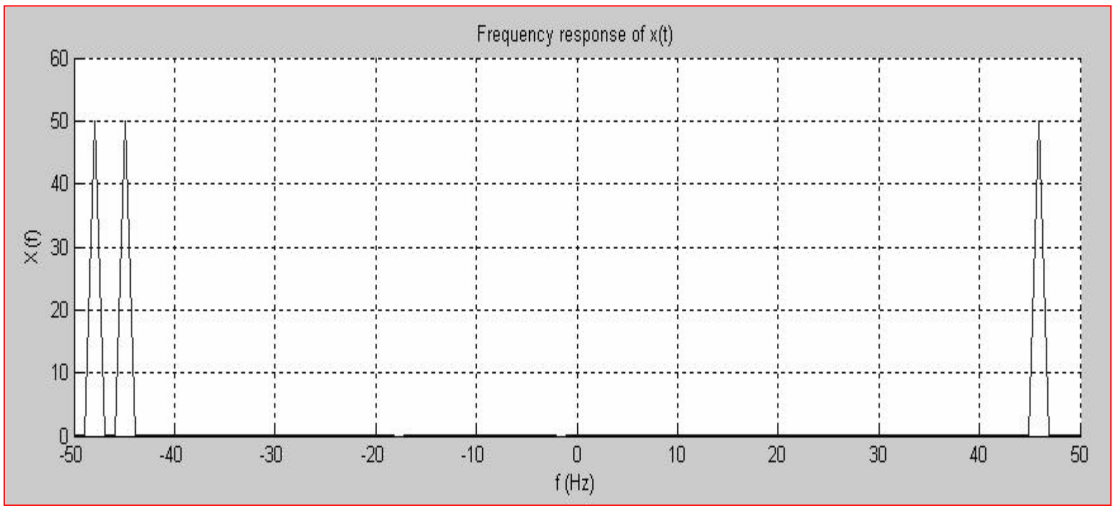

***Figure 2b*:**

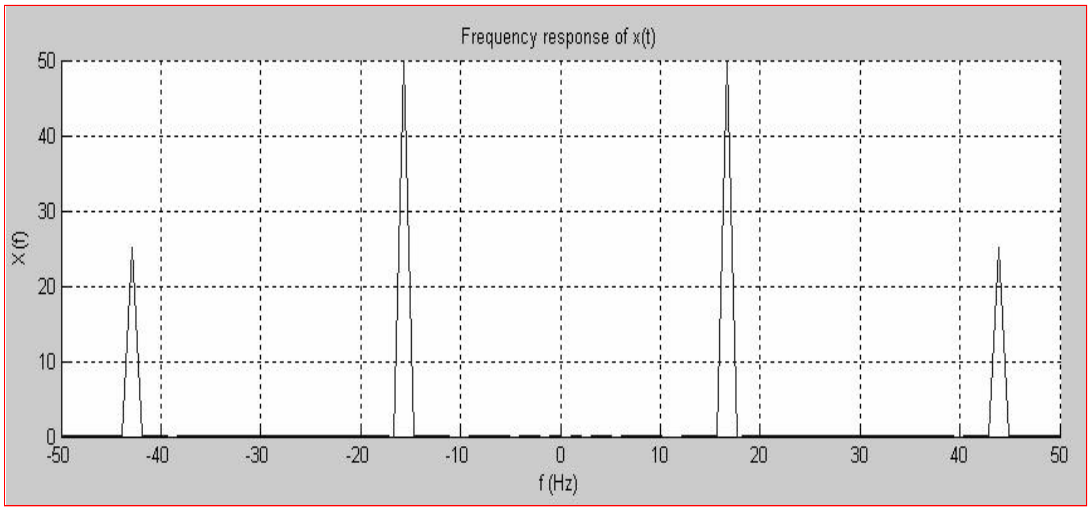

***Figure 2c*:**

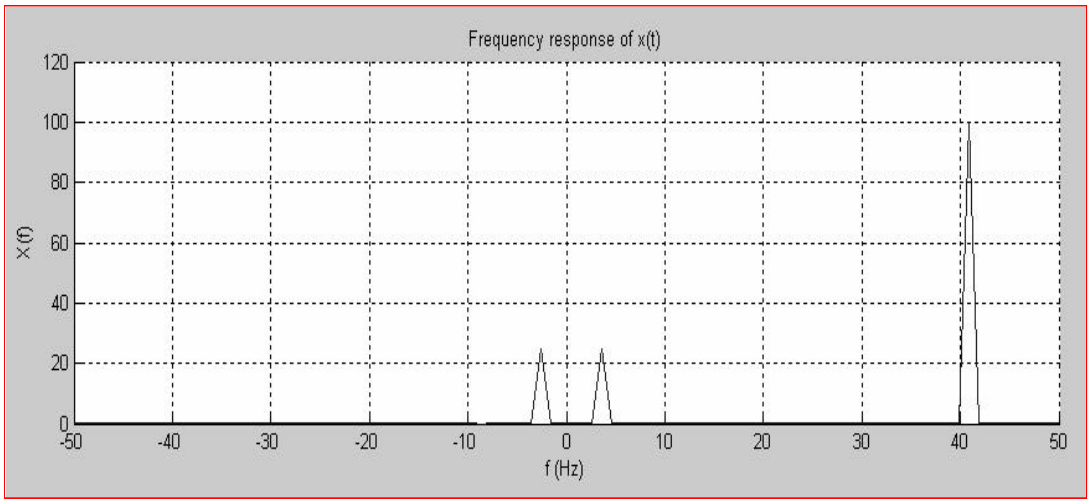

Figure 2d:

References

[1] Baron Jean Baptiste Fourier (see, e.g., http://bartleby.com/65/fo/Fouriers.html)

[2] HP, “The Fundamentals of Signal Analysis”, Hewlett Packard Application note 243, 1994

[3] L. Sevgi, F. Akleman, L. B. Felsen, “Ground Wave Propagation Modeling: Problem-matched Analytical Formulations and Direct Numerical Techniques”, IEEE Antennas and Propagation Magazine, Vol. 44, No.1, pp.55-75, Feb. 2002

[4] L. Sevgi, *Complex Electromagnetic Problems and Numerical Simulation Approaches*, IEEE Press – John Wiley and Sons, June 2003

[5] M. Levy, *Parabolic equation methods for electromagnetic wave propagation*, IEE, Institution of Electrical Engineers, 2000

[6] F. J. Harris, “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform”, Proc. IEEE, Vol. 66, no. 1, pp.51-83, Jan 1978

[7] Lima, F. (2018) What Exactly Is the Electric Field at the Surface of a Charged Conducting Sphere? Resonance, 23, 1215-1223. https://doi.org/10.1007/s12045-018-0731-y

[8] Assad, G. (2012) Electric Field “On the Surface” of a Spherical Conductor: An Issue to Be Clarified. Revista Brasileira de Fesica, 34, 4701.

[9] Griffiths, D. and Walborn, S. (1999) Dirac Deltas and Discontinuous Functions. American Journal of Physics, 67, 446-447. https://doi.org/10.1119/1.19283

[10] Fedak, W.A. (2002) Quantum Jumps and Classical Harmonics. American Journal of Physics, 70, 332-344. https://doi.org/10.1119/1.1445405

[11] Fan, G.-X. (2004) Fast Fourier Transform for Discontinuous Functions. IEEE Transactions on Antennas and Propagation, 52, 461-465. https://doi.org/10.1109/TAP.2004.823965

[12] Janssen, J.M. (1950) The Method of Discontinuities in Fourier Analysis. Philips Research Reports, 5, 435-460.

[13] Huang, X., Liu, X. and Mi, Y. (2013) The Fourier Series Approach to Investigate Phase-Locking Behaviors of the Sinoatrial Node Cell. Europhysics Letters, 104, Article ID: 38002. https://doi.org/10.1209/0295-5075/104/38002

[14] Thompson, W.J. (1992) Fourier Series and Gibbs Phenomenon. American Journal of Physics, 60, 425. https://doi.org/10.1119/1.16895

[15] Tsagareishvill, V. (2017) On Fourier Coefficients of Functions with Respect to General Orthonormal Systems. Izvestiya Mathematics, 81, 179. https://doi.org/10.1070/IM8394

[16] [Kvernadze, G. (2003) Approximating the Jump Discontinuities of a Function by Its Fourier-Jacobi Coefficients. Mathematics of Computation, 73, 731-751. https://doi.org/10.1090/S0025-5718-03-01594-1

[17] [Ageev, A.L. and Antonova, T.V. (2015) Approximation of Discontinuity Lines for aNoisy Function of Two Variables with Countably Many Singularities. Journal of