Appendix B: Supplementary Problems on Signal Analysis

B.1. Examples on Time-Domain and Frequency-Domain Signals

Example B.1.1: Given an asymmetric discrete signal find its reflection. It is obtained by reflecting every delta function in the left-hand sid e to the right-hand side and vice versa.

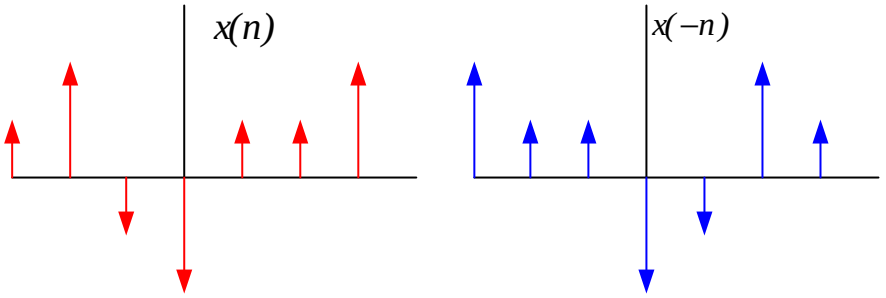


Figure B.1.1 An example of a discrete signal and its reflection.

Example B.1.2: Given a discrete signal, let us sketch its even and odd portions:

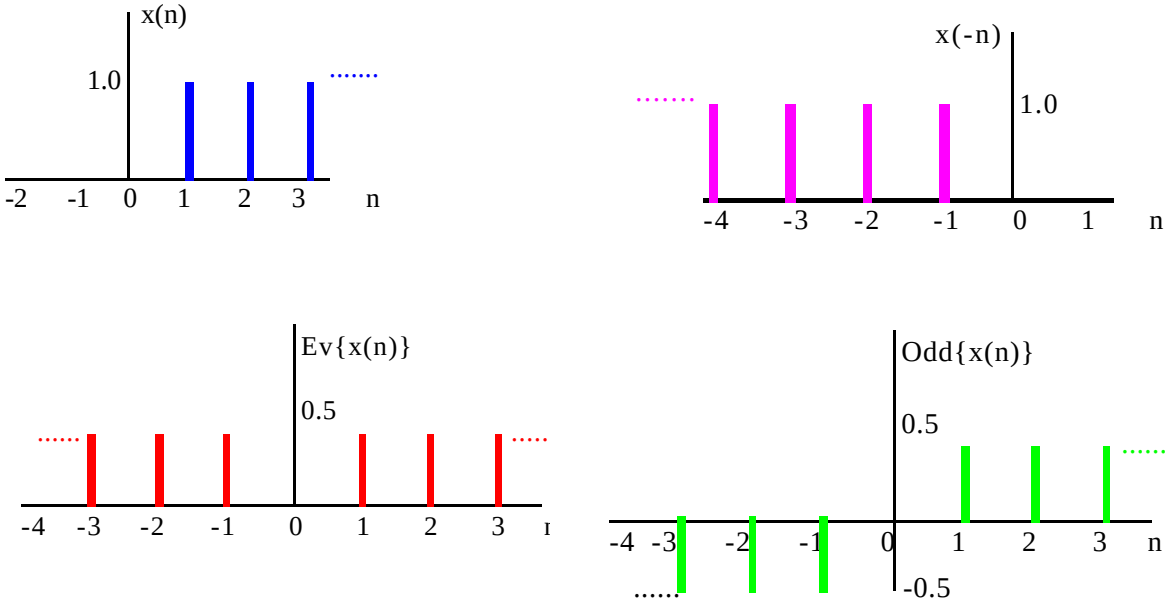


Figure B.1.2 Even and odd portions of a discrete signal.

B.2. Examples on Fourier Transforms and Properties

Example B.2.1. Fourier series expansion of an exponentially decaying function defined as $x(t) = e^{-t/2}$ and it is periodic satisfying $x(t) = x(t + nT_0) = x(t + n)$; where the period is 1.0 seconds. In the text we have computed the Fourier series coefficients for this function to be:

$c_0 = a_0 = 0.7869$; $c_n = 0.7869 * (\frac{2}{\sqrt{1+16\pi^2 n^2}})$ and $\theta_n = -arctan(4\pi n)$

```
% Magnitude & Phase Plots for an exponential decay function
n=0:16;
amp=zeros(size(n));
theta=zeros(size(n));
k=[1:1:length(n)];
k2=k .* k;
denom2=1+16*pi*pi .* k2; denom=sqrt(denom2);
coeff=0.79*2 ./denom;  coeffdb=20.0 .* log10(coeff);
theta=-atan(4*pi .* k); theta(1) =0;
coeffdb(1)=20*log10(0.79);

%Plotting funtions using “stem” with line thicness and color
hndl=stem(k,coeffdb, 'filled'); set(hndl,'lineWidth',3); set(hndl,'Color','red');
title(' Coefficients Plot in dB');  xlabel('frequency index'); ylabel('Amplitude in dB'); axis;
figure;
hndl=stem(k,theta, 'filled'); set(hndl,'lineWidth',3);  set(hndl,'Color','green');
title('Phase Plot of Coefficients'); xlabel('frequency index'); ylabel=('Phase');
```

Example B.2.2. Let us study the Example 2.9 in the text for the special case $a \to 0;\ \ A \to \infty$ such that $Aa \to 1$. The signal is plotted on top in Figure B.2.1. In this case, we can easily show that: $C_0 = 1/T_0$.

This is same as saying let the pulse train approach to an impulse train. In this case we have:

$$\sum_{n=-\infty}^{\infty} \delta(t - nT_0) = C_0 + \sum_{n=1}^{\infty} C_n . cos(nw_0 t)$$

The coefficients can be explicitly computed:

$$C_n = Lim_{a\to 0}(\frac{2A}{n\pi} . sin(\frac{n\pi a}{T_0})) = Lim_{a\to 0}(\frac{2A}{n\pi} . \frac{n\pi a}{T_0} . \frac{Sin(n\pi a / T_0)}{n\pi a / T_0} = \frac{2}{T_0}$$

which are all constant as shown as the bottom plot in Figure B.2.1. So, we have another impulse train with a different periodicity and heights.

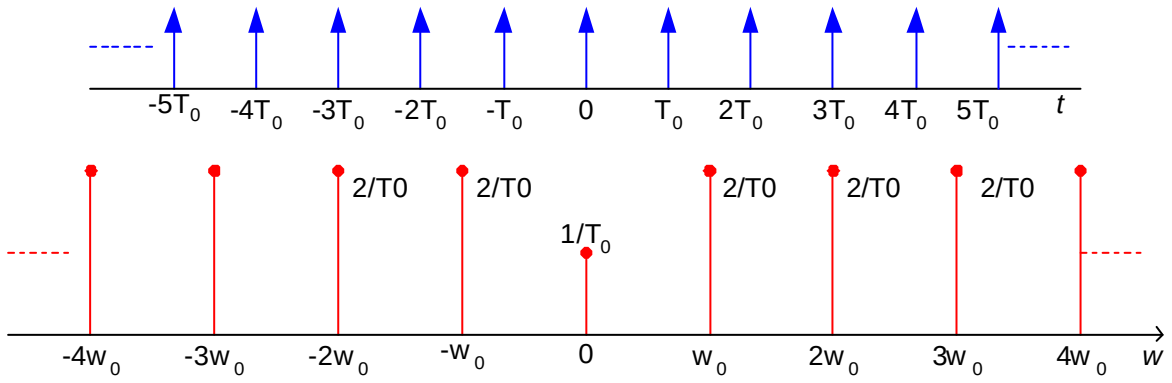
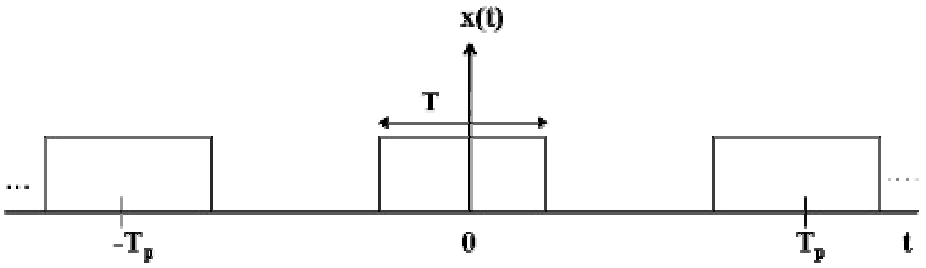


Figure B.2.1. Impluse Train and its spectral coefficients.

Example B.2.2. M-file for Example 2.11. Fourier Transform of a single Sinusoid.

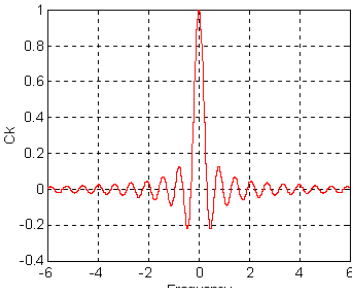
```
% Example  on Fourier Spectrum of a single sinusoid
% Initialization
M=4; N=2^M; f0=2;
w0=2*pi*f0;
n=0:1:N-1;
T0=1/f0;  delta_t=T0/N;   t=n*delta_t;

% Signal generation and FFT
xt = 5.0 * cos(w0*t);
XW = fft(xt); XW=XW(:);
nsym =-N/2:1:N/2-1;  fsym=nsym/T0; delta_f=1/delta_t;
Cn = 1/N * XW;  Cn=fftshift(Cn)';
Theta = (180/pi)*angle(Cn+0.001); Theta=fftshift(Theta)';
Cn = Cn(:); Theta=Theta(:);

% Plotting
figure; stem(fsym,abs(Cn));
title('Amplitude Spectrum'); xlabel('Frequency');ylabel('Amplitude');

figure; axis([-20,20,-2,2]); stem(fsym,Theta);
title('Phase Spectrum'); xlabel('Frequency'); ylabel('Phase');
```

Example B.2.3. Fourier transform of a pulse train for (i) the case when the period Tp is fixed and the pulse width T varies and (ii) when Tp is fixed and the pulse width T varies.

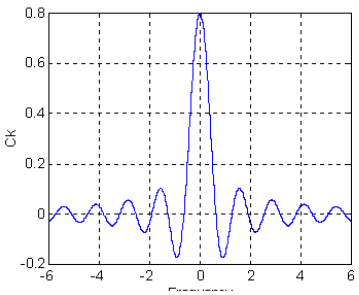


(i) Case for Tp is fixed and the pulse width T varies:

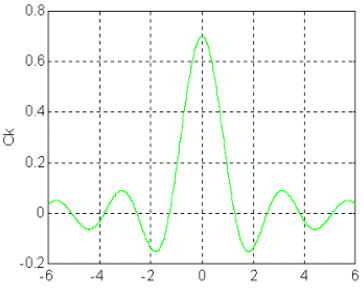
Pulse Width T = $0.2T_p$



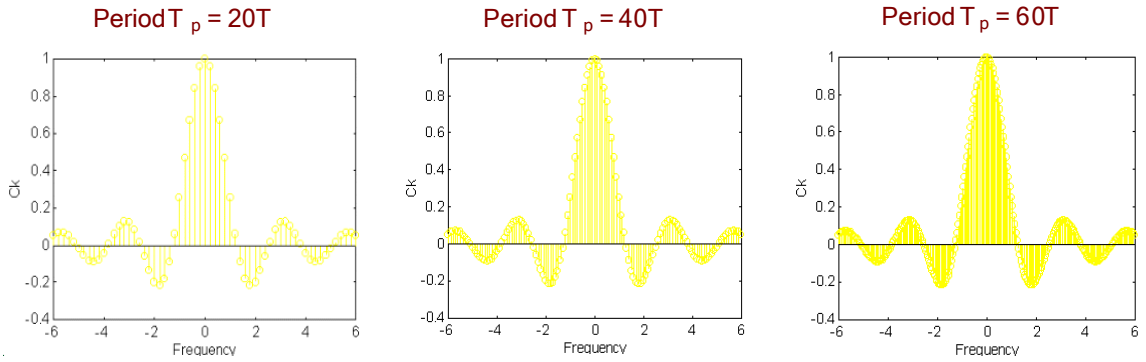
Pulse Width T = $0.1T_p$



Pulse Width T = $0.05T_p$

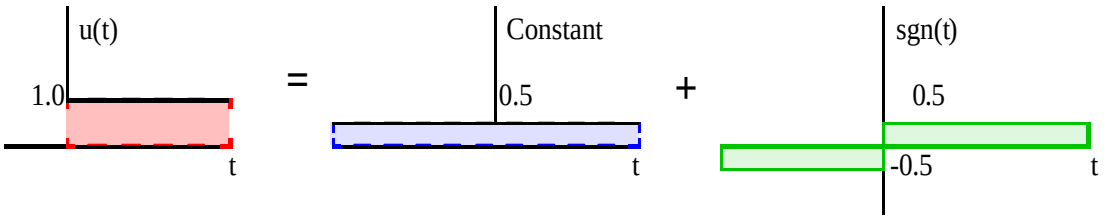


(ii) Case for T is fixed but Tp varies:



(More information on this example can be found in the "State-of-the-art Crossmedia Tools for a Course on Communication Systems" by the author at the website: http://akhisar.sdsu.edu/crossmedia/)

Example B.2.4. Fourier transform of a unit-step function. As we can see from the sketch below, we can write our unit-step function as a sum of two simple functions: $u(t) = \frac{1}{2} + \frac{1}{2}.Sgn(t)$



Here $Sgn(t)$ is the sign indicator function. But we can also write the derivative of this expression:

$$\frac{d}{dt}[1/2.Sgn(t)] = d(t) \quad \text{or equivalently:} \quad (jw).F\{1/2.Sgn(t)\} = 1,$$

which results in a form:

$$F\{1/2.Sgn(t)\} = 1/jw \quad \text{and} \quad F\{1/2\} = (1/2).2pd(w) = pd(w)$$

Finally, we combine these two transforms to obtain:

$$F\{u(t)\} = pd(w) + 1/jw$$

Example B.2.5. Let us perform the numerical convolution of a unit-step and exponential time functions. That is:

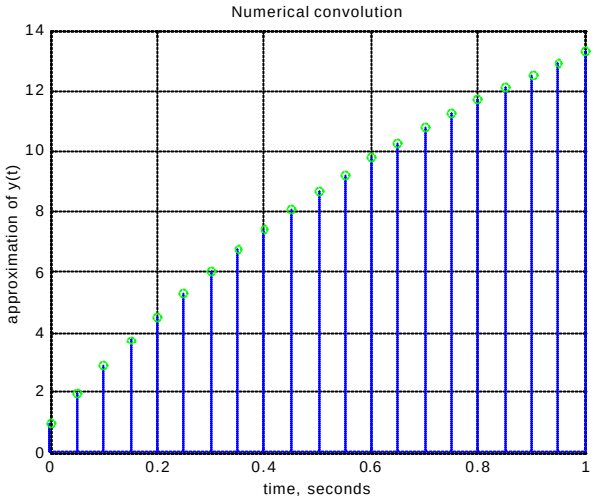
$$x(t) = u(t)$$

$$h(t) = e^{-t}.u(t)$$

and the output from the system is:

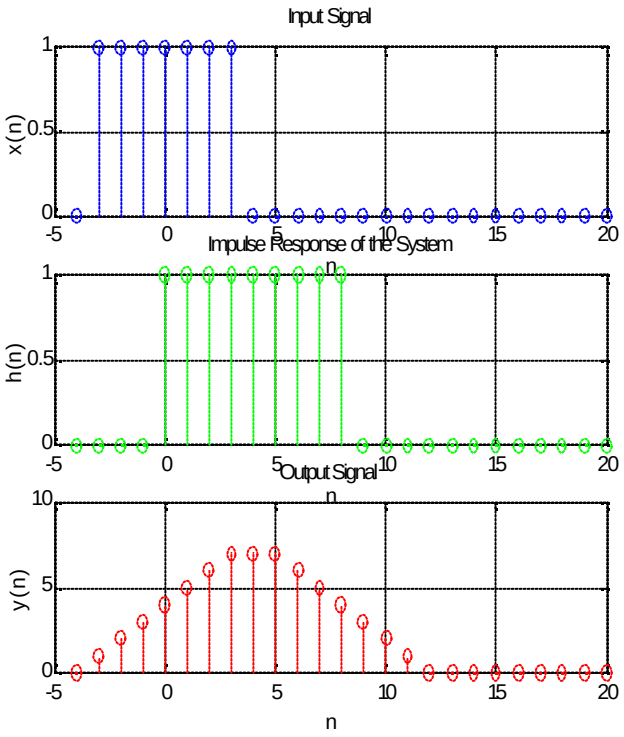
$$y(t) = x(t) \otimes h(t) = u(t) \otimes e^{-t}.u(t)$$

```
% Numerical convolution of a unit-step and an exponential signal.
t=0:.05:1; h=exp(-1*t)
x= ones(size(t))
y=conv(h,x)

ptit='Numerical convolution'
xlab='time, seconds'
ylab='approximation of y(t)';
displot(t,y(1:21),xlab,ylab,ptit)
grid; axis
```



Example B.2.6. Numerical convolution of two time-functions of different length.

```
% Convolution of two-discrete finite pulses
n=-4:20;
x=zeros(size(n));
h=zeros(size(n));
x(2:8)=ones(size(n(2:8)));
h(5:13)=ones(size(n(5:13)));

%Plot of the input and system
axis([-4,20,0,2])
subplot(311),stem(n,x);
grid;
title('Input Signal');
xlabel('n');
ylabel('x(n)');
subplot(312),stem(n,h,'g');
grid;
xlabel('n');
ylabel('h(n)');

title('Impulse Response of the System');

%convolution
y=conv(h,x);
subplot(313),stem(n,y(5:29),'r');grid;
title('Output Signal');
xlabel('n'); ylabel('y(n)');
```



Example B.2.7. Consider a message signal with characteristics:

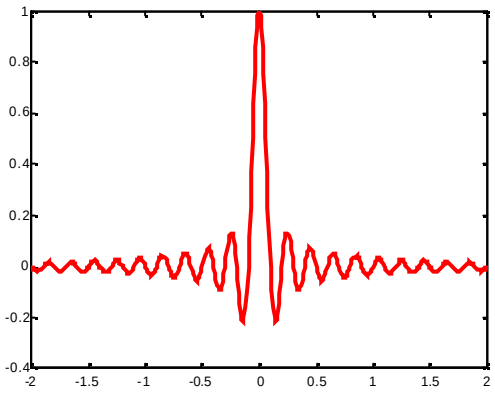
$$x(t)=\begin{cases}\text{sinc}(10t), & |t|\le 2\\ 0, & otherwise\end{cases}$$

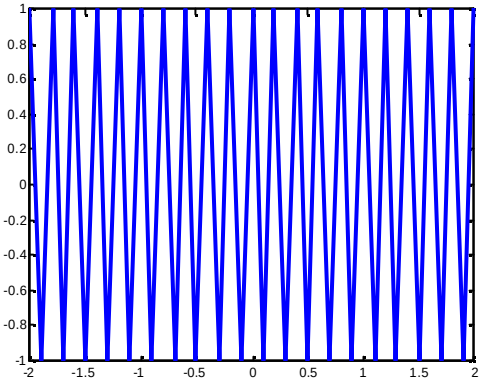
Let us use to modulate a 25 Hz sinusoidal carrier using DSB-SC AM and plot the modulated signal[1]. It is reasonable to use a 100 Hz sample rate so we can see more details. If we examined the spectrum of the modulated signal we should see the center frequency move from 0 to 25 Hz. But we were not asked to show this. Using MATLAB, we form the signal thus:

```
t=-2:.01:2;          % Look at 4 seconds of the signal total
xt=sinc(10*t);
plot(t,xt,'r'),set(gcf,'color',[1 1 1]),title('Plot of x(t)');
```

Now define a carrier to modulate the signal with:

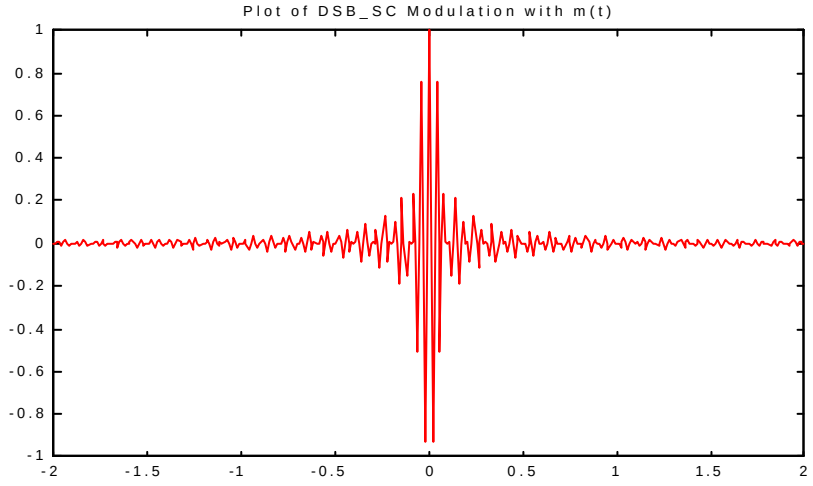
```
c=cos(2*pi*25.*t);          % 25 Hz carrier
plot(t, c,'b');
```





Now modulator output is given by:

```
dsbsc_am=xt.*c;
plot(t,dsbsc_am,'r'),title('Plot of DSB_SC Modulation with m(t)');
```



[1] Even thought the terminology DSB-SC AM is not covered yet, it is simply a time-domain multiplication of a signal $x(t)$ with a carrier signal, also a sinusoid.

B.3. Examples on Filters, Power Spectral Density and Noise

Example B.3.1: Let us design a Butterworth Filter with a normalized bandwidth of B=1.0 radians/second and an amplitude level $K = \pi$ and the order of the filter n=3 using Matlab Tools. For these parameters we write the expressions for the magnitude and phase responses:

$$|H(w)| = K\{u(w+1) - u(w-1)\} \quad \text{(B.3.1)}$$

$$\theta = -t_0.w \quad \text{(B.3.2)}$$

The combined frequency response is simply the product in frequency-domain:

$$H(w) = K\{u(w+1) - u(w-1)\}e^{-jt_0.w} \quad \text{(B.3.3)}$$

The inverse Fourier transform will give us the impulse response of this filter.

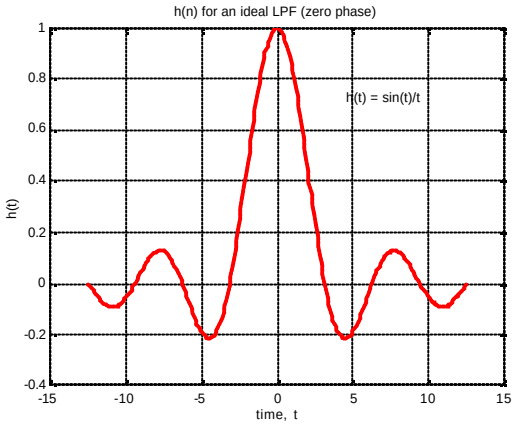
$$h'(t) = \frac{K}{pt}Sin(Bt) = \frac{KB}{p}Sinc(Bt) \quad for -\infty < t < \infty \quad \text{(B.3.4)}$$

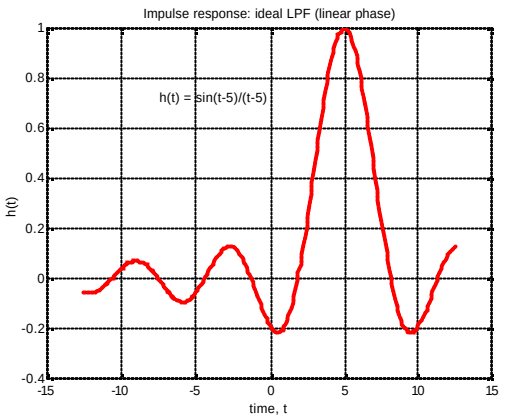
which is a non-causal function and it cannot be implemented. However, by delaying the impulse response by a reasonable amount and terminating appropriately, we can come close to this ideal filter:

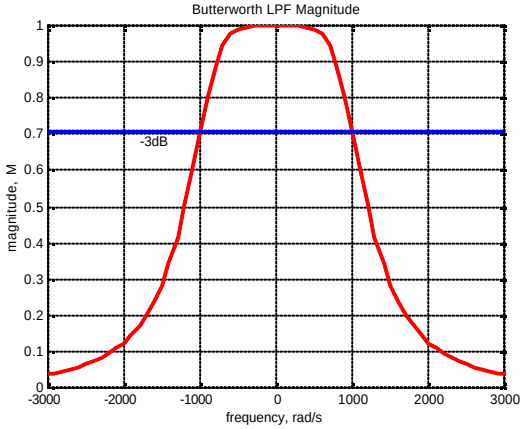
$$\hat{h}(t) = \frac{K}{\pi(t-t_0)}Sin[B(t-t_0)] = \frac{KB}{\pi}Sinc[B(t-t_0)] \quad for -\infty < t < \infty \quad \text{(B.3.5)}$$

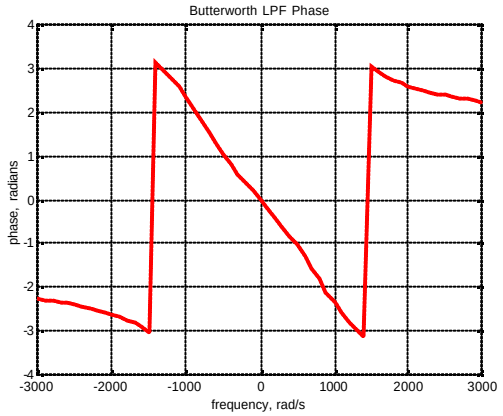
Using Matlab tools we have implemented (B.3.5) and the filter characteristics and responses are shown below together with the m-file. Other filters can easily be designed by appropriately modifying the parameters in this source code.

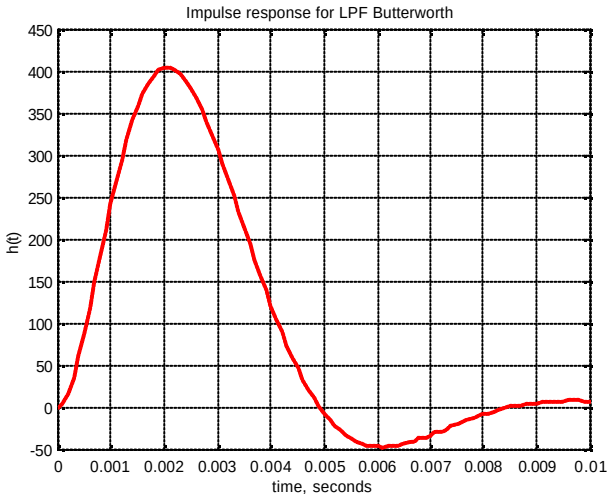
```
% Example  on Ideal and Butterworth Low Pass Filters
% h(n) for an ideal lowpass filter (zero phase)
t=-4*pi:.1:4*pi
y=sin(t) ./t; axis([1 2 3 4]); axis
plot(t,y,'r'); title('h(n) for an ideal LPF (zero phase)'); xlabel('time, t');ylabel('h(t)'); grid;
text(.65,.8,'h(t) = sin(t)/t','sc'); pause;

%Impulse response: ideal LPF (linear phase)
t=-4*pi:.1:4*pi;
y=sin(t-5) ./(t-5); axis([1 2 3 4]); axis;
plot(t,y,'r'); title('Impulse response: ideal LPF (linear phase)'); xlabel('time, t'); ylabel('h(t)'); grid;
text(.25,.8,'h(t) = sin(t-5)/(t-5)','sc'); pause;

%Butterworth lowpass filter
w=-3e3:100:3e3; b=[1e9];
a=[1,2e3,2e6,1e9]; h=freqs(b,a,w);
M=abs(h); g=.707*ones(size(w))*max(M); axis([1 2 3 4]); axis;
plot(w,M,'r',w,g,'b'); title('Butterworth LPF Magnitude'); xlabel('frequency, rad/s'); ylabel('magnitude, M');
grid; text(.2,.68,'-3dB','sc'); pause;
P=angle(h); plot(w,P,'r'); title('Butterworth LPF Phase'); xlabel('frequency, rad/s'); ylabel('phase, radians');
grid; pause;

%Impulse response for LPF Butterworth
num=[1e9]; den=[1,2e3,2e6,1e9];
t=0:.0001:.01;    % start time:increment:finis;
[y,v,t]=ksimptf(num,den,t);    % call ksimptf;
axis([-0.001,0.01,-100,500]);
plot(t,y,'r'); title('Impulse response for LPF Butterworth '); xlabel('time, seconds'); ylabel('h(t)'); grid; axis;
```





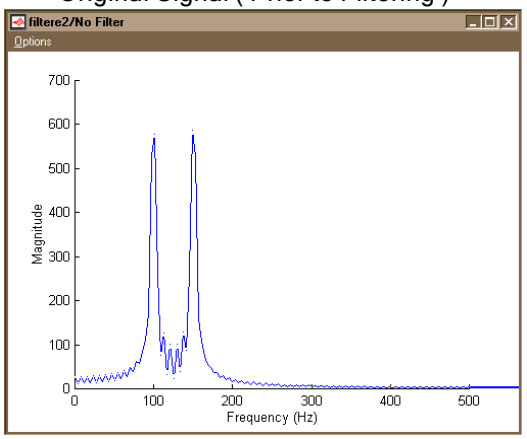




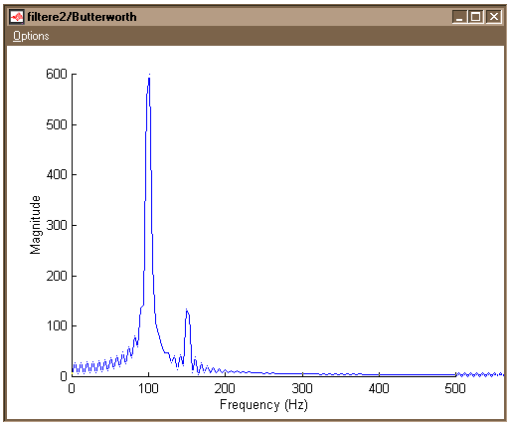


Example B.3.2: In this example, we try to recover a signal from a combination of two signals, one at 100Hz signal using Chebyshev Type 1 and 2 filters, Butterworth filter, FIR filter and Elliptic filter. All filters are lowpass filters are of order 8 and the cutoff is set at 125Hz.

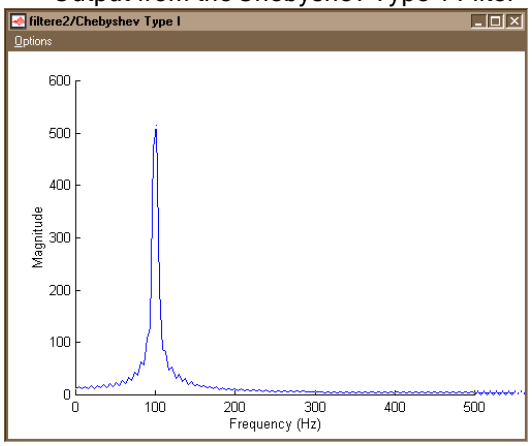
Original Signal (Prior to Filtering)



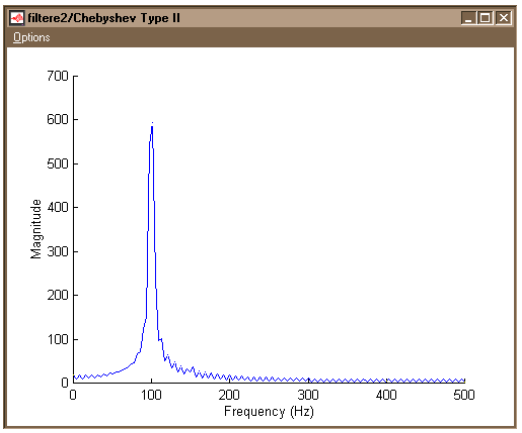
Output from the Butterworth Filter



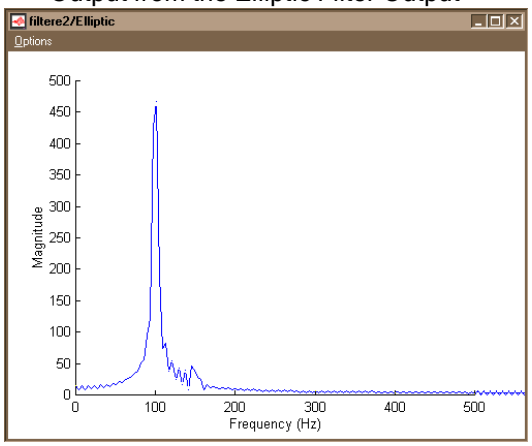
Output from the Chebyshev Type 1 Filter



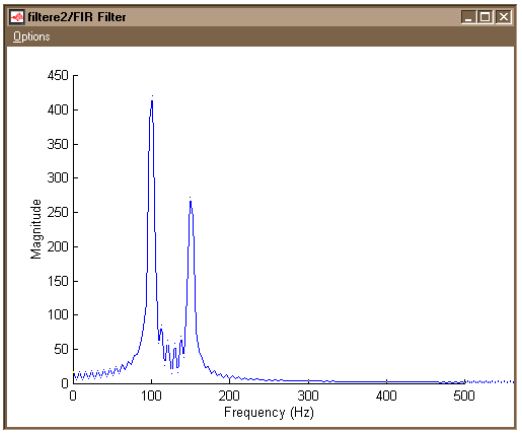
Output from the Chebyshev Type 2 Filter



Output from the Elliptic Filter Output



Output from the FIR type Filter Output



Example B.3.3: In this example, we will illustrate the technique of Fast Convolution using Fourier transform (FFT and IFFT).

For an N point signal and the system function, the ordinary convolution operation has a computational complexity of N^2. When N is large and the need to perform convolution in real-time is a necessity, the system requirement may become unrealistic.

For instance, for a speech segment of 0.5 seconds duration sampled at 8000 samples per second, N = 4000 and $N^2 = 16*10^6$ convolutions computations are needed. This puts an unrealistic burden on the computing power requirements of the system. On the other hand, the same task can be performed using FFT techniques.

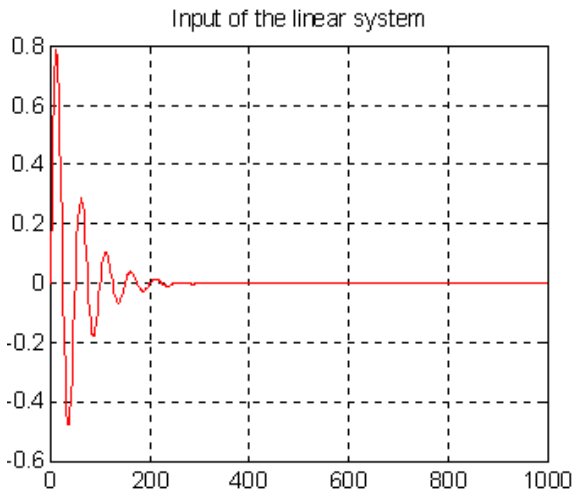
Consider a sequence of N = 4096 samples $[x_n]$, which will be convolved with another set of N = 4096 samples of a system transfer function: $[y_n]$. Instead of performing a convultion operation as we did in Example B.1.8 above, let us use the following procedure:

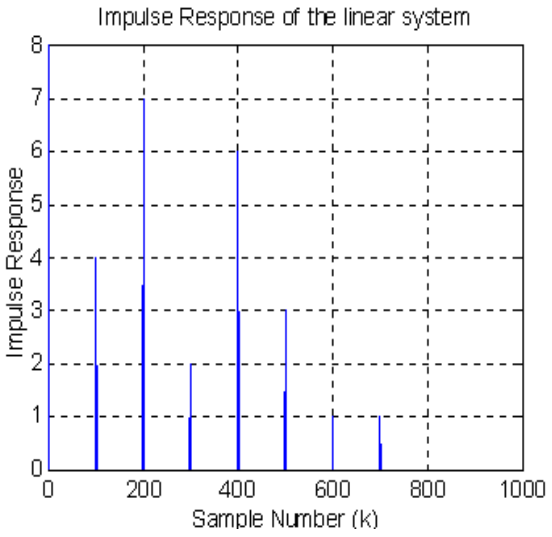
Step 1. Perform FFT on x_n and y_n to obtain X_k and Y_k.

Step 2. Multiply X_k and Y_k.

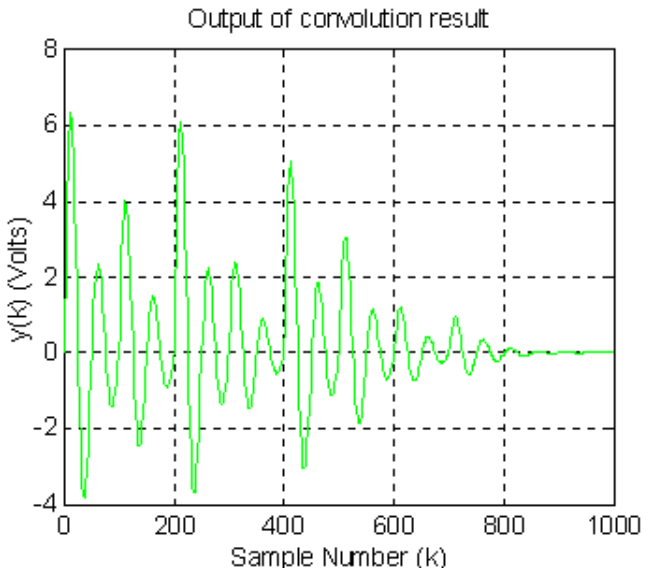
Step 3. Perform the summation and scaling operations required by the convolution equation.

If we count the number of operations in this procedure, we get 2*(4096 $\log_2$ 4096) = 98,304 for step 1, 4096 multipliers and 4096 sums for the next two steps. The overall operation count is approximately 10^5 versus $1.677 * 10^7$ in the direct convolution technique. Thius simple example demonstrates vividly that the technique of fast convolution using FFT has improved the performance by 167 fold.





These diagrams above and the next page show that convolution of a typical excitation signal with the impulse of an electromagnetic propagation channel. These impulses represent the paths of various reflectors. Therefore, the receiver input is the ensemble of all link responses with different delays and attenuation



Example B.3.5: In this example we compute the power and power spectral density of a sum of two sinusoids of unit amplitude and duration T=10 seconds; one located at 50 Hz and the other at 300 Hz.

$$x(t) = \begin{cases} Cos(2\pi 50t) + Cos(2\pi 300t) & 0 \le t \le 10 \\ 0 & Otherwise \end{cases}$$

Let us sample the combined signal at a rate 1000 samples per second. It is not difficult to see from the PSD curves of Figure B.3.2, we have two harmonics located at 50 and 300 Hz.

```
% MATLAB script Power Spectral Density and Power Computation
ts=0.001; fs=1/ts;
t=[0:ts:10];
x=cos(2*pi*50*t)+cos(2*pi*300*t);
p=spower(x);psd=spectrum(x,1024);
specplot(psd,fs)

function p=spower(x)
%       SPOWER, Returns the power in signal x
p=(norm(x)^2)/length(x);
```

Answer: P=1.0003

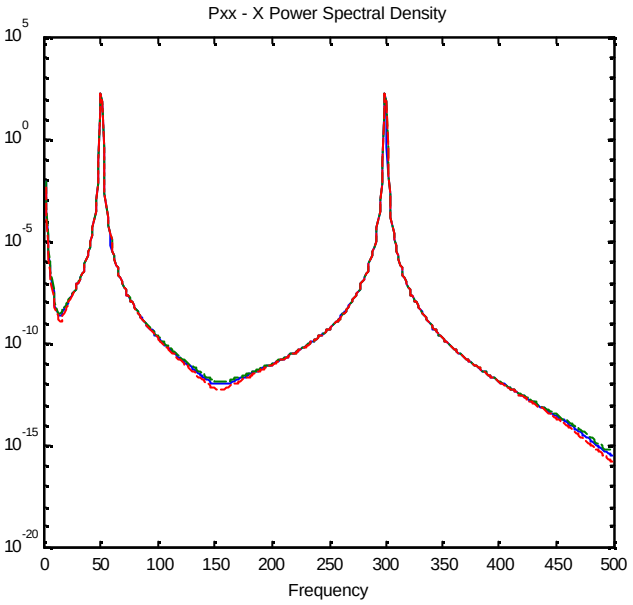


Figure B.3.2. One-sided power spectral density for Example B.2.5.

Example B.3.6: Using Matlab tools we will generate 100, 10,000, and 1,000,000 Gaussian random numbers and plot both the PDF and PSD. The m-file for this example can be found at the end of this chapter. The PDF and PSD plots are shown below. As the number of noise samples increases the PDF approaches to a true Gaussian (normal, bell-curve) and the PSD becomes whiter.

```
%   Example on Gaussian Noise Generation and PSD
%       We will use n=100,10000, and 1000000 samples in
%       generating almost true Gauassian PDF
n=1000000; s=1.0;
noise=s*randn(1,n);
x_coor=-4:0.1:4;
pdf=hist(noise,x_coor);
pdf=pdf/n;
bar(x_coor,pdf), xlabel('X'), ylabel('PDF'), title('Gaussian PDF');
f=1:256;
psd=spectrum(noise)
figure; bar(psd), xlabel('Frequency'), ylabel('PSD'),title('PSD');
```

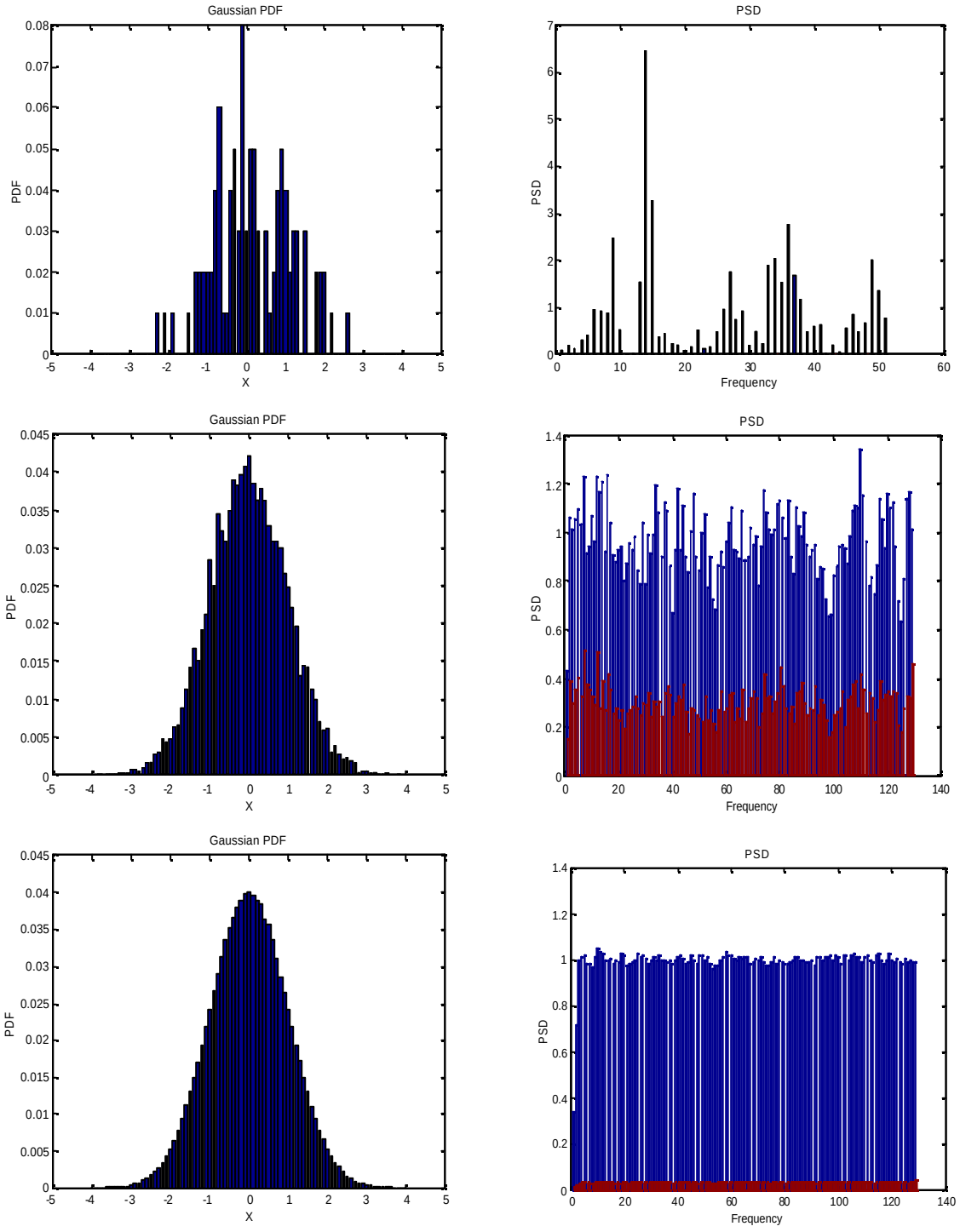


Figure B.3.2. Gaussian Noise PDF and PSD for 100, 10,000, and 1,000,000.

Example B.3.7: In this example, we demonstrate correlation of a signal corrupted with uniform white noise. The signal is given by

$$x_k = Sin(2*p*k/20) + \sqrt{60}.(r_k - 0.5) \quad for \;\; 0 < k < 1000$$

where r_k is a random function. In the given equation, we note that the sine wave is just barely discernible in the white noise, which is typical when the SNR is low. The first plot shows the white noise as the most dominant component. However, it is possible to detect the sinusoidal component in x_k by performing an autocorrelation analysis. The sinusoidal component in x_k is clearly detectable as shown in the second plot which is the autocorrelation function of the data sequence. This detection was successful as the sine wave is the only correlated or 'coherent' part of the signal.The autocorrelation plot shows an impulse at n=0 which is typically due to the white noise in x_k, which is uncorrelated for n not equal to zero , plus a cosine component due to the sine wave in x_k. Therefore, it can be deduced that the correlation operation can be used effectively to detect periodic components in a signal.