1. **Functions and Plots**

(a) Evaluate the following signal at an arbitrary set of time points:



That is, create an input of sampling time points, define an output vector with the values of *x(t)* evaluated at those time points. For time points falling outside [−3, 4], the output vector should return the value zero.

(b) Plot *x(t)* versus t, for *−6 ≤ t≤ 6*.

(c) Evaluate and plot *x(3 − t)* versus *t*.

1. **Convolution**

Convolve numerically the signals, *x1(t) = 3I[−2,−1]* and *x2(t) = 4I[1,3]*. Check that the result you get spans the correct interval and has the correct scaling (based on the analytical solution). Matlab approximates continuous-time convolution by using the “*conv(*X1*,* X2*)*” function multiplied by the sampling interval *dt*.

Thus, define scalars *t1*, *t2* and *dt*, representing the starting time for the samples of *x1(t)*, the starting time for the samples of *x2(t)*, and the spacing of the samples, respectively. Thus, define vectors X1 and X2 representing discrete samples of the two signals to be convolved. Then, define vectors Y *= dt conv(*X1*,* X2*)* and *t*, corresponding to the samples of the convolution output and the sampling time, respectively. Plot Y against *t*. Pay special attention on accurately defining the span (minimum and maximum values) for time *t*!

1. **Fourier transform**

The following Matlab code derives numerically the Fourier transform of the sinusoidal signal  using the function fft that stands for the fast Fourier transform (FFT). It also describes thoroughly the different operations and definitions of parameters required to derive the FFT. For the purpose of this lab, try to familiarize with definitions and steps of the code below. It is quite easy as it requires very basic programming and mathematical skills!

ts=1/16; %sampling interval

time\_interval = 0:ts:1; %sampling time instants

%%time domain signal evaluated at sampling instants

signal\_timedomain = sin(pi\*time\_interval); %sinusoidal pulse in our example

fs\_desired = 1/160; %desired frequency granularity

Nmin = ceil(1/(fs\_desired\*ts)); %minimum length DFT for desired frequency granularity

%for efficient computation, choose FFT size to be power of 2

Nfft = 2^(nextpow2(Nmin)) %FFT size = the next power of 2 at least as big as Nmin

%Alternatively, one could also use DFT size equal to the minimum length

%Nfft=Nmin;

%note: fft function in Matlab is just the DFT when Nfft is not a power of 2

%freq domain signal computed using DFT

%fft function of size Nfft automatically zeropads as needed

signal\_freqdomain = ts\*fft(signal\_timedomain,Nfft);

%fftshift function shifts DC to center of spectrum

signal\_freqdomain\_centered = fftshift(signal\_freqdomain);

fs=1/(Nfft\*ts); %actual frequency resolution attained

%set of frequencies for which Fourier transform has been computed using DFT

freqs = ((1:Nfft)-1-Nfft/2)\*fs;

%plot the magnitude spectrum

plot(freqs,abs(signal\_freqdomain\_centered));

xlabel(’Frequency’);

ylabel(’Magnitude Spectrum’);

(a) Once you understand the above code, compute the Fourier transform of , where the unit of time is microsecond, the signal is sampled at the rate of *16 MHz*, and truncated to the range [−8, 8] microseconds. We wish to attain a frequency resolution of *1 KHz* or better.

b) Plot the magnitude of the Fourier transform versus frequency, making sure you specify the units on the frequency axis. Check that the plot confirms your expectations.

(c) Plot the phase of the Fourier transform obtained in (a) versus frequency (again, make sure the units on the frequency axis are specified). What is the range of frequencies over which the phase plot has meaning?