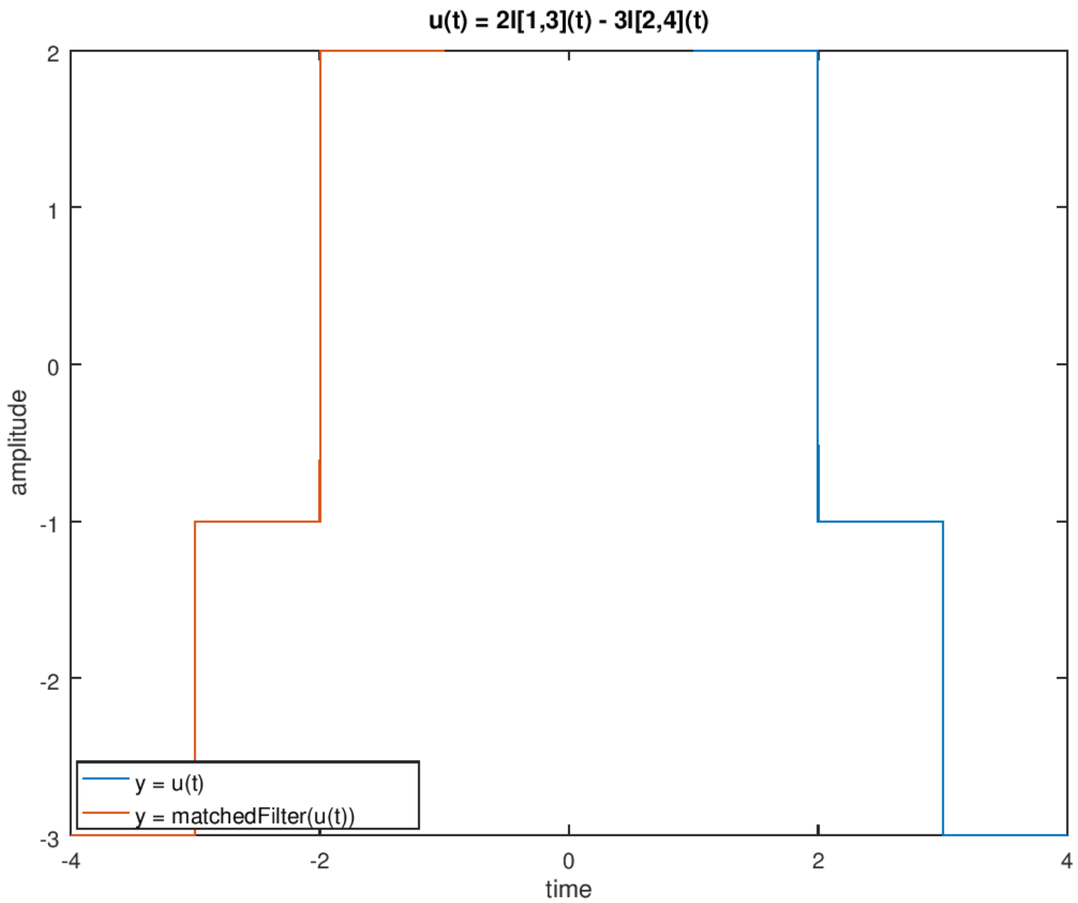
Principles of Communication Systems Lab
Lab 2 - August 30th, 2017

IMT2015524

Matched Filter

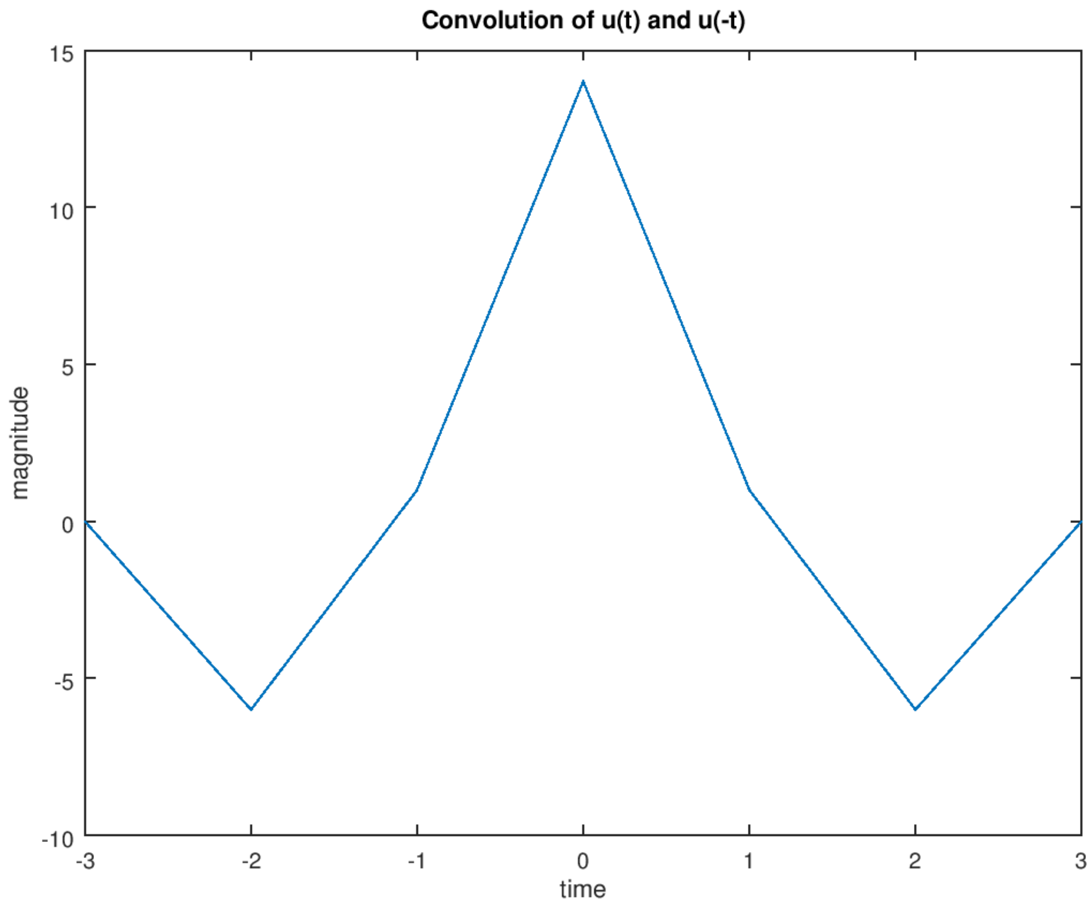
3a)



%Code:

```
% https://pastebin.com/3aCah2cE

resolution = 0.001;

first = ones(1/resolution,1);
first = first.*(2);

second = ones(1/resolution, 1);
second = second.*(-1);

third = ones((1/resolution)+1, 1);
third = third.*(-3);

signal = vertcat(first, second, third); % u(t)
time = [1:resolution:4];

signal_filtered = result(end:-1:1); % reversing magnitudes
time_filtered = time(end:-1:1).*(-1); %reverse and multiply my -1

plot(time, signal, time_filtered, signal_filtered);
title("u(t) = 2I[1,3](t) - 3I[2,4](t)");
xlabel ("time");
ylabel ("amplitude");
legend("y = u(t)", "y = matchedFilter(u(t))", "Location",
     "southwest");
```

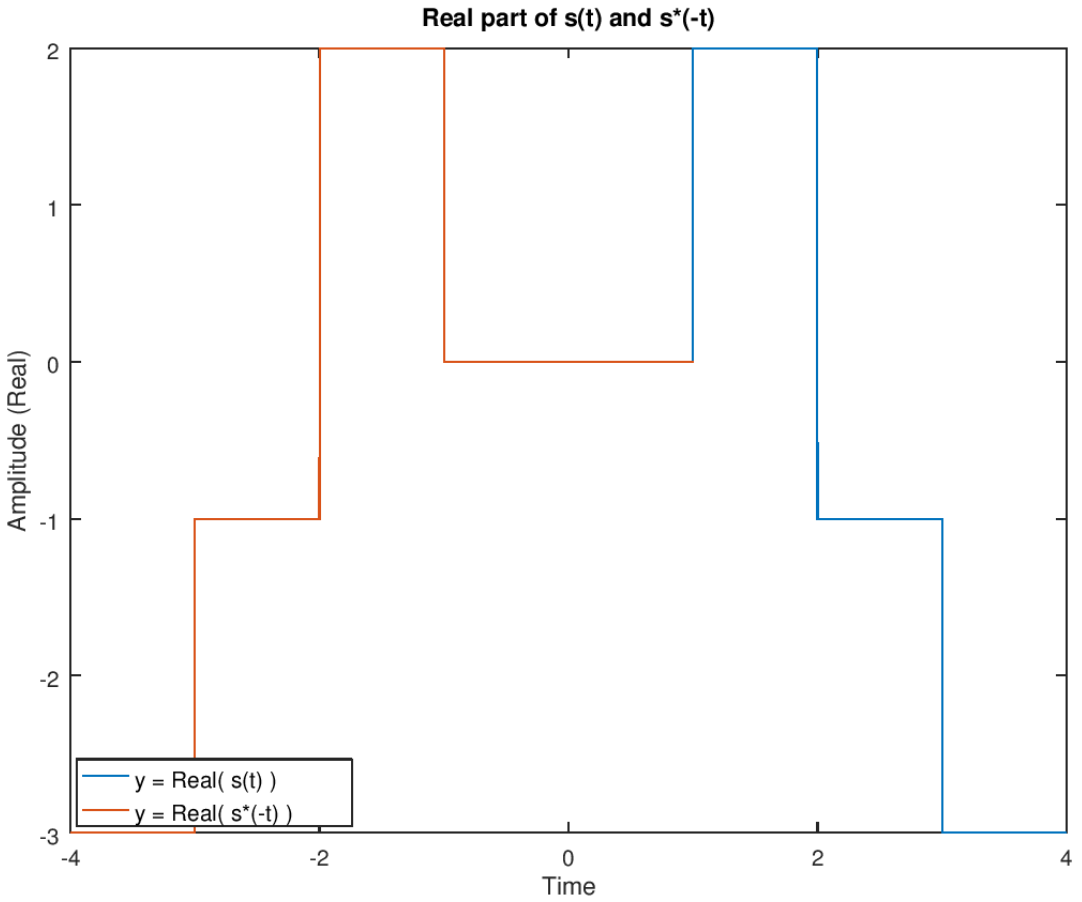
3B)

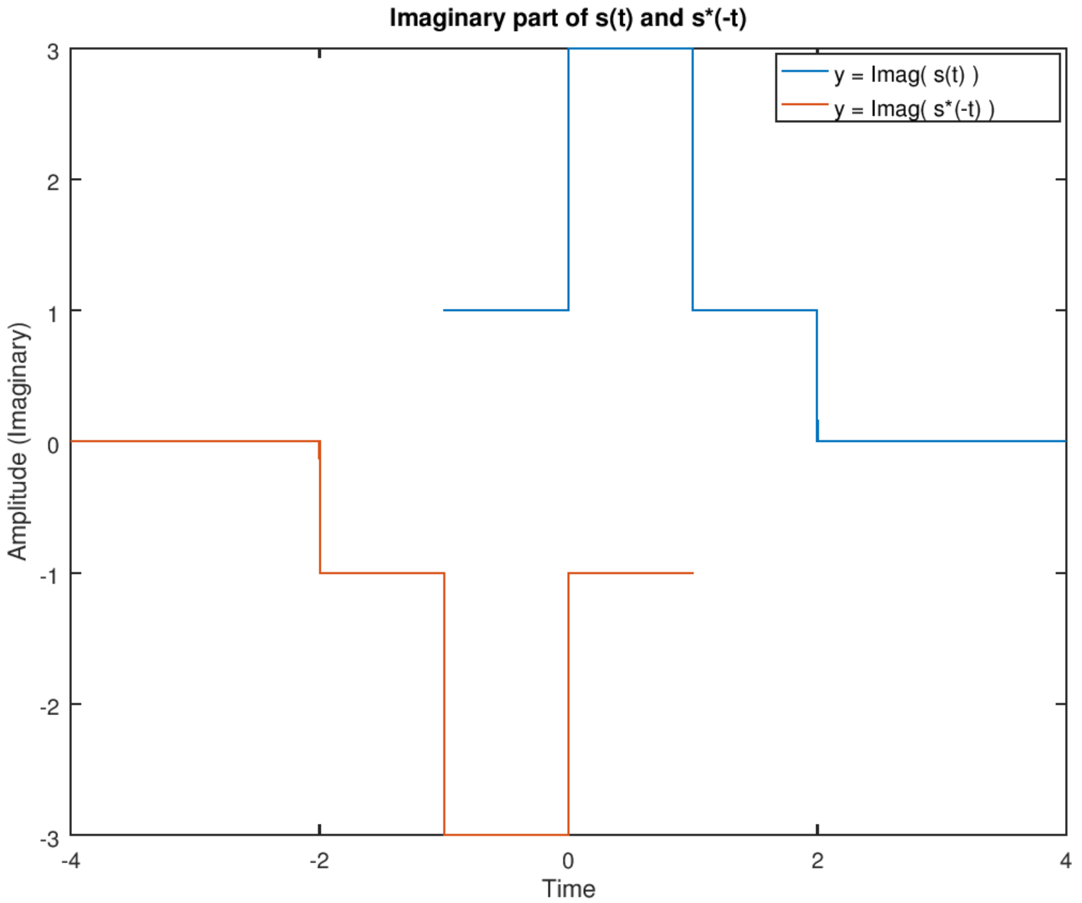


Signal peaked at 0 and had a magnitude of 14. (0, 14).

```
% Code:
% https://pastebin.com/MxKK2s13

function [convolution, time] = contconv (x1, x2, t1, t2, dt)
  % continous convolution
  Tstart1 = t1;
  Tstop1 = t1 + length(x1)*dt - dt;

  Tstart2 = t2;
  Tstop2 = t2 + length(x2)*dt - dt;

  startTime = Tstart1 + Tstart2;
  endTime = Tstop1 + Tstop2;

  time = startTime:dt:endTime;
  t = 1
  convolution = conv(x1,x2).*dt;
endfunction

function [time, signal, signal_filtered, time_filtered] =
     getSignalPair ()
  % get signal and filtered signal
  resolution = 0.001;

  first = ones(1/resolution,1);
  first = first.*(2);

  second = ones(1/resolution, 1);
  second = second.*(-1);

  third = ones((1/resolution)+1, 1);
  third = third.*(-3);

  signal = vertcat(first, second, third); % u(t)
  time = [1:resolution:4];

  signal_filtered = signal(end:-1:1); % reversing magnitudes
  time_filtered = time(end:-1:1).*(-1); %reverse and multiply my
     -1
endfunction

[time, signal, signal_filtered, time_filtered] = getSignalPair();

[convoluted_signal, convoluted_time] = contconv(signal,
     signal_filtered, time(1), time_filtered(1), 0.001);

plot(convoluted_time, convoluted_signal);

xlabel("time");
ylabel("magnitude");
title("Convolution of u(t) and u(-t)");
```

3C)

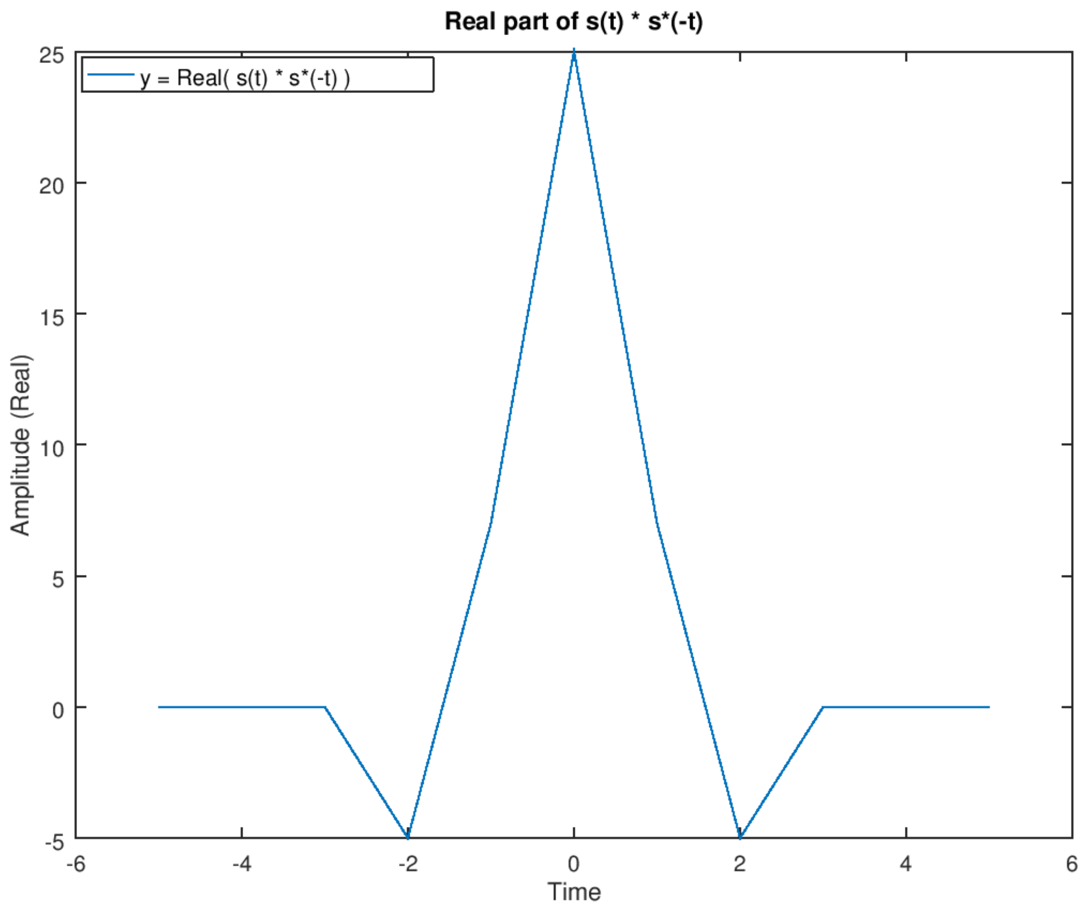

Real part of s(t) and s*(-t)
Amplitude (Real)
Time
y = Real(s(t))
y = Real(s*(-t))
2
1
0
-1
-2
-3
-4
-2
0
2
4

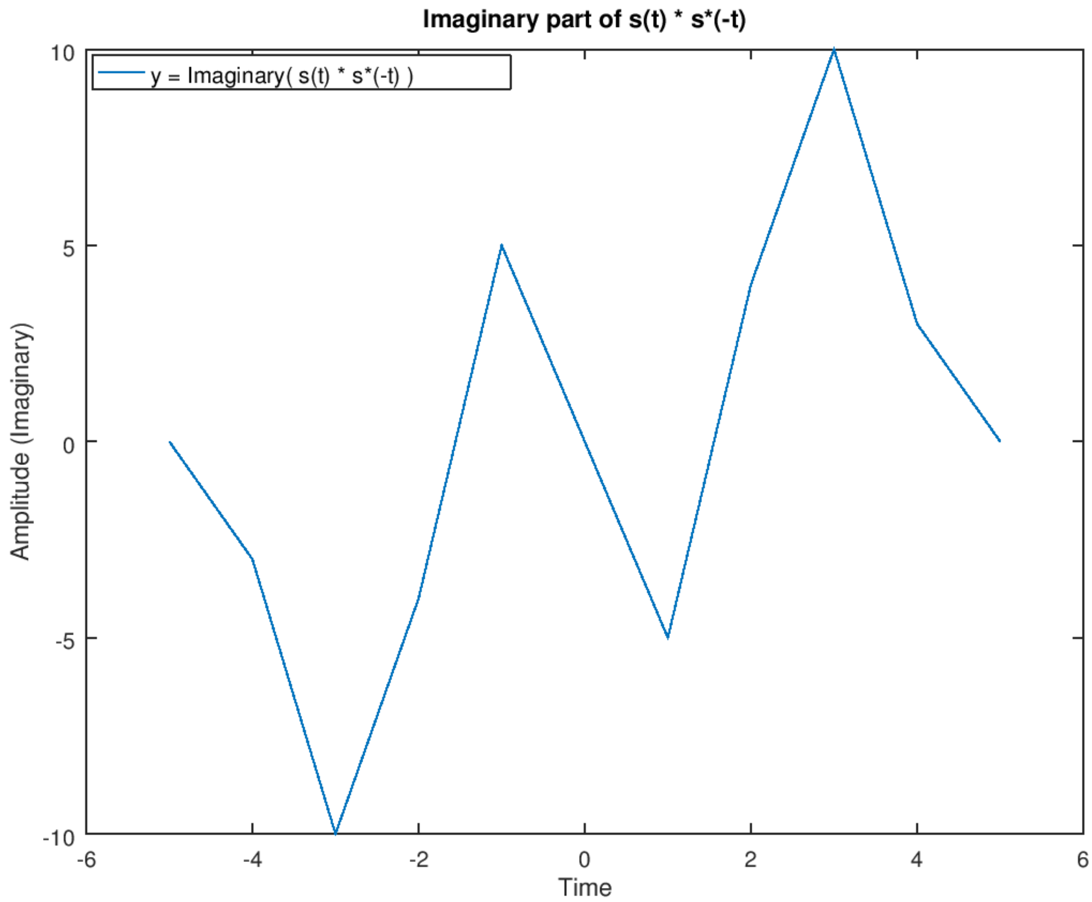

Imaginary part of s(t) and s*(-t)
Amplitude (Imaginary)
Time
y = Imag(s(t))
y = Imag(s*(-t))
3
2
1
0
-1
-2
-3
-4
-2
0
2
4

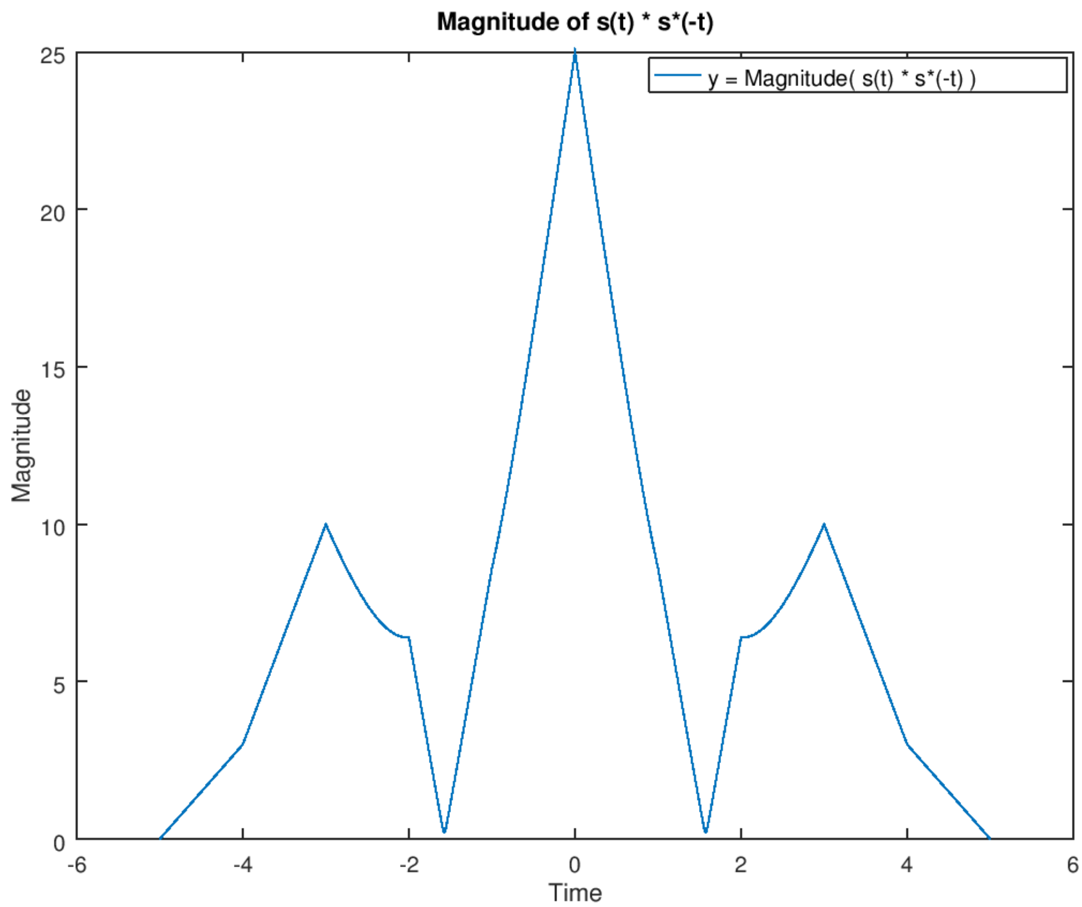
```
%Code:

% https://pastebin.com/szyzg5ND

function [mags] = getMagnitudes (dt, scalarArray)
  current = ones(1/dt, 1).*scalarArray(1);
  if(length(scalarArray) == 1)
     mags = current;
     return;
  end
  [next] = getMagnitudes(dt, scalarArray(2:1:end));
  mags = vertcat(current, next);
endfunction

function [times, magnitudes] = getSignal(Tstart, Tend, dt,
     scalars)
  % generates signals made of square waves
  times = Tstart:dt:Tend-dt;
  magnitudes = getMagnitudes(dt, scalars);
endfunction

function [times, magnitudes] = timeInvert(timeVct, magnitudeVct)
  magnitudes = magnitudeVct(end:-1:1);
  times = timeVct(end:-1:1).*(-1);
endfunction

% v(t) = I[-1, 2] + 2I[0, 1]
% v(t) = I[-1, 0] + 3I[0, 1] + I[1, 2]
%
% u(t) = 2I[1, 3] - 3I[2, 4]
% u(t) = 2I[1, 2] - I[2, 3] - 3I[3, 4]
%
% s(t) = u(t) + j*v(t)
% s(t) = (j)I[-1, 0] + (3j)I[0, 1] + (2+j)I[1, 2] - I(2, 3) -
     3I(3, 4)

resolution = 0.001
[time, signal] = getSignal(-1, 4, resolution, [i, 3i, 2 + i, -1,
     -3]);

[time2, signal2] = timeInvert(time, signal);
signal2 = conj(signal2);

Rsignal = real(signal);
Rsignal2 = real(signal2);

Isignal = imag(signal);
Isignal2 = imag(signal2);
```

```
plot(time, Rsignal, time2, Rsignal2);
xlabel("Time");
ylabel("Amplitude (Real)")
title("Real part of s(t) and s*(-t)");
legend("y = Real( s(t) )", "y = Real( s*(-t) )", "Location",
     "southwest")

%plot(time, Isignal, time2, Isignal2);
%xlabel("Time");
%ylabel("Amplitude (Imaginary)")
%title("Imaginary part of s(t) and s*(-t)");
%legend("y = Imag( s(t) )", "y = Imag( s*(-t) )")
```

3D)



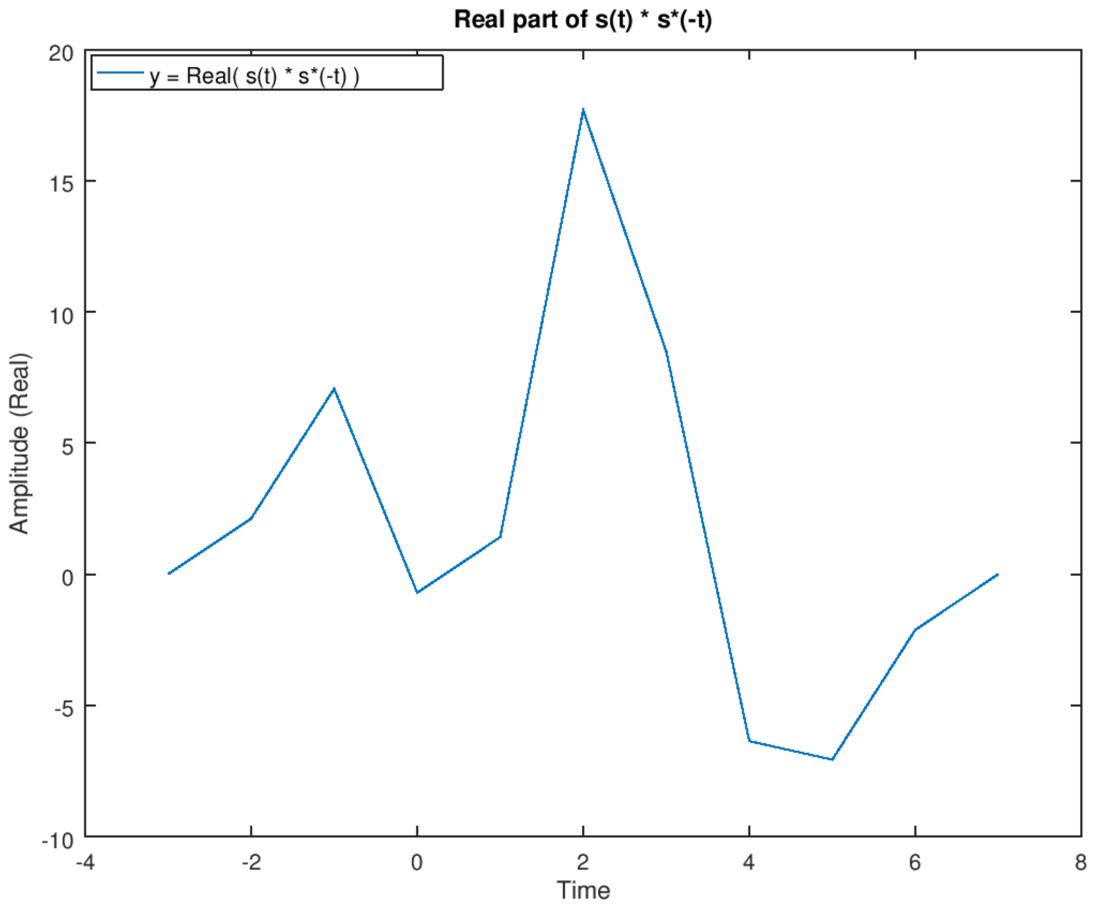

Imaginary part of s(t) * s*(-t)
y = Imaginary(s(t) * s*(-t))
Amplitude (Imaginary)
Time
10
5
0
-5
-10
-6
-4
-2
0
2
4
6

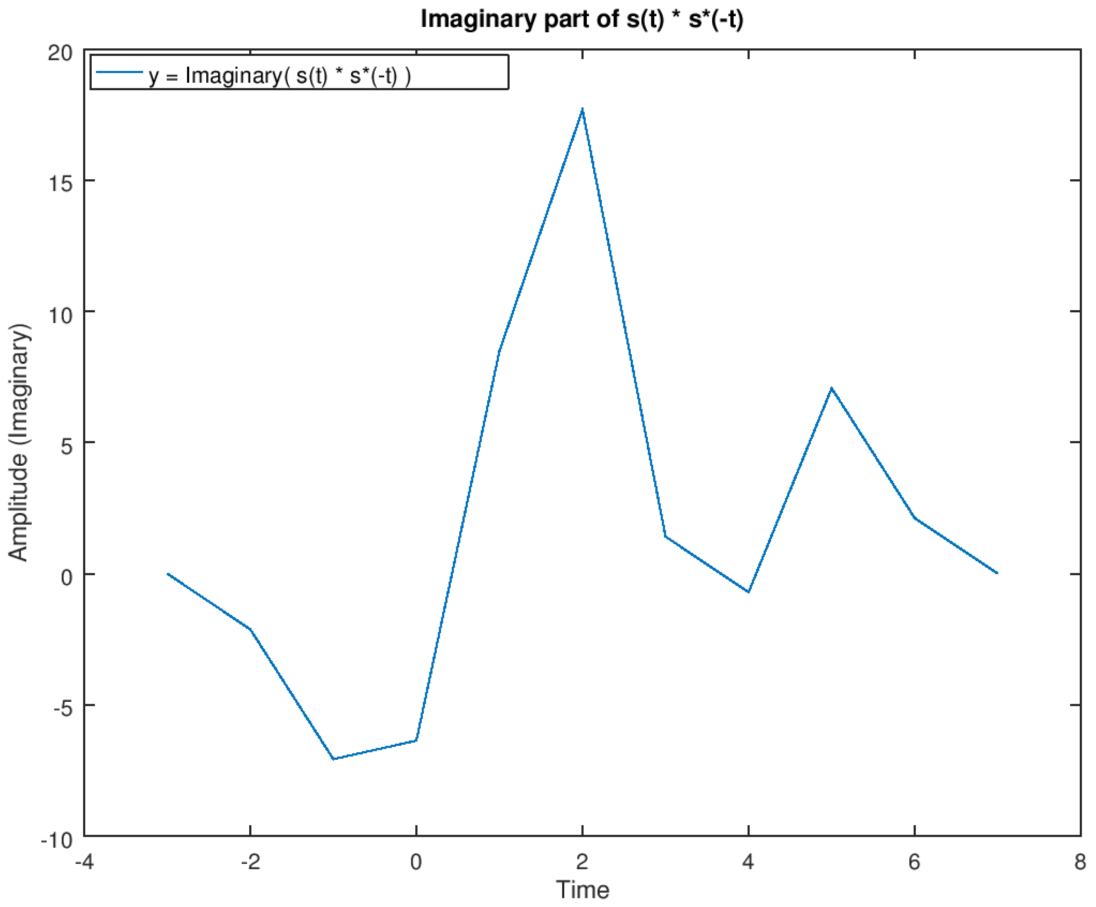

Magnitude of s(t) * s*(-t)
y = Magnitude(s(t) * s*(-t))
Magnitude
Time
25
20
15
10
5
0
-6
-4
-2
0
2
4
6

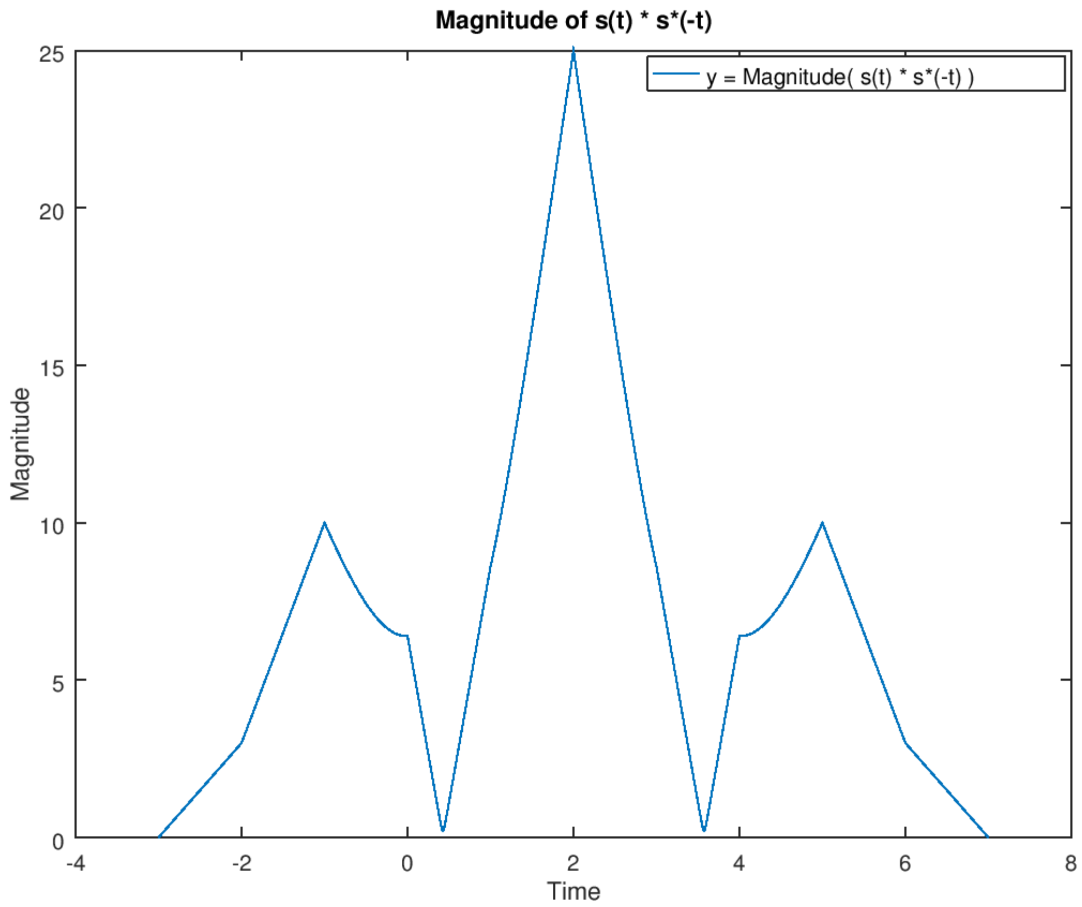
```
% Code:


% https://pastebin.com/YWuKddcJ

function [convolution, time] = contconv (x1, x2, t1, t2, dt)
  % continous convolution
  Tstart1 = t1;
  Tstop1 = t1 + length(x1)*dt - dt;

  Tstart2 = t2;
  Tstop2 = t2 + length(x2)*dt - dt;

  startTime = Tstart1 + Tstart2;
  endTime = Tstop1 + Tstop2;

  time = startTime:dt:endTime;
  convolution = conv(x1,x2).*dt;
endfunction

function [mags] = getMagnitudes (dt, scalarArray)
  current = ones(1/dt, 1).*scalarArray(1);
  if(length(scalarArray) == 1)
     mags = current;
     return;
  end
  [next] = getMagnitudes(dt, scalarArray(2:1:end));
  mags = vertcat(current, next);
endfunction

function [times, magnitudes] = getSignal(Tstart, Tend, dt,
     scalars)
  % generates signals made of square waves
  times = Tstart:dt:Tend-dt;
  magnitudes = getMagnitudes(dt, scalars);
endfunction

function [times, magnitudes] = timeInvert(timeVct, magnitudeVct)
  magnitudes = magnitudeVct(end:-1:1);
  times = timeVct(end:-1:1).*(-1);
endfunction

% v(t) = I[-1, 2] + 2I[0, 1]
% v(t) = I[-1, 0] + 3I[0, 1] + I[1, 2]
%
% u(t) = 2I[1, 3] - 3I[2, 4]
% u(t) = 2I[1, 2] - I[2, 3] - 3I[3, 4]
%
% s(t) = u(t) + j*v(t)
% s(t) = (j)I[-1, 0] + (3j)I[0, 1] + (2+j)I[1, 2] - I(2, 3) -
     3I(3, 4)
```

```
resolution = 0.001
[time, signal] = getSignal(-1, 4, resolution, [i, 3i, 2 + i, -1,
     -3]);

[time2, signal2] = timeInvert(time, signal);
signal2 = conj(signal2);

[signalC, timeC] = contconv(signal, signal2, time, time2,
     resolution);

plot(timeC, real(signalC));
xlabel("Time");
ylabel("Amplitude (Real)")
title("Real part of s(t) * s*(-t)");
legend("y = Real( s(t) * s*(-t) )", "Location", "northwest")

%plot(timeC, imag(signalC));
%xlabel("Time");
%ylabel("Amplitude (Imaginary)")
%title("Imaginary part of s(t) * s*(-t)");
%legend("y = Imaginary( s(t) * s*(-t) )", "Location", "northwest")

%plot(timeC, abs(signalC));
%xlabel("Time");
%ylabel("Magnitude")
%title("Magnitude of s(t) * s*(-t)");
%legend("y = Magnitude( s(t) * s*(-t) )", "Location", "northeast")
```

3E)


Real part of s(t) * s*(-t)
y = Real(s(t) * s*(-t))
Amplitude (Real)
Time
20
15
10
5
0
-5
-10
-4
-2
0
2
4
6
8


Imaginary part of s(t) * s*(-t)
y = Imaginary(s(t) * s*(-t))
Amplitude (Imaginary)
Time
20
15
10
5
0
-5
-10
-4
-2
0
2
4
6
8

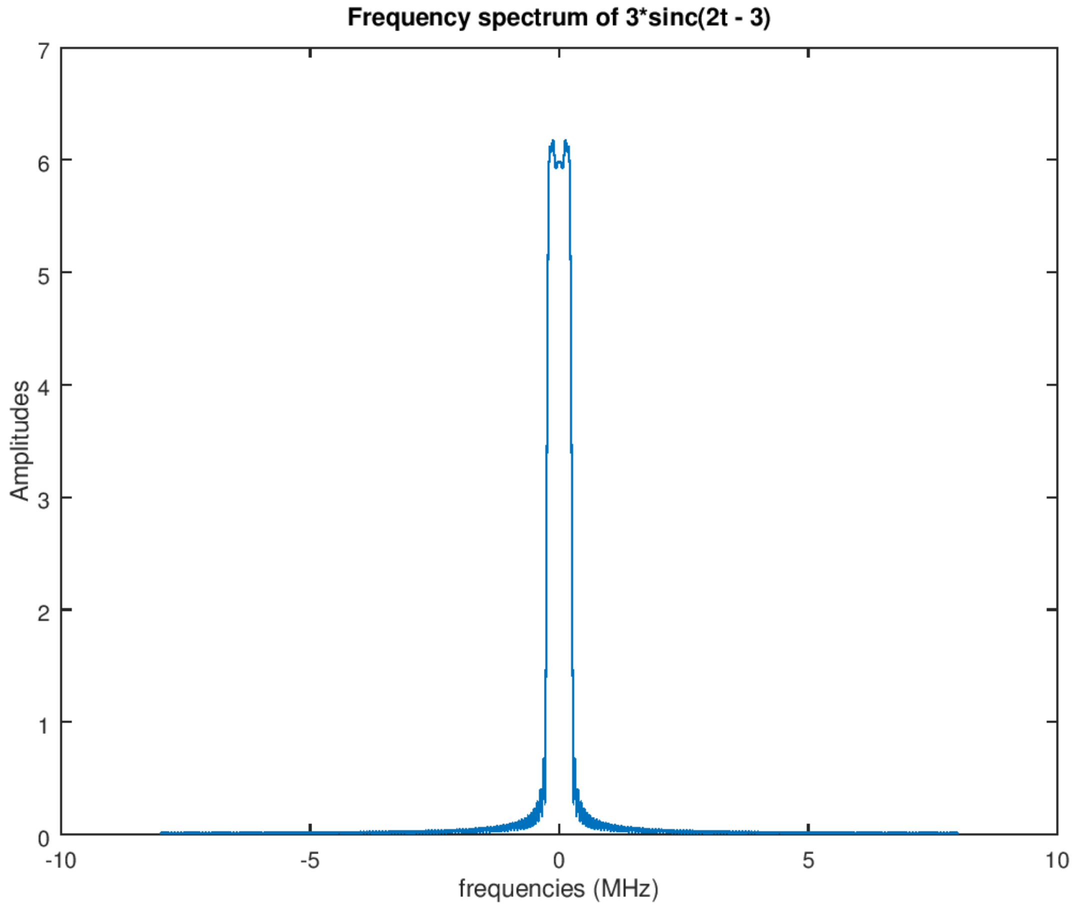



```
% Code

% https://pastebin.com/z0j9TRiR

function [convolution, time] = contconv (x1, x2, t1, t2, dt)
  % continous convolution
  Tstart1 = t1;
  Tstop1 = t1 + length(x1)*dt - dt;

  Tstart2 = t2;
  Tstop2 = t2 + length(x2)*dt - dt;

  startTime = Tstart1 + Tstart2;
  endTime = Tstop1 + Tstop2;

  time = startTime:dt:endTime;
  convolution = conv(x1,x2).*dt;
endfunction

function [mags] = getMagnitudes (dt, scalarArray)
```

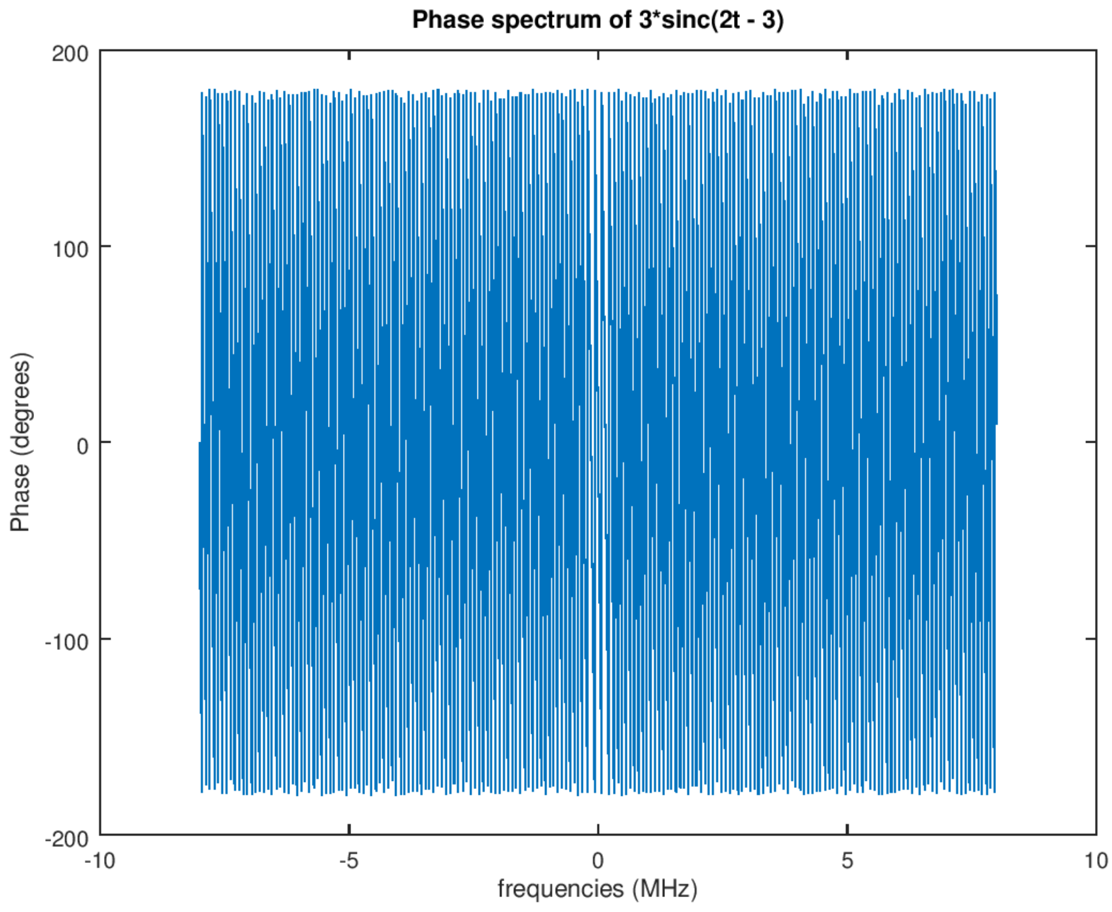
```
  current = ones(1/dt, 1).*scalarArray(1);
  if(length(scalarArray) == 1)
     mags = current;
     return;
  end
  [next] = getMagnitudes(dt, scalarArray(2:1:end));
  mags = vertcat(current, next);
endfunction

function [times, magnitudes] = getSignal(Tstart, Tend, dt,
     scalars)
  % generates signals made of square waves
  times = Tstart:dt:Tend-dt;
  magnitudes = getMagnitudes(dt, scalars);
endfunction

function [times, magnitudes] = timeInvert(timeVct, magnitudeVct)
  magnitudes = magnitudeVct(end:-1:1);
  times = timeVct(end:-1:1).*(-1);
endfunction

% v(t) = I[-1, 2] + 2I[0, 1]
% v(t) = I[-1, 0] + 3I[0, 1] + I[1, 2]
%
% u(t) = 2I[1, 3] - 3I[2, 4]
% u(t) = 2I[1, 2] - I[2, 3] - 3I[3, 4]
%
% s(t) = u(t) + j*v(t)
% s(t) = (j)I[-1, 0] + (3j)I[0, 1] + (2+j)I[1, 2] - I(2, 3) -
     3I(3, 4)

resolution = 0.001
[time, signal] = getSignal(-1, 4, resolution, [i, 3i, 2 + i, -1,
     -3]);

[time2, signal2] = timeInvert(time, signal);
signal2 = conj(signal2);

timeS = time.+2;  %timeshifting.
signalS = signal.*exp(i*pi/4);

[signalC, timeC] = contconv(signalS, signal2, timeS, time2,
     resolution);


plot(timeC, real(signalC));
xlabel("Time");
ylabel("Amplitude (Real)")
title("Real part of s(t) * s*(-t)");
legend("y = Real( s(t) * s*(-t) )", "Location", "northwest")

%plot(timeC, imag(signalC));
```

```
%xlabel("Time");
%ylabel("Amplitude (Imaginary)")
%title("Imaginary part of s(t) * s*(-t)");
%legend("y = Imaginary( s(t) * s*(-t) )", "Location", "northwest")

%plot(timeC, abs(signalC));
%xlabel("Time");
%ylabel("Magnitude")
%title("Magnitude of s(t) * s*(-t)");
%legend("y = Magnitude( s(t) * s*(-t) )", "Location", "northeast")
```

4A & 4B)



Frequency spectrum of 3*sinc(2t - 3)
Amplitudes
frequencies (MHz)
0
1
2
3
4
5
6
7
-10
-5
0
5
10



Phase spectrum of 3*sinc(2t - 3)
Phase (degrees)
frequencies (MHz)
200
100
0
-100
-200
-10
-5
0
5
10

The range of frequencies for which magnitude is not 0 matters.

Code:

```
function [X,f,df] = contFT(x,tstart,dt,df_desired)
%Use Matlab DFT for approximate computation of continuous time
     Fourier
%transform
%INPUTS
%x = vector of time domain samples, assumed uniformly spaced
%tstart= time at which first sample is taken
%dt = spacing between samples
%df_desired = desired frequency resolution
%OUTPUTS
%X=vector of samples of Fourier transform
%f=corresponding vector of frequencies at which samples are
     obtained %df=freq resolution attained (redundant--already
     available from %difference of consecutive entries of f)
%%%%%%%%%
%minimum FFT size determined by desired freq
Nmin=max(ceil(1/(df_desired*dt)),length(x));
%choose FFT size to be the next power of 2
Nfft = 2^(nextpow2(Nmin))
%compute Fourier transform, centering around
X=dt*fftshift(fft(x,Nfft));
%achieved frequency resolution
df=1/(Nfft*dt)
%range of frequencies covered
f = ((0:Nfft-1)-Nfft/2)*df; %same as f=-1/(2*dt):df:1/(2*dt) - df
%phase shift associated with start time
X=X.*exp(-j*2*pi*f*tstart);
end

sampling_freq = 16; % per micro second
dt = 1/sampling_freq;

temp_time = -8:dt:8; % -8 to 8 micro seconds
temp_time = temp_time.*0.5;
time = temp_time; % 2t - 3

temp_signal = sinc(time.-1.5);
signal = temp_signal.*3; % 3sinc(2t - 3)

df_desired = 10^-3 % 0.001 per micro second = 1 KHz.

%plot(time, signal);

[amplitudes, frequencies, df] = contFT(signal, time(1), dt,
     df_desired);

plot(frequencies, abs(amplitudes));
```

```
%plot(frequencies, amplitudes)

xlabel("frequencies (MHz)");
ylabel("Amplitudes");
title("Frequency spectrum of 3*sinc(2t - 3)")

%plot(frequencies, rad2deg(angle(amplitudes)));
%xlabel("frequencies (MHz)");
%ylabel("Phase (degrees)");
%title("Phase spectrum of 3*sinc(2t - 3)")
```