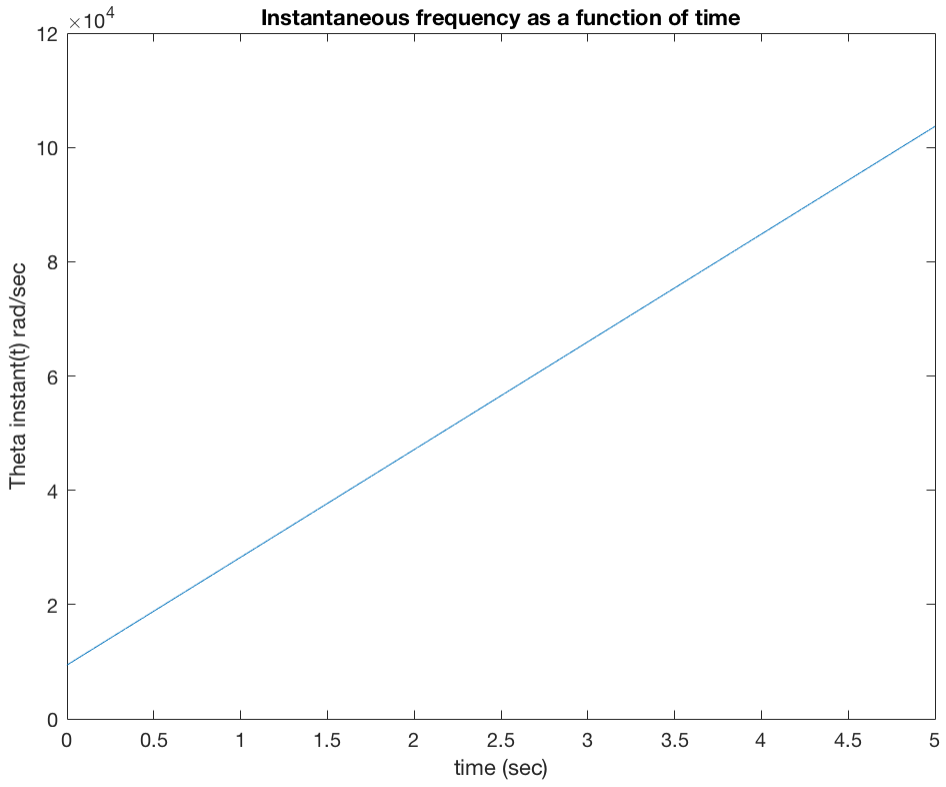
Part 1: Aliasing due to undersampling

Creating the vector and initializing variables

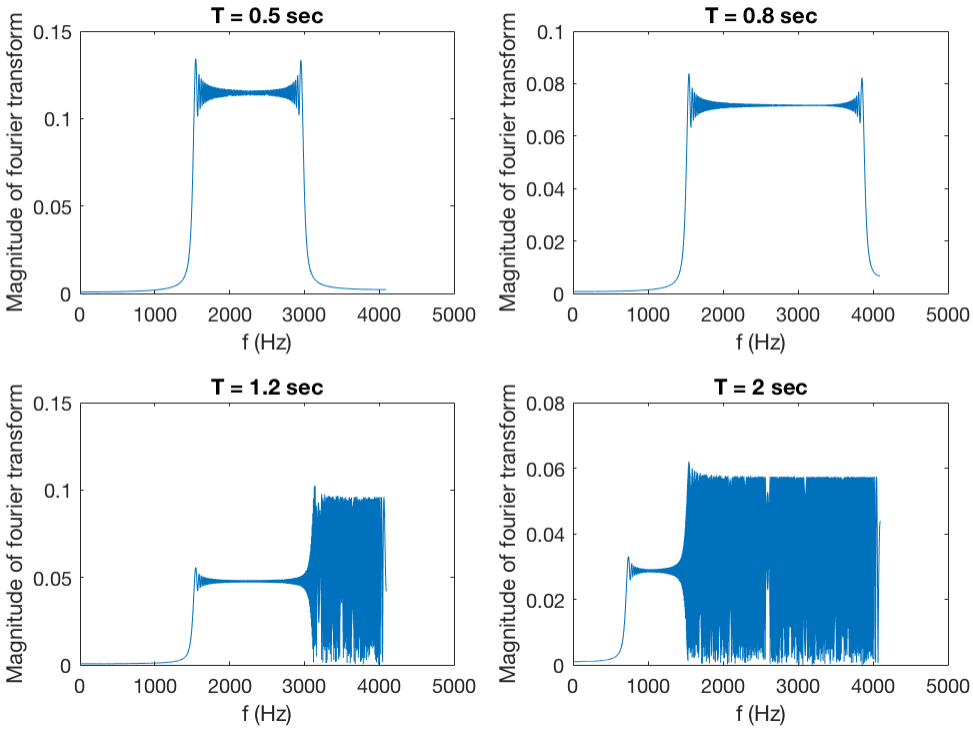
```
Omega = 2*pi*1500;
fs = 8192;
T = 2;
t = linspace(0,T,T*fs);
Beta = 2*pi*3000;

x = sin(Omega*t + 0.5*Beta*t.^2);

% Play the sound out loud
% soundsc(x);

% The sound that plays out sounds like it increases and then decreases
 in
% frequency like a bell curve kind of. It sounds like it hits its
 maximum
% pitch about halfway through the vector.

% Creating the second sound of 5 seconds
T2 = 5;
t2 = linspace(0,T2,T2*fs);
Beta = 2*pi*3000;

x2 = sin(Omega*t2 + 0.5*Beta*t2.^2);

% Play the sound out
% soundsc(x2)

% This sound sounds like it hits its minimum frequency twice, and is
% different from the one from the first time period.
inst = Omega + Beta*t2;
figure(1);
plot(t2,inst);
title('Instantaneous frequency as a function of time');
xlabel('time (sec)');
ylabel('Theta instant(t) rad/sec');

% Looking at the plot for the instantaneous frequency and the equation
 that
% that it is derived from the minimum should be at time t=0 and the
 maximum
% should be at the last time in the vector (in this case at t=5 sec).
 The
% reason why the sound is so different and the output from the vector
 is so
% different is because when frequencies past the sampling frequency
 are
% trying to be picked up in discrete time, there will be overlap in
 the
```

```
% fourier domain and some of the data will be lost (you'll miss some
 of the
% points you should have picked up) a visual equivalent of this is
 apparent
% in the video we watched in lecture of the plane's propellor and how
 it
% appears to be spinning backwards because of the sample rate of the
 camera

T3 = 0.5;
t3 = linspace(0,T3,T3*fs);
Beta = 2*pi*3000;
x3 = sin(Omega*t3 + 0.5*Beta*t3.^2);

T4 = 0.8;
t4 = linspace(0,T4,T4*fs);
Beta = 2*pi*3000;
x4 = sin(Omega*t4 + 0.5*Beta*t4.^2);

T5 = 1.2;
t5 = linspace(0,T5,T5*fs);
Beta = 2*pi*3000;
x5 = sin(Omega*t5 + 0.5*Beta*t5.^2);

X1 = ctft(x,fs);
X1p = X1(length(X1)/2+1:end);
X3 = ctft(x3,fs);
X3p = X3(length(X3)/2+1:end);
X4 = ctft(x4,fs);
X4p = X4((length(X4)+3)/2:end);
X5 = ctft(x5,fs);
X5p = X5(length(X5)/2+1:end);

f1 = (0:(length(X1)-1)/2)*fs/length(X1);
f3 = (0:(length(X3)-1)/2)*fs/length(X3);
f4 = (0:length(X4)/2-1)*fs/length(X4);
f5 = (0:(length(X5)-1)/2)*fs/length(X5);

figure(2);
subplot(2,2,4);
plot(f1,abs(X1p));
title('T = 2 sec');
xlabel('f (Hz)');
ylabel('Magnitude of fourier transform');

subplot(2,2,1);
plot(f3,abs(X3p));
title('T = 0.5 sec');
xlabel('f (Hz)');
ylabel('Magnitude of fourier transform');

subplot(2,2,2);
plot(f4,abs(X4p));
title('T = 0.8 sec');
```

```
xlabel('f (Hz)');
ylabel('Magnitude of fourier transform');

subplot(2,2,3);
plot(f5,abs(X5p));
title('T = 1.2 sec');
xlabel('f (Hz)');
ylabel('Magnitude of fourier transform');

suptitle('Magnitude of fourier transforms for signals of various
 lengths');

% As the chirp length changes the frequency domain of the function
 spreads
% out. This makes sense because parsevals relation must be preserved
 so the
% amplitude decreases as the width increases. It also makes sense
 because
% more higher frequency components are added in a linear relationship
 as
% suggested by the instantaneous frequency equation. You can also see
 the
% aliasing start to happen in the fourier domain where once the
% instantaneous frequency passes half of the sampling frequency there
% starts to become a mess on the right side of the graph (this would
 be the
% same on the left side because of the symmetry) due to the overlap
 from
% the frquencies cut off past 4096 Hz. This is most apparent in the
% transition between T=0.8 and T=1.2 and T=2.
```



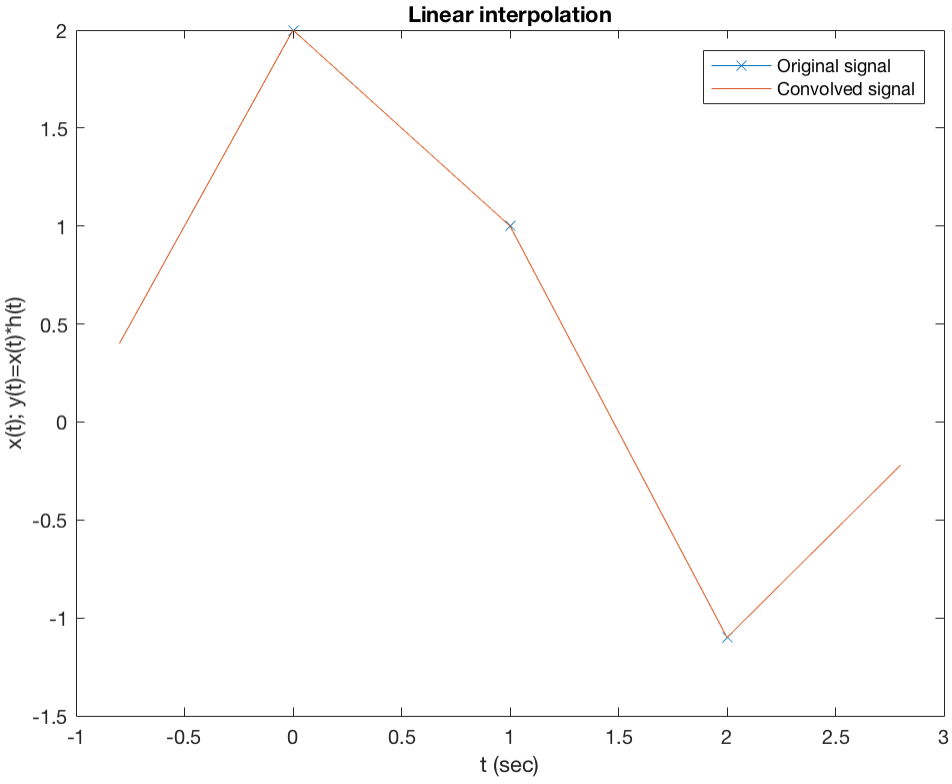
Magnitude of fourier transforms for signals of various lengths

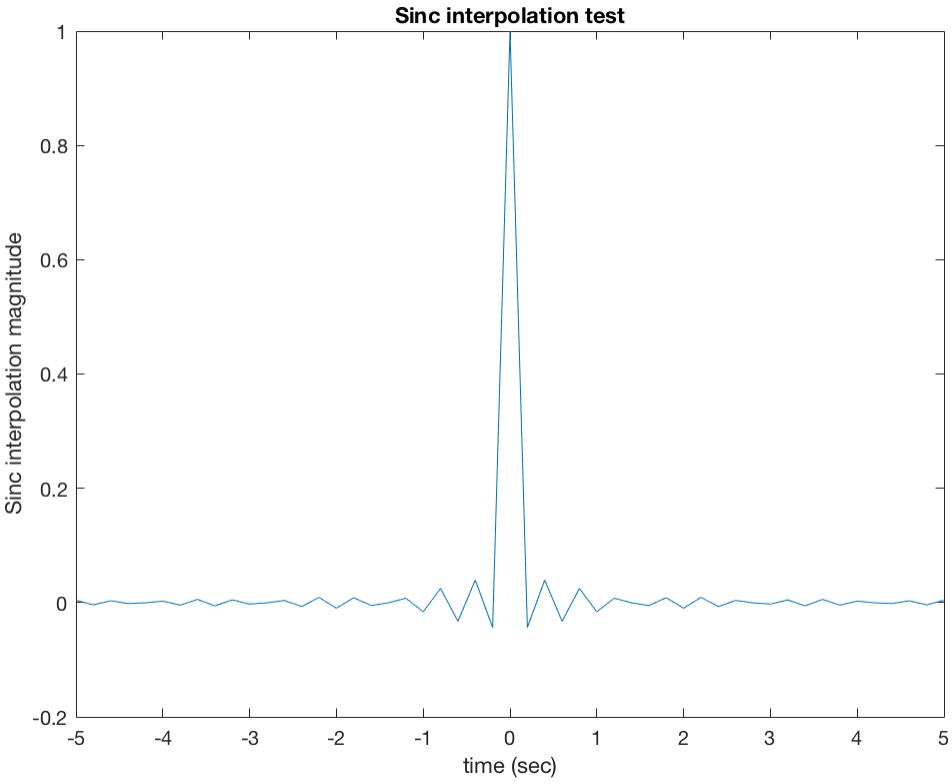


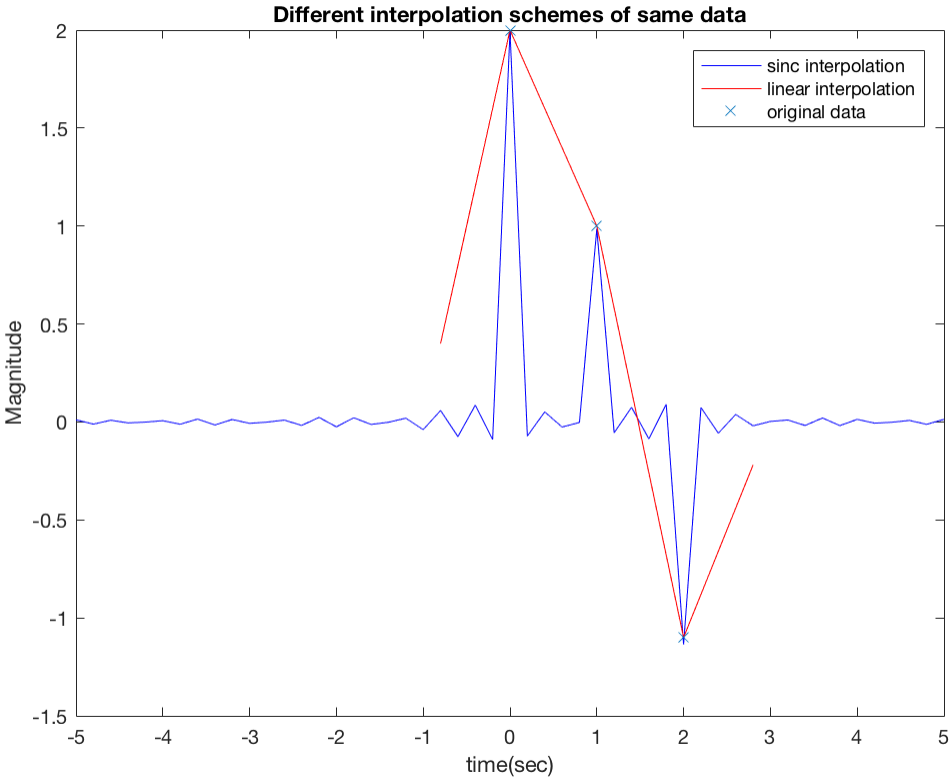
Part 2: Reconstruction

```
y = [2 1 -1.1];
t0 = 0:2;

yp = [2 0 0 0 0 1 0 0 0 0 -1.1];
h = [1 2 3 4 5 4 3 2 1]*.2;
ypp = conv(h,yp);
tp = linspace(-(length(h)-1)*0.2/2,2.8,length(ypp));

figure(3);
plot(t0,y,'-x'), hold on, plot(tp,ypp);
title('Linear interpolation');
xlabel('t (sec)');
ylabel('x(t); y(t)=x(t)*h(t)');
legend('Original signal','Convolved signal');

% To correct for the difference in the time for the convolution, you
 just
% have to use the amount of the left and right overlap for the filter,
% which works out to (length(h)-1)/2 scaled by the sampling period, in
 this
% case 0.2 seconds.
% Yes the result does match what you would expect for the linear
% interpolation! And if you assume the linearity to be accurate you
 can
% expect the values of x(-1) = x(3) = 0 by following the slope of the
 line.

% Sinc Interpolation
T = 0.2;
t = -5:T:5;

test = 1;

testout = sincInterp(test,T,t);
figure(4)
plot(t,testout)
title('Sinc interpolation test');
xlabel('time (sec)');
ylabel('Sinc interpolation magnitude');

% Yes, the test looks like a sinc function (but the sampling period is
 so
% large that is looks super chunky

out = sincInterp(yp,T,t);
figure(5)
plot(t,out,'-b',tp,ypp,'-r',t0,y,'x')
title('Different interpolation schemes of same data')
ylabel Magnitude; xlabel time(sec);
legend('sinc interpolation','linear interpolation','original data');
```

```
% In my personal opinion I think that the linear interpolation is a
 more
% natural looking wave than that of the sinc interpolation.

% For this case it can be assumed that the signal is band limited from
 0 to
% 2.5 Hz (not including symmetric negative frequencies to avoid
 redundancy)
% because the sample period is 0.2 seconds so the frequency is 5 Hz,
 and
% the sampled signal must be less than half the sampling frequency so
 the
% signal is band limited to 2.5 Hz

type('sincInterp.m')


function y = sincInterp(x,T,t)
% This function takes in a vector x, period T, and time vector t and
% creates a sinc interpolation y from the data

L = length(t);

y = zeros(1,L);

% Assuming causal signal so can start at n=0
for n = 0:length(x)-1
    d = x(n+1)*sinc(pi*(t-n*T)/T);
    y = y+d;
end

return
```


Linear interpolation
Original signal
Convolved signal
x(t); y(t)=x(t)*h(t)
t (sec)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3


Sinc interpolation test
Sinc interpolation magnitude
time (sec)
1
0.8
0.6
0.4
0.2
0
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5



Published with MATLAB® R2016b