7.1: Aliasing due to Undersampling

This exercise covers the effects of aliasing due to sampling on signals reconstructed by bandlimited interpolation. If a continuous-time signal $x(t)$ is sampled every T seconds, then its samples form the discrete-time sequence $x[n] = x(nT)$. The Nyquist sampling theorem states that if $x(t)$ has bandwidth less than $\Omega_s = 2\pi/T$, i.e., $X(j\Omega) = 0$ for $|\Omega| > \Omega_s/2$, then $x(t)$ can be completely reconstructed from its samples $x(nT)$. The bandlimited interpolation or signal reconstruction is most easily visualized by first multiplying $x(t)$ by and impulse train

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT).$$

The signal $x(t)$ can be recovered from $x_p(t)$ by filtering $x_p(t)$ with an ideal lowpass filter with cutoff frequency $\Omega_s/2$. Define $x_r(t)$ to be the reconstructed signal given by lowpass filtering $x_p(t)$. If the bandwidth of $x(t)$ is greater than Ω_s, then the samples $x(nT)$ do not completely determine $x(t)$, and $x_r(t)$ will not generally be queal to $x(t)$. In the following problems, you will examine the effects of undersampling a pure sinusoid and a chirp signal.

Basic Problems

Consider the sinusoidal signal

$$x(t) = sin(\Omega_0 t)$$

If $x(t)$ is sampled with frequency $\Omega_s = 2\pi/T$ rad/sec, then the discrete-time signal $x[n] = x(nT)$ is equal to

$$x[n] = sin(\Omega_0 nT).$$

Assume the sampling frequency is fixed at $\Omega_s = 2\pi(8192)$ rad/sec.

7.1:Part A

(a). Assume $\Omega_0 = 2\pi(1000)$ rad/sec and define T=1/8192. Create the vector n=[0:8191], so that $t = n * T$ contains the 8192 time samples of the interval $0 \leq t < 1$. Create a vector x which contains the samples of $x(t)$ at the time samples in t.

```matlab
%Part 1: Project 7.1, parts (a)

Omega_0 = 2*pi*1000;
T= 1/8192;
n=0:8191;
t=n*T;
x = sin(Omega_0*t); % x(t)
```

Code 7.1-1: matlab script for Part A

7.1:Part B

(b). Display the first fifty samples of x[n] versus n using stem. Display the first fifty samples of $x(t)$ versus the sampling times using plot. (use subplot to simuyltaneously display these two plots.)

7.1:Part B continued on next page...

```
%Part 2: Project 7.1, parts (b)

%Sampling frequency is Omega_s = 2*pi*8192
%Part A
Omega_0 = 2*pi*1000;
T= 1/8192;
n=0:8191;
t=n*T;
x = sin(Omega_0*t); % x(t)
x_discrete = sin(Omega_0*n*T); % x[n]

%Part B
figure(1)

subplot(211)
stem(n,x_discrete)%x[n]
xlim([0 49]); %first fifty samples
title('Project 7.1, Part(b)')
xlabel('time samples')
ylabel('x[t] = sin(\Omega_0nT)')

subplot(212)
plot(n,x)
xlim([0 49]); %first fifty samples
xlabel('time samples')
ylabel('x(t) = sin(\Omega_0t)')

if FINALPLOTS
    print -deps proj71PartB.eps
end
```

Code 7.1-2: matlab script for Part B

Note that plot(t,x) displays a continueous-time signal given the samples in x, using straight lines to interpolate between sample values. While this interpolation si not generally equal to the bandlimited reconstruction which follows from the smapling theorem, it can often be a ver good approximation.
To compute samples of the continuous-time Fourier transform of the bandlimited reconstruction $x_r(t)$, use the following function:
function [X,f] = ctfts(x,T) N=length(x); X = fftshift(fft(x,N))*(2*pi/N); f = linspace(-1,1-1/N,N)/(2*T);
This function uses fft to calculate the Fourier transform of the reconstructed signal. The M-file ctfts.m is provided in the Cmoputer Explorations Toolbox, and should be placed in your MATLABPATH.

7.1:Part C

(c). Use [X,f]=ctfts(x,T) to calculate the continuoues-time Fourier transform of the reconstructed signal $x_r(t)$. Plot the magnitude of X versus f. Is X nonzero at the proper frequency values? (Note that almost all the elements in X are nonzero, but most are small values due to numerical round-off errors.) Is the phase of X correct, assuming that the phase is equal to zero when X is nearly zero, I.E., nonzero only due to reound-off error.

```
%Part 1: Project 7.1, parts (c)

%Part A
Omega_0 = 2*pi*1000;
T= 1/8192;
```

7.1:Part C continued on next page. . .

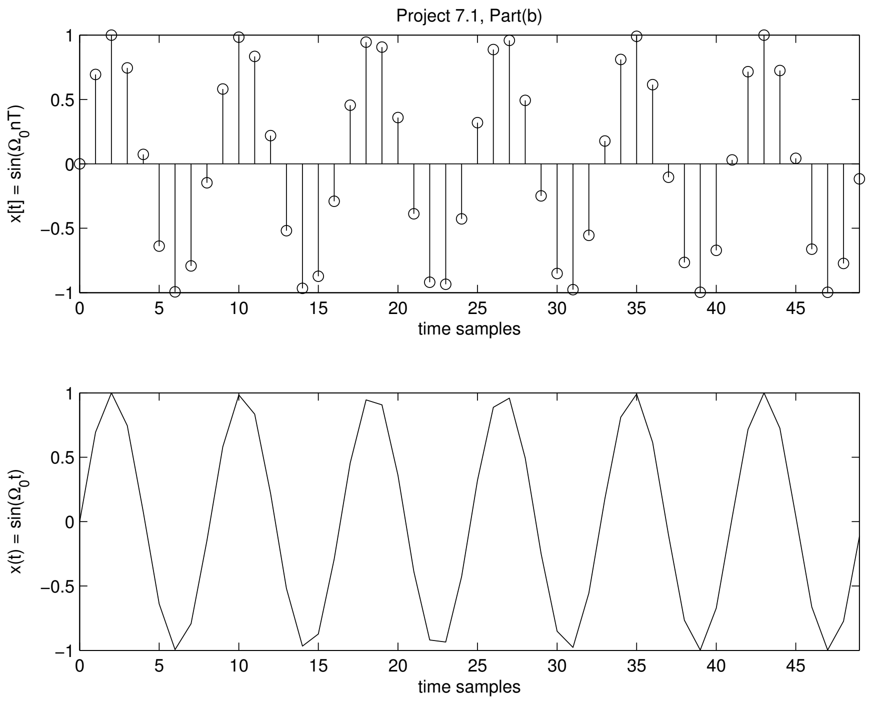

Figure 7.1-1: Output for 7.1 Part B

```
n=0:8191;
t=n*T;
x = sin(Omega_0*t); % x(t)

%Part C
[X,f]=ctfts(x,T);

figure(1)

plot(f,abs(X))
title('Project 7.1, Part(c)')
xlabel('time samples')
ylabel('x_r(t)  (CTFT)')


if FINALPLOTS
    print -deps proj71PartC.eps
end
```

Code 7.1-3: matlab script for Part C

Output:

Plot the magnitude of X versus f: see figure[!h] for the ouput
Is X nonzero at the proper frequency values?
Yes, the nonzero points were at 1000 and -1000 whihc corespond to $\Omega_0 = 2\pi(1000)$ where 1000 is the the bandlimit occurs.
Is the phase of X correct, assuming that the phase is equal to zero when X is nearly zero, I.E., nonzero only due to reound-off error.
Yes, the phase of X is correct.

7.1:Part C continued on next page. . .

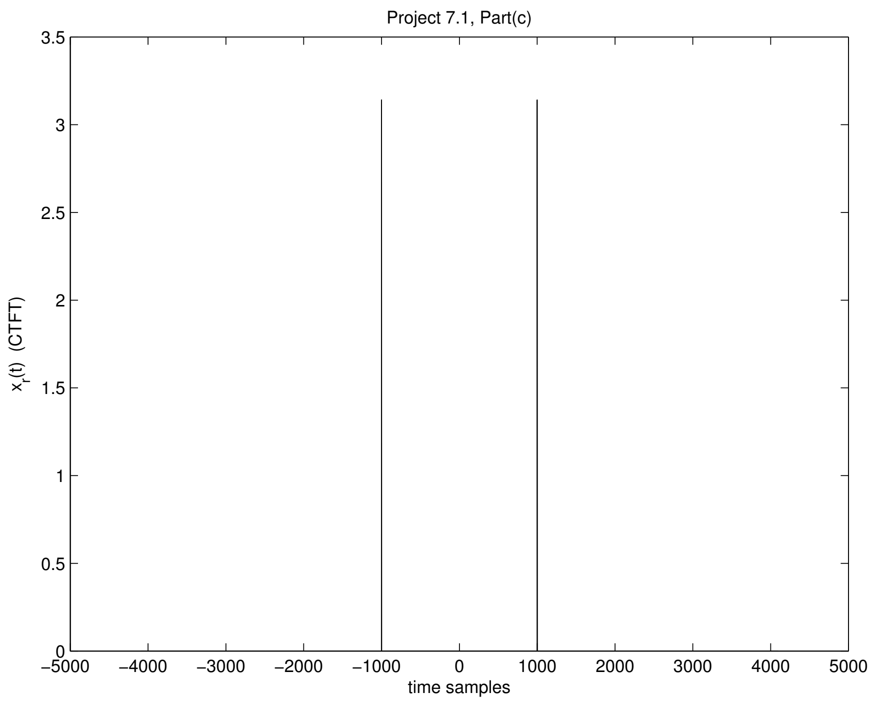

Figure 7.1-2: Output for 7.1 Part C

Intermediate Problems

you will now consider the effect of aliasing on the reconstructed signal $x_r(t)$.

7.1:Part D

(d). Repeat. Parts (a)-(c) for the sinusoidal frequencies $\Omega_0 = 2\pi(1500)$ and $2\pi(2000)$ rad/sec. Again, is the magnitude of X nonzero for the expected frequenies? Is the phase of X correct?

```
%Part 1: Project 7.1, parts (d)

%======================================
%Repeat (a)-(c) for omega_0 = 2pi(1500)
%======================================

Omega_0 = 2*pi*1500;
T= 1/8192;
n=0:8191;
t=n*T;
x = sin(Omega_0*t); % x(t)
x_discrete = sin(Omega_0*n*T); % x[n]


figure(1)
subplot(211)
stem(n,x_discrete)%x[n]
title('Project 7.1, Part(d) : \Omega_0 = 2 \pi (1500)')
xlim([0 49]); %first fifty samples
xlabel('time samples')
```

7.1:Part D continued on next page. . .

```
ylabel('x[t] = sin(\Omega_0nT)')

subplot(212)
plot(n,x)
xlim([0 49]); %first fifty samples
xlabel('time samples')
ylabel('x(t) = sin(\Omega_0t)')

if FINALPLOTS
    print -deps proj71PartD1.eps
end

[X,f]=ctfts(x,T);

figure(2)
plot(f,abs(X))
title('Project 7.1, Part(d): \Omega_0 = 2 \pi (1500)')
xlabel('time samples')
ylabel('x_r(t)  (CTFT)')


if FINALPLOTS
    print -deps proj71PartD2.eps
end

%======================================
%Repeat (a)-(c) for omega_0 = 2pi(2000)
%======================================

Omega_0 = 2*pi*2000;
x = sin(Omega_0*t); % x(t)
x_discrete = sin(Omega_0*n*T); % x[n]


figure(3)
subplot(211)
stem(n,x_discrete)%x[n]
title('Project 7.1, Part(d) : \Omega_0 = 2 \pi (2000)')
xlim([0 49]); %first fifty samples
xlabel('time samples')
ylabel('x[t] = sin(\Omega_0nT)')

subplot(212)
plot(n,x)
xlim([0 49]); %first fifty samples
xlabel('time samples')
ylabel('x(t) = sin(\Omega_0t)')

if FINALPLOTS
    print -deps proj71PartD3.eps
end

[X,f]=ctfts(x,T);

figure(4)
plot(f,abs(X))
title('Project 7.1, Part(d): \Omega_0 = 2 \pi (2000)')
xlabel('time samples')
ylabel('x_r(t)  (CTFT)')


if FINALPLOTS
```

7.1:Part D continued on next page. . .

```
    print -deps proj71PartD4.eps
end
```

Code 7.1-4: matlab script for Part D

Output:

Again, is the magnitude of X nonzero for the expected frequenies? Is the phase of X correct?
Yes, It matched the expectations I Stated from previous(part C)

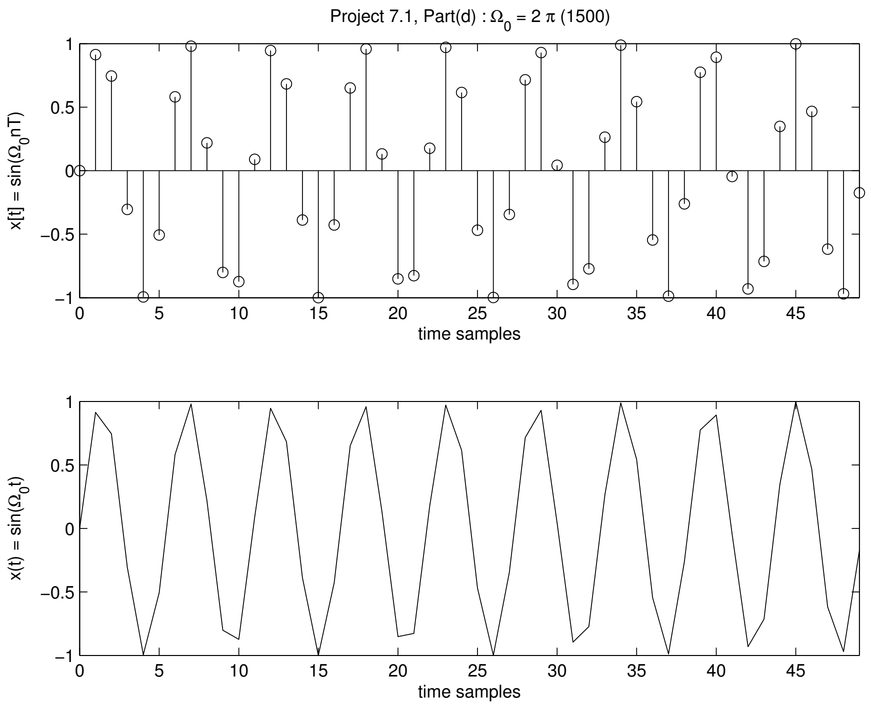

Figure 7.1-3: Output for 7.1 Part D —Part B for $\Omega_0 = 2\pi(1500)$

7.1:Part D continued on next page. . .

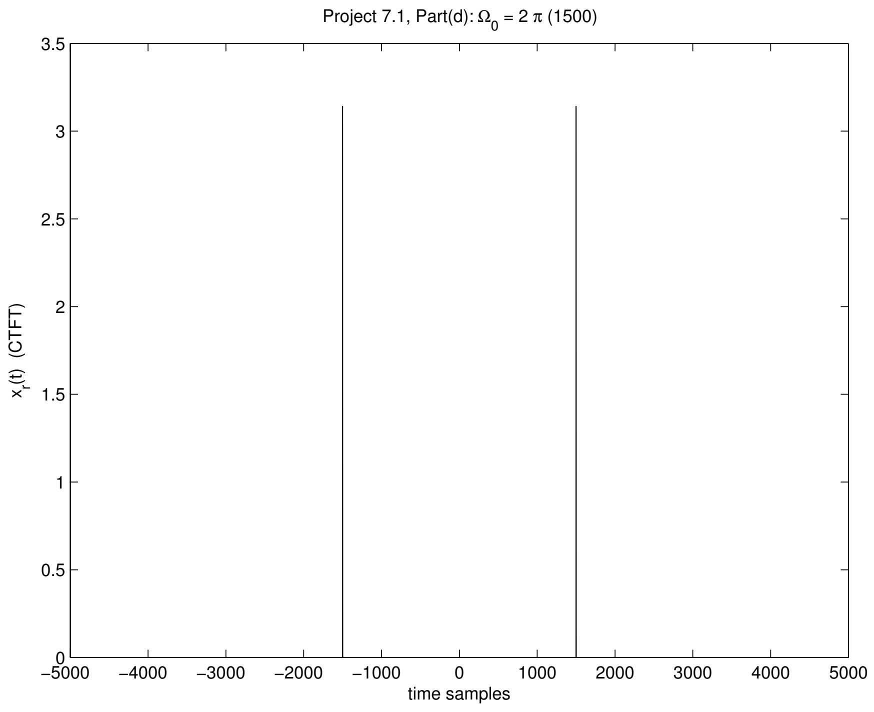

Figure 7.1-4: Output for 7.1 Part D —Part C for $\Omega_0 = 2\pi(1500)$

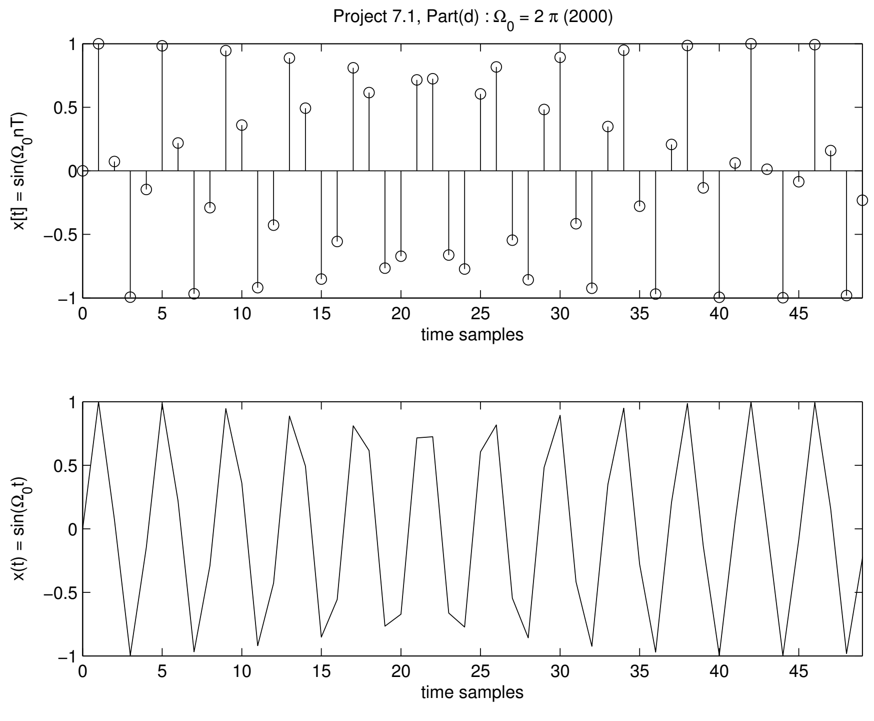

Figure 7.1-5: Output for 7.1 Part D —Part B for $\Omega_0 = 2\pi(2000)$

7.1:Part D continued on next page. . .

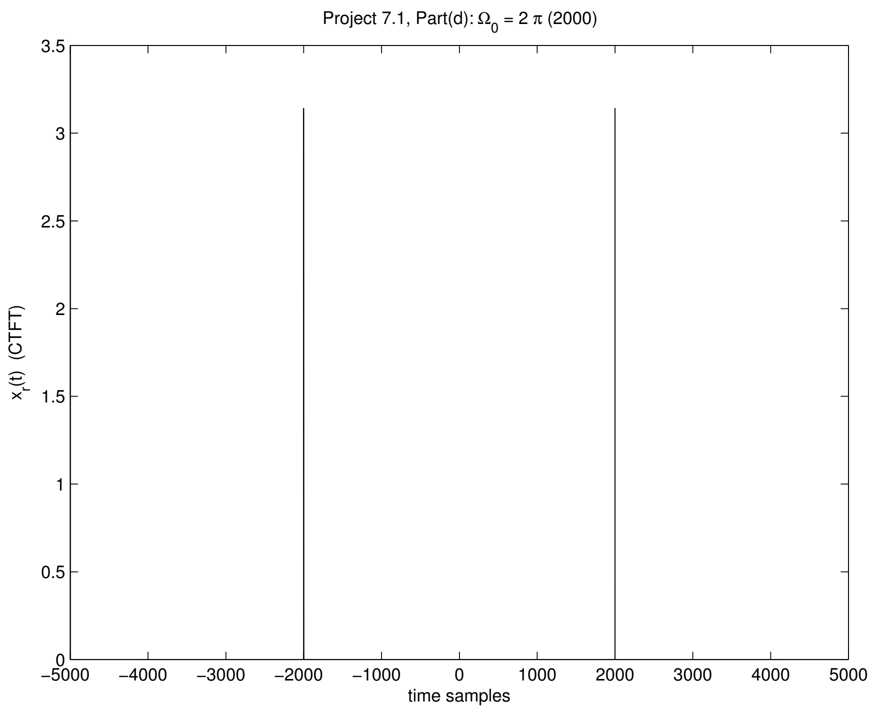

Figure 7.1-6: Output for 7.1 Part D —Part C for $\Omega_0 = 2\pi(2000)$

7.1:Part E

(e). Play each of the smapled signals created in Part(d) using sound(x,1/T). Does the pitch of the tone that you hear increase with increasing frequency Ω_0? Note that, like plot, the function sound performs interpolation. In essence, your computer converts the discrete-time signal in MATLAB into a continuous-time signal using a digital-to-analog converter, and then plays this continous-time signal on its speaker.

```
%Part 1: Project 7.1, parts (e)

Omega_0_1 = 2*pi*1500;
Omega_0_2 = 2*pi*2000;
T= 1/8192;
n=0:8191;
t=n*T;
x1 = sin(Omega_0_1*t); % x1(t)
x2 = sin(Omega_0_2*t); % x2(t)

sound(x1,1/T)
sound(x2,1/T)
```

Code 7.1-5: matlab script for Part E

Output:

Does the pitch of the tone that you hear increase with increasing frequency Ω_0? Note that, like plot, the function sound performs interpolation.
Yes, as Ω_0 increased so does the pitch. This can be observered when running the matlab script (Code 7.1-5)

7.1:Part E continued on next page...

7.1:Part F

(f). Now repeat Parts (a) and (c) - do not repeat Part (b) - for the sinusoidal frequencies $\Omega_0 = 2\pi(3500)$, $2\pi(4000)$, $2\pi(4500)$, $2\pi(5000)$, and $2\pi(5500)$ rad/sec. Also play each sample signal using sound. Does the pitch of the tone that you hear increase with each increase in the frequency Ω_0? If not, can you explain this behavior?

```matlab
%Part 1: Project 7.1, parts (f)
function Part1Code_F(freq, FINALPLOTS)
%
% This will do both Part A and C for Part F
% freq: what frequency will Omega_0 be based on 2*pi*freq
% FINALPLOTS: bool wheter to print out the graphs
%

%Part A
Omega_0 = 2*pi*freq;
T= 1/8192;
n=0:8191;
t=n*T;
x = sin(Omega_0*t); % x(t)

%Part C
[X,f]=ctfts(x,T);

figure(1)

plot(f,abs(X))
title(['Project 7.1, Part(F) | \Omega_o = 2\pi*(' num2str(freq) ')' ])
xlabel('time samples')
ylabel('x_r(t)  (CTFT)')

 output = sprintf('proj71PartF-%i.eps', freq);

if FINALPLOTS
    print('-deps', output)
end

%play the freq
sound(x,1/T)
```

Code 7.1-6: matlab script for Part F

Output:

Does the pitch of the tone that you hear increase with each increase in the frequency Ω_0? If not, can you explain this behavior?

The Pitch of the tone increase to the highest frequency at $\omega_0 = 4000$ and then it decreases in pitch. This Can be observed with the Matlab script which was created as a function(Code 7.1-6).

```
>> Part1Code_F(3500,0)
>> Part1Code_F(4000,0)
>> Part1Code_F(4500,0)
>> Part1Code_F(5000,0)
>> Part1Code_F(5500,0)
```

7.1:Part F continued on next page. . .

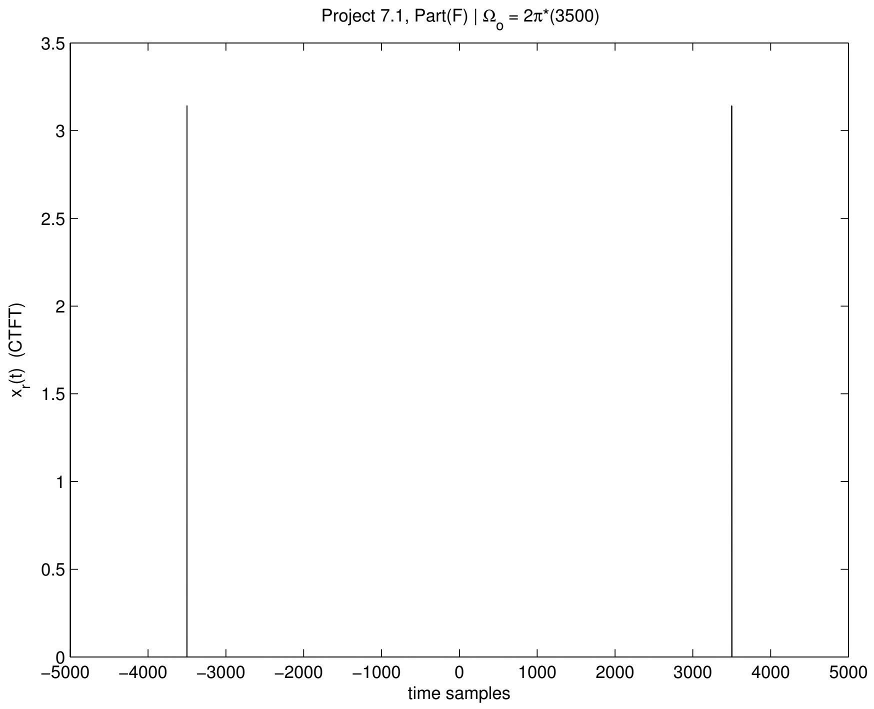

Figure 7.1-7: Output for 7.1 Part F — $\Omega_0 = 2\pi(3500)$

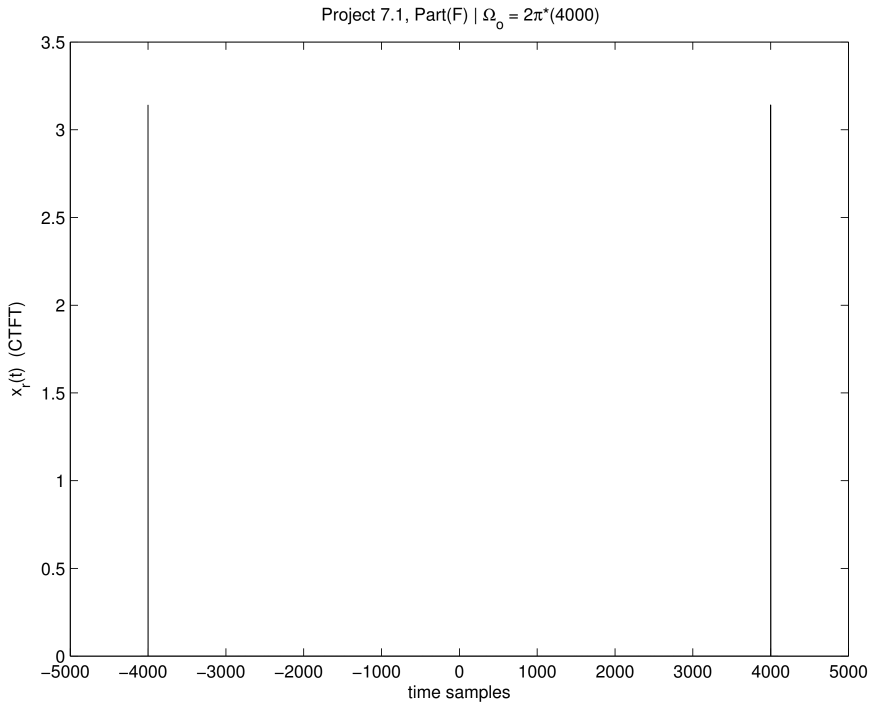

Figure 7.1-8: Output for 7.1 Part F — $\Omega_0 = 2\pi(4000)$

7.1:Part F continued on next page. . .

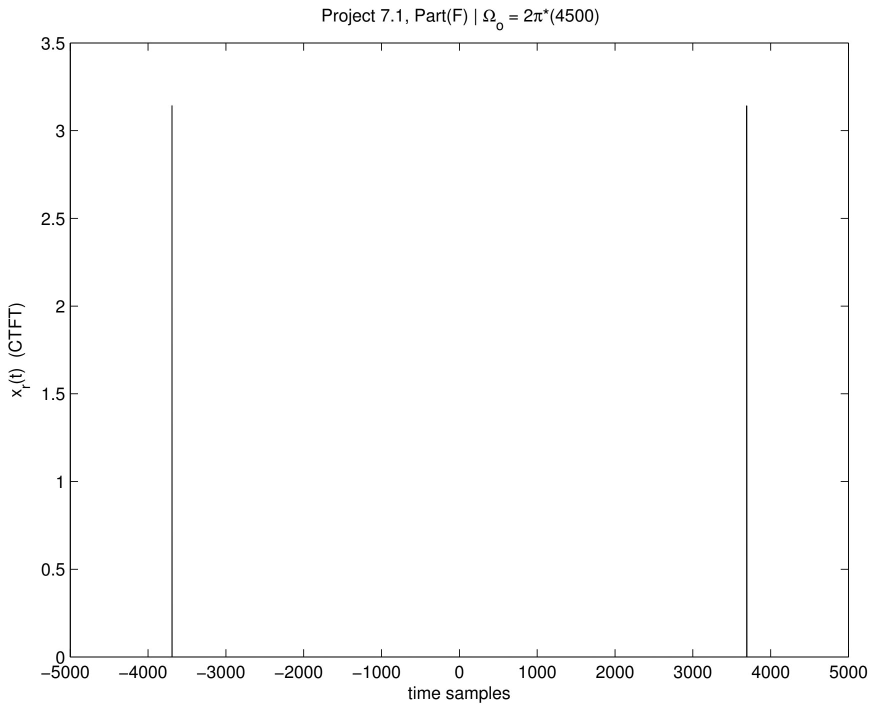

Figure 7.1-9: Output for 7.1 Part F — $\Omega_0 = 2\pi(4500)$

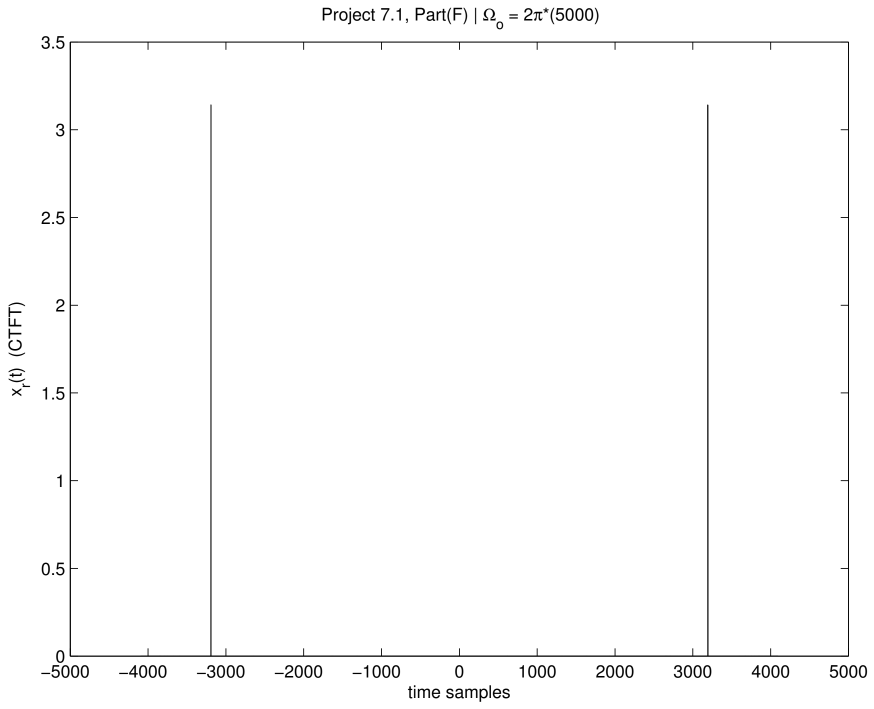

Figure 7.1-10: Output for 7.1 Part F — $\Omega_0 = 2\pi(5000)$

7.1:Part F continued on next page. . .

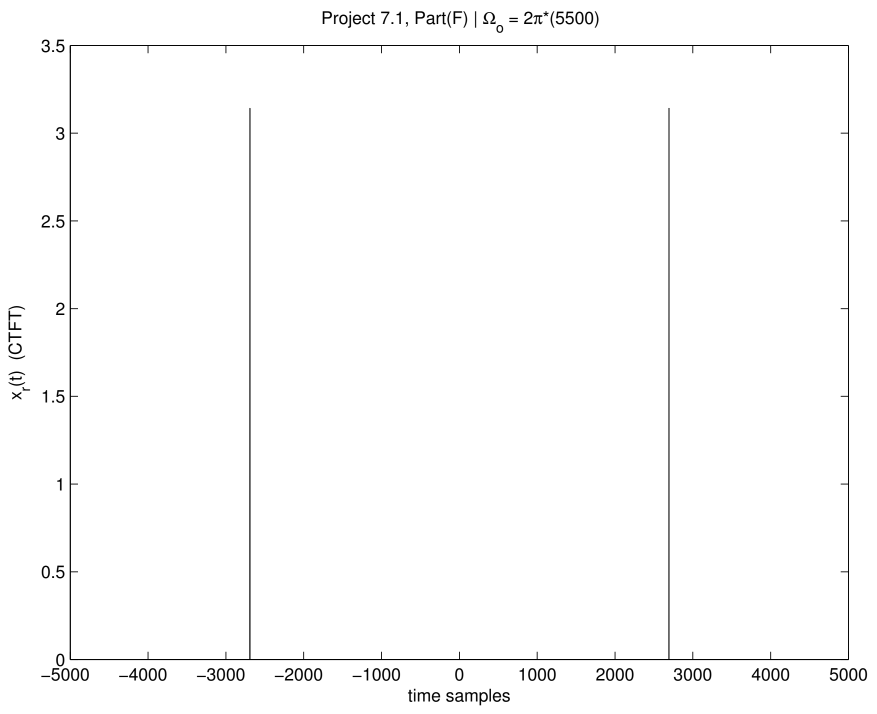

Figure 7.1-11: Output for 7.1 Part F — $\Omega_0 = 2\pi(5500)$

Advanced Problems

Now consider the signal

$$x(t) = sin(\Omega_0 t + \tfrac{1}{2}\beta t^2).$$

which is often called a chirp signal due to the sound it makes when played through a loudspeaker. The "chirp" sound is due to the increasing instantaneous frequency of the signal over time. The instatansoue frequency of a sinusoidal signal is given by the derivatives of its phase, i.e., the argument of $sin(\cdot)$. For the chirp signal, the instantaneous frequency is

$$\begin{aligned}\Omega_{inst}(t) &= \tfrac{d}{dx}\left(\Omega_0 t + \tfrac{1}{2}\beta t^2\right) . \\ &= \Omega_0 + \beta t\end{aligned}$$

Assume for the following problems that $\Omega_s = 2\pi(8192)$ rad/sec:

[Note: Use $\beta = 2\pi(2000)$ for (g)-(j), not $\beta = 2000$ as in the book.]

7.1:Part G

(g). Set $\Omega_0 = 2\pi(3000)$ rad/sec and $\beta = 2000 rad/sec^2$. Store in the vector x the samples of the chirp signal on the interval $0 \le t < 1$.

```
%Part 1: Project 7.1, parts (g)

```

7.1:Part G continued on next page...

```
Omega_0 = 2*pi*(3000);

%Based on matlab3.pdf infomation
beta = 2*pi*(2000);

%Omega_s=2*pi*(8192)
T= 1/8192;
n=0:8191;
t=n*T;

x = sin(Omega_0*t + (1/2)*beta*t.^2);
```

Code 7.1-7: matlab script for Part G

7.1:Part H

(h). Use sound to play the chirp signal contained in x. Can you explain what you just heard?

```
%Part 1: Project 7.1, parts (h)

Omega_0 = 2*pi*(5500);

%Based on matlab3.pdf infomation
beta = 2*pi*(2000);

%Omega_s=2*pi*(8192)
T= 1/8192;
n=0:8191;
t=n*T;

x = sin(Omega_0*t + (1/2)*beta*t.^2);

sound(x,1/T)
```

Code 7.1-8: matlab script for Part H

Output:

Can you explain what you just heard?
I Heard the sound of a mild high pitch which went higher then drop the pitch a little. This is what a chirp sounds like.

7.1:Part H continued on next page...

7.1:Part I

(i). Determine the approximate time sample at which the chirp signal has its maximum pitch. Given the linear equation for instantaneous frequency and your understanding of aliasing, explain how you could have predicted this time sample.

Output:

explain how you could have predicted this time sample.
Since the sample frequency $\omega_s = 2\pi * (8192)$ the sample frequency is 8192 which means T = 1/8192. In order of the highest frequency for the sinusoidal $\Omega_{inst}(t) = \frac{d}{dx}(\Omega_0 t + \frac{1}{2}\beta t^2)$.

7.1:Part I continued on next page...

7.1:Part J

(j). Store in x the samples for the first 10 seconds of the chirp signal. Play the signal using sound. Explain how you could have predicted the times at which the played signal has zero (or very low) frequency.

```
%Part 1: Project 7.1, parts (j)

Omega_0 = 2*pi*(3000);

%Based on matlab3.pdf infomation
beta = 2*pi*(2000);

%Omega_s=2*pi*(8192)
T= 1/8192;
n=0:8191*10;%increase the capture length by 10 seconds
t=n*T;

x = sin(Omega_0*t + (1/2)*beta*t.^2);

sound(x,1/T)
```

Code 7.1-8: matlab script for Part J

Output:

Explain how you could have predicted the times at which the played signal has zero (or very low) frequency.
Upon Listening to the chirp sound, I noticed that it fluctuates the pitch 2.5 times in the entire sound. This means that it alternates a pi means 2 seconds.
The patteren for the chirp went:
$[mid|start|0]->[high|1]->[mid|2]->[low|3]->[mid|4]->[high|5]->[mid|6]->[low|7]->[mid|8]->[high|9]->[mid|end|10]$
So therefore the lowest is at 3,7 seconds in the chirp

7.1:Part J continued on next page...