HW9: GPS Signal Modulation

Table of Contents

Initialize

```
close all
clearvars -except function_list pub_opt
global function_list;
function_list = {};
if ispc
    addpath('C:\Users\John\Documents\Astro\ASEN5090_GNSS\tools')
end
```

Sampling and Frequency Resolution

```
samp_rate = (20*1e6); % Hz
duration = 10*1e-3; % sec
time = 0:1/samp_rate:(duration-1/samp_rate);
num_samples = length(time);

% Nyquist frequency is half the sampling rate
Nyquist_freq = samp_rate/2;
fprintf('Nyquist Frequency: %f\n',Nyquist_freq);

% freq_resolution = samp_rate/num_samples;
freq_resolution = 1/duration;
fprintf('Frequency resolution: %f\n',freq_resolution);

dur_5Hz = 1/5;
fprintf('Required duration for fr = 5 Hz: %f\n',dur_5Hz);
```

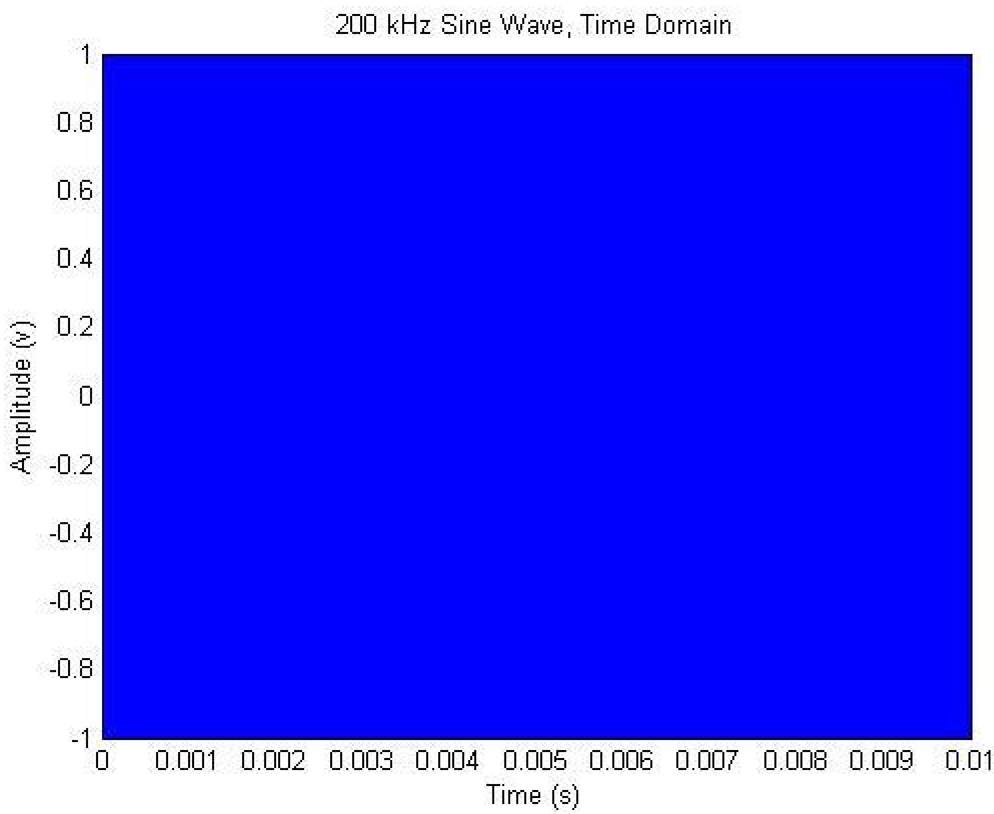
```
Nyquist Frequency: 10000000.000000
Frequency resolution: 100.000000
Required duration for fr = 5 Hz: 0.200000
```

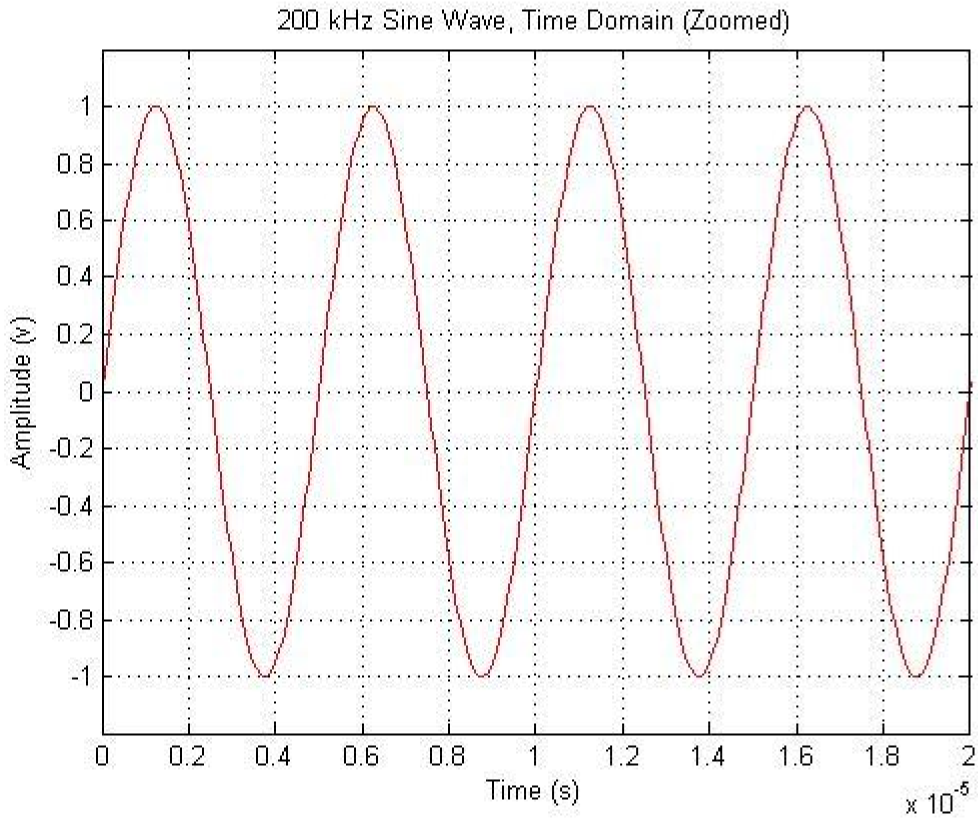
Oscilloscope and spectrum analyzer plots

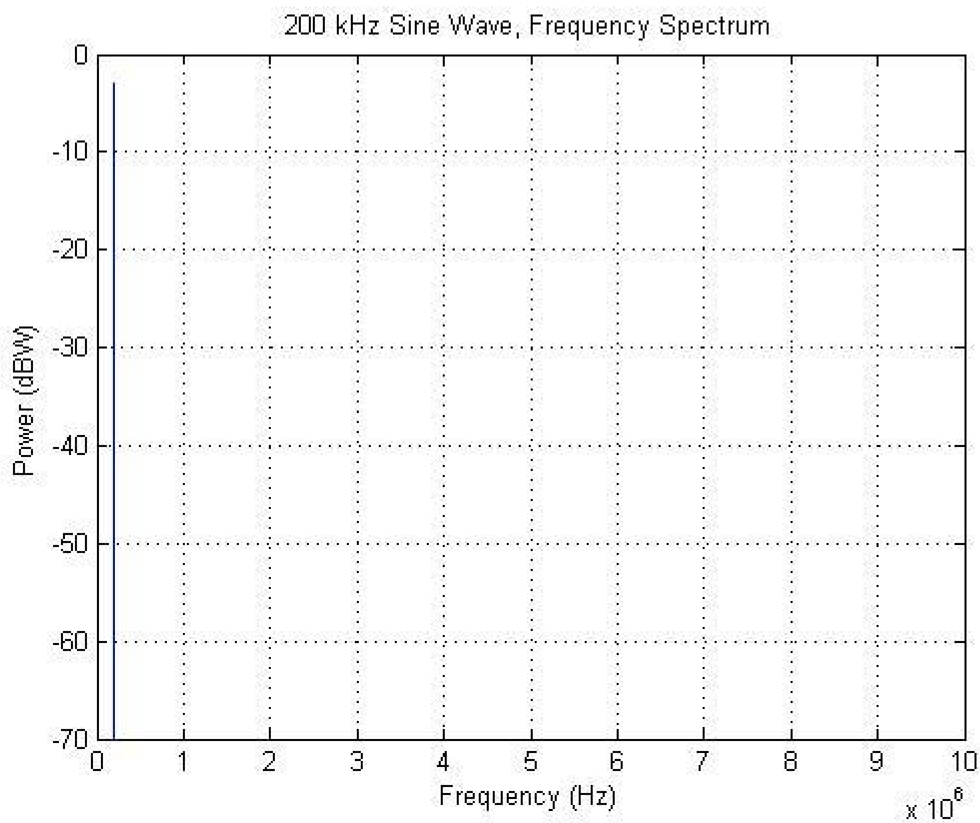
The time domain plots show a sine wave with frequency 200 kHz, amplitude = 1 volt, for 10 ms.

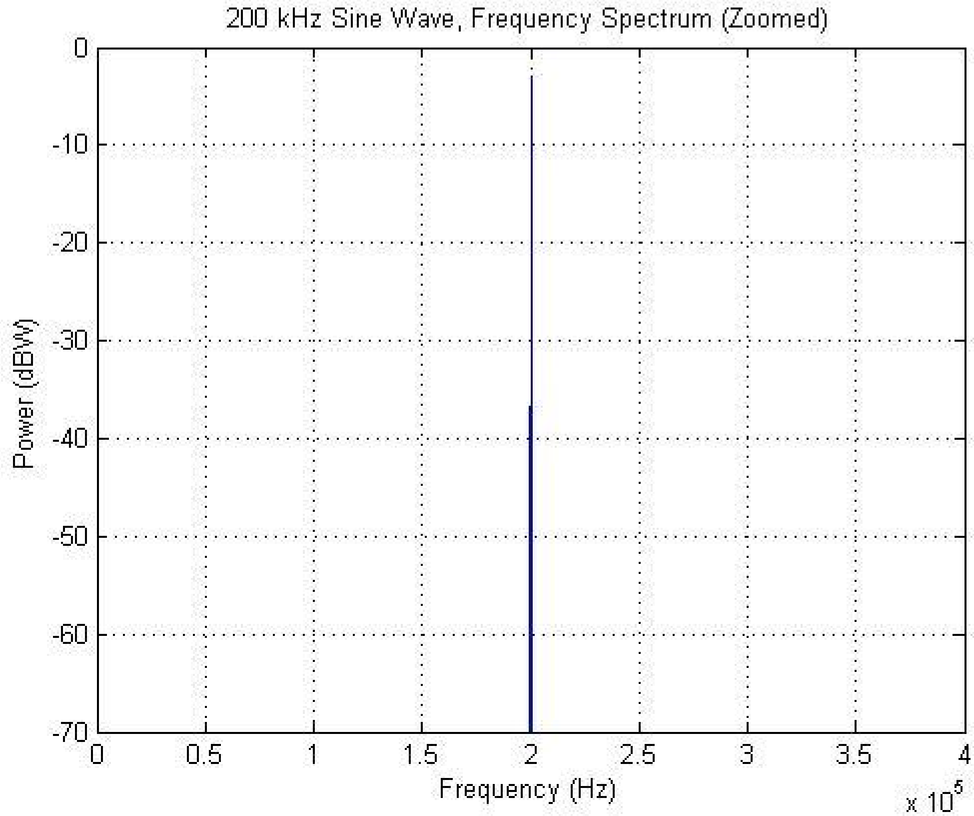
The frequency spectrum plots show a spike at 200 kHz, the frequency of the wave. The spike represents the Fourier coefficient of the wave. Since it's a perfect sine wave, there is one coefficient.

```
v = 1; % volts
signal_freq = 200*1e3; % Hz
signal = v*sin(2*pi*signal_freq*time);
plot_v_time_domain(time, signal, ...
    '200 kHz Sine Wave, Time Domain');
plot_v_time_domain(time, signal, ...
    '200 kHz Sine Wave, Time Domain (Zoomed)', ...
    [0,2e-5],[-1.2,1.2],'r');
sig_f = (fft(signal)*sqrt(2)/num_samples);
sig_f = power_dB(sig_f);
freqs = freq_resolution*[0:1:num_samples-1];
spectrum_analyzer(freqs, sig_f, ...
    '200 kHz Sine Wave, Frequency Spectrum', ...
    [0,Nyquist_freq], [-70,0]);
spectrum_analyzer(freqs, sig_f, ...
    '200 kHz Sine Wave, Frequency Spectrum (Zoomed)', ...
    [0,4e5], [-70,0]);
```




200 kHz Sine Wave, Time Domain (Zoomed)
Amplitude (v)
Time (s)
x 10^-5
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


200 kHz Sine Wave, Frequency Spectrum
Power (dBW)
Frequency (Hz)
x 10^6
0
-10
-20
-30
-40
-50
-60
-70
0
1
2
3
4
5
6
7
8
9
10



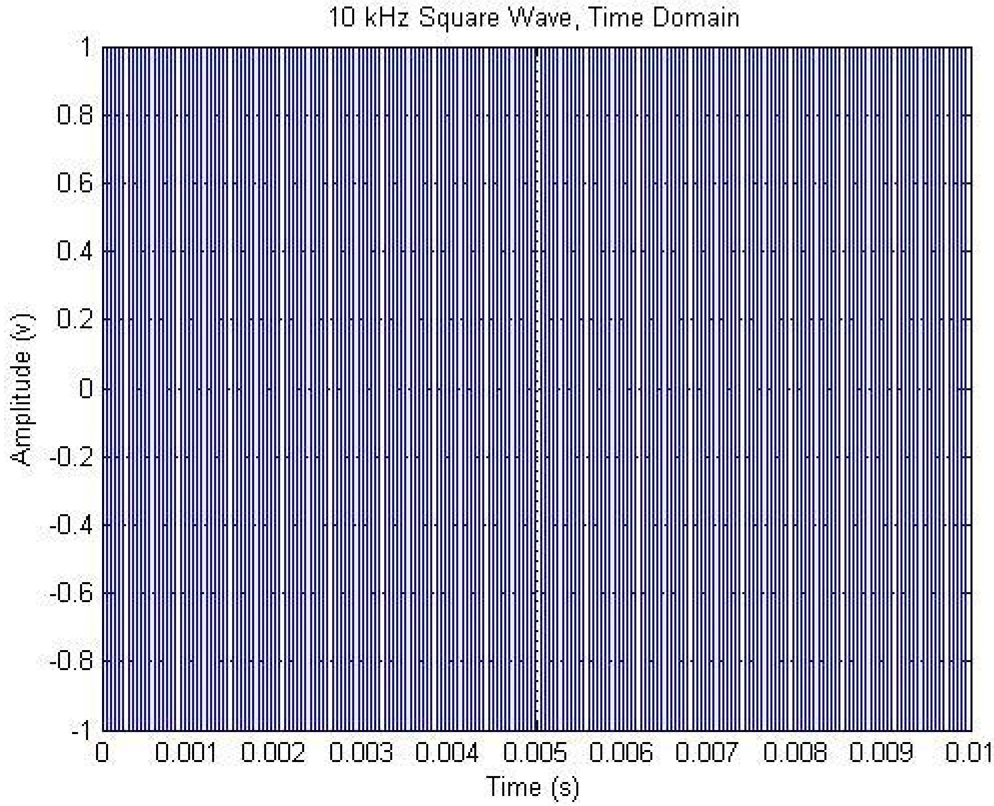
Square wave

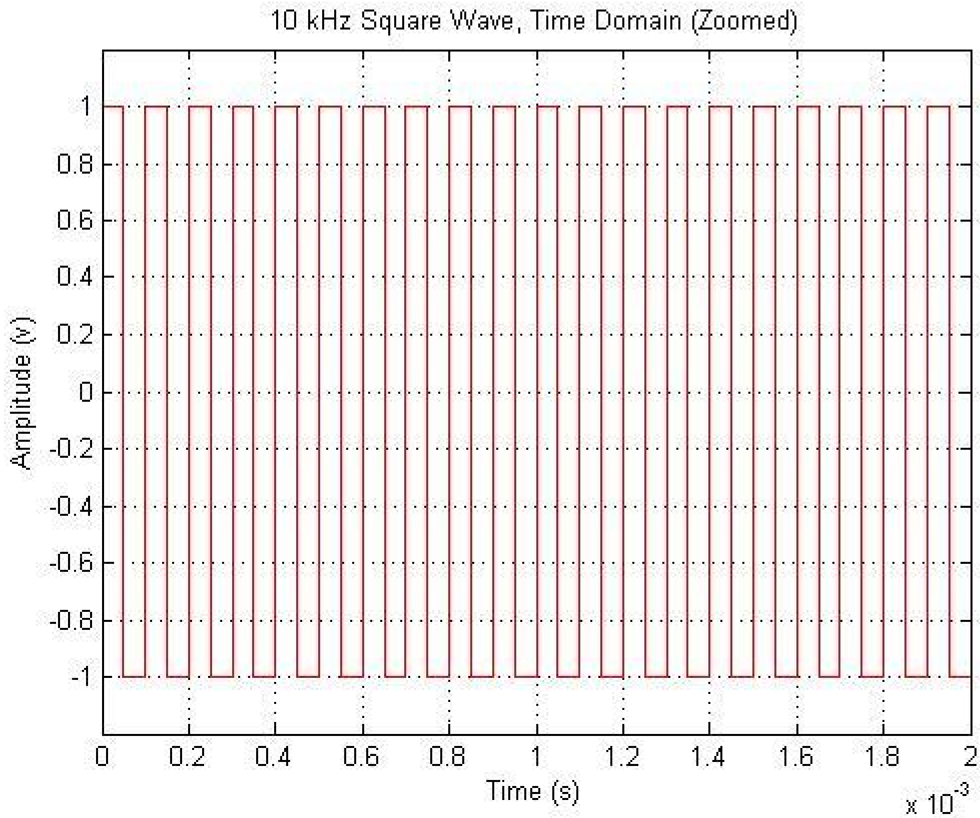
The time domain plots show a square wave with frequency 10 kHz, amplitude = 1 volt, for 10 ms.

The frequency spectrum plots show a spike at 10 kHz, the frequency of the wave. Side lobes follow in 20 kHz increments, each with less power. This represents Fourier coefficients. There are many (infinite) because it takes many sine waves to approximate a square wave.

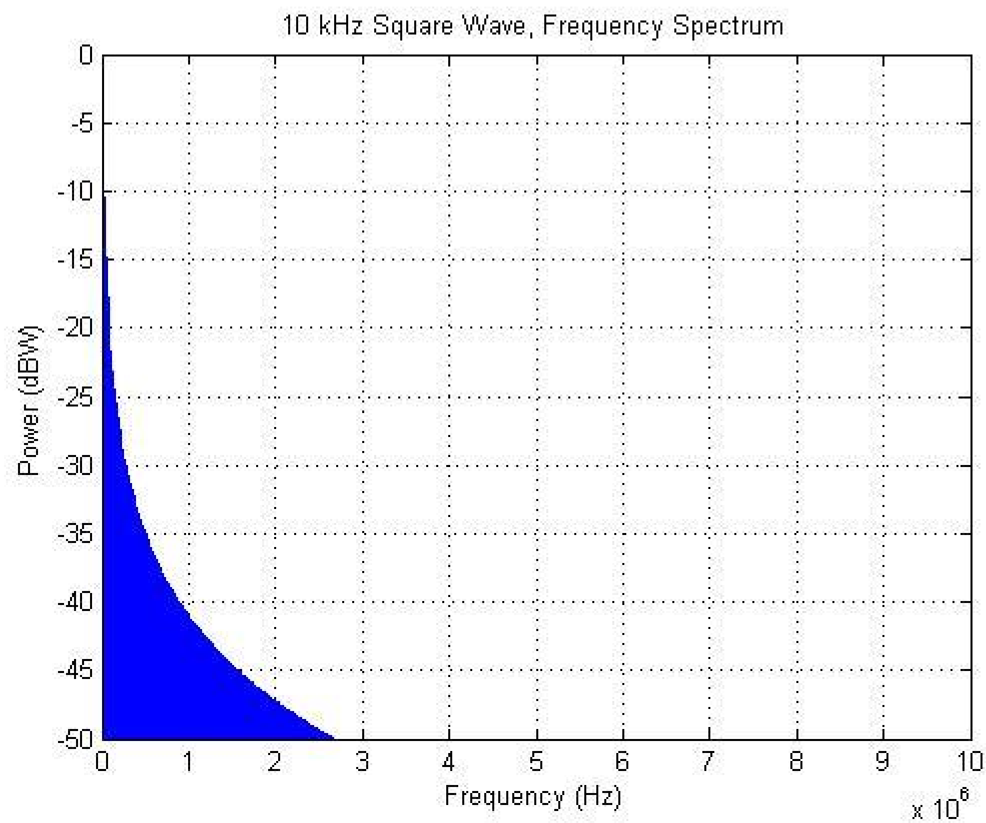
```
v = 1; % volts
signal_freq = 10*1e3; % Hz
signal = v*sin(2*pi*signal_freq*time);
signal(signal >= 0) = 1;
signal(signal < 0) = -1;

plot_v_time_domain(time, signal, ...
    '10 kHz Square Wave, Time Domain');
plot_v_time_domain(time, signal, ...
    '10 kHz Square Wave, Time Domain (Zoomed)', ...
    [0,2e-3],[-1.2,1.2],'r');

sig_f = (fft(signal)*sqrt(2)/num_samples);
sig_f = power_dB(sig_f);
freqs = freq_resolution*[0:1:num_samples-1];

spectrum_analyzer(freqs, sig_f, ...
    '10 kHz Square Wave, Frequency Spectrum', ...
```

```
    [0,Nyquist_freq], [-50,0]);
spectrum_analyzer(freqs, sig_f, ...
    '10 kHz Square Wave, Frequency Spectrum (Zoomed)', ...
    [0,1e5], [-50,0]);
```

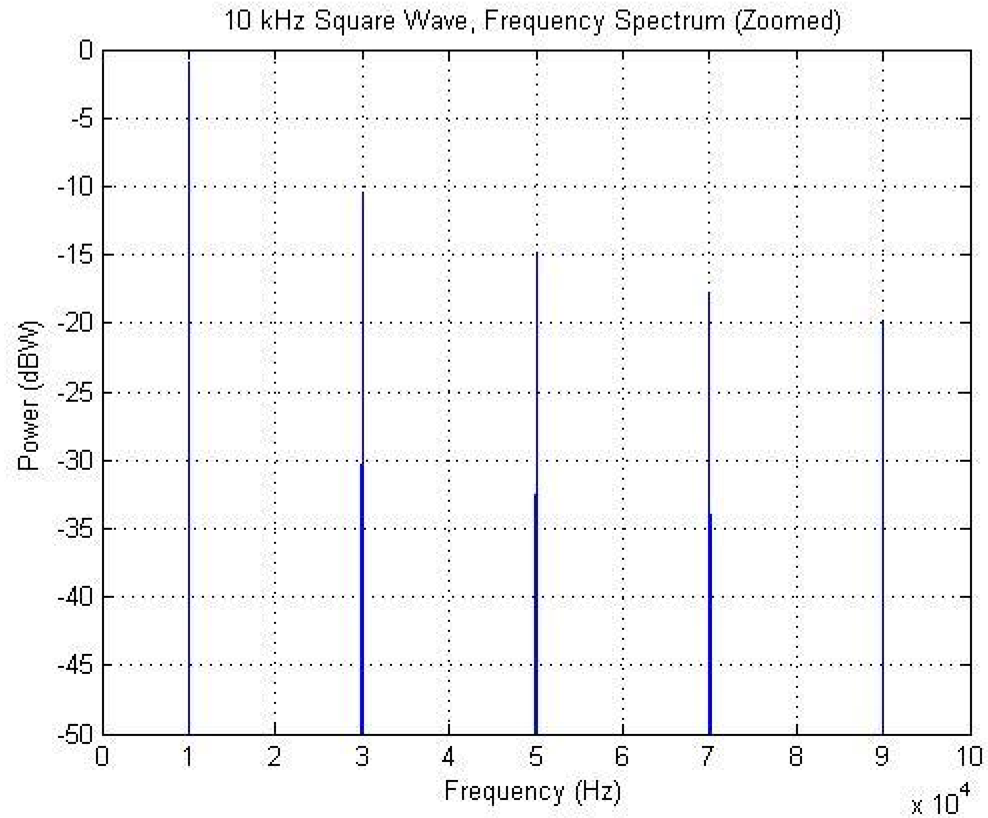




10 kHz Square Wave, Time Domain (Zoomed)
Amplitude (v)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Time (s)
x 10^-3



10 kHz Square Wave, Frequency Spectrum
Power (dBW)
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
0
1
2
3
4
5
6
7
8
9
10
Frequency (Hz)
x 10^6



PRN Codes - Maximal Length

The time domain plots show a square wave with frequency 1.023e6 Hz. Zoom plot shows the pseudo-randomness of the signal. It repeats every 1023 chips.

The frequency spectrum plots show a main lobe about zero with a width of 2.046e6 Hz. Side lobes are smaller with a width of 1.023e6 Hz. Zoom plot shows the code repeats at 1 kHz due to the repeat frequency (fc/num_chips).

The maximal code is fairly smooth at the main lobe.

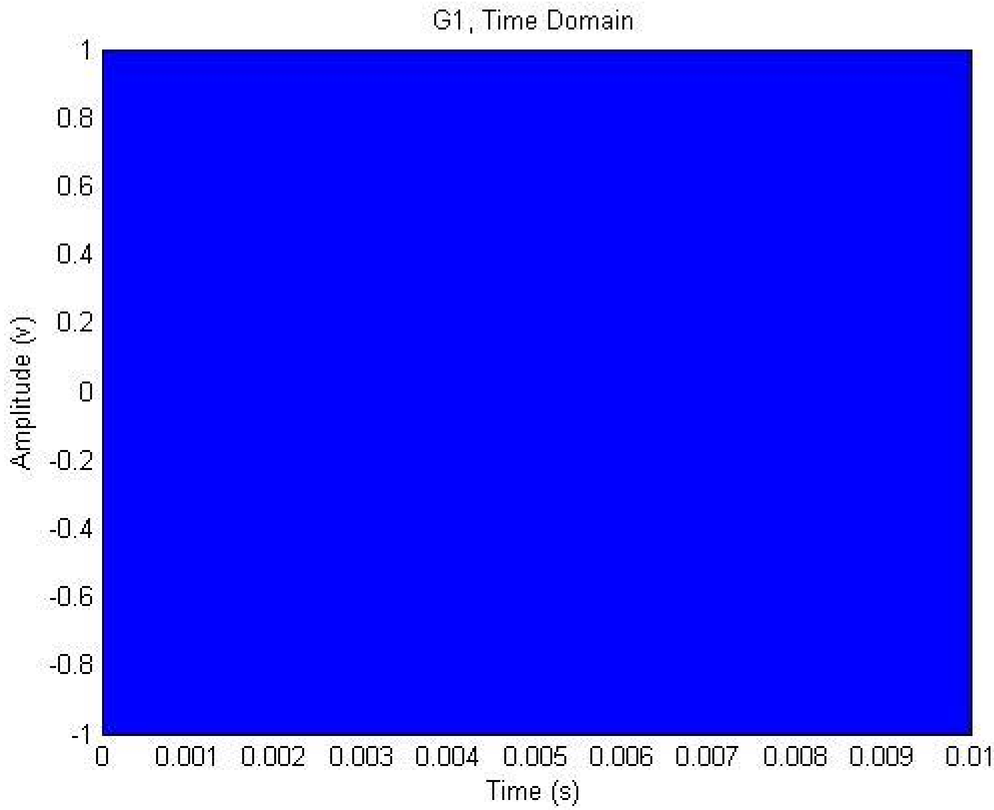
Example code bits are shown below.

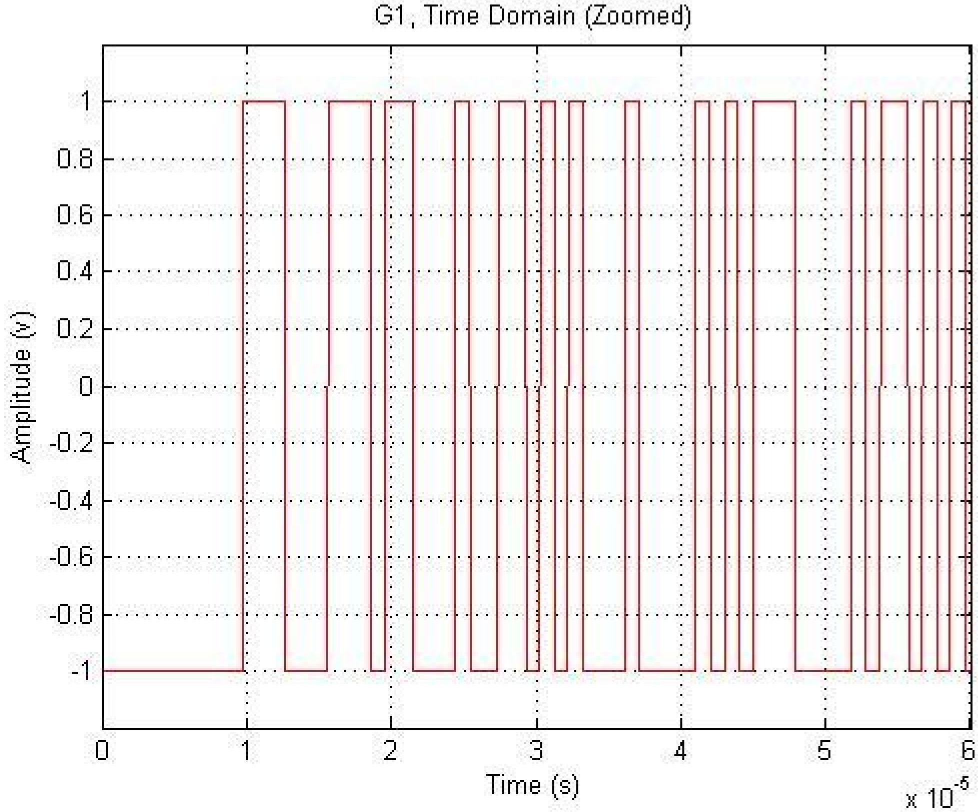
```
G1 = BitShiftRegister(10, [3,10]);
chip_rate = 1.023e6; % MHz
num_chips = chip_rate*duration;
G1_samples = zeros(num_samples,1);
G1_update_time = 0;
for ii = 1:num_samples
    if time(ii) >= G1_update_time - 1/chip_rate/100
        sample = G1.update();
        if sample == 1
            G1_samples(ii) = -1;
        else
            G1_samples(ii) = 1;
        end
        G1_update_time = G1_update_time + 1/chip_rate;
```

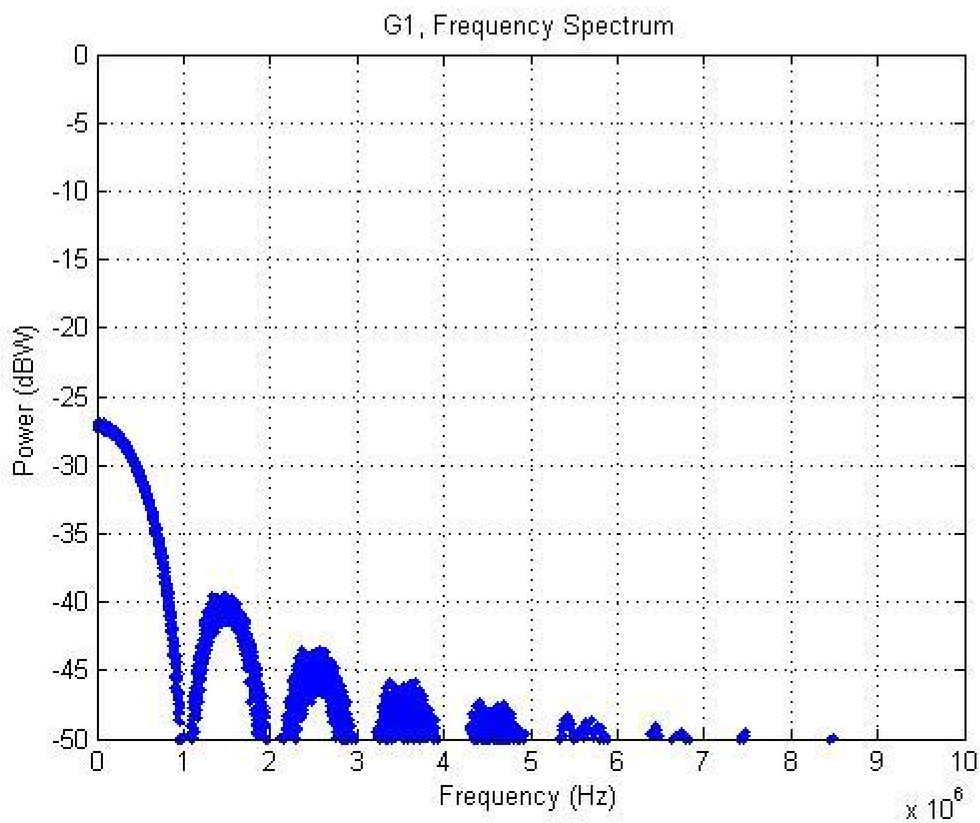
```
    else
        G1_samples(ii) = G1_samples(ii-1);
    end
end
time(92*samp_rate/1e7:102*samp_rate/1e7)'
G1_samples(92*samp_rate/1e7:102*samp_rate/1e7)
plot_v_time_domain(time, G1_samples, ...
    'G1, Time Domain');
plot_v_time_domain(time, G1_samples, ...
    'G1, Time Domain (Zoomed)', ...
    [0,6e-5],[-1.2,1.2],'r');

freqs = freq_resolution*[0:1:num_samples-1];
G1_samples_freq = fft(G1_samples)*sqrt(2)/num_samples;
Ps = 20*log10(abs((G1_samples_freq)));
spectrum_analyzer(freqs, Ps, ...
    'G1, Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
spectrum_analyzer(freqs, Ps, ...
    'G1, Frequency Spectrum (Zoomed)', ...
    [0,10000], [-50,0],'.');
```

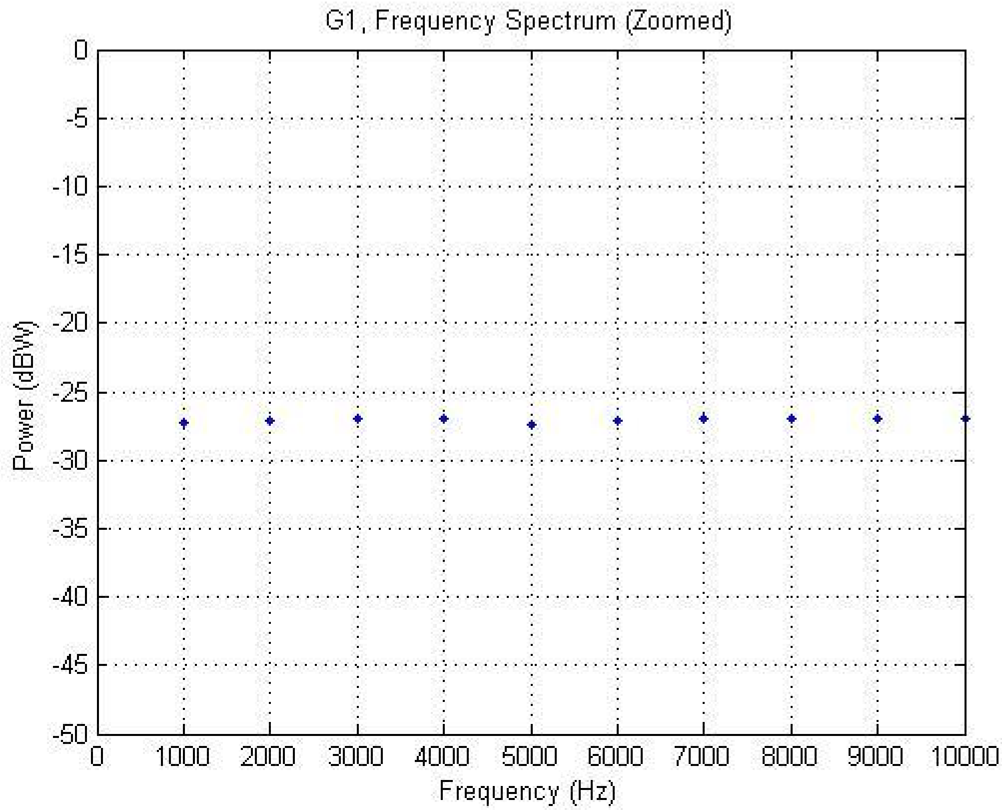
```
ans =

   1.0e-04 *

    0.0915
    0.0920
    0.0925
    0.0930
    0.0935
    0.0940
    0.0945
    0.0950
    0.0955
    0.0960
    0.0965
    0.0970
    0.0975
    0.0980
    0.0985
    0.0990
    0.0995
    0.1000
    0.1005
    0.1010
    0.1015


ans =

    -1
    -1
```

```
-1

-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
 1
 1
 1
 1
 1
 1
 1
 1
```




G1, Time Domain (Zoomed)
Amplitude (v)
Time (s)
x 10^-5


G1, Frequency Spectrum
Power (dBW)
Frequency (Hz)
x 10^6



PRN Codes - Gold Code

The time domain plots show a square wave with frequency 1.023e6 Hz. Zoom plot shows the pseudo-randomness of the signal. It repeats every 1023 chips.

The frequency spectrum plots show a main lobe about zero with a width of 2.046e6 Hz. Side lobes are smaller with a width of 1.023e6 Hz. Zoom plot shows the code repeats at 1 kHz due to the repeat frequency (fc/num_chips). The spectral lines are where the Fourier coefficients would be located.

This spectrum is noisier than the max-length code.

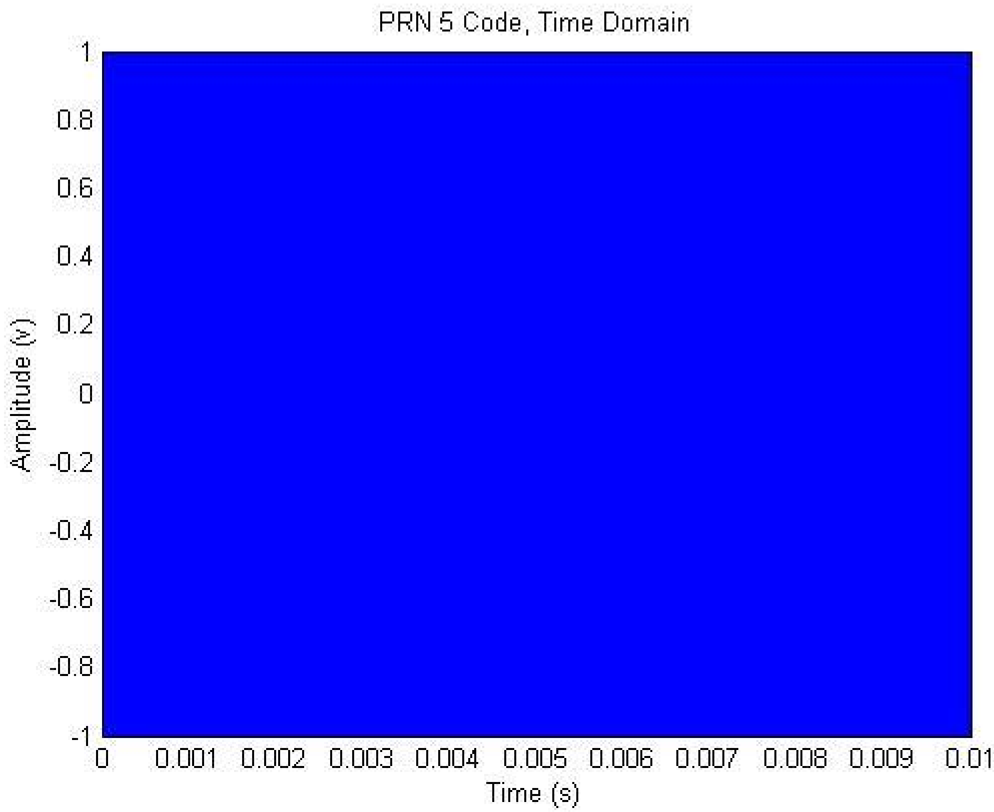
Example code bits are shown below.

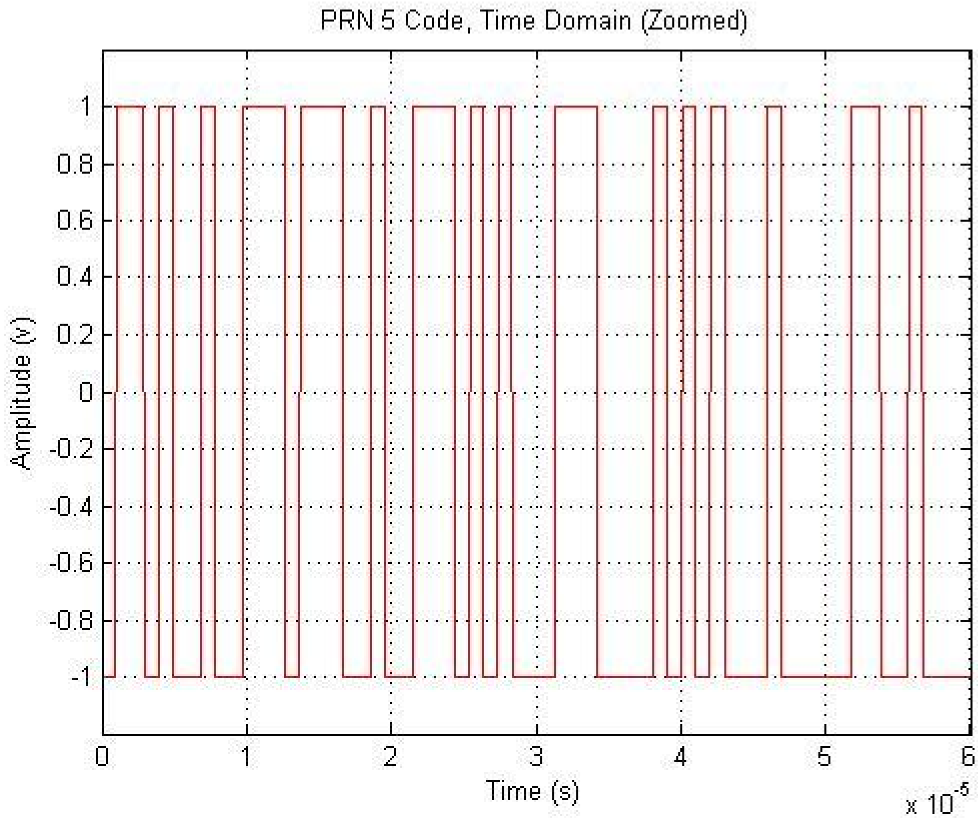
```
prn5=PRNCode(5);
for i = 1:1023
prn5.update()
end
prn5_samples = zeros(num_samples,1);
prn5_update_time = 0;
code_idx = 1;
for ii = 1:num_samples
    if time(ii) >= prn5_update_time - 1/chip_rate/100
        sample = prn5.CA_code(code_idx);
        if sample == 1
            prn5_samples(ii) = -1;
        else
            prn5_samples(ii) = 1;
```

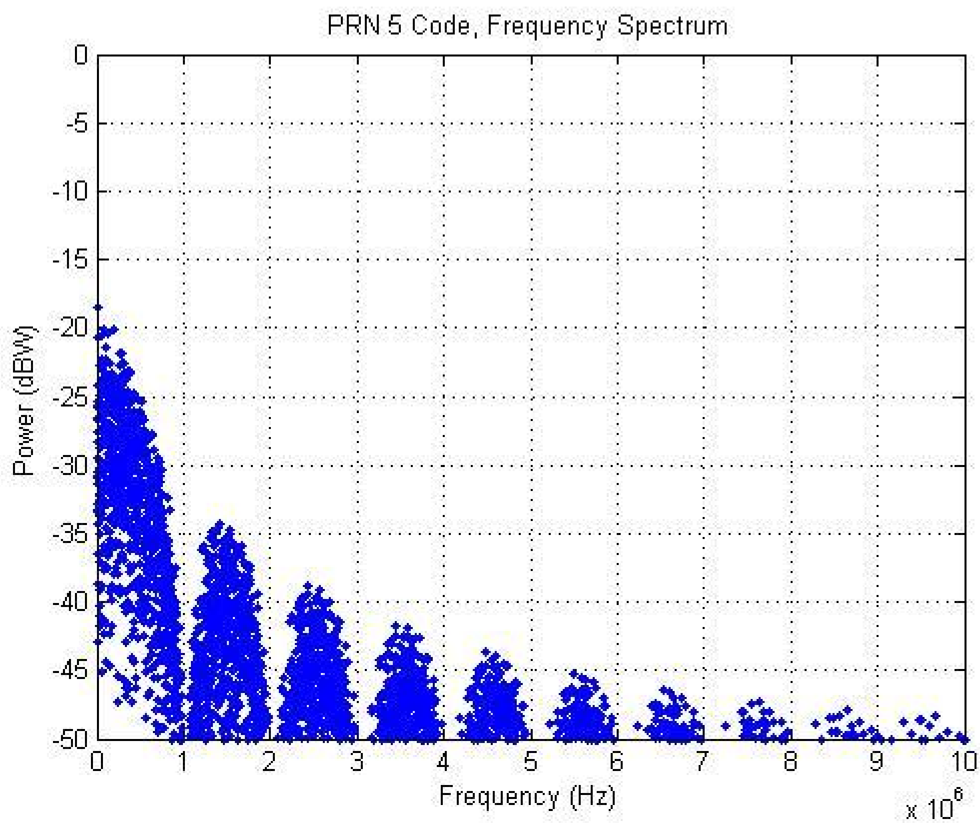
```
        end
        prn5_update_time = prn5_update_time + 1/chip_rate;
        code_idx = code_idx+1;
        if code_idx > 1023
            code_idx = 1;
        end
    else
        prn5_samples(ii) = prn5_samples(ii-1);
    end
end
time(92*samp_rate/1e7:102*samp_rate/1e7)'
prn5_samples(92*samp_rate/1e7:102*samp_rate/1e7)
plot_v_time_domain(time, prn5_samples, ...
    'PRN 5 Code, Time Domain');
plot_v_time_domain(time, prn5_samples, ...
    'PRN 5 Code, Time Domain (Zoomed)', ...
    [0,6e-5],[-1.2,1.2],'r');

freqs = freq_resolution*[0:1:num_samples-1];
Ps = 20*log10(abs(fft(prn5_samples)*sqrt(2)/num_samples));
spectrum_analyzer(freqs, Ps, ...
    'PRN 5 Code, Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
spectrum_analyzer(freqs, Ps, ...
    'PRN 5 Code, Frequency Spectrum (Zoomed)', ...
    [0,10000], [-50,0],'.');
```

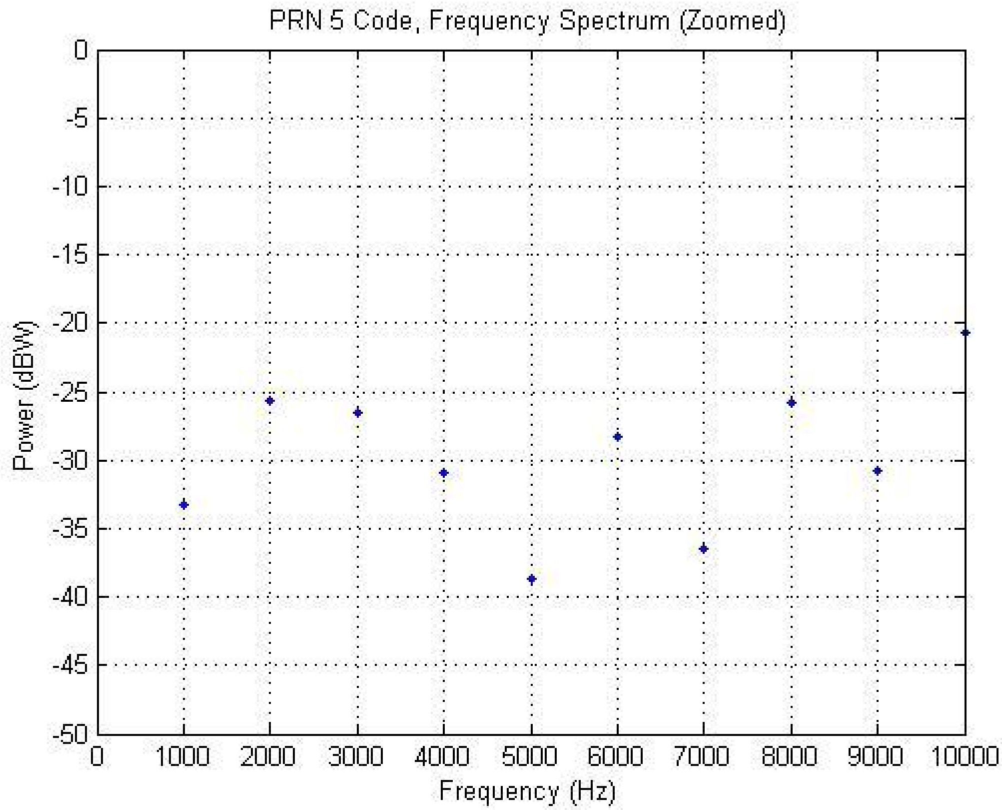
```
ans =

   1.0e-04 *

    0.0915
    0.0920
    0.0925
    0.0930
    0.0935
    0.0940
    0.0945
    0.0950
    0.0955
    0.0960
    0.0965
    0.0970
    0.0975
    0.0980
    0.0985
    0.0990
    0.0995
    0.1000
    0.1005
    0.1010
    0.1015
```

```
ans =

    -1
    -1
    -1
    -1
    -1
    -1
    -1
    -1
    -1
    -1
    -1
    -1
    -1
     1
     1
     1
     1
     1
     1
     1
     1
```




PRN 5 Code, Time Domain (Zoomed)
Amplitude (v)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
Time (s)
x 10^-5


PRN 5 Code, Frequency Spectrum
Power (dBW)
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
0
1
2
3
4
5
6
7
8
9
10
Frequency (Hz)
x 10^6



Direct Spread Spectrum Modulation

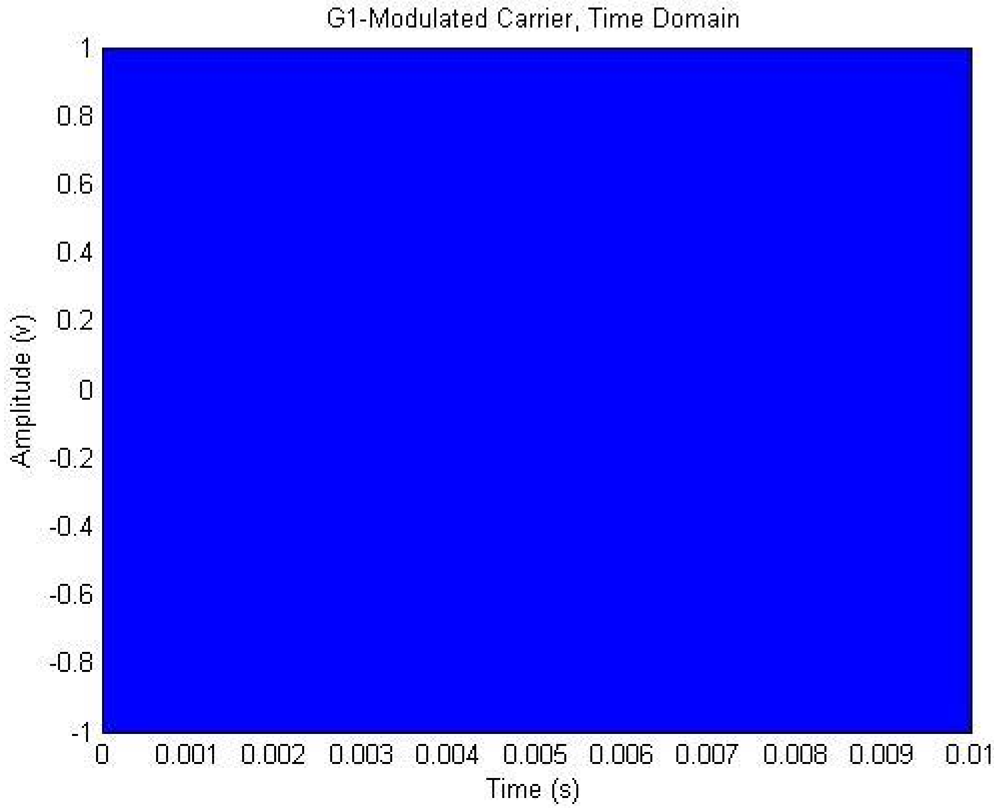
The time domain plots show a sampled sign wave that is biphase-shift keyed. That means the phase moves 180 degrees depending on the chip value of 1 or -1.

The frequency spectrum plots show a main lobe about the carrier frequency (5*1.023e6 Hz). It has a width of 2.046e6 Hz. Side lobes are smaller with a width of 1.023e6 Hz. Zoom plot shows the code repeats at 1 kHz due to the repeat frequency (fc/num_chips).

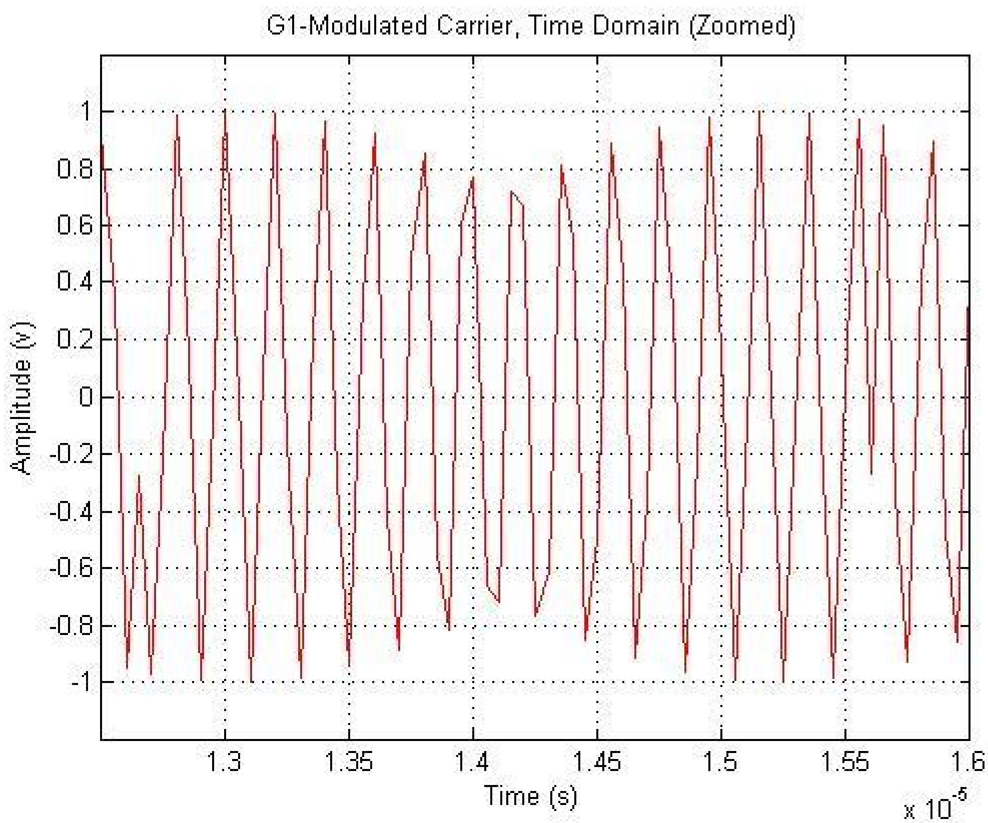
Changing the carrier frequency moves the center of the main lobe. Changing the chip rate affects the null-value given by the sinc function (the troughs of the spectrum). It also changes the space between the spectral lines.

```
carrier_freq = 5*chip_rate;
% carrier_freq = 1575.42e6;
carrier = cos(2*pi*carrier_freq*time);
mod_carrier = G1_samples.*carrier';
plot_v_time_domain(time, mod_carrier, ...
    'G1-Modulated Carrier, Time Domain');
plot_v_time_domain(time, mod_carrier, ...
    'G1-Modulated Carrier, Time Domain (Zoomed)', ...
    [1.25e-5,1.6e-5],[-1.2,1.2],'r');
% figure
% plot(time, mod_carrier)
Ps = 20*log10(abs(fft(mod_carrier)*sqrt(2)/num_samples));
spectrum_analyzer(freqs, Ps, ...
```

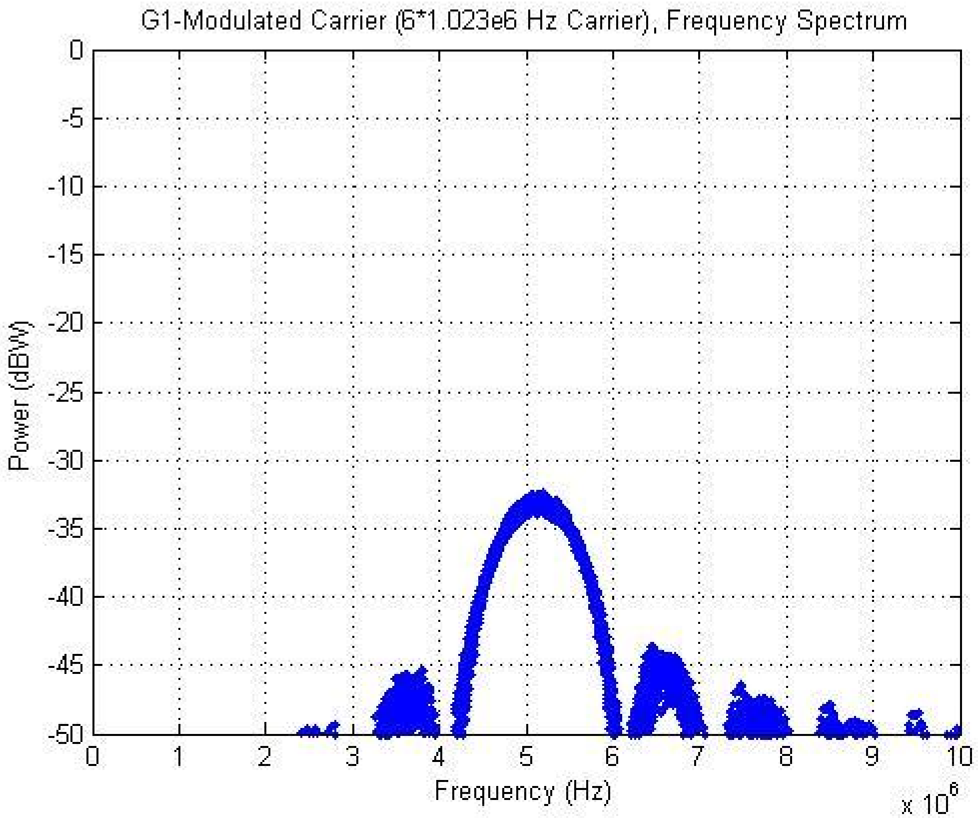
```
    'G1-Modulated Carrier (6*1.023e6 Hz Carrier), Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
spectrum_analyzer(freqs, Ps, ...
    'G1-Modulated Carrier (6*1.023e6 Hz Carrier), Frequency Spectrum (Zoomed)', ..
    [5.11e6,5.12e6], [-55,0],'.');

% Change Carrier freq
Ps = 20*log10(abs(fft(G1_samples.*cos(2*pi*6*chip_rate*time)')...
    *sqrt(2)/num_samples));
spectrum_analyzer(freqs, Ps, ...
    'G1-Modulated Carrier (2*1.023e6 Hz chip rate), Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
spectrum_analyzer(freqs, Ps, ...
    'G1-Modulated Carrier (2*1.023e6 Hz chip rate), Frequency Spectrum (Zoomed)',
    [5.11e6,5.12e6]+1.023e6, [-55,0],'.');

% Change chip rate
G1_fast = BitShiftRegister(10, [3,10]);
new_chip_rate = 2*1.023e6; % MHz
num_chips = new_chip_rate*duration;
G1_fast_samples = zeros(num_samples,1);
G1_fast_update_time = 0;
for ii = 1:num_samples
    if time(ii) >= G1_fast_update_time - 1/new_chip_rate/100
        sample = G1_fast.update();
        if sample == 1
            G1_fast_samples(ii) = -1;
        else
            G1_fast_samples(ii) = 1;
        end
        G1_fast_update_time = G1_fast_update_time + 1/new_chip_rate;
    else
        G1_fast_samples(ii) = G1_fast_samples(ii-1);
    end
end
Ps = 20*log10(abs(fft(G1_fast_samples.*carrier')...
    *sqrt(2)/num_samples));
spectrum_analyzer(freqs, Ps, ...
    'G1-Modulated Carrier, Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
spectrum_analyzer(freqs, Ps, ...
    'G1-Modulated Carrier, Frequency Spectrum (Zoomed)', ...
    [5.11e6,5.12e6], [-55,0],'.');
```

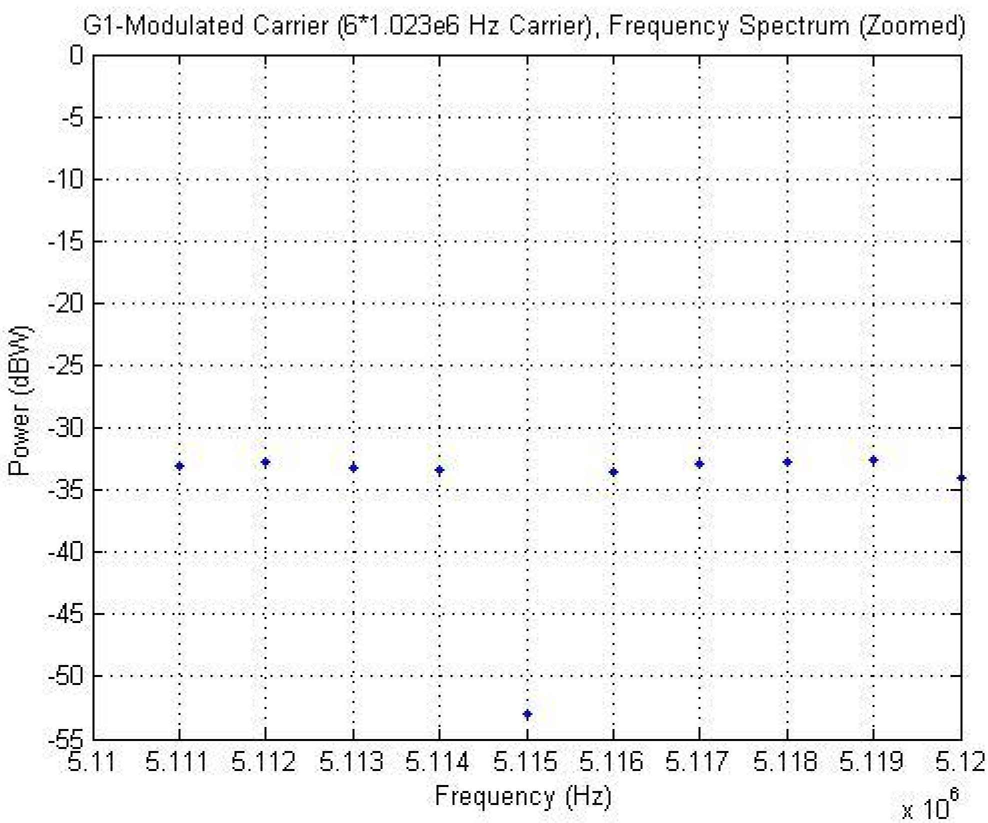


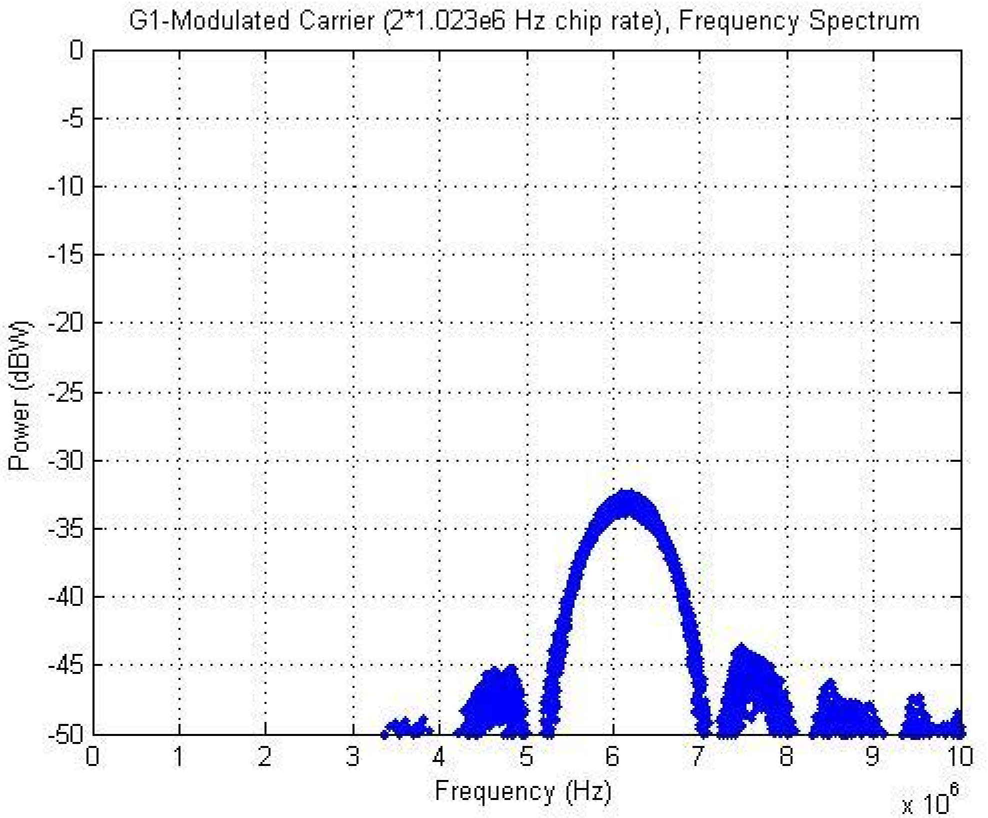
G1-Modulated Carrier, Time Domain
Amplitude (v)
Time (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01

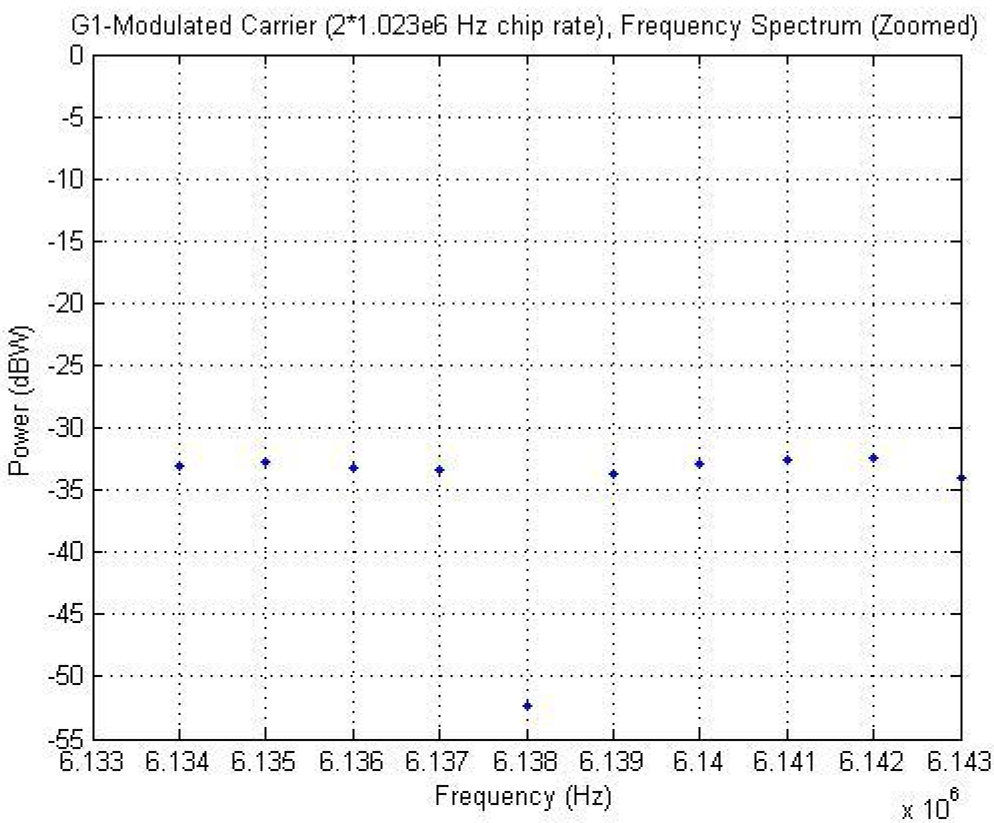


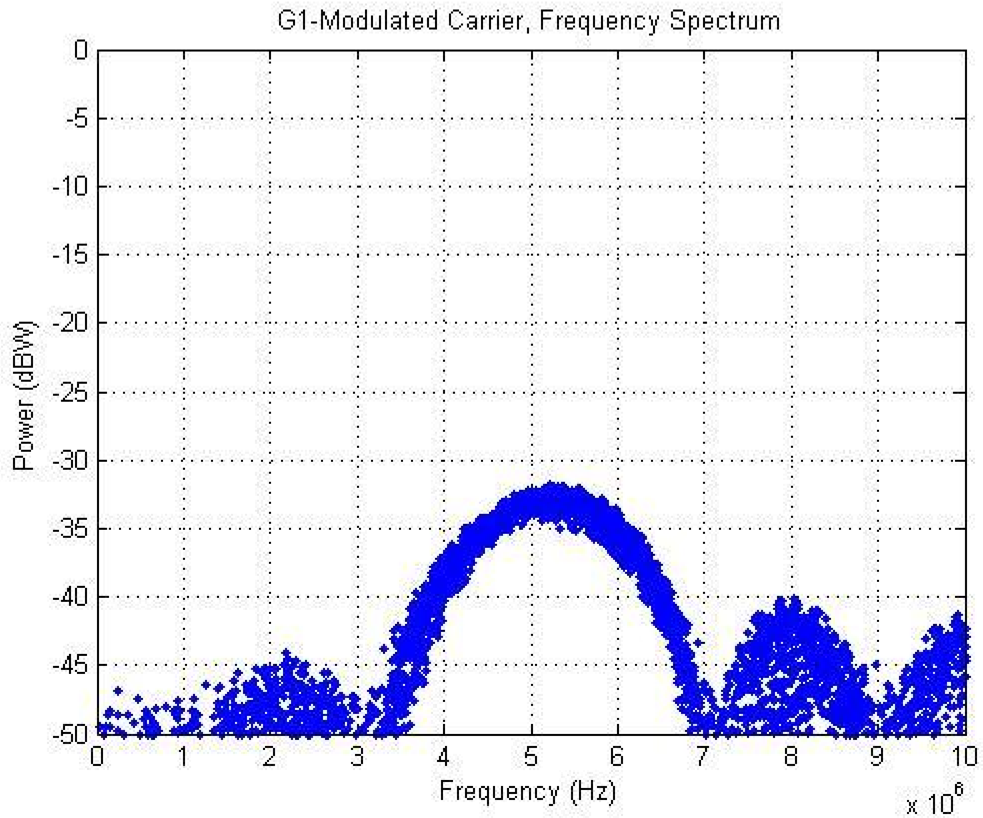
G1-Modulated Carrier, Time Domain (Zoomed)
Amplitude (v)
Time (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1.3
1.35
1.4
1.45
1.5
1.55
1.6
x 10-5

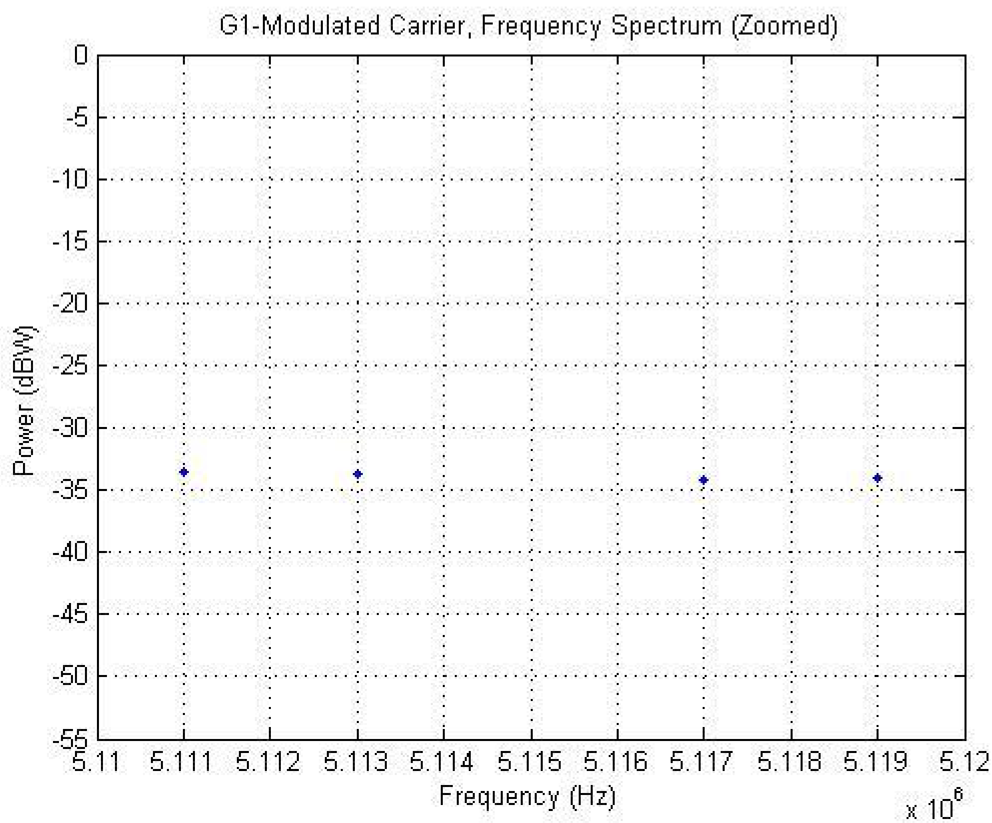

G1-Modulated Carrier (6*1.023e6 Hz Carrier), Frequency Spectrum
Power (dBW)
Frequency (Hz)
x 10^6
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
0
1
2
3
4
5
6
7
8
9
10


G1-Modulated Carrier (6*1.023e6 Hz Carrier), Frequency Spectrum (Zoomed)
Power (dBW)
Frequency (Hz)
x 10^6
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
5.11
5.111
5.112
5.113
5.114
5.115
5.116
5.117
5.118
5.119
5.12


G1-Modulated Carrier (2*1.023e6 Hz chip rate), Frequency Spectrum
Power (dBW)
Frequency (Hz)
x 10^6


G1-Modulated Carrier (2*1.023e6 Hz chip rate), Frequency Spectrum (Zoomed)
Power (dBW)
Frequency (Hz)
x 10^6


G1-Modulated Carrier, Frequency Spectrum
Power (dBW)
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
0 1 2 3 4 5 6 7 8 9 10
Frequency (Hz)
x 10^6

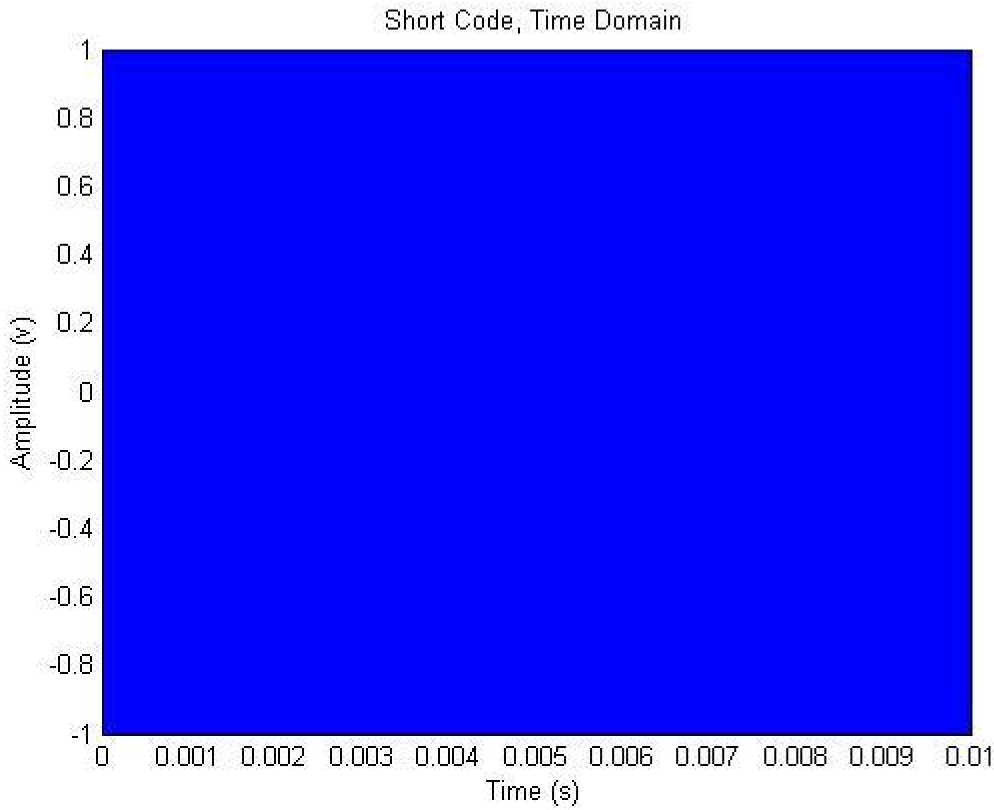

G1-Modulated Carrier, Frequency Spectrum (Zoomed)
Power (dBW)
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
5.11 5.111 5.112 5.113 5.114 5.115 5.116 5.117 5.118 5.119 5.12
Frequency (Hz)
x 10^6

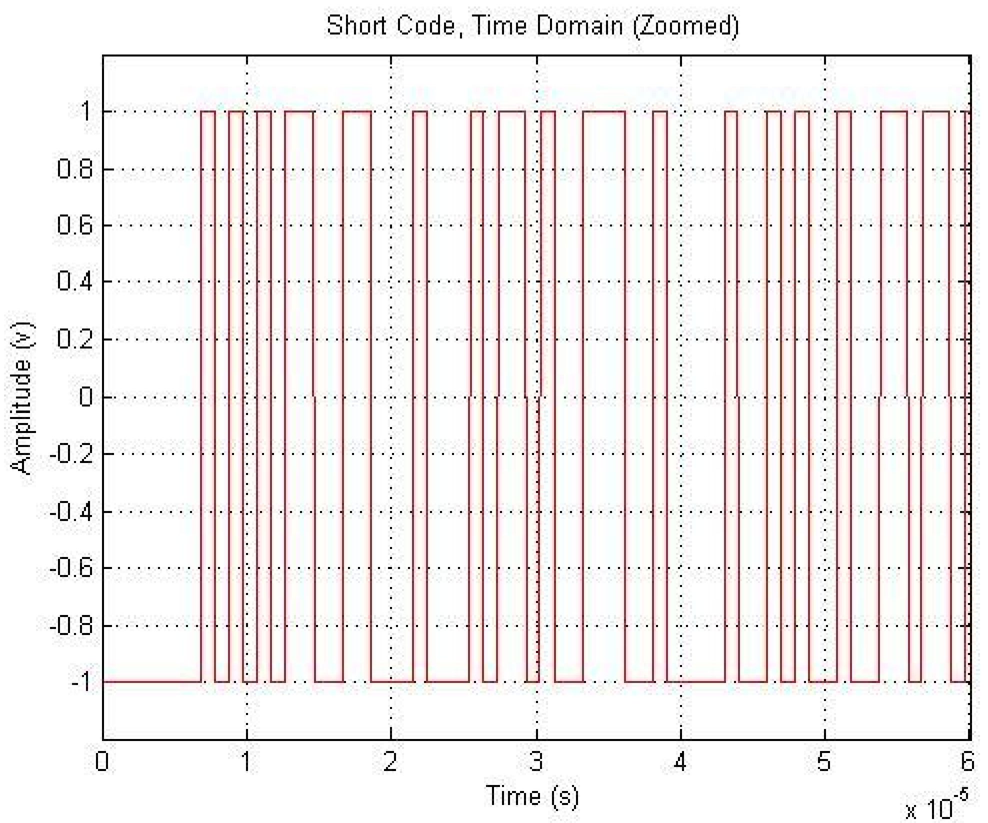
Shorter codes

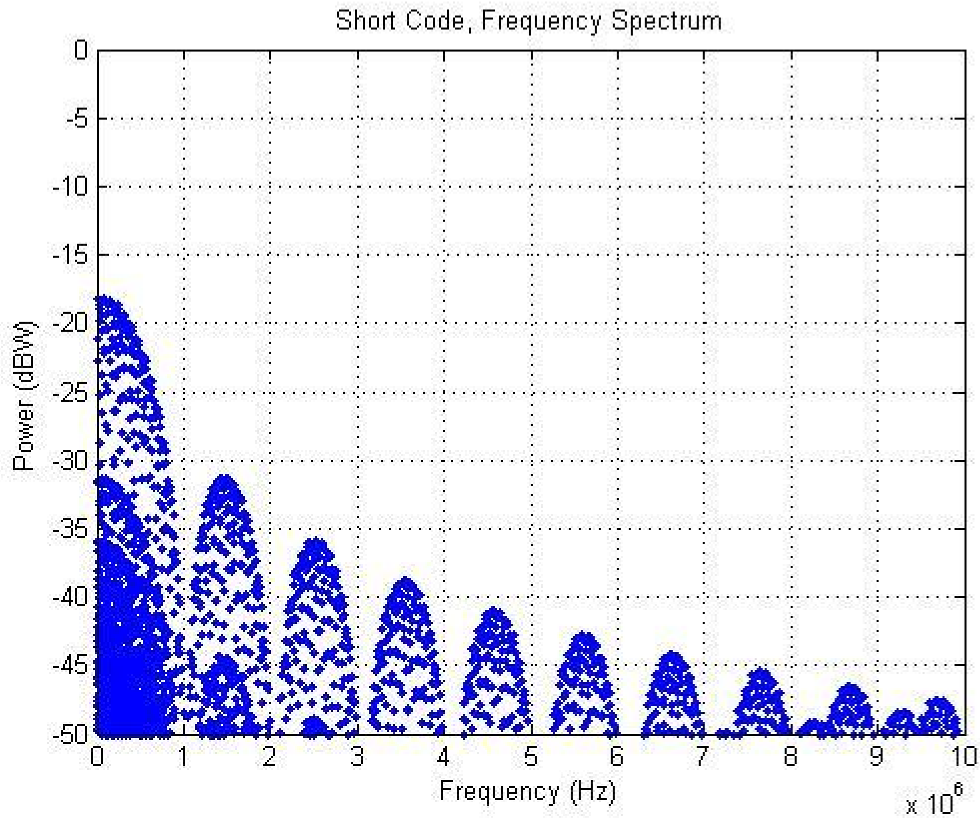
The time domain plots show a square wave with frequency 1.023e6 Hz. Zoom plot shows the pseudo-randomness of the signal. It repeats every 126 chips.

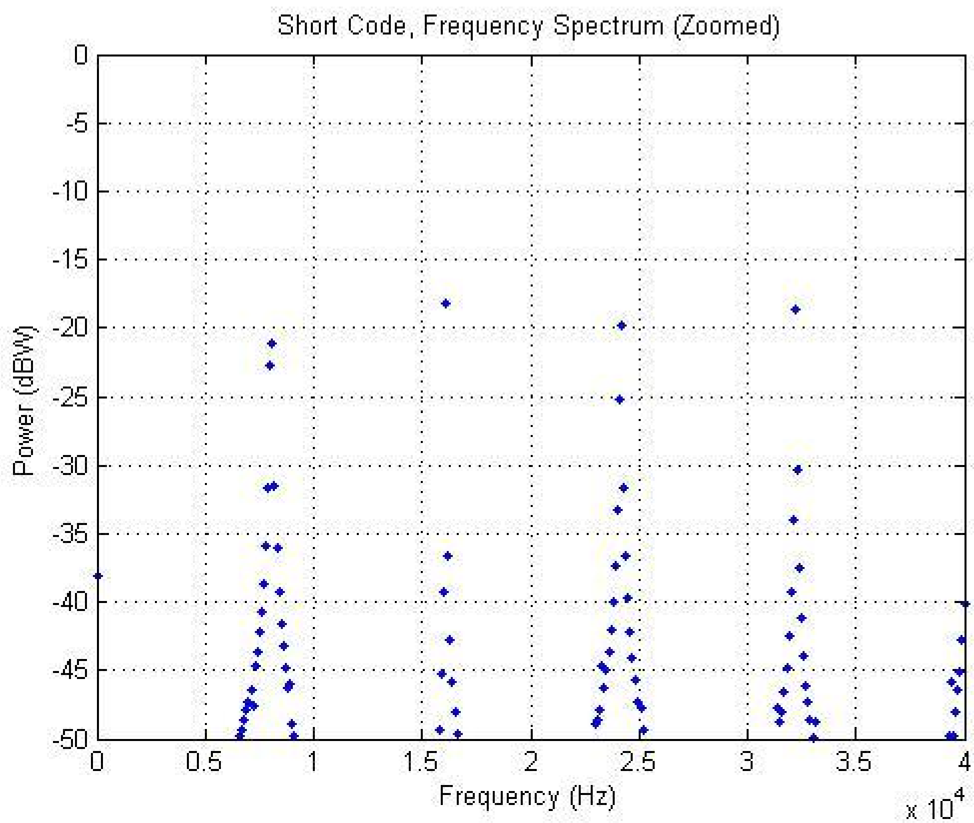
The frequency spectrum plots show a main lobe about zero with a width of 2.046e6 Hz. Side lobes are smaller with a width of 1.023e6 Hz. Zoom plot shows the code repeats at 8.1 kHz due to the repeat frequency (fc/num_chips) = fp. Thus, number of chips is 126 for this code before repeat.

```
small_register = BitShiftRegister(7, [7,1]);
SR_samples = zeros(num_samples,1);
SR_update_time = 0;
for ii = 1:num_samples
    if time(ii) >= SR_update_time - 1/chip_rate/100
%     if mod(ii,10) == 1
        sample = small_register.update();
        if sample == 1
            SR_samples(ii) = -1;
        else
            SR_samples(ii) = 1;
        end
%         G1_samples(ii) = G1.update();
        SR_update_time = SR_update_time + 1/chip_rate;
    else
        SR_samples(ii) = SR_samples(ii-1);
    end
end
SR_samples_freq = fft(SR_samples)*sqrt(2)/num_samples;
plot_v_time_domain(time, SR_samples, ...
    'Short Code, Time Domain');
plot_v_time_domain(time, SR_samples, ...
    'Short Code, Time Domain (Zoomed)', ...
    [0,6e-5],[-1.2,1.2],'r');
Ps = 20*log10(abs((SR_samples_freq)));
spectrum_analyzer(freqs, Ps, ...
    'Short Code, Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
spectrum_analyzer(freqs, Ps, ...
    'Short Code, Frequency Spectrum (Zoomed)', ...
    [0,4e4], [-50,0],'.');
```


Short Code, Time Domain
Amplitude (v)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time (s)


Short Code, Time Domain (Zoomed)
Amplitude (v)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
Time (s)
x 10^-5


Short Code, Frequency Spectrum
Power (dBW)
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
0
1
2
3
4
5
6
7
8
9
10
Frequency (Hz)
x 10^6


Short Code, Frequency Spectrum (Zoomed)
Power (dBW)
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
0
0.5
1
1.5
2
2.5
3
3.5
4
Frequency (Hz)
x 10^4

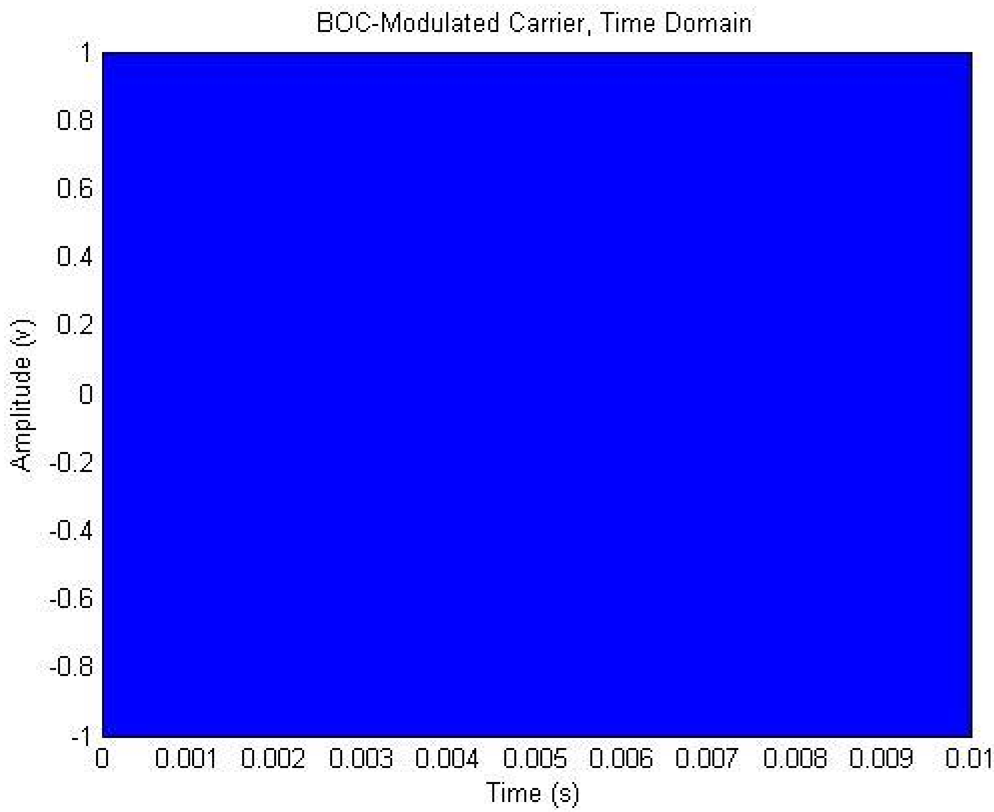
BOC

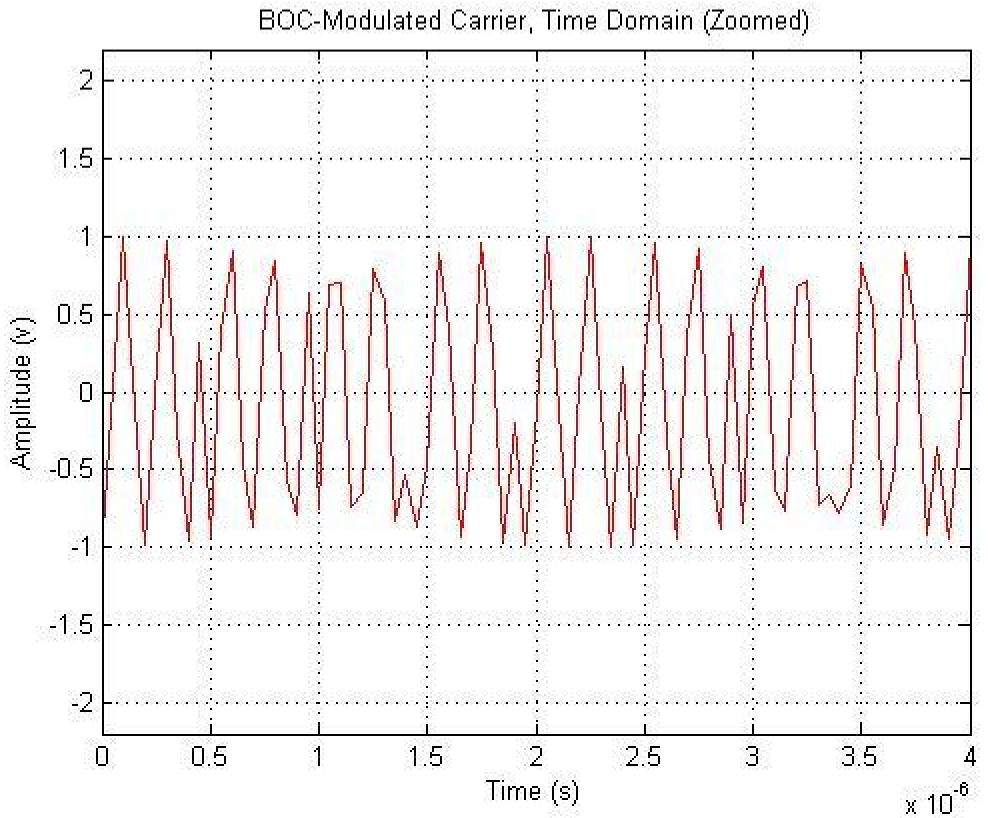
The time domain plots show a sampled sign wave that is biphase-shift keyed. That means the phase moves 180 degrees depending on the chip value of 1 or -1. The G1 wave has its amplitude doubled so you can clearly see how BOC follows it.

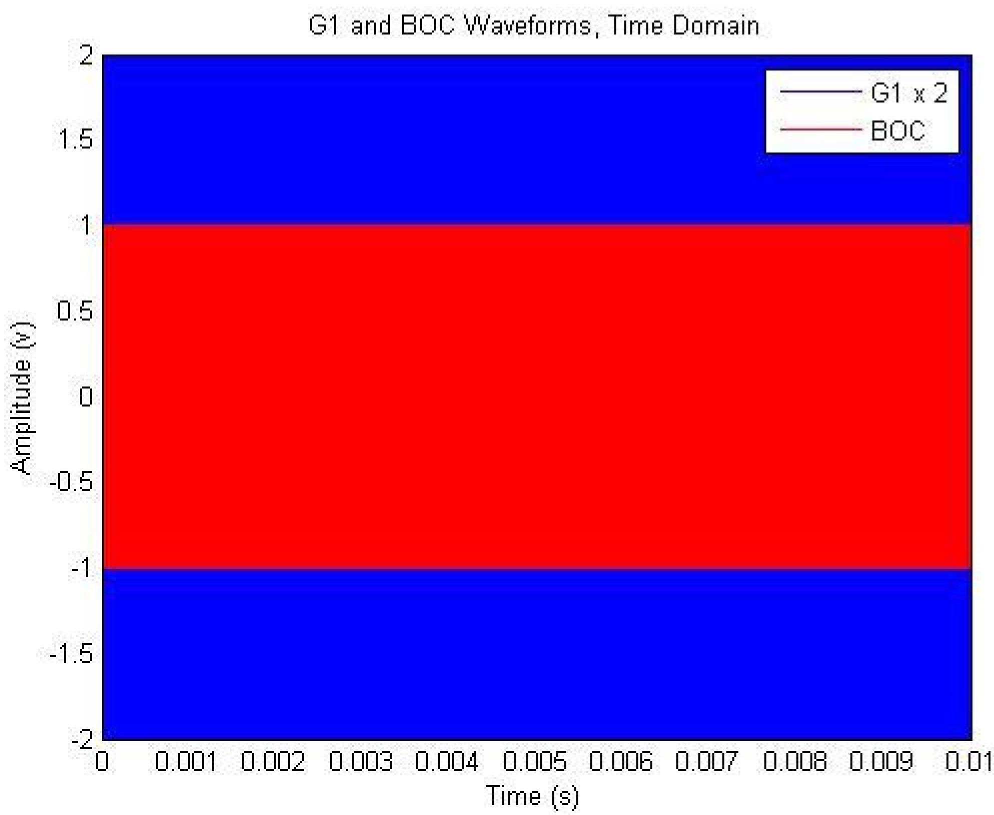
The frequency spectrum plots shows main lobes about the carrier frequency (5*1.023e6 Hz). It has a width of 1.023e6 Hz. Side lobes are with a width of 1.023e6 Hz. Zoom plot shows the code repeats at 1 kHz due to the repeat frequency (fc/num_chips), which remains the same as G1's.

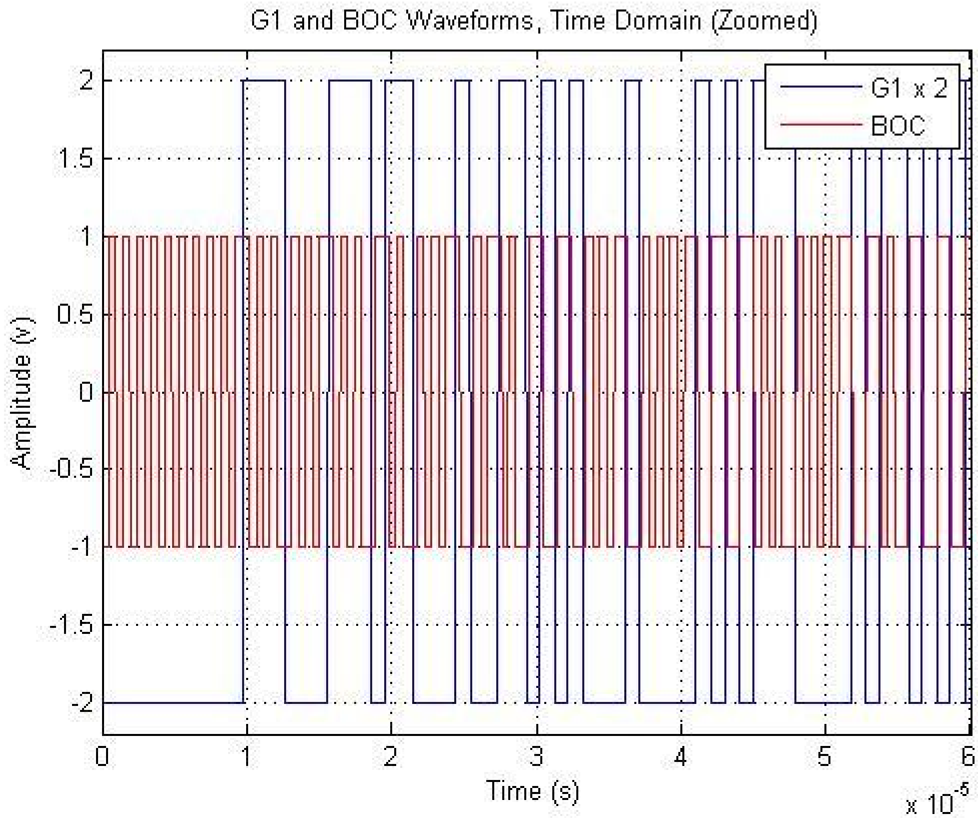
```
G1 = BitShiftRegister(10, [3,10]);
G1_code = [];
for ii = 1:1023
    G1_code = [G1_code G1.update()];
end
% 1/0 -> -1/1
G1_code = G1_code*-1;
G1_code(G1_code==0) = 1;

a = 1;
b = 1;
norm_chip_rate = 1.032e6; % cps
M = 2*a/b;
chip_rate = b*norm_chip_rate;
subcarrier_f = a*norm_chip_rate;
BOC_chip_rate = 2*subcarrier_f; % chip rate is 2x frequency of the subcarr.
% Create the BOC waveform
BOC = zeros(1023*M,1);
for ii = 1:1023
    for jj = 1:M
%         BOC((ii-1)*M+jj) = prn5.CA_code(ii)*-1^jj;
        BOC((ii-1)*M+jj) = G1_code(ii)*(-1)^(jj-1);
    end
end

% Sample the wave
BOC_samples = zeros(num_samples,1);
BOC_update_time = 0;
code_idx = 1;
for ii = 1:num_samples
    if time(ii) >= BOC_update_time - 1/BOC_chip_rate/100
        BOC_samples(ii) = BOC(code_idx);
        BOC_update_time = BOC_update_time + 1/BOC_chip_rate;
        code_idx = code_idx+1;
        if code_idx > 1023*M
            code_idx = 1;
        end
    else
        BOC_samples(ii) = BOC_samples(ii-1);
    end
end
plot_v_time_domain(time, BOC_samples.*carrier', ...
    'BOC-Modulated Carrier, Time Domain');
```

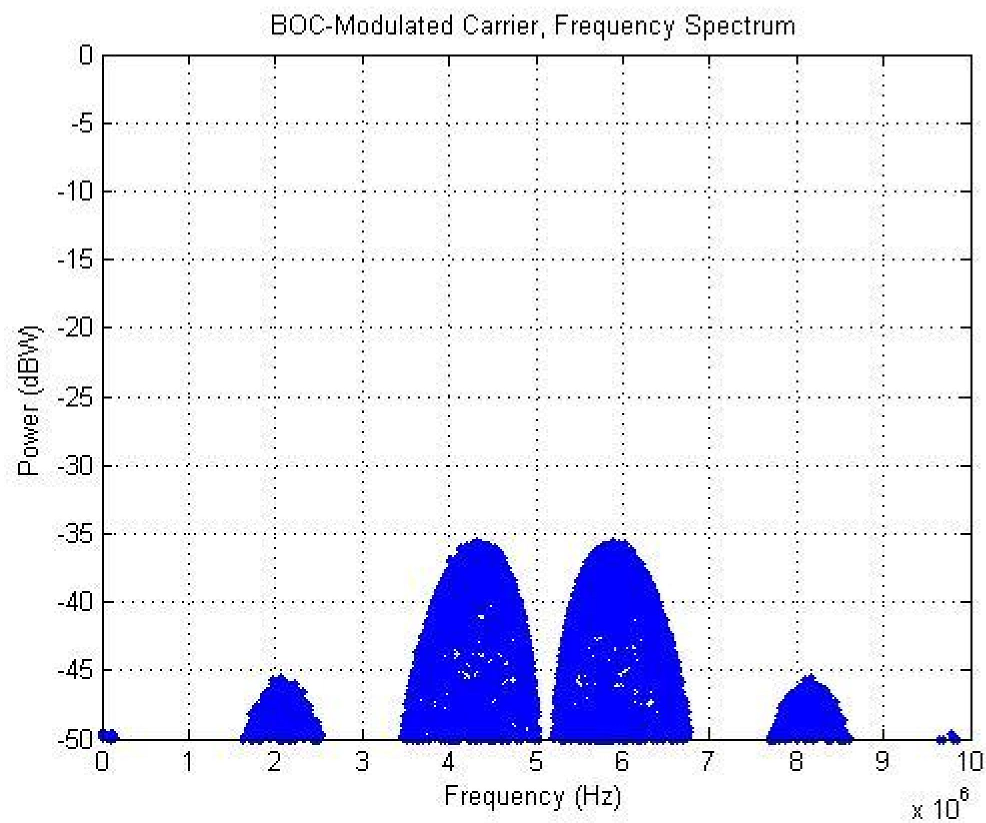
```
plot_v_time_domain(time, BOC_samples.*carrier', ...
    'BOC-Modulated Carrier, Time Domain (Zoomed)', ...
    [0,4e-6],[-2.2,2.2],'r');
plot_v_time_domain(time, 2*G1_samples, ...
    'G1 and BOC Waveforms, Time Domain');
hold on
plot(time,BOC_samples,'r')
legend('G1 x 2', 'BOC')
plot_v_time_domain(time, 2*G1_samples, ...
    'G1 and BOC Waveforms, Time Domain (Zoomed)', ...
    [0,6e-5],[-2.2,2.2]);
hold on
plot(time,BOC_samples,'r')
legend('G1 x 2', 'BOC')

Ps = 20*log10(abs(fft(BOC_samples.*carrier')*sqrt(2)/num_samples));
spectrum_analyzer(freqs, Ps, ...
    'BOC-Modulated Carrier, Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
spectrum_analyzer(freqs, Ps, ...
    'BOC-Modulated Carrier, Frequency Spectrum (Zoomed)', ...
    [6.133e6,6.14e6], [-55,0],'.');
```

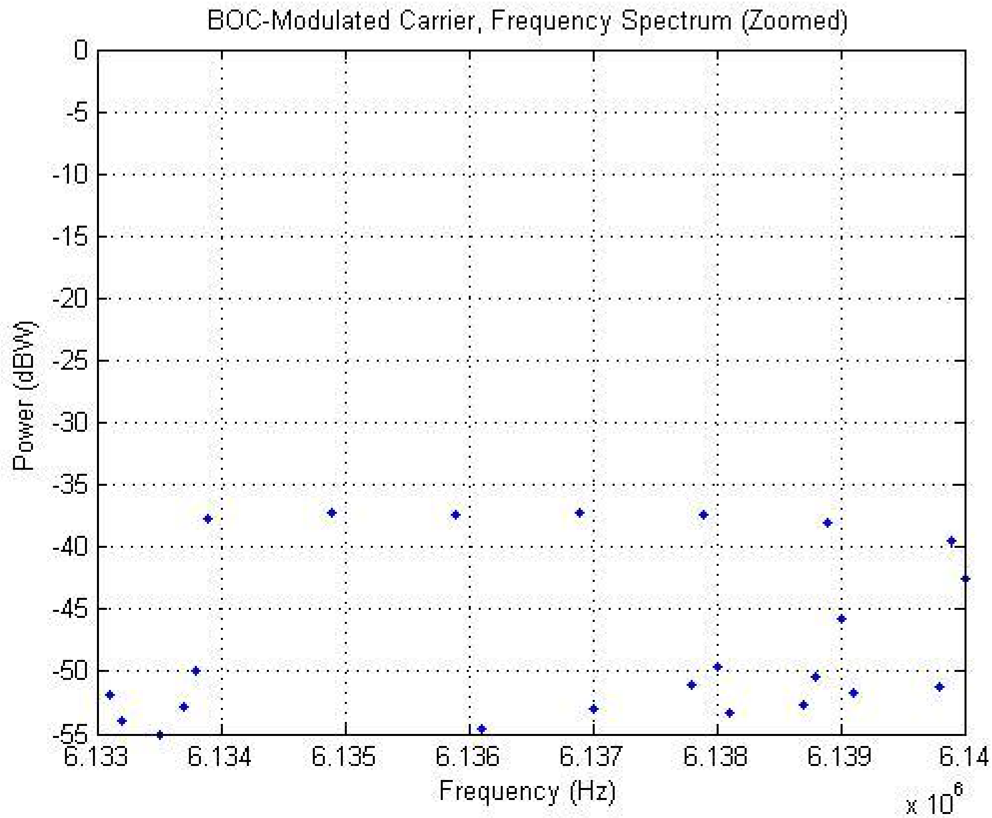



BOC-Modulated Carrier, Time Domain (Zoomed)
Amplitude (v)
Time (s)
x 10^-6
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.5
1
1.5
2
2.5
3
3.5
4


G1 and BOC Waveforms, Time Domain
G1 x 2
BOC
Amplitude (v)
Time (s)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01


G1 and BOC Waveforms, Time Domain (Zoomed)
G1 x 2
BOC
Amplitude (v)
Time (s)
x 10^-5


BOC-Modulated Carrier, Frequency Spectrum
Power (dBW)
Frequency (Hz)
x 10^6

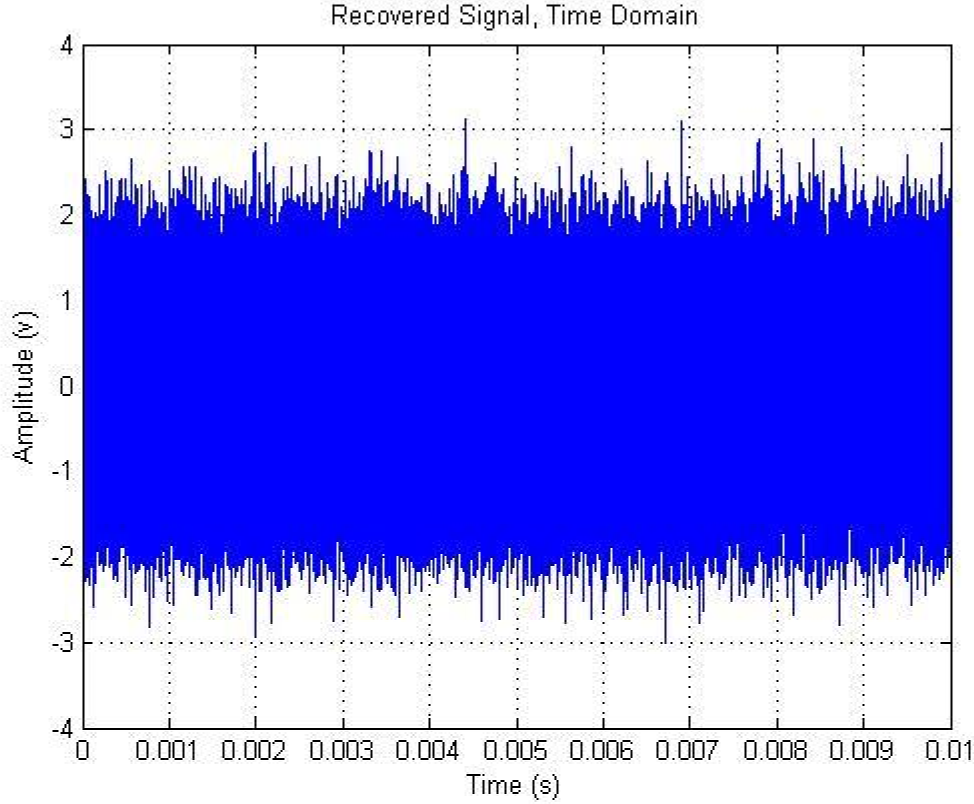


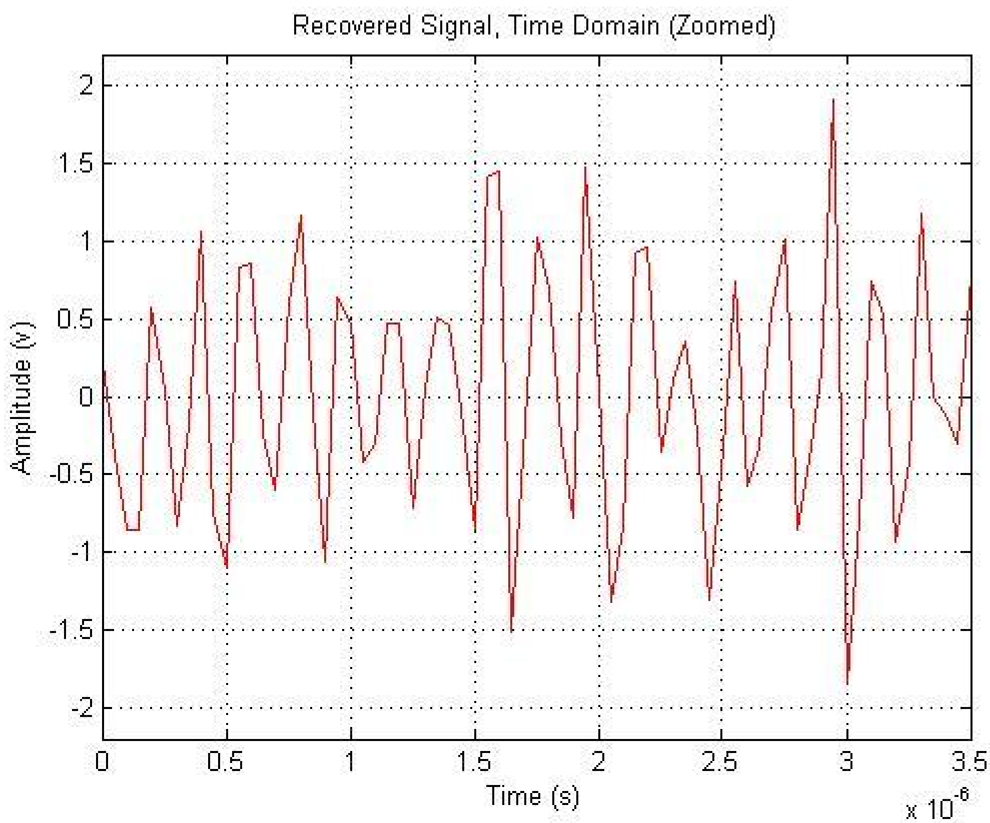
Demodulation/Carrier Recovery

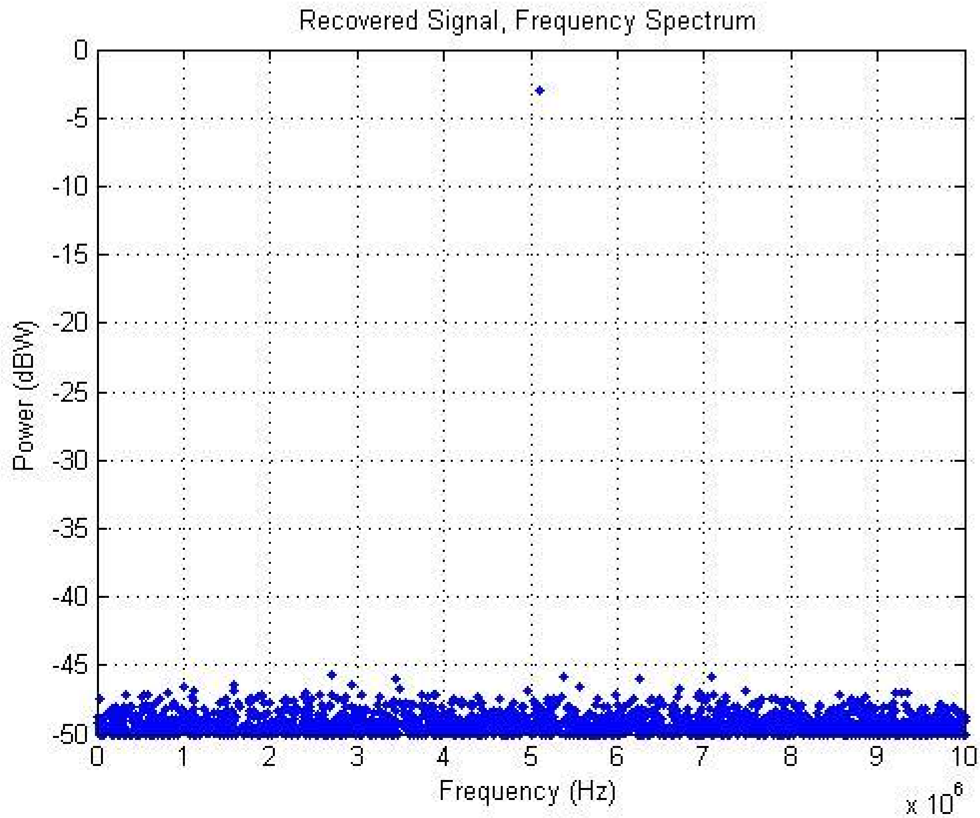
The time domain plots show a sampled sign wave with noise that is biphase-shift keyed. That means the phase moves 180 degrees depending on the chip value of 1 or -1.

The frequency spectrum plot shows the carrier is recovered, spiking at 5*1.023e6 Hz.

```
noisy_signal = mod_carrier + .5*randn(num_samples,1);
recovered_signal = noisy_signal.*G1_samples;
Ps = 20*log10(abs(fft(recovered_signal)*sqrt(2)/num_samples));
plot_v_time_domain(time, recovered_signal, ...
    'Recovered Signal, Time Domain');
plot_v_time_domain(time, recovered_signal, ...
    'Recovered Signal, Time Domain (Zoomed)', ...
    [0,3.5e-6],[-2.2,2.2],'r');
spectrum_analyzer(freqs, Ps, ...
    'Recovered Signal, Frequency Spectrum', ...
    [0,Nyquist_freq], [-50,0],'.');
```


Recovered Signal, Time Domain
Amplitude (v)
Time (s)
4
3
2
1
0
-1
-2
-3
-4
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01


Recovered Signal, Time Domain (Zoomed)
Amplitude (v)
Time (s)
x 10-6
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.5
1
1.5
2
2.5
3
3.5



Followup

L1 spectrum will now have military codes with main lobes offset from f_L1 due to BOC. L2 will also have that, plus a civil signal similar to L1 C/A code's spectrum (narrow lobe due to PRN repeat freq). L5 will have longer and faster codes, so it will have a wide main lobe similar to L1/2's P(Y) spectra. L2 and L5 will have a data-free component.

Higher chip rate -> wider signal bandwidth that the receiver has to deal with. But it can result in better range accuracy. (Navipedia)

Longer codes result in smaller cross correlation, but takes more time to acquire. (Navipedia)

Published with MATLAB® R2013b