HW2 Problem 1: PRN generation

Table of Contents

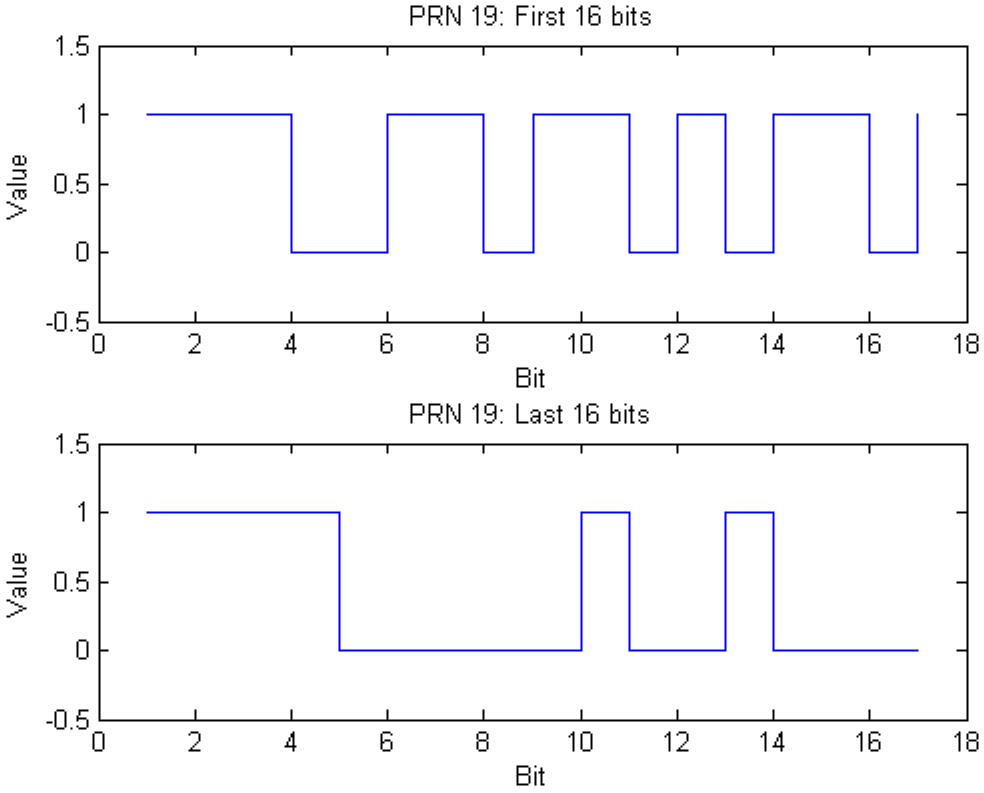
Initialize

```
fprintf('\n');
clearvars -except function_list pub_opt
close all
```

a) Plot PRN 19 chips

```
prn19=PRNCode(19);
for i = 1:1023
prn19.update()
end

binaryVectorToHex((prn19.CA_code(1:16)))
hexToBinaryVector('E6D6');
hw2_code_plot(prn19)
```

```
C:\Users\John\Documents\ASEN5090_GNSS\tools\PRNCode
C:\Users\John\Documents\ASEN5090_GNSS\tools\BitShiftRegister

ans =

E6D6

C:\Users\John\Documents\ASEN5090_GNSS\Homework\HW2\hw2_code_plot
```



b) PRN 19 chips 1024:2046

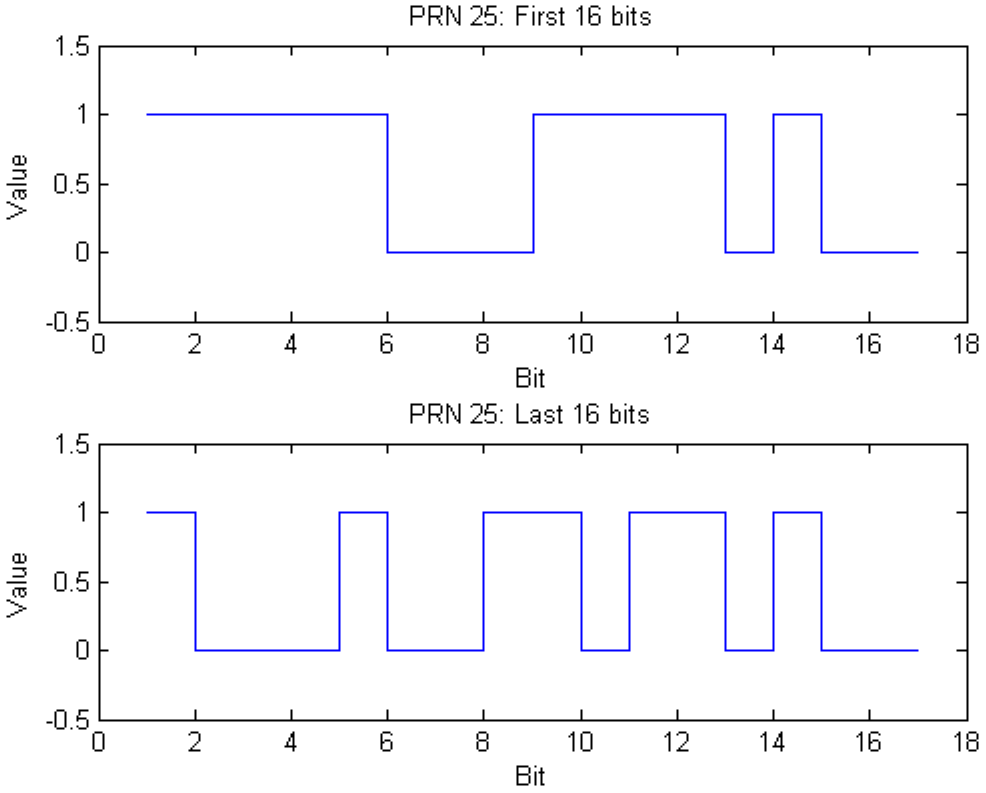
These chips are the same as 1:1023

```
for i = 1:1023
prn19.update()
end

sum(prn19.CA_code(1:1023)-prn19.CA_code(1024:end))
```

```
ans =

     0
```

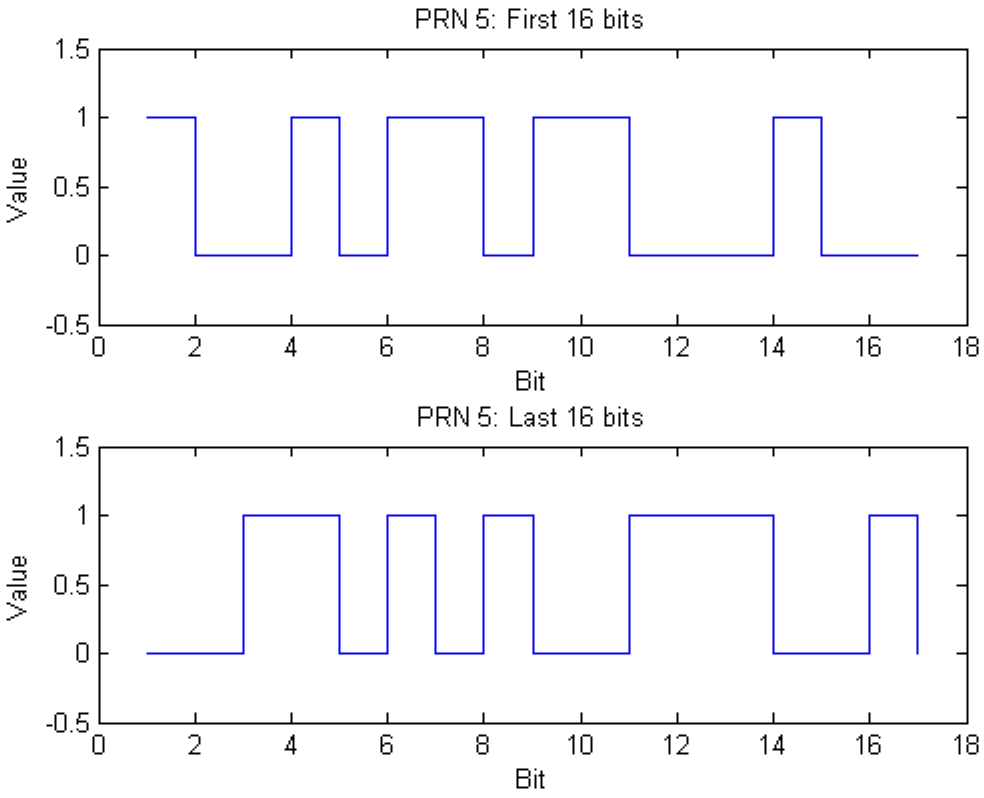
c) Plot PRN 25 chips

```
prn25=PRNCode(25);
for i = 1:1023
prn25.update()
end

hw2_code_plot(prn25)
```



d) Plot PRN 5 chips

```
prn5=PRNCode(5);
for i = 1:1023
prn5.update()
end

hw2_code_plot(prn5)
```



PRN 5: First 16 bits

PRN 5: Last 16 bits

Published with MATLAB® R2013b

HW2 Problem 2: Code correlations

Table of Contents

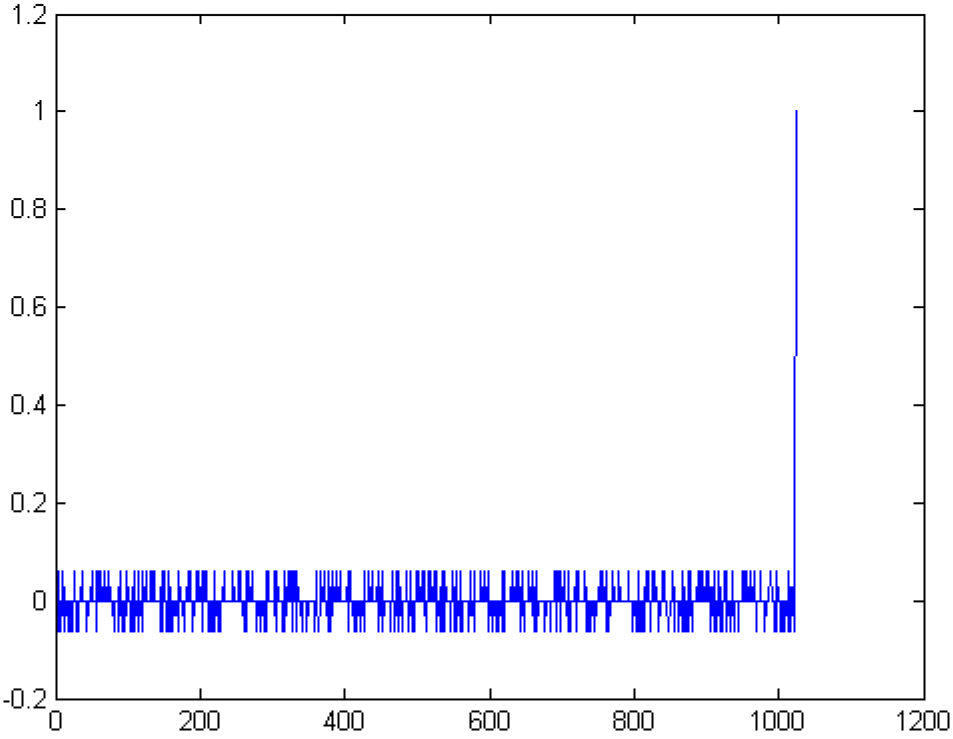
Initialize

```
fprintf('\n');
clearvars -except function_list pub_opt
close all

prn19=PRNCode(19);
prn25=PRNCode(25);
prn5=PRNCode(5);
make_delay = @(code, delay) [code(delay+1:end), code(1:delay)];
prn19_delay=350;
prn25_delay=905;
prn5_delay=75;

for i = 1:1023
    prn19.update()
    prn25.update()
    prn5.update()
end


offsets_correlation = zeros(1,1024);
```

a) auto-correlation of PRN19

```
for ii = 0:1023
    offsets_correlation(ii+1) = ...
        normalized_correlation(prn19.CA_code, prn19.CA_code, ii);
end
figure
plot(offsets_correlation)
```

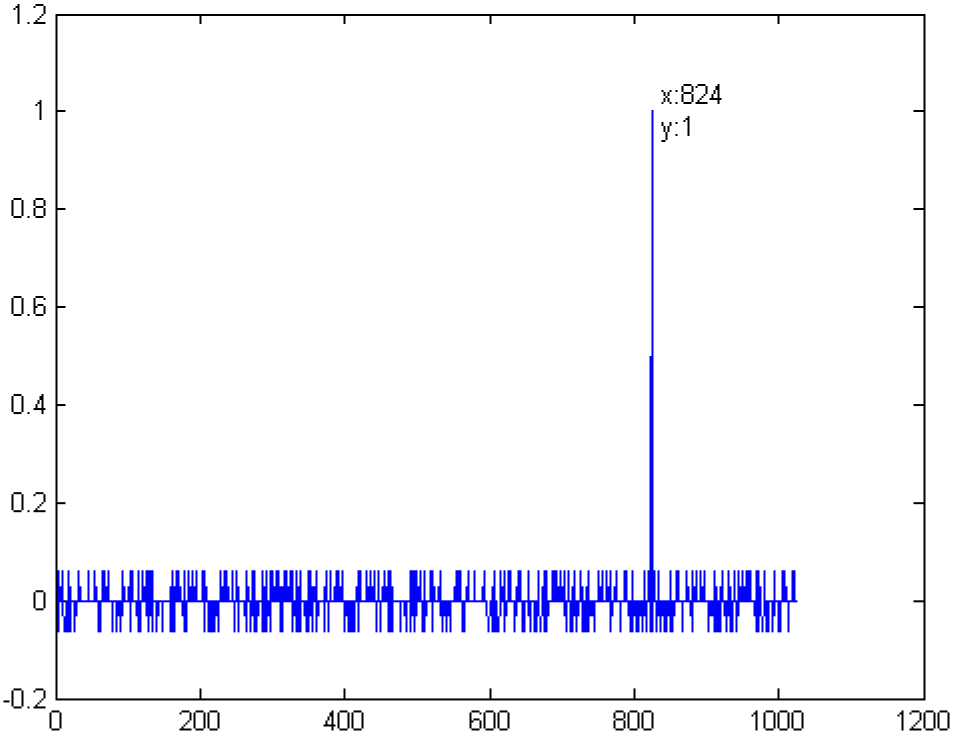
```
        C:\Users\John\Documents\ASEN5090_GNSS\tools\normalized_correlation
```



b) cross-correlation of PRN19 and PRN19 delayed by 200

The peak value's location makes sense because the 2nd code was shifted by 200, making the correlation happen 200 bits from the end.

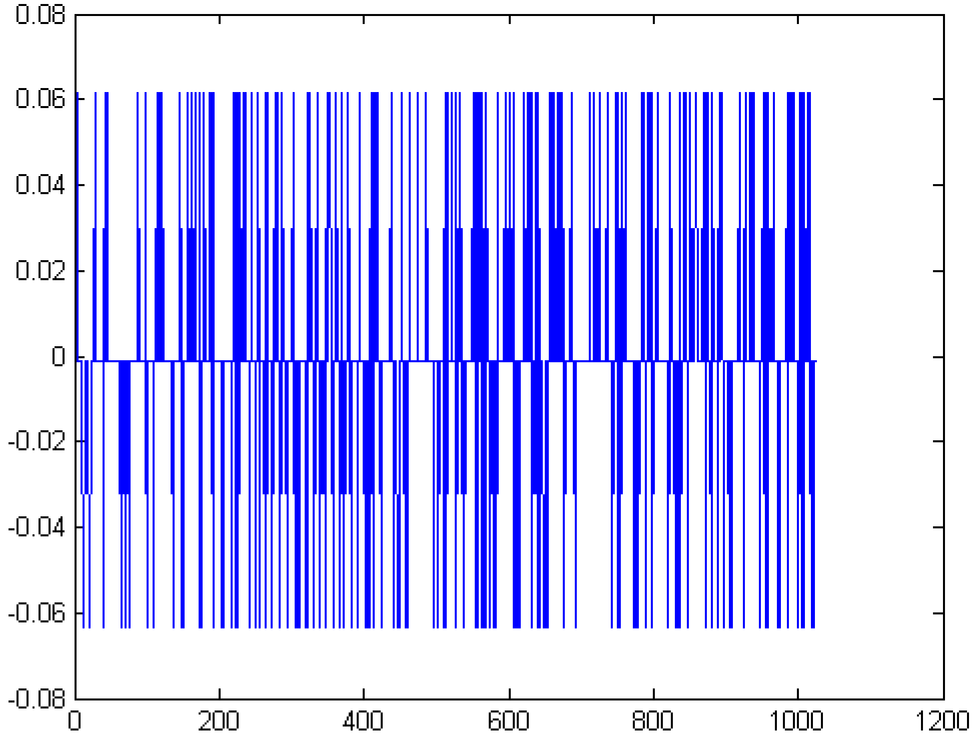
```
prn19_200delay = make_delay(prn19.CA_code, 200);
for ii = 0:1023
    offsets_correlation(ii+1) = ...
        normalized_correlation(prn19.CA_code, prn19_200delay, ii);
end
figure
plot(offsets_correlation);
[Y, I] = max(offsets_correlation);
text(I(1), Y, sprintf(' x:%d\n y:%g', I(1), Y))
```



c) cross-correlation of PRN19 and PRN25

No good correlations between PRNs 19 and 25

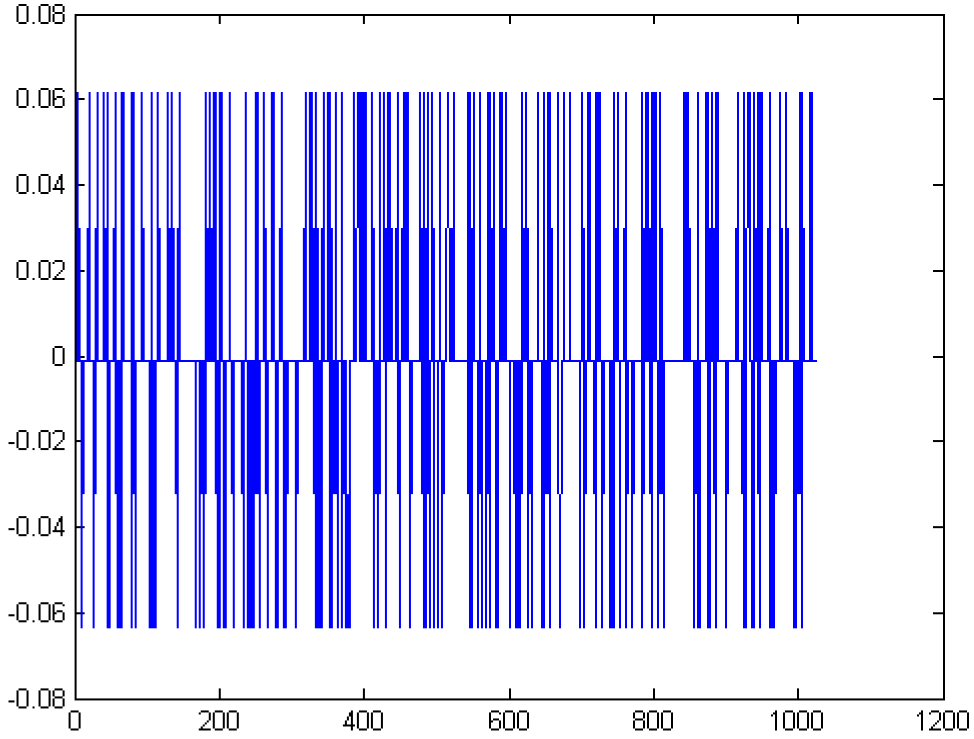
```
for ii = 0:1023
    offsets_correlation(ii+1) = ...
        normalized_correlation(prn19.CA_code, prn25.CA_code, ii);
end
figure
plot(offsets_correlation)
```



d) cross-correlation of PRN19 and PRN5

No good correlations between PRNs 19 and 5

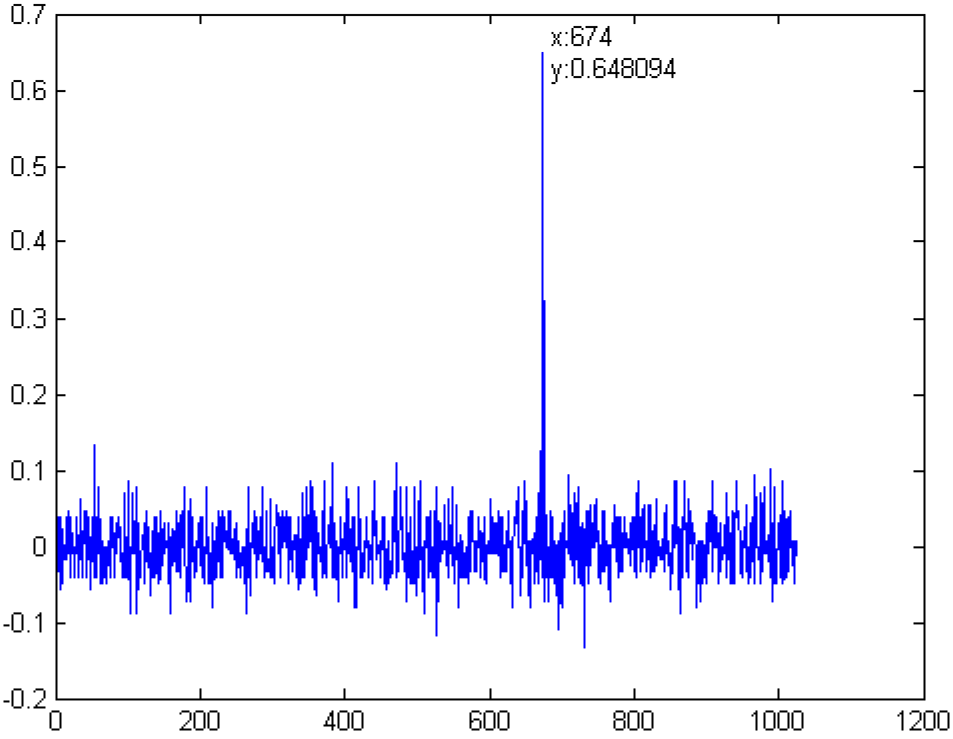
```
for ii = 0:1023
    offsets_correlation(ii+1) = ...
        normalized_correlation(prn19.CA_code, prn5.CA_code, ii);
end
figure
plot(offsets_correlation)
```



e) cross-correlation of PRN19 and summed PRNs

The peak value's location makes sense because the 2nd code was shifted by 350, making the correlation happen 350 bits from the end. The codes correlate despite the other codes on top of PRN 19.

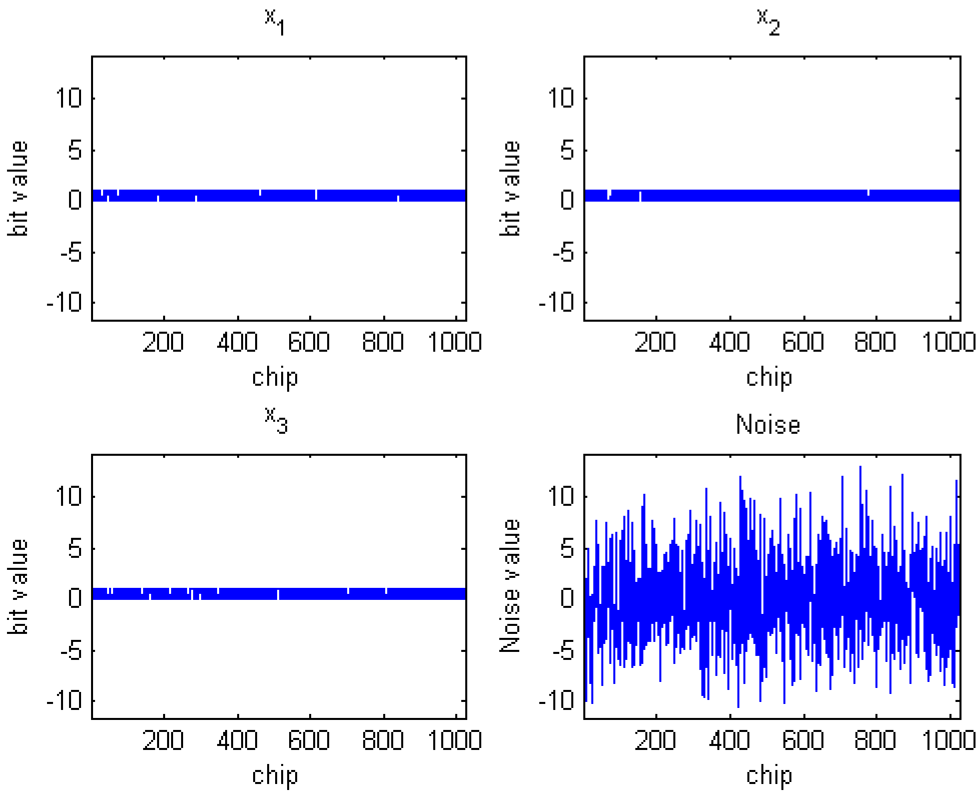
```
x1 = make_delay(prn19.CA_code, prn19_delay);
x2 = make_delay(prn25.CA_code, prn25_delay);
x3 = make_delay(prn5.CA_code, prn5_delay);
summed_prns = x1+x2+x3;
for ii = 0:1023
    offsets_correlation(ii+1) = ...
        normalized_correlation(prn19.CA_code, summed_prns, ii);
end
figure
plot(offsets_correlation)
[Y, I] = max(offsets_correlation);
text(I(1), Y, sprintf(' x:%d\n y:%g', I(1), Y))
```



f) Noise

```
noise = 4*randn(1,1023);
figure
subplot(2, 2, 1)
plot(x1)
title('x_1')
xlabel('chip')
ylabel('bit value')
ylim([min(noise)*1.1, max(noise)*1.1])
xlim([1, 1023])
subplot(2, 2, 2)
plot(x2)
title('x_2')
xlabel('chip')
ylabel('bit value')
ylim([min(noise)*1.1, max(noise)*1.1])
xlim([1, 1023])
subplot(2, 2, 3)
plot(x3)
title('x_3')
xlabel('chip')
ylabel('bit value')
ylim([min(noise)*1.1, max(noise)*1.1])
xlim([1, 1023])
```

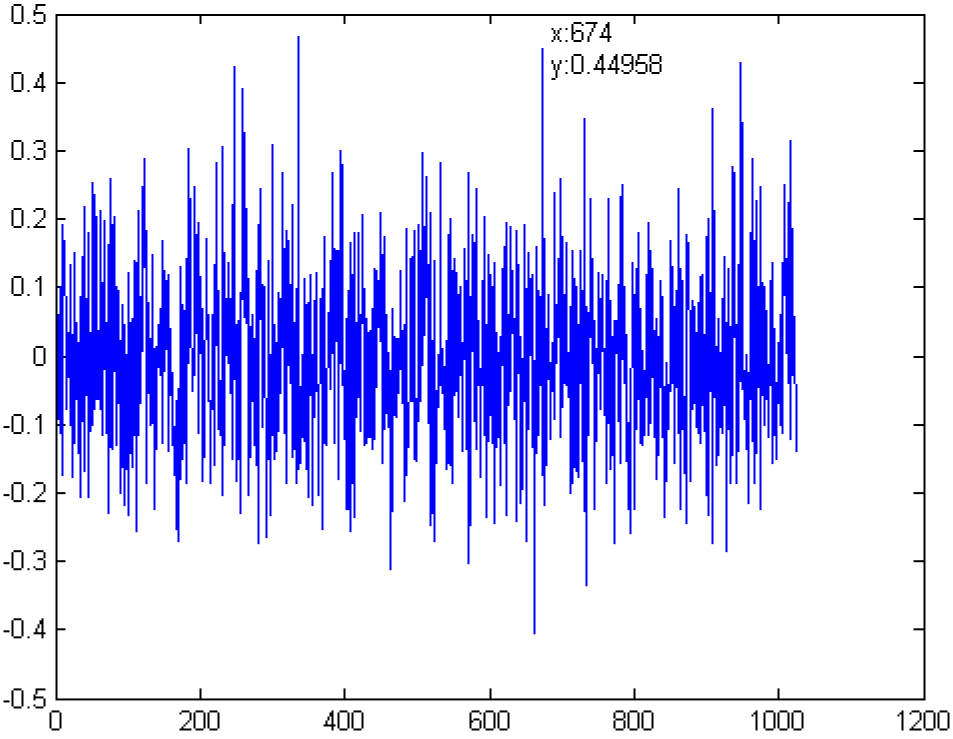
```
subplot(2, 2, 4)
plot(noise)
title('Noise')
xlabel('chip')
ylabel('Noise value')
ylim([min(noise)*1.1, max(noise)*1.1])
xlim([1, 1023])
```



g) cross-correlation of PRN19 and summed PRNs + noise

The peak is where I expect it to be, however it's much closer in scale to other local peaks. I ran the noise function multiple times and sometimes the correlation really stood out, sometimes it didn't.

```
summed_prns = x1+x2+x3+noise;
for ii = 0:1023
    offsets_correlation(ii+1) = ...
        normalized_correlation(prn19.CA_code, summed_prns, ii);
end
figure
plot(offsets_correlation)
text(I(1), offsets_correlation(I(1)), ...
    sprintf(' x:%d\n y:%g', I(1), offsets_correlation(I(1))))
```



Published with MATLAB® R2013b

HW 2 Master Script

Table of Contents

Initialize

```
if ispc
    addpath('C:\Users\John\Documents\ASEN5090_GNSS\tools')
end
clear all
clc

% Cell array to track what functions are used, so they can be published
% later
global function_list;
function_list = {};

% publishing options
pub_opt.format = 'pdf';
pub_opt.outputDir = '.\html';
pub_opt.imageFormat = 'bmp';
pub_opt.figureSnapMethod = 'entireGUIWindow';
pub_opt.useNewFigure = true ;
pub_opt.maxHeight = Inf;
pub_opt.maxWidth = Inf;
pub_opt.showCode = true;
pub_opt.evalCode = true;
pub_opt.catchError = true;
pub_opt.createThumbnail = true;
pub_opt.maxOutputLines = Inf;
```

Run Problem scripts and publish them

```
% Problem 1
publish('HW2_P1', pub_opt);

% Problem 2
publish('HW2_P2', pub_opt);
```

Publishing tools and support code

```
pub_opt.outputDir = '.\tools';
pub_opt.evalCode = false;
```

```
%Publish all used functions
function_list = ...
    [function_list, 'C:\Users\John\Documents\ASEN5090_GNSS\tools\fcnPrintQueue'];
for idx = 1:length(function_list)
    publish(function_list{idx}, pub_opt);
end
```

Published with MATLAB® R2013b

```
classdef BitShiftRegister < handle
    properties
        bits = []
        taps = []
        size = 0;
        num_taps = 0;
    end

    methods
        function BSR = BitShiftRegister(size, taps)
            %BitShiftRegister Construct a bit shift register with taps
            %   taps must be a vector
            fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

            %TODO: are the taps legal? Are there at least two?
            BSR.bits = ones(1,size);
            BSR.size = size;
            BSR.taps = taps; % at least two
            BSR.num_taps = length(taps);
        end

        function tap_result = bit_shift_register( BSR )
            %bit_shift_register Compute the tap result

            %TODO: are the taps legal? Are there at least two?
            xor_bits = zeros(1,BSR.num_taps);
            counter = 1;
            for i = BSR.taps
                xor_bits(counter) = BSR.bits(i);
                counter = counter+1;
            end
            tap_result = recurse_xor(xor_bits);
        end

        function code = update( BSR)
            %update Output the code and update the bits in the register
            code = BSR.bits(BSR.size);
            new_bits = [BSR.bit_shift_register() BSR.bits(1:BSR.size-1)];
            BSR.bits = new_bits;
        end
    end
end
```

Published with MATLAB® R2013b

```
function fcnPrintQueue( filename )
global function_list;
if exist('function_list', 'var')
    file_in_list = 0;
    for idx = 1:length(function_list)
        if strcmp(function_list(idx), filename);
            file_in_list = 1;
            break
        end
    end
    if ~file_in_list
        fprintf('%s\n', filename);
        function_list = [function_list, filename];
    end
end
end
```

Published with MATLAB® R2013b

```
function hw2_code_plot(code)
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

figure
subplot(2,1,1)
stairs(code.CA_code(1:17)), ylim([-0.5, 1.5])
title(sprintf('PRN %i: First 16 bits', code.PRN_Number))
xlabel('Bit')
ylabel('Value')
subplot(2,1,2)
stairs(code.CA_code(end-16:end)), ylim([-0.5, 1.5])
title(sprintf('PRN %i: Last 16 bits', code.PRN_Number))
xlabel('Bit')
ylabel('Value')
```

Published with MATLAB® R2013b

```
function out = normalized_correlation( in_code1, in_code2, shift )
%normalized_correlation Output the correlation between two codes.
%codes are row vectors
fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

%TODO Error check the codes
code_length = length(in_code1);

%Set the bits from 0,1 to 1,-1
code1 = in_code1*-1;
code2 = in_code2*-1;

for ii = 1:code_length
    if code1(ii) == 0
        code1(ii) = 1;
    end
    if code2(ii) == 0
        code2(ii) = 1;
    end
end

%Shift the second code
if shift > 0
    code2 = [code2(shift+1:end), code2(1:shift)];
elseif shift < 0
    code2 = [code2(code_length+shift+1:code_length), code2(1:code_length+shift)];
end

%Sum and divide
total = 0;
for ii = 1:code_length
    total = total + code1(ii) * code2(ii);
end

out = total/code_length;

end
```

Published with MATLAB® R2013b

```
classdef PRNCode < handle
    %UNTITLED3 Summary of this class goes here
    %   Detailed explanation goes here

    properties
        G1;
        G2;
        G1_out = [];
        G2_out = [];
        S1;
        S2;
        CA_code = [];
        PRN_Number = -1;
    end

    methods
        function PRN = PRNCode(PRN_number)
            fcnPrintQueue(mfilename('fullpath')) % Add this code to code app
            PRN.G1 = BitShiftRegister(10, [3,10]);
            PRN.G2 = BitShiftRegister(10, [2, 3, 6, 8, 9, 10]);

            %phase selection data obtained from
            %IS-GPS-200D
            phase_selections = [2 6;3 7;4 8;5 9;1 9;2 10;1 8;2 9;3 10;...
                2 3;3 4;5 6;6 7;7 8;8 9;9 10;1 4;2 5;3 6;4 7;5 8;6 9;...
                1 3;4 6;5 7;6 8;7 9;8 10;1 6;2 7;3 8;4 9;5 10;4 10;...
                1 7;2 8;4 10];
            PRN.S1 = phase_selections(PRN_number, 1);
            PRN.S2 = phase_selections(PRN_number, 2);
            PRN.PRN_Number = PRN_number;
        end
        function update(PRN)
            G1_out_scalar = PRN.G1.update();

            %TODO: consider reversing this?
            PRN.CA_code = [ PRN.CA_code recurse_xor([PRN.G2.bits(PRN.S1), ...
                PRN.G2.bits(PRN.S2), G1_out_scalar])];

            G2_out_scalar = PRN.G2.update();
            PRN.G1_out = [G1_out_scalar PRN.G1_out];
            PRN.G2_out = [G2_out_scalar PRN.G2_out];
        end
    end

end
```

Published with MATLAB® R2013b