

GPS - HW3

Walter Livingston

March 2024

Problem I

Write a controller (matlab or Simulink) for a simple $1/s$ plant. Assume a unit step input for the reference, $r(t)$.

Part A

What type of controller did you use. Provide the Gain Margin, Phase Margin, closed- loop eigenvalues, and steady state error.

Solution

I used a proportional controller. The characteristics of my controller are in the table below. Plots shown in C.

Gain Margin	∞
Phase Margin	90
Eigenvalues	0 ± -2
SS Error	~ 0

Part B

What is the steady state error if the reference $r(t)$ is a unit ramp input

Solution

Steady state error is 0.5.

Part C

Redesign the controller to track the ramp input and repeat Part A.

Solution

The controller was redesigned as a PI controller. The controller characteristics are listed in the table below.

Gain Margin	∞
Phase Margin	90
Eigenvalues	$-1 \& -1$
SS Error	~ 0

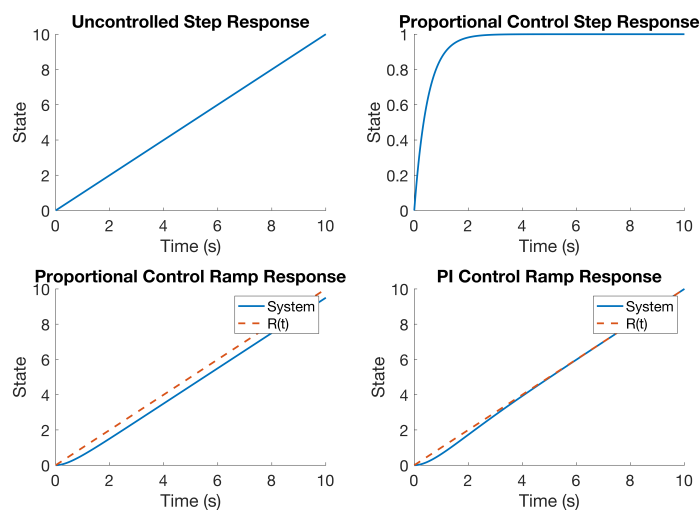


Figure 1: Problem 1 Step & Ramp Responses

Problem II

Take the sampled 100 Hz sine wave (`generate_signal(1)`) from the website (sampled at 1 MHz, i.e. $T_s = 1e - 6$)

Part A

Develop a simple (1 Hz) PLL to track the phase of the signal. Provide plots of phase, phase error, as well as the estimated signal vs. true signal.

Solution

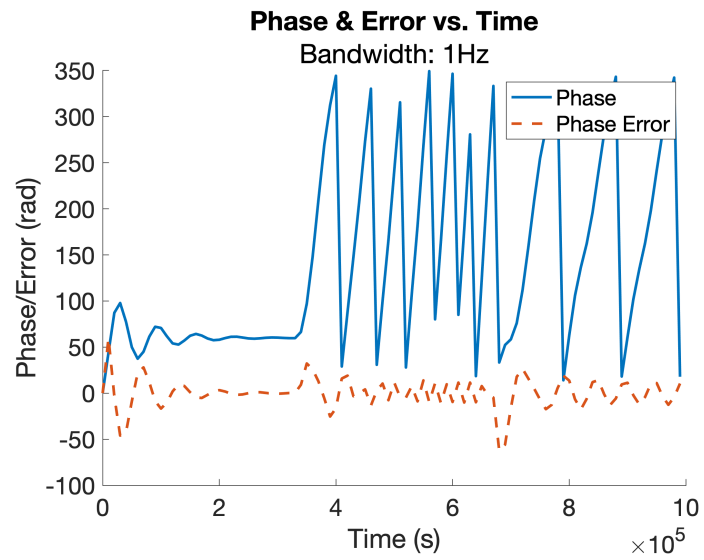


Figure 2: Phase & Error vs. Time for 1Hz Bandwidth

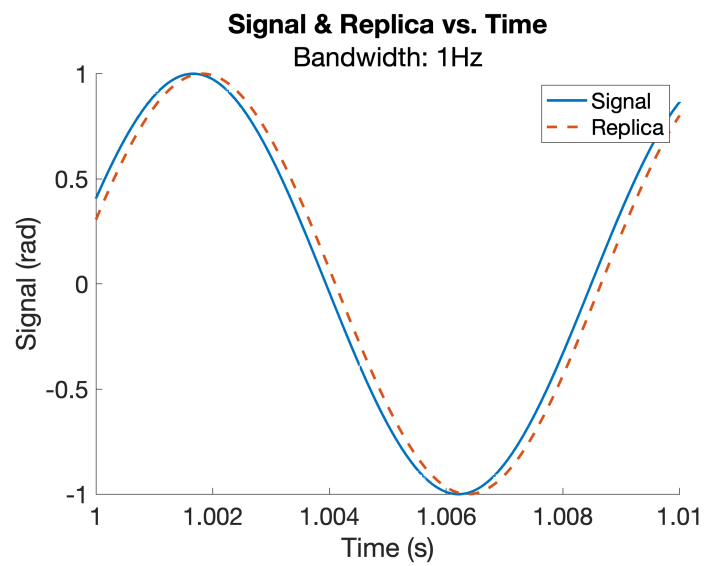


Figure 3: Signal & Replica vs. Time for 1Hz Bandwidth

Part B

Double the PLL bandwidth and repeat part a.

Solution

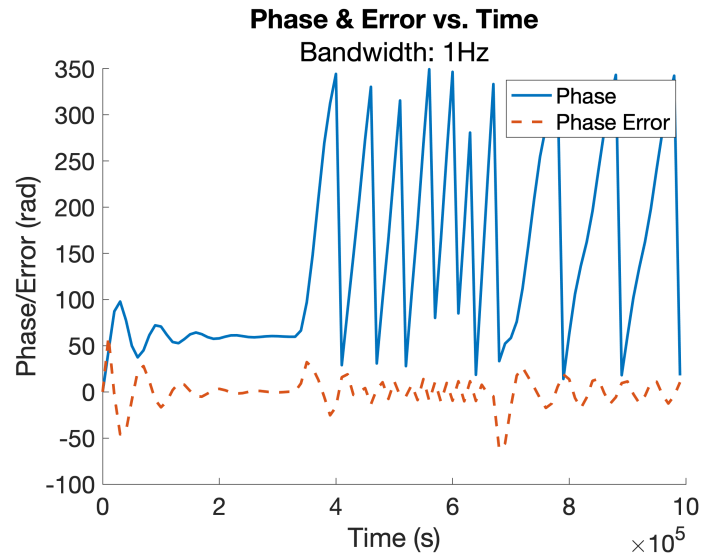


Figure 4: Phase & Error vs. Time for 2Hz Bandwidth

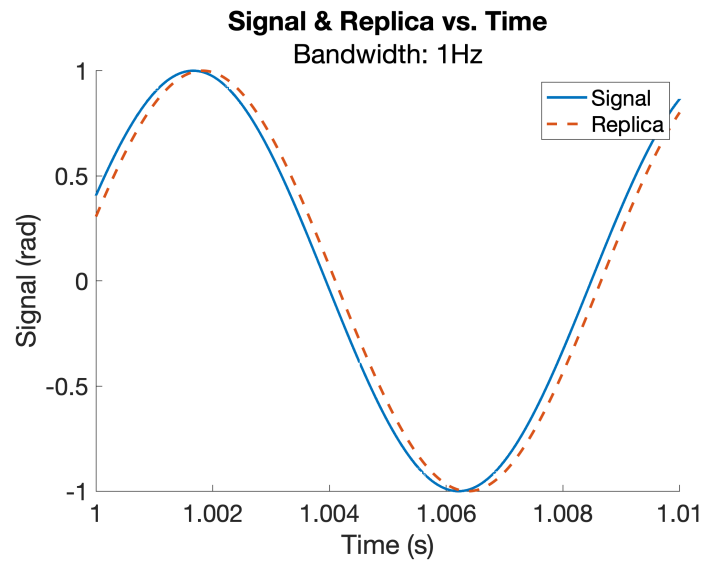


Figure 5: Signal & Replica vs. Time for 2Hz Bandwidth

Part C

Determine the true frequency and repeat part a.

Solution

Part D

Modify your PLL to work from the unknown frequency and repeat part a assuming 10 Hz signal.

Solution

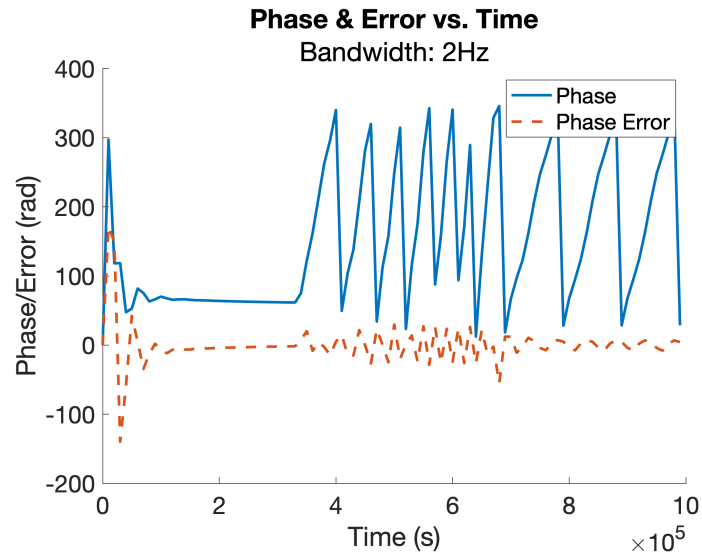


Figure 6: Phase & Error vs. Time for 10Hz Initial Frequency

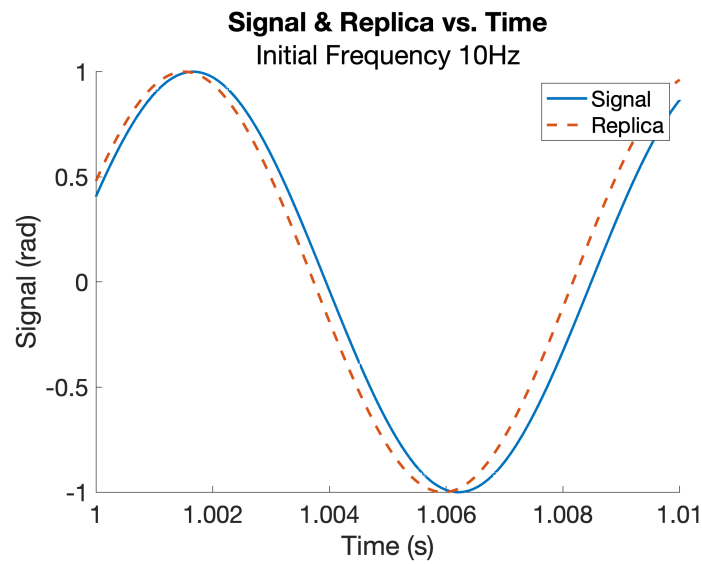


Figure 7: Signal & Replica vs. Time for 10Hz Initial Frequency

Problem III

Modify your PLL from problem #3 to operate as a Costas loop filter. Take the 88 second data (generate_signal(2)) sampled at 1 MHz and decode the data message on the 100 Hz sinusoid using your Costas loop filter. The data bits are 1 second wide and are comprised of 8 bit ascii characters. The following functions will be of use:

Solution

The databits translate to AU WarEagle.

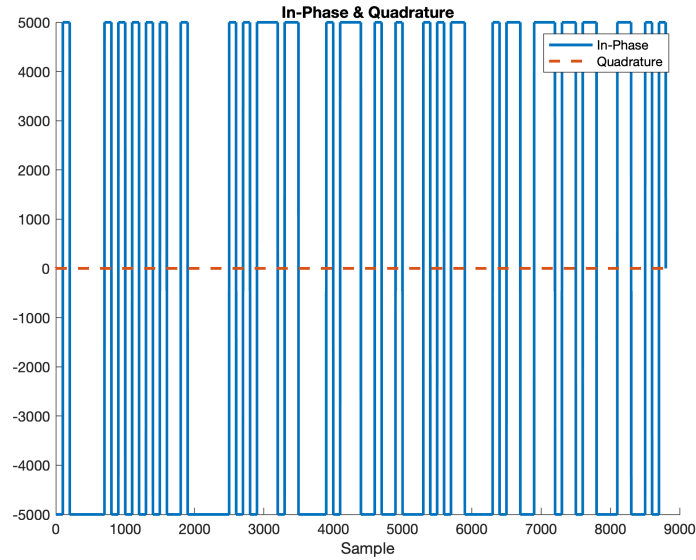


Figure 8: In-Phase & Quadrature

Problem IV

Develop a simple DLL to phase align the sequence shown below with the digital signal (generate_signal(3)) provided on the website. The sequence is sampled at 10 samples per chip. Plot the delay vs. time (or frequency vs. time depending on your implementation).

Solution

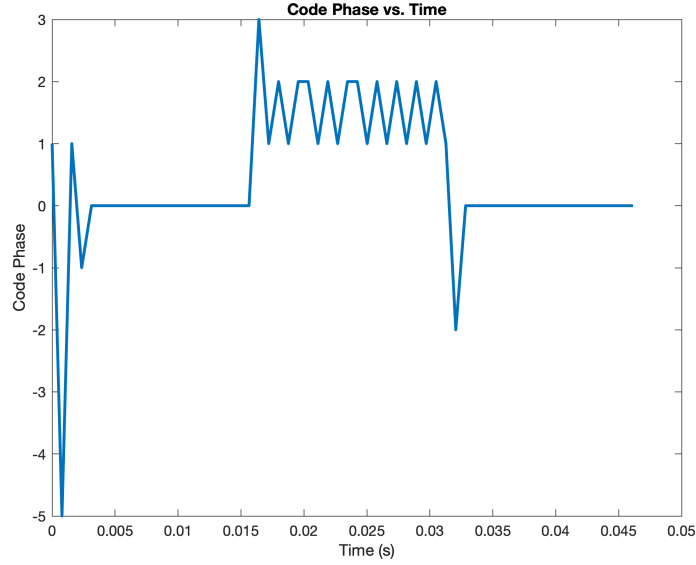


Figure 9: Code Phase vs. Time

Problem VI

Take the PRN code for SV #4 or #7 (i.e. from HW #3) and upsample it such that there are 16 samples at each chip (i.e. the length of this vector will be 1023×16 long).

Part A

Show the autocorrelation calculation from -5 chips to +5 chips in $1/16$ chip increments.

Solution

Combined Plot shown in Part B

Part B

Repeat with noise ($\sigma = 0.2$) added to the non-shifted signal.
Recall that the autocorrelation function can be calculated as:

$$R(\tau) = x^T x(\tau) = \sum_{k=0}^{16 \times 1023} x(k)x(k + \tau)$$

Note that you must roll $x(\tau)$ around when shifting it, i.e. $x(16 \times 1023 + 10) = x(11)$

You will need code that does the upsample and code shift for Lab #4 (as well as the problem below).

Solution

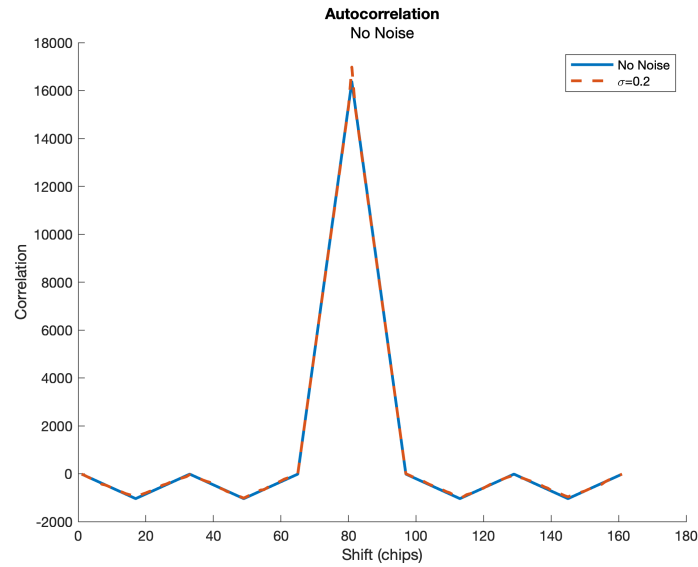


Figure 10: Autocorrelation Comparison

Problem VII

Write your own acquisition software to acquire a single satellite from the IFEN IF data file. You can write a serial or parallel search algorithm. Provide a plot of the acquisition plane (Code and Doppler) and provide the code phase and Doppler results. The C/A (Gold) codes for several satellites in view are on the website.

Solution

Parallel acquisition was used. The code shift is 20 chips and the doppler frequency is 990 Hz on PRN #7

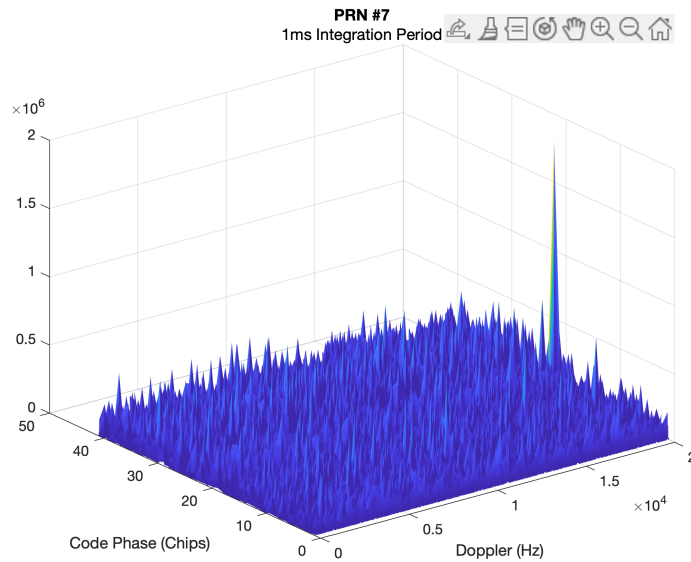


Figure 11: PRN #7 Acquisition

Appendix A: Part I Code

```

1 clear; close all; clc;
2
3 currentFile = mfilename('fullpath');
4 currentFolder = fileparts(currentFile);
5 addpath(genpath(currentFolder + "/.."));
6
7 time = 0:0.01:10;
8
9 s = tf('s');
10 plant = 1/s;
11 plant_eigs = pole(plant);
12 [plant_GM, plant_PM] = margin(plant);
13 [Y,~] = step(plant, time);
14 uc_ess = 1 - Y(end);
15
16 figure('Renderer', 'painters', 'Position', [10 10 900
17     600])
18 tiledlayout(2,2);
19 nexttile();
20 hold('on');
21 plot(time, Y, 'LineWidth', 2);
22 title('Uncontrolled Step Response');

```

```

22 xlabel('Time (s)');
23 ylabel('State');
24 ax = gca;
25 ax.FontSize = 18;
26
27 % (A) Proportional Controller
28 Kp = 2;
29 Pol = Kp*plant;
30 Pcl = Pol / (1 + Pol);
31 Pcl_eigs = pole(Pcl);
32 [P_GM, P_PM] = margin(Pol);
33 [YP,~] = step(Pcl, time);
34 P_ess = 1 - YP(end);
35
36 fprintf('A) Gain Margin: %0.4g\n', P_GM);
37 fprintf('A) Phase Margin: %0.4g\n', P_PM);
38 fprintf('A) EigenValues: %0.4g & %0.4g\n', Pcl_eigs);
39 fprintf('A) SS Error: %0.4g\n', P_ess);
40
41 nexttile();
42 hold('on');
43 plot(time, YP, 'LineWidth', 2);
44 title('Proportional Control Step Response');
45 xlabel('Time (s)');
46 ylabel('State');
47 ax = gca;
48 ax.FontSize = 18;
49
50 % (B)
51 [YP_ramp,~] = step(Pcl / s, time);
52 [Y_ramp,~] = step(1/s,time);
53
54 YP_ramp_ess = Y_ramp(end) - YP_ramp(end);
55 fprintf('B) SS Error: %0.4g\n', YP_ramp_ess);
56
57 nexttile();
58 hold('on');
59 plot(time, YP_ramp, 'LineWidth', 2);
60 plot(time, Y_ramp, '--', 'LineWidth', 2);
61 title('Proportional Control Ramp Response');
62 xlabel('Time (s)');
63 ylabel('State');
64 legend('System', 'R(t)');
65 ax = gca;
66 ax.FontSize = 18;
67

```

```

68 % (C)
69 Ki = 1;
70 PIol = ((Kp*s + Ki)/s)*plant;
71 PIcl = PIol / (1 + PIol);
72 PIcl_eigs = pole(PIcl);
73 [PI_GM, PI_PM] = margin(Pol);
74 [YPI_ramp,~] = step(PIcl / s, time);
75 PI_ramp_ess = Y_ramp(end) - YPI_ramp(end);
76
77
78 fprintf('C) Gain Margin: %0.4g\n', PI_GM);
79 fprintf('C) Phase Margin: %0.4g\n', PI_PM);
80 fprintf('C) EigenValues: %0.4g & %0.4g\n', PIcl_eigs);
81 fprintf('C) SS Error: %0.4g\n', PI_ramp_ess);
82
83 nexttile();
84 hold('on');
85 plot(time, YPI_ramp, 'LineWidth', 2);
86 plot(time, Y_ramp, '--', 'LineWidth', 2);
87 title('PI Control Ramp Response');
88 xlabel('Time (s)');
89 ylabel('State');
90 legend('System', 'R(t)');
91 ax = gca;
92 ax.FontSize = 18;
93
94 exportgraphics(gcf, currentFolder + '/../figures/p1.
  png', 'Resolution', 300);

```

Appendix B: Part II Code

```

1 clear; close all; clc;
2
3 currentFile = mfilename('fullpath');
4 currentFolder = fileparts(currentFile);
5 addpath(genpath(currentFolder + '/../'));
6
7 signal = generate_signal(1);
8 Ts = 1e-6;
9 Tint = 0.01;
10 time = 0:Ts:Ts*(length(signal)-2);
11 sampledTime = 0:Tint/Ts:length(time)-1;
12 intTime = 0:Ts:Tint;
13

```

```

14 % (A)
15 [freq1, phase1, err1] = costas(signal, 1/Ts, Tint);
16
17 figure();
18 hold('on');
19 plot(sampledTime, rad2deg(phase1), 'LineWidth', 2);
20 plot(sampledTime, rad2deg(err1), '--', 'LineWidth', 2)
21 ;
22 title('Phase & Error vs. Time');
23 subtitle('Bandwidth: 1Hz');
24 xlabel('Time (s)');
25 ylabel('Phase/Error (rad)');
26 legend('Phase', 'Phase Error');
27 ax = gca;
28 ax.FontSize = 18;
29
30 exportgraphics(gcf, currentFolder + '/../figures/
31 p2a_phase.png', 'Resolution', 300);
32
33 figure();
34 hold('on');
35 plot(length(time)*Ts - Tint*Ts + intTime, signal(
36     length(signal)-Tint/Ts:end), 'LineWidth', 2);
37 plot(length(time)*Ts - Tint*Ts + intTime, sin(2.*pi.*
38     freq1(end).*intTime + phase1(end)), '--', '
39     LineWidth', 2);
40 title('Signal & Replica vs. Time');
41 subtitle('Bandwidth: 1Hz');
42 xlabel('Time (s)');
43 ylabel('Signal (rad)');
44 legend('Signal', 'Replica');
45 ax = gca;
46 ax.FontSize = 18;
47
48 exportgraphics(gcf, currentFolder + '/../figures/
49 p2a_signal.png', 'Resolution', 300);
50
51 % (B)
52 [freq2, phase2, err2] = costas(signal, 1/Ts, Tint, 2);
53
54 figure();
55 hold('on');
56 plot(sampledTime, rad2deg(phase2), 'LineWidth', 2);
57 plot(sampledTime, rad2deg(err2), '--', 'LineWidth', 2)
58 ;
59 title('Phase & Error vs. Time');

```

```

53 subtitle('Bandwidth: 2Hz');
54 xlabel('Time (s)');
55 ylabel('Phase/Error (rad)')
56 legend('Phase', 'Phase Error');
57 ax = gca;
58 ax.FontSize = 18;
59
60 exportgraphics(gcf, currentFolder + '/../figures/
    p2b_phase.png', 'Resolution', 300);
61
62 figure();
63 hold('on');
64 plot(length(time)*Ts - Tint*Ts + intTime, signal(
    length(signal)-Tint/Ts:end), 'LineWidth', 2);
65 plot(length(time)*Ts - Tint*Ts + intTime, sin(2.*pi.*
    freq2(end).*intTime + phase2(end)), '--', '
    LineWidth', 2);
66 title('Signal & Replica vs. Time');
67 subtitle('Bandwidth: 2Hz');
68 xlabel('Time (s)');
69 ylabel('Signal (rad)');
70 legend('Signal', 'Replica');
71 ax = gca;
72 ax.FontSize = 18;
73
74 exportgraphics(gcf, currentFolder + '/../figures/
    p2b_signal.png', 'Resolution', 300);
75
76 % (D)
77 [freq10, phase10, err10] = costas(signal, 1/Ts, Tint,
    2, 10);
78
79 figure();
80 hold('on');
81 plot(sampledTime, rad2deg(phase10), 'LineWidth', 2);
82 plot(sampledTime, rad2deg(err10), '--', 'LineWidth',
    2);
83 title('Phase & Error vs. Time');
84 subtitle('Bandwidth: 2Hz');
85 xlabel('Time (s)');
86 ylabel('Phase/Error (rad)')
87 legend('Phase', 'Phase Error');
88 ax = gca;
89 ax.FontSize = 18;
90

```

```

91 exportgraphics(gcf, currentFolder + '/../figures/
    p2d_phase.png', 'Resolution', 300);
92
93 figure();
94 hold('on');
95 plot(length(time)*Ts - Tint*Ts + intTime, signal(
    length(signal)-Tint/Ts:end), 'LineWidth', 2);
96 plot(length(time)*Ts - Tint*Ts + intTime, sin(2.*pi.*
    freq10(end).*intTime + phase10(end)), '--', '
    LineWidth', 2);
97 title('Signal & Replica vs. Time');
98 subtitle('Initial Frequency 10Hz');
99 xlabel('Time (s)');
100 ylabel('Signal (rad)');
101 legend('Signal', 'Replica');
102 ax = gca;
103 ax.FontSize = 18;
104
105 exportgraphics(gcf, currentFolder + '/../figures/
    p2d_signal.png', 'Resolution', 300);
106
107 %% FUNCTIONS
108 function [freq, phase, err] = costas(signal, Fs, Tint,
    bandwidth, f0, phi0)
109     if ~exist('Tint', 'var'); Tint = 10e-3; end
110     if ~exist('bandwidth', 'var'); bandwidth = 1; end
111     if ~exist('f0', 'var'); f0 = 100; end
112     if ~exist('phi0', 'var'); phi0 = 0; end
113
114     wn = 2*pi*bandwidth;
115     zeta = 0.9;
116     Ki = wn^2;
117     Kp = 2*zeta*wn;
118
119     Ts = 1/Fs;
120     t = 0:Ts:(Tint-Ts);
121     time = 0:Ts:Ts*(length(signal)-2);
122     N = length(time);
123     M = Tint/Ts;
124
125     freq = f0.*ones(1,N/M);
126     phase = phi0.*ones(1,N/M);
127     err = zeros(1,N/M);
128     ierr = zeros(1,N/M);
129     for k = 1:N/M-1
130         X = signal(1 + (k-1)*M : k*M);

```

```

131         replicaI = sin(2.*pi.*freq(k).*t + phase(k));
132         replicaQ = cos(2.*pi.*freq(k).*t + phase(k));
133         I = X.*replicaI;
134         Q = X.*replicaQ;
135         err(k+1) = atan2(sum(Q), sum(I));
136         ierr(k+1) = ierr(k) + err(k+1)*Tint;
137         freq(k+1) = freq(k) + (Kp*err(k+1) + Ki*ierr(k
            +1));
138         phase(k+1) = rem(2.*pi*freq(k+1)*t(end) +
            phase(k), 2*pi);
139     end
140 end

```

Appendix C: Part III Code

```

1  clear; close all; clc;
2
3  currentFile = mfilename('fullpath');
4  currentFolder = fileparts(currentFile);
5  addpath(genpath(currentFolder + "/.."));
6
7  signal = generate_signal(2);
8  Ts = 1e-6;
9  Tint = 0.01;
10 t = 0:Ts:(Tint-Ts);
11 time = 0:Ts:Ts*(length(signal)-2);
12
13 [sumI, sumQ] = costas(signal, 1/Ts, Tint, 2);
14
15 figure();
16 hold on;
17 plot(sumI, 'LineWidth', 2);
18 plot(sumQ, '--', 'LineWidth', 2);
19 title('In-Phase & Quadrature');
20 xlabel('Sample');
21 legend('In-Phase', 'Quadrature');
22
23 exportgraphics(gcf, currentFolder + '/../figures/p3_IQ
    .png', 'Resolution', 300);
24
25 D = (1 + sign(sumI))./2;
26 D1 = D(1:1/Tint:end);
27 D2 = reshape(D1, 8, [])';
28 decode = bin2dec(num2str(D2));

```



```

29 fprintf('The message reads: %s\n', decode);
30
31 function [sumI, sumQ] = costas(signal, Fs, Tint,
    bandwidth, f0, phi0)
32     if ~exist('Tint', 'var'); Tint = 10e-3; end
33     if ~exist('bandwidth', 'var'); bandwidth = 1; end
34     if ~exist('f0', 'var'); f0 = 100; end
35     if ~exist('phi0', 'var'); phi0 = 0; end
36
37     wn = 2*pi*bandwidth;
38     zeta = 0.9;
39     Ki = wn^2;
40     Kp = 2*zeta*wn;
41
42     Ts = 1/Fs;
43     t = 0:Ts:(Tint-Ts);
44     time = 0:Ts:Ts*(length(signal)-2);
45     N = length(time);
46     M = Tint/Ts;
47
48     freq = f0.*ones(1,round(N/M));
49     phase = phi0.*ones(1,round(N/M));
50     err = zeros(1,round(N/M));
51     ierr = zeros(1,round(N/M));
52     sumI = zeros(1,round(N/M));
53     sumQ = zeros(1,round(N/M));
54     for k = 1:N/M-1
55         X = signal(1 + (k-1)*M : k*M);
56         replicaI = sin(2.*pi.*freq(k).*t + phase(k));
57         replicaQ = cos(2.*pi.*freq(k).*t + phase(k));
58         I = X.*replicaI;
59         Q = X.*replicaQ;
60         sumI(k) = sum(I);
61         sumQ(k) = sum(Q);
62         err(k+1) = atan(sumQ(k)/sumI(k));
63         ierr(k+1) = ierr(k) + err(k+1)*Tint;
64         freq(k+1) = freq(k) + (Kp*err(k+1) + Ki*ierr(k
            +1));
65         phase(k+1) = rem(2.*pi*freq(k+1)*t(end) +
            phase(k), 2*pi);
66     end
67 end

```

Appendix D: Part IV Code

```
1 clear; close all; clc;
2
3 currentFile = mfilename('fullpath');
4 currentFolder = fileparts(currentFile);
5 addpath(genpath(currentFolder + "/.."));
6
7 signal = generate_signal(3);
8 prn = [1 -1 -1 -1 1 -1 1 1];
9 spc = 10;
10 Tchip = 1e-3/1023;
11 Ts = Tchip*spc;
12 time = 0:Ts:Ts*(length(signal)-2);
13
14 wn = 0.5*2*pi;
15 zeta = 0.9;
16 Ki = wn^2;
17 Kp = 2*zeta*wn;
18
19 N = length(signal);
20 prnUp = upsample(prn, spc*length(prn));
21
22 M = length(prnUp);
23 shift = -M:M;
24 s = 0.5*spc;
25 Tint = M/spc*Tchip;
26
27 tau = ones(1,N/M);
28 err = zeros(1,N/M);
29 ierr = zeros(1,N/M);
30
31 figure();
32 for k = 1:N/M-1
33     clf;
34     hold('on');
35
36     X = signal(1 + (k-1)*M : k*M);
37     R = scorr(X, prnUp);
38     idx = find(tau(k)==shift);
39     RE = R(idx+s);
40     RL = R(idx-s);
41     err(k+1) = (RE - RL)/(RE + RL);
42     ierr(k+1) = ierr(k) + err(k+1)*Tint;
```

```

43     tau(k+1) = tau(k) + round(Kp*err(k+1) + Ki*ierr(k
        +1));
44     prnUp = circshift(prnUp, tau(k+1));
45
46     plot(X);
47     plot(prnUp);
48     pause(0.1);
49 end
50
51 figure();
52 plot(time(1:M:end), tau, 'LineWidth', 2);
53 title('Code Phase vs. Time');
54 xlabel('Time (s)');
55 ylabel('Code Phase');
56
57 exportgraphics(gcf, currentFolder + '/../figures/
    p4_tau.png', 'Resolution', 300);
58
59 %% FUNCTIONS
60
61 function [CE, CP, CL] = shift_code(code, shift, spc,
    spacing)
62 %shift_code.m Shift C/A Code
63 % Shifts the C/A code by a specified amount
64 % Inputs:
65 %     code      : PRN Code (already upsampled)
66 %     shift     : shift to the code (in terms of chips
        )
67 %     spc       : samples per chip
68 %     spacing   : I don't remember
69 %
70 % Outputs:
71 %     CE       : C/A Early
72 %     CP       : C/A Prompt
73 %     CL       : C/A Late
74 %
75 % Author: Walter Livingston
76
77     if ~exist('spc', 'var')
78         spc = 1;
79     end
80     if ~exist('spacing', 'var')
81         spacing = 1/2;
82     end
83     mult = round(spacing*spc);
84     tau = mult*shift;

```

```

85     CE = circshift(code, tau);
86     CP = code;
87     CL = circshift(code, -tau);
88 end

```

Appendix E: Part VI Code

```

1  clear; close all; clc;
2
3  currentFile = mfilename('fullpath');
4  currentFolder = fileparts(currentFile);
5  addpath(genpath(currentFolder + "/.."));
6
7  % (A)
8  spc = 16;
9
10 prn4 = genCA(4);
11 prn4(prn4 == 0) = -1;
12 prn4 = -prn4;
13 prn4up = upsample(prn4, spc*1023);
14
15 tauRange = -5:(1/16):5;
16 idx = 1;
17 for shift = tauRange
18     R(idx) = sum(prn4up.*circshift(prn4up, shift*spc))
19     ;
20     idx = idx + 1;
21 end
22 prn4_noisy = prn4up + 0.2*randn(1,spc*1023);
23
24 idx = 1;
25 for shift = tauRange
26     R_noisy(idx) = sum(prn4_noisy.*circshift(
27         prn4_noisy, shift*spc));
28     idx = idx + 1;
29 end
30 figure();
31 hold('on');
32 plot(R, 'LineWidth', 2);
33 plot(R_noisy, '--', 'LineWidth', 2);
34 title('Autocorrelation');
35 subtitle('No Noise');

```

```

36 xlabel('Shift (chips)');
37 ylabel('Correlation');
38 legend('No Noise', '\sigma=0.2');
39
40 exportgraphics(gcf, currentFolder + '/../figures/
    p6_corr.png', 'Resolution', 300);

```

Appendix C: Part VII Code

```

1  clear; close all; clc;
2
3  currentFile = mfilename('fullpath');
4  currentFolder = fileparts(currentFile);
5  addpath(genpath(currentFolder + '/../'));
6
7  [signalData, samplesRead] = ifen_parser('
    gpsBase_IFEN_IF.bin');
8  fint = 5000445.88565834;
9  prn = genCA(7);
10 pnr7(prn==0) = -1;
11 prn = -prn;
12
13 Tint = 0.001;
14 Tchip = (1e-3)/1023;
15 Fs = 20e6;
16 Ts = 1/Fs;
17 t = 0:Ts:(Tint-Ts);
18 spc = Ts/Tchip;
19
20 M = round(Tint/Ts);
21
22 dopRange = -10e3:500:10e3;
23 tauRange = 1:1023;
24
25 prnUp = upsample(prn, M);
26 batch = signalData(1:M)';
27 idx = 1;
28 for dop = dopRange
29     replicaI = sin(2*pi*(fint + dop)*t);
30     replicaQ = cos(2*pi*(fint + dop)*t);
31     I = batch.*replicaI;
32     Q = batch.*replicaQ;
33     result = I + Q;
34     fft_result = fft(result);

```

```

35     fft_prn = fft(prnUp);
36     conj_prn = conj(fft_prn);
37     combined = fft_result.*conj_prn;
38     corr(idx,:) = abs(ifft(combined)).^2;
39     idx = idx+1;
40 end
41
42 figure();
43 surf(corr, 'EdgeColor', 'none');
44 title('PRN #7');
45 subtitle('1ms Integration Period');
46 xlabel('Doppler (Hz)');
47 ylabel('Code Phase (Chips)');
48
49 exportgraphics(gcf, currentFolder + '/../figures/
    p7_prn7.png', 'Resolution', 300);

```