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

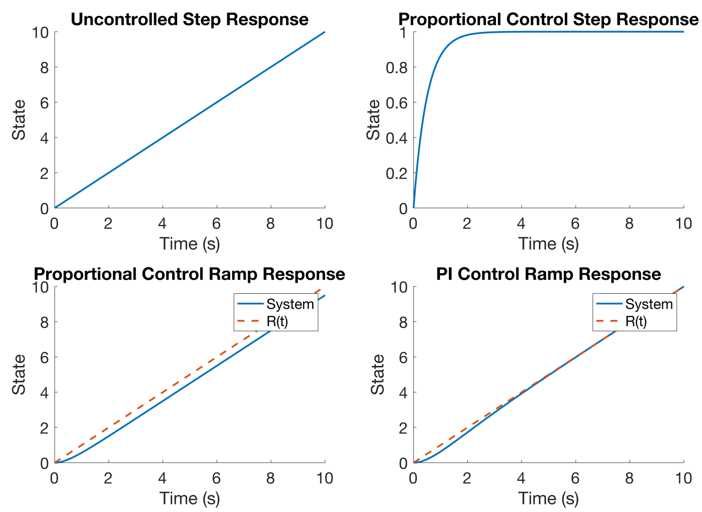

Figure 1: Problem 1 Step & Ramp Responses

Problem II

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

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

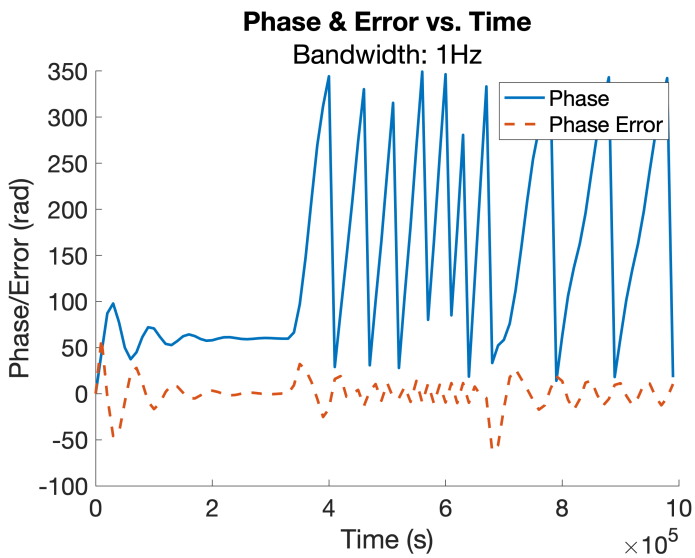

Figure 2: Phase & Error vs. Time for 1Hz Bandwidth

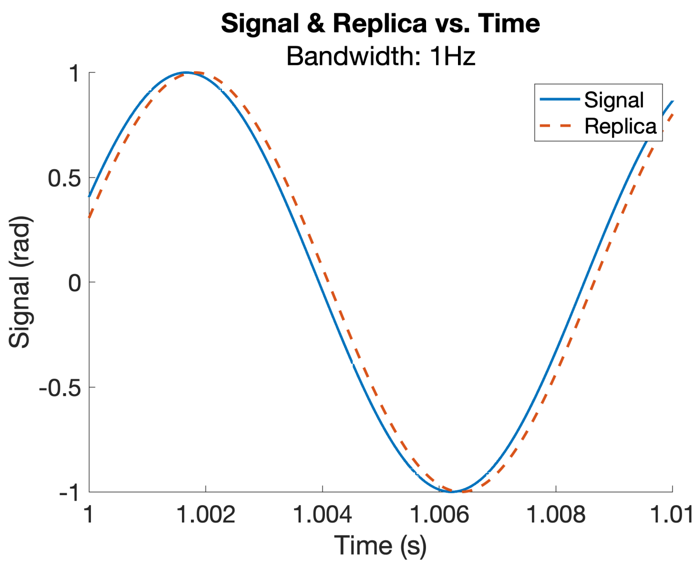

Figure 3: Signal & Replica vs. Time for 1Hz Bandwidth

Part B

Double the PLL bandwidth and repeat part a.

Solution

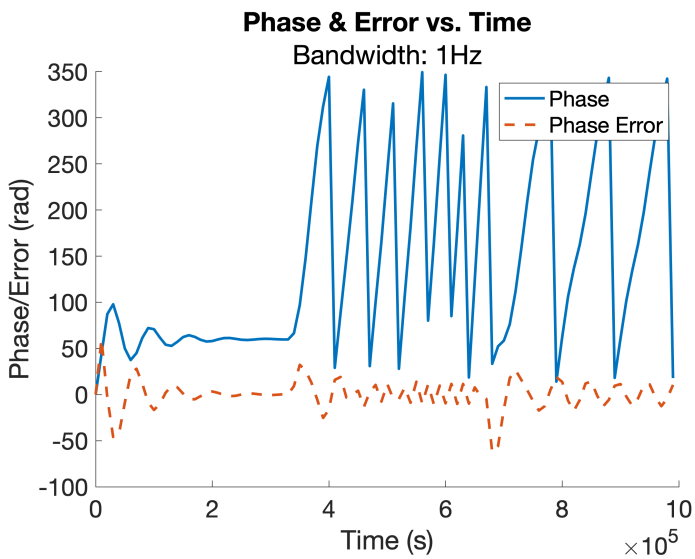

Figure 4: Phase & Error vs. Time for 2Hz Bandwidth

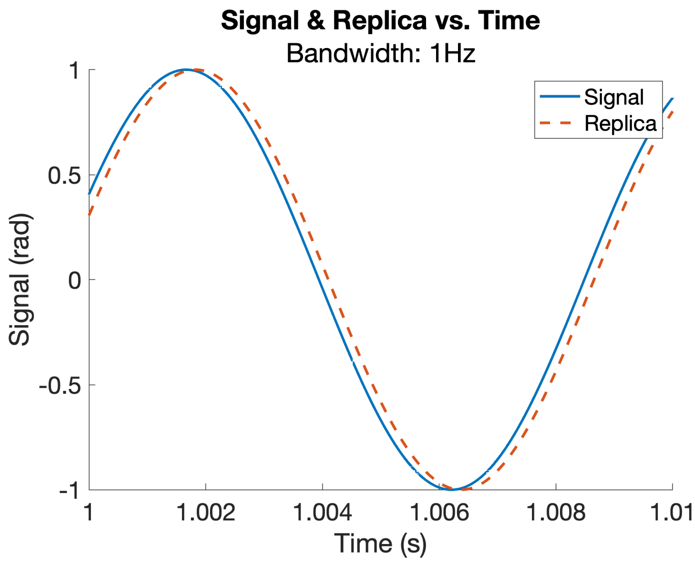

Figure 5: Signal & Replica vs. Time for 2Hz Bandwidth

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

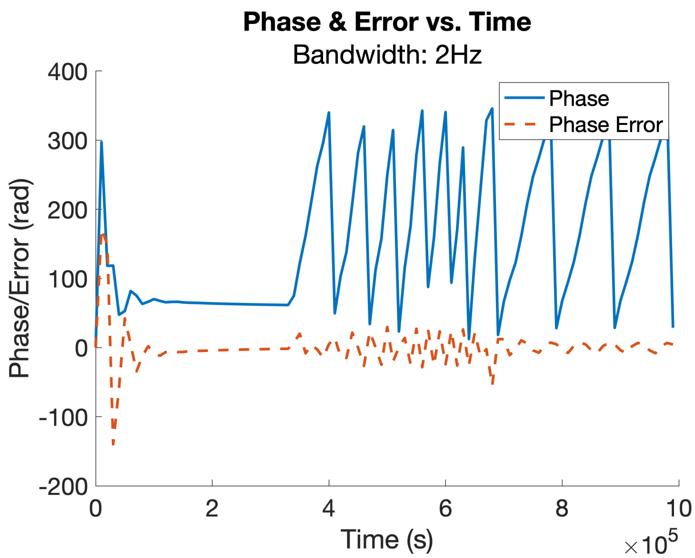

Figure 6: Phase & Error vs. Time for 10Hz Initial Frequency

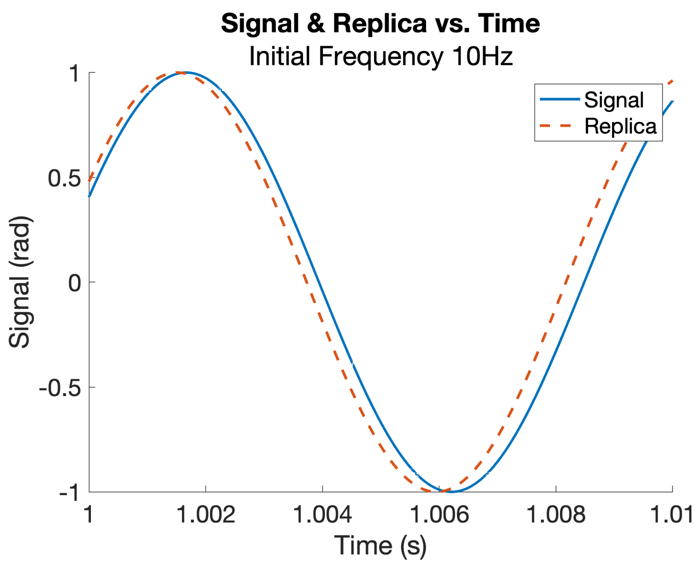

Figure 7: Signal & Replica vs. Time for 10Hz Initial Frequency

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.

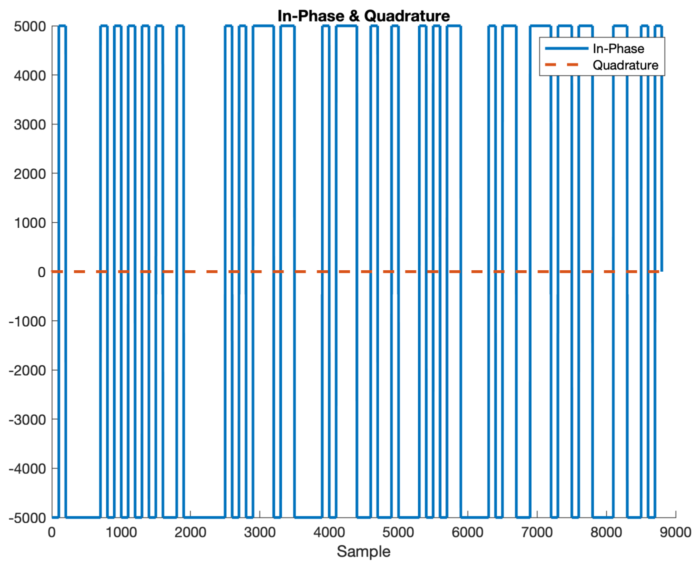

Figure 8: In-Phase & Quadrature

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

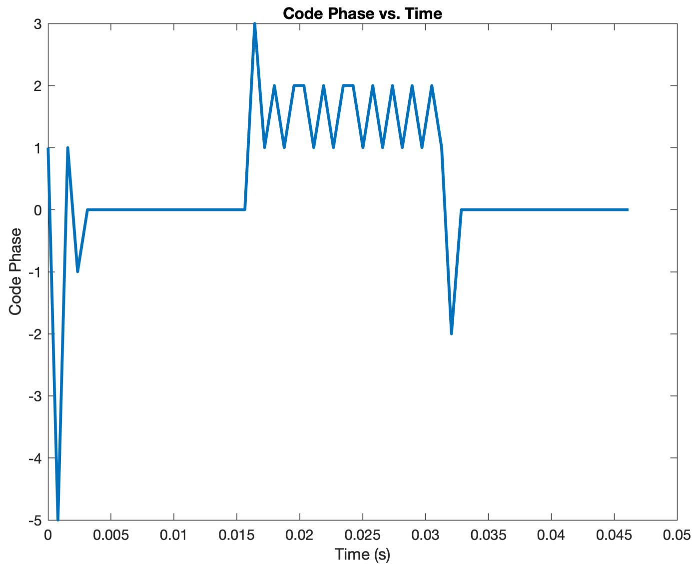

Figure 9: Code Phase vs. Time

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:

$$R(\tau) = x^T x(\tau) = \sum{}_{\_}1^{16*1023} x(k)x(k+\tau)$$

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

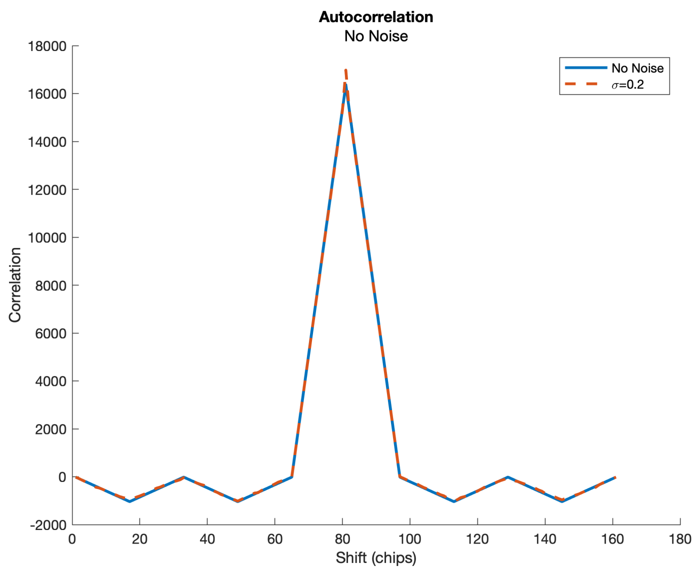

Figure 10: Autocorrelation Comparison

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

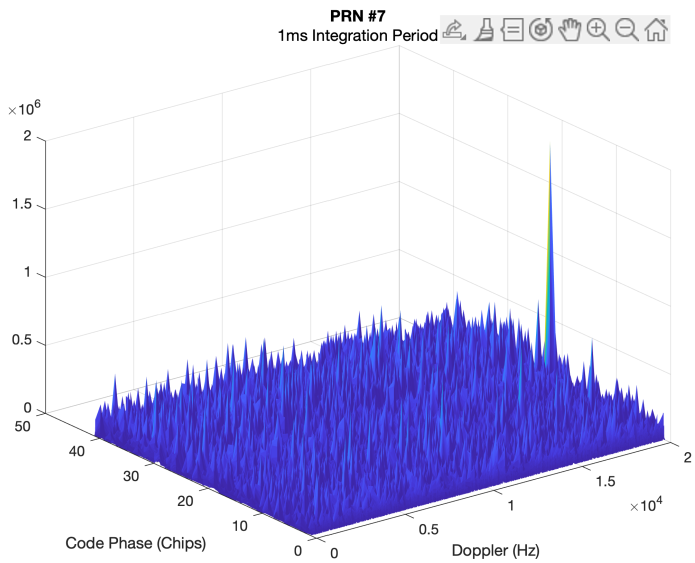

Figure 11: PRN #7 Acquisition

Appendix A: Part I Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/.."));

time = 0:0.01:10;

s = tf('s');
plant = 1/s;
plant_eigs = pole(plant);
[plant_GM, plant_PM] = margin(plant);
[Y,~] = step(plant, time);
uc_ess = 1 - Y(end);

figure('Renderer', 'painters', 'Position', [10 10 900
    600])
tiledlayout(2,2);
nexttile();
hold('on');
plot(time, Y, 'LineWidth', 2);
title('Uncontrolled Step Response');
```

```
xlabel('Time (s)');
ylabel('State');
ax = gca;
ax.FontSize = 18;

% (A) Proportional Controller
Kp = 2;
Pol = Kp*plant;
Pcl = Pol / (1 + Pol);
Pcl_eigs = pole(Pcl);
[P_GM, P_PM] = margin(Pol);
[YP,~] = step(Pcl, time);
P_ess = 1 - YP(end);

fprintf('A) Gain Margin: %0.4g\n', P_GM);
fprintf('A) Phase Margin: %0.4g\n', P_PM);
fprintf('A) EigenValues: %0.4g & %0.4g\n', Pcl_eigs);
fprintf('A) SS Error: %0.4g\n', P_ess);

nexttile();
hold('on');
plot(time, YP, 'LineWidth', 2);
title('Proportional Control Step Response');
xlabel('Time (s)');
ylabel('State');
ax = gca;
ax.FontSize = 18;

% (B)
[YP_ramp,~] = step(Pcl / s, time);
[Y_ramp,~] = step(1/s,time);

YP_ramp_ess = Y_ramp(end) - YP_ramp(end);
fprintf('B) SS Error: %0.4g\n', YP_ramp_ess);

nexttile();
hold('on');
plot(time, YP_ramp, 'LineWidth', 2);
plot(time, Y_ramp, '--', 'LineWidth', 2);
title('Proportional Control Ramp Response');
xlabel('Time (s)');
ylabel('State');
legend('System', 'R(t)');
ax = gca;
ax.FontSize = 18;

```

```
% (C)
Ki = 1;
PIol = ((Kp*s + Ki)/s)*plant;
PIcl = PIol / (1 + PIol);
PIcl_eigs = pole(PIcl);
[PI_GM, PI_PM] = margin(Pol);
[YPI_ramp,~] = step(PIcl / s, time);
PI_ramp_ess = Y_ramp(end) - YPI_ramp(end);


fprintf('C) Gain Margin: %0.4g\n', PI_GM);
fprintf('C) Phase Margin: %0.4g\n', PI_PM);
fprintf('C) EigenValues: %0.4g & %0.4g\n', PIcl_eigs);
fprintf('C) SS Error: %0.4g\n', PI_ramp_ess);

nexttile();
hold('on');
plot(time, YPI_ramp, 'LineWidth', 2);
plot(time, Y_ramp, '--', 'LineWidth', 2);
title('PI Control Ramp Response');
xlabel('Time (s)');
ylabel('State');
legend('System', 'R(t)');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p1.
    png", 'Resolution', 300);
```

Appendix B: Part II Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/.."));

signal = generate_signal(1);
Ts = 1e-6;
Tint = 0.01;
time = 0:Ts:Ts*(length(signal)-2);
sampledTime = 0:Tint/Ts:length(time)-1;
intTime = 0:Ts:Tint;

```

```
% (A)
[freq1, phase1, err1] = costas(signal, 1/Ts, Tint);

figure();
hold('on');
plot(sampledTime, rad2deg(phase1), 'LineWidth', 2);
plot(sampledTime, rad2deg(err1), '--', 'LineWidth', 2);
title('Phase & Error vs. Time');
subtitle('Bandwidth: 1Hz');
xlabel('Time (s)');
ylabel('Phase/Error (rad)')
legend('Phase', 'Phase Error');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p2a_phase.png", 'Resolution', 300);

figure();
hold('on');
plot(length(time)*Ts - Tint*Ts + intTime, signal(length(signal)-Tint/Ts:end), 'LineWidth', 2);
plot(length(time)*Ts - Tint*Ts + intTime, sin(2.*pi.*freq1(end).*intTime + phase1(end)), '--', 'LineWidth', 2);
title('Signal & Replica vs. Time');
subtitle('Bandwidth: 1Hz');
xlabel('Time (s)');
ylabel('Signal (rad)');
legend('Signal', 'Replica');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p2a_signal.png", 'Resolution', 300);

% (B)
[freq2, phase2, err2] = costas(signal, 1/Ts, Tint, 2);

figure();
hold('on');
plot(sampledTime, rad2deg(phase2), 'LineWidth', 2);
plot(sampledTime, rad2deg(err2), '--', 'LineWidth', 2);
title('Phase & Error vs. Time');
```

```
subtitle('Bandwidth: 2Hz');
xlabel('Time (s)');
ylabel('Phase/Error (rad)')
legend('Phase', 'Phase Error');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/
    p2b_phase.png", 'Resolution', 300);

figure();
hold('on');
plot(length(time)*Ts - Tint*Ts + intTime, signal(
    length(signal)-Tint/Ts:end), 'LineWidth', 2);
plot(length(time)*Ts - Tint*Ts + intTime, sin(2.*pi.*
    freq2(end).*intTime + phase2(end)), '--', '
    LineWidth', 2);
title('Signal & Replica vs. Time');
subtitle('Bandwidth: 2Hz');
xlabel('Time (s)');
ylabel('Signal (rad)');
legend('Signal', 'Replica');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/
    p2b_signal.png", 'Resolution', 300);

% (D)
[freq10, phase10, err10] = costas(signal, 1/Ts, Tint,
    2, 10);

figure();
hold('on');
plot(sampledTime, rad2deg(phase10), 'LineWidth', 2);
plot(sampledTime, rad2deg(err10), '--', 'LineWidth',
    2);
title('Phase & Error vs. Time');
subtitle('Bandwidth: 2Hz');
xlabel('Time (s)');
ylabel('Phase/Error (rad)')
legend('Phase', 'Phase Error');
ax = gca;
ax.FontSize = 18;

```

```
exportgraphics(gcf, currentFolder + "/../figures/p2d_phase.png", 'Resolution', 300);

figure();
hold('on');
plot(length(time)*Ts - Tint*Ts + intTime, signal(length(signal)-Tint/Ts:end), 'LineWidth', 2);
plot(length(time)*Ts - Tint*Ts + intTime, sin(2.*pi.*freq10(end).*intTime + phase10(end)), '--', 'LineWidth', 2);
title('Signal & Replica vs. Time');
subtitle('Initial Frequency 10Hz');
xlabel('Time (s)');
ylabel('Signal (rad)');
legend('Signal', 'Replica');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p2d_signal.png", 'Resolution', 300);

%% FUNCTIONS
function [freq, phase, err] = costas(signal, Fs, Tint, bandwidth, f0, phi0)
    if ~exist('Tint', 'var'); Tint = 10e-3; end
    if ~exist('bandwidth', 'var'); bandwidth = 1; end
    if ~exist('f0', 'var'); f0 = 100; end
    if ~exist('phi0', 'var'); phi0 = 0; end

    wn = 2*pi*bandwidth;
    zeta = 0.9;
    Ki = wn^2;
    Kp = 2*zeta*wn;

    Ts = 1/Fs;
    t = 0:Ts:(Tint-Ts);
    time = 0:Ts:Ts*(length(signal)-2);
    N = length(time);
    M = Tint/Ts;

    freq = f0.*ones(1,N/M);
    phase = phi0.*ones(1,N/M);
    err = zeros(1,N/M);
    ierr = zeros(1,N/M);
    for k = 1:N/M-1
        X = signal(1  + (k-1)*M : k*M);
```

```
        replicaI = sin(2.*pi.*freq(k).*t + phase(k));
        replicaQ = cos(2.*pi.*freq(k).*t + phase(k));
        I = X.*replicaI;
        Q = X.*replicaQ;
        err(k+1) = atan2(sum(Q), sum(I));
        ierr(k+1) = ierr(k) + err(k+1)*Tint;
        freq(k+1) = freq(k) + (Kp*err(k+1) + Ki*ierr(k
            +1));
        phase(k+1) = rem(2.*pi*freq(k+1)*t(end) +
            phase(k), 2*pi);
    end
end
```

Appendix C: Part III Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/.."));

signal = generate_signal(2);
Ts = 1e-6;
Tint = 0.01;
t = 0:Ts:(Tint-Ts);
time = 0:Ts:Ts*(length(signal)-2);

[sumI, sumQ] = costas(signal, 1/Ts, Tint, 2);

figure();
hold on;
plot(sumI, 'LineWidth', 2);
plot(sumQ, '--', 'LineWidth', 2);
title('In-Phase & Quadrature');
xlabel('Sample');
legend('In-Phase', 'Quadrature');

exportgraphics(gcf, currentFolder + "/../figures/p3_IQ
    .png", 'Resolution', 300);

D = (1 + sign(sumI))./2;
D1 = D(1:1/Tint:end);
D2 = reshape(D1, 8, [])';
decode = bin2dec(num2str(D2));
```

```
fprintf('The message reads: %s\n', decode);

function [sumI, sumQ] = costas(signal, Fs, Tint, bandwidth, f0, phi0)
    if ~exist('Tint', 'var'); Tint = 10e-3; end
    if ~exist('bandwidth', 'var'); bandwidth = 1; end
    if ~exist('f0', 'var'); f0 = 100; end
    if ~exist('phi0', 'var'); phi0 = 0; end

    wn = 2*pi*bandwidth;
    zeta = 0.9;
    Ki = wn^2;
    Kp = 2*zeta*wn;

    Ts = 1/Fs;
    t = 0:Ts:(Tint-Ts);
    time = 0:Ts:Ts*(length(signal)-2);
    N = length(time);
    M = Tint/Ts;

    freq = f0.*ones(1,round(N/M));
    phase = phi0.*ones(1,round(N/M));
    err = zeros(1,round(N/M));
    ierr = zeros(1,round(N/M));
    sumI = zeros(1,round(N/M));
    sumQ = zeros(1,round(N/M));
    for k = 1:N/M-1
        X = signal(1  + (k-1)*M : k*M);
        replicaI = sin(2.*pi.*freq(k).*t + phase(k));
        replicaQ = cos(2.*pi.*freq(k).*t + phase(k));
        I = X.*replicaI;
        Q = X.*replicaQ;
        sumI(k) = sum(I);
        sumQ(k) = sum(Q);
        err(k+1) = atan(sumQ(k)/sumI(k));
        ierr(k+1) = ierr(k) + err(k+1)*Tint;
        freq(k+1) = freq(k) + (Kp*err(k+1) + Ki*ierr(k+1));
        phase(k+1) = rem(2.*pi*freq(k+1)*t(end) + phase(k), 2*pi);
    end
end
```

Appendix D: Part IV Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/.."));

signal = generate_signal(3);
prn = [1 -1 -1 -1 1 -1 1 1];
spc = 10;
Tchip = 1e-3/1023;
Ts = Tchip*spc;
time = 0:Ts:Ts*(length(signal)-2);

wn = 0.5*2*pi;
zeta = 0.9;
Ki = wn^2;
Kp = 2*zeta*wn;

N = length(signal);
prnUp = upsample(prn, spc*length(prn));

M = length(prnUp);
shift = -M:M;
s = 0.5*spc;
Tint = M/spc*Tchip;

tau = ones(1,N/M);
err = zeros(1,N/M);
ierr = zeros(1,N/M);

figure();
for k = 1:N/M-1
    clf;
    hold('on');

    X = signal(1 + (k-1)*M : k*M);
    R = scorr(X, prnUp);
    idx = find(tau(k)==shift);
    RE = R(idx+s);
    RL = R(idx-s);
    err(k+1) = (RE - RL)/(RE + RL);
    ierr(k+1) = ierr(k) + err(k+1)*Tint;
```

```
    tau(k+1) = tau(k) + round(Kp*err(k+1) + Ki*ierr(k+1));
    prnUp = circshift(prnUp, tau(k+1));

    plot(X);
    plot(prnUp);
    pause(0.1);
end

figure();
plot(time(1:M:end), tau, 'LineWidth', 2);
title('Code Phase vs. Time');
xlabel('Time (s)');
ylabel('Code Phase');

exportgraphics(gcf, currentFolder + "/../figures/p4_tau.png", 'Resolution', 300);

%% FUNCTIONS

function [CE, CP, CL] = shift_code(code, shift, spc, spacing)
%shift_code.m Shift C/A Code
%   Shifts the C/A code by a specified amount
% Inputs:
%     code       : PRN Code (already upsampled)
%     shift      : shift to the code (in terms of chips)
%     spc        : samples per chip
%     spacing    : I don't remember
%
% Outputs:
%     CE         : C/A Early
%     CP         : C/A Prompt
%     CL         : C/A Late
%
% Author: Walter Livingston

    if ~exist('spc', 'var')
        spc = 1;
    end
    if ~exist('spacing', 'var')
        spacing = 1/2;
    end
    mult = round(spacing*spc);
    tau = mult*shift;
```

```
    CE = circshift(code, tau);
    CP = code;
    CL = circshift(code, -tau);
end
```

Appendix E: Part VI Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/.."));

% (A)
spc = 16;

prn4 = genCA(4);
prn4(prn4 == 0) = -1;
prn4 = -prn4;
prn4up = upsample(prn4, spc*1023);

tauRange = -5:(1/16):5;
idx = 1;
for shift = tauRange
    R(idx) = sum(prn4up.*circshift(prn4up, shift*spc))
        ;
    idx = idx + 1;
end

prn4_noisy = prn4up + 0.2*randn(1,spc*1023);

idx = 1;
for shift = tauRange
    R_noisy(idx) = sum(prn4_noisy.*circshift(
        prn4_noisy, shift*spc));
    idx = idx + 1;
end

figure();
hold('on');
plot(R, 'LineWidth', 2);
plot(R_noisy, '--', 'LineWidth', 2);
title('Autocorrelation');
subtitle('No Noise');
```

```
xlabel('Shift (chips)');
ylabel('Correlation');
legend('No Noise', '\sigma=0.2');

exportgraphics(gcf, currentFolder + "/../figures/
    p6_corr.png", 'Resolution', 300);
```

Appendix C: Part VII Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/.."));

[signalData, samplesRead] = ifen_parser('
    gpsBase_IFEN_IF.bin');
fint = 5000445.88565834;
prn = genCA(7);
pnr7(prn==0) = -1;
prn = -prn;

Tint = 0.001;
Tchip = (1e-3)/1023;
Fs = 20e6;
Ts = 1/Fs;
t = 0:Ts:(Tint-Ts);
spc = Ts/Tchip;

M = round(Tint/Ts);

dopRange = -10e3:500:10e3;
tauRange = 1:1023;

prnUp = upsample(prn, M);
batch = signalData(1:M)';
idx = 1;
for dop = dopRange
    replicaI = sin(2*pi*(fint + dop)*t);
    replicaQ = cos(2*pi*(fint + dop)*t);
    I = batch.*replicaI;
    Q = batch.*replicaQ;
    result = I + Q;
    fft_result = fft(result);
```

```
    fft_prn = fft(prnUp);
    conj_prn = conj(fft_prn);
    combined = fft_result.*conj_prn;
    corr(idx,:) = abs(ifft(combined)).^2;
    idx = idx+1;
end

figure();
surf(corr, 'EdgeColor', 'none');
title('PRN #7');
subtitle('1ms Integration Period');
xlabel('Doppler (Hz)');
ylabel('Code Phase (Chips)');

exportgraphics(gcf, currentFolder + "/../figures/
   p7_prn7.png", 'Resolution', 300);
```