THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

DEPARTMENT OF LAND SURVEYING AND GEO-INFORMATICS
土地測量及地理資訊學系

The Hong Kong Polytechnic University

The Department of Land Surveying and Geo-informatics

2021-2022 Semester 1: LSGI3322 - Satellite Positioning Systems

Subject Lecturer: Prof. George Liu & Prof. Wu Chen

Final Individual Project: GPS Positioning (30%)

Final Report: Computation of GPS Receiver Location (15%)

Due Date: 12-12-2021 23:59

Student Name: Tang Justin Hayse Chi Wing G.

Student ID: 20016345D

Final Year Student in BSc (Hons) in Land Surveying and Geo-informatics

Expected Grade: A Range

Table of Contents

Abstract
The LSGI3322 final project aims to perform the single point positioning (SPP) using pseudorange measurement by our own designed computer program, to compute the GPS receiver position. It used MATLAB program language to perform the readings of both RINEX navigation and observation data files, computation of satellite orbit positions and GPS receiver position. The coordinate transformation from Earth-Centred Earth-Fixed (ECEF) coordinate to WGS84 geodetic datum are provided. The result showed the GPS receiver position was at a car park nearby Albert Head Road, at Metchosin, British Columbia, Canada (**48.38979° N, 123.48712° W**).

Keywords: Pseudorange, Satellite Orbit Position, Global Positioning System, RINEX, MATLAB, Single Point

1 Introduction

Satellite positioning techniques have been a pivot to numerous state-of-the-art applications, such as smart city, location-based services, autonomous vehicle and Internet-of-Things (Bui et al, 2020; Malik and Gupta, 2021). Satellite positioning using pseudorange measurement has been extensively studied in literature (Torres, 2017).

In the general practice, there are two methods to compute the receiver position, namely pseudorange and carrier phase measurements (Næsset, 1999). Yet, the accuracy level of them are varied. Pseudorange measurement can achieve around five to ten meter level, whereas carrier phase measurement is capable of achieving centimetre level, or even reaching millimetre level (Li et al, 2019; Chen's LSGI3322 Lecture 7, 2021). Thus, the method pseudorange measurement is regarded as an initial stage to know the approximate coordinate position. Afterwards, it is followed by carrier phase measurement, to determine receiver position in high accuracy level.

2 Pseudorange and Satellite Orbit Positioning

2.1 Introduction to Pseudorange Measurement

The foundation of the pseudorange measurement is code correlation, in which the codes are correlated the replica code in the GPS receivers shifted and being identical with satellite code (Chen et al, 2019). GPS receiver measures the time offset which is the time difference ($\Delta t_i^k = (t_i - t^k)$) between the coded signals, either C/A code or P code, transmitted from the satellite clock and the coded signals received by the receiver clock.

When the code and is received by the GPS receiver, a replica of satellite code is then generated inside the GPS receiver. Once the time shift (Δt) is computed, the pseudorange, a geometric distance measurement between orbit satellite and GPS receiver position, can directly calculated by the below formula (1):

$$\boldsymbol{P_i^k = c \cdot \Delta t_i^k = c \cdot (t_i - t^k)} \quad \text{..... (1)}$$

c = speed of the light = $3 \times 10^8 m/s$

t_i = time when the coded signal is transmitted from satellite clock t^k = time when the coded signal is received by the satellite clock

Considering the errors involved in the pseudorange measurement, in which consist of three main parts, namely errors related to satellite, receiver and propagation. Error terms related to satellite comprise satellite clock and ephemeris errors. Error terms related to GPS receiver contain measurement noise and receiver clock error. Finally, errors related to the propagation involve tropospheric and ionospheric delays. The below formula (2) indicates the pseudorange measurement including the error terms.

$$\boldsymbol{P_i^k = \rho + d_{orb} + c(dt_i - dt^k) + d_{ion} + d_{trop} + others} \quad \text{..... (2)}$$

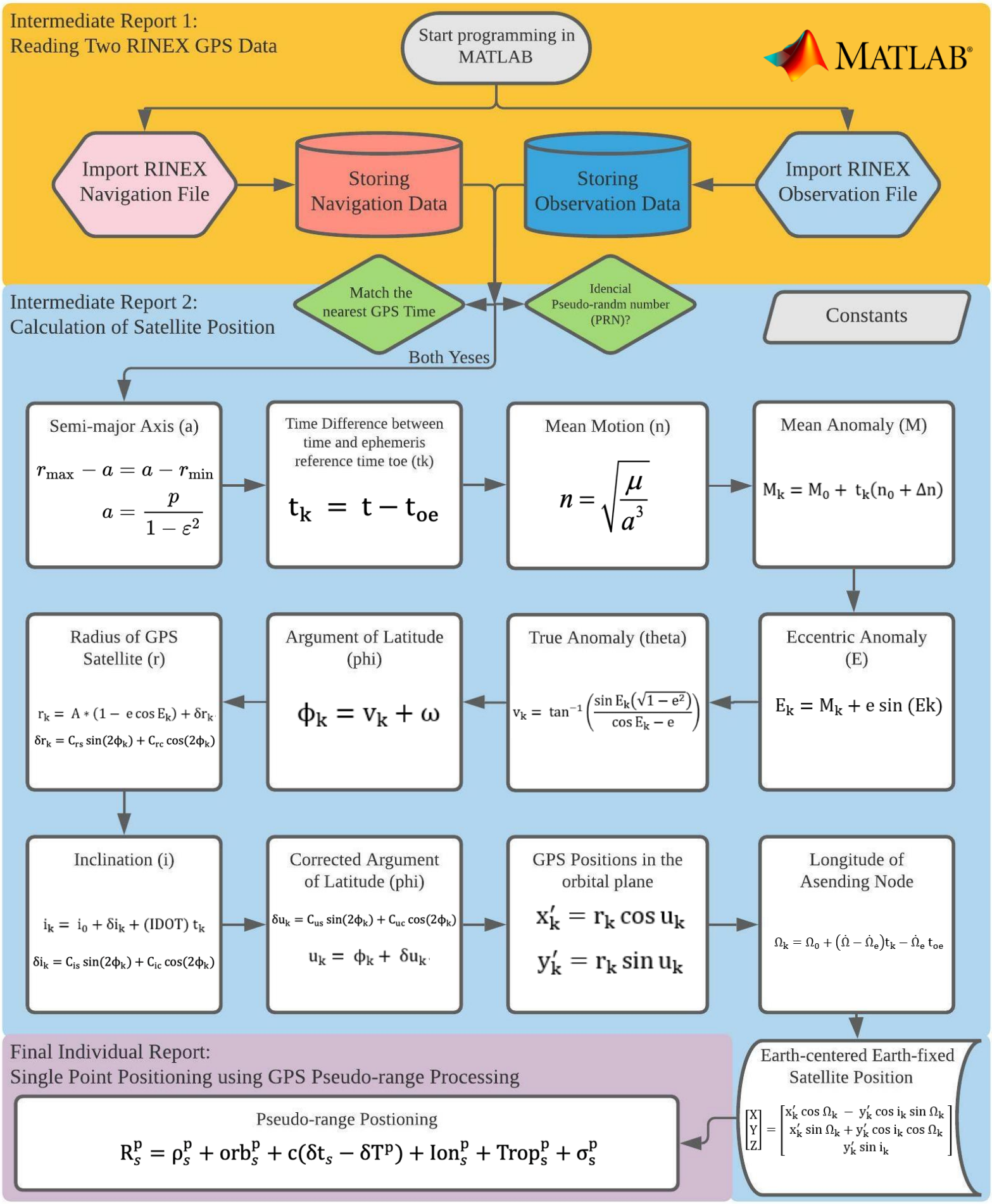
ρ: distance between satellite and receiver; c: speed of the light; dt_i: receiver clock error; dt^k: satellite clock error;

d_{ion}: ionospheric delay; d_{trop}: tropospheric delay; d_{orb}: satellite orbit error; $others$: multipath error and measurement noises

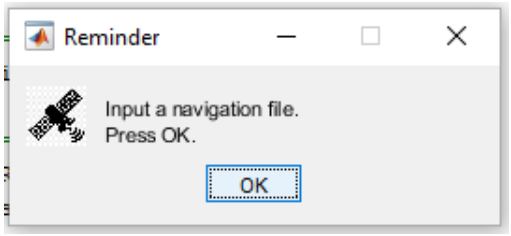
This assignment mainly divided into sections. The project begins with reading both RINEX files and computing orbit satellite positions. Then, computing GPS receiver position by linearization are performed. Additionally, the errors corrections, e.g., satellite clock error, ionospheric-free combination and tropospheric error are utilized.

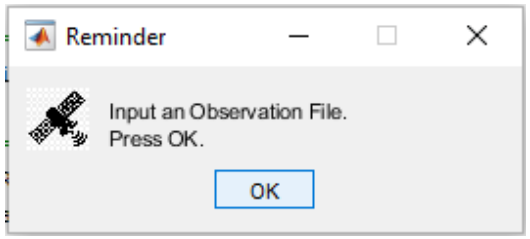
2.2 Framework of Satellite Orbit Positioning

The below framework displays the flow of reading both RINEX navigation and observation data storing in the MATLAB workplace (database) as well as the calculation of satellite positions. The detail of the computation steps and the corresponding formulas and MATLAB code, are well-tabulated as follows.



Workflow	Formula/ Constant	MATLAB Code
The Progress of Intermediate Report 1		

Step 1 - To create a program as user-friendly as possible, a pop-up window for selecting a RINEX nevigation file is made. The end-user can choose a navigation file and the file directory will automatically saved inside the MATLAB workplace.
 
 	
Figure 1. The pop-up window of selecting navigation file.
 	
Meanwhile, the program finds the header '**END OF HEADER**' in the nevigation file. The positioning information after header is used to perform the subsequent orbit and receiver positions. By using while loop, the emhemeris information is then saved in the structure array format in the workplace.
	N/A	%==============Read RINEX navigation File==========================
		[icondata,iconcmap] = imread('GPS3.png');
		z1=msgbox({'Input a navigation file.' ; 'Press OK.'},
		'Reminder','custom', icondata,iconcmap);
		uiwait(z1);
		[selNav, Nav_pathname] = uigetfile;
		Navigation_path = strcat(Nav_pathname,selNav);
		NavData = fopen(Navigation_path,'r');
		Read_NavLine = fgetl(NavData);
		while ischar(Read_NavLine)
		if contains(Read_NavLine, 'END OF HEADER')
		break;
		end
		Read_NavLine = fgetl(NavData);
		End
		
		No_Epoch_Nav = 1;
		while ischar(Read_NavLine)
		Read_NavLine = fgetl(NavData);
		if (Read_NavLine == -1)
		No_Epoch_Nav = No_Epoch_Nav - 1;
		break;
		end

Step 2 - Using the first row of the broadcast orbit as an example, these lines are used to find out the relevant information in the navigation file. For instance, recorder identifier, year, month, day, hour, minute, second, etc. Explanation of GPS Navigation Data Record 2 | 01 | 3 | 31 | 0 | 0 | 0 | -0.36158272088D-3 | -0.534328137292D-11 | 0.00000000000D+00 Recorder identifier | Year (2-digit) | Month | Day | Hour | Minute | Second (2 digits) | SV clock bias (seconds) | SV clock drift (sec/sec) | SV clock drift rate (sec/sec2)	N/A	%====================Broadcast Orbit - 1============================= Nav(Epoch_Nav).PRN=str2num(Read_NavLine(1:2)); Nav(Epoch_Nav).Year=str2num(Read_NavLine(4:5))+2000; Nav(Epoch_Nav).Month=str2num(Read_NavLine(7:8)); Nav(Epoch_Nav).Day=str2num(Read_NavLine(10:11)); Nav(Epoch_Nav).Hour=str2num(Read_NavLine(13:14)); Nav(Epoch_Nav).Minute=str2num(Read_NavLine(16:17)); Nav(Epoch_Nav).Second=str2num(Read_NavLine(18:22));
Step 3 - Similar to the Step 1, a pop-up window for selecting the observation file is also created.  Figure 2. The pop-up window of selecting observation file.	N/A	%====================Read RINEX Observation File==================== z2=msgbox({'Input an Observation File.' ; 'Press OK.'}, 'Reminder','custom',icondata,iconcmap); uiwait(z2); [selObs, Obs_pathname] = uigetfile; Observation_path = strcat(Obs_pathname,selObs); ObsData = fopen(Observation_path,'r'); fprintf('A pair of coordinate of GPS Receiver is now computing, Pls wait for a moment.\nThank you for your understanding.\n');

Step 4 - After saving the observation file to the file directory, use the while loop function to save observation data into MATLAB workplace (database).	N/A	`%======================Read Observation Data=========================`
		`while ischar(Read_ObsLine)`
		`Read_ObsLine = fgetl(ObsData);`
		`if (Read_ObsLine == -1)`
		`break;`
		`end`

Step 5 - This step is about the temporal condition regarding the observation file. Firstly, find out the observation time in the observation file through the first row again. After that, also count the total number and time used of satellites that were connected.
 	
In particular, considering the guideline of this project requires the program must read one epoch's GPS data from observation data file, C1 is selected. The epoch C1 includes the GPS time, satellite PRN information and pseudorange information.
	N/A	`%======================Find the Observation Time=======================`
		`if isequal(string(Read_ObsLine(2:3)),Obs_Time_of_FirstObs)`
		`No_of_PRN = str2num(Read_ObsLine(31:32));`
		`%================Find the total Number of Satellites=================`
		`for h = 1: No_of_PRN`
		`Obs(Epoch_OBS + h).PRN=str2num(Read_ObsLine(31+3*h:32+3*h));`
		`Obs(Epoch_OBS + h).Year=str2num(Read_ObsLine(1:3))+2000;`
		`Obs(Epoch_OBS + h).Month=str2num(Read_ObsLine(5:6));`
		`Obs(Epoch_OBS + h).Day=str2num(Read_ObsLine(7:9));`
		`Obs(Epoch_OBS + h).Hour=str2num(Read_ObsLine(11:12));`
		`Obs(Epoch_OBS + h).Minute=str2num(Read_ObsLine(14:15));`
		`Obs(Epoch_OBS + h).Second=str2num(Read_ObsLine(17:26));`
		`end`
		`%======================Find the C1 Observation Data=================`
		`for d = 1:No_of_PRN`
		`Obs(Epoch_OBS + d).C1 = str2num(Read_ObsLine(2+16*(IndexNo_C-1):15+16*(IndexNo_C-1)));`
		`end`

Step 6 - This is a critical step to first find out whether the PRNs from both RINEX navigation and observation data files are the same. Concerning the orbit satellites marked in the navigation file are not all are as well-tracked as the observation file, thus there are two prerequisites that must be satisfied.
 	
1) The first prerequisite is the PRNs in both navigation and observation data file are the same.
 	
2) The second prerequisite is the satellites' ephemerides are valid within four hours, as the nearest time of the satellite in both RINEX data are match.
	$T^{p,GPS} = t_{i,GPS} - \frac{\rho_i^p}{c} = (t_i - \delta t_i) - \frac{R_i^p - c(\delta t_i - \delta T^p)}{c} = (t_i - \delta t_i) - \frac{R_i^p}{c} + (\delta t_i - \delta T^p) = t_i - \frac{R_i^p}{c} - \delta T^p$	`%========Extract PRN Dada through Navigation and Observation file====`
		`Compare_Nav_PRN=[Nav(:).PRN];`
		`Compare_Obs_PRN=[Obs(:).PRN];`
		`%=============GPS Time of Navigation and Observation file============`
		`Compare_Nav_GPSTime=[Nav(:).Time_in_GPS];`
		`Compare_Obs_GPSTime=[Obs(:).Time_in_GPS];`
		`%=====Match the GPS Time of Navigation and Observation file==========`
		`RxClockError = 0.01;`
		`RxClockDiff = 1;`
		`dx = [0.01;0.01;0.01; RxClockError];`
		`Approximate_Coor = [Approx_X; Approx_Y; Approx_Z; RxClockError];`
		`%===============The nearest GPS time with 4 Hours Validation=========`
		`Difference_Minimum = 4*60*60; %4-hours validation`
		`for m = 1:No_RowOfSamePRN(i)`
		`Difference_GPSTime = abs(Compare_Obs_GPSTime(1,i) - Compare_Nav_GPSTime(1,RowOfSamePRN(m,i)));`
		`if (Difference_GPSTime < Difference_Minimum)`
		`Difference_Minimum = Difference_GPSTime;`
		`Sate_Row = RowOfSamePRN(m,i);`
		`end`

The Progress of Intermediate Report 2

Preparation - Make sure both RIXEX navigation and observation files are already correctly saved in MATLAB workplace (database). Moreover, the PRN number and the GPS time from both files should also be identical. In addition, input physical constants for the upcoming orbit calculation.

$c = 299792458$ m/s
$g = 9.80665$ m/s^2
$G = 6.67259 \times 10^{-11}$
$GM = 3.986005 \times 10^{14}$ m^3/s^2
$\dot{\Omega}_e = 7.2921151467 \times 10^{-5}$ rad/s

```
%==========================Basic Constants=====================================
c=299792458;            %Physical Constant: Speed of Light (m/s)
g=9.80665;              %Physical Constant: Acceleration of Gravity (m/s^2)
G=6.67259e-11;          %Physical Constant: Constant of Gravity (Nm^2/kg^2)
GM=3.986005e+14;            %Spaceflight Constant: GM(Earth) (m^3/s^2)
omega_E=7.2921151467e-05;   %Earth rotation rate (rad/s)
%==============================================================================
```

Step 7 - Semi-major Axis (a): It is the longest semi-diameter of the ellipse. The sqrt(a) could be read from RINEX navigation file.

$$r_{max} - a = a - r_{min}$$
$$a = \frac{p}{1 - \varepsilon^2}$$

```
%==========================Semi-major Axis (a)=================================
semi_major_axis=(Nav(Sate_Row).sqrt_a)^2;
a = semi_major_axis;
%==============================================================================
```

Step 8 - Time difference between time and ephemeris reference time (tk): This step is used to compute the differences between GPS transmission time, ephemeris reference. Afterwards, by using "if....else" condition, to determine the temporal differences and make time correction.

$$t_k = t - t_{oe}$$

```
%=================Time from Ephemeris Reference Epoch=========================
halfweek = 3.5*60*60*24;        %In terms of GPS Time
Transmission_Time = Obs(i).C1 /c;       %Signal transmission time
t = Obs(i).Time_in_GPS - Transmission_Time - RxClockError;

tk = (t - Nav(Sate_Row).Toe_time);
if (tk > halfweek)
tk = tk - (2*halfweek);
elseif (tk < -halfweek)
tk = tk + (2*halfweek);
end
%==============================================================================
```

Step 9 - Mean Motion (n): It is the angular speed that is required for a body in elliptical orbit.

$$n = \sqrt{\frac{\mu}{a^3}}$$

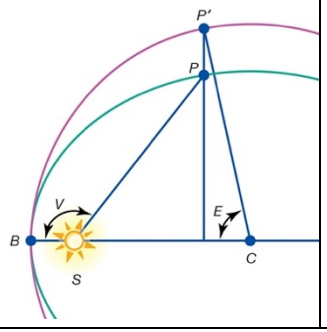
```
%==========================Mean motion (n)=====================================
n0 = sqrt (GM / (a^3));
n = n0 + Nav(Sate_Row).Delta_N;
%==============================================================================
```

Step 10 - Mean Anomaly (M): It gives an average value regarding the angular position of the satellite, with reference to the perigee.

$$M_k = M_0 + t_k(n_0 + \Delta n)$$

```
%==========================Mean anomaly (M)====================================
M = Nav(Sate_Row).M0 + (n * tk);
%==============================================================================
```

Step 11 - Eccentric Anomaly (E): An angle referring to the position of a body moving through the elliptic Kepler orbit. It is computed with iterations.



$$E_k = M_k + e \sin(Ek)$$

```
%==========================Eccentric anomaly (E)===============================
n = 1;
E = 1;
E0 = M;

while n <= 30 && abs(E0 - E) > 10^(-14)
E_new = M +(Nav(Sate_Row).e) * sin(E);
E = E0;
E0 = E_new;
n = n + 1;
end
%==============================================================================
```

Step 12 - True Anomaly (m): An angle from perigee to the satellite position, which gives true angular position of a satellite.	$v_k = \tan^{-1}\left(\frac{\sin E_k\left(\sqrt{1-e^2}\right)}{\cos E_k - e}\right)$	`%============================True anomaly (m)==============================` `True_Anomaly = 2*atan(sqrt((1 + Nav(Sate_Row).e)/(1 - Nav(Sate_Row).e)) * tan(E/2));`

Step 13 - Argument of Latitude: This is an angular parameter which refers the sum of the argument of perigee and true anomaly.	$\phi_k = v_k + \omega$	`%=============Argument of latitude (Coorected_ArgLat)=============-=========`
		`phi = True_Anomaly + Nav(Sate_Row).Omega;`
		`%================================================================================`

Step 14 - Orbit Radius of the Satellite Position: It is used to calculate the satellite orbit radius. To be noted, the e in the formula means eccentricity of the satellite orbit plane. The second Harmonic Perturbation is added.	$r_k = A * (1 - e\cos E_k) + \delta r_k$
	$\delta r_k = C_{rs}\sin(2\phi_k) + C_{rc}\cos(2\phi_k)$
		`%===============Orbit Radius of the GPS satellite position===========--=====`
		`r = a * (1 - Nav(Sate_Row).e * cos(E));`
		`delta_r = Nav(Sate_Row).Crs * sin(2*phi) + Nav(Sate_Row).Crc * cos(2*phi);`
		`Orbit_Radius = r + delta_r;`
		`%================================================================================`

Step 15 - Inclination: An angle between the earth equatorial plane and the orbital plane. It computes at the ascending node from the equator to the orbit, from the east to the north. The second Harmonic Perturbation is added.	$i_k = i_0 + \delta i_k + (IDOT)\, t_k$
	$\delta i_k = C_{is}\sin(2\phi_k) + C_{ic}\cos(2\phi_k)$
		`%========================Corrected GPS orbit Inclination=====================`
		`inclination_i = Nav(Sate_Row).i0 + Nav(Sate_Row).IDOT * tk;`
		`delta_i = Nav(Sate_Row).CIS* sin(2*phi) + Nav(Sate_Row).Cic * cos(2*phi);`
		`Corrected_Inclination = inclination_i + delta_i;`
		`%================================================================================`

Step 16 - Corrected Argument of Latitude: Similar to Step 7, it is used to confirm the sum of argument of perigee and true anomaly after computing the orbit radius and inclination. The second Harmonic Perturbation is added.	$\delta u_k = C_{us}\sin(2\phi_k) + C_{uc}\cos(2\phi_k)$
	$u_k = \phi_k + \delta u_k$
		`%========================Corrected Argument of Latitude=====================`
		`delta_phi = Nav(Sate_Row).Cus * sin(2*phi) + Nav(Sate_Row).Cuc * cos(2*phi);`
		`Coorected_ArgLat = phi + delta_phi;`
		`%================================================================================`

Step 17 - GPS Position in the orbital plane: This step is to find out the GPS position moving along the orbit by using corrected argument of latitude.	$x'_k = r_k \cos u_k$
	$y'_k = r_k \sin u_k$
		`%====================GPS positions in the orbital plane=====================`
		`X0 = Orbit_Radius * cos(Coorected_ArgLat);`
		`Y0 = Orbit_Radius * sin(Coorected_ArgLat);`
		`%================================================================================`

Step 18 - Longitude of the ascending node: This step is to find out the longitude of the ascending node, to confirm the orbit of the satellite in space.	$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e)t_k - \dot{\Omega}_e\, t_{oe}$	`%========================Longitude of ascending node=========================`
		`Omega_K = Nav(Sate_Row).Omega_0 + (Nav(Sate_Row).Omega_dot - omega_E) *tk - omega_E * Nav(Sate_Row).Toe_time;`
		`%================================================================================`

Step 19 - Earth-centered earth-fixed (ECEF) frame in orbital terrestrial coordinate system: Eventually, the satellite positions in the orbital plane. The coordinate is presumably generated by the control segment in ECEF frame. The coordinate is then saved in workplace.	$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x'_k\cos\Omega_k - y'_k\cos i_k\sin\Omega_k \\ x'_k\sin\Omega_k + y'_k\cos i_k\cos\Omega_k \\ y'_k\sin i_k \end{bmatrix}$	`%=========Orbital Coordinate System to Terrestrial Coordinate System=======`
		`%======================Earth-centred Earth-fixed Frame=====================`
		`Satellite_Position(i,1) = X0 * cos(Omega_K)- Y0 * cos(Corrected_Inclination) * sin(Omega_K);`
		`Satellite_Position(i,2) = X0 * sin(Omega_K)+ Y0 * cos(Corrected_Inclination) * cos(Omega_K);`
		`Satellite_Position(i,3) = Y0 * sin(Corrected_Inclination);`
		`%==================The END of GPS Satellite Positions=====================`

3 GPS Receiver Positioning

3.1 Algorithm for calculating GPS receiver position using pseudorange measurement

The below formula is the pseudorange measurement, including geometric distance and error terms.

$$P_i^k = \rho + d_{orb} + c(dt_i - dt^k) + d_{trop} + d_{ion} + others \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The signal transmitting path between satellite and receiver can be described by

$$\rho_i^k = \sqrt{(x^k - x_i)^2 + (y^k - y_i)^2) + (z^k - z_i)^2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

where $x^k, y^k, z^k = \ computed\ satellite\ position\ ;\quad x_i, y_i, z_i$ = approximate receiver position.

Since there are number of satellites observed in the epochs, the pseudorange values could be computed by the satellite coordinates and initial receiver coordinates. In each iteration, the computed coordinates would replace the initial receiver coordinates and update the satellite positions by applying the receiver clock correction and calculate the pseudorange again.

3.2 Linearization of pseudorange equation

(1) This process includes the differentiation of the transmitting path equation.

$$\rho_i^p = \sqrt{(X_i - X^p)^2 + (Y_i - Y^p)^2 + (Z_i - Z^p)^2} = rho + \frac{(X_0 - X^p)}{rho}\delta X_i + \frac{(Y_0 - Y^p)}{rho}\delta Y_i + \frac{(Z_0 - Z^p)}{rho}\delta Z_i$$

$$where\ rho = \sqrt{(X_0 - X^p)^2 + (Y_0 - Y^p)^2 + (Z_0 - Z^p)^2}$$

$rho = \sqrt{(X_0 - X^p)^2 + (Y_0 - Y^p)^2 + (Z_0 - Z^p)^2}$, $X_i = X_0 + \delta X$, $Y_i = Y_0 + \delta Y$, $Z_i = Z_0 + \delta Z$

(2) Therefore, the pseudorange can be represented as:

$$R_i^p = rho + \frac{(X_0 - X^p)}{rho}\delta X_i + \frac{(Y_0 - Y^p)}{rho}\delta Y_i + \frac{(Z_0 - Z^p)}{rho}\delta Z_i + orb_i^p + c(\delta t_i - \delta T^p) + Ion_i^p + Trop_i^p + \sigma_i^p$$

(3) Afterwards, the matrix form could be rearranged as shown below.

$$\frac{(X_0 - X^p)}{rho}\delta X_i + \frac{(Y_0 - Y^p)}{rho}\delta Y_i + \frac{(Z_0 - Z^p)}{rho}\delta Z_i + c\delta t_i - f = B \cdot \delta X - f = 0$$

$$B = \begin{bmatrix} \frac{(X_0 - X^p)}{rho} & \frac{(Y_0 - Y^p)}{rho} & \frac{(Z_0 - Z^p)}{rho} & 1 \end{bmatrix} \qquad \delta X = \begin{bmatrix} \delta X_i \\ \delta Y_i \\ \delta Z_i \\ c\delta t_i \end{bmatrix}$$

$$-f = rho - R_i^p - c\delta T^p + Ion_i^p + Trop_i^p + orb_i^p + \sigma_i^p$$

$$= \sqrt{(X_0 - X^p)^2 + (Y_0 - Y^p)^2 + (Z_0 - Z^p)^2} - R_i^p - c\delta T^p + Ion_i^p + Trop_i^p + orb_i^p + \sigma_i^p$$

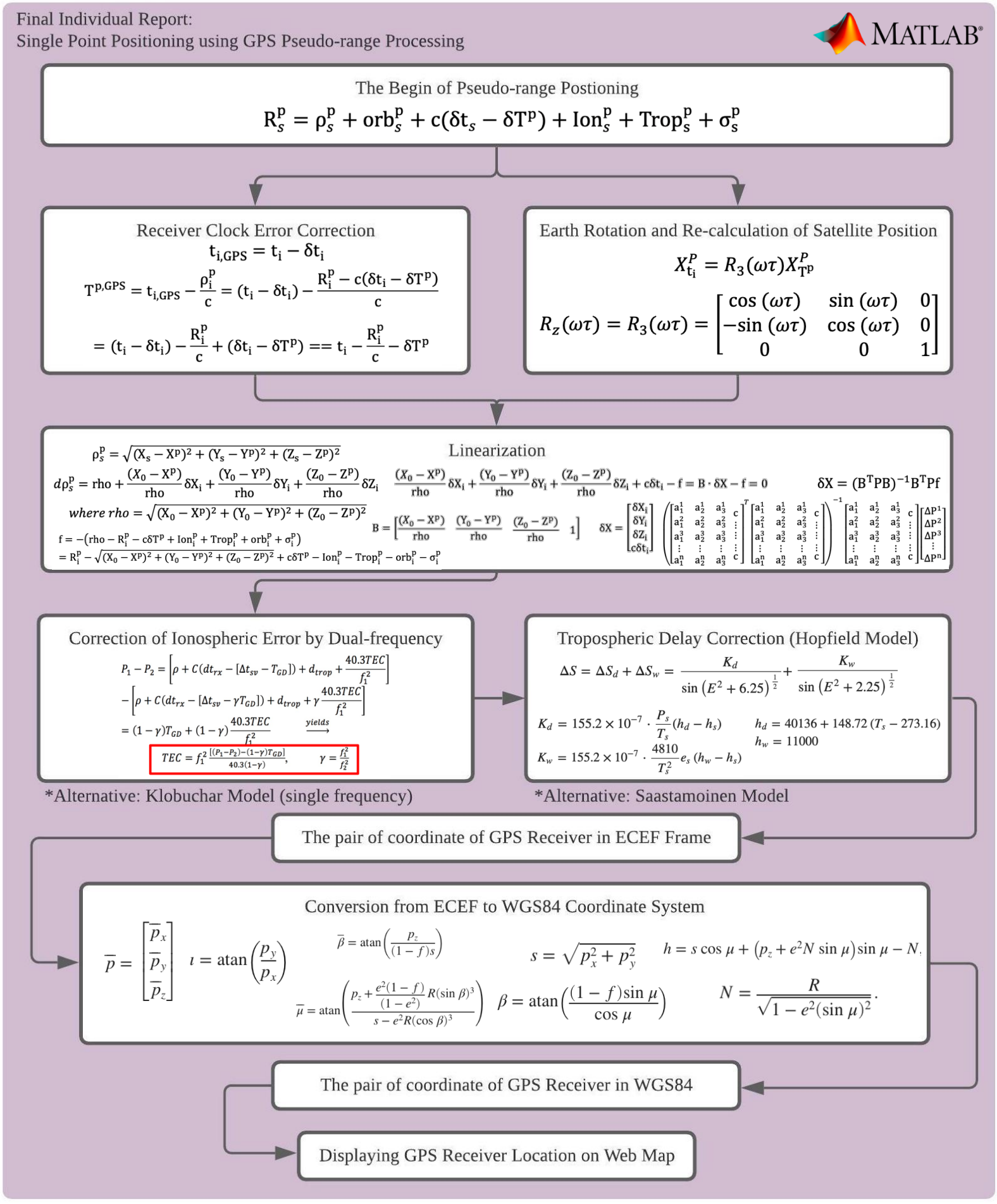
(4) Applying the Least Square estimator, it can be calculated by:

$$X = \ (B^T P B)^{-1} B^T P f$$

$$\left(\begin{bmatrix} a_1^1 & a_2^1 & a_3^1 & c \\ a_1^2 & a_2^2 & a_3^2 & \vdots \\ a_1^3 & a_2^3 & a_3^3 & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ a_1^n & a_2^n & a_3^n & c \end{bmatrix}^T \begin{bmatrix} a_1^1 & a_2^1 & a_3^1 & c \\ a_1^2 & a_2^2 & a_3^2 & \vdots \\ a_1^3 & a_2^3 & a_3^3 & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ a_1^n & a_2^n & a_3^n & c \end{bmatrix} \right)^{-1} \begin{bmatrix} a_1^1 & a_2^1 & a_3^1 & c \\ a_1^2 & a_2^2 & a_3^2 & \vdots \\ a_1^3 & a_2^3 & a_3^3 & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ a_1^n & a_2^n & a_3^n & c \end{bmatrix} \begin{bmatrix} \Delta P^1 \\ \Delta P^2 \\ \Delta P^3 \\ \vdots \\ \Delta P^n \end{bmatrix}$$

3.2 Framework of GPS Receiver Positioning

The second framework shows the computational procedures of the GPS receiver coordinate. Both receiver clock error and earth rotation correction can be undertaken simultaneously and the linearization for the pseudorange equation and atmospheric error corrections are followed. Transformation to WGS84 Datum is the last step.



<table>
<tr><th colspan="3">The Progress of Final Report</th></tr>
<tr>
<td>Step 20 - After the orbit satellite coordinates in ECER frame are successfully computed, the earth rotation correction can be conducted. The rotation corrections are computed in a 3x3 matrix format.

To be remarked, since the Sagnac Effect is easily occurred, suggesting the onboard satellite clock run too fast or too slow relative to the receiver clock on the geoid (equipotential surface), therefore the re-calculation of orbit satellite coordinates.</td>
<td>$$R_z(\omega\tau) = R_3(\omega\tau) = \begin{bmatrix} \cos(\omega\tau) & \sin(\omega\tau) & 0 \\ -\sin(\omega\tau) & \cos(\omega\tau) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$</td>
<td><pre>
%============================Earth Rotation=========================
WT = omega_E * t_emission;
Rotation_r(1,1) = cos(WT);
Rotation_r(1,2) = sin(WT);
Rotation_r(1,3) = 0;
Rotation_r(2,1) = -sin(WT);
Rotation_r(2,2) = cos(WT);
Rotation_r(2,3) = 0;
Rotation_r(3,1) = 0;
Rotation_r(3,2) = 0;
Rotation_r(3,3) = 1;
%================= Re-calculation of Orbit Coordinate ==============
GPS_Coordinate = GPS * Rotation_r;
</pre></td>
</tr>
<tr>
<td>Step 21 - This process is to cancel the satellite clock error. Using the coefficients (e.g., SV clock bias, drift, and drift rate) in the RINEX navigation data, to correct satellite clock errors.</td>
<td>$$\tilde{\delta t}^{sat} = a_0 + a_1(t - t_0) + a_2(t - t_0)^2$$
a_0 = SV Clock bias;
a_1 = SV Clock drift;
a_2 = SV Clock drift rate.</td>
<td><pre>
%================== Satellite Clock Error ============================
SxClockError(i) = c * (Nav(Sate_Row).SV_Clock_Bias +
(Nav(Sate_Row).SV_Clock_drift)* tk +
(Nav(Sate_Row).SV_Clock_drift_rate)* tk^2);
</pre></td>
</tr>
<tr>
<td>Step 22 - This process is to cancel the ionospheric delay by ionosphere-free combination for the dual-frequency receiver. In accordance with the ionospheric refraction, the first order of the ionospheric effect counts heavily on the inverse of squared signal frequency. As a result, the GPS receiver contains dual frequency is able to eliminate the ionospheric refraction.</td>
<td>$$P_1 - P_2 = \left[\rho + C(dt_{rx} - [\Delta t_{sv} - T_{GD}]) + d_{trop} + \frac{40.3TEC}{f_1^2}\right] - \left[\rho + C(dt_{rx} - [\Delta t_{sv} - \gamma T_{GD}]) + d_{trop} + \gamma\frac{40.3TEC}{f_1^2}\right] = (1-\gamma)T_{GD} + (1-\gamma)\frac{40.3TEC}{f_1^2} \xrightarrow{yields}$$
$$TEC = f_1^2 \frac{[(P_1 - P_2) - (1-\gamma)T_{GD}]}{40.3(1-\gamma)}, \qquad \gamma = \frac{f_1^2}{f_2^2}$$</td>
<td><pre>
%===== Ionosphere-free Combination for Dual-Frequency Receivers =====
F1 = 1575.42;
F2 = 1227.60;

for c = 1:length(Sate_Row)
L1(c) = GPS(c).Nav(Sate_Row).Cic;
L2(c) = str2num(GPS(c).C2W);
if strmp (GPS(c).C2W,'')
ionofree(c)=L1(c);
else
ionofree(c)=(1/(F1^2 - F2^2))*(L1(c)*f1^2 - L2(c)*f2^2);
end
P_SV (c) = ionofree(c)+GPS(c).clock_off* 299792458;
Positon_SV(c).P = P_sv(c);
end
</pre></td>
</tr>
<tr>
<td>Step 23 - This process is to cancel the tropospheric delay by the Hopfield Model, which is widely used for cancelling this error. Nonetheless, concerning there are no given weather information and atmospheric conditions at a certain place on 31st March 2001, it could not perform the program code regarding the cancellation of the tropospheric delay. Once the relevant information is given,</td>
<td>$$\Delta S = \Delta S_d + \Delta S_w = \frac{K_d}{\sin(E^2 + 6.25)^{\frac{1}{2}}} + \frac{K_w}{\sin(E^2 + 2.25)^{\frac{1}{2}}}$$</td>
<td><pre>
%========== Tropospheric Error Correction (Hopfield Model) ==========
Tropospheric_Error =
Error_Tropospheric_Hopfield(T_amb,P_amb,P_vap,Pos_Rcv,Pos_SV)
S=size(Pos_SV);
m=S(1);n=S(2);

for i=1:m
  [E,A0] = Calc_Azimuth_Elevation(Pos_Rcv,Pos_SV(i,:));
  El(i)=E;
</pre></td>
</tr>
</table>

the tropospheric error correction can be performed.
 	
Apart from The Hopfield Model, The Saastamoinen Model is an alternative for correcting the tropospheric delay of refraction. Likewise, the information of surface temperature and water vapor pressure at the surface level are remained unknown. As a result, it could not perform tropospheric error correction in this project.
	$K_d = 155.2 \times 10^{-7} \cdot \frac{P_s}{T_s}(h_d - h_s)$
	
	$K_w = 155.2 \times 10^{-7} \cdot \frac{4810}{T_s^2} e_s\,(h_w - h_s)$
	
	$h_d = 40136 + 148.72\,(T_s - 273.16)$
	$h_w = 11000$
		`A(i)=A0;`
		`end`
		
		`%=======================Zenith Hydrostatic Delay=====================`
		`Kd=1.55208*10^(-4)*P_amb*(40136+148.72*T_amb)/(T_amb+273.16);`
		`% %=========================Zenith Wet Delay=========================`
		`Kw=-.282*P_vap/(T_amb+273.16)+8307.2*P_vap/(T_amb+273.16)^2;`
		
		`for i=1:m`
		`Denom1(i)=sin(sqrt(El(i)^2+1.904*10^-3));`
		`Denom2(i)=sin(sqrt(El(i)^2+.6854*10^-3));`
		`% %====================Tropospheric Delay Correctoion================`
		`Tropospheric_Error(i)=Kd/Denom1(i)+Kw/Denom2(i);`
		`end`

Step 24 - This step is to calculate the geometric distance between satellite and GPS receiver by linearized pseudorange equations. According to the RINIX observation data, 23209 satellites are recorded, therefore it should be 23209 values of pseudoranges (distances) are linearized and computed.	$rho = \sqrt{(X_0 - X^p)^2 + (Y_0 - Y^p)^2 + (Z_0 - Z^p)^2}$	`%==========Geometric distance between satellite and receiver=========`
		`rho(i) = sqrt((Approximate_Coor(1,1)- GPS(i,1))^2 +`
		`(Approximate_Coor(2,1)- GPS(i,2))^2 + (Approximate_Coor(3,1)-`
		`GPS(i,3))^2);`
		`%======================= Formation of Function (f)==================`
		`f(i,1) = Obs(i).C1 - rho(i) + SxClockError(i);`
		`end`

Step 25 - As the iteration starts from the beginning of the orbit satellite computation, the iteration is stopped as final step before calculation of GPS receiver position. The corrections shall be applied to both position fix and GPS receiver clock correction.	$\delta X = (B^T P B)^{-1} B^T P f$
	
	$\left(\begin{bmatrix} a_1^1 & a_2^1 & a_3^1 & c \\ a_1^2 & a_2^2 & a_3^2 & \vdots \\ a_1^3 & a_2^3 & a_3^3 & \vdots \\ \vdots & \vdots & \vdots & c \\ a_1^n & a_2^n & a_3^n & c \end{bmatrix}^T \begin{bmatrix} a_1^1 & a_2^1 & a_3^1 & c \\ a_1^2 & a_2^2 & a_3^2 & \vdots \\ a_1^3 & a_2^3 & a_3^3 & \vdots \\ \vdots & \vdots & \vdots & c \\ a_1^n & a_2^n & a_3^n & c \end{bmatrix}\right)^{-1} \begin{bmatrix} a_1^1 & a_2^1 & a_3^1 & c \\ a_1^2 & a_2^2 & a_3^2 & \vdots \\ a_1^3 & a_2^3 & a_3^3 & \vdots \\ \vdots & \vdots & \vdots & c \\ a_1^n & a_2^n & a_3^n & c \end{bmatrix} \begin{bmatrix} \Delta P^1 \\ \Delta P^2 \\ \Delta P^3 \\ \vdots \\ \Delta P^n \end{bmatrix}$
		`%==================== Least Square Adjustment ======================`
		`dxprevious = RxClockError;`
		`dx = inv(transpose(B) * B) * transpose(B) * f;`
		`Approximate_Coor = Approximate_Coor + dx;`
		`RxClockError = dx(4,1)/c;`
		`RxClockDiff = RxClockError - dxprevious;`
		`iteration = iteration + 1;`
		`end`

Step 26 - The coordinate of GPS receiver position in Earth-Centered Earth-Fixed (ECEF) frame is stored in a matrix format.
 	
To show the result, the coordinate of GPS receiver position in ECEF frame is printed on the command window.
	N/A	`%============= The Coordinate of GPS Receiver Position ==============`
		`GPS_Receiver(1,1) = Approximate_Coor(1);`
		`GPS_Receiver(2,1) = Approximate_Coor(2);`
		`GPS_Receiver(3,1) = Approximate_Coor(3);`
		`GPS_Receiver(4,1) = Approximate_Coor(4);`
		`disp('-------------------------------------------------------------')`
		`fprintf('MATLAB Computation Output:\n The GPS Receiver Position (in`
		`ECEF Frame) is X: %f, Y: %f, Z: %f.\n',`
		`GPS_Receiver(1,1),GPS_Receiver(2,1),GPS_Receiver(3,1));`
		`disp('-------------------------------------------------------------')`

Step 27 - This step is to convert the ECEF coordinates to WGS84 coordinate system, to easily search the GPS receiver location.	N/A	`%==================== ECEF Frame to WGS84 Datum ==================`
		`X = GPS_Receiver(1,1);`
		`Y = GPS_Receiver(2,1);`
		`Z = GPS_Receiver(3,1);`

Step 28 - Similar to the orbit satellite computation, there are several physical constants must first define. For instance, semi-major axis and semi-minor axis, ellipsoid flattening, the first and second eccentricity.

$$\sqrt{1-\frac{b^2}{a^2}}\qquad \sqrt{\frac{a^2}{b^2}-1}$$

Left formula: First eccentricity
Right formula: Second eccentricity

```
a = 6378137;              %Physical Constant: Semi-major Axis
f = 1/298.257223563;      %Physical Constant: Ellipsoid Flattening
b = a*(1-f);              %Physical Constant: Semi-minor Axis
P = sqrt(X^2 + Y^2);
Theta = atan(Z*a/P*b);
E = sqrt(((a^2) - (b^2))/a^2); % First eccentricity
e2 = sqrt((a^2 - b^2)/b^2);     % Second eccentricity
```

Step 29 - This step is to compute the coordinate transformation from ECEF to WGS84 coordinate. The latitude, longitude, and altitude of WGS84 are computed, respectively.

$$\overline{\beta} = \operatorname{atan}\left(\frac{p_z}{(1-f)s}\right)$$

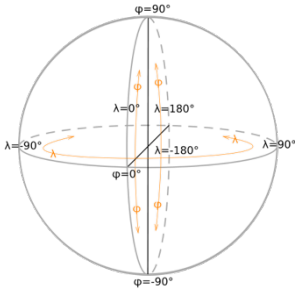
$$\overline{\mu} = \operatorname{atan}\left(\frac{p_z + \frac{e^2(1-f)}{(1-e^2)}R(\sin\beta)^3}{s - e^2R(\cos\beta)^3}\right)$$

```
%============= Iteration loop for estimate Latitude ================
Latitude = atan2(Z,(P*(1-e^2)));
for i = 1:10000
    N = a / sqrt(1-e^2*sin(Latitude).^2);           % Prime vertical
    Altitude = (P/cos(Latitude)) - N;               % Altitude
    Latitude = atan2(Z,(P*(1-e^2*(N/(N + Altitude))))); % Latitude
end
```

Step 30 - The coordinate of GPS receiver position in WGS84 coordinate system is converted. The coordinate is also printed on the command window.

The coordinate of the GPS Receiver is in at the car park nearby Albert Head Road, at Metchosin, in British Columbia, Canada:

(48.389786, -123.487116, 49.570577)

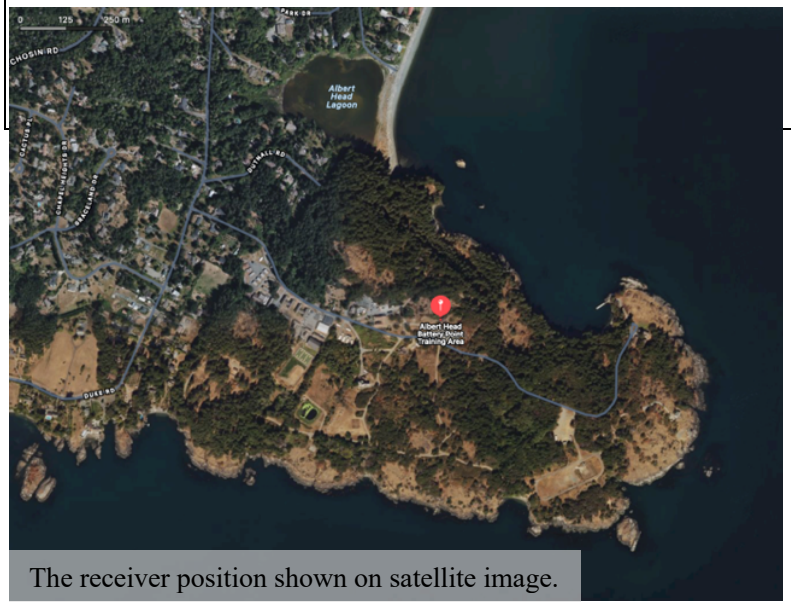



```
%==================== Latitude and Longitude =======================
Longitude = (atan2(Y,X))*180/pi;
Latitude   = Latitude*180/pi;
disp('------------------------------------------------------------------')
fprintf('2. MATLAB Computation Output: \n The GPS Receiver Position
(in WGS84 Datum) is X(Long): %f, Y(Lat): %f, Z(Alt): %f.\n',
Longitude, Latitude, Altitude);
disp('------------------------------------------------------------------')
```

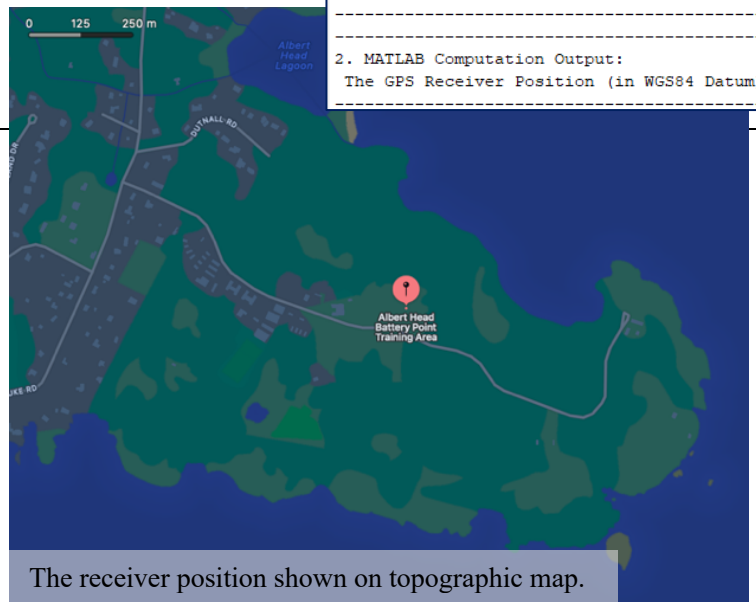
Final Output – The location of the GPS receiver shown in the screenshots of Web Map.

Address: ***9GQ7 + W56 Metchosin, British Columbia, Canada.***

```
A pair of coordinate of GPS Receiver is now computing, Pls wait for a moment.
Thank you for your understanding.
------------------------------------------------------------------
1. MATLAB Computation Output:
 The GPS Receiver Position (in ECEF Frame) is X: -2341317.379804, Y: -3539073.567852, Z: 4745804.978851.
------------------------------------------------------------------
------------------------------------------------------------------
2. MATLAB Computation Output:
 The GPS Receiver Position (in WGS84 Datum) is X(Long): -123.487116, Y(Lat): 48.389786, Z(Alt): 49.570577.
------------------------------------------------------------------
```



The receiver position shown on satellite image.



The receiver position shown on topographic map.

Zoom-in of GPS receiver position.

4 Discussion and Improvement

In the self-developed MATLAB program project, only satellite clock error, receiver clock error and ionospheric delay are cancelled. However, there are several source of errors could not be eliminated successfully, such as multipath effect as well as tropospheric delay. Owing to the relevant information is missing at the unknown position (students have to find out the GPS receiver position), such as the surface temperature and water vapor pressure at the surface level, it could not possibly eliminate these errors. The below shall discuss how different errors can be cancelled and improve the self-developed program.

4.1 Errors related to the Propagation in the Atmosphere

The performance of the GPS receiver counts heavily on the accuracy levels of the ways of range measurement. As long as the GPS signal is propagated from the satellite in the space to the GPS receiver, the signal is pass through various layers of the atmosphere and then being inevitably bent and/or reflected due to the properties of the atmospheric layers and buildings in high rise high dense urban areas, etc.

4.1.1 Multipath Effect (Signal Blockage)

Multipath effect is a common phenomenon, in which the GPS signal reaches a receiver's antenna through more than one path (Souza and Negri, 2017). Theoretical Speaking, the received signal inside the GPS receiver should be transmitted from satellite directly and without bending and reflection. Nonetheless, the indirect signal might be reflected by reflecting surfaces in the nearby environment, are also received by the GPS receiver. Such multipath is called receiver multipath. Reflecting surfaces can be on the ground, water, buildings, constructions, trees and mountain, etc (Wang et al., 2020). Multipath is the most critical error source in the urban area due to the existence of high-density buildings (Rumora, Sikirica and Filjar, 2018).

To avoid multipath effect, a proper site for situating the GPS receiver position is of significance. Furthermore, it can place the GPS receiver in a reflection-free location or place it nearby the open sky window. Adopting a "choke ring" antenna can also mitigate the multipath (Liu's LSGI3322 Lecture 3 (Error Modelling), 2021).

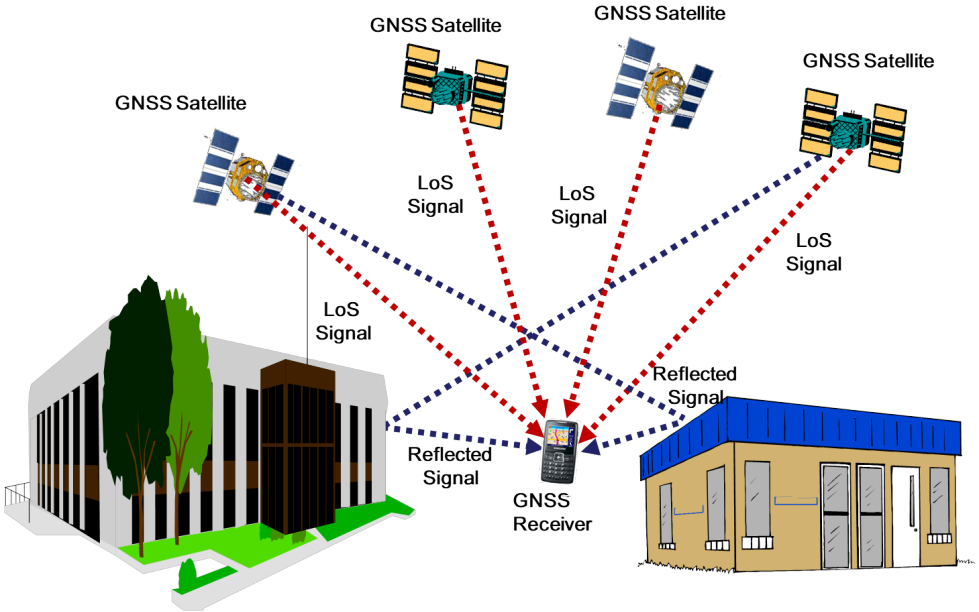


Figure 3. Illustration of the Multipath Effect (Hsu, 2017).

4.1.2 Tropospheric Delay

The troposphere is located in the lower layer of the Earth atmosphere, which is around 10 km above the sea level. It contains 80 % of the air mass on the Earth (Williams and Nievinski, 2017). Owing to the existence of the water vapour, the troposphere is the layer where most of the weather phenomenon happens, such as cloudy and thunderstorm (Doin et al., 2009; Li et al., 2012). The gas in the troposphere turns the troposphere into a

refractive layer. Hence, when GPS signal passes through the troposphere, it delays the signals. It means the distance between satellite and receiver is longer than the real range (Doin et al., 2009). The dry and wet components for tropospheric delay match with those dry and wet gases in the troposphere layer. Yet, the dry part is harder to model, it only accounts for 10% of the tropospheric delay (Yang et al., 2021). Whereas, the wet part, which is easier to model instead, accounts for the remaining delay (Elsobeiey, 2020).

Considering tropospheric error is independent from frequency, it cannot be eliminated by L1 and L2 GPS signals measurement combination (Elsobeiey, 2020), unlike ionospheric delay. To execute tropospheric error mitigation for meter-level accuracy, a couple of models can be deployed, including Hopefield and Saastamoinen models. These models generally compute the zenith delay for elevation angle which is 0° (Doin et al., 2009). Afterwards, the total slant delay is obtained by a mapping function, relying on the satellite elevation angles (Li et al., 2012). When a tropospheric error estimation with higher accuracy would like to be achieved, the dry component of tropospheric effect is modelled, and the wet component is evaluated as an extra unknown in the navigation filter (Williams and Nievinski, 2017).

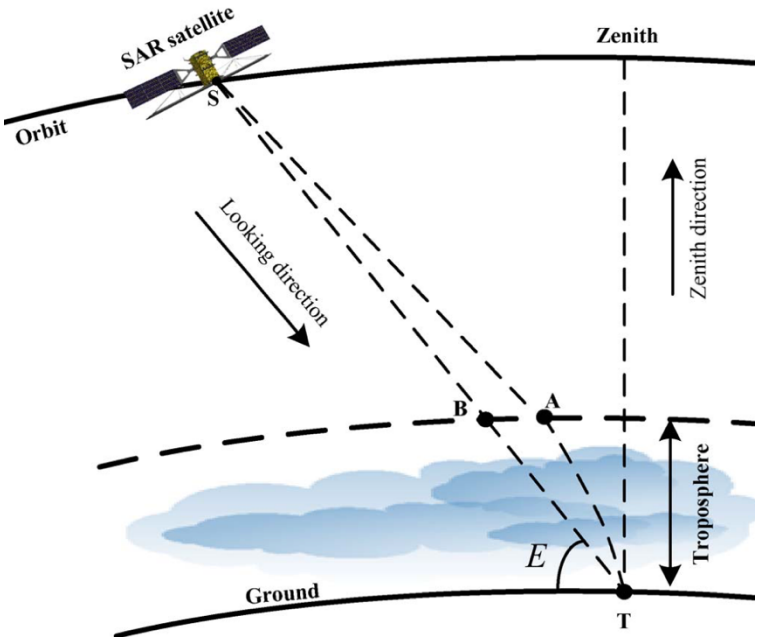


Figure 4. Illustration of Tropospheric Delay (Yu et al., 2015).

4.2 Transferring Time of The GPS Signals

In my MATLAB program code, it only adopts a simple algorithm to calculate the GPS signal travelling time. The transmission time shall not be improved with an iteration as the observation file in L1 frequency is fixed. Hence, only modified receiver clock correction being applied in iterations for the computing satellite position and receiver position (Dai et al., 2020). With the aim of optimizing the program in temporal perspective, the transfer time must be computed though another precise method as running iteration as follows (Wang, 2013):

- Calculate the geometric distance between receiver's initial position and satellite position previously.
- $\Delta t = \frac{geometric\ distance}{c\ (speed\ of\ light)}$
- Recalculate the satellite position at the time: t = observation-Δt with earth rotation correction
- Repeat the above steps until the difference between new computed transfer time and the previously computed transfer time is smaller than $1 \cdot 10^{-12}$ or any assigned values
- With the corrected transmission time and earth rotation, a more accurate satellite position results could be computed, and hence, it will substantially improve the GPS receiver position.

4.3 Running without Weighting in Least Square Adjustment (Linearization)

In my program code, it is assumed that all the measurement accuracies are identical. However, in the real measurement progress, the measurement error is smaller from the high elevation angle satellite, and vice versa. The satellite which has lower elevation angle must assign the smaller weighting resulting from serious effect of the atmospheric delays (Li et al., 2015). The weighting of each satellite is meaningful when it is calculated the unknown (dx, dy, dz) and receiver clock error by weighted least squares algorithm (Qian et al., 2016).

4.4 Running without Relativistic Clock Correction for the Satellite Clock Error

Satellite clock will appear running faster than the same clock on Earth by approximately 38.4 μs/day and scaled by the speed of light (Cocke, 1996). Indeed, this is equal to the range error of about 11512 m (Yao et al, 2021). In order to compensate the effect, we can apply the relativistic clock corrections.

$$\text{Relativistic clock correction} = \frac{2\sqrt{\mu}}{c^2} \cdot e\sqrt{A} \cdot sin(Ek)$$

μ= WGS84 Value of the earth's universal gravitational parameter for GPS user; e = eccentricity; $\sqrt{A}$= square root of the semi-major axis; Ek = eccentric anomaly.

5 Conclusion

In this LSGI3322 final individual project, it successfully developed a MATLAB program by myself, to perform the single point positioning (SPP) using pseudorange measurement, so as to compute the unknown GPS receiver position (site_0900.01n and site_0900.01o).

The MATLAB program performed both readings of RINEX navigation and observation data files, computation of satellite orbit positions and GPS receiver position. The satellite clock error, receiver clock error, earth rotation error and iono-free combination of dual-frequency were subsequently developed. The program output displayed the GPS receiver position was at (48.389786° N, 123.487116° W) in WGS84 coordinate system on 31st March 2001, where was a car park nearby and a battery point training centre nearby the Albert Head Road, at Metchosin, in British Columbia, Canada.

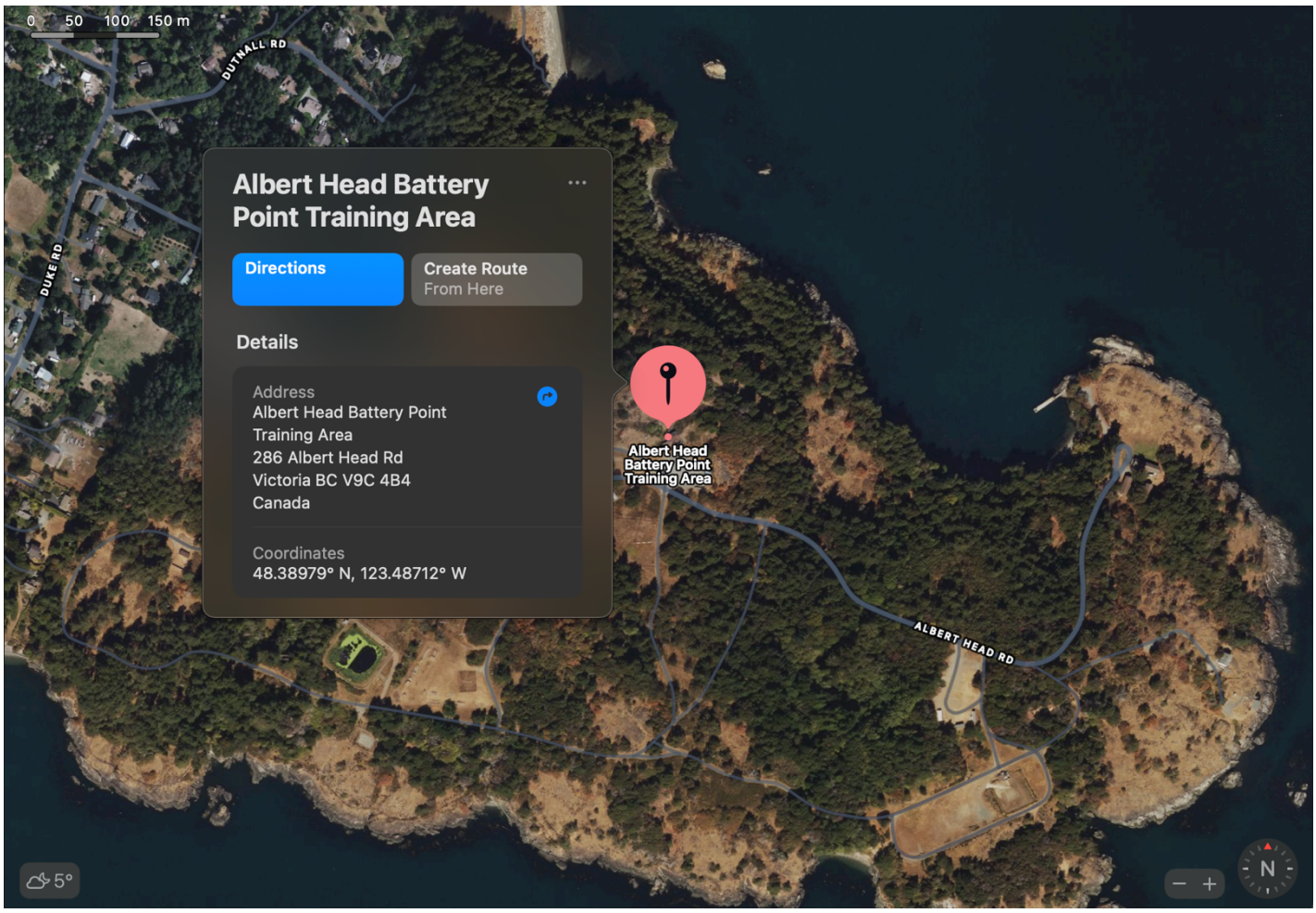


Figure 5. The computed GPS receiver position at Metchosin, in British Columbia, Canada.

Reference List

Bui V., Le, N. T., Vu, T. L., Nguyen, V. H., & Jang, Y. M. (2020). GPS-based indoor/outdoor detection scheme using machine learning techniques. *Applied Sciences*, 10(2), 500. https://doi.org/10.3390/app10020500

Chen Z., Zhang, Q., Li, L., Li, X., & Lü, H. (2019). An improved carrier phase smoothing pseudorange algorithm with self-modeling of ionospheric delay variation (电离层延迟变化自模型化的载波相位平滑伪距算法). *Ce Hui Xue Bao (Acta Geodaetica et Cartographica Sinica)*, 48(9), 1107–1118. https://doi.org/10.11947/j.AGCS.2019.20180404

Cocke, WJ. (1996). Relativistic corrections for terrestrial clock synchronization (vol 16, pg 662, 1966). *Physics Today*, 49(4), 84–84.

Dai H., Shen, Q., Wang, C.-Z., Li, S.-L., Liu, W.-Y., Cai, W.-Q., Liao, S.-K., Ren, J.-G., Yin, J., Chen, Y.-A., Zhang, Q., Xu, F., Peng, C.-Z., & Pan, J.-W. (2020). Towards satellite-based quantum-secure time transfer. *Nature Physics*, 16(8), 848–852. https://doi.org/10.1038/s41567-020-0892-y

Doin, Lasserre, C., Peltzer, G., Cavalié, O., & Doubre, C. (2009). Corrections of stratified tropospheric delays in SAR interferometry: Validation with global atmospheric models. *Journal of Applied Geophysics*, 69(1), 35–50. https://doi.org/10.1016/j.jappgeo.2009.03.010

Elsobeiey, M.E. (2020). Characteristic differences between IGS final and ray-traced tropospheric delays and their impact on precise point positioning and tropospheric delay estimates. *GPS Solutions*, 24(4). https://doi.org/10.1007/s10291-020-01012-y

Hsu L., & Interdisciplinary Division of Aeronautical Aviation Engineering. (2017). Analysis and modeling GPS NLOS effect in highly urbanized area. 22(1), 1–12. https://doi.org/10.1007/s10291-017-0667-9

Li L., Kuang, C., Zhu, J., Chen, W., Chen, Y., Li, H., & Department of Land Surveying Geo-Informatics. (2012). 基于实时精密单点定位技术的暴雨短临预报. *地球物理学报 (Chinese Journal of Geophysics).* 55(4), 1129–1136. http://hdl.handle.net/10397/7200

Li Y., Chang, X., Yu, K., Wang, S., & Li, J. (2019). Estimation of snow depth using pseudorange and carrier phase observations of GNSS single-frequency signal. *GPS Solutions*, 23(4), 1–13. https://doi.org/10.1007/s10291-019-0912-5

Li X., Zhang, X., Ren, X., Fritsche, M., Wickert, J., & Schuh, H. (2015). Precise positioning with current multi-constellation Global Navigation Satellite Systems: GPS, GLONASS, Galileo and BeiDou. *Scientific Reports*, 5(1), 8328–8328. https://doi.org/10.1038/srep08328

Malik, & Gupta, K. (2021). Smart City: A new phase of sustainable development using fog computing and IoT. The IOP Conference Series. *Materials Science and Engineering*, 1022(1), 12093. https://doi.org/10.1088/1757-899X/1022/1/012093

Næsset, E. (1999). Point accuracy of combined pseudorange and carrier phase differential GPS under forest canopy. *Canadian Journal of Forest Research*, 29(5), 547–553. https://doi.org/10.1139/x99-021

Qian, Liu, H., Zhang, M., Shu, B., Xu, L., & Zhang, R. (2016). A geometry-based cycle slip detection and repair method with time-differenced carrier phase (TDCP) for a single frequency global position system (GPS) + BeiDou navigation satellite system (BDS) receiver. *Sensors (Basel, Switzerland)*, 16(12), 2064. https://doi.org/10.3390/s16122064

Rumora I., Sikirica, N., & Filjar, R. (2018). An Experimental Identification of Multipath Effect in GPS Positioning Error. *TransNav* (Gdynia, Poland), 12(1), 29–32. https://doi.org/10.12716/1001.12.01.02

Souza E.M., & Negri, T. T. (2017). First prospects in a new approach for structure monitoring from GPS multipath effect and wavelet spectrum. *Advances in Space Research*, 59(10), 2536–2547. https://doi.org/10.1016/j.asr.2017.02.043

Torres, G. (2017). Global positioning system (GPS) : performance, challenges and emerging technologies. *Nova Science Publishers*, Inc.

Wang, H. (2013). GPS navigation : principles and applications. *Science Press* ; Alpha Science International.

Wang Y., Zou, X., Deng, C., Tang, W., Li, Y., Zhang, Y., & Feng, J. (2020). A novel method for mitigating the GPS multipath effect based on a multi-point hemispherical grid model. *Remote Sensing* (Basel, Switzerland), 12(18), 3045. https://doi.org/10.3390/RS12183045

Yang F., Meng, X., Guo, J., Yuan, D., & Chen, M. (2021). Development and evaluation of the refined zenith tropospheric delay (ZTD) models. *Satellite Navigation*, 2(1), 1–9. https://doi.org/10.1186/s43020-021-00052-0

Yao, J., Yoon, S., Stressler, B., Hilla, S., & Schenewerk, M. (2021). GPS satellite clock estimation using global atomic clock network. *GPS Solutions*, 25(3). https://doi.org/10.1007/s10291-021-01145-8

Yu Z., Li, Z., & Wang, S. (2015). An Imaging Compensation Algorithm for Correcting the Impact of Tropospheric Delay on Spaceborne High-Resolution SAR. *IEEE Transactions on Geoscience and Remote Sensing*, 53(9), 4825–4836. https://doi.org/10.1109/TGRS.2015.2411261

--- THIS IS THE END OF LSGI3322 FINAL INDIVIDUAL REPORT. THANK YOU. ---