Calculating the User Position using the Data Received from GPS Satellites

ENGR 6461 – Avionics Navigation Systems – Term Project

Instructor: Luis Rodrigues
Department of Electrical and Computer Engineering

By
Parthkumar Viradiya
(Student ID: 40125453)

Table of Contents

Abstract

The main purpose of this project is to design and understand the process and algorithm used in calculating the user position using the data received from the visible GPS(Global Positioning Systems) satellites and to study the errors that might affect the position measurements. GPS data from the 6 satellites were provided which includes the ephemeris data for calculating satellite position, time of signal transmission, and parameters for satellite clock corrections. First, the satellites' position in the ECEF frame is computed from the ephemeris data. Then it is used to compute the user position and clock bias is computed without considering any errors. This provides a good estimation of user position in the ECEF frame and it is also a good initial user position later for a solution with considering all known errors. Then satellites clock bias, ionosphere, and troposphere errors are computed for all satellites. These errors are then incorporated into the calculation. And finally, the user position in the ECEF frame is computed.

1) Introduction and Basics of GPS

1.1 Basic principles of GPS

Nowadays, GPS has become a standard for navigation and its application ranges from navigating in an urban environment by the civilians to the flight tracking for both military and civil aircraft and drones. This trend is largely due to availability and reduction in costs of electronics in recent decades that made it possible to use GPS receivers in most of the modern devices to provide a 3D position in real-time [4].

It has far outcast the conventional radio navigation systems which use ground waves and sky waves propagation because GPS provides line of sight propagation which provides higher accuracy and cover the mostly all area of the world [7].

The basic principle of GPS is the same as any simple navigation system. GPS satellites transmit signal(which includes the ephemeris data for calculating satellite position, time of signal transmission, and parameters for satellite clock corrections) at time t1 to the connected GPS receiver which receives it at time t2 and distance between them can be found using the time it takes the signals to travel from the satellite to the receiver.

The receiver connected to only one satellite will only calculate the spherical range between both. Therefore, for a position in a 2D application where the user is assumed to be on the surface of the earth requires connection with a minimum of 3 satellites to accurately measure its position. If altitude is also desired, at least 4 satellites must be in connection with the receiver. To cover all parts of the world GPS constellation consists of more than 24 satellites orbiting around earth in different orbits and inclinations designed such a way that at any time and any location, user has at least 3 satellites connected to provide navigation.

However, the signals received from the satellites are prone to error as they travel through the ionosphere and tropospheric region of the atmosphere that adds delay to the speed of signal transmission. Signals are also affected by the error in measuring the accurate time of transmission and receiving as electronics equipment add bias and Radom noises to the signals. Due to these and other errors in the measurements, the measured range between satellites and receiver is called "pseudorange", which consists of all the errors, to distinguish it from the geometric or true range between them.

1.2 Navigation equations for GPS

Below are the fundamental equations for GPS pseudorange measurement. The measured pseudorange is equal to the true pseudorange plus various errors as shown below. For the detailed derivation and discussion of equations, the reader is referred to [6] and [3].

$$\rho^k(t) = r^k(t-\tau) - c\Delta b^k(t-\tau) + c\Delta b^u(t) + c\Delta I^k(t) + c\Delta T^k(t) + \varepsilon^k(t) \tag{1}$$

Where,

ρ^k = the pseudorange measurement for satellite k (meters)

r^k = the true or geometric range (meters)

c = speed of the light in the vacuum (m/sec)

Δb^k = satellite bias clock error (sec)

Δb^u = user bias clock error (sec)

ΔI^k = ionospheric excess delay (sec)

ΔT^k = tropospheric excess delay (sec)

ε^k = miscellaneous unmodeled errors (meters)

k = individual satellite identifier

u = user identifier

τ = time of transmission (sec)

t = signal reception time (sec)

Dropping the explicit time reference and grouping all satellite errors with one term reduces the equation (1) to below equation [6].

$$\rho_c^k = r^k + c\Delta b^u + \upsilon^k \quad (2)$$

Where,

ρ_c^k = corrected pseudorange (meters)

υ^k = single term for all satellite errors (meters)

Δb^u = the receiver clock offset, (sec)

The corrected pseudorange (ρ_c^k) in equation (2) is corrected for all known errors, such as ionosphere delay, tropospheric delay, satellite clock error etc., to minimize the value of υ^k as it has a direct effect on the quality of the solution.

r^k is the true range between user position $P^u = [x, y, z]$ and satellite position $P^k = [x^k, y^k, z^k]$ in the cartesian coordinate frame. The most common reference frame used is Earth-Centered-Earth-Fixed(ECEF) coordinate frame. Positioning data transmitted from the GPS satellites is also described satellite position in the ECEF frame. Detailed information about the ECEF frame can be found in [4].

$$r^k = \sqrt{(x^k - x) + (y^k - y) + (z^k - z)} = \left\| P^k - P^u \right\| \quad (3)$$

Substituting equation (3) into (2) and simplifying the total receiver clock error to $b = c\Delta b^u$ gives following equation [6].

$$\rho_c^k = \left\| P^k - P^u \right\| + b + \upsilon^k \quad (4)$$

The unknown in above equation are the user position P^u and the clock bias. Thus, given a minimum of four pseudorange measurement, it’s possible to estimate the four unknowns [6].

One solution for the above nonlinear equation is to linearize the pseudorange measurement from equation (4) around a rough guess of the initial receiver position and clock bias and then to iterate until the difference between the guess and the measurements approaches zero. Initial rough guess has very little effect on the solution of the equation and mostly the center of the earth is considered for the initial user position.

Assuming initial user position and clock bias as $P_0^u = [x_0, y_0, z_0]$ and b_0, the corresponding pseudorange equation is,

$$\rho_0^k = \left\| P^k - P_0^u \right\| + b_0 \tag{5}$$

Now, expressing true user position P^u as $P^u = P_0^u + \Delta P$ and the true receiver clock offset as $b = b_0 + \Delta b$. Putting these equations into equation (4) yields equation for $\Delta\rho^k$, which is the difference between the estimated and measured pseudoranges [6].

$$\Delta\rho^k = \rho_c^k - \rho_0^k \tag{6}$$

By putting equations (4) and (5) into (6) and applying Tylor series expansion, following equation can be obtained [6].

$$\Delta\rho = G \begin{bmatrix} \Delta P \\ \Delta b \end{bmatrix} + v \tag{7}$$

Where,

$$\Delta\rho^k = \begin{bmatrix} \Delta\rho^1 \\ \Delta\rho^2 \\ \cdots \\ \Delta\rho^k \end{bmatrix} \qquad G = \begin{bmatrix} (-L_{unit}^1)^T & 1 \\ (-L_{unit}^2)^T & 1 \\ \cdots & 1 \\ (-L_{unit}^k)^T & 1 \end{bmatrix} \qquad L_{unit}^k = \frac{(P^k - P_0^u)}{\left\| P^k - P_0^u \right\|}$$

G = geometry matrix

L_{unit}^k = line-of-sight unit vector between the estimated receiver location and satellite k.

Assuming errors to be zero means and uncorrelated, the Least square solution for the equation (7) is given by following equation [6].

$$\begin{bmatrix} \Delta P \\ \Delta b \end{bmatrix} = (G^T G)^{-1} G^T \Delta\rho \tag{8}$$

These equation yields result only when above problem is over-deterministic(k ≥ 4) [6]. Then the user position(P^u) and clock bias(b) can be found by solving $P^u = P_0^u + \Delta P$ and $b = b_0 + \Delta b$ respectively.

As discussed in 1.1, various errors greatly affect the result of the user position. References [3] and [4] provide an in-depth explanation of each error and model for calculating, which are used in this project. The summary of GPS error sources is provided in the table(1) below [7].

Table (1) Summary of GPS Error Sources (from Reference [7] table 6.3)

Typical Error Budget (in meters) for Standard GPS	
Per Satellite Accuracy	Errors (in meters)
Satellite clocks	1.5
Orbit Errors	2.5
Ionosphere	5.0
Troposphere	0.5
Receiver Noise	0.3
Multipath	0.6

In addition to various errors, the geometry of the satellites relative to user position greatly affects the quality of the pseudorange measurements and user position calculations. This quality can be observed using the DOP(delusion of precision) matrix, which relates satellite geometry and pseudorange measurement to the position error [6]. DOP matrix is defined as follow,

$$DOP = (G^T G)^{-1} \tag{9}$$

$$DOP = \begin{bmatrix} DOP_{xx}^2 & \cdots & \cdots & \cdots \\ \cdots & DOP_{yy}^2 & \cdots & \cdots \\ \cdots & \cdots & DOP_{zz}^2 & \cdots \\ \cdots & \cdots & \cdots & DOP_{tt}^2 \end{bmatrix} \tag{10}$$

DOP matrix is sometimes combined to form new DOPs, such as, Total Geometric DOP, Position DOP, Horizontal DOP, etc. [6].

$$GDOP = \sqrt{DOP_{xx}^2 + DOP_{yy}^2 + DOP_{zz}^2 + DOP_{tt}^2} \tag{11}$$

$$PDOP = \sqrt{DOP_{xx}^2 + DOP_{yy}^2 + DOP_{zz}^2} \tag{12}$$

$$HDOP = \sqrt{DOP_{xx}^2 + DOP_{yy}^2} \tag{13}$$

1.3 Position calculation algorithm

This section described the steps for the calculation of user position from the GPS data received from the satellites. Data from the satellites have three main sections. 1) ephemeris data, 2) ionosphere data and 3) pseudorange data. It is assumed that this data is already obtained before the below calculation is started.

1) Load the data into the program to inspect the data and to see what data is available for position calculations.
2) Calculate the positions of satellites (P^k) in the ECEF frame using ephemeris data, GPS time of the week from iono data, and assuming the initial time of signal transmission is 0.075 seconds. The algorithm to compute the satellite position from the ephemeris data can be found in reference [3] and [4]. MATLAB program for this algorithm is provided in appendix A.

3) Assume the user initial position in the ECEF frame P_0^u = [0; 0; 0] and user clock bias b_0 = 0.
4) Calculate the distance between user initial position to all visible satellites to form pseudorange vector ρ_0^k.
5) To compute $\Delta\rho^k$, subtract ρ_0^k from pseudorange obtained from data ρ_c^k as described in equation (6).
6) Calculate the unit vectors (L_{unit}^k) in the direction from user initial position to all visible satellites to form G matrix as described in the equation (7).
7) Apply the Least Square solution from the equation (8) to obtain the values of ΔP and Δb.
8) Set the new initial user position values to $P_0^u = P_0^u + \Delta P$ and new initial clock bias to $b_0 = \Delta b$. And repeat the steps from 4 to 7 until the solution converges and ΔP becomes zero.
9) Obtain the User Position in ECEF frame and Clock bias using the equations $P^u = P_0^u + \Delta P$ and $b = b_0 + \Delta b$ respectively. (Note: Above obtained user position and clock bias is without any errors considered, but it can be great for using again as initial position and clock bias when computing the solution with all known errors considered as some errors model requires initial position to close to the true position. So, for this reason initial position as [0; 0; 0] would not be good for estimating errors as it can be very far from the true user position.)
10) Now, for accounting all known errors in pseudorange measurement, set values obtained from the above to user initial position in ECEF frame $P_0^u = P^u$ and user clock bias $b_0 = b$
11) Calculate the satellites clock errors for all visible satellites using the ephemeris data, GPS time of the week from iono data, and assuming the initial time of signal transmission is 0.075 seconds. The algorithm to compute this can be found in Reference [3] and [1]. MATLAB program for this is provided in Appendix A.
12) Calculate the ionosphere error for all satellites using the iono data, latitude and longitude of the user location and elevation and azimuth data of satellites. Klobuchar model used for calculation of this error can be found in reference [3] and [4]. MATLAB code for this is provided in Appendix A.

13) Calculate the troposphere error for all satellites using the standard atmosphere values and elevation information of satellites. Hopfield's model used for calculation of this error can be found in reference [3] and [1]. MATLAB code for this is provided in Appendix A.

14) Using the above measurements of all error, compute the corrected pseudorange (ρ_c^k) as described in equation (6).

15) Using the new initial user position, clock bias and corrected pseudorange(ρ_c^k), repeat the steps from 4 to 8 to obtain the new user position in the ECEF frame and user clock bias with all known error considered.

Flowchart for above the algorithm is given in the figure (1) below.

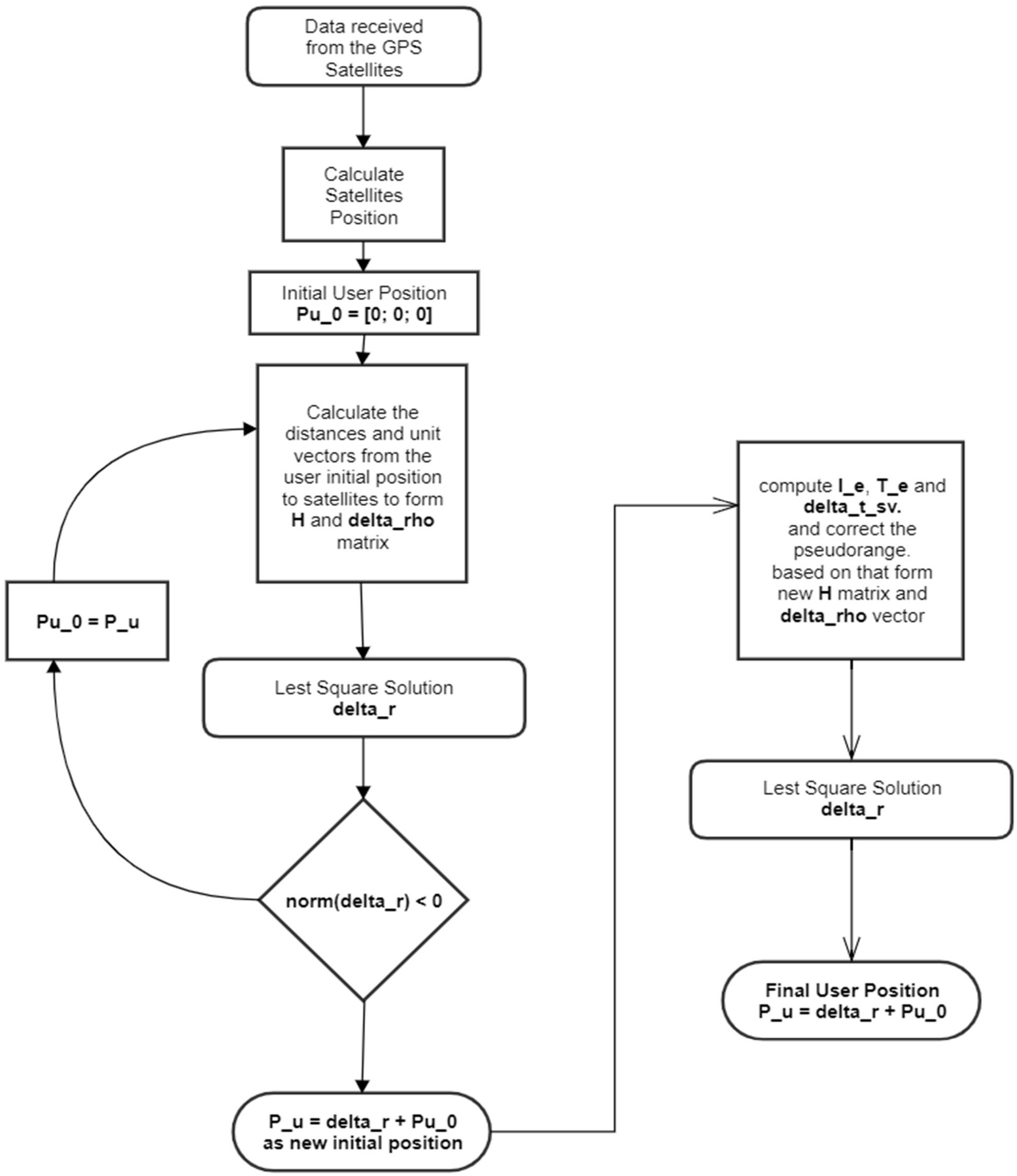

Figure 1: Flow chart for position calculation

2) Results

Below are the results obtained for calculation of the user position, DOP matrix and 1-sigma ellipse from the provided GPS data. MATLAB code used for the calculation is provided in Appendix A.

2.1 Assumptions

There are various assumptions are made during the calculations for user position. Below is the list of all assumptions made and the reason for them.

1) Initial user position and clock bias are assumed zero in computing the least square solution. Although this position is far from the true user position, this provides good estimation for user position as the solution is not very sensitive to initial guess [6].
2) In Troposphere error calculation, standard atmosphere values at sea level are used instead of actual measurements at the receiver antenna location. It is a reasonable assumption if the user is on the ground or even at very low altitude as these values remain similar.

2.2 User Position

Position of the user with all known errors considered is found to be on latitude 52.2913 deg, longitude 4.6756 deg and height 1.9191e+05 m which is near Hoofddorp, Netherlands as shown in the figure (2) below.

In ECEF frame it is $P^u = [x_u, y_u, z_u] = [3898.4, 318.8, 5025.3]$ Km.

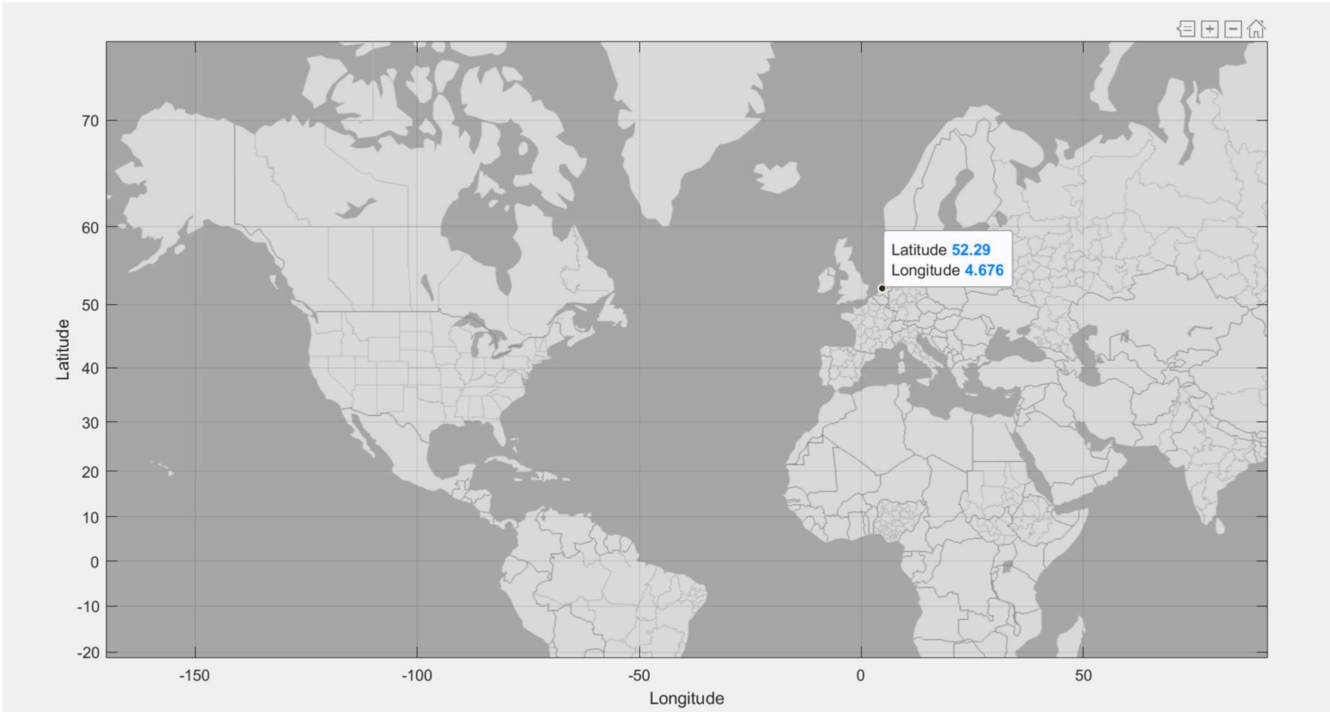

Figure 2: User Position in Latitude and Longitude on 2D earth map

Google map image of the user and positions of satellites with respect to user is shown on 3D earth map in the figure (3) and (4) respectively.

Figure 3: User Position on Google Map

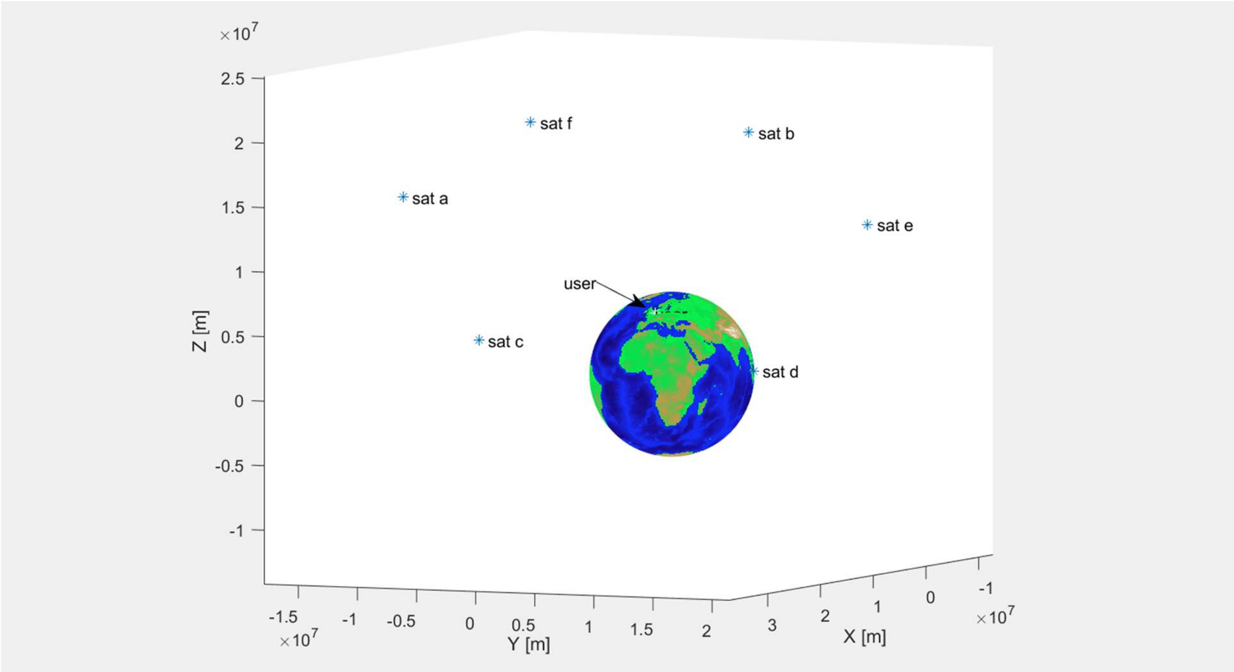

Figure 4: Positions of Satellites with respect to User

2.3 DOP parameters and 1-sigma ellipse

DOP matrix (in the ECEF frame) calculated from using MATLAB is shown below.

$$DOP = \begin{bmatrix} 10.65 & 2.25 & 4.95 & 8.46 \\ 2.25 & 0.98 & 1.18 & 1.91 \\ 4.95 & 1.18 & 3.86 & 4.50 \\ 8.46 & 1.91 & 4.50 & 7.11 \end{bmatrix}$$

From which various forms of DOPs are calculated from the equations (11) to (13)

GDOP = 4.7558 m

PDOP = 3.9375 m

HDOP = 3.4115 m

XDOP = 10.65 m

YDOP = 0.98 m

ZDOP = 3.86 m

TDOP = 7.11 m

For calculating 1-sigma ellipse in local east and north direction, DOP is needed in the ENU(East - North - Up) frame. Program for converting position from ECEF to ENU is provided in Appendix. 1-sigma is then calculated for east-north direction using DOP matrix in ENU frame. 1-sigma ellipse indicates that 68.2689492% of the time values of position in east-north plane will be within the boundary of ellipse [8]. Plot obtained form the MATLAB code for the 1-simgma is shown in figure (5). Values of semimajor axis, semiminor axis and alpha equal to 25.47 m, 11.22 m and 14.86 deg respectively.

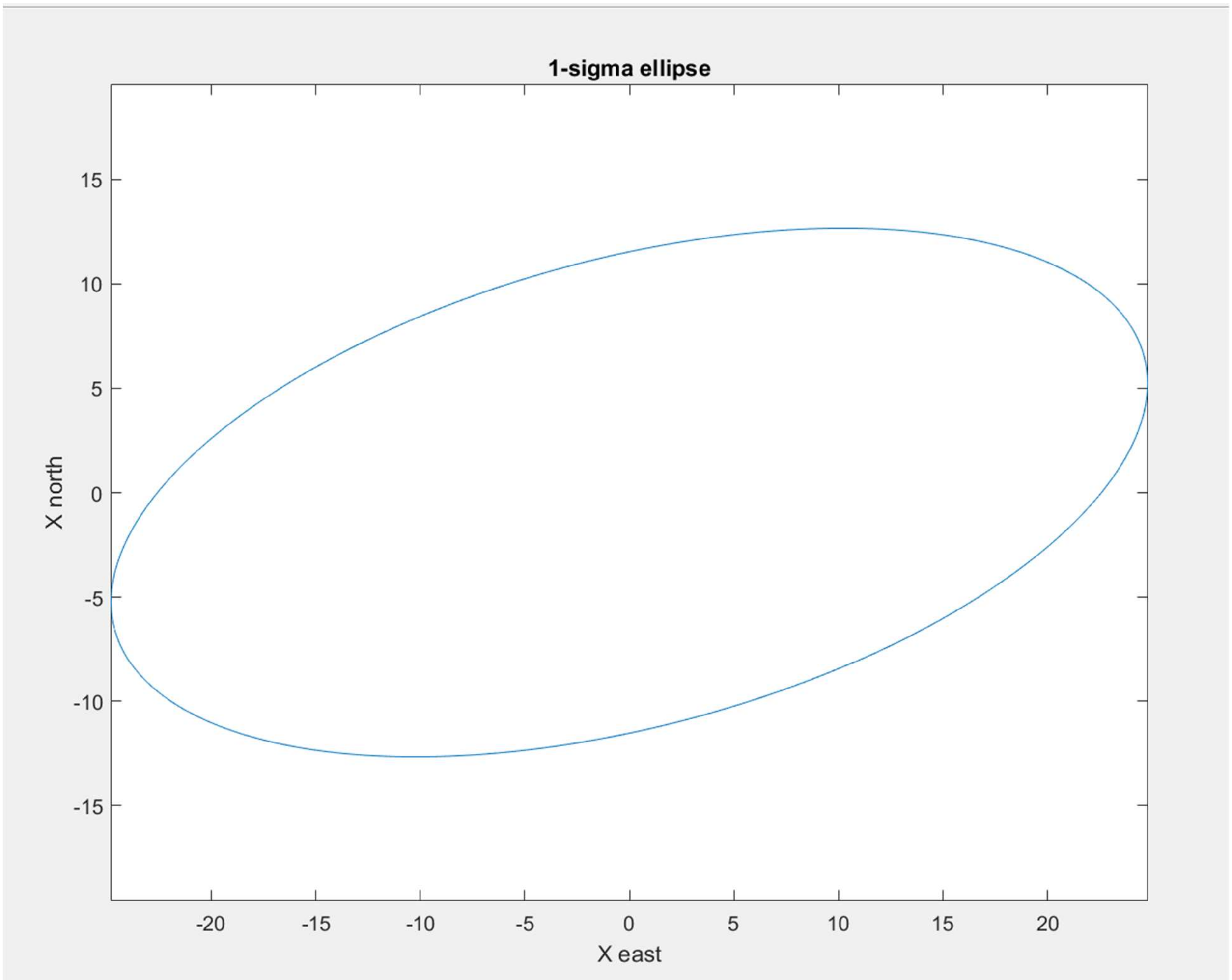

Figure 5: 1-sigma ellipse

References

1) https://moodle.concordia.ca/moodle/pluginfile.php/3799910/mod_resource/content/1/Project_Appendix.pdf
2) https://moodle.concordia.ca/moodle/pluginfile.php/3799909/mod_resource/content/1/ENGR6461_project.pdf
3) Parkinson, Spilker, Axelrad, Enge, "Global Positioning System: Theory and Applications", AIAA, 1996
4) A.V.Nebylv, J.Watson ,"Aerospace Navigation Systems".
5) https://ascelibrary.org/doi/pdf/10.1061/9780784411506.ap03
6) Scott Gleason, Demoz Gebre-Egzibher , "GNSS Applications and Methods".
7) R.P.G. Collinson, "Introduction of Avionics Systems", 2nd edition, Kluwer Academic Publishers 2003
8) https://en.wikipedia.org/wiki/Standard_deviation

Appendix. A: MATLAB code

Code can also be downloaded from https://github.com/parthp08/ENGR6461

A.1 main.m

```
%%% main script for calculating the user position from GPS data
clear all;
clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% user position calculation
%%% without considering any errors in the pseudoranges
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GPS data recieved from the satellite
load project_data.mat;

% obtain receiving GPS time of the week from the data
t_rcv = iono(1);
tau = 0.075; % intial estimation of transmission time between sat and user
c = 299792458; % m/sec % constant % speed of light

% Calculate the satellites position from the ephemeris data
P_a = sat_position(eph(:,1), t_rcv, tau);
P_b = sat_position(eph(:,2), t_rcv, tau);
P_c = sat_position(eph(:,3), t_rcv, tau);
P_d = sat_position(eph(:,4), t_rcv, tau);
P_e = sat_position(eph(:,5), t_rcv, tau);
P_f = sat_position(eph(:,6), t_rcv, tau);

P_sat_arr = [P_a, P_b, P_c, P_d, P_e, P_f]; % sat pos array

% inital user position and clock bias assumption
% at the centre of the earth in ECEF frame
Pu_0 = [0; 0; 0];
bu_0 = 0;

% Calculate the Inital Good Estimate of the inital Position
% to use later for error correction

% just to get loop started
delta_r = [100;100;100];
while norm(delta_r(1:3)) > 1e-5
```

```matlab
% Least Square Solution
[delta_r, ~] = least_square_sol(P_sat_arr, Pu_0, pr);

% calulate user position and user clock bias
Pu_0 = delta_r(1:3) + Pu_0;
bu_0 = delta_r(4);
end

% user position and clock error % without any errors considered
Pu = delta_r(1:3) + Pu_0;
bu = delta_r(4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% user position calculation
%%% with all available errors considered
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% use improved initial estimation for incorporating errors into the
% mesurements
Pu_0 = Pu;
bu_0 = bu;

%%% corrections for pseudorange (for all satellites)
% satellite clock corrections
delta_t_sv = [];
delta_t_sv(1) = sat_clock_error(eph(:,1), t_rcv, tau);
delta_t_sv(2) = sat_clock_error(eph(:,2), t_rcv, tau);
delta_t_sv(3) = sat_clock_error(eph(:,3), t_rcv, tau);
delta_t_sv(4) = sat_clock_error(eph(:,4), t_rcv, tau);
delta_t_sv(5) = sat_clock_error(eph(:,5), t_rcv, tau);
delta_t_sv(6) = sat_clock_error(eph(:,6), t_rcv, tau);

% ionosphere and troposphere errors
T_amb = 15;  % deg C
P_amb = 101.325; % kPa
P_vap = 0.85;    % kPa
[az1, el1] = sat_az_el(P_sat_arr(:,1), Pu_0);
[az2, el2] = sat_az_el(P_sat_arr(:,2), Pu_0);
[az3, el3] = sat_az_el(P_sat_arr(:,3), Pu_0);
[az4, el4] = sat_az_el(P_sat_arr(:,4), Pu_0);
[az5, el5] = sat_az_el(P_sat_arr(:,5), Pu_0);
[az6, el6] = sat_az_el(P_sat_arr(:,6), Pu_0);
[lat_u, lon_u, ~] = ECEF2WGS(Pu_0, 0);
I_e = [];
```

```
I_e(1) = iono_error(iono, lat_u, lon_u, el1, az1);
I_e(2) = iono_error(iono, lat_u, lon_u, el2, az2);
I_e(3) = iono_error(iono, lat_u, lon_u, el3, az3);
I_e(4) = iono_error(iono, lat_u, lon_u, el4, az4);
I_e(5) = iono_error(iono, lat_u, lon_u, el5, az5);
I_e(6) = iono_error(iono, lat_u, lon_u, el6, az6);
T_e = [];
T_e(1) = tropo_error(T_amb, P_amb, P_vap, el1);
T_e(2) = tropo_error(T_amb, P_amb, P_vap, el2);
T_e(3) = tropo_error(T_amb, P_amb, P_vap, el3);
T_e(4) = tropo_error(T_amb, P_amb, P_vap, el4);
T_e(5) = tropo_error(T_amb, P_amb, P_vap, el5);
T_e(6) = tropo_error(T_amb, P_amb, P_vap, el6);

% apply correction
pr_corrected = pr' + (c.*(delta_t_sv'))- bu_0 - I_e' - T_e';
pr = pr_corrected';

%%% Least Square Solution
[delta_r, DOP] = least_square_sol(P_sat_arr, Pu_0, pr);

% user position and clock error % with errors considered
Pu = delta_r(1:3) + Pu_0;
bu = delta_r(4);

% user position in Longitude and Lattitude
[lat, lon, h] = ECEF2WGS(Pu, 1);  % 1 = units in deg

% DOP Matrix and its various forms
DOP;
GDOP = sqrt(DOP(1,1)+DOP(2,2)+DOP(3,3)+DOP(4,4)); % geometric DOP
PDOP = sqrt(DOP(1,1)+DOP(2,2)+DOP(3,3)); % Position DOP
HDOP = sqrt(DOP(1,1)+DOP(2,2)); % Horizontal DOP

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% printing variables for easiness
disp("user position in ECEF frame");
disp(Pu);
disp("user position in Latitude and Longitude");
disp([lat; lon]);
disp("GDOP = ");
disp(GDOP);
disp("PDOP = ");
disp(PDOP);
disp("HDOP = ");
```

```
disp(HDOP);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plotting the user and satellites positions on ECEF frame
sat_pos_x = [P_a(1), P_b(1), P_c(1), P_d(1), P_e(1), P_f(1)];
sat_pos_y = [P_a(2), P_b(2), P_c(2), P_d(2), P_e(2), P_f(2)];
sat_pos_z = [P_a(3), P_b(3), P_c(3), P_d(3), P_e(3), P_f(3)];
figure();
earth_sphere('m'); % earth 3d map
hold on;
scatter3(sat_pos_x, sat_pos_y, sat_pos_z, '*');
scatter3(Pu(1), Pu(2), Pu(3), '*', 'w');
text(sat_pos_x(1),sat_pos_y(1),sat_pos_z(1),"  sat a");
text(sat_pos_x(2),sat_pos_y(2),sat_pos_z(2),"  sat b");
text(sat_pos_x(3),sat_pos_y(3),sat_pos_z(3),"  sat c");
text(sat_pos_x(4),sat_pos_y(4),sat_pos_z(4),"  sat d");
text(sat_pos_x(5),sat_pos_y(5),sat_pos_z(5),"  sat e");
text(sat_pos_x(6),sat_pos_y(6),sat_pos_z(6),"  sat f");
text(Pu(1),Pu(2),Pu(3),"user");

figure();
geoscatter(lat,lon, '*', 'r'); % on 2D map
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.2 sat_position.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%Calculate the Position of Satellite in ECEF frame given the ephemeris
%%% data of the satellite and GPS receiving time of the week
%
%%% references
% -------------
%[1]
https://moodle.concordia.ca/moodle/pluginfile.php/3799910/mod_resource/content/1/Project_Appendix.pdf
%[2]
https://moodle.concordia.ca/moodle/pluginfile.php/3799909/mod_resource/content/1/ENGR6461_project.pdf
%[3] 'Global Positioning System: Theory and Applications'; edited by
%     Parkinson, Spilker, Axelrad, Enge; AIAA; 1996
%[4] 'Aerospace Navigation Systems'; edited by A.V.Nebylv, J.Watson
%[5] https://ascelibrary.org/doi/pdf/10.1061/9780784411506.ap03
%[6] GNSS Applications and Methods; Scott Gleason, Demoz Gebre-Egzibher
%
```

```
%%% inputs
% ---------
% eph_data : array, shape(21,1), ephemeris data for a satellite
% t_rcv : float, receiving GPS time of the week, seconds
% tau: float, transmission time between sat and user, seconds
%
%%% outputs
% ----------
%P = [x; y; z] : array, size(3,1), Position of Satellte in ECEF frame,
%               meters
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function P = sat_position(eph_data, t_rcv, tau)

pi = 3.1415926535898;    % GPS standard value for pi

% eph parameters (Ref [2], page 2)
cr_sin = eph_data(1);
delta_n = eph_data(2) * pi;
M_0 = eph_data(3) * pi;
cu_cos = eph_data(4);
e = eph_data(5);
cu_sin = eph_data(6);
sqrta = eph_data(7);
t_0e = eph_data(8);
ci_cos = eph_data(9);
omg0 = eph_data(10) * pi;
ci_sin = eph_data(11);
i_0 = eph_data(12) * pi;
cr_cos = eph_data(13);
w = eph_data(14) * pi;
omg_dot = eph_data(15) * pi;
i_dot = eph_data(16) * pi;
%tgd = eph_data(17);
%toc = eph_data(18);
%af2 = eph_data(19);
%af1 = eph_data(20);
%af0 = eph_data(21);

%%% Satellite Position Calculation Algorithm (Ref[1] page 3, Ref [3] page 138)
% constants
mu = 3.986005e14;    % m^3 / s^2 % Earth universal gravitational parameter
omg_e_dot = 7.2921151467e-5; % Earth Rotation Rate
A = sqrta^2;    % semimajor axis
```

```
n0 = sqrt(mu / A^3); % computed mean motion % rad/sec

%tau = 0.075;
t = t_rcv - tau; % t == t_tr
t_k = t - (t_0e);  % time from ephemeris reference time(t_0e)
% from the footnote(page 138 Ref [3])
% accounting for beginning or end of week crossovers
if t_k > 302400  % seconds
        t_k = t_k - 604800;
end
if t_k < -302400
        t_k = t_k + 604800;
end

n = n0 + delta_n;    % corrected mean motion

M_k = M_0 + n*t_k;   % Mean anomaly

% (Ref [1] page 3(Note))
% E_k = M_k + e*np.sin(E_k)    % Kepler's Equ for the eccentric anomaly E_k % rad
E_old = M_k;
E_new = M_k + e*sin(E_old);
while E_new - E_old >= 1e-8  % stop iteration when E_new - E_old is less than 1e-8
        E_old = E_new;
        E_new = M_k + e*sin(E_old);
end
E_k = E_new; % Essentric anomaly at time t_k

% True anomaly as a function of E_k at time t_k
v_k = atan2(sqrt(1-e^2)*sin(E_k), cos(E_k)-e);

E_k = acos((e+cos(v_k)) / (1 + e*cos(v_k)));    % Eccentric anomaly

phi_k = v_k + w;   % Argument of latitude

% Second harmonic perturbations
phi_k_2 = 2*phi_k;
phi_k_2_cos = cos(phi_k_2);
phi_k_2_sin = sin(phi_k_2);
delta_u_k = cu_sin*phi_k_2_sin + cu_cos*phi_k_2_cos; % Argument of lattitide correction
delta_r_k = cr_sin*phi_k_2_sin + cr_cos*phi_k_2_cos; % Argument of radius correction
delta_i_k = ci_sin*phi_k_2_sin + ci_cos*phi_k_2_cos; % Argument of inclination correction

u_k = phi_k + delta_u_k; % Corrected argument of lattitude
r_k = A*(1 - e*cos(E_k)) + delta_r_k; % Corrected argument of radius
```

```
i_k = i_0 + delta_i_k + i_dot*t_k; % Corrected argument of inclination

% Satellite position in orbital plane (x_k_dash, y_k_dash)
x_k_dash = r_k * cos(u_k);
y_k_dash = r_k * sin(u_k);

% Corrected longitude of ascending node  % accounting for earth rotation rate
omg_k = omg0 + (omg_dot - omg_e_dot)*t_k - omg_e_dot*t_0e;

% Satellite Position in ECEF coordinates (x_k, y_k, z_k)
x_k = x_k_dash*cos(omg_k) - y_k_dash*cos(i_k)*sin(omg_k);
y_k = x_k_dash*sin(omg_k) + y_k_dash*cos(i_k)*cos(omg_k);
z_k = y_k_dash*sin(i_k);

P = [x_k; y_k; z_k]; % satellite position

end
```

A.3 least_square_sol.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Least Square solution of GPS pseudorange equation
%
%%% references
% -------------
%[1] 'GNSS Applications and Methods', Scott Gleason, Demoz Gebre-Egzibher
%
%%% inputs
% ----------
% P_sat_arr : array, size(3,*), array of satellites' position in ECEF
%             frame, meters
% Pu_0 : array, size(3,1), initial user position guess, meters
% pr_arr_data : array, size(1,*), psedorange data received from satellites,
%               meters
%
%%% outputs
%-----------
% delta_r : array, size(4,1),[delta_x, delta_y, delta_z, c*delta_b], meters
% DOP: array, size(4,4), DOP matrix
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [delta_r, DOP] = least_square_sol(P_sat_arr, Pu_0, pr_arr_data)

% number of satellite
```

```
n = size(P_sat_arr, 2); % 2 for size of columns

% vector dimension
p = size(P_sat_arr(:,1), 1); % 1 for size of rows

% initialize H matrix
H = ones(n, p+1);

% initialze rho_0
rho_0 = ones(n,1);

% fill rho_0 vector and H matrix
for i = 1:n
        H(i,1:p) = (-unit_vector(Pu_0, P_sat_arr(:,i)));
        rho_0(i,1) = norm(P_sat_arr(:,i) - Pu_0);
end

H_T = H';
DOP = inv(H_T*H);
H_terms = (DOP)*(H_T);

delta_rho = (pr_arr_data)' - rho_0;

% delta_r = P_u - Pu_0
delta_r = H_terms*delta_rho;

end
```

A.4 sat_clock_error.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Compute satellite clock correction
%
%%% references
% -------------
%[1]
https://moodle.concordia.ca/moodle/pluginfile.php/3799910/mod_resource/content/1/Project_Ap
pendix.pdf
%[2]
https://moodle.concordia.ca/moodle/pluginfile.php/3799909/mod_resource/content/1/ENGR646
1_project.pdf
%[3] 'Global Positioning System: Theory and Applications'; edited by
%     Parkinson, Spilker, Axelrad, Enge; AIAA; 1996
%[4] 'Aerospace Navigation Systems'; edited by A.V.Nebylv, J.Watson
%[5] https://ascelibrary.org/doi/pdf/10.1061/9780784411506.ap03
%
```

```
%%% inputs
% ----------
% eph_data : array, size(21,1), ephemeris data received from the GPS
%           satellites
% t_rcv : flot, float, receiving GPS time of the week, seconds
% tau : float, transmission time between sat and user, seconds
%
%%% outputs
%-----------
% delta_t_sv : float, satellite clock correction, seconds
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function delta_t_sv = sat_clock_error(eph_data, t_rcv, tau)

% eph parameters (Ref [2], page 2)
%cr_sin = eph_data(1);
delta_n = eph_data(2) * pi;
M_0 = eph_data(3) * pi;
%cu_cos = eph_data(4);
e = eph_data(5);
%cu_sin = eph_data(6);
sqrta = eph_data(7);
t_0e = eph_data(8);
%ci_cos = eph_data(9);
%omg0 = eph_data(10) * pi;
%ci_sin = eph_data(11);
%i_0 = eph_data(12) * pi;
%cr_cos = eph_data(13);
%w = eph_data(14) * pi;
%omg_dot = eph_data(15) * pi;
%i_dot = eph_data(16) * pi;
tgd = eph_data(17);
toc = eph_data(18);
af2 = eph_data(19);
af1 = eph_data(20);
af0 = eph_data(21);

F = -4.442807633e-10;    % relativistic constant

% inividual satellite time corrected to GPS system time 't'
% trasmission time  % Ref [2] page 1
% tau = 0.075 % initial estimation    % tau = t_rcv - t_tr
t_tr = t_rcv - tau; % t == t_tr
```

```
t_k = t_tr - t_0e;  % time from ephemeris reference time(t_0e)
% account for end of the week crossover correction % Ref [1] page 2
if t_k > 302400  % seconds
        t_k = t_k - 604800;
end
if t_k < -302400
        t_k = t_k + 604800;
end

mu = 3.986005e14;    % m^3 / s^2 % Earth universal gravitational parameter
a = sqrta^2;    % semimajor axis

n0 = sqrt(mu / a^3); % computed mean motion % rad/sec

n = n0 + delta_n;    % corrected mean motion

M_k = M_0 + n*t_k;   % Mean anomaly

% (Ref [1] page 3(Note))
% E_k = M_k + e*np.sin(E_k)    % Kepler's Equ for the eccentric anomaly E_k % rad
E_old = M_k;
E_new = M_k + e*sin(E_old);
while E_new - E_old >= 1e-8  % stop iteration when E_new - E_old is less than 1e-8
        E_old = E_new;
        E_new = M_k + e*sin(E_old);
end
E_k = E_new; % Essentric anomaly at time t_k

% Relativistic correction term
T_rel = F*e*sqrta*sin(E_k);   % sec

% Ref [1] page 1 and Ref [5]
delta_t_sv = af0 + af1*(t_tr - toc) + af2*((t_tr - toc)^2) + T_rel - tgd;
% t = t_tr - delta_t_sv

End
```

A.5 iono_error.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Compute the ionospheric errors in pseudorange mesurement
%
%%% references
% -------------
```

```
%[1]
https://moodle.concordia.ca/moodle/pluginfile.php/3799910/mod_resource/content/1/Project_Ap
pendix.pdf
%[2]
https://moodle.concordia.ca/moodle/pluginfile.php/3799909/mod_resource/content/1/ENGR646
1_project.pdf
%[3] 'Global Positioning System: Theory and Applications'; edited by
%     Parkinson, Spilker, Axelrad, Enge; AIAA; 1996
%[4] 'Aerospace Navigation Systems'; edited by A.V.Nebylv, J.Watson
%
%%% inputs
% ----------
% iono_data : array, size(9,1), data with GPS time for measurement and
%             eight parameters for ionosphere delay estimation
% lat_u : float, latitude of user, rad
% lon_u : float, longitude of user, rad
% E_s : float, elevation of sat from the user local tangent plane, rad
% A_s : float, azimuth of sat from the user local tangent plane, rad
%
%%% outputs
%-----------
% I_e : float, ionospheric delay, meters
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function I_e = iono_error(iono_data, lat_u, lon_u, E_s, A_s)

pi = 3.1415926535898;    % GPS standard value for pi

phi_u = lat_u;
lamda_u = lon_u;

% Ref [2] page 2
tow = iono_data(1);
a0 = iono_data(2);
a1 = iono_data(3);
a2 = iono_data(4);
a3 = iono_data(5);
b0 = iono_data(6);
b1 = iono_data(7);
b2 = iono_data(8);
b3 = iono_data(9);

%%% Klobuchar model for ionosphere  % Ref [1] page 4-6 and Ref [2] page 146-147 and Ref
[4] page 78-79
```

```
% Earth-centered angle between user and IP(ionosphere point % Ref[4] page 78)
psi_ip = (0.0137/(0.11 + E_s)) - 0.022; % semicircles
psi_ip = pi * psi_ip; % to convert to rad

% subionospheric latitide
phi_ip = phi_u + psi_ip*cos(A_s);    % semicircles
phi_ip = pi * phi_ip; % to convert to rad
if phi_ip > 0.416
            phi_ip = 0.416;
end
if phi_ip < -0.416
            phi_ip = -0.416;
end

% subionospheric longitude
lamda_ip = lamda_u + psi_ip*sin(A_s)/cos(phi_ip);  % semicircles
lamda_ip = pi * lamda_ip; % to convert to rad

% geomagnetic lattitude of the subionospheric location looking toward each GPS satellite
phi_m = phi_ip + 0.064*cos(lamda_ip-1.617);   % semicircles
phi_m = pi * phi_m; % to convert to rad

% local time(t) at the subionospheric point
t_gps = mod(tow, 86400);  % Appendix % moodle % second last page  % written note
t_ip = 4.32e4*lamda_ip + t_gps; % seconds
if t_ip > 86400
            t_ip = t_ip - 86400;
end
if t_ip < 0
            t_ip = t_ip + 86400;
end

% obliquity factor / slant factor (dimensionless)
% to convert to slant time delay
F_ip = 1.0 + 16.0*((0.53-E_s)^3);

% Amplitude % in seconds
AM_ip = a0 + a1*(phi_m^1) + a2*(phi_m^2) + a3*(phi_m^3);
if AM_ip < 0
            AM_ip = 0;
end

% Period % in seconds
PER_ip = b0 + b1*(phi_m^1) + b2*(phi_m^2) + b3*(phi_m^3);
if PER_ip < 72000
            PER_ip = 72000;
```

```
end

% Phase % in rad
X_ip = 2*pi*(t_ip - 50400) / PER_ip;

% Ionospheric Time delay (T_iono)  % in seconds
if abs(X_ip) >= 1.57
        T_iono = F_ip*5e-9;
else   % for abs(X_ip) < 1.57
        T_iono = F_ip*(5e-9 + AM_ip*(1 - (X_ip^2)/2 + (X_ip^4)/24));
end

% Ionosphere Error % in meters
c = 299792458;   % speed of light(GPS constant) % m/s
I_e = c * T_iono;    % m

End
```

A.6 tropo_error.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Compute the tropospheric errors in pseudorange mesurement
%
%%% references
% -------------
%[1]
https://moodle.concordia.ca/moodle/pluginfile.php/3799910/mod_resource/content/1/Project_Appendix.pdf
%[2] 'Aerospace Navigation Systems'; edited by A.V.Nebylv, J.Watson
%
%%% inputs
% ----------
% T_amb : float, ambient air tempreature, deg celsius
% P_amb : float, ambient air pressure, kPa
% P_vap : float, ambient vapour pressure, kPa
% E_s : float, elevation of sat from the user local tangent plane, rad
%
%%% outputs
%-----------
% T_e : float, tropospheric delay, meters
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function T_e = tropo_error(T_amb, P_amb, P_vap, E_s)
```

```
% zenith delay of the dry component
Kd = (1.55208e-4*P_amb*(40136.0 + 148.72*T_amb)) / (T_amb+273.16);   % m

% zenith delay of the wet component
Kw = (-0.282*P_vap/(T_amb+273.16)) + (8307.2*P_vap/((T_amb+273.16)^2)); % m

% Troposhperic delay correction (T_e), m
temp1 = sqrt(E_s*E_s + 1.904e-3);
temp2 = sqrt(E_s*E_s + 0.6854e-3);
T_e = (Kd/sin(temp1)) + (Kw/sin(temp2));   % m

End
```

A.7 unit_vector.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% unit vector between two points in caertesian frame
%
%%% inputs
% ----------
% X : array, size(3,1), point in cartesian frame
% Y : array, size(3,1), point in cartesian frame
%
%%% outputs
%-----------
% e : array, size(3,1), unit vector from X to Y
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function e = unit_vector(X,Y)
% unit vector from X to Y
e = (Y - X)./norm(Y - X);
end
```

A.8 ECEF2WGS.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Converts vector ocordinates from ECEF frame to WGS84 frame
%
%%% references
% -------------
%[1] 'Aerospace Navigation Systems'; edited by A.V.Nebylv, J.Watson
%
%%% inputs
```

```
% ----------
% P : array, size(3,1), vector in ECEF Frame, meters
% deg : int, deg=1 for units in deg
%          deg=0 for units in rad
%
%%% outputs
%-----------
% phi : float, latitide, deg or rad
% lambda : float, longitude, deg or rad
% h : altitude, meters
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [phi, lambda, h] = ECEF2WGS(P, deg)

x = P(1);
y = P(2);
z = P(3);
% WGS84 Model
a = 6378137.0;  % meters % semi-major axis
b = 6356752.314;  % meters % semi-minor axis
e = sqrt(1 - (b^2 / a^2));% eccentricity

%%% Step 1
if (x^2 + y^2) == 0  % if True algorithm stops here
        if z > 0
            phi = pi/2;
            h = z - b;
        else % z < 0
            phi = -pi/2;
            h = -z - b;
        end
        % return "Longitude cannot be defined", phi, h
        lambda = 0;

else

%%% Step 2
% assuming x^2 + y^2 > 0
if x == 0
        lambda_ = pi/2;
elseif x > 0
        lambda_ = atan(y/x);
elseif x < 0 && y >= 0
        lambda_ = pi + atan(y/x);
```

```
elseif x < 0 && y < 0
            lambda_ = -pi + atan(y/x);
end

%%% Step 3
% sensitivity levels
e_phi = 0.01;
e_h = 0.01;

% starting values
phi_s = pi/2;
d_s = a / sqrt(1 - e^2);
h_s = 0;

%%% Step 4
% computing new values related to the latitude
phi_n = atan2(z*(d_s+h_s) , ((sqrt(x^2 + y^2))*((d_s*(1 - e^2))+h_s)));
d_n = a / sqrt(1 - ((e*sin(phi_n))^2));

%%% Step 5
% computing new altitude value
if phi_n <= pi/4 || phi_n >= -pi/4
            h_n = (sqrt(x^2 + y^2)/cos(phi_n)) - d_n;
end
if phi_n > pi/4 || phi_n < -pi/4
            h_n = (z/sin(phi_n)) - (d_n*(1 - e^2));
end

%%% Step 6
% Evaluating the convergence of the obtained value for phi
while abs(phi_n - phi_s) >= e_phi && abs(h_n - h_s) >= e_h % if not converged
            phi_s = phi_n;
            d_s = d_n;
            h_s = h_n;

            %%% Step 4
            % computing new values related to the latitude
            phi_n = atan2(z*(d_s+h_s), (sqrt(x^2 + y^2))*(d_s*(1 - e^2) + h_s));
            d_n = a / sqrt(1 - ((e*sin(phi_n))^2));

            %%% Step 5
            % computing new altitude value
            if phi_n <= pi/4 || phi_n >= -pi/4
                h_n = (sqrt(x^2 + y^2)/cos(phi_n)) - d_n;
            end
            if phi_n > pi/4 || phi_n < -pi/4
```

```
                    h_n = (z/sin(phi_n)) - (d_n*(1 - e^2));
                end
end

% when converged
% abs(phi_n - phi_s) >= e_phi and abs(h_n - h_s) >= e_h
lambda = lambda_;   % in radians
phi = phi_n;
h = h_n;

if deg == 1
                lambda = rad2deg(lambda_);   % in degrees
                phi = rad2deg(phi_n);
                h = rad2deg(h_n);
end

end
end
```

A.9 sat_az_el.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Compute Azimuth and Elevation of satellite from the user local tangnent
%%% plane.
%
%%% references
% -------------
%[1] 'Global Positioning System: Signals, Measurements, and
%     Performance', Pratap Misra, Per Enge.
%[2] 'Aerospace Navigation Systems'; edited by A.V.Nebylv, J.Watson
%
%%% inputs
% ----------
% P_sat : array, size(3,1), satellite position in ECEF frame, meters
% P_u : array, size(3,1), user position in ECEF frame, meters
%
%%% outputs
%-----------
% az : float, azimuth of satellite from the user local tangent plane, rad
% el : float, elevation of satellite from the user local tangent plane, rad
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [az, el] = sat_az_el(P_sat,P_u)
```

```
xs = P_sat(1);
ys = P_sat(2);
zs = P_sat(3);

xu = P_u(1);
yu = P_u(2);
zu = P_u(3);

[lat, lon, ~] = ECEF2WGS(P_u, 0);

% vector from user to sat
X = [xs-xu; ys-yu; zs-zu];

% Transformation from the ECEF to ENU
lon_s = sin(lon);
lon_c = cos(lon);
lat_s = sin(lat);
lat_c = cos(lat);
R_L = [
        -lon_s, lon_c, 0;
        -lat_s*lon_c, -lat_s*lon_s, lat_c;
        lat_c*lon_c, lat_c*lon_s, lat_s
];

% X_L % representation of X in ENU
X_L = R_L*X;
x_e = X_L(1); % east position
x_n = X_L(2); % north position
x_u = X_L(3); % up position

az = atan2(x_e, x_n);

el = asin(x_u / sqrt(x_e^2 + x_n^2 + x_u^2));

end
```

A.10 ECEF2ENU.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Converts vector ocordinates from ECEF frame to WGS84 frame
%
%%% references
% -------------
%[1] 'Global Positioning System: Signals, Measurements, and
%     Performance', Pratap Misra, Per Enge.
%
```

```
%%% inputs
% ----------
% P : array, size(3,1), vector in ECEF Frame, meters
%
%%% outputs
%-----------
% X_L : array, size(3,1), vector in ENU Frame, meters
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function X_L = ECEF2ENU(P)

  [lat, lon, ~] = ECEF2WGS(P, 0);

  % Transformation from the ECEF to ENU
  lon_s = sin(lon);
  lon_c = cos(lon);
  lat_s = sin(lat);
  lat_c = cos(lat);
  R_L = [
    -lon_s, lon_c, 0;
    -lat_s*lon_c, -lat_s*lon_s, lat_c;
    lat_c*lon_c, lat_c*lon_s, lat_s
  ];

  % X_L % representation of X in ENU
  X_L = R_L*P;
  end
```

A.11 sigma_ellipse.m

```
%%% calculation for 1-sigma error
%%% have to compte the DOP matrix in ENU frame
%%% Note: have to run main.m file before running this file
%
%%% References
% -------------
%[1] 'Global Positioning System: Theory and Applications'; edited by
%    Parkinson, Spilker, Axelrad, Enge; AIAA; 1996
%[2] 'Aerospace Navigation Systems'; edited by A.V.Nebylv, J.Watson
%[3] https://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-
matrix/
%[4] https://stattrek.com/online-calculator/chi-square.aspx

% use Pu and P_sat_arr from main files and convert them to ENU Frame
```

```
Pu_enu = ECEF2ENU(Pu);
P_sat_arr_enu = ECEF2ENU(P_sat_arr);

%%%%%% construct H matrix
% number of satellite
n = size(P_sat_arr_enu, 2); % 2 for size of columns
% vector dimension
p = size(P_sat_arr_enu(:,1), 1); % 1 for size of rows
% initialize H matrix
H = ones(n, p+1);
% fill H matrix
for i = 1:n
   H(i,1:p) = (-unit_vector(Pu_enu, P_sat_arr_enu(:,i)));
end

%%%%%%% calculate DOP in ENU frame
H_T = H';
DOP_enu = inv(H_T*H);
GDOP_enu = sqrt(DOP_enu(1,1)+DOP_enu(2,2)+DOP_enu(3,3)+DOP_enu(4,4)); % geometric
DOP
PDOP_enu = sqrt(DOP_enu(1,1)+DOP_enu(2,2)+DOP_enu(3,3)); % Position DOP
HDOP_enu = sqrt(DOP_enu(1,1)+DOP_enu(2,2)); % Horizontal DOP

UERE = 5.3; % page 481 Ref[1] % 1-sigma user equivalent error

% for east and north plane error
ENDOP = DOP_enu(1:2,1:2);

% covarience matrix
K = (UERE^2)*ENDOP; % it is correlated  %%%%%%%%% put UERE^2
sigma_xe = sqrt(K(1,1));
sigma_xn = sqrt(K(2,2));

S = 2.28; % scale of the ellipse for 1-sigma (68.2% confidence)
[V,E] = eig(ENDOP);
a = sigma_xe*sqrt(S); % semi-major axis % in north direction
b = sigma_xn*sqrt(S); % semi-minor axis % in east direction
theta = atan(V(2,2)/V(2,1)); % orientation of the ellipse

xe0=0; % x0,y0 ellipse centre coordinates
xn0=0;
t=-pi:0.01:pi;
XE=xe0+a*cos(t);
XN=xn0+b*sin(t);
% rotate ellipse to theta angle
XE_rot = XE*cos(theta) + -XN*sin(theta);
```

```
XN_rot = XE*(sin(theta)) + XN*cos(theta);
plot(XE_rot,XN_rot);
title("1-sigma ellipse");
xlabel("X east");
ylabel("X north");
axis equal;
```

Appendix. B: Further Analysis of Result

B.1 Data from MATLAB code

Table (2) Satellites' position in ECEF frame

	Sat a	Sat b	Sat c	Sat d	Sat e	Sat f
X (km)	16126	12604	25943	21059	10073	15934
Y (km)	-15548	12117	-4760	16302	21064	-4820
Z(Km)	14384	20032	4339	2284	13011	20534

Table (3) Errors in pseudorange measurements

	Sat a	Sat b	Sat c	Sat d	Sat e	Sat f
Satellite clock error (s)	0.315e-4	1.2e-4	-0.003e-4	-0.164e-4	7.839e-4	-0.069e-4
Ionosphere error (m)	1.4094	-1.5280	1.4982	1.5761	1.4990	-7.8309
Troposphere error (m)	3.7807	2.7911	4.4837	6.3836	4.8097	2.5164
User clock error (m)	3.2614e+05					