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

Centre for Space Science Technology

Final Project: GPS Positioning (30%)

Intermediate Report 2: Computing GPS Satellite Positioning (8%)

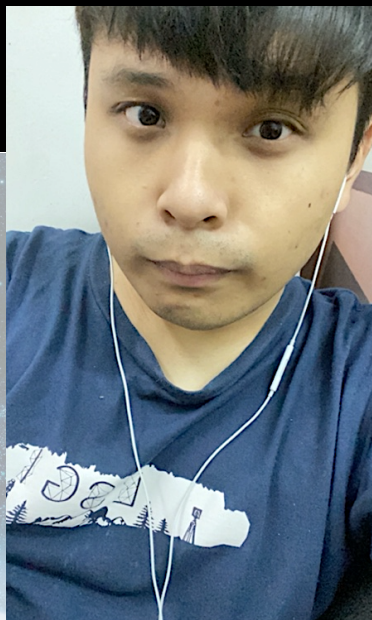
Due Date: 03-11-2021 23:59

Student Name: Tang Justin Hayse Chi Wing G.

Student ID: 20016345D

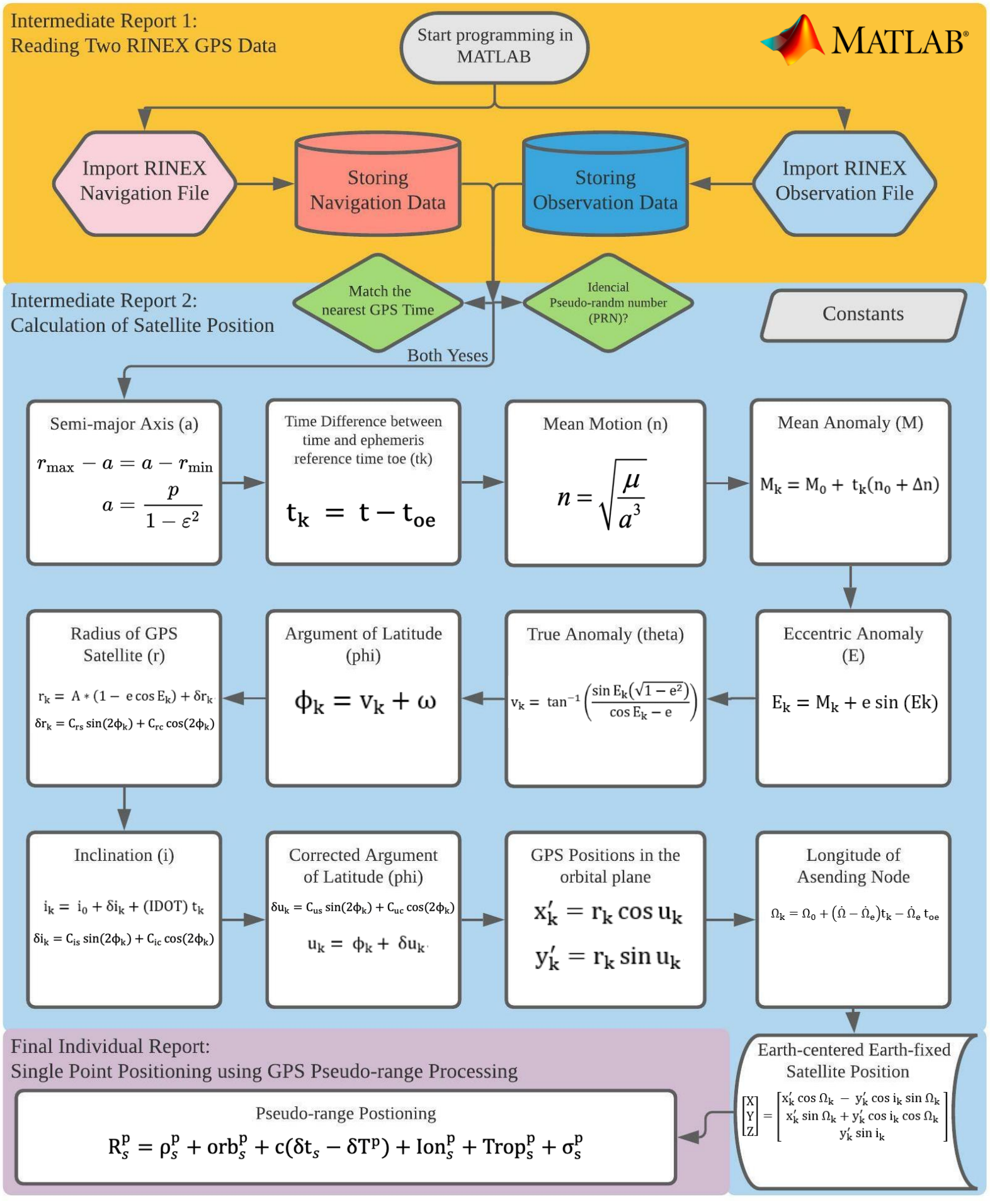
Final Student of BSc (Hons) in Land Surveying & Geo-Informatics 2022'

Expected Grade: A Range



1 Objective and Framework

For the **Second Intermediate Report**, which count for 8% of total grade, the aim is to calculate satellite position. Prior to start programming, a well-organized framework can help familiarise the whole idea of orbit calculation. The below framework displays the whole steps of calculating the satellite position with relevant formulas. For the detailed steps with explanations and MATLAB program code, it is discussed in Section 3.



2 Detailed Steps with Formula and MATLAB code

Workflow	Formula/ Constant	MATLAB Code

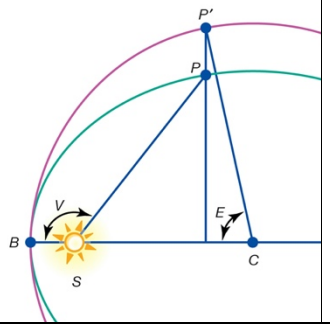
Preparation: Make sure both RIXEX navigation and observation files are already correctly saved in MATLAB workplace (database). Moreover, the PRN number and the GPS time from both files should also be identical. In addition, input physical constants at the beginning, for the upcoming orbit calculation.	c = 299792458 m/s
	g = 9.80665 m/s^2
	G = 6.67259 x 10^{-11}
	GM = 3.986005 x 10^{14} m^3/s^2
	$\dot{\Omega}_e$ = 7.2921151467 x 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 1 - 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 2 - 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 3 - 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 4 - 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 5 - 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 6 - True Anomaly (m): An angle from perigee to the satellite position, which gives the 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 7 - 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 8 - 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 9 - 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 10 - 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 11 - 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 12 - 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 13 - 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=====================`
		`end`

3 Output

Based on the MATLAB program code, ECEF GPS broadcast satellite positions are successfully computed. The outcome is in the "workplace". The file name called "Satellite Position" (23209 x 3 double) is the results for completing this middle progress report. The first, second and third columns are X, Y, Z coordinates, respectively.

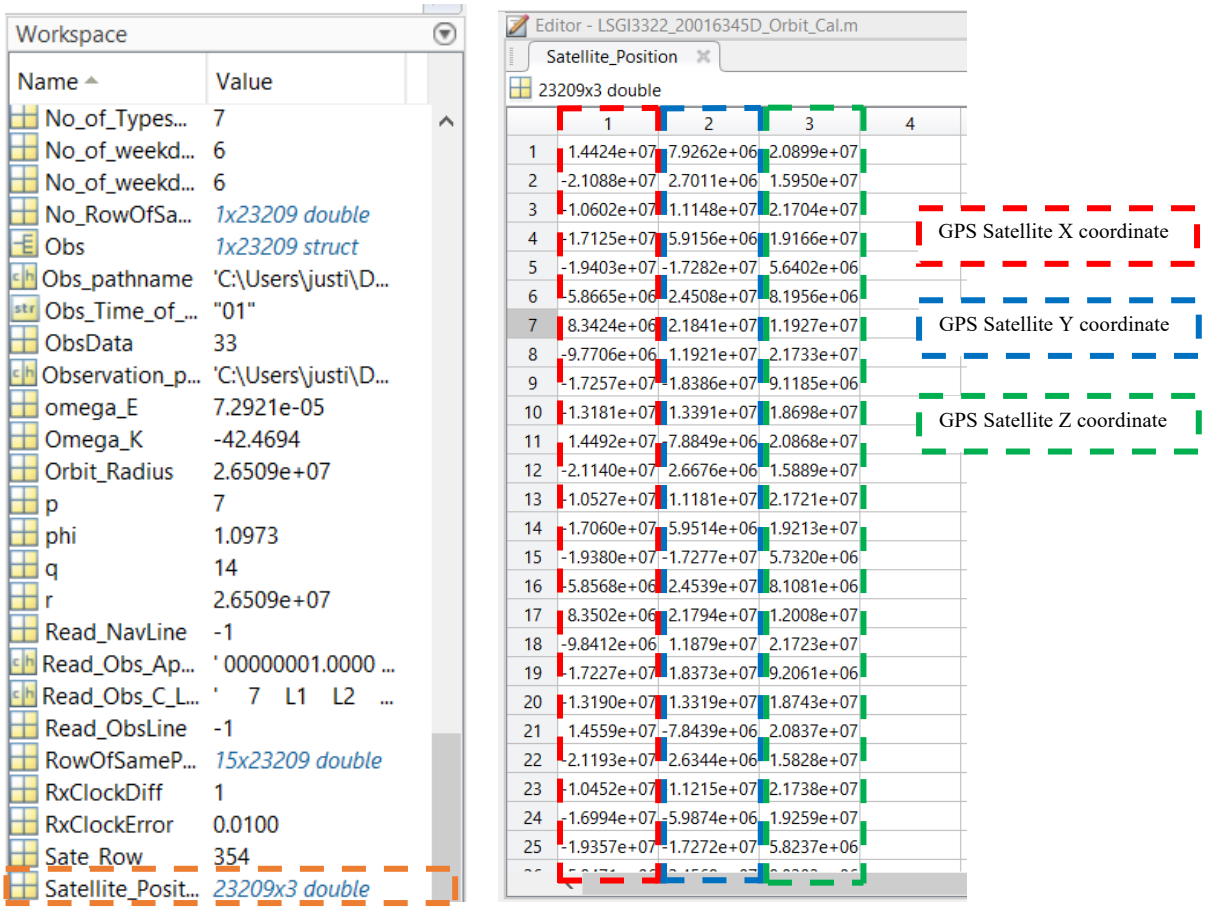


Figure 1. (Left) The workplace; Figure 2. (Right) The computed coordinate of GPS satellite position.

4 Difficulties and Improvements

Indeed, completing the second intermediate report found several problems had to encountered, mostly regarding the deep understanding of satellite orbit calculation and the use of MATLAB software.

4.1 Discovering the Relationships of all Algorithm and Formulas

Although the lecture notes have comprehensively explained the steps of the orbit calculation, the practical on MATLAB could be difficult than expected. It is because one have to know very well about the process of orbit calculation, otherwise it could be a hardship to complete this assignment. Writing a presentable flowchart (See Section 1) and drawing the relationships (e.g., mean motion, anomaly, radius of GPS satellite) can help clarify the orbit computation progress.

4.2 Completing MATLAB Programming with Limited Time

Frankly speaking, my understanding of MATLAB programme is not very deep, therefore it could be difficult to write MATLAB scripts. However, it could be an opportunity for me to comprehend a programme language intensively throughout one semester (only around three months!).

4.3 Difficulty of Proving the Output

The output of GPS satellite position computation is completed. Nonetheless, it is difficult to confirm the output (23209 x3 double) is correct or something(s) is/are missing throughout the computation process.

4 Conclusion and Future Planning

In this middle section, the calculation of satellite position is successfully completed. For the upcoming section, namely the final project, I have chosen single point positioning (SPP) through pseudo-range measurement to find out the GPS receiver position. Pseudo-range measurement means the measure of the distance between the receiver antenna and the satellite, the pseudo-range formula is shown as below.

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

ρ: the geometric distance between satellite and GPS receiver; c: the speed of light; dt_i : receiver clock error; dt^k: GPS satellite clock error ; d_{ion}: ionospheric delay; d_{trop}: tropospheric delay; d_{orb}: satellite orbit error; $others$: multipath error and measurement noises

Further, the least squares adjustment with linearization is required for calculate the coordinate of GPS receiver.

$$\tilde{P} + n = \rho(0) + \frac{(X_r^0 - X_s)}{\rho(0)} d_x + \frac{(Y_r^0 - Y_s)}{\rho(0)} d_y + \frac{(Z_r^0 - Z_s)}{\rho(0)} d_z + cd_t$$

$$\tilde{P} - \rho(0) + n = \frac{(X_r^0 - X_s)}{\rho(0)} d_x + \frac{(Y_r^0 - Y_s)}{\rho(0)} d_y + \frac{(Z_r^0 - Z_s)}{\rho(0)} d_z + cd_t$$

$$\Delta P^S + n = a_1 d_x + a_2 d_y + a_3 d_z + cd_t$$

With the linearized observation equation, the position of the GPS receiver can be computed:

$$L + n = Ax$$

$$\begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \vdots \\ \Delta P_n \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_n \end{bmatrix} = \begin{bmatrix} a_1^1 & a_2^1 & a_3^1 & c \\ a_1^2 & a_2^2 & a_3^2 & c \\ \vdots & \vdots & \vdots & \vdots \\ a_1^n & a_2^n & a_3^n & c \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \\ dt_n \end{bmatrix}$$

Solving equation,

$$X = (A^T W A)^{-1} A W L$$

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

These above processes will be soon conducted for the completion of the final individual project. These progress might presumably be take considerable times to find out the computed GPS receiver location.

Thank you for reading.

Reference List

[1] Leick, A., Rapoport, L., Tatarnikov, D., and Morrison, E. GPS satellite surveying (Fourth ed.). Hoboken, New Jersey: Wiley, 2015.

[2] Montenbruck, T., Montenbruck, O., and Teunissen, P. J. G., Global navigation satellite systems. Switzerland: Springer, 2017.

[3] Torge, W. and Müller, J. Geodesy (Fourth ed., De Gruyter textbook). Berlin ; Boston: De Gruyter, 2012.

[4] Travis H. Introduction to Satellite Attitude Control., InTech Open Book. Series, 2019.

[5] Wei, Z., Ruan R., Jia, X., Wu, X., Song, X, Mao, Y., Zhu Y. Satellite Positioning and Orbit Determination System SPODS: Theory and Test. Ce Hui Xue Bao, 43(1), 1-4, 2016.

[6] Gunter, S., Satellite geodesy, 2nd rev. and extended ed.. Berlin ; New York: Walter de Gruyter, 2003.

[7] Xu, G. and Xu, Y., GPS: Theory, Algorithm and Applications. Berlin, Heidelberg: Springer Berlin, 2016.

[8] Miura, S., Hsu, T.A., Chen, F., and Kamijo S., “GPS Error Correction With Pseudorange Evaluation Using Three-Dimensional Maps,” IEEE transactions on intelligent transportation systems, vol. 16, no. 6, pp. 3104–3115, 2015, doi: 10.1109/TITS.2015.2432122.

[9] Zhao, S., Cui, X., and Lu, X., “Single point positioning using full and fractional pseudo range measurements from GPS and BDS,” Survey review - Directorate of Overseas Surveys, vol. 53, no. 376, pp. 27–34, 2021, doi: 10.1080/00396265.2019.1683327.

[10] Ng, H.F., Zhang, G., Yang, K., Yang, S.X., and Hsu, L.T., “Improved weighting scheme using consumer-level GNSS L5/E5a/B2a pseudorange measurements in the urban area,” Advances in space research, vol. 66, no. 7, pp. 1647–1658, 2020, doi: 10.1016/j.asr.2020.06.002.

--- THIS IS THE END OF THE LSGI3322 SECOND INTERMEDIATE REPORT. THANK YOU. ---

Appendix - MATLAB Programme Source Code (Read RINEX Nav & Obs Files + Orbit Calculation)

```
clc

clear all
%=========================================================================
%================= LSGI3322 First Intermediate Report ====================
%============== Read Both Navigation and Observation Files ===============
%=========================================================================

%=========================================================================
%======================Read RINEX navigation File=========================
%=========================================================================
z1=msgbox({'Input a navigation file.' ; 'Press OK.'}, 'Reminder','warn');
uiwait(z1);
[selNav, Nav_pathname] = uigetfile;
Navigation_path = strcat(Nav_pathname,selNav);
NavData = fopen(Navigation_path,'r');
%==========Find out the header section "END OF HEADER"====================
Read_NavLine = fgetl(NavData);
while ischar(Read_NavLine)
if contains(Read_NavLine, 'END OF HEADER')
break;
end
Read_NavLine = fgetl(NavData);
end
%======================Read RINEX navigation File=========================
Epoch_NAV = 1;
while ischar(Read_NavLine)
Read_NavLine = fgetl(NavData);
if (Read_NavLine == -1)
Epoch_NAV = Epoch_NAV - 1;
break;
end
%=========================Broadcase 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));
Nav(Epoch_NAV).Date_Numerical=datenum(Nav(Epoch_NAV).Year,Nav(Epoch_NAV).Month,Nav(Epoch_NAV).Day,
Nav(Epoch_NAV).Hour,Nav(Epoch_NAV).Minute,Nav(Epoch_NAV).Second);
No_of_weekday_In_Nav=weekday(Nav(Epoch_NAV).Date_Numerical)-1;
Nav(Epoch_NAV).Time_in_GPS=No_of_weekday_In_Nav*60*60*24+Nav(Epoch_NAV).Hour*60*60+Nav(Epoch_NAV).
Minute*60+Nav(Epoch_NAV).Second;
Nav(Epoch_NAV).SV_Clock_Bias=str2num(Read_NavLine(23:41));
Nav(Epoch_NAV).SV_Clock_drift=str2num(Read_NavLine(42:60));
Nav(Epoch_NAV).SV_Clock_drift_rate=str2num(Read_NavLine(61:79));
Read_NavLine=fgetl(NavData);
%=========================Broadcase Orbit - 2=============================
Nav(Epoch_NAV).IODE=str2num(Read_NavLine(4:22));
Nav(Epoch_NAV).Crs=str2num(Read_NavLine(23:41));
Nav(Epoch_NAV).Delta_N=str2num(Read_NavLine(42:60));
Nav(Epoch_NAV).M0=str2num(Read_NavLine(61:79));
Read_NavLine=fgetl(NavData);
%=========================Broadcase Orbit - 3=============================
Nav(Epoch_NAV).Cuc=str2num(Read_NavLine(4:22));
Nav(Epoch_NAV).e=str2num(Read_NavLine(23:41));
Nav(Epoch_NAV).Cus=str2num(Read_NavLine(42:60));
Nav(Epoch_NAV).sqrt_a=str2num(Read_NavLine(61:79));
Read_NavLine=fgetl(NavData);
%=========================Broadcase Orbit - 4=============================
Nav(Epoch_NAV).Toe_time=str2num(Read_NavLine(4:22));
Nav(Epoch_NAV).Cic=str2num(Read_NavLine(23:41));
Nav(Epoch_NAV).Omega_0=str2num(Read_NavLine(42:60));
Nav(Epoch_NAV).CIS=str2num(Read_NavLine(61:79));
Read_NavLine=fgetl(NavData);
%=========================Broadcase Orbit - 5=============================
Nav(Epoch_NAV).i0=str2num(Read_NavLine(4:22));
Nav(Epoch_NAV).Crc=str2num(Read_NavLine(23:41));
Nav(Epoch_NAV).Omega=str2num(Read_NavLine(42:60));
```

```
Nav(Epoch_NAV).Omega_dot=str2num(Read_NavLine(61:79));
Read_NavLine=fgetl(NavData);
%=========================Broadcase Orbit - 6============================
Nav(Epoch_NAV).IDOT=str2num(Read_NavLine(4:22));
Nav(Epoch_NAV).weekNO=str2num(Read_NavLine(42:60));
Read_NavLine=fgetl(NavData);
%=========================Broadcase Orbit - 7============================
Nav(Epoch_NAV).SV_accuracy=str2num(Read_NavLine(4:22));
Nav(Epoch_NAV).SV_health=str2num(Read_NavLine(23:41));
Nav(Epoch_NAV).TGD=str2num(Read_NavLine(42:60));
Nav(Epoch_NAV).IODC=str2num(Read_NavLine(61:79));
Read_NavLine=fgetl(NavData);
%=========================Broadcase Orbit - 8============================
Nav(Epoch_NAV).Transmission_timeofMessage=str2num(Read_NavLine(4:22));

Epoch_NAV=Epoch_NAV+1;
%==================The END of Reading Navigation Data====================
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%=====================Read RINEX Observation File=========================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z2=msgbox({'Input an observation file.' ; 'Press OK.'}, 'Reminder','warn');
uiwait(z2);
[selObs, Obs_pathname] = uigetfile;
Observation_path = strcat(Obs_pathname,selObs);
ObsData = fopen(Observation_path,'r');
fprintf('Please Find the Satellite Position on "Workplace".');
%============Find the Wording: "'APPROX POSITION XYZ'"====================
Read_Obs_Approx_XYZ_Line = fgetl(ObsData);
while ischar(Read_Obs_Approx_XYZ_Line)
if contains(Read_Obs_Approx_XYZ_Line, 'APPROX POSITION XYZ')
break;
end
Read_Obs_Approx_XYZ_Line = fgetl(ObsData);
end
Approx_X = str2num(Read_Obs_Approx_XYZ_Line(2:14));
Approx_Y = str2num(Read_Obs_Approx_XYZ_Line(16:28));
Approx_Z = str2num(Read_Obs_Approx_XYZ_Line(30:42));
%============Find the Wording: "'# / TYPES OF OBSERV'"====================
Read_Obs_C_Line = fgetl(ObsData);
while ischar(Read_Obs_C_Line)
if contains(Read_Obs_C_Line, '# / TYPES OF OBSERV')
break;
end
Read_Obs_C_Line = fgetl(ObsData);
end
No_of_TypesOfObservation = str2num(Read_Obs_C_Line(4:6));
%=====Find the C1 Position in the '# / TYPES OF OBSERV'====================
for p = 1:No_of_TypesOfObservation
    Types_of_observation{p} = Read_Obs_C_Line(4+6*p:6+6*p);
end
IndexNo_C = find(contains(Types_of_observation,'C1'));
%============Find the Wording: "'TIME OF FIRST OBS'"=======================
Read_ObsLine = fgetl(ObsData);
while ischar(Read_ObsLine)
if contains(Read_ObsLine, 'TIME OF FIRST OBS')
Obs_Time_of_FirstObs = string(Read_ObsLine(5:6));
break;
end
Read_ObsLine = fgetl(ObsData);
end
%==================Find the Wording: "END OF HEADER"=======================
Read_ObsLine = fgetl(ObsData);
while ischar(Read_ObsLine)
if contains(Read_ObsLine, 'END OF HEADER')
break;
end
Read_ObsLine = fgetl(ObsData);
end
%=====================Read Observation Data================================
Epoch_OBS = 0;
while ischar(Read_ObsLine)
Read_ObsLine = fgetl(ObsData);
if (Read_ObsLine == -1)
```

```
break;
end
%====================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));
Obs(Epoch_OBS + h).Epoch_Flag=str2num(Read_ObsLine(28:29));
Obs(Epoch_OBS + h).Date_numerical=datenum(Obs(Epoch_OBS + h).Year,Obs(Epoch_OBS +
h).Month,Obs(Epoch_OBS + h).Day,Obs(Epoch_OBS + h).Hour,Obs(Epoch_OBS + h).Minute,Obs(Epoch_OBS +
h).Second);
No_of_weekday_In_Obs=weekday(Obs(Epoch_OBS + h).Date_numerical)-1;
Obs(Epoch_OBS + h).Time_in_GPS=No_of_weekday_In_Obs*60*60*24+Obs(Epoch_OBS +
h).Hour*60*60+Obs(Epoch_OBS + h).Minute*60+Obs(Epoch_OBS + h).Second;
end
%======================Find the C1 Observation Data=======================
for d = 1:No_of_PRN
Read_ObsLine = fgetl(ObsData);
Obs(Epoch_OBS + d).C1 = str2num(Read_ObsLine(2+16*(IndexNo_C-1):15+16*(IndexNo_C-1)));
Read_ObsLine = fgetl(ObsData);
end
Epoch_OBS = Epoch_OBS + No_of_PRN;
end
%==================The END of Reading Observation Data====================
end
%========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;
iteration=0;
dx = [0.01;0.01;0.01;RxClockError];
Approx_Coordinate = [Approx_X;Approx_Y;Approx_Z;RxClockError];
%==========Whether both files with the identical PRN number?===============
for i = 1:Epoch_OBS
q = 1;
for b = 1:Epoch_NAV
if isequal(Compare_Nav_PRN(1,b),Compare_Obs_PRN(1,i))
RowOfSamePRN(q,i) = b;
q = q + 1;
end
end
No_RowOfSamePRN = sum(RowOfSamePRN~=0);
%===============The nearest GPS time with 4 Hours Vaildation===============
Difference_Minimum = 4*60*60;
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
end
%==========================================================================
%================= LSGI3322 Second Intermediate Report ====================
%================ Calculation of GPS Satellite Positions ==================
%==========================================================================

%============================Basic Constants===============================
c=299792458;                %Physcial Constant: Speed of Light (m/s)
g=9.80665;                  %Physcial Constant: Accelera
G=6.67259e-11;              %Physcial 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)
```

```
%=========================Keplerian Elements=============================
%=========================Semi-major Axis (a)============================
semi_major_axis=(Nav(Sate_Row).sqrt_a)^2;
a = semi_major_axis;
%==================Time from Ephemeris Reference Epoch====================
half_week = 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 > half_week)
tk = tk - (2*half_week);
elseif (tk < -half_week)
tk = tk + (2*half_week);
end
%=========================Mean motion (n)=================================
n0 = sqrt (GM / (a^3));
n = n0 + Nav(Sate_Row).Delta_N;
%=========================Mean anomaly (M)================================
M = Nav(Sate_Row).M0 + (n * tk);
%=========================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
%=======================True anomaly (True_Anomaly)=====================
True_Anomaly = 2*atan(sqrt((1 + Nav(Sate_Row).e)/(1 - Nav(Sate_Row).e)) * tan(E/2));
%==================Argument of latitude (Coorected_ArgLat)================
phi = True_Anomaly + Nav(Sate_Row).Omega;
%================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;
%==================Corrected GPS orbit plane's 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;
%====================argument of latitude (Corr0ected_phi)================
delta_phi = Nav(Sate_Row).Cus * sin(2*phi) + Nav(Sate_Row).Cuc * cos(2*phi);
Coorected_ArgLat = phi + delta_phi;
%====================GPS positions in the orbital plane====================
X0 = Orbit_Radius * cos(Coorected_ArgLat);
Y0 = Orbit_Radius * sin(Coorected_ArgLat);
%=====================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;
%=========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====================
end
```