**AAE6102 Assignment Report**

**Name: Li Wang Student No: 20114095r**

**Requirement: calculating the receiver position and receiver clock error.**

The first step is to calculate the XYZ positions for all valid satellite at time 440992.

By using the equations listed in ICD, the positions of eight satellites are:

5 -8855453.41982622 -22060174.3786006 -11922092.1022657 56679.9157383027

6 -8087139.45347258 -16946005.6440138 18816194.5083126 -29.6839108349452

10 9027719.26443692 -12319179.5685345 21737387.5847633 9969.71388334742

17 -21277079.8368316 -7467235.53211887 14287503.9521099 -61434.6041448214

22 -13649576.2408342 8229427.43810091 21122958.6100403 66764.9603132400

23 -19452219.4402461 -16750492.1989638 -6918520.67331300 3115.31453068308

26 6163063.99824452 -25286737.0429405 -3541191.15788752 84239.2012273288

30 -17713788.1911051 -19797565.2288933 19209.1008794379 -3006.53072231040

The fist column is the PRN of each satellite, the second to the fourth are X, Y, and Z respectively. The last column is the satellite clock error.

Then, using least square and initial position we can get the first iteration result:

delta X = -5.714532986740690e+03

delta Y = 1.081755595342484e+03

delta Z = -2.606154365685373e+03

delta b = 5.194618917145593e+05

And the second iteration result is:

delta X = -0.128457002905858

delta Y = -0.098611297258279

delta Z = 0.105113323543143

delta b = -0.553073005475141

After the third iteration, the result is converged. In other words, delta X, delta Y, delta Z, delta b are all less than 1e-4. The final result is:

X = 2.700400134443800e+06

Y = -4.292560709015938e+06

Z = 3.855272874747667e+06

b = 5.194613386415655e+05

Transferring the receiver clock error to seconds we get 0.001732736514144 s, this explains why the receiver clock at this epoch is 44092.00173454 s.

The Github link of this assignment is: https://github.com/WangLeeeeeeee/AAE6102\_Assignment

Appendix (Matlab code)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% AAE6102 Assignment: Single Point Positioning

% Author: Li Wang

% Input: Ephemeris file and Observation file

% Output: Receiver position

% Date: 2021.10.11

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;

clc;

c = 2.99792458e8; % speed of light (m/s)

% Read ephemeris file

EphemerisFile = './eph.dat';

ephemeris = Read\_Ephemeris(EphemerisFile);

% Read observation file

ObserFile = './rcvr.dat';

observation = Read\_Observation(ObserFile);

% sort observation accoring to PRN number

observation = sortrows(observation,2);

% Calculate satellite position

SatellitePos = Compute\_SatellitePos(ephemeris,observation);

SatellitePos = sortrows(SatellitePos,1);

% Least Square

length = 8;

H = zeros(length,4);

L = zeros(length,1);

rho\_hat = zeros(length,1);

X = -2694685.473;

Y = -4293642.366;

Z = 3857878.924;

b = 0;

while 1

for ii = 1:length

t1 = (SatellitePos(ii,2)-X)^2;

t2 = (SatellitePos(ii,3)-Y)^2;

t3 = (SatellitePos(ii,4)-Z)^2;

rho\_hat(ii) = sqrt(t1+t2+t3);

end

for ii = 1:length

H(ii,1) = (SatellitePos(ii,2)-X)/rho\_hat(ii);

H(ii,2) = (SatellitePos(ii,3)-Y)/rho\_hat(ii);

H(ii,3) = (SatellitePos(ii,4)-Z)/rho\_hat(ii);

H(ii,4) = -1;

L(ii) = observation(ii,3) - rho\_hat(ii) + SatellitePos(ii,5) + SatellitePos(ii,6) - b;

end

dx = -H \ L;

X = X + dx(1);

Y = Y + dx(2);

Z = Z + dx(3);

b = b + dx(4);

if norm(dx) < 1e-8, break; end

end

%----------subfunction: Read Ephemeris File-----------------------------

function ephemeris = Read\_Ephemeris(EphemerisFile)

navFile = fopen(EphemerisFile);

if navFile == -1

disp('There is no such ephemeris file!');

return;

end

SateCount = 0;

while ~feof(navFile)

CurLine = fgetl(navFile);

if isempty(CurLine)

continue;

end

SateCount = SateCount + 1;

ephemeris(SateCount,:) = (str2double(split(CurLine)))';

end

end

%----------subfunction: Read Observation File-----------------------------

function observation = Read\_Observation(ObserFile)

obsFile = fopen(ObserFile);

if obsFile == -1

disp('There is no such observation file');

return;

end

SateCount = 0;

while ~feof(obsFile)

CurLine = fgetl(obsFile);

if isempty(CurLine)

continue;

end

SateCount = SateCount + 1;

observation(SateCount,:) = (str2double(split(CurLine)))';

end

end

%----------subfunction: Compute Satellite Position-----------------------------

function SatellitePos = Compute\_SatellitePos(ephemeris,observation)

GM = 3.986005e14; % earth's universal gravitational [m^3/s^2]

omegae = 7.2921151467e-5; % earth's rotation rate (rad/sec)

lightspeed = 2.99792458e8; % speed of light (m/s)

F = -4.442807633e-10; % Constant, [sec/(meter)^(1/2)]

lengthEph = size(ephemeris,1);

SatellitePos = zeros(lengthEph,6);

for i=1:lengthEph

% Calculate Velocity

[sqrtA,deltan] = deal(ephemeris(i,10),ephemeris(i,11));

n0 = sqrt(GM)/sqrtA^3;

n = n0 + deltan;

% Compute satellite clock correction

[tsv,PRN,toc,toe,af0,af1,af2] = deal(ephemeris(i,1),...

ephemeris(i,2),ephemeris(i,3),ephemeris(i,4),...

ephemeris(i,5),ephemeris(i,6),ephemeris(i,7));

deltaT = af0 + af1\*(tsv-toc) + af2\*(tsv-toc)^2;

tsv = 440992;

t = tsv - deltaT;

for ii=1:8

if observation(ii,2) == PRN

tSelfRotate = observation(ii,3)/lightspeed;

break;

end

end

tk = t - toe - tSelfRotate;

M0 = ephemeris(i,12);

Mk = M0 + n\*tk;

% Compute Kepler's equatin of eccentric anomaly

e = ephemeris(i,9);

Ek = kepOrb2E(Mk,e);

%Compute relativistic correction term

dtr = F \* e \* sqrtA \* sin(Ek);

RelDelay = dtr\*lightspeed;

vk = atan2(sqrt(1-e^2)\*sin(Ek),(cos(Ek)-e));

% Compute argument of latitude

w = ephemeris(i,13);

Phik = w + vk;

% Compute correction

[Cus,Cuc,Cis,Cic,Crs,Crc] = deal(ephemeris(i,18),ephemeris(i,19),...

ephemeris(i,20),ephemeris(i,21),ephemeris(i,22),ephemeris(i,23));

deltaUk = Cuc\*cos(2\*Phik) + Cus\*sin(2\*Phik);

deltaRk = Crc\*cos(2\*Phik) + Crs\*sin(2\*Phik);

deltaIk = Cic\*cos(2\*Phik) + Cis\*sin(2\*Phik);

% Corrected argument of latitude

[i0,idot] = deal(ephemeris(i,15),ephemeris(i,17));

uk = Phik + deltaUk;

rk = sqrtA^2\*(1-e\*cos(Ek)) + deltaRk;

ik = i0 + idot\*tk + deltaIk;

% Position in orbital plane

xp = rk\*cos(uk);

yp = rk\*sin(uk);

% Corrected longtitude of ascending node

[Omega0,Omegadot] = deal(ephemeris(i,14),ephemeris(i,16));

Omegak = Omega0 + (Omegadot-omegae)\*tk - omegae\*toe;

% Earth-fixed geocentric satellite coordinate

Xk = xp\*cos(Omegak) - yp\*sin(Omegak)\*cos(ik);

Yk = xp\*sin(Omegak) + yp\*cos(Omegak)\*cos(ik);

Zk = yp\*sin(ik);

SatellitePos(i,:) = [PRN, Xk, Yk, Zk, deltaT\*lightspeed, RelDelay];

end

end

function E = kepOrb2E(M,e)

% Inputs: - mean anomaly in radians

% - eccentricity

% Output: Eccentric anomaly

if (-pi < M < 0) | (M > pi)

E = M - e;

else

E = M + e;

end

check = 1;

while check > 10e-10

E\_new = (E + (M - E + e \* sin(E))/(1 - e \* cos(E)));

check = abs(E\_new - E);

E = E\_new;

end

end