AAE 575

Introduction to Satellite Navigation and Positioning

Homework 5: PVT Solutions

December 17, 2011

Josh Wildey

**Objective:**

Using the ephemeris and receiver data, calculate the position of the receiver at the time the data was collected.

**Given Data:**

*Ephemeris Data*

*Receiver Data*

*Receiver Position Estimate:*

*X = -2701206.38 m*

*Y = -4293642.366 m*

*Z = 3857878.924 m*

**Problem 1:**

The given ephemeris data provides us with data for 8 different satellites. The initial clock bias is set to 0. Using this data and the code produced from homework 4 the positions of the satellites is calculated. After these are calculated the positions of the satellites are recalculated to account for the rotation of the earth and the time the receiver actually receives the signal. The pseudoranges for all 8 satellites are then calculated between the receiver and each satellite. The linear observation matrix, H, is then calculated and ends up being an 8 by 4 matrix. Once this is computed, MATLAB is used to solve the system of equations and comes up with the receiver position and clock bias estimation. This process is put in a ‘while’ loop until the desired precision of .5 meters. To complete the problem, it took 3 iterations for the .5 meter precision parameter that was given. The output is shown below.

Estimate of Position:

X Position: -2701206.380000

Y Position: -4293642.366000

Z Position: 3857878.924000

t\_b: 0.000000

Estimate of Position:

X Position: -2700363.172502

Y Position: -4292553.165172

Z Position: 3855265.290738

t\_b: 0.001733

Estimate of Position:

X Position: -2700363.246131

Y Position: -4292553.664862

Z Position: 3855265.969368

t\_b: 0.001733

The final position estimate of the receiver in Geodetic Coordinates was calculated to be:

Geodetic Coordinates:

Latitude: 37.428187

Longitude: -122.173211

Altitude: 46.380768

**Problem 2:**

The dilution of precision values can be computed from the linear observation matrix as follows:

The following values were calculated in MATLAB:

Dilution of Position

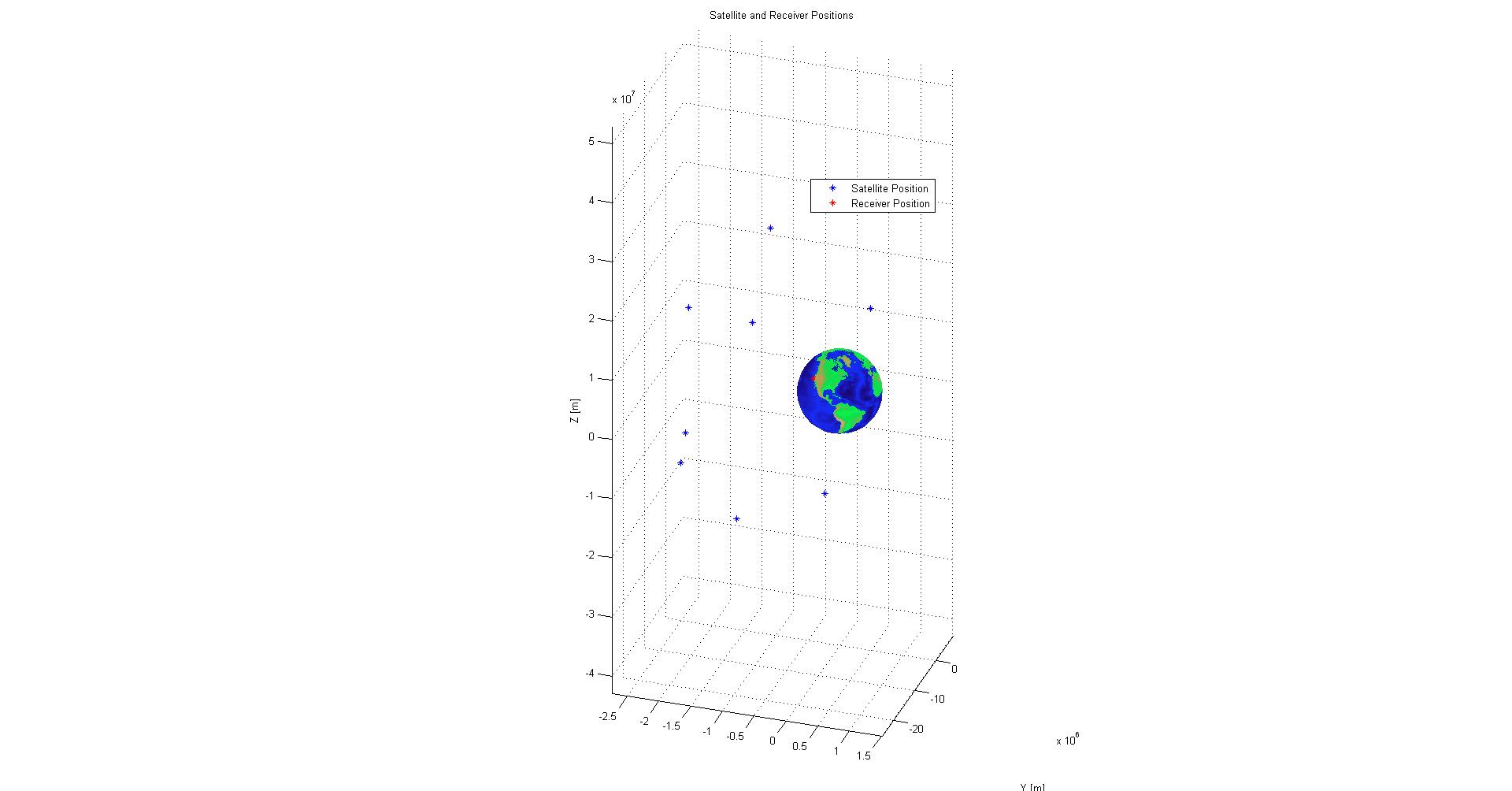
GDOP: 1.912511

PDOP: 1.718976

TDOP: 0.838344

**Conclusion:**

In conclusion, the receiver position was calculated in 3 iterations using the ephemeris data given. The final calculated positions of the satellites and the receiver can be seen in the following figure.



**Appendix A:**

**MATLAB Code: hw5.m**

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

% Author: Josh Wildey

% Class: AAE57500

% Homework 5 - 12/2/2011

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

close all

clear all

clc

format long

%%%%%%%%% Problem 1: %%%%%%%%%

load eph.dat

load rcvrc.dat

x\_rx = [-2701206.38; -4293642.366; 3857878.924]; %[m] Initial Position Guess (given)

tb\_R = 0; %[s] Initial Guess in receiver clock bias

x\_hat = [.6; .6; .6; 0]; % Initialize x\_hat for while loop control

% Display Initial Guess

disp('Estimate of Position:')

fprintf('X Position: %f\n',x\_rx(1))

fprintf('Y Position: %f\n',x\_rx(2))

fprintf('Z Position: %f\n',x\_rx(3))

fprintf('t\_b: %f\n\n',tb\_R)

rcvrc\_tow = rcvrc(:,1); %[s] Receiver Time of Week

svid = rcvrc(:,2); %[] Satellite Vehicle ID

pr = rcvrc(:,3); %[m] Pseudorange

c = 299792458; %[m/s] Speed of Light

toc = eph(:,3); %[s] Time of Clock

toe = eph(:,4); %[s] Time of Applicability

I\_0 = eph(:,15); %[rad] Orbital Inclination

omega\_dot = eph(:,16); %[rad/s] Rate of Right Ascen

e = eph(:,9); %[] Eccentricity

sqrt\_a = eph(:,10); %[m^.5] SQRT(A)

omega\_0 = eph(:,14); %[rad] Right Ascen at TOA

omega = eph(:,13); %[rad] Argument of Perigee

M\_0 = eph(:,12); %[rad] Mean Anom

delta\_n = eph(:,11); %[rad/s] mean motion diff

I\_dot = eph(:,17); %[rad/s] Rate of inclin

Cuc = eph(:,19); %[rad] lat cosine ampl

Cus = eph(:,18); %[rad] radius sin ampl

Crc = eph(:,23); %[m] inclin cos ampl

Crs = eph(:,22); %[m] radius sin ampl

Cic = eph(:,21); %[rad] inclin cos ampl

Cis = eph(:,20); %[rad] inclin sin ampl

omega\_dot\_e = 7.2921151467e-5; %[rad/sec] Number from IS-GPS-200E

mu = 3.986005e14; %[m^3/s^2] Number from IS-GPS-200E

t = rcvrc\_tow(1);

Y = pr;

I = eye(4);

while (abs(x\_hat(1))>.5 && abs(x\_hat(2))>.5 && abs(x\_hat(3))>.5)

tau = (pr - c.\*tb\_R)./c; %[s] Time delay of Satellite Signal

t\_tx = t - tau; %[s] Time of transmission

t\_k = t\_tx - toe; % Time from ephemeris refence time

A\_k = sqrt\_a.^2; % Semi-Major Axis

n\_0 = sqrt(mu./A\_k.^3); %[rad/s] Computed Mean Motion

n = n\_0 + delta\_n; % Correct Mean Motion

M\_k = M\_0 + n.\*t\_k; % Mean Anomaly

E\_k = kepler\_E(e,M\_k); % Eccentric Anomaly

v\_k = 2\*atan(sqrt((1+e)./(1-e)).\*tan(E\_k./2)); % True Anomaly

phi\_k = v\_k + omega; % Argument of latitude

% Second Harmonic Perturbations

delta\_u\_k = Cus.\*sin(2.\*phi\_k) + Cuc.\*cos(2.\*phi\_k); % Argument of latitude Correction

delta\_r\_k = Crs.\*sin(2.\*phi\_k) + Crc.\*cos(2.\*phi\_k); % Radial Correction

delta\_i\_k = Cis.\*sin(2.\*phi\_k) + Cic.\*cos(2.\*phi\_k); % Inclination Correction

u\_k = phi\_k + delta\_u\_k; % Corrected Argument of Latitude

r\_k = A\_k.\*(1-e.\*cos(E\_k)) + delta\_r\_k; % Corrected Radius

i\_k = I\_0 + I\_dot.\*t\_k + delta\_i\_k; % Corrected Inclination

% Positions in Orbital plane

x\_kprime = r\_k.\*cos(u\_k);

y\_kprime = r\_k.\*sin(u\_k);

% Corrected Longitude of Ascending Node

omega\_k = omega\_0 + (omega\_dot - omega\_dot\_e).\*t\_k - omega\_dot\_e.\*toe;

% Earth Fixed Coordinates of SV antenna phase center

x\_k = x\_kprime.\*cos(omega\_k) - y\_kprime.\*cos(i\_k).\*sin(omega\_k);

y\_k = x\_kprime.\*sin(omega\_k) + y\_kprime.\*cos(i\_k).\*cos(omega\_k);

z\_k = y\_kprime.\*sin(i\_k);

% Satellite Position at the time of reception

x\_k\_rx = cos(omega\_dot\_e.\*tau).\*x\_k + sin(omega\_dot\_e.\*tau).\*y\_k;

y\_k\_rx = -sin(omega\_dot\_e.\*tau).\*x\_k + cos(omega\_dot\_e.\*tau).\*y\_k;

z\_k\_rx = z\_k;

R = sqrt((x\_k- x\_rx(1)).^2 + (y\_k-x\_rx(2)).^2 + (z\_k-x\_rx(3)).^2);

H = [(x\_rx(1) - x\_k\_rx(1))/R(1), (x\_rx(2) - y\_k\_rx(1))/R(1), (x\_rx(3) - z\_k\_rx(1))/R(1) 1;

(x\_rx(1) - x\_k\_rx(2))/R(2), (x\_rx(2) - y\_k\_rx(2))/R(2), (x\_rx(3) - z\_k\_rx(2))/R(2) 1;

(x\_rx(1) - x\_k\_rx(3))/R(3), (x\_rx(2) - y\_k\_rx(3))/R(3), (x\_rx(3) - z\_k\_rx(3))/R(3) 1;

(x\_rx(1) - x\_k\_rx(4))/R(4), (x\_rx(2) - y\_k\_rx(4))/R(4), (x\_rx(3) - z\_k\_rx(4))/R(4) 1;

(x\_rx(1) - x\_k\_rx(5))/R(5), (x\_rx(2) - y\_k\_rx(5))/R(5), (x\_rx(3) - z\_k\_rx(5))/R(5) 1;

(x\_rx(1) - x\_k\_rx(6))/R(6), (x\_rx(2) - y\_k\_rx(6))/R(6), (x\_rx(3) - z\_k\_rx(6))/R(6) 1;

(x\_rx(1) - x\_k\_rx(7))/R(7), (x\_rx(2) - y\_k\_rx(7))/R(7), (x\_rx(3) - z\_k\_rx(7))/R(7) 1;

(x\_rx(1) - x\_k\_rx(8))/R(8), (x\_rx(2) - y\_k\_rx(8))/R(8), (x\_rx(3) - z\_k\_rx(8))/R(8) 1;];

M = linsolve(H',I');

M = M';

h = [(R(1) + c\*tb\_R); (R(2) + c\*tb\_R); (R(3) + c\*tb\_R); (R(4) + c\*tb\_R);

(R(5) + c\*tb\_R); (R(6) + c\*tb\_R); (R(7) + c\*tb\_R); (R(8) + c\*tb\_R);];

y = Y - h;

x\_hat = M\*y;

x\_rx = x\_rx + x\_hat(1:3);

tb\_R = tb\_R + x\_hat(4)/c;

disp('Estimate of Position:')

fprintf('X Position: %f\n',x\_rx(1))

fprintf('Y Position: %f\n',x\_rx(2))

fprintf('Z Position: %f\n',x\_rx(3))

fprintf('t\_b: %f\n\n',tb\_R)

end

lla = ecef2lla([x\_rx(1) x\_rx(2) x\_rx(3)], 'WGS84');

disp('Geodetic Coordinates:')

fprintf('Latitude: %f\n',lla(1))

fprintf('Longitude: %f\n',lla(2))

fprintf('Altitude: %f\n\n',lla(3))

% Geometry matrix:

G = inv(H'\*H);

%GDOP

GDOP = sqrt(trace(G));

% PDOP

PDOP = sqrt(G(1,1) + G(2,2)+ G(3,3));

%TDOP

TDOP = sqrt(G(4,4));

disp('Dilution of Position')

fprintf('GDOP: %f\n',GDOP)

fprintf('PDOP: %f\n',PDOP)

fprintf('TDOP: %f\n',TDOP)

% Plot Position of Satellites and Receiver around Earth

figure(1)

hold on

title('Satellite and Receiver Positions')

plot3(x\_k,y\_k,z\_k,'\*')

plot3(x\_rx(1),x\_rx(2),x\_rx(3),'r\*')

legend('Satellite Position','Receiver Position')

earth\_sphere('m')

grid on

hold off

**Kepler Equation Solver Function:**

function E = kepler\_E(e, M)

% ------------------------------------------------------------

%

% This function uses Newton’s method to solve Kepler’s

% equation E - e\*sin(E) = M for the eccentric anomaly,

% given the eccentricity and the mean anomaly.

%

% E - eccentric anomaly (radians)

% e - eccentricity, passed from the calling program

% M - mean anomaly (radians), passed from the calling program

% pi - 3.1415926...

%

% User M-functions required: none

% ------------------------------------------------------------

%...Set an error tolerance:

error = 1.e-12;

%...Select a starting value for E:

if M < pi

E = M + e/2;

else

E = M - e/2;

end

%...Iterate on Equation 3.14 until E is determined to within

%...the error tolerance:

ratio = 1;

while abs(ratio) > error

ratio = (E - e.\*sin(E) - M)./(1 - e.\*cos(E));

E = E - ratio;

end

% ˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜˜

**Earth Sphere Function:**

function [xx,yy,zz] = earth\_sphere(varargin)

%EARTH\_SPHERE Generate an earth-sized sphere.

% [X,Y,Z] = EARTH\_SPHERE(N) generates three (N+1)-by-(N+1)

% matrices so that SURFACE(X,Y,Z) produces a sphere equal to

% the radius of the earth in kilometers. The continents will be

% displayed.

%

% [X,Y,Z] = EARTH\_SPHERE uses N = 50.

%

% EARTH\_SPHERE(N) and just EARTH\_SPHERE graph the earth as a

% SURFACE and do not return anything.

%

% EARTH\_SPHERE(N,'mile') graphs the earth with miles as the unit rather

% than kilometers. Other valid inputs are 'ft' 'm' 'nm' 'miles' and 'AU'

% for feet, meters, nautical miles, miles, and astronomical units

% respectively.

%

% EARTH\_SPHERE(AX,...) plots into AX instead of GCA.

%

% Examples:

% earth\_sphere('nm') produces an earth-sized sphere in nautical miles

%

% earth\_sphere(10,'AU') produces 10 point mesh of the Earth in

% astronomical units

%

% h1 = gca;

% earth\_sphere(h1,'mile')

% hold on

% plot3(x,y,z)

% produces the Earth in miles on axis h1 and plots a trajectory from

% variables x, y, and z

% Clay M. Thompson 4-24-1991, CBM 8-21-92.

% Will Campbell, 3-30-2010

% Copyright 1984-2010 The MathWorks, Inc.

%% Input Handling

[cax,args,nargs] = axescheck(varargin{:}); % Parse possible Axes input

error(nargchk(0,2,nargs)); % Ensure there are a valid number of inputs

% Handle remaining inputs.

% Should have 0 or 1 string input, 0 or 1 numeric input

j = 0;

k = 0;

n = 50; % default value

units = 'km'; % default value

for i = 1:nargs

if ischar(args{i})

units = args{i};

j = j+1;

elseif isnumeric(args{i})

n = args{i};

k = k+1;

end

end

if j > 1 || k > 1

error('Invalid input types')

end

%% Calculations

% Scale factors

Scale = {'km' 'm' 'mile' 'miles' 'nm' 'au' 'ft';

1 1000 0.621371192237334 0.621371192237334 0.539956803455724 6.6845871226706e-009 3280.839895};

% Identify which scale to use

try

myscale = 6378.1363\*Scale{2,strcmpi(Scale(1,:),units)};

catch %#ok<\*CTCH>

error('Invalid units requested. Please use m, km, ft, mile, miles, nm, or AU')

end

% -pi <= theta <= pi is a row vector.

% -pi/2 <= phi <= pi/2 is a column vector.

theta = (-n:2:n)/n\*pi;

phi = (-n:2:n)'/n\*pi/2;

cosphi = cos(phi); cosphi(1) = 0; cosphi(n+1) = 0;

sintheta = sin(theta); sintheta(1) = 0; sintheta(n+1) = 0;

x = myscale\*cosphi\*cos(theta);

y = myscale\*cosphi\*sintheta;

z = myscale\*sin(phi)\*ones(1,n+1);

%% Plotting

if nargout == 0

cax = newplot(cax);

% Load and define topographic data

load('topo.mat','topo','topomap1');

% Rotate data to be consistent with the Earth-Centered-Earth-Fixed

% coordinate conventions. X axis goes through the prime meridian.

% http://en.wikipedia.org/wiki/Geodetic\_system#Earth\_Centred\_Earth\_Fixed\_.28ECEF\_or\_ECF.29\_coordinates

%

% Note that if you plot orbit trajectories in the Earth-Centered-

% Inertial, the orientation of the contintents will be misleading.

topo2 = [topo(:,181:360) topo(:,1:180)]; %#ok<NODEF>

% Define surface settings

props.FaceColor= 'texture';

props.EdgeColor = 'none';

props.FaceLighting = 'phong';

props.Cdata = topo2;

% Create the sphere with Earth topography and adjust colormap

surface(x,y,z,props,'parent',cax)

colormap(topomap1)

% Replace the calls to surface and colormap with these lines if you do

% not want the Earth's topography displayed.

% surf(x,y,z,'parent',cax)

% shading flat

% colormap gray

% Refine figure

axis equal

xlabel(['X [' units ']'])

ylabel(['Y [' units ']'])

zlabel(['Z [' units ']'])

view(127.5,30)

else

xx = x; yy = y; zz = z;

end