ENAE441 Group Project

Tyler Chotoo
Abubakr Hamid
Richard Quarles
Blaire Weinberg

Contents

1 Overview

In this project, the goal was to locate and estimate the orbit of two GPS satellites, GPS 10 and GPS 32, given the following variables:

- Pseudorange Data between the GNSS Satellite and three chosen Ground Stations.
- The Earth-Centered, Earth-Fixed Coordinates of the three chosen Ground Stations.
- A chosen date and time between January and October of 2019.

2 Work Done

2.1 Choosing the Ground Sites and the Date

2.1.1 Ground Sites

The following ground sites were chosen for this Project:

- GODE (Greenbelt, Maryland)
- NJTR (Trenton, New Jersy)
- LYCO (Middle of Pennsylvania)

September, 29, 2019 was chosen as the observation date for both GNSS satellite. We set t1 to 20:00 UTC Time for both satellites, as this was the time where the GNSS satellites were visible to all three of the ground stations. The end time, occurred twelve hours after t1. The date was chosen at the end of the allowable date range (January to October 2019). The idea behind choosing such a recent date was to improve the accuracy of our results by using the most recent data. The orbits of satellites change with time due to Perturbations and Drag. It is believed that the most recent data will give us the most recent orbital data which will be closest to the actual, current orbital elements of the GNSS satellite.

2.2 Trilateration

Trilateration was used by creating a function following the steps outlined in the book. There were issues with the first time of the first satellite, where it wouldn't converge. This is theorized to be due to the fact that the ground stations were too close together to provide far enough points to converge.

2.3 Position Vectors

The position vector was found by using a transformation matrix to change the coordinates from an ECEF frame to an IJK frame. This was done by taking the inverse of the Rz transformation matrix and then multiplying that to the ECEF vector. The resultant is the IJK vector. This is because the IJK frame and the ECEF frame share the same z axis so the transformation was simple.

2.4 Solving the Lambert Problem

2.5 Pseudoranges

3 Conclusion

3.1 Results

We were unable to get results due to the issue with trilateration of the first satellite time value.

3.2 Sources of Error

During the analysis, we had trouble getting the Trilateration Code for G32 to Converge at t1. The Trilateration code did converge for G32 at t2 and G10 at both t1 and t2. However, the position vectors we got out of the trilateration code were off from the position vectors by each ground site's sp3 files.We tested the trilateration code using an example from the textbook (Page 146) and Assignment 7 by inputting the given ranges and reference positions into the trilateration code and comparing the output of the code with the correct answers to the example problem and the homework.The output of the code matched the textbook and assignment answers, which led us to conclude that the code was written correctly, and is not the source of the error.

The error came from the chosen ground stations being too close to each other. When the ground stations were being selected, they were selected in a way where all three ground stations were as far away from each other as possible, but within 250 km of each other so that all three stations can receive the GNSS signal simultaneously. This Ground Station Selection method ended up becoming a source of error in our results.Trilateration works best when the reference locations are spread very far apart.

3.3 Limitations

One limitation of our analysis is that the trilateration code didn't include the transmitter clock error. This would have made the analysis much more accurate. GNSS trilateration is different from normal trilateration because GNSS trilateration includes the transmitter clock error. Including the transmitter clock error makes up for the time dilation GPS satellites go through in their orbits. (according to Special and General Relativity)

The analysis was also limited by the constrained range of dates given in the project. (January to October 2019) It was said before that the orbit of the GNSS satellite will change over time due to Drag and J2 Perturbations. Since the goal of the project is to measure the orbit of the GNSS satellite, it makes sense to get the most recent data from the ground stations, so that the results of the analysis (the estimated orbital elements of the chosen GNSS satellite) will be as up-to-date and accurate as possible.

If there is one thing about the project that can be changed to improve the accuracy of the results (and rid the analysis of this limitation), it would be to get rid of the constrained range of dates, so that students could pick a date from January 2019 to any day of 2019 where GNSS data is available. This will allow students to perform the analysis on more recent, up-to-date data and return more accurate results.

A Code

A.1 Main Analysis

```
%% Load data
% clear all
% clc
% close all
run set_data.m
run set_parameters.m

%% Trilateration
rho_t1_sat1 = Trilateration(rho_obsv_t1_sat1, ref_matrix, 1e-4, rho_njtr_t1_sat1)
rho_t2_sat1 = Trilateration(rho_obsv_t2_sat1, ref_matrix, 1e-4, rho_njtr_t2_sat1)
rho_t1_sat2 = Trilateration(rho_obsv_t1_sat2, ref_matrix, 1e-4, rho_njtr_t1_sat2)
rho_t2_sat2 = Trilateration(rho_obsv_t2_sat2, ref_matrix, 1e-4, rho_njtr_t2_sat2)

%% Frame Conversion
gst_t1 = GST(T1, t1);
gst_t2_sat1 = GST(T2, t2_sat1);
gst_t2_sat2 = GST(T2, t2_sat2);
```

```
ijk_t1_sat1 = eceftoijk(rho_njtr_t1_sat1, gst_t1)
ijk_t2_sat1 = eceftoijk(rho_njtr_t2_sat1, gst_t1)
ijk_t1_sat2 = eceftoijk(rho_njtr_t1_sat2, gst_t1)
ijk_t2_sat2 = eceftoijk(rho_njtr_t2_sat2, gst_t1)

%% Lambert Solver
dt1 = seconds(t_sat1(2) - t_sat1(1));
dt2 = seconds(t_sat2(2) - t_sat2(1));

[v01, v1] = Lambert(ijk_t1_sat1, ijk_t2_sat1, dt1, 1, mu_earth, 1e-5)
[v02, v2] = Lambert(ijk_t1_sat2, ijk_t2_sat2, dt2, 1, mu_earth, 1e-5)
%% PRN
mu = 3.986*10^5;
tle1 = ['1 29486U 06042A   19272.00430744 -.00000001  00000-0  00000+0 0  9995';
        '2 29486  54.9590 248.8813 0093935   0.5082  31.6022  2.00560148 95342'];
tle2 = ['1 27663U 03005A   19272.09385061 -.00000030  00000-0  00000+0 0  9998';
        '2 27663  56.2455 312.2850 0106625  32.1585 330.1378  2.00567379122108'];
tle3 = ['1 27663U 03005A   19272.09385061 -.00000030  00000-0  00000+0 0  9998';
        '2 27663  56.2455 312.2850 0106625  32.1585 330.1378  2.00567379122108'];
tle4 = ['1 24876U 97035A   19272.18807153  .00000003  00000-0  00000+0 0  9997';
        '2 24876  55.4568 194.4010 0036176  67.5612 292.8340  2.00563944162551'];
tle5 = ['1 38833U 12053A   19272.31629780 -.00000004  00000-0  00000+0 0  9997';
        '2 38833  53.7378 244.1349 0086285  33.5442 326.9658  2.00559943 51133'];
tle6 = ['1 25030U 97067A   19272.33803917  .00000001  00000-0  00000+0 0  9994';
        '2 25030  56.0213 256.2534 0190156 231.6008 293.5417  2.00516989160455'];
tle7 = ['1 26690U 01004A   19272.35974556  .00000032  00000-0  00000+0 0  9997';
        '2 26690  53.2943 123.9467 0001296 356.1546 182.7140  1.91595261136520'];
tle8 = ['1 39741U 14026A   19272.45032143 -.00000030  00000-0  00000+0 0  9993';
        '2 39741  55.9700  67.1394 0016102 289.8941  69.9389  2.00564685 39320'];
tle9 = ['1 20959U 90103A   19272.48391087 +.00000035 +00000-0 +00000-0 0  9997';
        '2 20959 054.4426 136.9451 0017876 042.7358 317.5074 01.88156272209715'];
tle10 = ['1 32384U 07062A   19272.48624791 -.00000087  00000-0  00000+0 0  9992';
         '2 32384  56.4834  11.0588 0009646 122.0956 203.3162  2.00562300 86374'];
tle11 = ['1 32384U 07062A   19272.48624791 -.00000087  00000-0  00000+0 0  9992';
         '2 32384  56.4834  11.0588 0009646 122.0956 203.3162  2.00562300 86374'];
tle12 = ['1 28474U 04045A   19272.50293218 -.00000038  00000-0  00000+0 0  9990';
         '2 28474  54.7934  63.5258 0194068 261.9012 100.0304  2.00578637109238'];
tle13 = ['1 22231U 92079A   19272.53710043 +.00000004 +00000-0 +00000-0 0  9995';
         '2 22231 055.1771 198.1532 0032550 313.3900 046.3442 01.89107504015768'];
tle14 = ['1 40534U 15013A   19272.55094475 -.00000032  00000-0  00000+0 0  9991';
         '2 40534  54.4286 305.6404 0039273   3.5376 356.5465  2.00563196 33065'];
tle15 = ['1 40534U 15013A   19272.55094569 -.00000032 +00000-0 +00000-0 0  9996';
         '2 40534 054.4286 305.6405 0039258 003.5533 356.5309 02.00563180032636'];
tle16 = ['1 35752U 09043A   19272.59462258  .00000032  00000-0  00000+0 0  9994';
         '2 35752  54.4690 125.6613 0057421  43.7292 316.6570  2.00551811 74149'];
tle17 = ['1 35752U 09043A   19272.59462258  .00000032  00000-0  00000+0 0  9994';
         '2 35752  54.4690 125.6613 0057421  43.7292 316.6570  2.00551811 74149'];
tle18 = ['1 35752U 09043A   19272.59462258  .00000032  00000-0  00000+0 0  9994';
         '2 35752  54.4690 125.6613 0057421  43.7292 316.6570  2.00551811 74149'];
tle19 = ['1 35752U 09043A   19272.59462258  .00000032  00000-0  00000+0 0  9994';
         '2 35752  54.4690 125.6613 0057421  43.7292 316.6570  2.00551811 74149'];
tle20 = ['1 35752U 09043A   19272.59462258  .00000032  00000-0  00000+0 0  9994';
         '2 35752  54.4690 125.6613 0057421  43.7292 316.6570  2.00551811 74149'];
tle21 = ['1 26605U 00071A   19272.69514253 +.00000007 +00000-0 +00000-0 0  9990';
         '2 26605 055.0312 191.7874 0109855 249.7786 332.1308 02.00554402138357'];
tle22 = ['1 26605U 00071A   19272.69514253  .00000007  00000-0  00000+0 0  9999';
```

```
         '2 26605  55.0312 191.7874 0109855 249.7786 332.1308  2.00554402138346'];
tle23 = ['1 28474U 04045A   19272.75033867 -.00000040 +00000-0 +00000-0 0  9993';
         '2 28474 054.7936 063.5158 0194091 261.9054 278.6796 02.00578618109248'];
tle24 = ['1 41019U 15062A   19272.76047173  .00000032  00000-0  00000+0 0  9999';
         '2 41019  55.2204 127.1099 0049663 205.3918 154.4583  2.00564610 28644'];
tle25 = ['1 26407U 00040A   19272.76359852 -.00000037 +00000-0 +00000-0 0  9990';
         '2 26407 056.1862 312.4812 0193416 276.6305 083.5227 02.00559992140769'];
tle26 = ['1 38833U 12053A   19272.81488285 -.00000003  00000-0  00000+0 0  9994';
         '2 38833  53.7377 244.1143 0086298  33.5504 326.9594  2.00559949 51143'];
tle27 = ['1 38833U 12053A   19272.81488285 -.00000003 +00000-0 +00000-0 0  9995';
         '2 38833 053.7377 244.1143 0086298 033.5504 326.9594 02.00559949050366'];
tle28 = ['1 28129U 03058A   19272.88447774  .00000024  00000-0  00000+0 0  9993';
         '2 28129  53.1966 122.8369 0071095 286.1149  73.1942  2.00560713115612'];
tle29 = ['1 40294U 14068A   19272.91473815  .00000031  00000-0  00000+0 0  9990';
         '2 40294  55.2208 127.2854 0023982  39.2856 320.9842  2.00560124 36006'];
tle30 = ['1 40294U 14068A   19272.91473815  .00000031  00000-0  00000+0 0  9990';
         '2 40294  55.2208 127.2854 0023982  39.2856 320.9842  2.00560124 36006'];
tle31 = ['1 29486U 06042A   19272.95201061 -.00000000  00000-0  00000+0 0  9996';
         '2 29486  54.9587 248.8433 0093968   0.5417 355.8484  2.00560077 95359'];
[OE1] = TLE2OE(tle1, mu);
[OE2] = TLE2OE(tle2, mu);
[OE3] = TLE2OE(tle3, mu);
[OE4] = TLE2OE(tle4, mu);
[OE5] = TLE2OE(tle5, mu);
[OE6] = TLE2OE(tle6, mu);
[OE7] = TLE2OE(tle7, mu);
[OE8] = TLE2OE(tle8, mu);
[OE9] = TLE2OE(tle9, mu);
[OE10] = TLE2OE(tle10, mu);
[OE11] = TLE2OE(tle11, mu);
[OE12] = TLE2OE(tle12, mu);
[OE13] = TLE2OE(tle13, mu);
[OE14] = TLE2OE(tle14, mu);
[OE15] = TLE2OE(tle15, mu);
[OE16] = TLE2OE(tle16, mu);
[OE17] = TLE2OE(tle17, mu);
[OE18] = TLE2OE(tle18, mu);
[OE19] = TLE2OE(tle19, mu);
[OE20] = TLE2OE(tle20, mu);
[OE21] = TLE2OE(tle21, mu);
[OE22] = TLE2OE(tle22, mu);
[OE23] = TLE2OE(tle23, mu);
[OE24] = TLE2OE(tle24, mu);
[OE25] = TLE2OE(tle25, mu);
[OE26] = TLE2OE(tle26, mu);
[OE27] = TLE2OE(tle27, mu);
[OE28] = TLE2OE(tle28, mu);
[OE29] = TLE2OE(tle29, mu);
[OE30] = TLE2OE(tle30, mu);
[OE31] = TLE2OE(tle31, mu);

[states1] = OE2Cart (OE1, mu);
[states2] = OE2Cart (OE2, mu);
[states3] = OE2Cart (OE3, mu);
[states4] = OE2Cart (OE4, mu);
[states5] = OE2Cart (OE5, mu);
[states6] = OE2Cart (OE6, mu);
```

```
[states7] = OE2Cart (OE7, mu);
[states8] = OE2Cart (OE8, mu);
[states9] = OE2Cart (OE9, mu);
[states10] = OE2Cart (OE10, mu);
[states11] = OE2Cart (OE11, mu);
[states12] = OE2Cart (OE12, mu);
[states13] = OE2Cart (OE13, mu);
[states14] = OE2Cart (OE14, mu);
[states15] = OE2Cart (OE15, mu);
[states16] = OE2Cart (OE16, mu);
[states17] = OE2Cart (OE17, mu);
[states18] = OE2Cart (OE18, mu);
[states19] = OE2Cart (OE19, mu);
[states20] = OE2Cart (OE20, mu);
[states21] = OE2Cart (OE21, mu);
[states22] = OE2Cart (OE22, mu);
[states23] = OE2Cart (OE23, mu);
[states24] = OE2Cart (OE24, mu);
[states25] = OE2Cart (OE25, mu);
[states26] = OE2Cart (OE26, mu);
[states27] = OE2Cart (OE27, mu);
[states28] = OE2Cart (OE28, mu);
[states29] = OE2Cart (OE29, mu);
[states30] = OE2Cart (OE30, mu);
[states31] = OE2Cart (OE31, mu);
range = zeros(31, 1);
range(1) = norm(states1);
range(2) = norm(states2);
range(3) = norm(states3);
range(4) = norm(states4);
range(5) = norm(states5);
range(6) = norm(states6);
range(7) = norm(states7);
range(8) = norm(states8);
range(9) = norm(states9);
range(10) = norm(states10);
range(11) = norm(states11);
range(12) = norm(states12);
range(13) = norm(states13);
range(14) = norm(states14);
range(15) = norm(states15);
range(16) = norm(states16);
range(17) = norm(states17);
range(18) = norm(states18);
range(19) = norm(states19);
range(20) = norm(states20);
range(21) = norm(states21);
range(22) = norm(states22);
range(23) = norm(states23);
range(24) = norm(states24);
range(25) = norm(states25);
range(26) = norm(states26);
range(27) = norm(states27);
range(28) = norm(states28);
range(29) = norm(states29);
range(30) = norm(states30);
range(31) = norm(states31);
```

```
satg32 = (24885.268756+25027.333239+24905.66238)/3
satg10 = (24741.064469+24965.346244+24859.33414)/3
```

A.2 Coordinate Transformation

```
function rijk = eceftoijk(r,gst)
% This function takes in a column vector r containing the ECEF coordinates
% and converts it into an IJK vector.
    Rz = [cos(gst) sin(gst) 0;
          -sin(gst) cos(gst) 0;
          0 0 1];
    rijk = inv(Rz)*r;
```

A.3 Lambert Solver

```
function [v0, v] = Lambert(r0, r, dt, direction, mu, tol_t)
% This function is a lambert solver for elliptical solutions only
% Inputs:
    % r0: initial position vector
    % r : final position vector
    % dt: time of flight
    % direction: +1 for short way, -1 for long way about the orbit
    % mu: orbital parameter for whichever body is being orbited
    % tol_t: tolerance for which to break the loop
% Outputs:
    % v0: Initial velocity vector
    % v : final velocity vector
% Author: Blaire Weinberg

cosdnu = dot(r0, r) / (norm(r0) * norm(r));
A = direction * sqrt(norm(r) * norm(r0) * (1 + cosdnu));

if A == 0
    error('Cannot Calculate Orbit, A = 0');
end

% Initial States
psi_n = 0;
c2 = 1/2;
c3 = 1/6;

psi_up = 4 * pi^2;
psi_down = -4 * pi;

dt_n = 0;
loop = 0;
loop_max = 100;

while ((abs(dt - dt_n) > tol_t)&&(loop < loop_max))
    loop = loop + 1;
    yn = norm(r0) + norm(r) + (A * (psi_n * c3 - 1) / sqrt(c2));
    chi_n = sqrt(yn / c2);
    dt_n = (chi_n^3 * c3 + A * sqrt(yn)) / sqrt(mu);

    if dt_n <= dt
        psi_down = psi_n;
    else
        psi_up = psi_n;
```

```
    end

    psi_n = (psi_up + psi_down) / 2;

    if psi_n > 1e-6
     c2 = (1 - cos(sqrt(psi_n))) / psi_n;
     c3 = (sqrt(psi_n) - sin(sqrt(psi_n))) / sqrt(psi_n^3);
    elseif psi_n < -1e-6
        c2 = (1 - cosh(sqrt(-psi_n))) / psi_n;
        c3 = (-sqrt(-psi_n) + sinh(sqrt(-psi_n))) / sqrt((-psi_n)^3);
    else
        c2 = 1/2;
        c3 = 1/6;
    end
end

f = 1 - (yn / norm(r0));
g = A * sqrt(yn / mu);
gdot = 1 - (yn / norm(r));

v0 = (r - f .* r0) ./ g;
v = (gdot .* r - r0) ./ g;
```

A.4 Transformation Angle

```
function [gst] = GST(T, t)
% Determines the Greenwich Sidereal Time
% Inputs:
    % T: Vector containing centuries since J2000 (UTC)
    % t: Vector containing seconds from midnight to the observation
% Outputs:
    % gst: Greenwich Sidereal Time in degrees
% Author: Blaire Weinberg
w = 0.004178074; % deg/s

gst0 = 100.4606184 + (36000.77004 .* T) + (0.000387933 .* T.^2) - (2.583e-8 .* T.^3)
    ;
gst = gst0 + (w .* t);
```

A.5 General Orbital Parameters

```
% Script to initialize global parameters/formatting for ENAE 404
% Inludes things like fundamental constants, planetary radii, etc.
% Author: Blaire Weinberg

% Formatting
format long g
clc
clear all

% Fundamental Constants
c = 3e5; % km/s

% Gravitational Parameters
mu_earth = 3.986e5; % km^3 / s^2
mu_mars = 4.305e4; % km^3/s^2
mu_saturn = 3.7931187e7; % km^3/s^2
```

```
mu_moon = 0.00490e6; % km^3/s^2 (nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html)
mu_jupiter = 126.687e6; % km^2/s^2 (nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html)
mu_mercury = 0.022032e6; % km^2/s^2 (https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html)
mu_sun = 132712e6; % km^2/s^2 (https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html)

% Radii
r_earth = 6378; % km
r_moon = 1738.1; % km (nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html)
r_jupiter = 71492; % km (nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html)
r_mercury = 2439.7; % km (https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html)
r_mars = 3396.2; % km (https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html)
r_geo = 42164; % km

% Semi-major axes
a_mercury = 57.91e6; % km (https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html)
a_earth = 149.6e6; % km (https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html)
a_jupiter = 778.57e6; % km (nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html)

% Mass
m_mercury = 0.33011e24;% kg (https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html)
m_jupiter = 1898.19e24; % kg (nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html)
m_sun = 1988500e24; % kg (nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html)

% Gravitational Parameters
g_earth = 9.798e-3; % km/s (https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html)
```

A.6 Project-Specific Data

```
% This script contains all of the data pulled from online for the project
% Author: Blaire Weinberg

%% G32
t_sat1 = [datetime(2019, 09, 29, 20, 0, 0);
          datetime(2019, 09, 30, 3, 12, 30)]; % YY-MM-DD hh:mm:ss

% [GODE; LYCO; NJTR]
rho_obsv_t1_sat1 = [25051697.711; 25250311.013; 25300673.18]; % m
rho_obsv_t2_sat1 = [24885268.756; 25027333.239; 24905662.38]; % m

rho_gode_t1_sat1 = [-16495.751222; -20282.441992; -4990.466293]; % m
rho_gode_t2_sat1 = [20213.786583; -14550.961176; -8980.879524]; % m

rho_lyco_t1_sat1 = [-16495.751222; -20282.441992; -4990.466293]; % m
rho_lyco_t2_sat1 = [20213.786583; -14550.961176;  -8980.879524]; % m

rho_njtr_t1_sat1 = [-16495.751222; -20282.441992; -4990.466293]; % m
```

```
rho_njtr_t2_sat1 = [20213.786583; -14550.961176; -8980.879524]; % m

%% G10
t_sat2 = [datetime(2019, 09, 29, 20, 0, 0);
          datetime(2019, 09, 30, 1, 22, 30)]; % YY-MM-DD hh:mm:ss

% [GODE; LYCO; NJTR]
rho_obsv_t1_sat2 = [23211963.276; 23183895.657; 23332124.46]; % m
rho_obsv_t2_sat2 = [24741064.469; 24965346.244; 24859334.14]; % m

rho_gode_t1_sat2 = [-16660.548228; -11669.484643; 17253.892228]; % m
rho_gode_t2_sat2 = [11443.232098; -19758.768196; -13321.664005]; % m

rho_lyco_t1_sat2 = [-16660.548228; -11669.484643; 17253.892228]; % m
rho_lyco_t2_sat2 = [11443.232098; -19758.768196; -13321.664005]; % m

rho_njtr_t1_sat2 = [-16660.548228; -11669.484643; 17253.892228]; % m
rho_njtr_t2_sat2 = [11443.232098; -19758.768196; -13321.664005]; % m

%% Ground Stations
% ECEF XYZ
GODE = [1130774.439; -4831255.071; 3994200.558]; % m
LYCO = [1080274.057; -4680045.229; 4182682.6660]; % m
NJTR = [1278261.278; -4703708.0040; 4099884.5550]; % m

ref_matrix = [GODE'; LYCO'; NJTR'];

%% Times
t1 = 20 * 3600; % s
t2_sat1 = (3 * 3600) + (12 * 60) + 30; % s
t2_sat2 = 3600 + (22 * 60) + 30; % s

T1 = (7181 + 29 + 0.5) / 32525;
T2 = (7181 + 30 + 0.5) / 32525;
```

A.7 Trilateration

```
function [r] = Trilateration(rho_obsv,ref,tolerance, r_guess)
% Determines the location of a satellite given reference information
% Inputs:
    % rho_obsv: vector containing observed ranges
    % ref: matrix containing all of the [X Y Z] information for the
    % reference locations for the observations
    % r_guess: guess [X Y Z] vector for where the satellite should be located

% Outputs:
    % r: [X Y Z] vector containing the iterated solution for where the
    % satellite is

% Author: Blaire Weinberg

r = r_guess;

rho_pred = sqrt((ref(:, 1)-r(1,1)).^2+(ref(:, 2)-r(2,1)).^2+(ref(:, 3)-r(3,1)).^2);
e = (rho_obsv-rho_pred);
while (norm(e)>tolerance)
    A = [(r(1,1)-ref(1, 1))/rho_pred(1),(r(2,1)-ref(1, 2))/rho_pred(1),(r(3,1)-ref
        (1, 3))/rho_pred(1);
```

```
        (r(1,1)-ref(2, 1))/rho_pred(2),(r(2,1)-ref(2, 2))/rho_pred(2),(r(3,1)-ref
            (2, 3))/rho_pred(2);
        (r(1,1)-ref(3, 1))/rho_pred(3),(r(2,1)-ref(3, 2))/rho_pred(3),(r(3,1)-ref
            (3, 3))/rho_pred(3)];
    rho_pred = sqrt((ref(:, 1)-r(1,1)).^2+(ref(:, 2)-r(2,1)).^2+(ref(:, 3)-r(3,1))
        .^2);
    e = (rho_obsv-rho_pred);
    r = r+(A\e);
    disp(e);
end
```

A.8 OE to Cartesian Transformation

```
function [states] = OE2Cart (oe, mu)
% Converts orbital elements to cartesian coordinates
% Inputs:
    % oe: array of orbital elements
        %[a, e, i, OMEGA, omega, nu]
    % mu: constant set in 'set_parameters'
% Outputs:
    % states: current state
        % [x y z xdot ydot zdot]

% Determine position vector
p = oe(1) * (1 - oe(2)^2);
r =  p / (1 + (oe(2) * cosd(oe(6))));
r = [r * cosd(oe(6)); r * sind(oe(6)); 0];

% Determine velocity vector
v =  sqrt(mu / p) .* [-sind(oe(6)); oe(2) + cosd(oe(6)); 0];

% Create rotation matrix
Rotate = [cosd(oe(5)) sind(oe(5)) 0; -sind(oe(5)) cosd(oe(5)) 0; 0 0 1] ...
        *[1 0 0; 0 cosd(oe(3)) sind(oe(3)); 0 -sind(oe(3)) cosd(oe(3))] ...
        *[cosd(oe(4)) sind(oe(4)) 0; -sind(oe(4)) cosd(oe(4)) 0; 0 0 1];
Rotate = Rotate';

% Rotate r, v vectors
r = Rotate * r;
v = Rotate * v;

% Set output to a standard state space vector
states = [r];
```

A.9 TLE to OE Transformation

```
function [OE] = TLE2OE(TLE, mu)

i = str2double(TLE(2, 9:16));
OMEGA = str2double(TLE(2,18:25));
e = str2double(strcat('0.', TLE(2,27:33)));
omega = str2double(TLE(2, 35:42));
M = str2double(TLE(2, 44:51));
n = str2double(TLE(2, 53:63)) * 2 * pi / (3600 * 24);
a = nthroot(mu / n^2, 3);
TA = rad2deg(M+(2*e-.25*e^3)*sin(M)+1.2*e^2*sin(2*M)+(13/12)*e^3*sin(3*M));
OE = [a; e; i; OMEGA; omega; TA];
```

B Group Work Distribution

B.1 Tyler Chotoo

- List of the five digit SDC numbers for all the NAVSTAR satellites.
- Found the satellites corresponding to our chosen PRN.
- Did part five of the analysis.
- Contributed to writing the report in the work done section.
- Attempted to find results

B.2 Abubakr Hamid

-

B.3 Richard Quarles

- Ground Site Determination (using the CORS Map)
- Data Collection from the CORS Website
- Pseudo-range and Ground Site ECEF Determination (using the Observation files)
- Spreadsheet
- Wrote the Overview and 2,1.1, created the Results Tables, and contributed to the Sources of Error subjection of the report, as well as the Limitations subjection.

B.4 Blaire Weinberg

- Lambert Solver Code §A.3
- Transformation Angle Code §A.4
- General Orbit Parameters Code §A.5
- Project Specific Data Code §A.6
- Trilateration Code §A.7
- Transformation Code §A.9 §A.8
- Getting Actual Positions **??**
- Trilateration Analysis 2.2