Brief Introduction to Global Positioning Systems

Steven Williams
Embry-Riddle Aeronautical University, Daytona Beach, Florida, 32114, United States

This report will include a GPS (global positioning satellite) space mission designed with the following requirements: four simulated satellites, none can be in the same orbit, each satellite will have a period of 12 hr, the place must have a unique latitude and longitude. The project is split into three parts: mission design, simulation using MATLAB, and the effects of perturbation. By completing this project, I have successfully designed a GPS space mission with four unique orbits, used MATLAB to create 3D plots of the orbits, used coordinate transformation, and showed my understanding of space mechanics.

I. Nomenclature

$\vec{r}$	=	Position vector
$\vec{v}$	=	Velocity vector
R_E	=	Radius of Earth
ω_E	=	Earth's angular velocity
μ	=	Standard gravitational parameter
$\vec{h}$	=	Specific angular momentum
i	=	Inclination
Ω	=	Right ascension of the ascending node
e	=	Eccentricity
ω	=	Argument of perigee
θ	=	True anomaly
$\vec{p}$	=	Perturbation force
T	=	Period
c	=	Speed of light
d_i	=	Distance between satellite and receiver
t_i	=	Time taken between transmission and received
t_c	=	Time correction using fourth satellite

II. Introduction

This document will explore various aspects of GPS mission design, principles, and time correction. This will be accomplished showing derivations for equations of motions (EOMs), and MATLAB to simulate orbits. To accomplish this, I will use all aspects learned from the course including: two-body problems, orbital elements, types of orbits and trajectories, reference frames, and more. By completing this project, I will strengthen my understanding of orbital mechanics, and have real experience in mission design by designing a GPS space mission.

Through this project, I designed and simulated four geocentric orbits in ECIF*, simulated one orbit in ECEF†, and plotted the ground tracks for an orbit in ECEF. These simulations were completed in MATLAB and designed around initial $\vec{r}$ and $\vec{v}$ values.

The effects of gravitational disturbances were also explored in this project. Similar to the first part, a simulation was made for an orbit in ECIF with the effect of perturbation included. This involved a coordinate transformation from

*Earth-centered inertial frame
†Earth-centered, Earth-Fixed frame

LVLH[‡] to ECIF. I will also be using LaTeX to write this report in the hopes during my study in higher level classes, the knowledge learned from writing this report will carry over.

III. Discussion of GPS Systems based on two-body problem

A. GPS system and simplified model using four satellites

GPS is a radio-navigation system consisting of satellites communicating with ground stations to determine position and control. There are currently 31 GPS satellites in orbit around Earth.[1] These satellites are in a circular orbit with an altitude of around 20200 km, and a period T of approximately 12 hours. These satellites are also in six different orbital planes.[2]

Data from these satellites are broadcasted and used for location, navigation, tracking, mapping, and timing. We use data from these GPS satellites in our everyday lives for navigating to work or school. The collection of the 31 satellites developed by the United States are called Global Navigation Satellite Systems (GNSS). The United States has made the use of these satellites free for the international community. Other constellation of GPS satellites are operated by the Russian Federation, European Union, and China to create GLONASS.[3]

These satellites use very accurate atomic clocks to provide an extremely accurate position on Earth. A minimum of four satellites is required to determine the location and time of the satellites themselves. The time the signals are broadcasted from the satellites is tracked to account for signal delay. The time taken between the emission of the signal and when it is received is used to determine the distance between the receiver and satellite. By using the distance from three other satellites and the location of those satellites, the receiving satellite can determine its own three-dimensional position. By using a fourth satellite, an extremely accurate atomic clock is no longer needed.

B. GPS position calculation

In order to determine the position of four GPS satellites orbiting the Earth, a receiver on the ground will perform the calculations below. Where $t_{t,i}$ is the time transmitted between signals, $t_{r,i}$ is the time received, and x_1, y_1, z_1, are the coordinates of the respective satellites.

The receiver will use t_c to correct for the time correction due to an inaccurate atomic clock.

$$d_1 = c(t_{t,1} - t_{r,1} + t_c) = \sqrt{(x_1 - x)^2 + (y_1 - y)^2 + \sqrt{(z_1 - z)^2}}$$

$$d_2 = c(t_{t,2} - t_{r,2} + t_c) = \sqrt{(x_2 - x)^2 + (y_2 - y)^2 + \sqrt{(z_2 - z)^2}}$$

$$d_3 = c(t_{t,3} - t_{r,3} + t_c) = \sqrt{(x_3 - x)^2 + (y_3 - y)^2 + \sqrt{(z_3 - z)^2}}$$

$$d_4 = c(t_{t,4} - t_{r,4} + t_c) = \sqrt{(x_4 - x)^2 + (y_4 - y)^2 + \sqrt{(z_4 - z)^2}}$$

[4] Note this system is nonlinear and multi-variable.

[‡]Local vertical local horizon frame

C. GPS system orbit design using four satellites

For a single GPS orbit, there are six orbital elements to describe the orbit. These six orbital elements, their equations, and what they are used for will be shown below. Certain orbital elements must be close to a certain value to generate the orbit needed for this mission. This mission requires a circular orbit with a period of 12 hours, and an altitude close to 20200 km.

The six orbital elements used will be: h, i, Ω, e, ω, θ.

In order to calculate these six orbital elements, initial position $\vec{r}$ and velocity $\vec{v}$ vectors are provided. Sample calculations are shown below to solve for the orbital elements.

Distance:

$$r = \sqrt{\vec{r} \cdot \vec{r}} \tag{1}$$

Speed:

$$v = \sqrt{\vec{v} \cdot \vec{v}} \tag{2}$$

Radial Velocity:

$$v_r = \vec{r} \cdot \vec{v}/r \tag{3}$$

The radial velocity will tell us if the satellite is flying away or toward perigee. If $v_r > 0$, it is going away perigee; if $v_r < 0$, it is going toward perigee.

Specific angular momentum:

$$\vec{h} = \vec{r} \times \vec{v} \tag{4}$$

From 4 we can calculate the magnitude of the specific angular momentum. This is the first of our six orbital elements.

$$h = \sqrt{\vec{h} \cdot \vec{h}} \tag{5}$$

The second orbital element is the inclination:

$$i = \cos^{-1}(h_z/h) \tag{6}$$

The inclination must be greater than 0° and less than 180°. With inclination being calculated with the inverse cosine function, there is no quadrant ambiguity.

Calculation of node line vector where $\hat{K}$ is [0 0 1]:

$$\vec{N} = \hat{K} \times \vec{h} \tag{7}$$

And the magnitude of the node line:

$$N = \sqrt{\vec{N} \cdot \vec{N}} \tag{8}$$

Using the x-component from the node line vector and the magnitude, the third orbital element: the right ascension of the ascending node can be calculated:

$$\Omega = \cos^{-1}(N_x/N) \tag{9}$$

For the purposes of our GPS satellite mission, quadrant ambiguity will not be factored for the ascending node.

Now the fourth orbital element, eccentricity can be calculated. For the GPS space mission, an eccentricity close to 0 is ideal meaning the orbit is circular:

$$e = \sqrt{1 + \frac{h^2}{\mu^2}(v^2 - \frac{2\mu}{r})^2} \tag{10}$$

The fifth orbital element, the amount of perigee can be found:

$$\omega = \cos^{-1}\left(\frac{\vec{N}}{N} \cdot \frac{\vec{e}}{e}\right) \tag{11}$$

The last orbital element, the true anomaly is calculated by:

$$\theta = \cos^{-1}\left(\frac{\vec{e}}{e} \cdot \frac{\vec{r}}{r}\right) \tag{12}$$

Note, while not one of the six orbital elements, a period of 12 hours is also required. The period for a circular orbit can be calculated by:

$$T = 2\pi\sqrt{\frac{r^3}{\mu}} \tag{13}$$

These orbital elements are calculated from initial conditions. For the purposes of this experiment, we require an eccentricity of 0 and an r of 20200 km.

In order to complete the MATLAB simulation, an equation of motion must be defined for a two-body problem:

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} \tag{14}$$

[5]

Note, this equation is a second order differential and cannot be solved using hand calculations, therefore MATLAB's ode45 differential equation solver is needed. In order to solve this EOM in MATLAB, we must first transform this second order into state space form:

$\ddot{\vec{r}}$ represents acceleration, where we need position.

$$\vec{r} = \begin{vmatrix} x \\ y \\ z \end{vmatrix} \qquad \dot{\vec{r}} = \begin{vmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{vmatrix}$$

The state vector can be written:

$$\vec{S} = \begin{bmatrix} \vec{r} \\ \dot{\vec{r}} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Where the the first three terms are position, and the last three are velocity. Therefore the state vector can be easily expressed in MATLAB by joining the position and velocity initial conditions.

Since the equation of motion is a second derivative, the derivative of the state vector can be expressed by:

$$\dot{\vec{s}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

Where the first three terms are $\vec{v}$ and the last three are the EOM $\ddot{\vec{r}}$. Now the second order differential can be solved using ode45 in state space form shown in section VI.

D. Perifocal frame to ECI frame using coordinate transformation

Coordinate transformation is used to convert from perifocal to ECI frame. This transformation uses the classical Euler angle sequence shown below:

$$[\vec{Q}] = [\vec{R}_3(\gamma)][\vec{R}_1(\beta)][\vec{R}_3(\alpha)] \tag{15}$$

Modifying this equation to suite a transformation from perifocal to ECI yields:

$$[\vec{Q}]_{xX} = [\vec{R}_3(\omega)][\vec{R}_1(i)][\vec{R}_3(\Omega)] \tag{16}$$

Where the first rotation angle is the right ascension of the ascending node Ω; the second rotation is the inclination angle i, and the third rotation is the argument of perigee ω. Ω is measured in ECIF while ω is measured in the perifocal frame.

Expanding:

$$[\vec{Q}_{Xx}] = \begin{bmatrix} -sin\Omega\cos i\sin\omega + \cos\Omega\cos\omega & \cos\Omega\cos i\sin\omega + \sin\Omega\cos\omega & \sin i\sin\omega \\ -sin\Omega\cos i\cos\omega - \cos\Omega\sin\omega & \cos\Omega\cos i\cos\omega - \sin\Omega\sin\omega & \sin i\cos\omega \\ \sin i\sin\omega & -\cos i\sin\omega & \cos i \end{bmatrix} \tag{17}$$

The inverse of this matrix yields:

$$[\vec{Q}_{xX}] = \begin{bmatrix} -sin\Omega\cos i\sin\omega + \cos\Omega\cos\omega & -\sin\Omega\cos i\cos\omega - \cos\Omega\sin\omega & \sin\Omega\sin i \\ \cos\Omega\cos i\sin\omega + \sin\Omega\cos\omega & \cos\Omega\cos i\cos\omega - \sin\Omega\sin\omega & -\cos\Omega\sin i \\ \sin i\sin\omega & \sin i\cos\omega & \cos i \end{bmatrix} \tag{18}$$

With the given position and velocity vectors in the geocentric equatorial frame, the state vectors for the perifocal frame can be found by:

$$\vec{r}_x = \begin{bmatrix} \overline{x} \\ \overline{y} \\ 0 \end{bmatrix} \tag{19}$$

$$\vec{v}_x = \begin{bmatrix} \overline{v_x} \\ \overline{v_y} \\ 0 \end{bmatrix} \tag{20}$$

Now the position and velocity vectors in perifocal can be found:

$$\vec{r_X} = [\vec{Q}_{xX}](\vec{r}_x) \tag{21}$$

$$\vec{v_X} = [\vec{Q}_{xX}](\vec{v}_x) \tag{22}$$

E. ECI to ECEF conversion using coordinate transformation

When calculating the latitude, longitude, and altitude of a satellite; the position of either an observer on Earth relative to the ECI frame or position of the satellite relative to the Earth is needed. In this case, the Greenwhich meridian was devised. This meridian runs north-south as the zero line for astronomical observations.[6] This line passes through the United Kingdom and terminates at the poles of the Earth.

Since ECI is an inertial non-rotating frame, when using ECEF, a rotating frame, the Earth's rotation with respect to celestial objects must be defined. This is called sidereal time. Greenwich mean sidereal time is "the hour angle of the average position of the vernal equinox, neglecting short term motions of the equinox due to nutation."[7]

Whenever time measurements are used, such as in GMST, a reference point is needed. An Epoch is a date used when describing astronomical objects. The current standard point is epoch J2000 which frame consists of an (X, Y, Z) Cartesian frame. J2000 is equivalent to noon on the 1st of January, 2000 in GMT time. The Z-axis in J200 is the normal to the mean equator at the date J2000. The X-axis is the intersection of the equatorial plane and ecliptic plane, this is called vernal equinox as discussed before. For the Y-axis, a simple cross product calculation is used of Z cross X.[8]

The transformation between ECIF and ECEF can be described using one matrix manipulation shown:

$$\vec{r} = (x' \cos\omega - y' \sin\omega)\hat{I} + (x' \cos\omega + y' \cos\omega)\hat{J} + z'\hat{K} \tag{23}$$

Therefore:

$$X = x' \cos\omega - y' \sin\omega$$
$$Y = x' \sin\omega - y' \cos\omega$$
$$Z = z'$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{\hat{I}\hat{J}\hat{K}} = \begin{bmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}_{Q_{(\hat{i}'\hat{j}'\hat{k}')(\hat{I}\hat{J}\hat{K})}} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}_{\hat{i}'\hat{j}'\hat{k}'}$$

F. Ground Tracks

A ground track is a projection of a satellite's orbit, in this case a GPS satellite, onto the Earth's surface. A line drawn from the center of the Earth to the satellites position at a point in time will be drawn, the intersection of that line onto Earth's (spherical) surface will be a data point for the ground track line. The position of these points are called latitude and longitude. After connecting these points after a certain period of time, a ground track is created.

Onto a Mercator projection of the Earth, a ground track will create a sine wave; since the Earth rotates, there will be multiple sine waves out of phase with respect to each other. Ground tracks can be calculated using the same six orbital elements used for the ECI calculation, however, ground tracks must be calculated in the ECE frame for latitude and longitude. Therefore we use:

$$[\vec{R}_3(\theta)] = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Where θ is:

$$\theta = \omega_E(t - t_0)$$

To transform the XYZ frame to $x'y'z'$ frame. To find the components of the position vector $\vec{r}$ in the Earth-fixed $x'y'z'$ we calculate:

$$\vec{r}_{x'} = [\vec{R_3}(\theta)]\vec{r}_X$$

Where $\vec{r}_X$ is the position vector in the inertial XYZ frame. With the new position vector in the $x'y'z'$ we can calculate the six orbital elements in the $x'y'z'$ frame.

IV. Orbits under external perturbation

Since the Earth is an oblate spheroid, a satellite will not experience equal forces throughout its entire orbit, even when perfectly circular (e = 0). The Earth's oblateness creates an unequal variation with latitude. The expression for this gravitational acceleration can be expressed by a variation of the EOM, with an added perturbation force $\vec{p}$. This section will explore the effect of perturbation on a satellite rather than a simplified two-body problem.

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} + \vec{p} \tag{25}$$

[5]

A. Earth's oblateness assumption

The variation with latitude is called a zonal variation. The quantifiable effect of this oblateness is called J_2, the second zonal harmonic which is a universal constant.[7] Each planet has its own J_2.

When breaking the perturbation force $\vec{p}$ into its components we find in the LVLH frame:

$$\vec{p} = p_r\hat{u_r} + p_\perp\hat{u_\perp} + p_h\hat{h}$$

Where $\hat{u_r}$ is the radial unit vector, $\hat{u_\perp}$ is the transverse unit vector, and $\hat{h}$ is the normal unit vector all attached to the satellite. Note this perturbation force is orders of magnitude smaller than the first term in the EOM. $\hat{u_\perp}$ lies in the orbital plane and points toward the direction of motion.

In order to find these components, we use a function of orbital parameters as well as the radius of the planet R:

$$p_r = -\frac{\mu}{r^2}\frac{3}{2}J_2(\frac{R}{r})^2[1 - 3\sin^2(\omega + \theta)]$$

$$p_\perp = -\frac{\mu}{r^2}\frac{3}{2}J_2(\frac{R}{r})^2\sin^2 i\sin[2(\omega + \theta)]$$

$$p_h = -\frac{\mu}{r^2}\frac{3}{2}J_2(\frac{R}{r})^2\sin 2i\sin(\omega + \theta)$$

The six orbital elements remain unchanged except for:

$$\dot{\Omega} = \frac{h}{\mu}\frac{\sin(\omega + \theta)}{\sin i(1 + e\cos\theta)}p_h \tag{26}$$

$$\dot{\omega} = -\frac{r\cos\theta}{eh}p_r + \frac{(2 + e\cos\theta)\sin\theta}{eh}p_\perp - \frac{r\sin(\omega + \theta)}{h\tan i}p_h \tag{27}$$

B. Conversion between local-vertical, local-horizontal (LVLH) frame and ECI frame

Transformation from ECI frame to local-vertical, local-horizontal frame can be performed using matrix operations. Where x^L, y^L, z^L are components in the LVLH frame. Components in ECI frame will be denoted by ^{E}.

$$x^L = \frac{\vec{r^E}}{|\vec{r^E}|}$$

$$y^L = \frac{\vec{v^E}}{|\vec{v^E}|}$$

$$z^L = \frac{x^L \times y^L}{|x^L \times y^L|}$$

Note in the next matrices, x^L, y^L, z^L are column vectors.

$$\vec{r}_L^E = \begin{bmatrix} x^L & y^L & z^L \end{bmatrix}$$

$$\vec{r}_E^L = \begin{bmatrix} x^L & y^L & z^L \end{bmatrix}^T$$

Note, transformation is only allowed for circular orbits. In the case of GPS satellites, this works.

For velocity, transformation can be expressed:

$$\vec{v}_L^E = \begin{bmatrix} v_x^L & v_y^L & v_z^L \end{bmatrix}$$

$$\vec{v}_E^L = \begin{bmatrix} v_x^L & v_y^L & v_z^L \end{bmatrix}^T$$

Therefore the new equation of motion follows:

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r^E}[1 - \frac{3}{2}J_2(\frac{R_E}{r})^2(5\frac{(r_z^E)^2}{r^2} - 1)] \tag{28}$$

[8]

V. Additional topics for extra credit

None

VI. Simulation

A. Simulation of four GPS satellite orbits in ECIF via MATLAB

The first simulation shows four GPS satellites plotted in 3D. This was done using ode45 to solve the second order EOM defined in 14.

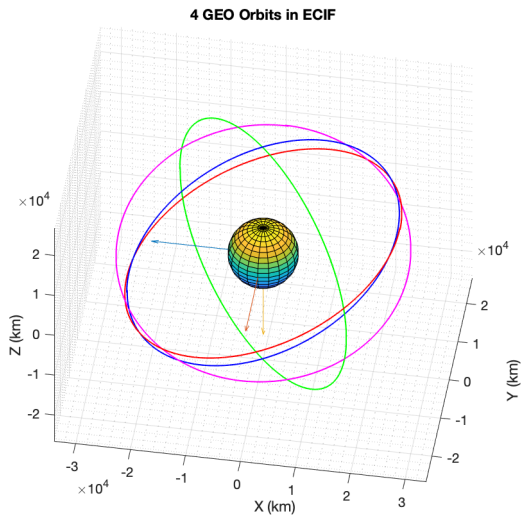

Fig. 1 Four circular orbits defined by initial position and velocity vectors for the GPS satellites designed in the space mission

Output for program file **A.1** *AE313GEOECIF.mlx*

```
Given r = [
            -22680I + -13924J +93K;
            24910I + -101J +9691K;
            -18220I + 19400J +-180K]
Given v = [
            0.000I + 0.000J +3.870K;
            0.000I + 3.862J +0.000K;
            0.000I + 0.000J +3.870K]

Eccentricity: 0.00 0.00 0.01 0.00

Period = 43208, 43490, 43212, 43200 (sec)

Eccentricity Vectors = [
            0.0000I + 0.0000J +-0.0035K;
            -0.0000I + 0.0038J +-0.0000K;
            0.0000I + 0.0000J +0.0068K]

Semi Major Axis = 26613, 26729, 26615, 26605 (km)

Right Ascension = 148.454, 90.000, 133.205, 180.000 (degrees)
```

B. Simulation of one GPS satellite orbit in ECEF via MATLAB

After coordinate transformation of the first orbit shown in Figure 1 from ECIF to ECEF, the orbit is now shown as:

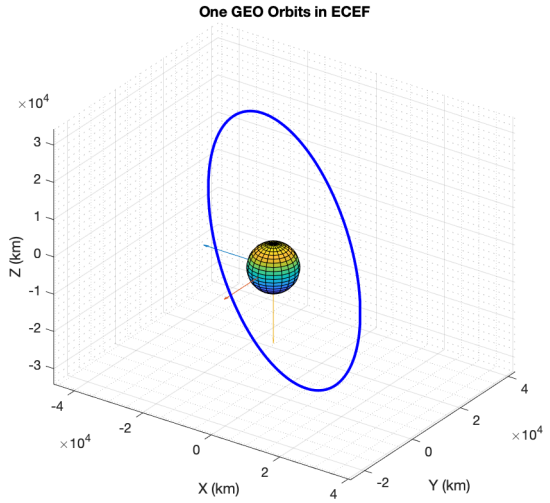

Fig. 2 One circular orbit defined by initial position and velocity vectors for a GPS satellite in ECEF

Output for program file **A.2** *AE313GEOECEF.mlx*

```
Given r = [-22680I + -13924J +93K]

Given v = [0.000I + 0.000J +3.870K]

Eccentricity: 0.25

Period = 66864 (sec)

Eccentricity Vectors = [0.0787I + -0.2400J +-0.0031K]

Semi Major Axis = 35606 (km)

```

```
Right Ascension = 287.797 (degrees)

ECEF r = [8112.722I + −25346.626J +44.805K]

ECEF v = [1.850I + −0.600J +3.870K]
```

C. Ground track for one GPS orbit in ECEF

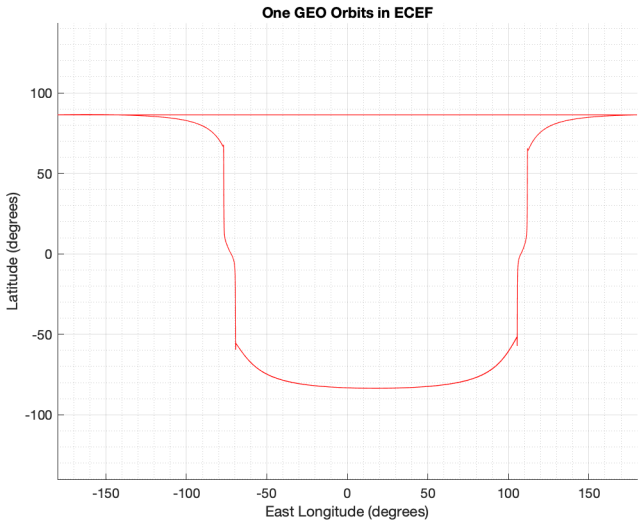

Fig. 3 Ground track for single GPS satellite orbit in ECEF

D. Effects of perturbation plotted in MATLAB for an orbit in ECIF

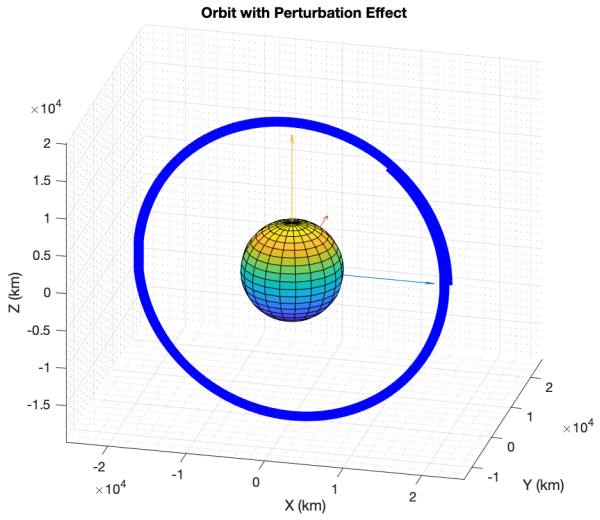

Fig. 4 GPS orbit in ECIF with perturbation force

Output for program file **A.4** *AE313PERTURBATION.mlx*

```
Given r = [20200I + 0J +0K]
Given v = [0.000I + 0.000J +4.442K]
Eccentricity: 0.00
```

```
Period = 28572 (sec)
Eccentricity Vectors = [
0.0000I + 0.0000J +0.0000K]
Semi Major Axis = 20200 (km)
Right Ascension = 0.000 (degrees)
```

VII. Conclusion

This document explored the orbital mechanics of GPS satellites. A space mission design was comprised using six orbital elements h, i, Ω, e, ω, θ. MATLAB was used to solve a second order differential equation using ode45, and plotting: four GPS satellites in ECIF, one in ECEF, and ground tracks in ECEF. The effects of perturbation were also explored and plotted showing the variation in orbit over time.

Appendix

Appendix Outline

A.1 Algorithm 1: numeric integration of two-body problem in ECIF using ode45
A.2 Algorithm 2: numeric integration of two-body problem in ECEF using ode45 with coordinate transformation
A.3 Algorithm 3: numeric integration of two-body problem in ECEF using ode45 with coordinate transformation and plotting of ground tracks
A.4 Algorithm 4: numeric integration of two-body problem in ECIF using ode45 with effect of perturbation

Program file *AE313GEOECIF.mlx* **A.1**

```
% Clear Functions
clear;
close all;
clc;
% Define Vectors for "For Loop"
r = zeros(4,1);
v = zeros(4,1);
vR = zeros(4,1);
hV = zeros(4,3);
h = zeros(4,1);
in = zeros(4,1);
NV = zeros(4,3);
K = [0,0,1;0,0,1;0,0,1;0,0,1];
N = zeros(4,1);
omegaRA = zeros(4,1);
eV = zeros(4,3);
e = zeros(4,1);
omega = zeros(4,1);
theta = zeros(4,1);
rp = zeros(4,1);
ra = zeros(4,1);
a = zeros(4,1);
T = zeros(4,1);

% Initial Conditions
rE = 6378; % Radius of Earth (km)
mu = 3.986e5; %mu for Earth

% Inital position and velocity vectors

r0 = [-22680.21,-13923.69,92.92;
    24910.21,-101,9691;
    -18220.41,19399.69,-180.213;
    0.31,25471.8221,7700];
v0 = [0,0,sqrt(mu/norm(r0(1,:)));
    0,sqrt(mu/norm(r0(2,:))),0;
    0,0,sqrt(mu/norm(r0(3,:)));
    sqrt(mu/norm(r0(3,:))),0,0];
```

```
39
40
41  % Calculating Orbital Elements
42  for i = 1:4
43
44
45      r(i) = norm(r0(i,:)); % Calc distance
46      v(i) = norm(v0(i,:)); % Calc speed
47      vR(i) = (dot(r0(i,:),v0(i,:)))/r(i); % Radial Velocity
48      hV(i,:) = cross(r0(i,:),v0(i,:)); % Vector specific angular momentum
49      h(i) = norm(hV(i,:)); % Magnitude of specific angular momentum
50      in(i) = acosd((hV(i,3))/h(i)); % Inclination
51
52
53      % Node line
54      NV(i,:) = cross(K(i,:),hV(i,:)); % Node line vector
55      N(i) = norm(NV(i,:)); % Magnitude of node line
56
57
58      omegaRA(i) = acosd(NV(i,1)/N(i)); % Right Ascension (RA) of ascending node
59
60      % Calculate eccentricity vector
61      %eV(i,:) = 1/mu*((v(i)^2-mu/r(i))*r0(i,:)-r(i)*vR(i)*v0(i,:));
62
63      eV(i,:) = 1/mu*(cross(v0(i,:),hV(i,:))-mu*(r0(i,:)/r(i)));
64      e(i) = norm(eV(i,:)); % Magnitude of eccentricity vector
65
66      omega(i) = acosd(dot(NV(i,:),eV(i,:))/(N(i)*e(i))); % Argument of perigee
67
68      theta(i) = acosd(dot(eV(i,:),r0(i,:))/(e(i)*r(i))); % True Anomaly
69
70      rp(i) = (h(i)^2/mu)*(1/(1+e(i)*cosd(0))); % Perigee radii
71      ra(i) = (h(i)^2/mu)*(1/(1+e(i)*cosd(180))); % apogee radii
72
73      %
74      a(i) = 1/2*(rp(i)+ra(i)); % Semimajor axis (km)
75
76      % a = (h^2/mu)/(1-e^2); % Semimajor axis (km)
77      % T(i) = 2*pi*(sqrt(a(i)^3/mu)); % Calculate period
78      T(i) = (2*pi)*sqrt(r(i)^3/mu);
79
80  end
81
82
83  % State Vector and Timespans
84  s0 = [r0(1,:),v0(1,:);r0(2,:),v0(2,:);r0(3,:),v0(3,:);r0(4,:),v0(4,:);]; % First Orbit
85
86  timeSpan1 = 0:T(1);
87  timeSpan2 = 0:T(2);
88  timeSpan3 = 0:T(3);
89  timeSpan4 = 0:T(4);
90  % Call ODE45 Function
91  % Note going to call multiple times for multiple orbits on same ECIF % frame
92  % Inputs: t, s,
93  % Output:
94  [¬, sol1] = ode45(@(t,s)diffEq(t,s,mu), timeSpan1, s0(1,:));
95  [¬, sol2] = ode45(@(t,s)diffEq(t,s,mu), timeSpan2, s0(2,:));
96  [¬, sol3] = ode45(@(t,s)diffEq(t,s,mu), timeSpan3, s0(3,:));
97  [¬, sol4] = ode45(@(t,s)diffEq(t,s,mu), timeSpan4, s0(4,:));
98  % Figure Configuration
99  figure;
100
101 hold on; grid on; grid minor; axis equal; rotate3d on
102 xlabel('X (km)')
103 ylabel('Y (km)')
104 zlabel('Z (km)')
105 title('4 GEO Orbits in ECIF')
106
```

```
107
108 [X, Y, Z] = sphere();
109 % Plotting Orbits and Earth
110 surf(X*rE, Y*rE, Z*rE)
111
112 % Draw Unit Vectors
113 quiver3(0,0,0,r0(1),0,0);
114 quiver3(0,0,0,0,r0(1),0);
115 quiver3(0,0,0,0,0,r0(1));
116
117 plot3(sol1(:,1),sol1(:,2),sol1(:,3),'-b','LineWidth',1)
118 plot3(sol2(:,1),sol2(:,2),sol2(:,3),'-r','LineWidth',1)
119 plot3(sol3(:,1),sol3(:,2),sol3(:,3),'-g','LineWidth',1)
120 plot3(sol4(:,1),sol4(:,2),sol4(:,3),'-m','LineWidth',1)
121
122 % Printing Outputs
123 fprintf('\nGiven r = [\n%.0fI + %.0fJ +%.0fK;\n%.0fI + %.0fJ +%.0fK;\n%.0fI + %.0fJ ...
        +%.0fK]\n\n', r0(1,1), r0(1,2), r0(1,3),r0(2,1), r0(2,2), r0(2,3),r0(3,1), r0(3,2), ...
        r0(3,3));
124 fprintf('\nGiven v = [\n%.3fI + %.3fJ +%.3fK;\n%.3fI + %.3fJ +%.3fK;\n%.3fI + %.3fJ ...
        +%.3fK]\n\n', v0(1,1), v0(1,2), v0(1,3),v0(2,1), v0(2,2), v0(2,3),v0(3,1), v0(3,2), ...
        v0(3,3));
125 fprintf('\nEccentrcity: %.2f %.2f %.2f %.2f', e(1),e(2),e(3),e(4));
126 fprintf('\nPeriod = %.0f, %.0f, %.0f, %.0f (sec)\n', T(1),T(2),T(3),T(4));
127 fprintf('\nEccentricity Vectors = [\n%.4fI + %.4fJ +%.4fK;\n%.4fI + %.4fJ +%.4fK;\n%.4fI + ...
        %.4fJ +%.4fK]', eV(1,1), eV(1,2), eV(1,3),eV(2,1), eV(2,2), eV(2,3),eV(3,1), eV(3,2), ...
        eV(3,3));
128 fprintf('\nSemi Major Axis = %.0f, %.0f, %.0f, %.0f (km)\n', a(1),a(2),a(3),a(4));
129 fprintf('Right Ascension = %.3f, %.3f, %.3f, %.3f ...
        (degrees)\n',omegaRA(1),omegaRA(2),omegaRA(3),omegaRA(4))
130
131
132
133 % ODE45 Function
134 function sdot = diffEq(t,s,mu)
135
136 % r is first three elements, v is last three
137 rDQ = s(1:3);
138 vDQ = s(4:6);
139
140 sdot(1:3,1) = vDQ; %First three elements are velocity
141 sdot(4:6,1) = (-mu*rDQ)/(norm(rDQ))^3; % Actual two body EOM
142
143
144 end
```

Program file *AE313GEOECEF.mlx* **A.2**

```
1  % Clear Functions
2  clear;
3  close all;
4  clc;
5
6  % Initial Conditions
7  rE = 6378; % Radius of Earth (km)
8  mu = 3.986e5; %mu for Earth
9
10 % Inital position and velocity vectors
11
12 r0 = [-22680.21,-13923.69,92.92];
13 v0 = [0,0,sqrt(mu/norm(r0))];
14
15 % Convert from ECIF to ECEF
16 utc = [2022 1 4 12 0 0];
17 [r_ecef,v_ecef] = eci2ecef(utc,r0,v0);
18
```

```
% Qxx = [cosd(omega), sind(omega), 0;
%     -sind(omega), cosd(omega), 0;
%     0, 0, 1];

% Convert the original matrices to vertial
% r0 = r0.';
% v0 = v0.';

% Calculate final state vector in ECEF
% rX = Qxx*r0;
% vX = Qxx*v0;
% Calculating Orbital Elements
r = norm(r_ecef); % Calc distance
v = norm(v_ecef); % Calc speed
vR = (r_ecef(1)*v_ecef(1) + r_ecef(2)*v_ecef(2) + r_ecef(3)*v_ecef(3))/r; %Radial velocity
hV = cross(r_ecef,v_ecef); % Vector specific angular momentum
h = norm(hV); % Magnitude of specific angular momentum

i = acosd(hV(3)/h); % Inclination
% Node line
K = [0,0,1];
NV = cross(K,hV); % Node line vector
N = norm(NV); % Magnitude of node line


omegaRA = acosd(NV(1)/N); % Right Ascension (RA) of ascending node

if NV(2) < 0
    omegaRA = 360-omegaRA;
end


% Calculate eccentricity vector
eV = 1/mu*((v^2-mu/r)*r_ecef-r*vR*v_ecef);
e = norm(eV); % Magnitude of eccentricity vector

omega = acosd(dot(NV,eV)/(N*e)); % Argument of perigee

theta = acosd(dot(eV,r_ecef)/(e*r)); % True Anomaly

rp = (h^2/mu)*(1/(1+e*cosd(0))); % Perigee radii
ra = (h^2/mu)*(1/(1+e*cosd(180))); % apogee radii


a = 1/2*(rp+ra); % Semimajor axis (km)
T = (2*pi)*sqrt(a^3/mu);

% State Vector and Timespans
s0 = [r_ecef;v_ecef];

timeSpan = 0:T;
% Call ODE45 Function
% Note going to call multiple times for multiple orbits on same ECIF frame
% Inputs: t, s,
% Output:
[¬, sol] = ode45(@(t,s)diffEq(t,s,mu), timeSpan, s0);
% Figure Configuration
figure;

hold on; grid on; grid minor; axis equal; rotate3d on
xlabel('X (km)')
ylabel('Y (km)')
zlabel('Z (km)')
title('One GEO Orbits in ECEF')


[X, Y, Z] = sphere();
% Plotting Orbit and Earth
```

```
surf(X*rE, Y*rE, Z*rE)

% Draw Unit Vectors
quiver3(0,0,0,r0(1),0,0);
quiver3(0,0,0,0,r0(1),0);
quiver3(0,0,0,0,0,r0(1));

plot3(sol(:,1),sol(:,2),sol(:,3),'-b','LineWidth',2);
% Printing Outputs
fprintf('\nGiven r = [\n%.0fI + %.0fJ +%.0fK]\n\n', r0(1), r0(2), r0(3));
fprintf('\nGiven v = [\n%.3fI + %.3fJ +%.3fK]\n\n', v0(1), v0(2), v0(3));
fprintf('\nEccentrcity: %.2f', e);
fprintf('\nPeriod = %.0f (sec)\n', T);
fprintf('\nEccentricity Vectors = [\n%.4fI + %.4fJ +%.4fK]', eV(1), eV(2), eV(3));
fprintf('\nSemi Major Axis = %.0f (km)\n', a);
fprintf('Right Ascension = %.3f (degrees)\n',omegaRA);


fprintf('ECEF r = [\n%.3fI + %.3fJ +%.3fK]\n\n', r_ecef(1), r_ecef(2), r_ecef(3)');
fprintf('ECEF v = [\n%.3fI + %.3fJ +%.3fK]\n\n', v_ecef(1), v_ecef(2), v_ecef(3)');


% ODE45 Function
function sdot = diffEq(t,s,mu)

% r is first three elements, v is last three
rDQ = s(1:3);
vDQ = s(4:6);

sdot(1:3,1) = vDQ; %First three elements are velocity
sdot(4:6,1) = (-mu*rDQ)/(norm(rDQ))^3; % Actual two body EOM


end
```

Program file *AE313GROUNDTRACKECEF.mlx* **A.3**

```
% Clear Functions
clear;
close all;
clc;

% Initial Conditions
rE = 6378; % Radius of Earth (km)
mu = 3.986e5; %mu for Earth
we = (2*pi + 2*pi/365.26)/(24*3600); % Earths Angular Velocity (rad/s)

% Inital position and velocity vectors

r0 = [-22680.21 -13923.69 92.92];
v0 = [0 0 sqrt(mu/norm(r0))];

% Convert from ECIF to ECEF
utc = [2000 1 1 12 0 0];
[r_ecef,v_ecef] = eci2ecef(utc,r0,v0);

% Qxx = [cosd(omega), sind(omega), 0;
%     -sind(omega), cosd(omega), 0;
%     0, 0, 1];

% Convert the original matrices to vertial
% r0 = r0.';
% v0 = v0.';

% Calculate final state vector in ECEF
% rX = Qxx*r0;
```

```
% vX = Qxx*v0;

% Calculating Orbital Elements
r = norm(r_ecef); % Calc distance
v = norm(v_ecef); % Calc speed
vR = (r_ecef(1)*v_ecef(1) + r_ecef(2)*v_ecef(2) + r_ecef(3)*v_ecef(3))/r; %Radial velocity
hV = cross(r_ecef,v_ecef); % Vector specific angular momentum
h = norm(hV); % Magnitude of specific angular momentum

i = acosd(hV(3)/h); % Inclination
% Node line
K = [0,0,1];
NV = cross(K,hV); % Node line vector
N = norm(NV); % Magnitude of node line


omegaRA = acosd(NV(1)/N); % Right Ascension (RA) of ascending node

if NV(2) < 0
    omegaRA = 360-omegaRA;
end


% Calculate eccentricity vector
eV = 1/mu*((v^2-mu/r)*r_ecef-r*vR*v_ecef);
e = norm(eV); % Magnitude of eccentricity vector

omega = acosd(dot(NV,eV)/(N*e)); % Argument of perigee

theta = acosd(dot(eV,r_ecef)/(e*r)); % True Anomaly

rp = (h^2/mu)*(1/(1+e*cosd(0))); % Perigee radii
ra = (h^2/mu)*(1/(1+e*cosd(180))); % apogee radii


a = 1/2*(rp+ra); % Semimajor axis (km)
T = (2*pi)*sqrt(a^3/mu);

% R1 = [ cos(W) sin(W) 0
%         -sin(W) cos(W) 0
%         0 0 1];
% R2 = [1 0 0
%         0 cos(incl) sin(incl)
%         0 -sin(incl) cos(incl)];
% R3 = [ cos(wp) sin(wp) 0
%         -sin(wp) cos(wp) 0
%         0 0 1];
% QxX = (R3*R2*R1)   ;



% State Vector and Timespans
s0 = [r_ecef;v_ecef];


timeSpan = 0:3*T;
% Call ODE45 Function
% Note going to call multiple times for multiple orbits on same ECIF %frame
Inputs: t, s,
Output:
[¬, sol] = ode45(@(t,s)diffEq(t,s,mu), timeSpan, s0);
% Convert to Latitude and Longitude
latlong = ecef2lla(sol(:,1:3));
% Figure Configuration
figure;

hold on; grid on; grid minor; axis equal;
xlabel('East Longitude (degrees)')
```

```
ylabel('Latitude (degrees)')

title('One GEO Orbits in ECEF');
% Plotting Groundtracks


plot(latlong(:,2), latlong(:,1), '-r');


% ODE45 Function
function sdot = diffEq(t,s,mu)

% r is first three elements, v is last three
rDQ = s(1:3);
vDQ = s(4:6);

sdot(1:3,1) = vDQ; %First three elements are velocity
sdot(4:6,1) = (-mu*rDQ)/(norm(rDQ))^3; % Actual two body EOM


end
```

Program file *AE313PERTURBATION.mlx* **A.4**

```
% Clear Functions
clear;
close all;
clc;

% Initial Conditions
rE = 6378; % Radius of Earth (km)
mu = 3.986e5; %mu for Earth
jTwo = 1.08263e-3; %J2 for Earth

% Inital position and velocity vectors

r0 = [20200,0,0];
v0 = [0,0,sqrt(mu/norm(r0))];

% Calculating Orbital Elements
r = norm(r0); % Calc distance
v = norm(v0); % Calc speed
vR = (r0(1)*v0(1) + r0(2)*v0(2) + r0(3)*v0(3))/r; %Radial velocity
hV = cross(r0,v0); % Vector specific angular momentum
h = norm(hV); % Magnitude of specific angular momentum

i = acosd(hV(3)/h); % Inclination
% Node line
K = [0,0,1];
NV = cross(K,hV); % Node line vector
N = norm(NV); % Magnitude of node line


% omegaRA = acosd(NV(1)/N); % Right Ascension (RA) of ascending node

if NV(2) < 0
    omegaRA = 360-omegaRA;
end


% Calculate eccentricity vector
eV = 1/mu*((v^2-mu/r)*r0-r*vR*v0);
e = norm(eV); % Magnitude of eccentricity vector

omega = acosd(dot(NV,eV)/(N*e)); % Argument of perigee

```

```
theta = acosd(dot(eV,r0)/(e*r)); % True Anomaly

rp = (h^2/mu)*(1/(1+e*cosd(0))); % Perigee radii
ra = (h^2/mu)*(1/(1+e*cosd(180))); % apogee radii


a = 1/2*(rp+ra); % Semimajor axis (km)
T = (2*pi)*sqrt(a^3/mu);

% State Vector and Timespans
s0 = [r0;v0];

timeSpan = 0:3*T;
% Call ODE45 Function
% Note going to call multiple times for multiple orbits on same ECIF %frame
% Inputs: t, s,
% Output:
[¬, sol] = ode45(@(t,s)diffEq(t,s,mu), timeSpan, s0);
Figure Configuration
figure;

hold on; grid on; grid minor; axis equal; rotate3d on
xlabel('X (km)')
ylabel('Y (km)')
zlabel('Z (km)')
title('Orbit with Perturbation Effect')


[X, Y, Z] = sphere();
% Plotting Orbit and Earth
surf(X*rE, Y*rE, Z*rE);

% Draw Unit Vectors
quiver3(0,0,0,r0(1),0,0);
quiver3(0,0,0,0,r0(1),0);
quiver3(0,0,0,0,0,r0(1));

plot3(sol(:,1),sol(:,2),sol(:,3),'-b','LineWidth',3);
Printing Outputs
fprintf('\nGiven r = [\n%.0fI + %.0fJ +%.0fK]\n\n', r0(1), r0(2), r0(3));
fprintf('\nGiven v = [\n%.3fI + %.3fJ +%.3fK]\n\n', v0(1), v0(2), v0(3));
fprintf('\nEccentrcity: %.2f', e);
fprintf('\nPeriod = %.0f (sec)\n', T);
fprintf('\nEccentricity Vectors = [\n%.4fI + %.4fJ +%.4fK]', eV(1), eV(2), eV(3));
fprintf('\nSemi Major Axis = %.0f (km)\n', a);
fprintf('Right Ascension = %.3f (degrees)\n',omegaRA);




% ODE45 Function
function sdot = diffEq(t,s,mu)
rE = 6378; % Radius of Earth (km)
jTwo = 1.08263e-3; %J2 for Earth


% r is first three elements, v is last three
rDQ = s(1:3);
vDQ = s(4:6);

sdot(1:3,1) = vDQ; %First three elements are velocity
sdot(4:6,1) =  ...
    -(mu*rDQ)/norm(rDQ)^3*(1-3/2*jTwo*(rE/norm(rDQ))^2*(5*(rDQ(3)^2/norm(rDQ)^2)-1)); % ...
    EOM with perturbation


end
```

References

[1] FAA, "Satellite Navigation - Global Positioning System (GPS)," 2022.

[2] FAA, "Satellite Navigation - GPS - Space Segment," *GPS - Space Segment*, 2022.

[3] FAA, "Satellite Navigation - GPS - How It Works," 2022.

[4] "Details of the GPS position calculation," *The Pennsylvania State University*, 2022.

[5] Curtis, H., *Orbital Mechanics: For Engineering Students*, Aerospace Engineering, Elsevier Science, 2015. URL `https://books.google.com/books?id=6aO9aGNBAgIC`.

[6] Fisher, R., "Astronomical Times," *Harvard University*, 2022.

[7] Navigation, and Facility, A. I., "Fundamental Concepts," *NASA Jet Propulsion Laboratory*, 2022.

[8] Scukins, E., "A Model Predictive Approach to Satellite Formation Control (Dissertation)," 2016.