Orbital Propagation

-Aryamman Bhatia as part of GNC subsystem in Mini Project Phase of IITBSSP Recruitment under mentorship of Ameya Marakarkandy

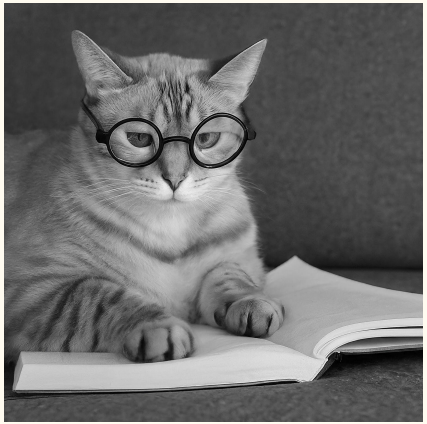
Problem Statement- Developing a simulation that finds the state of the satellite at any instant taking into account both Keplerian and perturbing forces.

Relevance- While in orbit, the satellite, for various mission objectives needs to know its state in space accurately. We need a mechanism to find this in a way that consumes low onboard power(a scarce resource) without compromising accuracy.

Expectations from a final deliverable

- Accurately outputs orbital elements at any requested instant given initial conditions/orbit of satellite.
- Information on limitations of the model in giving results within tolerance of accuracy.(confidently inaccurate outputs are dangerous)
- Easy calibration
- Make it more user friendly so as to make it possible to use not only on the satellite but also as a simulator to plan missions(like GMAT)

Week 1- Learning Centric



Re-familiarise myself with mechanics, and getting used to some terms and maths in the context of orbits.

- Reference frames and Coordinate systems
- Point mass dynamics
- Rigid body dynamics
- Keplerian Orbital Elements

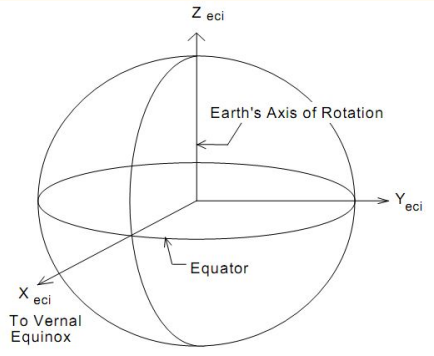
Some Reference Frames and Coordinate systems

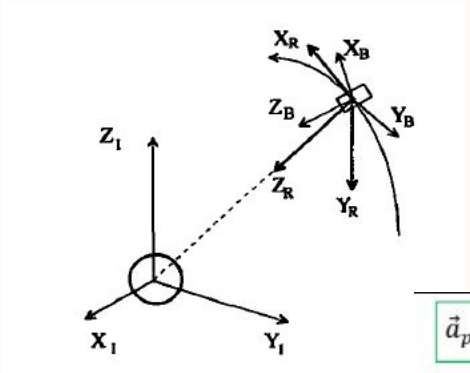
1. Earth Centred Inertial- Line joining poles as z axis, position of March equinox as x-axis, y- axis by RHR

2. J2000- earth's axis precesses over years due to various factors, changing long term equinox positions. Therefor J2000 is ECI with a time specified.

3. Body Frame- Fixed to the body, coordinate system also moves such that body is stationary in its body frame

4. Others- like R-T-N frame, perifocal frame





$$\vec{a}_p = \vec{a}_O + \dot{\vec{\Omega}} \times \vec{r} - \Omega^2 \vec{r} + 2\vec{\Omega} \times \vec{v}_{rel} + \vec{a}_{rel}$$

Basic Orbital Mechanics Results- Elliptical Orbit

- Kepler's Laws- planar elliptical orbits, time periods, h conserved. Therefore trajectory solved for.
- Basic problem statement- θ(t)
- Given initial and final theta, solve for time interval.
- Given initial theta and delta time solve for final theta.
- The latter involves implicit equations to be solved.

$$r = \frac{a(1-e^2)}{1+e\cos\theta}$$

$$\tan(\frac{v}{2}) = \sqrt{\frac{1+e}{1-e}}\tan(\frac{E}{2})$$

$$M = E - e\sin E$$

$$M = 2\pi t/T$$

Newton Raphson Iteration Method + MATLAB implementation

$$M = E - e\sin(E)$$

$$f(E) = E - e\sin(E) - M$$

Finding roots:

Start with guess:

$$If\ M < \pi : E(0) = M + \frac{e}{2}$$

$$If\ M > \pi : E(0) = M - \frac{e}{2}$$

$$E(i+1) = E(i) - \frac{f(E(i))}{f'(E(i))}$$

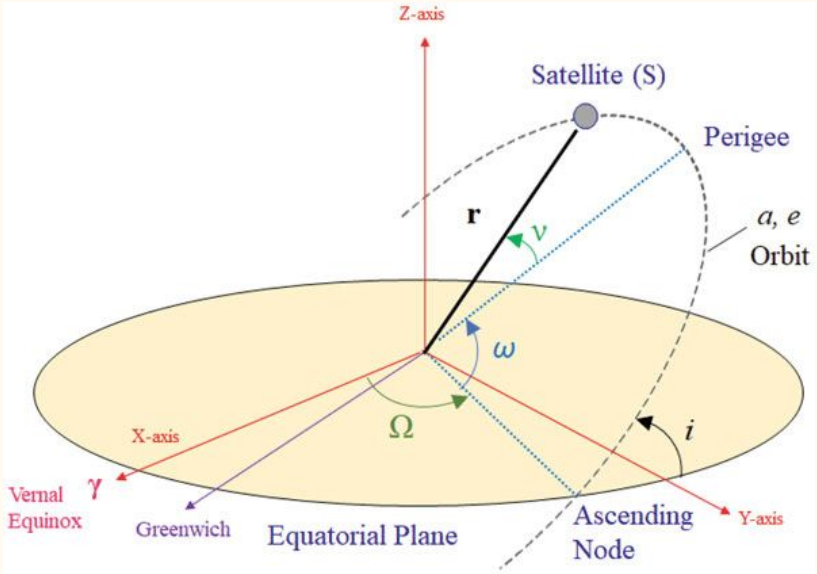
Repeat till $|f(E)| <$ tolerance

```
function nr= newtonraphson(M, e)
```

```
>> newtonraphson(0.7, 0.1)

ans =

    0.7696
```

Representing Orbits in 3d(in ECI frame)- Orbital Elements

- Preceding analysis was in the frame of this ellipse.
- How do we represent the ellipse and the position of the satellite in 3d.
- We use Orbital Elements



Converting from perifocal frame to ECI frame and vice versa

$$R_3(-\Omega) = \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1(i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix}$$

$$R_3(w) = \begin{bmatrix} \cos w & \sin w & 0 \\ -\sin w & \cos w & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Q = R_3(w)\, R_1(i)\, R_3(-\Omega)$$

$$\vec{r}_{perifocal} = Q\,\vec{r}_{geocentric} \qquad \vec{v}_{perifocal} = Q\,\vec{v}_{geocentric}$$

$$\vec{r}_{geocentric} = Q\,\vec{r}_{perifocal} \qquad \vec{v}_{geocentric} = Q\,\vec{v}_{perifocal}$$

Converting State Vector to Orbital Elements and vice versa- MATLAB

```
>> coe_from_sv([-4864.5, -2808, -4419.5], [5.1, 1.5, -5.7], 398600)

ans =

   1.0e+04 *

    5.5574    0.0000    0.0000    0.0001    0.0005    0.0006    0.7848
```

```
>> svfromcoeusingrotation([6438, 0.01, 0.1, 1, 0.5, 0.7])

ans =

   1.0e+03 *

   -3.7346
    5.1490
    0.5944
   -0.0064
   -0.0046
    0.0003
```

State Vector as a function of time- MATLAB

- Completes the problem of simulating Keplerian unperturbed orbits.
- Uses previous slide conversion techniques as well as technique to solve for θ (t) to get state vector for a requested time.

```
>> statevectortime([3734.6, 5149, 594.4, -6.4, -4.6, 0.3], 1200)

ans =

   1.0e+03 *

    7.0388
    2.6629
   -1.0780
    0.0050
    0.0041
   -0.0001
```

Week 2- Perturbations

Studying various forces which perturb satellite orbits, and how these forces are modelled mathematically

- Orbital perturbations
- Drag perturbation
- Non sphericity of earth and J2 perturbation
- Sun-synchronous orbits
- Basics of MATLAB
- Numerical ODE solvers-

  Runge Kutta Method

Introduction to Orbital Perturbations

- Various forces/phenomena exist which cause an orbit to deviate from its ideal elliptical condition.
- These forces are generally very small in magnitude compared to gravitational force.
- Some of these forces average out to zero over time and are called non secular. We do not take these into account because they do not significantly alter the orbit.
- Some forces make a net average change to the orbital elements, and therefore add up over time to make significant changes, called secular.
- Since they take time to make a significant difference, we assume a **piecewise elliptical orbit**, and accordingly update orbital elements.

Mathematical Modelling of Perturbing Forces

- All the orbital elements depend on some way or the other on ***h*** and E, which are constant in elliptical orbits.
- Due to secular perturbing forces, these change over the course of time, and therefore by chain rule the orbital elements change too.

$F = Re(R) + Ne(N) + Te(T)$: *normalised per mass*

How F *changes* h *and* E.

$$\frac{dh}{dt} = r \times F$$

$$\frac{dE}{dt} = F \,.\, v$$

Therefore other orbital elements change

Example $a = -\frac{\mu}{2E}$

$$\frac{da}{dt} = \frac{da}{dE} \cdot \frac{dE}{dt}$$

Mathematical Modelling of Perturbing Forces

Semi-major Axis

$$\dot{a} = 2\sqrt{\frac{a^3}{\mu(1-e^2)}}\,[eR\sin f + T(1+e\cos f)]$$

Eccentricity:

$$\dot{e} = \sqrt{\frac{a(1-e^2)}{\mu}}\,[R\sin f + T(\cos f + \cos E_{ecc})]$$

Inclination:

$$\frac{d}{dt}i = \sqrt{\frac{a(1-e^2)}{\mu}}\,\frac{N\cos(\omega+f)}{1+e\cos f}$$

RAAN:

$$\dot{\Omega} = \sqrt{\frac{a(1-e^2)}{\mu}}\,\frac{N\sin(\omega+f)}{\sin i(1+e\cos f)}$$

Argument of Perigee:

$$\dot{\omega} = -\dot{\Omega}\cos i + \sqrt{\frac{a(1-e^2)}{e^2\mu}}\left(-R\cos f + T\frac{(2+e\cos f)\sin f}{1+e\cos f}\right)$$

Drag Perturbation

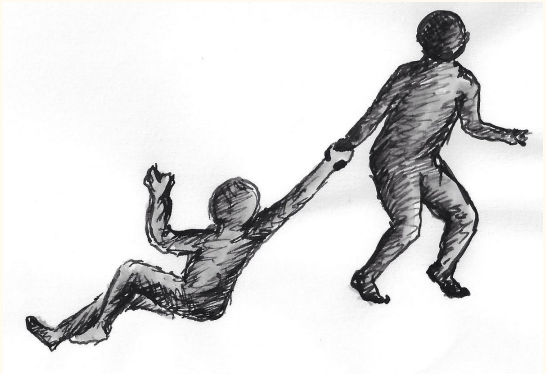
- Assumption of near circular LEO's
- Atmospheric effects still have consequences in this altitude range.

$$B = \frac{m}{C_D A}$$

$$N = R = 0, \qquad T = -\frac{1}{2}\frac{\rho}{m} C_D A v^2 = -\frac{1}{2}\frac{\rho v^2}{B}$$

Semi-Major Axis: Since $e = 0$, the dominant effect is on a.

$$a(t) = \left(\sqrt{a(0)} - 2\sqrt{\mu}\frac{\rho}{B}t\right)^2$$



Non-sphericity/Bulging of Earth

- Assumption that force acts from a point is valid under the assumption of a spherical Earth.
- In real, the Earth is not perfectly spherical, it is bulged at the equator, which causes the potential function and therefore the force to change.

$$U_{zonal}(r, \phi_{gc}) = \frac{\mu}{r} \sum_{i=2}^{\infty} J_i \left(\frac{R_e}{r} \right)^i P_i(\sin \phi_{gc})$$

- Use satellite data to fit this function for the Ji's.
- Result: J2 significantly dominates over the rest of the terms.

J2 Perturbation

$$\Delta U_{J2}(r, \phi_{gc}) = -\frac{\mu}{r}\frac{J_2}{2}\left(\frac{R_e}{r}\right)^2\left[\frac{3z^2}{r^2} - 1\right]$$

Calculate force as partial derivative of U, and later express in RTN frame to calculate perturbations.

$$\dot{\Omega}_{AV} = \left(\frac{d\Omega}{d\theta}\right)_{AV} \cdot \dot{\theta}_{AV} = \left(\frac{d\Omega}{d\theta}\right)_{AV} \cdot n = \frac{-3n}{2} J_2 \left(\frac{R}{p}\right)^2 \cos i$$

$$\left(\frac{d\omega}{d\theta}\right)_{AV} = \frac{3}{2} J_2 \left(\frac{R}{p}\right)^2 \left[2 - \frac{5}{2}\sin^2 i\right]$$

Other perturbing forces

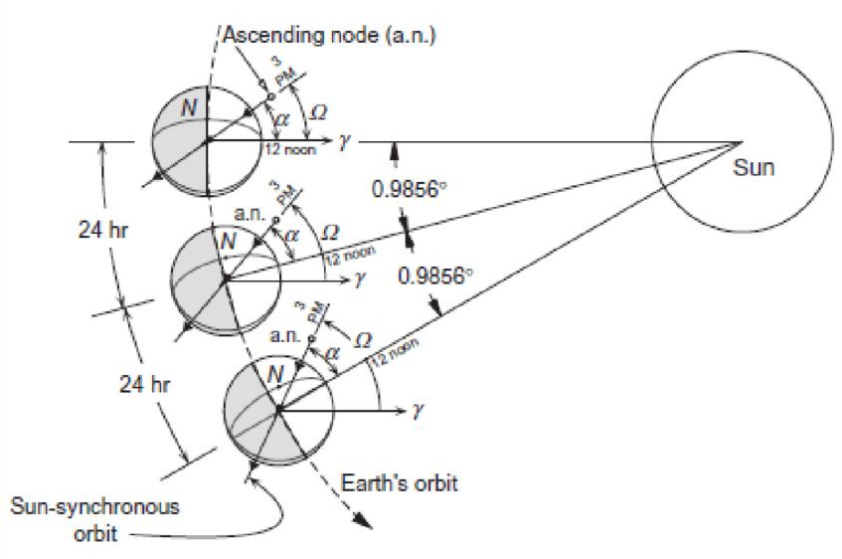
- Other perturbing forces exist, eg. moon and sun's gravity, solar radiation flux, tidal forces, etc.
- In our specific case of LEOs they do not contribute significantly compared to our current perturbations.
- Therefore we do not care about them, it would be counterproductive to.

Sun Synchronous Orbits

- Orbital plane makes constant angle with solar vector.
- Goes over same part of earth at same time of day.
- Has various applications like weather, imaging, etc.
- Surface illumination angle on planets underneath is same.

$$\frac{d\Omega}{dt} = \frac{0.9856^\circ}{day}$$

- For a given ‘a’ we can solve for an inclination.
- Most LEO’s are nearly polar since rate of change of RAAN is very low therefore cos(i) is also low.



Numerical ODE solvers, Runge Kutta Method- MATLAB implementation

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

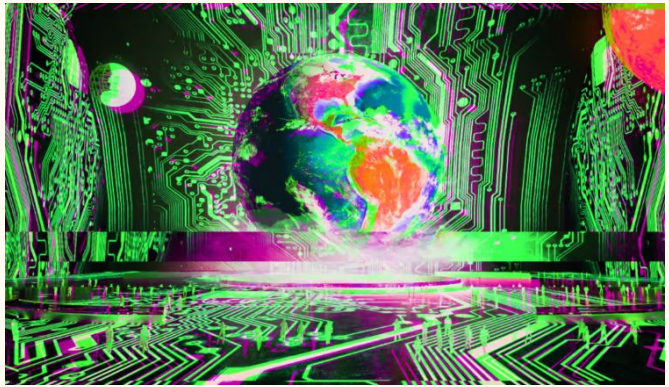
$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right), \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right), \\ k_4 &= f(t_n + h, y_n + hk_3). \end{aligned}$$

```
function rk = rungekutta(f, y, t, h, n)
%{
Applying Runge Kutta method to the ODE:
dy/dt=f(t, y)
the step size is h and the number of steps is n.
%}
```

```
>> rungekutta(test, 1, 0, 0.05, 20)

ans =

    0.1353
```

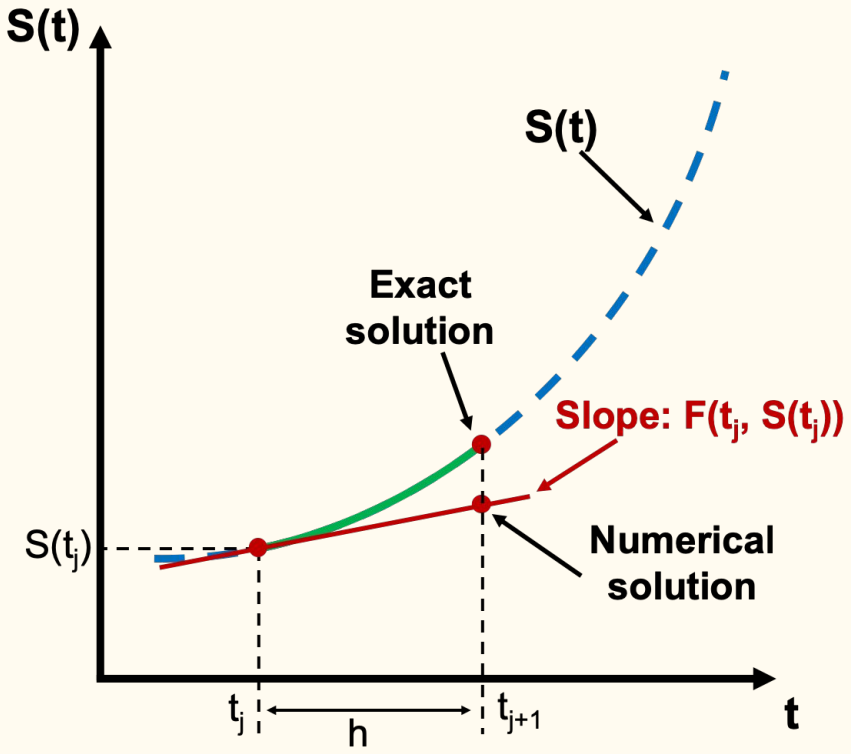
Week 3- Simulations



Coding up the final simulation on MATLAB

- Propagation of elements
- Encke's Method
- TLE
- J2 and SGP-4 propagator
- Design and implementation of LEO Sun-Synchronous Satellite Mission
- Learn how to use GMAT and analyse same orbit
- Compare MATLAB and GMAT results

Propagation of Elements- using Euler's Method



Example

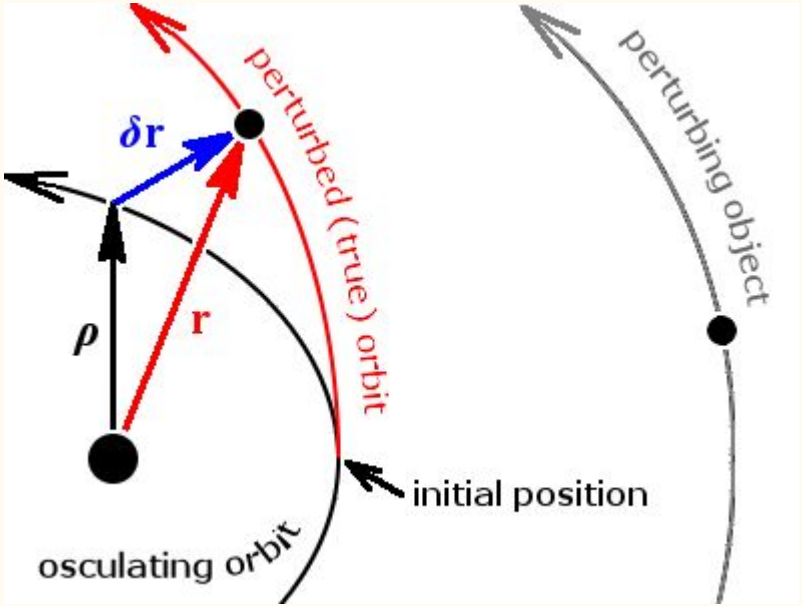
$$\Omega(t+\Delta t) = \Omega(t) + \Omega'(t).\Delta t$$

Propagation of a

- Propagation of a is problematic since it depends on the density.
- In the ionosphere, density data is very unreliable.
  (sometimes error >> 100% also)
- Implemented Jacchia Roberts model in my code, but this model requires a lot of tuning, which I am unaware how to do.

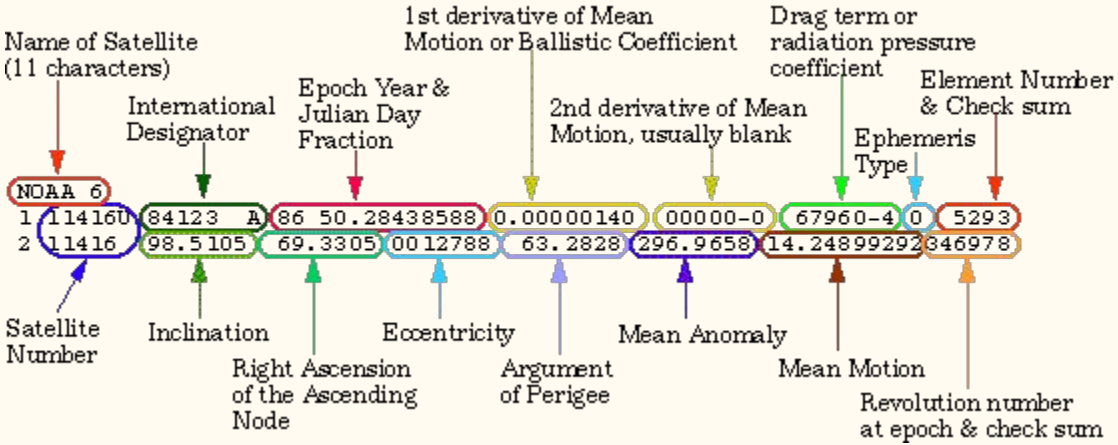
```
>> updateelements([6891, 0, 1.7, 0, 0, 0], 403000000)

ans =

   1.0e+03 *

  Columns 1 through 5

   6.839895051520148                   0   0.001700000000000   0.080600000000000  -0.283072389280984

  Column 6

                   0
```

Encke's Method



- Osculating orbit is assumed to be ideal Keplerian using state vector at some instant.
- Solve for deltar(t) using numerical integration.
- When deltar rises beyond some threshold, update the osculating orbit

TLE- Two Line Element



- Convention used to store, communicate, analyse and in general work with satellite data at a given instant.
- A satellites state at a particular instant corresponds to a particular TLE.

Designing Sun-Synchronous LEO missions

a=6645km e=0 i= 96.6 degrees(for SSO)	a=6891km e=0 i=97.4degrees

- Both having a time period of roughly 95 minutes.
- Both with a ballistic coefficient of 2.13 x 10^7 (in km units)
- Want to demonstrate how big of an effect the altitude can have on the lifetime of the satellite

Karman Line

- A line roughly 100km altitude above earth's surface, below which no satellite can survive(they tend to burn up)

Final Propagation Implementation

```
>> cosntantupdater([6645, 0, 1.686, 0, 0, 0], 833000)
```

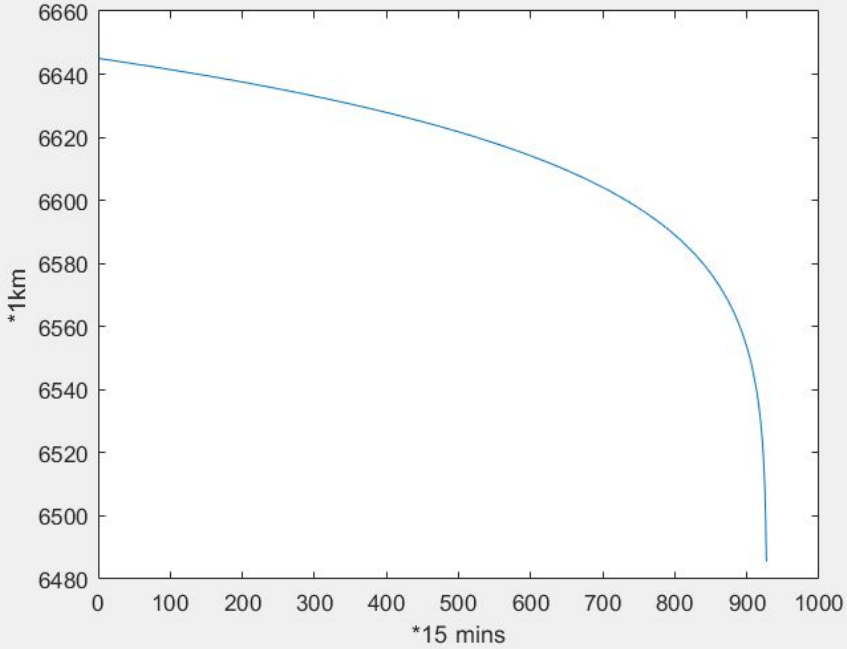
- User inputs initial elements of the satellite, and time period after which the elements/state are desired.
- Programme outputs new elements/state of satellite.

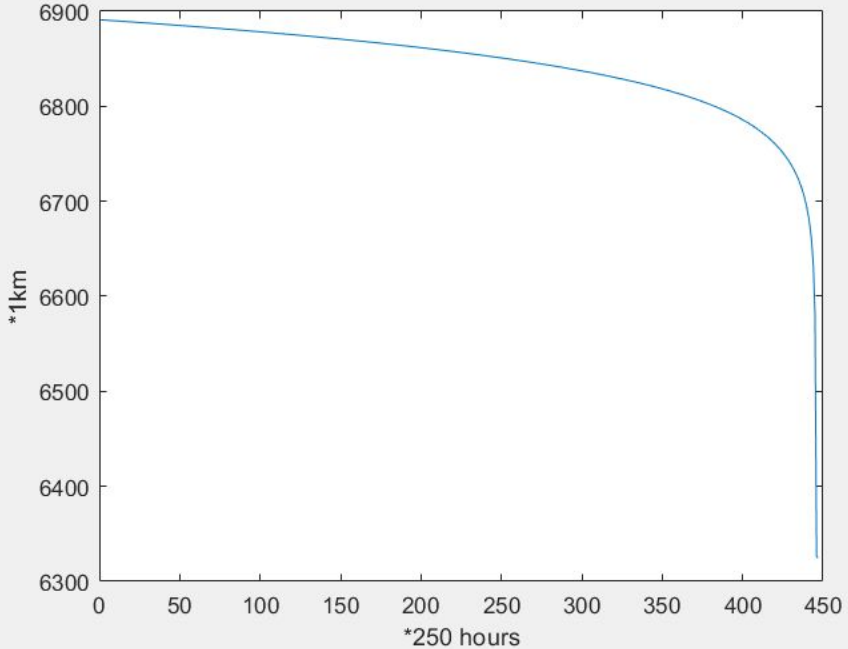
Flow

- First, the elements are constantly updated at a fixed frequency using Euler's Method.
- Care is taken that while updating theta, drag is also taken into account.
- Finally elements at users desired time are obtained.

```
 1.0e+03 *

Columns 1 through 5

 6.485449943666238                   0   0.001686000000000   0.000166600000000  -0.000686950154677

Column 6

 0.003202907072465
```

Analysis of Data Obtained for Two Orbits- SMA

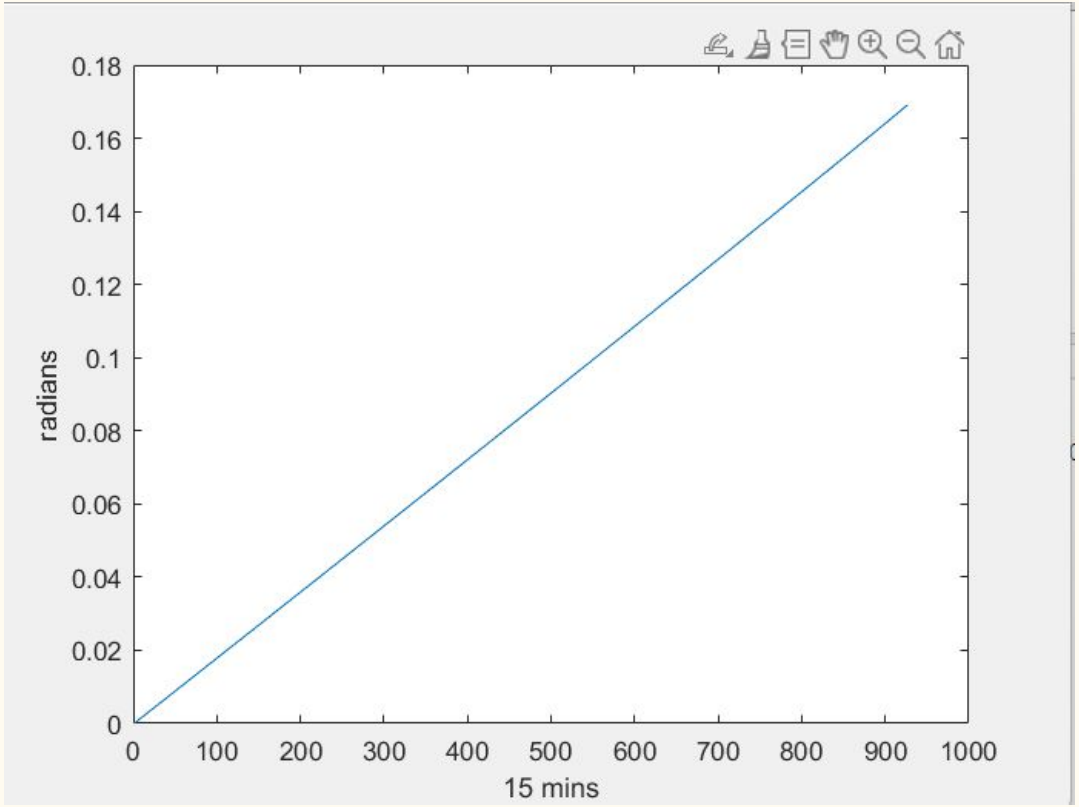




Lifetime of roughly 10 days

Lifetime of roughly 12-13 years

Analysis of Data Obtained for Orbit- RAAN



Increases linearly as expected for a SSO.

Rough Calculation of slope also gives us

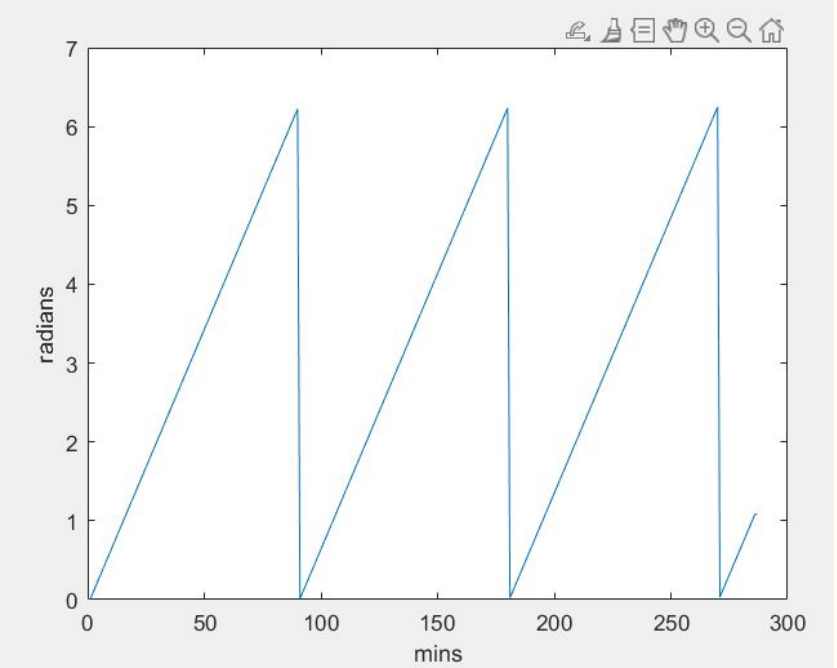
0.18rad/(15000)mins

= 0.99 degree per day
Which makes sense for SSO

Analysis of Data Obtained- Continued

- We assume that e and i do not change secularly based on the perturbing forces we take into account
- w is ill defined for circular orbits
- Now coming to θ(t)

Drag causes angular acceleration of the order 10^-12. Therefore it does not make a big change locally, but it's effects accumulate over time and that has been accounted for in the expression for θ(t)



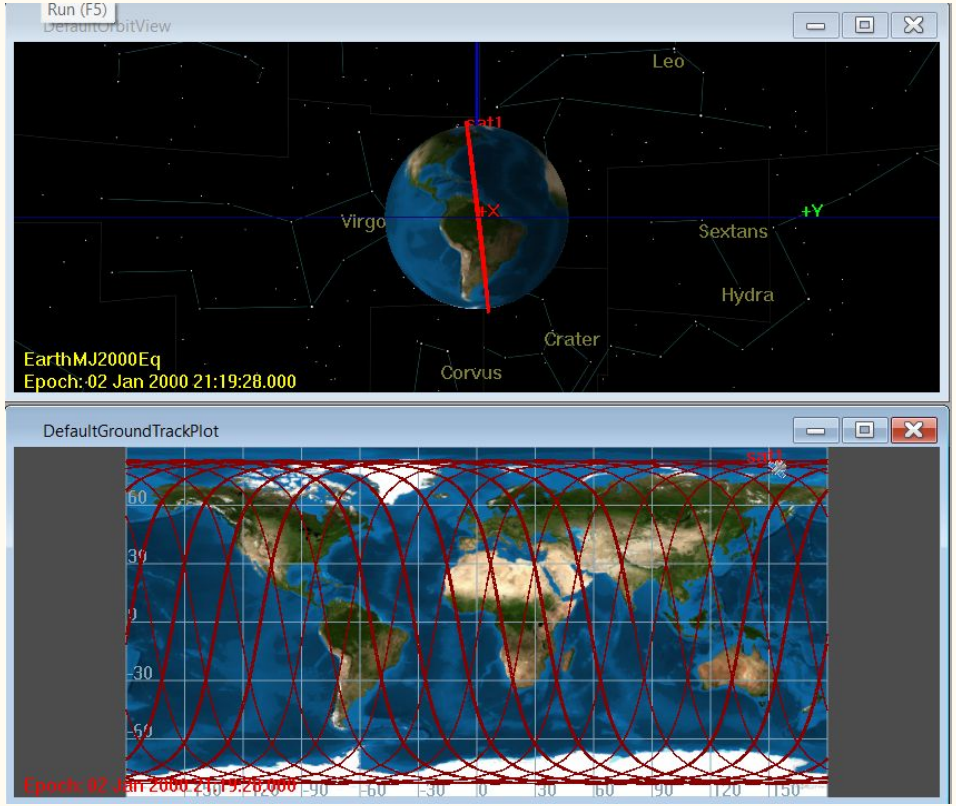
GMAT

- Used to plan and analyse missions.
- Very sophisticated, can take into account lots of forces/phenomena and run very in depth simulations.
- Also has animations to visualise the mission.

Learning how to use

- Pretty intuitive, it was mostly just specifying various parameters regarding the mission, when we want the simulation to end, etc.
- Can be customised/fine-tuned a lot more but more knowledge is required for that.
- Finally desired outputs can be exported to a text file.

GMAT



21546.36394053134	6623.899310667948	1.381181813838869	359.327962213334
21546.36495847518	6625.54989651425	1.38161779669105	0.959520133665446
21546.36597596305	6626.927114166205	1.381893486680232	1.285852044457727
21546.36699306136	6627.970811024956	1.382042572824412	1.096354594958524
21546.3680098605	6628.635320552599	1.382102729182883	0.6704946523836839
21546.36902646393	6628.891515591081	1.382114104504578	0.1394393431955651
21546.37004296606	6628.728099571282	1.382117874063659	359.5848638516285
21546.37105948834	6628.152051046967	1.38215485930286	359.0802984161576
21546.37207612011	6627.188221227349	1.382264208505472	358.7257232145195
21546.37309297551	6625.878060405049	1.382482135570391	358.7134133694132
21546.37411015045	6624.277659782014	1.382840706608251	359.5435963428167
21546.37512787924	6622.454873760057	1.383366779019048	3.173956378538962
21546.37614592521	6620.48738324823	1.384080600239513	30.23393970398997
21546.37716447021	6618.457719756608	1.384995333527478	148.9533361613641
21546.37818354733	6616.450733506486	1.386116088341065	159.6902043249592
21546.37920317051	6614.54986619517	1.387439541186118	161.6225446661461
21546.38022332396	6612.833801905177	1.388953827594235	161.7599698656048
21546.38124398621	6611.373311136804	1.390638876838203	161.2027600797816
21546.38226511124	6610.228523567857	1.392467104485998	160.2764581526783
21546.38328662955	6609.446579843648	1.394404530453218	159.1073956571652
21546.38430849481	6609.059764651341	1.396412345723353	157.7513860259455
21546.38533089747	6609.084347916441	1.398449089295432	156.2336417480402
21546.38635555302	6609.521075129106	1.400475022637617	154.5622616684629
21546.38737823125	6610.350343580033	1.402439357993026	152.7485506628264
21546.38840059819	6611.537913100667	1.404305769680434	150.7921879211184
21546.38888928683	6612.21832906356	1.405153010247618	149.80740169931

Comparing Results of GMAT vs MATLAB

- Does match for smaller time intervals, like over the course of an orbit or so.
- RAAN matches
- But ‘a’ falls over waaayyy faster in GMAT, for some reason it reaches the region where its drop is super fast way earlier, this screws everything up and causes my results not to match with GMAT.

An error I observed in GMAT- may not mean much just reporting it.

sat1.A1ModJulian	sat1.Earth.SMA	sat1.EarthMJ2000Eq.RAAN	sat1.Earth.TA
21545.00000039794	6644.999999999996	0	0
21545.00069484238	6644.885027889077	1.478779333471098e-06	0.9467278249868046
21545.00138928683	6644.579206912726	1.436269582099927e-05	1.96846049721598

TA should rise by (2/95)*360 = 7.6 degrees but it only rises by 2 degrees.

Why I believe my model is not totally useless even though the results do not match well with GMAT

- Density affects drag, drag affects a, and a affects everything else.
- Density is very volatile in ionosphere.
- Since mainly a is not matching, the discrepancy seems to be in the density model, according to GMAT, a should fall off way faster.
- Although I have integrated the same density model as GMAT, the model is highly sensitive and calibrable, and I do not know how to calibrate it well.

```
%  indata(1)  = geodetic altitude (kilometers)
%  indata(2)  = geodetic latitude (radians)
%  indata(3)  = geographic longitude (radians)
%  indata(4)  = calendar year (all digits)
%  indata(5)  = calendar month
%  indata(6)  = calendar day
%  indata(7)  = utc hours
%  indata(8)  = utc minutes
%  indata(9)  = geomagnetic index type
%               (1 = indata(12) is Kp, 2 = indata(12) is Ap)
%  indata(10) = solar radio noise flux (jansky)
%  indata(11) = 162-day average F10 (jansky)
%  indata(12) = geomagnetic activity index
```

- My model is still giving logically sound results.
- Density model can be calibrated easily once learnt how to, to produce correct results.

Some Problems Faced

- Was unable to find reliable density data- contacted mentor and found out about Jacchia Roberts model and integrated it into my code.
- A lot of bugs faced while coding- Eg. Output of 10^239 x NAN + 0.000000i Try to figure out if the logic is correct, use print functions at multiple places to figure out in what part the error is, use simpler test cases and then debug thee code and try again. If it still does not work, then re write the code of the problematic part from scratch with a fresh mind.
- In the learning phase, striking the right balance between depth and practicality. What do I need to understand in full depth and what are the things I just need to use? Optimisation is required when working on a deadline.

More stuff I could have done if time was better optimised

- Figure out how to tune the Jacchia Roberts density model, and try to get my results to match with GMAT.
- Make my programme more user friendly by implementing animations, visualisations, etc.
- Implement a control algorithm using GPS data to fine tune the algorithm.

Mistake/s I feel I made

- Started working on final algorithm slightly soon, should have read more about propagation techniques first, may have saved time in the long run.
- Following certain conventions in code/coding more neatly from day 1 could have saved time in the long run.

Future Plan: How to move forward from here to actual space system

- Take feedback on my project, and try to improve iteratively
- Read a lot more with respect to the entire subsystem in general, how attitude is controlled, working of magnetorquers, reaction wheels, etc.
- Get a more in depth idea regarding the entire subsystem by interacting with my peers more in detail about their mini-projects and how they implemented them.
- Understand from seniors how all tasks of our subsystem come together in the macro context of the satellite, develop ability to have both macro and micro in mind. Should not be totally clueless about any task given to me in the future.
- See how hardware is made/implemented irl in the lab, and how things actually manifest themselves physically- this was a software project, so that felt missing.
- Interact with all team members as much as possible to get as much exposure as possible because right now I know close to nothing.

Motivation to join SSP

- Solve real life engineering problems.
- Develop inter-disciplinary skills.
- Get good at going from theory to execution- getting things to work in general.
- Contribute to a big project such as a satellite, and see how things come together.
- Get practical exposure to parts of my branch(like control theory)
- Maybe in my time in association with SSP actually get to see a satellite we make get launched into orbit.
- Interact with smart people with varying perspectives.
- Satisfaction of seeing what I am working on coming to fruition, unlike some classes or courses.
- Want to try and learn how to maximise under various constraints associated to a student team.

Overall Experience/Feedback

- Very positive experience, got to learn and implement simultaneously which was very exciting.
- Learnt a lot of worldly skills such as posting work updates, making presentations, etc.
- Saw that I can actually get tangible outputs in a short span of 3 weeks.

Feedback

- Would be fun to do inter-disciplinary group projects/competitions, like 1 person from each subsystem, although I do not know how feasible that would be from a recruitment point of view.