A solution to Orbit Determination

Sattam Ghosal

Apr 9,2019

1 About myself/Motivation Letter:

I am Sattam Ghosal from India and presently, I am pursuing B.Tech degree(2nd year) in Computer Science and Engineering in Indian Institute of Information Technology Kalyani,India.

Physics and Mathematics are two of the most interesting subjects for me and I like implementing the laws that I come accross in physics,be it coding or numerical. I like playing with vectors and geometry.

I have been programming for six years,but I am not very old to the open source community or GSOC.I know python,java,c++,c etc.In spite of knowing these languages,I only coded the assignments that were given at college and few projects.

But my keen interest in the field of physics made me search for opportunities to solve for physics problems.It was then when I heard about GSOC from one of my classmates.

I searched for the different organizations and have gone through their problem statements.Then I found the organization aerospaceresearch.net. I went through all the problem statements,researched about the topics and found the topic "orbit determinator, community observation input" the best suitable for me.It was the opportunity,I was searching for months.

So I googled about AerospaceResearch.net and found out many positive thing for physics lovers like me.The best thing I found about aerospaceresearch.net is that this community has inputs from multiple other communities related to space.So being a part of it,I can learn much more and find some physics and mathematics background to code on.

I have gone through few of the projects undertaken by the community and like the tracking of Cubesats, rover mission ,the Buyancy Balloon Bus Lifted Experiments,the PAPELL etc.I wish to be contributing to the machine learning part of the rover missions.On this Google Summer of Code I am a planning to

work on "OrbitDeterminator: Community Observation Input of many station locations".

I will be having summer vacation from 1st June 2019 to 15 July 2019(current information from college office) and so I will have plenty of time to code if given an opportunity to work with AerospaceResearch.net.

2 My previous projects:

My GitHub profile:*(https://github.com/ghosalsattam/)*

1.Plotting a graph of a trigonometric function:*(https://github.com/ghosalsattam/Graph-plotting-program-for-trigonometric-functions-using-python)*:This program plots the graph of a function entered by the user in an interval.It can solve many complex combinations of trigonometric functions.Presently working on the introduction of algebraic functions.Applied symmetrisizing techniques.

2.Salary Management using PySide:*(https://github.com/ghosalsattam/GUI-representation-of-Salary-Management-system)*

3.Making Dyanmic Timetable Using genetic Algorithm:(https://github.com/ghosalsattam/Makin dynamic-timetable-using-genetic-algorithm)

3 My Contributions to current working code of orbit-determinator:

(Link:*https://github.com/ghosalsattam/orbitdeterminator)*

I have tried to contribute to the present version of orbit-determinator by pushing few codes, which, in my view will be helpful for the implimentation of larger parts.

1.Storing data to an online database:
(Link:*https://github.com/ghosalsattam/orbitdeterminator/tree/master/Store_To_Database)*This adds access to data observed from remote location and different communities can store their data on and use from an online database, thus increasing the communication between multiple workstations in different parts of the world.

2.Converting Topocentric data into geocentric frame:Link:*https://github.com/ghosalsattam/orbitdeterminator/tree/master/CONVERT_TO_GEOSPATIAL*:The current version allows data to be entered in only topocentric

format.This adds feature of converting the data in topocentric reference frame to geocentric reference frame and then carry on with the orbit determination.

3.Rotating a given datapath by an axis(Link: *https://github.com/ghosalsattam/orbitdeterminator/tree/master/ROTATE_BY_STANDARD_AXIS)*:This rotates the given orbit by a certain angle about x-,y- and z-axis.This feature is specifically helpful in finding an optimal plane for fitting the ellipse(rotating the curve about x-axis by inclination angle,i),thus helpful in interpolation and some of the processes discussed in consecutive sections.

4 Why Orbitdeterminator?

As mentioned earlier,about my keen interest in the field of physics and mathematics and my branch being CSE,I was finding something like "orbit determinator" to work on.Previously, my works on plotting an user "defined trigonometric function" have made me acquainted with a number of python libraries like 'scipy','numpy','matplotlib','sympy' etc.Besides ,symmetrisizing the trigonometric functions like (tan x) about $\pi/2$ have made me partially capable of handeling the smoothing part(though these are different,yet some projection concepts can be applied).
In the following paragraphs, I have tried to explain, how I am approaching to solve the problem.I have started with the filtering techniques,then went through the theoretical methods of orbitdetermination and their implementation(theory based) and at last tried to give some solution based on numerical methods(data based).
I have also taken help from the present working code of "orbit determinator" like interpolation techniques and lamberts-kalman solution.But I have tried to restrict the detailed description to only the 'features' which are not in current version.

IDEA DESCRIPTION

5 Filters

Moving Least Squares Projection Moving least squares is a method of reconstructing continuous functions from a set of unorganized point samples via the calculation of a weighted least squares measure biased towards the region around the point at which the reconstructed value is requested.
Consider a function $S = \{(x_i, f_i)|f(x_i) = f_i\}$
Then, $\sum(p(x_i) - f_i)^2\theta(x - x_i)$ is minimized where $\theta(x) = e^{-x^2}$.The following steps can be followed for smoothing a given set of data for a given orbit:
For every raw data point $\vec{r}$

1.Find a local reference line:Find a local neighborhood N_r consisting of N points. Suppose $\hat{u}$ is a unit vector in the direction of the optimal reference line and $\vec{q}$ is the projection of $\vec{r}$ on $\hat{u}$.
For every point $P_i \in N_r$

$$\sum ||(\vec{P_i} - \vec{q}) - < \vec{P_i} - \vec{q}, \hat{u} > u||^2 \theta(\vec{P_i} - \vec{q})$$

,where <> represents inner product,is minimized.We an use Powell's minimization method to calculate $\vec{q}$ *and* $\hat{u}$.
Now form a normal orthogonal basis with $\vec{q}$ as origin and $\vec{u}, \vec{r} - \vec{q}, \vec{u} \times (\vec{r} - \vec{q})$ Every point P_i in the local coordinate system can be written as $\vec{p_i} =< \vec{P_i} - \vec{q}, \vec{u} >, < \vec{P_i} - \vec{q}, \vec{r} - \vec{q} >, < \vec{P_i} - \vec{q}, \hat{u} \times (\vec{r} - \vec{q}) >$

2.Fit a local polynomial g:Use least square approximation to find a local polynomial g which satisfies the eqn:

$$\sum (g(x_i, y_i) - f_i)^2 \theta(\vec{p_i} - \vec{q})$$

We may also use interpolation.

3.Projection r' of r on g is found:

Triple Moving average: In moving average, by creating averages of full data.Weighted moving average can be used as follows: Consider a set $\{a_1, a_2, ...a_n\}$.Then, weighted average can be written as:

$$A_n = \frac{(na_1 + (n-1)a_2 ... + a_1)}{(n(n+1)/2)}$$

.This process can be used to approximate each point for smoothing and filtering noise.

Savitzky-Golay Filters:

6 Calculation of Orbital elements:

Calculation of velocity vector $\vec{v_2}$ at position vector $\vec{r_2}$:

Gibb's method: Consider 3 points with position vectors $\vec{r_1}, \vec{r_2}, \vec{r_3}$.We can construct three new vectors $\vec{D}, \vec{B}, \vec{S}$ as follows:

$$\vec{D} = \vec{r_1} \times \vec{r_2} + \vec{r_2} \times \vec{r_3} + \vec{r_3} \times \vec{r_1}$$

$$\vec{B} = \vec{D} \times \vec{r_2}$$

$$\vec{S} = (|\vec{r_2}| - |\vec{r_3}|)\vec{r_1} + (|\vec{r_3}| - |\vec{r_1}|)\vec{r_2} + (|\vec{r_1}| - |\vec{r_2}|)\vec{r_3}$$

$$\vec{N} = |\vec{r_3}|(\vec{r_1} \times \vec{r_2}) + |\vec{r_1}|(\vec{r_2} \times \vec{r_3}) + |\vec{r_2}|(\vec{r_3} \times \vec{r_1})$$

Then the velocity $\vec{v_2}$ at $\vec{r_2}$ is given by:

$$\vec{v_2} = \frac{1}{|\vec{r_2}|}\sqrt{\frac{GM}{|\vec{N}||\vec{D}|}}\vec{B} + \sqrt{\frac{GM}{|\vec{N}||\vec{D}|}}\vec{S}$$

M=mass of earth,G=Universal gravitation constant

Lambert's Solution: We can also use Lambert's Solution to find the velocity of a body at a given point.

$$V_r = \sqrt{\frac{GM}{p}}esin\theta$$

$$V_t = \sqrt{\frac{GM}{p}}(1 + ecos\theta)$$

Where V_r=Radial velocity,V_t=Tangential Velocity,p=Semiperimeter θ=True Anomaly
Implimentation: If $\vec{r_1} and \vec{r_2}$ are position vectors at time instant t1 and t2,we can use

$$l = pykep.lambert\_problem(\vec{r_1}, \vec{r_2}, t2 - t1)$$
$$v1 = l.get\_v1()[0]$$

vectors represent numpy arrays.

Calculation of $a, e, i, \omega, \Omega, \theta$: Consider a point $\vec{r}$ with velocity vector $\vec{v}$

$$Semimajor - axis, a = \frac{GM}{|\vec{v}|^2 + \frac{2GM}{|\vec{r}|}}$$

The angular momentum per unit mass

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

The eccentricity vector

$$\vec{e} = \frac{\vec{v} \times \vec{h}}{|\vec{v}|^2} - \frac{\vec{r}}{|\vec{r}|}$$

$$eccentricity, e = |\vec{e}|$$

$$inclination, i = \arccos(\frac{\hat{k}.\vec{h}}{|\vec{h}|})$$

Consider a new vector

$$\vec{N} = \hat{k} \times \vec{h}$$

$$Right\ Ascension\ for\ Ascending\ Node, \Omega = \arccos(\frac{\hat{i}.\vec{N}}{|\vec{N}|})$$

$$Argument\ of\ Perigee, \omega = \arccos \frac{\vec{N}.\vec{e}}{|\vec{N}||\vec{e}|}$$

$$True\ Anomaly, \theta = \arccos \frac{\vec{e}.\vec{r}}{|\vec{e}||\vec{r}|}$$

Gauss Method: When only angular measurements(declination,right ascension) are possible,three distinct observations are required.For a body observed at time instants t_1, t_2, t_3 at positions(vectors) $\vec{r_1}, \vec{r_2}, \vec{r_3}$ in geocentric reference system XYZ.Let $\vec{R_{E1}}, \vec{R_{E2}}, \vec{R_{E3}}$ be the position vectors of the point of observation.

$$\vec{r_1} = \vec{R_{E1}} + \vec{\rho_1} = \vec{R_{E1}} + \rho_1 \hat{u_1} \quad (1)$$

$$\vec{r_2} = \vec{R_{E2}} + \vec{\rho_2} = \vec{R_{E2}} + \rho_2 \hat{u_2} \quad (2)$$

$$\vec{r_3} = \vec{R_{E3}} + \vec{\rho_3} = \vec{R_{E3}} + \rho_3 \hat{u_3} \quad (3)$$

where ρ_i is the position of satellite measured in topocentric system and u_i is unit vector in direction of $\vec{\rho_i}$.Let δ be declination and α be right ascension.

$$\hat{u_1} = (\cos\delta_1 \cos\alpha_1)\hat{i} + (\cos\delta_1 \sin\alpha_1)\hat{j} + (\sin\delta_1)\hat{k} \quad (4)$$

$$\hat{u_2} = (\cos\delta_2 \cos\alpha_2)\hat{i} + (\cos\delta_2 \sin\alpha_2)\hat{j} + (\sin\delta_2)\hat{k} \quad (5)$$

$$\hat{u_3} = (\cos\delta_3 \cos\alpha_3)\hat{i} + (\cos\delta_3 \sin\alpha_3)\hat{j} + (\sin\delta_3)\hat{k} \quad (6)$$

The vectors r_1, r_2, r_3 lie in the same plane.

$$\vec{r_2} = c_1\vec{r_1} + c_3\vec{r_3} \quad (7)$$

Vectors r_1, r_3 can be written as

$$\vec{r_1} = f_1\vec{r_2} + g_1\vec{v_2} \quad (8)$$

$$\vec{r_3} = f_3\vec{r_2} + g_3\vec{v_2} \quad (9)$$

where f_1, g_1 are the lagerangian coefficients calculated at time t_1 and $\vec{v_2}$ is velocity of satellite at $\vec{r_2}$.
In the hypothesis we can calculate $\vec{v_2}, \vec{r_2}\ at\ t_2$ Consider equation 7:

$$c_1 = \frac{(\vec{r_1} \times \vec{r_3}).(\vec{r_2} \times \vec{r_3})}{|\vec{r_1} \times \vec{r_3}|^2}$$

$$c_3 = \frac{(\vec{r_3} \times \vec{r_1}).(\vec{r_2} \times \vec{r_1})}{|\vec{r_3} \times \vec{r_1}|^2}$$

From eqn 8 and 9

$$\vec{r_1} \times \vec{r_3} = (f_1 g_3 - g_1 f_3)\vec{h}$$

$$\vec{r_2} \times \vec{r_3} = g_3 \vec{h}$$

$$\vec{r_2} \times \vec{r_1} = g_1 \vec{h}$$

$$c_1 = \frac{g_3}{f_1 g_3 - g_1 f_3}$$

Let,

$$\tau_1 = t_1 - t_2$$

$$\tau_3 = t_3 - t_2$$

$$f_1 = 1 - \frac{1}{2}\frac{\mu}{|\vec{r_2}|^3}\tau_1^2 \qquad f_3 = 1 - \frac{1}{2}\frac{\mu}{|\vec{r_2}|^3}\tau_3^2$$

$$g_1 = \tau_1 - \frac{1}{6}\frac{\mu}{|\vec{r_2}|^3}\tau_1^3 \qquad g_3 = \tau_3 - \frac{1}{6}\frac{\mu}{|\vec{r_2}|^3}\tau_3^3$$

$$f_1 g_3 - g_1 f_3 = (\tau_3 - \tau_1) - \frac{1}{6}\frac{\mu}{r_2^3}(\tau_3 - \tau_1)^3 + \frac{1}{12}\frac{\mu^2}{r_2^6}(\tau_1^2\tau_3^3 - \tau_1^3\tau_3^2)$$

$$c_1 = \frac{\tau_3}{\tau}[1 + \frac{1}{6}\frac{\mu}{r_2^3}(\tau^2 - \tau_3^2)]$$

$$c_3 = \frac{\tau_1}{\tau}[1 + \frac{1}{6}\frac{\mu}{r_2^3}(\tau^2 - \tau_1^2)]$$

Equation 7 can also be written as :

$$c_1\rho_1\hat{u_1} - \rho_2\hat{u_2} + c_3\rho_3\hat{u_3} = -c_1\vec{r_{E1}} + \vec{r_{E2}} - c_3\vec{r_{E3}}$$

Take cross scalar product with $\hat{u_2} \times \hat{u_3}$

$$c_1\rho_1\hat{u_1}.(\hat{u_2} \times \hat{u_3})$$

$$D_0 = \hat{u_1}.(\hat{u_2} \times \hat{u_3})$$

$$D_{11} = \vec{r_{E1}}.(\hat{u_2} \times \hat{u_3})$$

$$D_{21} = \vec{r_{E2}}.(\hat{u_2} \times \hat{u_3})$$

$$D_{31} = \vec{r_{E3}}.(\hat{u_2} \times \hat{u_3})$$

Therefore

$$\rho_1 = (-D_{11} + \frac{1}{c_1}D_{21} - \frac{c_3}{c_1}D_{31})\frac{1}{D_0}$$

$$\rho_2 = (-c_1 D_{12} + D_{22} - c_3 D_{32})\frac{1}{D_0}$$

$$\rho_1 = (-\frac{c_1}{c_3}D_{13} + \frac{1}{c_3}D_{23} - D_{33})\frac{1}{D_0}$$

$D_{ij} = r_{Ei}.(\hat{u_{k1}} \times \hat{u_{k2}}), \qquad k1, k2 \neq j$

$$\rho_2 = -D_{12}\tau_3 \frac{6 + (\frac{\mu}{r_2^3})(\tau^2 - \tau_3^2)}{6D_0\tau} + \frac{D_{22}}{D_0} + D_{32}\tau_1 \frac{6 + (\frac{\mu}{r_2^3})(\tau^2 - \tau_3^2)}{6D_0\tau}$$

Put

$$A = (-D_{12}\frac{\tau_3}{\tau} + D_{22} + D_{32}\frac{\tau_1}{\tau})\frac{1}{D_0}$$

$$B = (-D_{12}\tau_3(\tau^2 - \tau_3^2) + D_{32}\tau_1(\tau^2 - \tau_1^2))\frac{1}{6D_0\tau}$$

Therefore,

$$\rho_2 = A + \frac{\mu B}{r_2^3} \qquad (a)$$

$$\rho_1 = [\frac{6(D_{31}\frac{\tau_1}{\tau_3} + D_{21}\frac{\tau}{\tau_3})r_2^3 + \mu D_{31(\tau^2 - \tau_1^2)\frac{\tau_1}{\tau_3}}}{6r_2^3 + \mu(\tau^2 - \tau_3^2)} - D_{11}]\frac{1}{D_0} \qquad (b)$$

$$\rho_3 = [\frac{6(D_{13}\frac{\tau_3}{\tau_1} - D_{23}\frac{\tau}{\tau_1})r_2^3 + \mu D_{13(\tau^2 - \tau_3^2)\frac{\tau_3}{\tau_1}}}{6r_2^3 + \mu(\tau^2 - \tau_3^2)} - D_{33}]\frac{1}{D_0} \qquad (c)$$

Equation 2 can be written as:

$$r_2^8 - (r_{E2}^2 + 2EA + A^2)r_2^6 - 2\mu B(E + A)r_2^3 - \mu^2 B^2 = 0$$

where E=$\vec{r_{E2}}.\hat{u_2}$ The solution of this equations give $|\vec{r_2}|$ Once we get $|\vec{r_2}|$ we can find ρ_1, ρ_2, ρ_3 from eqn a,b and c.Hence,equation 1,2 and 3 can be used to calculate $\vec{r_1}, \vec{r_2}, \vec{r_3}$ since $\hat{u_1}, \hat{u_1}, \hat{u_1}$ are calculated in eqn.4,5,6. Solving equation 8 and 9,

$$\vec{v_2} = \frac{f_1}{f_1 g_3 - g_1 f_3}\vec{r_3} - \frac{f_3}{f_1 g_3 - g_1 f_3}\vec{r_1}$$

.

Then we can use method described above to find the orbital elements.

7 Implimentation-Algorithm/Pseodo code:

Function GAUSS(δ, α, t, R):;

$\tau_1 = t[1] - t[2]$;

$\tau_3 = t[3] - t[2]$;

$\tau = t[3] - t[1]$;

$\vec{u_1} \leftarrow [\cos(\delta[1])cos(\alpha[1]), \cos(\delta[1])\sin(\alpha[1]), \sin(\delta[1])]$;

$\vec{u_2} \leftarrow [\cos(\delta[2])cos(\alpha[2]), \cos(\delta[2])\sin(\alpha[2]), \sin(\delta[2])]$;

$\vec{u_3} \leftarrow [\cos(\delta[3])cos(\alpha[3]), \cos(\delta[3])\sin(\alpha[3]), \sin(\delta[3])]$;

$\vec{p_1}$=$cross\_product(u_2,u_3)$;

$\vec{p_2} = cross\_product(u_1, u_3)$;

$\vec{p_3} = cross\_product(u_1, u_2)$;

$d_0 = \hat{u_1}.\vec{p_1}$;

$A = i = 1$, $j = 1$;

for $i <= 3$ **do**

  $j = 1$;

  for $j <= 3$ **do**

    D[i][j]=R[i].$\vec{p_j}$

  end

end

$A = (-D[1][2] * \tau_3/\tau + D[2][2] + D[3][2] * \tau_1/\tau)/d_0$;

$B = (D[1][2] * (\tau_3^2 - \tau^2) * \tau_3/\tau + D[3][2] * (\tau^2 - \tau_1^2) * \tau_1/\tau)/(6d_0)$;

$E = R[2].\hat{u_2}$;

$mag\_R_2 = sqrt(dot\_product(R[2], R[2])$;

$a = -(A^2 + 2AE + mag\_R_2^2)$;

$b = -2\mu * B * (A + E)$;

$c = -\mu^2 * B^2$;

$eqn \leftarrow r_2^8 + ar_2^6 + br_2^3 + c = 0$;

$solve(eqn, x)$; //*in python sympy.solve can be used*;

$\rho_1 = ((6(D[3][1] * \tau_1/\tau_3 + D[2][1]\tau/\tau_3)r_2^3 + \mu D[3][1](\tau^2 - \tau_1^2) *$
$\tau_1/\tau_3)/(6r_2^3 + \mu(\tau^2 - \tau_3^2)) - D[1][1])/d_0$;

$\rho_2 = A + \mu B/r_2^3$;

$\rho_3 = ((6(D[1][3] * \tau_3/\tau_1 + D[2][3]\tau/\tau_1)r_2^3 + \mu D[1][3](\tau^2 - \tau_3^2) *$
$\tau_3/\tau_1)/(6r_2^3 + \mu(\tau^2 - \tau_1^2)) - D[3][3])/d_0$;

$f_1 \leftarrow 1 - .5 * (\mu/r_2^3) * \tau_1^2$;

$f_3 \leftarrow 1 - .5 * (\mu/r_2^3) * \tau_3^2$;

$g_1 \leftarrow r_1 - (1/6) * (\mu/r_2^3) * \tau_1^3$;

$g_3 \leftarrow r_3 - (1/6) * (\mu/r_2^3) * \tau_3^3$ $\vec{r_1} = R[1] + \rho_1\hat{u_1}$;

$\vec{r_2} = R[2] + \rho_2\hat{u_2}$;

$\vec{r_3} = R[3] + \rho_3\hat{u_3}$;

$\vec{v_2} = (1/(f_1g_3 - f_3g_1)) * (f_1\vec{r_3} - \vec{r_1}f_1)$

EndFunction

8 Kalman Filters:

The Kalman filter uses a system's dynamic model (e.g., physical laws of motion), known control inputs to that system, and multiple sequential measurements (such as from sensors) to form an estimate of the system's varying quantities (its state) that is better than the estimate obtained by using only one measurement alone. As such, it is a common sensor fusion and data fusion algorithm.

Implimentation: Kalman Gain,

$$KG = \frac{Error_{Estimated}}{Error_{Estimated} + Error_{Measured}}$$

$$Estimation_t = Estimation_{t-1} + KG(MeasuredValue - Estimation_{t-1})$$

$$Estimation_t = \frac{Error_{measured} Error_{Estimated_{t-1}}}{Error_{Measured} + Error_{Estimated_{t-1}}}$$

9 Interpolation:

We can use Spline interpolation to find the velocity at apoint.
Algorithm:

```
function SPLINE(rawData);
    splineInput = dataPoints[index:index+2];
    spline = cubicSpline(splineInput);
    velocity = computeVelocity(spline, splineInput[0:,1:4][0])
EndFunction
```

Once we have found velocity we can find the orbital elements mentioned above.

Weighted interpolation: We can use the weighted in interpolation method for interpolation.

$$F(x, y, z) = \sum w_i f_i$$

where,w_i is the weight given to point i based on some condition.In *inverse distance interpolation* weight is inversely proportional to distance of point from the point to be interpolated.
A slight variation can be used in which,the weights of the points are inversely proportional to the optimal plane.

10 Fitting of the ellipse:

1.Fitting a Plane: We can choose an optimal plane with respect to the raw data.
2.Projection of all the points onto the plane:
3.Finding the corresponding 2D representation of the points. Convert the points in 3d with respect to a basis of 2d.

4.Define a function $f = Ax^2 + By^2 + 2Gx + 2Fy + 2Hxy + C$
5.Find an optimal conic with respect to the conic function f.
Till step 3 it is implimented on the present version of"orbitdeterminator".
Algorithm:

---

function DEFINE_CONIC(x,y,a,b,c,f,g,h);
return $(a * x^2 + b * y^2 + 2 * g * x + 2 * f * y + 2 * h * x * y + c)$
EndFunction
x,y ← x_coordinate_list,y_coordinate_list
scipy.optimize.curve_fit(DEFINE_CONIC,x,y)

---

11 Another method:

Overview:We can use the fact that in the whole ellipse curve,there are four points where the angle between the position vector and the velocity vectors changes from acute to obtuse and vice versa.Thus the dot product of the velocity vector and the position vector become 0 at 4 points.Two of them being the Periapsis and Apoapsis.Through this we can determine the semi-major axis,a.We can also use the fact that reflection of a ray passing through one of the foci passes through the other focus.
Implimentation:For the implimentation of the above method,we consider an abstract data type as described below:
1.**Members:**There are three members of the ADT.
a)Position vector $\vec{r}$
b) velocity vector $\vec{v}$
c)A Scalar flag f
2.**Operations:**It supports the operation dot_product.
Dot_product(i): It performs the dot product of the vectors prsent at ith index and stores 1 at f if(dot product¿0) else -1
Consider two points near the periapsis,one on each side of the periapsis.One has f=+1 and one have f=-1.We can take few surrounding points and apply interpolation technique to find periapsis distance.

In the pseudo codes, the V represent an array of object of the class DetDot.
Algorithm

Define the ADT

```
class DetDot
    Array r
    Array v
    int f
```

Take out dot_product node by node:

```
function CALCULATE_DOT(V):s
    i ← 0
    for i<len(V) do
        if dot_product(V[i].r,V[i].v)>0 then
            V[i].f=1
        else if dot_product(V[i].r,V[i].v)<0 then
            V[i].f=-1
        else
            V[i].f=0 //This may be periapsis or apoapsis
        end
    end
EndFunction
```

Among the few points with dot product 0 finds periapsis and apoapsis:

```
function FIND_PERIAPSIS_AND_APOAPSIS(V) i ← 0
    dot_0 ←[]//Stores possible indexes of apoapsis and periapsis points
    for i<len(V)-1 do
        if V[i].f==0 then
            dot_0.append(i)
        end
        else if (V[i].f>0 and V[i+1].f<0) or (V[i].f<0 and V[i+1].f>0)
 then
            dot_0.append(i)
        end
    end
    apo ← max(dot_0)//Extract max dist from (0,0,0)
    per ← min(dot_0)
    N_apo ← points ∈ Neighbour(apo)
    N_per ← points ∈ Neighbour(per)
    dist_apr[] = dist of each point in N_apo from(0,0,0)
    /*Calculating the real apoapsis which lies nearest to point apo*/
    fn=scipy.interpolate.RegularGridInterpolator(N_apo, dist_apr)
    pts=numpy.linspace(apo,V(index(apo)+1).r,1000)
    apoapsis_length ← maximum(fn(pts))
    /*Apply same method for geting periapsis*/
a=(apoapsis_length+periapsis_length)/2

EndFunction
```

Determination of eccentricity: Consider a ray of light passing through one of the foci and gets reflected at a point on the boundary.Before,the velocity has been calculated,so we know the direction cosines of the tangent passing through the point.So we can find out the direction of reflected ray.The point of intersection of the ray with vector $\vec{a}$ gives the coordinate of second focus.

Suppose a ray $\vec{p}$ is incident on the surface of the ellipse at point $\vec{r}$ having velocity vector $\vec{v}$. Then the normal(of reflection) lie at the same plane:

$$\hat{N} = k_1\hat{p} + k_2\hat{v}$$

Taking dot product with $\hat{v}$, we get

$$\hat{N}.\hat{v} = k_1(\hat{p}.\hat{v}) + k_2v^2 = 0$$

$$Let, \hat{v} = (v_x, v_y, v_z) \; and \; \hat{p} = (p_x, p_y, p_z)$$

Therefore,

$$\hat{N} = (k_1p_x + k_2v_x, k_1p_y + k_2v_y, k_1p_z + k_2v_z)$$

are the direction cosines.

$$(k_1p_x + k_2v_x)^2 + (k_1p_y + k_2v_y)^2 + (k_1p_z + k_2v_z)^2 = 1 \qquad (1)$$

$$\frac{k_1}{k_2} = \frac{-v^2}{p.v} \qquad (2)$$

From 1 and 2,

$$k_2^2 = \frac{1}{(v_x - \frac{v^2}{p.v}p_x)^2 + (v_y - \frac{v^2}{p.v}p_y)^2 + (v_z - \frac{v^2}{p.v}p_z)^2}$$

$$k_2 = \frac{p.v}{-v^2}k_1$$

Similar algorithm can be used to find that,For

$$\hat{L} = \alpha\hat{N} + \beta\hat{p}$$

$$\begin{aligned} &\alpha^2[(N_1\cos\theta - p_1)^2 + (N_2\cos\theta - p_2)^2 + (N_3\cos\theta - p_3)^2] \\ &\quad - 2\alpha[(N_1\cos\theta - p_1)p_1\cos\theta + (N_2\cos\theta - p_2)p_2\cos\theta \\ &\quad + (N_3\cos\theta - p_3)p_3\cos\theta] + p_1^2\cos^2\theta + p_2^2\cos^2\theta + p_3^2\cos^2\theta = 0 \end{aligned} \qquad (1)$$

$$\beta = 1 - \frac{\alpha}{\cos\theta}$$

Implimentation/Algorithm:

Thus we get the direction cosines of the reflected line. The line passes through

---

function FIND_REFLECTED_RAY($\hat{r}, \hat{v}, \hat{p}$);
$m \leftarrow dot(\hat{v}, \hat{v})/dot(\hat{p}, \hat{v})$;
$k_2 \leftarrow sqrt(1/((v[1]-m*p[1])**2+(v[2]-m*p[2])**2+(v[3]-m*p[3])**2))$;
$k_1 \leftarrow -v**2/dot(p, v)$;
$N \leftarrow k_1\hat{p} + k_2\hat{v}$;
$\theta \leftarrow \arccos(dot(\hat{N}, \hat{p}))$;
$a = (N[1]cos\theta - p[1])**2 + (N[2]cos\theta - p[2])**2 + (N[3]cos\theta - p[3])**2$;
$b = -2((N[1]cos\theta - p[1])p[1]cos\theta + (N[2]cos\theta - p[2])p[2]cos\theta +$
$(N[3]cos\theta - p[3])p[3]cos\theta)$;
$c \leftarrow p[1]**2*(cos\theta)**2 + p[2]**2*(cos\theta)**2 + p[3]**2*(cos\theta)**2$;
$\alpha = (-b + sqrt(b**2) - 4*a*c)/(2*a)$;
$\beta = (1 - \alpha/cos(\theta))$;
$L = \alpha N + \beta p$
EndFunction

---

point $\vec{r}$. So equation of the line becomes

$$\frac{x - r_x}{L_x} = \frac{y - r_y}{L_y} = \frac{z - r_z}{L_z} \qquad (1)$$

This line intersects $\vec{a}$ at focus F_2. If $\vec{A} = (A_x, A_y, A_z)$ be direction cosines of the line joining 2 foci, then

$$\frac{x}{A_x} = \frac{y}{A_y} = \frac{z}{A_z} \qquad (2)$$

Solving (1) and (2) we get Focus F_2

$$F_{2x} = \frac{2r_xL_x - r_xA_x}{A_x - L_x}$$

$$F_{2y} = \frac{2r_yL_y - r_yA_y}{A_y - L_y}$$

$$F_{2z} = \frac{2r_zL_z - r_zA_z}{A_z - L_z}$$

function FIND_FOCUS($\hat{r}, \hat{L}, \hat{A}$);
$F_2[1] = (2r[1] * L[1] - r[1]A[1])/(A[1] - L[1])$;
$F_2[3] = (2r[3] * L[3] - r[3]A[3])/(A[3] - L[3])$;
$F_2[2] = (2r[2] * L[2] - r[2]A[2])/(A[2] - L[2])$;
return(F_2)
EndFunction
function FIND_ECCENTRICITY($\hat{r}, \hat{L}, \hat{A}$);
$F_2 = FIND\_FOCUS(\hat{r}, \hat{L}, \hat{A})$;
ecc=$magnitude(F_2)/(2 * a)$;
return(ecc)
EndFunction

$$eccentricity, e = \frac{\sqrt{F_{2x}^2 + F_{2y}^2 + F_{2z}^2}}{2a}$$

Algorithm: There is another,rather simpler method for finding out the eccentricity. The mid-point of the periapsis and apoapsis calculated earlier gives the center of the ellipse.The distance of center from origin gives parameter ae. Therefore;

$$eccentricity, e = \frac{dist.of ellipse center from(0, 0, 0)}{magnitude of a}$$

Technology Stack:

1.Python,
2.Numpy,
3.Scipy,
4.Matplotlib,
5.MySql
6.MySql connector
7.some other spacescience related and other packages.

Complete CV

Github link:*https://github.com/ghosalsattam/*

Education:
1.10th-Bishop's School,Old HB Road,Ranchi-ICSE Board-91%
2.12th-Sarala Birla Public School,Ranchi-CBSE board–90% in PCM
3.B.Tech-Indian Institute of Information Technology,Kalyani(Year of pass-2021(Expected))

Project:
1.Implemented phone directory using C.
2.Implemented Dijkshtra's algorithm for finding shortest path using C.
3.Huffman Compression Technique.
4.Ford-Fulkersion Algorithm for determining maximum flow and minimum cut through network.
5.Made dynamic time-table management using Genetic Algorithm.

Work Experience:
1.Designed the salary Management System for the College.

Achievements::
1.NPTEL certificate on Data Structures and Algorithms using C(Awarded Elite ranking).
2.2nd ICSE topper at school level(10th)
3.2nd CBSE topper at high school level.

Courses enrolled:: 1.Introduction to programming with C.
2.Data Structures and Algorithms
3.Algorithms-1
4.Formal Languages and Automata Theory.
5.Computer Organization and Architecture.
6.Operating Systems
7.Object Oriented Programming
8.Linear Algebra
9.Probability and Statistics.
10.Differential Calculus and Transform Calculus
11.Numerical methods.

Skills::

Python,C,C++,Java,MySql,PySide,html,css,mysql.connector

My Interests::
1.Prefer reading space related books and articles.Have interst in space related technologies and a regular reader of articles published in www.space.com.
2.Coding
3.Searching for new approaches for solving physics and mathematics related problems.

Contacts:

1.contact no:8240657182
2.email:ghosalsattam@gmail.com
3.Skype:ghosalsattam
4.twitter:@GhosalSattam

Bibliography::

1.wikipedia
2.Smoothing Space Curves with MLS Projection by Lavanya Sita Tekumalla , Elaine Cohen School of Computing, University of Utah
3.Existing Code on Orbit Determination
4.https://sites.google.com/site/asen5050kemble/gibbs-method-orbit-determination