Determining the position, velocity and orbital elements of bodies orbiting Earth using Python

Noe Lepez Da Silva Duarte
Georgia Institute of Technology, Atlanta, Georgia, 30332, USA

This paper describes the method to determine position, velocity and orbital elements of bodies orbiting Earth using orbital mechanics and applies this to create a Python script able to determine the properties of any Earth orbit automatically.

Nomenclature

a	=	semi-major axis
D	=	number of mean solar days since epoch
e	=	eccentricity
$\vec{e}$	=	eccentricity vector
E	=	eccentric anomaly
$\vec{h}$	=	specific angular momentum
i	=	inclination
k	=	number of periapsis passages\
l	=	true longitude
L	=	latitude
M	=	mean anomaly
M_0	=	initial mean anomaly
$\vec{n}$	=	node vector
N	=	mean motion
p	=	semi-latus rectum
P	=	period of orbit
r	=	magnitude of the position vector in the planet-centered equatorial-equinox coordinate system
r_E	=	radius of the Earth
$\vec{r}$	=	position vector in the planet-centered equatorial-equinox coordinate system
$\overrightarrow{r_h}$	=	radius vector
$\overrightarrow{r_w}$	=	position vector in the perifocal-eccentricity coordinate system
TOF	=	time of flight
U	=	argument of latitude
V	=	magnitude of the velocity vector in the planet-centered equatorial-equinox coordinate system
$\vec{V}$	=	velocity vector in the planet-centered equatorial-equinox coordinate system
$\overrightarrow{V_w}$	=	velocity vector in the perifocal-eccentricity coordinate system
θ	=	local sidereal time
θ_{g0}	=	reference Greenwich mean sidereal time
λ_E	=	ground station location relative to the Greenwich meridian (west is considered negative)
μ	=	gravitational parameter
ν	=	true anomaly
Π	=	longitude of periapsis
$\vec{\rho}$	=	range
$\dot{\vec{\rho}}$	=	range rate
ω	=	argument of periapsis
$\vec{\omega}$	=	Coriolis acceleration
Ω	=	longitude of the ascending node

I. Introduction

For centuries humans have sought to explore the world around them. First, by conquering the seas, then soaring through the air, and today, mankind is trying to better understand the physics of space travel to explore new worlds. Orbital mechanics provides us with the tools to determine positions in and around orbits for any object in space. As such, this paper aims at creating a computer program able to determine the position, velocity vectors and orbital elements of an object orbiting around Earth at any time in its orbit, from a ground radar station. The program will then be tested by using the ISS's position on September 18th, 2020 at 8:15PM EST and making it determine its position 45 minutes later.

II. Methodology

All variables used were in canonical units, therefore, $\mu = 1$. All dates and times were also converted to GMT.

A. Position and velocity vectors

To determine the position and velocity vectors in the Earth-centered inertial, equatorial coordinate system, we must convert the range into a position a vector in the topocentric horizon coordinate. As such, the local sidereal time was first calculated using the Greenwich sidereal time equation,

$$\theta = \theta_{g0} + 1.00273779093 \times 360 \times D + \lambda_E \tag{1}$$

and reduced by

$$\theta = \theta - \mathrm{mod}(\theta, 360) \times 360 \tag{2}$$

where mod(x, y) represents the modulus of x divided by y. Then, the range was converted to the topocentric horizon frame relative to the center of the Earth as

$$\overrightarrow{r_h} = \vec{\rho} + r\hat{k} \tag{3}$$

$\overrightarrow{r_h}$ and $\dot{\vec{\rho}}$ were then transformed into the planet centered equatorial coordinate frame as

$$\begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \end{bmatrix} = [D]^{-1} \begin{bmatrix} \widehat{X_h} \\ \widehat{Y_h} \\ \widehat{Z_h} \end{bmatrix} \tag{4}$$

where

$$[D]^{-1} = \begin{bmatrix} sinLcos\theta & -sin\theta & cosLcos\theta \\ sinLsin\theta & cos\theta & cosLsin\theta \\ -\cos L & 0 & \sin L \end{bmatrix} \tag{5}$$

Finally, velocity was calculated by adding the impact of the Coriolis acceleration to the newly calculated $\dot{\vec{\rho}}$ as

$$\vec{V} = \dot{\vec{\rho}} + \vec{\omega} \times \vec{r} \tag{6}$$

B. Orbital elements

To calculate orbital elements, the three fundamental vectors must be calculated as

$$\vec{h} = \vec{r} \times \vec{V} \tag{7}$$

$$\vec{n} = \hat{Z} \times \vec{h} \tag{8}$$

$$\vec{e} = \frac{1}{\mu}\left[\left(V^2 - \frac{\mu}{r}\right)\vec{r} - (\vec{r} \cdot \vec{V})\vec{V}\right] \tag{9}$$

The semi-major axis was calculated as

$$a = \frac{\mathrm{mag}(\vec{h})^2}{1 - e^2} \tag{10}$$

where mag() represents the magnitude of the vector. Inclination was calculated as

$$i = \cos^{-1}\left(\frac{h_z}{\mathrm{mag}(\vec{h})}\right) \tag{11}$$

where h_z represents the z component of the angular momentum vector. The longitude of the ascending node was then calculated as

$$\Omega = \cos^{-1}\left(\frac{n_x}{\mathrm{mag}(\vec{n})}\right) \tag{12}$$

where n_x represents the x component of $\vec{n}$. If $n_y > 0$, $\Omega < 180$. The argument of periapsis and true anomaly were calculated as

$$\omega = \cos^{-1}\left(\frac{\vec{n} \cdot \vec{e}}{\mathrm{mag}(n) \times e}\right) \tag{13}$$

$$\nu = \cos^{-1}\left(\frac{\vec{e} \cdot \vec{r}}{r \times e}\right) \tag{14}$$

If $e_z > 0$, $\omega < 180°$ and, if $e_z < 0$, $\omega = 2\pi - \omega$. Similarly, if $\vec{r} \cdot \vec{V} > 0$, $\nu < 180°$. Finally, longitude of periapsis, argument of latitude and true longitude were determined as

$$\Pi = \Omega + \omega \tag{15}$$

$$U = \nu + \omega \tag{16}$$

$$l = \Pi + \nu \tag{17}$$

C. Kepler's problem

Determining the true anomaly requires an iterative solution known as Kepler's problem. Firstly, the number of periapsis passages must be determined by calculating the period of the orbit and finding mod(TOF, P). The period was calculated as

$$P = 2\pi\sqrt{\frac{a^3}{\mu}} \tag{18}$$

The mean anomaly was then calculated as

$$M = \mathrm{TOF}\sqrt{\frac{\mu}{a^3}} - 2\pi k + M_0 \tag{19}$$

where TOF is in seconds and M_0 is calculated using

$$M = E - e \sin E \tag{20a}$$

$$E = cos^{-1}\left(\frac{e + cos\nu}{1 + ecos\nu}\right) \tag{20b}$$

where ν is taken from eq. 14. E, and subsequently ν can be found using the Newton-Raphson method, an iterative solution shown below,

$$\mathrm{M_i = E_i - esinE_i} \tag{21}$$

$$E_{i+1} = E_i + \frac{m - m_i}{\frac{dM}{dE}|_{E=E_i}} \tag{22}$$

where

$$\frac{dM}{dE}|_{E=E_i} = 1 - ecosE_i \tag{23}$$

EE_i is 0.5 at the beginning of the iteration process. The iteration continues until m-m_i becomes very small.
Once the iteration process was completed and the value for E had converged, the new true anomaly was calculated as

$$\nu = cos^{-1}\left(\frac{cosE - e}{1 - ecosE}\right) \tag{24}$$

D. Position and velocity at new true anomaly

Before determining the position and velocity vectors in the planet-centered, equatorial-equinox coordinate system, they were calculated in the perifocal-eccentricity coordinate system as

$$\overrightarrow{\mathrm{r_w}} = \mathrm{rcos\nu}\widehat{\mathrm{X_w}} + \mathrm{rsin\nu}\widehat{\mathrm{Y_w}} \tag{25}$$

$$\overrightarrow{V_w} = \sqrt{\frac{\mu}{p}}\left(-sin\nu\widehat{X_w} + (e + cos\nu)\widehat{Y_w}\right) \tag{26}$$

Finally, these were converted to the to the planet-centered coordinate system as

$$\begin{bmatrix}\hat{X}\\ \hat{Y}\\ \hat{Z}\end{bmatrix} = [R]\begin{bmatrix}\widehat{X_w}\\ \widehat{Y_w}\\ \widehat{Z_w}\end{bmatrix} \tag{27}$$

where

$$[R] = \begin{bmatrix}R_{11} & R_{12} & R_{13}\\ R_{21} & R_{22} & R_{23}\\ \mathrm{R_{31}} & \mathrm{R_{32}} & \mathrm{R_{33}}\end{bmatrix} \tag{28}$$

$$R_{11} = \cos\Omega\cos\omega - \sin\Omega\sin\omega\cos i \tag{28a}$$

$$R_{12} = -\cos\Omega\sin\omega - \sin\Omega\sin\omega\cos i \tag{28b}$$

$$R_{13} = sin\Omega\sin i \tag{28c}$$

$$R_{21} = sin\Omega\cos\omega + \cos\Omega\sin\omega\cos i \tag{28d}$$

$$R_{22} = -\sin\Omega\sin\omega + \cos\Omega\cos\omega\cos i \tag{28e}$$

$$R_{23} = -cos\Omega \; \sin i \tag{28f}$$

$$R_{31} = \sin i \; \sin\omega \tag{28g}$$

$$R_{32} = \sin i \; \cos\omega \tag{28h}$$

$$R_{33} = \cos i \tag{28i}$$

E. Creating the Python program

Python was used to create the program due to its ease of use and the ability of adding multiple Python modules. Indeed, modules such as “datetime” to calculate the number of days since epoch, and “tkinter” to create a guide user interface (GUI) were used in the code. Information on how to run the code can be found in the Appendix.

III. Results and discussion

Using the following radar information for the ISS, collected on September 18, 2020 at 8:15PM EST or September 19, 2020 at 1:15 AM GMT,

$$\vec{\rho} = -0.118260\widehat{X_h} - 0.080977\widehat{Y_h} + 0.055150\widehat{Z_h}\mathrm{DU} \tag{29}$$

$$\dot{\vec{\rho}} = -0.513194\widehat{X_w} + 0.776045\widehat{Y_w} + 0.001608\widehat{Z_w}\,\frac{\mathrm{DU}}{\mathrm{TU}} \tag{30}$$

where the radar station has latitude 33.7718°N or 33.7718° and longitude 84.395°W or -84.395° and θ_{g0} taken on September 1 at 00:00 GMT is 340.62°, yields the following position and velocity vectors in the Earth-centered inertial, equatorial coordinate system,

$$\vec{r} = 0.23932\hat{X} - 0.77948\hat{Y} + 0.68485\hat{Z}\mathrm{DU} \tag{31}$$

$$\vec{V} = 0.65155\hat{X} + 0.57622\hat{Y} + 0.42749\hat{Z}\,\frac{\mathrm{DU}}{\mathrm{TU}} \tag{32}$$

The orbital elements for this orbit are shown in table 1.

Orbital element	**Value**
Eccentricity	0.000485 or $0.000342\hat{X}, 0.000115\hat{Y}, 0.000324\hat{Z}$
Semi-major axis	1.065055 DU
Inclination	51.263285°
Longitude of ascending node	244.709061°
Argument of periapsis	121.001276°
True anomaly	294.538147°
Longitude of periapsis	236.292214°
Argument of latitude	186.463128°
True longitude	301.754067°

Table 1. Orbital elements for the given orbit.

45 minutes or 0.75 hours later, the ISS had a true anomaly of 120.198° and its position in the Earth-centered inertial, equatorial coordinate system of

$$\vec{r} = -0.30877\hat{X} + 0.71328\hat{Y} - 0.72786DU \tag{33}$$

$$\vec{V} = -0.62683\hat{X} - 0.64251\hat{Y} - 0.36432\hat{Z}\,\frac{DU}{TU} \tag{34}$$

The iteration history is shown in Table 2.

Iteration (n)	**E_n**	**$M\text{-}M_n$**	**dM/dE**	**E_{n+1}**
0	0.5	3.6864035912	0.9995747820	4.1879717831
1	4.1879717831	-0.0022199092	1.0002426099	4.1857524123
2	4.1857524123	$1.0325056365\times10^{-9}$	1.0002435402	4.1857524133
3	4.1857524133	0.0	1.0002435402	4.1857524133

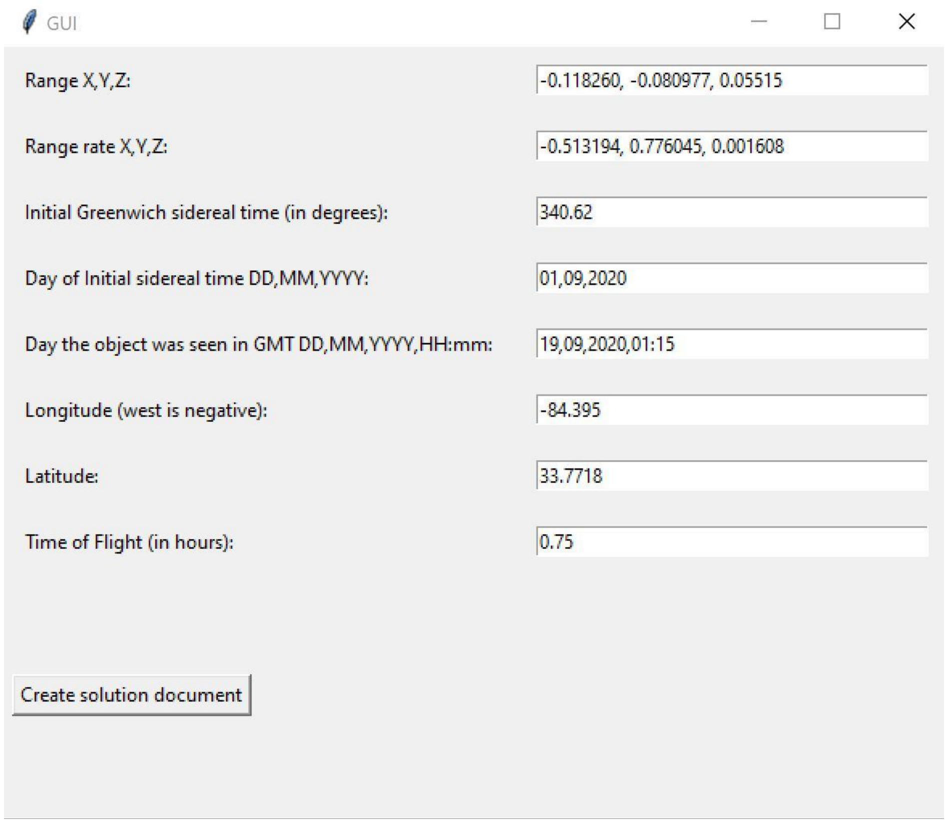
Table 2. Iteration history

IV. Conclusion

All in all, this paper was successful in creating a computer program able to calculate position, velocity and orbital elements of an object orbiting around Earth and showed the power of programming. Indeed, a simple Python program can calculate the properties of any orbit around Earth much faster than any human could. In the future, the code could become able to calculate orbits around any planet or star and a map showing where the object will be relative to the ground after the TOF input could be added.

Appendix

Two program files are available for use, both are the same. One has the .exe extension so that users without a Python interpreter can still easily use the program. To run the program, one simply has to double click the .exe file if on a Windows machine, or open the .py file with a Python Interpreter. A window similar to figure A.1. should open. Input the necessary information as shown in figure A.1. Make sure that the date and time the object was observed is in GMT, westward latitude is negative and TOF is in hours. DO NOT copy and paste any values into the GUI as this will lead to erroneous outputs.



Range X,Y,Z:	-0.118260, -0.080977, 0.05515
Range rate X,Y,Z:	-0.513194, 0.776045, 0.001608
Initial Greenwich sidereal time (in degrees):	340.62
Day of Initial sidereal time DD,MM,YYYY:	01,09,2020
Day the object was seen in GMT DD,MM,YYYY,HH:mm:	19,09,2020,01:15
Longitude (west is negative):	-84.395
Latitude:	33.7718
Time of Flight (in hours):	0.75

Figure A.1. How to use the GUI

Once all fields have been completed, press the “Create solution document” button, and a file named “Outputs_NoeDSDL.txt” should appear on your desktop. All solutions are on this document.