Planetary System MATLAB Simulation

December 26, 2018

Fahmy Ahmad
Diaa Eldin Malek
Nada Elmeligy

1 How to run the program

This program produces a simulation of the solar system using MATLAB. This simulation is accurate as far as introductory astronomy students are concerned. The program will show the orbits of the planets around the sun as they are known now, also shows the rotation of planets, but it doesn't show changes in orbits on an astronomical scale.

Caution:this program operates on MATLAB 2018:

1: Run PlanetarySystem(-) in the command prompt (replace - with 'ss' to run solar system simulation, or 'ps' simulate another system)
2: in case of 'ss', enter the speed level of the animation. (from 1 to 6). And choose the scaling of the orbits or orbits and sizes.
3: In case of 'ps', you will be asked to enter the name of your system, and prompted to enter data of the system in two matrices with each object in a row (1x10 for numerical data , 1x2 for string data).

Numerical data are as follows: (units between brackets)
Semi-major axis (AU), Eccentricity, Radius(Earth radius),
Orbital period(Earth year), direction of rotation (1 for clockwise,-1 for anti),
Rotational period(Earth day), Rotational direction(1 or -1),
Initial angular position in radians (choose your reference object),
The object that it rotates (1 for central star, the number of row of the orbited planet in case of moons), Orbital inclination to horizontal (radians)

String data are as follows: ["Name" , "Color.png"]
If the object is (or looks like) one of the solar system's objects, "Color.png" is replaced with "ObjectName.jpg".

Data example: [0.3871 0.2056 0.3825 0.24 1 58.646 1 6.0388 1 0.1222];
["Mercury" "Mercury.jpg"];

2 Derivation

For now, we will make 2 assumptions:
1- The solar system is a perfect plane. (z=0)
2- The sun lies at the center of the orbit, not the focus.
The equations describing the orbit are then:

$$x = a\cos\left(\frac{2\pi t}{P} + \phi\right) \qquad y = a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right) \qquad z = 0 \tag{1}$$

Where a is the semimajor axis, e is the orbit's eccentricity, P is the object's orbital period and α is the initial angular position in its orbit.

These equations plot a point at a given location of each object

To correct the second assumption, we shift the x-component of the origin of by a a factor of $a \times e$ hence re-locating the sun to the focus of the elliptical orbit instead. The equation then becomes:

$$x = ae + a\cos\left(\frac{2\pi t}{P} + \phi\right) \qquad y = a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right) \qquad z = 0 \tag{2}$$

To correct the first assumption, we have to rotate the elliptical orbit about the horizontal by an angle λ. By the laws of linear transformation, the equation becomes:

$$\begin{aligned} x &= ae + a\cos\left(\frac{2\pi t}{P} + \phi\right) \\ y &= a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right)\cos(\lambda) \\ z &= a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right)\sin(\lambda) \end{aligned} \tag{3}$$

We use spherical coordinates to construct spheres at those points and map each spheres with its respective picture, so the equation becomes.

$$\begin{aligned} x &= ae + a\cos\left(\frac{2\pi t}{P} + \phi\right) + R\cos(\theta)\sin(\alpha) \\ y &= a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right)\cos(\lambda) + R\sin(\theta)\sin(\alpha) \\ z &= a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right)\sin(\lambda) + R\cos(\alpha) \end{aligned} \tag{4}$$

Wheres R is the object's radius, $\quad 0 < \theta < 2\pi \; 0 < \alpha < \pi$.

But the object rotates about it axis, so the spherical coordinates vary with time, and the equation becomes:

$$x = ae + a\cos\left(\frac{2\pi t}{P} + \phi\right) + R\cos(\theta)\sin(\alpha + \frac{2\pi t}{p})$$
$$y = a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right)\cos(\lambda) + R\sin(\theta)\sin(\alpha + \frac{2\pi t}{p})$$
$$z = a\sqrt{1-e^2}\sin\left(\frac{2\pi t}{P} + \phi\right)\sin(\lambda) + R\cos(\alpha) \quad (5)$$

Where p is the axial rotation(in days)

In case of the moon:
The exact same procedure is followed, but the x,y,z coordinates of the moon are added to the objects' coordinates at that same point.

3 Program algorithm:

The program consists of 1 main functions and 5 secondary. The main function PlanetarySystem, This function takes 'ss' or 'ps' as input. If 'ss' is the input, the "SolarSystem" function is called which reads the data of the solar system which is already prepared in a function called "SOLARDATA", and stores it into two matricies one for numeric data (nx10) and one for string data (nx2). These the first matrix contains all the numeric data while the second contains all the string data of the planets. Prompts the user to launch a fully scaled animation or orbits scalled only animation. The data is then the input of the function "plotps" which starts the simulation for the solar system.

Secondly, if the input was 'ps', the function calls another function called "dataentry" which prompts the user to enter the data of the system. And then stores the data in 2 arrays one for the numeric data (nx10) and one for the string data (nx2).

After data is received and stored, this data is then the input of a function called "data" which loops on the data to check if the data is valid. Firstly, the data of the nx2 matrix will be checked to make sure all entries are string entries, if an entry is not a string the user will be prompted to enter this specific entry again providing the right format. Secondly, the nx10 matrix will be checked for numerical errors (values are positive, 0¡e¡1, etc) in the data given, and prompts the user to enter a valid entry providing the range of such a value.

The function “plotps”, takes the validated data matrices as input. At first, the user is prompted to enter the speed factor of the animation. Then this function constructs arrays containing the data points needed to plot the orbits and the moving objects and are stored into an array. secondly , the data points to construct a sphere in matlab are stored into 2 arrays. Thirdly, This data is then used in parallel with the numerical data to construct handles for each object using warp built in function. which will be stored in one handle to call them and change the values inside to produce moving animation. Fourthly, a while loop is constructed to loop on the handles and change the values of the xdata, ydata, and zdata. This change accounts for every single change that should happen, from rotation of moons and planets around themselves to their orbit, and the counter is incremented with each loop and reseted when it reaches the maximum index of the data matrices.