REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK
35. évfolyam (2024) 2. szám 61–72. • DOI: 10.32560/rk.2024.2.4

Tamás Szakács

Orbital Mechanics Two-Body Model for Educational Purposes

Teaching celestial mechanics is a highly rewarding task, as there is hardly a student who is not intrigued by how celestial bodies move across the sky, how an Earth-orbiting satellite stays in orbit, or how one can travel to the Moon, Mars, or further in the solar system and beyond. Even with only high school-level mathematics and physics knowledge, one can understand and describe key operations such as orbit insertion, orbital manoeuvres, and the concepts of escape velocities. To aid the teaching of the subject "Orbital Mechanics", I developed a model in the Matlab/Simulink environment, which is suitable for demonstrating these tasks, providing illustrations, and verifying homework. I emphasised keeping the model simple and easy to use while making it interesting for students. Additionally, it serves as a tool for practicing the use of Matlab/Simulink.

Keywords: *celestial mechanics, orbital mechanics, orbital manoeuvres*

1. Introduction

In his posthumous work, *De mundi systemate* (published in 1728 as *A Treatise of the System of the World* [1]), Isaac Newton included an illustration explaining the concept of the first cosmic velocity (Figure 1). While Newton could not calculate the exact velocity due to the lack of knowledge about Earth's precise mass and the gravitational constant, he introduced the theory of orbital motion. The second cosmic velocity was published by Konstantin Eduardovich Tsiolkovsky in 1903 [2]. Both works are accessible online.

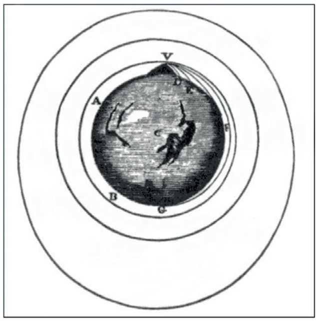


Figure 1.
Newton's explanation of 1st cosmic velocity [1]

The third name worth mentioning is Walter Hohmann, who in his 1925 publication introduced an energy-efficient orbital transfer between two orbits around a celestial body [3]. These concepts are part of the curriculum in orbital mechanics, and the author considered it essential to provide students with an interactive simulation tool, in addition to formulas and diagrams, to enhance their understanding of the material.

2. Modelling

The author developed the model in Matlab/Simulink environment. The Simulink block diagram model is extremely simple (Figure 2). It contains only a rocket propulsion block, a mechanical block named Space object, and a masked block, which at double click loads the necessary data, and launches GUI and animation windows. The core of the mechanical block (Figure 3) is a Matlab S-function, where the equations for orbiting a central celestial body are written in vector form in m-code.

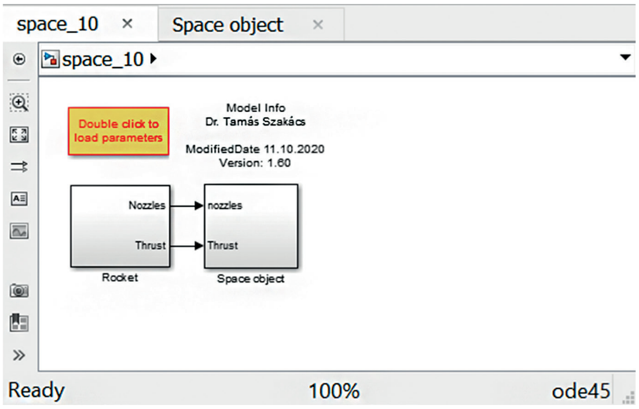


Figure 2.
Top layer of the orbital dynamics model [the author]

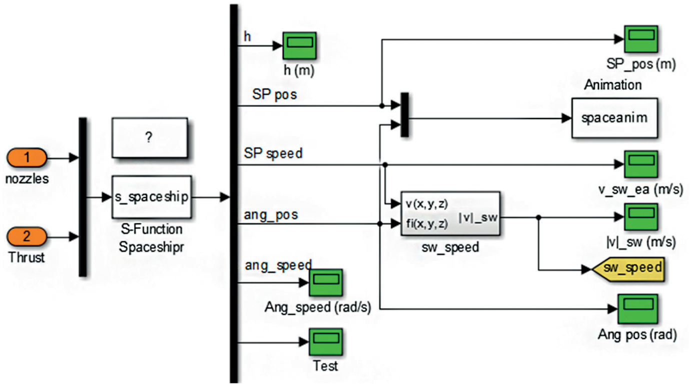
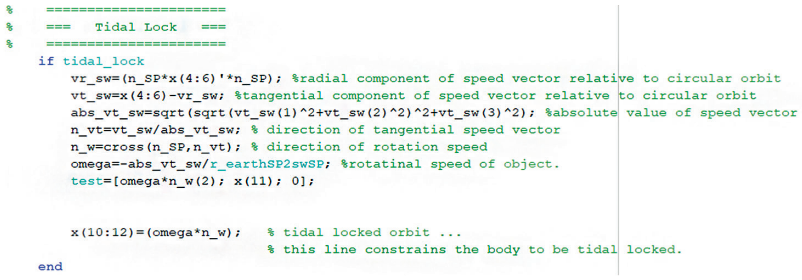


Figure 3.
Detailed view of the object block [the author]

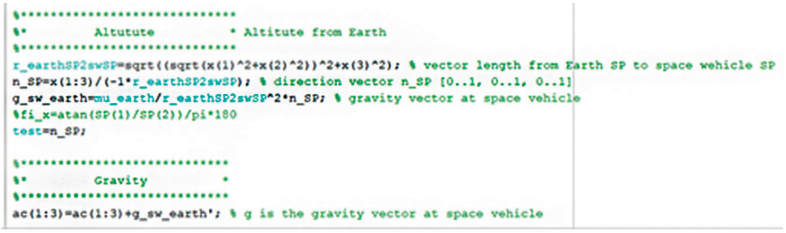
The mechanical part that calculates the mechanical motions of the body consists of a few code lines only. The code also includes an option to toggle between tidal locked and unlocked orbit (Figure 4). The total number of code lines are 175, about half of which are comments, while the rest are necessary lines for the S-function, such as setting initial values of state variables, calculating derivatives, and computing outputs. Thanks to Matlab's matrix-oriented structure, the equations for the derivatives take only three lines (Figure 5). The description for tidal-locked orbits, in contrast, takes six lines.



```
%    ======================
%    ===   Tidal Lock   ===
%    ======================
    if tidal_lock
        vr_sw=(n_SP*x(4:6)'*n_SP); %radial component of speed vector relative to circular orbit
        vt_sw=x(4:6)-vr_sw; %tangential component of speed vector relative to circular orbit
        abs_vt_sw=sqrt(sqrt(vt_sw(1)^2+vt_sw(2)^2)^2+vt_sw(3)^2); %absolute value of speed vector
        n_vt=vt_sw/abs_vt_sw; % direction of tangential speed vector
        n_w=cross(n_SP,n_vt); % direction of rotation speed
        omega=-abs_vt_sw/r_earthSP2swSP; %rotatinal speed of object.
        test=[omega*n_w(2); x(11); 0];


        x(10:12)=(omega*n_w);    % tidal locked orbit ...
                                 % this line constrains the body to be tidal locked.
    end
```

Figure 4.
Code lines for tidal-locked orbit [the author]

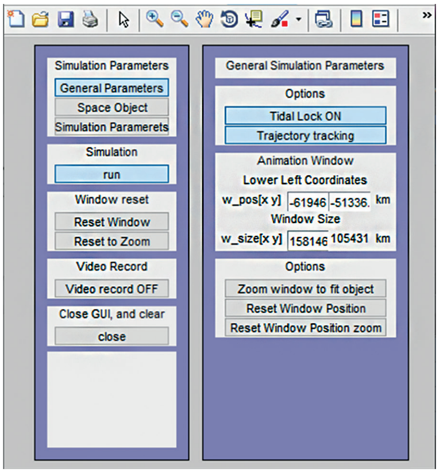


```
%******************************
%*          Altutute          * Altitute from Earth
%******************************
r_earthSP2swSP=sqrt((sqrt(x(1)^2+x(2)^2))^2+x(3)^2); % vector length from Earth SP to space wehicle SP
n_SP=x(1:3)/(-1*r_earthSP2swSP); % direction vector n_SP [0..1, 0..1, 0..1]
g_sw_earth=mu_earth/r_earthSP2swSP^2*n_SP; % gravity vector at space vehicle
%fi_x=atan(SP(1)/SP(2))/pi*180
test=n_SP;

%*****************************
%*          Gravity          *
%*****************************
ac(1:3)=ac(1:3)+g_sw_earth'; % g is the gravity vector at space vehicle
```

Figure 5.
Determining the magnitude and direction of gravitational force and acceleration acting on the object [the author]

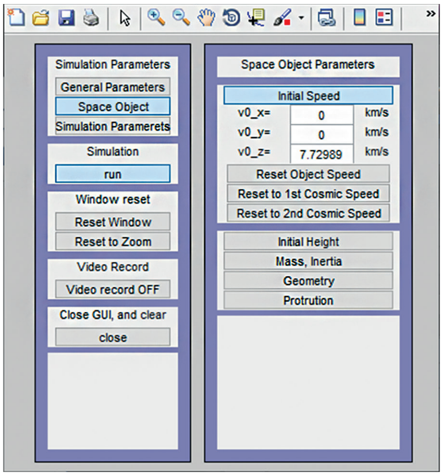
To ensure ease of use, the author developed a graphic user interface (GUI). The model is fully parametric, and parameter handling is managed through the graphical user interface (Figures 6–7). Interestingly, its code consists of 1,443 lines.



w_pos[x y] -61946 -51336. km

w_size[x y] 158146 105431 km

Figure 6.
Graphical User Interface for managing parameters: General settings and options [the author]



v0_x= 0 km/s

v0_y= 0 km/s

v0_z= 7.72989 km/s

Figure 7.
Graphical User Interface for managing object parameters [the author]

Since the essence of modelling is visualisation, an animation window was also created (Figure 8), where the relative motion of the two bodies can be tracked. This was also coded into the S-function of the animation.

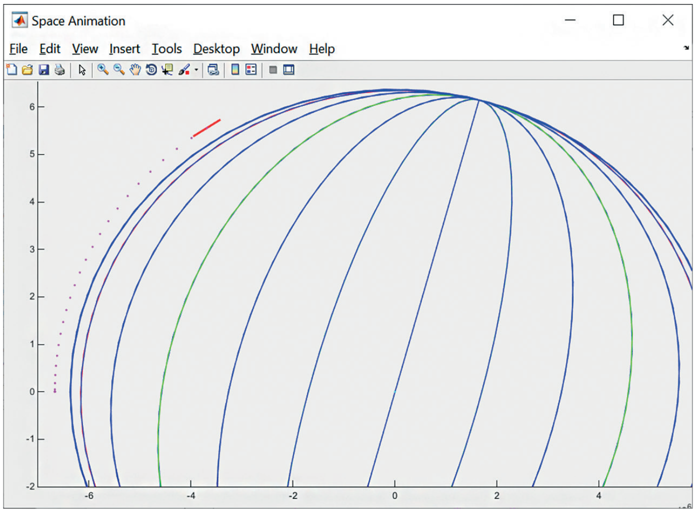
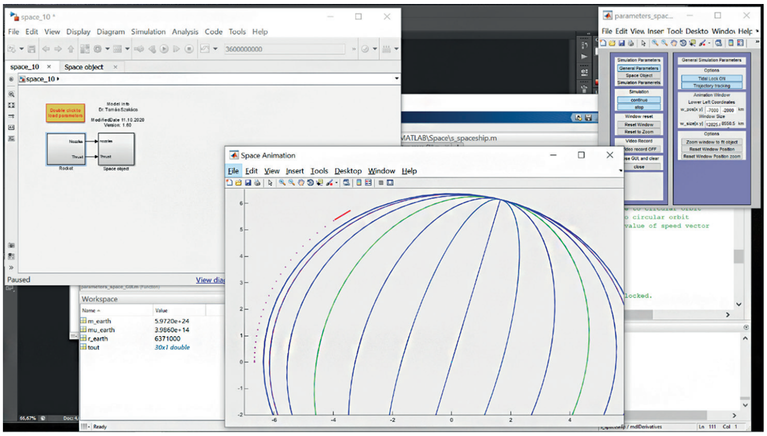


Figure 8.
Animation window [the author]

3. Model usage

Upon loading the model, the previously open windows are restored in the same state. As the model is parametric and the parameters can be managed via the graphical interface, the Matlab window can be minimised. By clicking on the parameter loading block (Double click to load parameters), the necessary simulation parameters are loaded, and the GUI and animation window open. After arranging the windows conveniently (Figure 9), the model can be run from the GUI (this is the recommended approach).


space_10
File Edit View Display Diagram Simulation Analysis Code Tools Help
Double click to load parameters
Rocket
Space object
Paused
Workspace
parameters_spac...
Simulation Parameters
General Simulation Parameters
Space Animation
File Edit View Insert Tools Desktop Window Help

Name	Value
m_earth	5.9720e+24
mu_earth	3.9860e+14
r_earth	6371000
tout	30x1 double

Figure 9.
Layout of the model windows [the author]

Before running, it is advisable to start with the simulation parameters in the general settings, found in the upper left section of the GUI. It includes three radio buttons: general parameters, object parameters, and simulation-related parameters. Pressing any of these will regenerate the right-hand window and display the corresponding parameters.

In the general parameters (Figure 6), the following options can be toggled:

- bounded orbit (Tidal Lock);
- object trajectory tracking in the animation window (Trajectory Tracking);
- screen zoom for simultaneous display of both objects (Zoom Window to Fit Object);
- reset window position before starting the simulation (Reset Window Position);
- reset zoom level before starting the simulation (Reset Window Position Zoom).

These last two functions are always accessible from the left menu for convenience. Additionally, the initial position and view range of the animation window can be set here.

Figure 6 shows the object parameters that can be modified:

- initial velocity in km/s;
- initial altitude from the center of the Earth in km;
- mass and moment of inertia;
- geometrical properties of the object;
- (the propulsion section is currently inactive).

Each tab contains further options, such as calculating, and setting the 1st and 2nd cosmic velocities for a given initial altitude, which the interface also displays. Doing this will result in a circular orbit at that predetermined height.

Additional features: The previously mentioned Reset Window and Reset to Zoom functions are available, as well as a button for recording the model's animation for video creation.

After setting the parameters, the model can be run. The phases of execution are shown in Figure 10.

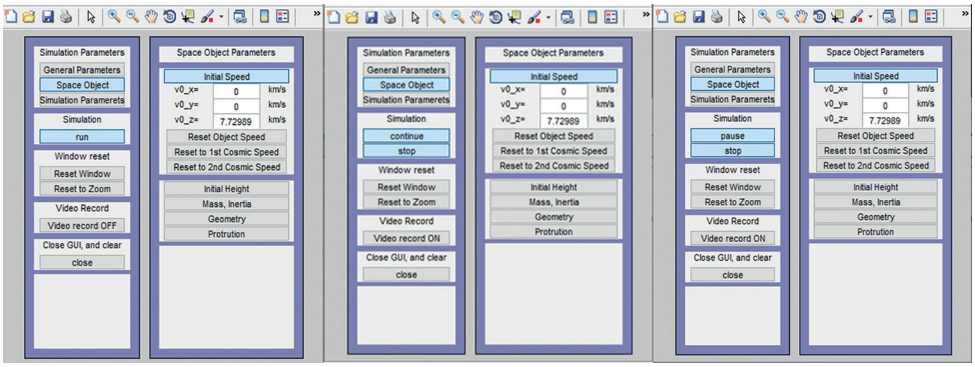


Figure 10.
Run Modes: Run, Pause/Stop, Continue/Stop [the author]

When clicking the run button, the label switches to pause and stop buttons, allowing the simulation to be paused or stopped. During suspension, the pause button changes to continue, resuming the simulation.

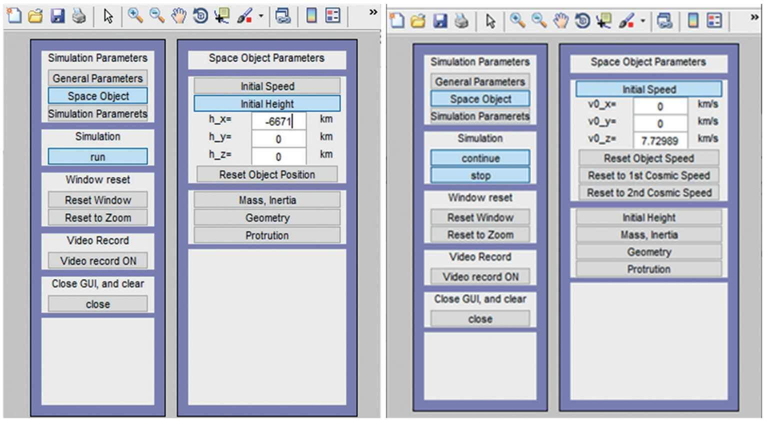
4. Example of running the model

As an example, an orbital transfer from 300 km circular orbit about Earth surface to a geostationary orbit using a Hohmann manoeuvre will be presented. First, we set a H = 300 km orbit by selecting the Space Object parameter and setting the Initial Height. The average radius of the Earth is R = 6,371 km, so the orbit is at R + H = 6,671 km measured from the centre of Earth. Then, we set the corresponding 1st cosmic speed for the altitude using the "Reset to 1st Cosmic Speed" button. The result is an initial velocity (v_{0z}) of 7.2989 km/s, as Figure 11 shows.

The first cosmic speed at 300 km height:

$$v_I = \sqrt{\gamma\frac{M}{r}} = \sqrt{\frac{K}{r}} = \sqrt{\frac{K}{(R+H)}} = \sqrt{\frac{398620}{6371+300}} = 7.2989\frac{km}{s}$$

Where γ is the universal gravitations constant: $\gamma = 6{,}67 \cdot 10^{-11}\,\frac{m^3}{s^2}$, and M is the mass of Earth, $\mathrm{M} = 5{,}976 \cdot 10^{24}\,kg,\ K = \gamma M = 6{,}67 \cdot 10^{-11} \cdot 5{,}976 \cdot 10^{24} = 398620\ \frac{kgkm^3}{s^2}$



h_x= -6671 km
h_y= 0 km
h_z= 0 km

v0_x= 0 km/s
v0_y= 0 km/s
v0_z= 7.72989 km/s

Figure 11.
Setting the parameters [the author]

The rocket propulsion setup follows. The rocket engines are programmed in the Rocket block of the Simulink model (Figure 12). Currently, the model includes two Hohmann manoeuvres: the first applies an initial impulse after completing one orbit, followed by a second impulse at the opposite side of the Earth when the new altitude is reached. The second variation completes the first impulse, returns to the original starting point on the Hohmann transfer ellipse, and only then performs the second impulse when the rocket is at the opposite point at the second time. This variation is for demonstration purposes only.

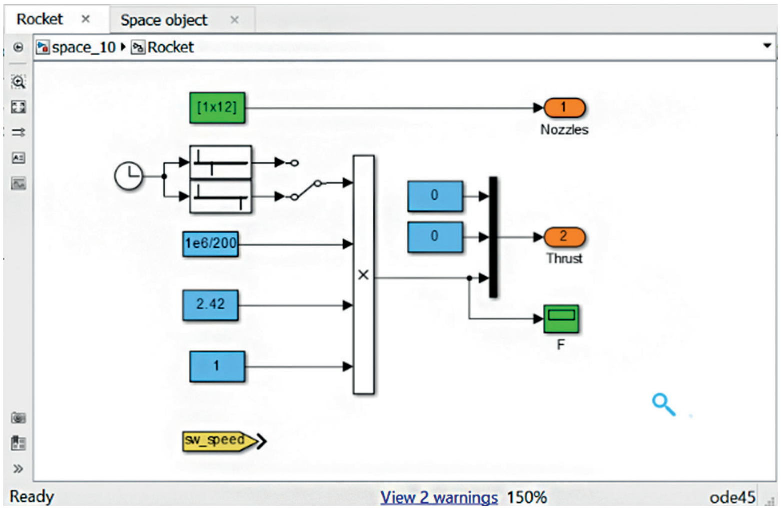
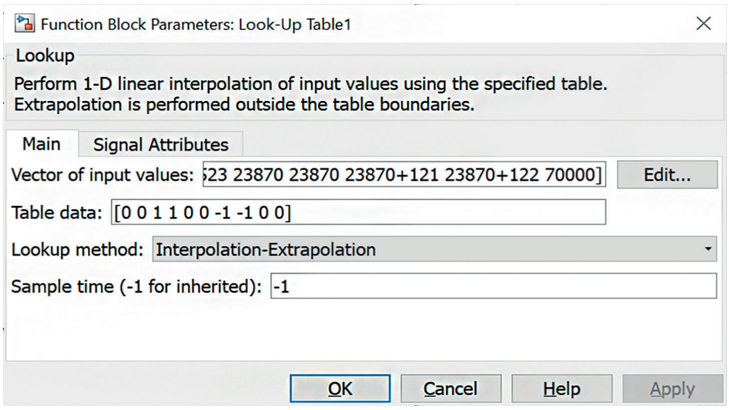


Figure 12.
The rocket and nozzle model [the author]


Function Block Parameters: Look-Up Table1

Lookup

Perform 1-D linear interpolation of input values using the specified table.
Extrapolation is performed outside the table boundaries.

Main | Signal Attributes

Vector of input values: ;23 23870 23870 23870+121 23870+122 70000]

Table data: [0 0 1 1 0 0 -1 -1 0 0]

Lookup method: Interpolation-Extrapolation

Sample time (-1 for inherited): -1

Figure 13.
Programming the rocket engine [the author]

After programming the rocket, clicking on Reset Window, Reset to Zoom, and Run starts the simulation.

The results are shown in Figures 14–18.

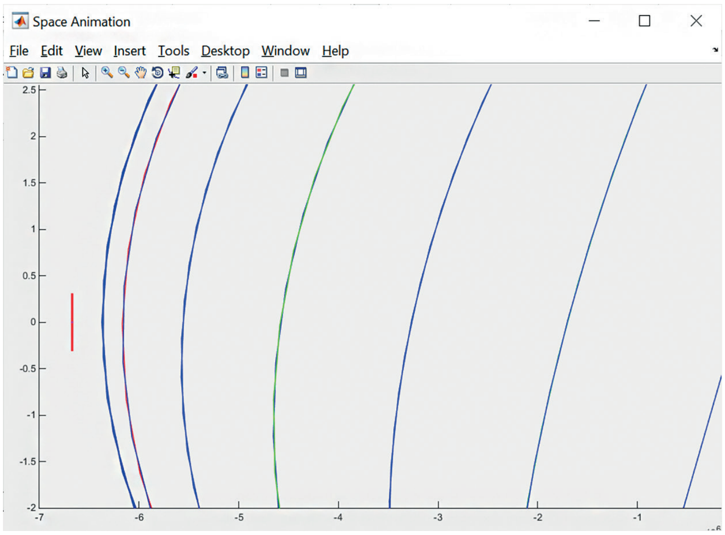


Figure 14.
Beginning of the simulation: the object in its initial position [the author]

Figure 14 shows the initial state at the beginning of the simulation. Initial position and speed are set in the parameters. The figure is to scale (except the space object), so one can see how far 300 km from Earth surface is.

As the space object moves, it is flying out of the window. Because autozoom is on, when it occurs, the window resolution is automatically resized to have centre of Earth, and the space object in the same window. Trajectory tracking is also on, that plots the purple dots behind the space object.

Completing one round, the space object returns to its original position. The rocket propulsion is programmed so that the first impulse occurs at this point. Burning time and thrust force are programmed in the rocket module (Figure 16).

In Figure 17 the object reached the new orbit height (apogee), where the second burn should occur to reach the first cosmic speed according to that distance. In this demonstration the burn does not occur, so the object completes the ellipse, and returns to the starting point (perigee). So, the orbit is now changed from circular to elliptical. Reaching the apogee again the second burn is initiated in the model, which accelerates the object to reach first cosmic velocity at that distance, so the object will orbit in a circular path again (Figure 18).

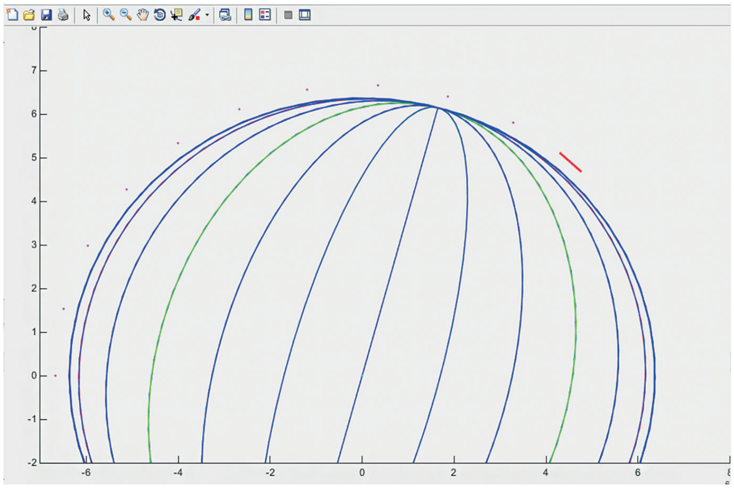


Figure 15.
As the object leaves the window's displayed area, the autozoom adjusts the window [the author]

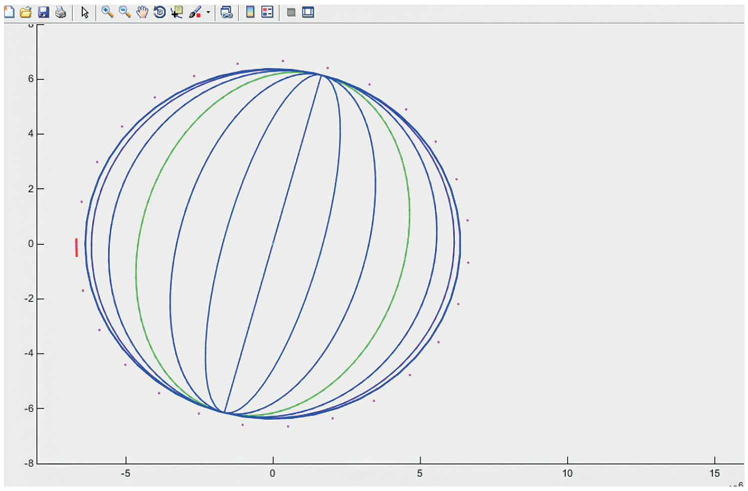


Figure 16.
The first impulse starts at the end of the first complete orbit [the author]

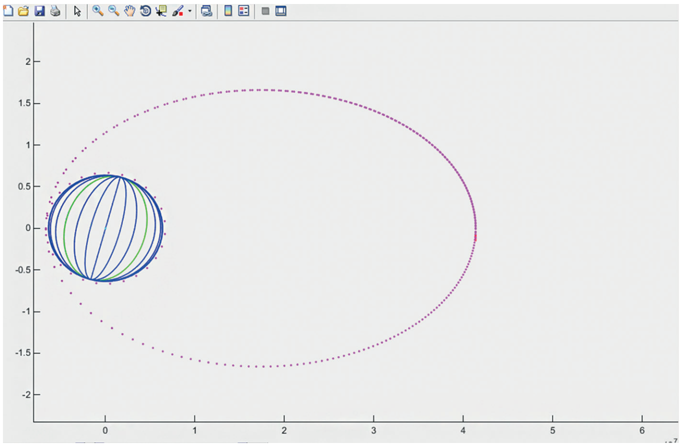


Figure 17.
The space object completes the Hohmann ellipse (demonstrational purpose). Second impulse [the author]

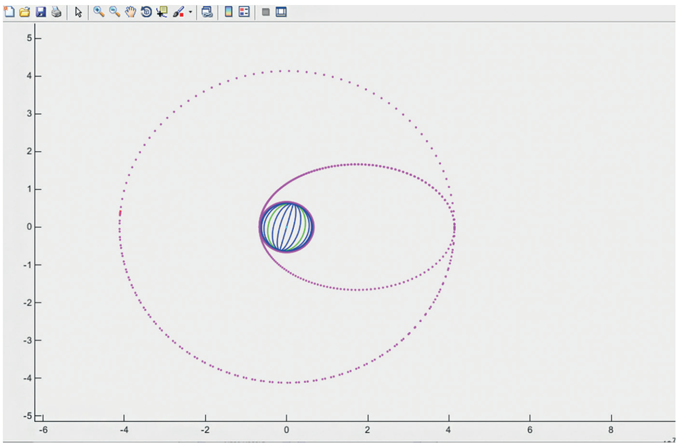


Figure 18.
The new geostationary orbit [the author]

So far, 61,400 simulation seconds have passed, which took only 60 seconds on a standard office laptop.

5. Further development suggestions

Currently, the model can simulate two-body systems. It is set to model Earth and a space object, but it can be adapted for any two-body system (e.g. Sun–Earth, Earth–Moon). Extending it to a three-body model would allow the simulation of slingshot manoeuvres or any three-body system (e.g. Sun–Earth–Moon or Earth–Moon–artificial satellite). After converting it into a three-body model, it can be further developed into an N-body model to model complex orbiting problems, or travelling with an object from planet to planet in the solar system.

6. Summary

This article presents the construction and usage of an orbital dynamics model. The model is intended for educational purposes in physics and orbital mechanics, allowing the demonstration of the first and second cosmic velocities, different orbits, and orbital manoeuvres. Developed in the Matlab/Simulink environment, it is also useful for teaching and practicing Matlab usage. The model is extremely simple in design and easy to use even without prior Matlab knowledge.

References

[1] I. Newton, *De mundi systemate. A Treatise of the System of the World.* 1728.
[2] К. Ціолковскій, *Изслѣдованіе мировыхъ пространствъ реактивными приборами.* 1903.
[3] W. Hohmann, *Die Erreichbarkeit der Himmelskörper.* München, R. Oldenbourg, 1925.

Űrdinamikai kéttest-modell oktatási célra

Az égi mechanika oktatása rendkívül hálás feladat, mivel nincs olyan diák, akit ne érdekelne az, hogy az égitestek hogyan mozognak az égbolton, vagy egy Föld körüli pályára állított műhold miért marad pályán, vagy hogyan lehet eljutni a Holdra, Marsra vagy nagyobb távolságokra a Naprendszerben, vagy azon túl. Már középiskolai szintű matematikai és fizikai ismeretekkel is meg lehet érteni, illetve le lehet írni a legfontosabb műveleteket, mint például a pályára állás vagy a pályamódosítások, az első és a többi kozmikus sebességek. Az űrdinamika című tárgy oktatásának segítésére létrehoztam egy olyan modellt Matlab/Simulink környezetben, amely alkalmas a fent említett feladatok bemutatására, szemléltetésére, házi feladatok leellenőrzésére. Fontosnak tartottam, hogy a modell egyszerű maradjon, könnyen kezelhető, mégis érdekes legyen a hallgatók számára. Nem utolsósorban a Matlab/Simulink programkörnyezet használatának gyakorlását is segíti.

Kulcsszavak: *égi mechanika, űrdinamika, pályamódosítások*

Szakács Tamás, PhD egyetemi docens Óbudai Egyetem Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar Jármű és Mechatronikai Intézet szakacs.tamas@bgk.uni-obuda.hu orcid.org/0000-0002-7636-7488	Tamás Szakács, PhD Associate Professor Óbuda University Donát Bánki Faculty of Mechanical and Safety Engineering Institute of Mechatronics and Vehicle Engineering szakacs.tamas@bgk.uni-obuda.hu orcid.org/0000-0002-7636-7488