Force Models and Numerical Methods for a Versatile Astrodynamics Simulator, OrbitSim3D

H. A. Güler

September 2024

Abstract

OrbitSim3D is a numerical physics simulator that can simulate a large variety of astrodynamics-related phenomena ranging from simple n-body gravitational problems to the effects of solar radiation, atmospheric drag and such, all in a quite simple manner; with possibility to account for many other forces, accelerations and perturbations easily. This report provides a short discussion of the mathematical models and the design philosophy.

1 Introduction

Kepler's Laws of Planetary Motion provided astronomers with the first model for describing the motions of heavenly bodies in three-dimensional space (as opposed to solutions for angular astrometric positions) for extended periods of time. This model describes the trajectory of the lighter body in a two-body orbital system, where the massive body sits in one of the ellipse's foci. The ellipse is then located in three dimensional space by the "orbital elements", which describe the length of the semi-major axis, the eccentricity of the ellipse, the inclination of the ellipse relative to a reference plane (usually the ecliptic plane for Solar System objects), the longitudes at which the periapsis and the ascending node are located, and the mean anomaly at a given epoch, locating the body on a particular point on the ellipse that was described by previously mentioned orbital elements. The model is elegant, and is quite accurate as long as;

- Modeling the two bodies as point-masses is an accurate approximation of the structure of the problem
- Gravitational acceleration is the only or the most dominant acceleration acting on the bodies at all times
- No extra massive bodies significantly affect the gravitational field in the solution domain

- Relativistic effects are negligible

The more closely these conditions are met, the better accuracy the two-body solution will obtain when compared to the real-life behaviour of celestial bodies. However, such a case can never be exactly replicated - the Solar System has multiple planets, some with many moons, and these bodies create gravitational perturbations on one another. In addition, some bodies such as the Moon can exhibit "lumpy" gravitational field characteristics, and their gravitational fields in close proximity to their surfaces may not be suitably replicated by a single point-mass. Furthermore, as observed with the perihelion precession of Mercury, some relativistic effects may have to be accounted for.

When all the effects mentioned above become significant contributors to the overall force models of objects in space, an analytical solution becomes impractical (or almost always outright impossible). For this reason, numerical schemes are used for utilizing higher accuracy force models.

OrbitSim3D attempts to provide the tools for creating complicated force models for spacecraft and celestial bodies and use numerical orbit propagation methods to simulate such complicated systems. In the following sections, how these force models are obtained will be described in detail.

2 Force and Acceleration Models

2.1 Single Point-Mass Gravity

"Point-mass gravity" is the most basic and fundamental force model for an n-body gravitational simulation. This model arises directly from Newton's Law of Gravitation and the point-mass approximation of a massive body.

Newton's Law of Gravitation is described as follows:

$$\boldsymbol{a} = G \cdot \frac{M}{r^2}\hat{\boldsymbol{r}} \tag{1}$$

...where $\boldsymbol{a}$ is the acceleration on a body, G is the universal gravitational constant, with an approximate value of $6.67 \cdot 10^{-11} m^3 kg^{-1} s^{-2}$, M is the mass of the body that is causing the acceleration, r is the distance between the affected body and the massive body that generates the gravitational field, and $\hat{\boldsymbol{r}}$ is the direction from the affected object towards the massive body.

To obtain the highest accuracy in evaluating the acceleration, the mass distribution within the volume that makes up the massive body should be considered, resulting in the following integral over the massive body's volume:

$$\boldsymbol{a} = G \cdot \int_V \frac{d\rho}{r^2}\hat{\boldsymbol{r}} dV \tag{2}$$

...where ρ is the density of the body at a given point within its volume, and r is the distance to the infinitesimal volume dV that is being considered, and $\hat{\boldsymbol{r}}$ is the direction (normalized relative position) of the infinitesimal volume relative to the affected body.

However, when the distance between the massive body and the affected body is much larger than the size of the massive body, the mass distribution (or the "lumpiness") of the massive body becomes insignificant, leading to the simplifying approximation which allows for its entire mass to be modeled as a point with finite mass, hence creating the "point-mass" approximation and allowing us to use Equation 1 instead.

2.2 Point-Mass-Cloud Gravity and Limited Spherical Harmonics Model

Equation 2 isn't without any use itself. When a "lumpy" body is considered, one whose density varies significantly within its volume, and if an affected body is close to the said lumpy body, the integral in Equation 2 can be approximated by a distribution of finite number of point masses. Equation 2 then turns into its discrete form:

$$\boldsymbol{a} = G \cdot \sum_{i=1}^{N} \frac{m_{pm}}{r_{pm}^2} \hat{\boldsymbol{r}}_{\boldsymbol{pm}} \tag{3}$$

...where N is the number of point masses that together describe a single body, with m_{pm} and r_{pm} representing the masses of and the distances to each individual point mass. As more point masses are used to describe a body, the spatial resolution of the gravitational field improves, and it becomes possible to theoretically model an infinitely complicated gravitational field - at the expense of computational cost.

The "spherical harmonics" approach is a more common and rather more standardized method for modeling complex gravitational fields created by oblate or lumpy bodies, which uses spherical harmonics coefficients in a matrix that describe the contributions of each spherical mode. OrbitSim3D is yet to implement that approach as of August 2024, as the point-mass cloud method was considered to be rather more straightforward and intuitive, especially for people who wish to model hypothetical bodies whose spherical harmonics coefficients can not be measured in real-life. The point-mass-cloud is also easier to use for modeling bodies of extremely non-spherical shapes such as contact-binary asteroids.

Regardless, a restricted spherical harmonics model called the "J2 model" (named after the J2 spherical harmonics coefficient) that only takes into account the first mode beyond the spherical field of the central point-mass, can be used in OrbitSim3D, that takes into account the equatorial oblateness of a celestial body. The acceleration due to the J2 coefficient can be modeled as follows. [1]

$$\boldsymbol{a}_{J2} = \frac{3J_2GMR_b^2}{2R^5} \left[\left(5\frac{Z^2}{R^2} - 1 \right) (X\hat{\boldsymbol{i}} + Y\hat{\boldsymbol{j}}) + Z \left(5\frac{Z^2}{R^2} - 3 \right) \hat{\boldsymbol{k}} \right] \tag{4}$$

...where R_b is the radius of the body, R is the distance of the affected object from the center of the body, X, Y and Z being the components of the affected

object's position in the body-centered inertial coordinate system, and $\hat{\boldsymbol{i}}$, $\hat{\boldsymbol{j}}$ and $\hat{\boldsymbol{k}}$ being the respective axes.

2.3 Atmospheric Drag

Atmospheric drag is known to scale with the square of the velocity of the airflow over a body, and the cross-sectional area that faces the flow, as well as the local density of the flow medium. But, in addition to these values, due to the exact way the shape of a body interacts with the airflow, or certain compressibility effects that occur at high velocities, a "coefficient of drag" is also required to scale the drag force to a correct value. The expression to evaluate for atmospheric drag force then becomes:

$$F_{drag} = 0.5 \cdot C_D \cdot \rho \cdot A \cdot v^2 \tag{5}$$

...where F_{drag} is the drag force acting on an object, C_D is the coefficient of drag, ρ is the density of the fluid medium, A is the cross-sectional area of the object that faces the airflow, and v is the velocity of the local fluid medium relative to the affected object. ρ is usually a function of altitude, such that $\rho = \rho(h)$ where h is the altitude. More complicated models can also consider temperature, so that $\rho = \rho(h, T)$ where T is the local temperature of the atmosphere. A and C_D may be a function of the object's orientation relative to the airflow.

The built-in atmospheric drag calculation method in OrbitSim3D only considers the variation in ρ with respect to altitude, where $\rho = \rho(h)$. One of the simpler ways to describe the density variation with altitude is to use an exponential fall-off function that uses a base density value and an atmospheric "scale height", such that:

$$\rho(r) = \rho_0 \cdot e^{\frac{R-r}{H}} \tag{6}$$

...where R is the sea-level radius of the body (or any meaningful reference radius), r is the distance of the measurement point from the center of the body, and H is the scale height that describes how rapidly the atmospheric density falls off.

While the above approach provides a simple solution, for higher accuracy solutions, the ideal way would be to use a curve fit or a lookup table that relies on body-specific parameters and measurements.

The local atmospheric flow velocity can be taken to match the rotational speed of the celestial body itself, or may be taken to be stationary in the body's inertial frame if the altitude is too great and space weather modeling is not available (or the inaccuracies are acceptable). Again, the ideal way to calculate such a velocity with high accuracy would be to rely on body-specific parameters, atmospheric measurements and weather models.

2.4 Radiation Pressure

Radiation pressure model describes the momentum transfer between a free-flying object and the photons colliding with the said object.

The force resulting from radiation pressure is calculated as:

$$F_{radiation} = P \cdot A_{scaled} \tag{7}$$

...where $F_{radiation}$ is the radiation pressure force, P is the light pressure and A_{scaled} is the equivalent pressure area.

The scaled area is calculated from the area of the face of the object that is illuminated by the radiation source, using the cosine of the light ray direction and the direction of the surface normal.

$$A_{scaled} = A_{illuminated} \cdot cos^2(\theta_{div}) \tag{8}$$

...where $cos^2(\theta_{div})$ is:

$$cos^2(\theta_{div}) = \frac{\boldsymbol{j} \cdot (-\boldsymbol{n})}{|\boldsymbol{j}| \cdot |-\boldsymbol{n}|} \tag{9}$$

...where $\boldsymbol{j}$ is the direction of the light rays and $\boldsymbol{n}$ is the (normalized) surface normal. When distant enough from a spherical or near-spherical radiation source, the light ray directions can be assumed to be in the direction of the "zenith", or directly away from the sphere.

The pressure is calculated using the flux density:

$$P = \frac{\phi}{c} \tag{10}$$

...where ϕ is the flux density and c is the speed of light, exactly 299792458 $m\ s^{-1}$.

For a spherical or near-spherical radiation source such as a star, the flux density at a given distance can be found as:

$$\phi(r) = \frac{L}{4\pi r^2} \tag{11}$$

...where L is the luminosity of the radiation source, and r is the distance from the center of the source.

To account for the contribution of multiple faces on radiation pressure effect, the effect of multiple radiation pressure force models can be added up on the same object.

2.5 Finite-Burn Maneuvers

Finite-burn maneuvers are orbital maneuvers during which the spacecraft travels a significant distance, prohibiting their modeling as "impulsive" maneuvers. These maneuvers can be modeled either by directly applying a pre-determined acceleration profile, or a predetermined thrust profile.

In case of a pre-determined acceleration, the model becomes quite simple.

$$\boldsymbol{a} = \boldsymbol{a}_{maneuver} \tag{12}$$

The pre-set thrust model is also quite simple, except the changing mass of the spacecraft has to be taken into account.

$$\boldsymbol{a} = \frac{\boldsymbol{F}_{maneuver}}{m(t)} \tag{13}$$

...where $F_{maneuver}$ is the pre-determined thrust, and $m(t)$ is the mass of the spacecraft at a given time. This requires the computation of mass flow with respect to time, which can be quite case-specific depending on the exact engineering of a propulsion system. A simple way to model it is to use a mass flow rate that is scaled proportionally to the given thrust and a maximum thrust rating, together with a maximum mass flow rate.

$$\dot{m} = -\frac{F_{maneuver} \cdot \dot{m}_{max}}{F_{max}} \tag{14}$$

This same scheme can also be used to model outgassing objects or similar other physical phenomena.

As opposed to a finite-burn maneuver, an impulsive maneuver does not require a force model, and can simply be used as a change in velocity, Δv. Such a velocity-change model can also model other sudden interactions such as a collision.

2.6 Relativistic Corrections

The relativistic corrections used in OrbitSim3D are discussed in detail in [2].

The so-called Schwarzschild term is calculated as [2]:

$$\Delta\ddot{\boldsymbol{r}} = \frac{GM}{c^2 r^3} \cdot \left\{ \left[2(\beta+\gamma)\frac{GM}{r} - \gamma\dot{\boldsymbol{r}} \cdot \dot{\boldsymbol{r}} \right] \boldsymbol{r} + 2(1+\gamma)(\boldsymbol{r} \cdot \dot{\boldsymbol{r}})\dot{\boldsymbol{r}} \right\} \tag{15}$$

...and the frame-dragging (Lense-Thirring) term is calculated as [2]:

$$\Delta\ddot{\boldsymbol{r}} = \frac{GM}{c^2 r^3}(1+\gamma) \left[\frac{3}{r^2}(\boldsymbol{r} \times \dot{\boldsymbol{r}})(\boldsymbol{r} \cdot \boldsymbol{J}) + (\dot{\boldsymbol{r}} \times \boldsymbol{J}) \right] \tag{16}$$

...where $\boldsymbol{r}$ is the position of a satellite with respect to the body, c is the speed of light, G is the gravitational constant, M is the mass of the body, β and γ are Parametrized Post-Newtonian Formalism parameters equal to 1 in General Relativity, J is the angular momentum of the body per unit mass.

The De-Sitter term is not implemented in OrbitSim3D.

It should be noted that these equations are approximations for weak fields and may not accurately model all relativistic cases.

3 Trajectory Computation

3.1 Modular Physical Models

Due to their discrete component nature, the force and acceleration models for different celestial bodies and spacecraft may be implemented individually. For example, an atmospheric drag model may have a significant impact on the trajectory of a Low Earth Orbit satellite, but not so much for that of Luna. In that case, the atmospheric drag model may only be implemented for the satellite. This "modular physics" approach can save time and computational resources as opposed to a model where all physical interactions are computed regardless of their estimated significance. However, this approach leaves it up to the user to decide what may be important or not. Therefore, it would be advisable for a user to compare results with more and more sophisticated models until where the complexity cut-off can be decided.

3.2 Time Integration

Due to its numerical nature and the potential complexity of the force and acceleration models, OrbitSim3D uses an explicit time-integration scheme. In each time step, all acceleration components from all physical models implemented for each object is calculated, and a time step is taken. Each time step may have multiple sub-steps if higher order integration schemes are used, in which case, the accelerations will have to be re-calculated for each sub-step as well. If enabled, a relative error tolerance will be checked at each time step by using two half-steps instead of a single step, and if the discrepancies in the results are too high, the time step will be attempted again from scratch with a lower time delta, Δt, until the relative error criteria are met.

In certain cases, the energy of the overall system represented by numerous spacecraft and/or celestial bodies may be conserved. To get the best performance regarding the non-divergence of overall energy, symplectic integrators are preferred, the most basic (and lowest order) of which is the Symplectic Euler method. It can be described easily as follows:

$$\begin{aligned} \boldsymbol{v}_{n+1} &= \boldsymbol{v}_n + \boldsymbol{a}_n \cdot \Delta t \\ \boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \boldsymbol{v}_n \cdot \Delta t \end{aligned} \tag{17}$$

...where $\boldsymbol{x}$ represents the position of an object, $\boldsymbol{v}$ represents the velocity, $\boldsymbol{a}$ represents the acceleration and Δt represents the time step size. This is an extremely simple modification to the usual Forward Euler method, but one that mostly eliminates the energy-accumulating trend that Forward Euler is prone to have during orbital simulations.

Symplectic Euler scheme can be limited in its uses due to its low-order nature, which requires low time-step sizes for good accuracy. Higher order schemes such as the Velocity Verlet method, or even higher order integrators can be used. OrbitSim3D allows the time-integration solver scheme to be switched

between the implemented solver functions, either before the beginning of the simulation or at any point during runtime.

3.3 Bringing It All Together

The force models and temporal solvers work together in an attempt to create as realistic of a model of the universe as possible. There can be many ways to design how the solver and the mathematical models interact. OrbitSim3D opts to use a method such as the one described below.

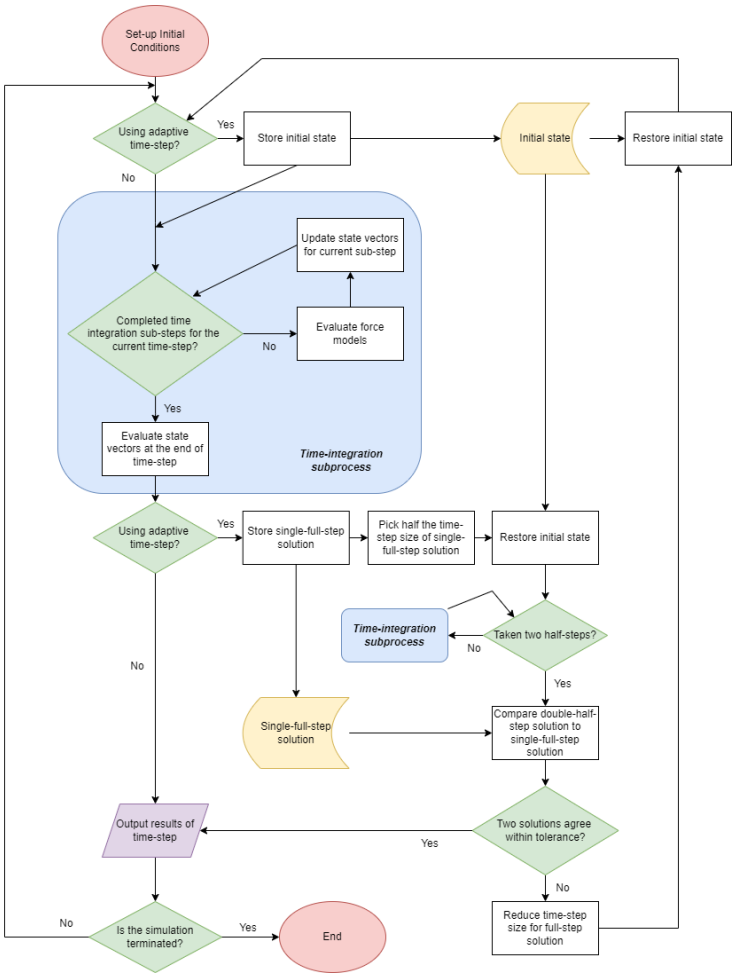

Figure 1: Simplified flowchart of the numerical model of OrbitSim3D

Figure 1 is a simplified model of the numerical solution method used in OrbitSim3D. The chart does not include user interaction processes, and the

way inputs and outputs are handled at runtime are not shown as to keep the figure clean.

4 Conclusion

OrbitSim3D connects various force models together with a few choices of numerical time-integration schemes, allowing for the creation of versatile simulations of astrodynamics and celestial mechanics-related systems. Owing to its modularity, implementation of new models and solvers is also made easy. The solution parameters can be changed at runtime without losing data, which provides great freedom to the user.

The primary downside of the approach is that since the solution methods are not optimized for any one sort of problem, and furthermore, the program is given the ability to take user input during runtime (and therefore the responsibility of listening to the said input), it tends to be quite slower than a dedicated software for particular types of problems. On the bright side, it should not be very difficult to derive a certain "stripped-down" version of OrbitSim3D for any particular need that can work faster, perhaps implemented in multiple different programming languages.

References

[1] Dennis C Pak. Linearized equations for j2 perturbed motion relative to an elliptical orbit. 2005.

[2] K. Sośnica, G. Bury, R. Zajdel, K. Kazmierski, J. Ventura-Traveset, R. Prieto-Cerdeira, and L. Mendes. General relativistic effects acting on the orbits of galileo satellites. *Celestial Mechanics and Dynamical Astronomy*, 133(4):14, Mar 2021.