HIGH ACCURACY SIMULATION OF ORBIT DYNAMICS: AN OBJECT-ORIENTED APPROACH

Francesco Casella, Marco Lovera

Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy

casella@elet.polimi.it(Francesco Casella)

Abstract

The development process for spacecraft control systems relies heavily on modelling and simulation tools for spacecraft dynamics. For this reason, there is an increasing need for adequate design tools in order to cope efficiently with tightening requirements for simulation accuracy and efficiency. In the last few years a Modelica library for spacecraft modelling and simulation has been developed, on the basis of the Modelica Multibody Library; the aim of this paper is to demonstrate improvements in terms of simulation accuracy and efficiency which can be obtained by using Keplerian parameters instead of Cartesian coordinates as state variables in the spacecraft model. Thanks to the features of the Modelica modelling language, and of the tools supporting it, it is straightforward to extend the rigid body model of the standard Multi-Body library, by adding the equations defining a transformation of the body center-of-mass coodinates from Keplerian parameters to Cartesian coordinates, and by setting the former as preferred states, instead of the latter; the tool then handles the state transformation automatically. The remaining parts of the model, including the model of the gravitational field, are left untouched, thus ensuring maximum re-usability of third-party code. The results shown in the paper demonstrate the superior accuracy and speed of computation in the reference case of a point-mass gravity field.

Keywords: Spacecraft dynamics, Object-oriented modelling, Modelica, Numerical integration.

Presenting Author's Biography

Francesco Casella. Francesco Casella got his Electronics Engineering degree in 1994 and a PhD in Computer and Control Science in 1999 from Politecnico di Milano, where he is Assistant Professor since 2001. His current research interests include modelling and control of energy conversion systems, and object-oriented modelling for control applications in general. He's an active member of the Modelica Association. He has published over 40 papers in referenced journals and international conference proceedings.

ISBN 978-3-901608-32-2

1 Introduction

The safe and satisfactory operation of a satellite, in terms of its mission objectives, is strongly related to the performance level of its on-board attitude and orbit control systems, which provide the ability to maintain a desired orientation in space (or, e.g., carry out predefined attitude maneuvers) and track a desired, nominal orbit in spite of the presence of external disturbances. In addition, the recent trend towards missions based on constellations or formations of small satellites has led to the formulation of even more complex control problems, related to the relative motion (both in terms of attitude and position) of more vehicles at a time. This has resulted in an increasing need for efficient design tools in every domain involved in spacecraft design, and particularly in the area of control-oriented modelling and simulation. Specific tools have to be developed for the design of both the system architecture and the Attitude and Orbit Control System (AOCS), bearing in mind the principles of reusability, flexibility and modularity. The main issue in the development of such tools should be to try and work out a unified environment to be used throughout the life cycle of the AOCS software, namely, the mission analysis stage, the preliminary and detailed design and simulation phases, the generation and testing of the on-board code, the development of the AOCS Electrical Ground Support Equipment (EGSE) and the post-launch data analysis activities. A number of commercial tools are available to support one or more of the above mentioned phases in the development of AOCS subsystems, however none of them seems capable of providing complete coverage of the whole development cycle in a sufficiently flexible way.

It is expected that a systematic approach to modelling and simulation, based on modern a-causal object-oriented languages such as Modelica (see [1, 2]), will eventually lead to the development of spacecraft simulation tools, the use of which would be made much more efficient by the very nature of the selected modelling approach. Note, in passing, that there is an increasing interest for multidomain problems in the spacecraft control design community (see, e.g., [3]), an area which would benefit from the availability of simulation tools based on the object-oriented approach. The development of simulation tools for satellite attitude and orbit dynamics within the object-oriented paradigm has been the subject of previous work (see [4], where an overview of the existing tools for AOCS modelling is presented). Surprisingly enough, however, while the use of Modelica for aerospace applications has recently led to the development of a library for flight dynamics (see [5]), very little activity in the spacecraft domain has been reported. Some preliminary results in the development of a Modelica spacecraft modelling library have been presented in [6, 7, 8]). More recently, the model components presented in the cited references have been revised in order to take advantage of the Modelica Multibody library (see [9]), which turns out to be extremely suitable to serve as a basis for the development of the basic model components for the mechanical parts of spacecraft models. In particular, a recent extension of the above mentioned library (see [10, 11]) is proving specially beneficial for the simulation of spacecraft with flexible appendages (see also [12]).

Therefore, the aims of this paper are the following:

- to demonstrate improvements in terms of simulation accuracy and efficiency which can be obtained by using Keplerian parameters instead of Cartesian coordinates as state variables in the spacecraft model;
- to illustrate how Keplerian parameters can be simply included in the existing multibody spacecraft model by exploiting the object-oriented features of the Modelica language and the symbolic manipulation capability of Modelica tools.

The paper is organised as follows: first an overview of the available choices for the state representation of satellite orbits is given in Section 2; subsequently, the use of Keplerian orbital elements for the simulation of orbit dynamics will be described in Section 3, while the corresponding Modelica implementation will be outlined in Section 4 and the results obtained in the implementation and application of the proposed approach to the simulation of a Low Earth orbit will be presented and discussed in Section 5.

2 Satellite State Representations

The state of the center of mass of a satellite in space needs six quantities to be defined. These quantities may take on many equivalent forms. Whatever the form, we call the collection of these quantities either a state vector (usually associated with position and velocity vectors) or a set of elements called orbital elements (typically used with scalar magnitude and angular representations of the orbit). Either set of quantities is referenced to a particular reference frame and completely specifies the two-body orbit from a complete set of initial conditions for solving an initial value problem class of differential equations.

In the following subsections, we will refer to a spacecraft subject only to the gravitational attraction of the Earth considered as a point mass (*unperturbed Keplerian conditions*).

2.1 Position and Velocity Coordinates

In the Earth-Centered Inertial (ECI) reference frame, the position and velocity vectors of a spacecraft influenced only by the gravitational attraction of the Earth considered with punctiform mass will be denotated as follows

$$r = \begin{bmatrix} x & y & z \end{bmatrix}^T, \tag{1}$$

$$v = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T = \frac{dr}{dt}. \tag{2}$$

ISBN 978-3-901608-32-2

The acceleration of such a spacecraft satisfies the equation of two-body motion

$$\frac{d^2r}{dt^2} = -GM_{\oplus}\frac{r}{\|r\|^3} \tag{3}$$

where $\mu = GM_{\oplus}$ is the gravitational coefficient of the Earth. A particular solution of this second order vector differential equation is called an orbit that can be elliptic or parabolic or hyperbolic, depending on the initial values of the spacecraft position and velocity vectors $r(t_0)$ and $v(t_0)$. Only circular and elliptic trajectories are considered in this study.

The state representation by position and velocity of a spacecraft in unperturbed Keplerian conditions is

$$x = [x \quad y \quad z \quad v_x \quad v_y \quad v_z]^T \tag{4}$$

at a given time t. Time t is always associated with a state vector and it is often considered as a seventh component. A time used as reference for the state vector or orbital elements is called the epoch.

2.2 Classical Orbital Elements

The most common element set used to describe elliptical orbits (including circular orbits) are the classical orbital elements (COEs), also called the Keplerian parameters, which are described in the sequel of this Section. The COEs are defined as follows:

- a : semi-major axis, [m];
- n : mean motion, [sec^{-1}]
- e : eccentricity, [dimensionless];
- i : inclination, [rad];
- Ω : right ascension of the ascending node, [rad];
- ω : argument of perigee, [rad];
- ν : true anomaly, [rad];
- E : eccentric anomaly, [rad];
- M : mean anomaly, [rad];

(see Figures 1 and 2). The definitions of the COEs are referenced to the ECI frame.

- The semi-major axis a specifies the size of the orbit. Alternatively, the mean motion

$$n = \sqrt{\frac{GM_{\oplus}}{a^3}} \tag{5}$$

can be used to specify the size.

- The eccentricity e specifies the shape of the ellipse. It is the magnitude of the eccentricity vector, which points toward the perigee along the line of apsis.

- The inclination i specifies the tilt of the orbit plane. It is defined as the angle between the angular momentum vector $h = r \times v$ and unit vector $\boldsymbol{Z}$

$$\cos i = \frac{\boldsymbol{Z} \cdot h}{\|h\|}. \tag{6}$$

- The right ascension of the ascending node Ω is the angle from the positive X axis to the node vector n pointing toward the ascending node, that is the point on the equatorial plane where the orbit crosses from south to north. The node vector n is defined as

$$n = \boldsymbol{Z} \times h. \tag{7}$$

The cosine of the right ascension of the ascending node is then

$$\cos\Omega = \frac{\boldsymbol{X} \cdot n}{\|n\|}. \tag{8}$$

A quadrant check must be done because Ω can vary from 0 to 2π. If the component of n along the Y axis is negative, then

$$\Omega = 2\pi - \arccos\left(\frac{\boldsymbol{X} \cdot n}{\|n\|}\right). \tag{9}$$

- The argument of perigee ω is measured from the ascending node to the perigee, i.e. to the eccentricity vector $\boldsymbol{e}$ pointing toward the perigee

$$\cos\omega = \frac{n \cdot \boldsymbol{e}}{\|n\|e}. \tag{10}$$

A quadrant check must be done because ω can vary from 0 to 2π. If the component of $\boldsymbol{e}$ along the Z axis is negative, then

$$\omega = 2\pi - \arccos\left(\frac{n \cdot \boldsymbol{e}}{\|n\|e}\right). \tag{11}$$

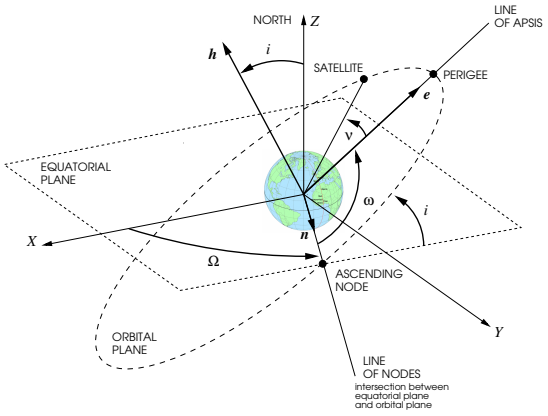

Fig. 1 *Classical Orbital Elements (COEs)*

- The eccentric anomaly E is defined on the auxiliary circle of radius a, that can be drawn around the elliptical orbit, as shown in Fig. 2. The sine and cosine of the eccentric anomaly are related to eccentricity and to the true anomaly ν according to the following relations

$$\sin E = \frac{\sin\nu\sqrt{1-e^2}}{1+e\cos\nu} \tag{12}$$

$$\cos E = \frac{e+\cos\nu}{1+e\cos\nu}. \tag{13}$$

In this work, satellite state representation in terms of classical orbital elements (Keplerian parameters) will be denoted as

$$\boldsymbol{x}_{COE} = [a \quad e \quad i \quad \Omega \quad \omega \quad E]^T \tag{14}$$

with the implicit choice of adopting E as a parameter to represent the spacecraft anomaly.

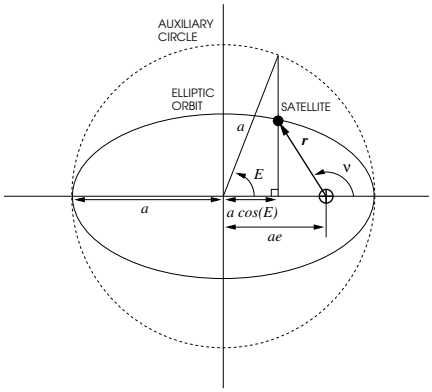

Fig. 2 *True and eccentric anomalies for elliptic motion*

2.3 Conversion Formulas

In this Section, the transformation from Keplerian to Cartesian coordinates will be briefly summarised (see [13] for details). With reference to Figure 2 and with obvious definitions for the x and y axes, z being normal to the orbit plane, we have

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a\cos(E) - ae \\ a\sin(E)\sqrt{1-e^2} \\ 0 \end{bmatrix}. \tag{15}$$

As depicted in Fig. 3, the orthogonal basis $\boldsymbol{RTN}$ of the Gaussian coordinate system can be obtained from the orthogonal basis $\boldsymbol{XYZ}$ of the ECI frame be means of three successive rotations

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \boldsymbol{\mathcal{R}}_{ZXZ}\left(\boldsymbol{x}_{COE}\right) \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \end{bmatrix}. \tag{16}$$

with

$$\boldsymbol{\mathcal{R}}_{ZXZ}\left(\boldsymbol{x}_{COE}\right) = \boldsymbol{\mathcal{R}}_Z(\omega)\boldsymbol{\mathcal{R}}_X(i)\boldsymbol{\mathcal{R}}_Z(\Omega) \tag{17}$$

where matrix

$$\boldsymbol{\mathcal{R}}_Z(\Omega) = \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

describes the first rotation around the Z axis of an angle Ω, matrix

$$\boldsymbol{\mathcal{R}}_X(\Omega) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix} \tag{19}$$

describes the second rotation around the X of an angle i, matrix

$$\boldsymbol{\mathcal{R}}_Z(\omega) = \begin{bmatrix} \cos(\omega) & \sin(\omega) & 0 \\ -\sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{20}$$

describes the third rotation around the Z axis of an angle $\omega + \nu$.

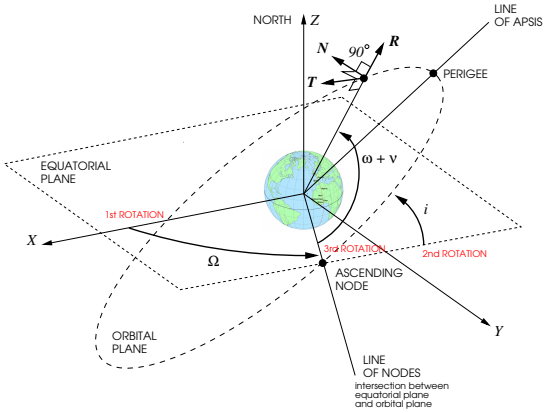

Fig. 3 *Rotations to convert from ECI to RTN reference frame*

3 COEs for simulation of orbit dynamics

When orbital control problems are studied, it is usually necessary to integrate the equations of motion of the satellite under the action of the earth (or any other celestial body) and of the actuators' thrust. The usual approach, known as Cowell's method (see [14]), is to integrate the equations of motion in cartesian coordinates

$$\dot{r} = v \tag{21}$$

$$\dot{v} = a_g(r) + f \tag{22}$$

where r is the cartesian coordinate vector of the satellite center of mass, a_g is the acceleration of gravity and f is the sum of the forces applied by the actuators. First-cut models assume a point-mass model

$$a_g = -GMr/\|r\|^3, \tag{23}$$

while accurate simulations require more detailed models of the gravitational field, usually in the form of a series expansion (see, e.g., [15]). In both cases, the differential equations are strongly non-linear; therefore, despite the use of high-order integration algorithm, tight tolerances end up in a fairly high number of simulation steps per orbit.

If the satellite motion is described in terms of COE's, it is easy to observe that the variability of the six orbit elements is much smaller than that of the cartesian coordinates. In particular, it is well-known that in case of a point-mass gravity field with no other applied forces, the first five parameters are constant, while the anomaly increases in time with slowly varying speed. All existing high-order integration methods have error bounds which depend on Taylor expansions of the state trajectory. One can then conjecture that if the COE's are used as state variables, instead of the Cartesian vectors r and v, the state trajectories will be smoother, and therefore the integration algorithm will be able to estimate them with the same relative precision using much larger time steps, or with increased precision using the same time steps as in the Cartesian coordinates case.

Defining the vectors

$$x = \begin{bmatrix} r^T & v^T \end{bmatrix}^T \tag{24}$$

$$z = \begin{bmatrix} a & e & i & \Omega & \omega & E \end{bmatrix}^T \tag{25}$$

equations (15)-(16) and (21)-(22) can be written in compact form as

$$\dot{x} = f(x) \tag{26}$$

$$x = g(z). \tag{27}$$

If a state variable change from x to z is performed, the following equations are obtained

$$\frac{\partial g(z)}{\partial z}\dot{z} = f(g(z)) \tag{28}$$

which can be solved for $\dot{z}$ provided that the COE's are uniquely defined

$$\dot{z} = \left(\frac{\partial g(z)}{\partial z}\right)^{-1} f(g(z)) \tag{29}$$

$$x = g(z). \tag{30}$$

If the orbit is circular and/or equatorial, the Jacobian becomes singular; in this case, a different parameterization of the orbital elements is needed (e.g., the equinoctial parameters), but this analysis is beyond the scope of this paper.

The model (29)-(30), which is now in standard state-space form, has two very important features:

- the right-hand side of (29) is much less variable than the right-hand side of (26), so it will be easier to integrate the equations with a given accuracy;
- in case an accurate model of the gravity field is used, it is not necessary to reformulate it in terms of the COE, because the right-hand side of (29) uses the compound function $f(g(z))$.

4 Modelica implementation

The concepts outlined in Section 3 are easily implemented using the object-oriented modelling language Modelica (see [16]). The goal is to enhance an already existing library for the simulation of satellite system dynamics [8, 17] with the increased accuracy in the determination of the position of the center of mass given by model (29)-(27).

The starting point is the BodyFrame model of the standard Modelica.Multibody library [9]: this is a 6 degrees-of-freedom model of a rigid body, which can be connected to other components to form a multibody system model. The original model has six potential state variables: the three cartesian coordinates of the center of mass, and three other suitable variables describing the body orientation. Assuming that the gravitational field is applied exactly at the center of mass (thus ignoring gravity gradient effects), the translational and rotational equations are completely decoupled, so it is possible to focus on the former ones, leaving the latter ones untouched.

The new model BodyKepler is obtained by inheritance from the BodyFrame model, with the following additions:

1. the equations to compute gravity acceleration *as a function of the cartesian coordinates* using more general models are added by inheritance to the standard World model of the MultiBody library (see [8, 17]);
2. the COE's a, e, i, ω, Ω, E are added as new model variables;
3. the equation relating them to the cartesian coordinates (15)-(16) are added;
4. the stateSelect attribute is switched to StateSelect.avoid for the r and v vectors, and to StateSelect.prefer for the COE's from the x variables, and switched on. The Modelica tool will then perform the transformation from (26)-(27) to (29)-(30) automatically, using symbolic manipulation algorithms.

The Modelica code defining the new model (listed in the Appendix and corresponding to steps 2-4), is very compact and easy to check, which is an important feature to ensure the correctness of the resulting model. As already noted, the accurate models of the gravity field, previously implemented in [8, 17], can still use the Cartesian coordinates as inputs, and are thus left unchanged.

5 Simulation results

In this Section, the results obtained in comparing the accuracy obtained by simulating the orbit dynamics for two Low Earth orbiting (LEO) spacecraft will be presented. As previously mentioned, for the purpose of the

present study we focus on the simulation of the unperturbed dynamics, i.e., only the gravitational acceleration computed from a point-mass model for the Earth is considered. In this case, the orbit is an ellipse (closed curve), having well-defined features. Therefore, this assumption allows us to introduce two simple criteria in order to evaluate the accuracy of the performed simulations, namely:

- The period of an unperturbed elliptical orbit can be computed a priori and is given by $T = 2\pi\sqrt{\frac{a^3}{\mu}}$, so a first measure of simulation accuracy can be given by the precision with which the orbit actually closes during the simulation. To this purpose, the following stopping criterion has been defined for the simulation: the integration is stopped when the position vector crosses a plane orthogonal to the initial velocity and passing through the initial position. Then, the final time is compared with the orbit period and the final position is compared with the initial one.
- Furthermore, for an unperturbed orbit the angular momentum $h = r \times v$ should remain constant, so a second measure of accuracy for the simulation is given by the relative error in the value of h, i.e., the quantity

$$e_h = \frac{\|h - h(0)\|}{\|h(0)\|}. \tag{31}$$

The considered orbits have been simulated using the Dymola tool, both using Cartesian and Keplerian coordinates, in order to evaluate the above-defined precision indicators. In both cases the DASSL integration algorithm has been used, with the smallest feasible relative tolerance 10^{-12}.

5.1 A near-circular, LEO orbit

The first considered orbit is a LEO, near circular one (see Figure 4), characterised by the following initial state, in Cartesian coordinates:

$$r(0) = \begin{bmatrix} 6828.140 \times 10^3 \\ 0 \\ 0 \end{bmatrix},$$

$$v(0) = \begin{bmatrix} 0 \\ 5.40258602956241 \times 10^3 \\ 5.40258602956241 \times 10^3 \end{bmatrix}$$

In Table 1 the precision achieved in the actual closure of the orbit is shown: as can be seen, the simulated period is very close to the actual one and both the period error and the position error are significantly smaller when simulating the orbital motion using Keplerian rather than Cartesian states.

Similarly, in Figure 5 the time histories of the relative error on the value of the orbital angular momentum are illustrated, for a simulation of about one day: the results are clearly very satisfactory in both cases, however while in the case of Cartesian states the relative

Tab. 1 Orbit closure errors, using Cartesian and Keplerian coordinates - near circular orbit.

States	ΔT [s]	$\|\Delta r\|$ [m]
Cartesian	-1.00332×10^{-6}	1.69711×10^{-3}
Keplerian	2.38369×10^{-8}	2.17863×10^{-5}

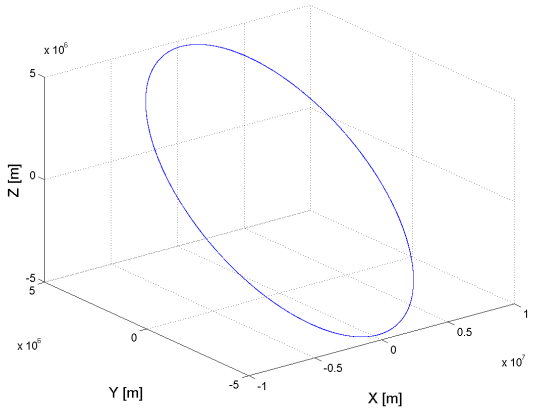

Fig. 4 The considered LEO, near circular orbit.

error is significantly larger than machine precision and is slowly increasing, in the case of Keplerian states the relative error is much smaller and appears to be more stable as a function of time (see also Table 2). Finally,

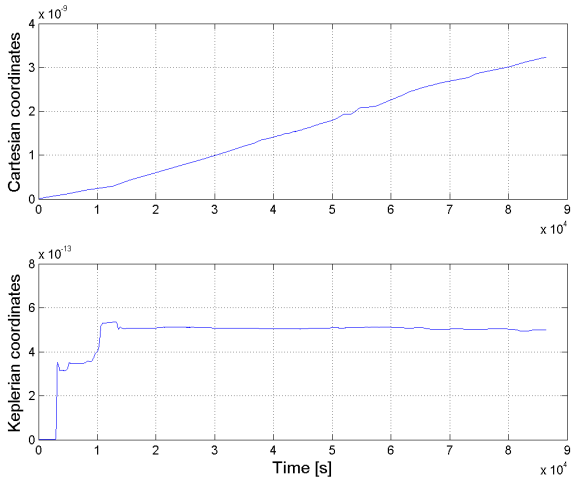

Fig. 5 Relative errors on the orbit angular momentum - near circular orbit: Cartesian (top) and Keplerian (bottom) coordinates.

note that the use of Keplerian parameters also leads to significant benefits in terms of simulation efficiency, as can be seen from Table 3.

5.2 A highly elliptical, LEO orbit

The second considered orbit is again a LEO one, but it is characterised by a high value of the eccentricity

Tab. 2 Mean angular momentum error, using Cartesian and Keplerian coordinates - near circular orbit.

State variables	Mean e_h
Cartesian	1.5373×10^{-9}
Keplerian	4.7528×10^{-13}

Tab. 3 Number of steps for a one-orbit simulation - near circular orbit.

State variables	Number of steps
Cartesian	959
Keplerian	376

(see Figure 6, where it is also compared with the circular orbit considered in the previous case) and by the following initial state, in Cartesian coordinates:

$$r(0) = \begin{bmatrix} 6828.140 \times 10^3 \\ 0 \\ 0 \end{bmatrix},$$

$$v(0) = \begin{bmatrix} 0 \\ 5.40258602956241 \times 10^3 \\ 7.29349113990925 \times 10^3 \end{bmatrix}$$

As in the previous case, Table 4 shows the precision achieved in the actual closure of the orbit: as can be seen, the errors on the simulated period are of the same order of magnitude for both choices of state variables. The position errors, on the other hand are significantly smaller when simulating the orbital motion using Keplerian rather than Cartesian states.

Tab. 4 Orbit closure errors, using Cartesian and Keplerian coordinates - highly elliptical orbit.

States	ΔT [s]	$\|\Delta r\|$ [m]
Cartesian	-1.17226×10^{-5}	4.39241×10^{-3}
Keplerian	1.48665×10^{-5}	2.67799×10^{-7}

Similarly, in Figure 7 the time histories of the relative error on the value of the orbital angular momentum are illustrated, for a simulation of about one day. In this case, the results show that using Cartesian states the relative error is again significantly larger than machine precision and is slowly increasing, while using Keplerian states the relative error is of the order of machine precision (see also Table 5).

Finally, the gain in terms of simulation efficiency can be verified from Table 6.

6 Concluding remarks

A method for the accurate simulation of satellite orbit dynamics on the basis of the Modelica MultiBody library has been presented. The proposed approach is

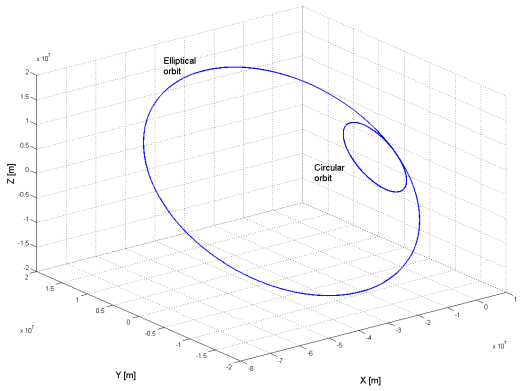

Fig. 6 The considered LEO, highly elliptical orbit, compared with the circular one considered in Section 5.1.

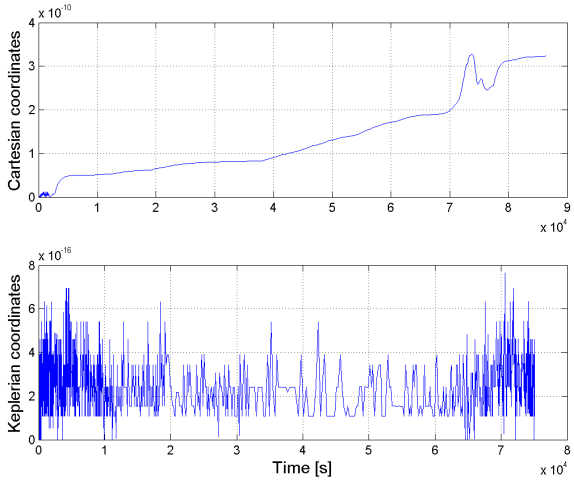

Fig. 7 Relative errors on the orbit angular momentum - highly elliptical orbit: Cartesian (top) and Keplerian (bottom) coordinates.

based on the use of Keplerian parameters instead of Cartesian coordinates as state variables in the spacecraft model. This is achieved by adding to the standard Body model the equations for the transformation from Keplerian parameters to Cartesian coordinates and exploiting automatic differentiation. The resulting model ensures a significant improvement in numerical accuracy, does not require the solution of implicit equations and keeps the same interface and multibody structure of the standard component. Simulation results with a point-mass gravity field show the good performance of the proposed approach. The validation with higher order gravity field models is currently being performed.

7 References

[1] Modelica - a unified object-oriented language for physical systems modelling. Language specification. Technical report, Modelica Association, 2002.

Tab. 5 Mean angular momentum error, using Cartesian and Keplerian coordinates - highly elliptical orbit.

State variables	Mean e_h
Cartesian	1.2927×10^{-10}
Keplerian	2.5223×10^{-16}

Tab. 6 Number of steps for a one-orbit simulation - highly elliptical orbit.

State variables	Number of steps
Cartesian	3650
Keplerian	1120

[2] P. Fritzson and Bunus. Modelica - a general object-oriented language for continuous and discrete-event system modelling and simulation. In *Proceedings of the 35th IEEE Annual Simulation Symposium*, 2002.

[3] C. Roithmayr, C. Karlgaard, R. Kumar, and D. Bose. Integrated power and attitude control with spacecraft flywheels and control moment gyroscopes. *Journal of Guidance, Control, and Dynamics*, 27(5):859–873, 2004.

[4] A. Turner. An open-source, extensible spacecraft simulation and modeling environment framework. Master's thesis, Virginia Polytechnic Institute and State University, 2003.

[5] D. Moorman and G. Looye. The Modelica flight dynamics library. In *Proceedings of the 2nd International Modelica Conference, Oberpfaffenhofen, Germany*, 2002.

[6] M. Lovera. Object-oriented modelling of spacecraft attitude and orbit dynamics. In *54th International Astronautical Congress, Bremen, Germany*, 2003.

[7] T. Pulecchi and M. Lovera. Object-oriented modelling of the dynamics of a satellite equipped with single gimbal control moment gyros. In *Proceedings of the 4th International Modelica Conference, Hamburg, Germany*, volume 1, pages 35–44, 2005.

[8] M. Lovera. Control-oriented modelling and simulation of spacecraft attitude and orbit dynamics. *Journal of Mathematical and Computer Modelling of Dynamical Systems, Special issue on Modular Physical Modelling*, 12(1):73–88, 2006.

[9] M. Otter, H. Elmqvist, and S. E. Mattsson. The new Modelica multibody library. In *Proceedings of the 3nd International Modelica Conference, Linköping, Sweden*, 2003.

[10] G. Ferretti, F. Schiavo, and L. Viganò. Object-Oriented Modelling and Simulation of Flexible Multibody Thin Beams in Modelica with the Finite Element Method. In 4^{th} *Modelica Conference*, Hamburg-Harburg, Germany, March 7-8, 2005.

[11] F. Schiavo, L. Viganò, and G. Ferretti. Modular modelling of flexible beams for multibody systems. *Multibody Systems Dynamics*, 12(1):73–88, 2006.

[12] F. Schiavo and M. Lovera. Modelling, simulation and control of spacecraft with flexible appendages. In *Proc. of the 5th International Symposium on Mathematical Modelling, Vienna, Austria*, 2006.

[13] M. Sidi. *Spacecraft dynamics and control*. Cambridge University Press, 1997.

[14] V. Chobotov. *Orbital Mechanics*. AIAA Education Series, Second edition, 1996.

[15] J. Wertz. *Spacecraft attitude determination and control*. D. Reidel Publishing Company, 1978.

[16] S. E. Mattsson, H. Elmqvist, and M. Otter. Physical system modeling with Modelica. *Control Engineering Practice*, 6(4):501–510, 1998.

[17] T. Pulecchi, F. Casella, and M. Lovera. A Modelica library for Space Flight Dynamics. In *Proceedings of the 5th International Modelica Conference, Vienna, Austria*, 2006.

Appendix: Modelica code for the BodyKepler model

```
model BodyKepler
  "Body model with computation of Kepler parameters"
  extends Modelica.Mechanics.MultiBody.Parts.Body(
    frame_a(r_0(stateSelect = StateSelect.never)),
    v_0(stateSelect = StateSelect.never));

  import Modelica.Math.*;
  constant Real pi = Modelica.Constants.pi;

  parameter Integer initCoordinates = 1 "0: none, 1: cartesian, 2: keplerian";
  parameter Integer stateChoice = 1 "0: cartesian, 1; Keplerian";
  parameter OrbitalPosition r_start[3] = {6800, 0, 0}
    "Start value of position in cartesian coordinates";
  parameter OrbitalVelocity v_start[3] = {0, 7200, 0}
    "Start value of initial velocity in cartesian coordinates";
  parameter OrbitalPosition a_start=
    sqrt(r_start*r_start)/(2-sqrt(r_start*r_start)*v_start*v_start/GM)
    "Start value of initial semi-major axis";
  parameter Real e_start = 0.1 "Start value of initial eccentricity";
  parameter Real i_start = pi/4 "Start value of initial inclination";
  parameter Real Omega_start = 0
    "Start value of initial longitude of ascending node";
  parameter Real omega_start = 0 "Start value of initial argument of perigee";
  parameter Real E_start = 0 "Start value of initial anomaly";
  parameter Real GM = world.mue "Gravitational constant times mass";

  // Cartesian position and velocity
  OrbitalPosition r[3](start = r_start) "Position in cartesian coordinates";
  OrbitalVelocity v[3](start = v_start) "Velocity in cartesian coordinates";
  OrbitalPosition r_orb[3] "Position in the orbit plane";
  OrbitalVelocity v_orb[3] "Velocity in the orbit plane";
  OrbitalPosition2 rmod2;
  OrbitalPosition rmod;
  OrbitalVelocity2 vmod2;

  // Rotation matrices to convert from ECI to RTN frame
  Real A_omega[3,3];
  Real A_Omega[3,3];
  Real A_i[3,3];

  // Keplerian parameters
  OrbitalPosition a(
    stateSelect = if stateChoice == 1 then StateSelect.always else StateSelect.default,
    start = a_start, min = 0) "Semi-major axis";
  Real e(
    stateSelect = if stateChoice == 1 then StateSelect.always else StateSelect.default,
    start = e_start, min = 0) "Eccentricity";
  Real i(
    stateSelect = if stateChoice == 1 then StateSelect.always else StateSelect.default,
    start = i_start) "Inclination";
  Real Omega(
    stateSelect = if stateChoice == 1 then StateSelect.always else StateSelect.default,
    start = Omega_start) "Longitude of ascending node";
  Real omega(
    stateSelect = if stateChoice == 1 then StateSelect.always else StateSelect.default,
    start = omega_start);
  Real E(
    stateSelect = if stateChoice == 1 then StateSelect.always else StateSelect.default,
    start = E_start);

  // Orbit period and mean angular frequency
  SI.Time T_orbit(start=10000);
  Real n "Mean orbital angular frequency";

equation
```



```
  // Connection to Body model
  r = frame_a.r_0;
  v = v_0;

  // Transformation from Keplerian parameters to cartesian coordinates
  T_orbit = 2*pi*sqrt(a^3/GM);
  n = 2*pi/T_orbit;

  rmod = sqrt(rmod2);
  rmod2 = r*r;
  vmod2 = v*v;

  r_orb[1] = a*(cos(E)-e);
  r_orb[2] = a*sqrt(1-e^2)*sin(E);
  r_orb[3] = 0;

  A_omega = [ cos(omega), sin(omega), 0;
             -sin(omega), cos(omega), 0;
                  0,          0,      1];
  A_Omega = [ cos(Omega), sin(Omega), 0;
             -sin(Omega), cos(Omega), 0;
                  0,          0,      1];
  A_i = [ 1,     0,     0;
          0,  cos(i), sin(i);
          0, -sin(i), cos(i)];

  r = transpose(A_Omega)*transpose(A_i)*transpose(A_omega)*r_orb;

  v_orb[1] = -a^2*n/rmod*sin(E);
  v_orb[2] = a^2*n/rmod*sqrt(1-e^2)*cos(E);
  v_orb[3] = 0;
  v = transpose(A_Omega)*transpose(A_i)*transpose(A_omega)*v_orb;

initial equation
  // Sets the initial conditions
  if initCoordinates == 1 then
    // initial conditions specified by start values of cartesian coordinates
    r = r_start;
    v = v_start;
  elseif initCoordinates == 2 then
    // initial conditions specified by start values of Keplerian parameters
    a = a_start;
    e = e_start;
    i = i_start;
    Omega = Omega_start;
    omega = omega_start;
    E = E_start;
  end if;

end BodyKepler;
```

ISBN 978-3-901608-32-2