AIAA/AAS Astrodynamics Specialist Conference
2 - 5 August 2010, Toronto, Ontario Canada

AIAA 2010-7658

Performance Aspects of Orbit Propagation using the Unified State Model

Vivek Vittaldev* , Erwin Mooij† , and Marc C. Naeije‡
Delft University of Technology, Delft, 2600 AA, The Netherlands

The Unified State Model is a method for expressing orbits using a set of seven elements. The elements consist of a quaternion and three parameters based on the velocity hodograph. The equations of this model and the background theory necessary to understand them have been shown here. Numerical simulations comparing the Unified State Model with the traditional Cartesian coordinates have been carried out for perturbed orbits, orbits with low-thrust propulsion, and a solar sailing trajectory. The Unified State Model outperforms Cartesian coordinates for all cases in terms of accuracy and computational speed, except for highly eccentric perturbed orbits. The performance of the Unified State Model is exceptionally better for the case of orbits with continuous low-thrust propulsion with CPU simulation time being an order of magnitude lower than for the simulation using Cartesian coordinates. This makes the Unified State Model especially suited for use in trajectory simulators and optimizers.

I. Introduction

Cartesian coordinates and classic Keplerian elements are well-established for the propagation and visualization of orbits. However, there are many other ways in which satellite orbits can be described. One of these methods is a novel concept known as the Unified State Model (USM). The USM, first proposed by Samuel P. Altman, uses quaternions and velocity hodograph parameters to express orbits.[1] The 4 quaternion elements express the orientation of a reference frame fixed to the orbiting body, with respect to an inertial frame fixed to the central body. The velocity hodograph parameters give information about the size and shape of the orbit. During an unperturbed orbit, the velocity hodograph parameters remain constant and only the quaternion varies with time. This is because the orbital shape remains unchanged for such an orbit, and only the location of the orbiting body within the orbit changes with time. When perturbations are present, the variation of the hodographic parameters will be small compared to the variation of Cartesian coordinates. Thus, the USM will have better numerical stability than Cartesian coordinates as there are 4 rapidly varying and 3 slowly varying elements, compared to the 6 rapidly varying elements. The classic Keplerian elements have only one rapidly varying element, which is the true anomaly. However, singularities exist and the equations for the influence of perturbations are rather complex.

There has not been much research carried out on the USM since its inception. If the theory has been mentioned, it has always been in relation to navigation. A method of satellite tracking in velocity space rather than the traditional position space was first shown by Altman,[2] which has been mentioned once,[3] but never investigated further. The USM equations have been shown in more detail in another technical note,[4] but with a term corrected with respect to the original model. The modified equations are also shown in the paper by Raol.[3] Both Chodas[4] and Raol[3] focus on satellite orbit tracking and mention that the USM allows for better trajectory simulation. It was concluded[4] that satellite orbit tracking from a ground station is slightly better using the traditional Cartesian coordinates than with the USM. Therefore, one strategy[3] is to use the USM for trajectory propagation and the Cartesian coordinates for state estimation.

It has been previously stated, without showing any results, that the USM is better for numerical integration than Cartesian coordinates.[1,3,4] One paper uses a modified version of the USM[5] as a way of

*Graduate Student, Faculty of Aerospace Engineering, v.vittaldev@student.tudelft.nl.
†Assistant Professor, Faculty of Aerospace Engineering, e.mooij@tudelft.nl, Senior member AIAA.
‡Assistant Professor, Faculty of Aerospace Engineering, m.c.naeije@tudelft.nl.

integrating rotating bodies. The comparison was performed with Euler angles and not orbital trajectories and Cartesian coordinates. Cartesian coordinates are used for expressing and integrating orbital trajectories in well known software like GEODYN-II,[6] the Satellite Tool Kit (STK),[7] and the General Mission Analysis Tool (GMAT).[8] If the USM improves the performance of numerical integration, it could be implemented in these software. This is investigated in this work by numerically integrating various orbital trajectories. With superior numerical performance, the USM would be especially suited for use in numerical optimization techniques like Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Especially, optimization of low-thrust orbits and solar sailing would benefit the most because of the long Time of Flight (TOF).

In Section II, the background theory required to understand the USM is presented. Section III shows how the equations of the USM can be derived. The results of some numerical simulations and the comparisons between the USM and Cartesian coordinates can be found in Section IV and finally some conclusions can be found in Section V.

II. Background Theory for the USM

In this section, the background information necessary for the derivation and the application of the USM is presented. The USM state consists of a quaternion and three hodographic velocities and therefore, quaternion and hodograph theory is discussed. First, however, some rotation theory is given to better understand quaternions.

A. Rotation

Reference frames are used to express the position and motion of bodies. Normally, these reference frames are dextral orthonormal triads, meaning that the first axis can be mapped onto the second axis by a 90° right-handed positive rotation along the third axis. Thus, a reference frame $\mathcal{F}_a$ consists of a set of three right-handed, mutually perpendicular vectors: $\hat{\mathbf{a}}_1$, $\hat{\mathbf{a}}_2$, and $\hat{\mathbf{a}}_3$.

It is often necessary to use different reference frames within the same set of calculations. In this case, it is required to convert vectors expressed in one reference frame to another. Suppose we have two reference frames, $\mathcal{F}_a$ and $\mathcal{F}_b$, that share an origin. The unit vectors that define $\mathcal{F}_b$ are related to the unit vectors that define $\mathcal{F}_a$ in the following manner:[9]

$$\hat{\mathbf{b}}_1 = c_{1,1}\hat{\mathbf{a}}_1 + c_{1,2}\hat{\mathbf{a}}_2 + c_{1,3}\hat{\mathbf{a}}_3 \tag{1a}$$

$$\hat{\mathbf{b}}_2 = c_{2,1}\hat{\mathbf{a}}_1 + c_{2,2}\hat{\mathbf{a}}_2 + c_{2,3}\hat{\mathbf{a}}_3 \tag{1b}$$

$$\hat{\mathbf{b}}_3 = c_{3,1}\hat{\mathbf{a}}_1 + c_{3,2}\hat{\mathbf{a}}_2 + c_{3,3}\hat{\mathbf{a}}_3 \tag{1c}$$

In Eq. (1), $c_{i,j}$ is the cosine of the angle between $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{a}}_j$, and is an element of the so-called Direction Cosine Matrix (DCM), $\mathbf{C}_{b,a}$. The frame transformation is carried out by pre-multiplying the vector defined in $\mathcal{F}_a$ with $\mathbf{C}_{b,a}$. To convert a vector from $\mathcal{F}_b$ to $\mathcal{F}_a$, the inverse of $\mathbf{C}_{b,a}$ should be used. As $\mathbf{C}_{b,a}$ is an orthonormal matrix, its inverse is simply the transpose.

Another fundamental concept in rotations of reference frames is Euler's theorem.[9] This theorem states that any rotation between two frames can be expressed by a rotation of an angle, Φ, around an axis $\hat{\mathbf{a}}$, through the shared origin of the two reference frames. This axis and angle are often referred to as the Euler axis and the Euler angle, respectively. Since the Euler axis-angle pair and the DCM both express the rotation between two frames, there exists a relationship between them. The DCM can be constructed from the Euler axis-angle pair using[9]

$$\mathbf{C} = \mathbf{I_3}\cos\Phi + (1-\cos\Phi)\hat{\mathbf{a}}\hat{\mathbf{a}}^T - [\hat{\mathbf{a}}\times]\sin\Phi \tag{2}$$

In Eq. (2) $\hat{\mathbf{a}} = [a_1, a_2, a_3]^T$ is the Euler axis, Φ is the Euler angle, and the term $[\hat{\mathbf{a}}\times]$ is the skew-symmetric form of the Euler axis and is defined as

$$[\hat{\mathbf{a}}\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3}$$

B. Quaternions

Attitude is synonymous with orientation and in the aerospace world it almost always means the orientation of a spacecraft or an aircraft. The attitude of a spacecraft relative to a reference frame actually means the orientation of a reference frame fixed to the body of the spacecraft with respect to another reference frame. Attitude parameters are just parameters that can describe this orientation. Thus, the DCM, Euler axis, and Euler angle are all attitude parameters.

A quaternion is a 4 dimensional hyper-complex number consisting of 1 real and 3 imaginary numbers. Quaternions of unit magnitude are used to express rotations, with the imaginary part being a vector, $\boldsymbol{\epsilon}$, and the real part a scalar, η. The quaternion can be created using the Euler axis-angle in the following manner:

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1, & \epsilon_2, & \epsilon_3 \end{bmatrix}^T = \hat{\mathbf{a}} \sin(\Phi/2) \tag{4a}$$

$$\eta = \cos(\Phi/2) \tag{4b}$$

Quaternions, especially when defined in the above manner, are also known as Euler parameters. In terms of complex numbers, the quaternion can be written as

$$Q = \eta + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k \tag{5}$$

where i,j, and k are all square roots of -1.

The rotation specified by the quaternion is expressed as $(\boldsymbol{\epsilon}, \eta)$. The 4 quaternion elements are not mutually independent because they satisfy the following constraint.

$$\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \eta^2 = \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2 = 1 \tag{6}$$

This shows that the quaternion describes a unit 3-sphere, meaning a sphere in 4 dimensional space. Any rotation is therefore a trajectory on the surface of the 3-sphere. Also, the orientation specified by $(\boldsymbol{\epsilon}, \eta)$ is the same as the orientation specified by $(-\boldsymbol{\epsilon}, -\eta)$. This is because the second quaternion describes the Euler rotation with Euler axis $-\mathbf{a}$ and Euler angle $-\Phi$. Thus, if $(\boldsymbol{\epsilon}, \eta)$ describes the shortest rotation then $(-\boldsymbol{\epsilon}, -\eta)$ describes the longest rotation. To ensure that the shortest rotation is taken, a necessary condition is that $\eta > 0$. To visualize this, quaternions can be viewed in another manner. The norm of the quaternion is 1, so the 4 quaternion elements represent the radial vector of a point on a 4-dimensional unit sphere. Since it is very difficult to imagine a 4-dimensional sphere, we can consider the vector part $\boldsymbol{\epsilon}$ to express the radial vector of a point on a 3-dimensional sphere. The radius of this 3-dimensional sphere, the norm of $\boldsymbol{\epsilon}$, can be found using the unit norm constraint of the quaternion in the following manner:

$$|\boldsymbol{\epsilon}| = \sqrt{1 - \eta^2} \tag{7}$$

Thus, the sphere expressing $\boldsymbol{\epsilon}$ changes with values of η. This will be a unit 3-dimensional sphere if $\eta = 0$, and only a point if $\eta = \pm 1$. Since η is in the fourth dimension, the 3-dimensional spheres of $\boldsymbol{\epsilon}$ corresponding to η can be thought to be present at various times like an animation. This can also be shown as a sphere with a linear offset from a point. Each offset corresponds to a certain value of η and all the points on the surface of the sphere express the possible values of $\boldsymbol{\epsilon}$. This can be seen in Fig. 1 along with a specific orientation.

A DCM can be constructed from a quaternion in the following manner:[9]

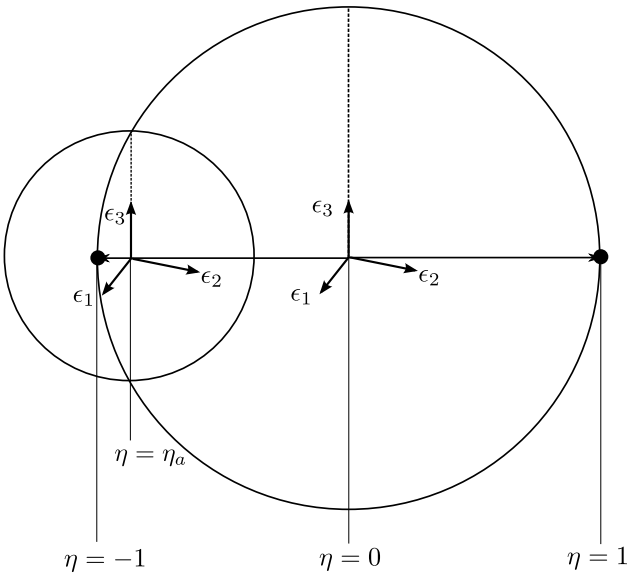
$$\mathbf{C} = \left(\eta^2 - \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}\right) \mathbf{I}_3 + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - 2\eta \left[\boldsymbol{\epsilon}\times\right] \tag{8}$$

This essential matrix can be written out in full as

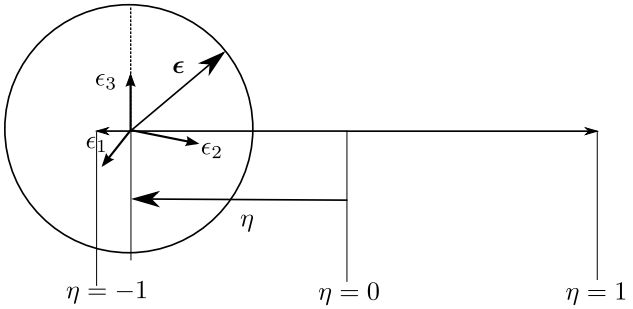
$$\mathbf{C} = \begin{bmatrix} 1 - 2\left(\epsilon_2^2 + \epsilon_3^2\right) & 2\left(\epsilon_1\epsilon_2 + \epsilon_3\eta\right) & 2\left(\epsilon_1\epsilon_3 - \epsilon_2\eta\right) \\ 2\left(\epsilon_2\epsilon_1 - \epsilon_3\eta\right) & 1 - 2\left(\epsilon_3^2 + \epsilon_1^2\right) & 2\left(\epsilon_2\epsilon_3 + \epsilon_1\eta\right) \\ 2\left(\epsilon_3\epsilon_1 + \epsilon_2\eta\right) & 2\left(\epsilon_3\epsilon_2 - \epsilon_1\eta\right) & 1 - 2\left(\epsilon_1^2 + \epsilon_2^2\right) \end{bmatrix} \tag{9}$$

Inversely, there are two methods to extract a quaternion from a DCM. The first method is very compact, but has a singularity at $\eta = 0$. In this method, the unit magnitude property of a quaternion is used to find the η from the DCM found in Eq. (9).

$$\eta = \pm\frac{1}{2}\sqrt{Tr\mathbf{C} + 1} \tag{10}$$



(a) 3 dimensional spheres expressing the constraint for $\boldsymbol{\epsilon}$ for various values of η



(b) Expression of a specific orientation ($\boldsymbol{\epsilon}$,η)

Figure 1. Visualizing quaternions

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \frac{1}{4\eta} \begin{bmatrix} c_{2,3} - c_{3,2} \\ c_{3,1} - c_{1,3} \\ c_{1,2} - c_{2,1} \end{bmatrix} \tag{11}$$

In Eq. (10), the sign can be chosen arbitrarily. This is because the orientation expressed by the quaternion $(\boldsymbol{\epsilon}, \eta)$ is the same as the orientation expressed by $(-\boldsymbol{\epsilon}, -\eta)$. As seen in Eq. (11), the signs of the elements of $\boldsymbol{\epsilon}$ change according to the sign of η. The appropriate sign should be chosen from the quaternion history. If the present step is considered to be k, the quaternion element with the largest magnitude in step $k-1$ should have the same sign in step k. For there to have been a smooth switch of sign for all the quaternion elements in one time step, the time step has to be extremely large because of the unit-magnitude constraint. If 3 of the quaternion elements pass through 0, one element still has to be approximately 1 or -1. Thus, for reasonably sized time steps and smooth uni-directional motion, only a maximum of 3 quaternion elements may change sign simultaneously.

A second method also exists that has no singularities.[10] In this method, the squares of the vector and scalar parts of the quaternion should first be computed using the following equations:

$$4\epsilon_i^2 = 1 - Tr\mathbf{C} + 2c_{i,i} \tag{12a}$$

$$4\eta^2 = 1 - Tr\mathbf{C} + 2Tr\mathbf{C} \tag{12b}$$

Starting from the quaternion element with the largest square in Eq. (12), the following relations can be used

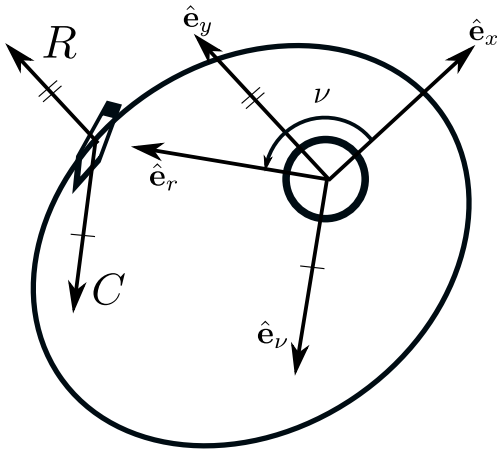


Figure 2. Cartesian and polar frame on the orbital plane

to compute the remaining quaternion elements:

$$4\epsilon_i\epsilon_j = c_{i,j} + c_{j,i} \tag{13a}$$

$$4\epsilon_i\eta = c_{j,k} - c_{k,j} \tag{13b}$$

In Eq. (12) and Eq. (13), the subscripts (i, j, k) are cyclic permutations of $(1, 2, 3)$. By default, the positive square root can be taken and then, the elements can get the appropriate sign based on the quaternion history.

If there are 2 successive rotations, $(\boldsymbol{\epsilon}', \eta')$ followed by $(\boldsymbol{\epsilon}'', \eta'')$, the full rotation, $(\boldsymbol{\epsilon}, \eta)$, can be expressed as

$$\boldsymbol{\epsilon} = \eta''\boldsymbol{\epsilon}' + \eta'\boldsymbol{\epsilon}'' + [\boldsymbol{\epsilon}'\times]\,\boldsymbol{\epsilon}'' \tag{14a}$$

$$\eta = \eta'\eta'' - \boldsymbol{\epsilon}'^T\boldsymbol{\epsilon}'' \tag{14b}$$

Finally, if $\mathcal{F}_b$ is a rotating reference frame with an angular velocity $\boldsymbol{\omega}$ expressed in $\mathcal{F}_b$, the quaternion that expresses its attitude with respect to an inertial frame has the time derivative[9]

$$\begin{bmatrix} \dot{\boldsymbol{\epsilon}} \\ \dot{\eta} \end{bmatrix} = \frac{1}{2}\boldsymbol{\Omega}\begin{bmatrix} \boldsymbol{\epsilon} \\ \eta \end{bmatrix} \tag{15}$$

with $\boldsymbol{\Omega}$ defined as

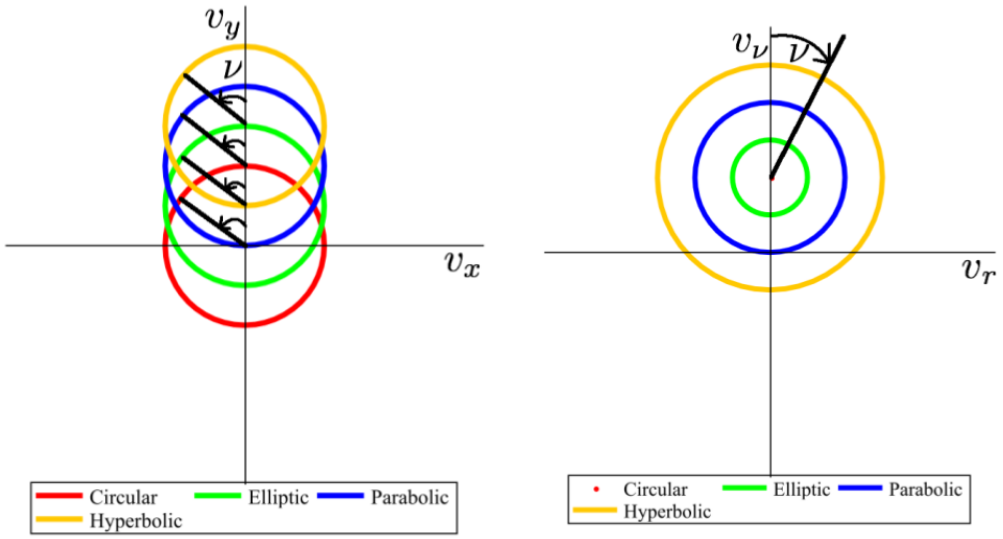
$$\boldsymbol{\Omega} = \begin{bmatrix} -[\boldsymbol{\omega}\times] & \boldsymbol{\omega} \\ \boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{16}$$

C. Hodographs

A hodograph, Greek for path-writing, is also known as a velocity diagram. Hodographs are not used often in contemporary research in Astrodynamics, but there are some excellent technical reports on hodographs from the 1960s.[11–15] There are various methods to derive the hodographs, but only one of the methods is presented here due to its simplicity and elegance.[15] The result is that the velocity at any point in an unperturbed orbit is the sum of a velocity normal to the radial vector laying in the orbital plane, C, and a velocity R, 90° ahead of the eccentricity vector. The eccentricity vector is the vector pointing from the focus of the orbit to the perifocus and is parallel to the unit vector $\hat{\mathbf{e}}_x$ found in Fig. 2. A set of Cartesian and polar reference frames are defined as seen in Fig. 2. In the Cartesian frame, $\hat{\mathbf{e}}_x$ points towards the perifocus of the orbit and $\hat{\mathbf{e}}_r$ points towards the orbiting body. C acts along $\hat{\mathbf{e}}_\nu$ and R acts along $\hat{\mathbf{e}}_y$.

The two velocities have the following magnitudes:[15]

$$C = \mu/h \tag{17}$$



(a) Classical velocity hodograph for various types of orbits

(b) Polar velocity hodograph for various types of orbits

Figure 3. Classical and polar velocity hodographs

$$R = \mu e/h = Ce \tag{18}$$

Expressions for the velocities in the polar and Cartesian frame give rise to circles in both the Cartesian-velocity and the polar-velocity space. The path in the Cartesian-velocity space is known as the classical hodograph and the path in the polar-velocity space is known as the polar hodograph. The equation for the classical hodograph is[15]

$$v_x^2 + (v_y - R)^2 = C^2 \tag{19}$$

where v_x and v_y are the velocity components in the Cartesian reference frame. The classical hodograph is a circle with its center at $(v_x,v_y) = (0,R)$ and radius C. If orbits with constant C are chosen, the hodographs will have the same size, but their centers will translate up along the v_y-axis as the eccentricity increases. Classical hodographs for orbits with various eccentricities, but the same C can be seen in Fig. 3(a).

The equation for the polar hodograph is

$$v_r^2 + (v_\nu - C)^2 = R^2 \tag{20}$$

The polar hodograph is a circle with its center at $(v_r\ ,v_\nu) = (0,C)$ and radius R. If orbits with constant C are chosen, the hodographs will be concentric circles with the radius increasing with increasing eccentricity. The limiting case of circular orbits with $e = 0$ has a point as the polar hodograph. Polar hodographs for orbits with various eccentricities, but the same C can be seen in Fig. 3(b).

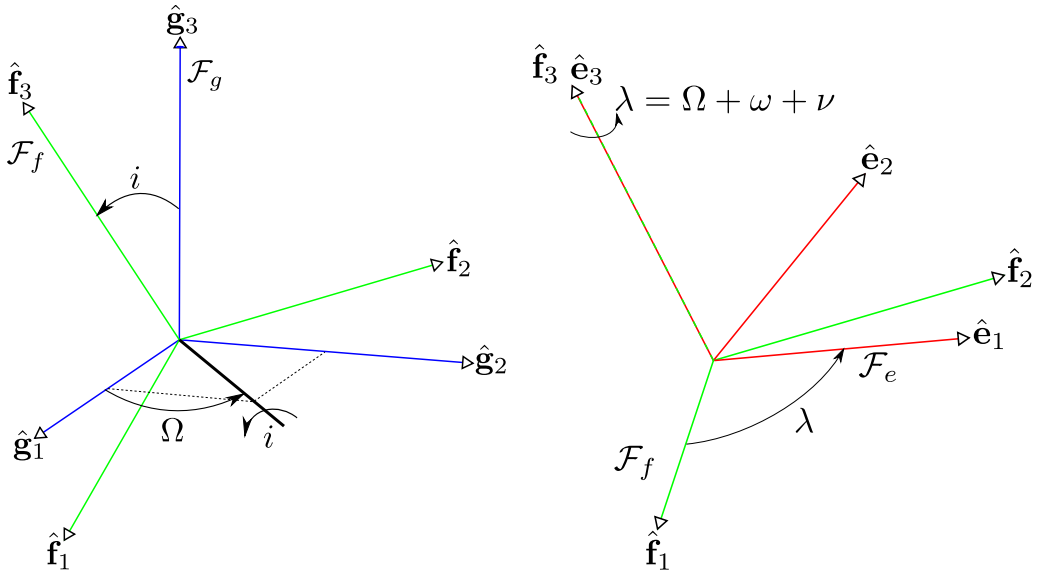
III. The Unified State Model

Using these hodograph theory and quaternions, it is possible to understand the USM. First, the method of formulating the USM state elements and state dynamics is presented, followed by the actual equations and the method of implementation. The USM is explained using the classic Keplerian elements in order to make it more understandable.

A. USM Theory

As mentioned previously, the USM is a 7 parameter set consisting of the four quaternion elements and three hodographic parameters. For the USM, the following three reference frames have to be defined:

$\mathcal{F}_g$ an inertial reference frame fixed on the central body



(a) Orientation of $\mathcal{F}_f$ with respect to $\mathcal{F}_g$ (b) Orientation of $\mathcal{F}_e$ with respect to $\mathcal{F}_f$

Figure 4. The three reference frames used in the USM

$\mathcal{F}_f$ an intermediate rotating reference frame

$\mathcal{F}_e$ a rotating reference frame similar to the polar reference frame found in Fig. 2

The frame $\mathcal{F}_g$ is defined to have the two unit axes $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$ laying in the equatorial plane of the central body, and the third unit axis $\hat{\mathbf{g}}_3$ is perpendicular to the equatorial plane. The frame $\mathcal{F}_e$ is defined to have the unit axis $\hat{\mathbf{e}}_1$ pointing along the radial vector of the orbiting body, $\hat{\mathbf{e}}_3$ laying along the angular momentum vector, and $\hat{\mathbf{e}}_2$ laying the orbital plane and completing the reference frame. Frame $\mathcal{F}_f$ has $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$ laying in the orbital plane, and $\hat{\mathbf{f}}_3$ laying along the angular momentum vector.

The USM state can then be written as[1]

$$\mathbf{x} = \begin{bmatrix} C \\ R_{f1} \\ R_{f2} \\ \epsilon_{O1} \\ \epsilon_{O2} \\ \epsilon_{O3} \\ \eta_O \end{bmatrix} \tag{21}$$

In Eq. (21), the quantities are

- C is a hodographic parameter
- R_{f1} and R_{f2} are the projection of R in $\mathcal{F}_f$
- ϵ_{O1}, ϵ_{O2}, ϵ_{O3}, and η_O are components of a quaternion that describes the orientation of $\mathcal{F}_e$ with respect to $\mathcal{F}_g$

To get from $\mathcal{F}_g$ to $\mathcal{F}_f$, an Euler rotation has to be made around the line of nodes. The line of nodes is the line joining the ascending and descending node of an orbit, with the positive pointing towards the ascending node. The Euler angle is equal to the inclination and the resulting rotation can be seen in Fig. 4(a).

To get from $\mathcal{F}_f$ to $\mathcal{F}_e$, an Euler rotation has to be made around $\hat{\mathbf{f}}_3$. The Euler angle is the true longitude λ, which is the sum of the right ascension of the ascending node, argument of pericenter, and the true anomaly:

$$\lambda = \Omega + \omega + \nu \tag{22}$$

The rotation and the resulting frame can be seen in Fig. 4(b)

The rotations from Figs. 4(a) and 4(b) can be combined into a single rotation and converted into the following quaternion.

$$\begin{bmatrix} \boldsymbol{\epsilon}_O \\ \eta_O \end{bmatrix} = \begin{bmatrix} \sin\frac{i}{2}\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\frac{i}{2}\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\frac{i}{2}\sin\left(\frac{\Omega+u}{2}\right) \\ \cos\frac{i}{2}\cos\left(\frac{\Omega+u}{2}\right) \end{bmatrix} \tag{23}$$

where u is the argument of latitude and is

$$u = \omega + \nu \tag{24}$$

The quaternion in Eq. 23 describes the orientation of $\mathcal{F}_e$ with respect to $\mathcal{F}_g$ and is part of the USM state. It should be noted that for equatorial orbits Ω is not defined, and ω is not defined for circular orbits. The USM is still valid in these cases with the undefined Keplerian elements set to 0.

To be able to use the USM, it is necessary to know how to compute the time derivatives. Unfortunately, the dynamic equations are not as simple to compute as for Cartesian coordinates. The intermediate quantities in Eqs. (25 - 29) have to be computed in the presented order. The quaternion elements depend on λ, so λ can be found after some algebraic manipulation of the quaternion elements.[1]

$$\begin{bmatrix} \sin\lambda \\ \cos\lambda \end{bmatrix} = \frac{1}{\epsilon_{O3}^2 + \eta_O^2} \begin{bmatrix} 2\epsilon_{O3}\eta_O \\ \eta_O^2 - \epsilon_{O3}^2 \end{bmatrix} \tag{25}$$

$$\lambda = \arctan\left(\frac{2\epsilon_{O3}\eta_O}{\eta_O^2 - \epsilon_{O3}^2 q}\right) \tag{26}$$

The velocities expressed in $\mathcal{F}_e$ can be found by projecting the hodographic velocities using λ.[1]

$$\begin{bmatrix} v_{e1} \\ v_{e2} \end{bmatrix} = \begin{bmatrix} 0 \\ C \end{bmatrix} + \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} R_{f1} \\ R_{f2} \end{bmatrix} \tag{27}$$

A quantity required for computing the time derivative of C is simply a relation of C and v_{e2}.[1]

$$p = \frac{C}{v_{e2}} \tag{28}$$

The last required relation is required for expressing the dependancy of R_{f1} and R_{f2} on out-of-plane perturbing accelerations, and is missing from the original model.[4]

$$\gamma = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{29}$$

As input to the dynamic equations of motion, the perturbing accelerations have now to be computed in $\mathcal{F}_e$. It should be noted that the 2-body problem is inherent to the USM. Thus, the accelerations to be computed should not include the acceleration due to the central gravity field. The dynamic equation for the quaternion requires the following angular velocities:[1]

$$\omega_1 = \frac{a_{e3}}{v_{e2}} \tag{30}$$

$$\omega_3 = \frac{C v_{e2}^2}{\mu} \tag{31}$$

The angular velocity $\omega_2 = 0$. Finally, the quaternion time derivative can be computed using the standard formulation:[9]

$$\begin{bmatrix} \dot{\epsilon}_{O1} \\ \dot{\epsilon}_{O2} \\ \dot{\epsilon}_{O3} \\ \dot{\eta}_O \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & 0 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & 0 \\ 0 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & 0 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} \epsilon_{O1} \\ \epsilon_{O2} \\ \epsilon_{O3} \\ \eta_O \end{bmatrix} \tag{32}$$

The time derivative of the hodographic velocity parameters can be computed in the following way:[4]

$$\begin{bmatrix} \dot{C} \\ \dot{R}_{f1} \\ \dot{R}_{f2} \end{bmatrix} = \begin{bmatrix} 0 & -p & 0 \\ \cos\lambda & -(1+p)\sin\lambda & -\gamma R_{f2}/v_{e2} \\ \sin\lambda & (1+p)\cos\lambda & \gamma R_{f1}/v_{e2} \end{bmatrix} \begin{bmatrix} a_{e1} \\ a_{e2} \\ a_{e3} \end{bmatrix} \tag{33}$$

The method of initializing all the USM elements, except for R_{f1} and R_{f2}, has been shown. R must be expressed in $\mathcal{F}_f$ to give R_{f1} and R_{f2}. The vector $\mathbf{R}$ lies in the orbital plane and is 90° ahead of the perifocus; this means that in $\mathcal{F}_e$, $\mathbf{R}$ can be written as

$$\mathbf{R}_e|_{\nu=90^\circ} = \begin{bmatrix} R, & 0, & 0 \end{bmatrix}^T \tag{34}$$

At the location where $\mathbf{R}$ is expressed, $\nu = 90^\circ$ and λ is

$$\lambda|_{\nu=90^\circ} = \Omega + \omega + 90^\circ \tag{35}$$

The rotation matrix $\mathbf{C}_{f,e}$ is the inverse of $\mathbf{C}_{e,f}$ and is

$$\mathbf{C}_{f,e}|_{\nu=90^\circ} = \begin{bmatrix} \cos\lambda & -\sin\lambda & 0 \\ \sin\lambda & \cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{36}$$

$\mathbf{R}$ can be converted to $\mathcal{F}_f$ using the definitions of λ and u.

$$\begin{aligned} \begin{bmatrix} R_{f1} \\ R_{f2} \\ 0 \end{bmatrix} &= \begin{bmatrix} \cos((\Omega+\omega)+90^\circ) & -\sin((\Omega+\omega)+90^\circ) & 0 \\ \sin((\Omega+\omega)+90^\circ) & \cos((\Omega+\omega)+90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}\Bigg|_{\nu=90^\circ} \times \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}\Bigg|_{\nu=90^\circ} \\ &= \begin{bmatrix} -R\sin(\Omega+\omega) \\ R\cos(\Omega+\omega) \\ 0 \end{bmatrix} \end{aligned} \tag{37}$$

B. Reflections on the USM

The USM has some attractive properties that make it suitable for numerical integration. The fact that the Euler parameters have a unit norm makes for a convenient way to check for numerical errors during integration. During numerical integration, normalizing the quaternion is suggested.[5] Three of the seven state elements vary only in the presence of perturbations. This means that during any orbit, only 4 of the 7 state elements vary rapidly. Also, the 4 rapidly varying elements are bounded between -1 and 1. It is also possible to scale the velocity parameters by dividing by C_0, which is the initial value of C. This will ensure that these parameters also stay in the unit region. It is realistic for most simulations that during the orbit C, R_{f1}, and R_{f2} do not increase by an order of magnitude. In case the attitude of the spacecraft is also simulated, the usage of Euler parameters for all 6 degrees of freedom may decrease the computation load.

Even though the Euler parameters are free from singularities, the USM is unfortunately not. There are 2 types of orbits where the USM breaks down. This can be seen in the mathematics whenever the denominator of a fraction is 0.

Case 1 $\epsilon_{O3}^2 + \eta_O^2$

Case 2 h from the definition of C

Case 2 signifies that the motion is rectilinear. This is not the case for normal trajectories in celestial mechanics. This might be the case for hyperbolic orbits when the true anomaly limit is reached. At the true anomaly limit the orbit degenerates into a rectilinear orbit. In the absence of perturbing accelerations, this would cause both ω_1 and ω_2 to become 0. Thus, the Euler parameters and the velocity parameters of the USM would remain constant and the state would be undeterminable, but without a singularity. This is logical because at this point the spacecraft has escaped from the central gravity field, thus violating the

a [km]	e [ ]	i [deg]	Ω [deg]	ω [deg]	ν [deg]
7213	0.01	98.9	269	205	174

Table 1. Keplerian elements of an almost circular orbit

underlying assumption of the USM: motion is in a central gravity field, albeit perturbed. This will not be a problem for orbital work because, using the notion of sphere of influence, the orbit should be out of the sphere of influence of the planet, and in orbit around the Sun. Escapes from the Solar System are not cases that occur very often for artificial spacecraft.

The other singularity occurs when

$$\epsilon_{O3}^2 + \eta_O^2 = 0 \tag{38}$$

Using the condition found in Eq. 38 and the condition of unit norm for Euler parameters results in

$$\epsilon_{O1}^2 + \epsilon_{O2}^2 = 1 \tag{39}$$

This can be deduced by using the definition of the Euler parameters in terms of the Keplerian elements.

$$\epsilon_{O3}^2 + \eta_O^2 = \cos^2\left(\frac{i}{2}\right) = 0 \tag{40}$$

which can subsequently be used to find that i is 180°. The reason for the singularity can be found in the rotation matrix from $\mathcal{F}_g$ to $\mathcal{F}_f$, $\mathbf{C}_{f,g}$.

$$\mathbf{C}_{f,g} = \begin{bmatrix} \cos i + \cos^2\Omega(1-\cos i) & \cos\Omega\sin\Omega(1-\cos i) & -\sin\Omega\sin i \\ \cos\Omega\sin\Omega(1-\cos i) & \cos i + \sin^2\Omega(1-\cos i) & \cos\Omega\sin i \\ \sin\Omega\sin i & -\cos i\sin i & \cos i \end{bmatrix} \tag{41}$$

When the case of $i = 180°$ is filled in, the resulting $\mathbf{C}_{f,g}$ is

$$\mathbf{C}_{f,g} = \begin{bmatrix} 2\cos^2\Omega - 1 & 2\cos\Omega\sin\Omega & 0 \\ 2\cos\Omega\sin\Omega & 2\sin^2\Omega - 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{42}$$

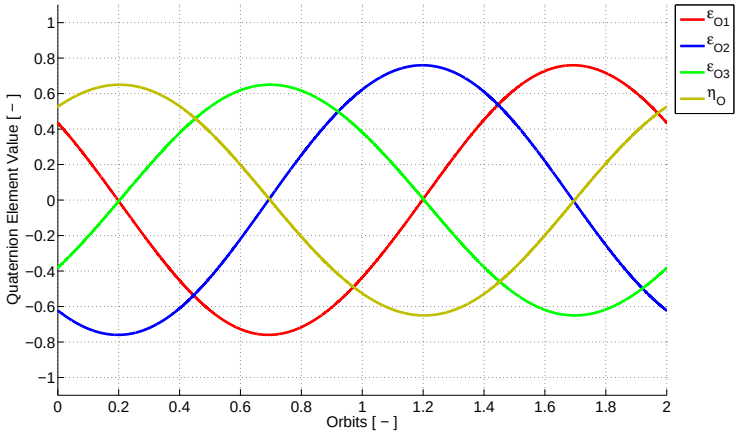
As can be seen in Eq. (42), different values of Ω result in different orientations of $\mathcal{F}_f$. This is obviously not the case in reality; thus, the singularity. This singularity can be removed by defining an alternate $\mathcal{F}_f$ based on a rotation about the descending node.[4] This translates mathematically into a rotation from $\mathcal{F}_g$ to $\mathcal{F}_f$ using the Euler axis and angle

$$\mathbf{a}_1 = \begin{bmatrix} \cos(\Omega+\pi) \\ \sin(\Omega+\pi) \\ 0 \end{bmatrix} = -\begin{bmatrix} \cos\Omega \\ \sin\Omega \\ 0 \end{bmatrix} \tag{43a}$$

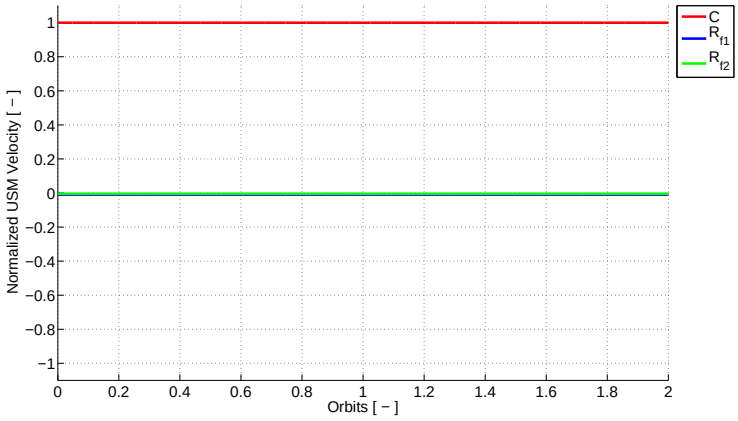
$$\Phi_1 = i \tag{43b}$$

Specialists in astronautics are accustomed to visualizing orbits using classic Keplerian elements. An orbital trajectory is also easy to see when plotted in Cartesian coordinates. There are no available visualizations, in existing literature, of orbits using the USM elements. Therefore, the behavior of the USM elements for an unperturbed orbit, with the initial Keplerian elements found in Table 1 is presented in Fig. 5.

The quaternion elements are bounded by 1, but C, R_{f1}, and R_{f2} are many orders of magnitude larger. To ensure that all the USM elements fit in one plot, the velocity parameters are scaled by using the initial value of C. Two orbital periods are plotted instead of one to show the periodicity of the quaternion elements. Figure 5 shows the evolution of the USM elements over two orbital periods of an almost polar, retrograde, and slightly eccentric orbit. The plots show that after a full orbit, all the quaternion elements have the same magnitude, but the opposite sign. This is correct because when $\nu = 360°$, the rotation is the same as when $\nu = 0°$, but the long way around. For the next orbit, the behavior is the same but with the signs reversed. This is because the quaternion elements use the half angle of the Euler angle and thus, are periodic over 720°. To be very consistent, the sign of the quaternion elements should be switched using some control logic once $\nu > 360°$. This would, however, not influence the results and it would add an unnecessary discontinuity in the ephemeris.



(a) Behavior of the quaternion elements



(b) Behavior of the normalized USM velocities

Figure 5. The behavior of the USM elements over 2 orbits with R_{f1} and R_{f2} overlapping ($C \approx 7434$ m/s, $R_{f1} \approx -68$ m/s, and $R_{f2} \approx -30$ m/s)

IV. Applications and Results

In this section, the USM is applied to five scenarios to compare its performance in numerical integration against the Cartesian coordinates. Two orbit types have been used to test the performance of the USM for perturbed Keplerian orbits without any thrust. An orbit raising maneuver from LEO to GEO has been used to test the performance of the USM under the influence of low-thrust propulsion and perturbations. Also, a trajectory is simulated where there are no perturbing forces on a satellite, except for a continuous low-thrust. Finally, the USM is applied to a solar-sailing scenario.[16]

A. Integrator

All the simulations carried out require a reference trajectory, which cannot be determined analytically for perturbed orbits. Thus, a more accurate integration is used to first generate the so-called truth model. A RK5(4) integrator is then used to find the error with respect to the truth model. A more accurate integration can be obtained by either using a more stringent tolerance or by increasing the order of the integrator. A higher order integrator with more stringent tolerances is used to get the truth model for the solar-sailing scenario. For the other scenarios, however, only the tolerances were made more stringent. Since simulations are normally carried out using variable step-size integrators, results of simulations using fixed step-size integrators are not presented.

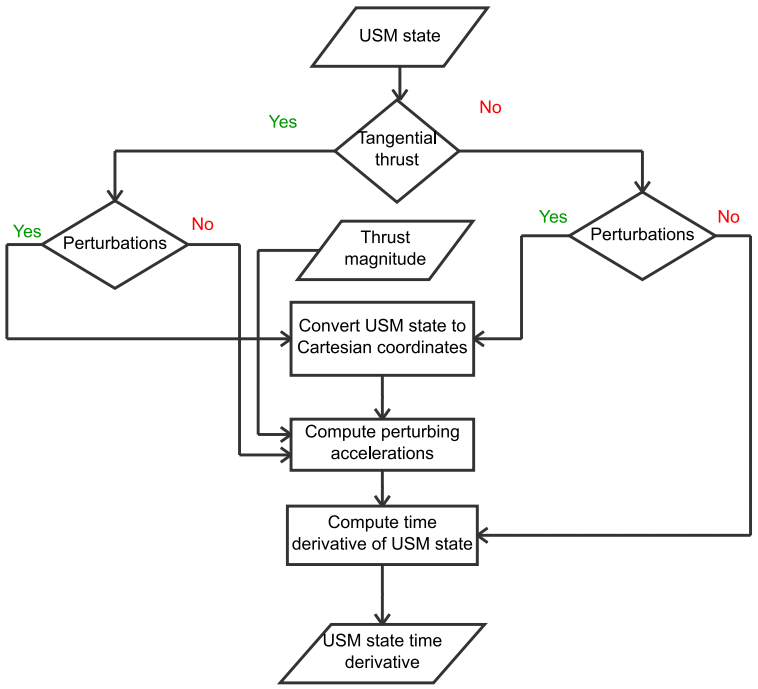


Figure 6. The procedure for a finding the state time derivative when carrying out integration using the USM

It is possible to derive all the various perturbations in terms of the USM,[1] however, users already have perturbations available in the Cartesian form. To ensure that the USM can be easily implemented without much overhead and because the time required to switch between the USM and Cartesian coordinates is very small compared to the computation time required to compute the perturbations, perturbation models expressed in Cartesian coordinates have been used. The perturbations applied are: atmospheric drag, $J_{2,0}$, $J_{2,2}$, solar and lunar 3rd body perturbation, and solar-radiation pressure. These perturbations are the major sources of perturbation for Earth orbiting satellites and the perturbation models can be easily found in literature.[17] The time derivative for USM was computed as shown in Fig. 6. This shows that if perturbations are to be used, the USM state is first converted to Cartesian coordinates prior to computing the perturbations. However, if only low-thrust propulsion is used, there is no need to convert the state to Cartesian coordinates since it is simpler to compute the perturbing accelerations in $\mathcal{F}_e$ than in $\mathcal{F}_g$.

When implementing the dynamics function for the USM, it is important to validate them with respect to an unperturbed analytical, and a perturbed orbit before they can be used to generate trajectories. For Cartesian coordinates the dynamics function is quite simple to implement and choosing an unperturbed, eccentric, and inclined orbit will sufficiently excite all the time derivatives. For the USM however, an unperturbed orbit only causes a change in the quaternion. Unless a perturbed orbit is chosen, $\dot{C}$, $\dot{R}_{f1}$, $\dot{R}_{f2}$, and ω_1 will not be excited. Thus, any bug in the dynamics regarding those elements would not be found.

It is important to note that the tolerances are checked during the USM integration by converting the propagated state to Cartesian coordinates. This way, a consistent time step-size adjuster is used for integration with both the USM and Cartesian coordinates. The variable step-size integrator requires two tolerance values ε_{pos} and ε_{vel}. After an integration step, both the actual 5th-order solution $\mathbf{y}$, and the embedded 4th-order solution $\mathbf{y}_{em}$ are used. If the integration is carried out using the USM, solutions are converted to Cartesian coordinates. The difference between the two vectors in Cartesian coordinates, Δ, is taken and then divided by the present time step-size h:

$$\Delta = \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\mathbf{v} \end{bmatrix} = \frac{\mathbf{y} - \mathbf{y}_{em}}{h} \tag{44}$$

In case $|\Delta\mathbf{r}| \leq \varepsilon_{pos}$ and $|\Delta\mathbf{v}| \leq \varepsilon_{vel}$, the solution is accepted. Otherwise, the integration is repeated using a new time step-size h_{new}. Even if the solution is accepted, h_{new} is computed for the next integration step.

Orbit Type	a [km]	e [-]	i [deg]	Ω [deg]	ω [deg]	ν [deg]
Elliptic	26559	0.70	63.2	206.3	281.6	0.0
Almost Circular	6936	0.00	28.5	194.8	272.3	0.0

Table 2. Initial Keplerian elements of the reference orbits

Two more values are defined to be

$$\delta r = 0.84 \left(\frac{\varepsilon_{pos}}{|\Delta \mathbf{r}|} \right)^{1/4} \tag{45a}$$

$$\delta v = 0.84 \left(\frac{\varepsilon_{vel}}{|\Delta \mathbf{v}|} \right)^{1/4} \tag{45b}$$

The difference δ to be used to find h_{new} is the smallest out of δr and δv. Finally, h_{new} is computed using one of the following three ways: If $\delta \leq 0.1$

$$h_{new} = 0.1 \cdot h \tag{46}$$

if $\delta \geq 5$

$$h_{new} = 5 \cdot h \tag{47}$$

and for all other cases

$$h_{new} = \delta \cdot h \tag{48}$$

B. Orbits without Thrust

An almost circular and an elliptic Molniya type of orbit with $e = 0.7$ are chosen as the reference orbits to be simulated. Trial runs using the USM showed that it has better integration performance and therefore, the truth-model is created using RK5(4) with the USM and $\varepsilon_{pos} = \varepsilon_{vel} = 10^{-9}$. The minimum time step-size was set to 5 s and the maximum time step-size to 250 s. The initial Keplerian elements for the two orbits can be found in Table 2.

Integration was again carried out using the RK5(4) with Cartesian coordinates and the USM for various tolerances. The position and velocity tolerances were set to the same value, $\varepsilon_{pos} = \varepsilon_{vel}$, and the tolerances were made more stringent by an order of magnitude for each simulation from 10^{-1} to 10^{-7}. The maximum and minimum time step-sizes were kept at 5 s and 250 s respectively. The upper limit was reached for the simulations with the least stringent tolerances and the lower limit was reached for the simulation with the most stringent tolerances. The simulations were carried out in MATLAB on a MacBook with a 2.4 GHz Intel Core 2 Duo processor and 2 GB of RAM. The mean error in position between the truth-model and the simulated trajectory is computed for each simulation and plotted against the CPU time as seen in Figs. 7 and 8. The mean error in velocity is not shown because it follows the same trend as the position error and therefore does not provide any additional information. The CPU time was measured using the `tic toc` function available in MATLAB.

The results of variable step-size integration of 100 initial orbital periods for the almost circular LEO can be seen in Fig. 7. During the variable step-size integration, the results of the Cartesian model form a front in the position error - CPU time space that is like a Pareto front used for optimization. Any USM results below or to the left of this front can be considered to perform better than the Cartesian results. As can be seen in Fig. 7, all of the USM results are better than the Cartesian results. For a given mean position error, the CPU time for the USM is approximately one tenth of the CPU time for Cartesian coordinates. Even for the most stringent tolerance, the mean error for the integration using Cartesian coordinates is around 5×10^{-2} with a CPU time of approximately 750 s. This accuracy is achieved by the USM with a CPU time of only 90 s.

The results of variable step-size integration of 50 initial orbit periods for the Molniya type of orbit can be seen in Fig. 8. Again, if a front is created using Cartesian coordinates, all of the USM results are better than the Cartesian results. For a given mean position error, the CPU time for the USM is in the range of a half to a quarter of the CPU time for Cartesian coordinates. Even though the results of the USM are better

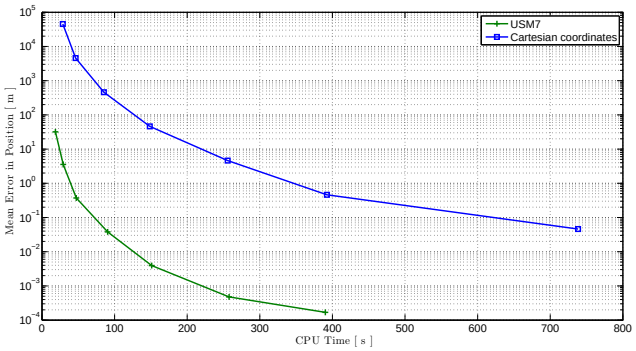


Figure 7. Mean error in position against CPU time for 100 original orbital periods of an almost circular orbit

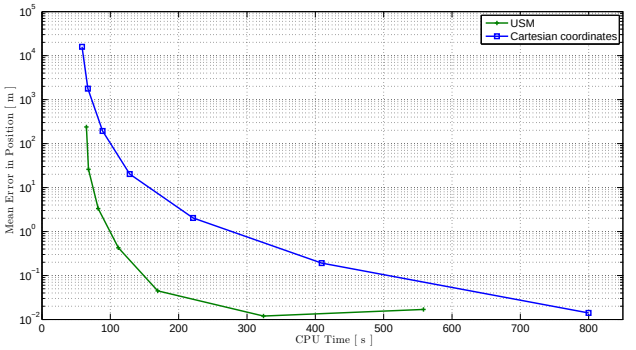


Figure 8. Mean error in position against CPU time for 50 original orbital periods of an eccentric orbit

than the results for Cartesian coordinates, the improvement due to the USM is not as dramatic as for the case of an almost circular orbit. In an eccentric orbit, the majority of the time is spent in the part of the trajectory that is almost rectilinear. In this part, the quaternion elements vary very slowly due to the low value of ω_3. Any small error in the quaternion results in a large error in the position and therefore, it is easy for numerical errors to creep in.

C. Orbit using Low-Thrust Propulsion

Using low-thrust propulsion to change orbital parameters is the focus of much work these days. There are a few analytical solutions for low-thrust orbits such as exponential sinusoids[18] and inverse polynomials.[19] These analytical solutions suffice for a first-order approximation, but to get more accurate results, optimizers have to be used that simulate the whole trajectory. However, low-thrust orbits take a long time to simulate because of the mission duration. Therefore, the USM could be readily applied if the CPU time for simulations decreases. To test the performance of the USM for trajectories as a result of low-thrust propulsion, the following two thrusting scenarios have been utilized:

Scenario 1 A circular parking orbit at an altitude of 838 km is used as the initial orbit and then simulated for 100 original orbit periods with a continuous tangential acceleration of 0.0005 g (0.004905 m/s^2). No other perturbing accelerations are used.

Scenario 2 A circular parking orbit at an altitude of 1000 km is used as the initial orbit. A continuous tangential acceleration of 0.001 g (0.00981 m/s^2) is then applied till the spacecraft reaches GEO altitude. The spacecraft is under the influence of all perturbations and this trajectory is comparable to a low-thrust GTO.

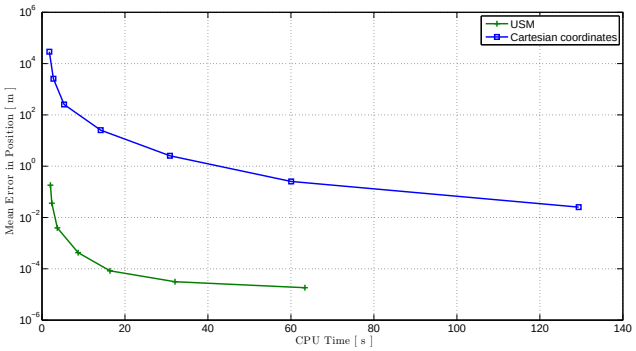


Figure 9. Mean error in position against CPU time for a trajectory with only low-thrust propulsion

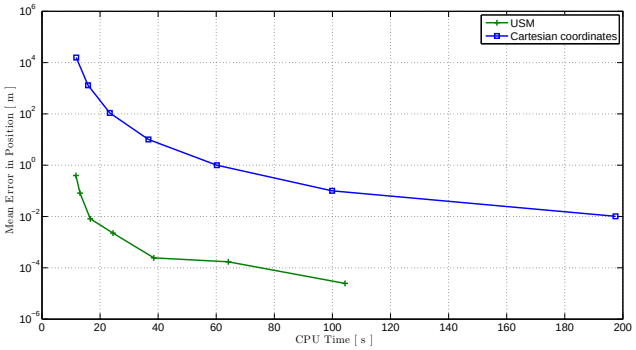


Figure 10. Mean error in position against CPU time for a trajectory with low-thrust propulsion and other perturbations

All the simulations were carried out with the same conditions for the integrators as for the orbits without thrust.

For Scenario 1, no other perturbations were used so that the abilities of the USM to handle low-thrust orbits could be showcased. For the low-thrust case, no conversions have to be made to the Cartesian model to find the perturbations as seen in Fig. 6. The results of the variable step-size integrator for 100 initial orbital periods can be seen in Fig. 9. This shows that the USM is more suited for simulating trajectories under the influence of continuous low-thrust. All the USM integrations are superior to Cartesian coordinates. The mean position error using Cartesian coordinates with $\varepsilon_{pos} = \varepsilon_{vel} = 10^{-7}$ is equal to the mean position error using the USM with $\varepsilon_{pos} = \varepsilon_{vel} = 10^{-2}$. However, the CPU time for the Cartesian coordinates is approximately 26 times larger.

In reality, low-thrust orbits also have other perturbations. Thus, a full test of the USM would be to have a low-thrust orbit with perturbations as found in Scenario 2. The results of the variable step-size integration can be seen in Fig. 10. The results with low-thrust and other perturbations are very similar to the results found for only low-thrust propulsion. Again, the USM performs better than the Cartesian model with the accuracy of the USM being much higher than that of Cartesian coordinates. For a given accuracy, the CPU time of the USM is an order of magnitude less than that of Cartesian coordinates. The mean position error using Cartesian coordinates with $\varepsilon_{pos} = \varepsilon_{vel} = 10^{-7}$ is equal to the ones using the USM with $\varepsilon_{pos} = \varepsilon_{vel} = 10^{-3}$. However, the CPU time for the Cartesian coordinates is approximately 13 times larger.

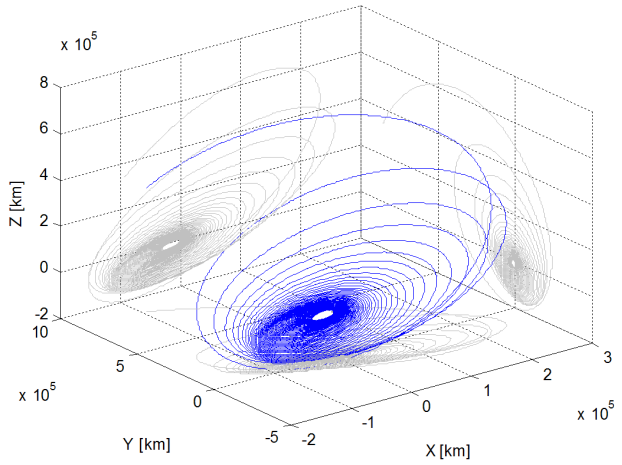


Figure 11. Optimum trajectory in the Geocentric frame

D. Solar Sailing Mission

Optimization of the trajectory of a mission to the Solar poles using Solar sailing has been previously carried out, where Evolutionary Optimization (EO) is used instead of Genetic Algorithms (GA), and a realistic model for the solar sail is used.[16] EO is very similar to optimization using GA because they both follow the Darwinian philosophy of "survival of the fittest". There is, however, one major difference between the two. GA techniques represent the positions in the search space in terms of binary code, while EO uses the actual floating numbers. EO techniques are usually more efficient and produce more realistic results when compared to GA.[20] The objective for the optimizer to minimize is the total mission cost, while still satisfying the constraint of arriving at a solar polar orbit by 2020.

This mission has the spacecraft starting off in a GTO , followed by a spiraling orbit to escape from the Earth using solar propulsion. From the escape from Earth onwards, a heliocentric reference frame is used instead of a geocentric reference frame. This trajectory is simulated until the spacecraft spirals inwards till it is 0.5 AU from the Sun. Finally, there is an heliocentric phase that cranks the orbit and increases the inclination to achieve the desired polar orbit. Simulation of this trajectory is carried out in this work using the USM and compared to Cartesian coordinates to investigate the possible gains due to switching to the USM.

There are many more specific factors that need to be taken into account to fully implement this mission, but these details can be found in ref. 16 and will not be repeated here. They include perturbations, guidance of the sail, modeling of the sail, etc. The final result of the optimization process is a vehicle of 218 kg with a sail area of 6424 m and a travel time of 9.46 years. The launch date is January 2011 and the arrival date is July 2020.[16] The geocentric and heliocentric phases of the optimum trajectory can be seen in Fig. 11 and Fig. 12, respectively.

Throughout the trajectory, the spacecraft is using solar-sailing as a method of low-thrust propulsion. As has been shown so far, integration of trajectories utilizing low-thrust propulsion is faster using the USM than Cartesian coordinates. Since the integration of the trajectory is expected to be faster using USM, the optimization can be carried out faster since numerous individual simulations need to be executed.

To check if the USM is indeed faster, a so-called truth model was created of a trajectory using Cartesian coordinates and an RKF7(8) integrator. This integrator has a relative tolerance of 10^{-9} and a maximum time step-size of 1000 s. This gives a total Time of Flight (TOF) of approximately 3652.5 days, which is approximately 10 years. The same trajectory was again integrated using the same guidance laws, but this time an RKF5(6) integrator was used instead of the RKF7(8). The integration was carried out with Cartesian coordinates and with the USM for various tolerances. This integrator uses an absolute and a relative tolerance like conventional variable step-size integrators.[21] The maximum time step-size was 10^7 s. For Cartesian coordinates, the velocity and position absolute tolerances are set to 0.1 m/s and 0.1 m, respectively. For the the USM, there were two cases used. Both cases had an absolute tolerance of 0.1 m/s

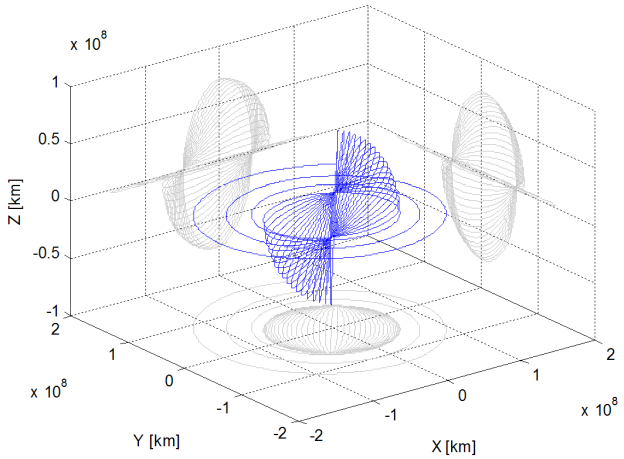


Figure 12. Optimum trajectory in the Heliocentric frame

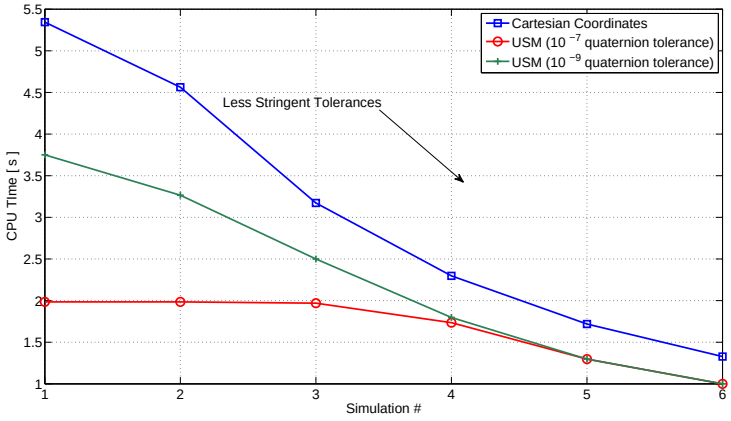


Figure 13. The CPU time for various simulations of a solar-sailing mission

for the hodographic velocity components. The first case of the USM had an absolute tolerance of 10^{-7} for the quaternion elements and the second case of the USM has an absolute tolerance of 10^{-9} for the quaternion elements. For the Cartesian and the two USM cases, the relative tolerance is varied from 10^{-9} to 10^{-4} and the integration is carried out. The CPU time for the various cases can be seen in Fig. 13 and the TOF for the various cases can be seen in Fig. 14 and the TOF mismatch when compared to the TOF of the truth model can be seen in Fig. 15. The simulation of the solar-sailing trajectories is carried out in a FORTRAN-coded model.

In Fig. 13, it can be seen that making the tolerance more stringent increases the CPU time, as expected. For a relative tolerance of 10^{-4}, the CPU time of both the USM cases is approximately 1 s and of the Cartesian case is approximately 1.3 s. As the tolerance becomes more stringent, the CPU times for all cases increase. However, for a tolerance including 10^{-7} and lower, the CPU time for the USM with an absolute tolerance of 10^{-7} for the quaternion elements tapers off at around 2 s. The USM case with an absolute tolerance of 10^{-9} for the quaternion elements results in increasing CPU times as the tolerances get more stringent and a final CPU time of 3.75 s is reached for a relative tolerance of 10^{-9}. The Cartesian coordinates case also requires increasing CPU times as the tolerances get more stringent and a final CPU time of 5.3 s. From this graph, it can be concluded that when a normal step-size controller is used for the USM, the absolute tolerance of the quaternion elements is very important. Making this tolerance more stringent forces

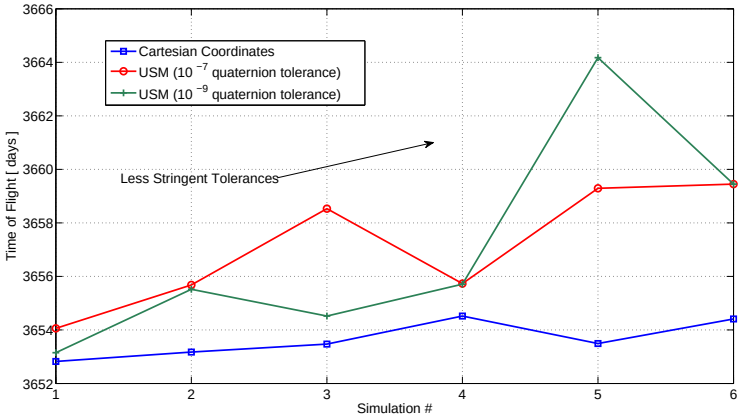


Figure 14. The time of flight for various simulations of a solar-sailing mission

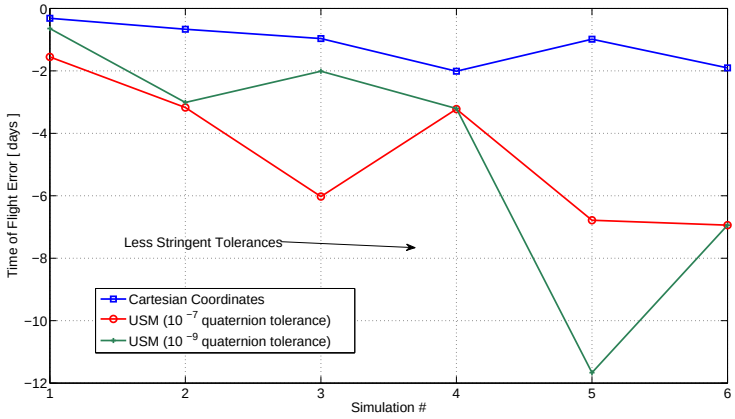


Figure 15. The time of flight mismatch for various simulations of a solar-sailing mission

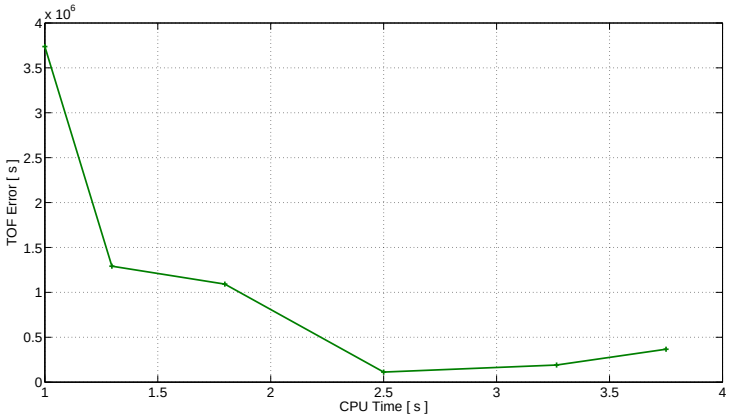


Figure 16. Time of Flight Error against CPU time for integration using the USM with an absolute tolerance of 10^{-9} on the quaternion elements

the integrator to take smaller time steps, especially when more stringent relative tolerances are also used. When the relative tolerance is not very stringent, making the absolute tolerance more stringent does not have any effect. With the most stringent relative tolerance, the Cartesian coordinates case takes 2.65 times the CPU time of the USM case with an absolute quaternion tolerance of 10^{-7}. The Cartesian coordinates case takes 1.41 times the CPU time of the USM case with an absolute quaternion tolerance of 10^{-9}.

The same information can be found in Figs. 14 and 15. All the cases have a larger TOF than the truth model. This, however, does not necessarily imply an error. The integration method checks if the present state has passed the stop criterion, e.g., $i = 90°$. If the present state of the integrator has not passed the stop criterion, the next integration step is made. The maximum time step-size is 10^7 s, which is approximately 116 days. This means that the integration step can have a large time step-size and therefore, the TOF when the integration stops can be much higher than that of the truth model. Once the integration step has been made, it is of course possible to interpolate back to the correct stop criterion location in time. This overshooting is a bigger problem for less stringent tolerances as the time step-sizes will be larger. Also, the overshooting is more for the USM cases than for the Cartesian coordinates, which suggests that larger time step-sizes are being used.

Integration using the USM with an absolute quaternion tolerance of 10^{-9} gives the most accurate results. The final phase of the trajectory involves cranking the orbit to reach an inclination of 90°. The error in the TOF is plotted against the CPU time for the accurate USM simulations by interpolating to reach the correct stop criterion in Fig. 16. Again, the general trend that can be seen is that as the tolerances are made more stringent, the TOF error decreases and the CPU time increases. At the very stringent tolerances, the behavior becomes a bit erratic due to numerical jitter since the relative errors in the TOF are well below 0.1%. For the simulations varying the tolerances of the Cartesian coordinates and the less accurate USM, the behavior of the TOF vs. CPU time is very erratic. This is again due to numerical jitter and is also related to the accuracy of the phase changes in the guidance, which, with large step sizes, can create differences. Despite all these issues, the USM has a tremendous potential for usage in optimizers, where even marginal relative decreases in CPU time can lead to large gains in terms of absolute CPU time.

V. Conclusions and Recommendations

The equations and the theory behind the elegant USM have been presented in this paper. Since none of the previous works dealing with the USM have presented any numerical simulations, the focus has been on showing the performance gains due to using the USM.

It was found that the USM can be used for all the applications for which Cartesian coordinates are used with very few adaptations. For unperturbed orbits, the behavior of the USM is excellent because the orbital energy and the angular momentum are conserved. The USM can also be used for parabolic and hyperbolic trajectories. Hyperbolic trajectories are well represented within the sphere of influence of planets and thus,

the patched conic method can be used for the USM for interplanetary trajectories. Other than the singularity present for pure retrograde orbits, there are no other theoretical scenarios where the USM cannot be used. However, the true anomaly limit for hyperbolic orbits causes a practical limit due to the slow change in the true anomaly near this limit.

For perturbed orbits, the USM performs better than Cartesian coordinates for variable step-size integration. For very small time step-sizes, the USM and Cartesian results are very similar. However, the error of the USM is much lower when larger time step-sizes are used. This can also be seen in the results for the variable step-size integration as the USM performs much better than the Cartesian model with smaller CPU times. The exception to this is an highly eccentric orbit, i.e., $e > 0.9$. In this case, the satellite spends much more time in an almost linear trajectory. Integration in Cartesian coordinates is inherently better suited for this as each state advancement is linear. For many commonly used orbits, however, the USM still performs better than Cartesian coordinates. As shown in this paper, a Molniya orbit, with an eccentricity of around 0.7, still has more accurate results when the USM is used. The scenario where the USM truly shines is low-thrust propulsion. When the thrust is the only perturbing force, the USM7 has position errors that are 5 - 6 orders of magnitude lower than the Cartesian model, when using a variable step-size integrator. When other perturbations are also present, the USM still has errors many orders of magnitude lower than the Cartesian model.

As the USM performs much better than Cartesian coordinates for low eccentricity and continuous thrust orbits, we highly recommend its usage in optimizers. For an optimization problem many different trajectories have to be simulated, which make the computation time very large. Thus, using the USM would help in reducing this CPU load. In particular, low-thrust trajectories with electric propulsion or solar-sailing, as shown in this paper, would be the optimal target group of the USM. Because of the inherent numeric stability, we also recommend the USM for long-term orbit simulation such as orbit debris.

Much work could be carried out on the USM in the future. The implementation of the solar-sailing scenario has to be further refined. Especially, backwards interpolation should be implemented for all the mission phase change criteria to improve the accuracies of the various integration runs. The whole optimization for solar-sailing and other types of low-thrust trajectories should be carried out in the USM search space to investigate if fewer population runs would be required. Each population run is already faster with the USM, thus fewer runs would further increase the gains in CPU time. The navigation performance of the USM could also be investigated. Instead of the Extended Kalman FIlter (EKF), which was used in the previous works to compare the navigation performance of the USM,[3,4] more recent filters can be used, such as the Unscented Kalman Filter[22] (UKF) and the Divided Difference Filter (DDF)[23] to find out if they improve the performance of the USM compared to Cartesian coordinates.

References

[1]S. P. Altman, "A unified state model of orbital trajectory and attitude dynamics," *Celestial Mechanics*, vol. 6, pp. 425–446, 1972.

[2]S. P. Altman, "Velocity-space maps and transforms of tracking observations for orbital trajectory state analysis," *Celestial Mechanics*, vol. 11, pp. 405–428, 1975.

[3]J. R. Raol and N. K. Sinha, "On the orbit determination problem," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-21, pp. 274–291, IEEE, May 1985.

[4]P. Chodas, "Application of the extended kalman filter to several formulations of orbit determination," UTIAS Technical Note 224, University of Toronto Institute for Aerospace Studies, 1981.

[5]T. Fukushima, "Simple, regular, and efficient numerical integration of rotational motion," *The Astronomical Journal*, vol. 135, pp. 2298 – 2322, May 2008.

[6]B. Putney and T. L. Felsentreger, "Geodyn systems development," Technical Memorandum NASA TM-87801, NASA, 1987.

[7]http://www.stk.com

[8]S. Hughes, "General mission analysis tool (gmat). mathematical specifications. draft," tech. rep., NASA Goddard Space Flight Center, 2009.

[9]P. C. Hughes, *Spacecraft Attitude Dynamics*. John Wiley & Sons, Inc., 1986.

[10]W. S. Stanley, "Quaternion from rotation matrix," *AIAA Journal of Guidance and Control*, vol. I, pp. 223–224, May 1978.

[11]F. Sun, "Hodograph analysis of the free-flight trajectories between two arbitrary terminal points," Contractor Report CR-153, NASA, 1965.

[12]S. P. Altman, *Orbital Hodograph Analysis*. AAS science and technology series 3, North Hollywood Western Periodicals, 1965.

[13]S. P. Altman, "Acceleration hodograph analysis techniques for powered orbital trajectories," nasa-cr-61616, NASA, 1967.

[14]S. P. Altman, "Hodographic treatment of the restricted three body problem," Tech. Rep. NASA-CR-61615, NASA, June 1967.

[15]J. B. Eades, "Orbit information derived from its hodograph," Tech. Rep. TM X-63301, NASA, 1968.

[16]E. Mooij, R. Noomen, and S. Candy, "Evolutionary optimization for a solar sailing polar mission," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006.

[17]O. Montenbruck and E. Gill, *Satellite Orbits Models Methods Applications*. Springer, 2005.

[18]A. E. Petropoulos and J. M. Longuski, "Shape-based algorithm for automatd design of low-thrust, gravity-assist trajectories," *Journal of Spacecraft and Rockets*, vol. 41, no. 5, pp. 787–790, 2004.

[19]B. J. Wall and B. A. Conway, "Shape-based approach to low-thrust trajectory design," *Journal of Guidance Control and Dynamics*, vol. 32, no. 1, pp. 95 – 101, 2009.

[20]Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, thrid ed., 1996.

[21]W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C The Art of Scientific Computing*. Cambridge University Press, second ed., 1994.

[22]S. J. Julier and J. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, F*, 1997.

[23]M. Nøgaard, N. K. Poulsen, and O. Ravn, "Advances in derivatice-free state estimation for nonlinear systems," Tech. Rep. IMM-REP-1998-15, Technical University of Denmark, October 2004.