Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

GRAVITYEFFECTOR

Rev	Change Description	By	Date
1.0	First version - Mathematical formulation and implementation	M. Diaz Ramos	2017-01-01
1.1	Added test documentation	S. Carnahan	2017-07-13
1.2	Added documentation on the multi-body gravity acceleration calculation	S. Carnahan	2018-07-20
1.3	Added documentation on the polyhedron model gravity acceleration computation	J.C. Sanchez	2022-07-05

Contents

1 Model Description

The gravity effector module is responsible for calculating the effects of gravity from a body on a spacecraft. A spherical harmonics model and implementation is developed and described below. The iterative methods used for the software algorithms are also described. Finally, the results of the code unit tests are presented and discussed.

1.1 Relative Gravitational Dynamics Formulation

The gravity effector module is critical to the propagation of spacecraft orbits in Basilisk. In order to increase the accuracy of spacecraft trajectories, a relative gravitational acceleration formulation can be used. Relative dynamics keep the acceleration, velocity, and distance magnitudes small, allowing more bits of a double variable to be used for accuracy, rather than magnitude. This additional accuracy is compounded via integration. This relative formulation is enforced when a user sets any planet in a multi-planet environment to have `isCentralBody = True`.

If no planets in a simulation are set as the central body, then an absolute formulation of gravitational acceleration is used. In the absolute formulation, acceleration of a spacecraft due to each massive body is summed directly to calculate the resultant acceleration of the spacecraft.

$$\ddot{\boldsymbol{r}}_{B/N,\text{grav}} = \sum_{i=1}^{n} \ddot{\boldsymbol{r}}_{B/N,i} \tag{1}$$

where the accelerations on the right hand side are the acceleration due to the i^{th} planet which is being modeled as a gravity body. In this absolute mode, spacecraft position and velocity are integrated with respect to the inertial origin, typically solar system barycenter.

In the relative formulation, the acceleration of the spacecraft is calculated *relative to* the central body. This is done by calculating the acceleration of the central body and subtracting it from the acceleration of the spacecraft.

$$\ddot{\boldsymbol{r}}_{B/C,\text{grav}} = \ddot{\boldsymbol{r}}_{B/N,\text{grav}} - \ddot{\boldsymbol{r}}_{C/N,\text{grav}} \tag{2}$$

where C is the central body. In this case, other accelerations of the central body (due to solar radiation pressure, for instance) are ignored. For relative dynamics, the Basilisk dynamics integrator uses only *relative* acceleration to calculate *relative* position and velocity. The gravity module then accounts for this and modifies the spacecraft position and velocity by the central body's position and velocity after each timestep.

The above relative formulation leads to some questions regarding the accuracy of the dynamics integration. First, if acceleration due to gravity is being handled in a relative form, but accelerations due to external foces are handled absolutely, does Basilisk always produce the correct absolute position and velocity? Second, if dynamic state effectors such as hinged rigid bodies are using the gravitational acceleration that the spacecraft receives from the gravity module, are their states being integrated correctly?

Absolute accelerations (i.e. due to thrust) being integrated alongside the relative gravitational acceleration is handled easily due to the linearity of integration. In the absolute dynamics formulation there is:

$$\ddot{\boldsymbol{r}}_{B/N} = \ddot{\boldsymbol{r}}_{B/N,\text{grav}} + \ddot{\boldsymbol{r}}_{B/N,\text{thrust}} + \ddot{\boldsymbol{r}}_{B/N,\text{SRP}} + \dots \tag{3}$$

and each term can be integrated separately on the right side so that

$$\boldsymbol{r}_{B/N} = \int\int \ddot{\boldsymbol{r}}_{B/N,\text{grav}}\text{dtdt} + \int\int \ddot{\boldsymbol{r}}_{B/N,\text{thrust}}\text{dtdt} + \int\int \ddot{\boldsymbol{r}}_{B/N,\text{SRP}}\text{dtdt} + \dots \tag{4}$$

In the derivation that follows, the double integral to position is used, but the logic holds for the first integral to velocity as well. Now, because accelerations also add linearly,

$$\ddot{\boldsymbol{r}}_{B/N} = \ddot{\boldsymbol{r}}_{B/C} + \ddot{\boldsymbol{r}}_{C/N} = \ddot{\boldsymbol{r}}_{B/C,\text{grav}} + \ddot{\boldsymbol{r}}_{C/N,\text{grav}} + \ddot{\boldsymbol{r}}_{B/N,\text{thrust}} + \ddot{\boldsymbol{r}}_{B/N,\text{SRP}} + \dots \tag{5}$$

which differs from Eq. 3 in the gravitational acceleration of the spacecraft being split at the acceleration of the central body. Applying the integrals:

$$\boldsymbol{r}_{B/N} = \boldsymbol{r}_{B/C} + \boldsymbol{r}_{C/N} = \int\int \ddot{\boldsymbol{r}}_{B/C,\text{grav}}\text{dtdt} + \boldsymbol{r}_{C/N} + \int\int \ddot{\boldsymbol{r}}_{B/N,\text{thrust}}\text{dtdt} + \int\int \ddot{\boldsymbol{r}}_{B/N,\text{SRP}}\text{dtdt} + \dots \tag{6}$$

where $\ddot{\boldsymbol{r}}_{C/N}$ is deliberately double integrated to $\boldsymbol{r}_{C/N}$ to show that it can be removed from both sides and $\boldsymbol{r}_{B/C}$ can be evaluated using relative gravitation acceleration combined with absolute accelerations due to external forces:

$$\boldsymbol{r}_{B/C} = \int\int \ddot{\boldsymbol{r}}_{B/C,\text{grav}}\text{dtdt} + \int\int \ddot{\boldsymbol{r}}_{B/N,\text{thrust}}\text{dtdt} + \int\int \ddot{\boldsymbol{r}}_{B/N,\text{SRP}}\text{dtdt} + \dots \tag{7}$$

Once that is done, it is clear that the absolute position can be found by simply adding the position of the central body to the relative position just found:

$$\boldsymbol{r}_{B/N} = \boldsymbol{r}_{B/C} + \boldsymbol{r}_{C/N} \tag{8}$$

This is how absolute position and velocity are found in Basilisk when using a relative dynamics formulation: the relative dynamics are integrated and the position and velocity of the central body are added afterward. The position and velocity of the central body are not integrated by Basilisk, but found from Spice.

Dynamic state effectors connected to the spacecraft hub can use the relative gravitational acceleration in their calculation for much the same reason. Effector positions and velocities are always integrated relative to the spacecraft. In fact, the absolute position and velocity of an effector is rarely, if ever, calculated or used. This is explains why a hinged body experiencing a relative acceleration does not quickly fall behind the spacecraft which is known to be moving along a course experiencing absolute gravitational acceleration. Additionally, because the effector is "pulled along" with the spacecraft when the spacecraft position is modified by the central body position, the effector sees the effect of absolute gravitational acceleration as well.

For intricacies related to using absolute vs relative dynamics, see the user manual at the end of this document.

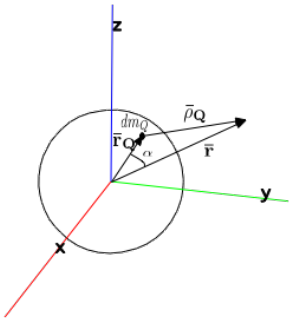

Fig. 1: Geometry of the Spherical Harmonics Representation.

1.2 Spherical harmonics gravity model

Gravity models are usually based on solutions of the Laplace equation ($\nabla^2 U(\bar{\mathbf{r}}) = 0$). It is very important to state that this equation only models a gravity potential outside a body. For computing a potential inside a body the Poisson equation is used instead.

The spherical harmonic potential is a solution of the Laplace equation using orthogonal spherical harmonics. It can be derived solving the Laplace equation in spherical coordinates, using the separation of variables technique and solving a Sturm-Liouville problem. In this work, the solution will be found using another technique, which essentially follows Vallado's book.[5]

For each element of mass dm_Q the potential can be written as

$$\mathrm{d}U(\bar{\mathbf{r}}) = G\frac{\mathrm{d}m_Q}{\rho_Q} \tag{9}$$

where ρ_Q is the distance between the element of mass and the position vector $\bar{\mathbf{r}}$ where the potential is computed. This position vector is usually given in a body-fixed frame. The relation between the position vector $\bar{\mathbf{r}}$, the position of the element of mass $\bar{\mathbf{r}}_Q$ and ρ_Q can be given using the cosine theorem and the angle α between the two position vectors, as can be appreciated in Figure 1.

$$\rho_Q = \sqrt{r^2 + r_Q^2 - 2rr_Q\cos(\alpha)} = r\sqrt{1 - 2\frac{r_Q}{r}\cos(\alpha) + \left(\frac{r_Q}{r}\right)^2} = r\sqrt{1 - 2\gamma\cos(\alpha) + \gamma^2} \tag{10}$$

where $\gamma = r_Q/r$.

The potential can be obtained by integrating dU through the whole body.

$$U(\bar{\mathbf{r}}) = G\int_{body}\frac{\mathrm{d}m_Q}{r\sqrt{1 - 2\gamma\cos(\alpha) + \gamma^2}} \tag{11}$$

If the potential is computed outside the body, γ will always be less than 1, and the inverse of the square root can be approximated using the Legendre polynomials $P_l[\beta]$.[5] Even though this derivation does not use the Laplace equation, it still assumes that the potential is computed outside the body.

The Legendre polynomials can be written as

$$P_l[\beta] = \frac{1}{2^l l!}\frac{d^l}{d\beta^l}(\beta^2 - 1)^l \tag{12}$$

The potential is

$$U(\bar{\mathbf{r}}) = \frac{G}{r}\int_{body}\sum_{l=0}^{\infty}\gamma^l P_l[\cos(\alpha)]\mathrm{d}m_Q \tag{13}$$

The angle α must be integrated. However, the cosine of the angle α can be decomposed using the geocentric latitude and the longitude associated to vectors $\bar{\mathbf{r}}$ and $\bar{\mathbf{r}}_{\mathsf{Q}}$. These angles will be called (ϕ, λ) and $(\phi_{\mathsf{Q}}, \lambda_{\mathsf{Q}})$ respectively. Using the addition theorem it is possible to write.[5]

$$P_l[cos(\alpha)] = P_l[\sin(\phi_{\mathsf{Q}})]P_l[\sin(\phi)] + 2\sum_{m=1}^{l}\frac{(l-m)!}{(l+m)!}(a_{l,m}a'_{l,m} + b_{l,m}b'_{l,m}) \tag{14}$$

where

$$a_{l,m} = P_{l,m}[\sin(\phi_{\mathsf{Q}})]\cos(m\lambda_{\mathsf{Q}}) \tag{15}$$
$$b_{l,m} = P_{l,m}[\sin(\phi_{\mathsf{Q}})]\sin(m\lambda_{\mathsf{Q}}) \tag{16}$$
$$a'_{l,m} = P_{l,m}[\sin(\phi)]\cos(m\lambda) \tag{17}$$
$$b'_{l,m} = P_{l,m}[\sin(\phi)]\sin(m\lambda) \tag{18}$$

where $P_{l,m}[x]$ are the associated Legendre functions. "l" is called degree and "m", order. The polynomials can be computed as

$$P_{l,m}[\beta] = (1-\beta^2)^{\frac{m}{2}}\frac{d^m}{d\beta^m}P_l[\beta] \tag{19}$$

As can be seen, $a_{l,m}$ and $b_{l,m}$ must be integrated, but $a'_{l,m}$ and $a'_{l,m}$ can be taken outside the integral. Therefore, it is possible to define

$$C'_{l,m} = \int_{body}(2-\delta_m)r_{\mathsf{Q}}^l\frac{(l-m)!}{(l+m)!}a_{l,m}\mathrm{d}m_{\mathsf{Q}} \tag{20}$$
$$S'_{l,m} = \int_{body}(2-\delta_m)r_{\mathsf{Q}}^l\frac{(l-m)!}{(l+m)!}b_{l,m}\mathrm{d}m_{\mathsf{Q}} \tag{21}$$

where δ_m is the Kronecker delta.

Then

$$U(\bar{\mathbf{r}}) = \frac{G}{r}\sum_{l=0}^{\infty}C'_{l,0}\frac{P_l[\sin(\phi)]}{r^l} + \frac{G}{r}\sum_{l=0}^{\infty}\sum_{m=1}^{l}\frac{P_{l,m}[\sin(\phi)]}{r^l}\big[C'_{l,m}\cos(m\lambda) + S'_{l,m}\sin(m\lambda)\big] \tag{22}$$

Non-dimensional coefficients $C_{l,m}$ and $S_{l,m}$ are usually used

$$C'_{l,m} = C_{l,m}R_{\mathsf{ref}}^l m_{\mathsf{Q}} \tag{23}$$
$$S'_{l,m} =_{\mathsf{CoM}} S_{l,m}R_{\mathsf{ref}}^l m_{\mathsf{Q}} \tag{24}$$

where m_{Q} is the total mass of the body and R_{ref} is a reference radius. If the coefficients $C_{l,m}$ and $S_{l,m}$ are given, the reference radius must be specified. Usually, the reference is chosen as the maximum radius or the mean radius.[4]

The potential is then

$$U(\bar{\mathbf{r}}) = \frac{\mu}{r}\sum_{l=0}^{\infty}C_{l,0}\left(\frac{R_{\mathsf{ref}}}{r}\right)^l P_l[\sin(\phi)] + \frac{\mu}{r}\sum_{l=0}^{\infty}\sum_{m=1}^{l}\left(\frac{R_{\mathsf{ref}}}{r}\right)^l P_{l,m}[\sin(\phi)]\big[C_{l,m}\cos(m\lambda) + S_{l,m}\sin(m\lambda)\big] \tag{25}$$

Since $P_l[x] = P_{l,0}[x]$ the potential can be written in a more compact way

$$U(\bar{\mathbf{r}}) = \frac{\mu}{r}\sum_{l=0}^{\infty}\sum_{m=0}^{l}\left(\frac{R_{\mathsf{ref}}}{r}\right)^l P_{l,m}[\sin(\phi)]\big[C_{l,m}\cos(m\lambda) + S_{l,m}\sin(m\lambda)\big] \tag{26}$$

Some coefficients have a very interesting interpretation.

$$C_{0,0} = 1 \tag{27}$$
$$S_{l,0} = 0 \quad \forall l \geqslant 0 \tag{28}$$
$$C_{1,0} = \frac{Z_{\text{CoM}}}{R_{\text{ref}}} \tag{29}$$
$$C_{1,1} = \frac{X_{\text{CoM}}}{R_{\text{ref}}} \tag{30}$$
$$S_{1,1} = \frac{Y_{\text{CoM}}}{R_{\text{ref}}} \tag{31}$$

where $[X_{\text{CoM}}, Y_{\text{CoM}}, Z_{\text{CoM}}]$ represents the center of mass of the celestial body. Therefore, if the origin of the coordinate system coincides with the center of mass, all these coefficients are identically zero. Similarly, the second order coefficients are related to the second order moments (moments of inertia).

Finally, the coefficients and Legendre polynomials are usually normalized to avoid computational issues. The factor $N_{l,m}$ is called the normalization factor

$$N_{l,m} = \sqrt{\frac{(l-m)!(2-\delta_m)(2l+1)}{(l+m)!}} \tag{32}$$

The normalized coefficients are

$$\bar{C}_{l,m} = \frac{C_{l,m}}{N_{l,m}} \tag{33}$$
$$\bar{S}_{l,m} = \frac{S_{l,m}}{N_{l,m}} \tag{34}$$

The normalized associated Legendre functions are

$$\bar{P}_{l,m}[x] = P_{l,m}[x] N_{l,m} \tag{35}$$

The potential may be written as

$$U(\bar{\mathbf{r}}) = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=0}^{l} \left(\frac{R_{\text{ref}}}{r}\right)^l \bar{P}_{l,m}[\sin(\phi)]\left[\bar{C}_{l,m}\cos(m\lambda) + \bar{S}_{l,m}\sin(m\lambda)\right] \tag{36}$$

1.2.1 Pines' Representation of Spherical Harmonics Gravity

There are many ways to algorithmically compute the potential and its first and secondary derivatives. One of such algorithms is the one proposed by Pines.[2]

The spherical harmonics representation as it was presented has a singularity at the poles for the gravity field. The Pines' formulation avoids this problem and is more numerically stable for high degree and high order terms.

Unfortunately, this formulation does not contain the normalization factor which is necessary if the coefficients are normalized. In a paper written by Lundberg and Schutz,[1] a normalized representation of the Pines' formulation is given, but it contains an approximation.

For this work, and in order to code the spherical harmonics formulation, a formulation similar to Pines' using the Lundberg-Schutz paper will be derived. However, no approximations will be used. Therefore, the algorithm will be developed here without using the exact formulations given in those

papers. For the sake of brevity, not every single derivation will be carried out, but it is possible to get the results following the expressions obtained in this section.

In the Pines' formulation the radius and the director cosines are used as coordinates. The potential will be given as $U[r, s, t, u]$, where

$$r = \sqrt{x^2 + y^2 + z^2} \tag{37}$$
$$s = \frac{x}{r} \tag{38}$$
$$t = \frac{y}{r} \tag{39}$$
$$u = \frac{z}{r} \tag{40}$$

For a function of these coordinates, the dependance will be given using square brackets (e.g. $f[r, s, t, u]$).

Since $u = \sin(\phi) = \cos(90^\circ - \phi)$, it is possible to write

$$P_{l,m}[\sin(\phi)] = P_{l,m}[u] \tag{41}$$

The derived Legendre functions $A_{l,m}[u]$ are defined such that

$$P_{l,m}[u] = (1 - u^2)^{\frac{m}{2}} A_{l,m}[u] \tag{42}$$

From the definition of $P_{l,m}$ (Equation 19), it is possible to write

$$A_{l,m}[u] = \frac{d^m}{du^m} P_l[u] = \frac{1}{2^l l!} \frac{d^{l+m}}{du^{l+m}} (u^2 - 1)^l \tag{43}$$

The term $(1 - u^2)^{\frac{m}{2}}$ can be written as $(1 - \sin^2(\phi))^{\frac{m}{2}} = |\cos(\phi)|^m = \cos^m(\phi)$.

If the complex number ξ is defined such that (j is the imaginary unit)

$$\xi = \cos(\phi)\cos(\lambda) + j\cos(\phi)\sin(\lambda) = \frac{x}{r} + j\frac{y}{r} = s + jt \tag{44}$$

it is possible to write

$$\xi^m = \cos^m(\phi) e^{jm\lambda} = (s + jt)^m \tag{45}$$

The following sequences may be defined

$$R_m[s, t] = Re\{\xi^m\} \tag{46}$$
$$I_m[s, t] = Im\{\xi^m\} \tag{47}$$

Putting all together, it is possible to write

$$U(\bar{\mathbf{r}}) = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=0}^{l} \left(\frac{R_{\text{ref}}}{r} \right)^l A_{l,m}[u] \{ C_{l,m} R_m[s, t] + S_{l,m} I_m[s, t] \} \tag{48}$$

In order to normalize the coefficients ($\bar{C}_{l,m}$ and $\bar{S}_{l,m}$) and the derived Legendre functions ($\bar{A}_{l,m} = N_{l,m} A_{l,m}$), each term is divided an multiplied by the normalization factor $N_{l,m}$. Then

$$U(\bar{\mathbf{r}}) = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=0}^{l} \left(\frac{R_{\text{ref}}}{r} \right)^l \bar{A}_{l,m}[u] \{ \bar{C}_{l,m} R_m[s, t] + \bar{S}_{l,m} I_m[s, t] \} \tag{49}$$

The sets $D_{l,m}[s,t]$, $E_{l,m}[s,t]$, and $F_{l,m}[s,t]$, are defined as

$$D_{l,m}[s,t] = \bar{C}_{l,m}R_m[s,t] + \bar{S}_{l,m}I_m[s,t] \tag{50}$$
$$E_{l,m}[s,t] = \bar{C}_{l,m}R_{m-1}[s,t] + \bar{S}_{l,m}I_{m-1}[s,t] \tag{51}$$
$$F_{l,m}[s,t] = \bar{S}_{l,m}R_{m-1}[s,t] - \bar{C}_{l,m}I_{m-1}[s,t] \tag{52}$$

The value $\rho_l[r]$ is also defined as

$$\rho_l[r] = \frac{\mu}{r}\left(\frac{R_{\text{ref}}}{r}\right)^l \tag{53}$$

The gravity potential may be finally computed as

$$U(\bar{\mathbf{r}}) = \sum_{l=0}^{\infty}\sum_{m=0}^{l}\rho_l[r]\bar{A}_{l,m}[u]D_{l,m}[s,t] \tag{54}$$

This is the final expression that will be used to compute the gravity potential.

1.2.2 Recursion Formulas

Several recursion formulas are needed in order to algorithmically implement the Pines' formulation. They will be given without proof, but they are easily derived using the definitions above.

- Recursion formula for $\rho_l[r]$

  Initial condition: $\rho_0[r] = \frac{\mu}{r}$

  $$\rho_l[r] = \rho \cdot \rho_{l-1}[r] \tag{55}$$

  where $\rho = R_{\text{ref}}/r$.

- Recursion formula for $R_m[s,t]$

  Initial condition: $R_0[s,t] = 1$

  $$R_m[s,t] = sR_{m-1}[s,t] - tI_{m-1}[s,t] \tag{56}$$

- Recursion formula for $I_m[s,t]$

  Initial condition: $I_0[s,t] = 0$

  $$I_m[s,t] = sI_{m-1}[s,t] + tR_{m-1}[s,t] \tag{57}$$

- Recursion formula for $\bar{A}_{l,m}[u]$

  From Equation (43), it is possible to see that

  $$A_{l,l}[u] = (2l-1)A_{l-1,l-1}[u] \tag{58}$$
  $$A_{l,l-1}[u] = uA_{l,l}[u] \tag{59}$$

There are several recursion formulas for computing Legendre polynomials $A_{l,m}[u]$, for $m < l-1$. The following formula, which is stable for high degrees,[1] will be used:

$$A_{l,m}[u] = \frac{1}{l-m}((2l-1)uA_{l-1,m}[u] - (l+m-1)A_{l-2,m}[u]) \tag{60}$$

Using Equations (58), (59), and (60), and the definition $\bar{A}_{l,m}[u] = N_{l,m}A_{l,m}[u]$, the following recursion formulas can be derived.

Initial condition: $\bar{A}_{0,0}[u] = 1$
The diagonal terms are computed as

$$\bar{A}_{l,l}[u] = \sqrt{\frac{(2l-1)(2-\delta_l)}{(2l)(2-\delta_{l-1})}}\bar{A}_{l-1,l-1}[u] \tag{61}$$

The low diagonal terms are then calculated as

$$\bar{A}_{l,l-1}[u] = u\sqrt{\frac{(2l)(2-\delta_{l-1})}{2-\delta_l}}\bar{A}_{l,l}[u] \tag{62}$$

Finally, for $l \geqslant (m+2)$, $N1_{l,m}$ and $N2_{l,m}$ are defined such that

$$N1_{l,m} = \sqrt{\frac{(2l+1)(2l-1)}{(l-m)(l+m)}} \tag{63}$$

$$N2_{l,m} = \sqrt{\frac{(l+m-1)(2l+1)(l-m-1)}{(l-m)(l+m)(2l-3)}} \tag{64}$$

and $\bar{A}_{l,m}[u]$ computed using

$$\bar{A}_{l,m}[u] = uN1_{l,m}\bar{A}_{l-1,m}[u] - N2_{l,m}\bar{A}_{l-2,m}[u] \tag{65}$$

1.2.3 Derivatives

The first order derivatives of many of the values given are necessary to compute the gravity field (second order derivatives are needed if the Hessian is to be computed).

It is easy to show that

$$\frac{\partial D_{l,m}}{\partial s}[s,t] = mE_{l,m}[s,t] \tag{66}$$

$$\frac{\partial D_{l,m}}{\partial t}[s,t] = mF_{l,m}[s,t] \tag{67}$$

$$\frac{d\rho_l}{dr}[r] = -\frac{(l+1)}{R_{\text{ref}}}\rho_{l+1}[r] \tag{68}$$

$$\frac{\partial R_m}{\partial s}[s,t] = mR_{m-1}[s,t] \tag{69}$$

$$\frac{\partial R_m}{\partial t}[s,t] = -mI_{m-1}[s,t] \tag{70}$$

$$\frac{\partial I_m}{\partial s}[s,t] = mI_{m-1}[s,t] \tag{71}$$

$$\frac{\partial I_m}{\partial t}[s,t] = mR_{m-1}[s,t] \tag{72}$$

$$\frac{d\bar{A}_{l,m}}{du}[u] = \frac{N_{l,m}}{N_{l,m+1}}\bar{A}_{l,m+1}[u] \tag{73}$$

The gravity field can be computed using all the equations given. However, the gradient of the potential is needed. As a change of variables was realized, the chain rule must be applied. In order

to avoid filling up pages with math derivations, the results will be given. With patience, the following results can be obtained applying the chain rule and using all the derivatives given.

The gravity field can be computed as

$$\bar{\mathbf{g}} = (a_1[r,s,t,u] + s\cdot a_4[r,s,t,u])\hat{\mathbf{i}} + (a_2[r,s,t,u] + t\cdot a_4[r,s,t,u])\hat{\mathbf{j}} + (a_3[r,s,t,u] + u\cdot a_4[r,s,t,u])\hat{\mathbf{k}} \tag{74}$$

where

$$a_1[r,s,t,u] = \sum_{l=0}^{\infty}\sum_{m=0}^{l}\frac{\rho_{l+1}[r]}{R_{\text{ref}}} m\bar{A}_{l,m}[u]E_{l,m}[s,t] \tag{75}$$

$$a_2[r,s,t,u] = \sum_{l=0}^{\infty}\sum_{m=0}^{l}\frac{\rho_{l+1}[r]}{R_{\text{ref}}} m\bar{A}_{l,m}[u]F_{l,m}[s,t] \tag{76}$$

$$a_3[r,s,t,u] = \sum_{l=0}^{\infty}\sum_{m=0}^{l}\frac{\rho_{l+1}[r]}{R_{\text{ref}}} m\frac{N_{l,m}}{N_{l,m+1}}\bar{A}_{l,m+1}[u]D_{l,m}[s,t] \tag{77}$$

$$a_4[r,s,t,u] = \sum_{l=0}^{\infty}\sum_{m=0}^{l}\frac{\rho_{l+1}[r]}{R_{\text{ref}}} m\frac{N_{l,m}}{N_{l+1,m+1}}\bar{A}_{l+1,m+1}[u]D_{l,m}[s,t] \tag{78}$$

In order to avoid computing factorials, it is easy to see that

$$\frac{N_{l,m}}{N_{l,m+1}} = \sqrt{\frac{(l-m)(2-\delta_m)(l+m+1)}{2-\delta_{m+1}}} \tag{79}$$

$$\frac{N_{l,m}}{N_{l+1,m+1}} = \sqrt{\frac{(l+m+2)(l+m+1)(2l+1)(2-\delta_m)}{(2l+3)(2-\delta_{m+1})}} \tag{80}$$

Using all these expressions, the potential and the gravity field can be computed.

1.3 Polyhedral gravity model

The polyhedral model, described in Werner and Scheeres publication[6] , computes the exterior gravity of a polyhedron with constant density σ. Let recall that a polyhedron is geometrically described by a number of planar faces composed by vertexes. An edge e is the line connecting two adjacent vertexes belonging to the same face f (the edges follow a counterclockwise concatenation). The potential of such object is

$$U(\bar{\mathbf{r}}) = -\frac{1}{2}G\sigma\sum_{f\in\text{faces}}\left(\mathbf{r}_f\cdot\mathbf{n}_f\mathbf{n}_f\cdot\mathbf{r}_f\omega_f - \sum_{e\in\text{face's edges}}\mathbf{r}_f\cdot\mathbf{n}_f\mathbf{n}_e\cdot\mathbf{r}_e L_e\right) \tag{81}$$

The vector $\mathbf{r}_f$ extends from the evaluation point to any vertex on the face. The variable $\mathbf{n}_f$ is the outward-pointing normal vector of face f. The term ω_f is the signed solid angle subtended by the face f when viewed from the evalution point. The variable $\mathbf{r}_e$ is a vector from the evaluation point to the initial vertex of edge e. The vector $\mathbf{n}_e$ is the normal of the edge lying on the face plane. The term L_e corresponds to the potential of a 1D wire.

Note that in Werner and Scheeres,[6] the double summation term of (81) was simplified to a single summation over all polyhedron's edges. Although that reduction is convenient for mathematical compactness, retaining the double summation simplifies the algorithmic implementation (so that there is no need to check for common edges between adjacent faces).

In order to provide consistency with other gravity models, the density σ is computed based on the polyhedron shape and the input gravity parameter μ. The volume of a trisurface polyhedron is

$$V = \frac{1}{6} \sum_{f \in \text{faces}} |(\mathbf{r}_1^f \times \mathbf{r}_2^f) \cdot \mathbf{r}_3^f|, \tag{82}$$

then $\sigma = \mu/(VG)$. The vector $\mathbf{r}_i^f$, $i = 1, 2, 3$, is the position of each face's vertex.

1.4 Simple Gravity

"Simple Gravity", or gravitational potential and acceleration without taking spherical harmonics into account, is equivalent to using only the $0 \sup th$ term of the spherical harmonics equations. This is the equation that is used in basics physics courses and is most often used in Basilisk simulations. It assumes the gravitational body to be a point mass:

$$U(\bar{\mathbf{r}}) = G\frac{m_{\text{Q}}}{\rho_{\text{Q}}} \tag{83}$$

2 Model Functions

The mathematical description of gravity effects are implemented in gravityEffector.cpp. This code performs the following primary functions

- **GravBody Creation**: The code creates gravity bodies which are capable of affecting spacecraft. It does not effect a spacecraft unless that spacecraft explicitly adds the body as a gravity effector.
- **Orbital Energy**: The code can calculate the total orbital energy as well as orbital kinetic and orbital potential energy of a spacecraft about a gravity body.
- **Simple Gravity**: The code can compute a gravity acceleration between two bodies according to Newton's law of universal gravitation given μ and the distance between the bodies.
- **Spherical Harmonics**: The code can compute gravity acceleration between two bodies using the more-complex method of spherical harmonics. To do this, it must be provided with the same inputs as for calculating simple gravity. In addition, it needs to be provided a "degree" of spherical harmonics to be used and spherical harmonics coefficients useful up to that degree.
- **Polyhedral:** The code can compute gravity acceleration between two bodies using the polyhedral model. To do this, it must be provided with the same inputs as for calculating simple gravity. In addition, it needs to be provided with the vertexes positions and their assignment to faces.
- **Multiple Body Effects**: The code can stack the effects of multiple gravity bodies on top of each other to determine the net effect on a spacecraft. The user must indicate in the spacecraft set-up which gravitational bodies should be taken into account.
- **Interface: Spacecraft States**: The code sends and receives spacecraft state information via the DynParamManager.
- **Interface: Energy Contributions**: The code sends spacecraft energy contributions via updateEnergyContributions() which is called by the spacecraft.
- **Interface: GravBody States**: The code outputs GravBody states(ephemeris information) via the Basilisk messaging system.

3 Model Assumptions and Limitations

3.1 Spherical harmonics gravity model

The limitations of spherical harmonics gravity model are well-known and clearly explained in Schaub and Junkins' book.[3] The limitations include:

- **Coefficient Accuracy**: The coefficients used in the spherical harmonics equations are typically calculated based on gravitational data gathered by satellites. Therefore, the accuracy of the model is determined by the accuracy of the satellite instrumentation and precision of the stored data. Furthermore, for some bodies, there may not be sufficient information available to provide accurate coefficients or higher-degree coefficients.
- **Maximum Degree**: The spherical harmonics equation is a series expansion. Therefore, any implementation must truncate the equation at some point. The truncated portion of the equation necessarily defines some amount of error in the final calculation. This error is, however, small after the first handful of terms. Additionally, a larger distance between gravity body and spacecraft requires fewer terms of the series to achieve equal accuracy as compared to a case with less distance. This code allows the user to request a maximum number of terms to evaluate rather than a specific accuracy. This could lead to less-than-desirable accuracy with small separation distances and greater-than-necessary run times with large separation distances.
- **Planetary Ephemeris Data**: This code generally relies on an external package for planetary ephemeris information. Errors included in this package will translate into error in the gravity calculations, but those errors should be small. Because the ephemeris data is tabulated, this code should not be used to try to project the orbits of the celestial bodies in question. This could be done, though, by treating any celestial body as a "spacecraft".

3.2 Polyhedral gravity model

The limitations of polyhedral gravity model are well-known and clearly explained in Werner and Scheere's article.[6] The limitations include:

- **Constant Density**: The polyhedron gravity computation assumes that the body has constant density. Consequently, this method does not account for spatial density variations that typically arise within the internal structure of bodies or in contact binary asteroids.
- **Shape Accuracy**: The polyhedral model assumes the body shape is described as a polyhedron which is an approximation of the continuous real shape. The resolution of the model can be augmented by increasing the number of vertexes and faces though, in turn, this may considerably slow down the gravity evaluation times. Let recall that the polyhedron gravity computation requires to loop over all faces and edges.
- **Trisurface Polyhedron:** The implemented computation is case-specific for polyhedrons with faces composed of three vertexes. This reduces the possible polyhedrons to a single geometrical topology. However, the trisurface polyhedron is the standardized shape for small bodies.

4 Test Description and Success Criteria

The unit test, test_gravityDynEffector.py, validates the internal aspects of the Basilisk spherical harmonics gravity effector module by comparing module output to expected output. It utilizes spherical harmonics for calulcations given the gravitation parameter for the massive body, a reference radius, and the maximum degree of spherical harmonics to be used. The unit test verifies basic set-up, single-body gravitational acceleration, and multi-body gravitational acceleration.

4.1 Model Set-Up Verification

This test verifies, via three checks, that the model is appropriately initialized when called.

- 1.1 The first check verifies that the normalized coefficient matrix for the spherical harmonics calculations is initialized appropriately as a 3×3 identity matrix.

- 1.2 The second check verifies that the magnitude of the gravity being calculated is reasonable(i.e. between 9.7 and 9.9 m/s^2).

- 1.3 The final check ensures that the maximum degrees value is truly acting as a ceiling on the maximum number of degrees being used in the spherical harmonics algorithms. For example, if the maximum degrees value is set to 20, an attempt is made to call the spherical harmonics algorithms with a degrees value of 100 and another attempt is made with a degrees value of 20. The results are compared and should be equal due to the enforcement of the maximum degrees value.

4.2 Independent Spherical Harmonics Check

This test compares the Basilisk gravity module spherical harmonics acceleration output to an independently formulated python solution. Gravity is measured at an arbitrary point. Note that the independent solution has singularities at the poles that lead to minor divergences in total acceleration.

4.3 Single-Body Gravity Calculations

This test compares calculated gravity values around the Earth with ephemeris data from the Hubble telescope. The simulation begins shortly after 0200 UTC May 1, 2012 and carries on for two hours, evaluating the gravitational acceleration at two second intervals.

4.4 Multi-Body Gravity Calculations

This test checks whether gravity from multiple sources is correctly stacked when applied to a spacecraft. First, a planet is placed in space near a spacecraft. Second, a planet with half the mass of the first is placed the same distance from the spacecraft but in the opposite direction. The gravitational acceleration along that axis is seen to be cut in half for the spacecraft. Finally, a third planet identical to the second is added coincident with the second and the net gravitational acceleration on the spacecraft is seen to be zero.

5 Test Parameters

This section summarizes the specific error tolerances for each test. Error tolerances are determined based on whether the test results comparison should be exact or approximate due to integration or other reasons. Error tolerances for each test are summarized in table 4.

Table 2: Error tolerance for each test. Note that relative tolerance $= \frac{truth - result}{truth}$

Test	Tolerance
Setup Test	1.0e-01 (Absolute)
Independent Spherical Harmonics Check	1.0e-12 (Relative)
Single-Body Gravity	1.0e-04 (Relative)
Multi-Body Gravity	1.0e-12 (Relative for first check, absolute for second)

The Setup Test has a large tolerance, which is acceptable because it is not trying to test exact values but only reasonableness. The Independent Spherical Harmonics check has tight tolerances because the results should be nearly identical for two different formulations of the same mathematics. Single Body Gravity also has relatively large tolerances because it is just trying to roughly match experimental data, but not all real effects are included in the model. Finally, the Multi-Body Gravity test has tight tolerances because it is set up in such a way that the results should be exact down to machine precision.

6 Test Results

All checks within test_gravityDynEffector.py passed as expected. Table 3 shows the test results.

Table 3: Test results.

Test	Pass/Fail	Notes
Setup Test	PASSED	
Independent Spherical Harmonics Check	PASSED	
Single-Body Gravity	PASSED	
Multi-Body Gravity	PASSED	

7 User Guide

This section contains descriptions of how the gravity effector code works and includes descriptions of and notes on variables. It should be helpful to users who wish to use the gravity effector module.

7.1 Code Diagram

The diagram in Fig. 2 demonstrates the basic iterative logic of the gravity effector module. There is extensive additional code that deals with things from the messaging system to transforming the spacecraft position from one frame to another. In general, the inputs shown must be given for each gravity body to be considered. There are, however, convenience functions which add the standard values of these inputs for common celestial bodies. An example is simIncludeGravbody.createEarth() in the test_scenarioOrbitManeuver.py tutorial.

After the inputs are given for each gravity body, the computeGravityInertial() method calculates the 0^{th} degree gravity term for each body and its effects on the spacecraft. If useSphericalHarmParams is True for a given body, then computeField() is called to calculate and add the non-Keplerian term higher degree spherical harmonics terms for that body. If usePolyhedral is True for a given body, then computeField() is called to compute the polyhedron gravity acceleration which overrides the previous 0^{th} degree gravity term (as it is implicitly considered in the polyhedron gravity computation).

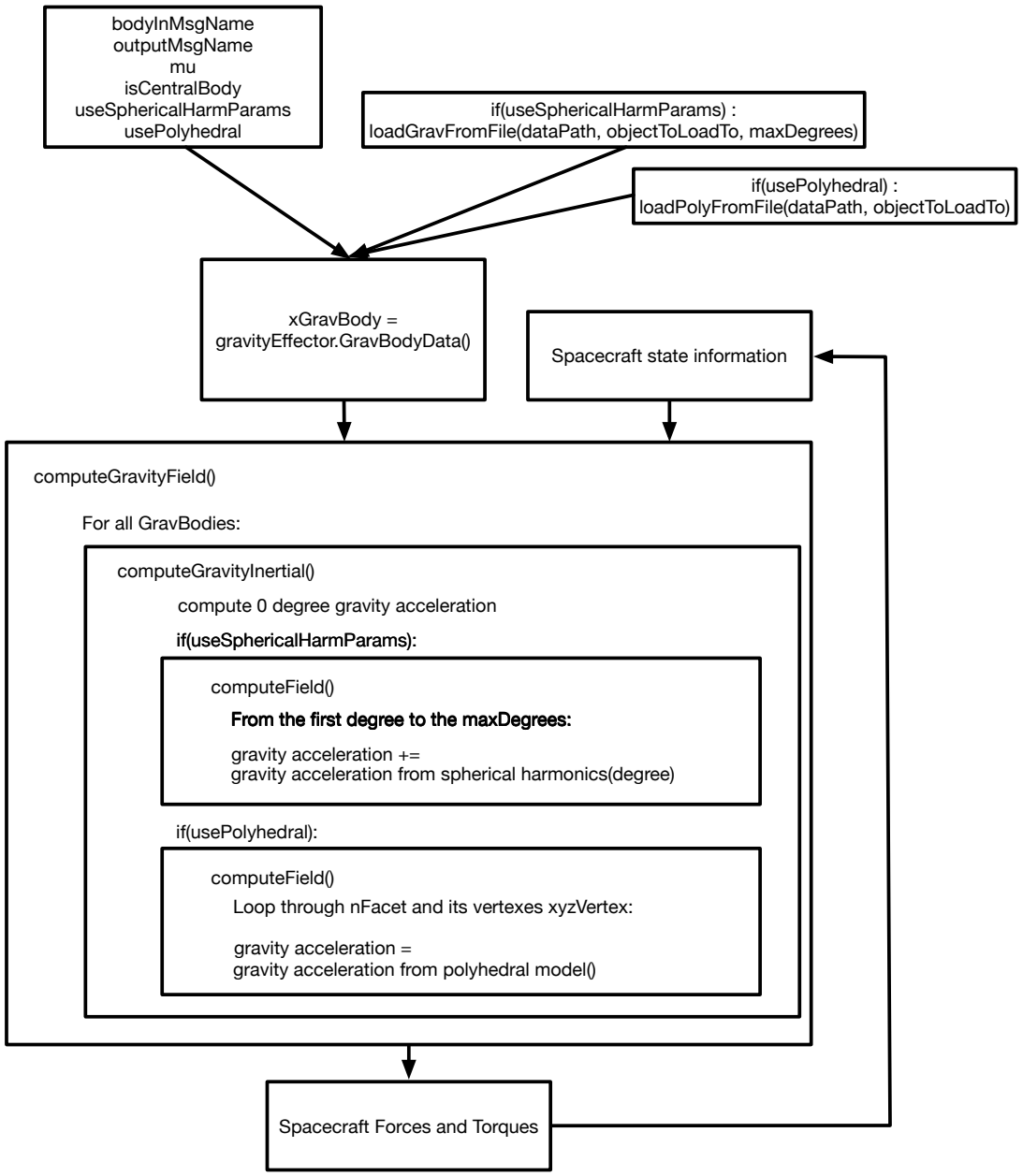

Fig. 2: A pseudo-code diagram demonstrating the flow of inputs and outputs in the gravity effector module.

7.2 Variable Definition and Code Description

The variables in Table 4 are available for user input. Variables used by the module but not available to the user are not mentioned here. Variables with default settings do not necessarily need to be changed by the user, but may be.

Table 4: Definition and Explanation of Variables Used.

Variable	LaTeX Equivalent	Variable Type	Notes
spherHarm.maxDeg	l_{max}	double	Default setting: 0"inertial_state number of degree to use when calculating gravity effects using sperical harmonics.
radEquator	R^l_{ref}	double	[m] Default setting: 0.0. This is the reference radius of the gravity body.
muBody	μ	double	[m3/s2] Default setting: 0.0f. This is the gravitational parameter of the body. Required Input to get any non-zero values out of the code.
isCentralBody	N/A	bool	Default setting: False. Determines whether the body in question is the central body and if initial spacecraft position and velocity are determined as relative or inertial.
isDisplayBody	N/A	bool	Default setting: False. Determines whether the body in question is the focus of the visualization
ephemTime	N/A	double	[s] Default setting: 0. The ephemeris time for the body in question
ephIntTime	N/A	double	[s] Default setting: 0. Required Input. The integration time associated with the ephem data.
planetEphemName	N/A	string	Required Input. An ephemeris name for the planet (user-named).

7.3 Using Central Bodies and Relative Dynamics

7.3.1 Using Central Bodies

In simulations with multiple planetary bodies, using dynamics relative to a central body can improve accuracy. Generally, this is the right thing to do rather than using an absolute coordinate set. If a user has a gravBody called `earth`, the central body flag should be set to True. `earth.isCentralBody = True` The dynamics will then take care of themselves, but the user needs to be careful to input initial position and velocity values as *relative to* the central body. This can be input from a set of Keplerian orbital elements using `orbitalMotion.elem2rv` as in
`Basilisk/tests/scenarios/scenarioBasicOrbit.py`.

The user should be aware that if spacecraft position and velocity are read back from a message log or plotted that the absolute position and velocity will be returned. It will take additional work to convert the outputs back to a relative form by subtracting out the central body positions and velocities. No rotation will be needed, though. It is critical that the relative position and velocities are given in a frame which is linearly translated but **not rotated** from the simulation inertial frame. There is no handling of rotated relative frames within the dynamics. The orbital element to position and velocity conversion in the section below can be used for relative dynamics inputs, as well.

7.3.2 Not Using Central Bodies

If no planets are designated as central bodies, an absolute initial position and velocity must be given. Again, `orbitalMotion.elem2rv` can be used if the orbital elements are given in a frame not rotated from the simulation inertial frame. However, now, the initial position and velocity of the central body must be accounted for. These can be retrieved from spice via the planetStates utility:

```
oe = om.ClassicElements()
oe.a = orbit_a * 1000 #m, orbit semi-major axis
```

```
oe.e = orbit_e #eccentricity
oe.i = radians(orbit_i) #inclination, radians.
oe.Omega = radians(orbit_O) # orbit RAA, radians
oe.omega = radians(orbit_o) #orbit argument of periapsis, radians
oe.f = radians(orbit_f) # orbit true anomaly, radians
r_sc_E, v_sc_E = om.elem2rv(muEarth, oe) #get xyz coordinates from keplerian elements

ephemerides = spice_interface.SpiceInterface()
ephemerides.ModelTag = "SpiceInterfaceData"
ephemerides.SPICEDataPath = splitPath[0] + '../supportData/EphemerisData/'
ephemerides.outputBufferCount = 2
ephemerides.planetNames = spice_interface.StringVector(["earth", "sun"])
ephemerides.UTCCalInit = simStart #pick a UTC string
earthPos_N, earthVel_N = planetStates.planetPositionVelocity('EARTH', simStart)
r_sc_N = array(r_sc_E).flatten() + array(earthPos_N).flatten()
v_sc_N = array(v_sc_E).flatten() + array(earthVel_N).flatten()
scObject.hub.r_CN_NInit = array(r_sc_N)
scObject.hub.v_CN_NInit = array(v_sc_N)
```

Of course, if a user has initial positions and velocities directly, those should be used. See scenarioCentralBody.py for a working example.

7.3.3 Reference Frames

An understanding of spice reference frames will help to explain the code above. The spice inertial frame is the ICRF. The ICRF is coplaner with the Earth's equator. Generally, the Earth Centered Inertial system one would give Keplerian elements in is aligned with ICRF. ICRF is referred to within spice as "j2000" for legacy reasons and because the J2000 system is only rotated from the ICRF by a few milliarcseconds.

7.4 Loading polyhedral shape files

The user has to load polyhedral files in a similar way as spherical harmonics ones. The following code, that loads a polyhedral shape from the file 'EROS856Vert1708Fac.txt', is shown as an example:

```
mu = 4.46275472004*1e5
gravFactory = simIncludeGravBody.gravBodyFactory()
polyBody = gravFactory.createCustomGravObject('eros', mu=mu)
polyBody.usePolyhedral = True
simIncludeGravBody.loadPolyFromFile('EROS856Vert1708Fac.txt', polyBody.poly)
```

7.4.1 Supported polyhedral shape files

The current supported polyhedral shapes files have extensions as '.obj', '.tab' and '.txt'. These describe polyhedral shape models with triangular faces (three vertexes per face). Generally speaking, these files contain the polyhedral vertexes coordinates followed by the composition of each face (in other words, the vertexes that compose a face). It is expected that the vertexes coordinates are stated in kilometers (km). Then, the file reader function `loadPolyFromFile` does the neccesary conversions to meters (m). The vertexes positions are assumed in a body-centered body-fixed reference frame, thus the z coordinate

is aligned with the body rotation axis. The expected format of each one of the supported files is detailed below (for a polyhedral shape of 32002 vertexes and 64000 faces).

The '.obj' file has to follow this format:

```
v -0.477031 0.084442 232.978500
...
v -1.560627 -1.586087 -225.401962
f 10 9 1
...
f 32002 31996 32000
```

The '.tab' file has to follow this format:

```
1 -0.477031 0.084442 232.978500
...
32002 -1.560627 -1.586087 -225.401962
1 10 9 1
...
64000 32002 31996 32000
```

The '.txt' file has to follow this format:

```
32002 64000
-0.477031 0.084442 232.978500
...
-1.560627 -1.586087 -225.401962
10 9 1
...
32002 31996 32000
```

REFERENCES

[1] J. Lundberg and B. Schutz. Recursion formulas of legendre functions for use with nonsingular geopotential models. *Journal of Guidance AIAA*, 11(1):31–38, 1988.

[2] Samuel Pines. Uniform representation of the gravitational potential and its derivatives. *AIAA Journal*, 11(11):1508–1511, 1973.

[3] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series, 3 edition, 2014.

[4] Daniel Scheeres. *Orbital Motion in Strongly Perturbed Environments*. Springer, 1 edition, 2012.

[5] David Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm press, 4 edition, 2013.

[6] R. Werner and D. Scheeres. Exterior gravitation of a polyhedron derived and compared with harmonic and mascon representations of asteroid 4769 castalia. *Celestial Mechanics and Dynamical Astronomy*, 65:313–344, 1996.