# HERCULES User Manual

Version 1.0.1

Simon J. Lock

Last updated: February 14, 2019

# 1 Introduction

HERCULES - short for Highly Eccentric Rotating Concentric $U$ (potential) Layers Equilibrium Structure - is a code designed to find the equilibrium structure of fluid planets, including bodies that are rotating rapidly. This code was originally developed to understand the corotation limit (CoRoL) [Lock and Stewart, 2017] and the origin of the Moon [Lock et al., 2018], but can be used for a wide range of studies where the equilibrium structure of a planet is important.

The code is based on a theoretical framework originally developed for studying the structure of giant planets [Hubbard, 2012, 2013], but extended to overcome the problems of numerical convergence in modeling highly oblate bodies [Hubbard et al., 2014; Kong et al., 2013]. A planet is modeled as a series of concentric, constant-density spheroids. The gravitational potential due to each of these spheroids can be calculated and summed to give the total potential at any point within or without the body. The shape of each of the spheroids is altered iteratively until the surface of each of the spheroids is an equipotential surface. The code can also iterate to conserve the mass of a number of material layers and the total angular momentum of the structure. HERCULES can use realistic equations of state (EOSs) to calculate thermodynamically consistent structures.

HERCULES cannot reach the level of accuracy in the gravitational moments needed for modern orbital mission data, but it has the advantage of being able to deal with many layered structures at a large range of rotation rates. It can also conserve mass and angular momentum, which is needed when modeling planets without an accurate determination of radius. The code is efficient, allowing for rapid exploration of phase space.

This user guide contains the theoretical background for the code (Section 2) and a description of its numerical implementation (Section 3). Sections 4 and 5 describe the structure of the code, Section 6 outlines the input file, Section 7 gives instructions as to how to run the code, and the output files are described in Section 8. Notes on compiling HERCULES are given in Section 9. We also include performance and accuracy tests (Section 10). The standard release of HERCULES includes scripts for the analysis of the output produced by the code and a simple tutorial for first-time users, the details of which are given in Sections 11 and 12. Finally, notes on the different versions of HERCULES, including a list of known issues with the code, are given in Section 13 and a list of published papers using HERCULES is given in Section 14.

## 1.1 Acquiring HERCULES

The most recent version of HERCULES can be acquired through the GitHub repository: `https://github.com/sjl499/HERCULESv1_user`.

### 1.2 Referencing HERCULES

In order to recognize the work in constructing the theoretical framework and in writing the HERCULES code, if you publish results using HERCULES, or a derivative, we ask that you cite the original Lock and Stewart [2017] paper that described the theoretical basis and core implementation of HERCULES:

```
@Article{Lock2017,
   author     =    {S. J. Lock and S. T. Stewart},
   title      =    {The structure of terrestrial bodies:
                     Impact heating, corotation limits, and synestias},
   journal    =    {Journal of Geophysical Research:  Planets},
   year       =    2017,
   volume     =    122,
   issue      =    5,
   pages      =    {950-982},
   doi        =    {10.1002/2016JE005239}
```

### 1.3 Reporting errors

If you find an error in either the code or this manual, please report them to Simon J. Lock (slock@caltech.edu) or via the GitHub repositoryXXXXX. We are keen to make the code and this manual as understandable and usable as possible so if you are confused about anything please get in touch. The email given is correct at the time of writing and will be subject to change. Up to date contact information can be found via a web search, e.g., 'Simon Lock HERCULES'.

### 1.4 Acknowledgments

The code was developed by S. J. Lock while on an NESSF, NASA grant number NNX13AO67H. I would like to thank Bill Hubbard for giving me his concentric Maclaurin spheroid (CMS) code [Hubbard, 2013] and for a useful discussion about the limitations of his method. I also thank Andrew Pepper for proofreading this manual, and Justin Lindeman, Dali Kong and Sarah Stewart for useful discussions.

## 2 Theoretical background

The development of the theoretical basis for models such as HERCULES was largely done in preparation for the interpretation of results from the *Juno* mission. Hubbard [2012] developed a technique for calculating the potential due to a single, constant density, spheroid as an expansion in Legendre polynomials as an alternative to using a small parameter expansion. Later, Hubbard [2013] built on this technique to describe the gravitational potential field from a series of nested concentric spheroids. Hubbard [2013] also devised an iterative procedure to find the equilibrium shape and internal structure of a planet modeled as a series of uniform-density spheroids. This concentric Maclaurin spheroid (CMS) model allowed for the calculation of the gravitational moments of a body to a high degree of precision, sufficient for interpreting the *Juno* mission data.

However, Kong et al. [2013] demonstrated that the Hubbard [2012] model did not correctly calculate the potential for radii between the semi-major and semi-minor axis of an ellipsoid, and the series expansion used by Hubbard [2012] does not converge for planets with $\ell > 1$, where $l$ is defined by

$$\ell^2 = \frac{a^2}{b^2} - 1\,, \tag{1}$$

and $a$ and $b$ are the equatorial and polar radius respectively [Hubbard et al., 2014]. For Jupiter, the error in Hubbard [2012] is not significant as Jupiter is only moderately oblate [Hubbard et al., 2014]. But, for more rapidly rotating planets, the error in Hubbard's method is a significant issue.

We have used the work of Kong et al. [2013] to develop a concentric spheroid model for planets that is stable for high rotation rates. In this section we outline the theoretical framework for our new model. We begin by considering the potential due to a single constant density spheroid. Depending on the radius of the point at which the potential is being calculated relative to the equatorial radius, $a$, and the polar radius, $b$, there are a number of different regimes for calculating the potential and we describe each in turn. We then use our results for a single spheroid to produce a concentric layered planet model that converges for rapidly rotating planets.

We refer to each of the constant-density bodies that make up the planetary structure as spheroids, but the constituent bodies can be of a variety of shapes as long as the radius of the surface of the body monotonically increases from the pole to the equator. It would be possible to generalize the solution to bodies with arbitrary shapes, but more regimes for calculating the potential due to a single, constant-density body would need to be included.

**A note on notation**

The notation used for the concentric spheroid models of planets in the literature is quite confused, with both Hubbard [2012, 2013] and Kong et al. [2013] using different notation. Here we attempt to use the notation from both groups when relevant and logical but have used our own notation when we feel there is a need for greater clarity. In some cases, we use slightly different definitions of some parameters for consistency. We have noted any differences in notation in the relevant sections and a table of relevant variables is given in Table 10.

## 2.1 Potential due to a single highly-oblate body

Here we extend the work of Kong et al. [2013] to consider the potential both within and without an oblate spheroid. The gravitational potential at a point, $\Phi(r, \mu, \phi)$, due to a body is given by

$$\Phi(r, \mu, \phi) = G \int_{\mathcal{V}} \frac{\rho(r', \mu', \phi')\mathrm{d}V'}{|\boldsymbol{r} - \boldsymbol{r'}|}\,, \tag{2}$$

where $(r, \theta, \phi)$ are the standard spherical coordinates: radius, angle from the rotation axis, and angle in the equatorial plane. $\mu = \cos(\theta)$ for $\theta$ between 0 and $\pi/2$. $\boldsymbol{r}$ is the position vector at the point $(r, \mu, \phi)$, $\boldsymbol{r'} = (r', \mu', \phi')$ is the position vector of the mass described by the density distribution $\rho(r', \mu', \phi')$. $G$ is the gravitational constant, $V$ is volume, and $\mathcal{V}$ is the volume in which there is mass. Note that previous studies have typically used $V$ for gravitational potential, but here we use $\Phi$ to avoid confusion with volume. We can expand the denominator in terms of spherical harmonics depending on the relative magnitude of $r$ and $r'$ (Equations 9 and 10 in Kong et al. [2013]). Here we will only consider axisymmetric bodies for which

$$\frac{1}{|\boldsymbol{r} - \boldsymbol{r'}|} = \begin{cases} \displaystyle\sum_{l=0}^{\infty} \frac{(r')^l}{r^{l+1}} P_l(\mu) P_l(\mu') & \text{when } \; r > r' \\ \displaystyle\sum_{l=0}^{\infty} \frac{r^l}{(r')^{l+1}} P_l(\mu) P_l(\mu') & \text{when } \; r < r' \end{cases}\,, \tag{3}$$

where $P_l(\mu)$ is the Legendre polynomial of degree $l$. The expression for the potential therefore changes depending on the position at which we are trying to calculate the potential relative to the body.

Kong et al. [2013] derived expressions for the potential in the exterior of a Maclaurin spheroid of constant density. Here we will consider the potential both inside and outside of a constant density body with a more general shape, the only condition being that the radius of the surface of the body is increasing monotonically from the pole to the equator.

We consider four regimes for calculating the potential: $r \leq b$; $b < r < a$ with the evaluation point within the body; $b < r < a$ with the evaluation point outside the body; and $r \geq a$ (see Figure 1). Note that these regimes are not the same as the domains used by Kong et al. [2013]. We consider each of the regimes for calculating the potential for a body of uniform density, $\rho$, in turn. This body is assumed to be rotationally symmetric and symmetric across the equatorial plane. Additionally, the expressions given here assume that the radius of the surface of the body is monotonically increasing from pole to equator, but there is no requirement for the body to be a Maclaurin spheroid as in Kong et al. [2013].

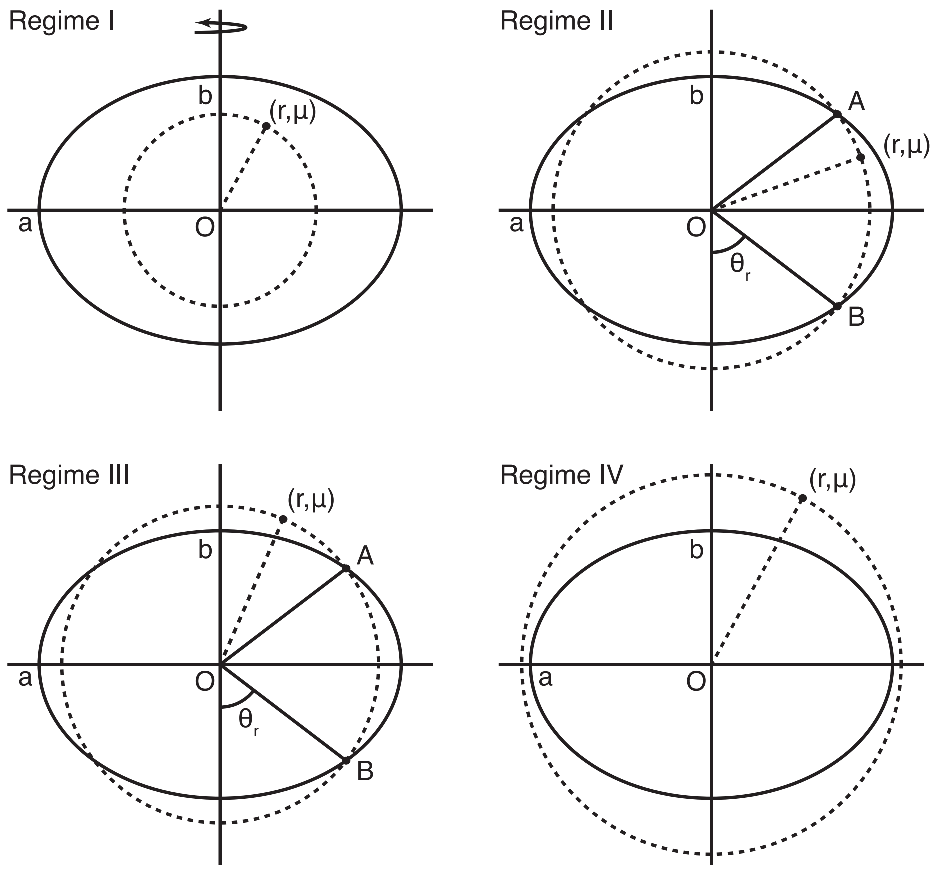

Figure 1: Schematic of the four regimes for calculating the potential of a constant density spheroid, see text for details. Black solid line shows a cross section through the spheroid, with the rotation axis vertical. The equatorial radius, $a$, and polar radius, $b$, are labeled. The point at which the potential is being evaluated, $(r, \mu)$, is marked by the point at the end of the dashed line. The dashed circle indicates the locus of all other points with a radius $r$ from the origin, O. The points on the spheroid where the radius of the surface is equal to $r$, i.e. $\tilde{r}(\mu)$ where $\mu = \pm\mu_r$, are labelled A and B.

### 2.1.1 Regime I: $r \leq b$

The first regime is that for calculating the potential at a point within the spheroid, where the volume interior to the radius $r$ is all within the spheroid. This case was considered by Hubbard [2013] but we rederive it here. In this regime the potential is a sum of two terms: the first due to the mass within a sphere with radius $r$; and the second an integral over the mass in the ellipsoidal shell exterior to $r$. Assuming that the body is axisymmetric and symmetrical relative to the equatorial plane, the potential function can be written as

$$\Phi^I(r,\mu) = \frac{4\pi}{3}\rho G r^2 + 2\pi\rho G \int_{-1}^{1}\int_{r}^{\tilde{r}(\mu')} \left[\sum_{l=0}^{\infty} \frac{r^l}{(r')^{l+1}} P_l(\mu)P_l(\mu')(r')^2 \mathrm{d}r'\right] \mathrm{d}\mu' \,, \tag{4}$$

where $\rho$ is the density of the spheroid and $\tilde{r}(\mu)$ is the radius of the surface of the body at $\mu$. Using the symmetry properties of the integrals, the expression simplifies to

$$\Phi^I(r,\mu) = 4\pi\rho G \left\{\frac{r^2}{3} + \int_{0}^{1}\int_{r}^{\tilde{r}(\mu')} \left[\sum_{k=0}^{\infty} \frac{r^{2k}}{(r')^{2k+1}} P_{2k}(\mu)P_{2k}(\mu')(r')^2 \mathrm{d}r'\right] \mathrm{d}\mu'\right\} . \tag{5}$$

Now we integrate over the radial direction, considering the $k = 0,1$ terms separately

$$\begin{aligned} \Phi^I(r,\mu) = 4\pi\rho G \Bigg\{ &\frac{r^2}{3} + \frac{1}{2}\int_0^1 \left[(\tilde{r}(\mu'))^2 - r^2\right] \mathrm{d}\mu' \\ &+ r^2 P_2(\mu) \int_0^1 \left[\ln\left(\frac{\tilde{r}(\mu')}{r}\right) P_2(\mu')\right] \mathrm{d}\mu' \\ &+ \sum_{k=2}^{\infty} \frac{r^{2k}P_{2k}(\mu)}{2-2k} \int_0^1 \left[(\tilde{r}(\mu'))^{2-2k} - r^{2-2k})P_{2k}(\mu')\right] \mathrm{d}\mu' \Bigg\} . \end{aligned} \tag{6}$$

This is similar in form to the expression found in Kong et al. [2013] for Regimes II and III. Therefore we use notation similar to that of Kong et al. [2013] and rewrite

$$\begin{aligned} \Phi^I(r,\mu) = \frac{4\pi}{3}\rho G r^2 + \frac{GM}{r} \Bigg[ &\left(\frac{r}{a}\right) N_0(\xi,1) - \left(\frac{r}{a}\right)^3 N_2(\xi,1) P_2(\mu) \\ &- \sum_{k=2}^{\infty} \left(\frac{r}{a}\right)^{2k+1} N_{2k}(\xi,1) P_{2k}(\mu) \Bigg] , \end{aligned} \tag{7}$$

where

$$M = \frac{4\pi\rho}{3} \int_0^1 \left[\tilde{r}(\mu')\right]^3 \mathrm{d}\mu' = \frac{4\pi\rho}{3} a^3 \int_0^1 \left[\tilde{\xi}(\mu')\right]^3 \mathrm{d}\mu' \tag{8}$$

is the mass of the spheroid,

$$N_0(\xi, \mu_r) = \left(\frac{3}{2}\right) \frac{\int_0^{\mu_r} \left[(\tilde{\xi}(\mu'))^2 - \xi^2\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}(\mu')\right]^3 \mathrm{d}\mu'}\,, \tag{9}$$

$$N_2(\xi, \mu_r) = -\frac{3\int_0^{\mu_r} \left[\ln\left(\tilde{\xi}(\mu')/\xi\right) P_2(\mu')\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}(\mu')\right]^3 \mathrm{d}\mu'}\,, \tag{10}$$

and

$$N_{2k}(\xi, \mu_r) = -\left(\frac{3}{2k-2}\right) \frac{\int_0^{\mu_r} \left[\left(\xi^{\,2-2k} - (\tilde{\xi}(\mu'))^{\,2-2k}\right) P_{2k}(\mu')\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}(\mu')\right]^3 \mathrm{d}\mu'} \quad \text{for } k \geq 2\,. \tag{11}$$

$\tilde{\xi}(\mu) = \tilde{r}(\mu)/a$ is the normalized radius on the surface of the body, $\xi = r/a$ is the normalized radius at which we are calculating the potential, and $a$ is the equatorial radius of the spheroid. We define $N_2$ with the opposite sign from that used in Kong et al. [2013] to be consistent with the definition of $N_{2k}$. This definition makes the expression for regimes II and III simpler.

The notation used here is similar to Kong et al. [2013], but not exactly the same. Kong et al. [2013] use $\xi_0 = r/a$ (which we denote $\xi$) and use $\xi(\mu)$ to denote the normalized radius at a point on the surface for which we have used $\tilde{\xi}(\mu')$. We feel that the notation used in Kong et al. [2013] is confusing when considering multiple spheroids and so have adopted this new notation for clarity.

### 2.1.2 Regime II: interior point with $b < r < a$

This case is the same as that considered by Kong et al. [2013], but with the point at which the potential is being calculated within the body. However, the location of the evaluation point relative to the surface does not change the expression for the potential in this regime and

$$\begin{aligned}\Phi^{II}(r,\mu) = 2\pi\rho G \int_{-\mu_r}^{+\mu_r} \int_0^r \left[\sum_{l=0}^{\infty} \frac{(r')^l}{r^{l+1}} P_l(\mu) P_l(\mu')(r')^2 \mathrm{d}r'\right] \mathrm{d}\mu' \\ + 2\pi\rho G \int_{-\mu_r}^{+\mu_r} \int_r^{\tilde{r}(\mu')} \left[\sum_{l=0}^{\infty} \frac{r^l}{(r')^{l+1}} P_l(\mu) P_l(\mu')(r')^2 \mathrm{d}r'\right] \mathrm{d}\mu' \\ + 2\pi\rho G \int_{\mu_r}^{1} \int_0^{\tilde{r}(\mu')} \left[\sum_{l=0}^{\infty} \frac{(r')^l}{r^{l+1}} P_l(\mu) P_l(\mu')(r')^2 \mathrm{d}r'\right] \mathrm{d}\mu' \\ + 2\pi\rho G \int_{-1}^{-\mu_r} \int_0^{\tilde{r}(\mu')} \left[\sum_{l=0}^{\infty} \frac{(r')^l}{r^{l+1}} P_l(\mu) P_l(\mu')(r')^2 \mathrm{d}r'\right] \mathrm{d}\mu' \,.\end{aligned} \tag{12}$$

$\mu_r = cos\,\theta_r$ (Figure 1) is the value of $\mu$ for which the observation radius and the surface intersect, i.e. $\tilde{r}(\mu_r) = r$, in the upper hemisphere. This can be integrated to give

$$\begin{aligned}\Phi^{II}(r,\mu) = \frac{GM}{r} \Bigg\{ \left[1 - K_0(\xi,\mu_r) + \left(\frac{r}{a}\right) N_0(\xi,\mu_r)\right] \\ - \sum_{k=1}^{\infty} \left[\left(\frac{r}{a}\right)^{2k+1} N_{2k}(\xi,\mu_r) + \left(\frac{a}{r}\right)^{2k} (J_{2k} - K_{2k}(\xi,\mu_r))\right] P_{2k}(\mu) \Bigg\} \,,\end{aligned} \tag{13}$$

where

$$K_0(\xi,\mu_r) = \frac{\int_0^{\mu_r} \left[(\tilde{\xi}(\mu'))^3 - \xi^3\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}(\mu')\right]^3 \mathrm{d}\mu'} \,, \tag{14}$$

$$K_{2k}(\xi,\mu_r) = -\left(\frac{3}{2k+3}\right) \frac{\int_0^{\mu_r} \left[\left((\tilde{\xi}(\mu'))^{2k+3} - \xi^{2k+3}\right) P_{2k}(\mu')\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}(\mu')\right]^3 \mathrm{d}\mu'} \quad \text{for } k \geq 1 \,, \tag{15}$$

and

$$J_{2k} = -\left(\frac{3}{2k+3}\right) \frac{\int_0^1 \left[(\tilde{\xi}(\mu'))^{2k+3} P_{2k}(\mu')\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}(\mu')\right]^3 \mathrm{d}\mu'} \,. \tag{16}$$

$J_{2k}$ are the gravitational moments and are not dependent on $r$. This is the same expression found by Kong et al. [2013].

In dealing with single Maclaurin spheroids, Kong et al. [2013] could assume that

$$\mu_r = \frac{\sqrt{a^2 - r^2}}{r\sqrt{\left(\frac{a}{b}\right)^2 - 1}} \,. \tag{17}$$

For a general surface this is not the case and we find $\mu_r$ for each potential surface numerically.

#### 2.1.3 Regime III: exterior point with $b < r < a$

This is the exact case that was considered by Kong et al. [2013]. The expression in Equation 13 for the potential applies and $\Phi^{III}(r, \mu) = \Phi^{II}(r, \mu)$.

#### 2.1.4 Regime IV: $r \geq a$

This regime was considered previously by Hubbard [2012] and Kong et al. [2013]. Since all the mass of the spheroid is interior to the evaluation point we need only consider a single term

$$\Phi^{IV}(r, \mu) = 2\pi\rho G \int_{-1}^{1} \int_{0}^{\tilde{r}} \left[ \sum_{l=0}^{\infty} \frac{(r')^l}{r^{l+1}} P_l(\mu) P_l(\mu') (r')^2 \mathrm{d}r' \right] \mathrm{d}\mu' \,. \tag{18}$$

Note that this expression in Equation 13 of Kong et al. [2013] is missing the $2\pi$ prefactor. Integrating Equation 18 radially we get the expression

$$\Phi^{IV}(r, \mu) = \frac{GM}{r} \left[ 1 - \sum_{k=1}^{\infty} \left( \frac{a}{r} \right)^{2k} J_{2k} P_{2k}(\mu) \right] , \tag{19}$$

which is the general expression for the potential outside of an axisymmetric body.

### 2.2 The general concentric Maclaurin spheroid model

Now that we have expressions for the potential due to a single spheroid, we can now build on on the work of Hubbard [2013] and develop a general model for a planet based on concentric Maclaurin spheroids. After Hubbard [2013] we consider modeling a planet as a number of uniform density layers made up of an array of $N_{\text{lay}}$ nested spheroids (Figure 2). The largest spheroid is numbered 0 and labels increase inwards. Layers are defined as the volume between two consecutive spheroids and are numbered by the spheroid that marks their outer surface. The 'real' densities of the layers, $\rho_i$, are given by the sum of the densities of each of the individual spheroids covering that layer. The density of each individual spheroid is therefore given by $\delta\rho_i = \rho_i - \rho_{i-1}$ for $i > 0$ and $\delta\rho_0 = \rho_0$. Conversely

$$\rho_k = \sum_{i \leq k} \delta\rho_i \,. \tag{20}$$

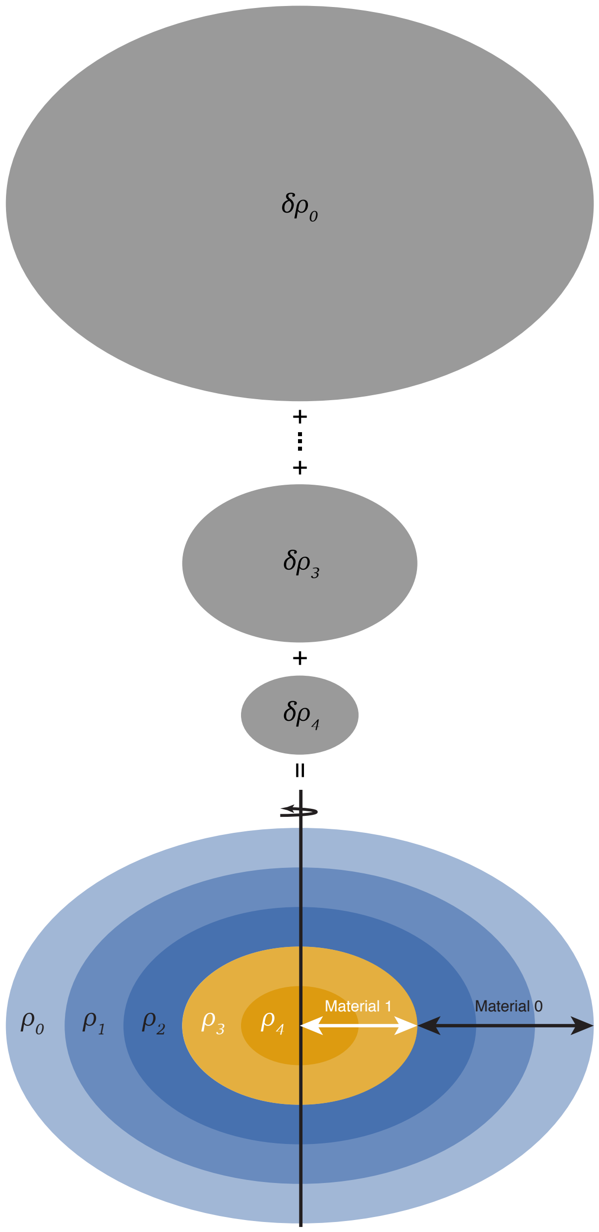

Figure 2: Caption on next page.

Figure 2: Schematic of how an axisymmetric planetary structure is modeled in the HERCULES code. A body is described as a superposition of a number of constant-density spheroids of density $\delta\rho_{\rm i}$ (shown in gray). The superposition of the spheroids gives a body with increasing density with depth. The volumes between successive spheroids are called layers. Each layer has a constant density, $\rho_{\rm i}$, given by the sum of the densities of all the spheroids larger than the spheroid that defines the inner edge of the layer. Each layer belongs to a material layer which determines the relationship between pressure and total density in that layer by use of the material's equation of state. Two material layers are shown separately in blues and oranges.

The gravitational potential at any point is given by summing the gravitational potentials from each of the spheroids, accounting for fact that they are in different regimes:

$$\Phi(r,\mu) = \sum_{i=0}^{i_I} \Phi_i^I(r,\mu) + \sum_{i=i_I+1}^{i_{IV}-1} \Phi_i^{II}(r,\mu) + \sum_{i=i_{IV}^{N_{\rm lay}-1}} \Phi_i^{IV}(r,\mu)\,, \tag{21}$$

where $i_I$ is the lowermost spheroid for which $r < b_i$, $i_{IV}$ is the uppermost spheroid for which $r > a_i$ and $N_{\rm lay}$ is the total number of spheroids. Note that as regime II and III have the same expression for potential, we have combined both into regime II here.

To calculate the equilibrium structure we solve for the position of the surfaces of the constant density spheroids, assuming they are equipotential surfaces. The radius of a point on the surface of the $i$th spheroid, $\tilde{r}_i(\mu)$, is then given by equating the potential at a point on the surface with the potential of that surface at the equator

$$U(\tilde{r}_i(\mu),\mu) - U(a_i,0) = 0\,, \tag{22}$$

where $U = \Phi + Q$ is the total potential. The centrifugal potential, $Q$, for a corotating body is given by

$$Q(r,\mu) = \frac{1}{3} r^2 \omega_{\rm rot}^2 \left[1 - P_2(\mu)\right]\,, \tag{23}$$

where $\omega_{\rm rot}$ is the corotating angular velocity. The total potential $U$ is given by summing the gravitational and centrifugal potentials i.e., $U = \Phi + Q$.

Combining our previous results, the equation that must be solved iteratively to find $\tilde{r}_j(\mu)$ is therefore

$$\Phi(\tilde{r}_j(\mu),\mu) + Q(\tilde{r}_j(\mu),\mu) - \Phi(a_j,0) - Q(a_j,0) = 0\,, \tag{24}$$

with

$$\Phi(r,\mu) = \sum_{i=0}^{i_I} \Phi_i^I(r,\mu) + \sum_{i=i_I+1}^{i_{IV}-1} \Phi_i^{II}(r,\mu) + \sum_{i=i_{IV}}^{N_{\rm lay}-1} \Phi_i^{IV}(r,\mu)\,, \tag{25}$$

$$\Phi_i^I(r,\mu) = \frac{4\pi}{3}\delta\rho_i G r^2 + \frac{GM_i}{r}\left[\left(\frac{r}{a_i}\right) N_{i,0}(\xi_i,1) \; - \sum_{k=1}^{\infty}\left(\frac{r}{a_i}\right)^{2k+1} N_{i,2k}(\xi_i,1) P_{2k}(\mu)\right]\,, \tag{26}$$

$$\begin{aligned}\Phi_i^{II}(r,\mu) = \frac{GM_i}{r}\Bigg\{&\left[1 - K_{i,0}(\xi_i,\mu_{r,i}) + \left(\frac{r}{a_i}\right) N_{i,0}(\xi_i,\mu_{r,i})\right] \\ &- \sum_{k=1}^{\infty}\left[\left(\frac{r}{a_i}\right)^{2k+1} N_{i,2k}(\xi_i,\mu_{r,i}) \; + \left(\frac{a_i}{r}\right)^{2k}\left(J_{i,2k} - K_{i,2k}(\xi_i,\mu_{r,i})\right)\right] P_{2k}(\mu)\Bigg\}\,,\end{aligned} \tag{27}$$

$$\Phi_i^{IV}(r,\mu) = \frac{GM_i}{r}\left[1 - \sum_{k=1}^{\infty}\left(\frac{a_i}{r}\right)^{2k} J_{i,2k}P_{2k}(\mu)\right], \tag{28}$$

$$Q(r,\mu) = \frac{1}{3}r^2\omega_{\rm rot}^2\left[1 - P_2(\mu)\right], \tag{29}$$

$$N_{i,0}(\xi_i,\mu_r) = \left(\frac{3}{2}\right)\frac{\int_0^{\mu_r}\left[(\tilde{\xi}_i(\mu'))^2 - \xi_i^2\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'}, \tag{30}$$

$$N_{i,2}(\xi_i,\mu_r) = -\frac{3\int_0^{\mu_r}\left[\ln\left(\tilde{\xi}_i(\mu')/\xi_i\right)P_2(\mu')\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'}, \tag{31}$$

$$N_{i,2k}(\xi_i,\mu_r) = -\left(\frac{3}{2k-2}\right)\frac{\int_0^{\mu_r}\left[\left(\xi_i^{\,2-2k} - (\tilde{\xi}_i(\mu'))^{\,2-2k}\right)P_{2k}(\mu')\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'} \quad \text{for } k \geq 2, \tag{32}$$

$$K_{i,0}(\xi_i,\mu_r) = \frac{\int_0^{\mu_r}\left[(\tilde{\xi}_i(\mu'))^3 - \xi_i^3\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'}, \tag{33}$$

$$K_{i,2k}(\xi_i,\mu_r) = -\left(\frac{3}{2k+3}\right)\frac{\int_0^{\mu_r}\left[\left((\tilde{\xi}_i(\mu'))^{\,2k+3} - \xi_i^{\,2k+3}\right)P_{2k}(\mu')\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'} \quad \text{for } k \geq 1, \tag{34}$$

$$J_{i,2k} = -\left(\frac{3}{2k+3}\right)\frac{\int_0^1\left[(\tilde{\xi}_i(\mu'))^{\,2k+3}P_{2k}(\mu')\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'}, \tag{35}$$

and

$$M_i = \frac{4\pi\delta\rho_i a_i^3}{3}\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'. \tag{36}$$

A subscript $i$ indicates the quantity was calculated for the $i$th spheroid. We define $\xi_i = r/a_i$, unlike Hubbard [2013], who normalizes all the $\xi$ to the equatorial radius of the outermost spheroid. $\mu_{r,i} = \mu_{r,i}(r)$ is different for each spheroid, $i$, and each point $r$. Also note that the the $J_{i,2k}$ defined here for each spheroid are not the true gravitational moments as in Hubbard [2013], but scaled versions of the moments. For a discussion of this see Section 3.9. These changes have been made to allow the spheroids to remain independent of each other for ease of calculation.

The Equations 24-36 are much more complicated than in Hubbard [2013] however they correctly account for all the different regimes for gravitational potential calculation and non-synchronous rotation. This will allow us to calculate the structure of rapidly rotating and super-CoRoL structures.

## 3 Numeric Implementation

HERCULES iteratively solves for the equilibrium structure of a planet, modeling the body as a series of constant density, nested spheroids. The code implements an algorithm to solve Equation 24 for a series of $N_\mu$ points on the surface of each spheroid, and uses the newly calculated equipotential surfaces to update the structure of the body. To make the iterative equation numerically tractable we truncate the series at a maximum spherical harmonic degree of $2k = 2k_{\max}$.

The code can also iterate to conserve the mass of several material layers and the AM of the body. The material layers (e.g. core, lower mantle, upper mantle etc.) are treated as real materials with equations of state (EOSs) that can be used to give self-consistent densities for each of the concentric layers. Layers are defined as the material that exists in the volume bounded by the surfaces of two consecutive spheroids. A layer is considered as being in a material layer if its outside surface is within or at the edge of that material layer. Layers, spheroids, materials and material layers are numbered from the outside in, starting from 0. A summary of the nomenclature used in describing spheroids, layers, materials and material layers is given in the box below. In this section, we outline the implementation of the iterative scheme in HERCULES and the various other features of the code. HERCULES is written using the C++ programming language.

Note on nomenclature

In this user guide we use a number of different terms to describe features of the planetary structure. Here we attempt to clarify the meaning of each of these.

*structure:*
: We use the term structure to describe the whole body/planet that is being calculated.

*spheroid* or *concentric spheroid:*
: This refers to one of the nested, concentric spheroids of density $\delta\rho_i$ that make up a planet in HERCULES. We use this term to refer to the whole spheroid.

*layer* or *concentric layer:*
: This refers to the volume or mass between the surface of a spheroid and the surface of the spheroid beneath it. The density of each layer is given by the sum of the density of the spheroids that intersect the volume of that layer (Equation 20). Layers are numbered corresponding to the spheroid that describes their outer surface.

*surface* or *outer surface:*
: The surface of a given spheroid.

*material:*
: The property of a layer that controls the relationship between pressure and density in that layer. For example, it could control the EOS used to calculate the density of that layer from the calculated pressure.

*material layer:*
: A material layer is the volume of the planet that is composed of a given material. The 'real' density of the layers in this region are calculated in the same way, typically using the EOS of the relevant material using the calculated pressure. A layer is considered to be part of a specific material layer if its outer surface lies within, or at the top of, the material layer. Spheroids are not attached to a given material The actual mass of a given material has contributions from spheroids that are not considered part to be of that material layer as they have a volume that intersects the material layer.

## 3.1 Overview of algorithm

Finding the equilibrium planetary structure in HERCULES is divided into a number of steps. Here, we outline the steps to provide a structure for discussing the different routines discussed below.

1. **Initialization:** This section reads in the input files and initializes the structure before the iteration routine begins. The root for this section is the `initialisation` function.

   i. **Read input:** The input file is read in and from it the lengths of various vectors are defined. This is done in the `read_input` function. This function also defines the material layers and reads in the EOS files. For a discussion of the input file see Section 6.

   ii. **Initialize the structure:** Next, an initial approximation of the planet's structure is made. This can be done assuming a single density for all spheroids, a constant density body or by reading in a binary output file (see Section 3.2).

   iii. **Calculate structure properties:** Calculate various properties of the structure. This includes: the precomputed integrals (see Section 3.8); the mass, AM, gravitational moments and moments of inertia for each spheroid (`concentric_layer.calc_M`, `planet.calc_Mtot`, `planet.calc_Ltot`, `concentric_layer.calc_Js` and `concentric_layer.calc_I`); and the potential at the equator of each spheroid. With the exception of the latter, the calculation of these properties is done in the routine for initializing the structure in Section 1ii. The initial structure is output to a binary file.

2. **Iteration:** The main body of the code is a single iteration to find the equilibrium structure with two nested iterations for calculating the shape of each of the spheroids and controlling the mass and AM of the body. A schematic of the iteration is given in Figure 3 and we will describe the main steps below. The root for the iteration section of the code is the function `iterate`.

   i. **Shape iteration:** The shape of the equipotential surfaces, corresponding to the potential at the equator of each of the spheroids from the previous iteration/initialization, is found. A Newton-Raphson method is used to find the $\xi_i$ point corresponding to the solution for Equation 24 at each $\mu_i$ point on each surface (see Section 3.3). The gradient of the function given in Equation 24 is determined numerically, calculating the potential at $\xi$ points separated by $\delta\xi$ to calculate the gradient of the function given in . This process is repeated for all the surface points. The structure of the body is kept constant throughout this process and only updated in the next step.

   ii. **Update planetary structure:** The structure of the planet is changed so that the surface of the spheroids correspond to the equipotential surfaces calculated in the shape iteration. By definition, changing the shape of the spheroids changes the equipotential surfaces, but over several iterations the shape of the structure will converge. The structure-specific precomputed integrals are recalculated (Section 3.8) along with the mass, AM, gravitational moments and moments of inertia for each spheroid and the total potential at the equator of each spheroid.

   iii. **Mass and AM control iteration:** Routines for controlling the mass and AM are applied (see Sections 3.5 and 3.6). The AM control routine is performed after the mass control routine. After each, if properties of the structure have changed, the relevant properties and precomputed integrals are recalculated. The routines are looped over until a convergence criteria is met (see Section 3.7). Both control routines are performed in each iteration.

   iv. **Access convergence:** The final step is to access whether the convergence criterion for the whole iteration has been met (see Section 3.7). The previous three steps are repeated until either the convergence criteria are met or a maximum number of iterations is reached.

3. **Output:** The final structure is output to a binary file. A file containing the initial structure is also output at the end of the initialization step, and additional files can be output at the end of each full iteration.

## 3.2 Initialization

HERCULES allows for two different ways of initializing a structure, either assuming a constant density for all spheroids or from a HERCULES binary output file. The choice of initialization procedure is made by the use of the start flag, $\mathcal{F}_{\text{start}}$, in the input file (see Section 6).

### 3.2.1 Constant density spheroids: $\mathcal{F}_{\text{start}} = 0$

This procedure (in `planet.initialise_ellipsoids` called from `initialisation`) initializes the structure as nested Maclaurin spheroids. The surfaces are equally spaced in equatorial radius within each material layer. The number of layers in each material, $N_{\text{lay}}^{\text{mat}}$, and the initial equatorial radii for each material layer, $a_{\text{mat}}$, are given in the input file. The shape of the spheroids is dictated by the initial aspect ratio, $\mathcal{A} = b/a$, given in the input file. The surface points on each ellipsoid are given by

$$\xi_i(\mu_j)^2 = \frac{b_i^2}{(a_i\mu_j)^2 + b_i^2\,(1-\mu_j)^2}\,. \tag{37}$$

The density of the spheroids is set depending on the mass control flag. For constant density layers ($\mathcal{F}_{\text{Mconc}} = 0$) the density of the layers is set to the reference density, $\rho_{\text{ref}}$, given in the input file. For all other routines, the density is calculated so that each material layer has the desired mass, with each spheroid in each material layer having the same density. The mass in a layer is given by the sum of the mass of all the spheroids covering the region with that material:

$$M_k^{\text{mat}} = \sum_{i<k}^{\text{material}} \delta\rho_i \sum_{j:i}^{\text{layers}} (V_{\text{out}}^k - V_{\text{in}}^k) + \delta\rho_k \sum_{i:k}^{\text{layers}} (V_i - V_{in}^k)\,, \tag{38}$$

where $M_k^{\text{mat}}$ is the mass of material $k$, $V_{\text{out}}^k$ and $V_{\text{in}}^k$ are the volume of the outside and inside layers of material $k$, and $\delta\rho_i$ is the constant density for spheroids in material $i$. $i : k$ indicates the layers that are of material $k$. This can be rearranged to give the density of spheroids in material layer $k$

$$\delta\rho_k = \frac{M_k^{\text{mat}} - \sum_{i<k}^{\text{material}} \delta\rho_i \sum_{j:i}^{\text{layers}} (V_{\text{out}}^k - V_{\text{in}}^k)}{\sum_{i:k}^{\text{layers}} (V_i - V_{\text{in}}^k)}\,. \tag{39}$$

The density for spheroids in each material layer can then be found, working from the outermost material inwards.

This routine also initializes a constant rotation for each layer. If $\mathcal{F}_{\text{Lconc}} = 0$ then the rotation rate is set to be that given in the input file; otherwise a constant rotation rate that satisfies AM conservation is set. This can be later overruled depending on the AM control routine used.

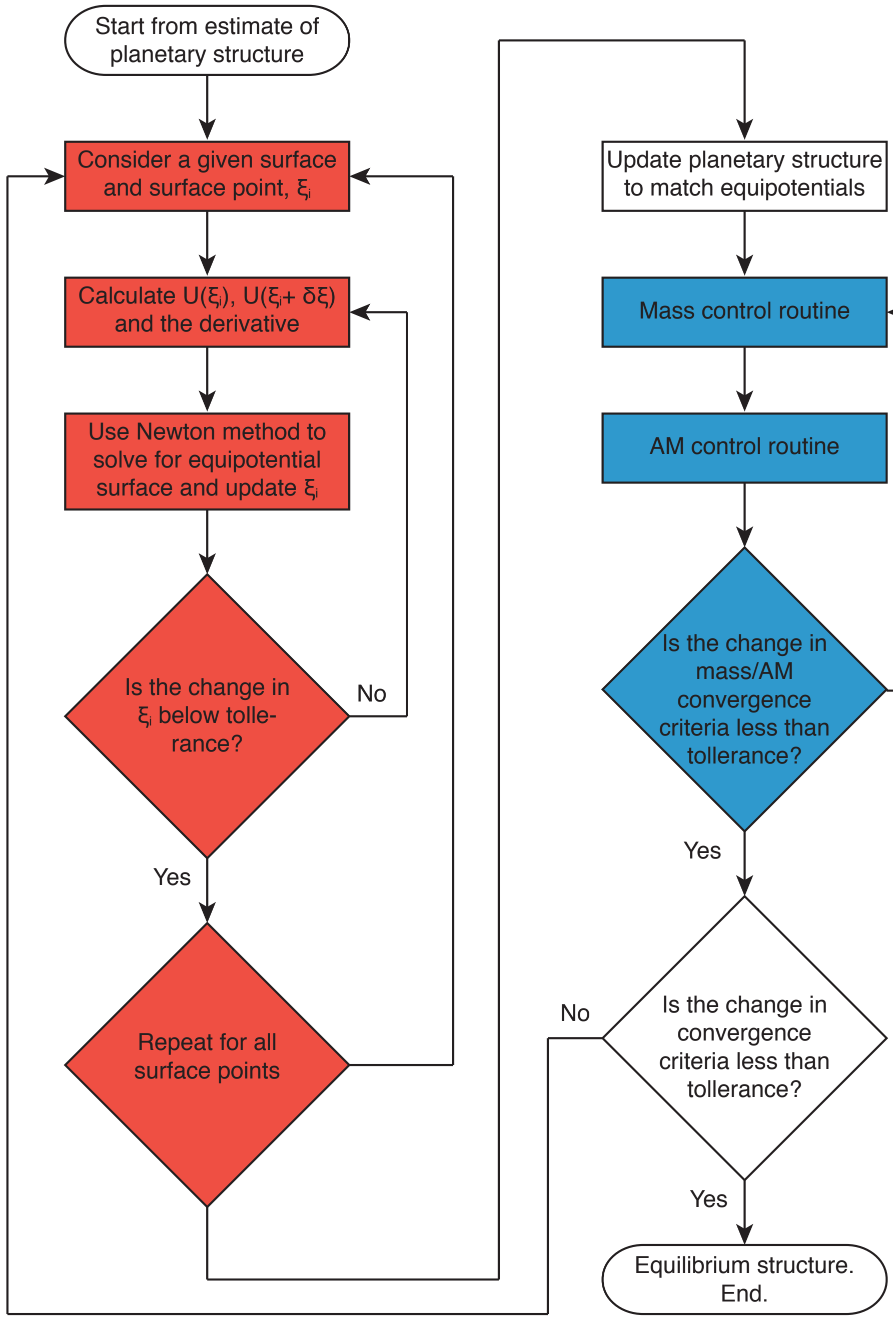
Start from estimate of planetary structure
Consider a given surface and surface point, $\xi_i$
Calculate $U(\xi_i)$, $U(\xi_i+\delta\xi)$ and the derivative
Use Newton method to solve for equipotential surface and update $\xi_i$
Is the change in $\xi_i$ below tolle-rance?
No
Yes
Repeat for all surface points
Update planetary structure to match equipotentials
Mass control routine
AM control routine
Is the change in mass/AM convergence criteria less than tollerance?
No
Yes
Is the change in convergence criteria less than tollerance?
No
Yes
Equilibrium structure. End.

Figure 3: Outline of HERCULES algorithm. Red indicates the shape iteration and blue the mass and AM control iteration.

### 3.2.2 From an output file: $\mathcal{F}_{\text{start}} = 1$

The structure can also be initialized by reading in a HERCULES binary output file whose file location is given in the input file. This is done in the `initialisation` function. The shape, thermodynamic properties and rotational state of the structure are set to be the same as the given start file. $k_{\text{max}}$, material EOSs, target mass and AM, $p_{\text{min}}$, and run parameters are all maintained from the original input file. The properties of the planet are then recalculated within the code to ensure consistency.

**Note**

At present, the body in the start file is required to have the same $N_{\text{lay}}$, $N_{\text{mat}}$, $N_{\text{lay}}^{\text{mat}}$ and $N_\mu$ as defined in the input file for the present run.

## 3.3 Shape iteration

The shape iteration takes each $\mu_j$ point on a surface in turn and, using a Newton-Raphson method, solves Equation 24 to find the location of the equipotential surface corresponding to the potential at the surface of that spheroid at the equator. This routine is in the function `itterate`. The equation this routine is solving is Equation 24 for a specific $\mu_j$:

$$X(\tilde{r}_i(\mu_j), \mu_j) = \Phi(\tilde{r}_i(\mu_j), \mu_j) + Q(\tilde{r}_i(\mu_j), \mu_j) - \Phi(a_i, 0) - Q(a_i, 0) = 0\,. \tag{40}$$

The current surface of spheroid $i$ is used as an initial guess to the position of the equipotential surface, i.e., $\xi_i^0 = \tilde{\xi}_i(\mu_j)$. The value of the function $X$ is calculated at a point $\xi_i$ and a point $\xi_i + \delta\xi$, to numerically determine the gradient of the function. $\delta\xi$ is defined by the user. The approximation to the equipotential surface is then updated using the Newton-Raphson formula with a first order determination of the $X$ function gradient

$$\xi_i^{n+1} = \xi_i^n - \frac{X(\xi_i^n, \mu_j)\delta\xi}{X(\xi_i^n + \delta\xi, \mu_j) - X(\xi_i^n, \mu_j)}\,. \tag{41}$$

The iteration continues with new determinations of $X$ until the convergence criteria are met (see Section 3.7).

**Note**

While the equipotential surfaces are found, the structure used to calculate the gravitational and rotational potential is not changed.

### 3.4 Thermodynamics and equations of state

In HERCULES, there are options to treat the concentric layers as being made of a real material with a defined EOS. Generally this is done in the mass control routines that change the density of a layer depending on the pressure and material within that layer (see Section 3.5). In the following sections we outline how the pressure of a given layer is calculated from the equatorial potential and density, and then describe the structure of the EOSs that HERCULES can use.

#### 3.4.1 Calculation of pressure

The pressure in the structure is calculated assuming hydrostatic equilibrium. Given that each of the layers is of constant density, the pressure at the top of a layer can be calculated as follows [Hubbard, 2013]:

$$p_i = p_{i-1} + \rho_{i-1} \left[ U(a_i, 0) - U(a_{i-1}, 0) \right] . \tag{42}$$

In HERCULES, the pressure at the surface of the outermost spheroid is specified, $p_0 = p_{\mathrm{min}}$, and so the pressure at the top of all subsequent layers can be determined sequentially. The pressure at the centre of the body is given by

$$p_{\mathrm{core}} = p_{N_{\mathrm{lay}}-1} + \rho_{N_{\mathrm{lay}}-1} \left[ U(0, 0) - U(a_{N_{\mathrm{lay}}-1}, 0) \right] . \tag{43}$$

which is calculated for reference and used for some convergence criteria (see Section 3.7).

To calculate the density, we use an approximation for the pressure in the middle of the layer by taking the average of the pressure at the top and bottom of the layer. This approximation amounts to assuming that the variation in potential is linear within each layer. The assumption of linear potential is a reasonably good one in most cases.

#### 3.4.2 Equations of state

In this version of HERCULES, equations of state must have a common structure, although future versions will allow for more EOS types. HERCULES accepts EOS as ASCII files with columns of pressure, density, temperature and entropy (although the latter two are not used in calculations or indeed stored for the structure). For the structure of these files see Section 6. Any monotonically increasing density profile is allowable, but the $p$-$\rho$ relationships are defined at runtime so properties like total energy are not conserved. Future versions of HERCULES will allow for the calculation of thermal structure within the code to conserve such properties.

In order to calculate the density from a given pressure, the EOS function `calc_rho` linearly interpolates the given $p$-$\rho$ points. There is also an option (see Section 8) to interpolate the points linearly in log space for densities below 3000 kg m$^{-3}$, which may give better results in the the low density regime.

## 3.5 Mass control

HERCULES contains a number of different routines to alter the density of spheroids to control the mass of each material layer. Typically these routines attempt to conserve the mass in each material layer, and are applied in the mass/AM iteration after the shape iteration and are in the `itterate` function. The properties of the planet are recalculated after mass control.

### 3.5.1 Constant density: $\mathcal{F}_{\rm Mconc} = 0$

This option does not alter the density of the spheroids and keeps them constant throughout the calculation. If $\mathcal{F}_{\rm start} = 0$, the density of each concentric layer is set to the reference density, $\rho_{\rm ref}$, at the start of the calculation. Note that this mass control option is not compatible with any AM conservation routine and only works for an imposed, constant rotation rate: AM control routine option $\mathcal{F}_{\rm Lconc} = 0$ (Section 3.6). This routine does not conserve the mass of the body or of material layers.

### 3.5.2 Density scaling: $\mathcal{F}_{\rm Mconc} = 1$

This option allows the mass of the body to be conserved by scaling the density of each spheroid. The density is determined from the EOS for the relevant material using the calculated pressure. The densities of all layers in a given material are then scaled by a constant factor that conserves the mass of that material. The scaling factors, $\lambda_i$, depend on each other as conserving the mass in one material layer requires knowledge of the density of all concentric spheroids that have a volume that intersects part of the volume of the material layer. After scaling, the mass of a given material $k$ is given by

$$M_k^{\rm mat} = \sum_{i<k}^{\rm material} \lambda_i \sum_{j:i}^{\rm layers} \delta\rho_j (V_{\rm out}^k - V_{\rm in}^k) + \sum_{i:k}^{\rm layers} \rho_i \lambda_k (V_i - V_{\rm in}^k)\,, \tag{44}$$

where $M_k^{\rm mat}$ is the mass of material layer $k$, $V_{\rm out}^k$ and $V_{\rm in}^k$ are the volume of the outside and inside spheroids, and $\lambda_k$ is the density scaling factor for material $k$. $i : k$ indicates the layers that are of material $k$. We can rewrite this to give

$$\lambda_k = \frac{M_k^{\rm mat} - \sum_{i<k}^{\rm material} \lambda_i \sum_{j:i}^{\rm layers} \delta\rho_j (V_{\rm out}^k - V_{\rm in}^k)}{\sum_{i:k}^{\rm layers} \delta\rho_i (V_i - V_{\rm in}^k)}\,, \tag{45}$$

which can be solved sequentially for each $\lambda_k$, starting from the outermost material layer. In this way, we require no prior knowledge of the other $\lambda_k$ that are yet to be determined.

The change in the density also feeds back to a change in the pressure structure and hence calculated densities. The routine is therefore iterated over as part of the mass and AM control iteration until the convergence criteria are met (see Section 3.7).

This density scaling preserves the mass of the body, but results in a distortion of the EOS for each material. This was the approach taken by Hubbard [2013], as for Jupiter the radius of the planet is better determined than the EOS of helium-hydrogen mixtures. However, for other application the distortion of the EOS may not be acceptable.

### 3.5.3 Volume scaling: $\mathcal{F}_{\mathrm{Mconc}} = 2$

HERCULES can also conserve the mass of the planet by using a consistent EOS but by scaling the radius of the material layers. For this routine the density of each layer is determined from the pressure and the EOS only. The equatorial radius of each spheroid in a material is then scaled by a constant factor to maintain the mass of that material layer. As for the previous routine, we calculate the required scaling factors by considering the mass of a material layer after scaling:

$$M_k^{\mathrm{mat}} = \sum_{i<i_{\mathrm{out}}^k} \delta\rho_i(\lambda_k V_{\mathrm{out}}^k - \lambda_{k+1} V_{\mathrm{in}}^k) + \sum_{i_{\mathrm{out}}^k \leq i < i_{\mathrm{in}}^k} \delta\rho_i(\lambda_k V_i - \lambda_{k+1} V_{\mathrm{in}}^k)\,, \tag{46}$$

where $M_k^{\mathrm{mat}}$ is the mass of material layer $k$, $i_{\mathrm{out}}$ is the index of the outermost layer in material layer $k$, $i_{\mathrm{in}}^k$ is the index of the shell on the inside boundary of material layer $k$ (note that this layer is not of material $k$), $V_{\mathrm{out}}^k$ and $V_{\mathrm{in}}^k$ are the volume of the outside and inside spheroids, and $\lambda_k$ is the volume scaling factor for material $k$ which applies to the volume of the spheroids inside material layer $k$. Rewriting this expression:

$$M_k^{\mathrm{mat}} = \lambda_k \left( \sum_{i<i_{\mathrm{out}}^k} \delta\rho_i V_{\mathrm{out}}^k + \sum_{i_{\mathrm{out}}^k \leq i < i_{\mathrm{in}}^k} \rho_i V_i \right) - \lambda_{k+1} \left( \sum_{i<i_{\mathrm{out}}^k} \delta\rho_i V_{\mathrm{in}}^k + \sum_{i_{\mathrm{out}}^k \leq i < i_{\mathrm{in}}^k} \rho_i V_{\mathrm{in}}^k \right)\,, \tag{47}$$

which can be rearranged to give

$$\lambda_k = \frac{M_k^{\mathrm{mat}} + \lambda_{k+1} \left( \sum_{i<i_{\mathrm{in}}} \rho_i V_{\mathrm{in}}^k \right)}{\left( \sum_{i<i_{\mathrm{out}}} \rho_i V_{\mathrm{out}}^k + \sum_{i_{\mathrm{out}} \leq i < i_{in}} \rho_i V_i \right)}\,. \tag{48}$$

We can then solve for each $\lambda_k$ sequentially from the inside out, as for the innermost material

$$\lambda_{N_{\mathrm{mat}}-1} = \frac{M_j^{\mathrm{mat}}}{\left( \sum_{i<i_{\mathrm{out}}^j} \rho_i V_{\mathrm{out}}^j + \sum_{i_{\mathrm{out}}^j \leq i < i_{\mathrm{in}}^j} \rho_i V_i \right)}\,, \tag{49}$$

where $j = N_{\mathrm{mat}} - 1$, which requires no prior knowledge of the other $\lambda$.

The volume scaling is applied by scaling the equatorial radius, $a$. The scaled $a_i$, $a_i'$, are given by

$$a' = \lambda^{\frac{1}{3}} a\,. \tag{50}$$

Each of the spheroids in the body are then given the new scaled equatorial radii, and hence polar radii, but their shape remains unchanged. This routine conserves mass but must be iterated over to give density and pressure profiles that are consistent with the EOS.

In some cases, the difference in scaling factors between material layers causes the surface of spheroids to change radial order i.e., $a_{i-1}$ can be less than $a_i$. Such an arrangement is not acceptable in HERCULES. The radii of any crossing spheroids are altered sequentially from the outside inwards so that

$$a_i = a'_i + 1.005 \times (a'_i - a'_{i-1}) \, . \tag{51}$$

where $a'_i$ are the equatorial radii calculated from scaling the spheroids as described above. A single crossing surface can cause several spheroids to have their radii altered which is done sequentially from the outside inwards.

Similarly, the routine ensures there are not large gaps between the surface of spheroids at the contact between material layers. If the gap between the outermost layer of one material and the innermost layer of the next is greater than 5% more than the spacing in the upper material, the equatorial radii of spheroids whose surfaces are in the upper material layer are altered. The radii of all the relevant spheroids are decreased such that the gap between the bounding spheroids is the same as the spacing of spheroids in the upper layer.

When the radii of the spheroids must be changed from their calculated positions from Equation 50, mass is not conserved on that iteration step. Over several iterations the structure will come to an arrangement where radii scaling corrections are not necessary and mass conservation is achieved.

## 3.6 Angular momentum control

HERCULES has a number of routines to control the rotational state of a planet. These routines are applied after the mass control routines. Since the rotational state feeds back into the structure of the planet, the AM control routines are iterated over, along with the mass control routines, to acquire a self consistent structure.

**Note**

In the current version of HERCULES, each layer has a constant rotational velocity, $\omega$, but this will be altered in future versions.

### 3.6.1 No conservation with constant rotation rate: $\mathcal{F}_{\text{Lconc}} = 0$

When using this flag, the rotational profile of the planet is not changed during the calculation. For $\mathcal{F}_{\text{start}} = 0$ the body is set as corotating with the angular velocity given in the input file. For $\mathcal{F}_{\rm Lconc} = 1$ the angular velocity of each of the layers is kept constant from the initial structure.

### 3.6.2 Angular momentum conservation with imposed $\omega$ profile: $\mathcal{F}_{\rm Lconc} = 1$

This routine conserves AM by prescribing a given angular velocity profile. In potential theory, a conserved field is achieved only if the angular velocity is purely a function of distance from the rotation axis. In the current version of HERCULES, each layer has its own rotation rate. To have a conservative field each layer must have the same angular velocity. However, while iterating to find the equilibrium structure, the angular velocity of the body may exceed the angular velocity of a Keplerian orbit at the equator, i.e., the body may exceed the CoRoL. A super-Keplerian angular velocity leads to negative pressure gradients in the structure which breaks the requirement for a monotonically decreasing density with radius. It is hence helpful to allow the outer layers of the structure to be on sub-Keplerian orbits at intermediate iteration steps to allow the iteration to continue, as long as the final body is corotating. Otherwise, the structure of the final body is not accurately determined using a potential field method.

The form of the profile used for this routine is that of a corotating planet and a sub-Keplerian outer region. The angular velocity of each layer is prescribed as:

$$\omega = \min(\omega_{\rm orb}, \omega_{\rm rot})\,, \tag{52}$$

where $\omega_{\rm orb}$ is the angular velocity of a particle in pressure supported keplerian orbit and $\omega_{\rm rot}$ is the angular velocity of the corotating body. In theory, $\omega_{\rm orb}$ is given by balancing the centrifugal force with the pressure gradient and the gravitational potential gradient in the equatorial plane

$$\text{acceleration} = -\omega_{\rm orb}^2 r = -\frac{1}{\rho}\frac{\partial p}{\partial r} + \frac{\partial \Phi}{\partial r}\,. \tag{53}$$

Solving for $\omega_{\rm orb}$ gives

$$\omega_{\rm orb} = \sqrt{\frac{1}{r}\left[\frac{1}{\rho}\frac{\partial p}{\partial r} - \frac{\partial \Phi}{\partial r}\right]}\,. \tag{54}$$

However, the expression above for $\omega_{\rm orb}$ is effectively the same expression we use to enforce hydrostatic equilibrium in the structure (Equation 42). The structure that satisfies conservation of AM is non-unique and, for this routine, we impose the orbital angular velocity profile to fully constrain the problem.

We define the orbital velocity by forcing it to be a fraction of the keplerian velocity that varies with radius

$$\omega_{\rm orb} = f\omega_{\rm kep} = f\sqrt{\frac{1}{r}\left[-\frac{\partial \Phi}{\partial r}\right]}\,, \tag{55}$$

where

$$f = \Omega_1 - \Omega_2 \left(1 - \frac{r_{eq}}{a_0}\right)^{\Omega_3} , \tag{56}$$

and $r_{eq}$ is the radius in the equatorial plane and $\Omega_i$ are user defined constants. Note that as $r_{eq} \to a_0$, $f \to \Omega_1$ and as $r_{eq} \to 0$, $f \to \Omega_1 - \Omega_2$. To calculate the structure of a body that is corotating except when it exceeds the CoRoL, set $\Omega_1 = 1$ and $\Omega_2 = 0$. Having defined $\omega_{\rm orb}$, the only free parameter we have now is $\omega_{\rm rot}$ which can be solved for by conserving AM.

To find $\omega_{\rm rot}$, we conserve AM, but to do this we need to know the moments of inertia of each layer and hence each spheroid. The moment of inertia about the $z$ axis of the $i$th constant-density spheroid is defined as

$$I_i = \int_{\mathcal{V}} \frac{2}{3} \delta\rho_i r^2 (1 - P_2(\mu)) \mathrm{d}V' , \tag{57}$$

or alternatively

$$I_i = \frac{4\pi}{3} \delta\rho_i \int_{-1}^{1} \int_{0}^{\tilde{r}_i(\mu')} (r')^4 (1 - P_2(\mu')) \mathrm{d}r' \mathrm{d}\mu' . \tag{58}$$

We use the symmetry of the spheroids and integrate over $r$ to give

$$I_i = \frac{8\pi}{15} \delta\rho_i \int_{0}^{1} (\tilde{r}_i(\mu'))^5 (1 - P_2(\mu')) \mathrm{d}\mu' . \tag{59}$$

This is the expression for the moment of inertia for a given spheroid.

Each layer in HERCULES has a constant angular velocity. We need to know the moment of inertia for each layer between two spheroids to calculate the AM. Consider the moment of inertia of the material in the $i$th spheroid that lies between the surfaces of the smaller spheroids $n$ and $j$. In this case the moment of inertia of the material in layer $i$ that lies between spheroids $n$ and $j$ is

$$I_{n\to j,i} = \frac{4\pi}{3} \delta\rho_i \int_{-1}^{1} \int_{\tilde{r}_j(\mu')}^{\tilde{r}_n(\mu')} (r')^4 (1 - P_2(\mu')) \mathrm{d}r' \mathrm{d}\mu' . \tag{60}$$

We use the symmetry of the spheroids and integrate over $r$ to give

$$I_{n\to j,i} = \frac{8\pi}{15} \delta\rho_i \int_{0}^{1} \left[(\tilde{r}_n(\mu'))^5 - \tilde{r}_j(\mu'))^5\right] (1 - P_2(\mu')) \mathrm{d}\mu' . \tag{61}$$

This is simply

$$I_{n\to j,i} = \frac{\delta\rho_i}{\delta\rho_n} I_n - \frac{\delta\rho_i}{\delta\rho_j} I_j , \tag{62}$$

where $i \leq n < j$.

We can now write the AM of the structure as

$$\begin{aligned} L_{\rm tot} = & \sum_{i=i_{\rm out}^{\rm orb}}^{N-1} I_i \omega_{\rm rot} + \sum_{i=0}^{i_{\rm out}^{\rm orb}-1} \frac{\delta\rho_i}{\delta\rho_{i_{\rm out}^{\rm orb}}} I_{i_{\rm out}^{\rm orb}} \omega_{\rm rot} \\ & + \sum_{i=0}^{i_{\rm out}^{\rm orb}-1} \sum_{j=0}^{i} \left[ \frac{\delta\rho_j}{\delta\rho_i} I_i - \frac{\delta\rho_j}{\delta\rho_{i+1}} I_{i+1} \right] \omega_{{\rm orb},i} \, , \end{aligned} \tag{63}$$

where $i_{\rm out}^{\rm orb}$ is the outermost layer that is corotating and $\omega_{{\rm orb},i}$ is the orbital angular velocity for layer $i$. The first term is the AM of all the spheroids that lie wholly within the corotating region. The second term is the AM of the material that lies within the corotating region but belongs to spheroids whose surfaces are not corotating. The third term is the AM of the layers that are rotating with orbital angular velocities. The second sum in the third term takes into account that portions of different spheroids are rotating at different orbital velocities. We can solve this equation to find the $\omega_{\rm rot}$ that conserves AM.

The conservation procedure works as follows. Firstly, the $\omega_{\rm orb}$ for each layer is calculated from the gravitational potential at the equator of the planet. Then the $\omega_{\rm orb}$ are compared with the present $\omega_{\rm rot}$ to identify the outermost spheroid that has $\omega_{\rm orb} > \omega_{\rm rot}$, hence determining $i_{\rm out}^{\rm orb}$. The $\omega$ of the non-corotating layers are then set to $\omega_{\rm orb,i}$. Equation 63 is then solved to give the new $\omega_{\rm rot}$ that conserves AM. The scheme is then iterated over to find an AM conserving angular velocity profile.

## 3.7 Convergence criteria

The convergence criteria available in each version of HERCULES are likely to vary as different criteria are found to be more effective for different purposes. Here we outline the convergence criteria for the present version for each of the iterative loops. Note that each loop has a maximum number of iterations ($n_{\rm int}^{\xi}$ for the shape iteration and $n_{\rm int}$ otherwise) that are allowed before the iteration exits and the code continues.

The convergence criterion for the main iteration for the equilibrium structure is based on the equatorial potentials. The iteration is stopped when

$$\left| \frac{\sum_{i=0}^{N_{\rm lay}} \left[ U^n(a_i^n, 0) - U^{n-1}(a_i^{n-1}, 0) \right]}{N_{\rm lay} U^n(a_0^n, 0)} \right| < \chi_{\rm toll} \, , \tag{64}$$

where $n$ indicates the current iteration and $\chi_{\rm toll}$ is given in the input file. This is similar to requiring that the average fractional change in equatorial potential is less than a given tolerance.

In the shape iteration, the Newton-Raphson method is repeated until the fractional change in $\xi$, scaled by the fractional equatorial radius of the given spheroid, is less than a value $\chi^{\xi}_{\text{toll}}$. In other words the iteration continues until

$$\left|\left(\xi_i^n - \xi_i^{n-1}\right)\left(\frac{a_i}{a_0}\right)\right| < \chi^{\xi}_{\text{toll}}\,, \tag{65}$$

where $n$ indicates the current iteration and $\chi^{\xi}_{\text{toll}}$ is given in the input file.

The convergence condition for the AM and mass control loop is that the fractional change in the core pressure is less than a certain value, $\chi_{\text{toll}}$, i.e.

$$\left|p^n_{\text{core}} - p^{n-1}_{\text{core}}\right| < \chi_{\text{toll}}\,. \tag{66}$$

Note that $\chi_{\text{toll}}$ is the same value used for the main iteration.

## 3.8 Pre-calculated integrals

During the iterative process in HERCULES the potential must be calculated a large number of times. In the formulation of Hubbard [2013], the $J_i$ can be computed using Gaussian quadrature at the start of each iteration, and are the only parameters required to calculate the potential at any point in the structure. However, in our more general formulation, there are integrals whose limits depend on $\mu_r$, which is dependent on the point for which the potential is being calculated. It is not possible to precompute these integrals in full before each iteration, making calculation of the equilibrium structure more numerically expensive. To minimize the computational intensity, HERCULES precomputes the integrals with an upper limit corresponding to each $\mu$ point. The integrands at each $\mu$ point are also recorded. A first order interpolation is used to find the value of the full integrals for a given $\mu_r$ point when required. The partial integrals and integrands are recalculated at specific points between iterations. In this section, we take each of the Equations 30-36 and recast them in terms of pre-computable integrals. Note that the $J_{i,2k}$, $M_i$ and some other integrals are fully pre-computable.

Note

Our approach for dealing with precomputed partial integrals and integrands is given in Section 3.8.1. Although we do not write it explicitly, the same approach is taken with all other $\mu_r$-dependent precomputed integrals.

### 3.8.1 $N_{i,0}(\xi, \mu_r)$

We can recast Equation 30,

$$N_{i,0}(\xi,\mu_r) = \left(\frac{3}{2}\right)\frac{\int_0^{\mu_r}\left[(\tilde{\xi}_i(\mu'))^2 - \xi^2\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'}\,, \tag{67}$$

as

$$N_{i,0}(\xi,\mu_r) = \left(\frac{3}{2}\right) \frac{\int_0^{\mu_r} (\tilde{\xi}_i(\mu'))^2 \mathrm{d}\mu' - \xi^2 \mu_r}{\int_0^1 \left[\tilde{\xi}_i(\mu')\right]^3 \mathrm{d}\mu'} \,. \tag{68}$$

The integral in the denominator is constant for a given structure and can be precomputed. The integral in the numerator varies depending on $\mu_r$, but HERCULES precomputes the integral evaluated using $\mu_r = \mu_i$ for each $\mu_i$ point on each surface. The value of the integrand is also precomputed at each $\mu$ point. For clarity we write

$$N_{i,0}(\xi,\mu_r) = \left(\frac{3}{2}\right) \frac{\bar{N}_{i,0}(\mu_r) - \xi^2 \mu_r}{\bar{M}_i} \,, \tag{69}$$

where

$$\bar{N}_{i,0}(\mu_r) = \int_0^{\mu_r} (\tilde{\xi}_i(\mu'))^2 \mathrm{d}\mu' \tag{70}$$

is partially precomputed and

$$\bar{M}_i = \int_0^1 \left[\tilde{\xi}_i(\mu')\right]^3 \mathrm{d}\mu' \,. \tag{71}$$

$\bar{M}_i$ can be precomputed in full each time we update the structure. Note that

$$M_i = \frac{4\pi\delta\rho_i a_i^3}{3} \bar{M}_i \,. \tag{72}$$

$\bar{N}_{i,0}(\mu_r)$ can be found from the precomputed values as

$$\bar{N}_{i,0}(\mu_r) = \bar{N}_{i,0}(\mu_j) + \int_{\mu_j}^{\mu_r} (\tilde{\xi}_i(\mu'))^2 \mathrm{d}\mu' \,, \tag{73}$$

where $\mu_j$ is the $\mu$ point below $\mu_r$. Using a first order trapezium rule

$$\bar{N}_{i,0}(\mu_r) = \bar{N}_{i,0}(\mu_j) + \frac{1}{2}\left[(\xi_i)^2 + (\tilde{\xi}_i(\mu_j))^2\right]\left[\mu_r - \mu_j\right] \,. \tag{74}$$

Note that the substitution $\tilde{\xi}_i(\mu_r) = \xi_i$, where $\xi_i$ is the point at which the function is being evaluated, is allowed by definition at $\mu_r$. We use the same approach for all of the precomputed integrals that depend on $\mu_r$.

### 3.8.2 $N_{i,2}(\xi, \mu_r)$

Similarly we can rewrite Equation 31,

$$N_{i,2}(\xi, \mu_r) = -\frac{3\int_0^{\mu_r}\left[\ln\left(\tilde{\xi}_i(\mu')/\xi\right) P_2(\mu')\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'}, \tag{75}$$

as

$$N_{i,2}(\xi, \mu_r) = -3\frac{\left[\bar{N}_{i,2}(\mu_r) - \bar{P}_2(\mu_r)\ln\xi\right]}{\bar{M}_i}, \tag{76}$$

where

$$\bar{N}_{i,2}(\mu_r) = \int_0^{\mu_r} P_2(\mu')\ln\tilde{\xi}_i(\mu')\mathrm{d}\mu', \tag{77}$$

$$\bar{P}_{2k}(\mu_r) = \int_0^{\mu_r} P_{2k}(\mu')\mathrm{d}\mu' = \frac{P_{2k+1}(\mu_r) - P_{2k-1}(\mu_r)}{4k+1}, \quad k > 0, \tag{78}$$

and $2k$ is the spherical harmonic degree. We can pre-calculate partial integrals and the integrand for $\bar{N}_{i,1}(\mu_r)$ and calculate $\bar{P}_{2k}(\mu_r)$ on the fly as it is needed.

Note that $\bar{P}_{2k}(1) = 0$ so

$$N_{i,2}(\xi, 1) = -3\frac{\bar{N}_{i,2}(1)}{\bar{M}_i}. \tag{79}$$

### 3.8.3 $N_{i,2k}(\xi, \mu_r)$ for $k \geq 2$

We write Equation 32,

$$N_{i,2k}(\xi, \mu_r) = -\left(\frac{3}{2k-2}\right)\frac{\int_0^{\mu_r}\left[\left(\xi^{\,2-2k} - (\tilde{\xi}_i(\mu'))^{\,2-2k}\right) P_{2k}(\mu')\right]\mathrm{d}\mu'}{\int_0^1\left[\tilde{\xi}_i(\mu')\right]^3\mathrm{d}\mu'}, \quad k \geq 2, \tag{80}$$

as

$$N_{i,2k}(\xi, \mu_r) = -\left(\frac{3}{2k-2}\right)\frac{\bar{P}_{2k}(\mu_r)\xi^{\,2-2k} - \bar{N}_{i,2k}(\mu_r)}{\bar{M}_i}, \quad k \geq 2, \tag{81}$$

where

$$\bar{N}_{i,2k}(\mu_r) = \int_0^{\mu_r}\left[\left((\tilde{\xi}_i(\mu'))^{\,2-2k}\right) P_{2k}(\mu')\right]\mathrm{d}\mu', \quad k \geq 2. \tag{82}$$

$\bar{N}_{i,2k}(\mu_r)$ is precomputed.

Note that $\bar{P}_{2k}(1) = 0$ so

$$N_{i,2k}(\xi, 1) = \left(\frac{3}{2k-2}\right)\frac{\bar{N}_{i,2k}(1)}{\bar{M}_i}, \quad k \geq 2. \tag{83}$$

### 3.8.4 $K_{i,0}(\xi, \mu_r)$

Equation 33,

$$K_{i,0}(\xi, \mu_r) = \frac{\int_0^{\mu_r} \left[(\tilde{\xi}_i(\mu'))^3 - \xi^3\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}_i(\mu')\right]^3 \mathrm{d}\mu'} \,, \tag{84}$$

can be rewritten

$$K_{i,0}(\xi, \mu_r) = \frac{\bar{K}_{i,0}(\mu_r) - \xi^3 \mu_r}{\bar{M}_i} \,, \tag{85}$$

where

$$\bar{K}_{i,0}(\mu_r) = \int_0^{\mu_r} (\tilde{\xi}_i(\mu'))^3 \mathrm{d}\mu' \,. \tag{86}$$

$\bar{K}_{i,0}(\mu_r)$ is precomputed at each $\mu$ point.

### 3.8.5 $K_{i,2k}(\xi, \mu_r)$ for $k \geq 1$

Similarly Equation 34

$$K_{i,2k}(\xi, \mu_r) = -\left(\frac{3}{2k+3}\right) \frac{\int_0^{\mu_r} \left[\left((\tilde{\xi}_i(\mu'))^{2k+3} - \xi^{2k+3}\right) P_{2k}(\mu')\right] \mathrm{d}\mu'}{\int_0^1 \left[\tilde{\xi}_i(\mu')\right]^3 \mathrm{d}\mu'}, \quad k \geq 1 \,, \tag{87}$$

is rewritten

$$K_{i,2k}(\xi, \mu_r) = -\left(\frac{3}{2k+3}\right) \frac{\bar{K}_{i,2k}(\mu_r) - \bar{P}_{2k}(\mu_r)\xi^{2k+3}}{\bar{M}_i}, \quad k \geq 1 \,, \tag{88}$$

where

$$\bar{K}_{i,2k}(\mu_r) = \int_0^{\mu_r} \left[(\tilde{\xi}_i(\mu'))^{2k+3} P_{2k}(\mu')\right] \mathrm{d}\mu', \quad k \geq 1 \,. \tag{89}$$

$\bar{K}_{i,2k}(\mu_r)$ is precomputed.

## 3.9 Gravitational moments

In HERCULES, the $J_{i,2k}$ are defined for each spheroid to calculate the potential. This definition of $J_{i,2k}$ is not the same as that typically used, e.g., in Hubbard [2013]. The two are straightforwardly related by a scaling factor

$$J^H_{i,2k} = \frac{M_i}{M} \left(\frac{a_i}{a_0}\right)^{2k} J_{i,2k} \,, \tag{90}$$

where $J^H_{i,2k}$ is the definition used by Hubbard.

In order to calculate the true gravitational moment we take the definition

$$J_n = -\frac{1}{Ma_0^n}\int_{\mathcal{V}} \rho(\boldsymbol{r})r^n P_n(\mu)\mathrm{d}V'\,. \tag{91}$$

Using the symmetry properties of the planet this reduces to

$$J_n = -\frac{1}{Ma_0^n}4\pi\sum_{i=0}^{N-1}\int_0^1\int_0^{\tilde{r}_i}\delta\rho_i P_n(\mu')(r')^{n+2}\mathrm{d}r'\mathrm{d}\mu'\,. \tag{92}$$

Integrating in $r$

$$J_n = -\frac{1}{Ma_0^n}\frac{4\pi}{n+3}\sum_{i=0}^{N-1}\int_0^1\delta\rho_i P_n(\mu')(\tilde{r})^{n+3}\mathrm{d}\mu'\,. \tag{93}$$

Rearranging

$$J_n = \sum_{i=0}^{N-1}\frac{4\pi}{3}\delta\rho_i\frac{\bar{M}_i}{M}\frac{1}{a_0^n}\frac{-3}{n+3}\frac{\int_0^1 P_n(\mu')(\tilde{\xi})^{n+3}a_i^{n+3}\mathrm{d}\mu'}{\int_0^1(\tilde{\xi})^3\mathrm{d}\mu'}\,. \tag{94}$$

We can then substitute in the expression from Equation 35 to give

$$J_n = \sum_{i=0}^{N-1}\frac{4\pi}{3}\delta\rho_i\frac{\bar{M}_i}{M}\frac{a_i^{n+3}}{a_0^n}J_{i,n}\,, \tag{95}$$

which simplifies to

$$J_n = \sum_{i=0}^{N-1}\frac{M_i}{M}\left(\frac{a_i}{a_0}\right)^n J_{i,n}\,. \tag{96}$$

This gives the whole planet's gravitational moment.

## 3.10 Analytical tools

In order to analyze the planetary structures calculated by HERCULES, a number of routines have been devised to calculate various properties that are not required directly in the calculation. Typically, the functions that calculate these properties are sub-functions of the `planet` class (see Section 4.2). In this section we detail these functions and the calculations they perform.

### 3.10.1 Radial mass distribution

This routine calculates the mass outside a given cylindrical radius. The mass outside a given cylindrical radius, $R$, for a single constant-density spheroid can be calculated by the integral

$$M^i_{r_{eq}>R} = 4\pi \int_0^{\mu_c} \int_{\frac{R}{\sin\theta'}}^{\tilde{r}_i} \delta\rho_i \left(r'\right)^2 \mathrm{d}r' \mathrm{d}\mu' \,, \tag{97}$$

where $\mu_c$ is the critical $\mu$ at which the surface has a cylindrical radius of $R$. We can integrate with respect to r to give

$$M^i_{r_{eq}>R} = \frac{4\pi}{3}\delta\rho_i \int_0^{\mu_c} \left[\tilde{r}_i^3 - \left(\frac{R}{\sqrt{1-\mu'^2}}\right)^3\right] \mathrm{d}\mu' \,. \tag{98}$$

This satisfies the result for a sphere and holds in the limits that $R \to 0$ and $R \to \tilde{r}(0)$. For a sphere this result reduces to

$$M^i_{r_{eq}>R} = \frac{4\pi}{3}\delta\rho_i \left(a^2 - R^2\right)^{\frac{3}{2}} \,, \tag{99}$$

where $a$ is the radius of the sphere. For ease of calculation in the code, we complete the integral in the second term and write the expression as

$$M^i_{r_{eq}>R} = \frac{4\pi}{3}\delta\rho_i \left[a_i^3 \bar{K}_{i,0}(\mu_c) - \frac{R^3 \mu_c}{\sqrt{1-\mu_c^2}}\right] \,. \tag{100}$$

Both of the terms are either able to be precomputed or their integrands can be precomputed as above.

In order to calculate the total mass outside a given radius, we must sum over all the spheroids

$$M_{r_{eq}>R} = \sum_i^{a_i>R} M^i_{r_{eq}>R} \,. \tag{101}$$

The surface density can then be found directly from these calculations.

The function `planet.calc_Mout` applies the above equations and calculates the mass outside of the equatorial radii of each spheroid. The value for each spheroid is stored as part of the `planet` class.

### 3.10.2 Radial angular momentum distribution

Similar to Section 3.10.1, this function calculates the AM outside a given cylindrical radius. The AM of the material between the spheroids $i$ and $i+1$ outside of a given cylindrical radius is

$$L^i_{r_{eq}>R} = 4\pi \int_0^{\mu_c} \int_{\max\left\{\tilde{r}_{i+1}, \frac{R}{\sin\theta'}\right\}}^{\tilde{r}_i} \rho_i \omega_i \left(r'\right)^2 \left(r' \sin\theta'\right)^2 \mathrm{d}r' \mathrm{d}\mu' \,, \tag{102}$$

where $\mu_c$ is the critical $\mu$ at which surface $i$ has a cylindrical radius of $R$. $\omega_i$ and $\rho_i$ are the angular velocity and 'real' density of layer $i$. We can integrate with respect to r to give

$$L^i_{r_{eq}>R} = \frac{4\pi}{5} \rho_i \omega_i \int_0^{\mu_c} \left[1-\mu'^2\right] \left[ \left(\tilde{r}_i\right)^5 - \left( \max\left\{ \tilde{r}_{i+1}, \frac{R}{\sqrt{1-\mu'^2}} \right\} \right)^5 \right] \mathrm{d}\mu' \,. \tag{103}$$

Neither of the terms in this equation is pre-calculated and have to be calculated on each function call. As this is an analytical tool, this does not affect code performance. In the limit of true spheroids, Equation 103 reduces to

$$\begin{aligned} L^i_{r_{eq}>R} = \frac{4\pi}{5} \rho_i \omega_i \Bigg\{ & a_i^5 \left[ \sqrt{1-\left(\frac{R}{a_i}\right)^2} - \frac{1}{3}\left(1-\left(\frac{R}{a_i}\right)^2\right)^{\frac{3}{2}} \right] \\ & - a_i^5 \left[ \sqrt{1-\left(\frac{R}{a_{i+1}}\right)^2} - \frac{1}{3}\left(1-\left(\frac{R}{a_{i+1}}\right)^2\right)^{\frac{3}{2}} \right] \\ & - R^4 \left[ a_i \sqrt{1-\left(\frac{R}{a_i}\right)^2} - a_{i+1}\sqrt{1-\left(\frac{R}{a_{i+1}}\right)^2} \right] \Bigg\} \,. \end{aligned} \tag{104}$$

To calculate the total AM outside a given cylindrical radius, we sum over all the spheroids that extend beyond that radius and

$$L_{r_{eq}>R} = \sum_i^{a_i>R} L^i_{r_{eq}>R} \,. \tag{105}$$

The function `planet.calc_Lout` applies the above equations and calculates the AM outside of the equatorial radii of each spheroid. The value for each spheroid is stored as part of the `planet` class.

## 3.11 Numerical Notes

This section contains miscellaneous notes about the numerical implementation of HERCULES.

#### 3.11.1 Differentiation of an uneven matrix

The standard expressions for numerical differentiation break down when the space between points is uneven. Fortunately, it is still possible to make an expression for the derivative that is accurate up to second order. I have not found a reference for this expression so I give the derivation here for future reference. Starting from the Taylor expansion of a function at a point:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3)\,, \tag{106}$$

we use the expansion at two separate points to write

$$h_2^2 f(x+h_1) - h_1^2 f(x+h_2) = h_2^2 f(x) - h_1^2 f(x) + h_1 h_2^2 f'(x) - h_2 h_1^2 f'(x) + O(h_i^3)\,. \tag{107}$$

This can be rearranged to give

$$f'(x) = \frac{h_2^2[f(x+h_1) - f(x)] + h_1^2[f(x) - f(x+h_2)]}{h_1 h_2 (h_2 - h_1)}\,. \tag{108}$$

This expression reduces to the normal centered difference, backward and forward difference formulae with the correct choice of $h_1$ and $h_2$. We use this expression in HERCULES to find the pressure gradient, as the spacing of the surfaces of the spheroids can be unequal.

#### 3.11.2 Manipulation of C++ vectors

In HERCULES we use the `std::vector` class to store the properties of the planet, layers, etc. There are limited inbuilt functions for manipulation of vectors in C++ as if they were mathematical vectors and so we have developed our own. These are used sporadically throughout the code and detailed in `vector_operations.h` and `vector_operations.cc`.

## 4 HERCULES structures

The HERCULES code relies heavily on the object orientated features of C++. The parameters of the calculation, the planet, individual layers, and EOSs each have their own class. The class for each layer also contains information for the spheroid whose surface defines the outside of the layer. The variables relevant to each structure are stored within the structure and a series of functions are defined within each structure. Here, we describe each of these structures.

### 4.1 `parameters` class

The `parameters` class is used to store the parameters for an individual code run. The variables and functions in the `parameters` class are given in Table 1.

Table 1: Structure of the `parameters` class.

| Name | Symbol | Description |
|---|---|---|
| Variables | | |
| `run_name` | | Name of the code run (up to 80 characters). |
| `nint_max` | $n_{\rm int}$ | Maximum number of iterations for the main iteration and mass/AM control iteration. |
| `toll` | $\chi_{\rm toll}$ | Desired value of the convergence criterion for both the main iteration and the mass and AM iteration. |
| `xi_int_max` | $n^{\xi}_{\rm int}$ | Maximum number of iterations for shape iteration. |
| `xi_toll` | $\chi^{\xi}_{\rm toll}$ | Desired value of convergence criterion for the shape iteration. |
| `dxi` | $\delta\xi$ | $\xi$ step used to calculate the $\mathrm{d}U/\mathrm{d}\xi$ used in shape iteration. |
| `flag_start` | $\mathcal{F}_{\rm start}$ | Flag for how to initiate the structure. See Section 3.2. |
| `start_file` | | File to use in some initialization routines. See Section 3.2. |
| `flag_Mconc` | $\mathcal{F}_{\rm Mconc}$ | Flag for mass control option. See Section 3.5. |
| `flag_Lconc` | $\mathcal{F}_{\rm Lconc}$ | Flag for AM control option. See Section 3.6. |
| `flag_iter_print` | $\mathcal{F}_{\rm print}$ | Flag for whether to print an output file on every iteration. See Section 8. |
| Vectors | | |
| `omega_param` | $\lambda_i$ | Parameters for angular velocity profile used in some AM control options. See Section 3.6. |
| Functions | | |
| `write_binary` | | Function to write out the parameter class to a binary file. See Section 8. |
| `read_binary` | | Function to read in a parameter class from a binary file. |

## 4.2 `planet` class

The `planet` class is used to store the properties of the planet and contains the functions to calculate the properties of the planet. The information about each of the layers/spheroids is also included as a vector containing `concentric_layer` classes. The variables and functions in the `planet` class are given in Table 2.

Table 2: Structure of the `planet` class.

| Name | Symbol | Description |
|---|---|---|
| Variables | | |
| `Nmu` | $N_\mu$ | Number of $\mu$ points used to describe surfaces. |
| `Nlayer` | $N_{\mathrm{lay}}$ | Number of concentric layers/spheroids. |
| `Nmaterial` | $N_{\mathrm{mat}}$ | Number of material layers. |
| `kmax` | $k_{\mathrm{max}}$ | Maximum value of $k$ considered. $2k_{\mathrm{max}}$ is the maximum spherical harmonic degree included in the calculation. |
| `Mtot` | $M$ | Total mass of the body. |
| `Ltot` | $L$ | Total AM of the body. |
| `omega_rot` | $\omega_{\mathrm{rot}}$ | Rotation rate of corotating section of planet. |
| `pmin` | $p_{\mathrm{min}}$ | Imposed pressure at the outside of the outermost surface of the structure. See Section 3.4.1. |
| `amax` | $a_0$ | The equatorial radius of the outermost spheroid. |
| `aspect` | $\mathcal{A}_0$ | Aspect ratio of the outermost spheroid. |
| `ref_rho` | $\rho_{\mathrm{ref}}$ | Reference density for some mass control routines. See Section 3.5. |
| `Mtot_tar` | $\sum M^i_{\mathrm{tar}}$ | Target mass of the whole body for mass conservation routines. |
| `Ltot_tar` | $\sum L^i_{\mathrm{tar}}$ | Target AM of the whole body for AM conservation routines. |
| `Ucore` | $U_{\mathrm{core}}$ | Total potential at the centre of the body. |
| `pcore` | $p_{\mathrm{core}}$ | Pressure at the centre of the body. |
| Vectors | | |
| `real_rho` | $\rho_i$ | The ‘real’ densities of each of the layers. |
| `press` | $p_i$ | Pressure at the top of each of the layers. |
| `dpdr` | $\mathrm{d}p/\mathrm{d}r$ | Equatorial pressure gradient. |
| `mu` | $\mu_i$ | Array of $\mu$ points. |
| `Ulayers` | $U(a_i, 0)$ | Total potential at the equator of each spheroid. |
| `flag_material` | | Flag for each layer as to what material layer it belongs to. Starting from 0 for the outside material layer. |
| `M_materials` | $M^i_{\mathrm{tar}}$ | The target mass for each of the material layers so they can be conserved separately. |
| `material_lay` | $N^{\mathrm{mat}}_{\mathrm{lay}}$ | Number of layers in each material. |
| `layers` | | Vector of `concentric_layer` classes containing the details of each of the layers/spheroids in the planet. |

continued on next page

Table 2 – continued from previous page

| | | |
|---|---|---|
| `materials` | | Array of EOS classes containing the EOS for each of the materials. |
| `Plegendre` | $P_{2k}(\mu_i)$ | Array of precomputed Legendre Polynomials for each $\mu$ point. |
| `Mout` | $M^i_{\text{out}}$ | Mass of material at cylindrical radii greater than the equatorial radius of each spheroid. |
| `Lout` | $L^i_{\text{out}}$ | AM at cylindrical radii greater than the equatorial radius of each spheroid. |
| `Js` | $J_{2k}$ | The gravitational moments for the whole structure. |
| Functions | | |
| `precalc_Plegendre` | | Function to pre-calculate the Legendre polynomials for each $\mu$ point. |
| `initialise_ellipsoids` | | Initialization routine for constant density ellipsoids. |
| `calc_V` | | Function to calculate the gravitational potential at a point. Requires calculation of $\bar{M}_i$, $\bar{N}_{i,2k}(\tilde{\xi}_i(\mu_j),\mu_j)$, $\bar{K}_{i,2k}(\tilde{\xi}_i(\mu_j),\mu_j)$, $\bar{P}_{2k}$ and $\bar{J}_{i,2k}$ for each spheroid (`concentric_layer.calc_M`, `concentric_layer.precalc_Nbar`, `concentric_layer.precalc_Kbar` and `concentric_layer.calc_Js`) before being called. |
| `calc_Q` | | Function to calculate the rotational potential at a point. |
| `calc_X` | | Function to calculate $U(r,\mu) - U(a_i,0)$ for the shape iteration. Requires calculation of $U(a_i,0)$ before being called. |
| `calc_Mtot` | | Function to calculate the present mass of the body. Requires calculation of $M_i$ for each spheroid (`concentric_layer.calc_M`) before being called. |
| `calc_Ltot` | | Function to calculate the total AM of the body. Requires calculation of $I_i$ for each spheroid (`concentric_layer.calc_I`) before being called. |
| `calc_Mout` | | Function to calculate $M^i_{\text{out}}$ for each layer. Requires calculation of $M_i$ for each spheroid (`concentric_layer.calc_M`) before being called. |
| `calc_Lout` | | Function to calculate $L^i_{\text{out}}$ for each layer. Requires calculation of $I_i$ for each spheroid (`concentric_layer.calc_I`) before being called. |

continued on next page

Table 2 – continued from previous page

| | |
|---|---|
| `calc_Js` | Function to calculate the gravitational moments, $J_{2k}$, for the whole body. Requires calculation of $J_{i,2k}$ for each layer (`concentric_layer.calc_Js`) before being called. |
| `write_binary` | Function to write out the structure to a binary file. See Section 8. |
| `read_binary` | Function to read in a structure from a binary file. |

## 4.3 `concentric_layer` class

The `concentric_layer` class is used to store the parameters for each of the layers/spheroids in a planet and includes functions to calculate properties for a single layer/spheroid. The properties of the layer and spheroid that share an exterior surface are stored in the same structure. The variables and functions in the `concentric_layer` class are given in Table 3.

Table 3: Structure of the `concentric_layer` class.

| Name | Symbol | Description |
|---|---|---|
| Variables | | |
| `Nmu` | $N_\mu$ | Number of $\mu$ points used to describe the surface of the spheroid. Same as in planet class. |
| `kmax` | $k_{\max}$ | Maximum value of $k$ considered. $2k_{\max}$ is the maximum spherical harmonic degree included in the calculation. Same as in `planet` class. |
| `omega` | $\omega_i$ | Rotation rate of layer. |
| `rho` | $\delta\rho_i$ | Density of concentric spheroid. |
| `M` | $M_i$ | Mass of the concentric spheroid. |
| `V` | $V_i$ | Volume of the concentric spheroid. |
| `a` | $a_i$ | Equatorial radius of the spheroid. |
| `b` | $b_i$ | Polar radius of the spheroid. |
| `I` | $I_i$ | Moment of inertia of the spheroid. |
| `Mbar` | $\bar{M}_i$ | Precomputed integral for the spheroid. See Section 3.8. |
| Vectors | | |
| `mu` | $\mu_j$ | $\mu$ for each of the points on the surface of the spheroid. |
| `xi` | $\tilde{\xi}_i(\mu_j)$ | Array of $\xi_i$ for each $\mu_j$ point. |

continued on next page

Table 3 – continued from previous page

| | | |
|---|---|---|
| `Js` | $J_{2k}$ | The scaled gravitational moments for the spheroid. See Section 3.9. |
| `Nbar_mu` | $\bar{N}_{i,2k}(\tilde{\xi}_i(\mu_j),\mu_j)$ | Array of $\bar{N}_{i,2k}$ for each of the surface points. |
| `Kbar_mu` | $\bar{K}_{i,2k}(\tilde{\xi}_i(\mu_j),\mu_j)$ | Array of $\bar{K}_{i,2k}$ for each of the surface points. |
| `Nbar_integrand` | | Array of the integrand to calculate $\bar{N}_{i,2k}$ evaluated at each surface point. |
| `Kbar_integrand` | | Array of the integrand to calculate $\bar{K}_{i,2k}$ evaluated at each surface point. |
| Functions | | |
| `calc_I` | | Function to calculate the moments of inertia, $I_i$, of the spheroid. |
| `calc_Mbar` | | Function to calculate $\bar{M}_i$ for the spheroid. |
| `calc_M` | | Function to calculate $M_i$ for the spheroid. Requires the calculation of $\bar{M}_i$ before being called. |
| `calc_Js` | | Function to calculate $\bar{J}_{i,2k}$ for the spheroid. |
| `precalc_Nbar` | | Function to precalculate `Nbar_mu` and `Nbar_integrand`. |
| `precalc_Kbar` | | Function to precalculate `Kbar_mu` and `Kbar_integrand`. |
| `calc_Pbar` | | Function to calculate $\bar{P}_{2k}(\mu_r)$. |
| `calc_VRegI` | | Function to calculate the gravitational potential for a point in Regime I. Requires calling `precalc_Nbar`, `precalc_Kbar`, `calc_Mbar` and `calc_Js` before being called. |
| `calc_VRegII` | | Function to calculate the gravitational potential for a point in Regime II or III. Requires calling `precalc_Nbar`, `precalc_Kbar`, `calc_Mbar` and `calc_Js` before being called. |
| `calc_VRegIV` | | Function to calculate the gravitational potential for a point in Regime IV. Requires calling `precalc_Nbar`, `precalc_Kbar`, `calc_Mbar` and `calc_Js` before being called. |
| `calc_mu_r` | | Function to calculate $\mu_r$ for a given point for this spheroid. |
| `calc_N` | | Function to calculate $N_{i,2k}(\tilde{\xi}_i(\mu),\mu)$. Requires calling `precalc_Nbar` before being called. |
| `calc_K` | | Function to calculate $K_{i,2k}(\tilde{\xi}_i(\mu),\mu)$. Requires calling `precalc_Kbar` before being called. |
| `write_binary` | | Function to write out a layer to a binary file. See Section 8. |
| `read_binary` | | Function to read in a layer from a binary output file. |

### 4.4 `EOS` class

The `EOS` class contains the information needed for a material EOS in HERCULES. The variables and functions in the `EOS` class are given in Table 4.

Table 4: Structure of the `EOS` class.

| Name | Symbol | Description |
|---|---|---|
| Variables | | |
| `fname` | | Name and location of EOS file used. |
| `Ndata` | | Number of EOS points. |
| `EOS_type` | | Flag for the type of EOS used. See Sections 3.4 and 6.2. |
| Vectors | | |
| `p` | $p$ | Pressure for EOS points. |
| `rho` | $\rho$ | Density for EOS points. |
| `T` | $T$ | Temperature for EOS points. |
| `S` | $S$ | Entropy for EOS points. |
| Functions | | |
| `read_EOSfile` | | Read in a HERCULES EOS file into the `EOS` structure. |
| `write_binary` | | Function to write out a layer to a binary file. See Section 8. |
| `read_binary` | | Function to read in an EOS from a binary output file. |
| `calc_rho` | | Calculate the density from a given pressure point by interpolating the EOS points. |

## 5 Code file structure

The code for HERCULES is split between several files. In Table 5 we outline the contents of each of these files for easy reference.

Table 5: Description of all HERCULES code files and the functions they contain.

| Name | Description |
|---|---|
| `concentric_layer_class.h` | Header file for `concentric_layer` class. |

continued on next page

Table 5 – continued from previous page

| | |
|---|---|
| `concentric_layer` | `concentric_layer` class. For description see Section 4.3. |
| `concentric_layer_functions.cc` | |
| `concentric_layer.calc_I` | Function to calculate the moments of inertia, $I_i$, of the spheroid. |
| `concentric_layer.calc_Mbar` | Function to calculate $\bar{M}_i$ for the spheroid. |
| `concentric_layer.calc_M` | Function to calculate $M_i$ for the spheroid. Requires the calculation of $\bar{M}_i$ before being called. |
| `concentric_layer.precalc_Nbar` | Function to precalculate `Nbar_mu` and `Nbar_integrand`. |
| `concentric_layer.precalc_Kbar` | Function to precalculate `Kbar_mu` and `Kbar_integrand`. |
| `concentric_layer.calc_Pbar` | Function to calculate $\bar{P}_{2k}(\mu_r)$. |
| `concentric_layer.calc_Js` | Function to calculate $\bar{J}_{i,2k}$ for the spheroid. |
| `concentric_layer.calc_VRegI` | Function to calculate the gravitational potential for a point in Regime I. Requires calling `precalc_Nbar`, `precalc_Kbar`, `calc_Mbar` and `calc_Js` before being called. |
| `concentric_layer.calc_VRegII` | Function to calculate the gravitational potential for a point in Regime II or III. Requires calling `precalc_Nbar`, `precalc_Kbar`, `calc_Mbar` and `calc_Js` before being called. |
| `concentric_layer.calc_VRegIV` | Function to calculate the gravitational potential for a point in Regime IV. Requires calling `precalc_Nbar`, `precalc_Kbar`,`calc_Mbar` and `calc_Js` before being called. |
| `concentric_layer.calc_mu_r` | Function to calculate $\mu_r$ for a given point for this spheroid. |
| `concentric_layer.calc_N` | Function to calculate $N_{i,2k}(\tilde{\xi}_i(\mu), \mu)$. Requires calling `precalc_Nbar` before being called. |
| `concentric_layer.calc_K` | Function to calculate $K_{i,2k}(\tilde{\xi}_i(\mu), \mu)$. Requires calling `precalc_Kbar` before being called. |
| `concentric_layer.write_binary` | Function to write out a layer to a binary file. See Section 8. |
| `concentric_layer.read_binary` | Function to read in a layer from a binary output file. |
| `constants.h` | Header file for `CONST` namespace. |
| `CONST` | `CONST` namespace. Used to store values not used in standard C++ libraries. |
| `EOS_class.h` | Header file for `EOS` class. |

continued on next page

Table 5 – continued from previous page

| | |
|---|---|
| `EOS` | `EOS` class. For description see Section 4.4. |
| `EOS_functions.cc` | |
| `EOS.read_EOSfile` | Read in a HERCULES EOS file into the `EOS` class. |
| `EOS.write_binary` | Function to write out a layer to a binary output file. See Section 8. |
| `EOS.read_binary` | Function to read in a layer from a binary output file. |
| `EOS.calc_rho` | Calculate the density from a given pressure point by interpolating the EOS points. |
| `general_functions.h` | Header file for miscellaneous functions. |
| `integrate` | See below. |
| `general_functions.cc` | |
| `integrate` | Function to integrate an array with equally spaced points between two given elements. Uses the most accurate method available for the number of points: Bode's rule; Simpson's 3/8ths rule; Simpson's rule; or the extended Simpson's rule [Ziegel et al., 2007]. |
| `header.h` | Main header file. |
| `initialization` | See below. |
| `itterate` | See below. |
| `hercules.cc` | |
| `main` | Root for HERCULES. Initializes the data structures, calls both `initialisation` and `itterate` and deals with the timing of the code. |
| `initialisation.cc` | |

continued on next page

Table 5 – continued from previous page

| | |
|---|---|
| `initialisation` | Function that orchestrates the initialization of the structure depending on the start flag (see Section 3.2). |
| `io_routines.h` | Input-output header file. |
| `read_input` | See below. |
| `write_binary_structure` | See below. |
| `read_binary_structure` | See below. |
| `io_routines.cc` | |
| `read_input` | Read the HERCULES input file (see Section 6). |
| `write_binary_structure` | Writes the structure out to a binary output file. This function relies on the class specific `write_binary` functions. |
| `read_binary_structure` | Read in a structure from a binary output file. This function relies on the class specific `read_binary` functions. |
| `itterate.cc` | |
| `itterate` | Function that orchestrates the iteration section of the code. This includes the shape (see Section 3.3) and mass/AM control iterations (see Sections 3.5 and 3.6). |
| `parameters_class.h` | Header file for parameters class. |
| `parameters` | For description of class see Section 4.1. |
| `parameters_functions.cc` | |
| `parameters.write_binary` | Function to write out the `parameter` class to a binary output file. See Section 8. |
| `parameters.read_binary` | Function to read in a `parameter` class from a binary output file. |
| `planet_class.h` | Header file for planet class. |
| `planet` | For description of class see Section 4.2. |

continued on next page

Table 5 – continued from previous page

| planet_functions.cc | |
|---|---|
| planet.precalc_Plegendre | Function to precalculate the Legendre polynomials. |
| planet.initialise_ellipsoids | Initialization routine for constant density ellipsoids. |
| planet.calc_V | Function to calculate the gravitational potential at a point. Requires calculation of $\bar{M}_i$, $\bar{N}_{i,2k}(\tilde{\xi}_i(\mu_j),\mu_j)$, $\bar{K}_{i,2k}(\tilde{\xi}_i(\mu_j),\mu_j)$, $\bar{P}_{2k}$ and $\bar{J}_{i,2k}$ for each spheroid (concentric_layer.calc_M, concentric_layer.precalc_Nbar, concentric_layer.precalc_Kbar and concentric_layer.calc_Js) before being called. |
| planet.calc_Q | Function to calculate the rotational potential at a point. |
| planet.calc_X | Function to calculate $U(r,\mu) - U(a_i,0)$ for the shape iteration. Requires calculation of $U(a_i,0)$ before being called. |
| planet.calc_Mtot | Function to calculate the present mass of the body. Requires calculation of $M_i$ for each spheroid (concentric_layer.calc_M) before being called. |
| planet.calc_Ltot | Function to calculate the total AM of the body. Requires calculation of $I_i$ for each spheroid (concentric_layer.calc_I) before being called. |
| planet.calc_Mout | Function to calculate $M^i_{\rm out}$ for each spheroid. Requires calculation of $M_i$ for each spheroid (concentric_layer.calc_M) before being called. |
| planet.calc_Lout | Function to calculate $L^i_{\rm out}$ for each spheroid. Requires calculation of $I_i$ for each spheroid (concentric_layer.calc_I) before being called. |
| planet.calc_Js | Function to calculate the gravitational moments, $J_{2k}$, for the whole body. Requires calculation of $J_{i,2k}$ for each spheroid (concentric_layer.calc_Js) before being called. |
| planet.write_binary | Function to write out the structure to a binary output file. See Section 8. |
| planet.read_binary | Function to read in a structure from a binary output file. |
| **vector_operations.h** | Header file for vector function namespace VecDoub. |
| VecDoub.VecMultiply | See below. |

continued on next page

Table 5 – continued from previous page

| | |
|---|---|
| `VecDoub.VecDivide` | See below. |
| `VecDoub.ScalMultiply` | See below. |
| `VecDoub.VecExp` | See below. |
| `VecDoub.VecSum` | See below. |
| `VecDoub.VecPow` | See below. |
| `VecDoub.VecAdd` | See below. |
| `VecDoub.VecSubtract` | See below. |
| `VecDoub.VecExponent` | See below. |
| `VecDoub.ScalAdd` | See below. |
| `VecDoub.VecGrad2` | See below. |
| **`vector_operations.cc`** | |
| `VecDoub.VecMultiply` | A function to multiply two vectors of doubles element-wise. |
| `VecDoub.VecDivide` | A function to divide two vectors of doubles element-wise. |
| `VecDoub.ScalMultiply` | A function to multiply all the elements in a vector of doubles by a single scalar. |
| `VecDoub.VecExp` | A function to take the exponential of a vector of doubles element wise i.e. $e^{x_i}$. |
| `VecDoub.VecSum` | A function to sum an array of doubles. |
| `VecDoub.VecPow` | A function to take the elements of a vector of doubles to a given power ($\alpha$), i.e., $x_i^\alpha$. |
| `VecDoub.VecAdd` | A function to add two vectors of doubles element-wise. |
| `VecDoub.VecSubtract` | A function to subtract one vector of doubles from another element-wise. |
| `VecDoub.VecExponent` | A function to take a vector of doubles and put each element as the exponent of a constant to create a new vector of doubles i.e. $c^{x_i}$. |
| `VecDoub.ScalAdd` | Add a scalar to each element of a vector of doubles. |
| `VecDoub.VecGrad2` | Take the gradient of an unevenly spaced vector of doubles to second order in d$x$. |

## 6 Description of input

HERCULES has three forms of potential input: a HERCULES input file; an EOS file; and a HERCULES structure start file. In this section, we outline the structure of each of the input files.

## 6.1 Main input file

The main input file for HERCULES, `HERCULES_input.txt`, is read from the directory in which HERCULES is run. This file contains the various parameters that HERCULES needs to initialize a structure and run. The file is divided into three segments demarcated by `#` signs. The top section is a header containing notes for the user which can be as long as desired and only ends at the first set of `#`. The next section contains the run parameters. The names of the variables are only given for reference and so the order of the parameters in the file is crucial. Spaces between the name, the = sign, the value for the parameter and the comments are required. For multiple value parameters the brackets and gaps between values are required. The last section begins at the second set of `#` and contains information about the structure of the planet to be modeled by HERCULES. The requirements are the same as in the parameters section.

In Table 6 we describe the parameters that must be provided in the input file.

Table 6: Description of the HERCULES input file.

| Name | Symbol | Description |
|---|---|---|
| Run parameters | | |
| `run_name` | | Name of the run. All output files are labelled with this run name and a string describing the parameters of the run (see Section 8). |
| `nint_max` | $n_{\mathrm{int}}$ | Maximum number of iterations for both the main iteration and the mass and AM iteration. |
| `toll` | $\chi_{\mathrm{toll}}$ | Desired value of the convergence criterion for both the main iteration and the mass and AM iteration. |
| `xi_nint_max` | $n^{\xi}_{\mathrm{int}}$ | Maximum number of iterations allowed for each point in the shape iteration. |
| `xi_toll` | $\chi^{\xi}_{\mathrm{toll}}$ | Desired value of convergence criterion for the shape iteration. |
| `dxi` | $\delta\xi$ | Step size for calculation of the potential gradient in the shape iteration. Code is generally insensitive to exact value. |
| `flag_start` | $\mathcal{F}_{\mathrm{start}}$ | Start flag (see Section 3.2). |
| `start_file` | | HERCULES structure file to use to initialize the structure if $\mathcal{F}_{\mathrm{start}} = 1$. Unused otherwise. |
| `flag_Mconc` | $\mathcal{F}_{\mathrm{Mconc}}$ | Mass control flag (see Section 3.5). |
| `flag_Lconc` | $\mathcal{F}_{\mathrm{Lconc}}$ | AM control flag (see Section 3.6). |

continued on next page

Table 6 – continued from previous page

| | | |
|---|---|---|
| `omega_param` | $\Omega_i$ | Array of parameters used to set the angular velocity profile if $\mathcal{F}_{\text{Lconc}} = 1$. |
| `flag_iter_print` | $\mathcal{F}_{\text{print}}$ | Flag for whether to print out the structure at every iteration, 1 for yes and 0 for no. |
| Planet properties | | |
| `aspect` | $\mathcal{A}$ | The initial aspect ratio used for the structure if $\mathcal{F}_{\text{start}} = 0$. |
| `Nlayers` | $N_{\text{lay}}$ | Total number of layers/spheroids. |
| `Nmaterial` | $N_{\text{mat}}$ | Number of material layers. |
| `Nmaterial_lay` | $N_{\text{lay}}^{\text{mat}}$ | Array of the number of layers in each material layer. The values must be in a form { $N_{\text{lay}}^0$ , $N_{\text{lay}}^1$ , ...} with the upper most material layer first. |
| `Nmaterial_amax` | $a_{\text{mat}}$ | The outer equatorial radii of each material layer used in initializing the planet if $\mathcal{F}_{\text{start}} = 0$. Must be given in the form { $a_{\text{mat}}^0$ , $a_{\text{mat}}^1$ , ...} with the upper most material layer first. Units [m]. |
| `material_files` | | An array of the EOS files to be read in for each material. Must be given in the form { `file0 , file1 , ...`} with the upper most material layer first. The space between the file and the comma is required. See Section 6.2 for a description of EOS files. |
| `Mass` | $M_{\text{tar}}^i$ | An array of the desired masses for each material layer. The masses are used in many initialization and mass control routines. Units [kg]. |
| `ref_rho` | $\rho_{\text{ref}}$ | The reference density for the structure. This is used if $\mathcal{F}_{\text{Mconc}} = 0$ and $\mathcal{F}_{\text{start}} = 0$ to define a constant density for layers in the structure. |
| `kmax` | $k_{\text{max}}$ | The maximum $k$ to use in calculating the structure. This corresponds to a maximum spherical harmonic degree of $2k_{\text{max}}$. |
| `Nmu` | $N_\mu$ | The number of $\mu$ points to describe each surface. |
| `omega_rot` | $\omega_{\text{rot}}$ | The corotating angular velocity of the planet, used to set the rotation rate if $\mathcal{F}_{\text{start}} = 0$ and $\mathcal{F}_{\text{Lconc}} = 0$. |
| `Ltot` | $L_{\text{tot}}$ | Total angular momentum of the planet. Units [kg m$^2$ s$^{-1}$]. |
| `pmin` | $p_{\text{min}}$ | Pressure at the outside of the outermost surface of the structure. Units [Pa]. |

## 6.2 EOS files

In the current version of HERCULES, the code accepts EOS with a one to one mapping of density to pressure. EOS files are ASCII files structured as follows. The top section is a header which can be of arbitrary length until the first # character. The number given after this character dictates the type of EOS and how the rest of the file is read. We detail the different options in Table 9.

Table 7: Description of EOS options

| Option | Description |
|---|---|
| 0 | This is the standard linear interpolation table option. The main part of the EOS file must contain a line of column headers and then four columns in the order: density; pressure; temperature; entropy. All units must be in base SI. When calculating thermodynamic properties the EOS is interpolated linearly. Temperature and entropy are not presently used in the code but values must be present for accurate reading of the EOS. |
| 1 | This is a mixed log and linear interpolation table option. The main part of the EOS file must contain a line of column headers and then four columns in the order: density; pressure; temperature; entropy. All units must be in base SI. When calculating thermodynamic properties the EOS is interpolated linearly for densities above 3000 kg $m^{-3}$ but log-linearly at lower densities. This routine may give better results for low density material such as vapor. Temperature and entropy are not presently used in the code but values must be present for accurate reading of the EOS. |
| 2 | This option deals with EOS of the type used by Hubbard [Hubbard, 2013]. The main part of the EOS file must contain a line of column headers and then three columns in the order: density; pressure; temperature. All units are in log of centimeter-grams-seconds (cgs) base units. When calculating thermodynamic properties, the EOS is interpolated linearly. Temperature is not presently used in the code but values must be present for accurate reading of the EOS. |

### 6.3 HERCULES output files

For $\mathcal{F}_{\text{start}} = 1$, HERCULES can read in a HERCULES output file for the initial structure. The file is of the same format as that which is output by the code (see Section 8). A separate structure is created to store the read in structure and the parameters that are required from it are then transferred to the structure to be used in the calculation. This is memory intensive, but does allow select transfer of planetary properties, such as being able to change the target mass or AM.

## 7 Running the code

After compilation (see Section 9) HERCULES is run by simply typing `.HERCULESv1.0` in the command line. In the directory the code is run there should be an input file `HERCULES_input.txt` with the desired parameters for the code (Section 6). An `Output` directory should be created in the run directory.

## 8 Description of output

HERCULES outputs information in two ways. Firstly, during the calculation the code outputs information about the iterative steps to the command line. This allows the user to keep track of the progress of the code. Secondly, the code outputs the structure of the planet as a binary file. This allows for post-run analysis of the structure.

### 8.1 Inline output

The inline output that is produced is self explanatory. During the initialization sequence, lines are printed describing the stages the code has reached and which options have been taken. During the iteration procedure, after completing each iteration, the progress in convergence and basic properties of the planet are printed. Convergence information is printed for both the main iteration and the mass/AM iteration. The properties that are printed out differ between the two iterations and are described in header lines in the inline output at the start of each iteration.

### 8.2 HERCULES structure files

HERCULES outputs the final, and optionally the intermediate, structures as binary files. These files contain the contents of the `parameters` and `planet` classes (and all subsidiary classes) for that model. The routine `write_binary_structure` is responsible for producing these files. HERCULES always outputs a structure file for the initial structure and after the completion of the main iteration. It is also possible to request the output of a structure file after each main iteration by setting $\mathcal{F}_{\text{print}} = 1$.

Currently, output files are named as follows:

$\textit{run\_name}$_L$L_{\rm tot}^{\rm tar}$_N$N_{\rm lay}$_Nm$N_\mu$_k$k_{\rm max}$\_f$\mathcal{F}_{\rm start}\mathcal{F}_{\rm Mconc}\mathcal{F}_{\rm Lconc}$_p$p$[bar]_l$\Omega_0$_$\Omega_1$_$\Omega_2$_*output*

where the red sections are obtained from the parameters of the run. The term *output* is either set to `0` for the initial structure, the iteration number after which the structure is output, or `final` if output after the main iteration has finished. The output files are put in a directory `Output` in the directory from which HERCULES is run. The output directory must be created before running the code.

## 9 Compiling notes

This section contains notes on compiling HERCULES. Examples of make files using different compilers are also given. Issues encountered compiling on different architectures are also noted and solutions provided where possible.

### 9.1 Required C++ libraries

Other than the standard C++ libraries, HERCULES requires the *boost* special function maths library (`http://www.boost.org/doc/libs/1_60_0/libs/math/doc/html/special.html`). The *boost* libraries are to efficiently calculate the Legendre polynomials and could be replaced by another library if preferred.

### 9.2 Example make files

We have compiled HERCULES using both the GNU (`https://gcc.gnu.org/`) and intel (`https://software.intel.com/en-us/c-compilers`) C++ compilers. An example make file, `Makefile`, is provided with the HERCULES source code. The main text of the make file is the same in both cases but the flags relevant to the two compilers are given below. Using these make files, HERCULES can be compiled using the `make` command.

#### 9.2.1 GNU compilers

I used the following flags for the GNU C++ compiler to compile HERCULES on a late-2013 iMac running macOS Sierra version 10.12.6. Note that the `-l` flag is the location of the boost libraries.

```
CFLAGS = -o1 -ansi -g -Wall -I /Applications/C_libraries/boost_1_59_0
LIBS = -l stdc++ -lm
CXX = g++
```

### 9.2.2 Intel compilers

The followings flags for the intel C++ compiler have been used to successfully compile HERCULES on the Harvard Odyssey cluster (`https://www.rc.fas.harvard.edu/odyssey/`). Note that a flag to the location of the boost libraries might be required on other systems.

```
CFLAGS = -O3 -std=c++11
LIBS =
CXX=icc
```

The build on Odyssey required some minor alterations to the code to run with the new compiler. The corrections were well flagged by the compiler and simple to make.

## 10 Code testing

Lock and Stewart [2017] compared the results of HERCULES to the analytical solution of a Maclaurin spheroid, and to previous techniques for calculating the structure of more complex bodies including: small-parameter expansions, the CMS method [Hubbard, 2013], and smoothed-particle hydrodynamics simulations. The results calculated by HERCULES compared well to these other methods and the analytical results. Lock and Stewart [2017] also explored the dependence of the structure calculated by HERCULES on parameters such as the number of concentric spheroids used, $N_{\rm lay}$, the number of points used to describe each surface, $N_\mu$, and the bounding pressure.

Here, we provide additional sensitivity tests for parameters not considered in Lock and Stewart [2017] and explore how the runtime of the code varies for different sets of parameters. For these tests we considered Earth-like bodies with isentropic cores and mantles and AM similar to that of the present-day Earth-Moon system, $L_{\rm EM}$. The mantles and cores were modeled as pure forsterite and iron respectively using M-ANEOS EOS [Melosh, 2007; Canup, 2012]. The specific entropies of the mantle and core were chosen to be 6 kJ K$^{-1}$ kg$^{-1}$ and 1.5 kJ K$^{-1}$ kg$^{-1}$ respectively, which corresponds to a hot, substantially vaporized body (see Figure 4 in Lock and Stewart [2017]). The parameters used for these tests, unless otherwise noted, are given in Table 8.

We encourage any users of HERCULES to perform sensitivity tests for their specific problem. The sensitivity and performance tests here are intended only as a guide.

Table 8: HERCULES parameters for parameter and performance testing. Given in the same order as in the input file.

| Parameter | Value | Units |
|---|---|---|
| $n_{\text{int}}$ | 200 | |
| $\chi_{\text{toll}}$ | $10^{-5}$ | |
| $n^{\xi}_{\text{int}}$ | 200 | |
| $\chi^{\xi}_{\text{toll}}$ | $10^{-10}$ | |
| $\delta\xi$ | $10^{-2}$ | |
| $\mathcal{F}_{\text{start}}$ | 0 | |
| $\mathcal{F}_{\text{Mconc}}$ | 2 | |
| $\mathcal{F}_{\text{Lconc}}$ | 1 | |
| $\Omega_i$ | $\{ 1, 0, 1.5 \}$ | |
| $\mathcal{A}$ | 0.8 | |
| $N_{\text{lay}}$ | 50 | |
| $N^{\text{mat}}_{\text{lay}}$ | $\{ 45, 5 \}$ | |
| $a_{\text{mat}}$ | $\{ 2.3 \times 10^{7}, 4 \times 10^{6} \}$ | m |
| $M^{i}_{\text{tar}}$ | $\{ 4.1729748 \times 10^{24}, 1.81499 \times 10^{24} \}$ | kg |
| $\rho_{\text{ref}}$ | $10^{3}$ | kg m$^{-3}$ |
| $k_{\text{max}}$ | 6 | |
| $N_{\mu}$ | 400 | |
| $L_{\text{tot}}$ | $3.53 \times 10^{34}$ | kg m$^{2}$ s$^{-1}$ |
| $p_{\text{min}}$ | $10^{6}$ | Pa |

## 10.1 Parameter testing

Figure 4 shows the fractional absolute error of bodies calculated with different convergence tolerances for the main iteration, $\chi_{\rm toll}$. As expected, the difference in key properties between each run and a reference run with a high $\chi_{\rm toll}$ is of the same order as the chosen $\chi_{\rm toll}$.

Figure 5 shows the fractional absolute error of bodies calculated with different convergence tolerances for the shape iteration, $\chi^{\xi}_{\rm toll}$. The results are weakly dependent on $\chi^{\xi}_{\rm toll}$ for the range of parameters we considered.

Figures 6 and 7 show the fractional absolute error of bodies calculated including different numbers of spherical harmonic degrees, $2k_{\rm max}$, for two different numbers of surface points. The results for the body we consider are not strongly dependent on the maximum spherical harmonic degree above $k_{\rm max} = 6$.

Figure 8 shows the fractional absolute error of bodies calculated using different $\delta\xi$. The results are very weakly dependent on $\delta\xi$ for the range of parameters we considered.

Figures 9 and 10 show the fractional absolute error of bodies calculated using different initial structures, determined by the initial radius of the starting body $a_0$ and the initial aspect ratio $\mathcal{A}$. The results are somewhat dependent on the starting structure. This is likely due to the fact that for different starting structures the final radial distribution of the spheroids varies.

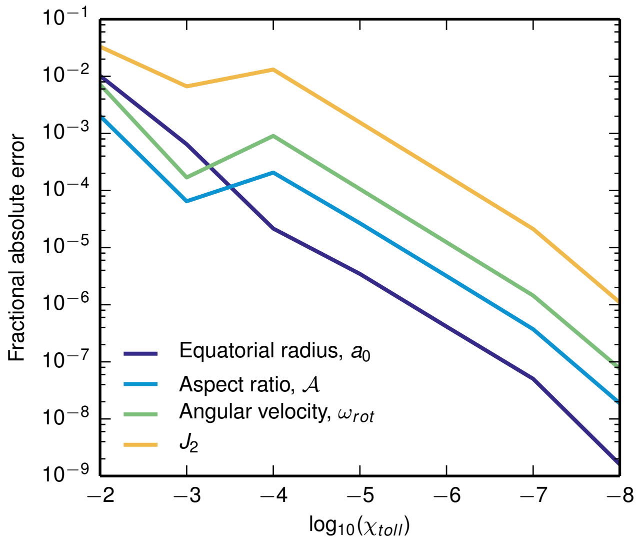

Figure 4: Testing of sensitivity to $\chi_{\rm toll}$. Fractional difference relative to a reference run with $\chi_{\rm toll} = 10^{-9}$.

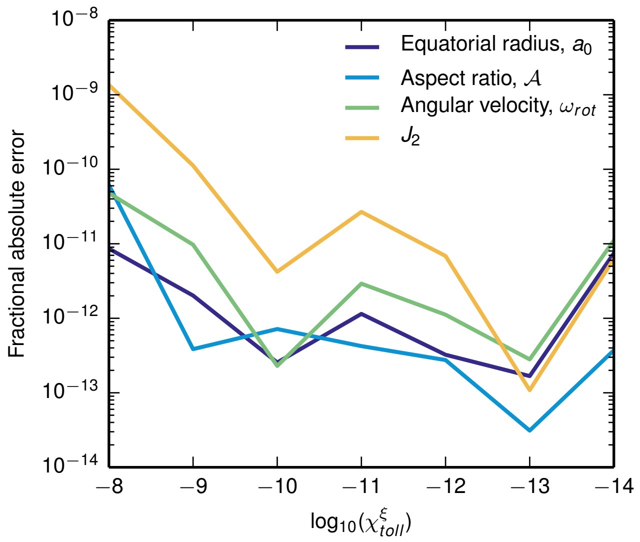

Figure 5: Testing of sensitivity to $\chi^{\xi}_{\rm toll}$. Fractional difference relative to a reference run with $\chi^{\xi}_{\rm toll} = 10^{-15}$.

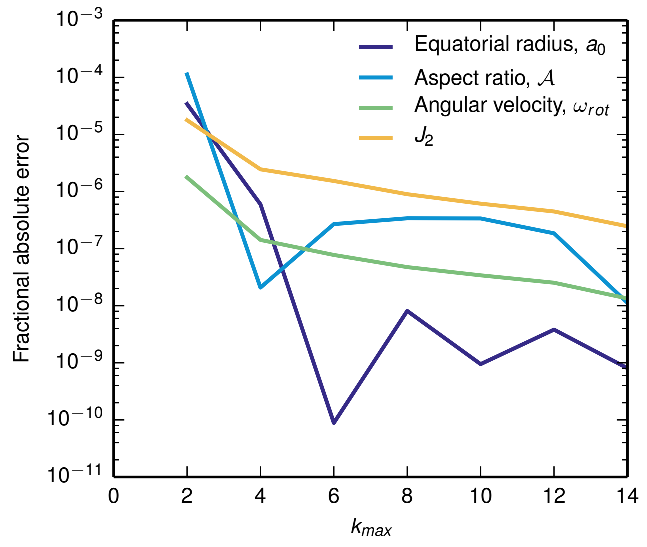

Figure 6: Testing of sensitivity to $k_{\max}$. Fractional difference relative to a reference run with $k_{\max} = 16$.

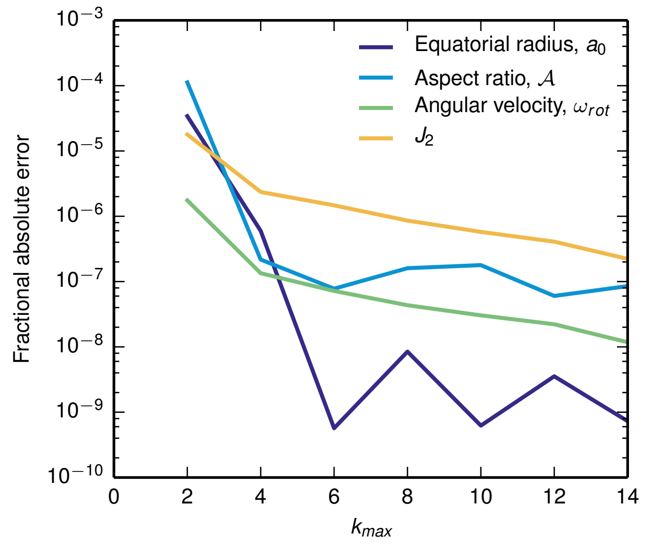

Figure 7: Testing of sensitivity to $k_{\max}$ for a body calculated using $N_\mu = 2000$. Fractional difference relative to a reference run with $k_{\max} = 16$.

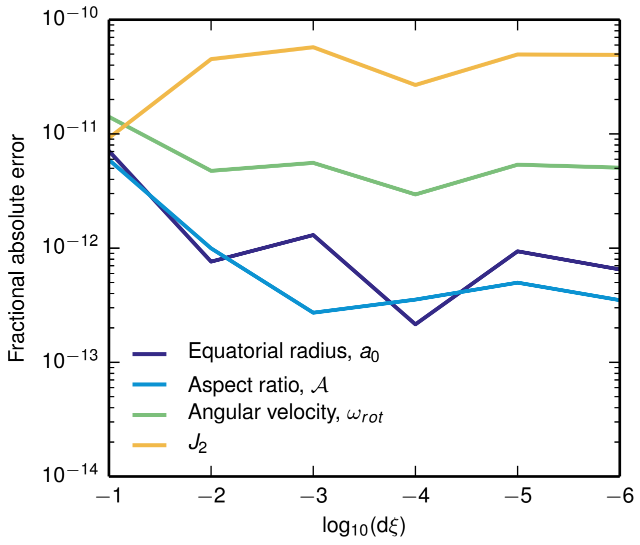

Figure 8: Testing of sensitivity to $\delta\xi$. Fractional difference relative to a reference run with $\delta\xi = 10^{-7}$.

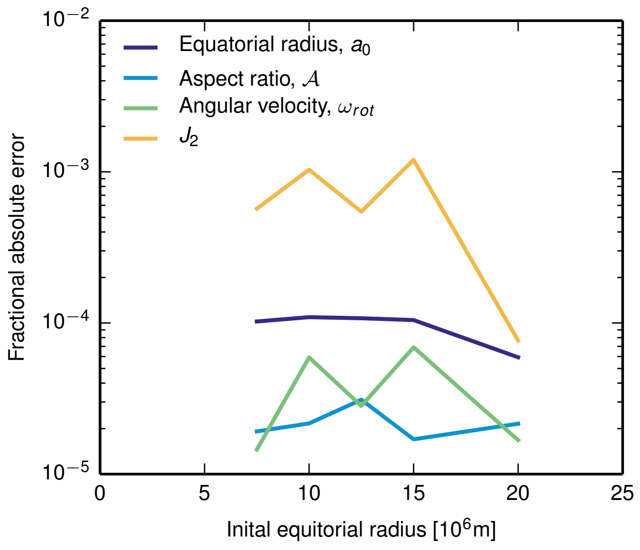

Figure 9: Testing of sensitivity to the initial $a_0$. Fractional difference relative to a reference run with an initial $a_0 = 25 \times 10^6$ m.

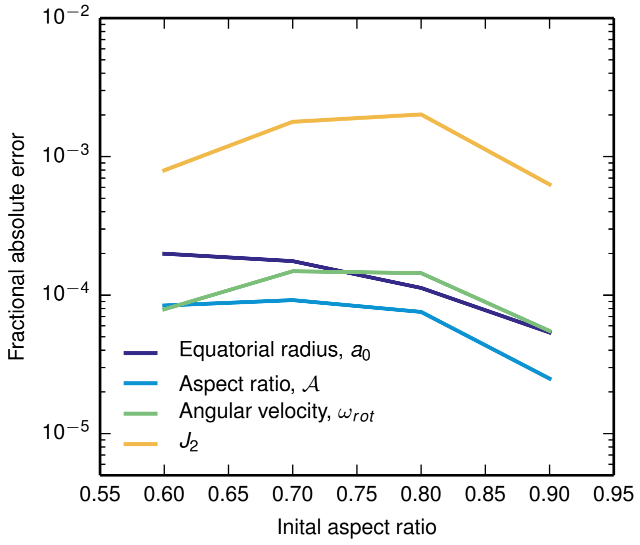

Figure 10: Testing of sensitivity to the initial $\mathcal{A}$. Fractional difference relative to a reference run with an initial $\mathcal{A} = 1$.

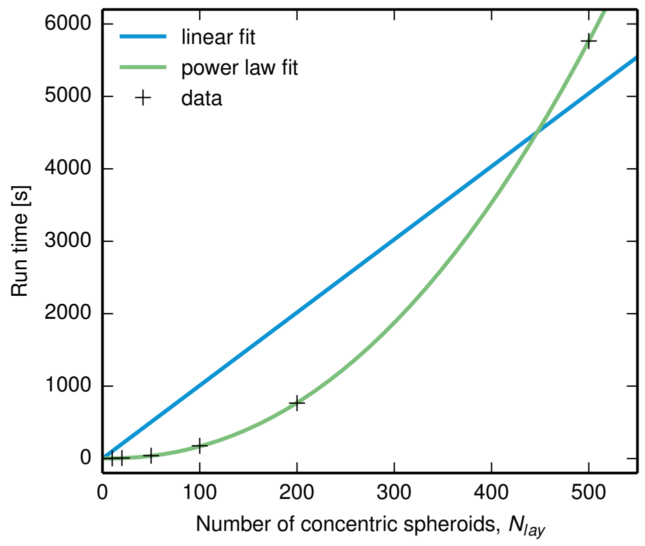

Figure 11: Dependence of run time on $N_{\text{lay}}$. The data is given in black, with a linear fit in blue and a power law fit with an exponent of 2.2 in green.

## 10.2 Performance testing

Figures 11, 12, 13, and 14 show the run time for bodies calculated using different: numbers of concentric spheroids, $N_{\text{lay}}$; numbers of points to describe each surface, $N_\mu$; main iteration tolerances, $\chi_{\text{toll}}$; and maximum spherical harmonic degrees, $2k_{\text{max}}$. The run time of HERCULES scales relatively well in all cases.

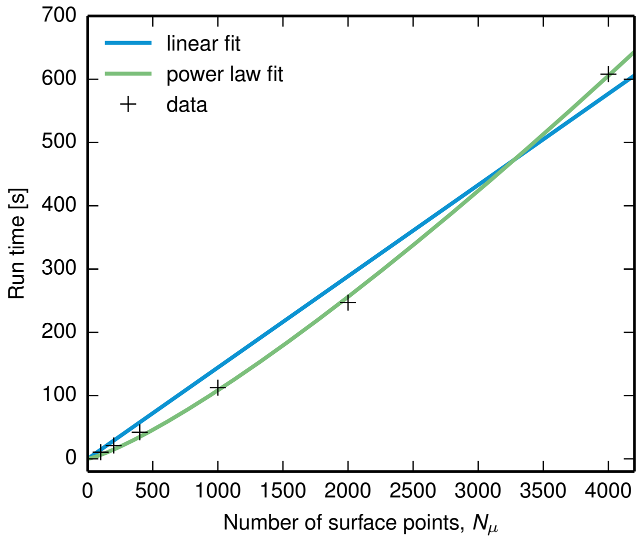

Figure 12: Dependence of run time on $N_\mu$. The data is given in black, with a linear fit in blue and a power law fit with an exponent of 1.24 in green.

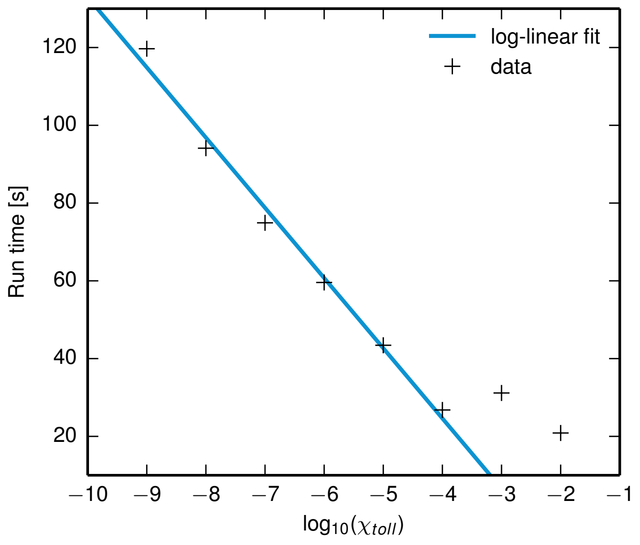

Figure 13: Dependence of run time on $\chi_{\mathrm{toll}}$. The data is given in black with a linear fit for the data below $\log_{10}(\chi_{\mathrm{toll}}) = -4$ in blue.

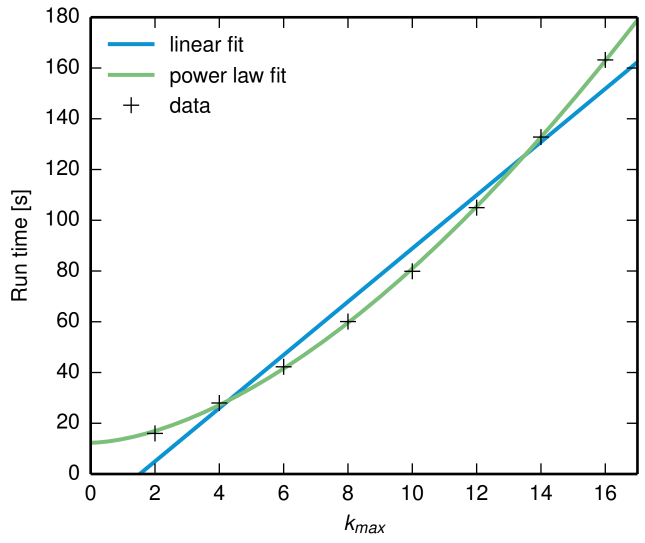

Figure 14: Dependence of run time on $k_{\max}$. The data is given in black, with a linear fit in blue and a power law fit with an exponent of 1.67 in green.

## 11 Supporting codes and scripts

The release of HERCULES contains a number of scripts to help users analyze the output from the code. Details are given below. The principal developer of HERCULES has a strong preference for `python` as a scripting language as it is open source and has a large user community. As a result, the supporting scripts provided in this and future releases of HERCULES will be in `python`.

### 11.1 `python` scripts

Table 9: Python scripts included with HERCULES

| Script | Description |
|---|---|
| `HERCULES_structures.py` | This script contains mirror structures to the `parameters`, `planet`, `concentric_layers`, and `EOS` classes in the HERCULES codes. It contains functions to read and write HERCULES binary output files, allowing users to quickly access the results of HERCULES calculations. |

## 12 Tutorial

The zip file `Tutorials.zip` contains an introductory tutorial for HERCULES. The file `HERCULES_tutorial_v1.0.pdf` contains step-by-step instructions for the tutorial. The necessary example input files, analysis scripts and EOS files are found within `Tutorials.zip`. The scripts for the tutorial are in `python`. The tutorial is still a work in progress so please report any errors to the contacts given above.

## 13 Version updates

This section records the major changes between different versions of the code. Where relevant, the details of each of these changes have been noted in the relevant section of the user manual, but this section provides a summary and a list of known issues with each version. A summary is also given at the top of `hercules.cc`.

### 13.1 Version 1.0

Version 1.0 is the first version of the code implementing the routines as described in this module.

### 13.1.1 Known issues

Although steps have been taken to catch as many errors as possible and give coherent information to the user about these errors, some are more subtle or the cause has not been identified. Here we give a list of known issues with running HERCULES as a record to help users and to guide future development.

- **Mass and/or AM not conserved:** Occasionally there is a small difference between the desired mass and/or AM of the body and its actual properties. This is due to the mass and AM iteration exiting after a forced re-scaling of the spheroids in one or more material layers (Section 3.5). There is currently no flag in the code to stop such behavior. The error is typically small and can be negated by using an initial structure closer to the equilibrium structure.
- **Failure to start with no error flag:** Occasionally the code will begin to run but not get past the initialization steps. This is usually associated with $\mathcal{F}_{\rm start} = 1$ and occurs if the start file does not exist. Check the given start file location. A similar error is found if the EOS file does not exist.
- **Failure after first few iterations:** This failure is often associated with properties of the planets being returned as `NAN`s or `INF`s. This error is usually solved by using a different starting structure, closer to the equilibrium structure. The actual cause of this error is not yet known.
- **Drift in polar points:** In some cases it has been observed that the surface point that defines the pole can drift away from the rotation axis by about 100 m for Earth-mass bodies. Cause unknown. Error introduced unknown.
- ***T* and *S* not calculated:** The $T$ and $S$ of each layer are not calculated in the code. These values must be extracted from the EOS in post-processing.

## 14 Papers published using HERCULES

Simon J. Lock and Sarah T. Stewart. The structure of terrestrial bodies: Impact heating, corotation limits, and synestias. *Journal of Geophysical Research: Planets*, 122(5):950–982, may 2017. ISSN 21699097. doi: 10.1002/2016JE005239. URL `http://doi.wiley.com/10.1002/2016JE005239`

## References

Robin M Canup. Forming a Moon with an Earth-like composition via a giant impact. *Science*, 338(6110):1052–5, 2012. ISSN 1095-9203. doi: 10.1126/science.1226073. URL `http://www.ncbi.nlm.nih.gov/pubmed/23076098`.

W. B. Hubbard. High-precision Maclaurin-based models of rotating liquid planets. *The Astrophysical Journal*, 756(1):L15, 2012. ISSN 2041-8205. doi: 10.1088/2041-8205/756/1/L15. URL `http://iopscience.iop.org/article/10.1088/2041-8205/756/1/L15`.

W.B. Hubbard, G. Schubert, D. Kong, and K. Zhang. On the convergence of the theory of figures. *Icarus*, 242:138–141, 2014. ISSN 00191035. doi: 10.1016/j.icarus.2014.08.014. URL `http://www.sciencedirect.com/science/article/pii/S001910351400428X`.

WB B. Hubbard. Concentric Maclaurin spheroid models of rotating liquid planets. *The Astrophysical Journal*, 768(1):43, 2013. ISSN 0004-637X. doi: 10.1088/0004-637X/768/1/43. URL `http://iopscience.iop.org/article/10.1088/0004-637X/768/1/43http://iopscience.iop.org/0004-637X/768/1/43`.

Dali Kong, Keke Zhang, and Gerald Schubert. On the gravitational fields of Maclaurin spheroid models of rotating fluid planets. *The Astrophysical Journal*, 764(1):67, 2013. ISSN 0004-637X. doi: 10.1088/0004-637X/764/1/67. URL `http://iopscience.iop.org/article/10.1088/0004-637X/764/1/67`.

Simon J. Lock and Sarah T. Stewart. The structure of terrestrial bodies: Impact heating, corotation limits, and synestias. *Journal of Geophysical Research: Planets*, 122(5):950–982, may 2017. ISSN 21699097. doi: 10.1002/2016JE005239. URL `http://doi.wiley.com/10.1002/2016JE005239`.

Simon J. Lock, Sarah T. Stewart, Michail I. Petaev, Zoë M. Leinhardt, Mia T. Mace, Stein B. Jacobsen, and Matija Ćuk. The origin of the Moon within a terrestrial synestia. *Journal of Geophysical Research: Planets*, 123(4):910–951, 2018. ISSN 21699097. doi: 10.1002/2017JE005333. URL `http://doi.wiley.com/10.1002/2017JE005333`.

H. J. Melosh. A hydrocode equation of state for SiO2. *Meteoritics & Planetary Science*, 42(12):2079–2098, 2007. ISSN 10869379. doi: 10.1111/j.1945-5100.2007.tb01009.x. URL `http://doi.wiley.com/10.1111/j.1945-5100.2007.tb01009.x`.

Eric Ziegel, William Press, Brian Flannery, Saul Teukolsky, and William Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd editio edition, 2007. ISBN 0521431085. doi: 10.2307/1269484. URL `http://www.jstor.org/stable/1269484?origin=crossref`.

## 15 Tables of variables

Table 10: Variables used in the theoretical background (Section 2).

| Symbol | Definition |
|---|---|
| Subscripts and superscripts. | |
| $I$ | Denotes regime 1, see Section 2.1.1. |
| $II$ | Denotes regime 2, see Section 2.1.2. |
| $III$ | Denotes regime 3, see Section 2.1.3. |
| $IV$ | Denotes regime 4, see Section 2.1.4. |
| $(i)$ | Denotes the layer/spheroid/material layer $i$. |
| $(i, 2k)$ | Denotes the layer/spheroid $i$ and the harmonic degree $2k$. |
| $'$ | Prime usually donates dummy variable for integration. |
| General variables. | |
| $G$ | Gravitational constant. |
| $k$ | Generally used to indicate spherical harmonic degree. |
| $l$ | Generally used to indicate spherical harmonic degree. |
| $\mu$ | The cosine of the angle of the position vector measured from the rotation axis: $\mu = \cos(\theta)$. |
| $\omega$ | Angular velocity. |
| $\boldsymbol{\omega}$ | The rotation vector. |
| $P_l$ | The Legendre polynomial of degree $l$. |
| $\phi$ | The angle of the position vector in the equatorial plane relative to a fixed position. |
| $Q$ | Rotational potential. |
| $r$ | Radius from the origin. |
| $\boldsymbol{r}$ | Position vector $(r, \mu, \phi)$, or $(r, \mu)$ if axisymmetry is assumed. |
| $\theta$ | Angle of the position vector measured from the rotation axis. |
| $U$ | Total potential: $U = \Phi + Q$. |
| $\Phi$ | Gravitational potential. |
| $\mathcal{V}$ | Integration volume. |
| Single body. | |
| $a$ | Equatorial radius (for a spheroid is equivalent to semi-major axis). |
| $\mathcal{A}$ | Aspect ratio of body: $\mathcal{A} = b/a$. |
| $b$ | Polar radius (for a spheroid is equivalent to semi-minor axis). |
| $J_l$ | Gravitational moment of degree $l$. For definition see Equation 16. |

continued on next page

Table 10 – continued from previous page

| | |
|---|---|
| $K_l(\xi, \mu_r)$ | For definition see Equations 14 and 15. |
| $M$ | Mass of body. For definition see Equation 8. |
| $\mu_r$ | $\mu$ at which the radius of the observation intersects the surface of the body i.e. $\tilde{r}(\mu_r) = r$. |
| $N_l(\xi, \mu_r)$ | For definition see Equations 9, 10 and 11. |
| $\tilde{r}(\mu)$ | Radius of the surface of the body at $\mu$. |
| $\rho$ | Density of the body. |
| $\xi$ | Normalized radius: $\xi = r/a$. |
| $\tilde{\xi}(\mu)$ | Normalized radius of the surface of the body at $\mu$: $\tilde{\xi} = \tilde{r}/a$. |

Concentric layer model.

| | |
|---|---|
| $a_i$ | Equatorial radius of spheroid/layer $i$. |
| $\mathcal{A}_i$ | Aspect ratio of spheroid $i$: $\mathcal{A}_i = b_i/a_i$. |
| $b_i$ | Polar radius of spheroid/layer $i$. |
| $i_I$ | The lowermost spheroid for which $r < b_i$ for a given $r$. |
| $i_{IV}$ | The uppermost spheroid for which $r > a_i$ for a given $r$. |
| $J_{i,2k}$ | Scaled gravitational moment of degree $2k$ for spheroid $i$. For definition see Equation 35. |
| $J_{2k}$ | Gravitational moment of degree $2k$ for whole body. |
| $K_{i,2k}(\xi_i, \mu_r)$ | For definition see Equations 33 and 34. |
| $M_i$ | Mass of concentric spheroid $i$. For definition see Equation 36. |
| $N$ or $N_{\text{lay}}$ | The number of layers/spheroids. |
| $N_{i,2k}(\xi_i, \mu_r)$ | For definition see Equations 30, 31 and 32. |
| $\tilde{r}_i(\mu)$ | Radius of the surface of spheroid $i$ at $\mu$. |
| $\rho_i$ | 'Real' density of layer $i$. |
| $\delta\rho_i$ | Density of spheroid $i$. |
| $\Phi_i(r, \mu)$ | The gravitational potential at a point $(r, \mu)$ due to spheroid $i$. |
| $\xi_i$ | Radius normalized to the equatorial radius of spheroid $i$: $\xi_i = r/a_i$. |
| $\tilde{\xi}_i(\mu)$ | Radius of surface of spheroid $i$ at $\mu$ normalized to the equatorial radius of spheroid $i$: $\xi_i = \tilde{r}_i/a_i$. |