LOV3D

Version 2.0

M.Rovira-Navarro & Allard Veenstra

May 28, 2024

Contents

List of Code Listings

1 Introduction

`LOV3D` is a code developed to obtain the tidal response of bodies with lateral variations of interior properties. The tidal response is obtained using the spectral semi-analytical approach detailed in Rovira-Navarro et al. (2024).

This manual corresponds to release 2.0 of `LOV3D`. A previous version of the software that considers an interior structure consisting of (1) a solid rigid core, (2) surrounded by a liquid layer and (3) a solid envelope with laterally varying properties can be accessed in Rovira-Navarro (2023). Compared to the previous version LOV3D 2.0

- can compute the tidal response of a body with *N* internal layers.
- can compute the tidal of a body with an internal liquid layer (e.g., subsurface ocean) without the assumption of a rigid interior.
- can use parallelization to speed up computations.

Section 2 details the input, outputs of the code, Section 3 introduces the main subroutines and functions and Section 4 describes the sample script `example.m` containing a test-case to reproduce the results in Figure 7.

2 Inputs & Outputs

The inputs and outputs produced by the code are listed here. For individual functions, inputs and outputs are described in the header.

Inputs

- `Interior_Model` Structure containing information about the interior of the body. The interior is subdivided in *N* layers, each one with different properties given by `Inpertior_Model(ilayer)`. A sketch of the interior model is shown in Figure 1 Dimensional quantities are indicated with a 0 and non-dimensional without. Some values are assigned inside `get_rheology`. The structure contains the following fields.

```
% Mean properties:
    %Interior_Model(ilayer).R0: radius [km]
    %Interior_Model(ilayer).rho0: density [kg.m^{-3}]
    %Interior_Model(ilayer).mu0: shear modulus [Pa]
    %Interior_Model(ilayer).Ks0: bulk modulus, if not given, the
       model is assumed incompressible, [Pa]
    %Interior_Model(ilayer).eta0: viscosity, if not give, the body
        is assumed elastic [Pa.s]
    %Interior_Model(ilayer).MaxTime: normalized Maxwell time, it
       can be given instead of the viscosity, else it is computed
       [-]
    %Interior_Model(ilayer).ocean: (0) no ocean, (1) ocean
    %Interior_Model(ilayer).Gg0: gravitational constant (should be equal for all layers) [Nm^2kg^{-2}]
% Lateral variations, if not given, the model is assumed to be spherically-symmetric
% Lateral variations can be provided in three different formats
    %%% (1) in complex spherical harmonics:
        %Interio_Model(ilayer).mu_variable: shear modulus variations
            %mu_variable(:,1): degree of variation
            %mu_variable(:,2): order of variation
            %mu_variable(:,3): amplitude of the variation (mu_l^m/mu^0_0)
        %Interio_Model(ilayer).K_variable: bulk modulus variations
            %K_variable(:,1): degree of variation
            %K_variable(:,2): order of variation
            %K_variable(:,3):  amplitude of bulk modulus variations (K_l^m/K^0_0)
        %Interio_Model(ilayer).eta_variable: viscosity
            %eta_variable(:,1): degree of variation
            %eta_variable(:,2): order of variation
            %eta_variable(:,3):  amplitude of viscosity variations (eta_l^m/eta^0_0)
    %%% (2) in peak to peak variation amplitude wrt to the mean value (in percent)
        %Interio_Model(ilayer).mu_variable_p2p: shear modulus variations
            %mu_variable_p2p(:,1): degree of variation
            %mu_variable_p2p(:,2): order of variation
            %mu_variable_p2p:,3): amplitude of the variation (mu_l^m/mu^0_0)
        %Interio_Model(ilayer).K_variable_p2p: bulk modulus variations
            %K_variable_p2p(:,1): degree of variation
            %K_variable_p2p(:,2): order of variation
            %K_variable_p2p(:,3):  amplitude of bulk modulus variations (K_l^m/K^0_0)
        %Interio_Model(ilayer).eta_variable: viscosity
            %eta_variable_p2p(:,1): degree of variation
            %eta_variable_p2p(:,2): order of variation
            %eta_variable_p2p(:,3):  amplitude of viscosity variations (eta_l^m/eta^0_0)
    %%% (3) as a map, in which case get_rheology will transform it to
            % a spherical harmonics expansion. The maps can be generated by generating SPH_LatLon in SPH_Tools and
```

```
      % for efficient transformation should be evaluated in the lat lon grid specified there,
      % which depends on the maximum degree of the expansion.
      % IMPORTANT NOTICE: The map(s) should already be scaled by their relevant average value
      % (Interior_Model(ilayer).eta0, Interior_Model(ilayer).mu0 or Interior_Model(ilayer).Ks0) as:
          % Interior_Model(ilayer).mu_latlon/Interior_Model(ilayer).mu0
          % Interior_Model(ilayer).eta_latlon/Interior_Model(ilayer).eta0
          % Interior_Model(ilayer).K_latlon/Interior_Model(ilayer).Ks0
      % This ensures that the maps that you input have a mean of 1, which is what
      % the code expects, and that the actual fields have an average equal
      % to the prescribed average (Interior_Model(ilayer).XXX0)
      % Each of the input fields has the components
          %.lon:  longitude [IN DEGREES]
          %.lat:  latitude [IN DEGREES]
          %.z:    pxq grid where p stands for latitude and q for
          %.lmax: maximum harmonic degree
  % For both (1) and (2) the expansion in spherical harmonics should result in a real field (there should be both m>0 and m<0 components). If this is not the case and only >0 components are provided, the <0 component is computed

% Assigned in get_rheology
  %%% Mean properties:
    % Interior_Model(ilayer).R: normalized radius (normalized with Interior_Model(end).R0)
    % Interior_Model(ilayer).rho: normalized density  (normalized with Interior_Model(end).rho0)
    % Interior_Model(ilayer).mu: normalized shear modulus of the layer (normalized with Interior_Model(end).mu0)
    % Interior_Model(ilayer).Ks: normalized bulk modulus of layer (normalized with Interior_Model(end).mu0)
    % Interior_Model(ilayer).Ks: normalized viscosity of layer (normalized with Interior_Model(end).mu0*Forcing.T)
    % Interior_Model(ilayer).Gg: normalized gravitational constant (can also be given by user in model given in non-dimensional units)
```

```
        % Interior_Model(ilayer).gs: normalized gravity at Interior_Model(ilayer).R0
        % Interior_Model(ilayer).gs0: gravity at Interior_Model(ilayer).R0
        %Interior_Model(ilayer).muC: normalized complex shear modulus
        % Interior_Model(ilayer).mu00R: normalized real component of the shear modulus
         % Interior_Model(ilayer).rho0_av: averaged density at Interior_Model(ilayer).R0
        % Interior_Model(ilayer).rho_av: normalized averaged density at Interior_Model(ilayer).R0
     %%% Lateral variations:
        % Interior_Model(ilayer).rheology_variable: rheology variable
        % rheology_variable(:,1): degree of variation
        % rheology_variable(:,2): order of variation
        % rheology_variable(:,3): amplitude of bulk modulus variations (K_l^m/K^0_0)
        % rheology_variable(:,4): amplitude of complex shear modulus variations (\hat mu_l^m/mu^0_0(N))
```

- `Forcing` Structure containing information about the forcing.

```
% Forcing.Td: forcing period
% Forcing.n: degree of the forcing
% Forcing.m: order of the forcing
% Forcing.F: amplitude of the component
```

- `Numerics` Structure containing numerical information. Including the radial discretisation, cut-off for lateral variations and parallelization options.

```
%Numerics.Nlayers: number of layers, simply length(Interior_Model)
%Numerics.method: used for the radial discretization in set_boundary_indices. Four methods are possible
%   - variable: This method sets the number of radial points in each layer equal to Numerics.Nrbase. The variable name thus comes from the fact that the stepsize is variable.
%   - fixed: This method sets the total number of points equal to Nrbase. Same as for the variable method, the name fixed thus means that the stepsize is kept fixed. WARNING: This method changes the physical boundary locations to ensure that they occur at an integer number of points.
%    - combination: This method is a combination of the fixed and variable methods. Numerics.Nrbase serves as the base number of points per layer onto which an additional number of points is added based on the physical size of the layer. The total number
```

```
 of points will depend on the number of layers but is roughly given by (Nlayers+1)*Nrbase
%   - manual: This method allows the user to set the number of points per layer manually. In order for it to work an array needs to be provided as a varargin after putting 'manual' as a varargin as well. The syntax of the array needs to be: [0, points in layer #1, points in layer #2, etc]
%Numerics.Nrbase: used for the radial discretization  (see above)
%Numerics.Nr: total number of radial points (computed inside set_boundary_indices)
%Numerics.Nrlayer(layer): number of radial points per layer (computed inside set_boundary_indices)
 %Numerics.BCindices(layer): index of the layer point at the boundaries
%Numerics.perturbation_order: maximum order of the perturbation. Default 2
%Numerics.rheology_cutoff: determines which terms of the rheology are included (only relevant for viscoelastic). terms with log10(mu_n^m)-log10(mu_n^m(leading))>=-Numerics.rheology_cutoff are included. Default 0 (only leading terms)
%Numerics.Nenergy: maximum degree to which energy dissipation is expanded. Default 8.
%Numerics.load_couplings:
     % (1) load coupling coefficintes from a file that contains specifically the coupling coefficients for the rheology specied, if it does not exisit, compute it.
     % (2) load coupling coefficients from file that contains ALL couplings up to rheology variations of a certain degree, if such a file does not exist compute it (default)
% Numerics.parallel_sol:  Use a parfor-loop to call get_Love, either 0 or 1
% Numerics.parallel_gen:  Calculate potential coupling files and the propagation inside get_solution using parfor-loops, either 0 or 1
```

Outputs

- `Love` Love number spectra (see Rovira-Navarro et al., 2024, Eq. 29). Output of `get_Love`

```
% Love_Spectra.nf: degree of the forcing
% Love_Spectra.mf: order of the forcing
% Love_Spectra.n: degree of the solution
% Love_Spectra.m: order of the solution
% Love_Spectra.order: order of the coupling
% Love_Spectra.k: gravity Love numbers
% Love_Spectra.h: radial displacement Love numbers
```

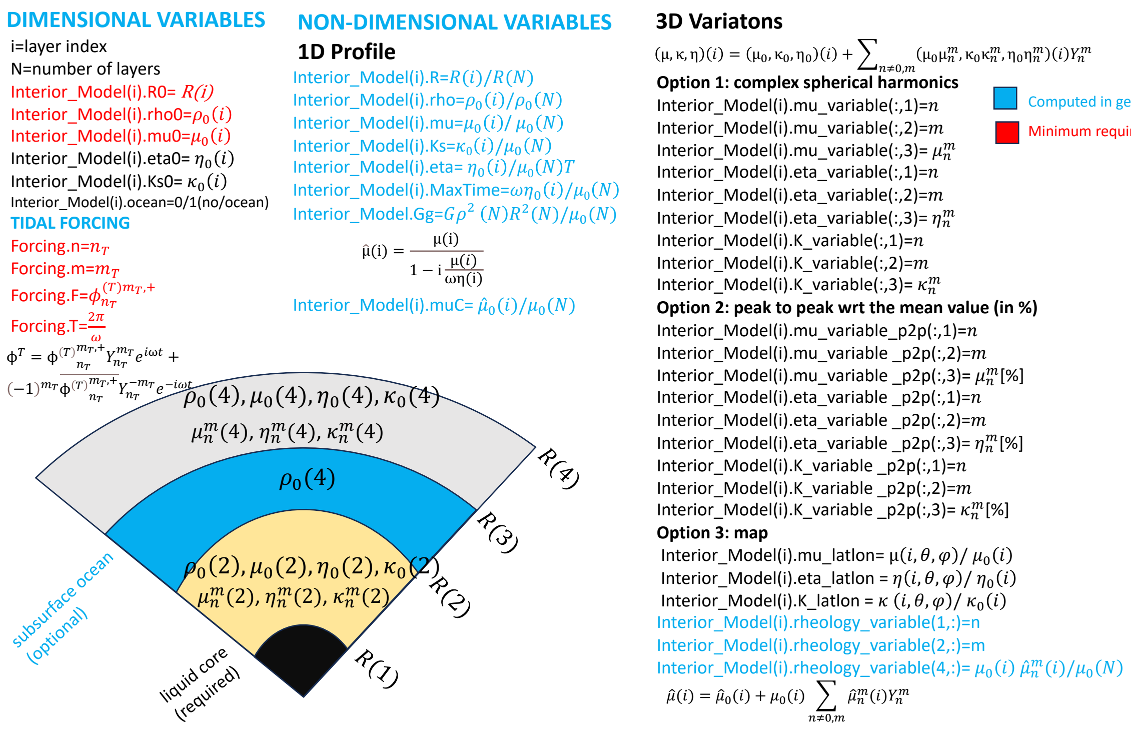
DIMENSIONAL VARIABLES
i=layer index
N=number of layers
Interior_Model(i).R0= $R(i)$
Interior_Model(i).rho0=$\rho_0(i)$
Interior_Model(i).mu0=$\mu_0(i)$
Interior_Model(i).eta0= $\eta_0(i)$
Interior_Model(i).Ks0= $\kappa_0(i)$
Interior_Model(i).ocean=0/1(no/ocean)
TIDAL FORCING
Forcing.n=n_T
Forcing.m=m_T
Forcing.F=$\phi_{n_T}^{(T)m_T,+}$
Forcing.T=$\frac{2\pi}{\omega}$
$\phi^T = \phi^{(T)}{}_{n_T}^{m_T,+} Y_{n_T}^{m_T} e^{i\omega t} + (-1)^{m_T} \overline{\phi^{(T)}{}_{n_T}^{m_T,+}} Y_{n_T}^{-m_T} e^{-i\omega t}$
NON-DIMENSIONAL VARIABLES
1D Profile
Interior_Model(i).R=$R(i)/R(N)$
Interior_Model(i).rho=$\rho_0(i)/\rho_0(N)$
Interior_Model(i).mu=$\mu_0(i)/\mu_0(N)$
Interior_Model(i).Ks=$\kappa_0(i)/\mu_0(N)$
Interior_Model(i).eta= $\eta_0(i)/\mu_0(N)T$
Interior_Model(i).MaxTime=$\omega\eta_0(i)/\mu_0(N)$
Interior_Model.Gg=$G\rho^2(N)R^2(N)/\mu_0(N)$
$\hat{\mu}(i) = \frac{\mu(i)}{1 - i\frac{\mu(i)}{\omega\eta(i)}}$
Interior_Model(i).muC= $\hat{\mu}_0(i)/\mu_0(N)$
3D Variatons
$(\mu,\kappa,\eta)(i) = (\mu_0,\kappa_0,\eta_0)(i) + \sum_{n\neq 0,m} (\mu_0\mu_n^m, \kappa_0\kappa_n^m, \eta_0\eta_n^m)(i) Y_n^m$
Option 1: complex spherical harmonics
Interior_Model(i).mu_variable(:,1)=n
Interior_Model(i).mu_variable(:,2)=m
Interior_Model(i).mu_variable(:,3)= μ_n^m
Interior_Model(i).eta_variable(:,1)=n
Interior_Model(i).eta_variable(:,2)=m
Interior_Model(i).eta_variable(:,3)= η_n^m
Interior_Model(i).K_variable(:,1)=n
Interior_Model(i).K_variable(:,2)=m
Interior_Model(i).K_variable(:,3)= κ_n^m
Option 2: peak to peak wrt the mean value (in %)
Interior_Model(i).mu_variable_p2p(:,1)=n
Interior_Model(i).mu_variable _p2p(:,2)=m
Interior_Model(i).mu_variable _p2p(:,3)= μ_n^m[%]
Interior_Model(i).eta_variable _p2p(:,1)=n
Interior_Model(i).eta_variable _p2p(:,2)=m
Interior_Model(i).eta_variable _p2p(:,3)= η_n^m[%]
Interior_Model(i).K_variable _p2p(:,1)=n
Interior_Model(i).K_variable _p2p(:,2)=m
Interior_Model(i).K_variable _p2p(:,3)= κ_n^m[%]
Option 3: map
Interior_Model(i).mu_latlon= $\mu(i,\theta,\varphi)/\mu_0(i)$
Interior_Model(i).eta_latlon = $\eta(i,\theta,\varphi)/\eta_0(i)$
Interior_Model(i).K_latlon = $\kappa(i,\theta,\varphi)/\kappa_0(i)$
Interior_Model(i).rheology_variable(1,:)=n
Interior_Model(i).rheology_variable(2,:)=m
Interior_Model(i).rheology_variable(4,:)= $\mu_0(i)\,\hat{\mu}_n^m(i)/\mu_0(N)$
$\hat{\mu}(i) = \hat{\mu}_0(i) + \mu_0(i) \sum_{n\neq 0,m} \hat{\mu}_n^m(i) Y_n^m$
🟦 Computed in get_rheology
🟥 Minimum required inputs
$\rho_0(4), \mu_0(4), \eta_0(4), \kappa_0(4)$
$\mu_n^m(4), \eta_n^m(4), \kappa_n^m(4)$
$\rho_0(4)$
$\rho_0(2), \mu_0(2), \eta_0(2), \kappa_0(2)$
$\mu_n^m(2), \eta_n^m(2), \kappa_n^m(2)$
$R(4)$
$R(3)$
$R(2)$
$R(1)$
subsurface ocean (optional)
liquid core (required)

Figure 1: Sketch of interior model and relation to inputs. Variables indicated in blue are computed inside `get_rheology`, variables indicated in red are the minimum inputs required.

- `y`: Radial Functions (see Rovira-Navarro et al., 2024, Eq. 40). Output of `get_Love`

```
% y.nf: degree of the forcing
% y.mf: order of the forcing
% y.n: degree of the solution mode
% y.m: order of the solution mode
% y.y(r,X,mode); where r stands for radial point and X
    % y.y(r,1,mode): r radial position
    % y.y(r,2,mode): U radial displacement
    % y.y(r,3,mode): V tangential displacement
    % y.y(r,4,mode): R normal stress
    % y.y(r,5,mode): S tangential stress
    % y.y(r,6,mode): \phi gravitational potential
    % y.y(r,7,mode): \dot\phi potential stress
    % y.y(r,8,mode): W toroidal displacement
    % y.y(r,9,mode): T toroidal stress
    % y.y(r,10,mode): u_{n,n-1}
    % y.y(r,11,mode): u_{n,n}
    % y.y(r,12,mode): u_{n,n+1}
    % y.y(r,13,mode): \sigma_{n,n,0}
    % y.y(r,14,mode): \sigma_{n,n-2,2}
    % y.y(r,15,mode): \sigma_{n,n-1,2}
    % y.y(r,16,mode): \sigma_{n,n,2}
    % y.y(r,17,mode): \sigma_{n,n+1,2}
    % y.y(r,18,mode): \sigma_{n,n+2,2}
    % y.y(r,19,mode): \epsilon_{n,n,0}
    % y.y(r,20,mode): \epsilon_{n,n-2,2}
    % y.y(r,21,mode): \epsilon_{n,n-1,2}
    % y.y(r,22,mode): \epsilon_{n,n,2}
    % y.y(r,23,mode): \epsilon{n,n+1,2}
    % y.y(r,24,mode): \epsilon_{n,n+2,2}
```

- `y_LatLon` `y` transformed to the spatial domain. Output of `get_map`

```
% y_LatLon.nf: degree of the forcing
% y_LatLon.mf: order of the forcing
% y_LatLon.lon: longitude
% y_LatLon.lat: latitude
% y_LatLon.r: radial point at which y functions are evaluated
% y_LatLon.mu(lon,lat): shear modulus
% y_LatLon.forcing(lon,lat): forcing
% y_LatLon.y(lon,lat,r,1):  Gravitational Potential
% y_LatLon.y(lon,lat,r,2):  Displacement e_r component
% y_LatLon.y(lon,lat,r,3):  Displacement e_theta component
% y_LatLon.y(lon,lat,r,4):  Displacement e_phi component
% y_LatLon.y(lon,lat,r,5):  stress   e_r e_r component
% y_LatLon.y(lon,lat,r,6):  stress   e_r e_theta component
% y_LatLon.y(lon,lat,r,7):  stress   e_r e_phi component
% y_LatLon.y(lon,lat,r,8):  stress   e_theta e_r component
% y_LatLon.y(lon,lat,r,9):  stress   e_theta e_theta component
% y_LatLon.y(lon,lat,r,10): stress   e_theta e_phi component
% y_LatLon.y(lon,lat,r,11): stress   e_phi e_r component
% y_LatLon.y(lon,lat,r,12): stress   e_phi e_theta component
% y_LatLon.y(lon,lat,r,13): stress   e_phi e_phi component
% y_LatLon.y(lon,lat,r,14): strain   e_r e_r component
% y_LatLon.y(lon,lat,r,15): strain   e_r e_theta component
% y_LatLon.y(lon,lat,r,16): strain   e_r e_phi component
% y_LatLon.y(lon,lat,r,17): strain   e_theta e_r component
% y_LatLon.y(lon,lat,r,18): strain   e_theta e_theta component
% y_LatLon.y(lon,lat,r,19): strain   e_theta e_phi component
% y_LatLon.y(lon,lat,r,20): strain   e_phi e_r component
% y_LatLon.y(lon,lat,r,21): strain   e_phi e_theta component
% y_LatLon.y(lon,lat,r,22): strain   e_phi e_phi component
```

- `Energy_Spectra` Energy dissipation spectra (see Rovira-Navarro et al., 2024, Eq. (32)). Output of `get_energy`

```
%Energy_Spectra.n: degrees with non-zero energy
%Energy_Spectra.m: orders with non-zero energy
%Energy_Spectra.n_v: degrees from 0 to Numerics.Nenergy
%Energy_Spectra.m_v: orders from 0 to Numerics.Nenergy
%Energy_Spectra.energy(r,mode): radial profile of energy spectra
```

```
%Energy_Spectra.energy_integral(mode): radially integrated energy
   for all non-zero degrees and orders (n,m)
%Energy_Spectra.energy_integral_v(mode): radially integrated
   energy for all degrees and orders (n_v,m_v)
```

A note about the *LOV3D* use of nondimensional units

LOV3D uses nondimensional units. The spatial and temporal timescales are the surface radii and the orbital period R and T. Stress and rheological parameters (e.g., shear and bulk modulii) are non-dimensionalized with the shear modulus of the uppermost layer $\mu_0(N)$ and density with its density of the solid envelope ($\rho_0(N)$). The following relations between dimensional and nondimensional quantities —indicated with a $'$— can be derived:

$$\mu_0'(i) = \frac{1}{\mu_0(N)}\mu_0(i) \quad \eta_0'(i) = \frac{1}{T\mu_0(N)}\eta_0(i) \quad \kappa_0'(i) = \frac{1}{\mu_0(N)}\kappa_0(i)$$

$$r' = \frac{1}{R(N)}r \quad u' = \frac{1}{R(N)}\mathbf{u} \quad \sigma' = \frac{1}{\mu_0(N)}\sigma$$

$$\dot{e}' = \frac{T}{\mu_0(N)}\dot{e} \quad \dot{E}' = \frac{T}{R^3(N)\mu_0(N)}\dot{E}$$

$$G' = \frac{\rho_0(N)^2R(N)^2}{\mu_0(N)}G \quad \phi' = \frac{\rho_0(N)}{\mu_0(N)}\phi \quad g' = \frac{\rho_0(N)R(N)}{\mu_0(N)}g$$

Here, σ, $\mathbf{u}$, $\dot{e}$, $\dot{E}$, g and ϕ stand for the stress tensor, the displacement field, the volumetric and total energy dissipation, and the gravitational acceleration and potential. Note that *LOV3D* computes the tidal response assuming a unit tidal forcing $\phi'^{T m_T}_{n_T} = 1$. Thus in order to recover the tidal response one needs to multiply the solution y by the factor $A\rho_0(N)/\mu_0(N)$ where A is the amplitude of the tidal force in dimensional units. For example, for a moon in an eccentric synchronous orbit $A = (\omega R)^2 e$, where e is the moon eccentricity (see Rovira-Navarro et al. (2024) Appendix D and Table 2). In such a case, the dimensional volumetric energy dissipation follows from:

$$\dot{e} = \frac{\rho_0^2(N)\omega^4 R^4(N)e^2}{\mu_0(N)T}\dot{e}' \tag{2}$$

3 Functions & Subroutines

The functions are stored in `/src`. The main functions are:

- `set_boundary_indices`: Radial discretization of the problem using the discretization method specified in `Numerics.method` and the number of radial points specified in `Numerics.Nrbase`. Updates the variable `Numerics`.
- `get_rheology`: Prepares the variable `Interior_Model` for the remaining functions. It
  - checks that there are no errors in the input

- nondimensionalizes the interior properties
- if the user has not provided some of the parameters sets them (e.g., sets a layer to elastic if no viscosity is provided).
- computes the complex shear modulus for spherically-symmetric and laterally-varying interior models

The function outputs an updated version of `Interior_Model` that can be used by `get_Love`

- `get_Love.m` computes the tidal response of the body for a given `Interior_Model` and `Forcing` using the numerical information specified in `Numerics`. It employs the subroutines `get_solution`

- `get_solution` computes the tidal response by numerically-integrating the governing equations using a Runge-Kutta integrator (see Section 4 of Rovira-Navarro et al. (2024) for details).

- `get_energy` obtains the energy dissipation spectra (`Energy_Spectra`) given the tidal response of the body (`y`). It uses the function `get_energy_couplings` and `get_energy_couplings_all`.

- `get_couplings` & `get_couplings_all` compute the rheology couplings for models with lateral variations (see Appendix B of Rovira-Navarro et al. (2024)). `get_couplings` computes only the couplings for the particular lateral variations and forcing specified (`Numerics.load_couplings=1`), while `get_couplings_all` computes all coupling coeffients up to the maximum degree of the lateral variations `Numerics.load_couplings_all=2` If multiple-runs with different lateral variations are employed, the later is recommended. They employ spherical harmonic functions contained in `src/SPH_Tools`. The coupling coefficients are stored in `data/couplings` for future use.

- `get_energy_couplings_all` & `get_energy_couplings` computes the energy dissipation integrals defined in Appendix B of Rovira-Navarro et al. (2024). The energy integrals are stored in `data/couplings` for future use. As for `get_couplings`, either only the required couplings are computed, or all up to the maximum considered degree of the rheology variations are computed.

The the directory `src/Plot_Tools` contains some useful functions that can be used for plotting.

- `get_map` computes the required tensor spherical harmonics and transforms the spectral solution, `y`, to the spatial domain `y_LatLon` using the transformations of Appendix A of Rovira-Navarro et al. (2024)

- `plot_map`, plots `y_LatLon`

- `plot_y`, in `/src/Plot\_Tools/`, plots the y functions $y(r)$ for a spherically-symmetric case, and one with lateral variations.

- `plot_energy_map`, plots the geographical distribution of tidal heating.

Additionally, the directory `/SPH_Tools` contains several functions used for Spherical Harmonics related computations, including:

- `Legendre.m` computes the normalized associated Legendre functions.
- `Wigner3j.m`, `Wigner6j.m` and `Wigner9j.m`, in `/Functions/SPH_Tools/Wigner/`, obtains the Wigner symbols required to obtain the coupling integrals. These functions were developed by Vladimir Sovkov and are available at MATLAB Central File Exchange
- `LatLon_SPH` converts a field in the spatial domain to the spectral domain (real spherical harmonics).
- `SPH_LatLon` converts a field from the spectral domain (real spherical harmonics) to the spatial domain.

The relation between the different functions is illustrated in the flow diagram of Figure 2.

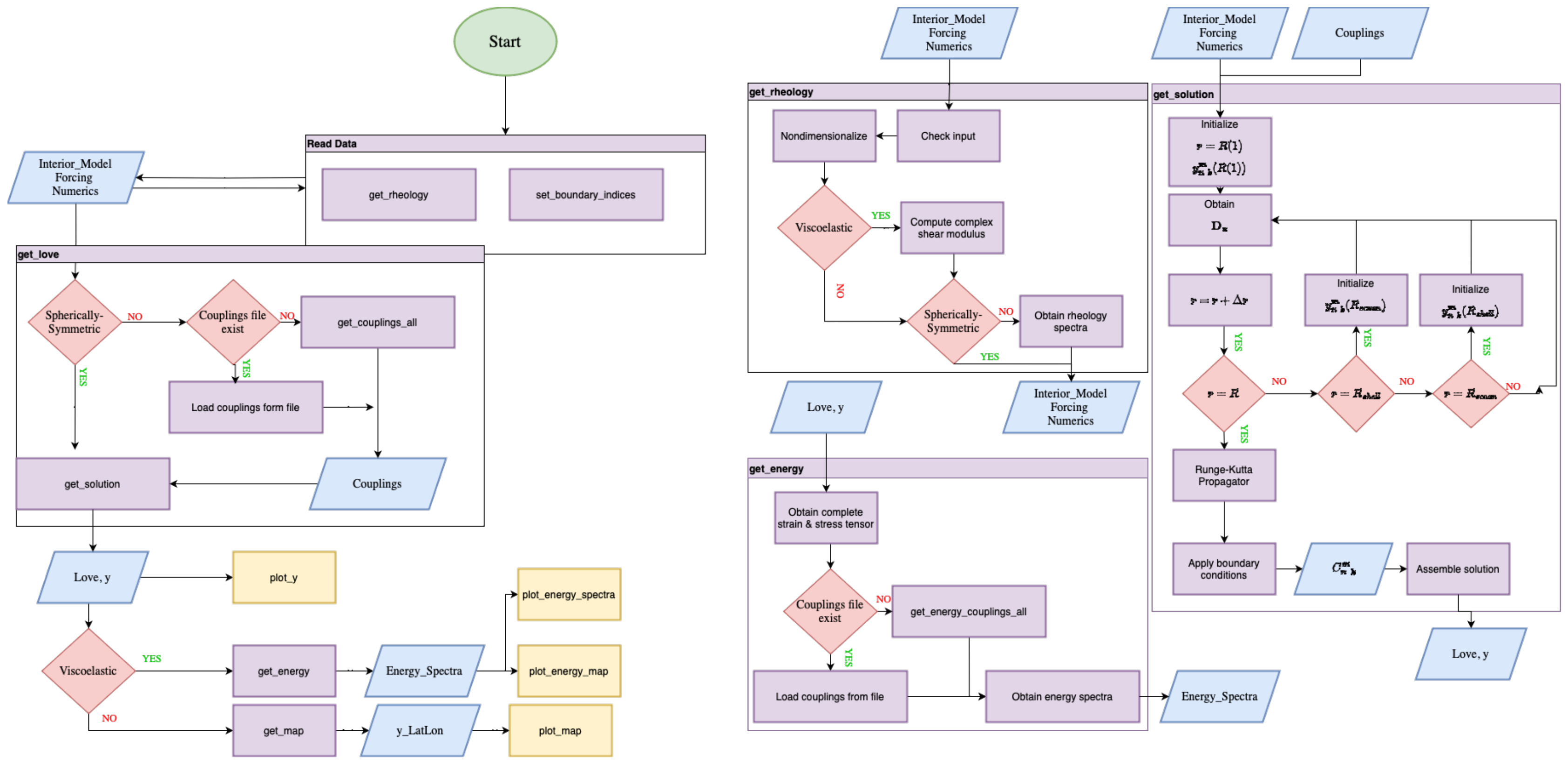

Figure 2: Code flow diagram with details of relevant subroutines. Plotting subroutines are indicated in yellow.

4 Examples

The directory `/tests` provides several example scripts, which include:

- `/tests/Test_One_Layer_Spherically_Symmetric.mlx`: Compares LOV3D Love numbers against love numbers obtained analytically for a uniform spherically-symmetric body
- `/tests/Test_Io_Multi_Layer_Spherically_Symmetric.mlx`: Multi-layered spherically-symmetric Io model based of Steinke et al. (2020), consisting of core, deep mantle, asthenosphere and lithosphere. The script obtains the Love numbers and compares them against results obtained with the spherically-symmetric code of Rovira-Navarro et al. (2022)
- `/tests/Test_Io_Multi_Layer_Spherically_Symmetric_Tidal_Heating.mlx`: Same as the previous test but:
  - tidal heating is computed using `/src/get_energy.m`
  - the geographical distribution of tidal heating is ploted using the `src/Plot_Tools/plot_energy_map.m` plotting function.
  - the y functions are computed and plotted using `/src/get_map.m` and `src/Plot_Tools/plot_map.m`
- `/tests/Test_Europa_Titan_Spherically_Symmetric.mlx`:Spherically-symmetric multi-layered icy moon model. The script computes the Love numbers for a multi-layered Europa and Titan models based on the interior models of Beuthe (2013)
- `/tests/Test_Enceladus_Two_Layers_Lateral_Variations.mlx`: 3 layer Enceladus model consisting of a rigid core, ocean and ice-shell with lateral variations. Compares LOV3D Love numbers against love numbers obtained using the perturbation method of Qin et al. (2014) and the FEM model of Berne et al. (2023a). Reproduces Figure 2 of Rovira-Navarro et al. (2024)
- `/tests/Test_Europa_Lateral_Variations.mlx`:Multi-layered icy moon model with lateral variations. It employs `plot_y` to show the y functions.

Please check the github repository for updates.

References

Berne, A., Simons, M., Keane, J., & Park, R. 2023a, , , doi: 10.1029/2022JE007712

Beuthe, M. 2013, Icarus, 223, 308, doi: https://doi.org/10.1016/j.icarus.2012.11.020

Qin, C., Zhong, S., & Wahr, J. 2014, , 199, 631, doi: 10.1093/gji/ggu279

Rovira-Navarro, M. 2023, LOV3D, https://github.com/mroviranavarro/LOV3D_open, GitHub

Rovira-Navarro, M., Katz, R. F., Liao, Y., van der Wal, W., & Nimmo, F. 2022, Journal of Geophysical Research: Planets, 127, e2021JE007117, doi: https://doi.org/10.1029/2021JE007117

Rovira-Navarro, M., Matsuyama, I., & Berne, A. 2024, Planetary Science Journal, 5, doi: 10.3847/PSJ/ad381f

Steinke, T., Hu, H., Höning, D., van der Wal, W., & Vermeersen, B. 2020, Icarus, 335, 113299, doi: https://doi.org/10.1016/j.icarus.2019.05.001