



**LOV3D**

Version 1.0

**M.Rovira-Navarro**

May 7, 2024

# Contents

1	Introduction	2
2	Inputs & Outputs	2
3	Functions & Subroutines	7
4	Script examples	9
5	Work in Progress	9

# List of Code Listings

# 1 Introduction

`L0V3D` is a code developed to obtain the tidal response of bodies with lateral variations of interior properties. The tidal response is obtained using the semi-analytical approach detailed in [Rovira-Navarro et al. \(2024\)](#). This version of `L0V3D` computes the Love numbers, radial functions and energy dissipation spectra for a 3 layer interior model consisting of (1) a solid inner core, (2) a liquid core and (3) a solid envelope with laterally varying properties.

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. Section 5 lists ongoing work to extend the software.

## 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. Both dimensional and non-dimensional quantities can be provided. Dimensional quantities are indicated with a 0 and non-dimensional without. If dimensional quantities are provided, they are non-dimensionalized inside the code. Some values are assigned inside the code and are output by `get_Love` as an optional output argument, called `Interior_ModelU` in `example.m`. The structure contains the following fields.

```
1 % Interior_Model.R0: radius
2     % R0(1) core radius (solid+liquid core)
3     % R0(2) surface radius
4 % Interior_Model.rho0: layers density
5     % rho0(1) core density (solid+liquid core)
6     % rho0(2) density of the outermost solid layer
7 % Interior_Model.Delta_rho0: Density difference between the liquid
    core and overlying solid layer. If not provided it is computed
    assuming that the two innermost layers have rho0(1).
8 % Interior_Model.mu0: shear modulus of the the outermost layer
9 % Interior_Model.Ks0: bulk modulus of the outermost layer
10 % Interior_Model.eta0: viscosity of the outermost layer
11 % Interior_Model.MaxTime: non-dimensional Maxwell time (assigned
    inside of the code if not provided)
12 % Interior_Model.mu_variable: shear modulus variations
13     % mu_variable(:,1): degree of variation
14     % mu_variable(:,2): order of variation
15     % mu_variable(:,3): amplitude of the variation (mu_l^m/mu^0_0)
16 % Interior_Model.K_variable: bulk modulus variations
17     % K_variable(:,1): degree of variation
18     % K_variable(:,2): order of variation
```

```

19      % K_variable(:,3): amplitude of bulk modulus variations ( $K_l^m/K^{\theta_0}$ )
20 % Interior_Model.eta_variable: viscosity
21      % eta_variable(:,1): degree of variation
22      % eta_variable(:,2): order of variation
23      % eta_variable(:,3): amplitude of viscosity variations ( $\eta_l^m/\eta^{\theta_0}$ )
24 % Interior_Model.rheology_variable: complex shear and bulkd
    modulus lateral variations (assigned inside the code)
25      % rheology_variable(:,1): degree of variation
26      % rheology_variable(:,2): order of variation
27      % rheology_variable(:,3): amplitude of bulk modulus
    variations ( $K_l^m/K^{\theta_0}$ ) --- this has not yet been used and
    tested
28      % rheology_variable(:,4): amplitude of complex shear modulus
    variations ( $\mu_l^m/\mu^{\theta_0}$ )
29 % Interior_Model.muC: complex shear modulus (assigned inside the
    code)

```

- **Forcing** Structure containing information about the forcing.

```

1 % Forcing.Td: forcing period
2 % Forcing.n: degree of the forcing
3 % Forcing.m: order of the forcing
4 % Forcing.F: amplitude of the component

```

- **Numerics** Structure containing numerical information.

```

1 % Numerics.Nr: number of radial points
2 % Numerics.perturbation_order: maximum order of the perturbation.
    Default is 2
3 % Numerics.rheology_cutoff: determines which terms of the rheology
    are include, (only relevant for viscoelastic rheology: terms
    with  $\log_{10}(\mu_n^m) - \log_{10}(\mu_n^m(\text{leading})) \geq -\text{Numerics.}$ 
    rheology_cutoff are included. Default 0 (only leading terms)
4 % Numerics.load_couplings: indicates if the coupling coefficients
    are computed or loaded from existing file
5     % (0) compute coupling coefficients from scratch
6     % (1) load coupling coefficients from file that contains the
    coupling coefficintes for the specific rheology variations
    considered. If the file does not exist, it is generated and
    stored in Files_Coupling
7     % (2) load coupling coefficintes from a file that contains all
    couplung coefficients up to a given degree. If the file
    does not exist, it is generated and stored in
    Files_Coupling
8 % Numerics.Nenergy: up to which degree the energy dissipation
    spectra is computed. Default 8

```

### DIMENSIONAL VARIABLES

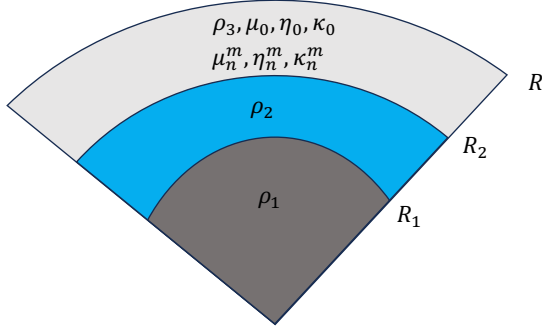
```
Interior_Model.R0(2)= R
Interior_Model.R0(1)= R2
Interior_Model.rho0(2)= ρ3
Interior_Model.rho0(1)= ρ2 + (R2/R1)3 (ρ1 - ρ2)
Interior_Model.mu0=μ0
Interior_Model.eta0=η0
Interior_Model.Ks0=κ0
```

### NON-DIMENSIONAL VARIABLES

$$(\mu, \kappa, \eta) = (\mu_0, \kappa_0, \eta_0) + \sum_{n \neq 0, m} (\mu_0 \mu_n^m, \kappa_0 \kappa_n^m, \eta_0 \eta_n^m) Y_n^m.$$

```
Interior_Model.mu_variable(:,1)=n
Interior_Model.mu_variable(:,2)=m
Interior_Model.mu_variable(:,3)=μnm
Interior_Model.eta_variable(:,1)=n
Interior_Model.eta_variable(:,2)=m
Interior_Model.eta_variable(:,3)=ηnm
Interior_Model.K_variable(:,1)=n
Interior_Model.K_variable(:,2)=m
Interior_Model.K_variable(:,3)=κnm
```

```
Interior_Model.R(2)=1
Interior_Model.R(1)= R2/R
Interior_Model.rho(2)= 1
Interior_Model.rho(1)= (ρ2 + (R2/R1)3 (ρ1 - ρ2)) / ρ3
Interior_Model.mu= 1
Interior_Model.Ks=κ0/μ0
Interior_Model.eta= η0 / μ0 T
Interior_Model.MaxTime=ωη / μ
Interior_Model.Gg=Gρ32 R2 / μ0
μ̂ = μ̂0 + μ0 ∑n≠0,m μ̂nm Ynm
Interior_Model.muC= μ̂0 / μ0
Interior_Model.rheology_variable(1,:)=n
Interior_Model.rheology_variable(2,:)=m
Interior_Model.rheology_variable(4,:)= μ̂nm
```



### TIDAL FORCING

$$\phi^T = \phi_{n_T}^{(T)m_T,+} Y_{n_T}^{m_T,+} e^{i\omega t} + (-1)^{m_T} \overline{\phi_{n_T}^{(T)m_T,+} Y_{n_T}^{m_T,+}} e^{-i\omega t}$$

```
Forcing.n=nT
Forcing.m=mT
Forcing.F=φnT(T)mT,+
Forcing.T=2π/ω
```

Figure 1

Figure 2: Sketch of interior model and relation to inputs. Fields indicated in blue are computed inside `get_Love`.

- 9 % Numerics.parallel: can be used to compute the tidal response per forcing in parallel (1) or not (0).

## Outputs

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

```
1 % Love_Spectra.nf: degree of the forcing
2 % Love_Spectra.mf: order of the forcing
3 % Love_Spectra.n: degree of the solution
4 % Love_Spectra.m: order of the solution
5 % Love_Spectra.order: order of the coupling
6 % Love_Spectra.k: gravity Love numbers
7 % Love_Spectra.h: radial displacement Love numbers
```

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

```
1 % y.nf: degree of the forcing
2 % y.mf: order of the forcing
3 % y.n: degree of the solution mode
4 % y.m: order of the solution mode
5 % y.y(r,X,mode); where r stands for radial point and X
```

```

6      % y.y(r,1,mode): r radial position
7      % y.y(r,2,mode): U radial displacement
8      % y.y(r,3,mode): V tangential displacement
9      % y.y(r,4,mode): R normal stress
10     % y.y(r,5,mode): S tangential stress
11     % y.y(r,6,mode): \phi gravitational potential
12     % y.y(r,7,mode): \dot{\phi} potential stress
13     % y.y(r,8,mode): W toroidal displacement
14     % y.y(r,9,mode): T toroidal stress
15     % y.y(r,10,mode): u_{n,n-1}
16     % y.y(r,11,mode): u_{n,n}
17     % y.y(r,12,mode): u_{n,n+1}
18     % y.y(r,13,mode): \sigma_{n,n,0}
19     % y.y(r,14,mode): \sigma_{n,n-2,2}
20     % y.y(r,15,mode): \sigma_{n,n-1,2}
21     % y.y(r,16,mode): \sigma_{n,n,2}
22     % y.y(r,17,mode): \sigma_{n,n+1,2}
23     % y.y(r,18,mode): \sigma_{n,n+2,2}
24     % y.y(r,19,mode): \epsilon_{n,n,0}
25     % y.y(r,20,mode): \epsilon_{n,n-2,2}
26     % y.y(r,21,mode): \epsilon_{n,n-1,2}
27     % y.y(r,22,mode): \epsilon_{n,n,2}
28     % y.y(r,23,mode): \epsilon_{n,n+1,2}
29     % y.y(r,24,mode): \epsilon_{n,n+2,2}

```

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

```

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

```

```

21 % y_LatLon.y(lon,lat,r,14): strain    e_r e_r component
22 % y_LatLon.y(lon,lat,r,15): strain    e_r e_theta component
23 % y_LatLon.y(lon,lat,r,16): strain    e_r e_phi component
24 % y_LatLon.y(lon,lat,r,17): strain    e_theta e_r component
25 % y_LatLon.y(lon,lat,r,18): strain    e_theta e_theta component
26 % y_LatLon.y(lon,lat,r,19): strain    e_theta e_phi component
27 % y_LatLon.y(lon,lat,r,20): strain    e_phi e_r component
28 % y_LatLon.y(lon,lat,r,21): strain    e_phi e_theta component
29 % 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`

```

1 %Energy_Spectra.n: degrees with non-zero energy
2 %Energy_Spectra.m: orders with non-zero energy
3 %Energy_Spectra.n_v: degrees from 0 to Numerics.Nenergy
4 %Energy_Spectra.m_v: orders from 0 to Numerics.Nenergy
5 %Energy_Spectra.energy(r,mode): radial profile of energy spectra
6 %Energy_Spectra.energy_integral(mode): radially integrated energy
   for all non-zero degrees and orders (n,m)
7 %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 moduli) are non-dimensionalized with the shear modulus of the solid envelope  $\mu_0$  and density with the density of the solid envelope ( $\rho_3$ ). The following relations between dimensional and nondimensional quantities —indicated with a '— can be derived:

$$\begin{aligned}
\mu'_0 &= 1 & \eta' &= \frac{1}{T\mu_0}\eta_0 & \kappa'_0 &= \frac{1}{\mu_0}\kappa_0 \\
r' &= \frac{1}{R}r & u' &= \frac{1}{R}\mathbf{u} & \boldsymbol{\sigma}' &= \frac{1}{\mu_0}\boldsymbol{\sigma} \\
\dot{e}' &= \frac{T}{\mu_0}\dot{e} & \dot{E}' &= \frac{T}{R^3\mu_0}\dot{E} \\
G' &= \frac{\rho_0^2 R^2}{\mu_0}G & \phi' &= \frac{\rho_0}{\mu_0}\phi & g' &= \frac{\rho_0 R}{\mu_0}g
\end{aligned}$$

Here,  $\boldsymbol{\sigma}$ ,  $\mathbf{u}$ ,  $\dot{e}$ ,  $\dot{E}$ ,  $g$  and  $\phi$  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  $A\rho_0/\mu_0$  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 Appendix D and Table 2). In such a case, the dimensional volumetric energy dissipation follows from:

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

### 3 Functions & Subroutines

The functions are stored in /Functions. The main functions are:

- `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_couplings`.
- `get_couplings` determines the modes that intervene in the tidal response and the coupling coefficients (see Appendix B). It employs spherical harmonic functions contained in `Functions/SPH_Tools`. The coupling coefficients are stored in `Files_Coupling` for future use.
- `get_solution` computes the tidal response by numerically-integrating the governing equations using a Runge-Kutta integrator (see Section 4 for more details).
- `get_energy` obtains the energy dissipation spectra (`Energy_Spectra`) given the tidal response of the body (`y`). It uses the function `couplings_energy`.
- `couplings_energy` computes the energy dissipation integrals defined in Appendix B. The energy integrals are stored in `Files_Coupling` for future use.
- `get_map` computes the required tensor spherical harmonics and transforms the spectral solution, `y`, to the spatial domain `y_LatLon` using the transformation of Appendix A
- `plot_map`, in `ff/Functions/Plot_Tools/`, plots `y_LatLon`

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 3.



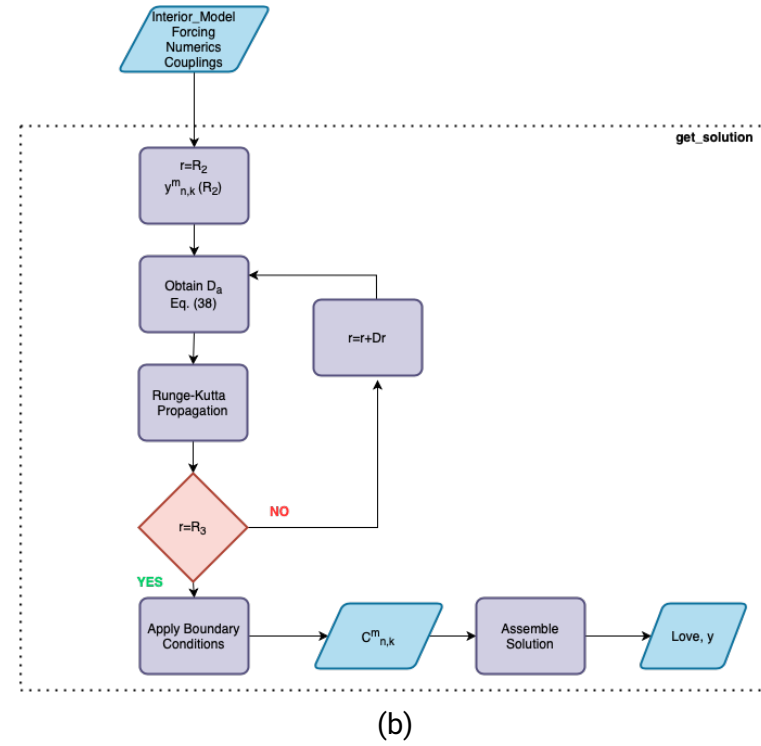
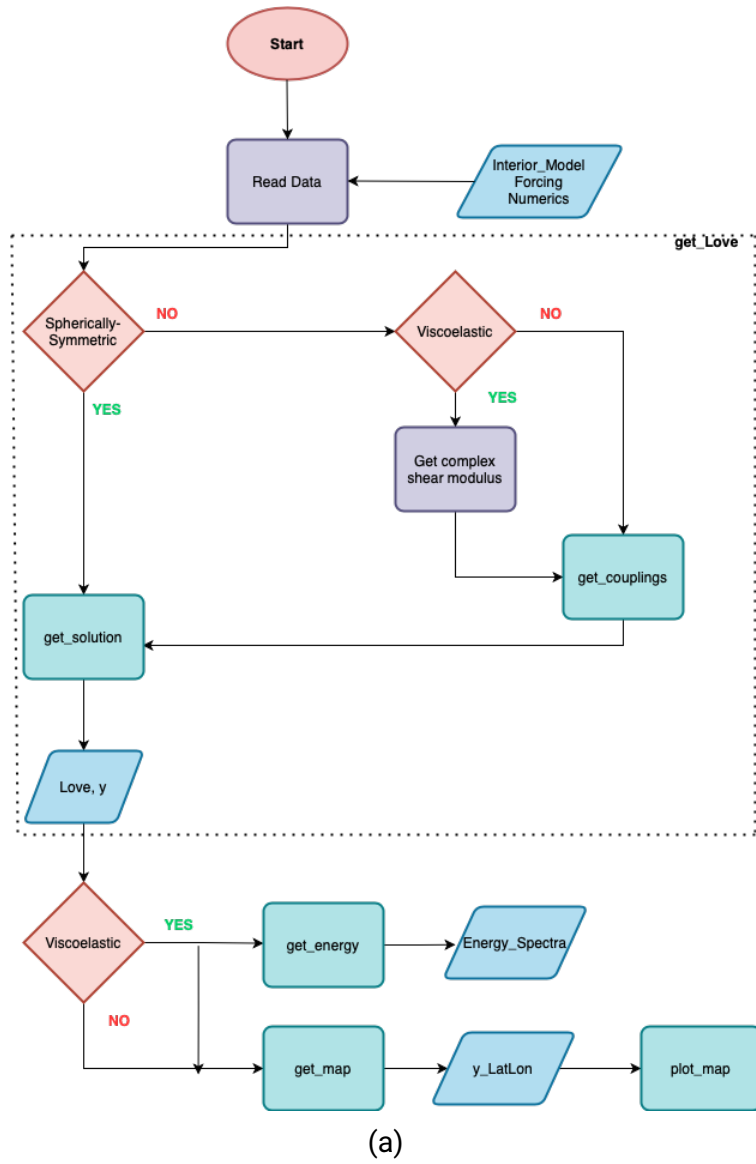


Figure 3: Code Flow Diagram (a) and detail of the function get\_solution flow diagram (b)

## 4 Script examples

The script `/Scripts/example.m` provides a test-run for the code. It can be used to reproduce the results in Section 7 of [Rovira-Navarro et al. \(2024\)](#). For this example:

- We consider the reference Io model of Table 1 with monochromatic lateral viscosity variations
- The Love numbers and energy spectra for the model and a spherically-symmetric counterpart are computed
- Gravitational potential, surface displacements, stress and strain tensors, and tidal heating maps are plotted.
- In all cases, non-dimensional units are used. See `get_Love.m` L161 for a reference to non-dimensional units.

## 5 Work in Progress

The authors of this code are currently working on:

- Including radial variations of interior properties.
- Extending boundary conditions to consider other types of loads (e.g., surface loads).

Please check the github repository for updates.

## References

Rovira-Navarro, M., Matsuyama, I., & Berne, A. 2024, The Planetary Science Journal, 5, doi: [10.3847/PSJ/ad381f](#)