LoadDef User Manual

LoadDef v. 1.1.4

Hilary R. Martens[1], Luis Rivera[2], and Mark Simons[3]

[1]*Department of Geosciences, University of Montana, Missoula, MT, USA*
[2]*Institut de Physique du Globe de Strasbourg (UMR7516), Université de Strasbourg/CNRS, Strasbourg, France*
[3]*Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena, CA, USA*

February 3, 2021

Abstract

`LoadDef` is a `Python`-based toolkit for modeling the spheroidal deformation-response of a spherically symmetric, non-rotating, elastic, and isotropic (SNREI) Earth to a variety of applied forces, including surface mass loading and gravitational fields. We designed the toolkit with ease-of-use in mind, such that it may be a resource for both experienced Earth-deformation modelers as well as newcomers to the field. Love numbers are generated by integrating the equations of motion for spheroidal deformation through a spherically-symmetric planetary model using Runge-Kutta techniques, with appropriate boundary conditions applied at the surface. The planetary model may be radially heterogeneous and optionally include a fluid layer. To quantify the effects of small perturbations to the density and elastic structure on the deformation-response, the toolkit also includes a module to compute partial derivatives of Love numbers with respect to perturbations in the density, bulk modulus, and shear modulus. Load Green's functions (displacement, gravity, strain, and tilt) are derived from combinations of load Love numbers in three separate reference frames (CE, CM, CF). The displacement load Green's functions may then be convolved with local- to global-scale models of surface mass loading to derive predicted surface displacements at user-defined geographic locations.

Preface

0.1 About This Document

This document is a user manual accompanying the `LoadDef` software. Section 1 discusses how to access and install the software. Section 2 describes how to access certain datasets that may be used with the software, some of which require registration and license agreements. We then explain the structure of the main `LoadDef` modules, including optional arguments and file formats, in Section 3. In Section 4, we provide several examples showing how to perform some of the primary functions of the software. A description of the theory and methods fundamental to the modeling objectives for which the code is designed is provided in Section 5. References are provided at the end of the document.

0.2 Attribution

The `LoadDef` software is a product of Hilary Martens's PhD studies at Caltech in collaboration with Mark Simons (Caltech) and Luis Rivera (Université de Strasbourg). We distribute the software free of charge with the hope that you may find it useful in your own research and educational pursuits. In the normal scientific practice, we request that you recognize the efforts of the authors by citing the appropriate peer-reviewed paper(s) in presentations and publications.

We aim to continue supporting and developing LoadDef based on community input, and we also welcome opportunities to collaborate as co-author(s) on project applications that make use of LoadDef. If you have any questions about the software or would like to discuss a particular project idea further, please get in touch! Thanks! (`hilary.martens@umontana.edu`)

To acknowledge use of this software, please cite:

1. Martens, H.R., L. Rivera, and M. Simons (2019), "LoadDef: A Python-based toolkit to model elastic deformation caused by surface mass loading on spherically symmetric bodies," *Earth & Space Science*, **6**: 1–13, doi:10.1029/2018EA000462.

Additional publications associated with the software include:

2. Martens, H.R., and M. Simons (2020), "A comparison of predicted and observed ocean tidal loading in

Alaska," *Geophys. J. Int.*, **223**(1): 454–470, doi:10.1093/gji/ggaa323.

3. Martens, H.R., L. Rivera, M. Simons, and T. Ito (2016), "The sensitivity of surface mass loading displacement response to perturbations in the elastic structure of the crust and mantle," *J. Geophys. Res. Solid Earth*, **121**: 3911–3938, doi:10.1002/2015JB012456.

4. Martens, H.R., M. Simons, S. Owen, and L. Rivera (2016), "Observations of ocean tidal load response in South America from sub-daily GPS positions," *Geophys. J. Int.*, **205**(3): 1637–1664, doi:10.1093/gji/ggw087.

For additional information, you may also be interested to consult:

Martens, Hilary R. (2016), "Using Earth deformation caused by surface mass loading to constrain the elastic structure of the crust and mantle," Dissertation (Ph.D.), California Institute of Technology. doi:10.7907/Z9N29TX7. `http://resolver.caltech.edu/CaltechTHESIS:04102016-211741759`

0.3 Support

`LoadDef` is based upon work supported by the National Science Foundation Geophysics Program funding under Grant No. EAR-1417245 as well as by the National Science Foundation Graduate Research Fellowship to HRM under Grant No. DGE-1144469. This software is based upon work supported by the NASA Earth and Space Science Fellowship to HRM under Grant No. NNX14AO04H as well as by NASA EPSCoR funding under Grant No. NNX15AK40A.

0.4 Request for Comments

We welcome comments, suggestions, and corrections aimed at improving the software and documentation. Please report errors, inaccuracies, typos, comments, and suggestions to Dr. Hilary Martens at the University of Montana (`hilary.martens@umontana.edu`).

0.5 `LoadDef` Software License and Copyright Notice

Contents

1 Download and Installation

`LoadDef` may be downloaded from GitHub under `https://github.com/hrmartens/LoadDef/`.

Questions about the download and installation of `LoadDef` should be directed to Dr. Hilary Martens at: `hilary.martens@umontana.edu`.

`LoadDef` is compatible with `Python2` and `Python3`, but `Python3` will be prioritized for future versions. `LoadDef` has been tested on Ubuntu and Debian Linux systems.

1.1 Installation with **`Anaconda`** (RECOMMENDED)

To install all packages using `Anaconda` (recommended), we suggest first creating a new environment.

First, you will need to install `Anaconda` on your system, if you haven't done so already:
`https://www.anaconda.com`

Next, at the command line, run the following three lines:

```
conda config --prepend channels conda-forge
conda config --set channel_priority strict
conda create -n LDEF python=3.7 basemap mpi4py scipy netCDF4 pyshp shapely
```

The commands above only need to be executed once on your system. From now on, each time you wish to run `LoadDef`, you only need to activate the LDEF environment (see next command).

To activate the new LDEF environment, run:

```
conda activate LDEF
```

If the command above did not work for you (e.g. for older versions of `Anaconda`), then you can try:

```
source activate LDEF
```

You're all set! `LoadDef` should now be ready to use.

To deactivate the new environment when you are finished, you can run:

```
conda deactivate
```

1.2 Installation with Other Package Managers

Several `Python` modules are required to use `LoadDef`. Most are quite standard. We recommend that modules be installed using a package-management system, such as `Anaconda`, `apt-get`, `python-pip`, or `macports`. We recommend `Anaconda`, since administrator privileges are not required.

Required modules include:

1. **`numpy`**
2. **`scipy`**
3. **`matplotlib`**
4. **`math`**
5. **`sys`**
6. **`os`**
7. **`netCDF4`**
8. **`datetime`**
9. **`mpi4py`**

Optional, but recommended, `Python` modules include:

1. **`mpl_toolkits.basemap`**
2. **`shapefile`**
3. **`shapely`**
4. **`pyproj`**

If you are using Ubuntu (Linux), you may install packages using `apt-get` and `python-pip`. Both require administrator privileges. To install `python-pip` using `apt-get`:

```
sudo apt-get install python-pip
```

To search for additional packages using `apt-get`:

```
apt-cache search []
```

To search for `Python` packages:

```
pip search []
```

To install most standard `Python` packages:

```
sudo pip install []
```

For example:

```
sudo pip install numpy

sudo pip install scipy

sudo pip install matplotlib

sudo pip install math

sudo pip install sys

sudo pip install os

sudo pip install datetime
```

The **`mpi4py`** package requires a pre-installed and working MPI distribution:

`https://mpi4py.scipy.org/docs/usrman/install.html`.

On a system running **Ubuntu 14.04.5 Linux**, we have found that all required packages for the **`mpi4py`** module may be installed using `apt-get`:

```
sudo apt-get install python-mpi4py
```

On a desktop PC running **Windows 10 with an Ubuntu Linux subsystem**, we installed the **`mpi4py`** module using the following sequence of commands:

1. **Optional**: If `python-mpi4py` was already installed using `apt-get`, first run:

```
sudo apt-get remove python-mpi4py
```

2. **Optional**: Ensure that `apt-get` is up-to-date:

```
sudo apt-get update

sudo apt-get -y upgrade
```

3. **Optional**: Search for available `OpenMPI` packages by running:

```
apt-cache search openmpi
```

4. Install `OpenMPI`:

```
sudo apt-get install libopenmpi-dev
```

5. Install `mpi4py` using `python-pip`:

```
sudo pip install mpi4py
```

Note: It is possible that problems may arise at run-time related to the `SciPy` sub-package `special`. In the */LOADGF/LN/fundamental_solutions_homsph.py* function, the line near the top reading "from scipy.special import **sph_jn**" may need to be modified to read "from scipy.special import **spherical_jn as sph_jn**".

1.3 Additional Notes on Optional `Python` Modules

The following modules are not required to use the primary functionalities of `LoadDef` (e.g., computing Love numbers and Green's functions), but are required to import and interpret certain load and coastline databases. The `basemap` addition to the `mpl_toolkits` module helps to improve the efficiency of grid interpolation for the convolution of Green's functions with a load model.

If not using `Anaconda`, the optional modules may be installed using `apt-get` and `python-pip`.

For example, on Ubuntu Linux:

```
sudo apt-get install python-mpltoolkits.basemap
sudo pip install pyshp
sudo pip install shapely
sudo pip install pyproj
sudo pip install netCDF4
```

The `pyshp` module includes the `shapefile` module.

If you do not have access to **`mpl_toolkits.basemap`**:

`/LoadDef/CONVGF/CN/interpolate_load.py`

must be modified by commenting *out* the import of **`mpl_toolkits.basemap`** (top of script), commenting *out* the reference to the `basemap interp` function, and commenting *in* the reference to the `scipy interpolate.RectBivariateSpline` function.

If you do not have access to **`shapefile`**, **`shapely`**, and/or **`pyproj`**:

`/LoadDef/GRDGEN/utility/process_add_shapefile.py`, which generates a land-sea mask for Antarctica, will not be available.

If you do not have access to **`netCDF4`**, you will not be able to read certain load models in netCDF format. You may also comment out the import of **`netCDF4`** in `CONVGF/utility/read_AmpPha.py`, and instead use load models in plain-text format.

2 Data Access

2.1 Land-Sea Mask

Global bathymetric and topographic datasets, such as ETOPO1 (**?**), may be used to generate a land-sea mask. The land-sea mask may then be refined using local models where available. For Antarctica, for example, coastline information is available from the Scientific Committee on Antarctic Research (SCAR) Antarctic Digital Database (ADD).

ETOPO1: `https://www.ngdc.noaa.gov/mgg/global/global.html`

ADD: `http://www.add.scar.org/home/add7`

2.2 Surface Mass Loading

2.2.1 Global Ocean Tide Models

FES2014 is the most recent installment in a series of tidal atlases produced by the "French Tidal Group" under the generic name of "Finite Element Solution (FES)" (**???**). The FES2014 atlas, generated by assimilating satellite altimetry observations into a global hydrodynamic model, provides the complex-valued amplitudes of 32 tidal harmonics on a $0.0625° \times 0.0625°$ grid. The model may be accessed from AVISO after site registration and license agreement:

`www.aviso.altimetry.fr/en/data/products/auxiliary-products/global-tide-fes.html`.

TPXO8-Atlas, developed at Oregon State University, integrates a global tidal solution (TPXO8) with a multitude of high-resolution local solutions produced for shelf and coastal regions at several locations around the world (**???**). Tidal amplitudes and phases are provided on a global grid of $\frac{1}{30}^{\circ}$ resolution for the harmonics M_2, S_2, N_2, K_2, K_1, O_1, P_1, Q_1, and M_4. Additional long-period and compound tidal harmonics (M_f, M_m, MS_4, and MN_4) are provided on a $\frac{1}{6}^{\circ}$-resolution grid. Satellite altimetry data from the TOPEX/Poseidon and Jason missions are assimilated into global solutions of the Laplace Tidal Equations (LTEs) to generate TPXO8. Tide gauges are primarily used for validation, particularly in the shelf and coastal regions. The model may be accessed from Oregon State University: `volkov.oce.orst.edu/tides/`.

EOT11A, a purely empirical ocean tide model generated from a harmonic analysis of multi-mission satellite altimetry data, provides amplitudes and phases for thirteen tidal harmonics (**?**). The harmonic analysis was performed on the combined altimetry residuals, using FES2004 (**?**) as a reference model. The model may be accessed from: `ftp://dgfi.tum.de/pub/EOT11a/`.

HAMTIDE11A (Hamburg direct data Assimilation Methods for TIDEs) provides amplitude and phase information for nine tidal harmonics on a regular grid of $0.125°$ resolution. A direct minimization of model and observational residuals using least-squares inversion generates the HAMTIDE11A atlas. The model may be accessed from: `ftp://ftp-icdc.cen.uni-hamburg.de/hamtide/`.

GOT4.10c, available by request from Dr. Richard Ray at the Goddard Space Flight Center (GSFC), is the latest release in a series of global ocean tide models developed by Dr. Ray at GSFC (**???**). The model is generated by performing a harmonic analysis of satellite altimetry residuals with respect to an *a priori* hydrodynamic model. The satellite altimetry observations of sea-surface height were corrected for tidal geocenter variations induced by the loading (**?**).

DTU10, developed and distributed by The Technical University of Denmark, is a global ocean tide model with $0.125° \times 0.125°$ resolution. The model may be accessed from: `http://www.space.dtu.dk/english/Research/Scientific\_data\_and\_models/Global\_Ocean\_Tide\_Model`.

2.2.2 Local Ocean Tide Models

Tidal predictions near coastlines and in shallow seas are less reliable than in the open ocean due to nonlinearities in the tidal equations and limited empirical constraints (e.g., **?**). The high-resolution, local ocean tide models may be obtained from Oregon State University (**???**): `volkov.oce.orst.edu/tides/region.html`.

Additional providers of local tidal models are listed in **?**, and the references therein. Local models for Canadian waters have also been developed by **?**.

2.2.3 Non-Tidal Ocean Models

ECCO2: NASA's *Estimating the Circulation and Climate of the Oceans* (ECCO) consortium provides time series of non-tidal ocean loading (NTOL) on global grids (e.g., **?**). A recent installment of the consortium, the ECCO2 model, includes daily estimates of ocean bottom-pressure potential anomalies on a global grid of resolution $0.25° \times 0.25°$ from 1992 onward. The ECCO2 estimates of bottom-pressure anomalies are diagnostic quantities derived from anomalous water-column height (e.g., **?**). It is recommended that global-spatial and temporal averages be removed from the ECCO2 bottom-pressure anomalies to account for drifts in the global balance of evaporation and precipitation (D. Menemenlis and H. Zhang, personal communication). Moreover, the ECCO2 simulations of ocean-bottom pressure anomalies do not include the effects of atmospheric or tidal forcing (D. Menemenlis, personal communication). To first-order, however, the ocean-bottom pressure does not change with variations in atmospheric pressure due to the inverted barometer effect, but could have issues on timescales less than a few days. `https://ecco.jpl.nasa.gov`

2.2.4 Atmospheric Models

Estimates of the atmospheric surface pressure on global grids are routinely computed by reanalysis centers such as the European Centre for Medium-Range Weather Forecasts (ECMWF) and the National Centers for Environmental Prediction/National Center for Atmospheric Research (NCEP/NCAR).

ECMWF: ECMWF models are distributed with six-hour temporal resolution on global grids of $0.75° \times 0.75°$ spatial resolution. ECMWF uses data assimilation techniques to incorporate a range of space- and land-based empirical weather measurements into general circulation models. Variations in atmospheric pressure tend to be largest at high latitude due to geostrophic force balance (e.g., **?**). `https://www.ecmwf.int/en/research/climate-reanalysis`

NCEP/NCAR: NCEP/NCAR models are distributed with six-hour temporal resolution on global grids. `https://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.html`

3 Main Programs

`LoadDef` includes four main programs, located in the **`working`** directory:

- **`run_ln.py`** | Computes potential, load, shear, and stress Love numbers
- **`run_pl.py`** | Computes partial derivatives of Love numbers
- **`run_gf.py`** | Computes displacement, gravity, tilt, and strain load Green's functions
- **`run_cn.py`** | Computes surface displacements caused by surface mass loading

3.1 Love Numbers

The program **`run_ln.py`**, located in the **`working`** directory, computes potential, load, shear, and stress Love numbers. The input planetary model is assumed to be spherically symmetric, non-rotating, elastic, and isotropic (SNREI), and should be in the following format: radius (km), Vp (km/s), Vs (km/s), density (g/cc).

REQUIRED INPUTS:

1. **Planet Model** : radius (km), Vp (km/s), Vs (km/s), density (g/cc)

   :: Assumed to be spherically symmetric, non-rotating, elastic, and isotropic (SNREI)

   :: May include one internal fluid layer (e.g. Earth's outer core), or be entirely solid

OPTIONAL INPUTS AND DEFAULTS:

1. **`startn`** : Starting value for spherical harmonic degrees (for Love numbers)

   Default is 0

2. **`stopn`** : Ending value for spherical harmonic degrees (for Love numbers)

   Default is 10000

3. **`period_hours`** : Forcing period (in hours)

   Default is 12.42 (M_2 period)

4. **`r_min`** : Minimum radius for variable planetary structural properties (m)

   Default is 1000

5. **`interp_emod`** : Optionally interpolate the planetary model to a different resolution

   Default is False (see LOADGF/LN/prepare_planet_model.py)

6. **`kx`** : Order of the spline fit for the planetary model (1=linear; 3=cubic)

   Default is 1 (recommended)

7. **`delim`** : Delimiter for the planetary model file

   Default is None (Whitespace)

8. **`inf_tol`** : Integration starting radius, r, for which $\left(\frac{r}{a}\right)^n$ drops below tolerance 'inf_tol'

   Default is 1E-5

9. **`rel_tol`** : Integration tolerance level (relative tolerance)

   Default is 1E-13

10. **`abs_tol`** : Integration tolerance level (absolute tolerance)

    Default is 1E-13

11. **`backend`** : Specify ODE solver

    :: Recommended to only use `dop853` or `dopri5` solvers, since they integrate only to a specified stopping point (no overshoot, like `lsoda` and `vode`)

    Default is `dop853`

12. **`nstps`** : Specify maximum number of (internally defined) steps allowed during each call to the ODE solver

    :: For more information, see Scipy.Integrate.Ode manual pages

    Default is 3000

13. **`num_soln`** : Set number of solutions for each integration (integer)

    :: Note that integration step size is adaptive based on the specified tolerance, but solutions are only computed at regular intervals determined by this user-specified value

Default is 100

14. `G` : Universal gravitational constant

    Default is 6.672E-11 (m^3 kg^{-1} s^{-2})

15. `nmaxfull` : Maximum spherical harmonic degree for which integration will be performed through the full planet. Beyond nmaxfull, the integration will begin in the mantle.

    Default is None (the parameter will be estimated from inf_tol within integrate_odes.py)

16. `file_out` : Extension for the output files

    Default is".txt"

OUTPUT FILES:

1. **Load Love Number Output File**

   "output/Love_Numbers/LLN/lln_" + `file_out`

2. **Potential Love Number Output File**

   "output/Love_Numbers/PLN/pln_" + `file_out`

3. **Stress Love Number Output File**

   "output/Love_Numbers/STR/str_" + `file_out`

4. **Shear Love Number Output File**

   "output/Love_Numbers/SHR/shr_" + `file_out`

OUTPUT VARIABLES:

1. Spherical harmonic degrees, n (array)

2. Vertical-displacement load Love numbers, h' (array)

3. Horizontal-displacement load Love numbers, nl' (array)

4. Gravitational-potential load Love numbers, nk' (array)

5. First-order asymptotic solution for h' (float)
6. First-order asymptotic solution for nl' (float)
7. First-order asymptotic solution for nk' (float)
8. Second-order asymptotic solution for h' (float)
9. Second-order asymptotic solution for nl' (float)
10. Second-order asymptotic solution for nk' (float)
11. Vertical-displacement potential/tide Love numbers, h (array)
12. Horizontal-displacement potential/tide Love numbers, nl (array)
13. Gravitational-potential potential/tide Love numbers, nk (array)
14. Vertical-displacement stress Love numbers, h_{str} (array)
15. Horizontal-displacement stress Love numbers, nl_{str} (array)
16. Gravitational-potential stress Love numbers, nk_{str} (array)
17. Vertical-displacement shear Love numbers, h_{shr} (array)
18. Horizontal-displacement shear Love numbers, nl_{shr} (array)
19. Gravitational-potential shear Love numbers, nk_{shr} (array)
20. Planet radius (float, m)
21. Planet mass (float, kg)
22. Integration points (array of radial distances)
23. Solutions for spheroidal deformation due to loading as a function of radius (array)
24. Solutions for spheroidal deformation due to a potential field as a function of radius (array)
25. Solutions for spheroidal deformation due to stress conditions as a function of radius (array)

26. Solutions for spheroidal deformation due to shear forcing as a function of radius (array)

27. Lamé parameter λ at the surface (float, Pa)

28. Lamé parameter μ at the surface (float, Pa)

3.2 Love Number Partial Derivatives

The program **`run_pl.py`**, located in the **`working`** directory, computes the partial derivates of Love numbers with respect to the bulk modulus, the shear modulus, and density (**??**). The program initially calls the same subroutines as **`run_ln.py`** to compute the solution vectors (solutions to the equations of motion as a function of radius), and then uses the solution vectors to compute partial derivatives as a function of radius. As a result, many of the inputs and outputs are the same as for **`run_ln.py`**. Note that if a specific file extension is specified and is identical to an extension used to compute Love numbers during a previous run of **`run_ln.py`**, the old Love number files will be overwritten.

REQUIRED INPUTS:

1. **Planet Model** : radius (km), Vp (km/s), Vs (km/s), density (g/cc)

   :: Assumed to be spherically symmetric, non-rotating, elastic, and isotropic (SNREI)

   :: May include one internal fluid layer (e.g. Earth's outer core), or be entirely solid

OPTIONAL INPUTS AND DEFAULTS:

1. **`startn`** : Starting value for spherical harmonic degrees (for Love numbers)

   Default is 0

2. **`stopn`** : Ending value for spherical harmonic degrees (for Love numbers)

   Default is 10000

3. **`period_hours`** : Forcing period (in hours)

   Default is 12.42 (M_2 period)

4. **`r_min`** : Minimum radius for variable planetary structural properties (m)

   Default is 1000

5. **`interp_emod`** : Optionally interpolate the planetary model to a different resolution

   Default is False (see LOADGF/LN/prepare_planet_model.py)

6. **`kx`** : Order of the spline fit for the planetary model (1=linear; 3=cubic)

   Default is 1 (recommended)

7. **`delim`** : Delimiter for the planetary model file

   Default is None (Whitespace)

8. **`inf_tol`** : Integration starting radius, r, for which $\left(\frac{r}{a}\right)^n$ drops below tolerance 'inf_tol'

   Default is 1E-5

9. **`rel_tol`** : Integration tolerance level (relative tolerance)

   Default is 1E-13

10. **`abs_tol`** : Integration tolerance level (absolute tolerance)

    Default is 1E-13

11. **`backend`** : Specify ODE solver

    :: Recommended to only use `dop853` or `dopri5` solvers, since they integrate only to a specified stopping point (no overshoot, like `lsoda` and `vode`)

    Default is `dop853`

12. **`nstps`** : Specify maximum number of (internally defined) steps allowed during each call to the ODE solver

    :: For more information, see Scipy.Integrate.Ode manual pages

    Default is 3000

13. **`num_soln`** : Set number of solutions for each integration (integer)

    :: Note that integration step size is adaptive based on the specified tolerance, but solutions are only computed at regular intervals determined by this user-specified value

    Default is 100

14. `G` : Universal gravitational constant

    Default is 6.672E-11 (m^3 kg^{-1} s^{-2})

15. `nmaxfull` : Maximum spherical harmonic degree for which integration will be performed through the full planet. Beyond nmaxfull, the integration will begin in the mantle.

    Default is None (the parameter will be estimated from inf_tol within integrate_odes.py)

16. `plot_figure` : Reproduce Figure 1 from **?** as a sanity check.

    Default is False

17. `par_out` : Extension for the output filenames

    Default is "partials.txt"

OUTPUT FILES:

1. **Load Love Number Output File**

   "output/Love_Numbers/LLN/lln_" + `par_out`

2. **Potential Love Number Output File**

   "output/Love_Numbers/PLN/pln_" + `par_out`

3. **Stress Love Number Output File**

   "output/Love_Numbers/STR/str_" + `par_out`

4. **Shear Love Number Output File**

   "output/Love_Numbers/SHR/shr_" + `par_out`

5. **Partial Derivative Output File** (one file per spherical-harmonic degree)

   "output/Love_Numbers/Partials/lln_n" + n + "_" + `par_out`

OUTPUT VARIABLES:

1. $\frac{\partial}{\partial\mu}h_{\text{potential}}(r)$

2. $\frac{\partial}{\partial\mu}l_{\text{potential}}(r)$

3. $\frac{\partial}{\partial \mu} k_{\text{potential}}(r)$
4. $\frac{\partial}{\partial \kappa} h_{\text{potential}}(r)$
5. $\frac{\partial}{\partial \kappa} l_{\text{potential}}(r)$
6. $\frac{\partial}{\partial \kappa} k_{\text{potential}}(r)$
7. $\frac{\partial}{\partial \rho} h_{\text{potential}}(r)$
8. $\frac{\partial}{\partial \rho} l_{\text{potential}}(r)$
9. $\frac{\partial}{\partial \rho} k_{\text{potential}}(r)$
10. $\frac{\partial}{\partial \mu} h_{\text{load}}(r)$
11. $\frac{\partial}{\partial \mu} l_{\text{load}}(r)$
12. $\frac{\partial}{\partial \mu} k_{\text{load}}(r)$
13. $\frac{\partial}{\partial \kappa} h_{\text{load}}(r)$
14. $\frac{\partial}{\partial \kappa} l_{\text{load}}(r)$
15. $\frac{\partial}{\partial \kappa} k_{\text{load}}(r)$
16. $\frac{\partial}{\partial \rho} h_{\text{load}}(r)$
17. $\frac{\partial}{\partial \rho} l_{\text{load}}(r)$
18. $\frac{\partial}{\partial \rho} k_{\text{load}}(r)$
19. $\frac{\partial}{\partial \mu} h_{\text{shear}}(r)$ ($h_{\text{stress}}(r)$ for $n = 1$)
20. $\frac{\partial}{\partial \mu} l_{\text{shear}}(r)$ ($l_{\text{stress}}(r)$ for $n = 1$)
21. $\frac{\partial}{\partial \mu} k_{\text{shear}}(r)$ ($k_{\text{stress}}(r)$ for $n = 1$)
22. $\frac{\partial}{\partial \kappa} h_{\text{shear}}(r)$ ($h_{\text{stress}}(r)$ for $n = 1$)
23. $\frac{\partial}{\partial \kappa} l_{\text{shear}}(r)$ ($l_{\text{stress}}(r)$ for $n = 1$)

24. $\frac{\partial}{\partial\kappa}k_{\text{shear}}(r)$ ($k_{\text{stress}}(r)$ for $n = 1$)

25. $\frac{\partial}{\partial\rho}h_{\text{shear}}(r)$ ($h_{\text{stress}}(r)$ for $n = 1$)

26. $\frac{\partial}{\partial\rho}l_{\text{shear}}(r)$ ($l_{\text{stress}}(r)$ for $n = 1$)

27. $\frac{\partial}{\partial\rho}k_{\text{shear}}(r)$ ($k_{\text{stress}}(r)$ for $n = 1$)

3.3 Load Green's Functions

The program **`run_gf.py`**, located in the **`working`** directory, computes displacement, gravity, tilt, and strain Green's functions for load-induced spheroidal deformation.

REQUIRED INPUTS:

1. **Love-Number File**: Output *load* Love number file from **`run_ln.py`**

OPTIONAL INPUTS AND DEFAULTS:

1. **`theta`** : Angular distances at which to compute the load Green's functions (list format; degrees)

   Default is:

   `theta` = [0.0001, 0.001, 0.01, 0.02, 0.03, 0.04, 0.06, 0.08, 0.1, 0.16, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0, 1.2, 1.6, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 12.0, 16.0, 20.0, 25.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0, 110.0, 120.0, 130.0, 140.0, 150.0, 160.0, 170.0, 180.0]

2. **`a`** : Planet mean radius (m)

   Default is 6371000.

3. **`me`** : Planet mean mass (kg)

   Default is 5.976E24

4. **`lmda_surface`** : Value of lambda elastic (Lamé) parameter at planet's surface (Pa)

   Default is 3.422E10

5. **mu_surface** : Value of the shear modulus at planet's surface (Pa)

   Default is 2.662E10

6. **g_surface** : Value of acceleration due to gravity at planet's surface (m s^{-2})

   Default is 9.81

7. **disk_factor** : Include a disk factor to improve convergence of the LGFs in the far field

   Default is True

8. **angdist** : Angular distance (in degrees) beyond which the disk factor will be included

   Default is 10 (if disk_factor = True)

9. **disk_size** : Radius of the disk in degrees

   Default is 0.1 (if disk_factor = True)

10. **apply_taper** : Option to apply a taper to the series summation in the computation of the LGFs.

    The taper mimics the recursive averaging desribed in Guo et al. 2004 (Eqs. 23 and 24).

    Default is True

11. **grn_out** : Extension for the output Green's function files

    Default is 'grn.txt'

OUTPUT FILES:

1. **Green's Function Output File (CE Frame)**

   "output/Greens_Functions/ce_" + **grn_out**

2. **Green's Function Output File (CM Frame)**

   "output/Greens_Functions/cm_" + **grn_out**

3. **Green's Function Output File (CF Frame)**

   "output/Greens_Functions/cf_" + **grn_out**

OUTPUT VARIABLES:

1. Vertical-displacement load Green's function, CE frame
2. Horizontal-displacement load Green's function, CE frame
3. Vertical-displacement load Green's function, CE frame (normalized by $10^{12}a\theta$, where a is planet's radius in meters and θ is angular distance in radians)
4. Horizontal-displacement load Green's function, CE frame (normalized by $10^{12}a\theta$)
5. Vertical-displacement load Green's function, CM frame
6. Horizontal-displacement load Green's function, CM frame
7. Vertical-displacement load Green's function, CM frame (normalized by $10^{12}a\theta$)
8. Horizontal-displacement load Green's function, CM frame (normalized by $10^{12}a\theta$)
9. Vertical-displacement load Green's function, CF frame
10. Horizontal-displacement load Green's function, CF frame
11. Vertical-displacement load Green's function, CF frame (normalized by $10^{12}a\theta$)
12. Horizontal-displacement load Green's function, CF frame (normalized by $10^{12}a\theta$)
13. Gravity load Green's function, indirect (elastic) component, CE frame
14. Gravity load Green's function, indirect (elastic) component, CE frame (normalized by $10^{18}a\theta$)
15. Gravity load Green's function, indirect (elastic) component, CM frame
16. Gravity load Green's function, indirect (elastic) component, CM frame (normalized by $10^{18}a\theta$)
17. Gravity load Green's function, indirect (elastic) component, CF frame
18. Gravity load Green's function, indirect (elastic) component, CF frame (normalized by $10^{18}a\theta$)
19. Tilt load Green's function, indirect (elastic) component, CE frame
20. Tilt load Green's function, indirect (elastic) component, CE frame (normalized by $10^{12}(a\theta)^2$)

21. Tilt load Green's function, indirect (elastic) component, CM frame

22. Tilt load Green's function, indirect (elastic) component, CM frame (normalized by $10^{12}(a\theta)^2$)

23. Tilt load Green's function, indirect (elastic) component, CF frame

24. Tilt load Green's function, indirect (elastic) component, CF frame (normalized by $10^{12}(a\theta)^2$)

25. Strain load Green's function, $\theta\theta$ component, CE frame

26. Strain load Green's function, $\lambda\lambda$ component, CE frame

27. Strain load Green's function, rr component, CE frame

28. Strain load Green's function, $\theta\theta$ component, CE frame (normalized by $10^{12}(a\theta)^2$)

29. Strain load Green's function, $\lambda\lambda$ component, CE frame (normalized by $10^{12}(a\theta)^2$)

30. Strain load Green's function, rr component, CE frame (normalized by $10^{18}a\theta$)

31. Strain load Green's function, $\theta\theta$ component, CM frame

32. Strain load Green's function, $\lambda\lambda$ component, CM frame

33. Strain load Green's function, rr component, CM frame

34. Strain load Green's function, $\theta\theta$ component, CM frame (normalized by $10^{12}(a\theta)^2$)

35. Strain load Green's function, $\lambda\lambda$ component, CM frame (normalized by $10^{12}(a\theta)^2$)

36. Strain load Green's function, rr component, CM frame (normalized by $10^{18}a\theta$)

37. Strain load Green's function, $\theta\theta$ component, CF frame

38. Strain load Green's function, $\lambda\lambda$ component, CF frame

39. Strain load Green's function, rr component, CF frame

40. Strain load Green's function, $\theta\theta$ component, CF frame (normalized by $10^{12}(a\theta)^2$)

41. Strain load Green's function, $\lambda\lambda$ component, CF frame (normalized by $10^{12}(a\theta)^2$)

42. Strain load Green's function, rr component, CF frame (normalized by $10^{18}a\theta$)
43. Direct (Newtonian) effect of gravity (normalized by $10^{18}a\theta$)
44. Direct (Newtonian) effect of tilt (normalized by $10^{12}(a\theta)^2$)

3.4 Displacement Response to Surface Mass Loading

The program **run_cn.py**, located in the **working** directory, computes predicted surface displacements caused by surface mass loading. The required inputs to **run_cn.py** include displacement load Green's functions for a SNREI planetary model, which may be generated using **run_gf.py**, and a load model.

REQUIRED INPUTS:

1. **rfm** : Reference frame (**?**)

   :: Used for input and output filenames

2. **grn_file** : Displacement load Green's function file output from **run_gf.py**

3. **norm_flag** :

   :: True = Displacement load Green's functions normalized by the **?** convention

   Format: [θ, u_{norm}, v_{norm}]

   θ is the angular distance (degrees)

   u_{norm} is the vertical-displacement load Green's function (normalized by $10^{12}a\theta$, with θ in radians)

   v_{norm} is the horizontal-displacement load Green's function (normalized by $10^{12}a\theta$, with θ in radians)

   :: False = Load Green's function file output directly from **run_gf.py**

4. **loadfile_directory** : Full path to the directory containing the input load files

5. **loadfile_prefix** : Prefix for the input load files

   :: The load-file directory will be searched for all files starting with this prefix

   :: Note: For load files organized by date, the end of the filename must be in the format yyyymmddhhmnsc.txt

:: Note: For load files not organized by date, the files may be organized by a unique filename ending (e.g. tidal harmonic)

:: Note: Output filenames will be determined in part by the extension following the last underscore character in the load-file name

:: **General format for the load files**: [Latitude (degrees, +N), Longitude (degrees, +E), Amplitude (m), Phase (degrees)]

6. **`loadfile_format`** : File format for the load models

   :: Acceptable formats include: “nc” (netCDF) and “txt” (plain text)

   :: See: /GRDGEN/load_files/ directory for scripts to generate `LoadDef`-compatible files of both formats for common load models.

7. **`time_series`** : Flag for whether or not the files are organized by date

   :: False = All files matching the directory and prefix criteria will be analyzed

   :: True = Files are organized by date and will be filtered by date

   :: Note: If true, filenames must end in the format yyyymmddhhmnsc.txt

8. **`frst_date`** : First date of the time series [year, month, day, hour, minute, second]

   :: Note: Only used if **`time_series`** is set to true

9. **`last_date`** : Last date of the time series [year, month, day, hour, minute, second]

   :: Note: Only used if **`time_series`** is set to true

10. **`regular`** : Flag for whether or not the load grid is regular (rectangular)

    :: True = Grid is regular (speeds up interpolation)

    :: False = Grid is not regular (if unsure, select false)

11. **`ldens`** : Density of the surface mass load (kg m^{-3})

    :: For oceanic loads, suggested density is 1030.0 (default)

    :: For atmospheric loads, suggested density is 1.0

12. **`lsmask_type`** : Type of land-sea mask

:: 0 = Do not mask either the ocean or the land (retain full model)

:: 1 = Mask out the land (retain the ocean load)

:: 2 = Mask out the oceans (retain the load over the land areas)

13. **`lsmask_file`** : Full path and filename of the land-sea mask

    :: Format = [Latitude (degrees, +N), Longitude (degrees, +E), Mask Value (0=ocean; 1=land)]

    :: Note: The land-sea mask may be irregular and sparse

14. **`sta_file`** : File containing the prediction locations for the convolution

    :: Format: [Latitude (degrees, +N), Longitude (degrees, +E), Station/Point Name (string)]

15. **`outstr`** : Additional string to append to output filenames (e.g. "_2019")

    :: Format: String

OPTIONAL INPUTS AND DEFAULTS:

1. **`load_density`** : Density of the surface mass load (kg m^{-3})

   Default is 1030.0

2. **`rad`** : Mean radius of the planet (m)

   Default is 6371000.

3. **Mesh Parameters** :

   (a) **`delinc1`** : Angular-distance (degrees) increment for inner zone

   Default is 0.0002

   (b) **`delinc2`** : Angular-distance (degrees) increment for zone 2

   Default is 0.001

   (c) **`delinc3`** : Angular distance (degrees) increment for zone 3

   Default is 0.01

   (d) **`delinc4`** : Angular distance (degrees) increment for zone 4

   Default is 0.1

(e) **`delinc5`** : Angular distance (degrees) increment for zone 5

Default is 0.5

(f) **`delinc6`** : Angular distance (degrees) increment for outer zone

Default is 1.0

(g) **`izb`** : Inner zone boundary (degrees; 0<**`izb`**<**`z2b`**)

Default is 0.02

(h) **`z2b`** : Zone 2 boundary (degrees; **`izb`**<**`z2b`**<**`z3b`**)

Default is 0.05

(i) **`z3b`** : Zone 3 boundary (degrees; **`z2b`**<**`z3b`**<**`z4b`**)

Default is 1.0

(j) **`z4b`** : Zone 4 boundary (degrees; **`z3b`**<**`z4b`**<**`z5b`**)

Default is 10.0

(k) **`z5b`** : Zone 5 boundary (degrees; **`z4b`**<**`z5b`**<180)

Default is 90.0

(l) **`azminc`** : Azimuthal increment (degrees)

Default is 0.1

(m) **`mass_cons`** : Option to enforce mass conservation by removing the spatial mean from the load grid

Default is False

OUTPUT FILES:

1. Predicted displacements at the specified convolution grid-point/station locations for each load file
   "output/Convolution/cn_" + **unmasked region** + "_" + **`cnv_out`**

OUTPUT VARIABLES:

1. East-component amplitude (mm)

2. East-component phase (degrees)

3. North-component amplitude (mm)
4. North-component phase (degrees)
5. Vertical-component amplitude (mm)
6. Vertical-component phase (degrees)

3.5 Formatting Surface Mass Load Models for Input to `LoadDef`

The **general format for the load model** must be:

Latitude [degrees, +N], Longitude [degrees, +E], Amplitude [m], Phase [degrees]

If the load is not a harmonic (e.g. not of tidal origin), then the phase should be set to zero.

Load grids may be provided in netCDF or plain-text format (see **`GRDGEN/load_files`** for automatic conversion of common datasets). Load models come in a wide variety of formats, and conversion to a `LoadDef`-standard format prior to input into the convolution allows for simplified program modules and greater ease of generating custom models. Note that small differences (e.g. fraction of a micron) can arise in the predicted load-induced surface displacements computed by `run_cn.py`, since the precision of the floating point values written to the netCDF and plain-text files can be different.

Note: In general, `LoadDef` is not currently set-up to handle very high-resolution load models, unless they are only local in extent. The memory usage can increase dramatically for very high-resolution grids, particularly during the grid interpolations. That said, `LoadDef` can currently handle most global-scale ocean-tide model grids as well as modern atmospheric and non-tidal oceanic pressure grids on modest computational systems. In addition to memory and storage, run-time also increases with finer grid resolution.

Scripts within the **`GRDGEN/load_files`** folder are already set up to generate `LoadDef`-formatted grids from common datasets, such as the TPXO8-Atlas and FES2014 ocean-tide models, ECMWF atmospheric models, and ECCO2 non-tidal oceanic-pressure models.

4 Examples

Assuming that the required `Python` modules have been properly installed (see Sec. 1), the `LoadDef working` directory contains prepared example scripts that illustrate the main functions of the software.

The example scripts are **`run_ln.py`**, **`run_pl.py`**, **`run_gf.py`**, and **`run_cn.py`**, which compute Love numbers, partial derivatives of Love numbers, load Green's functions, and predicted load-induced surface displacements, respectively. The `input` directory contains an Earth model (PREM), a pre-generated land-sea mask, and descriptions on how to access common load models. Once downloaded, many of the common load models may be converted into `LoadDef` format using utility scripts in the `GRDGEN/load_files/` folder. The example scripts are already set up to derive Love numbers and load Green's functions based on the provided data files in the `input` directory, with the exception of **`run_cn.py`**, which requires that a load model be downloaded first and then converted to `LoadDef` format; more details on the conversion procedure are discussed below.

In general, the scripts *should* be run in a particular order. An output file from the Love-number program **`run_ln.py`**, for example, may be used as input to the Green's-function program **`run_gf.py`**. The Green's functions are then used as input to the convolution program **`run_cn.py`**. Both **`run_ln.py`** and **`run_pl.py`** may be run simply by specifying an Earth model as input (nothing more is required).

All scripts are `MPI` compatible to reduce overall computation time. To run the **`run_ln.py`** script, for example, using `X` number of parallel processors:

```
mpirun -np X run_ln.py
```

To run the script in serial-processing mode:

```
mpirun -np 1 run_ln.py
```

or, on some systems, you may simply type:

```
run_ln.py
```

The following sections provide more detailed information on each program.

4.1 Computing Love Numbers

To demonstrate how to compute Love numbers using `LoadDef`, we have prepared a script in the `working` directory called `run_ln.py` to generate Love numbers based on PREM structure.

The program is `MPI` compatible to improve computational efficiency. From within the `working` directory, the script may be run by typing:

```
mpirun -np X run_ln.py
```

where `X` specifies the number of processors to use. For serial computations, specify `X` = 1. The specified number of processors should not exceed the number of jobs to be run.

Note that the top of the script includes a section for user inputs. An input Earth model is required, and we have selected PREM as an example. We also specify an optional parameter: an extension for the output filename. Without specifying a specific extension, the program would have defaulted to ".txt". To override the default value, we had to specify the parameter within the call to the main Love-number function (`compute_love_numbers.main`), which may be found at the bottom of the `run_ln.py` script.

For a list of additional optional input parameters, see Sec. 3.1. Importantly, the default integration parameters, including relative and absolute convergence tolerances, may be easily changed by passing in alternative values for the appropriate variables.

The program generates four output files: potential (PLN), load (LLN), shear (SHR), and stress (STR) Love numbers. Each file contains a particular type of Love numbers (PLN, LLN, SHR, or STR) for each spherical harmonic degree selected for the computation. By default, the program computes Love numbers from $n = 0$ to $n = 10000$, though the range may be modified using optional input parameters.

The output files are placed in an `output` directory, which will be created automatically if one does not already exist. The results from `run_ln.py` will specifically be placed under `output/Love_Numbers/`.

4.2 Computing Love-Number Partial Derivatives

To demonstrate how to compute Love-number partial derivatives using `LoadDef`, we have prepared a script in the `working` directory called `run_pl.py` to generate partial derivatives (with respect to the elastic moduli and density) based on PREM structure.

The program is `MPI` compatible to improve computational efficiency. From within the `working` directory, the script may be run by typing:

```
mpirun -np X run_pl.py
```

where `X` specifies the number of processors to use. For serial computations, specify `X` = 1. The specified number of processors should not exceed the number of jobs to be run.

The program `run_pl.py` only requires an Earth model as input (specified in the "user inputs" section at the top of the script). To compute partial derivatives, the program first computes Love numbers (using the `compute_love_numbers.main` function) and then uses the output parameters as input to a partial-derivative function (`compute_ln_partials.main`). Thus, the program calls the same subroutines to compute the Love numbers as `run_ln.py`. That said, it can still be worthwhile to have both programs (`run_ln.py` and `run_pl.py`). In particular, use-cases not requiring partial derivatives have no need for the additional functionality of `run_pl.py`. Furthermore, although `run_pl.py` does generate Love-number output files, some use-cases might only require partial derivatives for a few spherical-harmonic degrees. In this case, the number of spherical-harmonic degrees contained within the Love-number output file would be insufficient for input to the Green's-function program. As with `run_ln.py`, the range of spherical-harmonic degrees computed may be specified as optional inputs. In `run_pl.py`, we demonstrate how to restrict the range to $n = 0 – 4$. Partial derivatives will only be computed for degrees within the specified range.

Whenever partial derivatives for $n = 1$ are computed, a figure called `Okubo_Endo_Fig1.eps` may optionally be generated and placed under `output/Love_Numbers/Partials/`. The figure reproduces Figure 1 from **?** as a sanity check. The block of code used to generate the figure may be activated or inactivated within the script: `LOADGF/PL/ln_partials.py`.

Additional output files include partial derivatives for the load, potential, and shear Love numbers. All results are placed under the folder `output/Love_Numbers/Partials/`. The prefix of each filename indicates whether the partial derivatives correspond to load (lln), potential (pln), or shear (sln) Love numbers. For the case of $n = 1$, the partial derivates with the prefix "sln" correspond instead to stress Love numbers. One file is output for each type of Love number (lln, pln, or sln) and each spherical harmonic degree.

In addition, Love-number files are created in the same way as `run_ln.py`, and will only include Love numbers for the spherical-harmonic degrees used for computation of the partial derivatives.

4.3 Computing Load Green's Functions

The script `run_gf.py`, also located in the `working` directory, is already set up to use the output from `run_ln.py` to generate load Green's functions (LGFs) for displacement, gravity, strain, and tilt in three different reference frames (CE, CM, and CF). Note that `run_ln.py` must *finish* running (and have generated the necessary output files) prior to running `run_gf.py`. Since PREM structure was used to compute the Love numbers, we specify a special file extension within the "user inputs" section at the top of the `run_gf.py` script to indicate that the LGFs correspond to PREM structure. Additional optional parameters may be specified in the call to `compute_greens_functions.main` at the bottom of the script. See Sec 3.3 for more information on the optional inputs.

The program is `MPI` compatible to improve computational efficiency. From within the `working` directory, the script may be run by typing:

```
mpirun -np X run_gf.py
```

where `X` specifies the number of processors to use. For serial computations, specify `X` = 1. The specified number of processors should not exceed the number of jobs to be run.

Three output files are generated, one per reference frame, and placed under `output/Greens_Functions`. The prefix of each filename corresponds to the reference frame (CE, CM, or CF). The columns of the output files are: angular distance from the load point ("theta", degrees); radial displacement (m/kg); horizontal displacement (m/kg); normalized radial displacement ([m/kg] $\times$ [$10^{12}\, a\theta$], where a is Earth's radius and θ is the angular distance in degrees); normalized horizontal displacement ([m/kg] $\times$ [$10^{12}\, a\theta$]); elastic component of gravity ($\times 10^{18}\, a\theta$); elastic component of tilt ($\times 10^{12}\, [a\theta]^2$); strain component ε_{rr} ($\times 10^{18}\, a\theta$); strain component $\varepsilon_{\theta\theta}$ ($\times 10^{12}\, [a\theta]^2$); strain component $\varepsilon_{\lambda\lambda}$ ($\times 10^{12}\, [a\theta]^2$); ratio of elastic to Newtonian components of gravity ($\times 10^{18}\, a\theta$); and the ratio of elastic to Newtonian components of tilt ($\times 10^{12}\, [a\theta]^2$).

Tilt and strain LGFs are the slowest to converge, and their accuracy is directly affected by the precision of the Love numbers. To further speed convergence, a disk factor may be applied in the far field. Parameters associated with the application of the disk factor may be easily changed by passing alternative values to the appropriate variables. See Sec 3.3 for more information on the optional inputs.

4.4 Generating a Land-Sea Mask

Most surface mass loads are not global in extent. The ocean tides, for example, are zero in amplitude over land areas. Although mass-load models continue to improve in resolution every year, it can still be useful to apply a land-sea mask, whereby every point covered by the mask is ensured to be zero-amplitude. As another example, the continents and the oceans respond differently to atmospheric pressure forcing, and it may therefore be desirable to only apply the atmospheric load over land areas. A land-sea mask specifies which grid points are over land and which grid points are over water.

The format for the land-sea mask used by `LoadDef` is very simple: **latitude** (degrees, +N), **longitude** (degrees, +E), and a **land-sea flag**. The flag should be zero (0) for water areas and one (1) for land areas. A utility script called `landsea_mask.py`, within the `GRDGEN/landsea_mask/` directory, may be used to convert ETOPO1, and optionally a refined Antarctic coastline, into a land-sea mask (**?**). A second utility script called `antarctic_mask.py`, also within the `GRDGEN/landsea_mask/` directory, may be used to convert the Antarctic Digital Database (ADD) into a refined land-sea mask around the Antarctic `http://www.add.scar.org/`.

4.5 Preparing a Load Model

The format for load models used by `LoadDef` is designed to be accessible and general. `LoadDef` requires either a netCDF or plain-text file with geographic coordinates (latitude, longitude) and an amplitude and phase value for each coordinate location. In other words, the format is simply [latitude, longitude, amplitude, phase]. For non-harmonic loads, the phase may simply be set to zero. When the phase is set to zero, all values in the amplitude parameter are assumed to load the planet at the same moment in time.

The netCDF and/or plain-text files for common load models may be generated using the scripts in the `GRDGEN/load_files/` folder. The netCDF files are significantly more efficient, both in terms of storage and processing speed, but the plain-text files are more straight-forward to manipulate and interpret. Note that small differences (e.g. fraction of a micron) can arise in the predicted load-induced surface displacements computed by `run_cn.py`, since the precision of the floating point values written to the netCDF and plain-text files can be different.

For predicting load-induced surface displacements using the convolution algorithm (`run_cn.py`), the load model will be linearly interpolated onto an integration mesh. The resolution of the integration mesh may be adjusted using optional input parameters to `run_cn.py`. The load value mapped to each cell of the integration mesh will be integrated over the area of the cell. The grid interpolations typically take the longest time to run (for very high-resolution grids, the interpolation might take up to a few minutes or more to complete). Thus, the resolution of the load grid, as well as the resolution of the integration mesh (an optional parameter for the convolution algorithm), affects the run-time as well as the precision. That said, the inputs (load model and integration mesh) are adjustable to meet the specific needs of individual projects.

Moreover, if the load is on a regular grid, this may be specified in the call to the convolution algorithm (within the `run_cn.py` script) to speed up the computation. For loads on irregular grids, the convolution algorithm can handle the model but will take longer to run. If the model is local (as opposed to global) in extent, then the model should be surrounded by a grid of zero-amplitude values, such that the linear-interpolation algorithm will set areas beyond the local model to zero.

4.5.1 Harmonic Loads

Examples of harmonic loads include the ocean tides. To prepare an ocean-tide model for input into `LoadDef`, you can use the `gen_otl.py` utility in the `GRDGEN/load_files/` folder. We have included instructions for downloading various ocean-tide models within the `input/Load_Models/` folder. Some of the models may be downloaded straight away, and others require the acceptance of license agreements. For scientific and/or educational purposes, the models are generally available.

The `GRDGEN/load_files/gen_otl.py` script is already set up to read in several common ocean-tide models. The script will prepare an ocean-tide model for input into `LoadDef`, as well as, optionally, for plotting using the Generic Mapping Tools (GMT) (**?**). Note that the three user-inputs at the top of the script are set to the M_2 harmonic of the TPXO8-Atlas model, which is a recently released high-resolution ocean-tide model, but the parameters may easily be changed to read in a different model or harmonic instead (see the "user inputs" section at the top of the script). Since the TPXO8-Atlas model is defined on a high-resolution global grid, the script may take a few minutes or so to write out the necessary grid files. Lower resolution grids can be prepared in far less time.

For the examples presented here, we will assume that the M_2 harmonic of the TPXO8-Atlas ocean-tide models has been retrieved by the user from Oregon State University (`http://volkov.oce.orst.edu/`

tides/tpxo8_atlas.html). The script input/Load_Models/TPXO8-Atlas/get_TPXO8.sh will download the model automatically.

The TPXO8-Atlas netCDF files should be converted to LoadDef format using GRDGEN/load_files/gen_otl.py, as described above. The grids are output to output/Grid_Files/. Two sub-directories, GMT and LoadDef, were also generated to keep the files organized. If any of the directories do not exist, then LoadDef will create them automatically. The files that will be used as input to working/run_cn.py are placed in the output/Grid_Files/nc/OTL/ directory (or the output/Grid_Files/text/OTL/ directory if opting to work with plain-text files). Note that, for each different harmonic and/or model type, variables in the "user inputs" section of the gen_otl.py utility must be modified appropriately.

4.5.2 Time-Series Loads

Examples of non-harmonic, time-dependent loads include non-tidal oceanic loads and atmospheric loads. Models are typically available at 6-hourly or daily epochs. Each model will need to be converted to LoadDef format [latitude, longitude, amplitude, phase]. For non-harmonic, time-dependent loads, the phase should be set to zero. We have included utilities in the GRDGEN/load_files/ folder that will convert a variety of common load models into LoadDef format.

Time series of non-tidal oceanic models (based on the NASA ECCO2 model) may be prepared for input into LoadDef by running the gen_ecco2.py utility. Time series of atmospheric-pressure models (based on ECMWF) may be prepared for input into LoadDef by running the gen_ecmwf.py utility. Even harmonic ocean-tide models may be converted a time-series of load models using gen_otl_tseries.py. A utility function is also available to convert NASA GRACE mass-change grids into LoadDef format: gen_grace.py.

4.6 Predicting Load-Induced Surface Displacements

Once a set of load Green's functions has been computed (using run_gf.py) and a load model has been generated in the appropriate format (using one of the scripts in the GRDGEN/load_files/ folder), then the run_cn.py script in the working directory may be used to predict surface displacements caused by the load. We currently have two examples set up in the `run_cn.py` script. Both require that LGFs were computed using `run_gf.py`, as described above (Sec. 4.3). The first example, for harmonic ocean tidal loading response, requires that the TPXO8-Atlas model be downloaded and converted into the `LoadDef` format (Sec. 4.5.1). The second example, for a time series of non-tidal oceanic loading response, requires that a time series of ECCO2 models be downloaded and converted into `LoadDef` format.

The program is `MPI` compatible to improve computational efficiency. From within the `working` directory, the script may be run by typing:

```
mpirun -np X run_cn.py
```

where `X` specifies the number of processors to use. For serial computations, specify `X` = 1. For a small number of load models and/or for tight computational-memory constraints, it may be preferable to run the script in serial-mode. The specified number of processors should not exceed the number of jobs to be run.

The first example (Example 1) may be run straightaway (assuming the appropriate LGF and load files are available). To run the second example (Example 2), simply uncomment the `loadfile_directory`, `loadfile_prefix`, and `time_series` variables associated with "Example 2" (commented lines in the "user inputs" section at the top of the script).

The run-time of the convolution program depends on the resolution of the integration mesh, resolution of the land-sea mask, and resolution of the load model. For TPXO8-Atlas, the convolution may take a few minutes per station, primarily due to the interpolation of the high-resolution load model onto the integration mesh. The resolution of the integration mesh may be modified using optional input parameters (Sec. 3.4). Furthermore, the formats for the input land-sea mask and the input load model are designed to be straightforward. Output files are placed in the `output/Convolution/` directory.

4.7 Utility Functions

4.7.1 Station Grid

A utility script called `land_grid.py`, located within the `GRDGEN/station_grid/` directory, may be used to generate a grid of geographic point locations to be used with the convolution algorithm, `run_cn.py`. The user simply provides a desired grid spacing, bounds for the grid, and a land-sea mask (nominally used to return only those grid points that reside over designated land areas).

4.7.2 Plotting Results

Several simple plotting scripts are provided with `LoadDef` under `utility/plots/`. Following successful runs of `run_ln.py`, `run_pl.py`, and `run_gf.py`, most of the scripts should run straightaway. Figures will be plotted to screen and also saved as image files under the `utility/plots/output/` folder. If the folder does not exist, `LoadDef` will create one automatically.

The plotting scripts include the following:

1. **`plot_em.py`** : Plot parameters associated with Earth models
   :: Note: Multiple Earth models (and associated labels, colors, and weights) may be specified at the top of the script (in list format)
   :: Note: Panels F, G, and H, which show maximum differences *between* models, are only meaningful when at least two Earth models have been supplied as input (in the example, only PREM is shown)
2. **`plot_gf.py`** : Plot load Green's functions, which are generated by `run_gf.py`
3. **`plot_ld.py`** : Plot load models
   :: Note: The script is set up to plot the M_2 harmonic from the TPXO8-Atlas ocean-tide model (format: [lat, lon, amp, pha]), which may be generated using the `generate_otl_grid.py` utility in the `GRDGEN/load_files/` folder
   :: Note: Options within the script may be modified to plot values in the format [lat, lon, value] (e.g., the land-sea mask). To plot the land-sea mask that was provided in the `input/Load_Models/` directory, simply uncomment the fields marked with "Example 2" and comment *out* the fields marked with "Example 1". The land-sea mask is sparse, and only show values within a few grid-points of the coastlines. Land points are shown as magenta and ocean points are shown as cyan.
4. **`plot_ln.py`** : Plot load Love numbers, which are generated by `run_ln.py`
5. **`plot_pl.py`** : Plot Love-number partial derivatives, which are generated by `run_pl.py`
6. **`plot_ts.py`** : Plot time series of 3-component displacements, which are generated by `run_cn.py`

4.7.3 Custom Earth Models

A few simple scripts to generate and manipulate Earth models are provided in the `utility/emods/` folder. The subdirectory `PREM` includes scripts to generate an isotropic PREM model based on the polynomial functions provided in **?**. The `run_generate_PREM.py` script allows for user inputs, and calls the function `generate_PREM.py`.

The `utility/emods/` folder also contains the following scripts:

1. **`emod_dispersion.py`** : Adjusts the elastic parameters of a reference Earth model to non-seismic frequencies by accounting for physical dispersion
2. **`emod_homogeneous.py`** : Generates a homogeneous Earth model
3. **`emod_perturb.py`** : Perturbs the parameters (elastic moduli and density) of an Earth model

4.7.4 Conversion of Harmonic Displacements to Horizontal Particle Motion Ellipses

For the response to harmonic loads, such as surface displacements caused by ocean tidal loading, the horizontal components of the deformation may be converted to particle motion ellipses (PMEs) using utility scripts available in the `utility/pmes/` folder.

To facilitate the comparison of different models, stations should be grouped together into a single file for each tidal harmonic. The convolution program, however, generates a single file for each station, which contains all available harmonics. Thus, it may be necessary to sort through the station files, pick out the response to individual harmonics, and organize new response files by harmonic. The **`run_combine_harmonics.py`** script will do this for you, provided that the filenames for each station are in a certain format.

The "user inputs" section at the top of the script allows the user to specify the directory, prefix, and suffix corresponding to station files of interest. Only the name of the station should appear between the prefix and the suffix. The station name will be extracted within the code, as the string that exists between the prefix and the suffix. The model and harmonic should appear after the final underscore. The harmonic should be separated from the model name with a dash. The model name may optionally include dashes as well. For example, a filename might be `cn_OceanOnly_P402_cm_convgf_TPXO8-Atlas-M2.txt`, where `P402` is the station name, `TPXO8-Atlas` is the model name, and `M2` is the harmonic.

Whether the files are organized by station or harmonic, the **`run_env2pme.py`** script may be used to convert east and north displacements into particle motion ellipses. The input file for `run_env2pme.py` must be the same format as the files output by `run_cn.py`. The output file from `run_env2pme.py` will be in expanded "PME" format, which includes horizontal ellipse parameters.

Once files have been generated in "PME" format, the **`run_compute_residuals.py`** script may be used to compute residuals between the results contained within two PME-formatted files. Comparisons should only be made between ellipse parameters of the same frequency. In other words, each file should contain response estimates for a single harmonic. Optionally, the file may contain multiple stations, but the stations should be consistent in order and number between the two files being compared. In addition to computing the residuals directly between the two models, `run_compute_residuals.py` also generates a second set of residuals, whereby an amplitude and phase response common to the entire network is computed and removed. For more information on the "common-mode" response, see **?**.

5 Deformation Modeling

5.1 Introduction

For a more comprehensive discussion of the theory associated with computing Love numbers, load Green's functions, and predicted load-induced surface displacements, the reader is referred to **?**. Here, we review some of the key concepts.

Predicting the displacement-response of a radially symmetric Earth to surface mass loading (SML) involves a convolution of displacement load Green's functions (LGFs) with a load model (e.g., **??**):

$$\overline{U}(\overline{r}, S, Z, \rho) = \int_{\Omega'} \rho(\overline{r}')\, G(|\overline{r}' - \overline{r}|, S)\, Z(\overline{r}')\, d\Omega', \tag{5.1}$$

where $\overline{U}$ is the response of the solid Earth at position vector $\overline{r}$ (the measurement location), ρ is the mass density of the load at position vector $\overline{r}'$, G is the Green's function per kg load, and Z is the height of the load at position vector $\overline{r}'$. The integral is taken over the surface area of the Earth, Ω'. The LGFs depend on the distance between the load vector and the observation vector, $|\overline{r}' - \overline{r}|$, where $\overline{r}$ represents the position vector of the measurement location and $\overline{r}'$ represents the position vector of the load (e.g., **?**, Sec. 2.3). The LGFs additionally depend on Earth structure, S, where S represents radially symmetric structure (e.g., PREM). The LGFs are derived from combinations of load Love numbers (LLNs) (e.g., **?**).

In `LoadDef`, self-gravitating, spherically symmetric, non-rotating, elastic, and isotropic (SNREI) Earth models are assumed (e.g., **?**). In future releases, we plan to include the facility to adopt more sophisticated models, such as allowing for mantle anelasticity (e.g., **??**).

5.2 Love Number Computation

5.2.1 Introduction

Love numbers are dimensionless parameters that characterize the yielding of the elastic Earth to body forces and surface tractions (**??**). For the response of an elastic Earth to an external gravitational potential, V, three Love numbers are defined: $h_n(r)$, $k_n(r)$, and $l_n(r)$. The subscript n denotes a dependence on spherical harmonic degree and (r) indicates a radial dependence. Augustus Edward Hough (A.E.H.) Love introduced

$h_n(r)$ and $k_n(r)$ in 1909 (**?**); Toshi Shida subsequently introduced $l_n(r)$ in 1912. All three parameters are commonly referred to as *Love* numbers, or alternatively referred to as the *Love and Shida* numbers. Although the parameters exhibit a radial dependence, here we consider deformation observed only at Earth's surface, and thus drop the (r) notation.

The parameter h_n characterizes Earth's vertical displacement in response to an external potential. The radial displacement, u_n, of Earth's surface in response to an external gravitational potential of spherical harmonic degree, n, is given by (e.g., **?????**):

$$u_n = h_n \frac{V_n}{g}, \tag{5.2}$$

where g is the gravitational acceleration at Earth's surface, $\frac{V_n}{g}$ represents the equilibrium tidal height, and h_n scales the equilibrium height to a realistic vertical displacement based on the density and elastic properties of Earth's interior. Gravitational self-attraction, generated by the redistributed mass, is accounted for in the response parameter h_n (**?**).

The parameter k_n characterizes the change in the gravitational potential resulting from the redistribution of mass that occurs in response to the external potential field. Due to the self-attraction effect, the magnitude of the gravitational potential increases by a factor $k_n V_n$ from the location of the displaced surface. Thus, the total potential, accounting for self-attraction, becomes (e.g., **??**):

$$V_n^{\text{total}} = (1 + k_n) V_n. \tag{5.3}$$

The Shida number l_n is defined as the horizontal displacement of Earth relative to the gradient of the equilibrium tide. The two components of the horizontal displacement are:

$$\begin{aligned} v_n^{\text{north}} &= -\frac{l_n}{g} \frac{\partial V_n}{\partial \delta} \\ v_n^{\text{east}} &= \frac{l_n}{g} \frac{1}{\sin\delta} \frac{\partial V_n}{\partial \lambda}, \end{aligned} \tag{5.4}$$

where δ is colatitude and λ is east longitude (e.g., **??**).

The Love and Shida numbers described so far characterize Earth's response to an external gravitational potential; thus, we refer to them as **potential Love numbers**. A second class of Love numbers, referred to as **load Love numbers**, may also be introduced to describe the deformation of Earth under normal tractions, typically applied at Earth's surface (**??**). Surface mass loads come from a variety of sources, including glaciers, lakes, the atmosphere, and oceans. The load Love numbers are distinguished from the potential Love numbers by a superscript prime: h'_n, k'_n, and l'_n. A third class of Love numbers exists to characterize Earth's response to surface-tangential tractions (**?**), distinguished from the other classes by superscript double primes (h''_n, k''_n, l''_n) and known as **shear Love numbers**. Only six of the nine Love numbers from the three sets (potential, load, and shear) are independent, and expressions exist to relate the Love numbers to one another (**???**).

5.2.2 Equilibrium Equations for Material Deformation

The equilibrium equations describing spheroidal deformations of an elastic, self-gravitating, and hydrostatically pre-stressed body are derived from the principle of conservation of linear momentum (e.g., **??????????**). In continuum mechanics, conservation of linear momentum states:

$$\rho\,\overline{a} = \overline{\nabla}\cdot\overline{\overline{\sigma}} + \rho\,\overline{F}, \tag{5.5}$$

where $\overline{\overline{\sigma}}$ is the stress tensor, ρ is the density, $\overline{a}$ is the acceleration vector ($\overline{a}_i = \left(\frac{\partial^2\overline{u}}{\partial t^2}\right)_i = (\overline{u}_{,tt})_i = u_{i,tt}$, where $\overline{u}$ is a three-component displacement vector) and $\overline{F}$ is the body-force vector per unit mass. In indicial notation, $(\overline{\nabla}\cdot\overline{\overline{\sigma}})_i = \sigma_{ij,j}$, where Einstein's summation convention for repeated indices applies. The system of Eqs. 5.5 are also known as the *equations of motion* or the *momentum equations*. Note that we retain the acceleration term.

For a self-gravitating and hydrostatically pre-stressed body subjected to a small elastic displacement $\overline{u}$, the momentum equations are given by:

$$\begin{aligned}\rho\overline{a} = \rho_0\,\overline{u}_{,tt} \quad &= \quad \overline{\nabla}\cdot\overline{\overline{\sigma}}_1 + \rho_0\,\overline{\nabla}(\overline{u}\cdot\overline{g}_0) + (\overline{u}\cdot\overline{g}_0)\,\overline{\nabla}\rho_0 + \rho_0\,\overline{g}_1 - \\ &\qquad (\rho_0\,\Delta + \overline{u}\cdot\overline{\nabla}\rho_0)\,\overline{g}_0,\end{aligned} \tag{5.6}$$

where $\overline{\overline{\sigma}}_1$ represents the perturbation to the stress field due to the deformation, ρ_0 represents the equilibrium density of the unperturbed medium, $\overline{g}_0$ represents the equilibrium gravity of the unperturbed medium, and $\overline{g}_1$ represents the perturbation in gravity due to the perturbation in the gravitational potential (e.g., **??????**).

We define the displacement vector $\overline{u}$ as (e.g., **?**):

$$\begin{aligned} u_r &= u \\ u_\theta &= v \\ u_\phi &= w. \end{aligned} \tag{5.7}$$

Assuming that the vertical load is at the pole, it follows that the deformation is symmetric about $\theta = 0$ and the ϕ component of the equations of motion will be zero (e.g., **?**). Thus, we expand the momentum equations for displacements only in the directions of $\hat{r}$ ($u_r = u$) and $\hat{\theta}$ ($u_\theta = v$).

In spherical coordinates, the momentum equations become:

$$\begin{aligned} \rho_0 \frac{\partial^2 u}{\partial t^2} &= \frac{\partial}{\partial r}\sigma_{rr} + \frac{1}{r}\frac{\partial}{\partial \theta}\sigma_{r\theta} + \frac{1}{r}\left[2\sigma_{rr} - \sigma_{\theta\theta} - \sigma_{\phi\phi} + \cot\theta\, \sigma_{r\theta}\right] - \\ &\quad \rho_0 \frac{\partial (g_0\, u)}{\partial r} + \rho_0 \frac{\partial \psi}{\partial r} + \rho_0\, g_0\, \Delta \\ \rho_0 \frac{\partial^2 v}{\partial t^2} &= \frac{\partial}{\partial r}\sigma_{r\theta} + \frac{1}{r}\frac{\partial}{\partial \theta}\sigma_{\theta\theta} + \frac{1}{r}\left[(\sigma_{\theta\theta} - \sigma_{\phi\phi})\cot\theta + 3\sigma_{r\theta}\right] + \\ &\quad \frac{\rho_0}{r}\frac{\partial \psi}{\partial \theta} - \rho_0\, g_0 \frac{1}{r}\frac{\partial u}{\partial \theta}, \end{aligned} \tag{5.8}$$

where positive $\hat{r}$ is directed outward normal to the surface, positive $\hat{\theta}$ is directed outward from the load point, and ψ represents the perturbation to the gravitational potential. We have assumed radial forcing, which implies only spheroidal deformation, though we have not yet made any assumptions about a constitutive law. The Eqs. 5.8 are general to spheroidal deformation and may be used for multiple applications, including Earth's free oscillations, Earth's response to external gravitational potentials, and Earth's response to surface mass loading. The distinction between the different applications enters only through the boundary conditions applied at the surface, as we will discuss later on.

5.2.3 Linear Elastic Constitutive Relation

In the small-strain approximation, the strain tensor may be related to displacements by:

$$\overline{\epsilon} = \frac{1}{2}\left((\nabla\,\overline{u})^T + (\nabla\,\overline{u})\right), \tag{5.9}$$

or in indicial notation by:

$$\epsilon_{ij} = \frac{1}{2}\,(u_{i,j} + u_{j,i}). \tag{5.10}$$

For linear elasticity, the constitutive relation is given by:

$$\sigma_{ij} = \mathbb{C}_{ijkl}\,\epsilon_{kl}, \tag{5.11}$$

where $\mathbb{C}_{ijkl}$ represents the elasticity modulus (or *stiffness tensor*), which is a tensor of fourth-order. Eq. 5.11 is also known as Hooke's Law. For isotropic materials, the elasticity modulus reduces to:

$$\mathbb{C}_{ijkl} = \lambda\,\delta_{ij}\,\delta_{kl} + \mu(\delta_{ik}\,\delta_{jl} + \delta_{il}\,\delta_{jk}), \tag{5.12}$$

where λ and μ are Lamé's constants, which are related to compressional (P) and shear (S) sound-wave velocities by:

$$V_P^2 = \frac{\lambda + 2\,\mu}{\rho} \tag{5.13}$$
$$V_S^2 = \frac{\mu}{\rho}. \tag{5.14}$$

Thus, for linear elastic and isotropic materials, the constitutive relation is given by (Eqs. 5.11 and 5.12):

$$\sigma_{ij} = \lambda\,\epsilon_{kk}\,\delta_{ij} + 2\,\mu\,\epsilon_{ij}. \tag{5.15}$$

In spherical coordinates, the components of the stress tensor, $\overline{\overline{\sigma}}_1$, arising from the material deformation (without the hydrostatic pre-stress terms) are given by (e.g., **??**):

$$\sigma_{rr} = \lambda\,\Delta + 2\,\mu\,\epsilon_{rr} \tag{5.16}$$
$$\sigma_{\theta\theta} = \lambda\,\Delta + 2\,\mu\,\epsilon_{\theta\theta} \tag{5.17}$$
$$\sigma_{\phi\phi} = \lambda\,\Delta + 2\,\mu\,\epsilon_{\phi\phi} \tag{5.18}$$
$$\sigma_{r\theta} = 2\mu\,\epsilon_{r\theta} \tag{5.19}$$
$$\sigma_{r\phi} = 2\mu\,\epsilon_{r\phi} \tag{5.20}$$
$$\sigma_{\theta\phi} = 2\mu\,\epsilon_{\theta\phi}, \tag{5.21}$$

where

$$\begin{aligned}
\epsilon_{rr} &= \frac{\partial u}{\partial r} && (5.22)\\
\epsilon_{\theta\theta} &= \left(\frac{1}{r}\frac{\partial v}{\partial \theta} + \frac{u}{r}\right) && (5.23)\\
\epsilon_{\phi\phi} &= \left(\frac{1}{r\sin\theta}\frac{\partial w}{\partial \phi} + \frac{v}{r}\cot\theta + \frac{u}{r}\right) && (5.24)\\
\epsilon_{r\theta} &= \left(\frac{1}{2}\frac{\partial v}{\partial r} - \frac{v}{2r} + \frac{1}{2r}\frac{\partial u}{\partial \theta}\right) && (5.25)\\
\epsilon_{r\phi} &= \left(\frac{1}{2}\frac{1}{r\sin\theta}\frac{\partial u}{\partial \phi} + \frac{1}{2}\frac{\partial w}{\partial r} - \frac{w}{2r}\right) && (5.26)\\
\epsilon_{\theta\phi} &= \left[\frac{1}{2r}\left(\frac{\partial w}{\partial \theta} - w\cot\theta\right) + \frac{1}{2r\sin\theta}\frac{\partial v}{\partial \phi}\right]. && (5.27)
\end{aligned}$$

Thus, both elastic and gravitational restoring forces contribute to the deformation response. Note that definitions of the shear strain can differ by a factor of two in the literature, which must then be reflected consistently in the equations of motion (e.g., **??**).

Inserting the constitutive relations into the equations of motion yields:

$$\begin{aligned}
\rho_0\,\frac{\partial^2 u}{\partial t^2} &= \frac{\partial}{\partial r}\left(\lambda\,\Delta + 2\,\mu\,\frac{\partial u}{\partial r}\right) + \frac{2\mu}{r}\frac{\partial\left(\frac{1}{2}\frac{\partial v}{\partial r} - \frac{v}{2r} + \frac{1}{2r}\frac{\partial u}{\partial \theta}\right)}{\partial \theta} + \\
&\quad \frac{2\mu}{r}\left[2\frac{\partial u}{\partial r} - \frac{1}{r}\frac{\partial v}{\partial \theta} - \frac{2u}{r} + \cot\theta\left(\frac{1}{2}\frac{\partial v}{\partial r} - \frac{3v}{2r} + \frac{1}{2r}\frac{\partial u}{\partial \theta}\right)\right] - \\
&\quad \rho_0\frac{\partial(g_0\,u)}{\partial r} + \rho_0\frac{\partial \psi}{\partial r} + \rho_0\,g_0\,\Delta \\
\rho_0\,\frac{\partial^2 v}{\partial t^2} &= 2\frac{\partial}{\partial r}\left(\mu\left(\frac{1}{2}\frac{\partial v}{\partial r} - \frac{v}{2r} + \frac{1}{2r}\frac{\partial u}{\partial \theta}\right)\right) + \frac{1}{r}\frac{\partial}{\partial \theta}\left(\lambda\,\Delta + 2\,\mu\left(\frac{1}{r}\frac{\partial v}{\partial \theta} + \frac{u}{r}\right)\right) + \\
&\quad \frac{2\mu}{r}\left[\left(\frac{1}{r}\frac{\partial v}{\partial \theta} - \frac{v}{r}\cot\theta\right)\cot\theta + 3\left(\frac{1}{2}\frac{\partial v}{\partial r} - \frac{v}{2r} + \frac{1}{2r}\frac{\partial u}{\partial \theta}\right)\right] + \\
&\quad \frac{\rho_0}{r}\frac{\partial \psi}{\partial \theta} - \rho_0\,g_0\,\frac{1}{r}\frac{\partial u}{\partial \theta}, \qquad (5.28)
\end{aligned}$$

where we have allowed for radial heterogeneity in the density and elastic moduli.

5.2.4 Solutions to the Equations of Motion

The spheroidal-deformation Eqs. 5.28 for a self-gravitating, radially heterogeneous Earth may be satisfied by solutions of the form (e.g., **??**):

$$\begin{aligned} u_n &= U_n(r)\, P_n(\cos\theta)\, e^{i\omega t} \\ v_n &= V_n(r)\, \frac{\partial P_n(\cos\theta)}{\partial\theta}\, e^{i\omega t} \\ \psi_n &= \Psi_n(r)\, P_n(\cos\theta)\, e^{i\omega t}, \end{aligned} \tag{5.29}$$

where $U(r)$, $V(r)$, and $\Psi(r)$ are radial coefficients of the spherical harmonic expansions, $P_n(\cos\theta)$ is a Legendre polynomial of order n, u represents the radial displacement (Eq. 5.7), v represents the horizontal displacement (Eq. 5.7), and ψ represents the perturbation to the gravitational potential field caused by the deformation (e.g. **????**). For periodic forcing, such as loading by the ocean tides, ω represents the frequency of the forcing.

Since tides exhibit periods much longer than the free oscillations of the Earth ($\sim$1 hour), a quasi-static formulation of the equations of motion is sometimes adopted (e.g., **????**). For static or quasi-static formulations, $\omega = 0$, and temporal variations in the loading are removed from the problem. For the static case, additional considerations must be made for fluid regions (e.g. **??**). Since the divergence of the stress tensor must balance the sum of all forces, the static equations are referred to as the equilibrium equations (e.g., **?**). To maintain generality and consistency through fluid regions, we retain the time-varying factors in the development of the deformation solution (e.g., **???**).

The dilatation, Δ, may be written as:

$$\begin{aligned} \Delta &= \epsilon_{rr} + \epsilon_{\theta\theta} + \epsilon_{\phi\phi} \\ &= X(r)\, P_n(\cos\theta)\, e^{i\omega t}, \end{aligned} \tag{5.30}$$

where

$$X(r) = \dot{U} + \frac{2U}{r} - \frac{n(n+1)}{r} V, \tag{5.31}$$

and we have made use of the Legendre differential equation (e.g., **?**). A dot implies differentiation with respect to r.

From the set of Eqs. 5.29:

$$\begin{aligned}
\frac{\partial u}{\partial \theta} &= -U \sin\theta P'_n\, e^{i\omega t} \\
\frac{\partial v}{\partial \theta} &= V \sin^2\theta P''_n\, e^{i\omega t} - V\cos\theta P'_n\, e^{i\omega t} \\
\frac{\partial u}{\partial r} &= \dot{U} P_n\, e^{i\omega t} \\
\frac{\partial v}{\partial r} &= -\dot{V}\sin\theta P'_n\, e^{i\omega t} \\
\frac{\partial^2 u}{\partial \theta^2} &= U \sin^2\theta P''_n\, e^{i\omega t} - U\cos\theta P'_n\, e^{i\omega t} \\
\frac{\partial^2 v}{\partial \theta^2} &= \frac{\partial}{\partial\theta}[V\sin^2\theta P''_n - V\cos\theta P'_n]\, e^{i\omega t} \\
&= V[-\sin^3\theta P'''_n + 3P''_n \sin\theta\cos\theta + \sin\theta P'_n]\, e^{i\omega t}, \qquad (5.32)
\end{aligned}$$

where

$$P'_n = \frac{\partial P_n(\cos\theta)}{\partial\cos\theta}, P''_n = \frac{\partial^2 P_n(\cos\theta)}{\partial\cos^2\theta}, P'''_n = \frac{\partial^3 P_n(\cos\theta)}{\partial\cos^3\theta}, \qquad (5.33)$$

and, according to the Legendre differential equation (**?**),

$$\sin^2\theta P''_n - 2\cos\theta P'_n = -n(n+1)P_n. \qquad (5.34)$$

Only P_n and P'_n are required to derive additional derivatives of the Legendre polynomials recursively. For example, the third derivative of the Legendre polynomial, P'''_n, is given by:

$$\begin{aligned}
0 &= \frac{\partial}{\partial\cos\theta}[(1-\cos^2\theta)P''_n - 2\cos\theta P'_n + n(n+1)P_n] \\
&= P'''_n - 2\cos\theta P''_n - \cos^2\theta P'''_n - 2P'_n - 2\cos\theta P''_n + n(n+1)P'_n \\
&= (1-\cos^2\theta)P'''_n - 4\cos\theta P''_n + (-2+n(n+1))P'_n \\
&= \sin^2\theta P'''_n - 4\cos\theta P''_n - (2-n(n+1))P'_n \\
\sin^2\theta P'''_n &= 4\cos\theta P''_n + 2P'_n - n(n+1)P'_n. \qquad (5.35)
\end{aligned}$$

Rewriting the radial component of Eq. 5.28, we have:

$$\begin{aligned}
\rho_0 \frac{\partial^2 u}{\partial t^2} &= \left\{ \frac{\partial}{\partial r} \left(\lambda\, X + 2\, \mu\, \dot{U} \right) + \dot{V}\, \frac{\mu}{r}(-n(n+1)) + U\, \frac{\mu}{r^2}(-n(n+1)) + \right. \\
&\left. \frac{\mu}{r^2}[4\, r\dot{U} - 3V(-n(n+1)) - 4U] - \rho_0 \frac{\partial (g_0\, U)}{\partial r} + \rho_0 \dot{\Psi} + \rho_0\, g_0\, X \right\} P_n(\cos\theta)\, e^{i\omega t} \\
-\omega^2 \rho_0 U &= \frac{\partial}{\partial r} \left(\lambda\, X + 2\, \mu\, \dot{U} \right) + \frac{\mu}{r^2}[4\, r\dot{U} - 4U + n(n+1)(3V - U - r\dot{V})] - \\
&\rho_0 \frac{\partial (g_0\, U)}{\partial r} + \rho_0 \dot{\Psi} + \rho_0\, g_0\, X, \qquad (5.36)
\end{aligned}$$

where common terms in $e^{i\omega t}$ and P_n have been canceled in the second line (**?**).

Similarly, the surface-tangential component of Eq. 5.28 may be written as:

$$\begin{aligned}
\rho_0 \frac{\partial^2 v}{\partial t^2} &= \left\{ -\sin\theta P_n' \left(\frac{\partial}{\partial r} \left(\mu\dot{V} - \mu\frac{V}{r} + \mu\frac{U}{r} \right) + \frac{\lambda}{r} X \right) + \right. \\
&\frac{\rho_0}{r} \Psi P_n'(-\sin\theta) - \frac{\rho_0 g_0}{r} U P_n'(-\sin\theta) + \\
&\frac{\mu}{r^2} \left[2V(-\sin\theta P_n' + n(n+1)\sin\theta P_n') + \right. \\
&\left. \left. 5UP_n'(-\sin\theta) - 3VP_n'(-\sin\theta) - 3r\dot{V}\sin\theta P_n' \right] \right\} e^{i\omega t} \\
-\omega^2 \rho_0 V &= \frac{\partial}{\partial r} \left(\mu\dot{V} - \mu\frac{V}{r} + \mu\frac{U}{r} \right) + \frac{\lambda}{r} X + \frac{\rho_0}{r} \Psi - \frac{\rho_0 g_0}{r} U + \\
&\frac{\mu}{r^2} \left[5U - V - 2n(n+1)V + 3r\dot{V} \right], \qquad (5.37)
\end{aligned}$$

where common terms in $e^{i\omega t}$ and $[-\sin\theta P_n']$ have been canceled in the second line (**?**).

Finally, Poisson's equation may be written as (**?**):

$$\begin{aligned}
\nabla^2 \psi &= 4\,\pi\, G \left(\rho_0\, \Delta + u\, \frac{\partial \rho_0}{\partial r} \right) \\
&= \{ 4\,\pi\, G\ (\rho_0\, X + U\, \dot{\rho}_0) \}\ P_n(\cos\theta)\, e^{i\omega t}. \qquad (5.38)
\end{aligned}$$

We can also write the Laplacian in spherical coordinates as (**?**):

$$\nabla^2 \psi = \left\{ \frac{2}{r} \dot{\Psi} + \ddot{\Psi} - \frac{n(n+1)}{r^2} \Psi \right\} P_n(\cos\theta)\, e^{i\omega t}, \qquad (5.39)$$

where

$$\begin{aligned}
\frac{\partial\psi}{\partial\theta} &= -\sin\theta\,\Psi\,P_n'\,e^{i\omega t} \\
\frac{\partial^2\psi}{\partial\theta^2} &= \Psi\,[\sin^2\theta P_n'' - \cos\theta P_n']\,e^{i\omega t}.
\end{aligned}$$

Combining Eq. 5.38 with Eq. 5.39 yields (**?**):

$$\ddot{\Psi} + \frac{2}{r}\dot{\Psi} - \frac{n(n+1)}{r^2}\Psi = 4\,\pi\,G\,\left(\rho_0\,X + U\,\dot{\rho}_0\right), \tag{5.40}$$

where we have canceled common factors of $e^{i\omega t}$ and P_n.

The equations of motion for spheroidal deformation are now given by three second-order differential equations (Eqs. 5.36, 5.37, and 5.40) that may be solved for the coefficients U, V, and Ψ.

5.2.5 Reduction of the Equations of Motion to First Order

For convenience, a set of handy substitutions may be made to simplify Eqs. 5.36, 5.37, and 5.40 prior to numerical integration (e.g., **???**, Ch. 5):

$$\begin{aligned}
y_1 &= U \\
y_2 &= \lambda X + 2\mu\dot{U} \\
y_3 &= V \\
y_4 &= \mu\left(\dot{V} - \frac{V}{r} + \frac{U}{r}\right) \\
y_5 &= \Psi \\
y_6 &= \dot{\Psi} - 4\pi G\rho_0 U.
\end{aligned} \tag{5.41}$$

With the substitutions, Eqs. 5.36, 5.37, and 5.40 become a system of six first-order differential equations (e.g. **?**):

$$\begin{aligned}
\dot{y}_1 &= \frac{-2\lambda}{\lambda+2\mu}\frac{y_1}{r} + \frac{y_2}{\lambda+2\mu} + \frac{\lambda\,n(n+1)}{\lambda+2\mu}\frac{y_3}{r}, \\
\dot{y}_2 &= \left[-\omega^2\rho_0 r^2 - 4\rho_0 g_0 r + \frac{4\mu(3\lambda+2\mu)}{\lambda+2\mu}\right]\frac{y_1}{r^2} - \frac{4\mu}{\lambda+2\mu}\frac{y_2}{r}
\end{aligned}$$

$$
\begin{aligned}
&+ \quad \left[n(n+1)\rho_0 g_0 r - \frac{2\mu(3\lambda+2\mu)\,n(n+1)}{\lambda+2\mu}\right]\frac{y_3}{r^2} \\
&+ \quad n(n+1)\frac{y_4}{r} - \rho_0 y_6, \\
\dot{y}_3 &= -\frac{y_1}{r} + \frac{y_3}{r} + \frac{y_4}{\mu}, \\
\dot{y}_4 &= \left[g_0\rho_0 r - \frac{2\mu(3\lambda+2\mu)}{\lambda+2\mu}\right]\frac{y_1}{r^2} - \frac{\lambda}{\lambda+2\mu}\frac{y_2}{r} \\
&+ \quad \left[-\omega^2\rho_0 r^2 + \frac{2\mu}{\lambda+2\mu}[\lambda(2n^2+2n-1)+2\mu(n^2+n-1)]\right]\frac{y_3}{r^2} \\
&- \quad \frac{3y_4}{r} - \rho_0\frac{y_5}{r}, \\
\dot{y}_5 &= 4\pi G\rho_0 y_1 + y_6, \\
\dot{y}_6 &= -4\pi G\rho_0\, n(n+1)\frac{y_3}{r} + n(n+1)\frac{y_5}{r^2} - \frac{2y_6}{r}, \qquad (5.42)
\end{aligned}
$$

where dots represent differentiation with respect to r. Importantly, by reducing the equations of motion to first order, the derivatives of the elastic parameters no longer appear. The variables y_1 and y_3 characterize the radial and tangential displacements, respectively; y_2 and y_4 characterize the radial and tangential stress, respectively; y_5 characterizes the gravitational potential; and the equation for $\dot{y}_5$ defines y_6. Additional information about the equations of motion for spheroidal oscillations may be found in the literature (e.g., **????**).

Note that the six linear equations may now be written in the form $\dot{y} = A\, y$:

$$
\begin{bmatrix} \frac{dy_1}{dr} \\ \frac{dy_2}{dr} \\ \frac{dy_3}{dr} \\ \frac{dy_4}{dr} \\ \frac{dy_5}{dr} \\ \frac{dy_6}{dr} \end{bmatrix}
=
\begin{bmatrix}
-\frac{2\lambda\xi}{r} & \xi & \frac{k^2\lambda\xi}{r} & 0 & 0 & 0 \\
\left(-\omega^2\rho_0 - \frac{4g_0\rho_0}{r} + \frac{2\delta}{r^2}\right) & -\frac{4\mu\xi}{r} & \left(\frac{k^2 g_0\rho_0}{r} - \frac{k^2\delta}{r^2}\right) & \frac{k^2}{r} & 0 & -\rho_0 \\
-\frac{1}{r} & 0 & \frac{1}{r} & \frac{1}{\mu} & 0 & 0 \\
\left(\frac{g_0\rho_0}{r} - \frac{\delta}{r^2}\right) & -\frac{\lambda\xi}{r} & \left(-\omega^2\rho_0 + \frac{\epsilon}{r^2}\right) & -\frac{3}{r} & -\frac{\rho_0}{r} & 0 \\
4\pi G\rho_0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & -4\pi G\rho_0\frac{k^2}{r} & 0 & \frac{k^2}{r^2} & -\frac{2}{r}
\end{bmatrix}
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix}, \qquad (5.43)
$$

where

$$
k^2 = n(n+1) \qquad (5.44)
$$

$$
\xi = \frac{1}{\lambda+2\mu} \qquad (5.45)
$$

$$\delta = 2\mu(3\lambda + 2\mu)\xi \tag{5.46}$$

$$\epsilon = 4k^2\mu(\lambda + \mu)\xi - 2\mu. \tag{5.47}$$

The matrix equation, $\dot{y} = A\ y$, may be solved using numerical methods, such as the commonly adopted Runge-Kutta algorithm and the propagator matrix technique (Sec. 5.2.9) (**?**) .

For the special case of $n = 0$, the equations for y_3 and y_4 are undefined and the system reduces to (e.g., **?**):

$$\begin{bmatrix} \frac{dy_1}{dr} \\ \frac{dy_2}{dr} \\ \frac{dy_5}{dr} \\ \frac{dy_6}{dr} \end{bmatrix} = \begin{bmatrix} -\frac{2\lambda\xi}{r} & \xi & 0 & 0 \\ \left(-\omega^2\rho_0 - \frac{4g_0\rho_0}{r} + \frac{2\delta}{r^2}\right) & -\frac{4\mu\xi}{r} & 0 & -\rho_0 \\ 4\pi G\rho_0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{2}{r} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_5 \\ y_6 \end{bmatrix}. \tag{5.48}$$

5.2.6 Non-dimensionalization

Suitable scaling parameters must be introduced to maintain numerical stability during the integration of the equations of motion (e.g., **??**). We adopt a characteristic length scale, a, equal to Earth's mean radius (6371 km) and a characteristic mass scale close to Earth's mass, $4\pi a^3\bar{\rho}/3$, where $\bar{\rho}$ is an approximate value for Earth's average density (5500 kg/m^3). A handy way of scaling the time is to define the characteristic time scale as $1/\sqrt{\pi G\bar{\rho}}$, where G is the universal gravitational constant. Since G has units of $[M^{-1}L^3T^{-2}]$, it becomes

$$G' = G/(\bar{\rho}^{-1}a^{-3}a^3\pi G\bar{\rho}) = 1/\pi \tag{5.49}$$

in the scaled system. M, L, and T represent units of mass, length, and time, respectively.

In summary, we have

Length :

$$l' = l/a \tag{5.50}$$

Mass :

$$m' = m/(4\pi\bar{\rho}a^3/3) \tag{5.51}$$

Time :

$$t' = t\sqrt{\pi G \bar{\rho}} \tag{5.52}$$

Another way of stating the above mass scaling is simply to say that we adopt a characteristic density scale equal to $\bar{\rho}$, where

$$\rho' = \rho/\bar{\rho}. \tag{5.53}$$

5.2.7 Starting Solutions: Power Series Expansion

The six first-order ordinary differential equations (Eq. 5.42) require six sets of starting solutions to initialize the Runge-Kutta integration. Three of the functions, y_1, y_3, and y_5, diverge at $r = 0$; thus, the starting solutions for U $(= y_1)$, V $(= y_3)$, and Ψ $(= y_5)$ are necessarily set to zero to ensure regularity at Earth's center (e.g., **??**). The remaining three linearly independent solutions at Earth's center may be computed by a power series expansion near the geocenter (**??**). Following the development of **?** and **?**, the power series expansion of the y variables takes the form:

$$y_i(r) = r^{\alpha} \sum_{\nu=0}^{\infty} A_{i,\nu}\, r^{\nu}, \qquad i = 1, 2, \ldots, 6, \tag{5.54}$$

where α and ν are integers and the coefficients $A_{i,\nu}$ are to be determined. The variable α must be non-negative and the coefficients $A_{i,\nu} = 0$ for $\nu < 0$. The derivate of Eq. 5.54 is given by (**?**):

$$\frac{dy_i(r)}{dr} = r^{\alpha-1} \sum_{\nu=0}^{\infty} (\alpha + \nu)\, A_{i,\nu}\, r^{\nu}. \tag{5.55}$$

Additionally, gravity can be expanded as:

$$g_0(r) = \gamma r, \tag{5.56}$$

where terms beyond first-order have been dropped since the expansion will be implemented close to the geocenter, and

$$\gamma = \frac{4}{3}\pi G \rho_0. \tag{5.57}$$

Within an arbitrarily small radius, such as $r = 1$ km, the density and elastic moduli are assumed to be constant and equivalent to their values at the geocenter.

Eqs. 5.54, 5.55, and 5.56 are inserted back into the system of equations (Eq. 5.42). Matching powers of the radius are equated to obtain a set of characteristic, or *indicial*, equations. Eigenvalues of the characteristic equations yield the acceptable values for variable α, and the corresponding eigenvectors may then be used to compute the values of the coefficients $A_{i,\nu}$ in terms of three free constants: $A_{1,1}$, $A_{6,1}$, and $A_{4,0}$ (**?**). Choice of the free constants is arbitrary, since multiplying each of the y variables in a fundamental-solution set by the same constant factor does not affect the result; only ratios between the y variables are important. A convenient option is to set each of the three constants equal to one.

To facilitate numerical integration, we transform the variables (e.g., **??**):

$$\begin{aligned}
z_1(r) &= \frac{y_1(r)}{r^{\alpha+1}} \\
z_2(r) &= \frac{y_2(r)}{r^{\alpha}} \\
z_3(r) &= \frac{y_3(r)}{r^{\alpha+1}} \\
z_4(r) &= \frac{y_4(r)}{r^{\alpha}} \\
z_5(r) &= \frac{y_5(r)}{r^{\alpha+2}} \\
z_6(r) &= \frac{y_6(r)}{r^{\alpha+1}},
\end{aligned} \tag{5.58}$$

where $\alpha = n - 2$ for the starting solutions associated with the free constants $A_{1,1}$ and $A_{6,1}$ and $\alpha = n$ for the starting solution associated with the free constant $A_{4,0}$. Appropriate modifications must be made for the special cases of $n = 1$ and $n = 0$ (**?**). For $n = 0$, only two starting solutions are regular at the geocenter. More information about the starting solutions and how to compute them may be found in **?** and **?**.

5.2.8 Starting Solutions: Homogeneous Sphere

An alternative approach to the power series expansions is to compute the analytical solution for a homogeneous sphere (e.g., **????**). Here, we review the analytical solutions of the equations of motion for a homogeneous sphere. **?** provide a more comprehensive discussion.

Two of the independent solutions (note the $\mp$ in the equation for b^2 below) for spherical harmonic degree,

n, and forcing frequency, ω, are given by (e.g., **?**):

$$\begin{aligned}
y_1 &= -\frac{r^{n+1}}{2n+3}\left[\frac{1}{2}nh\chi_n(x) + f\phi_{n+1}(x)\right] \\
y_2 &= -(\lambda+2\mu)r^n f\phi_n(x) + \frac{\mu r^n}{2n+3}[-n(n+1)h\chi_n(x) + 2(2f+k^2)\phi_{n+1}(x)] \\
y_3 &= -\frac{r^{n+1}}{2n+3}\left[\frac{1}{2}h\chi_n(x) - \phi_{n+1}(x)\right] \\
y_4 &= \mu r^n\left[\phi_n(x) - \frac{1}{2n+3}[(n-1)h\chi_n(x) + 2(f+1)\phi_{n+1}(x)]\right] \\
y_5 &= r^{n+2}\left[\frac{V_P^2 f - (n+1)V_S^2}{r^2} - \frac{3\gamma f}{2(2n+3)}\chi_n(x)\right] \\
y_6 &= \frac{1}{r}\left[(2n+1)y_5 + \frac{3n\gamma h r^{n+2}}{2(2n+3)}\chi_n(x)\right], \qquad (5.59)
\end{aligned}$$

where V_P is the compressional wave velocity, V_S is the shear wave velocity, r is the radius, λ and μ are Lamé parameters, and

$$\begin{aligned}
\gamma &= \frac{4}{3}\pi\rho G \\
x &= b\,r \\
b^2 &= \frac{1}{2}\left[\frac{\omega^2+4\gamma}{V_P^2} + \frac{\omega^2}{V_S^2} \mp \left(\left(\frac{\omega^2}{V_S^2} - \frac{\omega^2+4\gamma}{V_P^2}\right)^2 + \frac{4k^2\gamma^2}{V_P^2V_S^2}\right)^{\frac{1}{2}}\right] \\
f &= \frac{V_S^2}{\gamma}\left(b^2 - \frac{\omega^2}{V_S^2}\right) \\
h &= f-(n+1). \qquad (5.60)
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\phi_n(x) &= \frac{(2n+1)!!}{x^n}j_n(x) \\
&= 1 - \frac{x^2}{2(2n+3)} + \frac{x^4}{2^2(2n+3)(2n+5)2} - \ldots \qquad (5.61) \\
\chi_n(x) &= \frac{2(2n+3)}{x^2}[1-\phi_n(x)] \\
&= 1 - \frac{x^2}{2(2n+5)2} + \frac{x^4}{2^2(2n+5)(2n+7)3} - \ldots \qquad (5.62)
\end{aligned}$$

where $j_n(x)$ represents the spherical Bessel function of the first kind (e.g., **??**). Since the Bessel functions tend toward zero at high n, we approximate them using the power series expansions given by Eqs. 5.61 and

5.62, carried out to several terms.

The third independent solution is given by (e.g., **?**):

$$\begin{aligned} y_1 &= nr^{n-1} \\ y_2 &= 2\mu n(n-1)r^{n-2} \\ y_3 &= r^{n-1} \\ y_4 &= 2\mu(n-1)r^{n-2} \\ y_5 &= (n\gamma - \omega^2)r^n \\ y_6 &= \frac{1}{r}(2n+1)y_5(r) - 3n\gamma r^{n-1}. \end{aligned} \tag{5.63}$$

5.2.9 Runge-Kutta Integration

We integrate the equations of motion from depth to Earth's surface using Runge-Kutta algorithms that are part of the `scipy` library. The starting radius for the integration can affect the numerical stability of the solution. `LoadDef` determines the starting radius for the integration based on the decaying influence of an external potential field with depth for any given harmonic (see Sec. 5.2.18). In practice, we have found that commencing the integration from Earth's center is appropriate for $n <$ about 20, after which numerical stability may be improved by starting the integration within the mantle. For integrations that start at Earth's center and move through the core, `LoadDef` uses starting solutions computed from power series expansions with transformed z variables (**?**). For integrations that start in the mantle, `LoadDef` uses starting solutions computed for a homogeneous sphere (**?**).

From a set of starting solutions $y_j(r_i)$ (or $z_j(r_i)$ within the inner core) at an initial radius r_i, the solution vector $y_j(r_{i+1})$ at the new radius $r_i + h$ is given by:

$$y_j(r_{i+1}) = y_j(r_i) + \frac{h}{6}(k_{1j} + 2k_{2j} + 2k_{3j} + k_{4j}), \tag{5.64}$$

where h represents a suitably small step in radius, $j = 1 \cdots 6$ for each of the six solutions to the system of six linear equations for spheroidal deformation (Eqs. 5.41), and

$$k_{1j} = A_{jl}(r_i)\, y_l(r_i) \tag{5.65}$$

$$\begin{aligned}
k_{2j} &= A_{jl}\left(r_i + \frac{h}{2}\right)\left[y_l(r_i) + \frac{k_{1l}}{2}\right] && (5.66)\\
k_{3j} &= A_{jl}\left(r_i + \frac{h}{2}\right)\left[y_l(r_i) + \frac{k_{2l}}{2}\right] && (5.67)\\
k_{4j} &= A_{jl}(r_i + h)\,[y_l(r_i) + k_{3l}]. && (5.68)
\end{aligned}$$

A_{jl} is given by Eq. 5.43. The fourth-order Runge-Kutta formulation in Eq. 5.64 is derived from a Taylor series expansion of $\dot{y}$. The system of equations, $\dot{y}$, may be computed at each radial step using the propagator matrix technique (i.e., $\dot{y} = A\ y$, as in Eqs. 5.65–5.68) (**?**).

5.2.10 Fluid Layers

In fluid regions, the shear modulus vanishes ($\mu = 0$). Hence $y_4 = 0$, $y_2 = \lambda X$, and $\dot{y}_3$ is undefined (**?**). Therefore, the system of equations for spheroidal deformation reduces to (e.g., **?**):

$$\begin{bmatrix} \frac{dy_1}{dr} \\ \frac{dy_2}{dr} \\ 0 \\ \frac{dy_5}{dr} \\ \frac{dy_6}{dr} \end{bmatrix} = \begin{bmatrix} -\frac{2}{r} & \frac{1}{\lambda} & \frac{k^2}{r} & 0 & 0 \\ \left(-\omega^2\rho_0 - \frac{4g_0\rho_0}{r}\right) & 0 & \frac{k^2 g_0\rho_0}{r} & 0 & -\rho_0 \\ \frac{g_0\rho_0}{r} & -\frac{1}{r} & -\omega^2\rho_0 & -\frac{\rho_0}{r} & 0 \\ 4\pi G\rho_0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -4\pi G\rho_0\frac{k^2}{r} & \frac{k^2}{r^2} & -\frac{2}{r} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_5 \\ y_6 \end{bmatrix}.$$

The third equation in the system can be solved for y_3 as follows (**?**):

$$\begin{aligned}
0 &= \frac{\rho_0 g_0}{r}y_1 - \frac{1}{r}y_2 - \omega^2\rho_0 y_3 - \frac{\rho_0}{r}y_5\\
\omega^2\rho_0 y_3 &= \frac{\rho_0 g_0}{r}y_1 - \frac{1}{r}y_2 - \frac{\rho_0}{r}y_5\\
y_3 &= \frac{1}{\omega^2\rho_0}\left(\frac{\rho_0 g_0}{r}y_1 - \frac{1}{r}y_2 - \frac{\rho_0}{r}y_5\right)\\
y_3 &= \frac{1}{\omega^2 r}\left(g_0 y_1 - \frac{1}{\rho_0}y_2 - y_5\right). && (5.69)
\end{aligned}$$

Note that, with ω in the denominator, the zero-frequency case is problematic for fluid layers (e.g. **?**).

The equation for y_3 is now written in terms of y_1, y_2, and y_5, so it may be substituted back into the matrix system, which reduces to:

$$\begin{bmatrix} \frac{dy_1}{dr} \\ \frac{dy_2}{dr} \\ \frac{dy_5}{dr} \\ \frac{dy_6}{dr} \end{bmatrix} = \begin{bmatrix} -\frac{2}{r} + \frac{k^2}{r} y_3^1 & \frac{1}{\lambda} + \frac{k^2}{r} y_3^2 & \frac{k^2}{r} y_3^5 & 0 \\ \left(-\omega^2 \rho_0 - \frac{4 g_0 \rho_0}{r} + \frac{k^2 g_0 \rho_0}{r} y_3^1\right) & \frac{k^2 g_0 \rho_0}{r} y_3^2 & \frac{k^2 g_0 \rho_0}{r} y_3^5 & -\rho_0 \\ 4\pi G \rho_0 & 0 & 0 & 1 \\ -4\pi G \rho_0 \frac{k^2}{r} y_3^1 & -4\pi G \rho_0 \frac{k^2}{r} y_3^2 & \left(\frac{k^2}{r^2} - 4\pi G \rho_0 \frac{k^2}{r} y_3^5\right) & -\frac{2}{r} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_5 \\ y_6 \end{bmatrix},$$

where

$$\begin{aligned} y_3^1 &= \frac{g_0}{\omega^2 r} \\ y_3^2 &= -\frac{1}{\rho_0 \omega^2 r} \\ y_3^5 &= -\frac{1}{\omega^2 r}. \end{aligned} \tag{5.70}$$

5.2.11 Boundary Conditions at Solid-Fluid Interfaces

All independent variables $y_1, y_2, \ldots y_6$ are continuous at solid-solid internal boundaries (e.g., **???**). At solid-fluid boundaries, however, the shear stress vanishes; thus, all y variables are continuous with the exception of y_3, which is undefined across the interface. The system of six first-order ODEs within the solid layer reduces to a system of four first-order ODEs within the fluid layer. Thus, whereas three independent solutions are required for solid layers, only two independent solutions are required for fluid layers. See **?** and **?** for more information on the boundary conditions at solid-fluid interfaces.

5.2.12 Surface Boundary Conditions

The surface boundary conditions differ depending on the type of response being investigated, such as Earth's free oscillations, Earth's response to an external gravitational potential, or Earth's response to surface mass loading. For a comprehensive discussion and derivation of the surface boundary conditions for several different response cases, the reader is referred to **?**. Here, we provide an abbreviated summary.

For free oscillations of the Earth, the absence of an applied load requires that the radial and tangential stresses vanish at the surface (e.g., **???**). For an external gravitational potential, caused by a mass m external to the Earth, the additional potential, ψ_E, must be added to the perturbed potential field, ψ, in the equations of

Surface Boundary Conditions ($n \neq 1$)				
	(a) Free Oscillations	(b) External Potential	(c) Surface Mass Loading	(d) Surface Shear Forcing
y_2	0	0	$-g_S^2\,\frac{2n+1}{4\pi G}$	0
y_4	0	0	0	$\frac{(2n+1)\,g_S^2}{4\pi G\,n\,(n+1)}$
$y_6 + \frac{n+1}{a}\,y_5$	0	$(2n+1)\,g_S$	$(2n+1)\,g_S$	0

Table 5.1: Summary of surface boundary conditions for the cases of (a) free oscillations, (b) the presence of an external gravitational potential field, (c) surface mass loading, and (d) surface shear forcing (e.g., **????????**). Note that the boundary conditions stated in **?** are presented in terms of normalized y-variables, whereas here we state the boundary conditions directly in terms of the y-variables. Also note that the Love number definitions of **?** differ from the definitions stated here by a factor of $(a\,g_S)$, where a is Earth's radius and g_S is the acceleration due to gravity. Different scalings for the Love numbers must be reflected in the boundary conditions.

Surface Boundary Conditions ($n = 1$, **CE frame)**					
	(a) Free Oscillations	(b) External Potential	(c) Surface Mass Loading	(d) Surface Shear Forcing	(e) Surface Stress
y_2	0	0	$-\frac{3\,g_S^2}{4\pi G}$	0	$-\frac{3\,g_S^2}{4\pi G}$
y_4	0	0	0	$\frac{3\,g_S^2}{8\pi G}$	$\frac{3\,g_S^2}{8\pi G}$
y_5	0	$a\,g_S$	$a\,g_S$	0	0

Table 5.2: Summary of surface boundary conditions for the cases of (a) free oscillations, (b) the presence of an external gravitational potential field, (c) surface mass loading, (d) surface shear forcing, and (e) surface stress (e.g., **???????**). The surface stress solution satisfies the consistency relation and provides a linearly independent secondary solution for static degree-1 modes (**?**).

motion (e.g., **?**).

Since mass m exists entirely outside Earth, however, the external potential field satisfies Laplace's equation everywhere inside the Earth and thus is only implicit in the equations of motion. The surface boundary conditions, however, contain ψ_E explicitly. At the surface, both ψ and ψ_E must be continuous. Furthermore, $\dot{\psi}_1 - 4\pi G\rho_0 u$ must be continuous, as stated previously, and additionally $\dot{\psi}_E + 4\pi G\gamma$ must also be continuous, where γ represents a unit of external mass distributed uniformly over a disk of radius α (e.g., **??**). Following the method of **?**, a unit of external mass γ distributed uniformly over a disk of radius α may be expanded as a harmonic Legendre series of the form:

$$\gamma = \sum_{n=0}^{\infty} K_n P_n(\cos\theta)\, e^{i\omega t}. \tag{5.71}$$

The coefficients K_n are then given by (e.g., **???**):

$$\begin{aligned} K_0 &= \frac{1}{4\pi a^2} \\ K_n &= \frac{P_{n-1}(\cos\alpha) - P_{n+1}(\cos\alpha)}{4\pi a^2(1-\cos\alpha)}, \qquad n > 0 \end{aligned} \tag{5.72}$$

where a is Earth's radius. In the limit $\alpha \to 0$,

$$K_n = \frac{2n+1}{4\pi a^2}\ . \tag{5.73}$$

From a Legendre recursion relation,

$$P_{n-1}(x) - P_{n+1}(x) = \frac{2n+1}{n(n+1)}(1-x^2)P_n'(x)\ ; \tag{5.74}$$

therefore (**?**),

$$K_n = \frac{2n+1}{4\pi a^2}\left[-\frac{(1+\cos\alpha)}{n(n+1)\sin\alpha}\frac{\partial P_n(\cos\alpha)}{\partial\alpha}\right]\ . \tag{5.75}$$

The quantity in brackets is known as the **disk factor** and represents a mass distribution of finite size. The quantity in front of the bracketed terms represents the Legendre expansion of the delta function in spherical coordinates (e.g., **??**).

For Earth's response to a gravitational potential field generated by an external mass *not* loading the Earth, the radial and tangential tractions vanish at the free surface (e.g., **?**). Thus, $y_2 = y_4 = 0$, as for the free oscillations (e.g., **??**). For the case of surface mass loading, however, the radial traction will be non-zero (e.g., **????**). The radial traction is given by the acceleration of gravity multiplied by the surface mass distribution (e.g., **?**, Sec. 3.06). Thus,

$$\begin{aligned} y_2 &= -g_S\, K_n \\ &= -g_S^2\, \frac{2n+1}{4\pi G}. \end{aligned} \tag{5.76}$$

The surface boundary conditions for free oscillations, external potential fields, and surface mass loading (SML) are summarized in Table 5.1. Solving the spheroidal-deformation equations using external-potential boundary conditions yields potential Love numbers. Solving the equations using SML boundary conditions yields load Love numbers. Another set of boundary conditions may be developed to represent surface shear forcing (also listed in Table 5.1). Solutions generated from applying shear-forcing boundary conditions to the equations of motion yield shear Love numbers. Only six of the nine Love numbers (potential, load, and shear) are independent and there exist three independent linear relationships among them (e.g., **??**).

For the special case of $n = 0$, the equations for y_3 and y_4 (the tangential components) are undefined and the system reduces to four equations (Eq. 5.48) (e.g., **??**). Furthermore, only two solutions and two boundary conditions exist for $n = 0$. The boundary conditions are identical to those listed in Table 5.1, with the exception that the conditions for y_4 must be excluded.

For the special case of $n = 1$, the gravitational-potential load Love number, k_1', must be zero in a reference frame centered at the center of mass of the solid Earth (CE) because deformation of the solid Earth caused by external surface loads, once the loads are in place, cannot affect the position of the center of mass of the solid Earth (**?**, Sec. 4.1). Ensuring that $k_1' = 0$ may be accomplished by adjusting the boundary conditions such that $Y_5(a) = a\, g_S$, since $k_1' = \frac{Y_5(a)}{a\, g_S} - 1$ (e.g. **?**).

Alternatively, the load Love numbers for $n = 1$ may be computed using the same boundary conditions as for all other spherical harmonic degrees (Table 5.1), and then subsequently adjusted to ensure that load-induced changes in the degree-one gravitational potential vanish outside the Earth (e.g. **?**). The required adjustments satisfy the constraint of a vanishing gravitational potential outside the Earth by accounting for a rigid-body translation of the center of mass. Suppose that the load Love numbers derived from the solutions to the equations of motion (using boundary conditions from Table 5.1) are $h_1'^e$, $l_1'^e$, and $k_1'^e$. To satisfy the constraint of the vanishing gravitational potential, $k_1'^e$ should be subtracted from each of the Love numbers (e.g. **?**):

$$\begin{aligned} h_1' &= h_1'^e - k_1'^e \\ l_1' &= l_1'^e - k_1'^e \\ k_1' &= k_1'^e - k_1'^e. \end{aligned} \tag{5.77}$$

Note that $k_1' = 0$ after applying the correction. The other two load Love numbers, h_1' and l_1', are also adjusted slightly to account for the translation of the center of mass.

As a third option, applying a rigid-body translation to the solution vectors at the surface *prior* to computation of the Love numbers, such that $Y_5(a) + (\alpha\, g_S) = 0$ where $\alpha = -\frac{Y_5(a)}{g_S}$ (e.g. **?**), yields $Y_5(a) = 0$. With $Y_5(a) = 0$, $k_1' = \frac{Y_5(a)}{a\, g_S} - 1 = -1$, which represents k_1' in the CM reference frame. The load Love numbers in the CM frame may then be converted to the CE frame using transformation equations (**?**). Sec. 5.3.8 provides a description of reference frames applicable to the loading problem, and more information on the special degree-1 case may be found in the literature (e.g. **??????**).

5.2.13 Load Love Numbers

Load Love numbers are computed by equating linear combinations of the three independent solutions to the equations of motion with boundary conditions at the surface. In matrix form,

$$\begin{bmatrix} -g_S^2 \frac{2n+1}{4\pi G} \\ 0 \\ (2n+1)\, g_S \end{bmatrix} = \begin{bmatrix} y_2^{\mathrm{I}} & y_2^{\mathrm{II}} & y_2^{\mathrm{III}} \\ y_4^{\mathrm{I}} & y_4^{\mathrm{II}} & y_4^{\mathrm{III}} \\ (y_6^{\mathrm{I}} + \frac{n+1}{a} y_5^{\mathrm{I}}) & (y_6^{\mathrm{II}} + \frac{n+1}{a} y_5^{\mathrm{II}}) & (y_6^{\mathrm{III}} + \frac{n+1}{a} y_5^{\mathrm{III}}) \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}, \tag{5.78}$$

where the superscript Roman numerals represent each of the three independent solutions that were propagated to the surface. Note that if the y-variables are non-dimensional, then the other variables must be scaled appropriately (Sec. 5.2.6). The system of equations may now be solved for the model parameters m_1, m_2, and m_3. When the system is solved using a direct matrix inversion or the normal equations, large instabilities in the Love numbers can arise, particularly at high spherical harmonic degrees (see Sec. 5.2.17).

To promote stability, we solve the system of equations directly for all n using `numpy`, without inverting the matrix explicitly.

With the model parameters in hand, Y_1, Y_3, and Y_5 at the surface may now be derived:

$$Y_1(a) = m_1\, y_1^{\mathrm{I}} + m_2\, y_1^{\mathrm{II}} + m_3\, y_1^{\mathrm{III}} \tag{5.79}$$

$$Y_3(a) = m_1\, y_3^{\mathrm{I}} + m_2\, y_3^{\mathrm{II}} + m_3\, y_3^{\mathrm{III}} \tag{5.80}$$

$$Y_5(a) = m_1\, y_5^{\mathrm{I}} + m_2\, y_5^{\mathrm{II}} + m_3\, y_5^{\mathrm{III}}\,. \tag{5.81}$$

The load Love numbers are given by (e.g., **???**):

$$h'_n = \frac{Y_1(a)}{a} \tag{5.82}$$

$$l'_n = \frac{Y_3(a)}{a} \tag{5.83}$$

$$k'_n = \frac{Y_5(a)}{a\, g_S} - 1. \tag{5.84}$$

Since we have assumed a spherically symmetric, non-rotating, elastic and isotropic (SNREI) Earth, the load Love numbers are real-valued and latitude independent. In future versions, we aim to relax the Earth-model assumptions. Rotation and ellipticity would introduce a latitudinal dependency (e.g., **?**) and anelastic effects would produce complex-valued load Love numbers (e.g., **?**).

5.2.14 Potential Love Numbers

The potential, or "tidal" (e.g., **?**), Love numbers are computed analogously to the load Love numbers, with the exception of different boundary conditions. For potential Love numbers, boundary conditions for an external gravitational potential are applied to the equations of motion at the surface (e.g., **??**):

$$\begin{bmatrix} 0 \\ 0 \\ (2n+1)\, g_S \end{bmatrix} = \begin{bmatrix} y_2^{\mathrm{I}} & y_2^{\mathrm{II}} & y_2^{\mathrm{III}} \\ y_4^{\mathrm{I}} & y_4^{\mathrm{II}} & y_4^{\mathrm{III}} \\ (y_6^{\mathrm{I}} + \frac{n+1}{a} y_5^{\mathrm{I}}) & (y_6^{\mathrm{II}} + \frac{n+1}{a} y_5^{\mathrm{II}}) & (y_6^{\mathrm{III}} + \frac{n+1}{a} y_5^{\mathrm{III}}) \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}. \tag{5.85}$$

The potential Love numbers are derived from

$$Y_1(a) = m_1\, y_1^{\mathrm{I}} + m_2\, y_1^{\mathrm{II}} + m_3\, y_1^{\mathrm{III}} \tag{5.86}$$

$$Y_3(a) = m_1\, y_3^{\mathrm{I}} + m_2\, y_3^{\mathrm{II}} + m_3\, y_3^{\mathrm{III}} \tag{5.87}$$

$$Y_5(a) = m_1\, y_5^{\mathrm{I}} + m_2\, y_5^{\mathrm{II}} + m_3\, y_5^{\mathrm{III}} \tag{5.88}$$

using the following formulae:

$$h_n = \frac{Y_1(a)}{a} \tag{5.89}$$

$$l_n = \frac{Y_3(a)}{a} \tag{5.90}$$

$$k_n = \frac{Y_5(a)}{a\, g_S} - 1. \tag{5.91}$$

5.2.15 Shear Love Numbers

The shear Love numbers are computed analogously to the load and potential Love numbers, except for different boundary conditions. Shear-traction boundary conditions are applied to derive the shear Love numbers (Table 5.1). Furthermore,

$$k_n = \frac{Y_5(a)}{a\, g_S}, \tag{5.92}$$

since the external force is free of a gravitational potential (e.g., **??**).

5.2.16 Stress Love Numbers and Degree-1 Modes

Since potential and shear Love numbers are undefined for the degree-1 static case, stress Love numbers may be introduced to satisfy the consistency relation (e.g., **??**). The consistency relation, given by

$$y_2(r) + 2\, y_4(r) + \frac{g(r)}{4\pi G}\, y_6(r) = 0, \tag{5.93}$$

ensures that, in the static case for $n = 1$, the solid Earth experiences no net force (e.g., **??**). As with the shear Love numbers, the stress Love numbers characterize Earth's response to potential-free external forcing (Table 5.1).

One additional special consideration must be made for the degree-1 mode: accounting for a rigid-body translation (e.g., **????**). For a reference frame centered at the center of mass of the solid Earth (CE), the degree-1 potential field must vanish outside the Earth. To satisfy the restriction, a rigid-body translation may be added to the solution vector derived from the equations of motion for spheroidal deformation (e.g., **??**):

$$\begin{aligned} y_1(r) &= \alpha \\ y_2(r) &= 0 \\ y_3(r) &= \alpha \\ y_4(r) &= 0 \\ y_5(r) &= g(r)\,\alpha \\ y_6(r) &= \frac{-2\,g(r)}{r}\,\alpha, \end{aligned} \tag{5.94}$$

where

$$y_5(a)^{Load} = 1 \tag{5.95}$$

$$y_5(a)^{Stress} = 0 \tag{5.96}$$

and a corresponds to an evaluation at Earth's surface. Thus,

$$\alpha^{Load} = -\frac{y_5(a)}{g(a)} \tag{5.97}$$

$$\alpha^{Stress} = -\frac{y_5(a)}{g(a)} + \frac{1}{g(a)}. \tag{5.98}$$

A careful reader might recognize that the set of Eqs. 5.94 correspond precisely to Eqs. 5.63 for the case of $n = 1$, $\omega = 0$, and $y_5(a)$ = 1 for the load solution or $y_5(a)$ = 0 for the stress solution (**?**). Note that definitions of the "y" equations (Eqs. 5.42) may differ in the literature. For example, to convert between the convention used here (Table 5.1) (e.g., **?**) and the convention of **?** and **?**, one must divide the y-variables by a factor $[a\,g(a)]$. Furthermore, y_6 is also defined differently; thus, the convention adopted here requires an extra factor of $[\frac{-(n+1)}{r}\,g(r)\,\alpha = \frac{-2\,g(r)}{r}\,\alpha]$ for the rigid-body translation.

5.2.17 Numerical Considerations

Numerical instabilities can easily arise in load Love number computations up to spherical harmonic degree $n = 10000$ and greater, particularly since the three linearly independent starting solutions become less linearly independent with integration to the surface. In addition to non-dimensionalization, we compute the surface model parameters (Eq. 5.85) using a linear-algebra solver in `NumPy`, rather than invert the matrix explicitly. Furthermore, the integration solver can also influence stability. `LoadDef` uses a Runge-Kutta scheme with adaptive step-sizing within `Python`. We have found good stability with the `scipy` differential-equation solvers *dopri5* and *dopri853*, which perform explicit fourth- and eighth-order Runge-Kutta integration, respectively. Reducing the absolute and relative tolerance values for the integration can improve precision, albeit at the expense of computation time (e.g., **?**). `LoadDef` allows the user to specify integration tolerances and solvers. Another method for improving stability involves variable transformations (e.g., **??**). In addition, the starting solutions and choice of starting radius also have significant effects on precision and stability, as discussed in the following section.

5.2.18 Starting Radius within the Mantle

Vertical and horizontal displacements induced by either an external gravitational potential or by surface mass loading at spherical harmonic degree, n, are proportional to the external potential field, which is proportional to $\left(\frac{r}{a}\right)^n$ (e.g., **???**). Therefore, since the influence of the potential drops off rapidly inside Earth with increasing n, integration through the inner and outer cores becomes less important (and may cause numerical instabilities in the solution vectors) for spherical harmonic degrees higher than about $n = 20$ (e.g., **?**). In practice, we have found that commencing the integrations from a radius at which $\left(\frac{r}{a}\right)^n$ just exceeds a tolerance level of 10^{-5} works well. With a tolerance level of 10^{-5}, the integration for spherical harmonic degrees $n >= 20$ will commence within the mantle. `LoadDef` allows the user to specify the tolerance level for integration starting-radius.

5.2.19 Asymptotic Solutions

As a function of spherical harmonic degree, the load Love numbers approach asymptotic values after about $n = 1000$ (e.g., **??**). The asymptotic expressions may be obtained by solving the flat-Earth Boussinesq problem (e.g., **?**) or by deriving asymptotic solutions to the system of governing ordinary differential equations (e.g., **?**). Using the latter method, the asymptotic expressions are given by:

$$\begin{aligned} h'_n &= h^*_\infty + \frac{1}{n}\, h^{**}_\infty \\ n l'_n &= l^*_\infty + \frac{1}{n}\, l^{**}_\infty \\ n k'_n &= k^*_\infty + \frac{1}{n}\, k^{**}_\infty, \end{aligned} \tag{5.99}$$

where

$$\begin{aligned} h^*_\infty &= -\frac{g_S^2\, \sigma_S}{4\pi\, G\, \mu_S\, \eta_S} \\ l^*_\infty &= \frac{g_S^2}{4\pi\, G\, \eta_S} \\ k^*_\infty &= -\frac{a\, \rho_S\, g_S}{2\, \mu_S} \\ h^{**}_\infty &= \frac{g_S^2}{4\pi\, G\, \eta_S}\left[-\frac{\mu_S}{\eta_S} + \frac{a\, \rho_S\, g_S\, (\lambda_S^2 + \lambda_S\, \mu_S - \mu_S^2)}{2\, \mu_S^2\, \eta_S} + \frac{2\pi\, G\, a\, \rho_S\, \eta_S}{g_S\, \mu_S}\right] \\ l^{**}_\infty &= \frac{g_S^2}{4\pi\, G\, \eta_S}\left[-\frac{3\lambda_S^2 + 8\lambda_S\, \mu_S + 3\mu_S^2}{2\mu_S\, \eta_S} + \frac{a\, \rho_S\, g_S\, \sigma_S}{2\mu_S\, \eta_S}\right] \\ k^{**}_\infty &= \frac{a\, g_S\, \rho_S}{\mu_S}\left[\frac{\lambda_S}{4\eta_S} + \frac{a\, \rho_S\, g_S\, (2\lambda_S + \mu_S)}{8\mu_S\, \eta_S} + \frac{\pi\, G\, a\, \rho_S}{g_S}\right] \end{aligned} \tag{5.100}$$

and

$$\begin{aligned} \sigma_S &= \lambda_S + 2\, \mu_S \\ \eta_S &= \lambda_S + \mu_S\, . \end{aligned} \tag{5.101}$$

A subscript S refers to the value of the parameter at Earth's surface.

5.3 Load Green's Functions

5.3.1 Introduction

Infinite sums of Love numbers may be formed to determine the impulse-response function, or **Green's Function**, of a body to a certain stimulus. `LoadDef` computes Green's functions for surface mass loading boundary conditions, though a similar procedure may be applied to other types of boundary conditions, such as an external gravitational potential. The variables Y_1, Y_3, and Y_5 obtained in Sec. 5.2.13 represent the radial coefficients of the spherical harmonic expansions (Eqs. 5.29). Namely, for mass loading at Earth's surface,

$$\begin{aligned} U_n(r) &= Y_1 = a\,h'_n \\ V_n(r) &= Y_3 = a\,l'_n \\ \Psi_n(r) &= Y_5 = a\,g\,(k'_n + 1), \end{aligned} \tag{5.102}$$

where a is Earth's radius. Referring back to Eqs. 5.29, the radial displacement for spherical harmonic degree n is given by:

$$\begin{aligned} u_n &= U_n(r)\,P_n(\cos\theta)\,e^{i\omega t} \\ &= a\,h'_n\,P_n(\cos\theta)\,e^{i\omega t}, \end{aligned} \tag{5.103}$$

where $e^{i\omega t}$ represents the temporal evolution of the applied load, such as a periodic ocean tide.

Since the boundary conditions in Sec. 5.2.12 were formulated based on the mass of the Earth, Eq. 5.103 may be rewritten to represent an arbitrary mass load m' (e.g., **?**):

$$\begin{aligned} u_n &= a\,\frac{m'}{m_E}\,h'_n\,P_n(\cos\theta)\,e^{i\omega t} \\ &= \frac{a}{m_E}\,h'_n\,P_n(\cos\theta)\,e^{i\omega t}, \end{aligned} \tag{5.104}$$

where, in the second line, we have taken m' to be a load of unit mass.

The vertical-displacement load Green's function (LGF) for a 1-kg load applied at Earth's surface is given by a summation of Eq. 5.104 over all n:

$$u = \frac{a}{m_E}\sum_{n=0}^{\infty} h'_n\,P_n(\cos\theta)\,e^{i\omega t}. \tag{5.105}$$

Similarly, the horizontal-displacement LGF for a 1-kg load at Earth's surface is given by:

$$v = \frac{a}{m_E}\sum_{n=1}^{\infty} l'_n\,\frac{\partial P_n(\cos\theta)}{\partial\theta}\,e^{i\omega t}. \tag{5.106}$$

Note that the sum for v begins at $n = 1$, since horizontal displacements do not apply to the degree-0 mode.

The radial coefficients of the spherical harmonic expansions may also be expressed in terms of the transformed surface potential for a point load of unit mass, $\Phi_{2,n}$ (e.g., **????**):

$$\begin{aligned} U_n(r) &= h'_n(r)\,\frac{\Phi_{2,n}}{g} \\ V_n(r) &= l'_n(r)\,\frac{\Phi_{2,n}}{g}\,, \end{aligned} \tag{5.107}$$

where

$$\Phi_{2,n} = \frac{4\pi G a}{2n+1}\,K_n = \frac{a\,g}{m_E}. \tag{5.108}$$

The displacement LGFs in Eqs. 5.105 and 5.106 may also be derived from Eqs. 5.107 and 5.108. In other words, the load Love numbers scale the equipotential height, $\frac{\Phi_{2,n}}{g}$, to the true displacements expected for Earth's material structure. To derive the predicted SML-induced displacements within Earth's interior, an extra factor of $\left(\frac{r}{a}\right)^n$ must be included in Eq. 5.108 (e.g., **??**).

In summary, the amplitudes of the displacement LGFs (i.e., written without the dynamic component of the forcing term) are:

$$u = \frac{a}{m_E}\sum_{n=0}^{\infty} h'_n\,P_n(\cos\theta) \tag{5.109}$$

and

$$v = \frac{a}{m_E}\sum_{n=1}^{\infty} l'_n\,\frac{\partial P_n(\cos\theta)}{\partial\theta}. \tag{5.110}$$

Displacement LGFs are appropriate for comparing predicted load-induced surface displacements with GNSS-inferred observations of surface displacements.

For surface displacements induced by an external gravitational potential, the potential Love numbers scale the equipotential height generated by the external gravitational potential, V_n (Eq. 5.2). For a mass a distance R away from Earth's center, the gravitational potential at Earth's surface, a, is given by:

$$V_n^{\text{potential}} = \frac{GM}{R}\left(\frac{a}{R}\right)^n P_n(\cos\theta)e^{i\omega t}, \tag{5.111}$$

where the time-dependent harmonic term ($e^{i\omega t}$) has been included to account for periodic dynamic forcing. Thus, the vertical and horizontal displacements due to an external gravitational potential may be derived by inserting the expression for V_n (Eq. 5.111) into Eqs. 5.2 and 5.4, respectively, and summing over all n.

`LoadDef` also computes Green's functions for gravity, tilt and strain (e.g., **?????**).

The equation for the indirect (elastic) effect of gravity is (e.g., **??**):

$$g^E = \frac{g_a}{M_E} \sum_{n=0}^{\infty} 2h'_n \, P_n(\cos\theta) - (n+1) \, k'_n \, P_n(\cos\theta), \tag{5.112}$$

where g_a is the acceleration due to gravity at the surface and M_E is the mass of the Earth. The direct, or Newtonian, effect of gravity is given by (e.g., **?**):

$$g^N = -\frac{g_a}{4 \, M_E \, \sin(\theta/2)}. \tag{5.113}$$

The equation for the indirect (elastic) effect of tilt is (e.g., **??**):

$$t^E = -\frac{1}{M_E} \sum_{n=1}^{\infty} (-h'_n + k'_n) \, \frac{d}{d\theta} P_n(\cos\theta). \tag{5.114}$$

The direct, or Newtonian, effect of tilt is given by (e.g., **?**):

$$t^N = \frac{\cos(\theta/2)}{4 \, M_E \, \sin^2(\theta/2)}. \tag{5.115}$$

The diagonal components of the strain tensor are given by (e.g., **????**, Sec. 3.06.4.2)

$$\epsilon_{\lambda\lambda} = \frac{1}{a} u + \frac{\cot\theta}{a} v \tag{5.116}$$

$$\epsilon_{\theta\theta} = \frac{1}{a} u + \frac{1}{a} \frac{\partial v}{\partial \theta} = \frac{1}{a} u + \frac{1}{M_E} \sum_{n=1}^{\infty} l'_n \frac{d^2}{d\theta^2} P_n(\cos\theta) \tag{5.117}$$

$$\epsilon_{rr} = \frac{\partial u}{\partial r} = -\frac{\lambda_a}{\lambda_a + 2\mu_a} (\epsilon_{\theta\theta} + \epsilon_{\lambda\lambda}), \tag{5.118}$$

where u represents radial displacement at the surface, v represents tangential displacement at the surface, and λ_a and μ_a are the Lamé parameters at the surface. In the spherical coordinate system, θ is the surface-tangential component in the longitudinal direction, λ is the surface-tangential component in the azimuthal direction, and r is the radial component. Eq. 5.135 may be solved directly using the displacement load

Green's functions. Plugging in expressions for u and v, Eq. 5.117 becomes

$$\epsilon_{\theta\theta} = \frac{1}{M_E}\sum_{n=0}^{\infty} h'_n P_n(\cos\theta) + \frac{1}{M_E}\sum_{n=0}^{\infty} l'_n \frac{\partial^2 P_n(\cos\theta)}{\partial\theta^2}. \tag{5.119}$$

? discuss how to rotate the components of the strain tensor into a geographical configuration (Eqs. 1.114–1.117). **?** provides an expression for computing strain along a desired azimuth (Eq. 52).

5.3.2 Kummer's Transformation and Series Convergence

The series in Eqs. 5.109, 5.110, and 5.112–5.118 can be slow to converge. Kummer's series transformation may be implemented to speed convergence (e.g., **????**). The general form of Kummer's transformation is given by (e.g., **??**):

$$\sum_n f(n)Q_n = f_\infty \sum_n Q_n + \sum_n \left(f(n) - f_\infty\right) Q_n, \tag{5.120}$$

where $f_\infty = \lim_{n\to\infty} f(n)$.

Thus, Eqs. 5.109 and 5.110 may be expressed in terms of the asymptotic expressions of the load Love numbers as:

$$u = \frac{a}{m_E} h^*_\infty \sum_{n=0}^{\infty} P_n(\cos\theta) + \frac{a}{m_E}\sum_{n=0}^{\infty}(h'_n - h^*_\infty)\, P_n(\cos\theta) \tag{5.121}$$

and

$$v = \frac{a}{m_E} l^*_\infty \sum_{n=1}^{\infty} \frac{1}{n}\frac{\partial P_n(\cos\theta)}{\partial\theta} + \frac{a}{m_E}\sum_{n=1}^{\infty}(nl'_n - l^*_\infty)\,\frac{1}{n}\frac{\partial P_n(\cos\theta)}{\partial\theta}. \tag{5.122}$$

The factor $\frac{1}{n}$ in Eq. 5.122 enters because l^*_∞ represents the asymptotic value of nl'_n (Eq. 5.99) (cf., **??**).

Eqs. 5.121 and 5.122 include only the first term in the asymptotic expressions of the load Love numbers (Eq. 5.99). **?** introduced an additional term to the asymptotic expressions, improving accuracy by a factor $\frac{1}{n}$. With the extra term included, Eqs. 5.109 and 5.110 become:

$$\begin{aligned} u \;=\; & \frac{a}{m_E} h^*_\infty \sum_{n=1}^{\infty} P_n(\cos\theta) + \frac{a}{m_E} h^{**}_\infty \sum_{n=1}^{\infty} \frac{1}{n} P_n(\cos\theta) + \\ & \frac{a}{m_E} h'_0 + \frac{a}{m_E} \sum_{n=1}^{N_{\max}} \left(h'_n - (h^*_\infty + \frac{1}{n} h^{**}_\infty)\right) P_n(\cos\theta) \end{aligned} \tag{5.123}$$

and

$$\begin{aligned} v &= \frac{a}{m_E} l^*_\infty \sum_{n=1}^{\infty} \frac{1}{n} \frac{\partial P_n(\cos\theta)}{\partial\theta} + \frac{a}{m_E} l^{**}_\infty \sum_{n=1}^{\infty} \frac{1}{n^2} \frac{\partial P_n(\cos\theta)}{\partial\theta} + \\ & \frac{a}{m_E} \sum_{n=1}^{N_{\max}} (n l'_n - (l^*_\infty + \frac{1}{n} l^{**}_\infty)) \frac{1}{n} \frac{\partial P_n(\cos\theta)}{\partial\theta}, \end{aligned} \tag{5.124}$$

where $N_{\max}$ represents the maximum degree n after which the asymptotic values are assumed. Note that the second-order terms for the asymptotes are undefined for $n = 0$. In practice, the load Love number computations are often carried out to spherical harmonic degree $n = 10000$, beyond which the load Love numbers are assumed to be equivalent to the asymptotic values (Eq. 5.99) (e.g., **??**). Thus, the arguments in Eqs. 5.123 and 5.124 that contain load Love numbers become zero beyond spherical harmonic degree, N. Furthermore, some Legendre sums are known analytically, which will be discussed in Sec. 5.3.3, and other Legendre polynomials and their derivatives may be computed recursively using the so-called *recursion relations*, which will be discussed in Sec. 5.3.5.

In summary, the displacement load Green's functions are computed using the formulae:

$$\begin{aligned} u(\theta) &= \frac{a}{m_E} h^*_\infty \sum_{n=1}^{\infty} P_n(\cos\theta) + \frac{a}{m_E} h^{**}_\infty \sum_{n=1}^{\infty} \frac{1}{n} P_n(\cos\theta) + \\ & \frac{a}{m_E} h'_0 + \frac{a}{m_E} \sum_{n=1}^{N_{\max}} (h'_n - (h^*_\infty + \frac{1}{n} h^{**}_\infty)) P_n(\cos\theta) \end{aligned} \tag{5.125}$$

for the vertical-displacement response, and

$$\begin{aligned} v(\theta) &= \frac{a}{m_E} l^*_\infty \sum_{n=1}^{\infty} \frac{1}{n} \frac{\partial P_n(\cos\theta)}{\partial\theta} + \frac{a}{m_E} l^{**}_\infty \sum_{n=1}^{\infty} \frac{1}{n^2} \frac{\partial P_n(\cos\theta)}{\partial\theta} + \\ & \frac{a}{m_E} \sum_{n=1}^{N_{\max}} (n l'_n - (l^*_\infty + \frac{1}{n} l^{**}_\infty)) \frac{1}{n} \frac{\partial P_n(\cos\theta)}{\partial\theta} \end{aligned} \tag{5.126}$$

for the horizontal-displacement response. The Legendre sums without Love-number coefficients may be determined analytically, as discussed in Sec. 5.3.3. The Legendre contributions to the final terms in Eqs. 5.125 and 5.126 are computed recursively (Sec. 5.3.5). For $\theta = 180°$, we compute the limits of the Legendre functions and their derivatives, as well as the limits of the Legendre sums, as θ approaches $180°$.

The Kummer transformations for gravity, tilt, and strain are performed in a similar way.

For the indirect effect of gravity,

$$\begin{aligned}
g^E &= \left(\frac{2g_a}{M_E}\sum_{n=0}^{\infty} h'_n\,P_n(\cos\theta)\right) - \left(\frac{g_a}{M_E}\sum_{n=0}^{\infty}(n+1)\,\frac{n\,k'_n}{n}\,P_n(\cos\theta)\right)\\
&= \left(\frac{2g_a}{M_E}\sum_{n=0}^{\infty} h'_n\,P_n(\cos\theta)\right) - \left(\frac{g_a}{M_E}\sum_{n=0}^{\infty}\left[n\,\frac{n\,k'_n}{n}\,P_n(\cos\theta) + \frac{n\,k'_n}{n}\,P_n(\cos\theta)\right]\right)\\
&= \left(\frac{2g_a}{M_E}\sum_{n=0}^{\infty} h'_n\,P_n(\cos\theta)\right) - \left(\frac{g_a}{M_E}\sum_{n=0}^{\infty}\left[n\,k'_n\,P_n(\cos\theta) + n\,k'_n\,\frac{1}{n}\,P_n(\cos\theta)\right]\right)\\
&= 2g_a\left(\frac{u}{a}\right) - \left(\frac{g_a}{M_E}\sum_{n=0}^{\infty}\left[n\,k'_n\,P_n(\cos\theta) + n\,k'_n\,\frac{1}{n}\,P_n(\cos\theta)\right]\right). \qquad (5.127)
\end{aligned}$$

Isolating components, we have:

$$\begin{aligned}
\sum_{n=0}^{\infty}(n\,k'_n)\,P_n(\cos\theta) &= k^*_\infty\sum_{n=0}^{\infty}P_n(\cos\theta) + k^{**}_\infty\sum_{n=0}^{\infty}\frac{1}{n}P_n(\cos\theta)\\
&\quad + \sum_{n=0}^{N_{\max}}\left[\left(n(n\,k'_n - k^*_\infty) - k^{**}_\infty\right)/n\right]\,P_n(\cos\theta) \qquad (5.128)
\end{aligned}$$

and

$$\sum_{n=0}^{\infty}(n\,k'_n)\,\frac{1}{n}\,P_n(\cos\theta) = k^*_\infty\sum_{n=0}^{\infty}\frac{1}{n}\,P_n(\cos\theta) + \sum_{n=0}^{N_{\max}}\left(n\,k'_n - k^*_\infty\right)\,\frac{1}{n}\,P_n(\cos\theta). \qquad (5.129)$$

For the second sum (Eq. 5.129), we opted to perform only a first-order Kummer transformation, which still facilitates a sufficient convergence of the series.

For the indirect effect of tilt,

$$\begin{aligned}
t^E &= -\frac{1}{M_E}\sum_{n=1}^{\infty}(-h'_n + k'_n)\,\frac{d}{d\theta}P_n(\cos\theta)\\
&= \left[-\frac{1}{M_E}\sum_{n=1}^{\infty}(-h'_n)\,\frac{d}{d\theta}P_n(\cos\theta)\right] + \left[-\frac{1}{M_E}\sum_{n=1}^{\infty}(k'_n)\,\frac{d}{d\theta}P_n(\cos\theta)\right]\\
&= \left[\frac{1}{M_E}\sum_{n=1}^{\infty}h'_n\,\frac{d}{d\theta}P_n(\cos\theta)\right] + \left[-\frac{1}{M_E}\sum_{n=1}^{\infty}\frac{nk'_n}{n}\,\frac{d}{d\theta}P_n(\cos\theta)\right]\\
&= \left[\frac{1}{M_E}\sum_{n=1}^{\infty}h'_n\,\frac{d}{d\theta}P_n(\cos\theta)\right] - \left[\frac{1}{M_E}\sum_{n=1}^{\infty}(nk'_n)\frac{1}{n}\,\frac{d}{d\theta}P_n(\cos\theta)\right]. \qquad (5.130)
\end{aligned}$$

Isolating components, we have:

$$\begin{aligned}\sum_{n=1}^{\infty} h'_n \frac{d}{d\theta} P_n(\cos\theta) &= h^*_\infty \sum_{n=1}^{\infty} \frac{d}{d\theta} P_n(\cos\theta) + h^{**}_\infty \sum_{n=1}^{\infty} \frac{1}{n}\frac{d}{d\theta} P_n(\cos\theta) \\ &\quad + \sum_{n=1}^{N_{\max}} \left[\left(n(h'_n - h^*_\infty) - h^{**}_\infty\right)/n\right] \frac{d}{d\theta} P_n(\cos\theta) \end{aligned} \tag{5.131}$$

and

$$\begin{aligned}\sum_{n=1}^{\infty} (n\, k'_n) \frac{1}{n}\frac{d}{d\theta} P_n(\cos\theta) &= k^*_\infty \sum_{n=1}^{\infty} \frac{1}{n}\frac{d}{d\theta} P_n(\cos\theta) + k^{**}_\infty \sum_{n=1}^{\infty} \frac{1}{n^2}\frac{d}{d\theta} P_n(\cos\theta) \\ &\quad + \sum_{n=1}^{N_{\max}} \left[\left(n(nk'_n - k^*_\infty) - k^{**}_\infty\right)/n\right] \frac{1}{n}\frac{d}{d\theta} P_n(\cos\theta). \end{aligned} \tag{5.132}$$

For the strain component $\epsilon_{\theta\theta}$,

$$\begin{aligned}\epsilon_{\theta\theta} &= \frac{u}{a} + \frac{1}{M_E} \sum_{n=1}^{\infty} l'_n \frac{d^2}{d\theta^2} P_n(\cos\theta) \\ &= \frac{u}{a} + \frac{1}{M_E} \sum_{n=1}^{\infty} (nl'_n) \frac{1}{n}\frac{d^2}{d\theta^2} P_n(\cos\theta), \end{aligned} \tag{5.133}$$

where

$$\begin{aligned}\sum_{n=1}^{\infty} (nl'_n) \frac{1}{n}\frac{d^2}{d\theta^2} P_n(\cos\theta) &= l^*_\infty \sum_{n=1}^{\infty} \frac{1}{n}\frac{d^2}{d\theta^2} P_n(\cos\theta) + l^{**}_\infty \sum_{n=1}^{\infty} \frac{1}{n^2}\frac{d^2}{d\theta^2} P_n(\cos\theta) \\ &\quad + \sum_{n=1}^{N_{\max}} \left[\left(n(nl'_n - l^*_\infty) - l^{**}_\infty\right)/n\right] \frac{1}{n}\frac{d^2}{d\theta^2} P_n(\cos\theta). \end{aligned} \tag{5.134}$$

The strain component $\epsilon_{\lambda\lambda}$ may be computed directly from the vertical and horizontal displacements:

$$\epsilon_{\lambda\lambda} = \frac{1}{a}\, u + \frac{\cot\theta}{a}\, v. \tag{5.135}$$

Finally, the strain component ϵ_{rr} may be computed directly from $\epsilon_{\theta\theta}$ and $\epsilon_{\lambda\lambda}$:

$$\epsilon_{rr} = -\frac{\lambda_a}{\lambda_a + 2\mu_a}(\epsilon_{\theta\theta} + \epsilon_{\lambda\lambda}). \tag{5.136}$$

To further improve the convergence of the series, we adopt the method of recursive averages suggested by **?**. The recursive averaging can be performed by developing a set of coefficients for each harmonic degree n that effectively "taper" the summands as the series approaches $n = n_{\max}$, where $n_{\max}$ is often taken to be 10 000. Most of the coefficients will be one, except for as $n = n_{\max}$ is approached and the coefficients are reduced towards zero. The point at which the taper starts depends on the number of recursive averages performed. `LoadDef` nominally selects 200 iterations for the taper.

5.3.3 Legendre Sums

?, **?**, **?**, **?** and **?** provide closed-form expressions for several Legendre sums. **?** include the most comprehensive collection of formulas. Equation B9 from **?**, however, contains a typo and should instead read (J.-Y. Guo, personal communication, 20 June 2016):

$$\begin{aligned}\sum_{n=1}^{\infty}\frac{1}{n^2}\,x^n\,\frac{d^2}{d\theta^2}P_n(\cos\theta) &= -\frac{\cos\theta}{\sin^2\theta}\ln\frac{(x-l+1)^2(1-\cos\theta)}{2(l-1+x\cos\theta)(1+\cos\theta)} \\ &\quad+\frac{1}{\sin^2\theta}\ln\frac{x^2\,\sin^2\theta}{2(l-1+x\cos\theta)} \\ &\quad+\frac{x(\cos\theta-1)(1-l)}{(l-1+x\cos\theta)l}-\frac{2x}{(x-l+1)l}+2. \end{aligned} \tag{5.137}$$

The only difference is that $(\cos\theta - l)$ should instead be $(\cos\theta - 1)$ in the final line. The equation may then be further reduced using half-angle formulas from trigonometry.

Many of the Legendre sums that appear in Eqs. 5.123–5.134 are known analytically (e.g., J.-Y. Guo, personal communication, 20 June 2016; **???**):

$$\begin{aligned}\sum_{n=0}^{\infty}P_n(\cos\theta) &= \frac{1}{2\sin(\theta/2)} \\ \sum_{n=0}^{\infty}n\,P_n(\cos\theta) &= -\frac{1}{4\,\sin(\theta/2)} \\ \sum_{n=1}^{\infty}\frac{1}{n}\,P_n(\cos\theta) &= \ln\frac{1}{\sin(\theta/2)\,[1+\sin(\theta/2)]} \\ \sum_{n=1}^{\infty}\frac{\partial P_n(\cos\theta)}{\partial\theta} &= -\frac{\cos(\theta/2)}{4\,\sin^2(\theta/2)} \\ \sum_{n=1}^{\infty}\frac{1}{n}\,\frac{\partial P_n(\cos\theta)}{\partial\theta} &= -\frac{\cos(\theta/2)\,[1+2\,\sin(\theta/2)]}{2\,\sin(\theta/2)\,[1+\sin(\theta/2)]}\end{aligned} \tag{5.138}$$

$$\begin{aligned}
\sum_{n=1}^{\infty} \frac{1}{n^2} \frac{\partial P_n(\cos\theta)}{\partial\theta} &= \frac{1}{\sin\theta} \ln \frac{\sin(\theta/2)\,[1-\sin(\theta/2)]}{\cos^2(\theta/2)} - \frac{\cos\theta}{\sin\theta} \ln \frac{\sin(\theta/2)\,\cos^2(\theta/2)}{1-\sin(\theta/2)} \\
\sum_{n=1}^{\infty} \frac{1}{n} \frac{\partial^2 P_n(\cos\theta)}{\partial\theta^2} &= \frac{1+\sin(\theta/2)+\sin^2(\theta/2)}{4\sin^2(\theta/2)\,[1+\sin(\theta/2)]} \\
\sum_{n=1}^{\infty} \frac{1}{n^2} \frac{\partial^2 P_n(\cos\theta)}{\partial\theta^2} &= -\frac{\cos\theta}{\sin^2\theta} \ln \frac{\sin(\theta/2)\,[1-\sin(\theta/2)]}{\cos^2(\theta/2)} \\
&\quad + \frac{1}{\sin^2\theta} \ln \frac{\sin(\theta/2)\,\cos^2(\theta/2)}{1-\sin(\theta/2)} - \frac{1-2\sin(\theta/2)}{2\sin(\theta/2)}. \qquad (5.139)
\end{aligned}$$

5.3.4 Legendre Sums: Limit as $\theta \to 180^\circ$

In the limit $\theta \to 180^\circ$, the Legendre sums reduce to:

$$\begin{aligned}
\sum_{n=0}^{\infty} P_n(\cos\theta) &= \frac{1}{2} \\
\sum_{n=0}^{\infty} n\,P_n(\cos\theta) &= -\frac{1}{4} \\
\sum_{n=1}^{\infty} \frac{1}{n} P_n(\cos\theta) &= \ln\left(\frac{1}{2}\right) = -\ln(2) \\
\sum_{n=1}^{\infty} \frac{\partial P_n(\cos\theta)}{\partial\theta} &= 0 \\
\sum_{n=1}^{\infty} \frac{1}{n} \frac{\partial P_n(\cos\theta)}{\partial\theta} &= 0 \qquad (5.140) \\
\sum_{n=1}^{\infty} \frac{1}{n^2} \frac{\partial P_n(\cos\theta)}{\partial\theta} &= 0 \\
\sum_{n=1}^{\infty} \frac{1}{n} \frac{\partial^2 P_n(\cos\theta)}{\partial\theta^2} &= \frac{3}{8} \\
\sum_{n=1}^{\infty} \frac{1}{n^2} \frac{\partial^2 P_n(\cos\theta)}{\partial\theta^2} &= \frac{\ln(2)}{2} + \frac{1}{4}. \qquad (5.141)
\end{aligned}$$

We computed the sums directly where possible, and otherwise used the `Python` package `SymPy`.

5.3.5 Legendre Polynomial Recursion Relations

To evaluate the final terms in Eqs. 5.125 and 5.126, the Legendre functions and their derivatives must be determined for every n. Recursion relations, derived from the Legendre generating function, are commonly used (e.g., **???**).

Two useful recursion, or recurrence, relations are (e.g., **?**):

$$
\begin{aligned}
nP_n(x) &= (2n-1)xP_{n-1}(x) - (n-1)P_{n-2}(x) && (5.142)\\
(1-x^2)\frac{\partial P_n(x)}{\partial x} &= nP_{n-1}(x) - nxP_n(x). && (5.143)
\end{aligned}
$$

For $x = \cos\theta$, the recursion relations become:

$$
\begin{aligned}
nP_n(\cos\theta) &= (2n-1)\cos\theta P_{n-1}(\cos\theta) - && (5.144)\\
&\quad (n-1)P_{n-2}(\cos\theta) &&\\
(1-\cos^2\theta)\frac{\partial P_n(\cos\theta)}{\partial\cos\theta} &= nP_{n-1}(\cos\theta) - n\cos\theta P_n(\cos\theta). && (5.145)
\end{aligned}
$$

Since Eq. 5.126 requires the derivative of the Legendre function with respect to θ (as opposed to $\cos\theta$), each side of Eq. 5.145 can be multiplied by $\frac{\partial\cos\theta}{\partial\theta}$, leading to:

$$
\begin{aligned}
(1-\cos^2\theta)\frac{\partial P_n(\cos\theta)}{\partial\cos\theta}\frac{\partial\cos\theta}{\partial\theta} &= [nP_{n-1}(\cos\theta) - n\cos\theta P_n(\cos\theta)]\frac{\partial\cos\theta}{\partial\theta}\\
(\sin^2\theta)\frac{\partial P_n(\cos\theta)}{\partial\theta} &= [nP_{n-1}(\cos\theta) - n\cos\theta P_n(\cos\theta)](-\sin\theta)\\
\frac{\partial P_n(\cos\theta)}{\partial\theta} &= -\frac{n}{\sin\theta}[P_{n-1}(\cos\theta) - \cos\theta P_n(\cos\theta)]. && (5.146)
\end{aligned}
$$

Legendre's differential equation is given by (e.g., **??**, Eq. 12.40)

$$
(1-x^2)\frac{\partial^2 P(x)}{\partial x^2} - 2x\frac{\partial P(x)}{\partial x} + l(l+1)P(x) = 0. \qquad (5.147)
$$

For $x = \cos\theta$, Eq. 5.147 becomes

$$
(1-\cos^2\theta)\frac{d^2P(\cos\theta)}{d\cos^2\theta} - 2\cos\theta\frac{dP(\cos\theta)}{d\cos\theta} + l(l+1)P(\cos\theta) = 0. \qquad (5.148)
$$

To evaluate the strain load Green's functions (SLGFs), however, we need to derive a recurrence relation for

$\frac{d^2P(\cos\theta)}{d\theta^2}$ rather than for $\frac{d^2P(\cos\theta)}{d\cos^2\theta}$. From the chain rule in calculus, we can write

$$\frac{d}{d\cos\theta} = \frac{d\theta}{d\cos\theta}\frac{d}{d\theta}, \tag{5.149}$$

which, when operating on $P(\cos\theta)$, becomes

$$\begin{aligned}\frac{dP(\cos\theta)}{d\cos\theta} &= \frac{d\theta}{d\cos\theta}\frac{dP(\cos\theta)}{d\theta} \\ &= -\frac{1}{\sin\theta}\frac{dP(\cos\theta)}{d\theta}.\end{aligned} \tag{5.150}$$

The second derivative term in Eq. 5.148, $\frac{d^2P(\cos\theta)}{d\cos^2\theta}$, becomes

$$\begin{aligned}\frac{d}{d\cos\theta}\left[\frac{dP(\cos\theta)}{d\cos\theta}\right] &= \frac{d\theta}{d\cos\theta}\frac{d}{d\theta}\left[\frac{dP(\cos\theta)}{d\cos\theta}\right] \\ &= -\frac{1}{\sin\theta}\frac{d}{d\theta}\left[\frac{d\theta}{d\cos\theta}\frac{d}{d\theta}P(\cos\theta)\right] \\ &= -\frac{1}{\sin\theta}\frac{d}{d\theta}\left[-\frac{1}{\sin\theta}\frac{dP(\cos\theta)}{d\theta}\right] \\ &= \frac{1}{\sin\theta}\frac{d}{d\theta}\left[\frac{1}{\sin\theta}\frac{dP(\cos\theta)}{d\theta}\right] \\ &= \frac{1}{\sin\theta}\left[\frac{dP(\cos\theta)}{d\theta}\left(\frac{-\cos\theta}{\sin^2\theta}\right) + \frac{1}{\sin\theta}\frac{d^2P(\cos\theta)}{d\theta^2}\right] \\ &= \frac{1}{\sin^2\theta}\left[\frac{d^2P(\cos\theta)}{d\theta^2} - \frac{\cos\theta}{\sin\theta}\frac{dP(\cos\theta)}{d\theta}\right].\end{aligned} \tag{5.151}$$

Thus, plugging Eq. 5.151 into Eq. 5.148, we have (e.g., **??**, Eq. 12.29)

$$\begin{aligned}0 &= \sin^2\theta\left[\frac{1}{\sin^2\theta}\left\{\frac{d^2P(\cos\theta)}{d\theta^2} - \frac{\cos\theta}{\sin\theta}\frac{dP(\cos\theta)}{d\theta}\right\}\right] \\ &\quad -2\cos\theta\left[-\frac{1}{\sin\theta}\frac{dP(\cos\theta)}{d\theta}\right] + l(l+1)P(\cos\theta) \\ &= \frac{d^2P(\cos\theta)}{d\theta^2} - \frac{\cos\theta}{\sin\theta}\frac{dP(\cos\theta)}{d\theta} + 2\frac{\cos\theta}{\sin\theta}\frac{dP(\cos\theta)}{d\theta} + l(l+1)P(\cos\theta) \\ &= \frac{d^2P(\cos\theta)}{d\theta^2} + \frac{\cos\theta}{\sin\theta}\frac{dP(\cos\theta)}{d\theta} + l(l+1)P(\cos\theta).\end{aligned} \tag{5.152}$$

Thus,

$$\frac{d^2P(\cos\theta)}{d\theta^2} = -\frac{\cos\theta}{\sin\theta}\frac{dP(\cos\theta)}{d\theta} - l(l+1)P(\cos\theta). \tag{5.153}$$

5.3.6 Legendre Polynomial Recursion Relations: Limit as $\theta \to 180°$

In the limit $\theta \to 180°$, the Legendre function are given by:

$$\begin{aligned}
P_n(\cos\theta) &= (-1)^n \\
\frac{dP_n(\cos\theta)}{d\theta} &= 0 \\
\frac{d^2P_n(\cos\theta)}{d\theta^2} &= (-1)^{n+1}\left(\frac{n\,(n+1)}{2}\right).
\end{aligned} \tag{5.154}$$

We evaluated the limits at $\theta = 180°$ using using the `Python` package `SymPy`.

For the tangential component of strain, $\epsilon_{\lambda\lambda}$ (Eq. 5.135), a factor of $\cos\theta$ appears in the second term that must also be evaluated in the limit that $\theta \to 180°$. For this component, we evaluate the limit as $\theta \to 180°$ of:

$$\cot(\theta)\,\frac{dP_n(\cos\theta)}{d\theta}, \tag{5.155}$$

where the Legendre polynomial arises from the series expansion for horizontal displacement, v (Eq. 5.126). From `SymPy`, the limit evaluates to:

$$(-1)^{n+1}\left(\frac{n\,(n+1)}{2}\right), \tag{5.156}$$

which is equivalent to the limit as $\theta \to 180°$ of $\frac{d^2P_n(\cos\theta)}{d\theta^2}$. As a result, $\epsilon_{\lambda\lambda} = \epsilon_{\theta\theta}$ at $\theta = 180°$.

5.3.7 Disk Factor

To speed the convergence of the series in Eqs. 5.125 and 5.126, distant loads (e.g., several tens of degrees away from the observer) may be approximated by finite circular caps rather than delta functions (e.g., **?**). In practice, the disk factor from Eq. 5.75 may be inserted back into Eqs. 5.125 and 5.126. For displacement LGFs, which converge relatively rapidly, disk factors are generally not necessary; however, disk factors can be very useful for other types of LGFs, such as tilt and strain (e.g., **?**).

The disk factor,

$$\left[-\frac{(1+\cos\alpha)}{n(n+1)\sin\alpha}\frac{\partial P_n(\cos\alpha)}{\partial\alpha}\right], \tag{5.157}$$

is only valid in the limit $\alpha \to 0$, where α specifies the finiteness of the circular cap (e.g., **?**); thus, the disk factor should only be invoked with small α (e.g., $\sim 0.004°$) or only at large angular distances.

5.3.8 Reference Frames

The vector displacement field generated by surface mass loading depends both on the physical characteristics of the deformation as well as the chosen reference frame (e.g., **??**). The load Love numbers and corresponding load Green's functions described thus far have been computed in a reference frame fixed to the center of mass of the solid Earth, abbreviated CE. The CE reference frame is convenient for computing Love numbers and Green's functions, but not directly observable in practice (e.g., **???**). More appropriate reference frames for GNSS-inferred surface displacements are defined by the center of mass of the entire Earth system, abbreviated CM, which includes the solid Earth as well as its fluid exterior (e.g., oceans and atmosphere), or the center of figure, abbreviated CF (e.g., **??**).

? demonstrated that conversions between the various reference frames involve only simple transformations of the degree-one load Love numbers. The conversions for CE to CM, for example, are given by:

$$\begin{aligned}
[h_1']_{CM} &= [h_1']_{CE} - 1 \\
[l_1']_{CM} &= [l_1']_{CE} - 1 \\
[1 + k_1']_{CM} &= [1 + k_1']_{CE} - 1.
\end{aligned} \tag{5.158}$$

Similarly, the conversions for CE to CF (center of figure) are given by:

$$\begin{aligned}
[h_1']_{CF} &= \frac{2}{3}\,[h_1' - l_1']_{CE} \\
[l_1']_{CF} &= -\frac{1}{3}\,[h_1' - l_1']_{CE} \\
[1 + k_1']_{CF} &= \left[1 - \frac{1}{3}h_1' - \frac{2}{3}l_1'\right]_{CE}.
\end{aligned} \tag{5.159}$$

From the equations for the displacement LGFs (Eqs. 5.109 and 5.110), the degree-1 components are:

$$u_1 = \frac{a}{m_E}\,h_1'\,\cos\theta \tag{5.160}$$

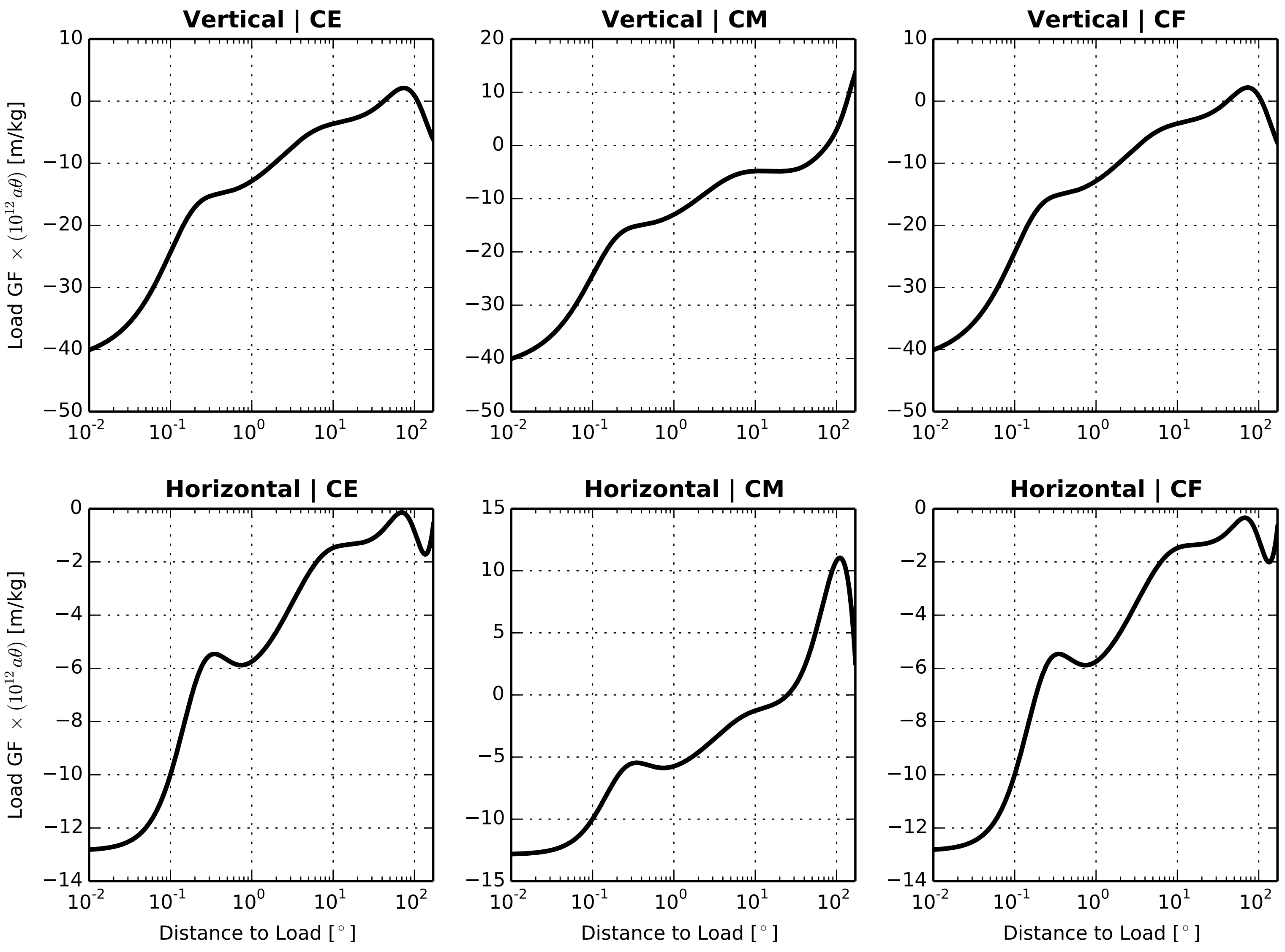

Figure 5.1: Vertical (top) and horizontal (bottom) displacement load Green's functions (LGFs) for a 1 kg surface mass load, computed using an isotropic and oceanless version of PREM. The left panels show the LGFs in the CE reference frame, the central panels show the LGFs in the CM reference frame, and the right panels show the LGFs in the CF reference frame (**?**). The load Green's functions have been normalized according to the Farrell convention (**?**): displacement $\times\,(a\,\theta)\,10^{12}$, where a is the radius of the Earth in meters and θ is the angular distance from the load in radians.

and

$$v_1 = \frac{a}{m_E} \, l_1' \, \frac{\partial}{\partial \theta} \cos\theta = -\frac{a}{m_E} \, l_1' \, \sin\theta \tag{5.161}$$

for the vertical- and horizontal-displacement components, respectively.

The difference between LGFs computed in the CM and CE frames involves only a degree-one transformation. Thus, (e.g., **?**):

$$u^{\mathrm{CM}} - u^{\mathrm{CE}} = u_1^{\mathrm{CM}} - u_1^{\mathrm{CE}} = \frac{a}{m_E} \cos\theta \left([h_1']_{\mathrm{CM}} - [h_1']_{\mathrm{CE}} \right) = -\frac{a}{m_E} \cos\theta \tag{5.162}$$

for the vertical-displacement component, and

$$v^{\mathrm{CM}} - v^{\mathrm{CE}} = v_1^{\mathrm{CM}} - v_1^{\mathrm{CE}} = -\frac{a}{m_E} \sin\theta \left([l_1']_{\mathrm{CM}} - [l_1']_{\mathrm{CE}} \right) = \frac{a}{m_E} \sin\theta \tag{5.163}$$

for the horizontal-displacement component.

The straightforward conversions between the CE and CM LGFs simplify to (**?**):

$$u^{\mathrm{CM}} = u^{\mathrm{CE}} - \frac{a}{m_E} \cos\theta \tag{5.164}$$

for the vertical-displacement component, and

$$v^{\mathrm{CM}} = v^{\mathrm{CE}} + \frac{a}{m_E} \sin\theta \tag{5.165}$$

for the horizontal-displacement component.

The vertical- and horizontal-displacement LGFs computed in the CE, CM, and CF reference frames for the Preliminary Reference Earth Model (PREM) (**?**) are shown in Fig. 5.1.

5.4 Convolution Methods

SML-induced surface displacements may now be computed by convolving LGFs with a load model. Eqs. 5.109 and 5.110 represent the load-induced displacements per 1-kg load. For an applied load of arbitrary mass, dm, the induced displacements, du and dv, are given by (e.g., **?**):

$$du = \frac{a}{m_E} \sum_{n=0}^{\infty} h'_n \, P_n(\cos\theta) \; dm = G_u(\theta) \; dm \tag{5.166}$$

and

$$dv = \frac{a}{m_E} \sum_{n=1}^{\infty} l'_n \, \frac{\partial P_n(\cos\theta)}{\partial\theta} \; dm = G_v(\theta) \; dm, \tag{5.167}$$

where $G_u(\theta)$ and $G_v(\theta)$ represent the vertical- and horizontal-displacement LGFs, respectively.

For a spatially variable, non-point-source load, the LGFs are convolved with a load model:

$$\overline{U}_j(\overline{r}, S, \rho_{sea}, Z_j) = \int_{\Omega'} G(|\overline{r} - \overline{r}'|, S) \, \rho_{sea}(\overline{r}') \, Z_j(\overline{r}') \, d\Omega'. \tag{5.168}$$

In the context of ocean tidal loading (OTL), $\overline{U}_j$ represents the surface displacement at observation point r due to loading by tidal harmonic j, ρ_{sea} is the density of seawater at the load point r', G represents the displacement LGF, and Z_j is the complex-valued tidal height at the load point r' (e.g., **???????????**). The LGF depends on distance to the load as well as Earth structure, S, which is assumed radially symmetric (e.g., PREM). The mass of the load, dm, at each load point, r', is equivalent to the product of the amplitude, area element for a spherical surface, and density of the load (Eqs. 5.166 and 5.167).

Since OTL is confined to Earth's surface, Eq. 5.168 may be re-expressed as:

$$\overline{U}_j(\Theta, \lambda, \rho_{sea}, Z_j, S) = \int_0^{2\pi} \int_0^{\pi} \rho_{sea}(\Theta', \lambda') \; Z_j(\Theta', \lambda') \; G(\theta, S) \; T(\alpha) \; a^2 \sin\Theta' \; d\Theta' d\lambda', \tag{5.169}$$

where Θ and λ are the co-latitude and longitude of the observation point, respectively; Θ' and λ' are the co-latitude and longitude of the load point; G is the displacement LGF; $T(\alpha)$ is a trigonometric factor that decomposes the horizontal-displacement response into two-component vectors (for the vertical response, $T(\alpha) = 1$); α is the azimuth (measured clockwise from north); Z_j is the complex-valued tide height at the load point; and a is Earth's radius.

As an alternative to the spatial-convolution approach, the load model and the predicted response may be developed in the frequency domain in terms of spherical harmonics (e.g., **??**, Sec. 3.06.4.1). The spherical-harmonic approach can be highly efficient, particularly when seeking the deformation response globally (**??**, Sec. 3.06.4.1). The spatial-convolution method, on the other hand, allows for enhanced accuracy very near to the receiver without requiring a global refinement in the integration mesh (e.g., **???**, Sec. 3.06.4.2). Furthermore, the spatial convolution allows for easy combination of multiple loading models from different grids (e.g., **??**, Sec. 3.06.4.2). We have adopted the spatial-convolution approach in the development presented here; however, with modern computing power and recent advancements in ocean-tide modeling, the spherical-harmonic method might now be a preferred option, and may be implemented in future releases.

5.4.1 Integration Mesh

For practical implementation, the integral in Eq. 5.169 is replaced by a sum over discrete cells. The discrete cells are collectively known as the integration mesh. The integration mesh may be defined in two primary ways (e.g., **?**): templates or gridlines. Templates are formed by subdividing concentric circles about the observation point. Gridlines are formed by subdividing a geographic coordinate system into discrete blocks. Since ocean-tide and other surface-fluid models are commonly distributed in gridline format, and the gridline method is not specific to an observation point, defining the integration mesh in terms of gridlines may seem the natural and obvious choice (e.g., **????**). Defining the integration mesh in terms of templates, however, has distinct advantages for SML analysis (e.g., **????**). First, since the LGFs depend only on the angular distance between the load and receiver, the station-centric coordinate system requires relatively few LGFs to be computed. Second, the integration mesh does not change with the load model and, thus, easily facilitates the combination of multiple load models, even on irregular grids. Third, the singularity in the LGFs at small θ is easily mitigated through appropriate scaling factors and integration over the area of the cells.

A drawback to the template method is that it requires interpolation of the load model onto the integration mesh and, therefore, takes time and does not represent the load model exactly. The ocean model itself, however, is also an approximation of the true load. Moreover, the gridline method also might require some interpolation of the load model (e.g., if a finer integration mesh is adopted near the station). In practice, **?** determined through rigorous testing that the choice of template or gridline method had little effect on predicted OTL-induced surface displacements.

As a result of the straightforward implementation and algorithmic flexibility, we have adopted the template method. Since the template grid is centered on the station, Eq. 5.169 becomes (e.g., **??**):

$$\overline{U}_j(\Theta, \lambda, \rho_{sea}, Z_j, S) = \int_0^{2\pi} \int_0^{\pi} \rho_{sea}(\theta, \alpha)\ Z_j(\theta, \alpha)\ G(\theta, S)\ T(\alpha)\ a^2 \sin\theta\ d\theta\ d\alpha, \tag{5.170}$$

where the load model, Z, and load density, ρ_{sea}, have been interpolated onto the integration mesh. Due to the rapid changes in the LGFs as $\theta \to 0$, the integration mesh should be refined near the station and neighboring coastlines in order to obtain accurate response predictions (e.g., **????**).

5.4.2 Interpolation and Integration of Load Green's Function

One method to alleviate the singularity in the displacement LGFs at $\theta = 0$ is to scale the LGFs by a factor proportional to θ. Following (**?**),

$$G'(\theta) = a^2\ G(\theta)\ 2\ \sin(\theta/2), \tag{5.171}$$

where $G'(\theta)$ is the normalized LGF, $G(\theta)$ is the original LGF, and a is Earth's radius. With the singularity reduced, the normalized LGFs are easily interpolated to intermediary values of θ using, e.g., cubic-spline interpolation. Since tabulated LGFs necessarily contain a greater number of entries at small θ, where the LGFs vary rapidly, interpolation on $\log\theta$ provides a more even spacing (e.g., **?**). When working with a template grid, another option is to integrate the LGFs directly at the LLN-summation stage (e.g., **?**), which has the added benefit of improving the convergence of the series.

Using the normalized LGFs, Eq. 5.170 becomes:

$$\overline{U}_j(\Theta, \lambda, \rho_{sea}, Z_j, S) = \int_0^{2\pi} \int_0^{\pi} \rho_{sea}\ Z(\theta, \alpha)\ G'(\theta)\ T(\alpha)\ \frac{\sin\theta}{2\ \sin(\theta/2)}\ d\theta\ d\alpha. \tag{5.172}$$

The tide height, seawater density, and normalized LGFs are given by the value at the mid-point of each cell. Thus, we rewrite Eq. 5.172 as:

$$\overline{U}_j(\Theta, \lambda, \rho_{sea}, Z_j, S) = \rho_{sea}\ Z(\theta, \alpha)\ G'(\theta) \int_0^{\pi} \frac{\sin\theta}{2\ \sin(\theta/2)}\ d\theta \int_0^{2\pi} T(\alpha)\ d\alpha. \tag{5.173}$$

In this form, the integration may be performed over individual cells and then summed together (**?**):

$$\overline{U}_j(\Theta, \lambda, \rho_{sea}, Z_j, S) = \sum_{i=1}^{N} \rho_i \; Z_i \; G'(\theta_i) \int_{\theta_i - \frac{\delta}{2}}^{\theta_i + \frac{\delta}{2}} \frac{\sin\theta}{2 \; \sin(\theta/2)} \; d\theta \int_{\alpha_i - \frac{\beta}{2}}^{\alpha_i + \frac{\beta}{2}} T(\alpha) \; d\alpha, \tag{5.174}$$

where N is the total number of cells in the integration mesh, Z_i is the tide height at the center of cell i, ρ_i is the seawater density at the center of cell i, $G'(\theta_i)$ is the normalized LGF computed for the center of cell i, θ_i is the angular separation between the station and the center of the load patch, δ_i is the inclination width of the load patch, α_i is the azimuth between the station and center of the load patch (as measured at the station in degrees clockwise from north), and β_i is the azimuthal width of the load patch.

For vertical-displacement response, $T(\alpha) = 1$. For the north component of the horizontal response, $T(\alpha) = -\cos(\alpha)$. For the east component of the horizontal response, $T(\alpha) = -\sin(\alpha)$. The minus signs are required because the horizontal response is directed radially outwards from the load point; thus, the azimuth of the response at the receiver is $\alpha + 180°$, where α is the vector geodesic pointing from the receiver to the load point (e.g., **?**).

The integral over θ reduces to (**?**):

$$\int_{\theta_i - \frac{\delta_i}{2}}^{\theta_i + \frac{\delta_i}{2}} \frac{\sin\theta}{2 \; \sin(\theta/2)} \; d\theta = 4 \; \cos\left(\frac{\theta_i}{2}\right) \; \sin\left(\frac{\delta_i}{4}\right). \tag{5.175}$$

Since θ_i and δ_i are determined solely from the integration mesh, the integrated LGFs may be computed and stored prior to convolution with a load model. The integral over α for the vertical-displacement response reduces to:

$$\int_{\alpha_i - \frac{\beta_i}{2}}^{\alpha_i + \frac{\beta_i}{2}} T(\alpha) \; d\alpha = \int_{\alpha_i - \frac{\beta_i}{2}}^{\alpha_i + \frac{\beta_i}{2}} d\alpha = \beta_i. \tag{5.176}$$

The integral over α for the horizontal-displacement response (north component) reduces to:

$$\begin{aligned}
\int_{\alpha_i - \frac{\beta_i}{2}}^{\alpha_i + \frac{\beta_i}{2}} T(\alpha) \; d\alpha &= -\int_{\alpha_i - \frac{\beta_i}{2}}^{\alpha_i + \frac{\beta_i}{2}} \cos(\alpha) \; d\alpha \\
&= -\left\{\sin\left(\alpha_i + \frac{\beta_i}{2}\right) - \sin\left(\alpha_i - \frac{\beta_i}{2}\right)\right\} \\
&= -\left[\sin(\alpha_i) \; \cos\left(\frac{\beta_i}{2}\right) + \cos(\alpha_i) \; \sin\left(\frac{\beta_i}{2}\right)\right] + \\
&\quad \left[\sin(\alpha_i) \; \cos\left(\frac{\beta_i}{2}\right) - \cos(\alpha_i) \; \sin\left(\frac{\beta_i}{2}\right)\right]
\end{aligned}$$

$$= \quad -2\ \sin\left(\frac{\beta_i}{2}\right)\ \cos(\alpha_i). \tag{5.177}$$

The integral over α for the horizontal-displacement response (east component) reduces to:

$$\begin{aligned}
\int_{\alpha_i-\frac{\beta_i}{2}}^{\alpha_i+\frac{\beta_i}{2}} T(\alpha)\ d\alpha \quad &= \quad -\int_{\alpha_i-\frac{\beta_i}{2}}^{\alpha_i+\frac{\beta_i}{2}} \sin(\alpha)\ d\alpha \\
&= \quad -\left\{\cos\left(\alpha_i - \frac{\beta_i}{2}\right) - \cos\left(\alpha_i + \frac{\beta_i}{2}\right)\right\} \\
&= \quad -\left[\cos(\alpha_i)\ \cos\left(\frac{\beta_i}{2}\right) + \sin(\alpha_i)\ \sin\left(\frac{\beta_i}{2}\right)\right] + \\
&\qquad \left[\cos(\alpha_i)\ \cos\left(\frac{\beta_i}{2}\right) - \sin(\alpha_i)\ \sin\left(\frac{\beta_i}{2}\right)\right] \\
&= \quad -2\ \sin\left(\frac{\beta_i}{2}\right)\ \sin(\alpha_i).
\end{aligned} \tag{5.178}$$

5.4.3 Convolution Procedure

After the integration mesh has been defined and the LGFs have been normalized and integrated, the next step is to determine the geographic coordinates for the midpoint of each cell. From spherical trigonometry, we compute the latitude and longitude of the cell midpoints (Φ_i, λ_i) using the so-called direct geodesic problem (e.g., **?**):

$$\Phi_i \quad = \quad \arcsin(\ \sin\Phi_R\ \cos\theta_i + \cos\Phi_R\ \sin\theta_i\ \cos\alpha_i) \tag{5.179}$$

$$\lambda_i \quad = \quad \lambda_R + \arctan\left(\frac{\sin\alpha_i\ \sin\theta_i\ \cos\Phi_R}{\cos\theta_i - \sin\Phi_R\ \sin\Phi_i}\right), \tag{5.180}$$

where $\Phi_i = 90 - \Theta_i$ is the latitude at the midpoint of mesh cell i, λ_i is the longitude at the midpoint of mesh cell i, Φ_R is the latitude of the station, λ_R is the longitude of the station, θ_i is the inclination angle between the station and the midpoint of mesh cell i, and α_i is the azimuth from the station to the midpoint of mesh cell i (measured at the station in degrees clockwise from north). Note that to account for Earth flattening effects, the geographic coordinates may be converted to geocentric coordinates when computing azimuth and inclination.

The load model, provided on a geographic coordinate grid, may now be interpolated to the specific geographic coordinates at the center of each integration-mesh cell. One option is to use bilinear interpolation of the four neighboring load points (e.g., **??**). Another option, as long as the grid is rectangular, is to use two-dimensional bivariate spline interpolation, which is provided as a built-in function within `scipy`.

The discrete convolution now may be written as (e.g., **??**):

$$\begin{aligned}
\overline{U}_{\text{vert}}(\Theta,\lambda) &= 4\sum_{i=1}^{N} Z_i\,\rho_i\,G'(\theta_i)\,\cos\left(\frac{\theta_i}{2}\right)\,\sin\left(\frac{\delta_i}{4}\right)\,\beta_i &(5.181)\\
\overline{U}_{\text{north}}(\Theta,\lambda) &= 8\sum_{i=1}^{N} Z_i\,\rho_i\,G'(\theta_i)\,\cos\left(\frac{\theta_i}{2}\right)\,\sin\left(\frac{\delta_i}{4}\right)\,\sin\left(\frac{\beta_i}{2}\right)\\
&\quad \cos(\alpha_i+180^\circ) &(5.182)\\
\overline{U}_{\text{east}}(\Theta,\lambda) &= 8\sum_{i=1}^{N} Z_i\,\rho_i\,G'(\theta_i)\,\cos\left(\frac{\theta_i}{2}\right)\,\sin\left(\frac{\delta_i}{4}\right)\,\sin\left(\frac{\beta_i}{2}\right)\\
&\quad \sin(\alpha_i+180^\circ). &(5.183)
\end{aligned}$$

For complex-valued loads, including the ocean tides, the amplitudes and phases are partitioned into real and imaginary components (e.g., **?**):

$$Z_i = A_i\ \cos\phi_i + i\ A_i\ \sin\phi_i = c + i\,s, \tag{5.184}$$

where A_i is the amplitude of the load at the center of mesh cell i and ϕ_i is the phase at the center of mesh cell i. The real and imaginary components are convolved separately over all cells in the integration mesh using Eqs. 5.181–5.183. The convolution results for each harmonic coefficient, U_c and U_s, are then recombined into the amplitude and phase for each spatial component:

$$\begin{aligned}
A &= \sqrt{U_c^2+U_s^2} &(5.185)\\
\phi &= \operatorname{atan2}(U_s,U_c). &(5.186)
\end{aligned}$$

In addition to a spatially variable load model, a spatially variable model for seawater densities may also be included (e.g., **???**). As with the load model, the densities would be interpolated onto the integration mesh, designed such that the average density within each cell is approximately equal to the value at the midpoint of each cell. Alternatively, the seawater density may be approximated as constant everywhere and applied after the convolution. The approximation of constant seawater density is good to about 1% (e.g., **?**).

5.4.4 Additional Considerations

To improve the convolution further, the load model may be refined around coastal boundaries using a land-sea mask and bilinear interpolation (e.g., **?**). Recent ocean tide models, such as TPXO8-Atlas and FES2014, do a much better job of fitting the coastline than former-generation models (e.g., **?**); thus, refining the ocean tide models around the coastline has become less critical, but still influential.

It is also worth noting that the development presented here applies to an elastic Earth. For a viscoelastic Earth, the elastic moduli are frequency dependent and the load Love numbers become complex-valued (e.g., **???**). **?** reported differences between OTL-induced surface displacements of up to 1.5% in amplitude and 0.3° in phase when comparing elastic and anelastic models. More recently, **?** found that discrepancies between observed and predicted OTL-induced surface displacements in western Europe could be reduced by about 0.2 mm on average by accounting for mantle anelasticity.