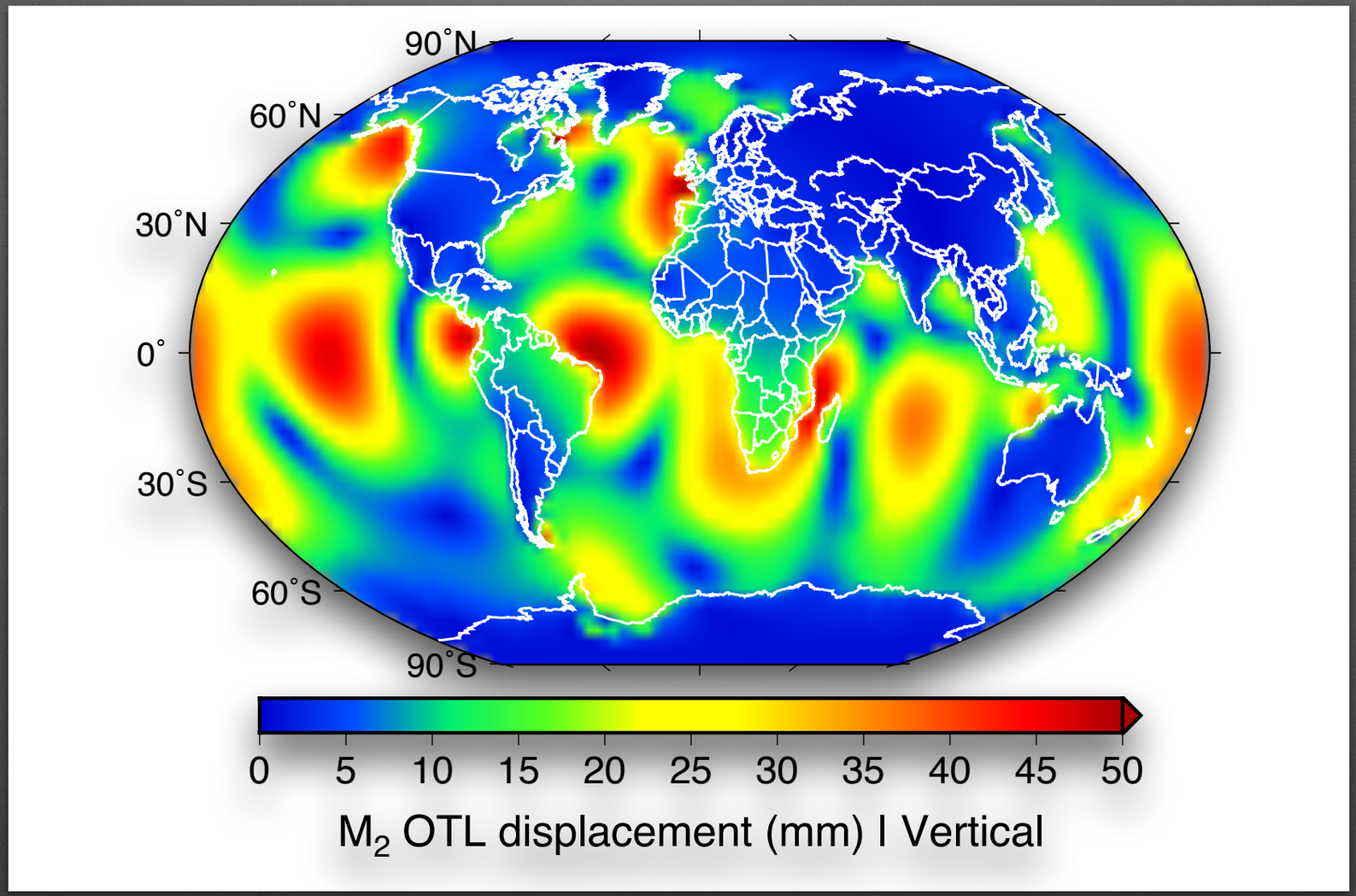


LoadDef Tutorial

Hilary Martens

July 2022

Setup Anaconda Environment

```
conda config --prepend channels conda-forge

conda config --set channel_priority strict

conda create -n LDEF python=3.10 mpi4py scipy netCDF4 pyshp shapely numba jupyter pygmt
```

Getting started

- Download *LoadDef* software from GitHub
  - https://github.com/hrmartens/LoadDef
  - Look for the green-colored button that says “Code”
  - Option 1: Download and unpack the zip file
  - Option 2: Copy the *https* link. Open a terminal. Type: git clone [link]
- Activate the conda environment:
  - >> conda activate LDEF
- Launch a Jupyter Notebook:
  - >> jupyter notebook

LOVE NUMBERS

★ *h*, *l*, *k* ★

★ Potential, Load, and Shear ★

Main program: run_ln.py
Basic input: SNREI planetary-structure model

LOAD GREEN'S FUNCTIONS

★ Vertical and Horizontal Displacement, Gravity, Tilt, and Strain ★

Main program: run_gf.py
Basic input: Load Love numbers

LOADDEF

PARTIAL DERIVATIVES OF LOVE NUMBERS

★ *h*, *l*, *k* ★

★ Potential, Load, and Shear ★

Main program: run_pl.py
Basic input: SNREI planetary-structure model

LOAD-INDUCED SURFACE DISPLACEMENTS

★ East, North, Up ★

Main program: run_cn.py
Basic inputs: (1) Load Green's functions; (2) Surface-load distribution

a M_2 OTL Displacement | CM Frame

b O_1 OTL Displacement | CM Frame

c M_f OTL Displacement | CM Frame

50˚N
45˚N
40˚N
35˚N
30˚N

10 x 2 mm

5 x 1 mm

1.0 x 0.2 mm

125˚W 120˚W 115˚W 110˚W

0 5 10 15 20 25 30

Vertical displacement (mm)

0 3 6 9 12 15

Vertical displacement (mm)

0.00 0.25 0.50 0.75 1.00

Vertical displacement (mm)

Computing Love Numbers

- Option A — From Jupyter Notebook:
    - Navigate to the web browser
    - Launch “working/run_ln.ipynb”
        - Click “Run”
- Option B — From command line:
    - >> cd working
    - >> mpirun -np 1 python run_ln.py

Plotting Love Numbers

- Option A — From Jupyter Notebook:
  - Launch “utility/plots/plot_ln.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../utility/plots
  - >> python plot_ln.py

Load Love Numbers

Love Number (dimensionless)

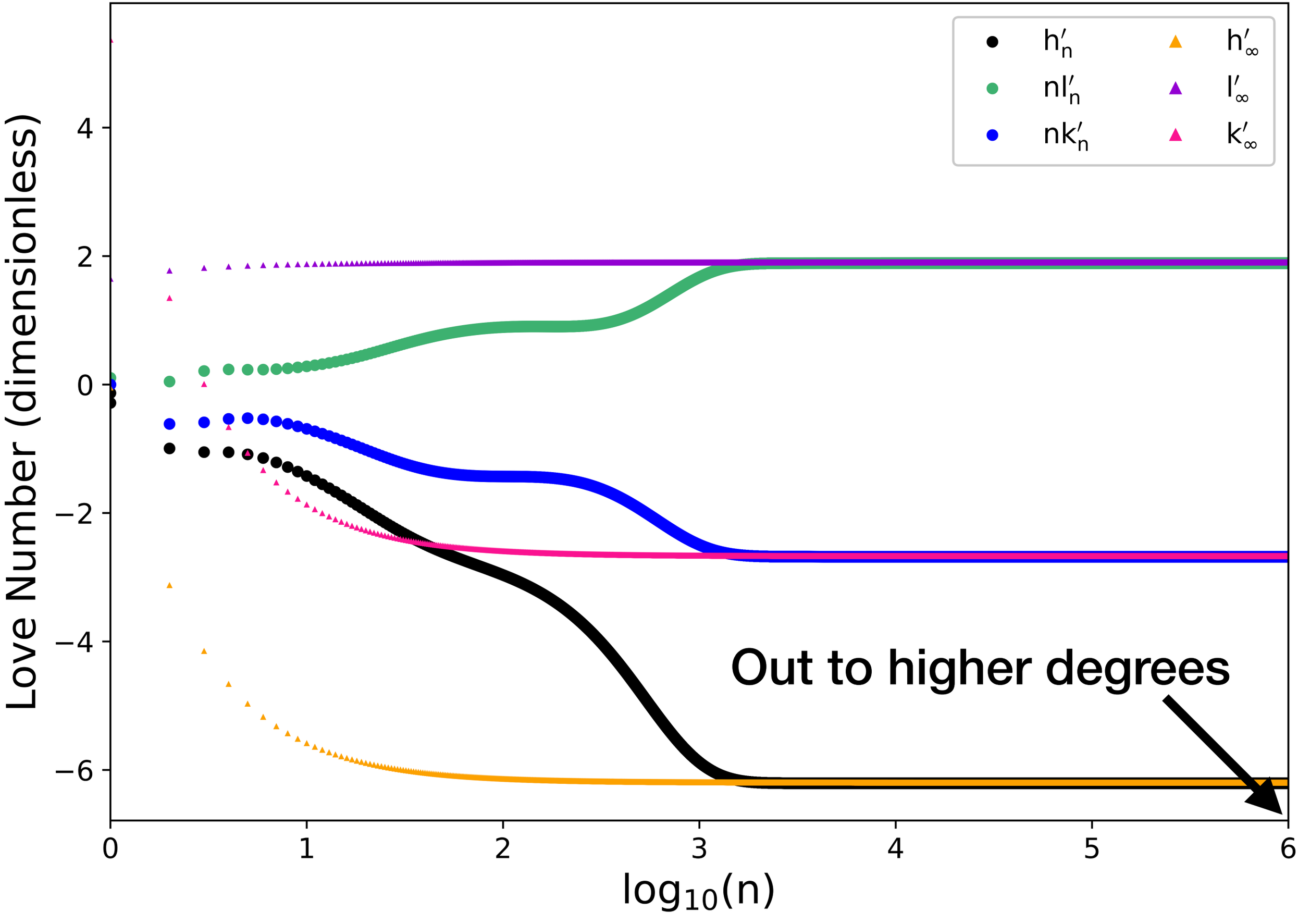
4
2
0
−2
−4
−6

h'_n
nl'_n
nk'_n
h'_∞
l'_∞
k'_∞

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

$\log_{10}(n)$

Load Love Numbers



Computing Green's Functions

- Option A — From Jupyter Notebook:
  - Launch "working/run_gf.ipynb"
  - Click "Run"
- Option B — From command line:
  - >> cd ../../working
  - >> mpirun -np 1 python run_gf.py

Plotting Green's Functions

- Option A — From Jupyter Notebook:
  - Launch "utility/plots/plot_gf.ipynb"
  - Click "Run"
- Option B — From command line:
  - >> cd ../utility/plots
  - >> python plot_gf.py

Horizontal | CM

Horizontal | CE

Horizontal | CF

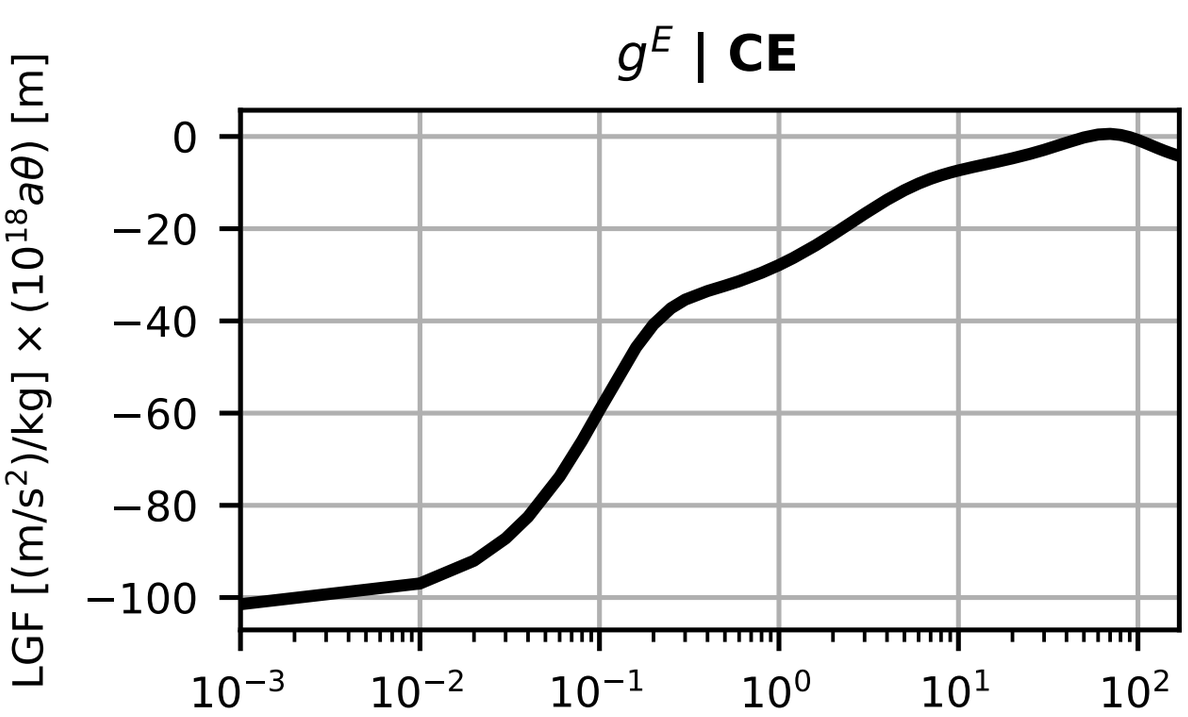
Vertical | CM

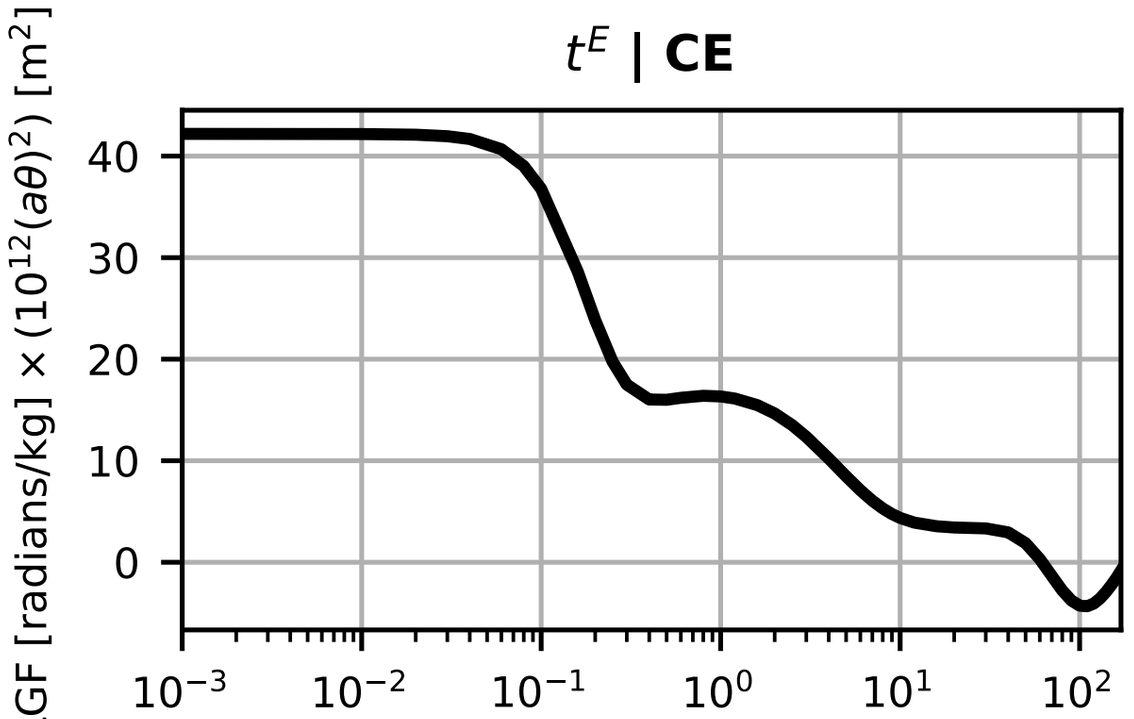
Vertical | CE

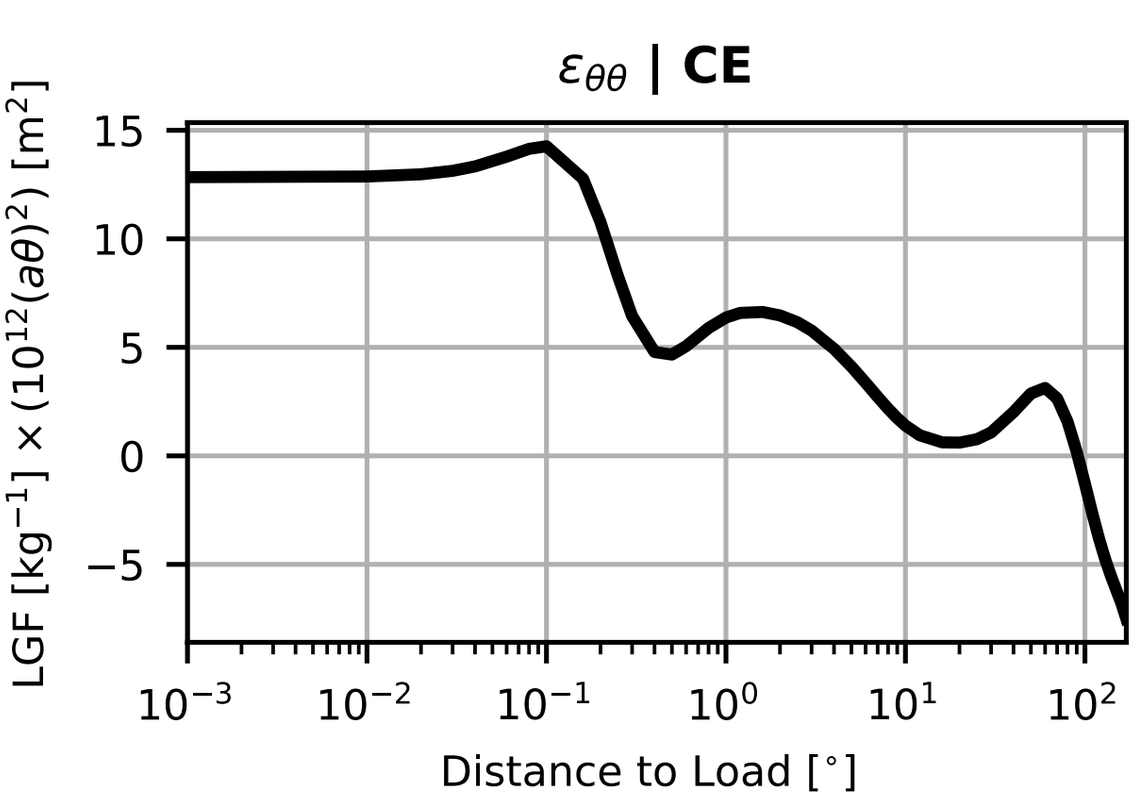
Vertical | CF

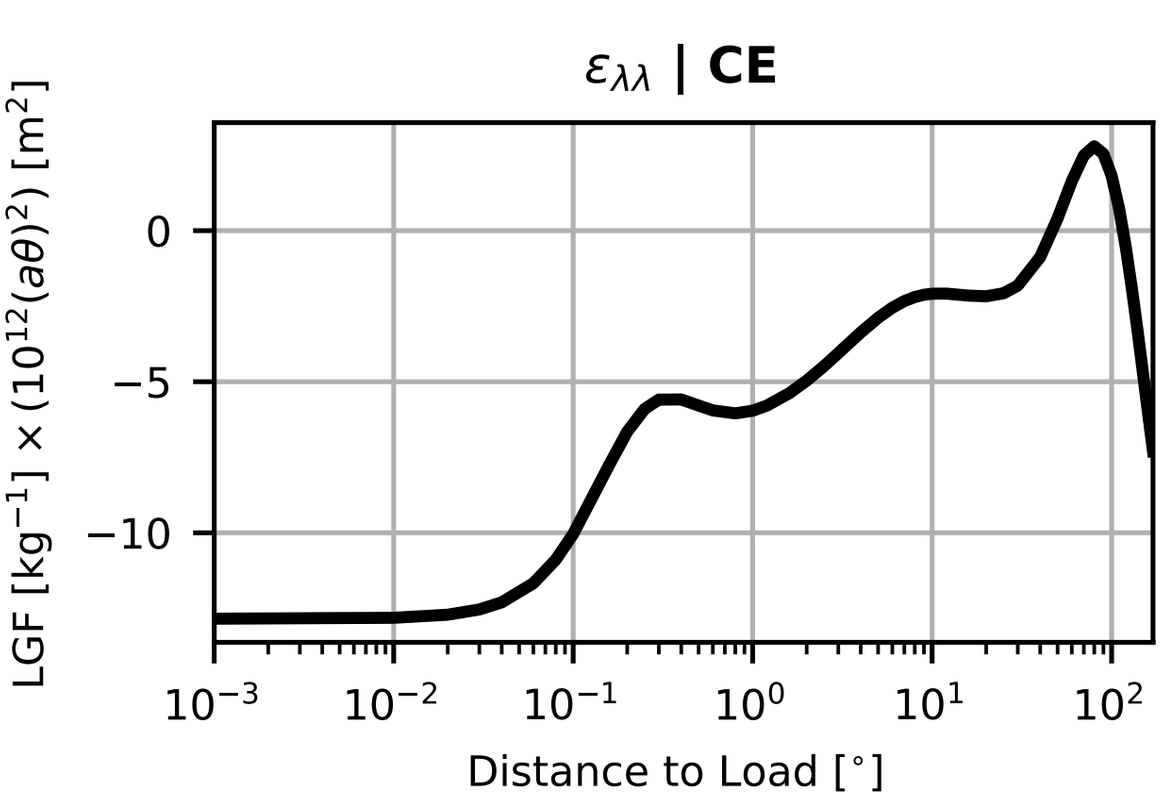
LGF [m/kg] × ($10^{12} a\theta$) [m]

Distance to Load [°]


g^E | **CE**
LGF [(m/s^2)/kg] × ($10^{18}a\theta$) [m]
0
−20
−40
−60
−80
−100
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}


t^E | **CE**
LGF [radians/kg] × ($10^{12}(a\theta)^2$) [m^2]
40
30
20
10
0
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}


$\varepsilon_{\theta\theta}$ | **CE**
LGF [kg^{-1}] × ($10^{12}(a\theta)^2$) [m^2]
15
10
5
0
−5
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Distance to Load [°]

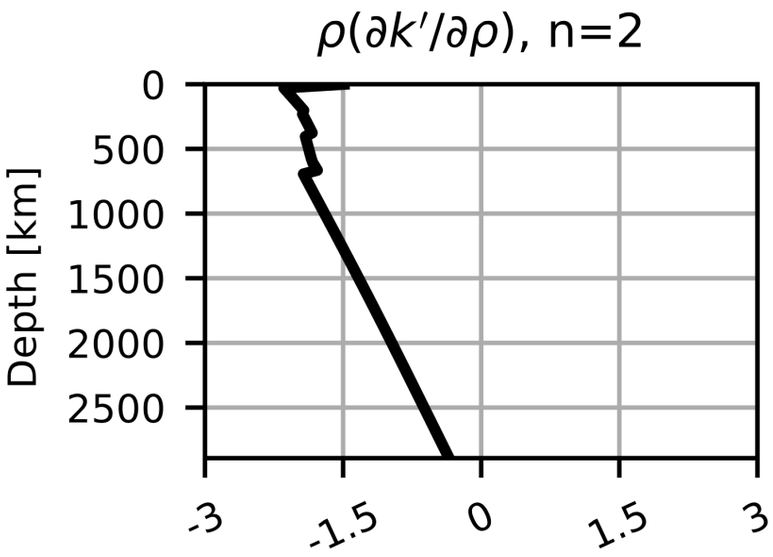

$\varepsilon_{\lambda\lambda}$ | **CE**
LGF [kg^{-1}] × ($10^{12}(a\theta)^2$) [m^2]
0
−5
−10
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Distance to Load [°]

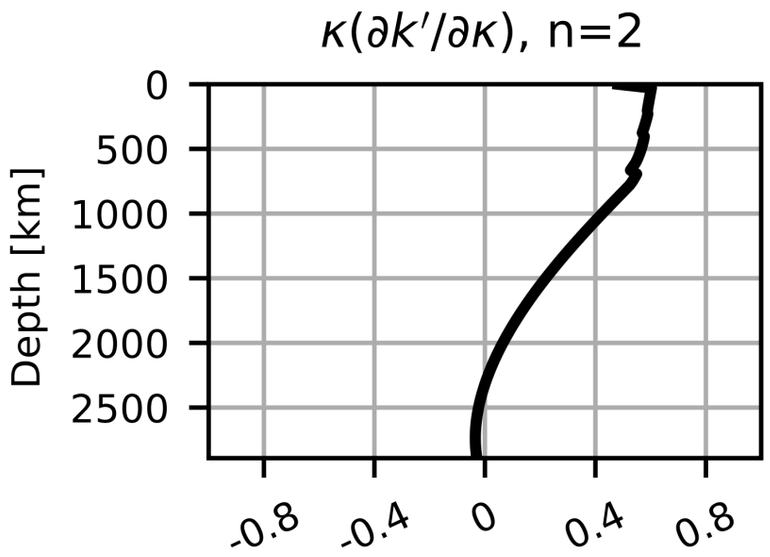
Computing Partial Derivatives of Love Numbers

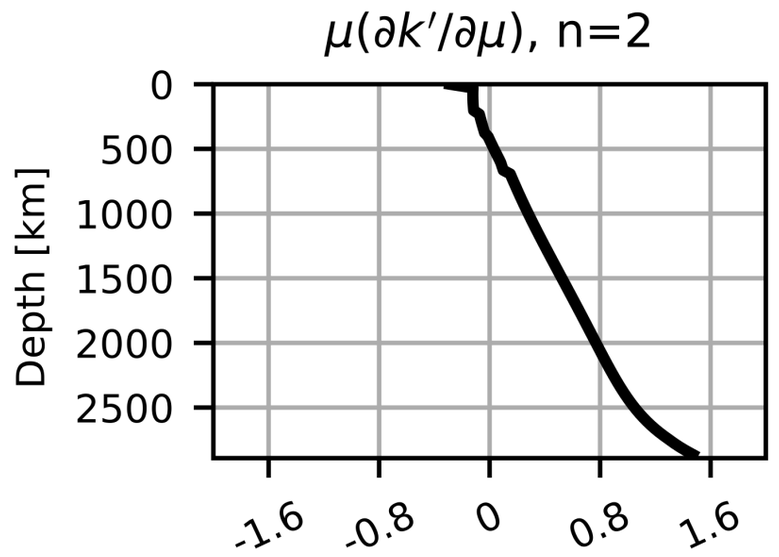
- Option A — From Jupyter Notebook:
    - Launch “working/run_pl.ipynb”
    - Click “Run”
- Option B — From command line:
    - >> cd ../../working
    - >> mpirun -np 1 python run_pl.py

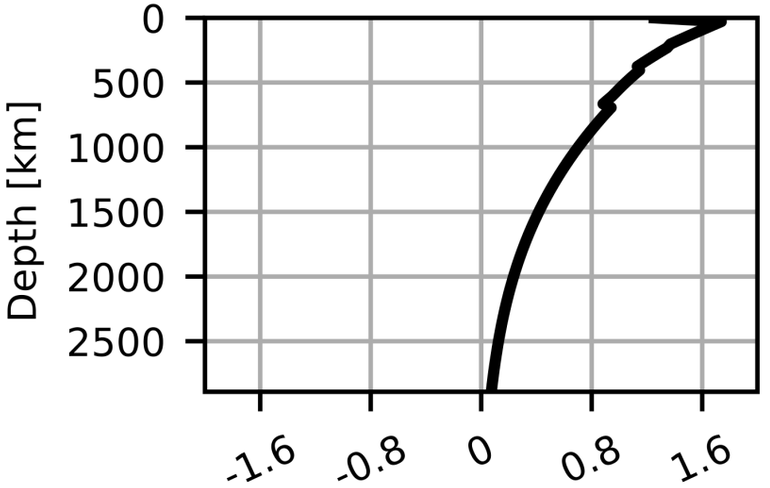
Plotting Partial Derivatives of Love Numbers

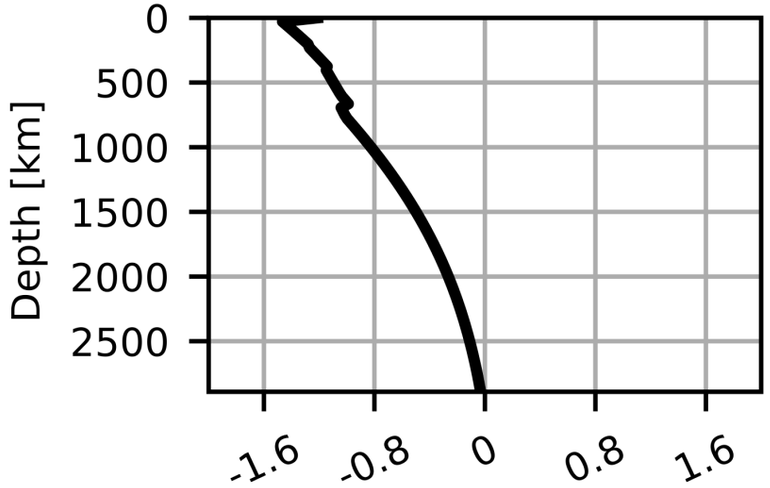
- Option A — From Jupyter Notebook:
  - Launch “utility/plots/plot_pl.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../utility/plots
  - >> python plot_pl.py

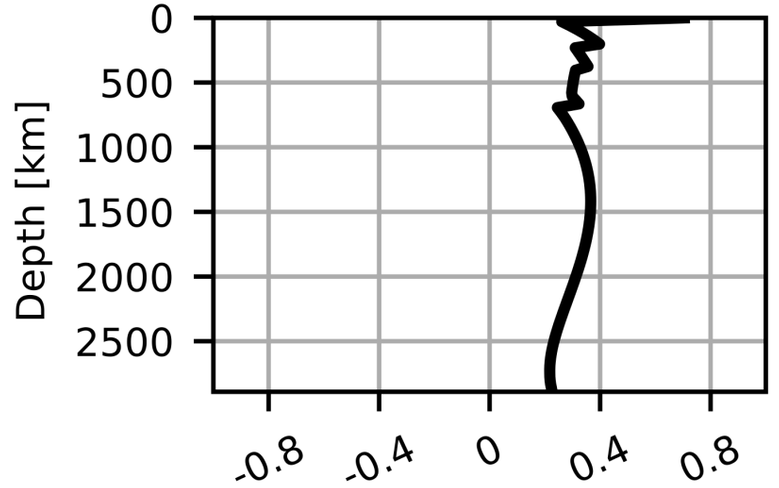

$\rho(\partial k'/\partial\rho)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-3
-1.5
0
1.5
3

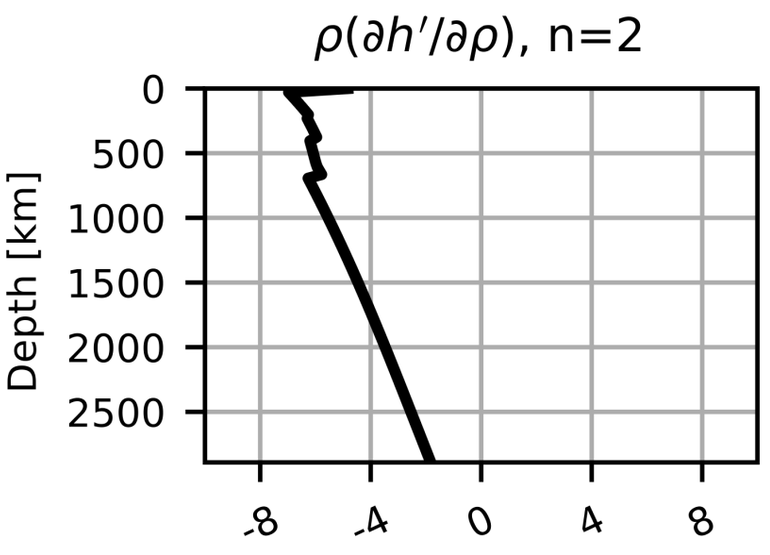

$\kappa(\partial k'/\partial\kappa)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-0.8
-0.4
0
0.4
0.8

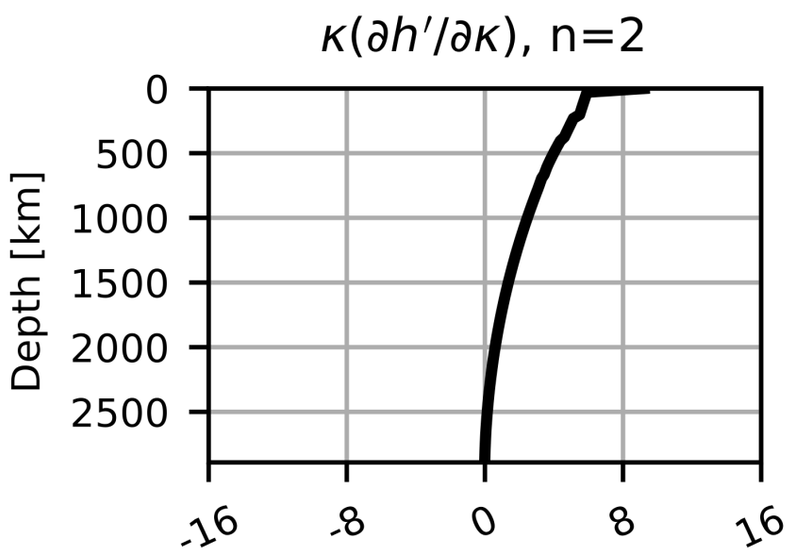

$\mu(\partial k'/\partial\mu)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-1.6
-0.8
0
0.8
1.6

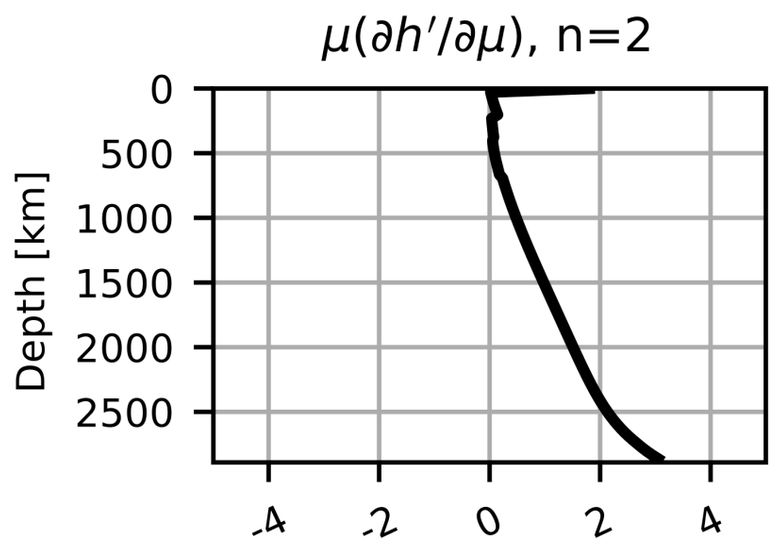

$\rho(\partial l'/\partial\rho)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-1.6
-0.8
0
0.8
1.6


$\kappa(\partial l'/\partial\kappa)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-1.6
-0.8
0
0.8
1.6


$\mu(\partial l'/\partial\mu)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-0.8
-0.4
0
0.4
0.8


$\rho(\partial h'/\partial\rho)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-8
-4
0
4
8


$\kappa(\partial h'/\partial\kappa)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-16
-8
0
8
16


$\mu(\partial h'/\partial\mu)$, n=2
Depth [km]
0
500
1000
1500
2000
2500
-4
-2
0
2
4

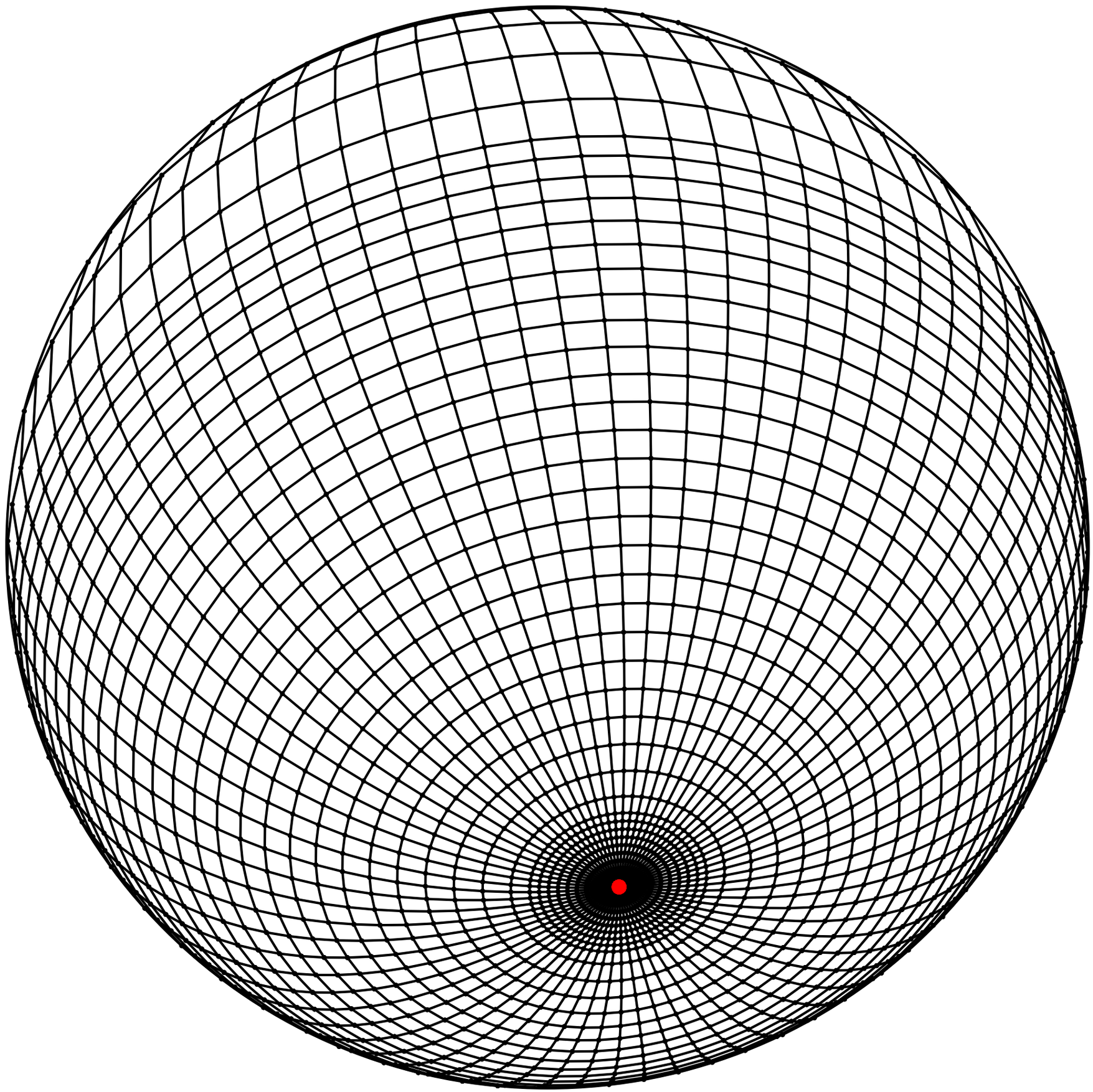
Predicted Load-Induced Displacements

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

Displacement — Displacement/Force — Force

- Response at observation point *r*
- Load density at load point *r'*
- Load Green's function
- Load height at load point *r'*
- Spherically symmetric structure
- Integrate over Earth's surface

To generate the forward model, both the load model (*Zρ)* and the load Green's functions (*G*) must be given



Copy external load files into appropriate directories

- IMPORTANT: The GRACE and tide models cannot be shared beyond this class. If you want to use them in your research, you must register for access to each data product.

- Place the GRACE data file into:
    - /input/Load_Models/GRACE-Tellus-RL06/
- Place the got410c data files into:
    - /input/Load_Models/GOT410c/

Three Example Applications

- Hydrological loading from GRACE mascons
- Disk loading
- Ocean tidal loading

Example 1: GRACE

- Example 1: Compute time series of displacements caused by global hydrological loading at the locations of GPS stations
    - Use data from GRACE mason solutions
        - Convolve with load Green's functions for PREM
- Step 1: Compute Love numbers and Green's functions for PREM.
- Step 2: Create a load grid for GRACE mascons.
- Step 3: Run the convolution.

Generating the Load Grids

- Option A — From Jupyter Notebook:
  - Launch “GRDGEN/load_files/gen_grace_tellus_rl06.ipynb”
    - Click “Run”
- Option B — From command line:
  - >> cd ../../GRDGEN/load_files
  - >> python gen_grace_tellus_rl06.py

Run the Convolution

- Option A — From Jupyter Notebook:
  - Launch “working/run_cn_grace.ipynb”
    - Click “Run”
- Option B — From command line:
  - >> cd ../../working
  - >> mpirun -np 1 python run_cn_grace.py

Plot the Results

- Option A — From Jupyter Notebook:
  - Launch “utility/plots/plot_ts.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../utility/plots
  - >> python plot_ts.py

Station : P144

East (mm)

0.5
0.0
−0.5

North (mm)

0.5
0.0
−0.5

Up (mm)

5
0

2019-10
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04
2021-07
2021-10

Date

Where are the Stations?

- https://www.unavco.org/instrumentation/networks/map/map.html#!/

- http://geodesy.unr.edu/NGLStationPages/GlobalStationList

- http://geodesy.unr.edu/NGLStationPages/gpsnetmap/GPSNetMap.html

Example 2a: Disks
(Analytical disk-factor approach)

- Example 2: Compute displacements caused by disk loads of two different sizes
    - Use disk-factor formulation from Farrell (1972)
        - No convolution needed
- Step 1: Compute Love numbers
- Step 2: Compute integrated Green's functions for the appropriate area of a disk.
- Step 3: Repeat Step 2 for disks of other sizes as desired.

Computing the Integrated Green's Functions

- Option A — From Jupyter Notebook:
    - Launch "working/run_an_disk.ipynb"
    - Click "Run"
- Option B — From command line:
    - >> cd ../../working
    - >> python run_an_disk.py
- Modify the input: set "disk_radius" to 25 km. Re-run the program.

Plotting the Results

- Option A — From Jupyter Notebook:
    - Launch “utility/plots/plot_dk_an.ipynb”
    - Click “Run”
- Option B — From command line:
    - >> cd ../plots/
    - >> python plot_dk_an.py

Disks | PREM | 1 m Fresh Water

Lateral Displ. (mm)

Up Displ. (mm)

Angular Distance from Load Center (deg)

10 km
25 km

Example 2b: Disks
(Convolution formulation)

- Example 2: Compute displacements caused by disk loads of two sizes
    - Create disk loads
    - Create a common mesh (optional, but faster)
        - Convolve with load Green's functions for PREM
- Step 1: Compute Love numbers and Green's functions for PREM.
- Step 2: Create a load grid for disks with various radii.
- Step 3: Create a common mesh to use for the convolution.
- Step 4: Run the convolution.

Generating the Load Grids

- Option A — From Jupyter Notebook:
  - Launch “GRDGEN/load_files/gen_disk.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../../GRDGEN/load_files
  - >> python gen_disk.py
- Repeat for a disk of radius 25 km

Generating a Common Mesh

- Option A — From Jupyter Notebook:
    - Launch “GRDGEN/common_mesh/disk_mesh.ipynb”
        - Click “Run”
- Option B — From command line:
    - >> cd ../../GRDGEN/common_mesh
    - >> python disk_mesh.py

Running the Convolution

- Option A — From Jupyter Notebook:
  - Launch “working/run_cn_disk.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../../working/
  - >> mpirun -np 2 python run_cn_disk.py

Combining the Stations

- Option A — From Jupyter Notebook:
  - Launch “utility/pmes/run_combine_stations.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../utility/pmes/
  - >> python run_combine_stations.py

Plotting the Results

- Option A — From Jupyter Notebook:
  - Launch “utility/plots/plot_dk.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../plots/
  - >> python plot_dk.py

Disks | PREM | 1 m Fresh Water

Lateral Displ. (mm)

1.0
0.8
0.6
0.4
0.2
0.0

10 km
25 km

Up Displ. (mm)

2
0
−2
−4

0.0
0.1
0.2
0.3
0.4
0.5

Angular Distance from Load Center (deg)

Disks | PREM | 1 m Fresh Water

Lateral Displacement (mm)

0
1
2
3
4

More disks!

1 km
2 km
5 km
10 km
20 km
30 km
50 km
100 km
200 km

Up Displacement (mm)

0
−5
−10
−15
−20

0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Angular Distance from Load Center (deg)

Example 3: Load Tides

- Example 3: Compute surface displacements caused by ocean tidal loading
    - Generate a load grid for an ocean-tide model
        - Convolve with load Green's functions for PREM
- Step 1: Compute Love numbers and Green's functions for PREM.
- Step 2: Create a load grid for an ocean-tide model.
- Step 3: Run the convolution.

Generating the Load Grids

- Option A — From Jupyter Notebook:
    - Launch “GRDGEN/load_files/gen_otl.ipynb”
    - Click “Run”
- Option B — From command line:
    - >> cd ../../GRDGEN/load_files
    - >> python gen_otl.py
- Optionally repeat for O1 tide
    - Note: Must change “loadfile” and “harmonic”

Running the Convolution

- Option A — From Jupyter Notebook:
  - Launch “working/run_cn_otl.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../../working/
  - >> mpirun -np 1 python run_cn_otl.py

Combining the Harmonics

- Option A — From Jupyter Notebook:
  - Launch “utility/pmes/run_combine_harmonics.ipynb”
    - Click “Run”
- Option B — From command line:
  - >> cd ../utility/pmes/
  - >> python run_combine_harmonics.py

Check out the Results

- Option A — From Jupyter Notebook:
  - Navigate into the “utility/pmes/output” folder
  - Open the file called: cn_OceanOnly_M2_cm_convgf_GOT410c_PREM.txt
- Option B — From command line:
  - >> cd ./output
  - >> more cn_OceanOnly_M2_cm_convgf_GOT410c_PREM.txt

Convert to Particle Motion Ellipses

- Option A — From Jupyter Notebook:
  - Launch “utility/pmes/run_env2pme.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ..
  - >> python run_env2pme.py

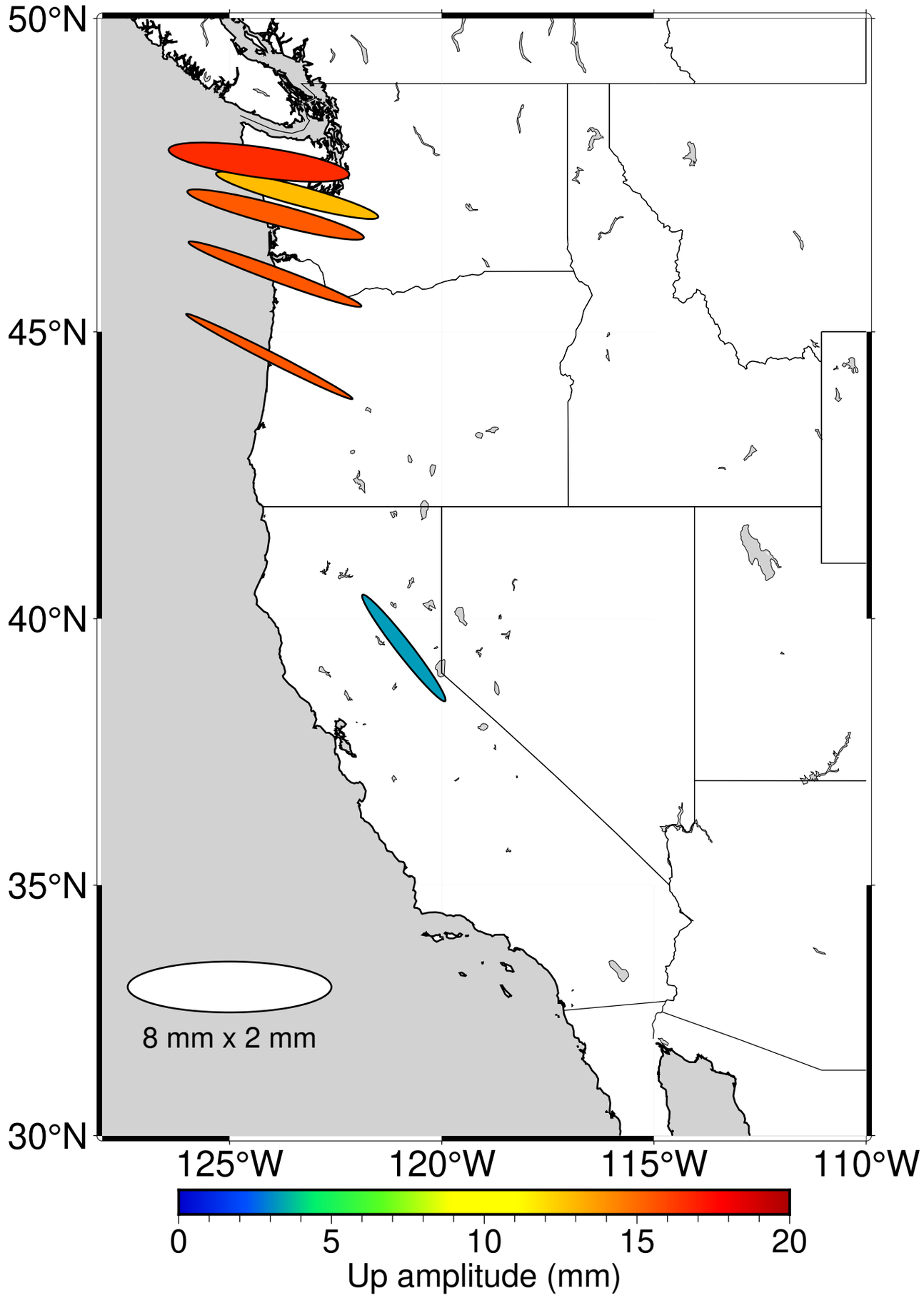
Convert to Particle Motion Ellipses: Repeat for O1 Tide

- Option A — From Jupyter Notebook:
  - Update “harmonic” in “run_env2pme.ipynb”
  - Click “Run”
- Option B — From command line:
  - Update “harmonic” in “run_env2pme.ipynb”
  - >> python run_env2pme.py

Plot the Particle Motion Ellipses

- Option A — From Jupyter Notebook:
  - Launch “utility/plots/plot_el.ipynb”
  - Click “Run”
- Option B — From command line:
  - >> cd ../plots
  - >> python plot_el.py

M_2 Ocean Tidal Loading



Theory

Equations of motion for spheroidal oscillations

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

$$\nabla^2\psi_1 = -4\pi G\rho_1$$

$$\frac{dy}{dr} = Ay$$

System of six first-order differential equations

radial displacement and stress

$$\begin{aligned}
\dot{y}_1 &= \frac{-2\lambda}{\lambda+2\mu}\frac{y_1}{r} + \frac{y_2}{\lambda+2\mu} + \frac{\lambda\, n(n+1)}{\lambda+2\mu}\frac{y_3}{r},\\
\dot{y}_2 &= \left[-\omega^2\rho r^2 - 4\rho g r + \frac{4\mu(3\lambda+2\mu)}{\lambda+2\mu}\right]\frac{y_1}{r^2} - \frac{4\mu}{\lambda+2\mu}\frac{y_2}{r}\\
&+ \left[n(n+1)\rho g r - \frac{2\mu(3\lambda+2\mu)\,n(n+1)}{\lambda+2\mu}\right]\frac{y_3}{r^2}\\
&+ n(n+1)\frac{y_4}{r} - \rho y_6,\\
\dot{y}_3 &= -\frac{y_1}{r} + \frac{y_3}{r} + \frac{y_4}{\mu},\\
\dot{y}_4 &= \left[g\rho r - \frac{2\mu(3\lambda+2\mu)}{\lambda+2\mu}\right]\frac{y_1}{r^2} - \frac{\lambda}{\lambda+2\mu}\frac{y_2}{r}\\
&+ \left[-\omega^2\rho r^2 + \frac{2\mu}{\lambda+2\mu}[\lambda(2n^2+2n-1) + 2\mu(n^2+n-1)]\right]\frac{y_3}{r^2}\\
&- \frac{3y_4}{r} - \rho\frac{y_5}{r},\\
\dot{y}_5 &= 4\pi G\rho y_1 + y_6,\\
\dot{y}_6 &= -4\pi G\rho\, n(n+1)\frac{y_3}{r} + n(n+1)\frac{y_5}{r^2} - \frac{2y_6}{r},
\end{aligned}$$

- Function of elastic and density structure
- Function of spherical harmonic degree

tangential displacement and stress

gravitational potential

Equations of motion for spheroidal oscillations

$$\frac{dy}{dr} = Ay$$

radial displacement and stress

tangential displacement and stress

gravitational potential

$$\begin{aligned}
\dot{y}_1 &= \frac{-2\lambda}{\lambda+2\mu}\frac{y_1}{r} + \frac{y_2}{\lambda+2\mu} + \frac{\lambda\, n(n+1)}{\lambda+2\mu}\frac{y_3}{r}, \\
\dot{y}_2 &= \left[-\omega^2\rho r^2 - 4\rho g r + \frac{4\mu(3\lambda+2\mu)}{\lambda+2\mu}\right]\frac{y_1}{r^2} - \frac{4\mu}{\lambda+2\mu}\frac{y_2}{r} \\
&+ \left[n(n+1)\rho g r - \frac{2\mu(3\lambda+2\mu)\, n(n+1)}{\lambda+2\mu}\right]\frac{y_3}{r^2} \\
&+ n(n+1)\frac{y_4}{r} - \rho y_6, \\
\dot{y}_3 &= -\frac{y_1}{r} + \frac{y_3}{r} + \frac{y_4}{\mu}, \\
\dot{y}_4 &= \left[g\rho r - \frac{2\mu(3\lambda+2\mu)}{\lambda+2\mu}\right]\frac{y_1}{r^2} - \frac{\lambda}{\lambda+2\mu}\frac{y_2}{r} \\
&+ \left[-\omega^2\rho r^2 + \frac{2\mu}{\lambda+2\mu}[\lambda(2n^2+2n-1) + 2\mu(n^2+n-1)]\right]\frac{y_3}{r^2} \\
&- \frac{3y_4}{r} - \rho\frac{y_5}{r}, \\
\dot{y}_5 &= 4\pi G\rho y_1 + y_6, \\
\dot{y}_6 &= -4\pi G\rho\, n(n+1)\frac{y_3}{r} + n(n+1)\frac{y_5}{r^2} - \frac{2y_6}{r},
\end{aligned}$$

Same equations used to compute normal-mode oscillations and solid Earth body tides

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

$$\begin{aligned} u_n &= U(r)\,P_n(\cos\theta)\,e^{i\omega t} \\ v_n &= V(r)\,\frac{\partial P_n(\cos\theta)}{\partial\theta}\,e^{i\omega t} \\ \psi_n &= P(r)\,P_n(\cos\theta)\,e^{i\omega t}, \end{aligned}$$

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

Surface Boundary Conditions					
	Free Oscillations	External Potential	Surface Mass Loading	Surface Shear Forcing	Surface Stress (n=1)
y_2	0	0	$-g_S^2\,\frac{2n+1}{4\pi G}$	0	$-\frac{3\,g_S^2}{4\pi G}$
y_4	0	0	0	$\frac{(2n+1)\,g_S^2}{4\pi G\,n\,(n+1)}$	$\frac{3\,g_S^2}{8\pi G}$
$y_6 + \frac{n+1}{a}\,y_5$	0	$(2n+1)\,g_S$	$(2n+1)\,g_S$	0	0

Love Number Development

Same as free-oscillation calculation

- We need to solve the equations of motion (i.e. *conservation of linear momentum*) for the spheroidal deformation of an elastic, self-gravitating and spherically symmetric Earth
- To eliminate derivatives in the elastic parameters, we reduce the equations of motion to a system of six first-order ODEs
- Compute three linearly independent starting solutions based on the analytical solutions for a homogeneous sphere
- Integrate the three solutions through the spherically symmetric Earth model, applying boundary conditions at all internal interfaces
- Apply stress and gravitational-potential boundary conditions at the surface to derive the remaining solutions to the equations of motion, or Love numbers, for each spherical harmonic degree

Love Number Development

- We need to solve the equations of motion (i.e. *conservation of linear momentum*) for the spheroidal deformation of an elastic, self-gravitating and spherically symmetric Earth

- To eliminate derivatives in the elastic parameters, we reduce the equations of motion to a system of six first-order ODEs

- Compute three linearly independent starting solutions based on the analytical solutions for a homogeneous sphere

- Integrate the three solutions through the spherically symmetric Earth model, applying boundary conditions at all internal interfaces

- Apply stress and gravitational-potential boundary conditions at the surface to derive the remaining solutions to the equations of motion, or Love numbers, for each spherical harmonic degree

Green's Function Development

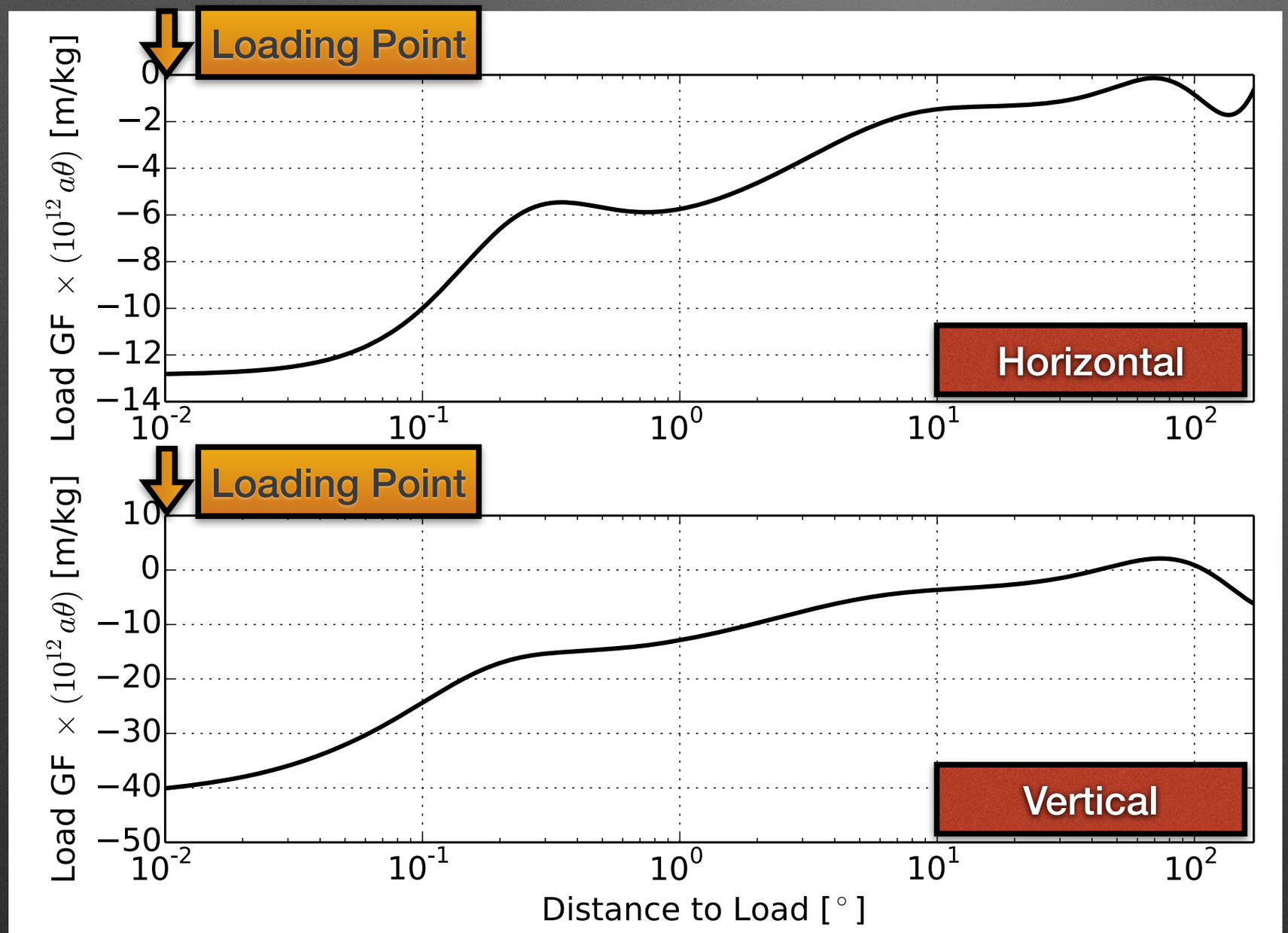
Same as free-oscillation calculation

- Develop the equations of motion (*conservation of momentum*) for the spheroidal deformation of an elastic, self-gravitating and hydrostatically pre-stressed body $\rho\,\overline{a} = \overline{\nabla} \cdot \overline{\overline{\sigma}} + \rho\,\overline{F}$
- Reduce to a system of six first-order ODEs that depend on the elastic moduli, density, spherical harmonic degree, and radius
- Compute analytical starting solutions for a homogeneous sphere
- Integrate through a spherically symmetric Earth model (including fluid layers) to the surface
- Apply boundary conditions at the surface to derive solutions to the equations of motion, and Love numbers for each spherical harmonic degree

$$u_n = U_n P_n(\cos\theta) \qquad v_n = V_n \frac{\partial P_n(\cos\theta)}{\partial\theta}$$

Load Green's Functions

Characterize the deformation response to a delta-function unit normal force



Horizontal Displacement

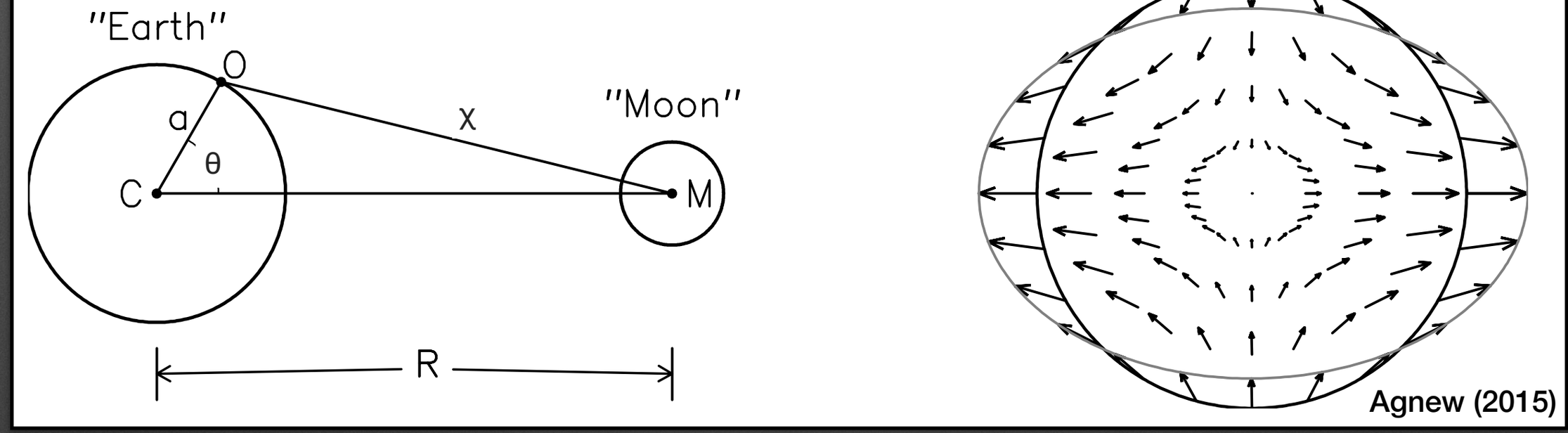
$$v = \frac{a}{m_E} \sum_{n=1}^{\infty} l'_n \frac{\partial P_n(\cos\theta)}{\partial\theta}$$

Vertical Displacement

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

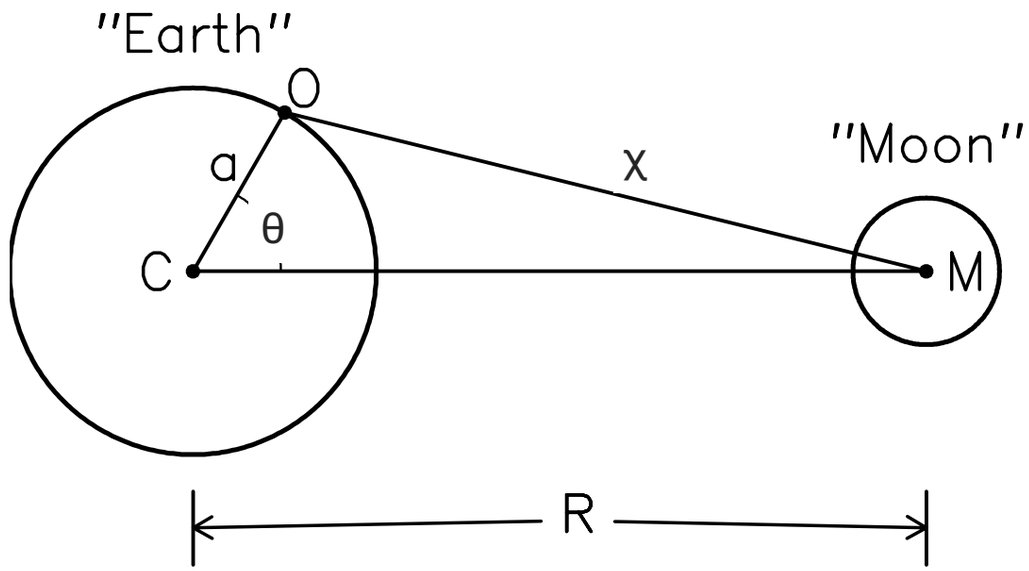
Tides

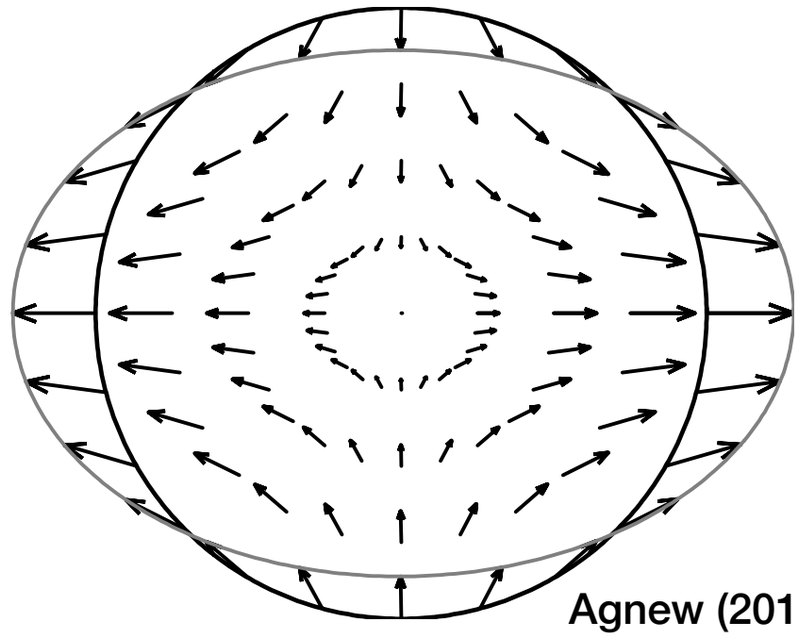
Tidal Potential



Tides arise due to differential, or unbalanced, gravitational forces within the oceans and the solid Earth

Tidal Potential



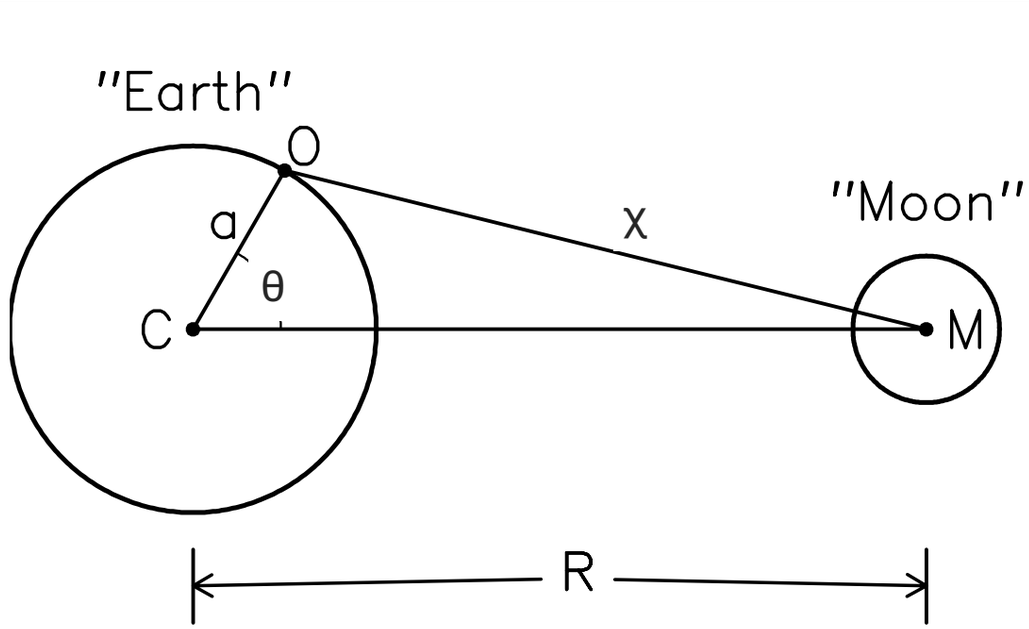


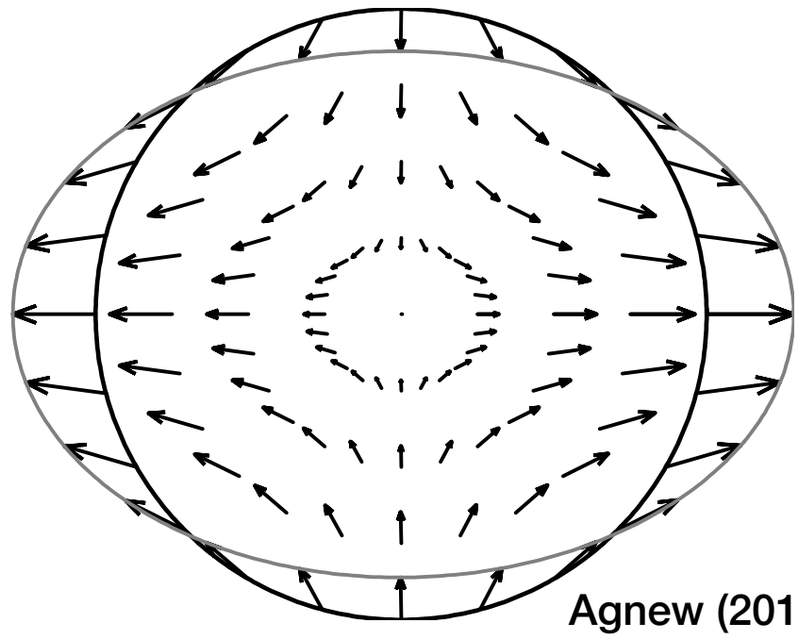
Agnew (2015)

Tides arise due to differential, or unbalanced, gravitational forces within the oceans and the solid Earth

$$V_O = \frac{GM_{\text{ext}}}{\chi} = \frac{GM_{\text{ext}}}{R} \sum_{n=0}^{\infty} \left(\frac{a}{R}\right)^n P_n(\cos\theta)$$

Tidal Potential





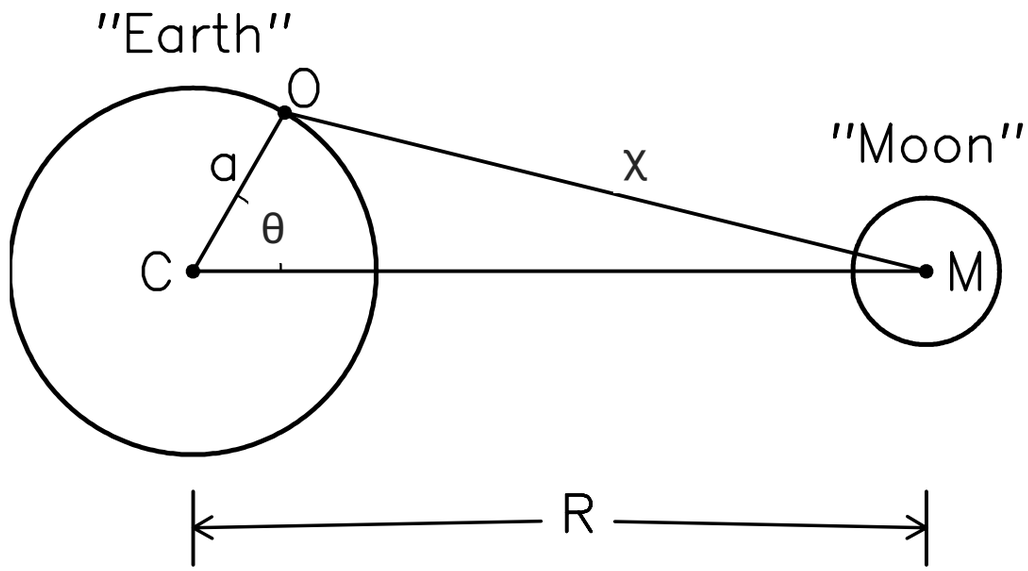
Agnew (2015)

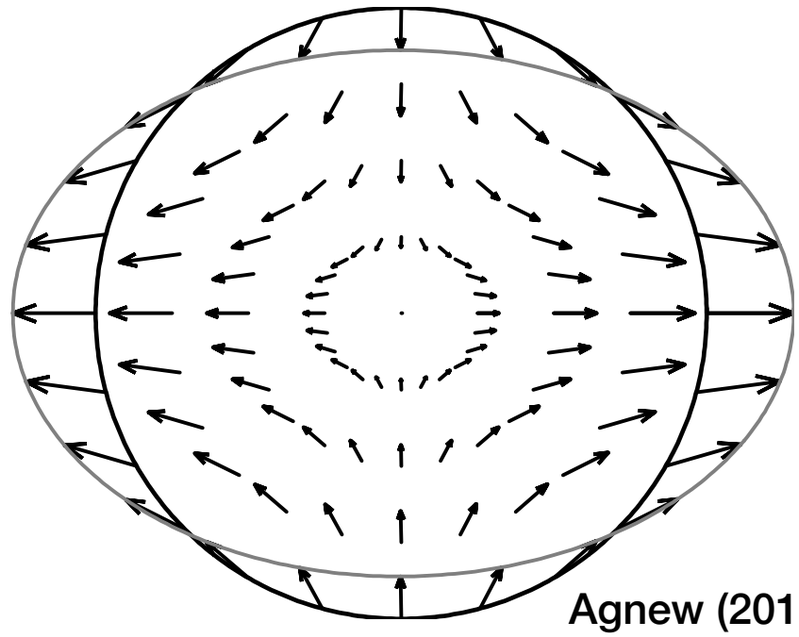
Tides arise due to differential, or unbalanced, gravitational forces within the oceans and the solid Earth

$$V_O = \frac{GM_{\text{ext}}}{\chi} = \frac{GM_{\text{ext}}}{R} \sum_{n=0}^{\infty} \left(\frac{a}{R}\right)^n P_n(\cos\theta)$$

The first two terms in the expansion (*n*=0, 1) represent a constant and uniform force along the CM direction, respectively

Tidal Potential



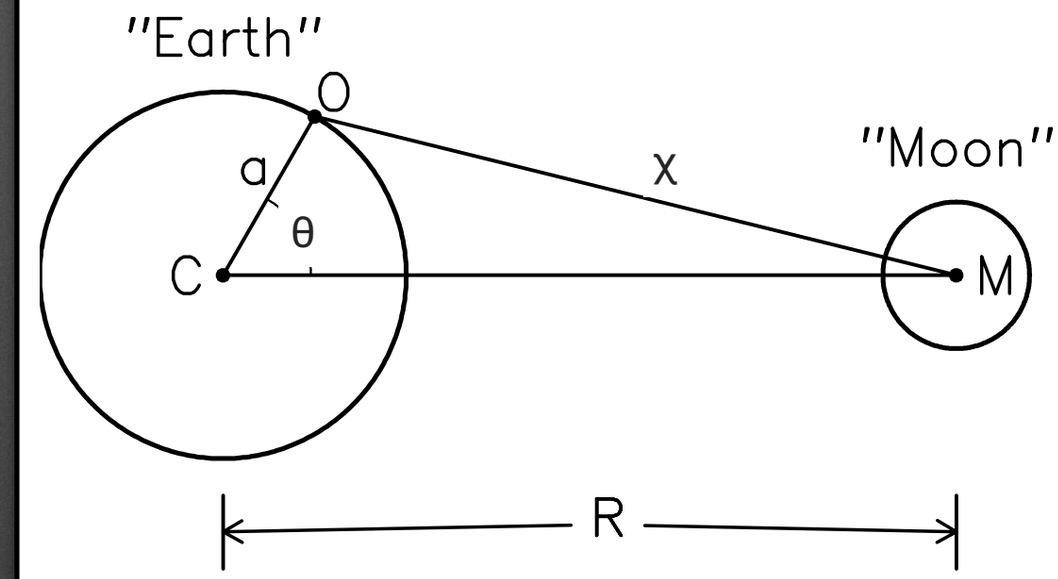


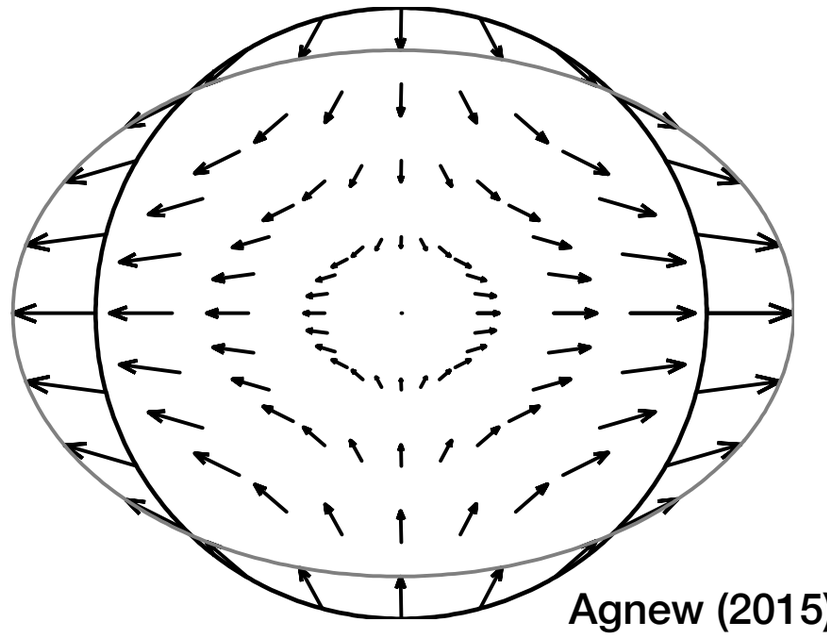
Agnew (2015)

The degree-2 term generates the largest tidal response

$$V_O = GM_{\text{ext}} \frac{a^2}{R^3} \left(\frac{3}{2} \cos^2 \theta - \frac{1}{2} \right)$$

Tidal Potential





Agnew (2015)

The degree-2 term generates the largest tidal response

$$V_O = GM_{\text{ext}} \frac{a^2}{R^3} \left(\frac{3}{2} \cos^2 \theta - \frac{1}{2} \right)$$

θ and *R*: Complicated functions of the astronomical ephemeris

Astronomical Parameters

τ = mean lunar time	1.035 days
s = mean longitude of moon	27.322 days
h = mean longitude of sun	365.242 days
p = mean longitude of lunar perigee	8.85 years
N′ = negative mean longitude of lunar ascending node	18.61 years
p′ = mean longitude of perihelion	20942 years

Astronomical Parameters

τ = mean lunar time	1.035 days
s = mean longitude of moon	27.322 days
h = mean longitude of sun	365.242 days
p = mean longitude of lunar perigee	8.85 years
N' = negative mean longitude of lunar ascending node	18.61 years
p' = mean longitude of perihelion	20942 years

Doodson Number

$$\begin{bmatrix} \eta_\tau & \eta_s & \eta_h & \eta_p & \eta_{N'} & \eta_{p'} \end{bmatrix}$$

Astronomical Parameters

τ = mean lunar time	1.035 days
s = mean longitude of moon	27.322 days
h = mean longitude of sun	365.242 days
p = mean longitude of lunar perigee	8.85 years
N′ = negative mean longitude of lunar ascending node	18.61 years
p′ = mean longitude of perihelion	20942 years

Doodson Number

$$\begin{bmatrix} \eta_\tau & \eta_s & \eta_h & \eta_p & \eta_{N'} & \eta_{p'} \end{bmatrix}$$

Astronomical argument (reference phase)

$$\Phi_\eta(t) = \eta_\tau\ \tau(t) + \eta_s\ s(t) + \eta_h\ h(t) + \eta_p\ p(t) + \eta_{N'}\ N'(t) + \eta_{p'}\ p'(t)$$

Astronomical Parameters

τ = mean lunar time	1.035 days
s = mean longitude of moon	27.322 days
h = mean longitude of sun	365.242 days
p = mean longitude of lunar perigee	8.85 years
N' = negative mean longitude of lunar ascending node	18.61 years
p' = mean longitude of perihelion	20942 years

Doodson Number

$$\begin{bmatrix} \eta_\tau & \eta_s & \eta_h & \eta_p & \eta_{N'} & \eta_{p'} \end{bmatrix}$$

Astronomical argument (reference phase)

$$\Phi_\eta(t) = \eta_\tau\ \tau(t) + \eta_s\ s(t) + \eta_h\ h(t) + \eta_p\ p(t) + \eta_{N'}\ N'(t) + \eta_{p'}\ p'(t)$$

Harmonics = Unique sums/differences of astronomical parameters

Total Tide = Superposition of tidal harmonics