Assess Documentation

Stephan Kramer

Jun 12, 2020

Contents:

CHAPTER 1

Analytical Solutions for the Stokes Equations in Spherical Shells

Assess is a python package that implements a number of analytical solutions to the Stokes equations in cylindrical and spherical shell domains.

CHAPTER 2

Download and installation

Assess is available from the Python Package Index (PyPi), so that it can be installed with a simple

```
$ pip install assess
```

It can also be downloaded from github.

2.1 Overview

Assess is a python package that implements a number of analytical solutions, in cylindrical and spherical shell domains, to the Stokes equations

$$\begin{aligned} -\nabla \cdot \tau + \nabla p &= -g\rho' \hat{\mathbf{r}}, \\ \tau &= \nu \left[\nabla \mathbf{u} + \nabla \mathbf{u}^T \right], \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned}$$

with velocity $\mathbf{u}$ and pressure p. The domain is assumed to be a spherical shell consisting of points with radius $R_- \leq r \leq R_+$. $\hat{\mathbf{r}}$ denotes the radial, outward unit-vector.

The gravitational acceleration g and viscosity ν are two user specified constants and the perturbation density ρ' is assumed to **either** have the following smooth form

$$\rho'(r, \varphi) = \frac{r^k}{R_+^k} \cos(n\varphi),$$

(2D-smooth)

$$\rho'(r, \theta, \varphi) = \frac{r^k}{R_+^k} Y_{lm}(\theta, \varphi),$$

(3D-smooth)

where in 2D, we use cylindrical coordinates with radius r and azimuthal angle φ, and in 3D, spherical coordinates with radius r, co-latitude θ, and longitude φ. The radial dependency is a simple polynomial (monomial) of order k.

In 2D, n is the wave number and in 3D l and m are the degree and order of the spherical harmonic function Y_{lm} (see *assess.Y()* for definition).

Or, ρ' is a perturbation at a specified radius r'

$$\rho'(r, \varphi) = \delta(r - r') \cos(n\varphi),$$

(2D-delta)

$$\rho'(r, \theta, \varphi) = \delta(r - r') Y_{lm}(\theta, \varphi),$$

(3D-delta)

where $\delta(r - r')$ is the Dirac delta function. Combined with two types of boundary conditions

$$-\mathbf{n} \cdot \tau \cdot \mathbf{n} + p = 0, \mathbf{n} \cdot \mathbf{u} = 0,$$

at $r = R_-$ and $r = R_+$

(free-slip)

$$\mathbf{u} = 0,$$

at $r = R_-$ and $r = R_+$

(zero-slip)

this leads to eight analytical solutions, which are implemented in the following classes

- *assess.CylindricalStokesSolutionSmoothFreeSlip*
- *assess.CylindricalStokesSolutionSmoothZeroSlip*
- *assess.CylindricalStokesSolutionDeltaFreeSlip*
- *assess.CylindricalStokesSolutionDeltaZeroSlip*
- *assess.SphericalStokesSolutionSmoothFreeSlip*
- *assess.SphericalStokesSolutionSmoothZeroSlip*
- *assess.SphericalStokesSolutionDeltaFreeSlip*
- *assess.SphericalStokesSolutionDeltaZeroSlip*

2.2 Usage

2.2.1 Smooth cases

To evaluate pressure p and velocity for the cylindrical, smooth, free-slip case

```
import assess
Rp, Rm = 2.22, 1.22  # outer and inner radius of domain
n = 2  # wave number
k = 3  # polynomial order of radial density variation
solution = assess.CylindricalStokesSolutionSmoothFreeSlip(n, k, Rp=Rp, Rm=Rm)
r, phi = 2, pi/2.  # location to evaluate
print("Radial component of velocity:", solution.u_r(r, phi))
print("Transverse component of velocity:", solution.u_phi(r, phi))
print("Pressure:", solution.p(r, phi))
```

To evaluate the radial component of stress:

```
print("Radial deviatoric stress:", solution.tau_rr(r,phi))
# or, radial component of full stress (including pressure)
print("Radial stress:", solution.radial_stress(r, phi))
```

To evaluate the density perturnation ρ' that was used:

```
print("Density perturbation:", solution.delta_rho(r,phi))
```

To setup a spherical, smooth, zero-slip case:

```
Rp, Rm = 2.22, 1.22  # outer and inner radius of domain
l = 2  # spherical degree
m = 3  # spherical order
k = 3  # polynomial order of radial density variation
solution = assess.SphericalStokesSolutionSmoothZeroSlip(l, m, k, Rp=Rp, Rm=Rm)
r, theta, phi = 2, pi/2., pi.  # location to evaluate
print("Radial component of velocity:", solution.u_r(r, theta, phi))
print("Colatitudinal (southward) component of velocity:", solution.u_theta(r, theta,␣
↪phi))
print("Longitudinal (eastward) component of velocity:", solution.u_phi(r, theta, phi))
print("Pressure:", solution.p(r, phi))
```

To simplify working with Cartesian coordinates, the methods `pressure_cartesian`, `delta_rho_cartesian`, `radial_stress_cartesian`, and `velocity_cartesian` allow providing 2d (cylindrical cases) or 3d (spherical cases) coordinates (tuple/list/array). The `velocity_cartesian` method also returns the velocity as a Cartesian xy or xyz-vector.

2.2.2 Delta-function cases

To evaluate the analytical solution with a delta function forcing, one must additionaly specify the radius r' used in the delta function $\delta(r - r')$. The analytical solution is split in two halves: one that is valid above the anomaly $r' \leq r \leq R_+$ and one below $R_- \leq r \leq r'$. Which of the two solutions is evaluated is chosen by setting the `sign` parameter: `sign=1` for the upper half and `sign=-1` for the lower half. **Note** that the methods do not check in which half the provided coordinates are actually located. This is done so that the discontinuous solutions at $r = r'$ can be evaluated without ambiguity.

```
Rp, Rm = 2.22, 1.22  # outer and inner radius of domain
rp = (Rp+Rm)/2.  # density anomaly at
l = 2  # spherical degree
m = 3  # spherical order
solution_above = assess.SphericalStokesSolutionDeltaFreeSlip(l, m, +1, Rp=Rp, Rm=Rm,␣
↪rp=rp)
solution_below = assess.SphericalStokesSolutionDeltaFreeSlip(l, m, -1, Rp=Rp, Rm=Rm,␣
↪rp=rp)
r, theta, phi = rp, pi/2., pi.  # location to evaluate
print("Radial component of velocity:", solution_above.u_r(r, theta, phi), solution_
↪below.u_r(r, theta, phi))
print("Colatitudinal (southward) component of velocity:", solution_above.u_theta(r,␣
↪theta, phi), solution_below.u_theta(r, theta, phi))
print("Longitudinal (eastward) component of velocity:", solution_above.u_phi(r, theta,
↪ phi), solution_below.u_phi(r, theta, phi))
print("Pressure:", solution.p(r, phi))
```

The delta-function classes implement the same methods as the smooth-case classes except for the `delta_rho` method.

2.2.3 Keyword arguments

All eight classes take the following (optional) keyword arguments with defaults

`Rp=2.22`	outer radius
`Rm=1.22`	inner radius
`nu=1.00`	viscosity
`g=1.00`	gravitation/magnitude of forcing

Additionally, the delta-function cases have the following default for `rp`

`rp=1.72`	radius of perturbation

2.3 User Reference

This page documents the eight classes, and its members, corresponding to the eight analytical solutions provided in the access python package.

Assess is a python package that implements analytical solutions to the Stokes equations in cylindrical and spherical domains.

class assess.**CylindricalStokesSolutionSmoothFreeSlip**(*n*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)

Bases: *assess.cylindrical.CylindricalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and free-slip boundary conditions

Parameters

- **n** – wave number
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

delta_rho(*r*, *phi*)

Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters

- **r** – radius
- **phi** – angle with x-axis

delta_rho_cartesian(*X*)

Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters **X** – 2D Cartesian coordinate

dpsi_rdr(*r*)

Radial derivative of radial part of streamfunction

Parameters **r** – radius

dpsi_rdr2 (*r*)
Second radial derivative of radial part of streamfunction

Parameters **r** – radius

p (*r*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **phi** – angle with x-axis

pressure_cartesian (*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 2D Cartesian location

psi_r (*r*)
Radial part of streamfunction

Parameters **r** – radius

radial_stress (*r*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

radial_stress_cartesian (*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 2D Cartesian location

tau_rphi (*r*, *phi*)
Return shear stress $\tau_{r\varphi}$.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

tau_rr (*r*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_phi (*r*, *phi*)
Return tangential component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_r (*r*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 2D Cartesian location

class assess.**CylindricalStokesSolutionSmoothZeroSlip**(*n*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.cylindrical.CylindricalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and zero-slip boundary conditions

Parameters

- **n** – wave number
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

delta_rho(*r*, *phi*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters

- **r** – radius
- **phi** – angle with x-axis

delta_rho_cartesian(*X*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters **X** – 2D Cartesian coordinate

dpsi_rdr(*r*)
Radial derivative of radial part of streamfunction

Parameters **r** – radius

dpsi_rdr2(*r*)
Second radial derivative of radial part of streamfunction

Parameters **r** – radius

p(*r*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **phi** – angle with x-axis

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 2D Cartesian location

psi_r(*r*)
Radial part of streamfunction

Parameters **r** – radius

radial_stress(*r*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 2D Cartesian location

tau_rphi(*r*, *phi*)
Return shear stress $\tau_{r\varphi}$.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

tau_rr(*r*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_phi(*r*, *phi*)
Return tangential component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_r(*r*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 2D Cartesian location

class assess.**CylindricalStokesSolutionDeltaFreeSlip**(*n*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)
Bases: *assess.cylindrical.CylindricalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and free-slip boundary conditions

Parameters

- **n** – wave number

- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

dpsi_rdr(*r*)
Radial derivative of radial part of streamfunction

Parameters **r** – radius

dpsi_rdr2(*r*)
Second radial derivative of radial part of streamfunction

Parameters **r** – radius

p(*r*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **phi** – angle with x-axis

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 2D Cartesian location

psi_r(*r*)
Radial part of streamfunction

Parameters **r** – radius

radial_stress(*r*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 2D Cartesian location

tau_rphi(*r*, *phi*)
Return shear stress $\tau_{r\varphi}$.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

tau_rr(*r*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius

- **phi** – angle with x-axis.

u_phi (*r*, *phi*)
Return tangential component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_r (*r*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

velocity_cartesian (*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 2D Cartesian location

class assess.**CylindricalStokesSolutionDeltaZeroSlip**(*n*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)
Bases: *assess.cylindrical.CylindricalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and zero-slip boundary conditions

Parameters

- **n** – wave number
- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

dpsi_rdr (*r*)
Radial derivative of radial part of streamfunction

Parameters **r** – radius

dpsi_rdr2 (*r*)
Second radial derivative of radial part of streamfunction

Parameters **r** – radius

p (*r*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **phi** – angle with x-axis

pressure_cartesian (*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 2D Cartesian location

psi_r(*r*)
Radial part of streamfunction

Parameters **r** – radius

radial_stress(*r*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 2D Cartesian location

tau_rphi(*r*, *phi*)
Return shear stress $\tau_{r\varphi}$.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

tau_rr(*r*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_phi(*r*, *phi*)
Return tangential component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_r(*r*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 2D Cartesian location

class assess.**SphericalStokesSolutionSmoothFreeSlip**(*l*, *m*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and free-slip boundary conditions

Parameters

- **l** – degree of the harmonic

- **m** – order of the harmonic
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

Pl(*r*)
Radial part of poloidal function

Parameters **r** – radius

dPldr(*r*)
Radial derivative of radial part of poloidal function

Parameters **r** – radius

dPldr2(*r*)
Second radial derivative of radial part of poloidal function

Parameters **r** – radius

delta_rho(*r*, *theta*, *phi*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

delta_rho_cartesian(*X*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters **X** – 3D Cartesian coordinate

p(*r*, *theta*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 3D Cartesian location

radial_stress(*r*, *theta*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 3D Cartesian location

tau_rphi(*r*, *theta*, *phi*)
Return longitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rr(*r*, *theta*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rtheta(*r*, *theta*, *phi*)
Return colatitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_phi(*r*, *theta*, *phi*)
Return longitudinal (eastward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_r(*r*, *theta*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_theta(*r*, *theta*, *phi*)
Return colatitudinal (southward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]

- **phi** – longitude in [0, 2*pi]

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 3D Cartesian location

class assess.**SphericalStokesSolutionSmoothZeroSlip**(*l*, *m*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and zero-slip boundary conditions

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

Pl(*r*)
Radial part of poloidal function

Parameters **r** – radius

dPldr(*r*)
Radial derivative of radial part of poloidal function

Parameters **r** – radius

dPldr2(*r*)
Second radial derivative of radial part of poloidal function

Parameters **r** – radius

delta_rho(*r*, *theta*, *phi*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

delta_rho_cartesian(*X*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters **X** – 3D Cartesian coordinate

p(*r*, *theta*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]

- **phi** – longitude in [0, 2*pi]

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 3D Cartesian location

radial_stress(*r*, *theta*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 3D Cartesian location

tau_rphi(*r*, *theta*, *phi*)
Return longitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rr(*r*, *theta*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rtheta(*r*, *theta*, *phi*)
Return colatitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_phi(*r*, *theta*, *phi*)
Return longitudinal (eastward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_r(*r*, *theta*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_theta(*r*, *theta*, *phi*)
Return colatitudinal (southward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 3D Cartesian location

class assess.**SphericalStokesSolutionDeltaFreeSlip**(*l*, *m*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and free-slip boundary conditions

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

Pl(*r*)
Radial part of poloidal function

Parameters **r** – radius

dPldr(*r*)
Radial derivative of radial part of poloidal function

Parameters **r** – radius

dPldr2(*r*)
Second radial derivative of radial part of poloidal function

Parameters **r** – radius

p(*r*, *theta*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]

- **phi** – longitude in [0, 2*pi]

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 3D Cartesian location

radial_stress(*r*, *theta*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 3D Cartesian location

tau_rphi(*r*, *theta*, *phi*)
Return longitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rr(*r*, *theta*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rtheta(*r*, *theta*, *phi*)
Return colatitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_phi(*r*, *theta*, *phi*)
Return longitudinal (eastward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_r(*r*, *theta*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_theta(*r*, *theta*, *phi*)
Return colatitudinal (southward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 3D Cartesian location

class assess.**SphericalStokesSolutionDeltaZeroSlip**(*l*, *m*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and zero-slip boundary conditions

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

Pl(*r*)
Radial part of poloidal function

Parameters **r** – radius

dPldr(*r*)
Radial derivative of radial part of poloidal function

Parameters **r** – radius

dPldr2(*r*)
Second radial derivative of radial part of poloidal function

Parameters **r** – radius

p(*r*, *theta*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]

- **phi** – longitude in [0, 2*pi]

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 3D Cartesian location

radial_stress(*r*, *theta*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 3D Cartesian location

tau_rphi(*r*, *theta*, *phi*)
Return longitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rr(*r*, *theta*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rtheta(*r*, *theta*, *phi*)
Return colatitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_phi(*r*, *theta*, *phi*)
Return longitudinal (eastward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_r(*r*, *theta*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_theta(*r*, *theta*, *phi*)
Return colatitudinal (southward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 3D Cartesian location

assess.**Y**(*l*, *m*, *theta*, *phi*)
Real-valued spherical harmonic function $Y_{lm}(\theta, \varphi)$

This is based on the following definition:

$$Y_{lm}(\theta, \varphi) = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos(\theta))\cos(m\varphi)$$

which is equal to the real part of scipy.special.sph_harm.

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

assess.**dYdphi**(*l*, *m*, *theta*, *phi*)
Colatitudinal derivative of spherical harmonic function $Y_{lm}(\theta, \varphi)$

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

assess.**dYdtheta**(*l*, *m*, *theta*, *phi*)
Longitudinal derivative of spherical harmonic function $Y_{lm}(\theta, \varphi)$

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

assess.**to_spherical**(*X*)
Convert Cartesian 3D coordinates X to spherical r, theta, phi

Parameters **X** – Cartesian 3D coordinates

Returns r, theta, phi radius, colatitude, longitude

assess.**from_spherical**(*r*, *theta*, *phi*)
Convert spherical r, theta, phi to 3D Cartesian coordinates.

Parameters

- **r,** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

Returns X Cartesian 3D coordinates

2.4 Developer Reference

This page gathers all docstrings of all classes (including class hierarchy), its members and separate functions in the submodules of assess.

2.4.1 assess.cylindrical module

Analytical solutions in cylindrical domains.

class assess.cylindrical.**AnalyticalStokesSolution**(*Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: object

Base class for solutions in sperical or cylindrical shell domains.

Parameters

- **Rp** – outer radius
- **Rm** – inner radius
- **g** – magnitude of source term
- **nu** – viscosity

class assess.cylindrical.**CylindricalStokesSolution**(*n*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.cylindrical.AnalyticalStokesSolution*

Base class for solutions in cylindrical shell domains.

This implements analytical solutions based on a streamfunction of the form.

$$\psi(r, \varphi) = \psi_r(r) \sin(n\varphi)$$

where ψ_r should be defined in a method *psi_r()* and its derivative in *dpsi_rdr()*.

Initialize basic parameters of analytical solution in cylindrical domain.

dpsi_rdr(*r*)
Abstract method to be implemented by subclass.

Parameters **r** – radius

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 2D Cartesian location

psi_r(*r*)
Abstract method to be implemented by subclass.

Parameters **r** – radius

radial_stress(*r*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 2D Cartesian location

tau_rphi(*r*, *phi*)
Return shear stress $\tau_{r\varphi}$.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

tau_rr(*r*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_phi(*r*, *phi*)
Return tangential component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

u_r(*r*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **phi** – angle with x-axis.

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 2D Cartesian location

class assess.cylindrical.**CylindricalStokesSolutionDelta**(*ABCD*, *n*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.cylindrical.CylindricalStokesSolution*

Base class for solutions in cylindrical shell domains with delta(r-r') forcing

This implements the analytical solution in one half (above or below r') of the domain which is based on a biharmonic streamfunction

$$\psi(r, \varphi) = \psi_r(r) \sin(n\varphi)$$

determinded by 4 coefficients, A, B, C, and D.

Parameters

- **ABCD** – list or tuple of 4 floats, coefficients for the 4 biharmonic solutions of the streamfunction
- **n** – wave number
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

dpsi_rdr(*r*)
Radial derivative of radial part of streamfunction

Parameters **r** – radius

dpsi_rdr2(*r*)
Second radial derivative of radial part of streamfunction

Parameters **r** – radius

p(*r*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **phi** – angle with x-axis

psi_r(*r*)
Radial part of streamfunction

Parameters **r** – radius

class assess.cylindrical.**CylindricalStokesSolutionDeltaFreeSlip**(*n*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)

Bases: *assess.cylindrical.CylindricalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and free-slip boundary conditions

Parameters

- **n** – wave number
- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius

- **nu** – viscosity
- **g** – forcing strength

class assess.cylindrical.**CylindricalStokesSolutionDeltaZeroSlip**(*n*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)

Bases: *assess.cylindrical.CylindricalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and zero-slip boundary conditions

Parameters

- **n** – wave number
- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

class assess.cylindrical.**CylindricalStokesSolutionSmooth**(*ABCDE*, *k*, *n*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)

Bases: *assess.cylindrical.CylindricalStokesSolution*

Base class for solutions in cylindrical shell domains with r^k forcing

The analytical solution is based on a streamfunction

determinded by 4 coefficients, A, B, C, and D corresponding to 4 independent biharmonic (i.e. homogenous) solutions and a fifth coefficient E that corresponds to the inhomogeneous part.

Parameters

- **ABCDE** – list or tuple of 5 floats, coefficients for the 4 biharmonic and one inhomogenous solutions of the streamfunction
- **k** – polynomial order of forcing
- **n** – wave number
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

delta_rho(*r*, *phi*)

Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters

- **r** – radius
- **phi** – angle with x-axis

delta_rho_cartesian(*X*)

Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters **X** – 2D Cartesian coordinate

dpsi_rdr(*r*)
Radial derivative of radial part of streamfunction

Parameters **r** – radius

dpsi_rdr2(*r*)
Second radial derivative of radial part of streamfunction

Parameters **r** – radius

p(*r*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **phi** – angle with x-axis

psi_r(*r*)
Radial part of streamfunction

Parameters **r** – radius

class assess.cylindrical.**CylindricalStokesSolutionSmoothFreeSlip**(*n*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.cylindrical.CylindricalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and free-slip boundary conditions

Parameters

- **n** – wave number
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

class assess.cylindrical.**CylindricalStokesSolutionSmoothZeroSlip**(*n*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.cylindrical.CylindricalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and zero-slip boundary conditions

Parameters

- **n** – wave number
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius

- **nu** – viscosity
- **g** – forcing strength

2.4.2 assess.delta module

This module contains functions that compute the coefficients in analytical solution to the Stokes equations in cylindrical and spherical shell domains with a delta function RHS forcing term, which in the 2D cylindrical shell domain takes the form:

$$f = -g\delta(r - r')cos(n\varphi)\hat{r}$$

where r is the radius (distance from origin), φ is the angle with the x-axis, r' is the radius of the density anomaly, and $\hat{r}$ is the outward pointing radial unit vector. The magnitude g, degree l and wave number n can be chosen arbitrarily. Similarly in a 3D spherical domain we consider the forcing term

$$f = -g\delta(r - r')Y_{lm}(\theta, \varphi)\hat{r}$$

where θ and φ are the co-latitude and longitude respectively, and Y_{lm} is Laplace's spherical harmonic function of degree l and order m.

We consider the following cases:

cylinder_delta_fs: cylindrical, with free-slip boundary condition at r=Rm and r=Rp cylinder_delta_ns: cylindrical, with zero-slip boundary conditions at r=Rm and r=Rp sphere_delta_fs: spherical, with free-slip boundary condition at r=Rm and r=Rp sphere_smooh_ns: spherical, with zero-slip boundary condition at r=Rm and r=Rp

The functions below take the following parameters:

param Rp outer radius

param Rm inner radius

param rp radius of density anomaly with Rm<rp<Rp

param n wave number (2D cylindrical only)

param l spherical degree (3D only)

param g scalar magnitude of forcing

param nu viscosity

param sign +1 or -1 for the upper (r>rp) and lower (r<rp) half of the solution

and return four coefficients A, B, C, D for the linear combination of biharmonic spherical functions that constitute the analytical solution.

This module has been automatically generated by extracting the solutions from the latex source of the associated paper. If sign=+1 we use the following substitutions

alpha_pm = Rp/rp, alpha_mp = Rm/rp, pm=+1, mp=-1

i.e. we take the top case for the symbols alpha_pm, alpha_mp, pm, and mp in the paper to obtain A_+, B_+, C_+, and D_+ corresponding to the upper half of the domain. If sign=-1 we have

alpha_pm = Rm/rp, alpha_mp = Rp/rp, pm=-1, mp=+1

to obtain the A_-, B_-, C_-, and D_- coefficients of the lower half of the solution.

`assess.delta.`**`coefficients_cylinder_delta_fs`**(*Rp*, *Rm*, *rp*, *n*, *g*, *nu*, *sign*)

`assess.delta.`**`coefficients_cylinder_delta_ns`**(*Rp*, *Rm*, *rp*, *n*, *g*, *nu*, *sign*)

assess.delta.**coefficients_sphere_delta_fs**(*Rp*, *Rm*, *rp*, *l*, *g*, *nu*, *sign*)

assess.delta.**coefficients_sphere_delta_ns**(*Rp*, *Rm*, *rp*, *l*, *g*, *nu*, *sign*)

2.4.3 assess.smooth module

This module contains functions that compute the coefficients in analytical solution to the Stokes equations in cylindrical and spherical shell domains with a smooth RHS forcing term, which in the 2D cylindrical shell domain takes the form:

$$f = -g(r/Rp)^k cos(n\varphi)\hat{r}$$

where r is the radius (distance from origin), φ is the angle with the x-axis, Rp and Rm are the outer and inner radius of the shell domain, and $\hat{r}$ is the outward pointing radial unit vector. The magnitude g, degree l and wave number n can be chosen arbitrarily. Similarly in a 3D spherical domain we consider the forcing term

$$f = -g(r/Rp)^k Y_{lm}(\theta, \varphi)\hat{r}$$

where θ and φ are the co-latitude and longitude respectively, and Y_{lm} is Laplace's spherical harmonic function of degree l and order m.

We consider the following cases:

> cylinder_smooth_fs: cylindrical, with free-slip boundary condition at r=Rm and r=Rp cylinder_smooth_ns: cylindrical, with zero-slip boundary conditions at r=Rm and r=Rp sphere_smooth_fs: spherical, with free-slip boundary condition at r=Rm and r=Rp sphere_smooth_ns: spherical, with zero-slip boundary condition at r=Rm and r=Rp

The functions below take the following parameters:

> **param Rp** outer radius
>
> **param Rm** inner radius
>
> **param k** polynomial degree of forcing term
>
> **param n** wave number (2D cylindrical only)
>
> **param l** spherical degree (3D only)
>
> **param g** scalar magnitude of forcing
>
> **param nu** viscosity

and return the five coefficients A, B, C, D, and E used in the analytical solution.

This module has been automatically generated by extracting the solutions from the latex source of the associated paper.

assess.smooth.**coefficients_cylinder_smooth_fs**(*Rp*, *Rm*, *k*, *n*, *g*, *nu*)

assess.smooth.**coefficients_cylinder_smooth_ns**(*Rp*, *Rm*, *k*, *n*, *g*, *nu*)

assess.smooth.**coefficients_sphere_smooth_fs**(*Rp*, *Rm*, *k*, *l*, *g*, *nu*)

assess.smooth.**coefficients_sphere_smooth_ns**(*Rp*, *Rm*, *k*, *l*, *g*, *nu*)

2.4.4 assess.spherical module

Analytical solutions in spherical shell domains.

class assess.spherical.**SphericalStokesSolution**(*l*, *m*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)

Bases: *assess.cylindrical.AnalyticalStokesSolution*

Base class for solutions in spherical shell domains.

This implements analytical solutions based on a poloidal function of the form.

$$\mathcal{P}(r,\theta,\varphi) = \mathcal{P}_l(r)Y_{lm}(\theta,\varphi)$$

so that the velocity solution takes the form:

$$\mathbf{u} = \nabla \times (\mathbf{r} \times \nabla\mathcal{P})$$

The function $\mathcal{P}_l$ and its derivative should be implement in methods *Pl()* and *dPldr()* in the subclass.

Pl(*r*)
Abstract method to be implemented by subclass.

Parameters **r** – radius

dPldr(*r*)
Abstract method to be implemented by subclass.

Parameters **r** – radius

pressure_cartesian(*X*)
Return pressure solution at Cartesian location.

Parameters **X** – 3D Cartesian location

radial_stress(*r*, *theta*, *phi*)
Return radial component of stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

radial_stress_cartesian(*X*)
Return radial component of stress at Cartesian location.

Parameters **X** – 3D Cartesian location

tau_rphi(*r*, *theta*, *phi*)
Return longitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

tau_rr(*r*, *theta*, *phi*)
Return radial component of deviatoric stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]

- **phi** – longitude in [0, 2*pi]

tau_rtheta(*r*, *theta*, *phi*)
Return colatitudinal component of shear stress.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_phi(*r*, *theta*, *phi*)
Return longitudinal (eastward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_r(*r*, *theta*, *phi*)
Return radial component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

u_theta(*r*, *theta*, *phi*)
Return colatitudinal (southward) component of velocity.

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

velocity_cartesian(*X*)
Return Cartesian velocity at Cartesian location.

Parameters **X** – 3D Cartesian location

class assess.spherical.**SphericalStokesSolutionDelta**(*ABCD*, *l*, *m*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)

Bases: *assess.spherical.SphericalStokesSolution*

Base class for solutions in spherical shell domains with delta(r-r') forcing

This implements the analytical solution in one half (above or below r') of the domain which is based on a poloidal function

$$\mathcal{P}(r,\theta,\varphi) = \mathcal{P}_l(r)Y_{lm}(\theta,\varphi)$$

and velocity

$$\mathbf{u} = \nabla \times (\mathbf{r} \times \nabla\mathcal{P})$$

where for biharmonic solutions, $\mathcal{P}_l(r)$ is determinded by four coefficients, A, B, C, and D.

Parameters

- **ABCD** – list or tuple of 4 floats, coefficients for the 4 independent biharmonic solutions.
- **l** – degree of the harmonic
- **m** – order of the harmonic
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

Pl(*r*)
Radial part of poloidal function

Parameters **r** – radius

dPldr(*r*)
Radial derivative of radial part of poloidal function

Parameters **r** – radius

dPldr2(*r*)
Second radial derivative of radial part of poloidal function

Parameters **r** – radius

p(*r*, *theta*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

class assess.spherical.**SphericalStokesSolutionDeltaFreeSlip**(*l*, *m*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and free-slip boundary conditions

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

class assess.spherical.**SphericalStokesSolutionDeltaZeroSlip**(*l*, *m*, *sign*, *Rp=2.22*, *Rm=1.22*, *rp=1.72*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionDelta*

Analytical Solution in cylindrical domain with delta(r-r') forcing and zero-slip boundary conditions

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **sign** – +1 for upper half solution r'<r<Rp -1 for lower half solution Rm<r<r'
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

class assess.spherical.**SphericalStokesSolutionSmooth**(*ABCDE*, *k*, *l*, *m*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)

Bases: *assess.spherical.SphericalStokesSolution*

Base class for solutions in spherical shell domains with r^k forcing

This implements the analytical solution in one half (above or below r') of the domain which is based on a poloidal function

$$\mathcal{P}(r,\theta,\varphi) = \mathcal{P}_l(r) Y_{lm}(\theta,\varphi)$$

and velocity

$$\mathbf{u} = \nabla \times (\mathbf{r} \times \nabla \mathcal{P})$$

where the solution $\mathcal{P}_l(r)$ is determinded by four coefficients, A, B, C, and D of four independent biharmonic solutions, and one coefficient E associated with the inhomogenous part.

Parameters

- **ABCDE** – list or tuple of 5 floats, coefficients for the 4 biharmonic and one inhomogenous solution.
- **l** – degree of the harmonic
- **m** – order of the harmonic
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

Pl(*r*)
Radial part of poloidal function

Parameters **r** – radius

dPldr(*r*)
Radial derivative of radial part of poloidal function

Parameters **r** – radius

dPldr2(*r*)
Second radial derivative of radial part of poloidal function

Parameters **r** – radius

delta_rho(*r*, *theta*, *phi*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

delta_rho_cartesian(*X*)
Perturbation density ρ' in forcing term: $g\rho'\hat{r}$

Parameters **X** – 3D Cartesian coordinate

p(*r*, *theta*, *phi*)
Pressure solution

Parameters

- **r** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

class assess.spherical.**SphericalStokesSolutionSmoothFreeSlip**(*l*, *m*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and free-slip boundary conditions

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius
- **nu** – viscosity
- **g** – forcing strength

class assess.spherical.**SphericalStokesSolutionSmoothZeroSlip**(*l*, *m*, *k*, *Rp=2.22*, *Rm=1.22*, *nu=1.0*, *g=1.0*)
Bases: *assess.spherical.SphericalStokesSolutionSmooth*

Analytical Solution in cylindrical domain with smooth r^k forcing and zero-slip boundary conditions

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **k** – polynomial order of forcing
- **Rp** – outer radius
- **Rm** – inner radius

- **nu** – viscosity
- **g** – forcing strength

assess.spherical.**Y**(*l*, *m*, *theta*, *phi*)
Real-valued spherical harmonic function $Y_{lm}(\theta, \varphi)$

This is based on the following definition:

$$Y_{lm}(\theta, \varphi) = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos(\theta))\cos(m\varphi)$$

which is equal to the real part of scipy.special.sph_harm.

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

assess.spherical.**Y_cartesian**(*l*, *m*, *X*)
Real-valued spherical harmonic function that takes Cartesian coordinates

See *Y()* :param m: order of the harmonic :param l: degree of the harmonic :param X: Cartesian 3D coordinates

assess.spherical.**dYdphi**(*l*, *m*, *theta*, *phi*)
Colatitudinal derivative of spherical harmonic function $Y_{lm}(\theta, \varphi)$

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

assess.spherical.**dYdtheta**(*l*, *m*, *theta*, *phi*)
Longitudinal derivative of spherical harmonic function $Y_{lm}(\theta, \varphi)$

Parameters

- **l** – degree of the harmonic
- **m** – order of the harmonic
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

assess.spherical.**from_spherical**(*r*, *theta*, *phi*)
Convert spherical r, theta, phi to 3D Cartesian coordinates.

Parameters

- **r,** – radius
- **theta** – co-latitude in [0, pi]
- **phi** – longitude in [0, 2*pi]

Returns X Cartesian 3D coordinates

assess.spherical.**to_spherical**(*X*)

Convert Cartesian 3D coordinates X to spherical r, theta, phi

Parameters **X** – Cartesian 3D coordinates

Returns r, theta, phi radius, colatitude, longitude

Python Module Index

a

Index

F

P

R

S

T

U

V

Y