# MEEUUW
# Mantle modelling Early Earth Utrecht University Work in progress

C. Thieulot

December 26, 2025

## Contents

# 1 Foreword

I have quite a history of naming codes and regretting my choice afterwards. MEEUUW is one more in my collection: SoPHa, SPHene, FANTOM, ELEFANT, FLAPS, FieldStone[1].

# 2 Install process

The code consists of about two dozens of Python files. Aside from standard Python modules (numpy, scipy, etc ...) the numba[2] package also needs to be installed on the machine you are running the code. The reason for the use of this module is as follows: "Numba translates Python functions to optimized machine code at runtime using the industry-standard LLVM compiler library. Numba-compiled numerical algorithms in Python can approach the speeds of C or FORTRAN. You don't need to replace the Python interpreter, run a separate compilation step, or even have a C/C++ compiler installed. Just apply one of the Numba decorators to your Python function, and Numba does the rest." [Taken from the numba website]

If numba is not available on your machine, simply delete all occurences of `@numba.njit` before the functions.

[1] `https://cedrict.github.io/`
[2] `https://numba.pydata.org/`

# 3 Philosophy

The code is entirely written in Python. The domain is only 2d, in the $xz$-plane. The geometry of the domain is controlled by the geometry parameter which can take four values:

- box: A Cartesian domain of size Lx,Ly
- half: Half an annulus, parameterised by an inner radius Rinner, an outer radius Router and an opening angle $\theta \in [-\pi/2, \pi/2]$
- quarter: Quarter of an annulus, parameterised by an inner radius Rinner, an outer radius Router and an opening angle $\theta \in [0, \pi/2]$
- eighth: Eighth of an annulus, parameterised by an inner radius Rinner, an outer radius Router and an opening angle $\theta \in [\pi/4, \pi/2]$

An axisymmetric option has been implemented for all four geometries[3].

add here figure of all 4 geometries

There is no adaptive mesh refinement nor mesh 'stretching'. The mesh also does not deform (i.e. no ALE-type algorithm). The mesh consists of nelx*nelz elements. In the case of curved domains the topology of the mesh is the same but it is curved to conform to the domain boundaries.

The code is not meant to compete with ASPECT or any similar code. It is meant to be an educational tool, with MSc or PhD students in mind. The interface is simple and flexible. Inside a experimentX.py file the user needs to specify the necessary parameters (e.g. the resolution, domain size, number of timesteps, etc ...) and fill in a few template functions:

```
initial_temperature
assign_boundary_conditions_V
assign_boundary_conditions_T
particle_layout
material_model
gravity_model
```

Run parameters also need to be defined this file, and if not the value in the `set_default_parameters.py` file apply.

Because the code is only 2d the memory requirements (aside from the linear solver) are rather mild with respect with the amount of RAM even laptops now have. As such I make no attempt at being cautious when it comes to memory use. This means for example that each particle carries a dozen or so field. Likewise the material parameters are projected from the particles onto the quadrature points and these projected fields are permanently stored in large arrays. The rationale is as follows: Lets us assume that we are running a $200 \times 200$ element simulation. There are then at least 9 times the number of quadrature points, i.e. $\sim 360,000$, each carrying approx. 10 double precision floats (64 bits). In total the memory use for these is $360000 \cdot 10 \cdot 64 = 230400000$bits, or about 220Mb of memory.

There are at the moment 18 so-called experiments (also called cookbooks in ASPECT) that either aim at reproducing existing articles results (numerical benchmarks) or built as setups to be used in class or for research purposes. Each experiment is entirely contained in its corresponding experimentX.py file. The current list of experiments is as follows:

```
# experiment  0: Blankenbach et al, 1993 - isoviscous convection
# experiment  1: van Keken et al, JGR, 1997 - Rayleigh-Taylor experiment
# experiment  2: Schmeling et al, PEPI 2008 - Newtonian subduction
# experiment  3: Tosi et al, 2015 - visco-plastic convection
# experiment  4: not sure. mantle size convection
```

[3]Only for the Stokes solver! no temp eq yet

```
# experiment  5: Trompert & Hansen, Nature 1998 - convection w/ plate-like
# experiment  6: Crameri et al, GJI 2012 (cosine perturbation & plume)
# experiment  7: ESA workshop
# experiment  8: quarter - sinker
# experiment  9: axisymmetric Mars setup
# experiment 10: axisymmetric 4D dyn Earth benchmark of Stokes sphere
# experiment 11: rising plume
# experiment 12: hollow earth gravity benchmark
# experiment 13: sinking block benchmark
# experiment 14: slab detachment (Schmalholz 2011)
# experiment 15: stokes sphere axisymmetric
# experiment 16: subduction initiation from Matsumoto and Tomoda (1983)
# experiment 17: sinking sphere 512km with sticky air
# experiment 18: sinking sphere unit square with sticky air
```

The results obtained with the code for these experiments are available in the `RESULTS` folder (simply open the `results.pdf` file – if it not present use the `lat` script to compile the LaTeXcode)

In order to select one of these experiments, simply open `meeuuw.py` and edit line 66 which reads experiment=X.

In order to run the code, simply then type the following in the terminal:

```
python3 meeuuw.py
```

Note that a script carrying this also exists and can be executed as follows:

```
./run
```

The code will then run and will produce *a lot* of information while running in the terminal. If you wish to keep this information you can redirect the screen output to a file as follows for example:

```
python3 meeuuw.py > screen_output.text
```

The code also generates `.ascii` files (to be plotted with gnuplot for example) and `.vtu` files to be opened with ParaView. All are placed automatically in the `OUTPUT` folder.

After a simulation, if results are to be discarded, and before running a new model, it is recommended to remove all results by running the following script:

```
./cleandata
```

# 4 Numerical methods

## 4.1 Finite Element Method

The Stokes equations are discretised by means of the Finite Element Method (FEM). As explained in Thieulot et al. (2022) and Thieulot et al. (2025), there are many so-called element pairs for the velocity and pressure spaces. The MEEUUW code relies on quadrilateral Taylor-Hood elements only, i.e. $Q_2 \times Q_1$. The energy equation relies on $Q_2$ elements for the temperature. The FEM formulation of these equations is explained in the FieldStone manual[4] and plenty of textbooks.

The discretisation of the Stokes equations (mass and momentum equation) yields the following linear system:

$$\begin{pmatrix} \mathbb{K} & \mathbb{G} \\ \mathbb{G}^T & 0 \end{pmatrix} \cdot \begin{pmatrix} \vec{\mathcal{V}} \\ \vec{\mathcal{P}} \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{h} \end{pmatrix}$$

At the moment the whole matrix and right hand side is built and stored in the following arrays:

```
bignb_V=nel*(  (m_V*ndof_V)**2  +  2*(m_V*ndof_V*m_P)  )
II_V=np.zeros(bignb_V,dtype=np.int32)
JJ_V=np.zeros(bignb_V,dtype=np.int32)
VV_T=np.zeros(bignb,dtype=np.float64)
rhs=np.zeros(Nfem_T,dtype=np.float64)
```

while the discretisation of the energy equation yields

$$\left(\mathbb{M} + \frac{1}{2}(\mathbb{K}_a + \mathbb{K}_d)\delta t\right) \cdot \vec{\mathcal{T}}^{k+1} = \left(\mathbb{M} - \frac{1}{2}(\mathbb{K}_a + \mathbb{K}_d)\delta t\right) \cdot \vec{\mathcal{T}}^{k} + \vec{H}$$

where $\mathbb{M}$ is the mass matrix, $\mathbb{K}_a$ is the advection matrix, $\mathbb{K}_d$ is the diffusion matrix.

The fully assembled matrix and right hand side are stored in the following arrays:

```
bignb_T=nel*m_T**2
II_T=np.zeros(bignb_T,dtype=np.int32)
JJ_T=np.zeros(bignb_T,dtype=np.int32)
VV_T=np.zeros(bignb,dtype=np.float64)
rhs=np.zeros(Nfem_T,dtype=np.float64)
```

At the moment there is no stabilisation scheme implemented which can help with highly advective problems (high Peclet number).

## 4.2 Particle-In-Cell method

The code relies on the Particle-in-Cell method to track compositional fields. This is not the only method used in geodynamics, see for example the textbooks by Gerya, or by Tackley & Ismael-Zaddeh. All arrays bearing the name swarm_xxxx are nparticle long. Note that the total number of particles is constant (no addition, no deletion). swarm_x, swarm_z contain all their Cartesian coordinates, swarm_rad, swarm_theta contain all their polar coordinates, swarm_u, swarm_w contain all their velocities, swarm_r, swarm_t contain all their local coordinates ($r, t \in [-1:1]$), swarm_iel contains their parent element number, etc ... The particles are advected by means of either Runge-Kutta 1st, 2nd and 4th order in space algorithms (as controlled by the global parameter RKorder). Note that RKorder can also be set to -1 explain. The global parameter nparticle_per_dim controls the initial number of particles per element and the total number of particles:

```
nparticle_per_element=nparticle_per_dim**2
nparticle=nel*nparticle_per_element
```

[4] https://cedric.github.io/

The initial type of spatial distribution of the particles is controlled by the parameter particle_distribution which can take 4 values:

1. random: nparticle_per_element particles are generated at random inside the element using the local coordinates and an isoparametric mapping;
2. regular: the particles are arranged in an nparticle_per_dim∗nparticle_per_dim array;
3. Poisson disc sampling[5]. At the moment this is only available for the 'box' geometry.
4. pseudo-random: this is similar to 2, but a small perturbation is added to the positions.

Fields can be easily added to particles depending on the needs of a given experiment, for example swarm_F tracks the depletion.

The core of the Particle-In-Cell method that is implemented is as follows. At the beginning of each time step the temperature, pressure and strain rate is interpolated on a particle. The so-called material_model is then called and returns the following particle fields: swarm_rho,swarm_eta,swarm_hcond, swarm_hcapa,swarm_hprod.

The fields carried by the particle must then be projected where they are needed for the Finite Element matrix building algorithm, i.e. the Gauss quadrature points (typically and array of $3 \times 3$ points inside each element).

There are multiple options, controlled by the particle_rho_projection and particle_eta_projection parameters:

- 'elemental
- 'nodal'
- 'least_squares' – see subsection below

The fields are then projected onto the mesh nodes, the resulting fields belonging to the $Q_1$ space. An arithmetic average is used for the heat capacity, heat conductivity, heat production and density, while the user may choose between and arithmetic, geometric, or harmonic averaging for the viscosity.

Finally these nodal quantities are interpolated onto the quadrature points and these values will be used to build the FE matrices for the Stokes and/or energy equations.

### 4.2.1 Least square algorithms

One could think that with high(er) order elements optimal convergence is unlikely to be reached if viscosity (and density) are assumed to be constant inside each element (see Gassmöller, Lokavarapu, et al. (2019)). One could therefore use the least-square method to arrive at a functional representation of the field inside the element which is as close as possible (in the least-squares sense, then) to the particle-based field.

Thielmann et al. (2014) use the $Q_2P_{-1}$ element and introduce an element-wise interpolation scheme based on a least squares fitting of the particle properties and choose the functional to be a linear function to match the pressure space. They define the error $\epsilon$ such that

$$\epsilon^2 = \sum_{i=1}^{n} (\tilde{f}(x_i, y_i) - f_i)^2$$

[5] https://www.jasondavies.com/poisson-disc/

with $\tilde{f}(x,y) = a + bx + cy$ and proceed to look for the minimum of $\epsilon^2$, i.e. $\vec{\nabla}(\epsilon^2) = 0$ in the $\{a, b, c\}$ space:

$$\begin{aligned}
0 = \frac{\partial \epsilon^2}{\partial a} &= 2\sum_i (\tilde{f}(x_i, y_i) - f_i) \\
&= 2\sum_i (a + bx_i + cy_i - f_i) \\
&= 2\left[a\sum_i 1 + b\sum_i x_i + c\sum y_i - \sum_i f_i\right] \\
0 = \frac{\partial \epsilon^2}{\partial b} &= 2\sum_i (\tilde{f}(x_i, y_i) - f_i)x_i \\
&= 2\sum_i (a + bx_i + cy_i - f_i)x_i \\
&= 2\left[a\sum_i x_i + b\sum_i x_i^2 + c\sum x_i y_i - \sum_i x_i f_i\right] \\
0 = \frac{\partial \epsilon^2}{\partial c} &= 2\sum_i (\tilde{f}(x_i, y_i) - f_i)y_i \\
&= 2\sum_i (a + bx_i + cy_i - f_i)y_i \\
&= 2\left[a\sum_i y_i + b\sum_i x_i y_i + c\sum y_i^2 - \sum_i y_i f_i\right]
\end{aligned}$$

so

$$\begin{pmatrix} \sum_i 1 & \sum_i x_i & \sum_i y_i \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i y_i \\ \sum_i y_i & \sum_i x_i y_i & \sum_i y_i^2 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum_i f_i \\ \sum_i x_i f_i \\ \sum_i y_i f_i \end{pmatrix}$$

This method can trivially be extended to three dimensions. It must also be noted that it is not cheap: for each element the matrix and rhs above must be formed and the system solved for $a, b, c$.

We could also then decide to use a bi-linear function $\tilde{f}$, i.e.

$$\tilde{f}(x,y) = a + bx + cy + dxy$$

which lies in the $Q_1$ space of Taylor-Hood quadrilateral elements. In this case the error is

$$\epsilon^2 = \sum_{i=1}^{n} (\tilde{f}(x_i, y_i) - f_i)^2 = \sum_{i=1}^{n} (a + bx_i + cy_i + dx_i y_i - f_i)^2$$

and one has to solve a $4 \times 4$ system this time:

$$\begin{pmatrix} \sum_i 1 & \sum_i x_i & \sum_i y_i & \sum_i x_i y_i \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i y_i & \sum_i x_i^2 y_i \\ \sum_i y_i & \sum_i x_i y_i & \sum_i y_i^2 & \sum_i x_i y_i^2 \\ \sum_i x_i y_i & \sum_i x_i y_i & \sum_i y_i^2 & \sum_i x_i^2 y_i^2 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \sum_i f_i \\ \sum_i x_i f_i \\ \sum_i y_i f_i \\ \sum_i x_i y_i f_i \end{pmatrix}$$

which we write $\boldsymbol{A} \cdot \vec{c} = \boldsymbol{b}$. Note that the matrix $\boldsymbol{A}$ is symmetric. We see that this is a potentially numerically problematic equation. Distances/coordinates in geodynamic calculations are of the order of 100-1000km and viscosities are between $10^{19}$ and $10^{26}$Pa s. The matrix would contain very large terms, which may compromise the accuracy of the system solve.

Once this linear system (or the previous one) has been solved we have obtained the coefficients $a, b, c(, d)$ which allow us to compute $\tilde{f}$ anywhere inside the element, and especially at the quadrature points. Once these coefficients have been obtained one can compute $\tilde{f}$ anywhere in the element, and in particular at the quadrature points.

Using a different (bi)linear function $\tilde{f}$ for each element means that it is likely to be discontinuous from one element to another in regions of high gradients.

There is however one drawback with this approach (linear or bi-linear alike): in the areas of steep gradients the computed coefficients can be such that the function $\tilde{f}$ evaluated on a quadrature point is negative which 1) would be wrong but not numerically dramatic for density, 2) would be wrong and physically and numerically problematic for viscosity (a viscosity cannot be negative, and this would automatically destroy the SPD nature of the viscous block of the Stokes matrix).

This problem is discussed in Thielmann et al. (2014) in Section 3.2.1 and they call this "Over- and Under-shooting". A simple (iterative) fix is then designed which insures that the computed value is within user-defined acceptable bounds. This is also mentioned in Gassmöller, Lokavarapu, et al. 2019 but the authors explain that this problem was not encountered in the context of the publication.

Remark: One could consider the above least-square approach with $\tilde{f} = a$, i.e. $\tilde{f}$ is a zero-th order polynomial. In this case

$$\epsilon^2 = \sum_{i=1}^{n} (\tilde{f}(x_i, y_i) - f_i)^2 = \sum_{i=1}^{n} (a - f_i)^2$$

The gradient becomes

$$\vec{\nabla}(\epsilon^2) = \frac{d\epsilon^2}{da} = \sum_{i=1}^{n} 2(a - f_i) = 0$$

or $a = \frac{1}{n} \sum_i f_i$. We here recover the arithmetic averaging!

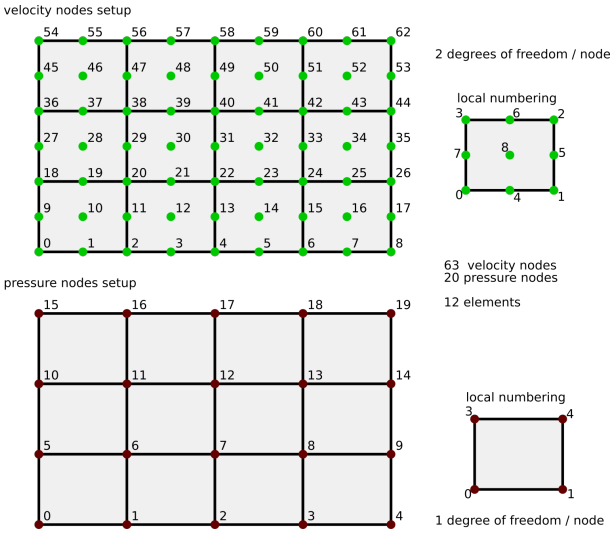
# 5 Code structure

```
- quadrature rule points and weights
- open output files & write headers
- build velocity nodes coordinates
- connectivity for velocity nodes
- build pressure grid
- build pressure connectivity array
- define velocity boundary conditions
- define temperature boundary conditions
- initial temperature.
- define_mapping
- compute area of elements / sanity check
- precompute basis functions values at quadrature points
- precompute basis functions and jacobian values at V nodes
- compute coordinates of quadrature points
- compute gravity vector at quadrature points
- compute gravity on mesh points
- compute normal vector of domain - NOT needed anymore
- compute array for assembly
- fill II_V,JJ_V arrays for Stokes matrix
- fill II_T,JJ_T arrays for temperature matrix & plith matrix
- particle coordinates setup
- particle paint
- particle layout
- --------------------- time stepping loop ---------------------------------
    - interpolate strain rate, pressure and temperature on particles
    - compute depletion and super solidus temperature
    - evaluate density and viscosity on particles (and hcond, hcapa, hprod)
    - project particle properties on elements
    - project particle properties on V nodes
    - remove nodal rho profile
    - assign values to quadrature points
    - compute lithostatic pressure a la Jourdon & May, Solid Earth, 2022
    - build FE matrix
    - solve stokes system
    - split solution into separate u,v,p velocity arrays
    - convert velocity to polar coordinates
    - compute timestep
    - normalise pressure: simple approach to have avrg p = 0 (volume or surface)
    - compute elemental pressure
    - project Q1 pressure onto Q2 (vel,T) mesh
    - project velocity on quadrature points
    - compute L2 errors
    - build temperature matrix
    - solve temperature system
    - compute vrms
    - compute nodal heat flux
    - compute heat flux and Nusselt at top and bottom
    - compute temperature profile
```

```
    - compute nodal strainrate
    - compute nodal deviatoric strainrate
    - compute deviatoric stress tensor components (elemental & nodal)
    - compute full stress tensor components
    - compute dynamic topography at bottom and surface topo
    - compute nodal pressure gradient
    - advect particles
    - locate particles and compute reduced coordinates
    - compute strain on particles
    - output min/max coordinates of each material in one single file
    - generate/write in pvd files
    - output solution to vtu file
    - output particles to vtu file
    - output quadrature points to vtu file
    - compute gravitational field above domain
    - assess steady state
- ---------------------- end time stepping loop -----------------------------
```

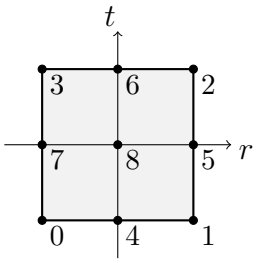
# 6 Nomenclature

All calculations take place in the $x-z$ plane. The corresponding reduced/local coordinates are then $r, t$. There is a single mesh with nel elements, but the velocity is in the space $Q_2$ while the pressure is the space $Q_1$. There are then (2∗nelx+1)∗(2∗nelz+1) velocity nodes and (nelx+1)∗(nelz+1) pressure nodes. All quantities ending in _V (e.g. nn_V, m_V, x_V,y_V) are related to/defined on the velocity nodes while those ending in _P (e.g. nn_P, m_P, x_P,y_P) are related to/defined on the pressure nodes.



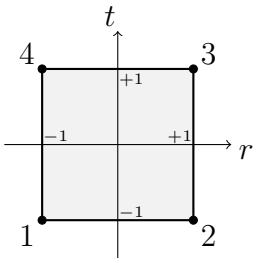

The internal numbering of the velocity nodes is as follows:

(tikz_q22d.tex)

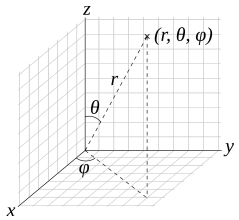



while the numbering of the pressure nodes is as follows:

(tikz_q12d.tex)




For the spherical coordinates we adopt the following notations for the angles $\theta$ and $\phi$:



## 6.1 Mesh related parameters

- nelx, nelz: number of elements in each direction
- nnx,nnz: number of V nodes in each direction
- nel: number of elements
- nn_V: number of velocity nodes
- nn_P: number of pressure nodes
- ndof_V_el: number of V dofs per element (=18)
- m_V: number of velocity nodes per element (=9)
- m_P: number of pressure nodes per element (=4)
- m_T: number of temperature nodes per element (=9)
- hx,hz: size of an element
- Nfem_V: number of velocity degrees of freedom
- Nfem_P: number of pressure degrees of freedom
- Nfem_T: number of temperature degrees of freedom
- Nfem: total number of degrees of freedom (Stokes eqs)
- vrms: root mean square velocity $\mathbf{v}_{rms}$

## 6.2 Mesh related arrays

- exx_n,ezz_n,exz_n: strain rate tensor $\boldsymbol{\varepsilon}(\vec{\mathbf{v}})$ on the nodes
- exx_e, ezz_e, exz_e: strain rate tensor $\boldsymbol{\varepsilon}(\vec{\mathbf{v}})$ in the element center
- dxx_n,dzz_n,dxz_n: deviatoric strain rate tensor $\boldsymbol{\varepsilon}^d(\vec{\mathbf{v}})$ on the nodes
- tauxx_n,tauzz_n,tauxz_n: deviatoric stress $\boldsymbol{\tau}$ on the nodes
- sigmaxx_n,sigmazz_n,sigmaxz_n: full stress tensor $\boldsymbol{\sigma}$ on the nodes
- T: temperature array
- p: pressure field (on $Q_1$ mesh)
- q: pressure field (on $Q_2$ mesh)
- u,w: velocity components arrays
- x_V,y_V: coordinates arrays of velocity nodes
- x_P,y_P: coordinates arrays of pressure nodes
- icon_V: connectivity array for velocity nodes
- icon_P: connectivity array for pressure nodes

- bc_fix_V, bc_val_V: boundary conditions arrays for velocity
- bc_fix_T, bc_val_T: boundary conditions arrays for temperature
- N_V, dNdr_V, dNdt_V, dNdx_V, dNdz_V: velocity basis functions and derivatives
- N_P: pressure basis functions
- qx_n, qz_n: nodal heat flux component
- II_V, JJ_V, VV_V: arrays to store Stokes FEM matrix
- II_T, JJ_T, VV_T: arrays to store energy FEM matrix
- r_V, s_V: arrays of size m_V containing red coords of V nodes
- rad_V, theta_V: polar coordinates of V nodes, i.e. $\theta_V \in [-\pi/2, \pi/2]$
- rad_P, theta_P: polar coordinates of P nodes

## 6.3 Particle swarm related arrays and parameters

- swarm_x:
- swarm_z:
- swarm_rho:
- swarm_eta:
- swarm_T:
- swarm_p:
- swarm_F:
- swarm_sst:
- swarm_exx:
- swarm_ezz:
- swarm_exz:
- swarm_iel:
- swarm_rad:
- swarm_theta:
- swarm_u:
- swarm_w:
- swarm_r:
- swarm_t:
- swarm_strain:

- swarm_active:
- swarm_hcond:
- swarm_hcapa:
- swarm_hprod:

## 6.4 Quadrature related arrays and parameters

- nq_per_element=nq_per_dim**ndim
- nq=nq_per_element*nel
- xq,zq
- rhoq
- etaq
- Tq
- hcondq
- hcapaq
- hprodq
- dpdxq,dpdzq
- gxq,gzq
- exxq,ezzq,exzq
- JxWq
- jcbi00q,jcbi01q,jcbi10q,jcbi11q
- nqperdim,qcoords,qweights

# 7 Conversions between coordinate systems

## 7.1 vectors

### 7.1.1 Polar coordinates (2d)

The equations and the code are formulated in Cartesian coordinates. However, when an annulus-like geometry is used one may wish to look at certain quantities expressed in polar coordinates. For example, the velocity $\vec{\mathbf{v}} = (u, w)$ is then given by

$$\mathbf{v}_r = u\cos\theta + w\sin\theta \tag{1}$$
$$\mathbf{v}_\theta = -u\sin\theta + w\cos\theta \tag{2}$$

### 7.1.2 Spherical coordinates (3d)

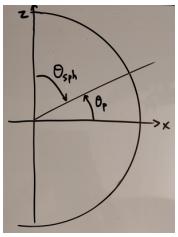
The velocity in spherical coordinates is given by

$$v_r = u\sin\theta\cos\phi + v\sin\theta\sin\phi + w\cos\theta \tag{3}$$
$$v_\theta = u\cos\theta\cos\phi + v\cos\theta\sin\phi - w\sin\theta \tag{4}$$
$$v_\phi = -u\sin\phi + v\cos\phi \tag{5}$$

We will be using these expressions only when the axisymmetric mode is activated, since in fact the problem is in fact 3d. Because of the symmetry we expect $v_\phi = 0$ and $\phi = 0$ sine we compute in the $xz$ plane, so that in the end we have the following expression:

$$v_r = u\sin\theta + w\cos\theta \tag{6}$$
$$v_\theta = u\cos\theta - w\sin\theta \tag{7}$$

One thing to be aware of, is the fact that the $\theta_{sph}$ angle of spherical harmonics is measured from the vertical axis, while the theta angle $\theta_p$ is measured from the horizontal axis in polar coordinates, i.e. $\theta_{sph} = \frac{\pi}{2} - \theta_{pol}$ in the $x > 0$ half-plane where calculations take place.




The theta_V quantity in the code corresponds to the polar coordinates angle $\theta_p$.

Then

$$v_r = u\sin(\frac{\pi}{2} - \theta_p) + w\cos(\frac{\pi}{2} - \theta_p) = u\cos\theta_p + w\sin\theta_p \tag{8}$$
$$v_\theta = u\cos(\frac{\pi}{2} - \theta_p) - w\sin(\frac{\pi}{2} - \theta_p) = u\sin\theta_p - w\cos\theta_p \tag{9}$$

This translates as follows:

```
if geometry=='quarter' or geometry=='half' or geometry=='eighth':
    if axisymmetric:
        vr=u*np.cos(theta_V)+v*np.sin(theta_V)
        vt=u*np.sin(theta_V)-v*np.cos(theta_V)
    else:
        vr=u*np.cos(theta_V)+v*np.sin(theta_V)
        vt=-u*np.sin(theta_V)+v*np.cos(theta_V)
```

## 7.2 tensors

### 7.2.1 Polar coordinates

$$
\begin{aligned}
\boldsymbol{T}_{\text{polar}} &= \begin{pmatrix} T_{rr} & T_{r\theta} \\ T_{\theta r} & T_{\theta\theta} \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} T_{xx} & T_{xy} \\ T_{yx} & T_{yy} \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} T_{xx}\cos\theta + T_{xy}\sin\theta & -T_{xx}\sin\theta + T_{xy}\cos\theta \\ T_{yx}\cos\theta + T_{yy}\sin\theta & -T_{yx}\sin\theta + T_{yy}\cos\theta \end{pmatrix} \\
&= \begin{pmatrix} T_{xx}c_\theta^2 + T_{xy}s_\theta c_\theta + T_{yx}s_\theta c_\theta + T_{yy}s_\theta^2 & -T_{xx}s_\theta c_\theta + T_{xy}c_\theta^2 - T_{yx}s_\theta^2 + T_{yy}s_\theta c_\theta \\ -T_{xx}s_\theta c_\theta - T_{xy}s_\theta^2 + T_{yx}c_\theta^2 + T_{yy}s_\theta c_\theta & T_{xx}s_\theta^2 - T_{xy}s_\theta c_\theta - T_{yx}s_\theta c_\theta + T_{yy}c_\theta^2 \end{pmatrix} \\
&= \begin{pmatrix} T_{xx}c_\theta^2 + 2T_{xy}s_\theta c_\theta + T_{yy}s_\theta^2 & (T_{yy} - T_{xx})s_\theta c_\theta + T_{xy}(c_\theta^2 - s_\theta^2) \\ (T_{yy} - T_{xx})s_\theta c_\theta + T_{xy}(c_\theta^2 - s_\theta^2) & T_{xx}s_\theta^2 - 2T_{xy}s_\theta c_\theta + T_{yy}c_\theta^2 \end{pmatrix}
\end{aligned}
$$

where we have assumed that the tensor $\boldsymbol{T}$ is symmetric at some point. Note also that the trace of $\boldsymbol{T}_{\text{polar}}$ is

$$Tr[\boldsymbol{T}_{\text{polar}}] = T_{xx}c_\theta^2 + 2T_{xy}s_\theta c_\theta + T_{yy}s_\theta^2 + T_{xx}s_\theta^2 - 2T_{xy}s_\theta c_\theta + T_{yy}c_\theta^2 = T_{xx} + T_{zz}$$

Indeed, the trace of a tensor is an invariant.

Finally we can also use $\sin 2x = 2\sin x \cos x$ and $\cos 2x = \cos^2 x - \sin^2 x$ to arrive at

$$
\begin{aligned}
\boldsymbol{T}_{\text{polar}} &= \begin{pmatrix} T_{rr} & T_{r\theta} \\ T_{\theta r} & T_{\theta\theta} \end{pmatrix} \\
&= \begin{pmatrix} T_{xx}\cos^2\theta + T_{xy}\sin 2\theta + T_{yy}\sin^2\theta & \frac{1}{2}(T_{yy} - T_{xx})\sin 2\theta + T_{xy}\cos 2\theta \\ \frac{1}{2}(T_{yy} - T_{xx})\sin 2\theta + T_{xy}\cos 2\theta & T_{xx}\sin^2\theta - T_{xy}\sin 2\theta + T_{yy}\cos^2\theta \end{pmatrix}
\end{aligned}
$$

or,

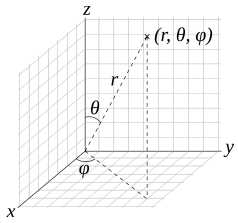
$$
\begin{aligned}
T_{rr} &= T_{xx}\cos^2\theta + T_{xy}\sin 2\theta + T_{yy}\sin^2\theta && (10) \\
T_{\theta\theta} &= T_{xx}\sin^2\theta - T_{xy}\sin 2\theta + T_{yy}\cos^2\theta && (11) \\
T_{r\theta} &= \frac{1}{2}(T_{yy} - T_{xx})\sin 2\theta + T_{xy}\cos 2\theta && (12)
\end{aligned}
$$

This translates as follows modify code !:

```
def convert_tensor_to_polar_coords(theta,Txx,Tyy,Txy):
Trr=Txx*(np.cos(theta))**2 +2*Txy*np.sin(theta)*np.cos(theta) +Tyy*(np.sin(theta))**2
Ttt=Txx*(np.sin(theta))**2 -2*Txy*np.sin(theta)*np.cos(theta) +Tyy*(np.cos(theta))**2
Trt=(Tyy-Txx)*np.sin(theta)*np.cos(theta)+Txy*((np.cos(theta))**2-(np.sin(theta))**2)
return Trr,Ttt,Trt
```

### 7.2.2 Spherical coordinates

In order to compute the dynamic topography we will need $\sigma_{rr}$, which in fact will require $\dot{\varepsilon}_{rr}$. This means that having obtained the strain rate tensor in Cartesian coordinates we must rewrite it in spherical coordinates with the following conventions:




We have FIND SOURCE!!

$$
\begin{aligned}
\boldsymbol{T}_{\text{spherical}} &= \begin{pmatrix} T_{rr} & T_{r\theta} & T_{r\phi} \\ T_{\theta r} & T_{\theta\theta} & T_{\theta\phi} \\ T_{\phi r} & T_{\phi\theta} & T_{\phi\phi} \end{pmatrix} \\
&= \begin{pmatrix} \sin\theta\cos\phi & \sin\theta\sin\phi & \cos\theta \\ \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ -\sin\phi & \cos\phi & 0 \end{pmatrix} \cdot \begin{pmatrix} T_{xx} & T_{xy} & T_{xz} \\ T_{yx} & T_{yy} & T_{yz} \\ T_{zx} & T_{zy} & T_{zz} \end{pmatrix} \cdot \begin{pmatrix} \sin\theta\cos\phi & \cos\theta\cos\phi & -\sin\phi \\ \sin\theta\sin\phi & \cos\theta\sin\phi & \cos\phi \\ \cos\theta & -\sin\theta & 0 \end{pmatrix}
\end{aligned}
$$

In our case, calculations take place in the $(x, z)$-plane so we have $\phi = 0$ and the equation above becomes

$$
\begin{pmatrix} T_{rr} & T_{r\theta} & T_{r\phi} \\ T_{\theta r} & T_{\theta\theta} & T_{\theta\phi} \\ T_{\phi r} & T_{\phi\theta} & T_{\phi\phi} \end{pmatrix} = \begin{pmatrix} \sin\theta & 0 & \cos\theta \\ \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} T_{xx} & T_{xy} & T_{xz} \\ T_{yx} & T_{yy} & T_{yz} \\ T_{zx} & T_{zy} & T_{zz} \end{pmatrix} \cdot \begin{pmatrix} \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \\ \cos\theta & -\sin\theta & 0 \end{pmatrix}
$$

We define $c_\theta = \cos\theta$, $s_\theta = \sin\theta$ so that the following calculations fit on the page:

$$
\begin{aligned}
& \begin{pmatrix} T_{rr} & T_{r\theta} & T_{r\phi} \\ T_{\theta r} & T_{\theta\theta} & T_{\theta\phi} \\ T_{\phi r} & T_{\phi\theta} & T_{\phi\phi} \end{pmatrix} \\
= & \begin{pmatrix} s_\theta & 0 & c_\theta \\ c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} T_{xx} & T_{xy} & T_{xz} \\ T_{yx} & T_{yy} & T_{yz} \\ T_{zx} & T_{zy} & T_{zz} \end{pmatrix} \cdot \begin{pmatrix} s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \\ c_\theta & -s_\theta & 0 \end{pmatrix} \\
= & \begin{pmatrix} s_\theta & 0 & c_\theta \\ c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} T_{xx}s_\theta + T_{xz}c_\theta & T_{xx}c_\theta - T_{xz}s_\theta & T_{xy} \\ T_{yx}s_\theta + T_{yz}c_\theta & T_{yx}c_\theta - T_{yz}s_\theta & T_{yy} \\ T_{zx}s_\theta + T_{zz}c_\theta & T_{zx}c_\theta - T_{zz}s_\theta & T_{zy} \end{pmatrix} \\
= & \begin{pmatrix} T_{xx}s_\theta^2 + T_{xz}s_\theta c_\theta + T_{zx}s_\theta c_\theta + T_{zz}c_\theta^2 & T_{xx}s_\theta c_\theta - T_{xz}s_\theta^2 + T_{zx}c_\theta^2 - T_{zz}s_\theta c_\theta & T_{xy}s_\theta + T_{zy}c_\theta \\ T_{xx}s_\theta c_\theta + T_{xz}c_\theta^2 - T_{zx}s_\theta^2 - T_{zz}s_\theta c_\theta & T_{xx}c_\theta^2 - T_{xz}s_\theta c_\theta - T_{zx}s_\theta c_\theta + T_{zz}s_\theta^2 & T_{xy}c_\theta - T_{zy}s_\theta \\ T_{yx}s_\theta + T_{yz}c_\theta & T_{yx}c_\theta - T_{yz}s_\theta & T_{yy} \end{pmatrix}
\end{aligned}
$$

Using the fact that $\boldsymbol{T}$ is a symmetric tensor:

$$
\begin{pmatrix} T_{rr} & T_{r\theta} & T_{r\phi} \\ T_{\theta r} & T_{\theta\theta} & T_{\theta\phi} \\ T_{\phi r} & T_{\phi\theta} & T_{\phi\phi} \end{pmatrix} = \begin{pmatrix} T_{xx}s_\theta^2 + 2T_{xz}s_\theta c_\theta + T_{zz}c_\theta^2 & (T_{xx} - T_{zz})s_\theta c_\theta + T_{xz}(c_\theta^2 - s_\theta^2) & T_{xy}s_\theta + T_{zy}c_\theta \\ (T_{xx} - T_{zz})s_\theta c_\theta + T_{xz}(c_\theta^2 - s_\theta^2) & T_{xx}c_\theta^2 - 2T_{xz}s_\theta c_\theta + T_{zz}s_\theta^2 & T_{xy}c_\theta - T_{zy}s_\theta \\ T_{yx}s_\theta + T_{yz}c_\theta & T_{yx}c_\theta - T_{yz}s_\theta & T_{yy} \end{pmatrix} \tag{13}
$$

Note that the trace of this tensor is equal to $T_{xx} + T_{yy} + T_{zz}$ (which is an invariant).

As before we can also use $\sin 2x = 2\sin x \cos x$ and $\cos 2x = \cos^2 x - \sin^2 x$ to simplify the expressions above:

$$\begin{pmatrix} T_{rr} & T_{r\theta} & T_{r\phi} \\ T_{\theta r} & T_{\theta\theta} & T_{\theta\phi} \\ T_{\phi r} & T_{\phi\theta} & T_{\phi\phi} \end{pmatrix} = \begin{pmatrix} T_{xx}s_\theta^2 + T_{xz}\sin 2\theta + T_{zz}c_\theta^2 & \frac{1}{2}(T_{xx} - T_{zz})\sin 2\theta + T_{xz}\cos 2\theta & T_{xy}s_\theta + T_{zy}c_\theta \\ \frac{1}{2}(T_{xx} - T_{zz})\sin 2\theta + T_{xz}\cos 2\theta & T_{xx}c_\theta^2 - T_{xz}\sin 2\theta + T_{zz}s_\theta^2 & T_{xy}c_\theta - T_{zy}s_\theta \\ T_{yx}s_\theta + T_{yz}c_\theta & T_{yx}c_\theta - T_{yz}s_\theta & T_{yy} \end{pmatrix} \tag{14}$$

or,

$$T_{rr} = T_{xx}\sin^2\theta + T_{xz}\sin 2\theta + T_{zz}\cos^2\theta \tag{15}$$

$$T_{\theta\theta} = T_{xx}c_\theta^2 - T_{xz}\sin 2\theta + T_{zz}\sin^2\theta \tag{16}$$

$$T_{r\theta} = \frac{1}{2}(T_{xx} - T_{zz})\sin 2\theta + T_{xz}\cos 2\theta \tag{17}$$

which translates as follows in the code modify code:

```
def convert_tensor_to_spherical_coords(theta_polar,Txx,Tzz,Txz):
    theta_sph=np.pi/2-theta_polar
    sin_theta=np.sin(theta_sph)
    cos_theta=np.cos(theta_sph)
    Trr=Txx*sin_theta**2 +2*Txz*sin_theta*cos_theta +Tzz*cos_theta**2
    Ttt=Txx*cos_theta**2 -2*Txz*sin_theta*cos_theta +Tzz*sin_theta**2
    Trt=(Txx-Tzz)*sin_theta*cos_theta +Txz*(cos_theta**2-sin_theta**2)
```

FLAPS:

```
err1[i]=exx1[i]*np.sin(theta[i])**2+2*exz1[i]*np.sin(theta[i])*np.cos(theta[i])+ezz1[i]*np
ett1[i]=exx1[i]*np.cos(theta[i])**2-2*exz1[i]*np.sin(theta[i])*np.cos(theta[i])+ezz1[i]*np
ert1[i]=(exx1[i]-ezz1[i])*np.sin(theta[i])*np.cos(theta[i])+exz1[i]*(-np.sin(theta[i])**2+
```

Remark: We could arrive at $T_{xx}$ also by stating that $T_{rr} = \vec{n}\cdot\boldsymbol{T}\cdot\vec{n}$, where $\vec{n} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$ is the normal to the sphere surface. Since we are computing in the $xz$-plane then $\phi = 0$ and then $\vec{n} = (\sin\theta, 0, \cos\theta)$. The calculation above will yield the same expression for $T_{rr}$ as before.

As mentioned before we assume that deformation takes place in the $x, z$-plane, so that derivatives with respect to $y$ and $v_y$ are automatically zero:

$$\dot{\boldsymbol{\varepsilon}}_{Cart} = \begin{pmatrix} \dot{\varepsilon}_{xx} & 0 & \dot{\varepsilon}_{xz} \\ 0 & \dot{\varepsilon}_{yy} & 0 \\ \dot{\varepsilon}_{xz} & 0 & \dot{\varepsilon}_{zz} \end{pmatrix}$$

so

$$\begin{aligned}
\dot{\boldsymbol{\varepsilon}}_{Sph} = \begin{pmatrix} \dot{\varepsilon}_{rr} & \dot{\varepsilon}_{r\theta} & \dot{\varepsilon}_{r\phi} \\ \dot{\varepsilon}_{\theta r} & \dot{\varepsilon}_{\theta\theta} & \dot{\varepsilon}_{\theta\phi} \\ \dot{\varepsilon}_{\phi r} & \dot{\varepsilon}_{\phi\theta} & \dot{\varepsilon}_{\phi\phi} \end{pmatrix} &= \begin{pmatrix} s_\theta & 0 & c_\theta \\ c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \dot{\varepsilon}_{xx} & 0 & \dot{\varepsilon}_{xz} \\ 0 & \dot{\varepsilon}_{yy} & 0 \\ \dot{\varepsilon}_{xz} & 0 & \dot{\varepsilon}_{zz} \end{pmatrix} \cdot \begin{pmatrix} s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \\ c_\theta & -s_\theta & 0 \end{pmatrix} \\
&= \begin{pmatrix} s_\theta & 0 & c_\theta \\ c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \dot{\varepsilon}_{xx}s_\theta + \dot{\varepsilon}_{xz}c_\theta & \dot{\varepsilon}_{xx}c_\theta - \dot{\varepsilon}_{xz}s_\theta & 0 \\ 0 & 0 & \dot{\varepsilon}_{yy} \\ \dot{\varepsilon}_{xz}s_\theta + \dot{\varepsilon}_{zz}c_\theta & \dot{\varepsilon}_{xz}c_\theta - \dot{\varepsilon}_{zz}s_\theta & 0 \end{pmatrix} \\
&= \begin{pmatrix} \dot{\varepsilon}_{xx}s_\theta^2 + 2\dot{\varepsilon}_{xz}c_\theta s_\theta + \dot{\varepsilon}_{zz}c_\theta^2 & \dot{\varepsilon}_{xx}c_\theta s_\theta - \dot{\varepsilon}_{xz}s_\theta^2 + \dot{\varepsilon}_{xz}c_\theta^2 - \dot{\varepsilon}_{zz}s_\theta c_\theta & 0 \\ \dot{\varepsilon}_{xx}s_\theta c_\theta + \dot{\varepsilon}_{xz}c_\theta^2 - \dot{\varepsilon}_{xz}s_\theta^2 - \dot{\varepsilon}_{zz}c_\theta s_\theta & \dot{\varepsilon}_{xx}c_\theta^2 - 2\dot{\varepsilon}_{xz}s_\theta c_\theta + \dot{\varepsilon}_{zz}s_\theta^2 & 0 \\ 0 & 0 & \dot{\varepsilon}_{yy} \end{pmatrix}
\end{aligned}$$

# 8 Pressure normalisation

When normal velocities are prescribed on all sides of the domain the computed pressure is not unique (nullspace of size 1). One then typically arbitrarily enforce and additional condition on the pressure field (in a postprocessing step), typically 'average surface pressure is zero' or 'volume average is zero'.

In what follows we make the assumption that all elements are identical, i.e. no mesh stretching.

## 8.1 Plane strain - volume average

In this case the volume average is simply:

$$< p >= \frac{\iint p \, dV}{\iint \, dV} = \frac{\sum_e \iint_{\Omega_e} p \, dV}{\sum_e \iint_{\Omega_e} \, dV}$$

which simply translates as follows:

```
pressure_avrg=0
for iel in range(0,nel):
    for iq in range(0,nqel):
        pressure_avrg+=np.dot(N_P[iq,:],p[icon_P[:,iel]])*JxWq[iel,iq]
pressure_avrg/=volume
```

This works equally well for Cartesian and Annulus domains.

## 8.2 Plane strain - surface average

Let us assume that the surface in question is the top surface.

$$< p >= \frac{\int p \, dS}{\int \, dS} = \frac{\sum_{e,s} \int p \, dS}{\sum_{e,s} \int \, dS} \tag{18}$$

where the sum of elements only only runs on those elements touching the surface.

### 8.2.1 Cartesian geometry

If the geometry is Cartesian this sum is trivial and runs over the nelx elements at the top of the domain. The integral over the top edge can be approximated by a 1-point quadrature[6] so that

$$\int_{edge} p \, dS = \frac{x_2 - x_3}{2}(p_2 + p_3) = \frac{h_x}{2}(p_2 + p_3)$$

since the dof layout of the pressure grid is as follows

```
3--2
|  |
0--1
```

In the end we see that the denominator in Eq. (18) is equal to $L_x$ and then

$$< p >= \frac{1}{L_x} \sum_{e,s} \frac{h_x}{2}(p_{e,2} + p_{e,3}) = \frac{1}{nelx} \sum_{e,s} \frac{1}{2}(p_{e,2} + p_{e,3})$$

[6] Pressure is $Q_1$

### 8.2.2 Annulus geometry

We here consider a half-annulus for simplicity. The approach is similar, except that the edge length can be seen as being $R_{outer}h_\theta$ long, where $h_\theta$ is the angular opening of the element, so that

$$\int_{edge} p\ dS = \frac{1}{2}(p_2 + p_3)R_{outer}(\theta_2 - \theta_3)$$

Then in the end, since the denominator in Eq. (18) is equal to $\pi R_{outer}$

$$< p >= \frac{1}{\pi R_{outer}} \sum_{e,s} \frac{R_{outer}}{2}(p_{e,2} + p_{e,3})(\theta_{e,2} - \theta_{e,3}) = \frac{1}{\pi} \sum_{e,s} \frac{1}{2}(p_{e,2} + p_{e,3})(\theta_{e,2} - \theta_{e,3})$$

## 8.3 Axisymmetric

### 8.3.1 Cartesian geometry

Although technically possible, the axisymmetric approximation is not implemented for Cartesian geometries.

### 8.3.2 Annulus geometry

In spherical coordinates, the surface element is $dS = R^2_{outer} \sin\theta_s\ d\theta_s d\phi$ (we here use the subscript 's' to indicate that this is the $\theta$ angle of the spherical coordinates).

$$< p >= \frac{\iint p(\theta_s, \phi) dS}{\iint dS} = \frac{\iint p(\theta_s, \phi) R^2_{outer} \sin\theta_s d\theta_s d\phi}{\iint R^2_{outer} \sin\theta_s d\theta_s d\phi}$$

and since $p$ is independent of $\phi$ then

$$< p >= \frac{2\pi R^2_{outer} \int_0^\pi p(\theta_s) \sin\theta_s d\theta_s}{4\pi R^2_{outer}} = \frac{\int_0^\pi p(\theta_s) \sin\theta_s d\theta_s}{2}$$

The integral over $\theta_s$ can be simplified by using the average pressure along the edge and the angle of the edge middle point:

$$\int_0^\pi p(\theta_s) \sin\theta_s d\theta_s = \sum_{e,s} \int_{\theta_{s,e,3}}^{\theta_{s,e,2}} p(\theta_s) \sin\theta_s d\theta_s \simeq \sum_{e,s} \frac{1}{2}(p_{e,2} + p_{3,e}) \sin\frac{\theta_{s,e,2} + \theta_{s,e,3}}{2}(\theta_{s,e,2} - \theta_{s,e,3})$$

Remember that the sum runs over all elements ('e') at the surface ('s'). In the end

$$< p >= \frac{1}{2} \sum_{e,s} \frac{1}{2}(p_{e,2} + p_{3,e}) \sin\frac{\theta_{s,e,2} + \theta_{s,e,3}}{2}(\theta_{s,e,2} - \theta_{s,e,3})$$

Note again that this is valid for $\theta_s$, not $\theta_p$, so that we will need to convert to $\theta_s$ in the code:

$$\frac{\theta_{s,e,2} + \theta_{s,e,3}}{2} = \frac{1}{2}(\frac{\pi}{2} - \theta_{p,e,2} + \frac{\pi}{2} - \theta_{p,e,3}) = \frac{\pi}{2} - \frac{1}{2}(\theta_{p,e,2} + \theta_{p,e,3})$$

and

$$\theta_{s,e,2} - \theta_{s,e,3} = \frac{\pi}{2} - \theta_{p,e,2} - \frac{\pi}{2} + \theta_{p,e,3} = \theta_{p,e,3} - \theta_{p,e,2}$$

# 9 Generic mass, momentum, energy conservation equations

We focus on the system of equations in a $d = 2$- or $d = 3$-dimensional domain $\Omega$ that describes the motion of a highly viscous fluid (i.e. near infinite Prandlt number) driven by differences in the gravitational force due to a density variations. In the following, we largely follow the exposition of this material in Schubert, Turcotte and Olson Schubert et al. 2001.

$$\boldsymbol{\sigma} = -p\,\mathbf{1} + \boldsymbol{\tau}$$
$$\boldsymbol{\tau} = 2\eta\dot{\boldsymbol{\varepsilon}}^d(\vec{\mathbf{v}}) = 2\eta\left(\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) - \frac{1}{3}(\vec{\nabla}\cdot\vec{\mathbf{v}})\mathbf{1}\right)$$

Specifically, we consider the following set of equations for velocity $\vec{\mathbf{v}}$, pressure $p$ and temperature $T$:

$$-\vec{\nabla}p + \vec{\nabla}\cdot\left[2\eta\left(\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) - \frac{1}{3}(\vec{\nabla}\cdot\vec{\mathbf{v}})\mathbf{1}\right)\right] + \rho\vec{g} = \vec{0} \qquad \text{in } \Omega, \tag{19}$$

$$\frac{\partial\rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{\mathbf{v}}) = 0 \qquad \text{in } \Omega, \tag{20}$$

$$\begin{aligned}\rho C_p\left(\frac{\partial T}{\partial t} + \vec{\mathbf{v}}\cdot\vec{\nabla}T\right) - \vec{\nabla}\cdot k\vec{\nabla}T &= \rho H \\ &\quad + 2\eta\left(\dot{\varepsilon}(\vec{\mathbf{v}}) - \frac{1}{3}(\vec{\nabla}\cdot\vec{\mathbf{v}})\mathbf{1}\right) : \left(\dot{\varepsilon}(\vec{\mathbf{v}}) - \frac{1}{3}(\vec{\nabla}\cdot\vec{\mathbf{v}})\mathbf{1}\right) \qquad (21)\\ &\quad + \alpha T\left(\frac{\partial p}{\partial t} + \vec{\mathbf{v}}\cdot\vec{\nabla}p\right) \qquad \text{in } \Omega, \end{aligned} \tag{22}$$

where $\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) = \frac{1}{2}(\vec{\nabla}\vec{\mathbf{v}} + \vec{\nabla}\vec{\mathbf{v}}^T)$ is the symmetric gradient of the velocity (often called the *strain rate* tensor).

In this set of equations, (19) and (20) represent the compressible Stokes equations in which $\vec{\mathbf{v}} = \vec{\mathbf{v}}(\mathbf{x}, t)$ is the velocity field and $p = p(\mathbf{x}, t)$ the pressure field. Both fields depend on space $\mathbf{x}$ and time $t$. Fluid flow is driven by the gravity force that acts on the fluid and that is proportional to both the density of the fluid and the strength of the gravitational pull.

Coupled to this Stokes system is equation (21) for the temperature field $T = T(\mathbf{x}, t)$ that contains heat conduction terms as well as advection with the flow velocity $\vec{\mathbf{v}}$. The right hand side terms of this equation correspond to

- internal heat production for example due to radioactive decay;
- friction (shear) heating;
- adiabatic compression of material;

### 9.0.1 A word about the shear heating term

It is defined as

$$\Phi = 2\eta\left(\dot{\varepsilon}(\vec{\mathbf{v}}) - \frac{1}{3}(\vec{\nabla}\cdot\vec{\mathbf{v}})\mathbf{1}\right) : \left(\dot{\varepsilon}(\vec{\mathbf{v}}) - \frac{1}{3}(\vec{\nabla}\cdot\vec{\mathbf{v}})\mathbf{1}\right)$$

We start from the 3d strain rate tensor

$$\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) = \begin{pmatrix} \dot{\varepsilon}_{xx} & \dot{\varepsilon}_{xy} & \dot{\varepsilon}_{xz} \\ \dot{\varepsilon}_{yx} & \dot{\varepsilon}_{yy} & \dot{\varepsilon}_{yz} \\ \dot{\varepsilon}_{zx} & \dot{\varepsilon}_{zy} & \dot{\varepsilon}_{zz} \end{pmatrix}$$

The plane strain assumption is such that the problem at hand is assumed to be infinite in a given direction. In the case of computational geodynamics, most 2D modelling is a vertical section of the crust-lithosphere-mantle and the underlying implicit assumption is then that the orogen/rift/subduction/etc ... is infinite in the direction perpendicular to the screen/paper.

Let us assume that the deformation takes place in the $x, z$-plane, so that $v = 0$ (velocity in the $y$ direction is zero) and $\partial_y \to 0$ (no change in the $y$ direction). We then have $\dot{\varepsilon}_{yy} = 0$ as well as $\dot{\varepsilon}_{xy} = 0$ and $\dot{\varepsilon}_{yz} = 0$, so that the strain rate tensor is

$$\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) = \begin{pmatrix} \dot{\varepsilon}_{xx} & 0 & \dot{\varepsilon}_{xz} \\ 0 & 0 & 0 \\ \dot{\varepsilon}_{xz} & 0 & \dot{\varepsilon}_{zz} \end{pmatrix}$$

where we have also used the fact that the tensor is symmetric. The deviatoric strain rate tensor in plane strain is given by

$$\dot{\boldsymbol{\varepsilon}}^d(\vec{\mathbf{v}}) = \dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) - \frac{1}{3}\mathrm{tr}[\dot{\boldsymbol{\varepsilon}}]\mathbf{1} = \dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) - \frac{1}{3}(\dot{\varepsilon}_{xx} + \dot{\varepsilon}_{zz})\mathbf{1} = \begin{pmatrix} \frac{2}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz} & \dot{\varepsilon}_{xy} & 0 \\ \dot{\varepsilon}_{yx} & -\frac{1}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz} & 0 \\ 0 & 0 & -\frac{1}{3}\dot{\varepsilon}_{xx} + \frac{2}{3}\dot{\varepsilon}_{zz} \end{pmatrix}$$

We then have

$$\begin{aligned}
& \dot{\boldsymbol{\varepsilon}}^d(\vec{\mathbf{v}}) : \dot{\boldsymbol{\varepsilon}}^d(\vec{\mathbf{v}}) \\
= & \begin{pmatrix} \frac{2}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz} & \dot{\varepsilon}_{xy} & 0 \\ \dot{\varepsilon}_{yx} & -\frac{1}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz} & 0 \\ 0 & 0 & -\frac{1}{3}\dot{\varepsilon}_{xx} + \frac{2}{3}\dot{\varepsilon}_{zz} \end{pmatrix} : \begin{pmatrix} \frac{2}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz} & \dot{\varepsilon}_{xy} & 0 \\ \dot{\varepsilon}_{yx} & -\frac{1}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz} & 0 \\ 0 & 0 & -\frac{1}{3}\dot{\varepsilon}_{xx} + \frac{2}{3}\dot{\varepsilon}_{zz} \end{pmatrix} \\
= & \left(\frac{2}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz}\right)^2 + \left(-\frac{1}{3}\dot{\varepsilon}_{xx} - \frac{1}{3}\dot{\varepsilon}_{zz}\right)^2 + \left(-\frac{1}{3}\dot{\varepsilon}_{xx} + \frac{2}{3}\dot{\varepsilon}_{zz}\right)^2 + 2\dot{\varepsilon}_{xz}^2 \\
= & (\frac{4}{9} + \frac{1}{9} + \frac{1}{9})\dot{\varepsilon}_{xx}^2 + (\frac{1}{9} + \frac{4}{9} + \frac{1}{9})\dot{\varepsilon}_{zz}^2 + (-\frac{2}{9} - \frac{2}{9} + \frac{1}{9})\dot{\varepsilon}_{xx}\dot{\varepsilon}_{zz} + 2\dot{\varepsilon}_{xz}^2 \\
= & \frac{2}{3}\dot{\varepsilon}_{xx}^2 + \frac{2}{3}\dot{\varepsilon}_{zz}^2 - \frac{1}{3}\dot{\varepsilon}_{xx}\dot{\varepsilon}_{zz} + 2\dot{\varepsilon}_{xz}^2
\end{aligned} \tag{23}$$

This translates as follows in the code:

```
b_el[:]+=N[:]*JxWq[iq]*2*etaq[iel,iq]*\
        (2./3.*(exxq[iel,iq]**2+ezzq[iel,iq]**2)\
        -exxq[iel,iq]*ezzq[iel,iq]/3+2*exzq[iel,iq]**2)
```

# 10 Equation of state

The equation of state gives the density as a function of the pressure and temperature: $\rho = \rho(p, T)$. The density is then either obtained from precomputed lookup tables or a simpler functional approach is often taken by means of a linearisation.

The widely used Boussinesq approximation (see below) linearizes these basic conservation laws near the reference hydrostatic state. If density changes caused by the pressure deviations $p' = p - p_0$ are neglected, we may linearize the state equation with respect to the temperature deviations $T - T_0$, where $T_0$ is a reference temperature, and write:

$$\rho = \rho_0(1 - \alpha(T - T_0))$$

This approximation thus means that the influence of hydrostatic pressure (as well as temperature $T_0$) on density is hidden in a spatial dependence of the reference density $\rho_0$.

The reference density $\rho_0$ is assumed to be a time-independent function. Considering only the largest term in the equation of continuity, that is, neglecting thermal expansion, we arrive at the simplified equation:

$$\vec{\nabla} \cdot (\rho_0 \vec{\mathbf{v}}) = 0$$

# 11 Anelastic liquid approximation (ALA)

This comes from Matyska et al. (2007).

If we assume that there is a reference hydrostatic state characterized by $\vec{\mathbf{v}} = 0$ in which the hydrostatic pressure $p_0$, hydrostatic density $\rho_0$, and hydrostatic gravity acceleration $g_0$ are related by $\vec{\nabla} p_0 = \rho_0 \vec{g}_0$, and moreover that pressure deviations $p' = p - p_0$ are negligible in the heat equation, the transfer of heat in a homogeneous material (i.e., entropy may be considered as a function of only p and T) is then described by the well-known equation:

$$\rho C_p \left( \frac{\partial T}{\partial t} + \vec{\mathbf{v}} \cdot \vec{\nabla} T \right) = \vec{\nabla} \cdot k \vec{\nabla} T - \alpha T \vec{\mathbf{v}} \cdot \rho \vec{g} + \boldsymbol{\tau} : \vec{\nabla} \vec{\mathbf{v}} + Q$$

where $C_p$ is the isobaric specific heat, $\alpha$ is the thermal expansion coefficient and vr denotes the radial component of velocity. The left-hand side of equation 8 represents local changes of heat balance; the second (third) term on the right-hand side describes ad- vection of heat (adiabatic heating and/or cooling).

# 12 Boussinesq approximation (BA)

In the case of an incompressible flow, then $\partial\rho/\partial t = 0$ and $\vec{\nabla}\rho = 0$, i.e. $D\rho/Dt = 0$ and the mass conservation equation becomes:

$$\vec{\nabla} \cdot \vec{\mathbf{v}} = 0$$

A vector field that is divergence-free is also called solenoidal[7].

In this case the equations above can now be written

$$-\vec{\nabla}p + \vec{\nabla} \cdot [2\eta\,(\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}))] + \rho\vec{g} = \vec{0} \qquad \text{in } \Omega, \tag{24}$$

$$\vec{\nabla} \cdot \vec{\mathbf{v}} = 0 \qquad \text{in } \Omega, \tag{25}$$

$$\rho C_p \left( \frac{\partial T}{\partial t} + \vec{\mathbf{v}} \cdot \vec{\nabla} T \right) - \vec{\nabla} \cdot k\vec{\nabla}T = \rho H + 2\eta\dot{\varepsilon}(\vec{\mathbf{v}}) : \dot{\varepsilon}(\vec{\mathbf{v}}) + \alpha T \left( \frac{\partial p}{\partial t} + \vec{\mathbf{v}} \cdot \vec{\nabla} p \right) \qquad \text{in } \Omega, \tag{26}$$

As nicely explained in Spiegel et al. (1960):

> In the study of problems of thermal convection it is a frequent practice to simplify the basic equations by introducing certain approximations which are attributed to Boussinesq (1903). The Boussinesq approximations can best be summarized by two statements:
>
> 1. The fluctuations in density which appear with the advent of motion result principally from thermal (as opposed to pressure) effects.
> 2. In the equations for the rate of change of momentum and mass, density variations may be neglected except when they are coupled to the gravitational acceleration in the buoyancy force."

Note that their paper examines the Boussinesq approximation for compressible fluids.

The Boussinesq approximation assumes that the density can be considered constant in all occurrences in the equations with the exception of the buoyancy term on the right hand side of (19). The primary result of this assumption is that the continuity equation (20) will now read $\vec{\nabla} \cdot \vec{\mathbf{v}} = 0$. This implies that the strain rate tensor is deviatoric. Under the Boussinesq approximation, the equations are much simplified:

$$-\vec{\nabla}p + \vec{\nabla} \cdot [2\eta\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}})] + \rho\vec{g} = \vec{0} \qquad \text{in } \Omega, \tag{27}$$

$$\vec{\nabla} \cdot \vec{\mathbf{v}} = 0 \qquad \text{in } \Omega, \tag{28}$$

$$\rho_0 C_p \left( \frac{\partial T}{\partial t} + \vec{\mathbf{v}} \cdot \vec{\nabla} T \right) - \vec{\nabla} \cdot k\vec{\nabla}T = \rho H \qquad \text{in } \Omega \tag{29}$$

Note that all terms on the rhs of the temperature equations have disappeared, with the exception of the source term.

In Zelst et al. (2022) we read:

> The Boussinesq approximation (Oberbeck (1879); Boussinesq, 1903; Rayleigh, 1916) assumes that density variations are so small that they can be neglected everywhere except in the buoyancy term in the momentum equation, which is equivalent to using a constant reference density profile. This implies incompressibility [...]. In addition, adiabatic heating and shear heating are not considered in the energy equation. This approximation is valid as long as density variations are small and the modelled processes would cause no

[7] `https://en.wikipedia.org/wiki/Solenoidal_vector_field`

substantial shear or adiabatic heating. The Boussinesq approximation is often used in lithosphere- scale models. Due to its simplicity, the approximation of in- compressibility is sometimes also adopted for whole-mantle convection models, wherein it is only approximately valid, and it has been shown that compressibility can have a large effect on the pattern of convective flow Tackley 1996.

# 13 Extended Boussinesq approximation (EBA)

In Zelst et al. (2022) we read:

> The extended Boussinesq approximation (Christensen et al. 1985; Oxburgh et al. 1978) is based on the same assumptions as the BA but does consider adiabatic and shear heating. Since it includes adiabatic heating, but not the associated volume and density changes, it can lead to artificial changes of energy in the model, i.e. material is being heated or cooled based on the assumption that it is compressed or it expands, but the mechanical work that causes compression or expansion is not done. Consequently, the extended Boussinesq approximation should only be used in models without substantial adiabatic temperature changes.
>
> For a comparison between some of these approximations using benchmark models, see e.g. Steinbach et al. (1989), Leng et al. (2008), King et al. (2010), Gassmöller, Dannberg, et al. (2020). In addition, the choice of approximation may also be limited by the numerical methods being employed (for example, the accuracy of the solution for the variables that affect the density). Also note that, technically, these approximations are all internally inconsistent to varying degrees, since they do not fulfil the definitions of thermodynamic variables but use linearised versions instead, and they use different density formulations in the different equations. Nevertheless, many of them are generally accepted and widely used in geodynamic modelling studies, as they allow for simpler equations and more easily obtained solutions.

# 14 Computing the lithostatic pressure

# 15 Initial adiabatic temperature profile

# 16 Dynamic topography

# 17 Notes

## 17.1 Plane strain case

```
ggx=0.
ggy=0.
for iel in range(0,nel):
    xx=xs-xc[iel]
    yy=ys-yc[iel]
    rr=np.sqrt(xx**2+yy**2)
    ggx+=Ggrav*(rho[iel]-rho_ref)*vol[iel]*xx/rr**3
    ggy+=Ggrav*(rho[iel]-rho_ref)*vol[iel]*yy/rr**3
gnorm=np.sqrt(ggx**2+ggy**2)
```

## 17.2 Axisymmetric case

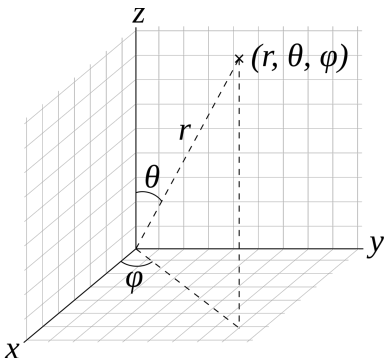
Although the FEM calculations take place in the $xz$-plane, we need to think of the system as a 3d one, i.e. the half annulus mass distribution is repeated from $\phi = 0$ to $\phi = 2\pi$. Each element we consider must be seen as a torus whose cross section is the element under consideration.

Let us assume that the satellite is at position $(x_s, y_s, z_s)$. Let us for now isolate a single element $e$ of surface $S_e$ and $\phi-$angular opening $\delta\phi$, and density $\rho_e$ with its center of mass at position $(x_e, y_e, z_e)$). The gravity that the element mass exerts on the satellite is a vector with components

$$g_x(x_s, y_s, z_s, e) = \mathcal{G}\frac{S_e\rho_e\delta\phi}{[(x_s-x_e)^2+(y_s-y_e)^2+(z_s-z_e)^2]^{3/2}}(x_s-x_e) \tag{30}$$

$$g_y(x_s, y_s, z_s, e) = \mathcal{G}\frac{S_e\rho_e\delta\phi}{[(x_s-x_e)^2+(y_s-y_e)^2+(z_s-z_e)^2]^{3/2}}(y_s-y_e) \tag{31}$$

$$g_z(x_s, y_s, z_s, e) = \mathcal{G}\frac{S_e\rho_e\delta\phi}{[(x_s-x_e)^2+(y_s-y_e)^2+(z_s-z_e)^2]^{3/2}}(z_s-z_e) \tag{32}$$




We can rewrite the position of the element center in spherical coordinates ollows:

$$x_e = r_e\sin\theta_e\cos\phi_e \tag{33}$$

$$y_e = r_e\sin\theta_e\sin\phi_e \tag{34}$$

$$z_e = r_e\cos\theta_e \tag{35}$$

In this case $\theta \in [0:\pi]$ and $\phi \in ]-\pi:\pi]$ and we have the following relationships: Let us focus on $g_x$ for now.

$$\begin{aligned} g_x(x_s, y_s, z_s, e) &= \mathcal{G}\frac{S_e\rho_e\delta\phi}{[(x_s-x_e)^2+(y_s-y_e)^2+(z_s-z_e)^2]^{3/2}}(x_s-x_e) \\ &= \mathcal{G}\frac{S_e\rho_e\delta\phi}{[(x_s-r_e\sin\theta_e\cos\phi_e)^2+(y_s-r_e\sin\theta_e\sin\phi_e)^2+(z_s-r_e\cos\theta_e)^2]^{3/2}}(x_s-r_e\sin\theta_e\cos\phi_e) \end{aligned}$$

At this stage we must acknowledge that inside the torus $r_e$ and $\theta_e$ are constant and only the $\phi_e$ value changes. Since the position of the satellite is assumed to be fixed for now then the last term of the denominator $(z_s - r_e \cos\theta_e)$ is then a constant. Likewise, since the domain is a half-annulus and we only consider the center of mass of element then we have $r_e > 0$ and $\theta_e > 0$ and $\theta_e < \pi$ so that $r_e \sin\theta_e > 0$. We can then define:

$$x'_s = \frac{x_s}{r_e \sin\theta_e} \qquad y'_s = \frac{y_s}{r_e \sin\theta_e} \qquad z'_s = \frac{z_s}{r_e \sin\theta_e}$$

We can then write:

$$g_x(x_s, y_s, z_s, e) = \int_0^{2\pi} \frac{\mathcal{G}}{r_e \sin\theta_e} \frac{S_e \rho_e}{[(x'_s - \cos\phi)^2 + (y'_s - \sin\phi)^2 + (z'_s - \tan^{-1}\theta_e)^2]^{3/2}} (x'_s - \cos\phi) d\phi$$

We then find ourselves having to compute and integral of the form:

$$\int_0^{2\pi} \frac{a - \cos\phi}{[(a - \cos\phi)^2 + (b - \sin\phi)^2 + c^2]^{3/2}} d\phi$$

In Bull et al. 2014 we find

"We employ a temperature- and depth-dependent rheology of the non-dimensional form:

$$\eta(T, z) = \eta_r(z) \exp(A(0.5 - T))$$

where $\eta_r(z) = 1$ for $z < 663$km and $\eta_r(z) = 0.1225z - 51.2$ for $663 \leq z \leq 2867$km. $\eta$ and $z$ are the non-dimensional viscosity and dimensional depth respectively. [...] it leads to a weak upper mantle, a $30\times$ viscosity step at the boundary between the upper and lower mantle, and a $10\times$ linear increase with depth to the base of the mantle. The non-dimensional activation coefficient is chosen to be $A = 9.2103$, which leads to a temperature-induced viscosity contrast of $10^4$. "

This is a beautiful example of reviewing/editing failure. The above equation is clearly formulated for dimensionless quantities, but the $\eta_r(z) = 0.1225z - 51.2$ only makes sense if $z$ is expressed in km. Likewise the value 2867km is surprisingly off with regards to the agreed upon cmb depth of about 2890km. Finally the real outer radius of the domain is never specified, and neither are the temperature boundary conditions.

At 663km depth we find $\eta_r(z) = 0.1225 * 663 - 51.2 = 30$ and at 2867km depth we find $\eta_r(z) = 0.1225 * 2867 - 51.2 = 300$, so this adds up with respect to the text.

In the end, if we want to use this rheology in an Earth-dimensioned model we will need to use

$$\eta(T, z) = \eta_0 \eta_r(z) \exp(A(0.5 - T))$$

where $\eta_0 \sim 10^{21}$ .

# 18 TO DO

- which is the correct way of computing DT? remove average stress, or removing average pressure only?
- test
- vrms in axisymmetry
- check dev strain rate in axisymm
- top_element vs top_nodeS
- when element is empty , stop AND create vtu file
- run schmeling subduction -
- if ss detected make sure vtus are produced

# References


Bull, A.L. et al. (2014). "The effect of plate motion history on the longevity of deep mantle heterogeneities". In: *Earth Planet. Sci. Lett.* 401, pp. 172–182. DOI: 10.1016/j.epsl.2014.06.008.

Christensen, Ulrich R et al. (1985). "Layered convection induced by phase transitions". In: *J. Geophys. Res.: Solid Earth* 90.B12, pp. 10291–10300. DOI: 10.1029/JB090iB12p10291.

Gassmöller, Rene, Juliane Dannberg, et al. (2020). "On formulations of compressible mantle convection". In: *Geophy. J. Int.* 221.2, pp. 1264–1280. DOI: 10.1093/gji/ggaa078.

Gassmöller, Rene, Harsha Lokavarapu, et al. (2019). "Evaluating the accuracy of hybrid finite element/particle-in-cell methods for modelling incompressible Stokes flow". In: *Geophy. J. Int.* 219.3, pp. 1915–1938. DOI: 10.1093/gji/ggz405.

King, S. et al. (2010). "A community benchmark for 2D Cartesian compressible convection in the Earth's mantle". In: *Geophy. J. Int.* 180, pp. 73–87.

Leng, W. et al. (2008). "Viscous heating, adiabatic heating and energetic consistency in compressible mantle convection". In: *Geophy. J. Int.* 173, pp. 693–702. DOI: 10.1111/j.1365-246X.2008.03745.x.

Matyska, Ctirad et al. (2007). "Lower-mantle material properties and convection models of multiscale plumes". In: *Special Papers – Geological Society of America* 430, p. 137.

Oberbeck, Anton (1879). "Über die Wärmeleitung der Flüssigkeiten bei Berücksichtigung der Strömungen infolge von Temperaturdifferenzen". In: *Annalen der Physik* 243.6, pp. 271–292.

Oxburgh, E R et al. (1978). "Mechanisms of continental drift". In: *Reports on Progress in Physics* 41.8, p. 1249. DOI: 10.1088/0034-4885/41/8/003.

Schubert, G. et al. (2001). *Mantle Convection in the Earth and Planets*. Cambridge University Press. ISBN: 0-521-70000-0. DOI: 10.1017/CBO9780511612879.

Spiegel, Edward A et al. (1960). "On the Boussinesq approximation for a compressible fluid." In: *The Astrophysical Journal* 131, p. 442.

Steinbach, Volker et al. (1989). "Compressible convection in the earth's mantle: a comparison of different approaches". In: *Geophys. Res. Lett.* 16.7, pp. 633–636. DOI: 10.1029/GL016i007p00633.

Tackley, P.J. (1996). "Effects of strongly variable viscosity on three-dimensional compressible convection in planetary mantles". In: *J. Geophys. Res.: Solid Earth* 101.B2, pp. 3311–3332.

Thielmann, M. et al. (2014). "Discretization errors in the Hybrid Finite Element Particle-In-Cell Method". In: *Pure Appl. Geophys.* 171.9, pp. 2164–2184. DOI: 10.1007/s00024-014-0808-9.

Thieulot, C. et al. (2022). "On the choice of finite element for applications in geodynamics". In: *Solid Earth* 13, pp. 229–249. DOI: 10.5194/se-13-229-2022.

— (2025). "On the choice of finite element for applications in geodynamics. Part II: A comparison of simplex and hypercube elements". In: *Solid Earth* 16, pp. 457–476. DOI: 10.5194/se-16-457-2025.

Zelst, Iris van et al. (2022). "101 Geodynamic modelling: How to design, interpret, and communicate numerical studies of the solid Earth". In: *Solid Earth* 13, pp. 583–637. DOI: 10.5194/se-13-583-2022.