FLAPS
the FLexible Axisymmetric Planet Solver

C. Thieulot

November 7, 2023

Utrecht University

Contents

1 Introduction **3**

2 The physics **4**
2.1 About the gravity field 4
2.2 Axisymmetric formulation 5
2.3 Tensors from Cartesian to Spherical coordinates 6
2.4 Dynamic topography 7

3 Numerical methods **9**
3.1 Finite elements 9
3.2 Nomenclature 9
3.3 Mapping 10
3.4 Quadrature 11
3.5 Computing normals 11
3.6 Solving the linear system 13
3.7 Computing velocity derivatives 13
3.8 Implementing free slip boundary conditions 14
3.9 Pressure normalisation 14
3.10 Code structure 16

4 The data **17**
4.1 Earth 17
4.2 Mars 17

5 Running the code **18**

6 Benchmarking **19**
6.1 Computing volume/mass 19
6.2 Annulus convection manufactured solution 20
6.2.1 Timings 20

7 The '4D dynamic earth' inter-code benchmark **20**

A Misc **21**
A.1 Notes to self 21
A.2 To do list 21

1 Introduction

This code first started as a stone in the fieldstone project [9]. As it was growing and it appeared that it would be used for different projects I decided to take it out and turn it into a standalone code with its own manual.

The code only supports one type of Finite Elements, the $Q_2 \times Q_1$ pair, which is stable and commonly used in geodynamics [8].

2 The physics

The domain is a spherical shell centered around the origin of the Cartesian axis system (x, y, z). The inner radius is denoted by R_1 and the outer radius by R_2. The Stokes equations are solved in the domain and the flow is assumed to be instantaneous, incompressible and isothermal.

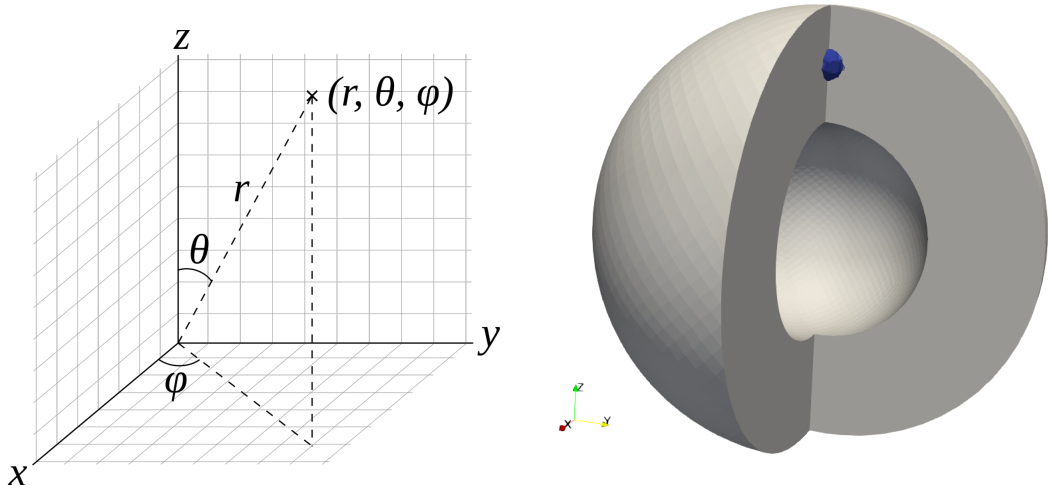
The velocity in Cartesian coordinates is denoted by $\vec{v} = (v_x, v_y, v_z)$ and $\vec{v} = (v_r, v_\theta, v_\phi)$ in spherical coordinates. The equations to solve are then

$$-\vec{\nabla}p + \vec{\nabla} \cdot (2\eta\dot{\vec{\epsilon}}(\vec{v})) + \rho\vec{g} = 0 \quad (1)$$

$$\vec{\nabla} \cdot \vec{v} = 0 \quad (2)$$

The domain has two boundaries: the inner boundary $r = R_1$ and the outer boundary at $r = R_2$. Boundary conditions can be either no slip, free slip or open ('free surface').

The domain is filled with a fluid (the mantle 'm') of viscosity $\eta_m(\vec{r})$ and density $\rho_m(\vec{r})$. A sphere ('s') of a different fluid or radius R_s is placed at location $(0, 0, z_s)$ with viscosity η_s and density ρ_s. As such the setup is axisymmetric along the North-South poles line. The gravity $\vec{g} = -g\vec{e}_r$ points towards the center of the shell[1].



Models with free slip boundary conditions on both inner and outer surfaces have a rotational nullspace which needs to be removed.

In such case the pressure also showcases a constant value nullspace which can be removed by ensuring that the average surface pressure is zero.

2.1 About the gravity field

We need to specify the gravity acceleration vector $\vec{g}$ inside the domain. We have several option.

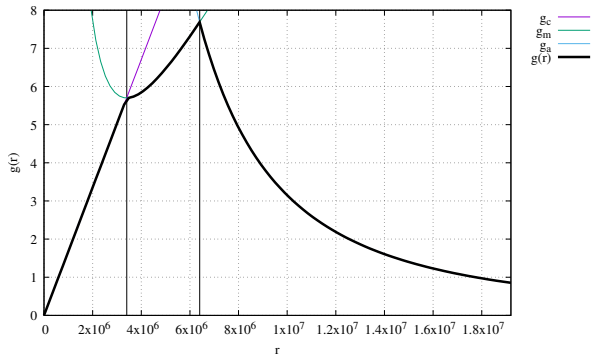
- the simplest option is to set $\vec{g} = -g_0\vec{e}_r$ with g_0 being a constant (e.g. $9.8\ \mathrm{m\,s^{-1}}$). This is gravity_model=0
- We can assume the domain to be spherically symmetric: the core has a density ρ_c and the mantle has a density ρ_m. In that case we have In the end we have:

$$g_c(r) = \frac{4\pi}{3}\mathcal{G}\rho_c r$$

$$g_m(r) = \frac{4\pi}{3}\mathcal{G}\rho_m r + \frac{C}{r^2} \qquad C = \frac{4\pi R_1^3}{3}\mathcal{G}(\rho_c - \rho_m) \quad (3)$$

[1] unless self gravitation is used

In the core we have $\vec{g}_c = -g_c(r)\vec{e}_r$ and in the mantle $\vec{g}_m = -g_m(r)\vec{e}_r$.



This is gravity_model=1

- This is no more an axisymmetric case. We have the same structure (core and mantle) but there is now a sphere ('blob') of density ρ_{blob} at location $x = 0$ $z = z_{blob}$ of radius R_{blob}.

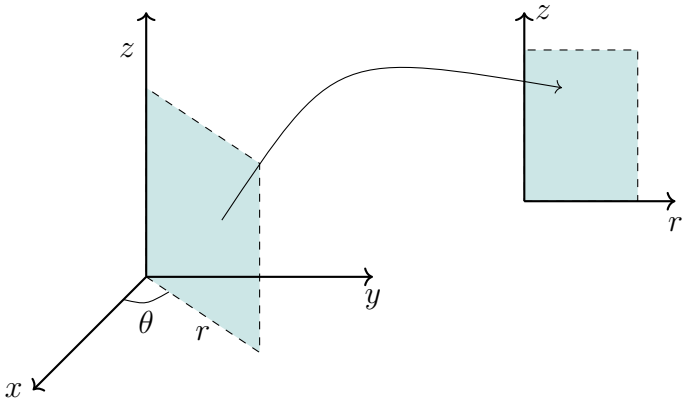
  The gravity field is then the sum of the spherically symmetric one above and the one generated by the density anomaly in the blob.

  This is gravity_model=2

2.2 Axisymmetric formulation

We here rely on axisymmetric cylindrical coordinates, see Section **??**. As shown on the following figure we assume that the deformation/flow is independent of the angle θ so that the remaining space coordinates are r and z.

(tikz_axi.tex)



Given the symmetry of the problem any term containing ∂_θ or $\mathbf{v}_\theta$ is zero. The strain rate tensor given in Section **??** then simplifies to:

$$
\begin{aligned}
\dot{\varepsilon}_{rr} &= \frac{\partial \mathbf{v}_r}{\partial r} \\
\dot{\varepsilon}_{\theta\theta} &= \frac{\mathbf{v}_r}{r} \\
\dot{\varepsilon}_{\theta r} = \dot{\varepsilon}_{r\theta} &= 0 \\
\dot{\varepsilon}_{zz} &= \frac{\partial \mathbf{v}_z}{\partial z} \\
\dot{\varepsilon}_{rz} = \dot{\varepsilon}_{zr} &= \frac{1}{2}\left(\frac{\partial \mathbf{v}_r}{\partial z} + \frac{\partial \mathbf{v}_z}{\partial r}\right) \\
\dot{\varepsilon}_{\theta z} = \dot{\varepsilon}_{z\theta} &= 0
\end{aligned}
$$

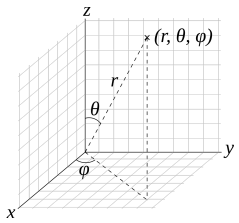
or,

$$\dot{\boldsymbol{\varepsilon}}(\vec{\mathbf{v}}) = \begin{pmatrix} \dot{\varepsilon}_{rr} & 0 & \dot{\varepsilon}_{rz} \\ 0 & \dot{\varepsilon}_{\theta\theta} & 0 \\ \dot{\varepsilon}_{zr} & 0 & \dot{\varepsilon}_{zz} \end{pmatrix}$$

Note that the term $\dot{\varepsilon}_{\theta\theta}$ is not zero! The deviatoric stress tensor $\boldsymbol{\tau} = 2\eta\dot{\boldsymbol{\varepsilon}}$ can be computed as well as the full stress tensor $\boldsymbol{\sigma} = -p\mathbf{1} + \boldsymbol{\tau}$.

2.3 Tensors from Cartesian to Spherical coordinates

In order to compute the dynamic topography we will need σ_{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

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

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

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

As we have seen above we have

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

so

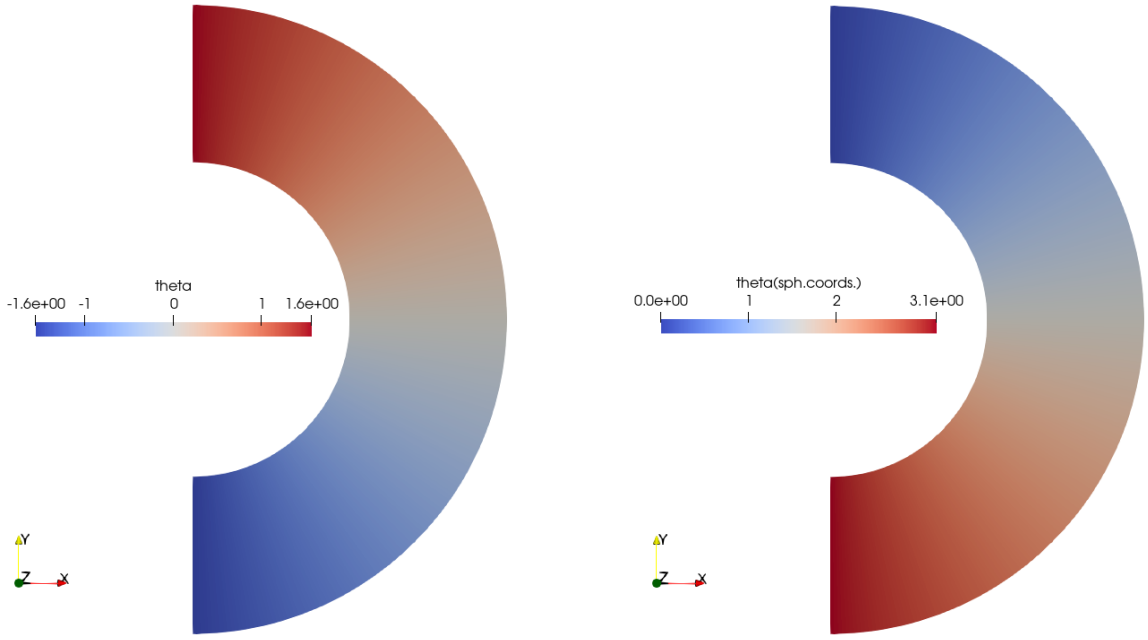
$$\dot{\boldsymbol{\varepsilon}}_{Sph} = \begin{pmatrix} \dot{\varepsilon}_{rr} & \dot{\varepsilon}_{r\theta} & \dot{\varepsilon}_{r\phi} \\ \dot{\varepsilon}_{\theta r} & \dot{\varepsilon}_{\theta\theta} & \dot{\varepsilon}_{\theta\phi} \\ \dot{\varepsilon}_{\phi r} & \dot{\varepsilon}_{\phi\theta}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

$$= \begin{pmatrix} s_\theta & 0 & c_\theta \\ c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \dot{\varepsilon}_{xx}s_\theta + \dot{\varepsilon}_{xz}c_\theta & \dot{\varepsilon}_{xx}c_\theta - \dot{\varepsilon}_{xz}s_\theta & 0 \\ 0 & 0 & \dot{\varepsilon}_{yy} \\ \dot{\varepsilon}_{xz}s_\theta + \dot{\varepsilon}_{zz}c_\theta & \dot{\varepsilon}_{xz}c_\theta - \dot{\varepsilon}_{zz}s_\theta & 0 \end{pmatrix}$$

$$= \begin{pmatrix} \dot{\varepsilon}_{xx}s_\theta^2 + 2\dot{\varepsilon}_{xz}c_\theta s_\theta + \dot{\varepsilon}_{zz}c_\theta^2 & \dot{\varepsilon}_{xx}c_\theta s_\theta - \dot{\varepsilon}_{xz}s_\theta^2 + \dot{\varepsilon}_{xz}c_\theta^2 - \dot{\varepsilon}_{zz}s_\theta c_\theta & 0 \\ \dot{\varepsilon}_{xx}s_\theta c_\theta + \dot{\varepsilon}_{xz}c_\theta^2 - \dot{\varepsilon}_{xz}s_\theta^2 - \dot{\varepsilon}_{zz}c_\theta s_\theta & \dot{\varepsilon}_{xx}c_\theta^2 - 2\dot{\varepsilon}_{xz}s_\theta c_\theta + \dot{\varepsilon}_{zz}s_\theta^2 & 0 \\ 0 & 0 & \dot{\varepsilon}_{yy} \end{pmatrix}$$

We can easily verify that the trace of this tensor is zero.

In the code we need to be careful and use theta_sph:




```
for i in range(0,NV):
    theta_sph[i]=np.pi/2-math.atan(yV[i]/abs(xV[i]))
    if xV[i]>=0:
        e_rr[i]=exx2[i]*np.sin(theta_sph[i])**2+\
                2*exy2[i]*np.sin(theta_sph[i])*np.cos(theta_sph[i])+\
                eyy2[i]*np.cos(theta_sph[i])**2
        e_tt[i]=exx2[i]*np.cos(theta_sph[i])**2-\
                2*exy2[i]*np.sin(theta_sph[i])*np.cos(theta_sph[i])+\
                eyy2[i]*np.sin(theta_sph[i])**2
        e_rt[i]=(exx2[i]-eyy2[i])*np.sin(theta_sph[i])*np.cos(theta_sph[i])+\
                exy2[i]*(-np.sin(theta_sph[i])**2+\
                np.cos(theta_sph[i])**2)
```

2.4 Dynamic topography

Looking at the ASPECT manual, we find:

"we evaluate the stress and evaluate the component of it in the direction in which gravity acts. In other words we compute

$$\sigma_{rr} = \hat{g}^T(2\eta\varepsilon(\mathbf{u}) - \frac{1}{3}(\text{div } \mathbf{u})I)\hat{g} - p_d$$

where $\hat{g} = \mathbf{g}/\|\mathbf{g}\|$ is the direction of the gravity vector $\mathbf{g}$ and $p_d = p - p_a$ is the dynamic pressure computed by subtracting the adiabatic pressure p_a from the total pressure p computed as part of the Stokes solve. From this, the dynamic topography is computed using the formula

$$h = \frac{\sigma_{rr}}{(\mathbf{g} \cdot \mathbf{n})\rho}$$

where ρ is the density. For the bottom surface we chose the convection that positive values are up (out) and negative values are in (down), analogous to the deformation of the upper surface. The file format then consists of lines with Euclidean coordinates followed by the corresponding topography value."

Here the viscosity is constant in the domain, the flow is incompressible and $\vec{g} = -g_0\vec{e}_r$ so we can compute the dynamic topography as follows:

$$h = -\frac{-p + 2\eta_0\dot{\varepsilon}_{rr}}{\rho_0 g_0}$$

This is of course valid if what is above the surface is a fluid with zero density. This translates into:

```
dyn_topo=np.zeros(NV,dtype=np.float64)
for i in range(0,NV):
    if surfaceV[i] and xV[i]>=0:
       dyn_topo[i]= -(2*viscosity*e_rr[i]-q[i])/(rho0*g0)
```

Note that in our case $\rho_0 = 1$ and $g_0 = 1$ so that in fact the dynamic topography is simply $-\sigma_{rr}$.

3 Numerical methods

3.1 Finite elements

The discretisation of the Stokes equations by means of the Finite Element Method is presented in Chapter 5 of fieldstone. Numerical quadrature is explained in Section 3.1 and two-dimensional basis functions in Section 3.4. Note that the special case of cylindrical axisymmetry is discussed in Section 5.4.3.

3.2 Nomenclature

- Mesh & geometry
  - nelr: number of elements in the radial direction (integer)
  - nelt: number of elements in the tangential direction (integer)
  - xi: ratio of nelt and nelr (integer)
  - nel: total number of elements (integer)
  - NV: number of velocity nodes (integer)
  - NP: number of pressure nodes (integer)
  - xV,zV: coordinates of velocity nodes (float64 array size NV)
  - xP,zP: coordinates of pressure nodes (float64 array size NP)
  - xc,zc: coordinates of element centers (float64 array size nel)
  - rad,theta: r and θ spherical coordinates of velocity nodes (float64 array size NV)
  - nx,ny:
  - u,v: velocity field (float64 array size NV)
  - p: pressure field (float64 array size NP)
  - q: pressure field projected on velocity nodes (float64 array size NV)
- Finite elements
  - mV: number of velocity nodes per element (integer)
  - mP: number of pressure nodes per element (integer)
  - ndofV: number of velocity dof per node (integer)
  - ndofP: number of pressure dof per node (integer)
  - NfemV: total number of velocity dofs (integer)
  - NfemP: total number of pressure dofs (integer)
  - Nfem: total number of dofs (integer)
  - iconV: connectivity array for velocity nodes (integer array)
  - iconP: connectivity array for pressure nodes (integer array)
  - bc_fix: (boolean array size NfemV)
  - bc_val: (float 64 array size NfemV)
- Mapping
  - mapping: type of mapping ('Q1','Q2','Q3','Q4','Q5','Q6')

- mmapping: number of support points (4,9,16,25,36,49)
- xmapping: x-coordinates of all mapping support points, array (mmapping,nel)
- zmapping: z-coordinates of all mapping support points, array (mmapping,nel)
- jcb: Jacobian matrix (float64 array, size (2,2))
- jcbi: inverse of Jacobian matrix (float64 array, size (2,2))
- jcob: determinant of Jacobian matrix

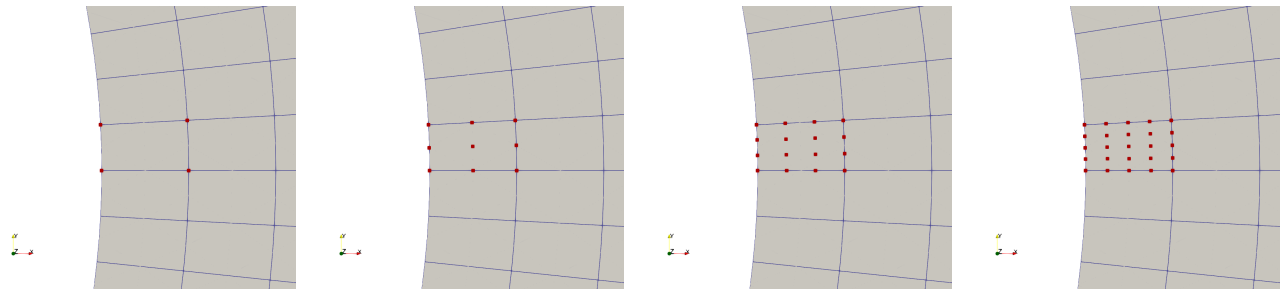
- Quadrature
  - nqperdim: number of quadrature points in 1D
  - nqel: number of quadrature points pr element (nqperdim**2)
  - qcoords_r: r-coordinates of the nqel quadrature points (float 64 array)
  - qcoords_s: s-coordinates of the nqel quadrature points (float 64 array)
  - qweights: associated weights of the nqel quadrature points (float 64 array)
  - xq: x-coordinates of all quadrature points (float64 array size (nel*nqel))
  - zq: z-coordinates of all quadrature points (float64 array size (nel*nqel))

3.3 Mapping

After discretising the domain in nel elements, and having decided the FE pair we want to use to solve the Stokes equations (in this case $Q_2 \times Q_1$), we end up having to compute elemental integrals such as

$$\mathbb{K}_e = \int_{\Omega_e} \boldsymbol{B}^T \cdot \boldsymbol{C}_\eta \cdot \boldsymbol{B} \, d\Omega$$

where Ω_e denotes an element. The way we carry out this integration is by means of the Gauss-Legendre quadrature, which forces us to carry out a change of variables from the original element Ω_e to the reference element $(r, s) \in [-1, 1] \times [-1, 1]$. For this we establish a mapping between both as explained in Section 7.13 of fieldstone. Basis functions $Q_{1,2,3,4}$ are defined in Section 3.4.



Layout of the mapping nodes in element #0 of the mesh. From left to right: Q_1, Q_2, Q_3 and Q_4. Rows of nodes are placed on concentric circles and columns of nodes are equidistant in θ space.

For each element we store the coordinates of these mapping points into two arrays:

```
xmapping=np.zeros((X,nel),dtype=np.float64)
ymapping=np.zeros((X,nel),dtype=np.float64)
```

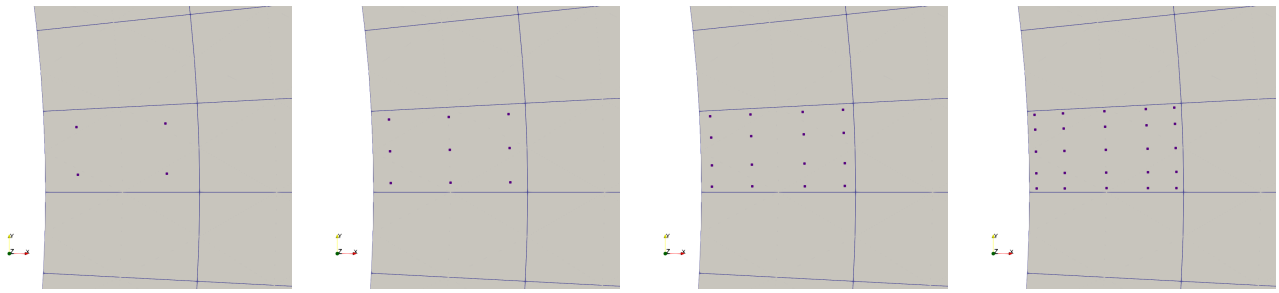
where X stands for the number of nodes for each mapping.

The reduced coordinates for the quadrature points are given by the Gauss-Legendre quadrature approach. The real coordinates of these points is a function of the mapping used so that

```
for iel in range(0,nel):
    for kq in range(0,nqel):
        rq=qcoords_r[kq]
        sq=qcoords_s[kq]
        NNNV=NNN(rq,sq,mapping)
        xq=np.dot(NNNV[:],xmapping[:,iel])
        yq=np.dot(NNNV[:],ymapping[:,iel])
```

Likewise the Jacobian matrix is by definition a function of the chosen mapping so that

```
for iel in range(0,nel):
    for kq in range(0,nqel):
        rq=qcoords_r[kq]
        sq=qcoords_s[kq]
        dNNNVdr=dNNNdr(rq,sq,mapping)
        dNNNVds=dNNNds(rq,sq,mapping)
        jcb[0,0]=np.dot(dNNNVdr[:],xmapping[:,iel])
        jcb[0,1]=np.dot(dNNNVdr[:],ymapping[:,iel])
        jcb[1,0]=np.dot(dNNNVds[:],xmapping[:,iel])
        jcb[1,1]=np.dot(dNNNVds[:],ymapping[:,iel])
        jcob=np.linalg.det(jcb)
        jcbi=np.linalg.inv(jcb)
```



Layout of the quadrature points in element #0 of the mesh. From left to right: nqperdim=2,3,4,5.

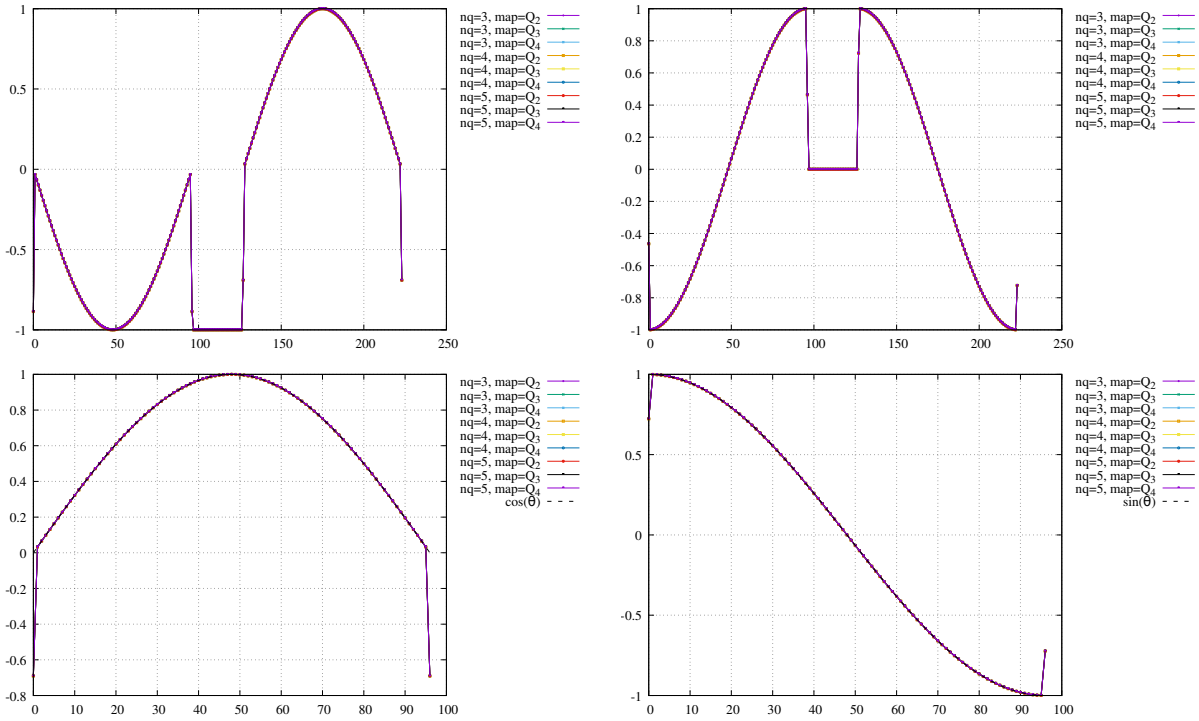
Note that the axisymmetric flag controls whether the Stokes equations are solved in plane strain or under the assumption that there is axisymmetry. In the latter case the mesh is a demi-annulus in the $x > 0$ half plane.

3.4 Quadrature

3.5 Computing normals

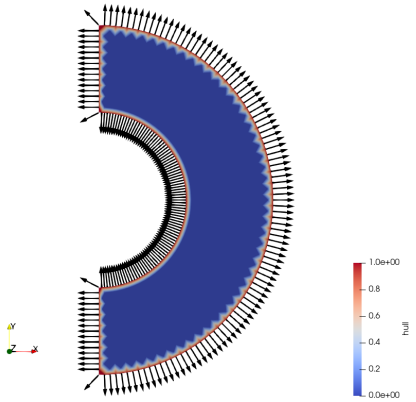
- axisymmetric case only

As seen in stone f151, there are (at least) two ways to compute the normal at the nodes on the surface: $\vec{n}_1$ which is purely geometric and $\vec{n}_2$ which is based on an integration of the basis function derivatives. We found that in the case of the annulus the two coincided to machine precision. Let us now turn the half annulus. As an experiment, *all* nodes on the hull are flagged and the normal $\vec{n}_2$ is computed. Since this normal is a function of basis functions and requires an integration over elements, we can ask ourselves whether the mapping and/or the number of quadrature points influence the results of the normal calculations. The `script_normals` bash script runs the required models - note that the Q_1 mapping and nqperdim=2 have been removed from the loops.



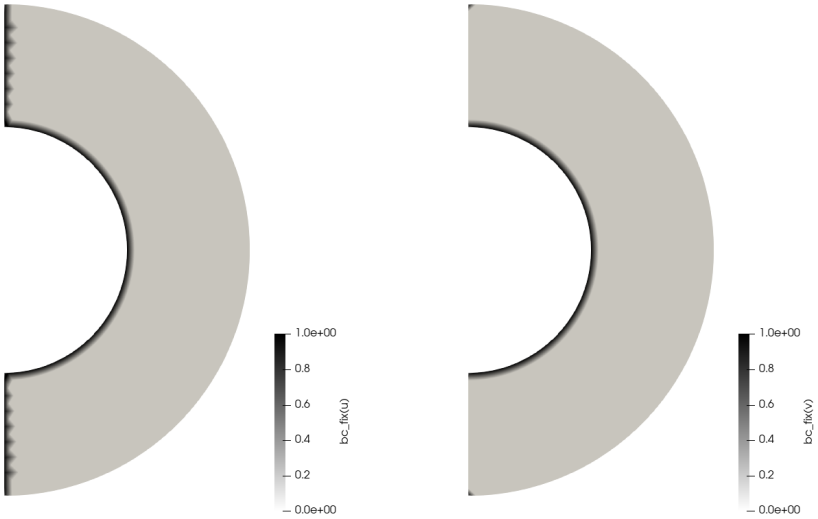
components of the normal vector on the hull (first line), on the surface (second line). Mesh is 8x96.

We find that these normal vector components do not seem to depend on the mapping nor quadrature, and that on the curved parts they match their geometrical counterparts $\vec{n}_1$. These normal vectors are shown here:



Obviously, we need to look closer at the four nodes that belong to the surface and the cmb with $x = 0$. On the one hand they belong the vertical boundary $x = 0$ so their horizontal velocity component should be zero (axisymmetry). On the other hand they also belong to the curved boundaries. In the case of a near infinite resolution the normal to the curved part would align with the vertical axis so that we would then have $v = 0$. In the end these 4 points should be prescribed no-slip boundary conditions.

In our case here only the surface can be prescribed free slip boundary conditions and there are nnt points at the surface. Removing the 2 extremities, we have nnt-2 Lagrange multipliers. Note that then how we compute normals is not relevant.



horizontal and vertical boundary condition indicators.

3.6 Solving the linear system

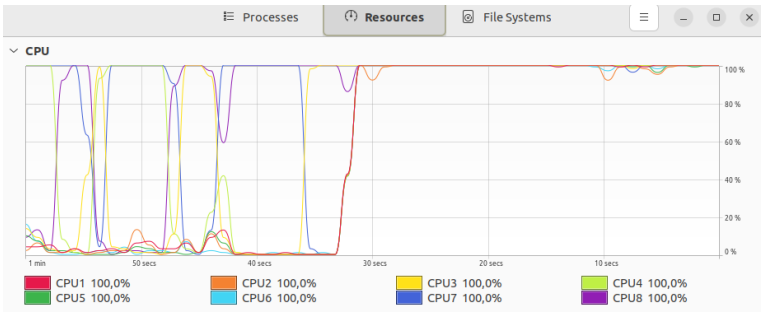
Once the linear system has been built, we must solve it. The simplest approach is to use spsolve[2].

```
sol=sps.linalg.spsolve(sparse_matrix,rhs)
```

Another possibility is available to us: (L)GMRES[3].

```
sol = scipy.sparse.linalg.gmres(sparse_matrix, rhs, restart=2000,tol=1e-8)[0]
sol = scipy.sparse.linalg.lgmres(sparse_matrix, rhs,atol=1e-16,tol=1e-5)[0]
```

Note that these solvers tend to use all resources, as shown here on my laptop with 8 cores:



GMRES[4] and LGMRES are Krylov subspace solvers. They require a (relative) tolerance which determines the accuracy of the solution. Interestingly LGRMRES does not require a restart length parameter.

Note that a preconditioner could and should be supplied to the (L)GMRES algorithms for optimal performance [4].

3.7 Computing velocity derivatives

There are three different methods implemented in the code. after various tests it was found that method 2 was the cheapest and most accurate. The other two remain in the code but are switched off by default.

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.spsolve.html#scipy.sparse.linalg.spsolve

[3] https://docs.scipy.org/doc/scipy/reference/sparse.linalg.html#module-scipy.sparse.linalg

[4] See the excellent blog post on GMRES at https://www.rikvoorhaar.com/gmres/

1. This is only

2.

3.

3.8 Implementing free slip boundary conditions

Finally free slip boundary conditions have been implemented, but only at the surface, and only with the method of Lagrange Multipliers (stone 151 taught us that it works as well as the other method). change

$$\mathbb{K}_e \cdot \vec{\mathcal{V}} + \mathbb{G}_e \cdot \vec{\mathcal{P}} = \vec{f} \quad (4)$$
$$\mathbb{G}_e \cdot \vec{\mathcal{V}} = \vec{0} \quad (5)$$

We multiply the first line by the rotation matrix $\boldsymbol{R}$:

$$\boldsymbol{R} \cdot \mathbb{K}_e \cdot \vec{\mathcal{V}} + \boldsymbol{R} \cdot \mathbb{G}_e \cdot \vec{\mathcal{P}} = \boldsymbol{R} \cdot \vec{f} \quad (6)$$
$$\mathbb{G}_e \cdot \vec{\mathcal{V}} = \vec{0} \quad (7)$$

and then introduce the identity matrix $\boldsymbol{I} = \boldsymbol{R}^T \cdot \boldsymbol{R}$ before the velocity vector:

$$\boldsymbol{R} \cdot \mathbb{K}_e \cdot \boldsymbol{R}^T \cdot \boldsymbol{R} \cdot \vec{\mathcal{V}} + \boldsymbol{R} \cdot \mathbb{G}_e \cdot \vec{\mathcal{P}} = \boldsymbol{R} \cdot \vec{f} \quad (8)$$
$$\mathbb{G}_e \cdot \boldsymbol{R}^T \cdot \boldsymbol{R} \cdot \vec{\mathcal{V}} = \vec{0} \quad (9)$$

The second line can also be written

$$(\boldsymbol{R} \cdot \mathbb{K}_e \cdot \boldsymbol{R}^T) \cdot (\boldsymbol{R} \cdot \vec{\mathcal{V}}) + (\boldsymbol{R} \cdot \mathbb{G}_e) \cdot \vec{\mathcal{P}} = \boldsymbol{R} \cdot \vec{f} \quad (10)$$
$$(\boldsymbol{R} \cdot \mathbb{G}_e)^T \cdot (\boldsymbol{R} \cdot \vec{\mathcal{V}}) = \vec{0} \quad (11)$$

which translates at the elemental level into

```
K_el=RotMat.dot(K_el.dot(RotMat.T))
f_el=RotMat.dot(f_el)
G_el=RotMat.dot(G_el)
```

Note that the matrix $\mathbb{K}_e$ is $(m * ndofV) \times (m * ndofV)$ in size, and so is the matrix $\boldsymbol{R}$.

After boundary conditions are imposed, the system is rotated back:

```
K_el=RotMat.T.dot(K_el.dot(RotMat))
f_el=RotMat.T.dot(f_el)
G_el=RotMat.T.dot(G_el)
```

3.9 Pressure normalisation

Plane strain

In polar coordinates the surface element is

$$dS = Rd\theta$$

so that $\int_0^{2\pi} dS = \int_0^{2\pi} Rd\theta = 2\pi R$ which is the perimeter of the circle.

We wish to normalise the pressure so that it is on average zero on the surface:

$$p_{normalised} = p - < p >$$

where

$$< p >= \frac{\int p(r,\theta) dS}{\int dS} = \frac{\int p(\theta) R d\theta}{\int R d\theta d\theta} = \frac{1}{2\pi R} R \int p(\theta) d\theta = \frac{1}{2\pi} \int_0^{2\pi} p(\theta) d\theta$$

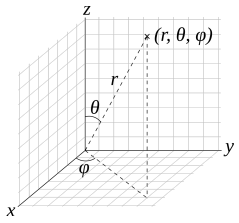
This integral is broken up in a summation over element edges which are at the surface.

$$< p >= \frac{1}{2\pi} \sum_e^{nelt} \int_{\theta_2^e}^{\theta_3^e} p(\theta) d\theta$$

where θ_2^e and θ_3^e are the θ values of nodes 2 and 3 of element e that lie on the surface. These edge integrals are simplifed by assuming a 1-point quadrature:

$$< p >= \frac{1}{2\pi} \sum_e^{nelt} p(\frac{\theta_2^e + \theta_3^e}{2})(\theta_3^e - \theta_2^e)$$

Axisymmetric case



In spherical coordinates, the surface element is

$$dS = R^2 \sin\theta d\theta d\phi$$

We wish to normalise the pressure so that it is on average zero on the surface:

$$p_{normalised} = p - < p >$$

where

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

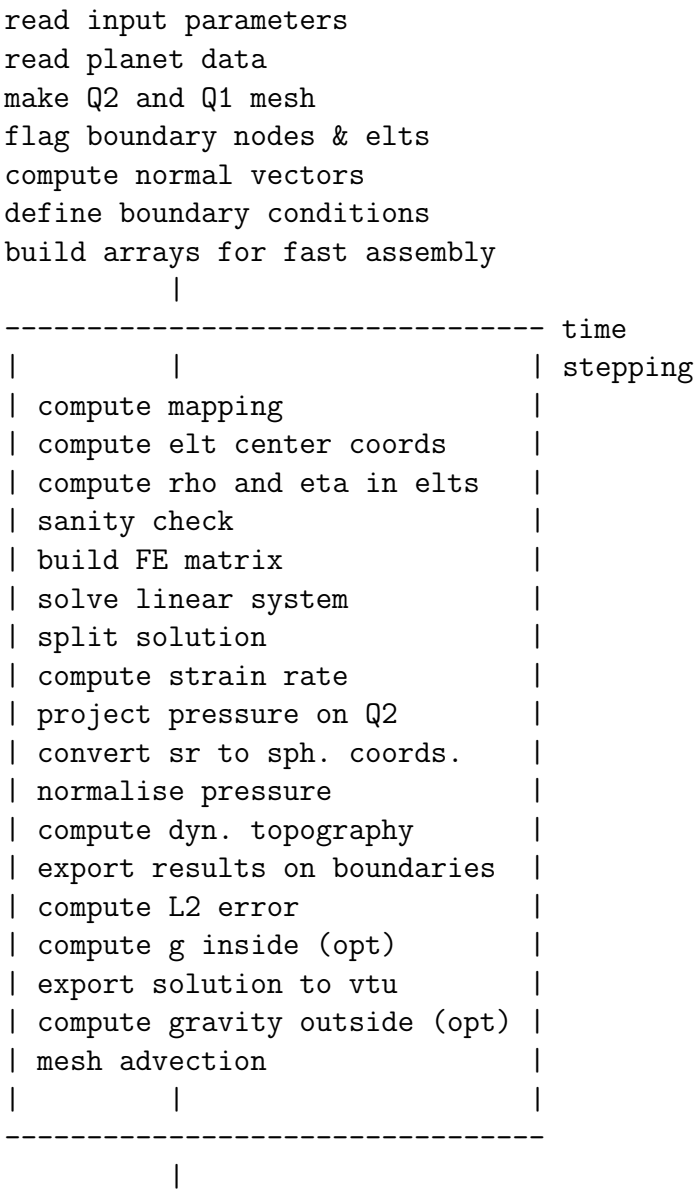
and since p is independent of ϕ then

$$< p >= \frac{R_2^2 \cdot 2\pi \cdot \int_0^\pi p(\theta) \sin\theta d\theta}{R_2^2 \cdot 2\pi \cdot \int \sin\theta d\theta} = \frac{2\pi R_2^2 \int_0^\pi p(\theta) \sin\theta d\theta}{4\pi R_2^2}$$

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

```
poffset=0
for iel in range(0,nel):
    if surface_element[iel]:
        dtheta=theta_sph[iconV[2,iel]]-theta_sph[iconV[3,iel]]
        pmean=0.5*(p[iconP[2,iel]]+p[iconP[3,iel]])
        poffset+=np.sin((theta_sph[iconV[2,iel]]+theta_sph[iconV[3,iel]])/2)*dtheta\
                 *2*np.pi*R2**2 * pmean
poffset/=4*np.pi*R2**2
```

3.10 Code structure



4 The data

4.1 Earth

We assume that viscosity is purely a function of depth[5]..

Five radial viscosity profiles are available:

- The first viscosity profile is a constant viscosity for all depths of 10^{22} Pa s. This value is an estimated value of what is normally found in the literature.
- The second viscosity profile comes from Yoshida et al (2001) [10]. It uses three different regions: lithosphere (0 km to 150 km), upper mantle (150 km to 670 km) and lower mantle (670 km to 2900 km).
- The third viscosity profile comes from Steinberger & Holmes (2008) [7] which is comparable to [6], but of the latter no available data was available. Data is read from the file `DATA/EARTH/eta_stho08.ascii`.
- The fourth and fifth profile come from Ciskova et al (2012) [1]. Data is read from the file `DATA/EARTH/eta_civs.ascii`. The paper showcases two main families of radial viscosity profiles in literature. Family A, which has a sharp increase below the 660 km transition zone and remains constant for most of the lower mantle and family B which is much smoother over the transition zone and increases with depth in the lower mantle.

Three radial density profiles are available:

- PREM [2]
- ak135f [3] `http://rses.anu.edu.au/seismology/ak135/ak135f.html` Data is read from the file `DATA/EARTH/rho_ak135f.ascii`.
- stw105 [5] `http://ds.iris.edu/ds/products/emc-stw105/` Data is read from the file `DATA/EARTH/rho`

eta_model
rho_model

4.2 Mars

[5]This is borrowed from stone 71

5 Running the code

./run
- scripts
- paraview
- gnuplot

6 Benchmarking

The goal here is to explore the influence of the mapping polynomial order and/or the number of quadrature points on the accuracy of the solution of various benchmarks and test cases.

Concretely, in this section we will explore the effect of:

- resolution via the number of elements in the radial direction: nelr=2-32 (we automatically set nelt=12*nelr)
- the number of quadrature points per dimension: nqperdim=2,3,4,5
- the polynomial order of the mapping: mapping='Q1','Q2','Q3','Q4'

and we will monitor the computed area/volume, the root mean square velocity and the velocity and pressure errors.

6.1 Computing volume/mass

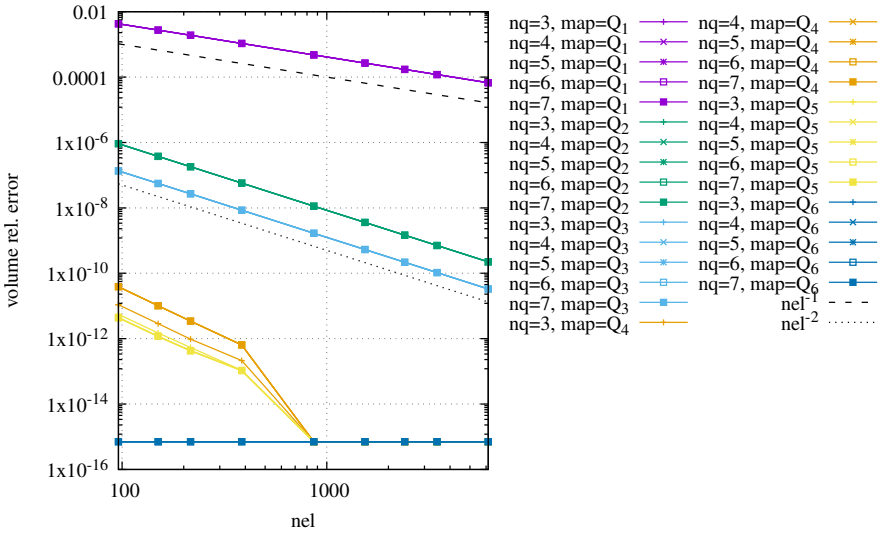
Here we simply compute

$$\mathcal{V} = \sum_e \int_{\Omega_e} 2\pi x dV$$

where the sum runs over all elements of the domain. We of course have

$$\mathcal{V}_{analytical} = \frac{4}{3}\pi(R_2^3 - R_1^3)$$

so that we can compute the relative errors as a function of resolution, mapping and quadrature:

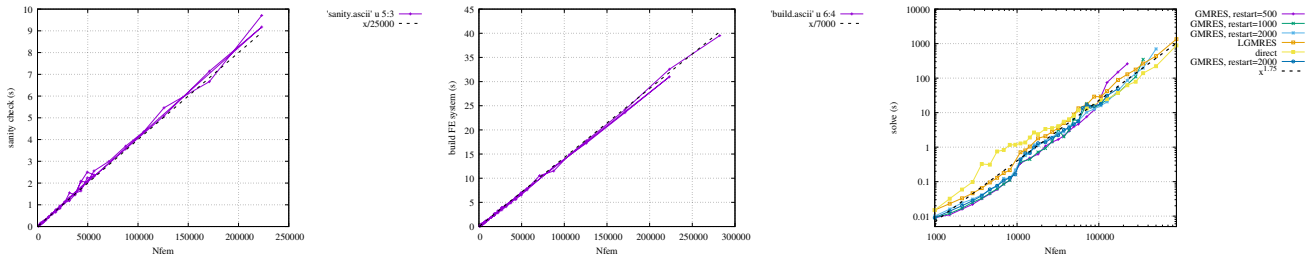


Conclusions:

- unsurprisingly Q_1 mapping yields the worst results
- mapping is the controling factor, much more than quadrature
- for the isoparametric element (mapping Q_2) the number of quadrature points is not critical. Results are virtually identical for nqperdim=2,3,4,5
- Q_3 about one order of magnitude more accurate than Q_2
- Q_4 mapping 3-4 orders of magnitude more accurate than Q_2 mapping

6.2 Annulus convection manufactured solution

6.2.1 Timings



Obtained with xi=6, nqperdim=3, mapping=Q2.

Solve times obtained with constant viscosity (since $\eta = 1$ in this benchmark). These times will probably change substantially when/if the viscosity of the blob is large compared to the mantle viscosity. Note that solve times do not substantially change if the tolerance is brought from 1e-8 to 1e-5.

7 The '4D dynamic earth' inter-code benchmark

[4]

A Misc

A.1 Notes to self

What I have tried to cure the pb of the weird anomalies at the poles.

- turning elements into real trapezes. Made things worse
- different mappings. not much difference
- when using blob, reduced densities. no difference
- nb of quad points, no real difference
- nb of elements in tangential direction, some difference but no cure
- when using blob, drho/rho, no diff
- type of b.c. at point corner below poles, no real diff
- scaling of G matrix
- different rotations/bc for free slip, no difference
- using cmat matrix for dev strain rate, helped a little bit, no cure

A.2 To do list

- visc profiles
- rho profiles
- time stepping
- gravity calculations. import from f96, re-benchmark
- CBF?
- compute self gravity for reduced density case
- export exx1 and exx3 to outside function, clean their code too?
- remove call to math
- bottom free slip
- change y for z in stone
- use PREM gravity value
- aspect with GMG ?
- compute moment of inertia
- by default code now uses elemental rho and eta. it changes things wrt exp0 benchmark results!
- change all eyy and exy to ezz exz
- difference between surfaceV and outerQ2 ?!?

References

[1] H. Čížková, A.P. van den Berg, W. Spakman, and Ctirad Matyska. The viscosity of the earth's lower mantle inferred from sinking speed of subducted lithosphere. *Phys. Earth. Planet. Inter.*, 200–201:56–62, 2012.

[2] A.M. Dziewonski and D.L. Anderson. Preliminary reference Earth model. *Phys. Earth. Planet. Inter.*, 25:297–356, 1981.

[3] B.L.N. Kennett, E.R. Engdahl, and R. Buland. Travel times for global earthquake location and phase association. *Geophy. J. Int.*, 122:108–124, 1995.

[4] M. Kronbichler, T. Heister, and W. Bangerth. High accuracy mantle convection simulation through modern numerical methods. *Geophy. J. Int.*, 191:12–29, 2012.

[5] B Kustowski, G Ekström, and AM Dziewoński. Anisotropic shear-wave velocity structure of the earth's mantle: A global model. *Journal of Geophysical Research: Solid Earth*, 113(B6), 2008.

[6] B. Steinberger and A.R. Calderwood. Models of large-scale viscous flow in the Earth's mantle with constraints from mineral physics and surface observations. *Geophy. J. Int.*, 167:1461–1481, 2006.

[7] B Steinberger and R Holme. Mantle flow models with core-mantle boundary constraints and chemical heterogeneities in the lowermost mantle. *Journal of Geophysical Research: Solid Earth*, 113(B5), 2008.

[8] C. Thieulot and W. Bangerth. On the choice of finite element for applications in geodynamics. *Solid Earth*, 13:229–249, 2022.

[9] Cedric Thieulot. Fieldstone: a computational geodynamics (self-)teaching tool. 2023.

[10] Masaki Yoshida, Satoru Honda, Motoyuki Kido, and Yasuyuki Iwase. Numerical simulation for the prediction of the plate motions. *Earth, planets and space*, 53(7):709–721, 2001.