The Finite Element Method in Geodynamics

C. Thieulot

February 8, 2019

Contents

WARNING: this is work in progress

1 Introduction

1.1 Philosophy

This document was writing with my students in mind, i.e. 3rd and 4th year Geology/Geophysics students at Utrecht University. I have chosen to use jargon as little as possible unless it is a term that is commonly found in the geodynamics literature (methods paper as well as application papers). There is no mathematical proof of any theorem or statement I make. These are to be found in generic Numerical Analysic, Finite Element and Linear Algebra books.

The codes I provide here are by no means optimised as I value code readability over code efficiency. I have also chosen to avoid resorting to multiple code files or even functions to favour a sequential reading of the codes. These codes are not designed to form the basis of a real life application: Existing open source highly optimised codes shoud be preferred, such as ASPECT, CITCOM, LAMEM, PTATIN, PYLITH, ...

All kinds of feedback is welcome on the text (grammar, typos, ...) or on the code(s). You will have my eternal gratitude if you wish to contribute an example, a benchmark, a cookbook.

All the python scripts and this document are freely available at

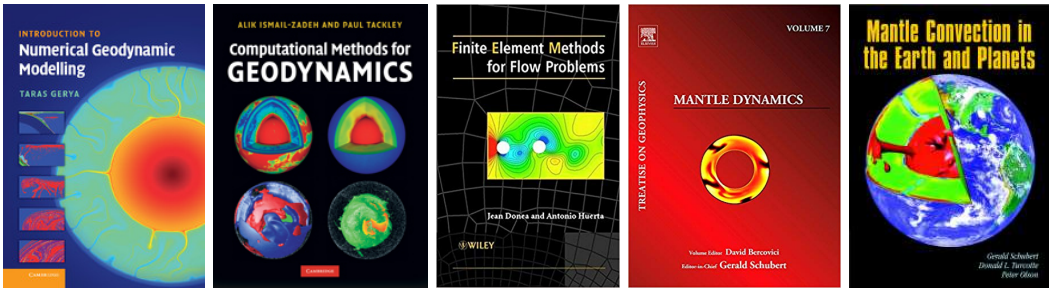
`https://github.com/cedrict/fieldstone`

1.2 Acknowledgments

I have benefitted from many discussions, lectures, tutorials, coffee machine discussions, debugging sessions, conference poster sessions, etc ... over the years. I wish to name these instrumental people in particular and in alphabetic order: Wolfgang Bangerth, Jean Braun, Philippe Fullsack, Menno Fraters, Anne Glerum, Timo Heister, Robert Myhill, John Naliboff, Lukas van de Wiel, Arie van den Berg, and the whole ASPECT family/team.

I wish to acknowledge many BSc and MSc students for their questions and feedback. and wish to mention Job Mos in particular who wrote the very first version of fieldstone as part of his MSc thesis. and Tom Weir for his contributions to the compressible formulations.

1.3 Essential literature



1.4 Installation

```
python3.6 -m pip install --user numpy scipy matplotlib
```

2 The physical equations of Fluid Dynamics

Symbol	meaning	unit
t	Time	s
x, y, z	Cartesian coordinates	m
$\boldsymbol{v}$	velocity vector	m$\cdot$ s^{-1}
ρ	mass density	kg/m^3
η	dynamic viscosity	Pa$\cdot$ s
λ	penalty parameter	Pa$\cdot$ s
T	temperature	K
$\boldsymbol{\nabla}$	gradient operator	m^{-1}
$\boldsymbol{\nabla}\cdot$	divergence operator	m^{-1}
p	pressure	Pa
$\dot{\boldsymbol{\varepsilon}}(\boldsymbol{v})$	strain rate tensor	s^{-1}
α	thermal expansion coefficient	K^{-1}
k	thermal conductivity	W/(m $\cdot$ K)
C_p	Heat capacity	J/K
H	intrinsic specific heat production	W/kg
β_T	isothermal compressibility	Pa^{-1}
$\boldsymbol{\tau}$	deviatoric stress tensor	Pa
$\boldsymbol{\sigma}$	full stress tensor	Pa

2.1 The heat transport equation - energy conservation equation

Let us start from the heat transport equation as shown in Schubert, Turcotte and Olson [82]:

$$\rho C_p \frac{DT}{Dt} - \alpha T \frac{Dp}{Dt} = \boldsymbol{\nabla} \cdot k \boldsymbol{\nabla} T + \Phi + \rho H$$

with D/Dt being the total derivatives so that

$$\frac{DT}{Dt} = \frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} T \qquad \frac{Dp}{Dt} = \frac{\partial p}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} p$$

Solving for temperature, this equation is often rewritten as follows:

$$\rho C_p \frac{DT}{Dt} - \boldsymbol{\nabla} \cdot k \boldsymbol{\nabla} T = \alpha T \frac{Dp}{Dt} + \Phi + \rho H$$

A note on the shear heating term Φ: In many publications, Φ is given by $\Phi = \tau_{ij} \partial_j u_i = \boldsymbol{\tau} : \boldsymbol{\nabla} \boldsymbol{v}$.

$$\begin{aligned}
\Phi &= \tau_{ij} \partial_j u_i \\
&= 2\eta \dot{\varepsilon}^d_{ij} \partial_j u_i \\
&= 2\eta \frac{1}{2} \left(\dot{\varepsilon}^d_{ij} \partial_j u_i + \dot{\varepsilon}^d_{ji} \partial_i u_j \right) \\
&= 2\eta \frac{1}{2} \left(\dot{\varepsilon}^d_{ij} \partial_j u_i + \dot{\varepsilon}^d_{ij} \partial_i u_j \right) \\
&= 2\eta \dot{\varepsilon}^d_{ij} \frac{1}{2} \left(\partial_j u_i + \partial_i u_j \right) \\
&= 2\eta \dot{\varepsilon}^d_{ij} \dot{\varepsilon}_{ij} \\
&= 2\eta \dot{\boldsymbol{\varepsilon}}^d : \dot{\boldsymbol{\varepsilon}} \\
&= 2\eta \dot{\boldsymbol{\varepsilon}}^d : \left(\dot{\boldsymbol{\varepsilon}}^d + \frac{1}{3} (\boldsymbol{\nabla} \cdot \boldsymbol{v}) \mathbf{1} \right) \\
&= 2\eta \dot{\boldsymbol{\varepsilon}}^d : \dot{\boldsymbol{\varepsilon}}^d + 2\eta \dot{\boldsymbol{\varepsilon}}^d : \mathbf{1} (\boldsymbol{\nabla} \cdot \boldsymbol{v}) \\
&= 2\eta \dot{\boldsymbol{\varepsilon}}^d : \dot{\boldsymbol{\varepsilon}}^d
\end{aligned} \tag{1}$$

Finally

$$\Phi = \boldsymbol{\tau} : \boldsymbol{\nabla} \boldsymbol{v} = 2\eta \dot{\boldsymbol{\varepsilon}}^d : \dot{\boldsymbol{\varepsilon}}^d = 2\eta \left((\dot{\varepsilon}^d_{xx})^2 + (\dot{\varepsilon}^d_{yy})^2 + 2(\dot{\varepsilon}^d_{xy})^2 \right)$$

2.2 The momentum conservation equations

Because the Prandlt number is virtually zero in Earth science applications the Navier Stokes equations reduce to the Stokes equation:

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \rho \boldsymbol{g} = 0$$

Since

$$\boldsymbol{\sigma} = -p\mathbf{1} + \boldsymbol{\tau}$$

it also writes

$$-\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot \boldsymbol{\tau} + \rho \boldsymbol{g} = 0$$

Using the relationship $\boldsymbol{\tau} = 2\eta \dot{\boldsymbol{\varepsilon}}^d$ we arrive at

$$\boxed{-\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot (2\eta \dot{\boldsymbol{\varepsilon}}^d) + \rho \boldsymbol{g} = 0}$$

2.3 The mass conservation equations

The mass conservation equation is given by

$$\frac{D\rho}{Dt} + \rho \boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$$

or,

$$\boxed{\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0}$$

In the case of an incompressible flow, then $\partial\rho/\partial t = 0$ and $\boldsymbol{\nabla}\rho = 0$, i.e. $D\rho/Dt = 0$ and the remaining equation is simply:

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$$

2.4 The equations in ASPECT manual

The following is lifted off the ASPECT manual. We focus on the system of equations in a $d = 2$- or $d = 3$-dimensional domain Ω that describes the motion of a highly viscous fluid driven by differences in the gravitational force due to a density that depends on the temperature. In the following, we largely follow the exposition of this material in Schubert, Turcotte and Olson [82].

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

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

$$\nabla \cdot (\rho \boldsymbol{v}) = 0 \qquad \text{in } \Omega, \quad (3)$$

$$\begin{aligned} \rho C_p \left(\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T \right) - \nabla \cdot k \nabla T &= \rho H \\ &\quad + 2\eta \left(\dot{\varepsilon}(\boldsymbol{v}) - \frac{1}{3}(\nabla \cdot \boldsymbol{v})\mathbf{1} \right) : \left(\dot{\varepsilon}(\boldsymbol{v}) - \frac{1}{3}(\nabla \cdot \boldsymbol{v})\mathbf{1} \right) \\ &\quad + \alpha T \left(\boldsymbol{v} \cdot \nabla p \right) \end{aligned} \qquad (4)$$

$$\text{in } \Omega,$$

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

In this set of equations, (110) and (111) represent the compressible Stokes equations in which $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$ 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 (112) for the temperature field $T = T(\mathbf{x}, t)$ that contains heat conduction terms as well as advection with the flow velocity $\mathbf{v}$. 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;

In order to arrive at the set of equations that ASPECT solves, we need to

- neglect the $\partial p/\partial t$. WHY?
- neglect the $\partial \rho/\partial t$. WHY?

from equations above.

Also, their definition of the shear heating term Φ is:

$$\Phi = k_B(\boldsymbol{\nabla}\cdot\boldsymbol{v})^2 + 2\eta\dot{\boldsymbol{\varepsilon}}^d : \dot{\boldsymbol{\varepsilon}}^d$$

For many fluids the bulk viscosity k_B is very small and is often taken to be zero, an assumption known as the Stokes assumption: $k_B = \lambda + 2\eta/3 = 0$. Note that η is the dynamic viscosity and λ the second viscosity. Also,

$$\boldsymbol{\tau} = 2\eta\dot{\boldsymbol{\varepsilon}} + \lambda(\boldsymbol{\nabla}\cdot\boldsymbol{v})\mathbf{1}$$

but since $k_B = \lambda + 2\eta/3 = 0$, then $\lambda = -2\eta/3$ so

$$\boldsymbol{\tau} = 2\eta\dot{\boldsymbol{\varepsilon}} - \frac{2}{3}\eta(\boldsymbol{\nabla}\cdot\boldsymbol{v})\mathbf{1} = 2\eta\dot{\boldsymbol{\varepsilon}}^d$$

2.5 the Boussinesq approximation: an Incompressible flow

[from aspect manual] 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 (110). The primary result of this assumption is that the continuity equation (111) will now read

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$$

This implies that the strain rate tensor is deviatoric. Under the Boussinesq approximation, the equations are much simplified:

$$-\nabla \cdot [2\eta \dot{\boldsymbol{\varepsilon}}(\boldsymbol{v})] + \nabla p = \rho \boldsymbol{g} \qquad \text{in } \Omega, \tag{5}$$

$$\nabla \cdot (\rho \boldsymbol{v}) = 0 \qquad \text{in } \Omega, \tag{6}$$

$$\rho_0 C_p \left(\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T \right) - \nabla \cdot k \nabla T = \rho H \qquad \text{in } \Omega \tag{7}$$

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

3 The building blocks of the Finite Element Method

3.1 Numerical integration

As we will see later, using the Finite Element method to solve problems involves computing integrals which are more often than not too complex to be computed analytically/exactly. We will then need to compute them numerically.

[wiki] In essence, the basic problem in numerical integration is to compute an approximate solution to a definite integral

$$\int_a^b f(x)dx$$

to a given degree of accuracy. This problem has been widely studied [**?**] and we know that if $f(x)$ is a smooth function, and the domain of integration is bounded, there are many methods for approximating the integral to the desired precision.

There are several reasons for carrying out numerical integration.

- The integrand $f(x)$ may be known only at certain points, such as obtained by sampling. Some embedded systems and other computer applications may need numerical integration for this reason.
- A formula for the integrand may be known, but it may be difficult or impossible to find an antiderivative that is an elementary function. An example of such an integrand is $f(x) = exp(-x^2)$, the antiderivative of which (the error function, times a constant) cannot be written in elementary form.
- It may be possible to find an antiderivative symbolically, but it may be easier to compute a numerical approximation than to compute the antiderivative. That may be the case if the antiderivative is given as an infinite series or product, or if its evaluation requires a special function that is not available.

3.1.1 in 1D - theory

The simplest method of this type is to let the interpolating function be a constant function (a polynomial of degree zero) that passes through the point $((a+b)/2, f((a+b)/2))$.

This is called the midpoint rule or rectangle rule.

$$\int_a^b f(x)dx \simeq (b-a)f(\frac{a+b}{2})$$

insert here figure

The interpolating function may be a straight line (an affine function, i.e. a polynomial of degree 1) passing through the points $(a, f(a))$ and $(b, f(b))$.

This is called the trapezoidal rule.

$$\int_a^b f(x)dx \simeq (b-a)\frac{f(a)+f(b)}{2}$$

insert here figure

For either one of these rules, we can make a more accurate approximation by breaking up the interval [a, b] into some number n of subintervals, computing an approximation for each subinterval, then adding up all the results. This is called a composite rule, extended rule, or iterated rule. For example, the composite trapezoidal rule can be stated as

$$\int_a^b f(x)dx \simeq \frac{b-a}{n}\left(\frac{f(a)}{2} + \sum_{k=1}^{n-1} f(a+k\frac{b-a}{n}) + \frac{f(b)}{2}\right)$$

where the subintervals have the form $[kh, (k+1)h]$, with $h = (b-a)/n$ and $k = 0, 1, 2, \ldots, n-1$.

a) b)

The interval $[-2, 2]$ is broken into 16 sub-intervals. The blue lines correspond to the approximation of the red curve by means of a) the midpoint rule, b) the trapezoidal rule.

There are several algorithms for numerical integration (also commonly called 'numerical quadrature', or simply 'quadrature') . Interpolation with polynomials evaluated at equally spaced points in $[a, b]$ yields the NewtonCotes formulas, of which the rectangle rule and the trapezoidal rule are examples. If we allow the intervals between interpolation points to vary, we find another group of quadrature formulas, such as the Gauss(ian) quadrature formulas. A Gaussian quadrature rule is typically more accurate than a NewtonCotes rule, which requires the same number of function evaluations, if the integrand is smooth (i.e., if it is sufficiently differentiable).

An $n-$point Gaussian quadrature rule, named after Carl Friedrich Gauss, is a quadrature rule constructed to yield an exact result for polynomials of degree $2n-1$ or less by a suitable choice of the points x_i and weights w_i for $i = 1, \ldots, n$.

The domain of integration for such a rule is conventionally taken as $[-1, 1]$, so the rule is stated as

$$\int_{-1}^{+1} f(x)dx = \sum_{i_q=1}^{n} w_{i_q} f(x_{i_q})$$

In this formula the x_{i_q} coordinate is the i-th root of the Legendre polynomial $P_n(x)$.

It is important to note that a Gaussian quadrature will only produce good results if the function $f(x)$ is well approximated by a polynomial function within the range $[-1, 1]$. As a consequence, the method is not, for example, suitable for functions with singularities.

Number of points, n	Points, x_i	Weights, w_i
1	0	2
2	$\pm\sqrt{\frac{1}{3}}$	1
3	0	$\frac{8}{9}$
	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$
5	0	$\frac{128}{225}$
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}$
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	$\frac{322-13\sqrt{70}}{900}$

Gauss-Legendre points and their weights.

As shown in the above table, it can be shown that the weight values must fulfill the following condition:

$$\sum_{i_q} w_{i_q} = 2 \tag{8}$$

and it is worth noting that all quadrature point coordinates are symmetrical around the origin.

Since most quadrature formula are only valid on a specific interval, we now must address the problem of their use outside of such intervals. The solution turns out to be quite simple: one must carry out a change of variables from the interval $[a, b]$ to $[-1, 1]$.

We then consider the reduced coordinate $r \in [-1, 1]$ such that

$$r = \frac{2}{b-a}(x-a) - 1$$

This relationship can be reversed such that when r is known, its equivalent coordinate $x \in [a,b]$ can be computed:

$$x = \frac{b-a}{2}(1+r) + a$$

From this it follows that

$$dx = \frac{b-a}{2}dr$$

and then

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{+1} f(r)dr \simeq \frac{b-a}{2}\sum_{i_q=1}^{n} w_{i_q} f(r_{i_q})$$

3.1.2 in 1D - examples

example 1 Since we know how to carry out any required change of variables, we choose for simplicity $a=-1$, $b=+1$. Let us take for example $f(x)=\pi$. Then we can compute the integral of this function over the interval $[a,b]$ exactly:

$$I = \int_{-1}^{+1} f(x)dx = \pi\int_{-1}^{+1} dx = 2\pi$$

We can now use a Gauss-Legendre formula to compute this same integral:

$$I_{gq} = \int_{-1}^{+1} f(x)dx = \sum_{i_q=1}^{n_q} w_{i_q} f(x_{i_q}) = \sum_{i_q=1}^{n_q} w_{i_q}\pi = \pi\underbrace{\sum_{i_q=1}^{n_q} w_{i_q}}_{=2} = 2\pi$$

where we have used the property of the weight values of Eq.(8). Since the actual number of points was never specified, this result is valid for all quadrature rules.

example 2 Let us now take $f(x)=mx+p$ and repeat the same exercise:

$$I = \int_{-1}^{+1} f(x)dx = \int_{-1}^{+1}(mx+p)dx = [\frac{1}{2}mx^2 + px]_{-1}^{+1} = 2p$$

$$I_{gq} = \int_{-1}^{+1} f(x)dx = \sum_{i_q=1}^{n_q} w_{i_q} f(x_{i_q}) = \sum_{i_q=1}^{n_q} w_{i_q}(mx_{i_q}+p) = m\underbrace{\sum_{i_q=1}^{n_q} w_{i_q}x_{i_q}}_{=0} + p\underbrace{\sum_{i_q=1}^{n_q} w_{i_q}}_{=2} = 2p$$

since the quadrature points are symmetric w.r.t. to zero on the x-axis. Once again the quadrature is able to compute the exact value of this integral: this makes sense since an n-point rule exactly integrates a $2n-1$ order polynomial such that a 1 point quadrature exactly integrates a first order polynomial like the one above.

example 3 Let us now take $f(x)=x^2$. We have

$$I = \int_{-1}^{+1} f(x)dx = \int_{-1}^{+1} x^2 dx = [\frac{1}{3}x^3]_{-1}^{+1} = \frac{2}{3}$$

and

$$I_{gq} = \int_{-1}^{+1} f(x)dx = \sum_{i_q=1}^{n_q} w_{i_q} f(x_{i_q}) = \sum_{i_q=1}^{n_q} w_{i_q} x_{i_q}^2$$

- $n_q = 1$: $x_{iq}^{(1)} = 0$, $w_{i_q} = 2$. $I_{gq} = 0$
- $n_q = 2$: $x_q^{(1)} = -1/\sqrt{3}$, $x_q^{(2)} = 1/\sqrt{3}$, $w_q^{(1)} = w_q^{(2)} = 1$. $I_{gq} = \frac{2}{3}$
- It also works $\forall n_q > 2$!

3.1.3 in 2D/3D - theory

Let us now turn to a two-dimensional integral of the form

$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(x,y) dx dy$$

The equivalent Gaussian quadrature writes:

$$I_{gq} \simeq \sum_{i_q=1}^{n_q} \sum_{j_q}^{n_q} f(x_{i_q}, y_{j_q}) w_{i_q} w_{j_q}$$

3.2 The mesh

3.3 A bit of FE terminology

We introduce here some terminology for efficient element descriptions [40]:

- For triangles/tetrahedra, the designation $P_m \times P_n$ means that each component of the velocity is approximated by continuous piecewise complete Polynomials of degree m and pressure by continuous piecewise complete Polynomials of degree n. For example $P_2 \times P_1$ means

  $$u \sim a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2$$

  with similar approximations for v, and

  $$p \sim b_1 + b_2 x + b_3 y$$

  Both velocity and pressure are continuous across element boundaries, and each triangular element contains 6 velocity nodes and three pressure nodes.

- For the same families, $P_m \times P_{-n}$ is as above, except that pressure is approximated via piecewise *discontinuous* polynomials of degree n. For instance, $P_2 \times P_{-1}$ is the same as $P_2 P_1$ except that pressure is now an independent linear function in each element and therefore discontinuous at element boundaries.

- For quadrilaterals/hexahedra, the designation $Q_m \times Q_n$ means that each component of the velocity is approximated by a continuous piecewise polynomial of degree m *in each direction* on the quadrilateral and likewise for pressure, except that the polynomial is of degree n. For instance, $Q_2 \times Q_1$ means

  $$u \sim a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 + a_7 x^2 y + a_8 xy^2 + a_9 x^2 y^2$$

  and

  $$p \sim b_1 + b_2 x + b_3 y + b_4 xy$$

- For these same families, $Q_m \times Q_{-n}$ is as above, except that the pressure approximation is not continuous at element boundaries.

- Again for the same families, $Q_m \times P_{-n}$ indicates the same velocity approximation with a pressure approximation that is a discontinuous complete piecewise polynomial of degree n (not of degree n in each direction !)

- The designation P_m^+ or Q_m^+ means that some sort of bubble function was added to the polynomial approximation for the velocity. You may also find the term 'enriched element' in the literature.

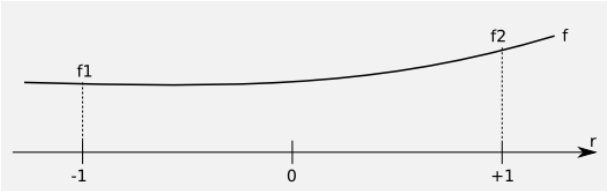
- Finally, for $n = 0$, we have piecewise-constant pressure, and we omit the minus sign for simplicity.

Another point which needs to be clarified is the use of so-called 'conforming elements' (or 'non-conforming elements'). Following again [40], conforming velocity elements are those for which the basis functions for a subset of H^1 for the continuous problem (the first derivatives and their squares are integrable in Ω). For instance, the rotated $Q_1 \times P_0$ element of Rannacher and Turek (see section **??**) is such that the velocity is discontinous across element edges, so that the derivative does not exist there. Another typical example of non-conforming element is the Crouzeix-Raviart element [21].

3.4 Elements and basis functions in 1D

3.4.1 Linear basis functions (Q_1)

Let $f(r)$ be a C^1 function on the interval $[-1:1]$ with $f(-1) = f_1$ and $f(1) = f_2$.



Let us assume that the function $f(r)$ is to be approximated on $[-1, 1]$ by the first order polynomial

$$f(r) = a + br \tag{9}$$

Then it must fulfill

$$\begin{aligned} f(r=-1) &= a - b = f_1 \\ f(r=+1) &= a + b = f_2 \end{aligned}$$

This leads to

$$a = \frac{1}{2}(f_1 + f_2) \qquad b = \frac{1}{2}(-f_1 + f_2)$$

and then replacing a, b in Eq. (9) by the above values on gets

$$f(r) = \left[\frac{1}{2}(1-r)\right] f_1 + \left[\frac{1}{2}(1+r)\right] f_2$$

or

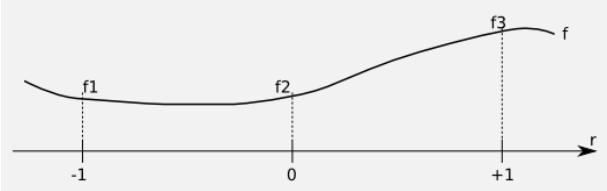
$$f(r) = \sum_{i=1}^{2} N_i(r) f_1$$

with

$$\begin{aligned} N_1(r) &= \frac{1}{2}(1-r) \\ N_2(r) &= \frac{1}{2}(1+r) \end{aligned} \tag{10}$$

3.4.2 Quadratic basis functions (Q_2)

Let $f(r)$ be a C^1 function on the interval $[-1:1]$ with $f(-1) = f_1$, $f(0) = f_2$ and $f(1) = f_3$.



Let us assume that the function $f(r)$ is to be approximated on $[-1, 1]$ by the second order polynomial

$$f(r) = a + br + cr^2 \tag{11}$$

Then it must fulfill

$$\begin{aligned} f(r=-1) &= a - b + c = f_1 \\ f(r=0) &= a \qquad\quad = f_2 \\ f(r=+1) &= a + b + c = f_3 \end{aligned}$$

This leads to

$$a = f_2 \quad b = \frac{1}{2}(-f1 + f3) \quad c = \frac{1}{2}(f_1 + f_3 - 2f_2)$$

and then replacing a, b, c in Eq. (11) by the above values on gets

$$f(r) = \left[\frac{1}{2}r(r-1)\right] f_1 + (1-r^2)f_2 + \left[\frac{1}{2}r(r+1)\right] f_3$$

or,

$$f(r) = \sum_{i=1}^{3} N_i(r) f_i$$

with

$$\begin{aligned} N_1(r) &= \frac{1}{2}r(r-1) \\ N_2(r) &= (1-r^2) \\ N_3(r) &= \frac{1}{2}r(r+1) \end{aligned} \tag{12}$$

3.4.3 Cubic basis functions (Q_3)

The 1D basis polynomial is given by

$$f(r) = a + br + cr^2 + dr^3$$

with the nodes at position -1,-1/3, +1/3 and +1.

$$\begin{aligned} f(-1) &= a - b + c - d = f_1 \\ f(-1/3) &= a - \frac{b}{3} + \frac{c}{9} - \frac{d}{27} = f_2 \\ f(+1/3) &= a - \frac{b}{3} + \frac{c}{9} - \frac{d}{27} = f_3 \\ f(+1) &= a + b + c + d = f_4 \end{aligned}$$

Adding the first and fourth equation and the second and third, one arrives at

$$f_1 + f_4 = 2a + 2c \qquad f_2 + f_3 = 2a + \frac{2c}{9}$$

and finally:

$$a = \frac{1}{16}\left(-f_1 + 9f_2 + 9f_3 - f_4\right)$$

$$c = \frac{9}{16}\left(f_1 - f_2 - f_3 + f_4\right)$$

Combining the original 4 equations in a different way yields

$$2b + 2d = f_4 - f_1 \qquad \frac{2b}{3} + \frac{2d}{27} = f_3 - f_2$$

so that

$$b = \frac{1}{16}\left(f_1 - 27f_2 + 27f_3 - f_4\right)$$

$$d = \frac{9}{16}\left(-f_1 + 3f_2 - 3f_3 + f_4\right)$$

Finally,

$$
\begin{aligned}
f(r) &= a + b + cr^2 + dr^3 \\
&= \frac{1}{16}(-1 + r + 9r^2 - 9r^3)f_1 \\
&+ \frac{1}{16}(9 - 27r - 9r^2 + 27r^3)f_2 \\
&+ \frac{1}{16}(9 + 27r - 9r^2 - 27r^3)f_3 \\
&+ \frac{1}{16}(-1 - r + 9r^2 + 9r^3)f_4 \\
&= \sum_{i=1}^{4} N_i(r) f_i
\end{aligned}
$$

where

$$
\begin{aligned}
N_1 &= \frac{1}{16}(-1 + r + 9r^2 - 9r^3) \\
N_2 &= \frac{1}{16}(9 - 27r - 9r^2 + 27r^3) \\
N_3 &= \frac{1}{16}(9 + 27r - 9r^2 - 27r^3) \\
N_4 &= \frac{1}{16}(-1 - r + 9r^2 + 9r^3)
\end{aligned}
$$

Verification:

- Let us assume $f(r) = C$, then

$$\hat{f}(r) = \sum N_i(r) f_i = \sum_i N_i C = C \sum_i N_i = C$$

  so that a constant function is exactly reproduced, as expected.

- Let us assume $f(r) = r$, then $f_1 = -1$, $f_2 = -1/3$, $f_3 = 1/3$ and $f_4 = +1$. We then have

$$
\begin{aligned}
\hat{f}(r) &= \sum N_i(r) f_i \\
&= -N_1(r) - \frac{1}{3}N_2(r) + \frac{1}{3}N_3(r) + N_4(r) \\
&= [-(-1 + r + 9r^2 - 9r^3) \\
&\quad -\frac{1}{3}(9 - 27r - 9r^2 - 27r^3) \\
&\quad +\frac{1}{3}(9 + 27r - 9r^2 + 27r^3) \\
&\quad +(-1 - r + 9r^2 + 9r^3)]/16 \\
&= [-r + 9r + 9r - r]/16 + ...0... \\
&= r
\end{aligned} \tag{13}
$$

The basis functions derivative are given by

$$
\begin{aligned}
\frac{\partial N_1}{\partial r} &= \frac{1}{16}(1 + 18r - 27r^2) \\
\frac{\partial N_2}{\partial r} &= \frac{1}{16}(-27 - 18r + 81r^2) \\
\frac{\partial N_3}{\partial r} &= \frac{1}{16}(+27 - 18r - 81r^2) \\
\frac{\partial N_4}{\partial r} &= \frac{1}{16}(-1 + 18r + 27r^2)
\end{aligned}
$$

Verification:

- Let us assume $f(r) = C$, then

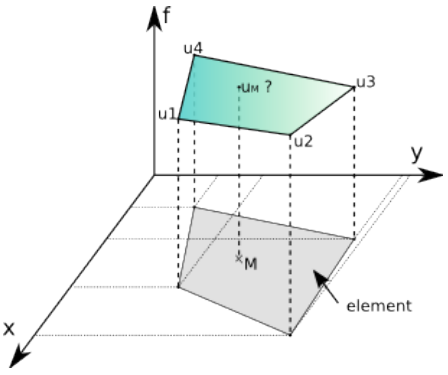
$$
\begin{aligned}
\frac{\partial \hat{f}}{\partial r} &= \sum_i \frac{\partial N_i}{\partial r} f_i \\
&= C \sum_i \frac{\partial N_i}{\partial r} \\
&= \frac{C}{16}[(1 + 18r - 27r^2) \\
&\quad +(-27 - 18r + 81r^2) \\
&\quad +(+27 - 18r - 81r^2) \\
&\quad +(-1 + 18r + 27r^2)] \\
&= 0
\end{aligned}
$$

- Let us assume $f(r) = r$, then $f_1 = -1$, $f_2 = -1/3$, $f_3 = 1/3$ and $f_4 = +1$. We then have

$$
\begin{aligned}
\frac{\partial \hat{f}}{\partial r} &= \sum_i \frac{\partial N_i}{\partial r} f_i \\
&= \frac{1}{16}[-(1 + 18r - 27r^2) \\
&\quad -\frac{1}{3}(-27 - 18r + 81r^2) \\
&\quad +\frac{1}{3}(+27 - 18r - 81r^2) \\
&\quad +(-1 + 18r + 27r^2)] \\
&= \frac{1}{16}[-2 + 18 + 54r^2 - 54r^2] \\
&= 1
\end{aligned}
$$

3.5 Elements and basis functions in 2D

Let us for a moment consider a single quadrilateral element in the xy-plane, as shown on the following figure:



Let us assume that we know the values of a given field u at the vertices. For a given point M inside the element in the plane, what is the value of the field u at this point? It makes sense to postulate that $u_M = u(x_M, y_M)$ will be given by

$$u_M = \phi(u_1, u_2, u_3, u_4, x_M, y_M)$$

where ϕ is a function to be determined. Although ϕ is not unique, we can decide to express the value u_M as a weighed sum of the values at the vertices u_i. One option could be to assign all four vertices the same weight, say $1/4$ so that $u_M = (u_1 + u_2 + u_3 + u_4)/4$, i.e. u_M is simply given by the arithmetic mean of the vertices values. This approach suffers from a major drawback as it does not use the location of point M inside the element. For instance, when $(x_M, y_M) \to (x_2, y_2)$ we expect $u_M \to u_2$.

In light of this, we could now assume that the weights would depend on the position of M in a continuous fashion:

$$u(x_M, y_M) = \sum_{i=1}^{4} N_i(x_M, y_M)\ u_i$$

where the N_i are continous ("well behaved") functions which have the property:

$$N_i(x_j, y_j) = \delta_{ij}$$

or, in other words:

$$N_3(x_1, y_1) = 0 \tag{14}$$
$$N_3(x_2, y_2) = 0 \tag{15}$$
$$N_3(x_3, y_3) = 1 \tag{16}$$
$$N_3(x_4, y_4) = 0 \tag{17}$$

The functions N_i are commonly called basis functions.

Omitting the M subscripts for any point inside the element, the velocity components u and v are given by:

$$\hat{u}(x, y) = \sum_{i=1}^{4} N_i(x, y)\ u_i \tag{18}$$

$$\hat{v}(x, y) = \sum_{i=1}^{4} N_i(x, y)\ v_i \tag{19}$$

Rather interestingly, one can now easily compute velocity gradients (and therefore the strain rate tensor) since we have assumed the basis functions to be "well behaved" (in this case differentiable):

$$\dot{\epsilon}_{xx}(x, y) = \frac{\partial u}{\partial x} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial x}\ u_i \tag{20}$$

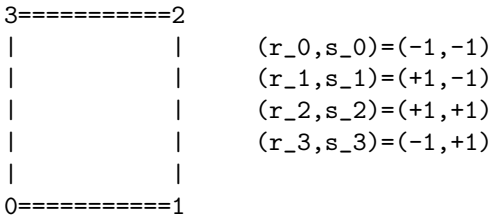
$$\dot{\epsilon}_{yy}(x, y) = \frac{\partial v}{\partial y} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial y}\ v_i \tag{21}$$

$$\dot{\epsilon}_{xy}(x, y) = \frac{1}{2}\frac{\partial u}{\partial y} + \frac{1}{2}\frac{\partial v}{\partial x} = \frac{1}{2}\sum_{i=1}^{4} \frac{\partial N_i}{\partial y}\ u_i + \frac{1}{2}\sum_{i=1}^{4} \frac{\partial N_i}{\partial x}\ v_i \tag{22}$$

How we actually obtain the exact form of the basis functions is explained in the coming section.

3.5.1 Bilinear basis functions in 2D (Q_1)

In this section, we place ourselves in the most favorables case, i.e. the element is a square defined by $-1 < r < 1$, $-1 < s < 1$ in the Cartesian coordinates system (r, s):



This element is commonly called the reference element. How we go from the (x, y) coordinate system to the (r, s) once and vice versa will be dealt later on. For now, the basis functions in the above reference element and in the reduced coordinates system (r, s) are given by:

$$\begin{aligned} N_1(r,s) &= 0.25(1-r)(1-s) \\ N_2(r,s) &= 0.25(1+r)(1-s) \\ N_3(r,s) &= 0.25(1+r)(1+s) \\ N_4(r,s) &= 0.25(1-r)(1+s) \end{aligned}$$

The partial derivatives of these functions with respect to r ans s automatically follow:

$$\begin{aligned} \frac{\partial N_1}{\partial r}(r,s) &= -0.25(1-s) & \frac{\partial N_1}{\partial s}(r,s) &= -0.25(1-r) \\ \frac{\partial N_2}{\partial r}(r,s) &= +0.25(1-s) & \frac{\partial N_2}{\partial s}(r,s) &= -0.25(1+r) \\ \frac{\partial N_3}{\partial r}(r,s) &= +0.25(1+s) & \frac{\partial N_3}{\partial s}(r,s) &= +0.25(1+r) \\ \frac{\partial N_4}{\partial r}(r,s) &= -0.25(1+s) & \frac{\partial N_4}{\partial s}(r,s) &= +0.25(1-r) \end{aligned}$$

Let us go back to Eq.(19). And let us assume that the function $v(r, s) = C$ so that $v_i = C$ for $i = 1, 2, 3, 4$. It then follows that

$$\hat{v}(r,s) = \sum_{i=1}^{4} N_i(r,s)\ v_i = C \sum_{i=1}^{4} N_i(r,s) = C[N_1(r,s) + N_2(r,s) + N_3(r,s) + N_4(r,s)] = C$$

This is a very important property: if the v function used to assign values at the vertices is constant, then the value of $\hat{v}$ *anywhere* in the element is exactly C. If we now turn to the derivatives of v with respect to r and s:

$$\frac{\partial \hat{v}}{\partial r}(r,s) = \sum_{i=1}^{4} \frac{\partial N_i}{\partial r}(r,s)\ v_i = C \sum_{i=1}^{4} \frac{\partial N_i}{\partial r}(r,s) = C\left[-0.25(1-s) + 0.25(1-s) + 0.25(1+s) - 0.25(1+s)\right] = 0$$

$$\frac{\partial \hat{v}}{\partial s}(r,s) = \sum_{i=1}^{4} \frac{\partial N_i}{\partial s}(r,s)\ v_i = C \sum_{i=1}^{4} \frac{\partial N_i}{\partial s}(r,s) = C\left[-0.25(1-r) - 0.25(1+r) + 0.25(1+r) + 0.25(1-r)\right] = 0$$

We reassuringly find that the derivative of a constant field anywhere in the element is exactly zero.

If we now choose $v(r,s) = ar + bs$ with a and b two constant scalars, we find:

$$\begin{aligned}
\hat{v}(r,s) &= \sum_{i=1}^{4} N_i(r,s)\, v_i && (23)\\
&= \sum_{i=1}^{4} N_i(r,s)(ar_i + bs_i) && (24)\\
&= \underbrace{a\sum_{i=1}^{4} N_i(r,s) r_i}_{r} + \underbrace{b\sum_{i=1}^{4} N_i(r,s) s_i}_{s} && (25)\\
&= a\,[0.25(1-r)(1-s)(-1) + 0.25(1+r)(1-s)(+1) + 0.25(1+r)(1+s)(+1) + 0.25(1-r)(1+s)(-1)]\\
&+ b\,[0.25(1-r)(1-s)(-1) + 0.25(1+r)(1-s)(-1) + 0.25(1+r)(1+s)(+1) + 0.25(1-r)(1+s)(+1)]\\
&= a\,[-0.25(1-r)(1-s) + 0.25(1+r)(1-s) + 0.25(1+r)(1+s) - 0.25(1-r)(1+s)]\\
&+ b\,[-0.25(1-r)(1-s) - 0.25(1+r)(1-s) + 0.25(1+r)(1+s) + 0.25(1-r)(1+s)]\\
&= ar + bs && (26)
\end{aligned}$$

verify above eq. This set of bilinear shape functions is therefore capable of exactly representing a bilinear field. The derivatives are:

$$\begin{aligned}
\frac{\partial \hat{v}}{\partial r}(r,s) &= \sum_{i=1}^{4} \frac{\partial N_i}{\partial r}(r,s)\, v_i && (27)\\
&= a\sum_{i=1}^{4} \frac{\partial N_i}{\partial r}(r,s) r_i + b\sum_{i=1}^{4} \frac{\partial N_i}{\partial r}(r,s) s_i && (28)\\
&= a\,[-0.25(1-s)(-1) + 0.25(1-s)(+1) + 0.25(1+s)(+1) - 0.25(1+s)(-1)]\\
&+ b\,[-0.25(1-s)(-1) + 0.25(1-s)(-1) + 0.25(1+s)(+1) - 0.25(1+s)(+1)]\\
&= \frac{a}{4}\,[(1-s) + (1-s) + (1+s) + (1+s)]\\
&+ \frac{b}{4}\,[(1-s) - (1-s) + (1+s) - (1+s)]\\
&= a && (29)
\end{aligned}$$

Here again, we find that the derivative of the bilinear field inside the element is exact: $\frac{\partial \hat{v}}{\partial r} = \frac{\partial v}{\partial r}$.

However, following the same methodology as above, one can easily prove that this is no more true for polynomials of degree strivtly higher than 1. This fact has serious consequences: if the solution to the problem at hand is for instance a parabola, the Q_1 shape functions cannot represent the solution properly, but only by approximating the parabola in each element by a line. As we will see later, Q_2 basis functions can remedy this problem by containing themselves quadratic terms.

3.5.2 Biquadratic basis functions in 2D (Q_2)

Inside an element the local numbering of the nodes is as follows:

```
3=====6=====2
|     |     |   (r_0,s_0)=(-1,-1)   (r_4,s_4)=( 0,-1)
|     |     |   (r_1,s_1)=(+1,-1)   (r_5,s_5)=(+1, 0)
7=====8=====5   (r_2,s_2)=(+1,+1)   (r_6,s_6)=( 0,+1)
|     |     |   (r_3,s_3)=(-1,+1)   (r_7,s_7)=(-1, 0)
|     |     |                       (r_8,s_8)=( 0, 0)
0=====4=====1
```

The velocity shape functions are then given by:

$$
\begin{aligned}
N_0(r,s) &= \frac{1}{2}r(r-1)\frac{1}{2}s(s-1) \\
N_1(r,s) &= \frac{1}{2}r(r+1)\frac{1}{2}s(s-1) \\
N_2(r,s) &= \frac{1}{2}r(r+1)\frac{1}{2}s(s+1) \\
N_3(r,s) &= \frac{1}{2}r(r-1)\frac{1}{2}s(s+1) \\
N_4(r,s) &= (1-r^2)\frac{1}{2}s(s-1) \\
N_5(r,s) &= \frac{1}{2}r(r+1)(1-s^2) \\
N_6(r,s) &= (1-r^2)\frac{1}{2}s(s+1) \\
N_7(r,s) &= \frac{1}{2}r(r-1)(1-s^2) \\
N_8(r,s) &= (1-r^2)(1-s^2)
\end{aligned}
$$

and their derivatives by:

$$
\begin{aligned}
\frac{\partial N_0}{\partial r} &= \frac{1}{2}(2r-1)\frac{1}{2}s(s-1) & \frac{\partial N_0}{\partial s} &= \frac{1}{2}r(r-1)\frac{1}{2}(2s-1) \\
\frac{\partial N_1}{\partial r} &= \frac{1}{2}(2r+1)\frac{1}{2}s(s-1) & \frac{\partial N_1}{\partial s} &= \frac{1}{2}r(r+1)\frac{1}{2}(2s-1) \\
\frac{\partial N_2}{\partial r} &= \frac{1}{2}(2r+1)\frac{1}{2}s(s+1) & \frac{\partial N_2}{\partial s} &= \frac{1}{2}r(r+1)\frac{1}{2}(2s+1) \\
\frac{\partial N_3}{\partial r} &= \frac{1}{2}(2r-1)\frac{1}{2}s(s+1) & \frac{\partial N_3}{\partial s} &= \frac{1}{2}r(r-1)\frac{1}{2}(2s+1) \\
\frac{\partial N_4}{\partial r} &= (-2r)\frac{1}{2}s(s-1) & \frac{\partial N_4}{\partial s} &= (1-r^2)\frac{1}{2}(2s-1) \\
\frac{\partial N_5}{\partial r} &= \frac{1}{2}(2r+1)(1-s^2) & \frac{\partial N_5}{\partial s} &= \frac{1}{2}r(r+1)(-2s) \\
\frac{\partial N_6}{\partial r} &= (-2r)\frac{1}{2}s(s+1) & \frac{\partial N_6}{\partial s} &= (1-r^2)\frac{1}{2}(2s+1) \\
\frac{\partial N_7}{\partial r} &= \frac{1}{2}(2r-1)(1-s^2) & \frac{\partial N_7}{\partial s} &= \frac{1}{2}r(r-1)(-2s) \\
\frac{\partial N_8}{\partial r} &= (-2r)(1-s^2) & \frac{\partial N_8}{\partial s} &= (1-r^2)(-2s)
\end{aligned}
$$

3.5.3 Bicubic basis functions in 2D (Q_3)

Inside an element the local numbering of the nodes is as follows:

```
12===13===14===15   (r,s)_{00}=(-1,-1)      (r,s)_{08}=(-1,+1/3)
||   ||   ||   ||   (r,s)_{01}=(-1/3,-1)    (r,s)_{09}=(-1/3,+1/3)
08===09===10===11   (r,s)_{02}=(+1/3,-1)    (r,s)_{10}=(+1/3,+1/3)
||   ||   ||   ||   (r,s)_{03}=(+1,-1)      (r,s)_{11}=(+1,+1/3)
04===05===06===07   (r,s)_{04}=(-1,-1/3)    (r,s)_{12}=(-1,+1)
||   ||   ||   ||   (r,s)_{05}=(-1/3,-1/3)  (r,s)_{13}=(-1/3,+1)
00===01===02===03   (r,s)_{06}=(+1/3,-1/3)  (r,s)_{14}=(+1/3,+1)
                    (r,s)_{07}=(+1,-1/3)    (r,s)_{15}=(+1,+1)
```

The velocity shape functions are given by:

$$N_1(r) = (-1 + r + 9r^2 - 9r^3)/16 \qquad N_1(t) = (-1 + t + 9t^2 - 9t^3)/16$$
$$N_2(r) = (+9 - 27r - 9r^2 + 27r^3)/16 \qquad N_2(t) = (+9 - 27t - 9t^2 + 27t^3)/16$$
$$N_3(r) = (+9 + 27r - 9r^2 - 27r^3)/16 \qquad N_3(t) = (+9 + 27t - 9t^2 - 27t^3)/16$$
$$N_4(r) = (-1 - r + 9r^2 + 9r^3)/16 \qquad N_4(t) = (-1 - t + 9t^2 + 9t^3)/16$$

$$\begin{aligned}
N_{01}(r,s) &= N_1(r)N_1(s) = (-1 + r + 9r^2 - 9r^3)/16 * (-1 + t + 9s^2 - 9s^3)/16 \\
N_{02}(r,s) &= N_2(r)N_1(s) = (+9 - 27r - 9r^2 + 27r^3)/16 * (-1 + t + 9s^2 - 9s^3)/16 \\
N_{03}(r,s) &= N_3(r)N_1(s) = (+9 + 27r - 9r^2 - 27r^3)/16 * (-1 + t + 9s^2 - 9s^3)/16 \\
N_{04}(r,s) &= N_4(r)N_1(s) = (-1 - r + 9r^2 + 9r^3)/16 * (-1 + t + 9s^2 - 9s^3)/16 \\
N_{05}(r,s) &= N_1(r)N_2(s) = (-1 + r + 9r^2 - 9r^3)/16 * (9 - 27s - 9s^2 + 27s^3)/16 \\
N_{06}(r,s) &= N_2(r)N_2(s) = (+9 - 27r - 9r^2 + 27r^3)/16 * (9 - 27s - 9s^2 + 27s^3)/16 \\
N_{07}(r,s) &= N_3(r)N_2(s) = (+9 + 27r - 9r^2 - 27r^3)/16 * (9 - 27s - 9s^2 + 27s^3)/16 \\
N_{08}(r,s) &= N_4(r)N_2(s) = (-1 - r + 9r^2 + 9r^3)/16 * (9 - 27s - 9s^2 + 27s^3)/16 \\
N_{09}(r,s) &= N_1(r)N_3(s) = \qquad (30) \\
N_{10}(r,s) &= N_2(r)N_3(s) = \qquad (31) \\
N_{11}(r,s) &= N_3(r)N_3(s) = \qquad (32) \\
N_{12}(r,s) &= N_4(r)N_3(s) = \qquad (33) \\
N_{13}(r,s) &= N_1(r)N_4(s) = \qquad (34) \\
N_{14}(r,s) &= N_2(r)N_4(s) = \qquad (35) \\
N_{15}(r,s) &= N_3(r)N_4(s) = \qquad (36) \\
N_{16}(r,s) &= N_4(r)N_4(s) = \qquad (37)
\end{aligned}$$

4 Solving the Stokes equations with the FEM

In the case of an incompressible flow, we have seen that the continuity (mass conservation) equation takes the simple form $\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$. In other word flow takes place under the constraint that the divergence of its velocity field is exactly zero eveywhere (solenoidal constraint), i.e. it is divergence free.

We see that the pressure in the momentum equation is then a degree of freedom which is needed to satisfy the incompressibilty constraint (and it is not related to any constitutive equation) [28]. In other words the pressure is acting as a Lagrange multiplier of the incompressibility constraint.

Various approaches have been proposed in the literature to deal with the incompressibility constraint but we will only focus on the penalty method (section 4.2) and the so-called mixed finite element method 4.3.

4.1 strong and weak forms

The strong form consists of the governing equation and the boundary conditions, i.e. the mass, momentum and energy conservation equations supplemented with Dirichlet and/or Neumann boundary conditions on (parts of) the boundary.

To develop the finite element formulation, the partial differential equations must be restated in an integral form called the weak form. In essence the PDEs are first multiplied by an arbitrary function and integrated over the domain.

4.2 The penalty approach

In order to impose the incompressibility constraint, two widely used procedures are available, namely the Lagrange multiplier method and the penalty method [6, 44]. The latter is implemented in ELEFANT, which allows for the elimination of the pressure variable from the momentum equation (resulting in a reduction of the matrix size).

Mathematical details on the origin and validity of the penalty approach applied to the Stokes problem can for instance be found in [22], [73] or [42].

The penalty formulation of the mass conservation equation is based on a relaxation of the incompressibility constraint and writes

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} + \frac{p}{\lambda} = 0 \tag{38}$$

where λ is the penalty parameter, that can be interpreted (and has the same dimension) as a bulk viscosity. It is equivalent to say that the material is weakly compressible. It can be shown that if one chooses λ to be a sufficiently large number, the continuity equation $\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$ will be approximately satisfied in the finite element solution. The value of λ is often recommended to be 6 to 7 orders of magnitude larger than the shear viscosity [30, 45].

Equation (38) can be used to eliminate the pressure in Eq. (**??**) so that the mass and momentum conservation equations fuse to become :

$$\boldsymbol{\nabla} \cdot (2\eta\dot{\varepsilon}(\boldsymbol{v})) + \lambda\boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot \boldsymbol{v}) = \rho\boldsymbol{g} = 0 \tag{39}$$

[61] have established the equivalence for incompressible problems between the reduced integration of the penalty term and a mixed Finite Element approach if the pressure nodes coincide with the integration points of the reduced rule.

In the end, the elimination of the pressure unknown in the Stokes equations replaces the original saddle-point Stokes problem [7] by an elliptical problem, which leads to a symmetric positive definite (SPD) FEM matrix. This is the major benefit of the penalized approach over the full indefinite solver with the velocity-pressure variables. Indeed, the SPD character of the matrix lends itself to efficient solving stragegies and is less memory-demanding since it is sufficient to store only the upper half of the matrix including the diagonal [39] .

list codes which use this approach

The stress tensor $\boldsymbol{\sigma}$ is symmetric (*i.e.* $\sigma_{ij} = \sigma_{ji}$). For simplicity I will now focus on a Stokes flow in two dimensions.

Since the penalty formulation is only valid for incompressible flows, then $\dot{\boldsymbol{\epsilon}} = \dot{\boldsymbol{\epsilon}}^d$ so that the d superscript is ommitted in what follows. The stress tensor can also be cast in vector format:

$$\begin{aligned}
\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} &= \begin{pmatrix} -p \\ -p \\ 0 \end{pmatrix} + 2\eta \begin{pmatrix} \dot{\epsilon}_{xx} \\ \dot{\epsilon}_{yy} \\ \dot{\epsilon}_{xy} \end{pmatrix} \\
&= \lambda \begin{pmatrix} \dot{\epsilon}_{xx} + \dot{\epsilon}_{yy} \\ \dot{\epsilon}_{xx} + \dot{\epsilon}_{yy} \\ 0 \end{pmatrix} + 2\eta \begin{pmatrix} \dot{\epsilon}_{xx} \\ \dot{\epsilon}_{yy} \\ \dot{\epsilon}_{xy} \end{pmatrix} \\
&= \left[\lambda \underbrace{\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\boldsymbol{K}} + \eta \underbrace{\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{C}} \right] \cdot \begin{pmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial v}{\partial y} \\ \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{pmatrix}
\end{aligned}$$

Remember that

$$\frac{\partial u}{\partial x} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial x}\, u_i \qquad \frac{\partial v}{\partial y} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial y}\, v_i$$

$$\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial y}\, u_i + \sum_{i=1}^{4} \frac{\partial N_i}{\partial x}\, v_i$$

so that

$$\begin{pmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial v}{\partial y} \\ \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} \\ \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_4}{\partial x} \end{pmatrix}}_{\boldsymbol{B}} \cdot \underbrace{\begin{pmatrix} u1 \\ v1 \\ u2 \\ v2 \\ u3 \\ v3 \\ u4 \\ v4 \end{pmatrix}}_{\boldsymbol{V}}$$

Finally,

$$\vec{\sigma} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} = (\lambda \boldsymbol{K} + \eta \boldsymbol{C}) \cdot \boldsymbol{B} \cdot \boldsymbol{V}$$

We will now establish the weak form of the momentum conservation equation. We start again from

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0}$$

For the N_i's 'regular enough', we can write:

$$\int_{\Omega_e} N_i \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} d\Omega + \int_{\Omega_e} N_i \boldsymbol{b} d\Omega = 0$$

We can integrate by parts and drop the surface term[1]:

$$\int_{\Omega_e} \boldsymbol{\nabla} N_i \cdot \boldsymbol{\sigma} d\Omega = \int_{\Omega_e} N_i \boldsymbol{b} d\Omega$$

or,

$$\int_{\Omega_e} \begin{pmatrix} \frac{\partial N_i}{\partial x} & 0 & \frac{\partial N_i}{\partial y} \\ \\ 0 & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} d\Omega = \int_{\Omega_e} N_i \boldsymbol{b} d\Omega$$

[1] We will come back to this at a later stage

Let $i = 1, 2, 3, 4$ and stack the resulting four equations on top of one another.

$$\int_{\Omega_e} \begin{pmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_1}{\partial y} \\ & & \\ 0 & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} d\Omega = \int_{\Omega_e} N_1 \begin{pmatrix} b_x \\ b_y \end{pmatrix} d\Omega \tag{40}$$

$$\int_{\Omega_e} \begin{pmatrix} \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_2}{\partial y} \\ & & \\ 0 & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} d\Omega = \int_{\Omega_e} N_i \begin{pmatrix} b_x \\ b_y \end{pmatrix} d\Omega \tag{41}$$

$$\int_{\Omega_e} \begin{pmatrix} \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_3}{\partial y} \\ & & \\ 0 & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} d\Omega = \int_{\Omega_e} N_3 \begin{pmatrix} b_x \\ b_y \end{pmatrix} d\Omega \tag{42}$$

$$\int_{\Omega_e} \begin{pmatrix} \frac{\partial N_4}{\partial x} & 0 & \frac{\partial N_4}{\partial y} \\ & & \\ 0 & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} d\Omega = \int_{\Omega_e} N_4 \begin{pmatrix} b_x \\ b_y \end{pmatrix} d\Omega \tag{43}$$

We easily recognize $\boldsymbol{B}^T$ inside the integrals! Let us define

$$\boldsymbol{N}_b^T = (N_1 b_x, N_1 b_y, ...N_4 b_x, N_4 b_y)$$

then we can write

$$\int_{\Omega_e} \boldsymbol{B}^T \cdot \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} d\Omega = \int_{\Omega_e} \boldsymbol{N}_b d\Omega$$

and finally:

$$\int_{\Omega_e} \boldsymbol{B}^T \cdot [\lambda \boldsymbol{K} + \eta \boldsymbol{C}] \cdot \boldsymbol{B} \cdot \boldsymbol{V} d\Omega = \int_{\Omega_e} \boldsymbol{N}_b d\Omega$$

Since V contains the velocities at the corners, it does not depend on the x or y coordinates so it can be taking outside of the integral:

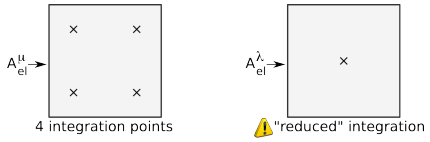
$$\underbrace{\left(\int_{\Omega_e} \boldsymbol{B}^T \cdot [\lambda \boldsymbol{K} + \eta \boldsymbol{C}] \cdot \boldsymbol{B} d\Omega \right)}_{A_{el}(8\times 8)} \cdot \underbrace{\boldsymbol{V}}_{(8x1)} = \underbrace{\int_{\Omega_e} \boldsymbol{N}_b d\Omega}_{B_{el}(8\times 1)}$$

or,

$$\left[\underbrace{\left(\int_{\Omega_e} \lambda \boldsymbol{B}^T \cdot \boldsymbol{K} \cdot \boldsymbol{B} d\Omega \right)}_{A^{\lambda}_{el}(8\times 8)} + \underbrace{\left(\int_{\Omega_e} \eta \boldsymbol{B}^T \cdot \boldsymbol{C} \cdot \boldsymbol{B} d\Omega \right)}_{A^{\eta}_{el}(8\times 8)} \right] \cdot \underbrace{\boldsymbol{V}}_{(8x1)} = \underbrace{\int_{\Omega_e} \boldsymbol{N}_b d\Omega}_{B_{el}(8\times 1)}$$

INTEGRATION - MAPPING
reduced integration [45]

1. partition domain Ω into elements Ω_e, $e = 1, ...n_{el}$.
2. loop over elements and for each element compute $\boldsymbol{A}_{el}$, $\boldsymbol{B}_{el}$



3. a node belongs to several elements
   → need to assemble $\boldsymbol{A}_{el}$ and $\boldsymbol{B}_{el}$ in $\boldsymbol{A}$, $\boldsymbol{B}$
4. apply boundary conditions
5. solve system: $\boldsymbol{x} = \boldsymbol{A}^{-1} \cdot \boldsymbol{B}$
6. visualise/analyse $\boldsymbol{x}$

4.3 The mixed FEM

5 Solving the elastic equations with the FEM

6 Additional techniques

6.1 Picard and Newton

6.2 The SUPG formulation for the energy equation

6.3 Tracking materials and/or interfaces

6.4 Dealing with a free surface

6.5 Convergence criterion for nonlinear iterations

6.6 Static condensation

6.7 The method of manufactured solutions

The method of manufactured solutions is a relatively simple way of carrying out code verification. In essence, one postulates a solution for the PDE at hand (as well as the proper boundary conditions), inserts it in the PDE and computes the corresponding source term. The same source term and boundary conditions will then be used in a numerical simulation so that the computed solution can be compared with the (postulated) true analytical solution.

Examples of this approach are to be found in [28, 15, **?**].

- ▷ *python_codes/fieldstone*
- ▷ *python_codes/fieldstone_saddlepoint*
- ▷ *python_codes/fieldstone_saddlepoint_q2q1*
- ▷ *python_codes/fieldstone_saddlepoint_q3q2*
- ▷ *python_codes/fieldstone_burstedde*

6.7.1 Analytical benchmark I

Taken from [30].

$$\begin{aligned} u(x,y) &= x^2(1-x)^2(2y-6y^2+4y^3) \\ v(x,y) &= -y^2(1-y)^2(2x-6x^2+4x^3) \\ p(x,y) &= x(1-x)-1/6 \end{aligned}$$

Note that the pressure obeys $\int_\Omega p\, d\Omega = 0$

The corresponding components of the body force $\boldsymbol{b}$ are prescribed as

$$\begin{aligned} b_x &= (12-24y)x^4+(-24+48y)x^3+(-48y+72y^2-48y^3+12)x^2 \\ &\quad +(-2+24y-72y^2+48y^3)x+1-4y+12y^2-8y^3 \\ b_y &= (8-48y+48y^2)x^3+(-12+72y-72y^2)x^2 \\ &\quad +(4-24y+48y^2-48y^3+24y^4)x-12y^2+24y^3-12y^4 \end{aligned}$$

With this prescribed body force, the exact solution is

6.7.2 Analytical benchmark II

Taken from [27]

$$u(x,y) = x+x^2-2xy+x^3-3xy^2+x^2y \quad (44)$$

$$v(x,y) = -y-2xy+y^2-3x^2y+y^3-xy^2 \quad (45)$$

$$p(x,y) = xy+x+y+x^3y^2-4/3 \quad (46)$$

Note that the pressure obeys $\int_\Omega p\, d\Omega = 0$

$$b_x = -(1+y-3x^2y^2) \quad (47)$$

$$b_y = -(1-3x-2x^3y) \quad (48)$$

6.8 Assigning values to quadrature points

As we have seen in Section **??**, the building of the elemental matrix and rhs requires (at least) to assign a density and viscosity value to each quadrature point inside the element. Depending on the type of modelling, this task can prove more complex than one might expect and have large consequences on the solution accuracy.

Here are several options:

- The simplest way (which is often used for benchmarks) consists in computing the 'real' coordinates (x_q, y_q, z_q) of a given quadrature point based on its reduced coordinates (r_q, s_q, t_q), and passing these coordinates to a function which returns density and/or viscosity at this location. For instance, for the Stokes sphere:

```
def rho(x,y):
    if (x-.5)**2+(y-0.5)**2<0.123**2:
       val=2.
    else:
       val=1.
    return val

def mu(x,y):
    if (x-.5)**2+(y-0.5)**2<0.123**2:
       val=1.e2
    else:
       val=1.
    return val
```

  This is very simple, but it has been shown to potentially be problematic. In essence, it can introduce very large contrasts inside a single element and perturb the quadrature. Please read section 3.3 of [43] and/or have a look at the section titled "Averaging material properties" in the ASPECT manual.

- another similar approach consists in assigning a density and viscosity value to the nodes of the FE mesh first, and then using these nodal values to assign values to the quadrature points. Very often ,and quite logically, the shape functions are used to this effect. Indeed we have seen before that for any point (r, s, t) inside an element we have

$$f_h(r, s, t) = \sum_i^m f_i N_i(r, s, t)$$

  where the f_i are the nodal values and the N_i the corresponding basis functions.

  In the case of linear elements (Q_1 basis functions), this is straightforward. In fact, the basis functions N_i can be seen as moving weights: the closer the point is to a node, the higher the weight (basis function value).

  However, this is quite another story for quadratic elements (Q_2 basis functions). In order to illustrate the problem, let us consider a 1D problem. The basis functions are

$$N_1(r) = \frac{1}{2}r(r-1) \qquad N_2(r) = 1 - r^2 \qquad N_3(r) = \frac{1}{2}r(r+1)$$

  Let us further assign: $\rho_1 = \rho_2 = 0$ and $\rho_3 = 1$. Then

$$\rho_h(r) = \sum_i^m \rho_i N_i(r) = N_3(r)$$

  There lies the core of the problem: the $N_3(r)$ basis function is negative for $r \in [-1, 0]$. This means that the quadrature point in this interval will be assigned a negative density, which is nonsensical and numerically problematic!

  use 2X Q1. write about it !

The above methods work fine as long as the domain contains a single material. As soon as there are multiple fluids in the domain a special technique is needed to track either the fluids themselves or their interfaces. Let us start with markers. We are then confronted to the infernal trio (a *menage a trois*?) which is present for each element, composed of its nodes, its markers and its quadrature points.

Each marker carries the material information (density and viscosity). This information must ultimately be projected onto the quadrature points. Two main options are possible: an algorithm is designed and projects the marker-based fields onto the quadrature points directly or the marker fields are first projected onto the FE nodes and then onto the quadrature points using the techniques above.

At a given time, every element e contains n^e markers. During the FE matrix building process, viscosity and density values are needed at the quadrature points. One therefore needs to project the values carried by the markers at these locations. Several approaches are currently in use in the community and the topic has been investigated by [26] and [31] for instance.

ELEFANT adopts a simple approach: viscosity and density are considered to be elemental values, i.e. all the markers within a given element contribute to assign a unique constant density and viscosity value to the element by means of an averaging scheme.

While it is common in the literature to treat the so-called arithmetic, geometric and harmonic means as separate averagings, I hereby wish to introduce the notion of generalised mean, which is a family of functions for aggregating sets of numbers that include as special cases the arithmetic, geometric and harmonic means.

If p is a non-zero real number, we can define the generalised mean (or power mean) with exponent p of the positive real numbers a_1, ... a_n as:

$$M_p(a_1, ...a_n) = \left(\frac{1}{n} \sum_{i=1}^{n} a_i^p \right)^{1/p} \tag{49}$$

and it is trivial to verify that we then have the special cases:

$$M_{-\infty} = \lim_{p \to -\infty} M_p = \min(a_1, ...a_n) \quad \text{(minimum)} \tag{50}$$

$$M_{-1} = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_n}} \qquad \text{(harm. avrg.)} \tag{51}$$

$$M_0 = \lim_{p \to 0} M_p = \left(\prod_{i=1}^{n} a_i \right)^{1/n} \qquad \text{(geom. avrg.)} \tag{52}$$

$$M_{+1} = \frac{1}{n} \sum_{i=1}^{n} a_i \qquad \text{(arithm. avrg.)} \tag{53}$$

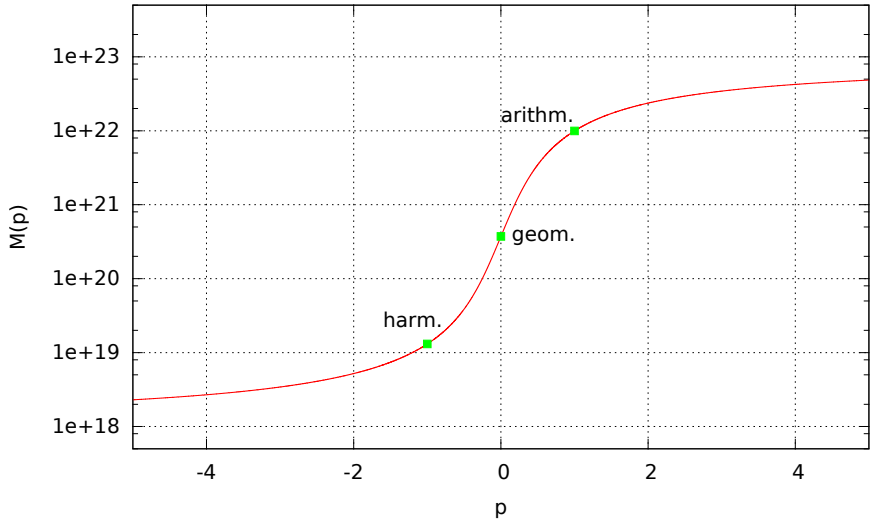
$$M_{+2} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} a_i^2} \qquad \text{(root mean square)} \tag{54}$$

$$M_{+\infty} = \lim_{p \to +\infty} M_p = \max(a_1, ...a_n) \quad \text{(maximum)} \tag{55}$$

Note that the proofs of the limit convergence are given in [14].

An interesting property of the generalised mean is as follows: for two real values p and q, if $p < q$ then $M_p \leq M_q$. This property has for instance been illustrated in Fig. 20 of [80].

One can then for instance look at the generalised mean of a randomly generated set of 1000 viscosity values within $10^{18} Pa.s$ and $10^{23} Pa.s$ for $-5 \leq p \leq 5$. Results are shown in the figure hereunder and the arithmetic, geometric and harmonic values are indicated too. The function M_p assumes an arctangent-like shape: very low values of p will ultimately yield the minimum viscosity in the array while very high values will yield its maximum. In between, the transition is smooth and occurs essentially for $|p| \leq 5$.



▷ *python_codes/fieldstone_markers_avrg*

6.9 Matrix (Sparse) storage

The FE matrix is the result of the assembly process of all elemental matrices. Its size can become quite large when the resolution is being increased (from thousands of lines/columns to tens of millions).

One important property of the matrix is its sparsity. Typically less than 1% of the matrix terms is not zero and this means that the matrix storage can and should be optimised. Clever storage formats were designed early on since the amount of RAM memory in computers was the limiting factor 3 or 4 decades ago. [76]

There are several standard formats:

- compressed sparse row (CSR) format
- compressed sparse column format (CSC)
- the Coordinate Format (COO)
- Skyline Storage Format
- ...

I focus on the CSR format in what follows.

6.9.1 2D domain - One degree of freedom per node

Let us consider again the 3×2 element grid which counts 12 nodes.

```
8=======9======10======11
|       |       |       |
|  (3)  |  (4)  |  (5)  |
|       |       |       |
4=======5=======6=======7
|       |       |       |
|  (0)  |  (1)  |  (2)  |
|       |       |       |
0=======1=======2=======3
```

In the case there is only a single degree of freedom per node, the assembled FEM matrix will look like this:

$$\begin{pmatrix}
X & X & & & X & X & & & & & & \\
X & X & X & & X & X & X & & & & & \\
 & X & X & X & & X & X & X & & & & \\
 & & X & X & & & X & X & & & & \\
X & X & & & X & X & & & X & X & & \\
X & X & X & & X & X & X & & X & X & X & \\
 & X & X & X & & X & X & X & & X & X & X \\
 & & X & X & & & X & X & & & X & X \\
 & & & & X & X & & & X & X & & \\
 & & & & X & X & X & & X & X & X & \\
 & & & & & X & X & X & & X & X & X \\
 & & & & & & X & X & & & X & X
\end{pmatrix}$$

where the X stand for non-zero terms. This matrix structure stems from the fact that

- node 0 sees nodes 0,1,4,5
- node 1 sees nodes 0,1,2,4,5,6
- node 2 sees nodes 1,2,3,5,6,7
- ...

- node 5 sees nodes 0,1,2,4,5,6,8,9,10
- ...
- node 10 sees nodes 5,6,7,9,10,11
- node 11 sees nodes 6,7,10,11

In light thereof, we have

- 4 corner nodes which have 4 neighbours (counting themselves)
- 2(nnx-2) nodes which have 6 neighbours
- 2(nny-2) nodes which have 6 neighbours
- (nnx-2)×(nny-2) nodes which have 9 neighbours

In total, the number of non-zero terms in the matrix is then:

$$NZ = 4 \times 4 + 4 \times 6 + 2 \times 6 + 2 \times 9 = 70$$

In general, we would then have:

$$NZ = 4 \times 4 + [2(nnx - 2) + 2(nny - 2)] \times 6 + (nnx - 2)(nny - 2) \times 9$$

Let us temporarily assume $nnx = nny = n$. Then the matrix size (total number of unknowns) is $N = n^2$ and

$$NZ = 16 + 24(n - 2) + 9(n - 2)^2$$

A full matrix array would contain $N^2 = n^4$ terms. The ratio of NZ (the actual number of reals to store) to the full matrix size (the number of reals a full matrix contains) is then

$$R = \frac{16 + 24(n - 2) + 9(n - 2)^2}{n^4}$$

It is then obvious that when n is large enough $R \sim 1/n^2$.

CSR stores the nonzeros of the matrix row by row, in a single indexed array A of double precision numbers. Another array COLIND contains the column index of each corresponding entry in the A array. A third integer array RWPTR contains pointers to the beginning of each row, which an additional pointer to the first index following the nonzeros of the matrix A. A and COLIND have length NZ and RWPTR has length N+1.

In the case of the here-above matrix, the arrays COLIND and RWPTR will look like:

$$COLIND = (0, 1, 4, 5,\ 0, 1, 2, 4, 5, 6,\ 1, 2, 3, 5, 6, 7, ..., 6, 7, 10, 11)$$

$$RWPTR = (0, 4, 10, 16, ...)$$

6.9.2 2D domain - Two degrees of freedom per node

When there are now two degrees of freedom per node, such as in the case of the Stokes equation in two-dimensions, the size of the $\mathbb{K}$ matrix is given by

NfemV=nnp∗ndofV

In the case of the small grid above, we have `NfemV=24`. Elemental matrices are now 8×8 in size.

We still have

- 4 corner nodes which have 4 ,neighbours,
- 2(nnx-2) nodes which have 6 neighbours
- 2(nny-2) nodes which have 6 neighbours

- (nnx-2)x(nny-2) nodes which have 9 neighbours,

but now each degree of freedom from a node sees the other two degrees of freedom of another node too. In that case, the number of nonzeros has been multiplied by four and the assembled FEM matrix looks like:

$$\begin{pmatrix}
X & X & X & X & & & & & X & X & X & X & & & & & & & & & & & & \\
X & X & X & X & & & & & X & X & X & X & & & & & & & & & & & & \\
X & X & X & X & X & X & & & X & X & X & X & X & X & & & & & & & & & & \\
X & X & X & X & X & X & & & X & X & X & X & X & X & & & & & & & & & & \\
 & & X & X & X & X & X & X & & & X & X & X & X & X & X & & & & & & & & \\
 & & X & X & X & X & X & X & & & X & X & X & X & X & X & & & & & & & & \\
 & & & & X & X & X & X & & & & & X & X & X & X & & & & & & & & \\
 & & & & X & X & X & X & & & & & X & X & X & X & & & & & & & & \\
X & X & X & X & & & & & X & X & X & X & & & & & X & X & X & X & & & & \\
X & X & X & X & & & & & X & X & X & X & & & & & X & X & X & X & & & & \\
X & X & X & X & X & X & & & X & X & X & X & X & X & & & X & X & X & X & X & X & & \\
X & X & X & X & X & X & & & X & X & X & X & X & X & & & X & X & X & X & X & X & & \\
 & & X & X & X & X & X & X & & & X & X & X & X & X & X & & & X & X & X & X & X & X \\
 & & X & X & X & X & X & X & & & X & X & X & X & X & X & & & X & X & X & X & X & X \\
 & & & & X & X & X & X & & & & & X & X & X & X & & & & & X & X & X & X \\
 & & & & X & X & X & X & & & & & X & X & X & X & & & & & X & X & X & X \\
 & & & & & & & & X & X & X & X & & & & & X & X & X & X & & & & \\
 & & & & & & & & X & X & X & X & & & & & X & X & X & X & & & & \\
 & & & & & & & & X & X & X & X & X & X & & & X & X & X & X & X & X & & \\
 & & & & & & & & X & X & X & X & X & X & & & X & X & X & X & X & X & & \\
 & & & & & & & & & & X & X & X & X & X & X & & & X & X & X & X & X & X \\
 & & & & & & & & & & X & X & X & X & X & X & & & X & X & X & X & X & X \\
 & & & & & & & & & & & & X & X & X & X & & & & & X & X & X & X \\
 & & & & & & & & & & & & X & X & X & X & & & & & X & X & X & X
\end{pmatrix}$$

Note that the degrees of freedom are organised as follows:

$$(u_0, v_0, u_1, v_1, u_2, v_2, ...u_{11}, v_{11})$$

In general, we would then have:

$$NZ = 4\left[4 \times 4 + [2(nnx-2) + 2(nny-2)] \times 6 + (nnx-2)(nny-2) \times 9\right]$$

and in the case of the small grid, the number of non-zero terms in the matrix is then:

$$NZ = 4\left[4 \times 4 + 4 \times 6 + 2 \times 6 + 2 \times 9\right] = 280$$

In the case of the here-above matrix, the arrays COLIND and RWPTR will look like:

$$COLIND = (0, 1, 2, 3, 8, 9, 10, 11,\ 0, 1, 2, 3, 8, 9, 10, 11,\ ...)$$

$$RWPTR = (0, 8, 16, 28, ...)$$

6.9.3 in fieldstone

The majority of the codes have the FE matrix being a full array

```
a_mat = np.zeros((Nfem,Nfem),dtype=np.float64)
```

and it is converted to CSR format on the fly in the solve phase:

```
sol = sps.linalg.spsolve(sps.csr_matrix(a_mat),rhs)
```

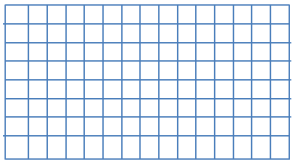
Note that linked list storages can be used (lil_matrix). Substantial memory savings but much longer compute times.

6.10 Mesh generation

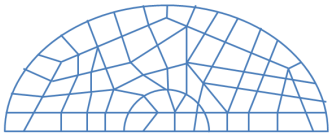
Before basis functions can be defined and PDEs can be discretised and solved we must first tesselate the domain with polygons, e.g. triangles and quadrilaterals in 2D, tetrahedra, prisms and hexahedra in 3D.

When the domain is itself simple (e.g. a rectangle, a sphere, ...) the mesh (or grid) can be (more or less) easily produced and the connectivity array filled with straightforward algorithms [91]. However, real life applications can involve etremely complex geometries (e.g. a bridge, a human spine, a car chassis and body, etc ...) and dedicated algorithms/softwares must be used (see [93, 34, 99]).

We usually distinguish between two broad classes of grids: structured grids (with a regular connectivity) and unstructured grids (with an irregular connectivity).

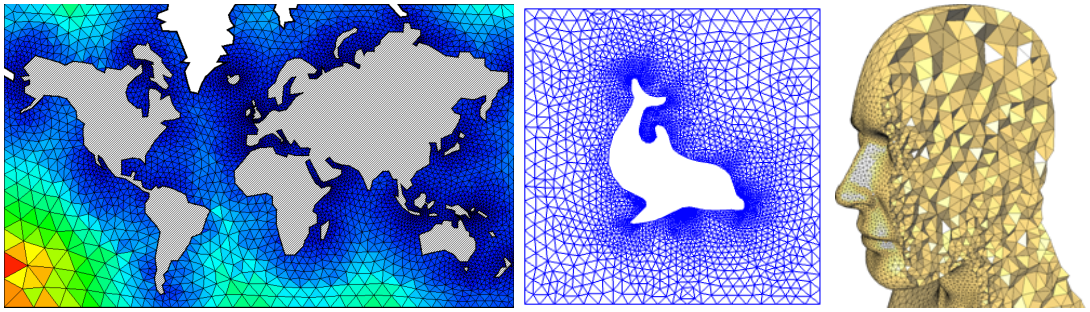


Structured Grid



Unstructured Grid

On the following figure are shown various triangle- tetrahedron-based meshes. These are obviously better suited for simulations of complex geometries:



add more examples coming from geo

Let us now focus on the case of a rectangular computational domain of size `Lx` × `Ly` with a regular mesh composed of `nelx`×`nely`=`nel` quadrilaterals. There are then `nnx`×`nny`=`nnp` grid points. The elements are of size `hx`×`hy` with `hx`=`Lx`/`nelx`.

We have no reason to come up with an irregular/ilogical node numbering so we can number nodes row by row or column by column as shown on the example hereunder of a 3×2 grid:

```
8=======9======10======11       2=======5=======8======11
|       |       |       |       |       |       |       |
|  (3)  |  (4)  |  (5)  |       |  (1)  |  (3)  |  (5)  |
|       |       |       |       |       |       |       |
4=======5=======6=======7       1=======4=======7======10
|       |       |       |       |       |       |       |
|  (0)  |  (1)  |  (2)  |       |  (0)  |  (2)  |  (4)  |
|       |       |       |       |       |       |       |
0=======1=======2=======3       0=======3=======6=======9

     "row by row"                   "column by column"
```

The numbering of the elements themselves could be done in a somewhat chaotic way but we follow the numbering of the nodes for simplicity. The row by row option is the adopted one in **fieldstone**and the coordinates of the points are computed as follows:

```
x = np.empty(nnp, dtype=np.float64)
y = np.empty(nnp, dtype=np.float64)
```

```
counter = 0
for j in range(0,nny):
    for i in range(0,nnx):
        x[counter]=i*hx
        y[counter]=j*hy
        counter += 1
```

The inner loop has `i` ranging from `0` to `nnx-1` first for `j`=0, 1, ... up to `nny-1` which indeed corresponds to the row by row numbering.

We now turn to the connectivity. As mentioned before, this is a structured mesh so that the so-called connectivity array, named `icon` in our case, can be filled easily. For each element we need to store the node identities of its vertices. Since there are `nel` elements and `m=4` corners, this is a `m`×`nel` array. The algorithm goes as follows:

```
icon =np.zeros((m,nel),dtype=np.int16)
counter = 0
for j in range(0,nely):
    for i in range(0,nelx):
        icon[0,counter] = i + j * nnx
        icon[1,counter] = i + 1 + j * nnx
        icon[2,counter] = i + 1 + (j + 1) * nnx
        icon[3,counter] = i + (j + 1) * nnx
        counter += 1
```

In the case of the 3×2 mesh, the `icon` is filled as follows:

element id→ / node id↓	0	1	2	3	4	5
0	0	1	2	4	5	6
1	1	2	3	5	6	7
2	5	6	7	9	10	11
3	4	5	6	8	9	10

It is to be understood as follows: element #4 is composed of nodes 5, 6, 10 and 9. Note that nodes are always stored in a counter clockwise manner, starting at the bottom left. This is very important since the corresponding basis functions and their derivatives will be labelled accordingly.

In three dimensions things are very similar. The mesh now counts `nelx`×`nely`×`nelz`=`nel` elements which represent a cuboid of size `Lx`×`Ly`×`Lz`. The position of the nodes is obtained as follows:

```
x = np.empty(nnp,dtype=np.float64)
y = np.empty(nnp,dtype=np.float64)
z = np.empty(nnp,dtype=np.float64)
counter=0
for i in range(0,nnx):
    for j in range(0,nny):
        for k in range(0,nnz):
            x[counter]=i*hx
            y[counter]=j*hy
            z[counter]=k*hz
            counter += 1
```

The connectivity array is now of size `m`×`nel` with `m=8`:

```
icon =np.zeros((m,nel),dtype=np.int16)
counter = 0
for i in range(0,nelx):
    for j in range(0,nely):
        for k in range(0,nelz):
            icon[0,counter]=nny*nnz*(i  )+nnz*(j  )+k
            icon[1,counter]=nny*nnz*(i+1)+nnz*(j  )+k
            icon[2,counter]=nny*nnz*(i+1)+nnz*(j+1)+k
            icon[3,counter]=nny*nnz*(i  )+nnz*(j+1)+k
            icon[4,counter]=nny*nnz*(i  )+nnz*(j  )+k+1
            icon[5,counter]=nny*nnz*(i+1)+nnz*(j  )+k+1
            icon[6,counter]=nny*nnz*(i+1)+nnz*(j+1)+k+1
            icon[7,counter]=nny*nnz*(i  )+nnz*(j+1)+k+1
            counter += 1
```

produce drawing of node numbering

6.11 Visco-Plasticity

6.11.1 Tensor invariants

Before we dive into the world of nonlinear rheologies it is necessary to introduce the concept of tensor invariants since they are needed further on. Unfortunately there are many different notations used in the literature and these can prove to be confusing.

Given a tensor $\boldsymbol{T}$, one can compute its (moment) invariants as follows:

- first invariant :

$$\begin{aligned} T_I|^{2D} &= Tr[\boldsymbol{T}] = T_{xx} + T_{yy} \\ T_I|^{3D} &= Tr[\boldsymbol{T}] = T_{xx} + T_{yy} + T_{zz} \end{aligned}$$

- second invariant :

$$\begin{aligned} T_{II}|^{2D} &= \frac{1}{2} Tr[\boldsymbol{T}^2] = \frac{1}{2} \sum_{ij} T_{ij} T_{ji} = \frac{1}{2}(T_{xx}^2 + T_{yy}^2) + T_{xy}^2 \\ T_{II}|^{3D} &= \frac{1}{2} Tr[\boldsymbol{T}^2] = \frac{1}{2} \sum_{ij} T_{ij} T_{ji} = \frac{1}{2}(T_{xx}^2 + T_{yy}^2 + T_{yy}^2) + T_{xy}^2 + T_{xz}^2 + T_{yz}^2 \end{aligned}$$

- third invariant :

$$T_{III} = \frac{1}{3} Tr[\boldsymbol{T}^3] = \frac{1}{3} \sum_{ijk} T_{ij} T_{jk} T_{ki}$$

The implementation of the plasticity criterions relies essentially on the second invariants of the (deviatoric) stress $\boldsymbol{\tau}$ and the (deviatoric) strainrate tensors $\dot{\boldsymbol{\varepsilon}}$:

$$\begin{aligned} \tau_{II}|^{2D} &= \frac{1}{2}(\tau_{xx}^2 + \tau_{yy}^2) + \tau_{xy}^2 \\ &= \frac{1}{4}(\sigma_{xx} - \sigma_{yy})^2 + \sigma_{xy}^2 \\ &= \frac{1}{4}(\sigma_1 - \sigma_2)^2 \end{aligned}$$

$$\begin{aligned} \tau_{II}|^{3D} &= \frac{1}{2}(\tau_{xx}^2 + \tau_{yy}^2 + \tau_{zz}^2) + \tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2 \\ &= \frac{1}{6}\left[(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{xx} - \sigma_{zz})^2\right] + \sigma_{xy}^2 + \sigma_{xz}^2 + \sigma_{yz}^2 \\ &= \frac{1}{6}\left[(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_1 - \sigma_3)^2\right] \end{aligned}$$

$$\begin{aligned} \varepsilon_{II}|^{2D} &= \frac{1}{2}\left[(\dot{\varepsilon}_{xx}^d)^2 + (\dot{\varepsilon}_{yy}^d)^2\right] + (\dot{\varepsilon}_{xy}^d)^2 \\ &= \frac{1}{2}\left[\frac{1}{4}(\dot{\varepsilon}_{xx} - \dot{\varepsilon}_{yy})^2 + \frac{1}{4}(\dot{\varepsilon}_{yy} - \dot{\varepsilon}_{xx})^2\right] + \dot{\varepsilon}_{xy}^2 \\ &= \frac{1}{4}(\dot{\varepsilon}_{xx} - \dot{\varepsilon}_{yy})^2 + \dot{\varepsilon}_{xy}^2 \end{aligned}$$

$$\begin{aligned} \varepsilon_{II}|^{3D} &= \frac{1}{2}\left[(\dot{\varepsilon}_{xx}^d)^2 + (\dot{\varepsilon}_{yy}^d)^2 + (\dot{\varepsilon}_{zz}^d)^2\right] + (\dot{\varepsilon}_{xy}^d)^2 + (\dot{\varepsilon}_{xz}^d)^2 + (\dot{\varepsilon}_{yz}^d)^2 \\ &= \frac{1}{6}\left[(\dot{\epsilon}_{xx} - \dot{\epsilon}_{yy})^2 + (\dot{\epsilon}_{yy} - \dot{\epsilon}_{zz})^2 + (\dot{\epsilon}_{xx} - \dot{\epsilon}_{zz})^2\right] + \dot{\epsilon}_{xy}^2 + \dot{\epsilon}_{xz}^2 + \dot{\epsilon}_{yz}^2 \end{aligned}$$

Note that these (second) invariants are almost always used under a square root so we define:

$$\underline{\tau}_{II} = \sqrt{\tau_{II}} \qquad \underline{\dot{\varepsilon}}_{II} = \sqrt{\dot{\varepsilon}_{II}}$$

Note that these quantities have the same dimensions as their tensor counterparts, i.e. Pa for stresses and s^{-1} for strain rates.

6.11.2 Scalar viscoplasticity

This formulation is quite easy to implement. It is widely used, e.g. [98, 92, 83], and relies on the assumption that a scalar quantity η_p (the 'effective plastic viscosity') exists such that the deviatoric stress tensor

$$\boldsymbol{\tau} = 2\eta_p \dot{\boldsymbol{\varepsilon}} \tag{56}$$

is bounded by some yield stress value Y. From Eq. (56) it follows that $\underline{\tau}_{II} = 2\eta_p \underline{\dot{\varepsilon}}_{II} = Y$ which yields

$$\eta_p = \frac{Y}{2\underline{\dot{\varepsilon}}_{II}}$$

This approach has also been coined the Viscosity Rescaling Method (VRM) [50].

insert here the rederivation 2.1.1 of spmw16

It is at this stage important to realise that (i) in areas where the strainrate is low, the resulting effective viscosity will be large, and (ii) in areas where the strainrate is high, the resulting effective viscosity will be low. This is not without consequences since (effective) viscosity contrasts up to 8-10 orders of magnitude have been observed/obtained with this formulation and it makes the FE matrix very stiff, leading to (iterative) solver convergence issues. In order to contain these viscosity contrasts one usually resorts to viscosity limiters η_{min} and η_{max} such that

$$\eta_{min} \leq \eta_p \leq \eta_{max}$$

Caution must be taken when choosing both values as they may influence the final results.

▷ *python_codes/fieldstone_indentor*

6.11.3 about the yield stress value Y

In geodynamics the yield stress value is often given as a simple function. It can be constant (in space and time) and in this case we are dealing with a von Mises plasticity yield criterion. . We simply assume $Y_{vM} = C$ where C is a constant cohesion independent of pressure, strainrate, deformation history, etc ...

Another model is often used: the Drucker-Prager plasticity model. A friction angle ϕ is then introduced and the yield value Y takes the form

$$Y_{DP} = p \sin\phi + C \cos\phi$$

and therefore depends on the pressure p. Because ϕ is with the range $[0°, 45°]$, Y is found to increase with depth (since the lithostatic pressure often dominates the overpressure).

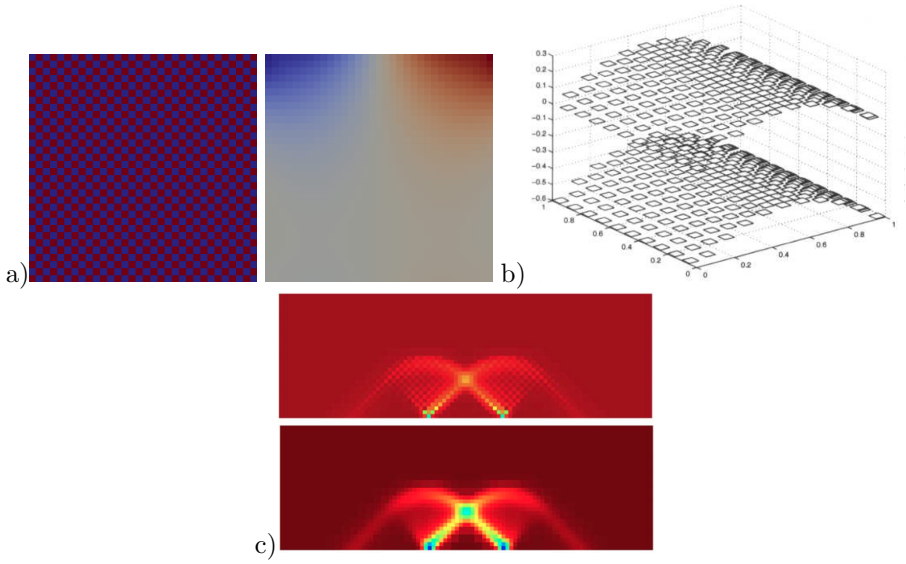
Note that a slightly modified verion of this plasticity model has been used: the total pressure p is then replaced by the lithostatic pressure p_{lith}.

6.12 Pressure smoothing

It has been widely documented that the use of the $Q_1 \times P_0$ element is not without problems. Aside from the consequences it has on the FE matrix properties, we will here focus on another unavoidable side effect: the spurious pressure checkerboard modes.

These modes have been thoroughly analysed [41, 17, 77, 78]. They can be filtered out [17] or simply smoothed [55].

On the following figure (a,b), pressure fields for the lid driven cavity experiment are presented for both an even and un-even number of elements. We see that the amplitude of the modes can sometimes be so large that the 'real' pressure is not visible and that something as simple as the number of elements in the domain can trigger those or not at all.



a) element pressure for a 32x32 grid and for a 33x33 grid;
b) image from [28, p307] for a manufactured solution;
c) elemental pressure and smoothed pressure for the punch experiment [92]

The easiest post-processing step that can be used (especially when a regular grid is used) is explained in [92]: "The element-to-node interpolation is performed by averaging the elemental values from elements common to each node; the node-to-element interpolation is performed by averaging the nodal values element-by-element. This method is not only very efficient but produces a smoothing of the pressure that is adapted to the local density of the octree. Note that these two steps can be repeated until a satisfying level of smoothness (and diffusion) of the pres- sure field is attained."

In the codes which rely on the $Q_1 \times P_0$ element, the (elemental) pressure is simply defined as

```
p=np.zeros(nel,dtype=np.float64)
```

while the nodal pressure is then defined as

```
q=np.zeros(nnp,dtype=np.float64)
```

The element-to-node algorithm is then simply (in 2D):

```
count=np.zeros(nnp,dtype=np.int16)
for iel in range(0,nel):
    q[icon[0,iel]]+=p[iel]
    q[icon[1,iel]]+=p[iel]
    q[icon[2,iel]]+=p[iel]
    q[icon[3,iel]]+=p[iel]
    count[icon[0,iel]]+=1
    count[icon[1,iel]]+=1
    count[icon[2,iel]]+=1
    count[icon[3,iel]]+=1
q=q/count
```

Pressure smoothing is further discussed in [45].

produce figure to explain this

link to proto paper

link to least square and nodal derivatives

6.13 Pressure scaling

As perfectly explained in the step 32 of deal.ii[2], we often need to scale the $\mathbb{G}$ term since it is many orders of magnitude smaller than $\mathbb{K}$, which introduces large inaccuracies in the solving process to the point that the solution is nonsensical. This scaling coefficient is η/L. We start from

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

and introduce the scaling coefficient as follows (which in fact does not alter the solution at all):

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

We then end up with the modified Stokes system:

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

where

$$\underline{\mathbb{G}} = \frac{\eta}{L}\mathbb{G} \qquad \underline{\mathcal{P}} = \frac{L}{\eta}\mathcal{P} \qquad \underline{h} = \frac{\eta}{L}h$$

After the solve phase, we recover the real pressure with $\mathcal{P} = \frac{\eta}{L}\mathcal{P}'$.

[2] https://www.dealii.org/9.0.0/doxygen/deal.II/step_32.html

6.14 Pressure normalisation

When Dirichlet boundary conditions are imposed everywhere on the boundary, pressure is only present by its gradient in the equations. It is thus determined up to an arbitrary constant. In such a case, one commonly impose the average of the pressure over the whole domain or on a subsect of the boundary to be have a zero average, i.e.

$$\int_\Omega p dV = 0$$

Another possibility is to impose the pressure value at a single node.

Let us assume that we are using $Q_1 \times P_0$ elements. Then the pressure is constant inside each element. The integral above becomes:

$$\int_\Omega p dV = \sum_e \int_{\Omega_e} p dV = \sum_e p_e \int_{\Omega_e} dV = \sum_e p_e A_e =$$

where the sum runs over all elements e of area A_e. This can be rewritten

$$\mathbb{L}^T \mathcal{P} = 0$$

and it is a constraint on the pressure. As we have seen before **??**, we can associate to it a Lagrange multiplier λ so that we must solve the modified Stokes system:

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

When higher order spaces are used for pressure (continuous or discontinuous) one must then carry out the above integration numerically by means of (usually) a Gauss-Legendre quadrature.

- ▷ *python_codes/fieldstone_stokes_sphere_3D_saddlepoint/*
- ▷ *python_codes/fieldstone_burstedde/*
- ▷ *python_codes/fieldstone_busse/*
- ▷ *python_codes/fieldstone_RTinstability1/*
- ▷ *python_codes/fieldstone_saddlepoint_q2q1/*
- ▷ *python_codes/fieldstone_compressible2/*
- ▷ *python_codes/fieldstone_compressible1/*
- ▷ *python_codes/fieldstone_saddlepoint_q3q2/*

6.15 The choice of solvers

Let us first look at the penalty formulation. In this case we are only solving for velocity since pressure is recovered in a post-processing step. We also know that the penalty factor is many orders of magnitude higher than the viscosity and in combination with the use of the $Q_1 \times P_0$ element the resulting matrix condition number is very high so that the use of iterative solvers in precluded. Indeed codes such as SOPALE [35], DOUAR [11], or FANTOM [90] relying on the penalty formulation all use direct solvers (BLKFCT, MUMPS, WSMP).

The main advantage of direct solvers is used in this case: They can solve ill-conditioned matrices. However memory requirements for the storage of number of nonzeros in the Cholesky matrix grow very fast as the number of equations/grid size increases, especially in 3D, to the point that even modern computers with tens of Gb of RAM cannot deal with a 100^3 element mesh. This explains why direct solvers are often used for 2D problems and rarely in 3D with noticeable exceptions [92, 100, 12, 60, 2, 3, 4, 97, 66].

In light of all this, let us start again from the (full) Stokes system:

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

We need to solve this system in order to obtain the solution, i.e. the $\mathcal{V}$ and $\mathcal{P}$ vectors. But how?

Unfortunately, this question is not simple to answer and the 'right' depends on many parameters. How big is the matrix ? what is the condition number of the matrix $\mathbb{K}$? Is it symmetric ? (i.e. how are compressible terms handled?).

6.15.1 The Schur complement approach

Let us write the above system as two equations:

$$\mathbb{K}\mathcal{V} + \mathbb{G}\mathcal{P} = f \quad (57)$$
$$\mathbb{G}^T\mathcal{V} = h \quad (58)$$

The first line can be re-written $\mathcal{V} = \mathbb{K}^{-1}(f - \mathbb{G}\mathcal{P})$ and can be inserted in the second:

$$\mathbb{G}^T\mathcal{V} = \mathbb{G}^T[\mathbb{K}^{-1}(f - \mathbb{G}\mathcal{P})] = h$$

or,

$$\mathbb{G}^T K^{-1}\mathbb{G}\mathcal{P} = \mathbb{G}^T\mathbb{K}^{-1}f - h$$

The matrix $\mathbb{S} = \mathbb{G}^T\mathbb{K}^{-1}\mathbb{G}$ is called the Schur complement. It is Symmetric (since $\mathbb{K}$ is symmetric) and Postive-Definite[3] (SPD) if $Ker(\mathbb{G}) = 0$. look in donea-huerta book for details Having solved this equation (we have obtained $\mathcal{P}$), the velocity can be recovered by solving $\mathbb{K}\mathcal{V} = f - \mathbb{G}\mathcal{P}$. For now, let us assume that we have built the $\mathbb{S}$ matrix and the right hand side $\tilde{f} = \mathbb{G}^T\mathbb{K}^{-1}f - h$. We must solve $\mathbb{S}\mathcal{P} = \tilde{f}$.

Since $\mathbb{S}$ is SPD, the Conjugate Gradient (CG) method is very appropriate to solve this system. Indeed, looking at the definition of Wikipedia: *In mathematics, the conjugate gradient method is an algorithm for the numerical solution of particular systems of linear equations, namely those whose matrix is symmetric and positive-definite. The conjugate gradient method is often implemented as an iterative algorithm, applicable to sparse systems that are too large to be handled by a direct implementation or other direct methods such as the Cholesky decomposition. Large sparse systems often arise when numerically solving partial differential equations or optimization problems.*

The Conjugate Gradient algorithm is as follows:

[3] M positive definite $\iff x^T Mx > 0 \; \forall \; x \in \mathbb{R}^n \setminus \mathbf{0}$

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
$\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
$\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

Algorithm obtained from `https://en.wikipedia.org/wiki/Conjugate_gradient_method`

Let us look at this algorithm up close. The parts which may prove to be somewhat tricky are those involving the matrix (in our case the Schur complement). We start the iterations with a guess pressure $\mathcal{P}_0$ (and an initial guess velocity which could be obtained by solving $\mathbb{K}\mathcal{V}_0 = f - \mathbb{G}\mathcal{P}_0$).

$$\begin{aligned} r_0 &= \tilde{f} - \mathbb{S}\mathcal{P}_0 && (59)\\ &= \mathbb{G}^T\mathbb{K}^{-1}f - h - (\mathbb{G}^T\mathbb{K}^{-1}\mathbb{G})\mathcal{P}_0 && (60)\\ &= \mathbb{G}^T\mathbb{K}^{-1}(f - \mathbb{G}\mathcal{P}_0) - h && (61)\\ &= \mathbb{G}^T\mathbb{K}^{-1}\mathbb{K}\mathcal{V}_0 - h && (62)\\ &= \mathbb{G}^T\mathcal{V}_0 - h && (63)\\ & && (64) \end{aligned}$$

We now turn to the α_k coefficient:

$$\alpha_k = \frac{r_k^T r_k}{p_k \mathbb{S} p_k} = \frac{r_k^T r_k}{p_k \mathbb{G}^T\mathbb{K}^{-1}\mathbb{G} p_k} = \frac{r_k^T r_k}{(\mathbb{G}p_k)^T\mathbb{K}^{-1}(\mathbb{G}p_k)}$$

We then define $\tilde{p}_k = \mathbb{G}p_k$, so that α_k can be computed as follows:

1. compute $\tilde{p}_k = \mathbb{G}p_k$
2. solve $\mathbb{K}d_k = \tilde{p}_k$
3. compute $\alpha_k = (r_k^T r_k)/(\tilde{p}_k^T d_k)$

Then we need to look at the term $\mathbb{S}p_k$:

$$\mathbb{S}p_k = \mathbb{G}^T\mathbb{K}^{-1}\mathbb{G}p_k = \mathbb{G}^T\mathbb{K}^{-1}\tilde{p}_k = \mathbb{G}^T d_k$$

We can then rewrite the CG algorithm as follows [105]:

- $r_0 = \mathbb{G}^T\mathcal{V}_0 - h$
- if r_0 is sufficiently small, then return $(\mathcal{V}_0, \mathcal{P}_0)$ as the result
- $p_0 = r_0$
- $k = 0$
- repeat
  - compute $\tilde{p}_k = \mathbb{G}p_k$
  - solve $\mathbb{K}d_k = \tilde{p}_k$
  - compute $\alpha_k = (r_k^T r_k)/(\tilde{p}_k^T d_k)$

- $\mathcal{P}_{k+1} = \mathcal{P}_k + \alpha_k p_k$
- $r_{k+1} = r_k - \alpha_k \mathbb{G}^T d_k$
- if r_{k+1} is sufficiently small, then exit loop
- $\beta_k = (r_{k+1}^T r_{k+1})/(r_k^T r_k)$
- $p_{k+1} = r_{k+1} + \beta_k p_k$
- $k = k + 1$

- return $\mathcal{P}_{k+1}$ as result

We see that we have managed to solve the Schur complement equation with the Conjugate Gradient method without ever building the matrix $\mathbb{S}$. Having obtained the pressure solution, we can easily recover the corresponding velocity with $\mathbb{K}\mathcal{V}_{k+1} = f - \mathbb{G}\mathcal{P}_{k+1}$. However, this is rather unfortunate because it requires yet another solve with the $\mathbb{K}$ matrix. As it turns out, we can slightly alter the above algorithm to have it update the velocity as well so that this last solve is unnecessary.

We have

$$\mathcal{V}_{k+1} = \mathbb{K}^{-1}(f - \mathbb{G}\mathcal{P}_{p+1}) = \mathbb{K}^{-1}(f - \mathbb{G}(\mathcal{P}_k + \alpha_k p_k)) = \mathbb{K}^{-1}(f - \mathbb{G}\mathcal{P}_k) - \alpha_k \mathbb{K}^{-1}\mathbb{G}p_k = \mathcal{V} - \alpha_k \mathbb{K}^{-1}\tilde{p}_k = \mathcal{V}_k - \alpha_k d_k$$

and we can insert this minor extra calculation inside the algorithm and get the velocity solution nearly for free. The final CG algorithm is then

solver_cg:

- compute $\mathcal{V}_0 = \mathbb{K}^{-1}(f - \mathbb{G}\mathcal{P}_0)$
- $r_0 = \mathbb{G}^T\mathcal{V}_0 - h$
- if r_0 is sufficiently small, then return $(\mathcal{V}_0, \mathcal{P}_0)$ as the result
- $p_0 = r_0$
- $k = 0$
- repeat
  - compute $\tilde{p}_k = \mathbb{G}p_k$
  - solve $\mathbb{K}d_k = \tilde{p}_k$
  - compute $\alpha_k = (r_k^T r_k)/(\tilde{p}_k^T d_k)$
  - $\mathcal{P}_{k+1} = \mathcal{P}_k + \alpha_k p_k$
  - $\mathcal{V}_{k+1} = \mathcal{V}_k - \alpha_k d_k$
  - $r_{k+1} = r_k - \alpha_k \mathbb{G}^T d_k$
  - if r_{k+1} is sufficiently small ($|r_{k+1}|_2/|r_0|_2 < tol$), then exit loop
  - $\beta_k = (r_{k+1}^T r_{k+1})/(r_k^T r_k)$
  - $p_{k+1} = r_{k+1} + \beta_k p_k$
  - $k = k + 1$
- return $\mathcal{P}_{k+1}$ as result

This iterative algorithm will converge to the solution with a rate which depends on the condition number of the $\mathbb{S}$ matrix, which is not easily obtainable since $\mathbb{S}$ is never built. Also, we know that large viscosity contrasts in the domain will influence this too. Finally, we notice that this algorithm requires one solve with matrix $\mathbb{K}$ per iteration but says nothing about the method employed to do so.

One thing we know improves the convergence of any iterative solver is the use of a preconditioner matrix and therefore use the Preconditioned Conjugate Gradient method:

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{z}_0 := \mathbf{M}^{-1}\mathbf{r}_0$
$\mathbf{p}_0 := \mathbf{z}_0$
$k := 0$
repeat
$\alpha_k := \dfrac{\mathbf{r}_k^\mathsf{T}\mathbf{z}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small **then** exit loop **end if**
$\mathbf{z}_{k+1} := \mathbf{M}^{-1}\mathbf{r}_{k+1}$
$\beta_k := \dfrac{\mathbf{z}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{z}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
The result is $\mathbf{x}_{k+1}$

Algorithm obtained from `https://en.wikipedia.org/wiki/Conjugate_gradient_method`

In the algorithm above the preconditioner matrix M has to be symmetric positive-definite and fixed, i.e., cannot change from iteration to iteration.

We see that this algorithm introduces an additional vector z and a solve with the matrix M at each iteration, which means that M must be such that solving $Mx = f$ where f is a given rhs vector must be cheap. Ultimately, the PCG algorithm applied to the Schur complement equation takes the form:

solver_pcg:

- compute $\mathcal{V}_0 = \mathbb{K}^{-1}(f - \mathbb{G}\mathcal{P}_0)$
- $r_0 = \mathbb{G}^T\mathcal{V}_0 - h$
- if r_0 is sufficiently small, then return $(\mathcal{V}_0, \mathcal{P}_0)$ as the result
- $z_0 = M^{-1}r_0$
- $p_0 = z_0$
- $k = 0$
- repeat
  - compute $\tilde{p}_k = \mathbb{G}p_k$
  - solve $\mathbb{K}d_k = \tilde{p}_k$
  - compute $\alpha_k = (r_k^T z_k)/(\tilde{p}_k^T d_k)$
  - $\mathcal{P}_{k+1} = \mathcal{P}_k + \alpha_k p_k$
  - $\mathcal{V}_{k+1} = \mathcal{V}_k - \alpha_k d_k$
  - $r_{k+1} = r_k - \alpha_k\mathbb{G}^T d_k$
  - if r_{k+1} is sufficiently small ($|r_{k+1}|_2/|r_0|_2 < tol$), then exit loop
  - $z_{k+1} = M^{-1}r_{k+1}$
  - $\beta_k = (z_{k+1}^T r_{k+1})/(z_k^T r_k)$
  - $p_{k+1} = z_{k+1} + \beta_k p_k$
  - $k = k + 1$
- return $\mathcal{P}_{k+1}$ as result

how to compute M for the Schur complement ?

6.16 The GMRES approach

6.17 The consistent boundary flux (CBF)

The Consistent Boundary Flux technique was devised to alleviate the problem of the accuracy of primary variables derivatives (mainly velocity and temperature) on boundaries, where basis function (nodal) derivatives do not exist. These derivatives are important since they are needed to compute the heat flux (and therefore the NUsselt number) or dynamic topography and geoid.

The idea was first introduced in [63] and later used in geodynamics [103]. It was finally implemented in the CitcomS code [104] and more recently in the ASPECT code (dynamic topography postprocessor). Note that the CBF should be seen as a post-processor step as it does not alter the primary variables values.

The CBF method is implemented and used in **??**.

6.17.1 applied to the Stokes equation

We start from the strong form:

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = \boldsymbol{b}$$

and then write the weak form:

$$\int_\Omega N \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} dV = \int_\Omega N \boldsymbol{b} dV$$

where N is any test function. We then use the two equations:

$$\boldsymbol{\nabla} \cdot (N\boldsymbol{\sigma}) = N\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{\nabla} N \cdot \boldsymbol{\sigma} \qquad \text{(chain rule)}$$

$$\int_\Omega (\boldsymbol{\nabla} \cdot \boldsymbol{f})\, dV = \int_\Gamma \boldsymbol{f} \cdot \boldsymbol{n}\, dS \qquad \text{(divergence theorem)}$$

Integrating the first equation over Ω and using the second, we can write:

$$\int_\Gamma N\boldsymbol{\sigma} \cdot \boldsymbol{n}\, dS - \int_\Omega \nabla N \cdot \boldsymbol{\sigma}\, dV = \int_\Omega N \boldsymbol{b} dV$$

On Γ, the traction vector is given by $\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n}$:

$$\int_\Gamma N \boldsymbol{t} dS = \int_\Omega \nabla N \cdot \boldsymbol{\sigma} dV + \int_\Omega N \boldsymbol{b} dV$$

Considering the traction vector as an unknown living on the nodes on the boundary, we can expand (for Q_1 elements)

$$t_x = \sum_{i=1}^{2} t_{x|i} N_i \qquad t_y = \sum_{i=1}^{2} t_{y|i} N_i$$

on the boundary so that the left hand term yields a mass matrix M'. Finally, using our previous experience of discretising the weak form, we can write:

$$M' \cdot \mathcal{T} = -\mathbb{K}\mathcal{V} - \mathbb{G}\mathcal{P} + f$$

where $\mathcal{T}$ is the vector of assembled tractions which we want to compute and $\mathcal{V}$ and $\mathcal{T}$ are the solutions of the Stokes problem. Note that the assembly only takes place on the elements along the boundary.

Note that the assembled mass matrix is tri-diagonal can be easily solved with a Conjugate Gradient method. With a trapezoidal integration rule (i.e. Gauss-Lobatto) the matrix can even be diagonalised and the resulting matrix is simply diagonal, which results in a very cheap solve [103].

6.17.2 applied to the heat equation

We start from the strong form of the heat transfer equation (without the source terms for simplicity):

$$\rho c_p \left(\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} T \right) = \boldsymbol{\nabla} \cdot k \boldsymbol{\nabla} T$$

The weak form then writes:

$$\int_\Omega N\rho c_p \frac{\partial T}{\partial t} dV + \rho c_p \int_\Omega N\boldsymbol{v}\cdot\boldsymbol{\nabla} T dV = \int_\Omega N\boldsymbol{\nabla}\cdot k\boldsymbol{\nabla} T dV$$

Using once again integration by parts and divergence theorem:

$$\int_\Omega N\rho c_p \frac{\partial T}{\partial t} dV + \rho c_p \int_\Omega N\boldsymbol{v}\cdot\boldsymbol{\nabla} T dV = \int_\Gamma Nk\boldsymbol{\nabla} T\cdot\boldsymbol{n} d\Gamma - \int_\Omega \boldsymbol{\nabla} N\cdot k\boldsymbol{\nabla} T dV$$

On the boundary we are interested in the heat flux $\boldsymbol{q} = -k\boldsymbol{\nabla} T$

$$\int_\Omega N\rho c_p \frac{\partial T}{\partial t} dV + \rho c_p \int_\Omega N\boldsymbol{v}\cdot\boldsymbol{\nabla} T dV = -\int_\Gamma N\boldsymbol{q}\cdot\boldsymbol{n} d\Gamma - \int_\Omega \boldsymbol{\nabla} N\cdot k\boldsymbol{\nabla} T dV$$

or,

$$\int_\Gamma N\boldsymbol{q}\cdot\boldsymbol{n} d\Gamma = -\int_\Omega N\rho c_p \frac{\partial T}{\partial t} dV - \rho c_p \int_\Omega N\boldsymbol{v}\cdot\boldsymbol{\nabla} T dV - \int_\Omega \boldsymbol{\nabla} N\cdot k\boldsymbol{\nabla} T dV$$

Considering the normal heat flux $q_n = \boldsymbol{q}\cdot\boldsymbol{n}$ as an unknown living on the nodes on the boundary,

$$q_n = \sum_{i=1}^{2} q_{n|i} N_i$$

so that the left hand term becomes a mass matrix for the shape functions living on the boundary. We have already covered the right hand side terms when building the FE system to solve the heat transport equation, so that in the end

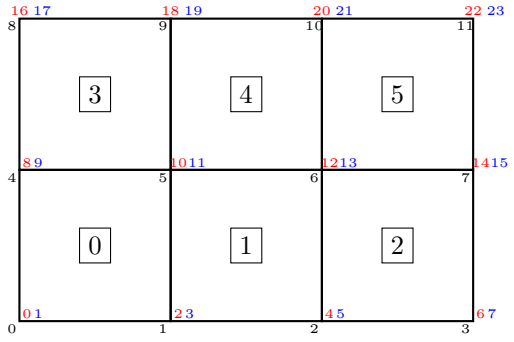
$$M'\cdot\mathcal{Q}_n = -M\cdot\frac{\partial \boldsymbol{T}}{\partial t} - K_a\cdot\boldsymbol{T} - K_d\cdot\boldsymbol{T}$$

where $\mathcal{Q}_n$ is the assembled vector of normal heat flux components. Note that in all terms the assembly only takes place over the elements along the boundary.

Note that the resulting matrix is symmetric.

6.17.3 implementation - Stokes equation

Let us start with a small example, a 3x2 element FE grid:



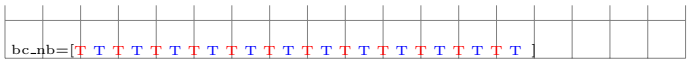
Red color corresponds to the dofs in the x direction, blue color indicates a dof in the y direction.

We have nnp=12, nel=6, NfemV=24. Let us assume that free slip boundary conditions are applied. The `fix_bc` array is then:

bc_fix=[T T T T T T T T T T T T T T T T T T T T T T T T]

Note that since corners belong to two edges, we effectively prescribed no-slip boundary conditions on those.

We wish to compute the tractions on the boundaries, and more precisely for the dofs for which a Dirichlecht velocity boundary condition has been prescribed. The number of (traction) unknowns NfemTr is then the number of `T` in the `bc_fix` array. In our specific case, we wave NfemTr= . This means that we need for each targeted dof to be able to find its identity/number between 0 and NfemTr-1. We therefore create the array `bc_nb` which is filled as follows:



This translates as follows in the code:

```
NfemTr=np.sum(bc_fix)
bc_nb=np.zeros(NfemV,dtype=np.int32)
counter=0
for i in range(0,NfemV):
    if (bc_fix[i]):
       bc_nb[i]=counter
       counter+=1
```

Algorithm 1 dksjfhfjk ghdf

0: kjhg kf
0: gfhfk

The algorithm is then as follows

A Prepare two arrays to store the matrix M_{cbf} and its right hand side rhs_{cbf}

B Loop over all elements

C For each element touching a boundary, compute the residual vector $R_{el} = -f_{el} + \mathbb{K}_{el}\mathcal{V}_{el} + \mathbb{G}_{el}\mathcal{P}_{el}$

D Loop over the four edges of the element using the connectivity array

E For each edge loop over the number of degrees of freedom (2 in 2D)

F For each edge assess whether the dofs on both ends are target dofs.

G If so, compute the mass matrix M_{edge} for this edge

H extract the 2 values off the element residual vector and assemble these in rhs_{cbf}

I Assemble M_{edge} into NfemTrxNfemTr matrix using bc_nb

```
M_cbf = np.zeros((NfemTr,NfemTr),np.float64)          # A
rhs_cbf = np.zeros(NfemTr,np.float64)

for iel in range(0,nel):                              # B

    ... compute elemental residual ...                # C

    #boundary 0-1                                     # D
    for i in range(0,ndofV):                          # E
        idof0=2*icon[0,iel]+i
        idof1=2*icon[1,iel]+i
        if (bc_fix[idof0] and bc_fix[idof1]):         # F
           idofTr0=bc_nb[idof0]
           idofTr1=bc_nb[idof1]
           rhs_cbf[idofTr0]+=res_el[0+i]              # H
           rhs_cbf[idofTr1]+=res_el[2+i]
           M_cbf[idofTr0,idofTr0]+=M_edge[0,0]        #
           M_cbf[idofTr0,idofTr1]+=M_edge[0,1]        # I
           M_cbf[idofTr1,idofTr0]+=M_edge[1,0]        #
           M_cbf[idofTr1,idofTr1]+=M_edge[1,1]        #

    #boundary 1-2                                     #[D]

    ...

    #boundary 2-3                                     #[D]

    ...
```

```
#boundary 3-0                                    #[D]
...
```

6.18 The value of the timestep

The chosen time step dt used for time integration is chosen to comply with the Courant-Friedrichs-Lewy condition [5].

$$\delta t = C \min\left(\frac{h}{\max|\boldsymbol{v}|}, \frac{h^2}{\kappa}\right)$$

where h is a measure of the element size, $\kappa = k/\rho c_p$ is the thermal diffusivity and C is the so-called CFL number chosen in $[0, 1[$.

In essence the CFL condition arises when solving hyperbolic PDEs . It limits the time step in many explicit time-marching computer simulations so that the simulation does not produce incorrect results.

This condition is not needed when solving the Stokes equation but it is mandatory when solving the heat transport equation or any kind of advection-diffusion equation. Note that any increase of grid resolution (i.e. h becomes smaller) yields an automatic decrease of the time step value.

6.19 mappings

The name isoparametric derives from the fact that the same ('iso') functions are used as basis functions and for the mapping to the reference element.

6.20 Exporting data to vtk format

This format seems to be the universally accepted format for 2D and 3D visualisation in Computational Geodynamics. Such files can be opened with free softwares such as Paraview [4], MayaVi [5] or Visit [6].

Unfortunately it is my experience that no simple tutorial exists about how to build such files. There is an official document which describes the vtk format[7] but it delivers the information in a convoluted way. I therefore describe hereafter how **fieldstone** builds the vtk files.

I hereunder show vtk file corresponding to the 3x2 grid presented earlier 6.10. In this particular example there are:

- 12 nodes and 6 elements
- 1 elemental field: the pressure `p`)
- 2 nodal fields: 1 scalar (the smoothed pressure `q`), 1 vector (the velocity field `u,v,0`)

Note that vtk files are inherently 3D so that even in the case of a 2D simulation the z-coordinate of the points and for instance their z-velocity component must be provided. The file, usually called *solution.vtu* starts with a header:

```
<VTKFile type='UnstructuredGrid' version='0.1' byte_order='BigEndian'>
<UnstructuredGrid>
<Piece NumberOfPoints='12' NumberOfCells='6'>
```

We then proceed to write the node coordinates as follows:

```
<Points>
<DataArray type='Float32' NumberOfComponents='3' Format='ascii'>
0.000000e+00 0.000000e+00 0.000000e+00
3.333333e-01 0.000000e+00 0.000000e+00
6.666667e-01 0.000000e+00 0.000000e+00
1.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 5.000000e-01 0.000000e+00
3.333333e-01 5.000000e-01 0.000000e+00
6.666667e-01 5.000000e-01 0.000000e+00
1.000000e+00 5.000000e-01 0.000000e+00
0.000000e+00 1.000000e+00 0.000000e+00
3.333333e-01 1.000000e+00 0.000000e+00
6.666667e-01 1.000000e+00 0.000000e+00
1.000000e+00 1.000000e+00 0.000000e+00
</DataArray>
</Points>
```

These are followed by the elemental field(s):

```
<CellData Scalars='scalars'>
<DataArray type='Float32' Name='p' Format='ascii'>
-1.333333e+00
-3.104414e-10
1.333333e+00
-1.333333e+00
8.278417e-17
1.333333e+00
</DataArray>
</CellData>
```

Nodal quantities are written next:

```
<PointData Scalars='scalars'>
<DataArray type='Float32' NumberOfComponents='3' Name='velocity' Format='ascii'>
0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
```

[4] https://www.paraview.org/
[5] https://docs.enthought.com/mayavi/mayavi/
[6] https://wci.llnl.gov/simulation/computer-codes/visit/
[7] https://www.vtk.org/wp-content/uploads/2015/04/file-formats.pdf

```
0.000000e+00 0.000000e+00 0.000000e+00
8.888885e-08 -8.278405e-24 0.000000e+00
8.888885e-08 1.655682e-23 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
1.000000e+00 0.000000e+00 0.000000e+00
1.000000e+00 0.000000e+00 0.000000e+00
1.000000e+00 0.000000e+00 0.000000e+00
1.000000e+00 0.000000e+00 0.000000e+00
</DataArray>
<DataArray type='Float32' NumberOfComponents='1' Name='q' Format='ascii'>
-1.333333e+00
-6.666664e-01
6.666664e-01
1.333333e+00
-1.333333e+00
-6.666664e-01
6.666664e-01
1.333333e+00
-1.333333e+00
-6.666664e-01
6.666664e-01
1.333333e+00
</DataArray>
</PointData>
```

To these informations we must append 3 more datasets. The first one is the connectivity, the second one is the offsets and the third one is the type. The first one is trivial since said connectivity is needed for the Finite Elements. The second must be understood as follows: when reading the connectivity information in a linear manneer the offset values indicate the beginning of each element (omitting the zero value). The third simply is the type of element as given in the vtk format document (9 corresponds to a generic quadrilateral with an internal numbering consistent with ours).

```
<Cells>
<DataArray type='Int32' Name='connectivity' Format='ascii'>
0 1 5 4
1 2 6 5
2 3 7 6
4 5 9 8
5 6 10 9
6 7 11 10
</DataArray>
<DataArray type='Int32' Name='offsets' Format='ascii'>
4
8
12
16
20
24
</DataArray>
<DataArray type='Int32' Name='types' Format='ascii'>
9
9
9
9
9
9
</DataArray>
</Cells>
```

The file is then closed with

```
</Piece>
</UnstructuredGrid>
</VTKFile>
```

The *solution.vtu* file can then be opened with ParaView, MayaVi or Visit and the reader is advised to find tutorials online on how to install and use these softwares.

6.21 Runge-Kutta methods

These methods were developed around 1900 by the German mathematicians Carl Runge and Martin Kutta. The RK methods are methods for the numerical integration of ODEs. These methods are well documented in any numerical analysis textbook and the reader is referred to [37, 47].

Any Runge-Kutta method is uniquely identified by its Butcher tableau.

The following method is called the Runge-Kutta-Fehlberg method and is commonly abbreviated RKF45. Its Butcher tableau is as follows:

0						
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	-7200/2197	7296/2197			
1	439/216	-8	3680/513	-845/4104		
1/2	-8/27	2	-3544/2565	1859/4104	-11/40	
	16/135	0	6656/12825	28561/56430	-9/50	2/55
	25/216	0	1408/2565	2197/4104	-1/5	0

The first row of coefficients at the bottom of the table gives the fifth-order accurate method, and the second row gives the fourth-order accurate method.

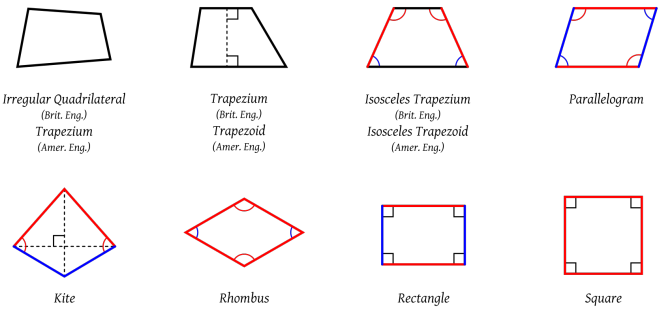
6.22 Am I in or not?

It is quite common that at some point one must answer the question: "Given a mesh and its connectivity on the one hand, and the coordinates of a point on the other, how do I accurately and quickly determine in which element the point resides?"

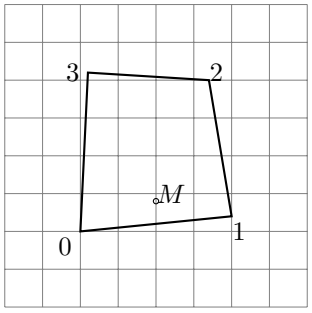
One typical occurence of such a problem is linked to the use of the Particle-In-Cell technique: particles are advected and move through the mesh, and need to be localised at every time step. This question could arise in the context of a benchmark where certain quantities need to be measured at specific locations inside the domain.

6.22.1 Two-dimensional space

We shall first focus on quadrilaterals. There are many kinds of quadrilaterals as shown hereunder:



I wish to arrive at a single algorithm which is applicable to all quadrilaterals and therefore choose an irregular quadrilateral. For simplicity, let us consider a Q_1 element, with a single node at each corner.



Several rather simple options exist:

- we could subdivide the quadrilateral into two triangles and check whether point M is inside any of them (as it turns out, this problem is rather straightforward for triangles. Simply google it.)
- We could check that point M is always on the left side of segments $0 \to 1$, $1 \to 2$, $2 \to 3$, $3 \to 0$.
- ...

Any of these approaches will work although some might be faster than others. In three-dimensions all will however become cumbersome to implement and might not even work at all. Fortunately, there is an elegant way to answer the question, as detailed in the following subsection.

6.22.2 Three-dimensional space

If point M is inside the quadrilateral, there exist a set of reduced coordinates $r, s, t \in [-1:1]^3$ such that

$$\sum_{i=1}^{4} N_i(r_M, s, t) x_i = x_M \qquad \sum_{i=1}^{4} N_i(r_M, s, t) y_i = y_M \qquad \sum_{i=1}^{4} N_i(r_M, s, t) z_i = z_M$$

This can be cast as a system of three equations and three unknowns. Unfortunately, each shape function N_i contains a term rst (as well as rs, rt, and st) so that it is not a linear system and standard techniques are not applicable. We must then use an iterative technique: the algorithm starts with a guess for values r, s, t and improves on their value iteration after iteration.

The classical way of solving nonlinear systems of equations is Newton's method. We can rewrite the equations above as $\boldsymbol{F}(r, s, t) = 0$:

$$\begin{aligned} \sum_{i=1}^{8} N_i(r,s,t)x_i - x_M &= 0 \\ \sum_{i=1}^{8} N_i(r,s,t)y_i - y_M &= 0 \\ \sum_{i=1}^{8} N_i(r,s,t)z_i - z_M &= 0 \end{aligned} \tag{65}$$

or,

$$\begin{aligned} F_r(r,s,t) &= 0 \\ F_s(r,s,t) &= 0 \\ F_t(r,s,t) &= 0 \end{aligned}$$

so that we now have to find the zeroes of continuously differentiable functions $\boldsymbol{F} : \mathbb{R} \to \mathbb{R}$. The recursion is simply:

$$\begin{pmatrix} r_{k+1} \\ s_{k+1} \\ t_{k+1} \end{pmatrix} = \begin{pmatrix} r_k \\ s_k \\ t_k \end{pmatrix} - J_F(r_k, s_k, t_k)^{-1} \begin{pmatrix} F_r(r_k, s_k, t_k) \\ F_s(r_k, s_k, t_k) \\ F_t(r_k, s_k, t_k) \end{pmatrix}$$

where J the Jacobian matrix:

$$\begin{aligned} J_F(r_k, s_k, t_k) &= \begin{pmatrix} \frac{\partial F_r}{\partial r}(r_k, s_k, t_k) & \frac{\partial F_r}{\partial s}(r_k, s_k, t_k) & \frac{\partial F_r}{\partial t}(r_k, s_k, t_k) \\ \frac{\partial F_s}{\partial r}(r_k, s_k, t_k) & \frac{\partial F_s}{\partial s}(r_k, s_k, t_k) & \frac{\partial F_s}{\partial t}(r_k, s_k, t_k) \\ \frac{\partial F_t}{\partial r}(r_k, s_k, t_k) & \frac{\partial F_t}{\partial s}(r_k, s_k, t_k) & \frac{\partial F_t}{\partial t}(r_k, s_k, t_k) \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^{8} \frac{\partial N_i}{\partial r}(r_k, s_k, t_k)x_i & \sum_{i=1}^{8} \frac{\partial N_i}{\partial s}(r_k, s_k, t_k)x_i & \sum_{i=1}^{8} \frac{\partial N_i}{\partial t}(r_k, s_k, t_k)x_i \\ \sum_{i=1}^{8} \frac{\partial N_i}{\partial r}(r_k, s_k, t_k)y_i & \sum_{i=1}^{8} \frac{\partial N_i}{\partial s}(r_k, s_k, t_k)y_i & \sum_{i=1}^{8} \frac{\partial N_i}{\partial t}(r_k, s_k, t_k)y_i \\ \sum_{i=1}^{8} \frac{\partial N_i}{\partial r}(r_k, s_k, t_k)z_i & \sum_{i=1}^{8} \frac{\partial N_i}{\partial s}(r_k, s_k, t_k)z_i & \sum_{i=1}^{8} \frac{\partial N_i}{\partial t}(r_k, s_k, t_k)z_i \end{pmatrix} \end{aligned}$$

In practice, we solve the following system:

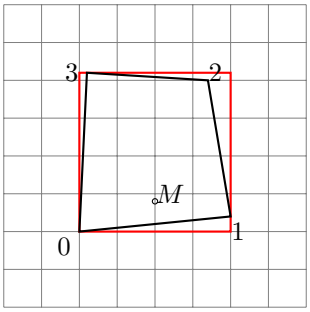
$$J_F(r_k, s_k, t_k) \left[\begin{pmatrix} r_{k+1} \\ s_{k+1} \\ t_{k+1} \end{pmatrix} - \begin{pmatrix} r_k \\ s_k \\ t_k \end{pmatrix} \right] = - \begin{pmatrix} F_r(r_k, s_k, t_k) \\ F_s(r_k, s_k, t_k) \\ F_t(r_k, s_k, t_k) \end{pmatrix}$$

Finally, the algorithm goes as follows:

- set guess values for r, s, t (typically 0)
- loop over k=0,...
- Compute rhs= $-\boldsymbol{F}(r_k, s_k, t_k)$
- Compute matrix $J_F(r_k, s_k, t_k)$

- solve system for (dr_k, ds_k, dt_k)
- update $r_{k+1} = r_k + dr_k$, $s_{k+1} = s_k + ds_k$, $t_{k+1} = t_k + dt_k$
- stop iterations when (dr_k, ds_k, dt_k) is small
- if $r_k, s_k, t_k \in [-1, 1]^3$ then M is inside.

This method converges quickly but involves iterations, and multiple solves of 3×3 systems which, when carried out for each marker and at each time step can prove to be expensive. A simple modification can be added to the above algorithm: iterations should be carried out *only* when the point M is inside of a cuboid of size $[\min_i x_i : \max_i x_i] \times [\min_i y_i : \max_i y_i] \times [\min_i z_i : \max_i z_i]$ where the sums run over the vertices of the element. In 2D this translates as follows: only carry out Newton iterations when M is inside the red rectangle!



Note that the algorithm above extends to high degree elements such as Q_2 and higher, even with curved sides.

To Do:

- write about impose bc on el matrix
- full compressible
- total energy calculations
- constraints
- compositions, marker chain
- free-slip bc on annulus and sphere . See for example p540 Gresho and Sani book.
- non-linear rheologies (two layer brick spmw16, tosn15)
- Picard vs Newton
- periodic boundary conditions
- free surface
- newton method to localise markers
- zaleski disk advection
- cvi !!!
- TOSI !!!!
- matrix singular annulus conv
- pure elastic
- including phase changes (w. R. Myhill)
- discontinuous galerkin
- nonlinear poiseuille
- formatting of code style
- navier-stokes ? (LUKAS)
- compute strainrate in middle of element or at quad point for punch?
- GEO1442 code
- GEO1442 indenter setup in plane ?
- in/out flow on sides for lith modelling
- Fehlberg RK advection
- redo puth17 2 layer experiment

https://peterkovesi.com/projects/colourmaps/ velocity: CET-D1A pressure: CET-D1 divv: CET-L1 density: CET-D3 strainrate: CET-R2
Problems to solve:
colorscale
better yet simple matrix storage ?
write Scott about matching compressible2 setup with his paper
deal with large matrices.

7 List of tutorials

tutorial number	element	outer solver	formulation	physical problem	3D	temperature	time stepping	nonlinear	compressible	analytical benchmark	numerical benchmark
1	$Q_1 \times P_0$		penalty	analytical benchmark							
2	$Q_1 \times P_0$		penalty	Stokes sphere							
3	$Q_1 \times P_0$		penalty	Blankenbach et al., 1989		†	†				
4	$Q_1 \times P_0$		penalty	Lid driven cavity							
5	$Q_1 \times P_0$		penalty	SolCx benchmark							
6	$Q_1 \times P_0$		penalty	SolKz benchmark							
7	$Q_1 \times P_0$		penalty	SolVi benchmark							
8	$Q_1 \times P_0$		penalty	Indentor				†			
9	$Q_1 \times P_0$		penalty	annulus benchmark							
10	$Q_1 \times P_0$		penalty	Stokes sphere	†						
11	$Q_1 \times P_0$	full matrix	mixed	Stokes sphere	†						
12	$Q_1 \times P_0$		penalty	analytical benchmark + consistent press recovery							
13	$Q_1 \times P_0$		penalty	Stokes sphere + markers averaging							
14	$Q_1 \times P_0$	full matrix	mixed	analytical benchmark							
15	$Q_1 \times P_0$	Schur comp. CG	mixed	analytical benchmark							
16	$Q_1 \times P_0$	Schur comp. PCG	mixed	Stokes sphere							
17	$Q_2 \times Q_1$	full matrix	mixed	Burstedde benchmark	†						
18	$Q_2 \times Q_1$	full matrix	mixed	analytical benchmark							
19	$Q_3 \times Q_2$	full matrix	mixed	analytical benchmark							
20	$Q_1 \times P_0$		penalty	Busse et al., 1993	†	†	†				
21	$Q_1 \times P_0$ R-T		penalty	analytical benchmark							
22	$Q_1 \times Q_1$-stab	full matrix	mixed	analytical benchmark							
23	$Q_1 \times P_0$		mixed	analytical benchmark					†		
24	$Q_1 \times P_0$		mixed	convection box		†	†		†		
25	$Q_1 \times P_0$	full matrix	mixed	Rayleigh-Taylor instability							
26	$Q_1 \times P_0$	full matrix	mixed	Slab detachment				†			
27	$Q_1 \times P_0$	full matrix	mixed	CBF benchmarks				†		†	
28	$Q_1 \times P_0$	full matrix	mixed	Tosi et al, 2015		†	†	†			†
29	$Q_1 \times P_0$	full matrix	mixed	Open Boundary conditions						†	
30	Q_1, Q_2	X	X	Cons. Vel. Interp (cvi)			†			†	

8 `fieldstone_01`: simple analytical solution

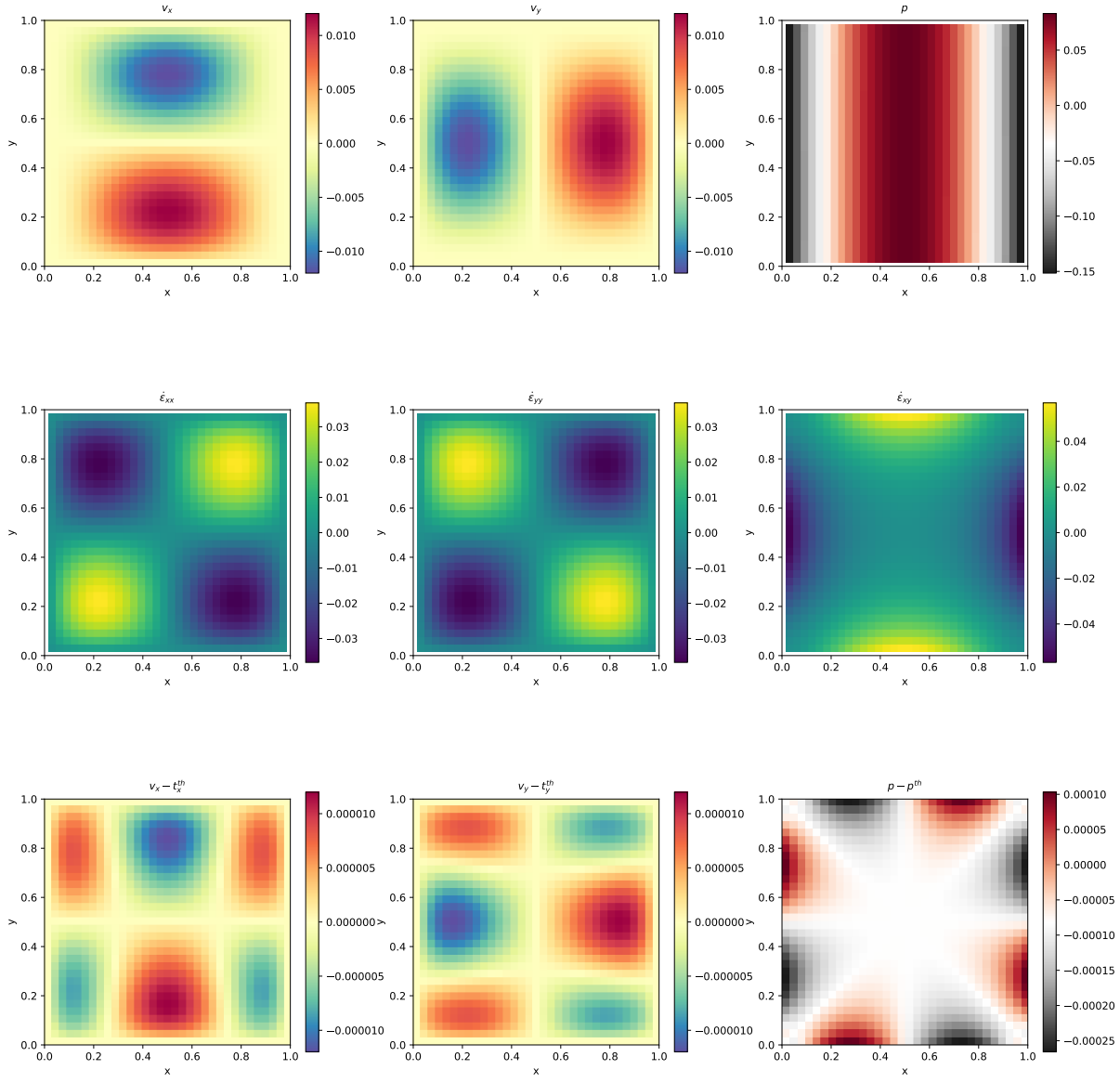
From [28]. In order to illustrate the behavior of selected mixed finite elements in the solution of stationary Stokes flow, we consider a two-dimensional problem in the square domain $\Omega = [0,1]\times[0,1]$, which possesses a closed-form analytical solution. The problem consists of determining the velocity field $\boldsymbol{v} = (u, v)$ and the pressure p such that

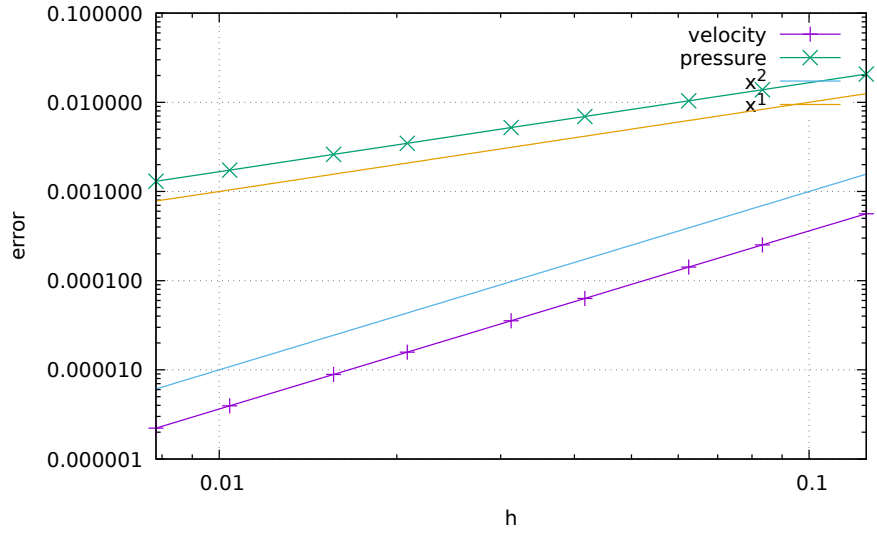
$$-\nu\Delta\boldsymbol{v} + \boldsymbol{\nabla}p = \boldsymbol{b} \qquad \text{in } \Omega$$
$$\boldsymbol{\nabla}\cdot\boldsymbol{v} = 0 \qquad \text{in } \Omega$$
$$\boldsymbol{v} = \boldsymbol{0} \qquad \text{on } \Gamma$$

where the fluid viscosity is taken as $\nu = 1$.

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (no-slip)
- direct solver
- isothermal
- isoviscous
- analytical solution





Quadratic convergence for velocity error, linear convergence for pressure error, as expected.

ToDo:
pressure normalisation?
different cmat, a la schmalholz
To go further:

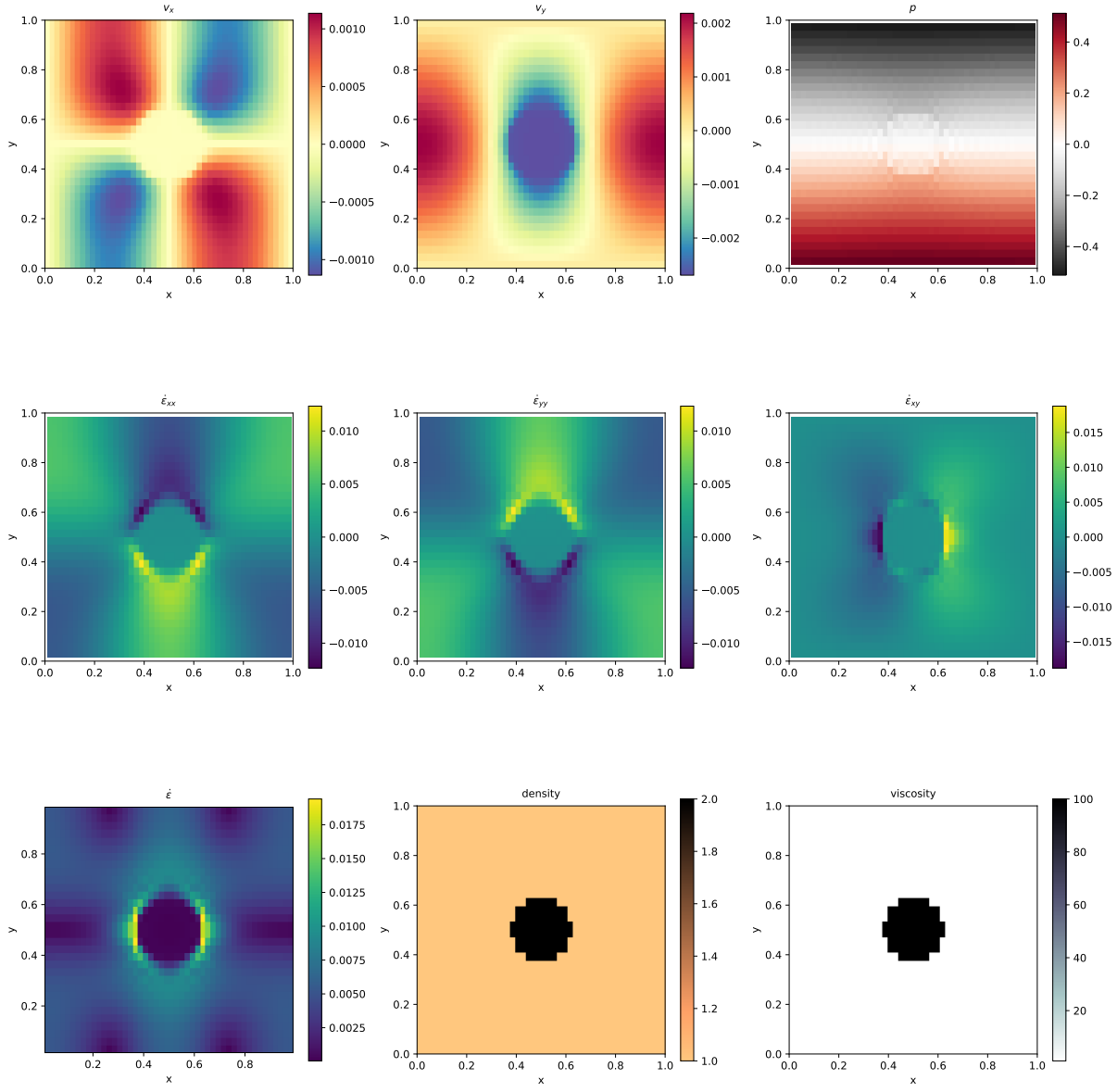
1. make your own analytical solution

9 `fieldstone_02`: Stokes sphere

Viscosity and density directly computed at the quadrature points.

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (free-slip)
- isothermal
- non-isoviscous
- buoyancy-driven flow
- Stokes sphere



10 `fieldstone_03`: Convection in a 2D box

This benchmark deals with the 2-D thermal convection of a fluid of infinite Prandtl number in a rectangular closed cell. In what follows, I carry out the case 1a, 1b, and 1c experiments as shown in [9]: steady convection with constant viscosity in a square box.

The temperature is fixed to zero on top and to ΔT at the bottom, with reflecting symmetry at the sidewalls (i.e. $\partial_x T = 0$) and there are no internal heat sources. Free-slip conditions are implemented on all boundaries.

The Rayleigh number is given by

$$Ra = \frac{\alpha g_y \Delta T h^3}{\kappa \nu} = \frac{\alpha g_y \Delta T h^3 \rho^2 c_p}{k \mu} \tag{66}$$

In what follows, I use the following parameter values: $L_x = L_y = 1$,$\rho_0 = c_P = k = \mu = 1$, $T_0 = 0$, $\alpha = 10^{-2}$, $g = 10^2 Ra$ and I run the model with $Ra = 10^4, 10^5$ and 10^6.

The initial temperature field is given by

$$T(x,y) = (1-y) - 0.01 \cos(\pi x) \sin(\pi y) \tag{67}$$

The perturbation in the initial temperature fields leads to a perturbation of the density field and sets the fluid in motion.

Depending on the initial Rayleigh number, the system ultimately reaches a steady state after some time.

The Nusselt number (i.e. the mean surface temperature gradient over mean bottom temperature) is computed as follows [9]:

$$Nu = L_y \frac{\int \frac{\partial T}{\partial y}(y = L_y) dx}{\int T(y=0) dx} \tag{68}$$

Note that in our case the denominator is equal to 1 since $L_x = 1$ and the temperature at the bottom is prescribed to be 1.

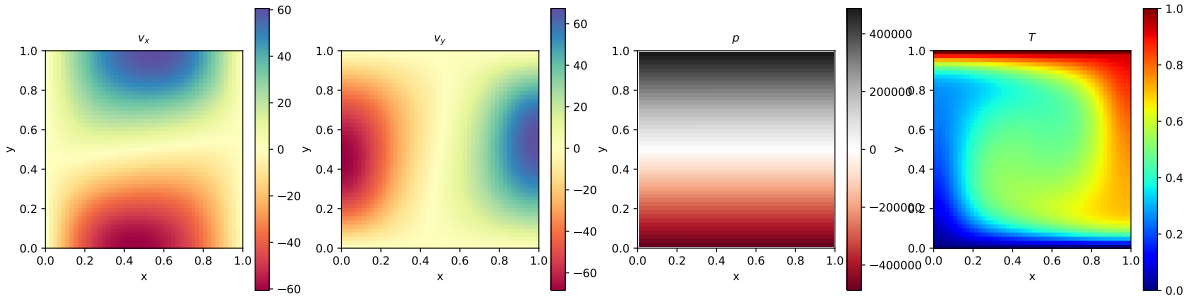
Finally, the steady state root mean square velocity and Nusselt number measurements are indicated in Table **??** alongside those of [9] and [86]. (Note that this benchmark was also carried out and published in other publications [94, 1, 37, 23, 57] but since they did not provide a complete set of measurement values, they are not included in the table.)

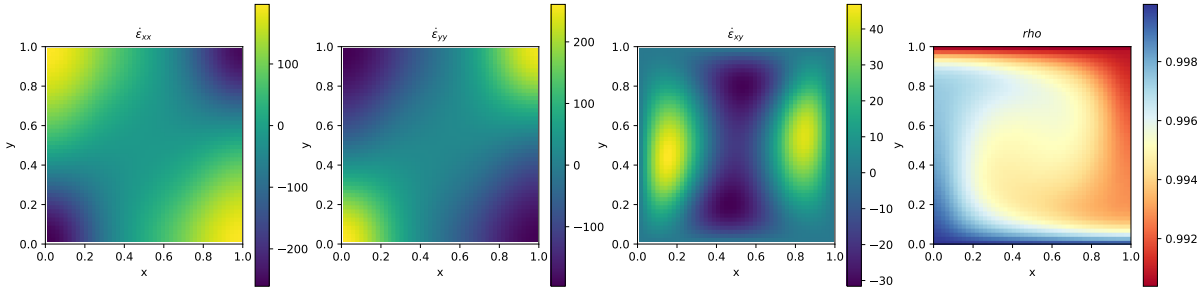
		Blankenbach et al	Tackley [86]
$Ra = 10^4$	V_{rms}	42.864947 ± 0.000020	42.775
	Nu	4.884409 ± 0.000010	4.878
$Ra = 10^5$	V_{rms}	193.21454 ± 0.00010	193.11
	Nu	10.534095 ± 0.000010	10.531
$Ra = 10^6$	V_{rms}	833.98977 ± 0.00020	833.55
	Nu	21.972465 ± 0.000020	21.998

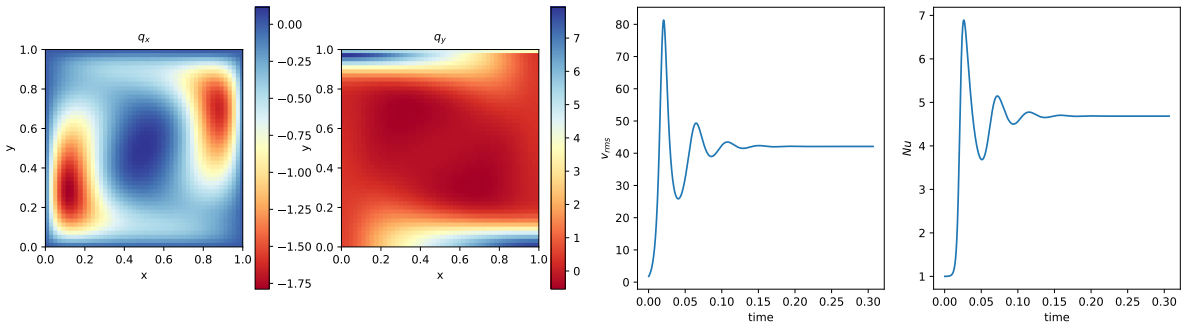
Steady state Nusselt number Nu and V_{rms} measurements as reported in the literature.

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (free-slip)
- Boussinesq approximation
- direct solver
- non-isothermal
- buoyancy-driven flow
- isoviscous
- CFL-condition







ToDo:
implement steady state criterion
reach steady state
do Ra=1e4, 1e5, 1e6
plot against blankenbach paper and aspect
look at critical Ra number

11 `fieldstone_04`: The lid driven cavity

The lid driven cavity is a famous Computational Fluid Dynamics test case and has been studied in countless publications with a wealth of numerical techniques (see [33] for a succinct review) and also in the laboratory [53].

It models a plane flow of an isothermal isoviscous fluid in a rectangular (usually square) lid-driven cavity. The boundary conditions are indicated in the Fig. **??**a. The gravity is set to zero.

11.1 the lid driven cavity problem (`ldc=0`)

In the standard case, the upper side of the cavity moves in its own plane at unit speed, while the other sides are fixed. This thereby introduces a discontinuity in the boundary conditions at the two upper corners of the cavity and yields an uncertainty as to which boundary (side or top) the corner points belong to. In this version of the code the top corner nodes are considered to be part of the lid. If these are excluded the recovered pressure showcases and extremely large checkboard pattern.

This benchmark is usually dicussed in the context of low to very high Reynolds number with the full Navier-Stokes equations being solved (with the noticeable exception of [77, 78, 17, 32] which focus on the Stokes equation). In the case of the incompressible Stokes flow, the absence of inertia renders this problem instantaneous so that only one time step is needed.

11.2 the lid driven cavity problem - regularisation I (`ldc=1`)

We avoid the top corner nodes issue altogether by prescribing the horizontal velocity of the lid as follows:

$$u(x) = x^2(1-x)^2. \tag{69}$$

In this case the velocity and its first derivative is continuous at the corners. This is the so-called regularised lid-driven cavity problem [70].

11.3 the lid driven cavity problem - regularisation II (`ldc=2`)

Another regularisation was presented in [25]. Here, a regularized lid driven cavity is studied which is consistent in the sense that $\boldsymbol{\nabla}\cdot\boldsymbol{v}=0$ holds also at the corners of the domain. There are no-slip conditions at the boundaries $x=0$, $x=1$, and $y=0$.

The velocity at $y=1$ is given by

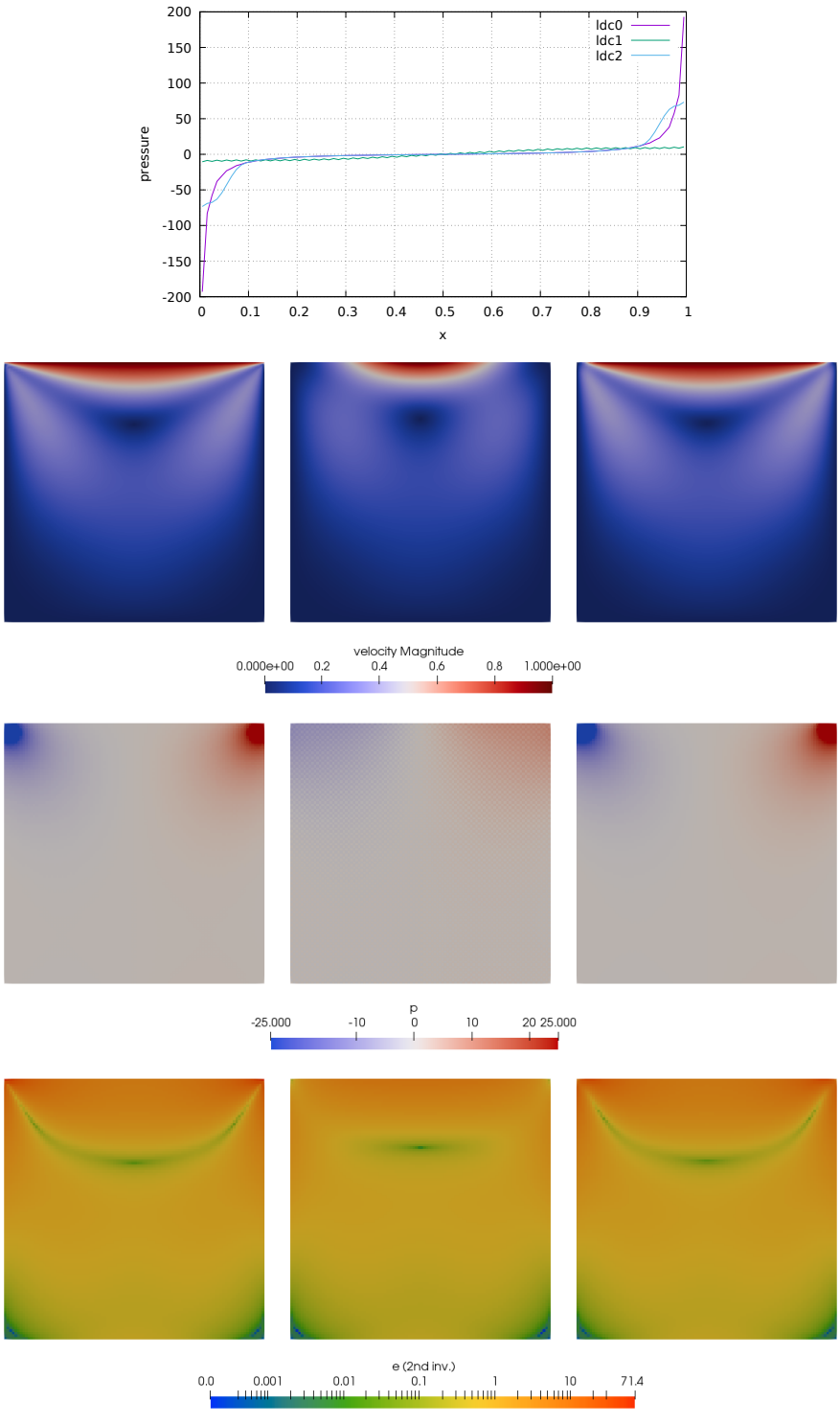
$$\begin{aligned}
u(x) &= 1-\frac{1}{4}\left(1-\cos(\frac{x_1-x}{x_1}\pi)\right)^2 \qquad x\in[0,x_1] \\
u(x) &= 1 \qquad x\in[x_1,1-x_1] \\
u(x) &= 1-\frac{1}{4}\left(1-\cos(\frac{x-(1-x_1)}{x_1}\pi)\right)^2 \qquad x\in[1-x_1,1]
\end{aligned} \tag{70}$$

Results are obtained with $x_1=0.1$.

features

- $Q_1\times P_0$ element
- incompressible flow
- penalty formulation
- isothermal
- isoviscous

A 100x100 element grid is used. No-slip boundary conditions are prescribed on sides and bottom. A zero vertical velocity is prescribed at the top and the exact form of the prescribed horizontal velocity is controlled by the `ldc` parameter.



12 `fieldstone_05`: SolCx benchmark

The SolCx benchmark is intended to test the accuracy of the solution to a problem that has a large jump in the viscosity along a line through the domain. Such situations are common in geophysics: for example, the viscosity in a cold, subducting slab is much larger than in the surrounding, relatively hot mantle material.

The SolCx benchmark computes the Stokes flow field of a fluid driven by spatial density variations, subject to a spatially variable viscosity. Specifically, the domain is $\Omega = [0,1]^2$, gravity is $\boldsymbol{g} = (0,-1)^T$ and the density is given by

$$\rho(x,y) = \sin(\pi y)\cos(\pi x) \tag{71}$$

Boundary conditions are free slip on all of the sides of the domain and the temperature plays no role in this benchmark. The viscosity is prescribed as follows:

$$\mu(x,y) = \begin{cases} 1 & for \quad x < 0.5 \\ 10^6 & for \quad x > 0.5 \end{cases} \tag{72}$$

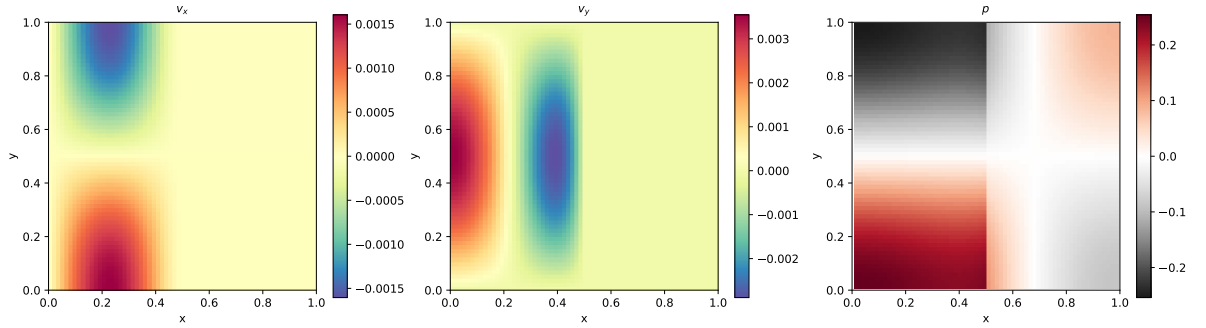
Note the strongly discontinuous viscosity field yields a stagnant flow in the right half of the domain and thereby yields a pressure discontinuity along the interface.

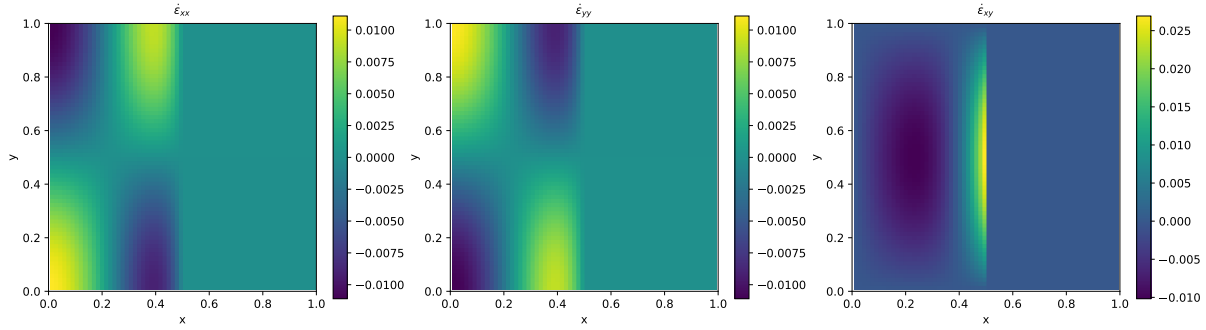
The SolCx benchmark was previously used in [31] (references to earlier uses of the benchmark are available there) and its analytic solution is given in [102]. It has been carried out in [54] and [38]. Note that the source code which evaluates the velocity and pressure fields for both SolCx and SolKz is distributed as part of the open source package Underworld ([65], http://underworldproject.org).

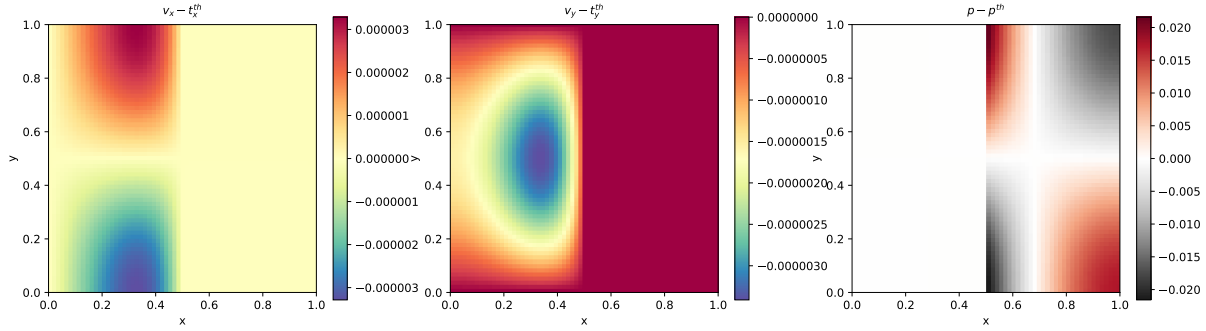
In this particular example, the viscosity is computed analytically at the quadrature points (i.e. tracers are not used to attribute a viscosity to the element). If the number of elements is even in any direction, all elements (and their associated quadrature points) have a constant viscosity(1 or 10^6). If it is odd, then the elements situated at the viscosity jump have half their integration points with $\mu = 1$ and half with $\mu = 10^6$ (which is a pathological case since the used quadrature rule inside elements cannot represent accurately such a jump).

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (free-slip)
- direct solver
- isothermal
- non-isoviscous
- analytical solution







What we learn from this

13 `fieldstone_06`: SolKz benchmark

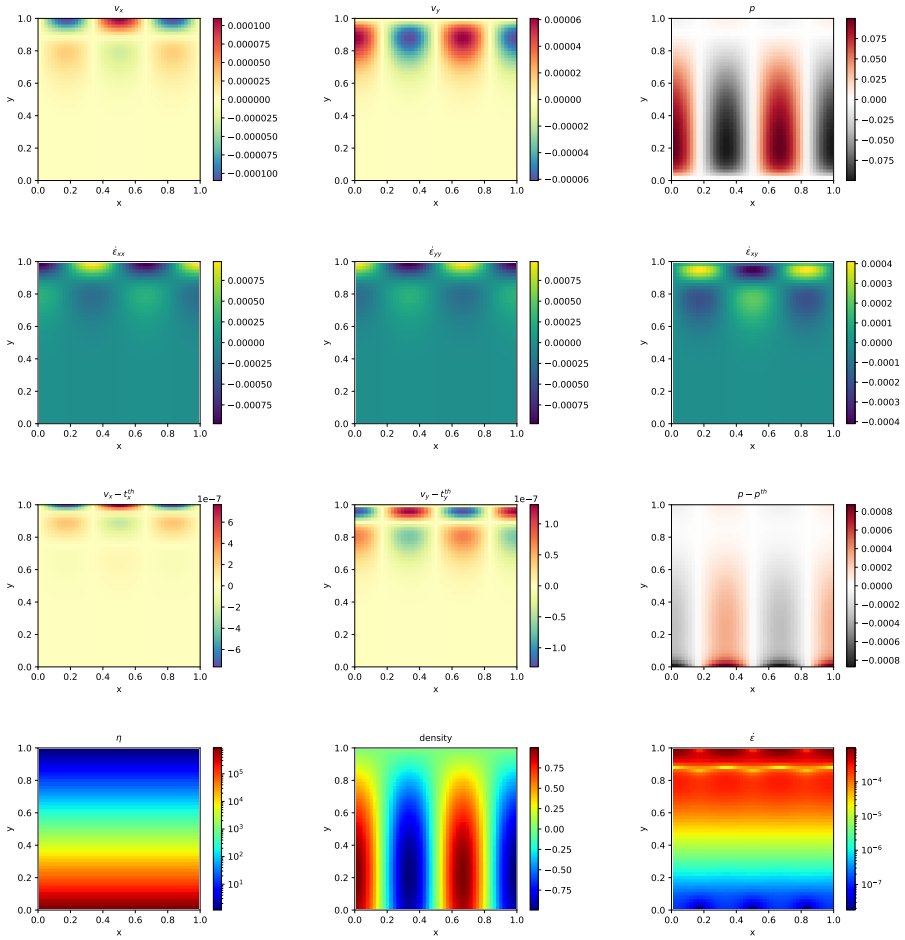
The SolKz benchmark [74] is similar to the SolCx benchmark. but the viscosity is now a function of the space coordinates:

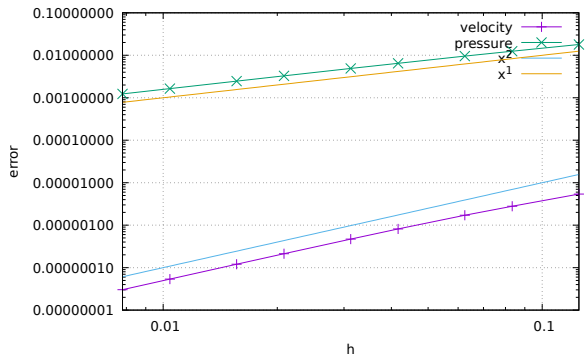
$$\mu(y) = \exp(By) \quad \text{with} \quad B = 13.8155 \tag{73}$$

It is however not a discontinuous function but grows exponentially with the vertical coordinate so that its overall variation is again 10^6. The forcing is again chosen by imposing a spatially variable density variation as follows:

$$\rho(x, y) = \sin(2y)\cos(3\pi x) \tag{74}$$

Free slip boundary conditions are imposed on all sides of the domain. This benchmark is presented in [102] as well and is studied in [31] and [38].





14 `fieldstone_07`: SolVi benchmark

Following SolCx and SolKz, the SolVi inclusion benchmark solves a problem with a discontinuous viscosity field, but in this case the viscosity field is chosen in such a way that the discontinuity is along a circle. Given the regular nature of the grid used by a majority of codes and the present one, this ensures that the discontinuity in the viscosity never aligns to cell boundaries. This in turns leads to almost discontinuous pressures along the interface which are difficult to represent accurately. [81] derived a simple analytic solution for the pressure and velocity fields for a circular inclusion under simple shear and it was used in [26], [85], [31], [54] and [38].

Because of the symmetry of the problem, we only have to solve over the top right quarter of the domain (see Fig. **??**a).

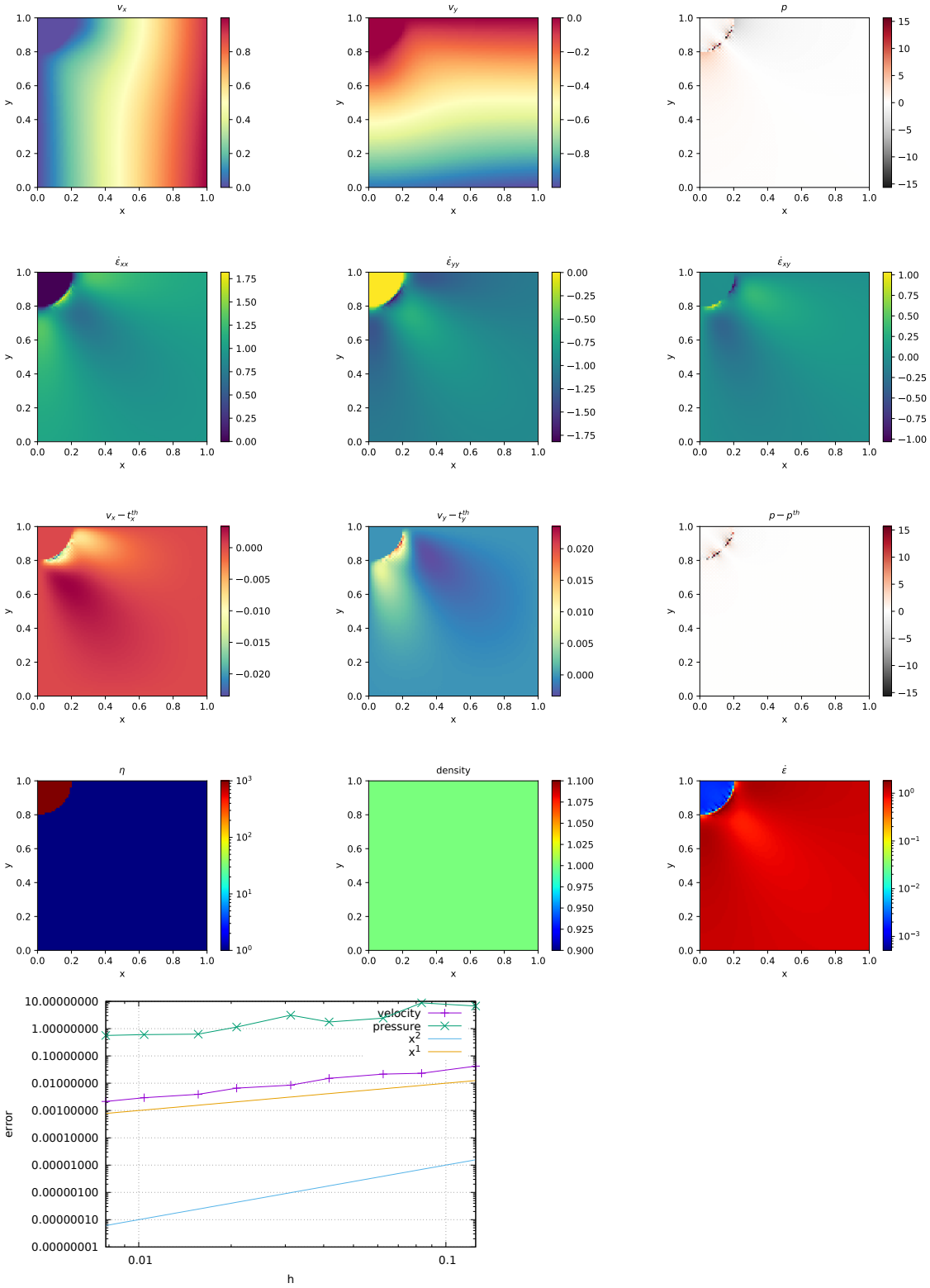
The analytical solution requires a strain rate boundary condition (e.g., pure shear) to be applied far away from the inclusion. In order to avoid using very large domains and/or dealing with this type of boundary condition altogether, the analytical solution is evaluated and imposed on the boundaries of the domain. By doing so, the truncation error introduced while discretizing the strain rate boundary condition is removed.

A characteristic of the analytic solution is that the pressure is zero inside the inclusion, while outside it follows the relation

$$p_m = 4\dot{\epsilon}\frac{\mu_m(\mu_i-\mu_m)}{\mu_i+\mu_m}\frac{r_i^2}{r^2}\cos(2\theta) \tag{75}$$

where $\mu_i = 10^3$ is the viscosity of the inclusion and $\mu_m = 1$ is the viscosity of the background media, $\theta = \tan^{-1}(y/x)$, and $\dot{\epsilon} = 1$ is the applied strain rate.

[26] thoroughly investigated this problem with various numerical methods (FEM, FDM), with and without tracers, and conclusively showed how various averagings lead to different results. [31] obtained a first order convergence for both pressure and velocity, while [54] and [38] showed that the use of adaptive mesh refinement in respectively the FEM and FDM yields convergence rates which depend on refinement strategies.


v_x
v_y
p
$\dot{\varepsilon}_{xx}$
$\dot{\varepsilon}_{yy}$
$\dot{\varepsilon}_{xy}$
$v_x - t_x^{th}$
$v_y - t_y^{th}$
$p - p^{th}$
η
density
$\dot{\varepsilon}$
x
y
velocity
pressure
x^2
x^1
error
h
0.01
0.1

15 `fieldstone_08`: the indentor benchmark

The punch benchmark is one of the few boundary value problems involving plastic solids for which there exists an exact solution. Such solutions are usually either for highly simplified geometries (spherical or axial symmetry, for instance) or simplified material models (such as rigid plastic solids) [50].

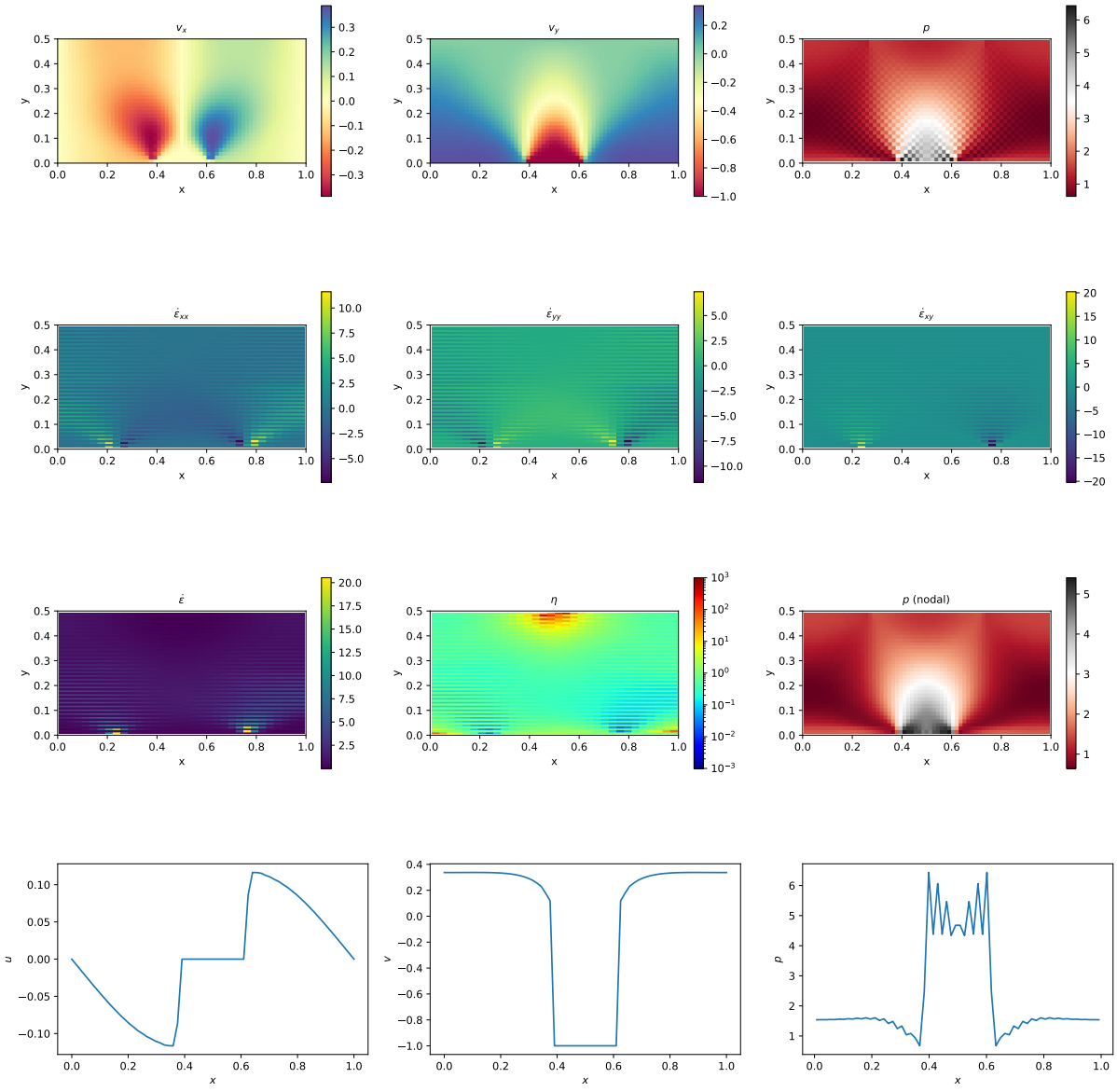
In this experiment, a rigid punch indents a rigid plastic half space; the slip line field theory gives exact solutions as shown in Fig. **??**a. The plane strain formulation of the equations and the detailed solution to the problem were derived in the Appendix of [92] and are also presented in [36].

The two dimensional punch problem has been extensively studied numerically for the past 40 years [108, 107, 20, 19, 46, 101, 13, 72] and has been used to draw a parallel with the tectonics of eastern China in the context of the India-Eurasia collision [88, 64]. It is also worth noting that it has been carried out in one form or another in series of analogue modelling articles concerning the same region, with a rigid indenter colliding with a rheologically stratified lithosphere [69, 24, 49].

Numerically, the one-time step punch experiment is performed on a two-dimensional domain of purely plastic von Mises material. Given that the von Mises rheology yield criterion does not depend on pressure, the density of the material and/or the gravity vector is set to zero. Sides are set to free slip boundary conditions, the bottom to no slip, while a vertical velocity $(0, -v_p)$ is prescribed at the top boundary for nodes whose x coordinate is within $[L_x/2 - \delta/2, L_x/2 + \delta/2]$.

The following parameters are used: $L_x = 1$, $L_y = 0.5$, $\mu_{min} = 10^{-3}$, $\mu_{max} = 10^3$, $v_p = 1$, $\delta = 0.123456789$ and the yield value of the material is set to $k = 1$.

The analytical solution predicts that the angle of the shear bands stemming from the sides of the punch is $\pi/4$, that the pressure right under the punch is $1 + \pi$, and that the velocity of the rigid blocks on each side of the punch is $v_p/\sqrt{2}$ (this is simply explained by invoking conservation of mass).



ToDo: smooth punch

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (no-slip)
- isothermal
- non-isoviscous
- nonlinear rheology

16 `fieldstone_09`: the annulus benchmark

This benchmark is based on Thieulot & Puckett [Subm.] in which an analytical solution to the isoviscous incompressible Stokes equations is derived in an annulus geometry. The velocity and pressure fields are as follows:

$$
\begin{aligned}
v_r(r,\theta) &= g(r)k\sin(k\theta), && (76)\\
v_\theta(r,\theta) &= f(r)\cos(k\theta), && (77)\\
p(r,\theta) &= kh(r)\sin(k\theta), && (78)\\
\rho(r,\theta) &= \aleph(r)k\sin(k\theta), && (79)
\end{aligned}
$$

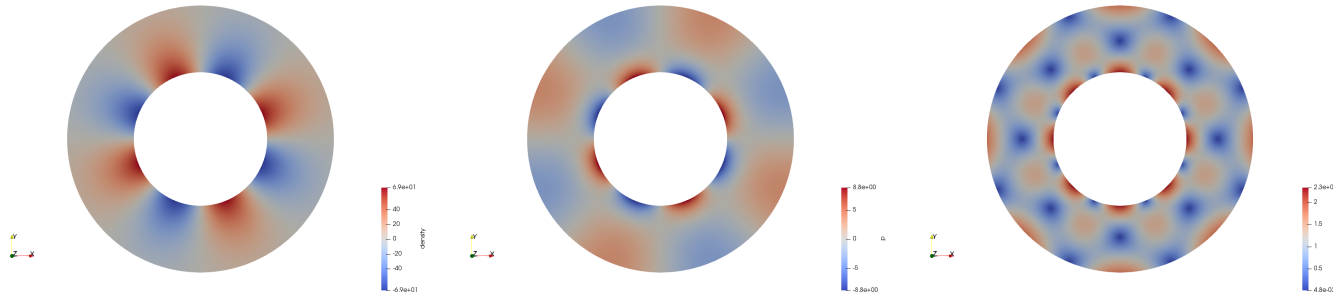
with

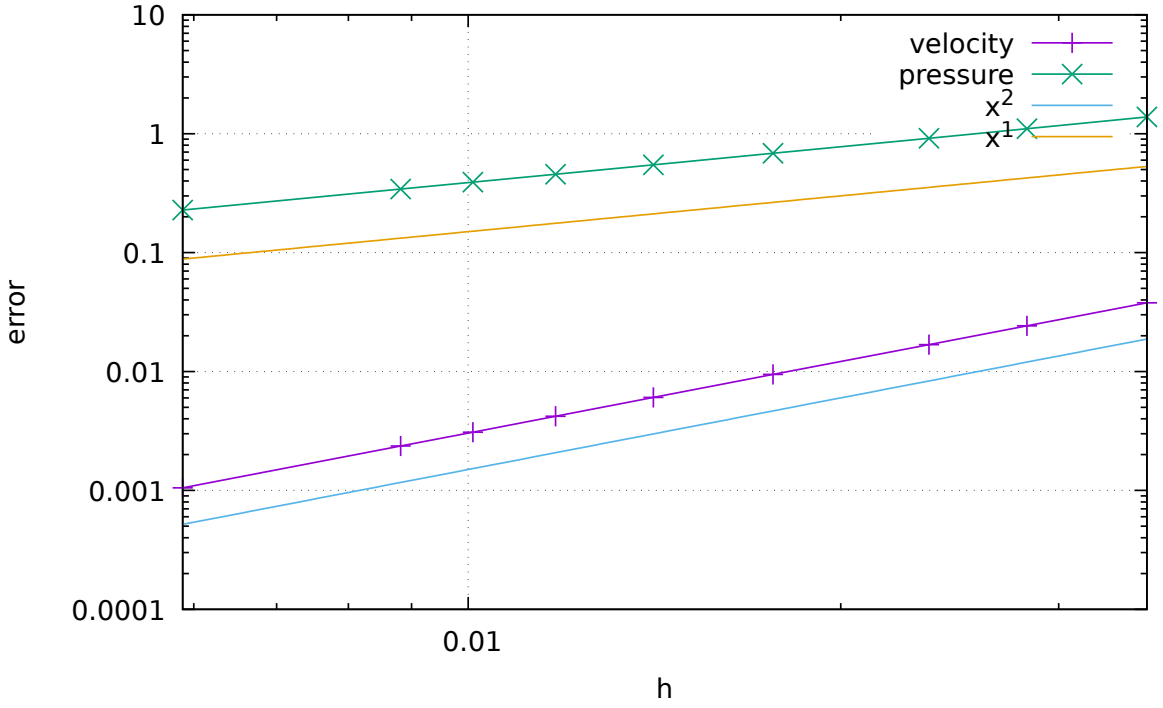
$$
\begin{aligned}
f(r) &= Ar + B/r, && (80)\\
g(r) &= \frac{A}{2}r + \frac{B}{r}\ln r + \frac{C}{r}, && (81)\\
h(r) &= \frac{2g(r)-f(r)}{r}, && (82)\\
\aleph(r) &= g'' - \frac{g'}{r} - \frac{g}{r^2}(k^2-1) + \frac{f}{r^2} + \frac{f'}{r}, && (83)\\
A &= -C\frac{2(\ln R_1 - \ln R_2)}{R_2^2\ln R_1 - R_1^2\ln R_2}, && (84)\\
B &= -C\frac{R_2^2 - R_1^2}{R_2^2\ln R_1 - R_1^2\ln R_2}. && (85)
\end{aligned}
$$

The parameters A and B are chosen so that $v_r(R_1) = v_r(R_2) = 0$, i.e. the velocity is tangential to both inner and outer surfaces. The gravity vector is radial and of unit length. In the present case, we set $R_1 = 1$, $R_2 = 2$ and $C = -1$.

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions
- direct solver
- isothermal
- isoviscous
- analytical solution
- annulus geometry
- elemental boundary conditions

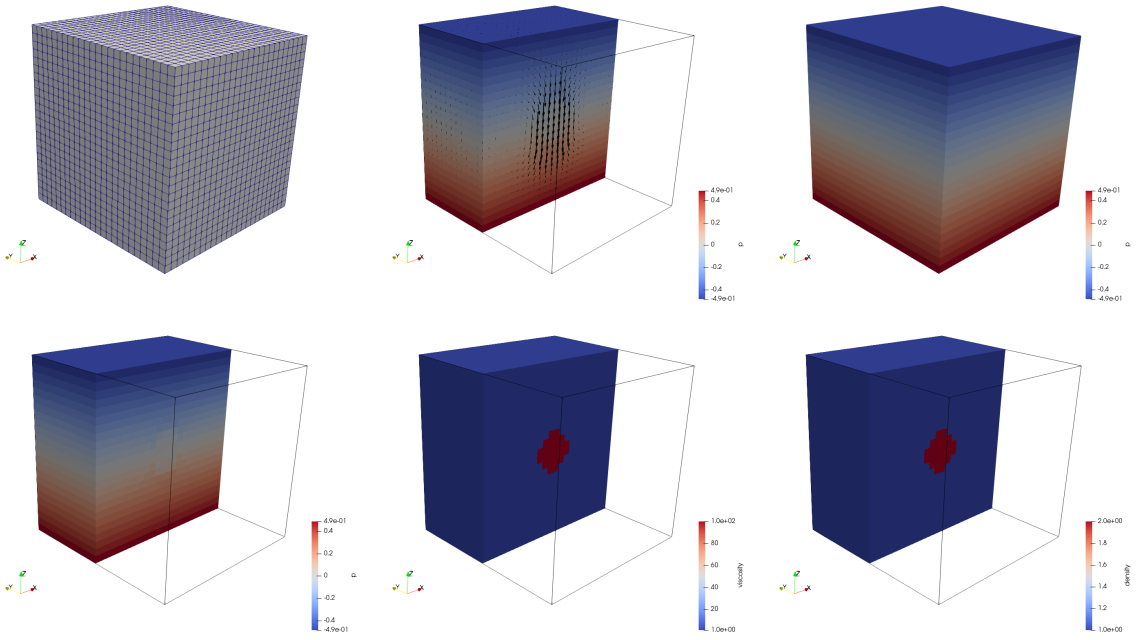



velocity
pressure
x^2
x^1
error
h
10
1
0.1
0.01
0.001
0.0001
0.01

17 `fieldstone_10`: Stokes sphere (3D) - penalty

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (free-slip)
- direct solver
- isothermal
- non-isoviscous
- 3D
- elemental b.c.
- buoyancy driven



resolution is 24x24x24

18 `fieldstone_11`: stokes sphere (3D) - mixed formulation

This is the same setup as Section 17.

features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- Dirichlet boundary conditions (free-slip)
- direct solver
- isothermal
- non-isoviscous
- 3D
- elemental b.c.
- buoyancy driven

19 `fieldstone_12`: consistent pressure recovery

What follows is presented in [106]. The second part of their paper wishes to establish a simple and effective numerical method to calculate variables eliminated by the penalisation process. The method involves an additional finite element solution for the nodal pressures using the same finite element basis and numerical quadrature as used for the velocity.

Let us start with:

$$p = -\lambda \boldsymbol{\nabla} \cdot \boldsymbol{v}$$

which lead to

$$(q, p) = -\lambda (q, \boldsymbol{\nabla} \cdot \boldsymbol{v})$$

and then

$$\left(\int \boldsymbol{N} \boldsymbol{N} d\Omega \right) \cdot \boldsymbol{P} = - \left(\lambda \int \boldsymbol{N} \boldsymbol{\nabla} \boldsymbol{N} d\Omega \right) \cdot \boldsymbol{V}$$

or,

$$\boldsymbol{M} \cdot \boldsymbol{P} = -\boldsymbol{D} \cdot \boldsymbol{V}$$

and finally

$$\boldsymbol{P} = -\boldsymbol{M}^{-1} \cdot \boldsymbol{D} \cdot V$$

with $\boldsymbol{M}$ of size $(np \times np)$, $\boldsymbol{D}$ of size $(np * ndof \times np * ndof)$ and $\boldsymbol{V}$ of size $(np * ndof)$. The vector $\boldsymbol{P}$ contains the np nodal pressure values directly, with no need for a smoothing scheme. The mass matrix $\boldsymbol{M}$ is to be evaluated at the full integration points, while the constraint part (the right hand side of the equation) is to be evaluated at the reduced integration point.

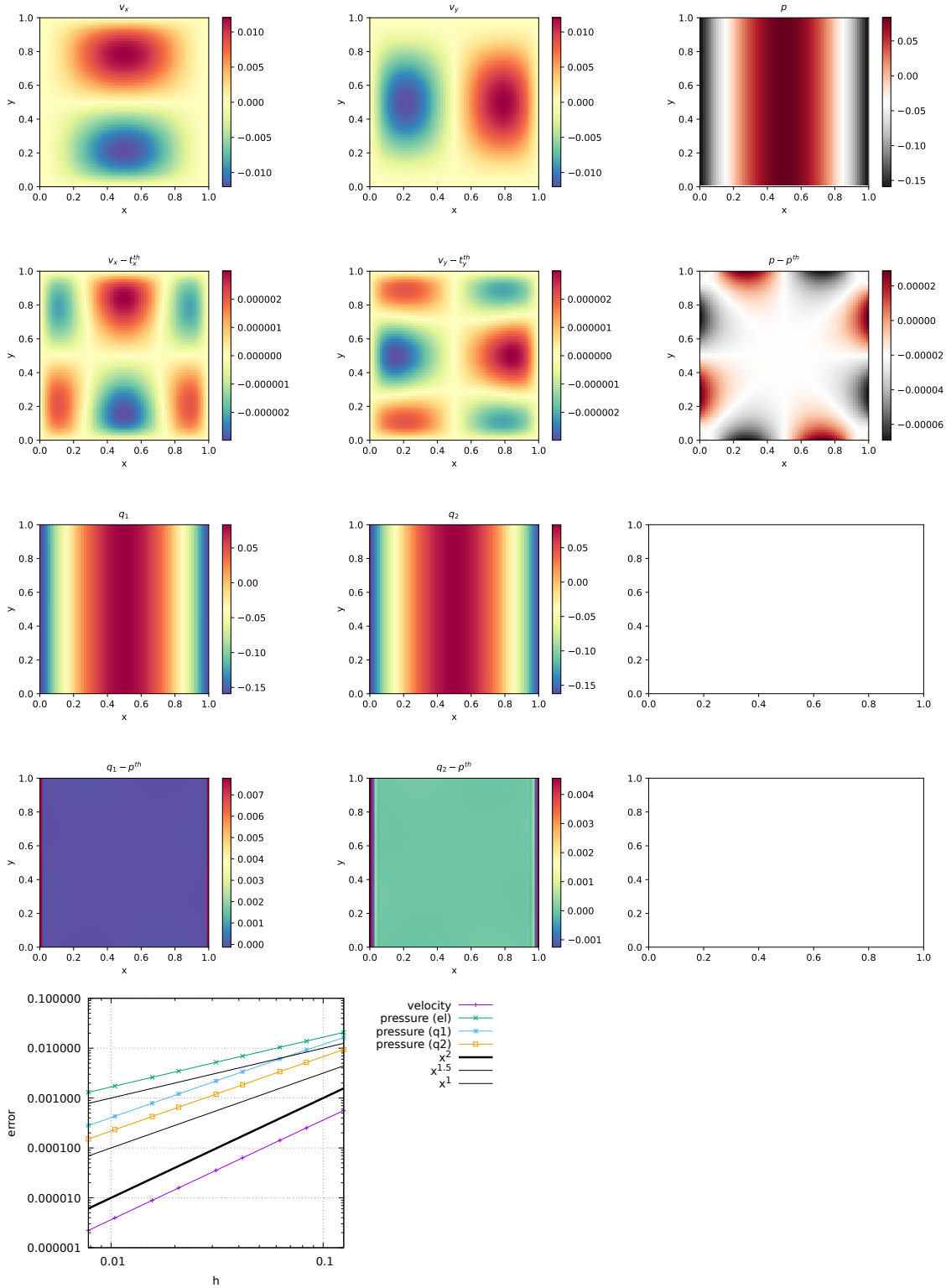
As noted by [106], it is interesting to note that when linear elements are used and the lumped matrices are used for the $\boldsymbol{M}$ the resulting algebraic equation is identical to the smoothing scheme based on the averaging method only if the uniform square finite element mesh is used. In this respect this method is expected to yield different results when elements are not square or even rectangular.

———-

q_1 is smoothed pressure obtained with the center-to-node approach.

q_2 is recovered pressure obtained with [106].

All three fulfill the zero average condition: $\int p d\Omega = 0$.



In terms of pressure error, q_2 is better than q_1 which is better than elemental.
QUESTION: why are the averages exactly zero ?!
TODO:

- add randomness to internal node positions.
- look at elefant algorithms

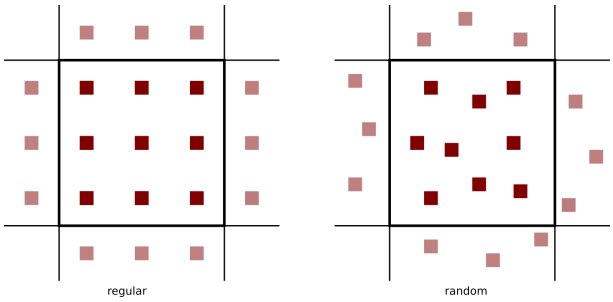
20 `fieldstone_13`: the Particle in Cell technique (1) - the effect of averaging

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (no-slip)
- isothermal
- non-isoviscous
- particle-in-cell

After the initial setup of the grid, markers can then be generated and placed in the domain. One could simply randomly generate the marker positions in the whole domain but unless a *very* large number of markers is used, the chance that an element does not contain any marker exists and this will prove problematic. In order to get a better control over the markers spatial distribution, one usually generates the marker per element, so that the total number of markers in the domain is the product of the number of elements times the user-chosen initial number of markers per element.

Our next concern is how to actually place the markers inside an element. Two methods come to mind: on a regular grid, or in a random manner, as shown on the following figure:



In both cases we make use of the basis shape functions: we generate the positions of the markers (random or regular) in the reference element first (r_{im}, s_{im}), and then map those out to the real element as follows:

$$x_{im} = \sum_i^m N_i(r_{im}, s_{im}) x_i \qquad y_{im} = \sum_i^m N_i(r_{im}, s_{im}) y_i$$

where x_i, y_i are the coordinates of the vertices of the element.

When using *active* markers, one is faced with the problem of transferring the properties they carry to the mesh on which the PDEs are to be solved. As we have seen, building the FE matrix involves a loop over all elements, so one simple approach consists of assigning each element a single property computed as the average of the values carried by the markers in that element. Often in colloquial language "average" refers to the arithmetic mean:

$$\langle \phi \rangle_{am} = \frac{1}{n} \sum_k^n \phi_i$$

where $< \phi >_{am}$ is the arithmetic average of the n numbers ϕ_i. However, in mathematics other means are commonly used, such as the geometric mean:

$$\langle \phi \rangle_{gm} = \left(\prod_i^n \phi_i \right)$$

and the harmonic mean:

$$\langle \phi \rangle_{hm} = \left(\frac{1}{n} \sum_i^n \frac{1}{\phi_i} \right)^{-1}$$

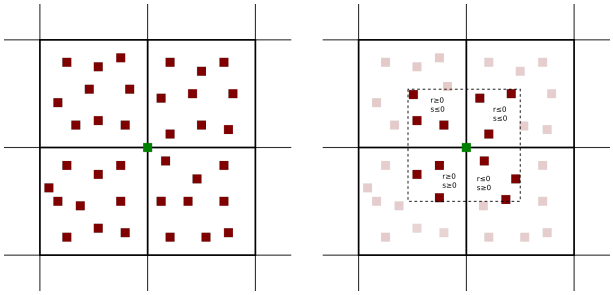
Furthermore, there is a well known inequality for any set of positive numbers,

$$\langle \phi \rangle_{am} \quad \geq \quad \langle \phi \rangle_{gm} \quad \geq \quad \langle \phi \rangle_{hm}$$

which will prove to be important later on.

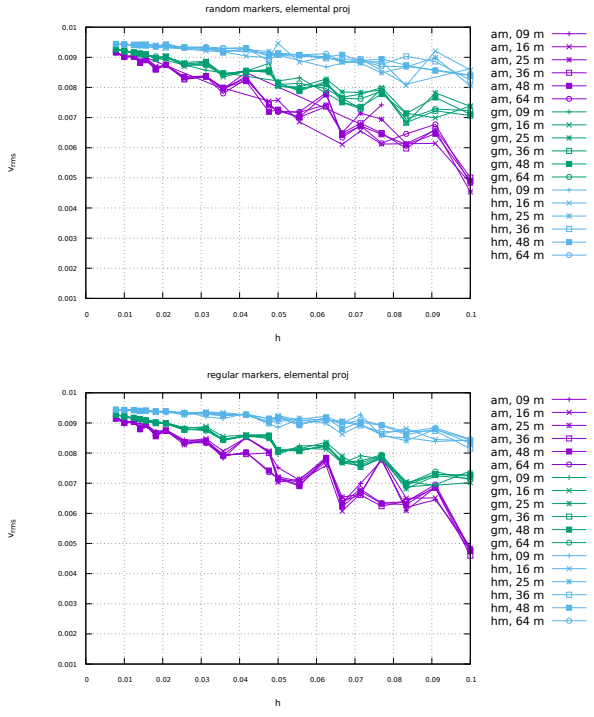
Let us now turn to a simple concrete example: the 2D Stokes sphere. There are two materials in the domain, so that markers carry the label "mat=1" or "mat=2". For each element an average density and viscosity need to be computed. The majority of elements contains markers with a single material label so that the choice of averaging does not matter (it is trivial to verify that if $\phi_i = \phi_0$ then $\langle \phi \rangle_{am} = \langle \phi \rangle_{gm} = \langle \phi \rangle_{hm} = \phi_0$. Remain the elements crossed by the interface between the two materials: they contain markers of both materials and the average density and viscosity inside those depends on 1) the total number of markers inside the element, 2) the ratio of markers 1 to markers 2, 3) the type of averaging.

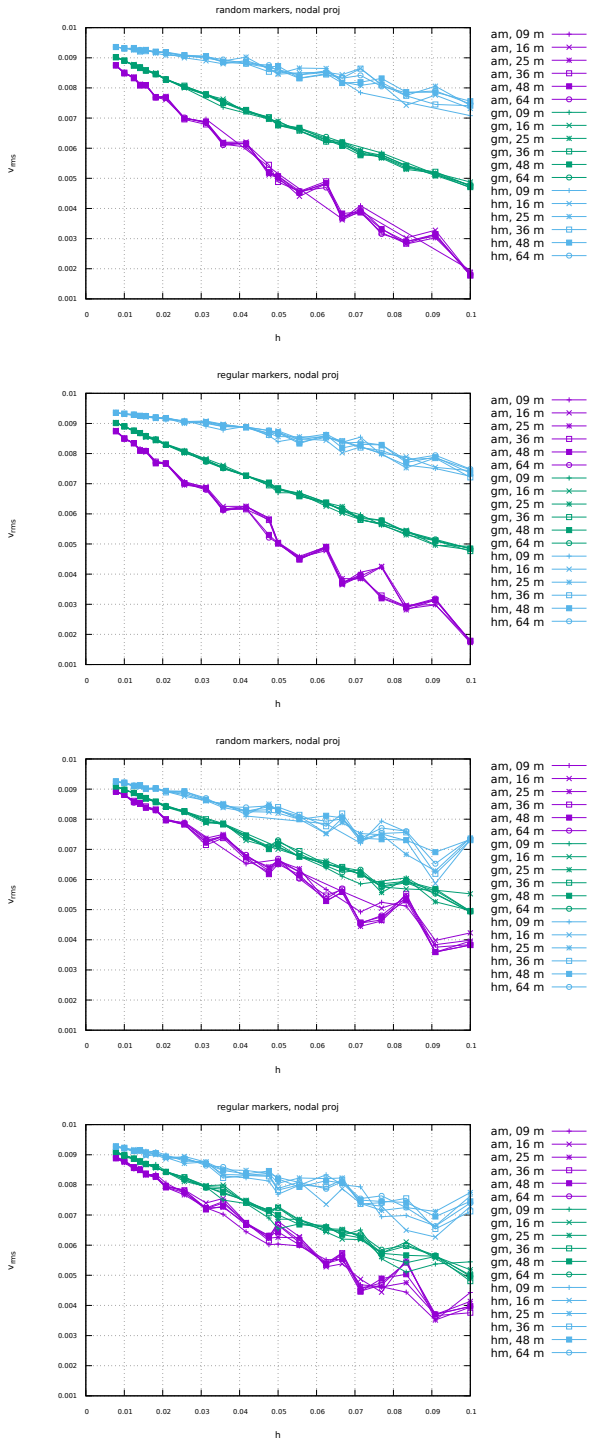
This averaging problem has been studied and documented in the literature [80, 26, 89, 71]



Nodal projection. Left: all markers inside elements to which the green node belongs to are taken into account. Right: only the markers closest to the green node count.

The setup is identical as the Stokes sphere experiment. The bash script *runall* runs the code for many resolutions, both initial marker distribution and all three averaging types. The viscosity of the sphere has been set to 10^3 while the viscosity of the surrounding fluid is 1. The average density is always computed with an arithmetic mean. Root mean square velocity results are shown hereunder:



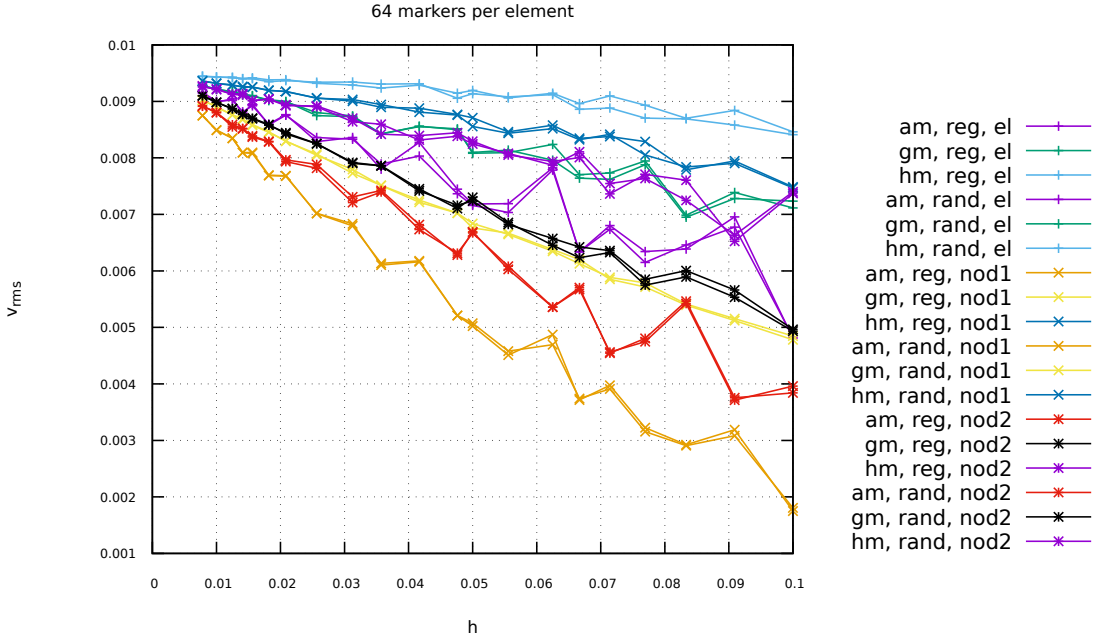


Left column: random markers. Right column: regular markers. Top row: elemental projection. Middle row: nodal 1 projection. Bottom row: nodal 2 projection.

Conclusions:

- With increasing resolution ($h \to 0$) vrms values seem to converge towards a single value, irrespective of the number of markers.
- At low resolution, say 32x32 (i.e. h=0.03125), vrms values for the three averagings differ by about 10%. At higher resolution, say 128x128, vrms values are still not converged.

- The number of markers per element plays a role at low resolution, but less and less with increasing resolution.
- Results for random and regular marker distributions are not identical but follow a similar trend and seem to converge to the same value.
- At low resolutions, elemental values yield better results.
- harmonic mean yields overal the best results



21 `fieldstone_f14`: solving the full saddle point problem

The details of the numerical setup are presented in Section **??**.

The main difference is that we no longer use the penalty formulation and therefore keep both velocity and pressure as unknowns. Therefore we end up having to solve the following system:

$$\begin{pmatrix} \mathbb{K} & \mathbb{G} \\ \mathbb{G}^T & 0 \end{pmatrix} \cdot \begin{pmatrix} V \\ P \end{pmatrix} = \begin{pmatrix} f \\ h \end{pmatrix} \qquad \text{or,} \qquad \mathbb{A} \cdot X = rhs$$

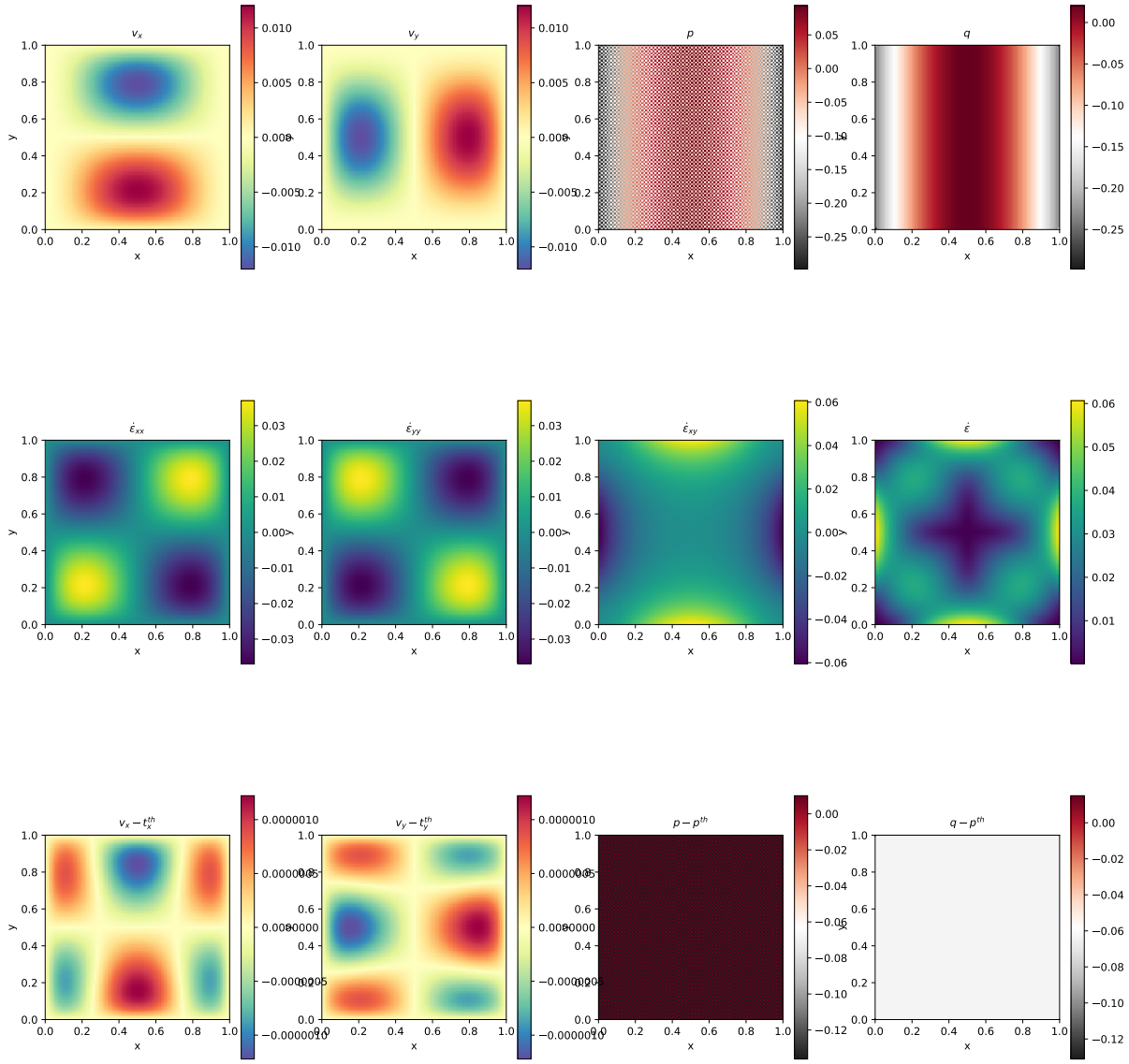
Each block $\mathbb{K}$, $\mathbb{G}$ and vector f, h are built separately in the code and assembled into the matrix $\mathbb{A}$ and vector rhs afterwards. $\mathbb{A}$ and rhs are then passed to the solver. We will see later that there are alternatives to solve this approach which do not require to build the full Stokes matrix $\mathbb{A}$.

Each element has $m = 4$ vertices so in total $ndofV \times m = 8$ velocity dofs and a single pressure dof, commonly situated in the center of the element. The total number of velocity dofs is therefore $NfemV = nnp \times ndofV$ while the total number of pressure dofs is $NfemP = nel$. The total number of dofs is then $Nfem = NfemV + NfemP$.

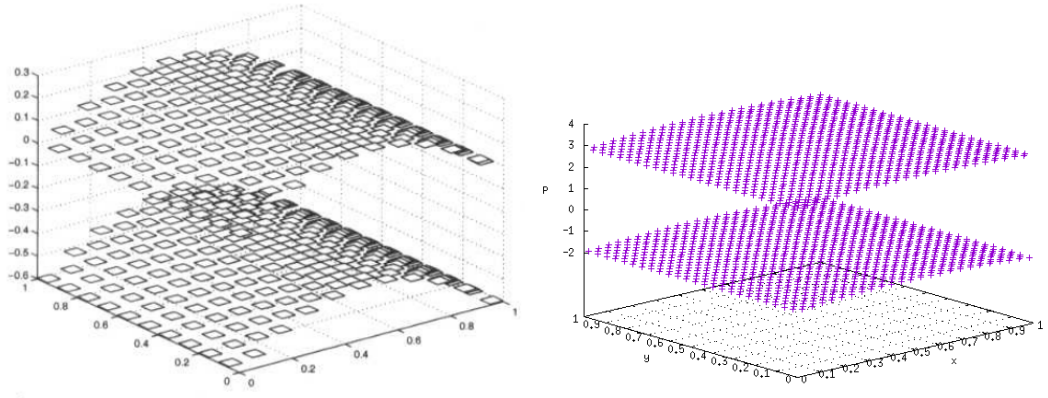
As a consequence, matrix $\mathbb{K}$ has size $NfemV, NfemV$ and matrix $\mathbb{G}$ has size $NfemV, NfemP$. Vector f is of size $NfemV$ and vector h is of size $NfemP$.

features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- Dirichlet boundary conditions (no-slip)
- direct solver (?)
- isothermal
- isoviscous
- analytical solution
- pressure smoothing



Unlike the results obtained with the penalty formualtion (see Section **??**), the pressure showcases a very strong checkerboard pattern, similar to the one in [29].

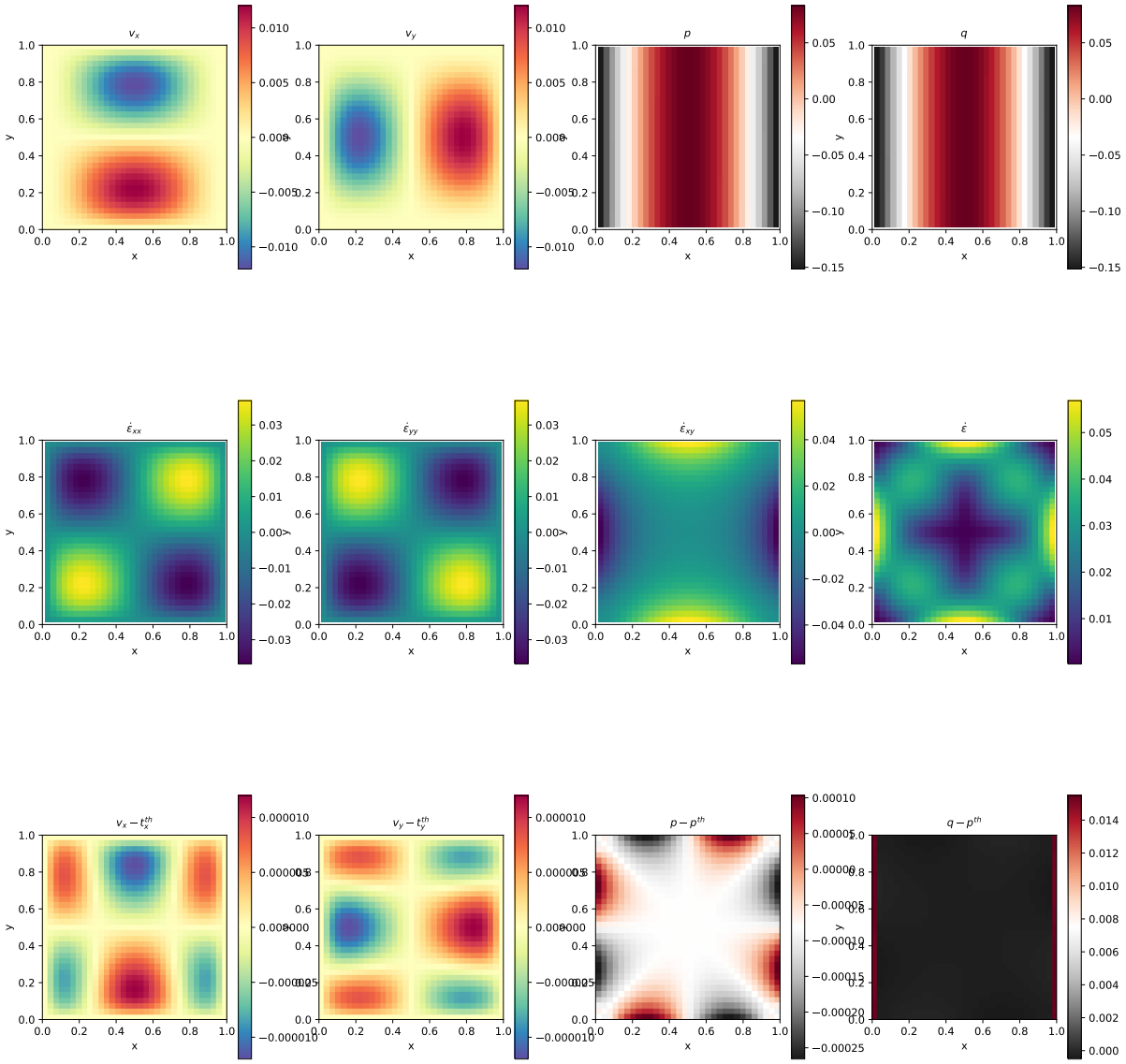


Left: pressure solution as shown in [29]; Right: pressure solution obtained with fieldstone.

Rather interestingly, the nodal pressure (obtained with a simple center-to-node algorithm) fails to recover a correct pressure at the four corners.

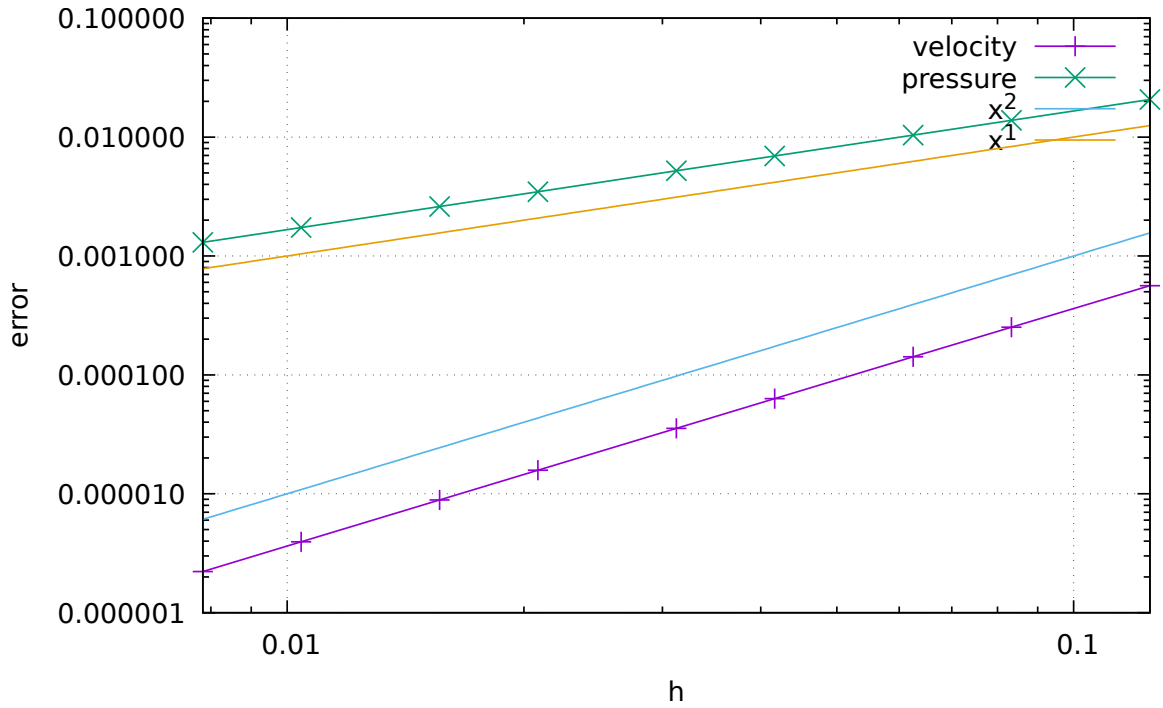
22 `fieldstone_f15`: saddle point problem with Schur complement approach - benchmark

The details of the numerical setup are presented in Section **??**. The main difference resides in the Schur complement approach to solve the Stokes system, as presented in Section 6.15 (see **solver_cg**). This iterative solver is very easy to implement once the blocks $\mathbb{K}$ and $\mathbb{G}$, as well as the rhs vectors f and h have been built.

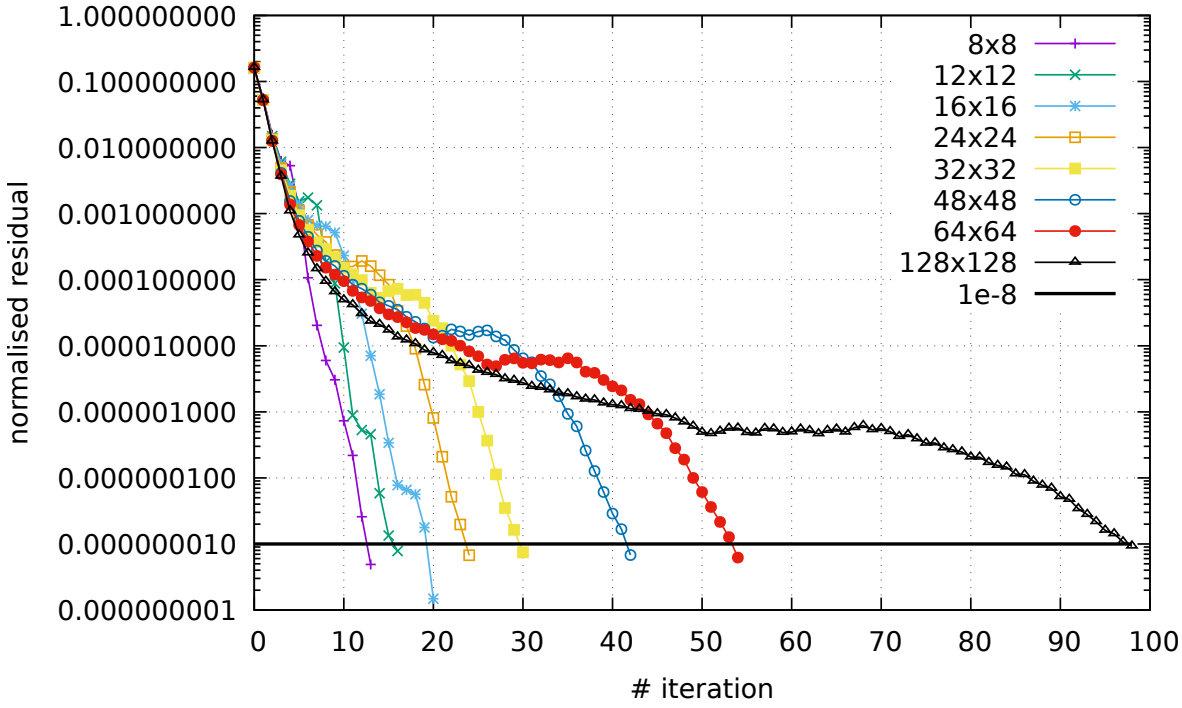


Rather interestingly the pressure checkerboard modes are not nearly as present as in Section **??** which uses a full matrix approach.

Looking at the discretisation errors for velocity and pressure, we of course recover the same rates and values as in the full matrix case.



Finally, for each experiment the normalised residual (see **solver_cg**) was recorded. We see that all things equal the resolution has a strong influence on the number of iterations the solver must perform to reach the required tolerance. This is one of the manifestations of the fact that the $Q_1 \times P_0$ element is not a stable element: the condition number of the matrix increases with resolution. We will see that this is not the case of stable elements such as $Q_2 \times Q_1$.



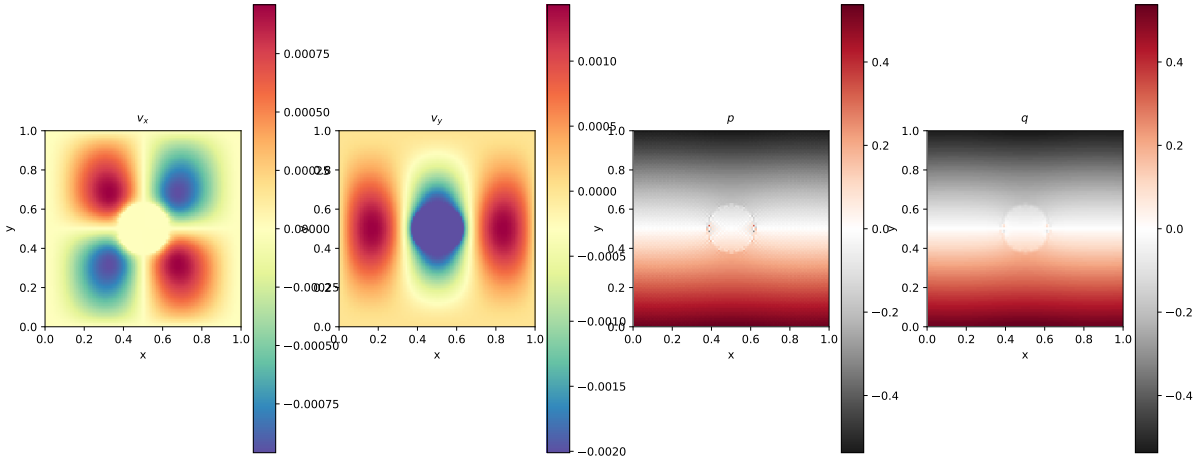
features

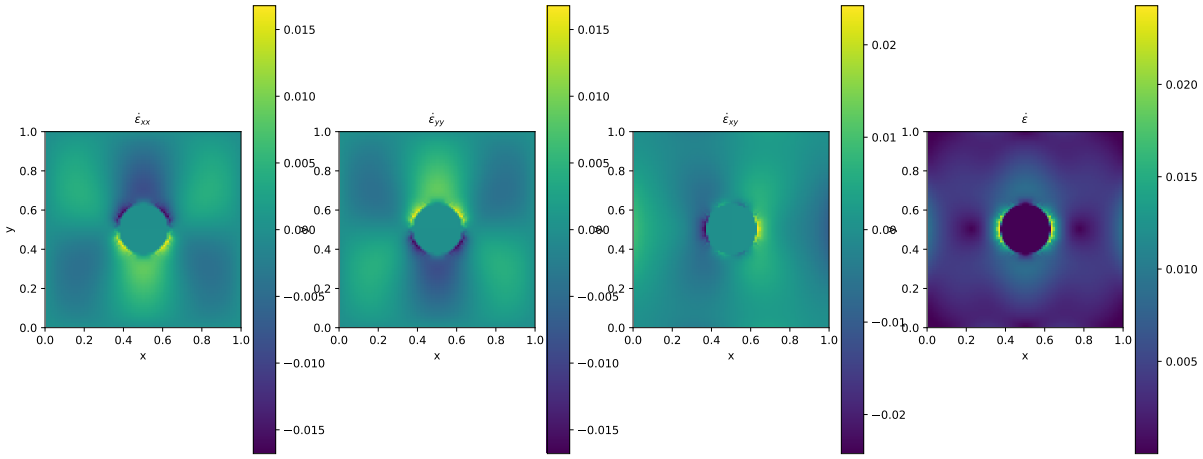
- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- Schur complement approach
- isothermal
- isoviscous
- analytical solution

build S and have python compute its smallest and largest eigenvalues as a function of resolution?

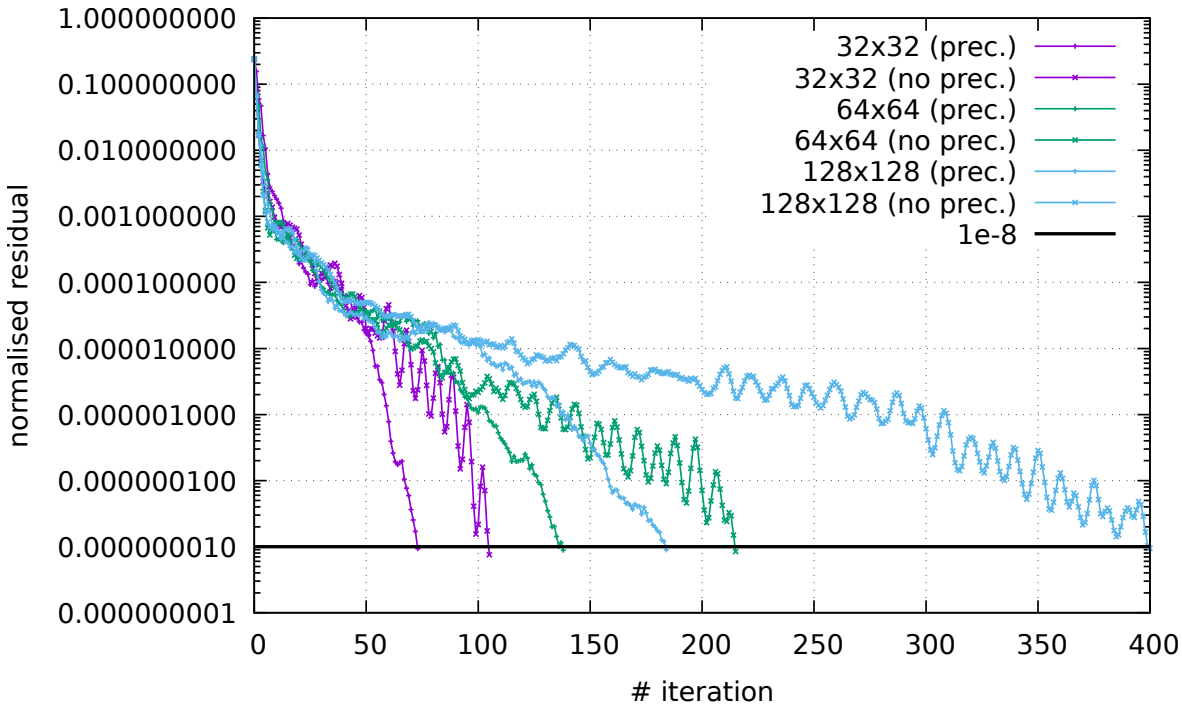
23 `fieldstone_f16`: saddle point problem with Schur complement approach - Stokes sphere

We are revisiting the 2D Stokes sphere problem, but this time we use the Schur complement approach to solve the Stokes system, Because there are viscosity contrasts in the domain, it is advisable to use the Preconditioned Conjugate Gradient as presented in Section 6.15 (see **solver_pcg**).





The normalised residual (see **solver_pcg**) was recorded. We see that all things equal the resolution has a strong influence on the number of iterations the solver must perform to reach the required tolerance. However, we see that the use of the preconditioner can substantially reduce the number of iterations inside the Stokes solver. At resolution 128x128, this number is halved.



features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- Schur complement approach
- isothermal
- non-isoviscous
- Stokes sphere

24 `fieldstone_17`: solving the full saddle point problem in 3D

When using $Q_1 \times P_0$ elements, this benchmark fails because of the Dirichlet b.c. on all 6 sides and all three components. However, as we will see, it does work well with $Q_2 \times Q_1$ elements. .

This benchmark begins by postulating a polynomial solution to the 3D Stokes equation [27]:

$$\boldsymbol{v} = \begin{pmatrix} x + x^2 + xy + x^3 y \\ y + xy + y^2 + x^2 y^2 \\ -2z - 3xz - 3yz - 5x^2 yz \end{pmatrix} \tag{86}$$

and

$$p = xyz + x^3 y^3 z - 5/32 \tag{87}$$

While it is then trivial to verify that this velocity field is divergence-free, the corresponding body force of the Stokes equation can be computed by inserting this solution into the momentum equation with a given viscosity μ (constant or position/velocity/strain rate dependent). The domain is a unit cube and velocity boundary conditions simply use Eq. (86). Following [15], the viscosity is given by the smoothly varying function

$$\mu = \exp(1 - \beta(x(1-x) + y(1-y) + z(1-z))) \tag{88}$$

One can easily show that the ratio of viscosities $\mu^\star$ in the system follows $\mu^\star = \exp(-3\beta/4)$ so that choosing $\beta = 10$ yields $\mu^\star \simeq 1808$ and $\beta = 20$ yields $\mu^\star \simeq 3.269 \times 10^6$.

We start from the momentum conservation equation:

$$-\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot (2\mu \dot{\boldsymbol{\epsilon}}) = \boldsymbol{f}$$

The x-component of this equation writes

$$\begin{aligned} f_x &= -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}(2\mu \dot{\epsilon}_{xx}) + \frac{\partial}{\partial y}(2\mu \dot{\epsilon}_{xy}) + \frac{\partial}{\partial z}(2\mu \dot{\epsilon}_{xz}) && (89) \\ &= -\frac{\partial p}{\partial x} + 2\mu \frac{\partial}{\partial x}\dot{\epsilon}_{xx} + 2\mu \frac{\partial}{\partial y}\dot{\epsilon}_{xy} + 2\mu \frac{\partial}{\partial z}\dot{\epsilon}_{xz} + 2\frac{\partial \mu}{\partial x}\dot{\epsilon}_{xx} + 2\frac{\partial \mu}{\partial y}\dot{\epsilon}_{xy} + 2\frac{\partial \mu}{\partial z}\dot{\epsilon}_{xz} && (90) \end{aligned}$$

Let us compute all the block separately:

$$\begin{aligned} \dot{\epsilon}_{xx} &= 1 + 2x + y + 3x^2 y \\ \dot{\epsilon}_{yy} &= 1 + x + 2y + 2x^2 y \\ \dot{\epsilon}_{zz} &= -2 - 3x - 3y - 5x^2 y \\ 2\dot{\epsilon}_{xy} &= (x + x^3) + (y + 2xy^2) = x + y + 2xy^2 + x^3 \\ 2\dot{\epsilon}_{xz} &= (0) + (-3z - 10xyz) = -3z - 10xyz \\ 2\dot{\epsilon}_{yz} &= (0) + (-3z - 5x^2 z) = -3z - 5x^2 z \end{aligned}$$

In passing, one can verify that $\dot{\epsilon}_{xx} + \dot{\epsilon}_{yy} + \dot{\epsilon}_{zz} = 0$. We further have

$$\begin{aligned}
\frac{\partial}{\partial x} 2\dot{\epsilon}_{xx} &= 2(2 + 6xy) \\
\frac{\partial}{\partial y} 2\dot{\epsilon}_{xy} &= 1 + 4xy \\
\frac{\partial}{\partial z} 2\dot{\epsilon}_{xz} &= -3 - 10xy \\
\frac{\partial}{\partial x} 2\dot{\epsilon}_{xy} &= 1 + 2y^2 + 3x^2 \\
\frac{\partial}{\partial y} 2\dot{\epsilon}_{yy} &= 2(2 + 2x^2) \\
\frac{\partial}{\partial z} 2\dot{\epsilon}_{yz} &= -3 - 5x^2 \\
\frac{\partial}{\partial x} 2\dot{\epsilon}_{xz} &= -10yz \\
\frac{\partial}{\partial y} 2\dot{\epsilon}_{yz} &= 0 \\
\frac{\partial}{\partial z} 2\dot{\epsilon}_{zz} &= 2(0)
\end{aligned}$$

$$\frac{\partial p}{\partial x} = yz + 3x^2y^3z \tag{91}$$

$$\frac{\partial p}{\partial y} = xz + 3x^3y^2z \tag{92}$$

$$\frac{\partial p}{\partial z} = xy + x^3y^3 \tag{93}$$

Pressure normalisation Here again, because Dirichlet boundary conditions are prescribed on all sides the pressure is known up to an arbitrary constant. This constant can be determined by (arbitrarily) choosing to normalised the pressure field as follows:

$$\int_\Omega p \, d\Omega = 0 \tag{94}$$

This is a single constraint associated to a single Lagrange multiplier λ and the global Stokes system takes the form

$$\begin{pmatrix} \mathbb{K} & \mathbb{G} & 0 \\ \mathbb{G}^T & 0 & \mathcal{C} \\ 0 & \mathcal{C}^T & 0 \end{pmatrix} \begin{pmatrix} V \\ P \\ \lambda \end{pmatrix}$$

In this particular case the constraint matrix $\mathcal{C}$ is a vector and it only acts on the pressure degrees of freedom because of Eq.(94). Its exact expression is as follows:

$$\int_\Omega p \, d\Omega = \sum_e \int_{\Omega_e} p \, d\Omega = \sum_e \int_{\Omega_e} \sum_i N_i^p p_i \, d\Omega = \sum_e \sum_i \left(\int_{\Omega_e} N_i^p \, d\Omega \right) p_i = \sum_e \mathcal{C}_e \cdot \boldsymbol{p}_e$$

where $\boldsymbol{p}_e$ is the list of pressure dofs of element e. The elemental constraint vector contains the corresponding pressure basis functions integrated over the element. These elemental constraints are then assembled into the vector $\mathcal{C}$.

24.0.1 Constant viscosity

Choosing $\beta = 0$ yields a constant velocity $\mu(x, y, z) = \exp(1) \simeq 2.718$ (and greatly simplifies the right-hand side) so that

$$\frac{\partial}{\partial x}\mu(x,y,z) = 0 \quad (95)$$

$$\frac{\partial}{\partial y}\mu(x,y,z) = 0 \quad (96)$$

$$\frac{\partial}{\partial z}\mu(x,y,z) = 0 \quad (97)$$

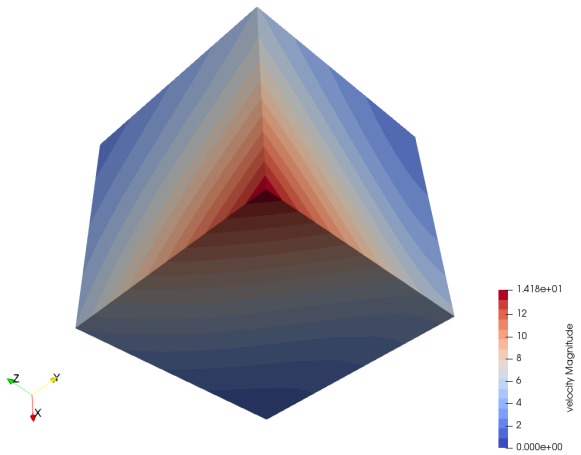
and

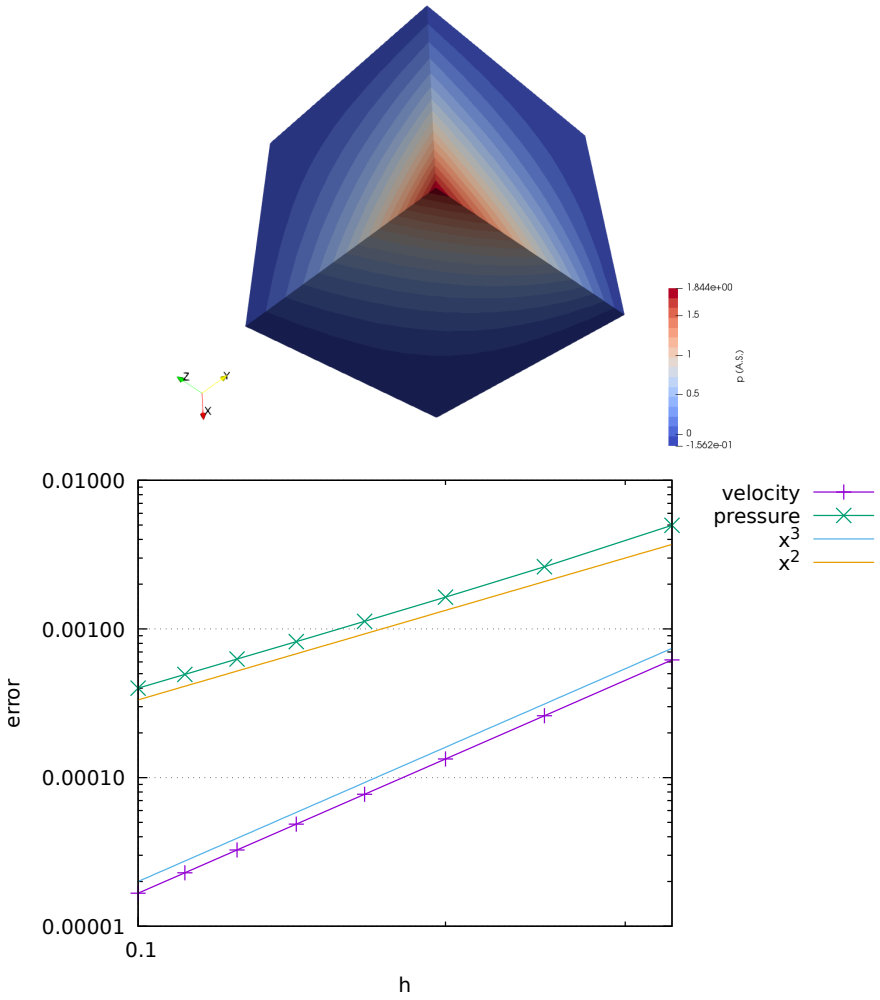
$$\begin{aligned}
f_x &= -\frac{\partial p}{\partial x} + 2\mu\frac{\partial}{\partial x}\dot{\epsilon}_{xx} + 2\mu\frac{\partial}{\partial y}\dot{\epsilon}_{xy} + 2\mu\frac{\partial}{\partial z}\dot{\epsilon}_{xz} \\
&= -(yz + 3x^2y^3z) + 2(2 + 6xy) + (1 + 4xy) + (-3 - 10xy) \\
&= -(yz + 3x^2y^3z) + \mu(2 + 6xy) \\
f_y &= -\frac{\partial p}{\partial y} + 2\mu\frac{\partial}{\partial x}\dot{\epsilon}_{xy} + 2\mu\frac{\partial}{\partial y}\dot{\epsilon}_{yy} + 2\mu\frac{\partial}{\partial z}\dot{\epsilon}_{yz} \\
&= -(xz + 3x^3y^2z) + \mu(1 + 2y^2 + 3x^2) + \mu 2(2 + 2x^2) + \mu(-3 - 5x^2) \\
&= -(xz + 3x^3y^2z) + \mu(2 + 2x^2 + 2y^2) \\
f_z &= -\frac{\partial p}{\partial z} + 2\mu\frac{\partial}{\partial x}\dot{\epsilon}_{xz} + 2\mu\frac{\partial}{\partial y}\dot{\epsilon}_{yz} + 2\mu\frac{\partial}{\partial z}\dot{\epsilon}_{zz} \\
&= -(xy + x^3y^3) + \mu(-10yz) + 0 + 0 \\
&= -(xy + x^3y^3) + \mu(-10yz)
\end{aligned}$$

Finally

$$\boldsymbol{f} = -\begin{pmatrix} yz + 3x^2y^3z \\ xz + 3x^3y^2z \\ xy + x^3y^3 \end{pmatrix} + \mu\begin{pmatrix} 2 + 6xy \\ 2 + 2x^2 + 2y^2 \\ -10yz \end{pmatrix}$$

Note that there seems to be a sign problem with Eq.(26) in [15].





24.0.2 Variable viscosity

The spatial derivatives of the viscosity are then given by

$$\begin{aligned}
\frac{\partial}{\partial x}\mu(x,y,z) &= -(1-2x)\beta\mu(x,y,z) \\
\frac{\partial}{\partial y}\mu(x,y,z) &= -(1-2y)\beta\mu(x,y,z) \\
\frac{\partial}{\partial z}\mu(x,y,z) &= -(1-2z)\beta\mu(x,y,z)
\end{aligned}$$

and thr right-hand side by

$$
\begin{aligned}
\boldsymbol{f} &= -\begin{pmatrix} yz+3x^2y^3z \\ xz+3x^3y^2z \\ xy+x^3y^3 \end{pmatrix} + \mu \begin{pmatrix} 2+6xy \\ 2+2x^2+2y^2 \\ -10yz \end{pmatrix} \\
&\quad -(1-2x)\beta\mu(x,y,z)\begin{pmatrix} 2\dot{\epsilon}_{xx} \\ 2\dot{\epsilon}_{xy} \\ 2\dot{\epsilon}_{xz} \end{pmatrix} - (1-2y)\beta\mu(x,y,z)\begin{pmatrix} 2\dot{\epsilon}_{xy} \\ 2\dot{\epsilon}_{yy} \\ 2\dot{\epsilon}_{yz} \end{pmatrix} - (1-2z)\beta\mu(x,y,z)\begin{pmatrix} 2\dot{\epsilon}_{xz} \\ 2\dot{\epsilon}_{yz} \\ 2\dot{\epsilon}_{zz} \end{pmatrix} \\
&= -\begin{pmatrix} yz+3x^2y^3z \\ xz+3x^3y^2z \\ xy+x^3y^3 \end{pmatrix} + \mu \begin{pmatrix} 2+6xy \\ 2+2x^2+2y^2 \\ -10yz \end{pmatrix} \\
&- (1-2x)\beta\mu \begin{pmatrix} 2+4x+2y+6x^2y \\ x+y+2xy^2+x^3 \\ -3z-10xyz \end{pmatrix} - (1-2y)\beta\mu \begin{pmatrix} x+y+2xy^2+x^3 \\ 2+2x+4y+4x^2y \\ -3z-5x^2z \end{pmatrix} - (1-2z)\beta\mu \begin{pmatrix} -3z-10xyz \\ -3z-5x^2z \\ -4-6x-6y-10x \end{pmatrix}
\end{aligned}
$$

Note that at $(x,y,z)=(0,0,0)$, $\mu=\exp(1)$, and at $(x,y,z)=(0.5,0.5,0.5)$, $\mu=\exp(1-3\beta/4)$ so that the maximum viscosity ratio is given by

$$\mu^\star = \frac{\exp(1-3\beta/4)}{\exp(1)} = \exp(-3\beta/4)$$

By varying β between 1 and 22 we can get up to 7 orders of magnitude viscosity difference.

features

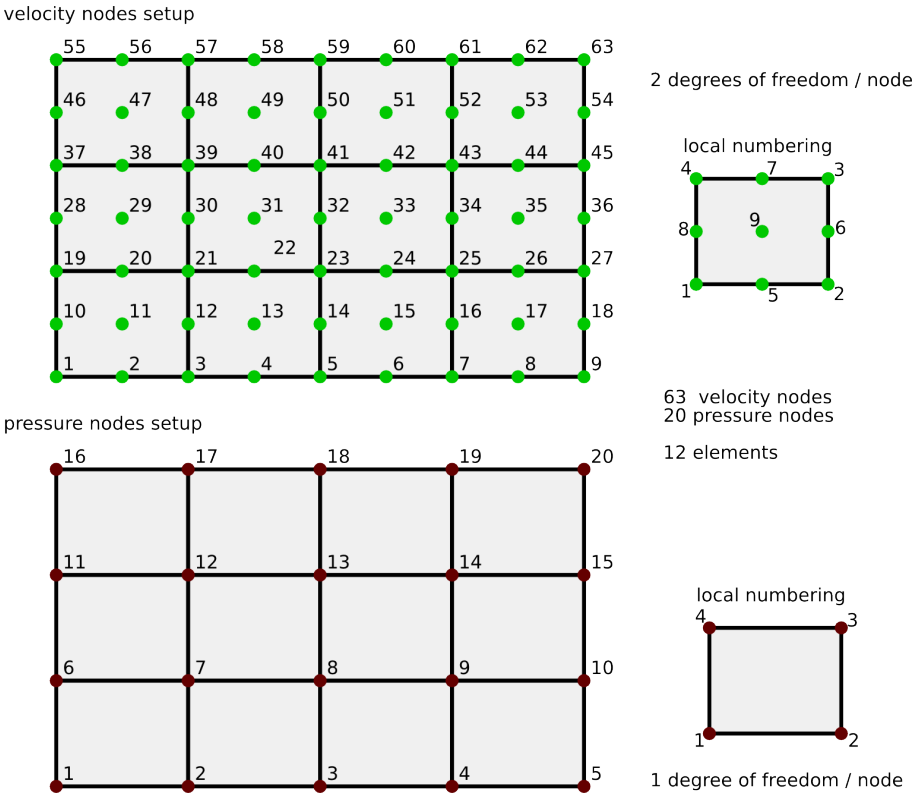
- $Q_1 \times P_0$ element
- incompressible flow
- saddle point system
- Dirichlet boundary conditions (free-slip)
- direct solver
- isothermal
- non-isoviscous
- 3D
- elemental b.c.
- analytical solution

25 `fieldstone_18`: solving the full saddle point problem with $Q_2 \times Q_1$ elements

The details of the numerical setup are presented in Section **??**.

Each element has $m_V = 9$ vertices so in total $ndof_V \times m_V = 18$ velocity dofs and $ndof_P * m_P = 4$ pressure dofs. The total number of velocity dofs is therefore $NfemV = nnp \times ndofV$ while the total number of pressure dofs is $NfemP = nel$. The total number of dofs is then $Nfem = NfemV + NfemP$.

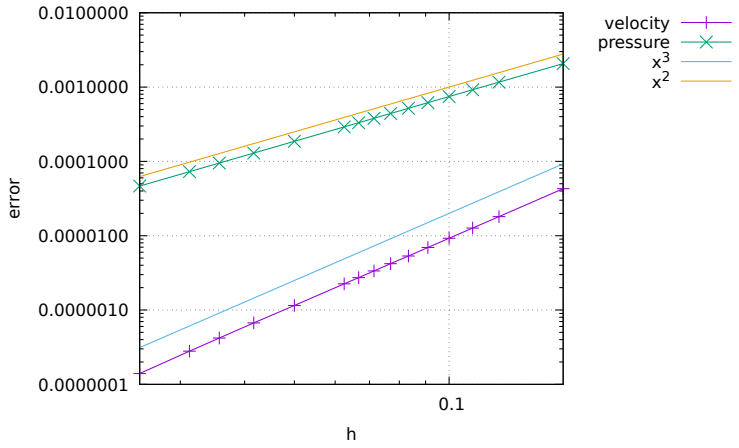
As a consequence, matrix $\mathbb{K}$ has size $NfemV, NfemV$ and matrix $\mathbb{G}$ has size $NfemV, NfemP$. Vector f is of size $NfemV$ and vector h is of size $NfemP$.



renumber all nodes to start at zero!! Also internal numbering does not work here

features

- $Q_2 \times Q_1$ element
- incompressible flow
- mixed formulation
- Dirichlet boundary conditions (no-slip)
- isothermal
- isoviscous
- analytical solution


0.0100000
0.0010000
0.0001000
0.0000100
0.0000010
0.0000001
error
0.1
h
velocity
pressure
x3
x2

26 `fieldstone_19`: solving the full saddle point problem with $Q_3 \times Q_2$ elements

The details of the numerical setup are presented in Section **??**.

Each element has $m_V = 16$ vertices so in total $ndof_V \times m_V = 32$ velocity dofs and $ndof_P * m_P = 9$ pressure dofs. The total number of velocity dofs is therefore $NfemV = nnp \times ndofV$ while the total number of pressure dofs is $NfemP = nel$. The total number of dofs is then $Nfem = NfemV + NfemP$.

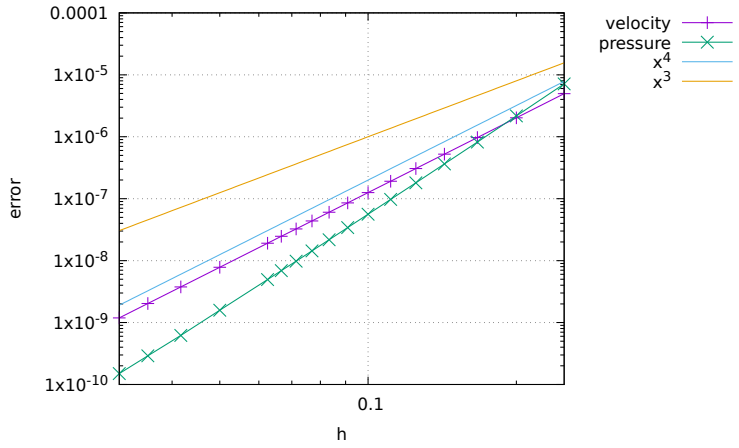
As a consequence, matrix $\mathbb{K}$ has size $NfemV, NfemV$ and matrix $\mathbb{G}$ has size $NfemV, NfemP$. Vector f is of size $NfemV$ and vector h is of size $NfemP$.

```
60===61===62===63===64===65===66===67===68===70
||             ||             ||             ||
50   51   52   53   54   55   56   57   58   59
||             ||             ||             ||
40   41   42   43   44   45   46   47   48   49
||             ||             ||             ||
30===31===32===33===34===35===36===37===38===39
||             ||             ||             ||
20   21   22   23   24   25   26   27   28   29
||             ||             ||             ||
10   11   12   13   14   15   16   17   18   19
||             ||             ||             ||
00===01===02===03===04===05===06===07===08===09

Example of 3x2 mesh. nnx=10, nny=7, nnp=70, nelx=3, nely=2, nel=6

12===13===14===15           06=====07=====08
||   ||   ||   ||           ||     ||     ||
08===09===10===11           ||     ||     ||
||   ||   ||   ||           03=====04=====05
04===05===06===07           ||     ||     ||
||   ||   ||   ||           ||     ||     ||
00===01===02===03           00=====01=====02

Velocity (Q3)               Pressure (Q2)

(r,s)_{00}=(-1,-1)          (r,s)_{00}=(-1,-1)
(r,s)_{01}=(-1/3,-1)        (r,s)_{01}=(0,-1)
(r,s)_{02}=(+1/3,-1)        (r,s)_{02}=(+1,-1)
(r,s)_{03}=(+1,-1)          (r,s)_{03}=(-1,0)
(r,s)_{04}=(-1,-1/3)        (r,s)_{04}=(0,0)
(r,s)_{05}=(-1/3,-1/3)      (r,s)_{05}=(+1,0)
(r,s)_{06}=(+1/3,-1/3)      (r,s)_{06}=(-1,+1)
(r,s)_{07}=(+1,-1/3)        (r,s)_{07}=(0,+1)
(r,s)_{08}=(-1,+1/3)        (r,s)_{08}=(+1,+1)
(r,s)_{09}=(-1/3,+1/3)
(r,s)_{10}=(+1/3,+1/3)
(r,s)_{11}=(+1,+1/3)
(r,s)_{12}=(-1,+1)
(r,s)_{13}=(-1/3,+1)
(r,s)_{14}=(+1/3,+1)
(r,s)_{15}=(+1,+1)
```

Write about 4 point quadrature.

features

- $Q_3 \times Q_2$ element
- incompressible flow
- mixed formulation
- isothermal
- isoviscous
- analytical solution



velocity error rate is cubic, pressure superconvergent since the pressure field is quadratic and therefore lies into the Q2 space.

27 `fieldstone_20`: the Busse benchmark

This three-dimensional benchmark was first proposed by [16]. It has been subsequently presented in [86, 94, 1, 68, 23, 54]. We here focus on Case 1 of [16]: an isoviscous bimodal convection experiment at $Ra = 3 \times 10^5$.

The domain is of size $a \times b \times h$ with $a = 1.0079h$, $b = 0.6283h$ with $h = 2700$km. It is filled with a Newtonian fluid characterised by $\rho_0 = 3300\text{kg.m}^{-3}$, $\alpha = 10^{-5}\text{K}^{-1}$, $\mu = 8.0198 \times 10^{23}$Pa.s, $k = 3.564\text{W.m}^{-1}\text{.K}^{-1}$, $c_p = 1080\text{J.K}^{-1}\text{.kg}^{-1}$. The gravity vector is set to $\boldsymbol{g} = (0, 0, -10)^T$. The temperature is imposed at the bottom ($T = 3700°$C) and at the top ($T = 0°$C).

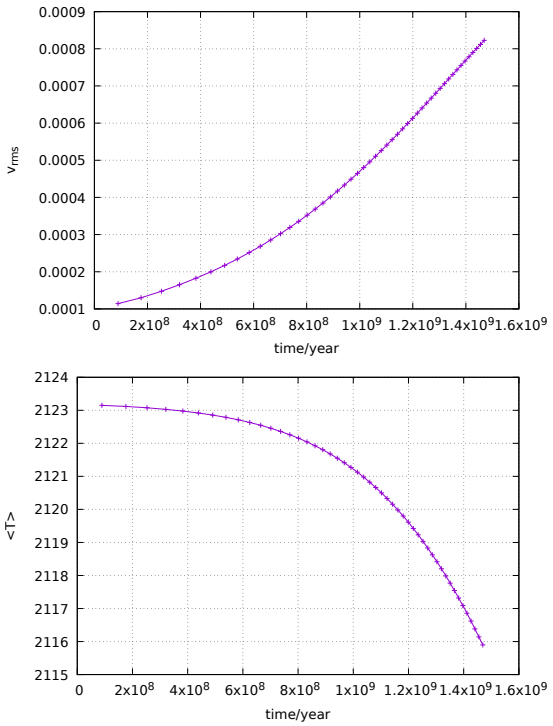
The various measurements presented in [16] are listed hereafter:

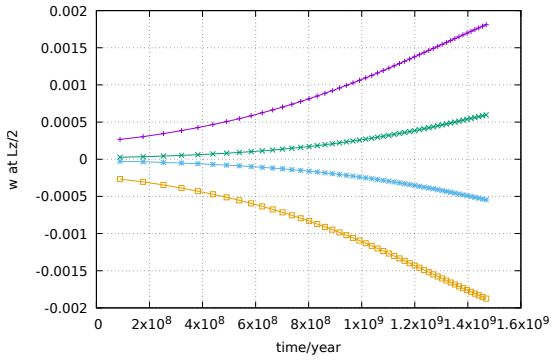
- The Nusselt number Nu computed at the top surface following Eq. (68):

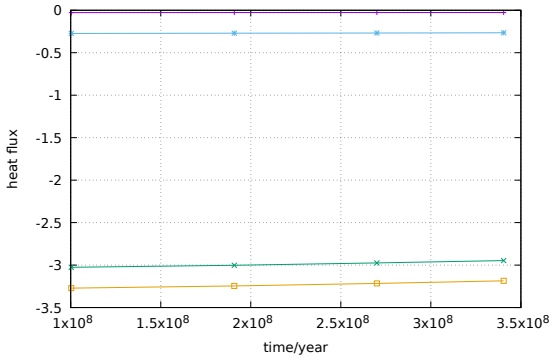
$$Nu = L_z \frac{\int\int_{z=L_z} \frac{\partial T}{\partial y} dxdy}{\int\int_{z=0} T dxdy}$$

- the root mean square velocity v_{rms} and the temperature mean square velocity T_{rms}
- The vertical velocity w and temperature T at points $\boldsymbol{x}_1 = (0, 0, L_z/2)$, $\boldsymbol{x}_2 = (L_x, 0, L_z/2)$, $\boldsymbol{x}_3 = (0, L_y, L_z/2)$ and $\boldsymbol{x}_4 = (L_x, L_y, L_z/2)$;
- the vertical component of the heat flux Q at the top surface at all four corners.

The values plotted hereunder are adimensionalised by means of a reference temperature (3700K), a reference lengthscale 2700km, and a reference time L_z^2/κ.







features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- Dirichlet boundary conditions (free-slip)
- direct solver
- isothermal
- non-isoviscous
- 3D
- elemental b.c.
- buoyancy driven

ToDo: look at energy conservation. run to steady state and make sure the expected values are retrieved.

28 `fieldstone_21`: The non-conforming $Q_1 \times P_0$ element

features

- Non-conforming $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- isothermal
- non-isoviscous
- analytical solution
- pressure smoothing

try Q1 mapping instead of isoparametric.

29 `fieldstone_22`: The stabilised $Q_1 \times Q_1$ element

The details of the numerical setup are presented in Section **??**.

We will here focus on the $Q_1 \times Q_1$ element, which, unless stabilised, violates the LBB stability condition and therefore is unusable. Stabilisation can be of two types: least-squares (see for example [30]) [**?**, 51, **?**], or by means of an additional term in the weak form as first introduced in [27, 10]. It is further analysed in [67, 58]. Note that an equal-order velocity-pressure formulation that does not exhibit spurious pressure modes (without stabilisaion) has been presented in [75].

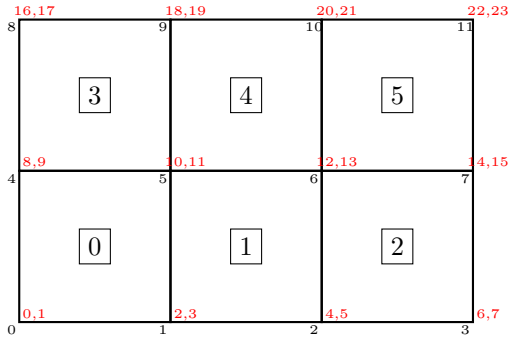
This element corresponds to Bilinear velocities, bilinear pressure (equal order interpolation for both velocity and pressure) which is very convenient in terms of data structures since all dofs are colocated.

In geodynamics, it is used in the Rhea code [84, 15] and in Gale. It is also used in [57] in its stabilised form, in conjunction with AMR.

The stabilisation term $\mathbb{C}$ enters the Stokes matrix in the (2,2) position:

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

Let us take a simple example: a 3x2 element grid.



The $\mathbb{K}$ matrix is of size $NfemV \times NfemV$ with $NfemV = ndofV \times nnp = 2 \times 12 = 24$. The $\mathbb{G}$ matrix is of size $NfemV \times NfemP$ with $NfemP = ndofP \times nnp = 1 \times 12 = 12$. The $\mathbb{C}$ matrix is of size $NfemP \times NfemP$.

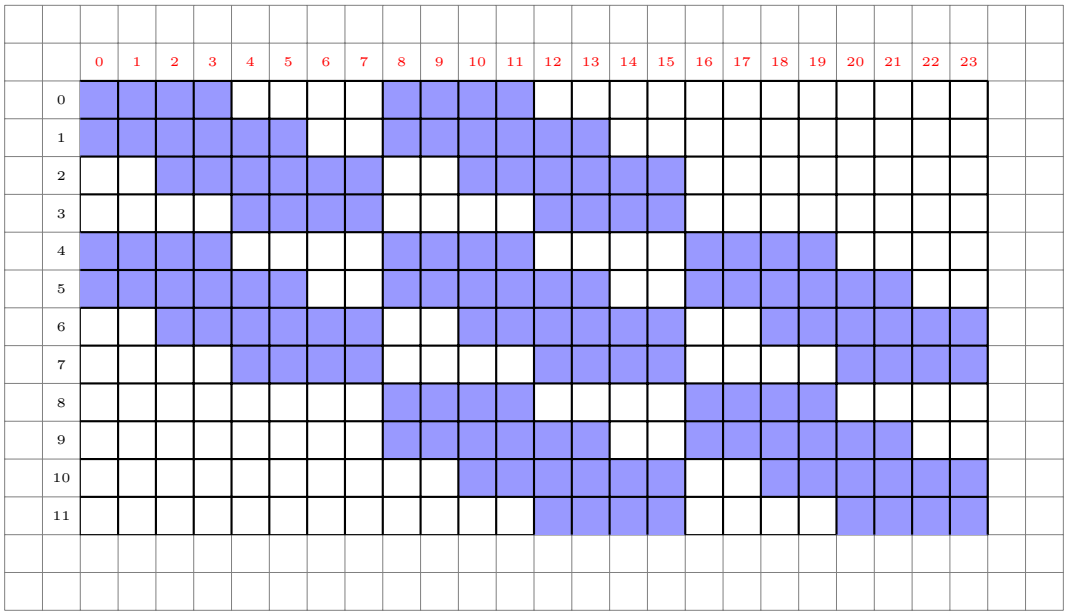
A corner pdof sees 4 vdofs, a side pdof sees 12 vdofs and an inside pdof sees 18 vdofs, so that the total number of nonzeros in $\mathbb{G}$ can be computed as follows:

$$NZ_{\mathbb{G}} = \underbrace{4}_{corners} + \underbrace{2(nnx-2)*12}_{2hor.sides} + \underbrace{2(nny-2)*12}_{2vert.sides} + \underbrace{(nnx-2)(nny-2)*18}_{insidenodes}$$

Concretely,

- pdof #0 sees vdofs 0,1,2,3,8,9,10,11
- pdof #1 sees vdofs 0,1,2,3,4,5,8,9,10,11,12,13
- pdof #5 sees vdofs 0,1,2,3,4,5,8,9,10,11,12,13,16,17,18,19,20,21

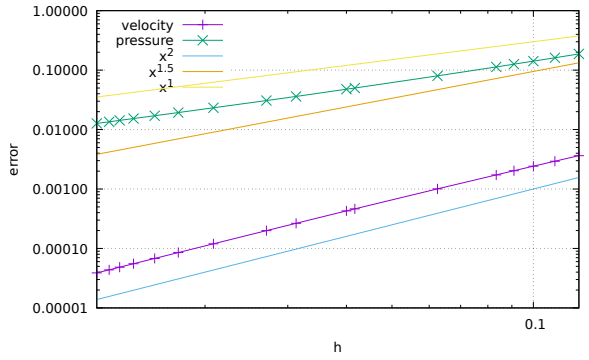
so that the $\mathbb{G}^T$ matrix non-zero structure then is as follows:

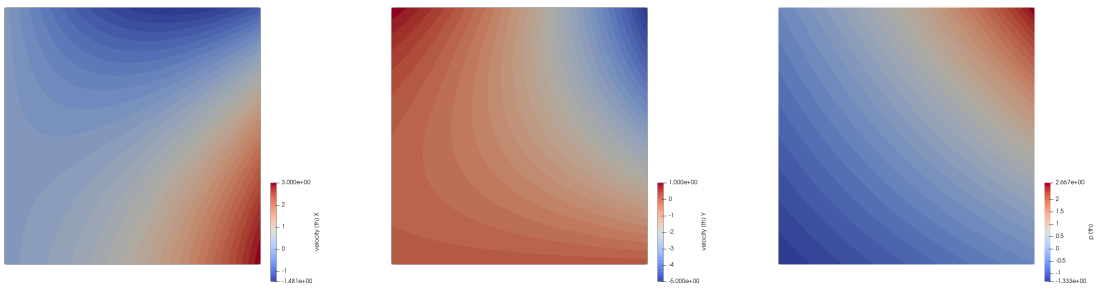


We will here use the formulation of [27], which is very appealing since it does not require any free (user-chosen) parameter. Furthermore the matrix $\mathbb{C}$ is rather simple to compute as we will see hereafter. We also impose $\int pdV = 0$ which means that the following constraint is added to the Stokes matrix:

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

As in [27] we solve the benchmark problem presented in 6.7.2





30 `fieldstone_23`: compressible flow (1) - analytical benchmark

This work is part of the MSc thesis of T. Weir (2018).

We first start with an isothermal Stokes flow, so that we disregard the heat transport equation and the equations we wish to solve are simply:

$$-\nabla\cdot\left[2\eta\left(\dot{\varepsilon}(\boldsymbol{v})-\frac{1}{3}(\nabla\cdot\boldsymbol{v})\mathbf{1}\right)\right]+\nabla p=\rho\boldsymbol{g} \qquad \text{in } \Omega, \tag{99}$$

$$\nabla\cdot(\rho\boldsymbol{v})=0 \qquad \text{in } \Omega \tag{100}$$

The second equation can be rewritten $\nabla\cdot(\rho\boldsymbol{v})=\rho\nabla\cdot\boldsymbol{v}+\boldsymbol{v}\cdot\boldsymbol{\nabla}\rho=0$ or,

$$\nabla\cdot\boldsymbol{v}+\frac{1}{\rho}\boldsymbol{v}\cdot\boldsymbol{\nabla}\rho=0$$

Note that this presupposes that the density is not zero anywhere in the domain.

We use a mixed formulation and therefore keep both velocity and pressure as unknowns. We end up having to solve the following system:

$$\begin{pmatrix}\mathbb{K} & \mathbb{G}\\ \mathbb{G}^T+\mathbb{Z} & 0\end{pmatrix}\cdot\begin{pmatrix}\mathcal{V}\\ \mathcal{P}\end{pmatrix}=\begin{pmatrix}f\\ h\end{pmatrix} \qquad \text{or,} \qquad \mathbb{A}\cdot X=rhs$$

Where $\mathbb{K}$ is the stiffness matrix, $\mathbb{G}$ is the discrete gradient operator, $\mathbb{G}^T$ is the discrete divergence operator, $\mathcal{V}$ the velocity vector, $\mathcal{P}$ the pressure vector. Note that the term $\mathbb{Z}\mathcal{V}$ derives from term $\boldsymbol{v}\cdot\boldsymbol{\nabla}\rho$ in the continuity equation.

Each block $\mathbb{K}$, $\mathbb{G}$, $\mathbb{Z}$ and vectors f and h are built separately in the code and assembled into the matrix $\mathbb{A}$ and vector rhs afterwards. $\mathbb{A}$ and rhs are then passed to the solver. We will see later that there are alternatives to solve this approach which do not require to build the full Stokes matrix $\mathbb{A}$.

Remark: the term $\mathbb{Z}\mathcal{V}$ is often put in the rhs (i.e. added to h) so that the matrix $\mathbb{A}$ retains the same structure as in the incompressible case. This is indeed how it is implemented in ASPECT. This however requires more work since the rhs depends on the solution and some form of iterations is needed.

In the case of a compressible flow the strain rate tensor and the deviatoric strain rate tensor are no more equal (since $\boldsymbol{\nabla}\cdot\boldsymbol{v}\neq 0$). The deviatoric strainrate tensor is given by[8]

$$\dot{\boldsymbol{\epsilon}}^d(\boldsymbol{v})=\dot{\boldsymbol{\epsilon}}(\boldsymbol{v})-\frac{1}{3}Tr(\dot{\boldsymbol{\epsilon}})\mathbf{1}=\dot{\boldsymbol{\epsilon}}(\boldsymbol{v})-\frac{1}{3}(\boldsymbol{\nabla}\cdot\boldsymbol{v})\mathbf{1}$$

In that case:

$$\dot{\epsilon}^d_{xx} = \frac{\partial u}{\partial x}-\frac{1}{3}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)=\frac{2}{3}\frac{\partial u}{\partial x}-\frac{1}{3}\frac{\partial v}{\partial y} \tag{101}$$

$$\dot{\epsilon}^d_{yy} = \frac{\partial v}{\partial y}-\frac{1}{3}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)=-\frac{1}{3}\frac{\partial u}{\partial x}+\frac{2}{3}\frac{\partial v}{\partial y} \tag{102}$$

$$2\dot{\epsilon}^d_{xy} = \frac{\partial u}{\partial y}+\frac{\partial v}{\partial x} \tag{103}$$

and then

$$\dot{\boldsymbol{\epsilon}}^d(\boldsymbol{v})=\begin{pmatrix}\frac{2}{3}\frac{\partial u}{\partial x}-\frac{1}{3}\frac{\partial v}{\partial y} & \frac{1}{2}\frac{\partial u}{\partial y}+\frac{1}{2}\frac{\partial v}{\partial x}\\ \\ \frac{1}{2}\frac{\partial u}{\partial y}+\frac{1}{2}\frac{\partial v}{\partial x} & -\frac{1}{3}\frac{\partial u}{\partial x}+\frac{2}{3}\frac{\partial v}{\partial y}\end{pmatrix}$$

From $\vec{\tau}=2\eta\vec{\epsilon}^d$ we arrive at:

$$\begin{pmatrix}\tau_{xx}\\ \tau_{yy}\\ \tau_{xy}\end{pmatrix}=2\eta\begin{pmatrix}\dot{\epsilon}^d_{xx}\\ \dot{\epsilon}^d_{yy}\\ \dot{\epsilon}^d_{xy}\end{pmatrix}=2\eta\begin{pmatrix}2/3 & -1/3 & 0\\ -1/3 & 2/3 & 0\\ 0 & 0 & 1/2\end{pmatrix}\cdot\begin{pmatrix}\frac{\partial u}{\partial x}\\ \frac{\partial v}{\partial y}\\ \frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\end{pmatrix}=\eta\begin{pmatrix}4/3 & -2/3 & 0\\ -2/3 & 4/3 & 0\\ 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}\frac{\partial u}{\partial x}\\ \frac{\partial v}{\partial y}\\ \frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\end{pmatrix}$$

or,

$$\vec{\tau}=\boldsymbol{C}_\eta\boldsymbol{B}V$$

[8] See the ASPECT manual for a justification of the 3 value in the denominator in 2D and 3D.

In order to test our implementation we have created a few manufactured solutions:

- benchmark #1 (`ibench=1`)): Starting from a density profile of:

$$\rho(x,y) = xy \tag{104}$$

We derive a velocity given by:

$$v_x(x,y) = \frac{C_x}{x}, v_y(x,y) = \frac{C_y}{y} \tag{105}$$

With $g_x(x,y) = \frac{1}{x}$ and $g_y(x,y) = \frac{1}{y}$, this leads us to a pressure profile:

$$p = -\eta\left(\frac{4C_x}{3x^2} + \frac{4C_y}{3y^2}\right) + xy + C_0 \tag{106}$$

This gives us a strain rate:

$$\dot{\epsilon}_{xx} = \frac{-C_x}{x^2} \qquad \dot{\epsilon}_{yy} = \frac{-C_y}{y^2} \qquad \dot{\epsilon}_{xy} = 0$$

In what follows, we choose $\eta = 1$ and $C_x = C_y = 1$ and for a unit square domain $[1:2] \times [1:2]$ we compute C_0 so that the pressure is normalised to zero over the whole domain and obtain $C_0 = -1$.

- benchmark #2 (`ibench=2`): Starting from a density profile of:

$$\rho = \cos(x)\cos(y) \tag{107}$$

We derive a velocity given by:

$$v_x = \frac{C_x}{\cos(x)}, v_y = \frac{C_y}{\cos(y)} \tag{108}$$

With $g_x = \frac{1}{\cos(y)}$ and $g_y = \frac{1}{\cos(x)}$, this leads us to a pressure profile:

$$p = \eta\left(\frac{4C_x\sin(x)}{3\cos^2(x)} + \frac{4C_y\sin(y)}{3\cos^2(y)}\right) + (\sin(x) + \sin(y)) + C_0 \tag{109}$$

$$\dot{\epsilon}_{xx} = C_x\frac{\sin(x)}{\cos^2(x)} \qquad \dot{\epsilon}_{yy} = C_y\frac{\sin(y)}{\cos^2(y)} \qquad \dot{\epsilon}_{xy} = 0$$

We choose $\eta = 1$ and $C_x = C_y = 1$. The domain is the unit square $[0:1] \times [0:1]$ and we obtain C_0 as before and obtain

$$C_0 = 2 - 2\cos(1) + 8/3(\frac{1}{\cos(1)} - 1) \simeq 3.18823730$$

(thank you WolframAlpha)

- benchmark #3 (`ibench=3`)
- benchmark #4 (`ibench=4`)
- benchmark #5 (`ibench=5`)

features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- Dirichlet boundary conditions (no-slip)
- isothermal
- isoviscous
- analytical solution
- pressure smoothing

ToDo:

- pbs with odd vs even number of elements
- q is 'fine' everywhere except in the corners - revisit pressure smoothing paper?
- redo A v d Berg benchmark (see Tom Weir thesis)

31 `fieldstone_24`: compressible flow (2) - convection box

This work is part of the MSc thesis of T. Weir (2018).

31.1 The physics

Let us start with some thermodynamics. Every material has an equation of state. The equilibrium thermodynamic state of any material can be constrained if any two state variables are specified. Examples of state variables include the pressure p and specific volume $\nu = 1/\rho$, as well as the temperature T.

After linearisation, the density depends on temperature and pressure as follows:

$$\rho(T,p) = \rho_0 \left((1 - \alpha(T - T_0) + \beta_T p \right)$$

where α is the coefficient of thermal expansion, also called thermal expansivity:

$$\alpha = -\frac{1}{\rho} \left(\frac{\partial \rho}{\partial T} \right)_p$$

α is the percentage increase in volume of a material per degree of temperature increase; the subscript p means that the pressure is held fixed.

β_T is the isothermal compressibility of the fluid, which is given by

$$\beta_T = \frac{1}{K} = \frac{1}{\rho} \left(\frac{\partial \rho}{\partial P} \right)_T$$

with K the bulk modulus. Values of $\beta_T = 10^{-12} - 10^{-11}$ Pa^{-1} are reasonable for Earth's mantle, with values decreasing by about a factor of 5 between the shallow lithosphere and core-mantle boundary. This is the percentage increase in density per unit change in pressure at constant temperature. Both the coefficient of thermal expansion and the isothermal compressibility can be obtained from the equation of state.

The full set of equations we wish to solve is given by

$$-\nabla \cdot \left[2\eta \dot{\boldsymbol{\epsilon}}^d(\boldsymbol{v}) \right] + \nabla p = \rho_0 \left((1 - \alpha(T - T_0) + \beta_T p \right) \boldsymbol{g} \qquad \text{in } \Omega \tag{110}$$

$$\nabla \cdot \boldsymbol{v} + \frac{1}{\rho} \boldsymbol{v} \cdot \boldsymbol{\nabla} \rho = 0 \qquad \text{in } \Omega \tag{111}$$

$$\rho C_p \left(\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T \right) - \nabla \cdot k \nabla T = \rho H + 2\eta \dot{\boldsymbol{\epsilon}}^d : \dot{\boldsymbol{\epsilon}}^d + \alpha T \left(\frac{\partial p}{\partial t} + \boldsymbol{v} \cdot \nabla p \right) \qquad \text{in } \Omega, \tag{112}$$

Note that this presupposes that the density is not zero anywhere in the domain.

31.2 The numerics

We use a mixed formulation and therefore keep both velocity and pressure as unknowns. We end up having to solve the following system:

$$\begin{pmatrix} \mathbb{K} & \mathbb{G} + \mathbb{W} \\ \mathbb{G}^T + \mathbb{Z} & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathcal{V} \\ \mathcal{P} \end{pmatrix} = \begin{pmatrix} f \\ h \end{pmatrix} \qquad \text{or,} \qquad \mathbb{A} \cdot X = rhs$$

Where $\mathbb{K}$ is the stiffness matrix, $\mathbb{G}$ is the discrete gradient operator, $\mathbb{G}^T$ is the discrete divergence operator, $\mathcal{V}$ the velocity vector, $\mathcal{P}$ the pressure vector. Note that the term $\mathbb{Z}\mathcal{V}$ derives from term $\boldsymbol{v} \cdot \boldsymbol{\nabla} \rho$ in the continuity equation.

As perfectly explained in the step 32 of deal.ii[9], we need to scale the $\mathbb{G}$ term since it is many orders of magnitude smaller than $\mathbb{K}$, which introduces large inaccuracies in the solving process to the point that the solution is nonsensical. This scaling coefficient is η/L. After building the $\mathbb{G}$ block, it is then scaled as follows: $\mathbb{G}' = \frac{\eta}{L}\mathbb{G}$ so that we now solve

[9] https://www.dealii.org/9.0.0/doxygen/deal.II/step_32.html

$$\begin{pmatrix} \mathbb{K} & \mathbb{G}' + \mathbb{W} \\ \mathbb{G}'^T + \mathbb{Z} & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathcal{V} \\ \mathcal{P}' \end{pmatrix} = \begin{pmatrix} f \\ h \end{pmatrix}$$

After the solve phase, we recover the real pressure with $\mathcal{P} = \frac{\eta}{L}\mathcal{P}'$.

adapt notes since I should scale $\mathbb{W}$ and $\mathbb{Z}$ too. h should be caled too !!!!!!!!!!!!!!

Each block $\mathbb{K}$, $\mathbb{G}$, $\mathbb{Z}$ and vectors f and h are built separately in the code and assembled into the matrix $\mathbb{A}$ and vector rhs afterwards. $\mathbb{A}$ and rhs are then passed to the solver. We will see later that there are alternatives to solve this approach which do not require to build the full Stokes matrix $\mathbb{A}$.

Remark 1: the terms $\mathbb{Z}\mathcal{V}$ and $\mathbb{W}\mathcal{P}$ are often put in the rhs (i.e. added to h) so that the matrix $\mathbb{A}$ retains the same structure as in the incompressible case. This is indeed how it is implemented in ASPECT, see also appendix A of [56]. This however requires more work since the rhs depends on the solution and some form of iterations is needed.

Remark 2: Very often the adiabatic heating term $\alpha T\left(\boldsymbol{v}\cdot\nabla p\right)$ is simplified as follows: If you assume the vertical component of the gradient of the dynamic pressure to be small compared to the gradient of the total pressure (in other words, the gradient is dominated by the gradient of the hydrostatic pressure), then $-\rho\boldsymbol{g}\simeq\boldsymbol{\nabla}p$ and then $\alpha T\left(\boldsymbol{v}\cdot\nabla p\right)\simeq-\alpha\rho T\boldsymbol{v}\cdot\boldsymbol{g}$. We will however not be using this approximation in what follows.

We have already established that

$$\vec{\tau} = \boldsymbol{C}_\eta \boldsymbol{B} V$$

The following measurements are carried out:

- The root mean square velocity (`vrms`):

$$v_{rms} = \sqrt{\frac{1}{V}\int_V v^2 dV}$$

- The average temperature (`Tavrg`):

$$< T >= \frac{1}{V}\int_V T dV$$

- The total mass (`mass`):

$$M = \int_V \rho dV$$

- The Nusselt number (`Nu`):

$$Nu = -\frac{1}{Lx}\frac{1}{\Delta T}\int_0^{L_x} \frac{\partial T(x, y = L_y)}{\partial y} dx$$

- The kinetic energy (`EK`):

$$E_K = \int_V \frac{1}{2}\rho v^2 dV$$

- The work done against gravity

$$< W >= -\int_V \rho g_y v_y dV$$

- The total viscous dissipation (`visc_diss`)

$$< \Phi >= \int \Phi dV = \frac{1}{V}\int 2\eta\dot{\boldsymbol{\varepsilon}} : \dot{\boldsymbol{\varepsilon}} dV$$

- The gravitational potential energy (`EG`)

$$E_G = \int_V \rho g_y (L_y - y) dV$$

- The internal thermal energy (`ET`)

$$E_T = \int_V \rho_{(0)} C_p T dV$$

Remark 3: Measuring the total mass can be misleading: indeed because $\rho = \rho_0(1 - \alpha T)$, then measuring the total mass amounts to measuring a constant minus the volume-integrated temperature, and there is no reason why the latter should be zero, so that there is no reason why the total mass should be zero...!

31.3 The experimental setup

The setup is as follows: the domain is $Lx = Ly = 3000$km. Free slip boundary conditions are imposed on all four sides. The initial temperature is given by:

$$T(x,y) = \left(\frac{L_y - y}{Ly} - 0.01 \cos(\frac{\pi x}{L_x}) \sin(\frac{\pi y}{Ly}) \right) \Delta T + T_{surf}$$

with $\Delta T = 4000$K, $T_{surf} = T_0 = 273.15$K. The temperature is set to $\Delta T + T_{surf}$ at the bottom and T_{surf} at the top. We also set $k = 3$, $C_p = 1250$, $|g| = 10$, $\rho_0 = 3000$ and we keep the Rayleigh number Ra and dissipation number Di as input parameters:

$$Ra = \frac{\alpha g \Delta T L^3 \rho_0^2 C_p}{\eta k} \qquad Di = \frac{\alpha g L}{C_p}$$

From the second equation we get $\alpha = \frac{Di C_p}{gL}$, which we can insert in the first one:

$$Ra = \frac{Di C_p^2 \Delta T L^2 \rho_0^2}{\eta k} \qquad \text{or,} \qquad \eta = \frac{Di C_p^2 \Delta T L^2 \rho_0^2}{Ra\ k}$$

For instance, for $Ra = 10^4$ and $Di = 0.75$, we obtain $\alpha \simeq 3 \cdot 10^{-5}$ and $\eta \simeq 10^{25}$ which are quite reasonable values.

31.4 Scaling

Following [52], we non-dimensionalize the equations using the reference values for density ρ_r, thermal expansivity α_r, temperature contrast ΔT_r (`refTemp`), thermal conductivity k_r, heat capacity C_p, depth of the fluid layer L and viscosity η_r. The non-dimensionalization for velocity, u_r , pressure p_r and time, t_r become

$$u_r = \frac{k_r}{\rho_r C_p L} \quad (\texttt{refvel})$$

$$p_r = \frac{\eta_r k_r}{\rho_r C_p L^2} \quad (\texttt{refpress})$$

$$t_r = \frac{\rho_r C_p L^2}{k_r} \quad (\texttt{reftime})$$

In the case of the setup described hereabove, and when choosing $Ra = 10^4$ and $Di = 0.5$, we get:

```
alphaT 2.083333e-05
eta 8.437500e+24
reftime 1.125000e+19
refvel 2.666667e-13
refPress 7.500000e+05
```

31.5 Conservation of energy 1

31.5.1 under BA and EBA approximations

Following [56], we take the dot product of the momentum equation with the velocity $\boldsymbol{v}$ and integrate over the whole volume[10]:

$$\int_V [-\nabla \cdot \boldsymbol{\tau} + \boldsymbol{\nabla} p] \cdot \boldsymbol{v} dV = \int_V \rho \boldsymbol{g} \cdot \boldsymbol{v} dV$$

or,

$$-\int_V (\nabla \cdot \boldsymbol{\tau}) \cdot \boldsymbol{v} dV + \int_V \boldsymbol{\nabla} p \cdot \boldsymbol{v} dV = \int_V \rho \boldsymbol{g} \cdot \boldsymbol{v} dV$$

Let us look at each block separately:

$$-\int_V (\nabla \cdot \boldsymbol{\tau}) \cdot \boldsymbol{v} dV = -\int_S \boldsymbol{\tau} \underbrace{\boldsymbol{v} \cdot \boldsymbol{n}}_{=0\ (b.c.)} dS + \int_V \boldsymbol{\tau} : \boldsymbol{\nabla} \boldsymbol{v} dV = \int_V \boldsymbol{\tau} : \dot{\boldsymbol{\varepsilon}} dV = \int_V \Phi dV$$

which is the volume integral of the shear heating. Then,

$$\int_V \boldsymbol{\nabla} p \cdot \boldsymbol{v} dV = \int_S p \underbrace{\boldsymbol{v} \cdot \boldsymbol{n}}_{=0\ (b.c.)} dS - \int_V \underbrace{\boldsymbol{\nabla} \cdot \boldsymbol{v}}_{=0\ (incomp.)} p dV = 0$$

which is then zero in the case of an incompressible flow. And finally

$$\int_V \rho \boldsymbol{g} \cdot \boldsymbol{v} dV = W$$

which is the work against gravity.

Conclusion for an *incompressible* fluid: we should have

$$\int_V \Phi dV = \int_V \rho \boldsymbol{g} \cdot \boldsymbol{v} dV \tag{113}$$

This formula is hugely problematic: indeed, the term ρ in the rhs is the full density. We know that to the value of ρ_0 corresponds a lithostatic pressure gradient $p_L = \rho_0 g y$. In this case one can write $\rho = \rho_0 + \rho'$ and $p = p_L + p'$ so that we also have

$$\int_V [-\nabla \cdot \boldsymbol{\tau} + \boldsymbol{\nabla} p'] \cdot \boldsymbol{v} dV = \int_V \rho' \boldsymbol{g} \cdot \boldsymbol{v} dV$$

which will ultimately yield

$$\int_V \Phi dV = \int_V \rho' \boldsymbol{g} \cdot \boldsymbol{v} dV = \int_V (\rho - \rho_0) \boldsymbol{g} \cdot \boldsymbol{v} dV \tag{114}$$

Obviously Eqs.(113) and (114) cannot be true at the same time. The problem comes from the nature of the (E)BA approximation: $\rho = \rho_0$ in the mass conservation equation but it is not constant in the momentum conservation equation, which is of course inconsistent. Since the mass conservation equation is $\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0$ under this approximation then the term $\int_V \boldsymbol{\nabla} p \cdot \boldsymbol{v} dV$ is always zero for any pressure (full pressure p, or overpressure $p - p_L$), hence the paradox. This paradox will be lifted when a consistent set of equations will be used (compressible formulation). On a practical note, Eqs.(113) is not verified by the code, while (114) is.

In the end:

$$\boxed{\underbrace{\int_V \Phi dV}_{\texttt{visc\_diss}} = \underbrace{\int_V (\rho - \rho_0) \boldsymbol{g} \cdot \boldsymbol{v} dV}_{\texttt{work\_grav}}} \tag{115}$$

[10] Check: this is akin to looking at the power, force*velocity, says Arie

31.5.2 under no approximation at all

$$\int_V \boldsymbol{\nabla} p \cdot \boldsymbol{v} dV = \int_S p \underbrace{\boldsymbol{v} \cdot \boldsymbol{n}}_{=0\ (b.c.)} dS - \int_V \boldsymbol{\nabla} \cdot \boldsymbol{v}\ p dV = 0 \tag{116}$$

$$= \int_V \frac{1}{\rho} \boldsymbol{v} \cdot \boldsymbol{\nabla} \rho\ p dV = 0 \tag{117}$$

$$\tag{118}$$

ToDo:see section 3 of [56] where this is carried out with the Adams-Williamson eos.

31.6 Conservation of energy 2

Also, following the Reynold's transport theorem [62],p210, we have for a property A (per unit mass)

$$\frac{d}{dt} \int_V A \rho dV = \int_V \frac{\partial}{\partial t} (A\rho) dV + \int_S A \rho \boldsymbol{v} \cdot \boldsymbol{n} dS$$

Let us apply to this to $A = C_p T$ and compute the time derivative of the internal energy:

$$\frac{d}{dt} \int_V \rho C_p T dV = \int_V \frac{\partial}{\partial t} (\rho C_p T) dV + \int_S A\rho \underbrace{\boldsymbol{v} \cdot \boldsymbol{n}}_{=0\ (b.c.)} dS = \underbrace{\int_V C_p T \frac{\partial \rho}{\partial t} dV}_{I} + \underbrace{\int_V \rho C_p \frac{\partial T}{\partial t} dV}_{II} \tag{119}$$

In order to expand I, the mass conservation equation will be used, while the heat transport equation will be used for II:

$$I = \int_V C_p T \frac{\partial \rho}{\partial t} dV = -\int_V C_p T \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) dV = -\int_V C_p T \rho \underbrace{\boldsymbol{v} \cdot \boldsymbol{n}}_{=0\ (b.c.)} dS + \int_V \rho C_p \boldsymbol{\nabla} T \cdot \boldsymbol{v} dV \tag{120}$$

$$II = \int_V \rho C_p \frac{\partial T}{\partial t} dV = \int_V \left[-\rho C_p \boldsymbol{v} \cdot \boldsymbol{\nabla} T + \boldsymbol{\nabla} \cdot k \boldsymbol{\nabla} T + \rho H + \Phi + \alpha T \left(\frac{\partial p}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} p \right) \right] dV \tag{121}$$

$$= \int_V \left[-\rho C_p \boldsymbol{v} \cdot \boldsymbol{\nabla} T + \rho H + \Phi + \alpha T \left(\frac{\partial p}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} p \right) \right] dV + \int_V \boldsymbol{\nabla} \cdot k \boldsymbol{\nabla} T dV \tag{122}$$

$$= \int_V \left[-\rho C_p \boldsymbol{v} \cdot \boldsymbol{\nabla} T + \rho H + \Phi + \alpha T \left(\frac{\partial p}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} p \right) \right] dV + \int_S k \boldsymbol{\nabla} T \cdot \boldsymbol{n} dS \tag{123}$$

$$= \int_V \left[-\rho C_p \boldsymbol{v} \cdot \boldsymbol{\nabla} T + \rho H + \Phi + \alpha T \left(\frac{\partial p}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} p \right) \right] dV - \int_S \boldsymbol{q} \cdot \boldsymbol{n} dS \tag{124}$$

Finally:

$$I + II = \underbrace{\frac{d}{dt} \int_V \rho C_p T dV}_{\texttt{ET}} = \int_V \left[\rho H + \Phi + \alpha T \left(\frac{\partial p}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} p \right) \right] dV - \int_S \boldsymbol{q} \cdot \boldsymbol{n} dS \tag{125}$$

$$= \int_V \rho H dV + \underbrace{\int_V \Phi dV}_{\texttt{visc\_diss}} + \underbrace{\int_V \alpha T \frac{\partial p}{\partial t} dV}_{\texttt{extra}} + \underbrace{\int_V \alpha T \boldsymbol{v} \cdot \boldsymbol{\nabla} p dV}_{\texttt{adiab\_heating}} - \underbrace{\int_S \boldsymbol{q} \cdot \boldsymbol{n} dS}_{\texttt{heatflux\_boundary}} \tag{126}$$

This was of course needlessly complicated as the term $\partial \rho / \partial t$ is always taken to be zero, so that $I = 0$ automatically. The mass conservation equation is then simply $\boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0$. Then it follows that

$$0 = \int_V C_p T \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) dV = -\int_V C_p T \rho \underbrace{\boldsymbol{v} \cdot \boldsymbol{n}}_{=0\ (b.c.)} dS + \int_V \rho C_p \boldsymbol{\nabla} T \cdot \boldsymbol{v} dV \tag{127}$$

$$= \int_V \rho C_p \boldsymbol{\nabla} T \cdot \boldsymbol{v} dV \tag{128}$$

so that the same term in Eq.(124) vanishes too, and then Eq.(126) is always valid, although one should be careful when computing E_T in the BA and EBA cases as it should use ρ_0 and not ρ.

31.7 The problem of the onset of convection

[wiki] In geophysics, the Rayleigh number is of fundamental importance: it indicates the presence and strength of convection within a fluid body such as the Earth's mantle. The mantle is a solid that behaves as a fluid over geological time scales.

The Rayleigh number essentially is an indicator of the type of heat transport mechanism. At low Rayleigh numbers conduction processes dominate over convection ones. At high Rayleigh numbers it is the other way around. There is a so-called critical value of the number with delineates the transition from one regime to the other.

This problem has been studied and approached both theoretically and numerically [95, e.g.] and it was found that the critical Rayleigh number Ra_c is

$$Ra_c = (27/4)\pi^4 \simeq 657.5$$

in setups similar to ours.

VERY BIG PROBLEM

The temperature setup is built as follows: T_{surf} is prescribed at the top, $T_{surf} + \Delta T$ is prescribed at the bottom. The initial temperature profile is linear between these two values. In the case of BA, the actual value of T_{surf} is of no consequence. However, for the EBA the full temperature is present in the adiabatic heating term on the rhs of the hte, and the value of T_{surf} will therefore influence the solution greatly. This is very problematic as there is no real way to arrive at the surface temperature from the King paper. On top of this, the density uses a reference temperature T_0 which too will influence the solution without being present in the controlling Ra and Di numbers!!

In light thereof, it will be very difficult to recover the values of King et al for EBA!

features

- $Q_1 \times P_0$ element
- compressible flow
- mixed formulation
- Dirichlet boundary conditions (no-slip)
- isoviscous
- analytical solution
- pressure smoothing

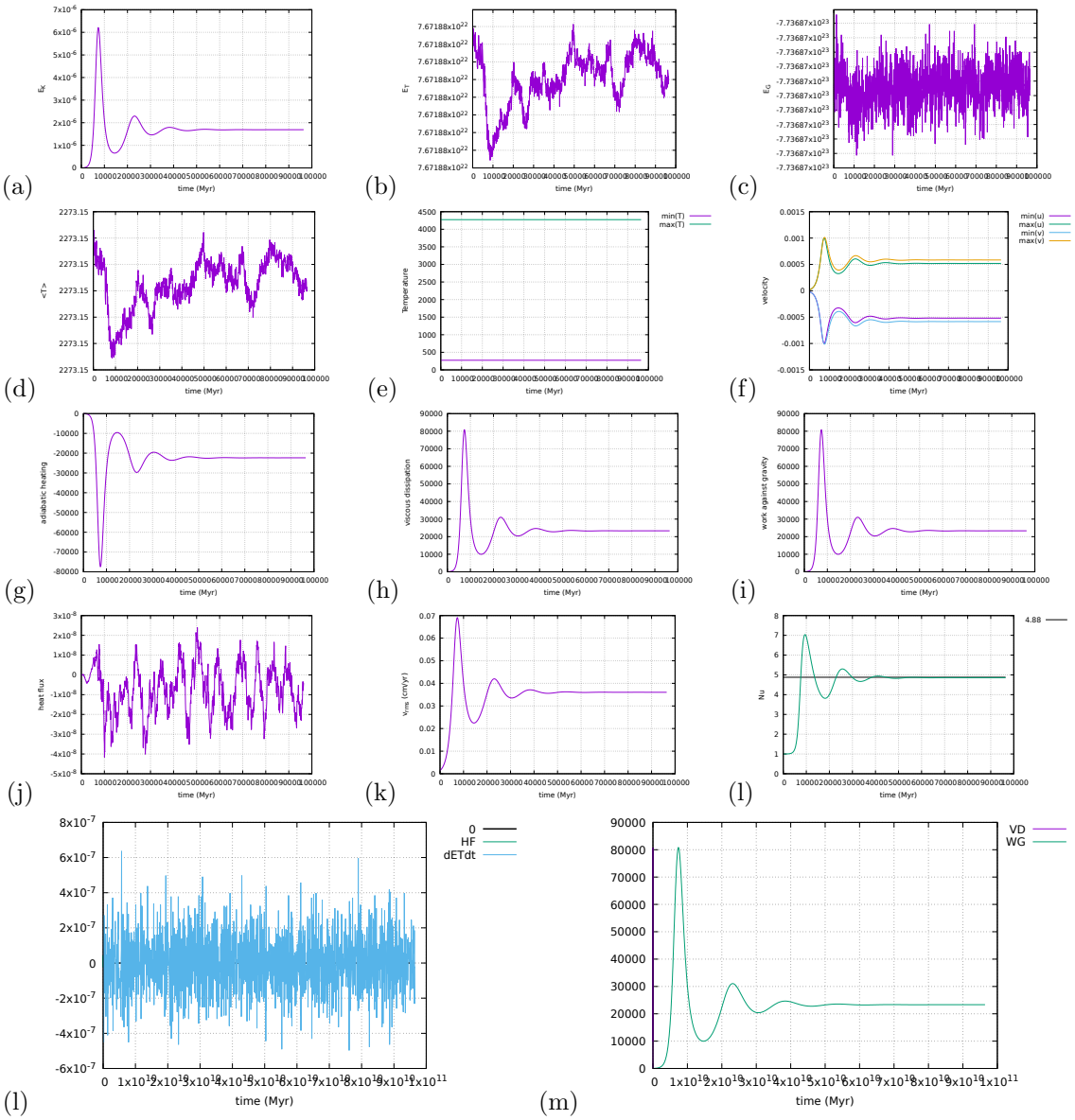
Relevant literature: [8, 48, 87, 56, 52, 57, 59, 43]
ToDo:

- heat flux is at the moment elemental, so Nusselt and heat flux on boundaries measurements not as accurate as could be.

- implement steady state detection
- do $Ra = 10^5$ and $Ra = 10^6$
- velocity average at surface
- non dimensional heat flux at corners [9]
- depth-dependent viscosity (case 2 of [9])

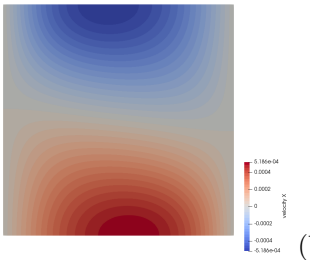
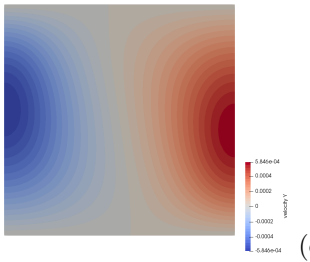
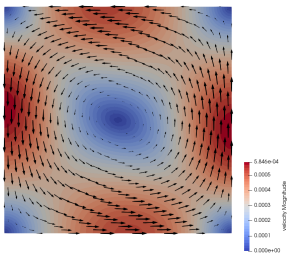
31.8 results - BA - $Ra = 10^4$

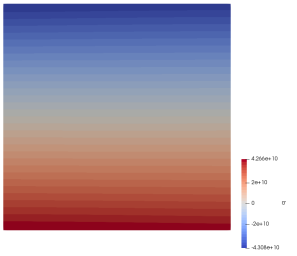
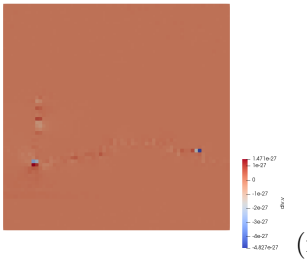
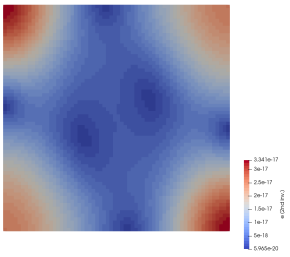
These results were obtained with a 64x64 resolution, and CFL number of 1. Steady state was reached after about 1250 timesteps.

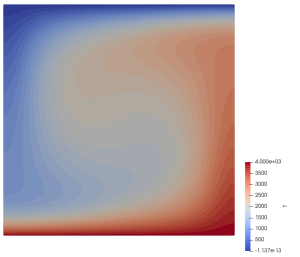
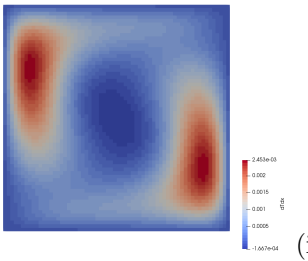
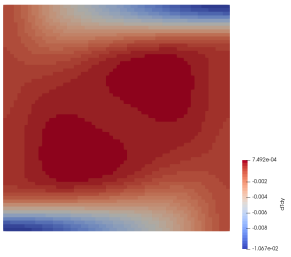


AH: adiabatic heating, VD: viscous dissipation, HF: heat flux, WG: work against gravity

Eq.(126) is verified by (l) and Eq.(115) is verified by (m).

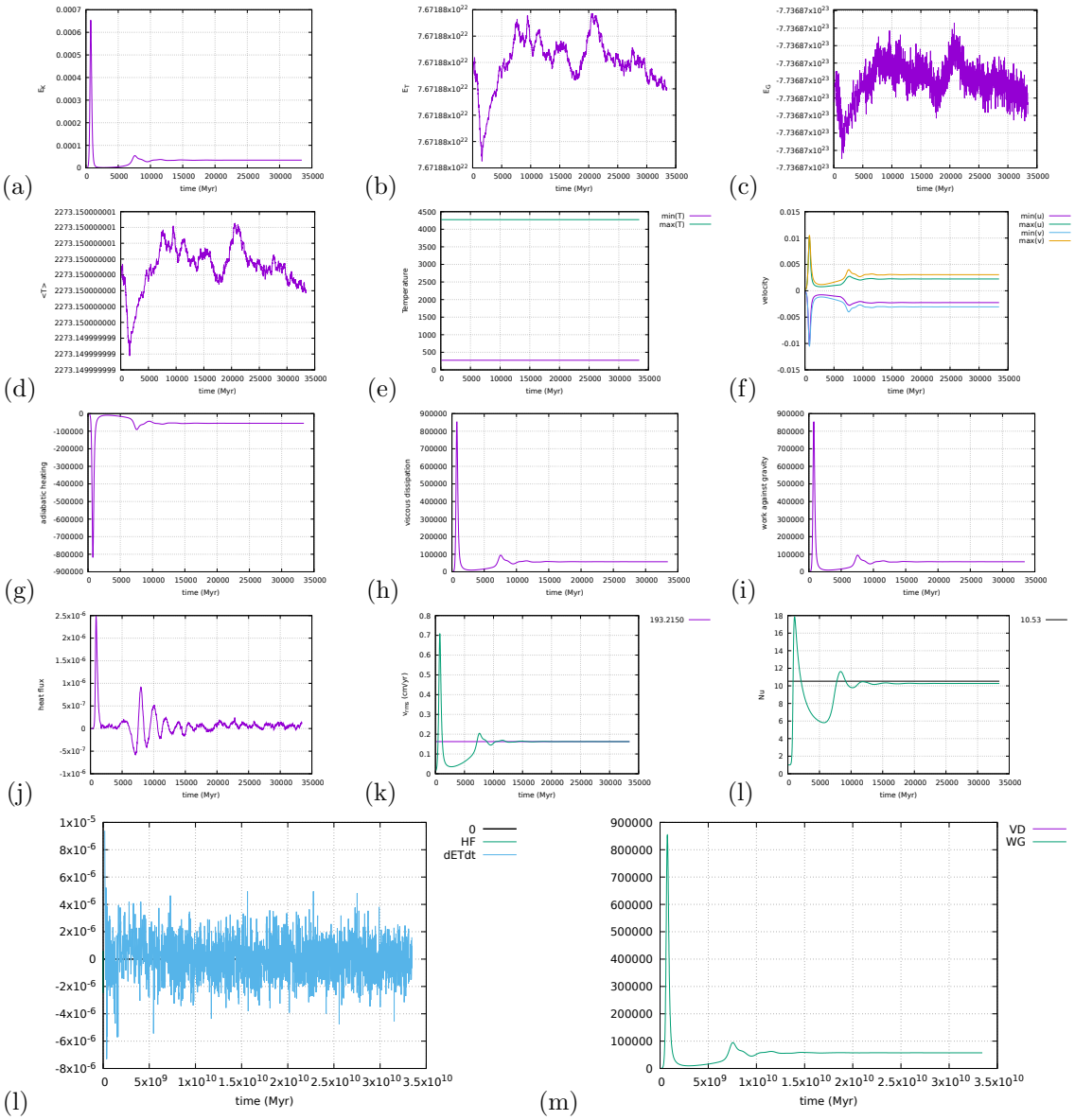
(a)  (b)  (c) 

(d)  (e)  (f) 

(g)  (h)  (i) 

31.9 results - BA - $Ra = 10^5$

These results were obtained with a 64x64 resolution, and CFL number of 1. Steady state was reached after about 1250 timesteps.



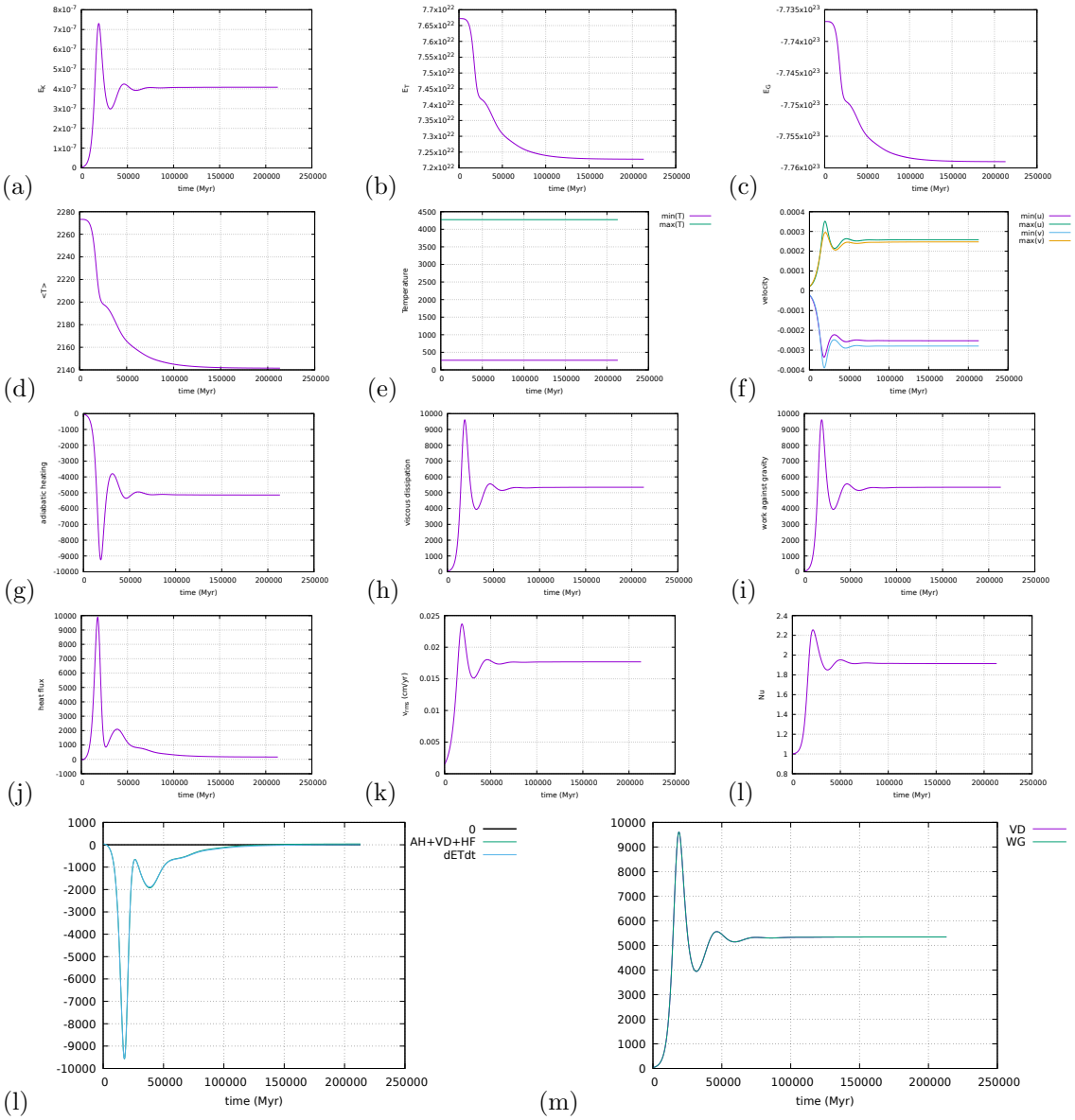
AH: adiabatic heating, VD: viscous dissipation, HF: heat flux, WG: work against gravity

Eq.(126) is verified by (l) and Eq.(115) is verified by (m).

31.10 results - BA - $Ra = 10^6$

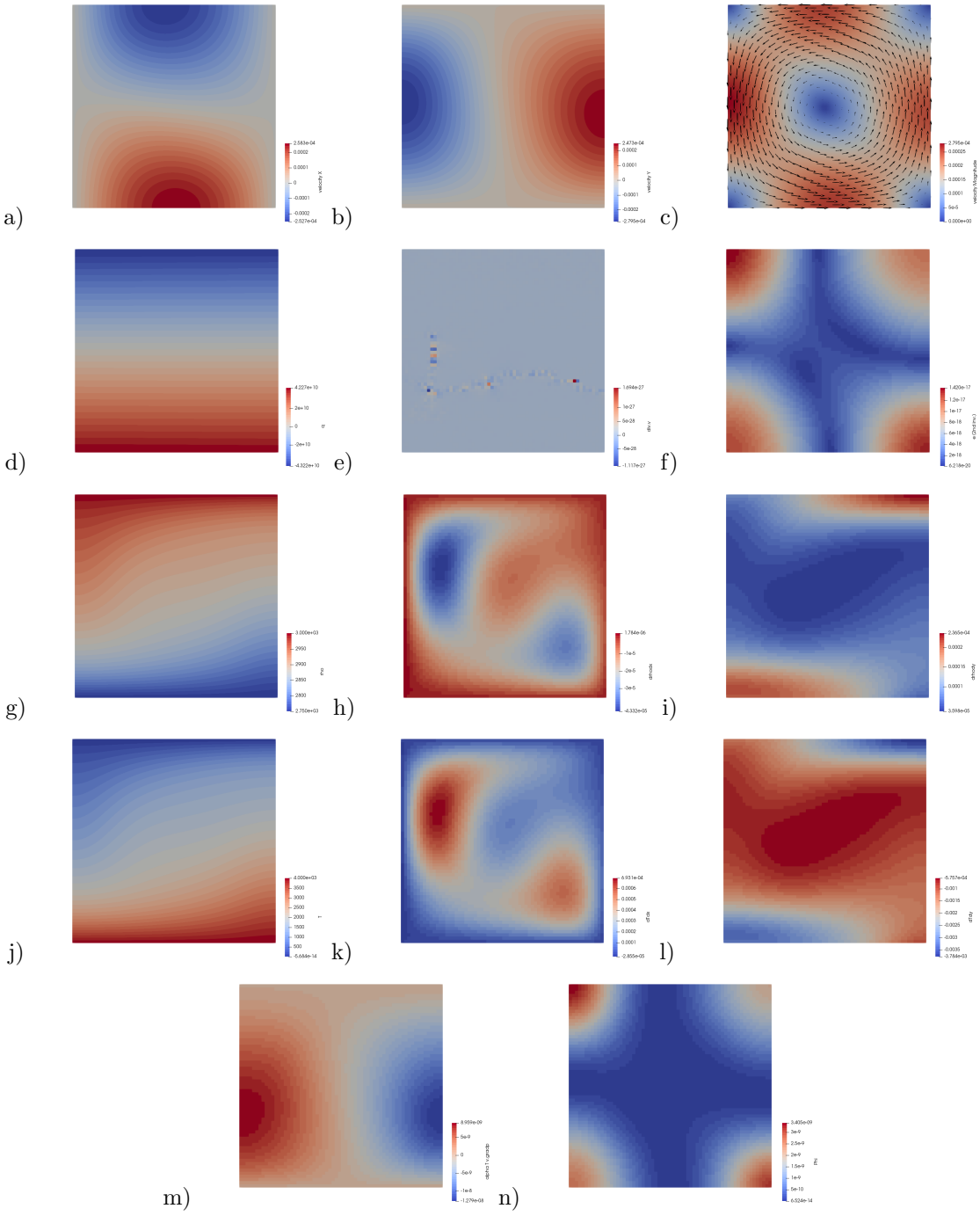
31.11 results - EBA - $Ra = 10^4$

These results were obtained with a 64x64 resolution, and CFL number of 1. Steady state was reached after about 2500 timesteps



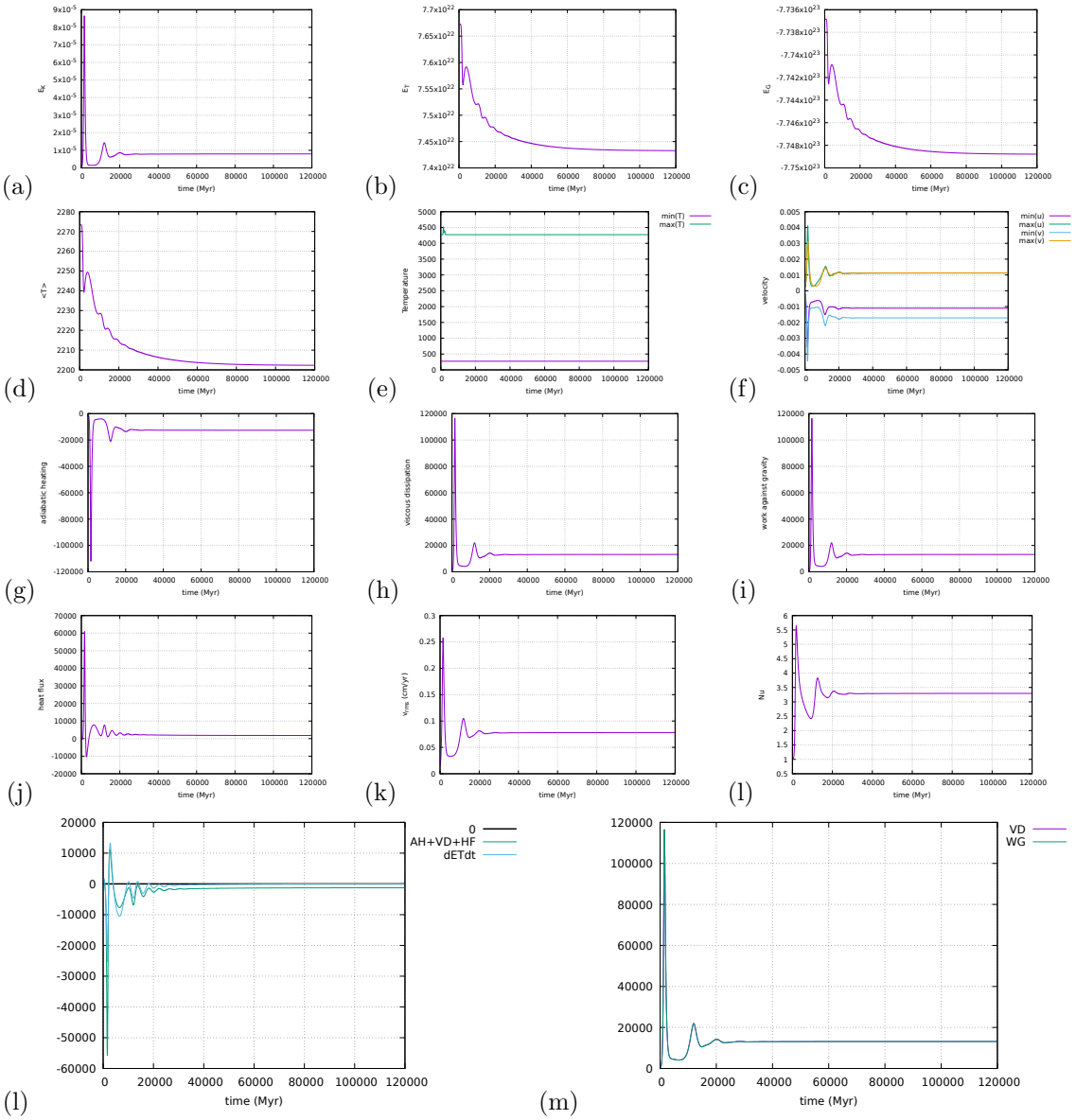
AH: adiabatic heating, VD: viscous dissipation, HF: heat flux, WG: work against gravity

Eq.(126) is verified by (l) and Eq.(115) is verified by (m).


a)
b)
c)
d)
e)
f)
g)
h)
i)
j)
k)
l)
m)
n)

31.12 results - EBA - $Ra = 10^5$

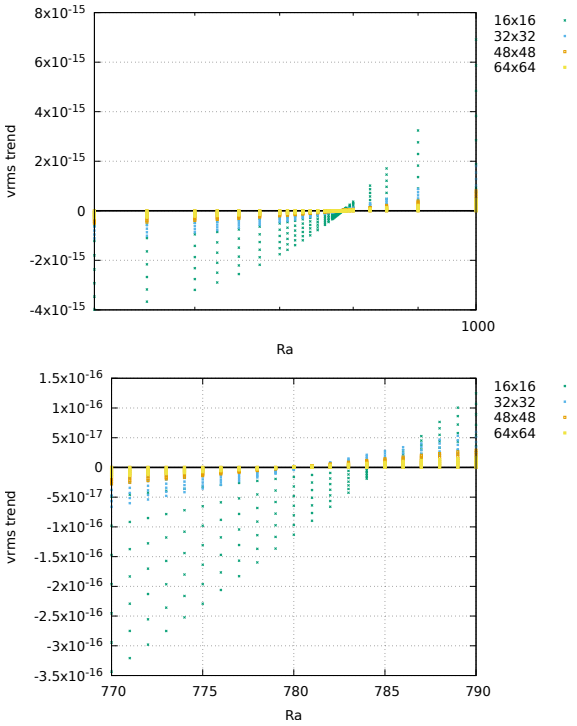
These results were obtained with a 64x64 resolution, and CFL number of 1. Simulation was stopped after about 4300 timesteps.



AH: adiabatic heating, VD: viscous dissipation, HF: heat flux, WG: work against gravity

31.13 Onset of convection

The code can be run for values of Ra between 500 and 1000, at various resolutions for the BA formulation. The value $v_{rms}(t) - v_{rms}(0)$ is plotted as a function of Ra and for the 10 first timesteps. If the v_{rms} is found to decrease, then the Rayleigh number is not high enough to allow for convection and the initial temperature perturbation relaxes by diffusion (and then $v_{rms}(t) - v_{rms}(0) < 0$. If the v_{rms} is found to increase, then $v_{rms}(t) - v_{rms}(0) > 0$ and the system is going to showcase convection. The zero value of $v_{rms}(t) - v_{rms}(0)$ gives us the critical Rayleigh number, which is found between 775 and 790.

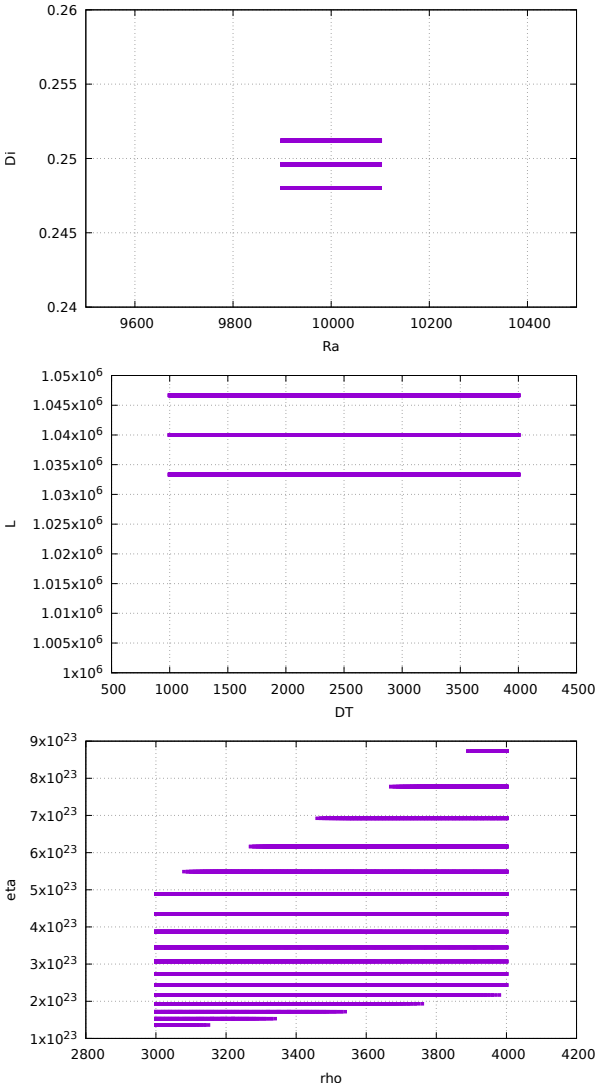


Appendix: Looking for the right combination of parameters for the King benchmark.

I run a quadruple do loop over L, ΔT, ρ_0 and η_0 between plausible values (see code targets.py) and write in a file only the combination which yields the required Rayleigh and Dissipation number values (down to 1% accuracy).

```
alpha=3e-5
g=10
hcapa=1250
hcond=3
DTmin=1000   ; DTmax=4000   ; DTnpts=251
Lmin=1e6     ; Lmax=3e6     ; Lnpts=251
rhomin=3000  ; rhomax=3500  ; rhonpts=41
etamin=19    ; etamax=25    ; etanpts=100
```

On the following plots the 'winning' combinations of these four parameters are shown:



We see that:

- the parameter L (being to the 3rd power in the Ra number) cannot vary too much. Although it is varied between 1000 and 3000km there seems to be a 'right' value at about 1040 km. (why?)
- viscosities are within 10^{23} and 10^{24} which are plausible values (although a bit high?).
- densities can be chosen freely between 3000 and 3500
- ΔT seems to be the most problematic value since it can range from 1000 to 4000K ...

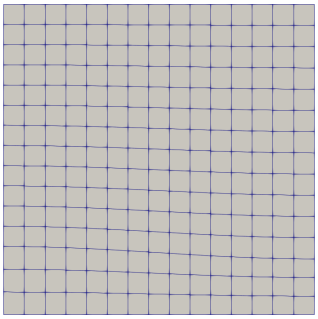
32 `fieldstone_25`: Rayleigh-Taylor instability (1) - instantaneous

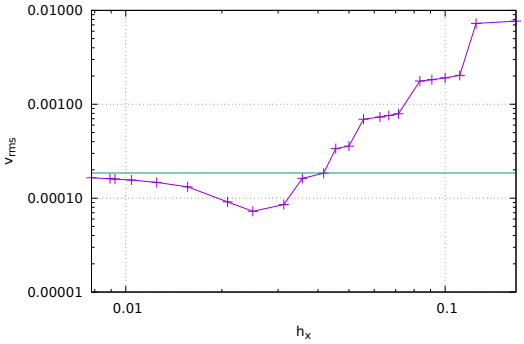
This numerical experiment was first presented in [96]. It consists of an isothermal Rayleigh-Taylor instability in a two-dimensional box of size $Lx = 0.9142$ and $L_y = 1$. Two Newtonian fluids are present in the system: the buoyant layer is placed at the bottom of the box and the interface between both fluids is given by $y(x) = 0.2 + 0.02 \cos\left(\frac{\pi x}{L_x}\right)$ The bottom fluid is parametrised by its mass density ρ_1 and its viscosity μ_1, while the layer above is parametrised by ρ_2 and μ_2.

No-slip boundary conditions are applied at the bottom and at the top of the box while free-slip boundary conditions are applied on the sides.

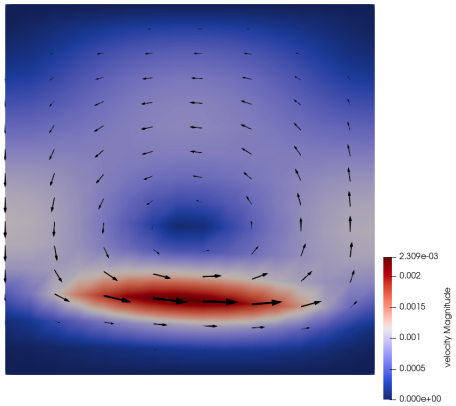
In the original benchmark the system is run over 2000 units of dimensionless time and the timing and position of various upwellings/downwellings is monitored. In this present experiment only the root mean square velocity is measured at $t = 0$: the code is indeed not yet foreseen of any algorithm capable of tracking deformation.

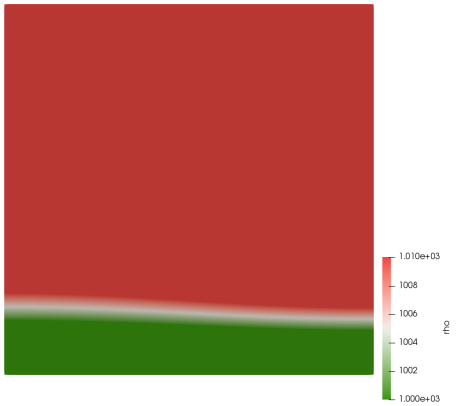
Another approach than the ones presented in the extensive literature which showcases results of this benchmark is taken. The mesh is initially fitted to the fluids interface and the resolution is progressively increased. This results in the following figure:

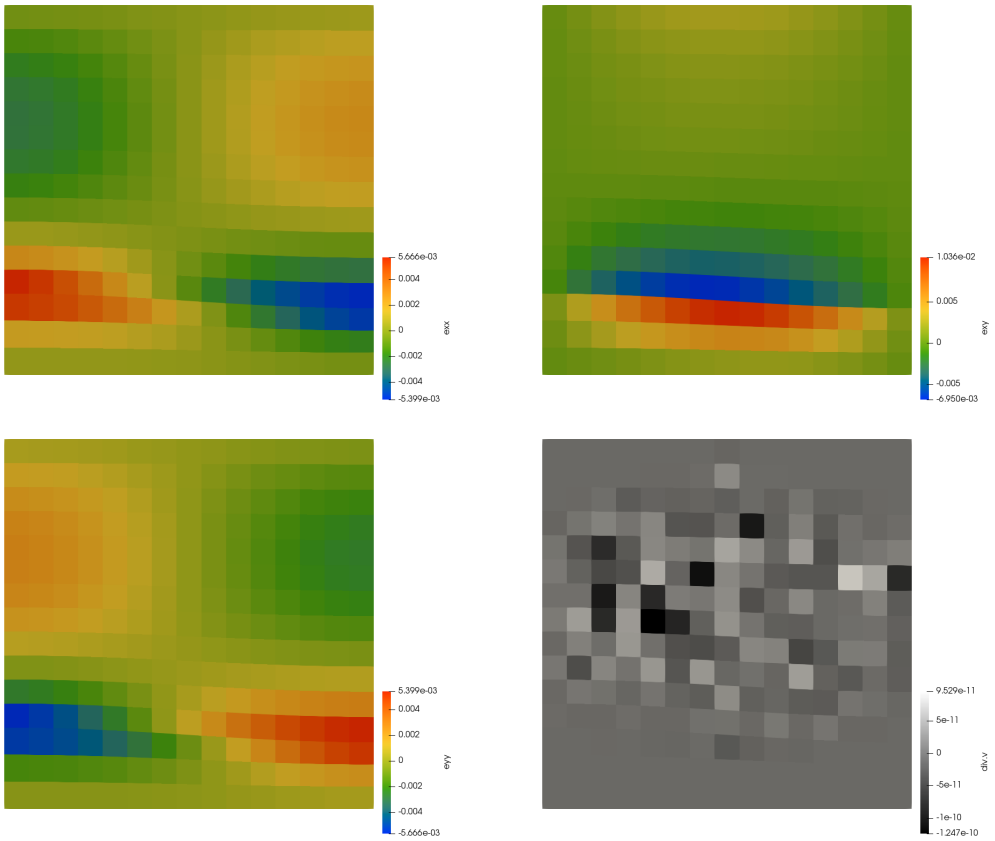




The green line indicates results obtained with my code ELEFANT with grids up to 2000x2000 with the exact same methodology.







features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- isothermal
- numerical benchmark

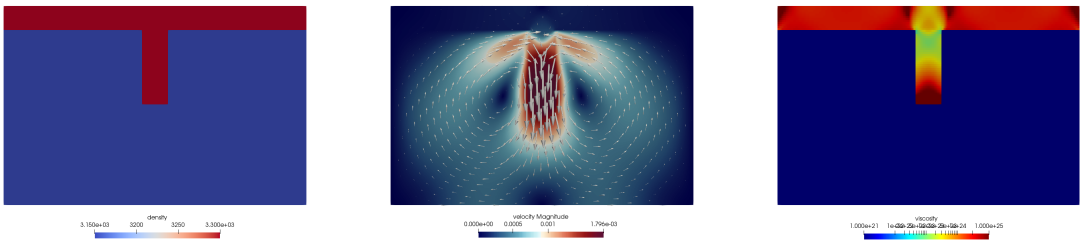
33 `fieldstone_26`: Slab detachment benchmark (1) - instantaneous

As in [79], the computational domain is $1000km \times 660km$. No-slip boundary conditions are imposed on the sides of the system while free-slip boundary conditions are imposed at the top and bottom. Two materials are present in the domain: the lithosphere (mat.1) and the mantle (mat.2). The overriding plate (mat.1) is $80km$ thick and is placed at the top of the domain. An already subducted slab (mat.1) of $250km$ length hangs vertically under this plate. The mantle occupies the rest of the domain.

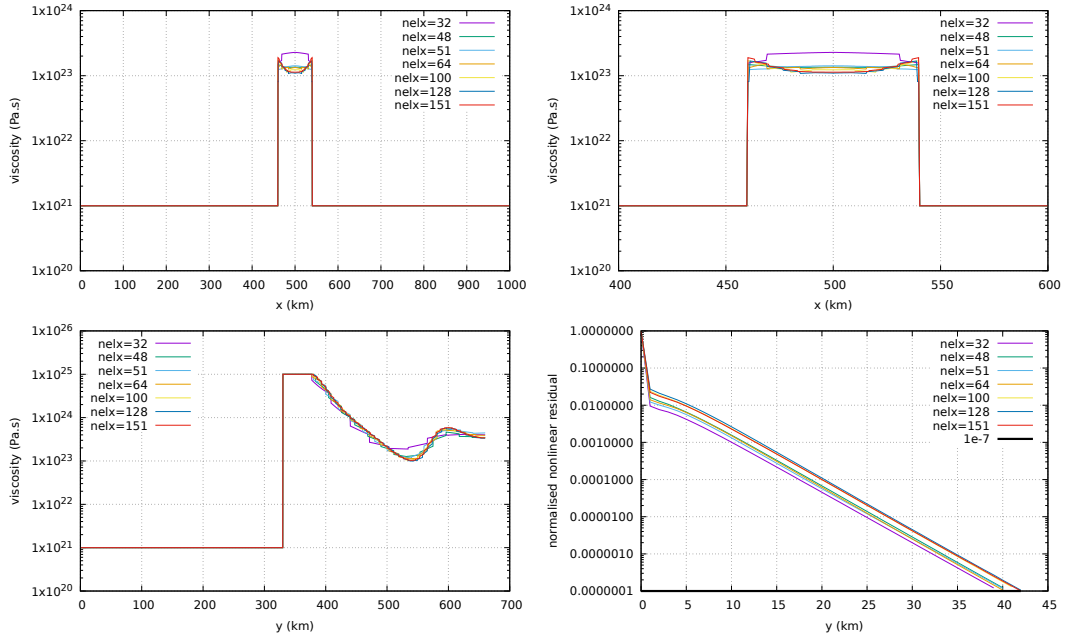
The mantle has a constant viscosity $\eta_0 = 10^{21} Pa.s$ and a density $\rho = 3150 kg/m^3$. The slab has a density $\rho = 3300 kg/m^3$ and is characterised by a power-law flow law so that its effective viscosity depends on the second invariant of the strainrate I_2 as follows:

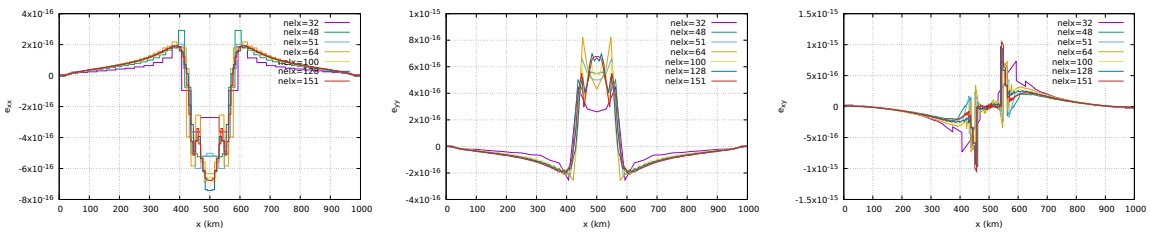
$$\eta_{eff} = \frac{1}{2} A^{-1/n_s} I_2^{1/n_s - 1} = \frac{1}{2}[(2 \times 4.75 \times 10^{11})^{-n_s}]^{-1/n_s} I_2^{1/n_s - 1} = 4.75 \times 10^{11} I_2^{1/n_s - 1} = \eta_0 I_2^{1/n_s - 1} \quad (129)$$

with $n_s = 4$ and $A = (2 \times 4.75 \times 10^{11})^{-n_s}$, or $\eta_0 = 4.75 \times 10^{11}$.

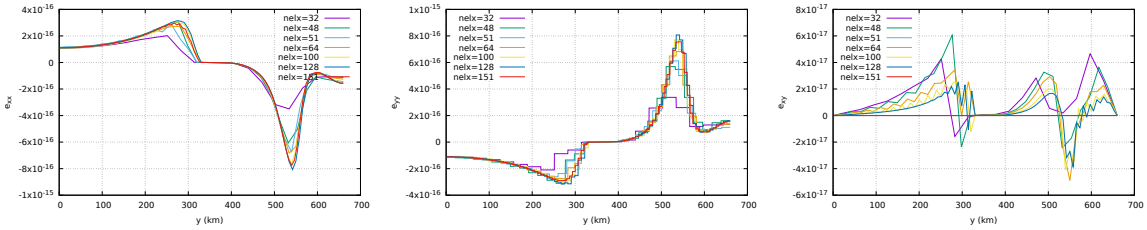


Fields at convergence for 151x99 grid.





Along the horizontal line



Along the vertical line

features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- isothermal
- nonlinear rheology
- nonlinear residual

Todo: nonlinear mantle, pressure normalisation

34 `fieldstone_27`: Consistent Boundary Flux

In what follows we will be re-doing the numerical experiments presented in Zhong et al. [103].

The first benchmark showcases a unit square domain with free slip boundary conditions prescribed on all sides. The resolution is fixed to 64×64 $Q_1 \times P_0$ elements. The flow is isoviscous and the buoyancy force $\boldsymbol{f}$ is given by

$$\begin{aligned} f_x &= 0 \\ f_y &= \rho_0 \alpha T(x,y) \end{aligned}$$

with the temperature field given by

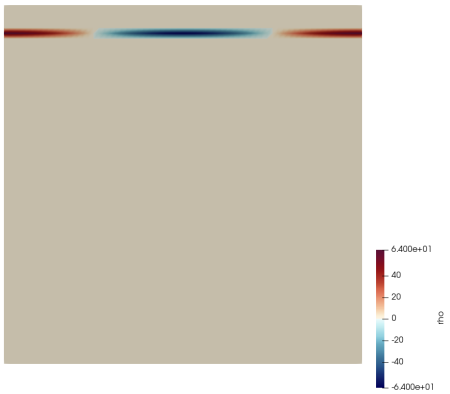
$$T(x,y) = \cos(kx)\delta(y - y_0)$$

where $k = 2\pi/\lambda$ and λ is a wavelength, and y_0 represents the location of the buoyancy strip. We set $g_y = -1$ and prescribe $\rho(x,y) = \rho_0 \alpha \cos(kx)\delta(y - y_0)$ on the nodes of the mesh.

One can prove ([103] and refs. therein) that there is an analytic solution for the surface stress σ_{zz} [11]

$$\frac{\sigma_{yy}}{\rho\alpha g h} = \frac{\cos(kx)}{\sinh^2(k)} \left[k(1-y_0)\sinh(k)\cosh(ky_0) - k\sinh(k(1-y_0)) + \sinh(k)\sinh(ky_0) \right]$$

We choose $\rho_0\alpha = 64$, $\eta = 1$ (note that in this case the normalising coefficient of the stress is exactly 1 (since $h = L_x/nelx = 1/64$) so it is not implemented in the code). $\lambda = 1$ is set to 1 and we explore $y_0 = \frac{63}{64}, \frac{62}{64}, \frac{59}{64}$ and $y_0 = 32/64$. Under these assumptions the density field for $y_0 = 59/64$ is:

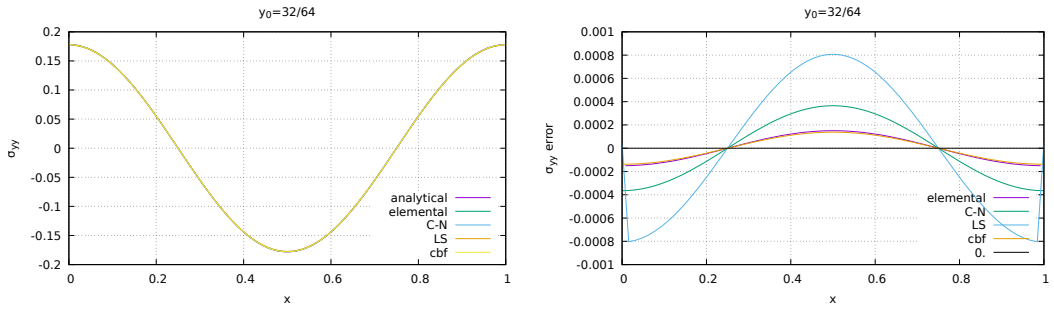


We can recover the stress at the boundary by computing the yy component of the stress tensor in the top row of elements:

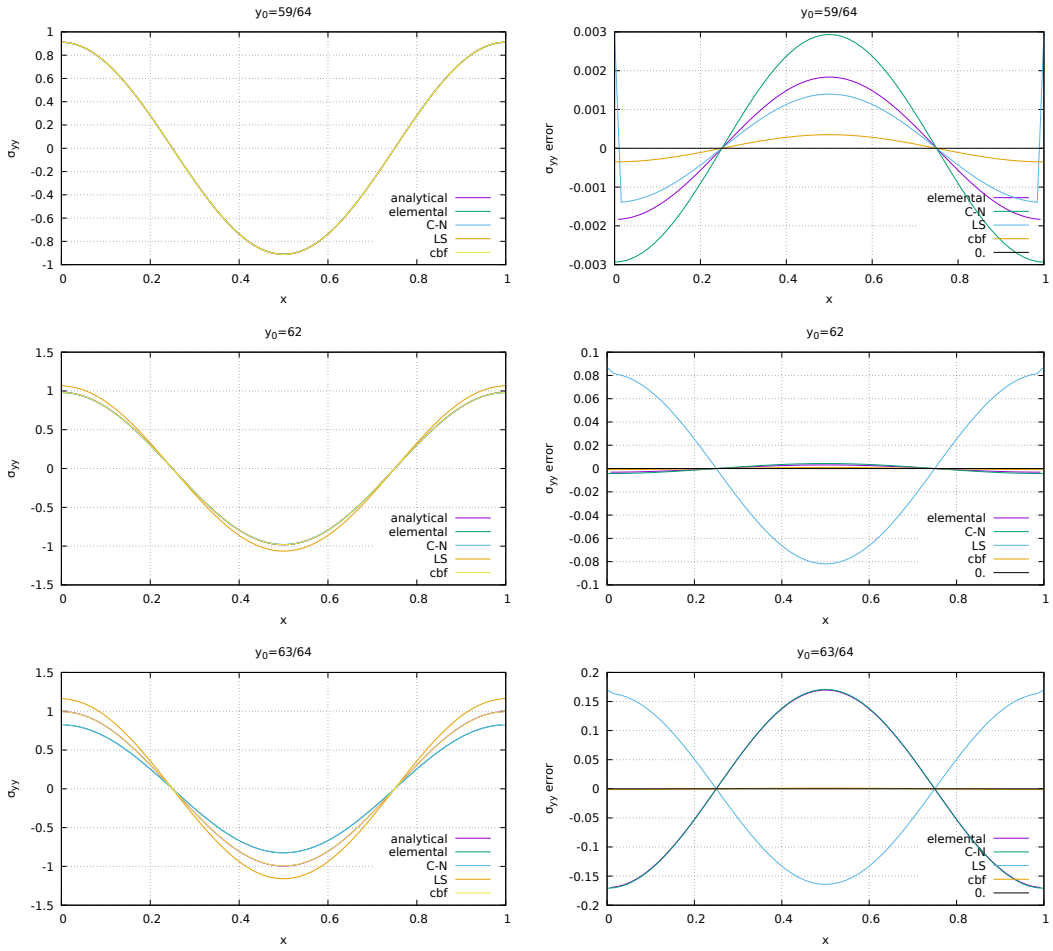
$$\sigma_{yy} = -p + 2\eta\dot{\epsilon}_{yy}$$

Note that pressure is by definition elemental, and that strain rate components are then also computed in the middle of each element.

These elemental quantities can be projected onto the nodes (see section **??**) by means of the C→N algorithm or a least square algorithm (LS).



[11] Note that in the paper the authors use $\rho\alpha g$ which does not have the dimensions of a stress



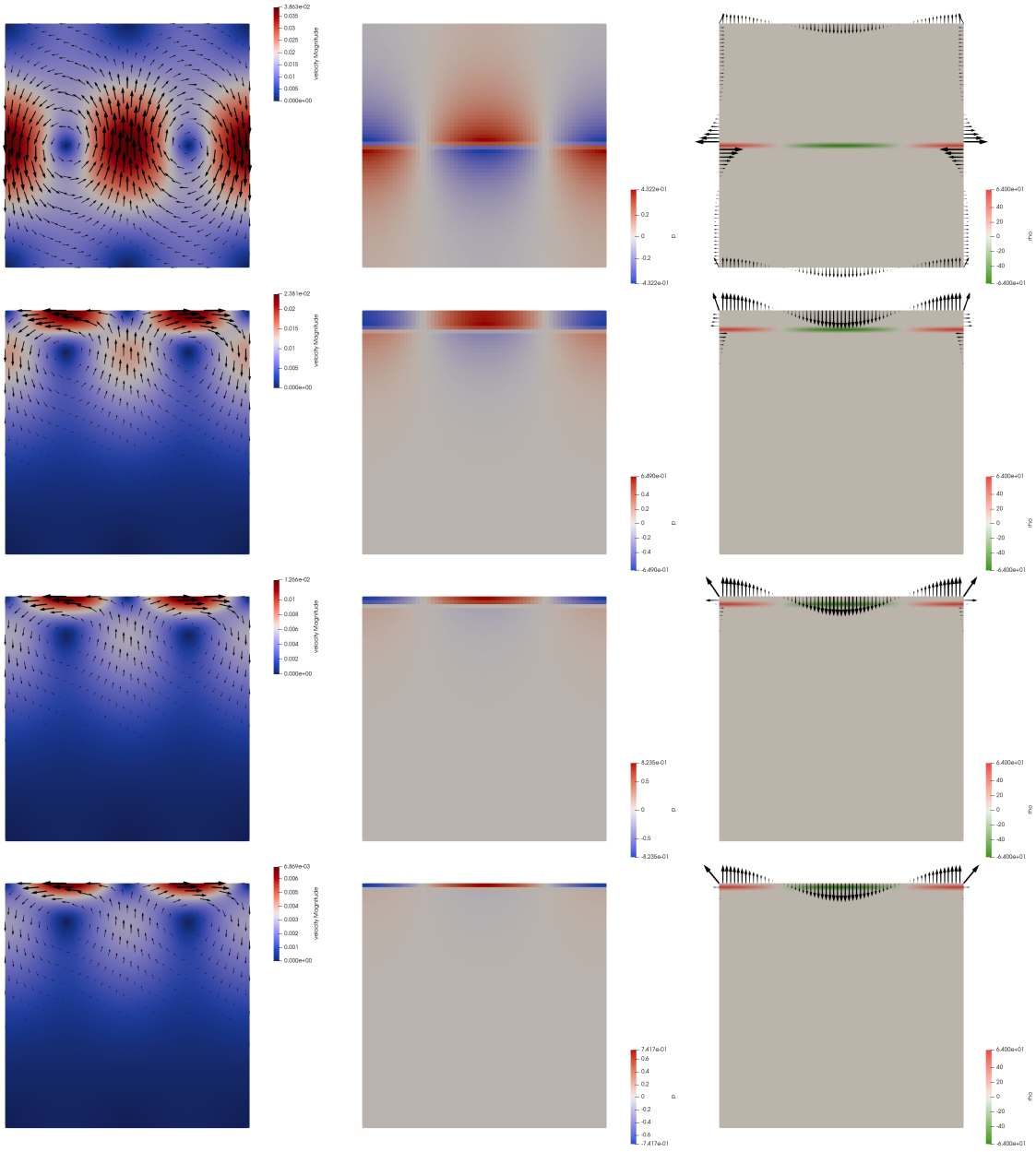
The consistent boundary flux (CBF) method allows us to compute traction vectors $\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n}$ on the boundary of the domain. On the top boundary, $\boldsymbol{n} = (0, 1)$ so that $\boldsymbol{t} = (\sigma_{xy}, \sigma_{yy})^T$ and t_y is the quantity we need to consider and compare to other results.

In the following table are shown the results presented in [103] alongside the results obtained with Fieldstone:

Method	$y_0 = 63/64$	$y_0 = 62/64$	$y_0 = 59/64$ [12]	$y_0 = 32/64$
Analytic solution	0.995476	0.983053	0.912506	0.178136
Pressure smoothing [103]	1.15974	1.06498	0.911109	n.a.
CBF [103]	0.994236	0.982116	0.912157	n.a.
fieldstone: elemental	0.824554 (-17.17 %)	0.978744 (-0.44%)	0.909574 (-0.32 %)	0.177771 (-0.20 %)
fieldstone: nodal (C→N)	0.824554 (-17.17 %)	0.978744 (-0.44%)	0.909574 (-0.32 %)	0.177771 (-0.20 %)
fieldstone: LS	1.165321 (17.06 %)	1.070105 (8.86%)	0.915496 (0.33 %)	0.178182 (0.03 %)
fieldstone: CBF	0.994236 (-0.13 %)	0.982116 (-0.10%)	0.912157 (-0.04 %)	0.177998 (-0.08 %)

We see that we recover the published results with the same exact accuracy, thereby validating our implmentation.

On the following figures are shown the velocity, pressure and traction fields for two cases $y_0 = 32/64$ and $y_0 = 63/64$.



Here lies the superiority of our approach over the one presented in the original article: our code computes all traction vectors on all boundaries at once.

explain how Medge is arrived at!

compare with ASPECT ??!

gauss-lobatto integration?

pressure average on surface instead of volume ?

features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- isothermal
- isoviscous
- analytical solution
- pressure smoothing
- consistent boundary flux

35 `fieldstone_28`: convection 2D box - Tosi et al, 2015

The viscosity field μ is calculated as the harmonic average between a linear part μ_{lin} that depends on temperature only or on temperature and depth d , and a non-linear, plastic part μ_{plast} dependent on the strain rate:

$$\mu(T,z,\dot{\boldsymbol{\epsilon}}) = 2\left(\frac{1}{\mu_{\rm lin}(T,z)} + \frac{1}{\mu_{\rm plast}(\dot{\boldsymbol{\epsilon}})}\right)^{-1}. \tag{130}$$

The linear part is given by the linearized Arrhenius law (the so-called Frank-Kamenetskii approximation [**?**]):

$$\mu_{\rm lin}(T,z) = \exp(-\gamma_T T + \gamma_z z), \tag{131}$$

where $\gamma_T = \ln(\Delta\mu_T)$ and $\gamma_z = \ln(\Delta\mu_z)$ are parameters controlling the total viscosity contrast due to temperature ($\Delta\mu_T$) and pressure ($\Delta\mu_z$). The non-linear part is given by [**?**]:

$$\mu_{\rm plast}(\dot{\boldsymbol{\epsilon}}) = \mu^* + \frac{\sigma_Y}{\sqrt{\dot{\boldsymbol{\epsilon}} : \dot{\boldsymbol{\epsilon}}}}, \tag{132}$$

where μ^* is a constant representing the effective viscosity at high stresses [**?**] and σ_Y is the yield stress, also assumed to be constant. In 2-D, the denominator in the second term of equation (132) is given explicitly by

$$\sqrt{\dot{\boldsymbol{\epsilon}} : \dot{\boldsymbol{\epsilon}}} = \sqrt{\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}} = \sqrt{\left(\frac{\partial u_x}{\partial x}\right)^2 + \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right)^2 + \left(\frac{\partial u_y}{\partial y}\right)^2}. \tag{133}$$

The viscoplastic flow law (equation 130) leads to linear viscous deformation at low stresses (equation (131)) and to plastic deformation for stresses that exceed σ_Y (equation (132)), with the decrease in viscosity limited by the choice of μ^* [**?**].

In all cases that we present, the domain is a two-dimensional square box. The mechanical boundary conditions are for all boundaries free-slip with no flux across, i.e. $\tau_{xy} = \tau_{yx} = 0$ and $\boldsymbol{u} \cdot \boldsymbol{n} = 0$, where $\boldsymbol{n}$ denotes the outward normal to the boundary. Concerning the energy equation, the bottom and top boundaries are isothermal, with the temperature T set to 1 and 0, respectively, while side-walls are assumed to be insulating, i.e. $\partial T/\partial x = 0$. The initial distribution of the temperature field is prescribed as follows:

$$T(x,y) = (1-y) + A\cos(\pi x)\sin(\pi y), \tag{134}$$

where $A = 0.01$ is the amplitude of the initial perturbation.

In the following Table **??**, we list the benchmark cases according to the parameters used.

Case	Ra	$\Delta\mu_T$	$\Delta\mu_y$	μ^*	σ_Y	Convective regime
1	10^2	10^5	1	–	–	Stagnant lid
2	10^2	10^5	1	10^{-3}	1	Mobile lid
3	10^2	10^5	10	–	–	Stagnant lid
4	10^2	10^5	10	10^{-3}	1	Mobile lid
5a	10^2	10^5	10	10^{-3}	4	Periodic
5b	10^2	10^5	10	10^{-3}	3 – 5	Mobile lid – Periodic – Stagnant lid

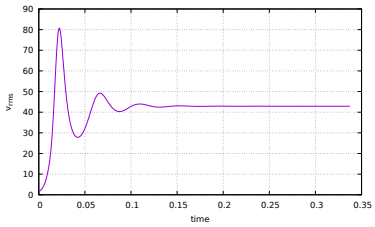
Benchmark cases and corresponding parameters.

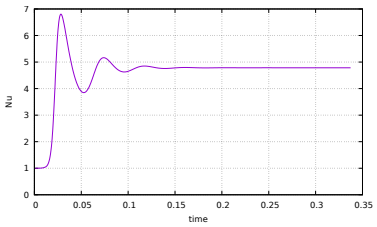
In Cases 1 and 3 the viscosity is directly calculated from equation (131), while for Cases 2, 4, 5a, and 5b, we used equation (130). For a given mesh resolution, Case 5b requires running simulations with yield stress varying between 3 and 5

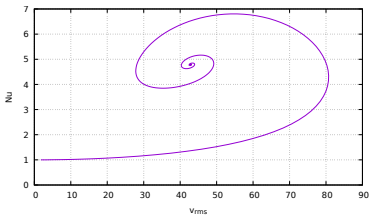
In all tests, the reference Rayleigh number is set at the surface ($y = 1$) to 10^2, and the viscosity contrast due to temperature $\Delta\mu_T$ is 10^5, implying therefore a maximum effective Rayleigh number of 10^7 for $T = 1$. Cases 3, 4, 5a, and 5b employ in addition a depth-dependent rheology with viscosity contrast $\Delta\mu_z = 10$. Cases 1 and 3 assume a linear viscous rheology that leads to a stagnant lid regime. Cases 2 and 4 assume a viscoplastic rheology that leads instead to a mobile lid regime. Case 5a also

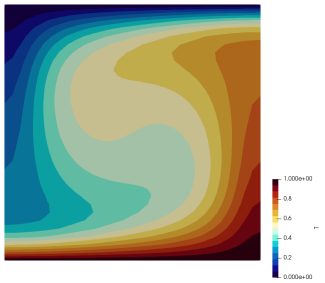
assumes a viscoplastic rheology but a higher yield stress, which ultimately causes the emergence of a strictly periodic regime. The setup of Case 5b is identical to that of Case 5a but the test consists in running several simulations using different yield stresses. Specifically, we varied σ_Y between 3 and 5 in increments of 0.1 in order to identify the values of the yield stress corresponding to the transition from mobile to periodic and from periodic to stagnant lid regime.

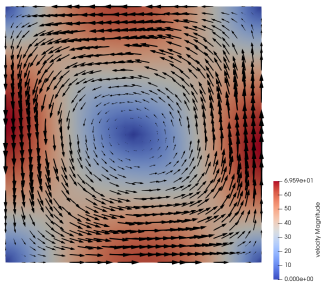
35.0.1 Case 0: Newtonian case, a la Blankenbach et al., 1989







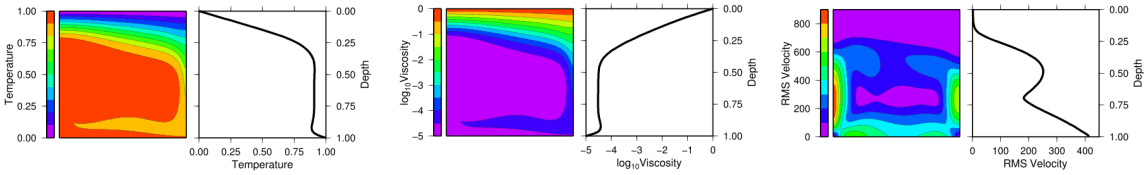


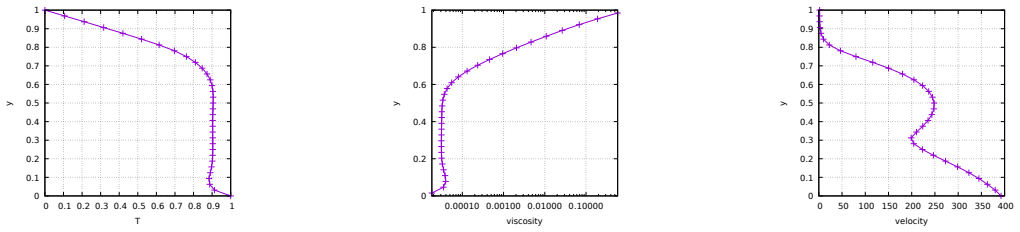


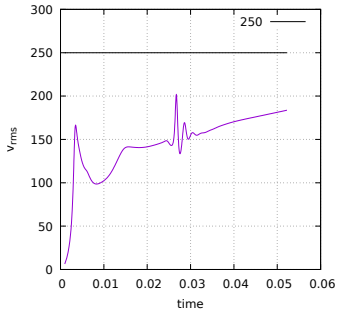
35.0.2 Case 1

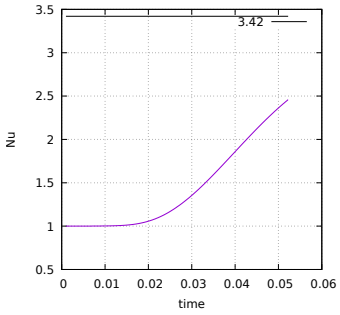
In this case $\mu^\star = 0$ and $\sigma_Y = 0$ so that μ_{plast} can be discarded. The CFL number is set to 0.5 and the viscosity is given by $\mu(T, z, \dot{\epsilon}) = \mu_{\rm lin}(T, z)$. And since $\Delta\mu_z = 1$ then $\gamma_z = 0$ so that $\mu_{\rm lin}(T, z) = \exp(-\gamma_T T)$

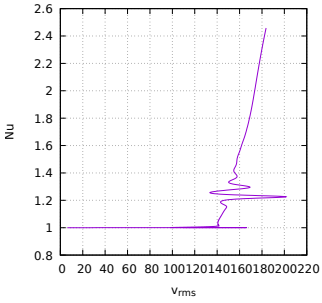
Code	YACC	Plaatjes	CHIC	GAIA	StreamV	StagYY	FEniCS	Fluidity	ELEFANT	ASPECT	MC3D
Resolution	100×100	$128{\times}128r$	80×80	$100r{\times}100r$	80×80	$128{\times}128r$	80×80	128×128	100×100	64×64	100×100
Case 1											
$\langle T \rangle$	0.7767	0.7759	0.7758	0.7759	0.776	0.776	0.7759	0.7758	0.7758	0.7768	0.779
$Nu_{\rm top}$	3.4298	3.4159	3.4260	3.4213	3.4091	3.419	3.5889	3.4253	3.4214	3.4305	3.3129
$Nu_{\rm bot}$	3.3143	3.4159	3.4259	3.4213	3.4091	3.419	3.4231	3.3795	3.313	3.4142	3.3139
$u_{\rm RMS}$	251.7997	249.54	249.2985	250.0738	252.0906	249.541	249.5730	248.9252	249.134	251.3069	296.6156
$u^{\rm surf}_{\rm RMS}$	1.8298	1.878	1.8999	1.8836	1.8823	1.8723	1.8698	1.8474	1.8642	1.8695	1.1114
$u^{\rm surf}_{\rm max}$	2.5516	2.618	2.6477	2.6254	2.64	2.6104	2.6066	2.5761	2.6119	2.6064	1.5329
$\langle W \rangle$	2.4583	2.369	2.431	2.4121	2.4071	2.4189	2.4246	2.4148	2.4316	2.4282	2.5548
$\langle \Phi \rangle$/Ra	2.4333	2.4119	2.4189	2.4165	2.392	2.4182	2.4246	2.4148	2.4276	2.4281	2.31916
δ	1.02%	1.78%	0.50%	0.18%	0.63%	0.03%	$< 0.01\%$	$< 0.01\%$	0.16%	$< 0.01\%$	9.22%

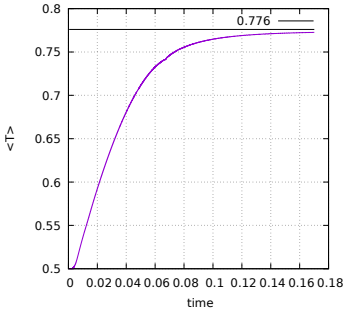








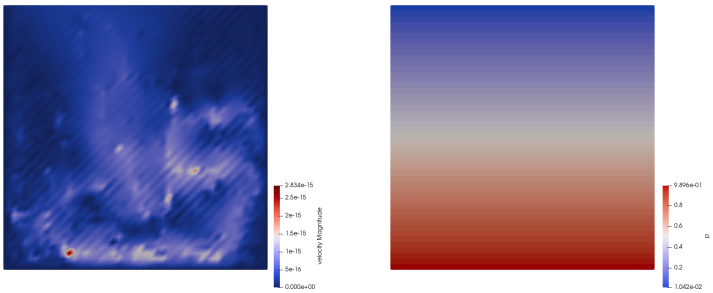




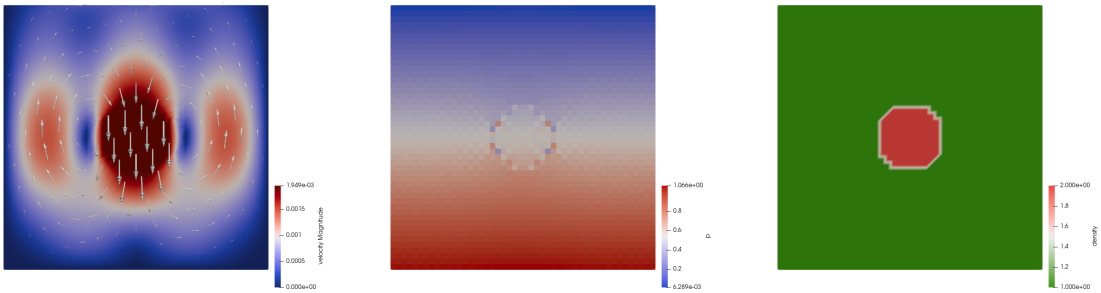
36 `fieldstone_29`: open boundary conditions

In what follows we will investigate the use of the so-called open boundary conditions in the very simple context of a 2D Stokes sphere experiment.

We start with a domain without the sphere. Essentially, it is what people would call an aquarium. Free slip boundary conditions are prescribed on the sides and no-slip conditions at the bottom. The top surface is left free. The fluid has a density $\rho_0 = 1$ and a viscosity $\eta_0 = 1$. In the absence of any density difference in the domain there is no active buoyancy force so that we expect a zero velocity field and a lithostatic pressure field. This is indeed what we recover:



If we now implement a sphere parametrised by its density $\rho_s = \rho_0 + 1$, its viscosity $\eta_s = 10^3$ and its radius $R_s = 0.123$ in the middle of the domain, we see clear velocity field which logically shows the sphere falling downward and a symmetric return flow of the fluid on each side:

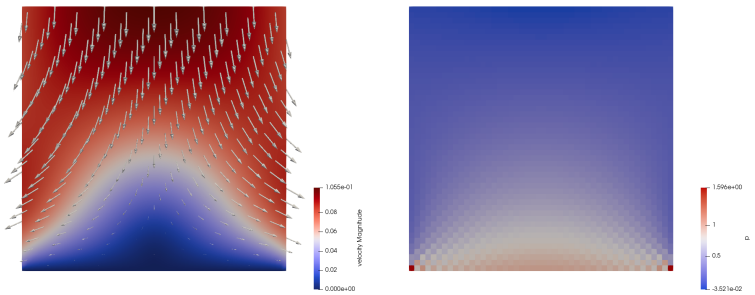


Unfortunately it has been widely documented that the presence of free-slip boundary conditions affects the evolution of subduction [18], even when these are placed rather far from the subduction zone. A proposed solution to this problem is the use of 'open boundary conditions' which are in fact stress boundary conditions. The main idea is to prescribe a stress on the lateral boundaries (instead of free slip) so that it balances out exactly the existing lithostatic pressure inside the domain along the side walls. Only pressure deviations with respect to the lithostatic are responsible for flow and such boundary conditions allow flow across the boundaries.

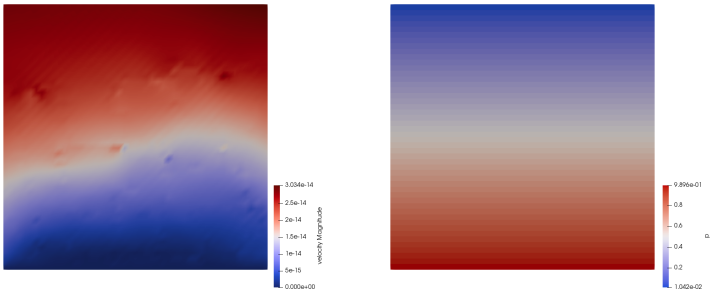
We need the lithostatic pressure and compute it before hand (which is trivial in our case but can prove to be a bit more tedious in real life situations when for instance density varies in the domain as a function of temperature and/or pressure).

```
plith = np.zeros(nnp,dtype=np.float64)
for i in range(0,nnp):
    plith[i]=(Ly-y[i])*rho0*abs(gy)
```

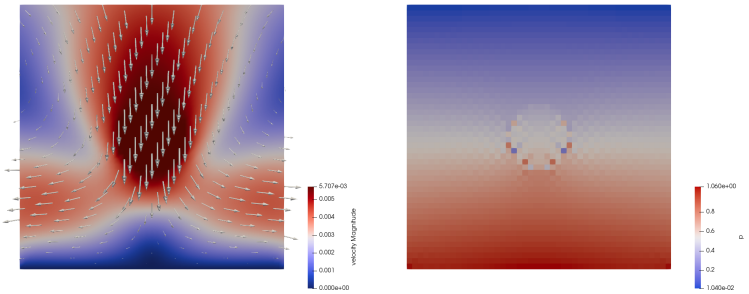
Let us start with a somewhat pathological case: even in the absence of the sphere, what happens when no boundary conditions are prescribed on the sides? The answer is simple: think about an aquarium without side walls, or a broken dam. The velocity field indeed shows a complete collapse of the fluid left and right of the bottom.



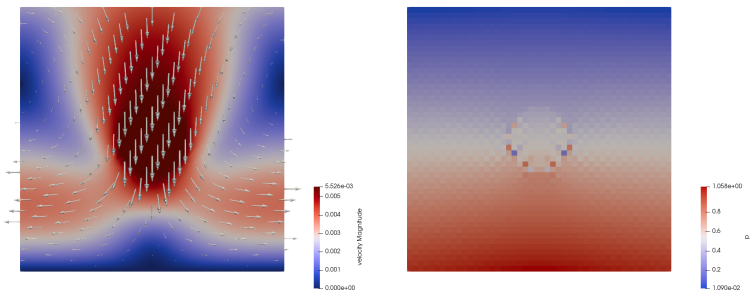
Let us then continue (still with no sphere) but let us now switch on the open boundary conditions. Since the side boundary conditions match the lithostatic pressure we expect no flow at all in the absence of any density perturbation in the system. This is indeed what is recovered:



Finally, let us reintroduce the sphere. This time flow is allowed through the left and right side boundaries:



Finally, although horizontal velocity Dirichlet boundary conditions and open boundary conditions are not compatible, the same is not true for the vertical component of the velocity: the open b.c. implementation acts on the horizontal velocity dofs only, so that one can fix the vertical component to zero, as is shown hereunder:



We indeed see that the in/outflow on the sides is perpendicular to the boundaries.

Turning now to the actual implementation, we see that it is quite trivial, since all element edges are vertical, and all have the same vertical dimension h_x. Since we use a Q_0 approximation for the pressure

we need to prescribe a single pressure value in the middle of the element. Finally because of the sign of the normal vector projection onto the x-axis, we obtain:

```
if open_bc_left and x[icon[0,iel]]<eps: # left side
   pmid=0.5*(plith[icon[0,iel]]+plith[icon[3,iel]])
   f_el[0]+=0.5*hy*pmid
   f_el[6]+=0.5*hy*pmid
if open_bc_right and x[icon[1,iel]]>Lx-eps: # right side
   pmid=0.5*(plith[icon[1,iel]]+plith[icon[2,iel]])
   f_el[2]-=0.5*hy*pmid
   f_el[4]-=0.5*hy*pmid
```

These few lines of code are added after the elemental matrices and rhs are built, and before the application of other Dirichlet boundary conditions, and assembly.

features

- $Q_1 \times P_0$ element
- incompressible flow
- mixed formulation
- open boundary conditions
- isoviscous

37 `fieldstone_30`: conservative velocity interpolation

features

- $Q_1 \times P_0$ element
- incompressible flow
- penalty formulation
- Dirichlet boundary conditions (free-slip)
- direct solver
- isothermal
- non-isoviscous
- analytical solution

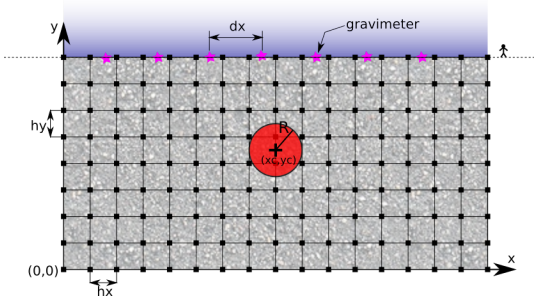
38 `fieldstone`: Gravity: buried sphere

Before you proceed further, please read :

`http://en.wikipedia.org/wiki/Gravity_anomaly`

`http://en.wikipedia.org/wiki/Gravimeter`

Let us consider a vertical domain $Lx \times Ly$ where $L_x = 1000km$ and $Ly = 500km$. This domain is discretised by means of a grid which counts $nnp = nnx \times nny$ nodes. This grid then counts $nel = nelx \times nely = (nnx - 1) \times (nny - 1)$ cells. The horizontal spacing between nodes is hx and the vertical spacing is hy.



Assume that this domain is filled with a rock type which mass density is given by $\rho_{medium} = 3000kg/m^3$, and that there is a circular inclusion of another rock type ($\rho_{sphere} = 3200kg/m^3$) at location ($xsphere$, $ysphere$) of radius $rsphere$. The density in the system is then given by

$$\rho(x,y) = \begin{cases} \rho_{sphere} \text{ inside the circle} \\ \rho_{medium} \text{ outside the circle} \end{cases}$$

Let us now assume that we place $nsurf$ gravimeters at the surface of the model. These are placed equidistantly between coordinates $x = 0$ and coordinates $x = Lx$. We will use the arrays $xsurf$ and $ysurf$ to store the coordinates of these locations. The spacing between the gravimeters is $\delta_x = Lx/(nsurf - 1)$.

At any given point (x_i, y_i) in a 2D space, one can show that the gravity anomaly due to the presence of a circular inclusion can be computed as follows:

$$g(x_i,y_i) = 2\pi\mathcal{G}(\rho_{sphere} - \rho_0)R^2 \frac{y_i - y_{sphere}}{(x_i - x_{sphere})^2 + (y_i - y_{sphere})^2} \tag{135}$$

where r_{sphere} is the radius of the inclusion, (x_{sphere}, y_{sphere}) are the coordinates of the center of the inclusion, and ρ_0 is a reference density.

However, the general formula to compute the gravity anomaly at a given point (x_i, y_i) in space due to a density anomaly of any shape is given by:

$$g(x_i,y_i) = 2\mathcal{G} \int\int_\Omega \frac{\Delta\rho(x,y)(y - y_i)}{(x - x_i)^2 + (y - y_i)^2} dxdy \tag{136}$$

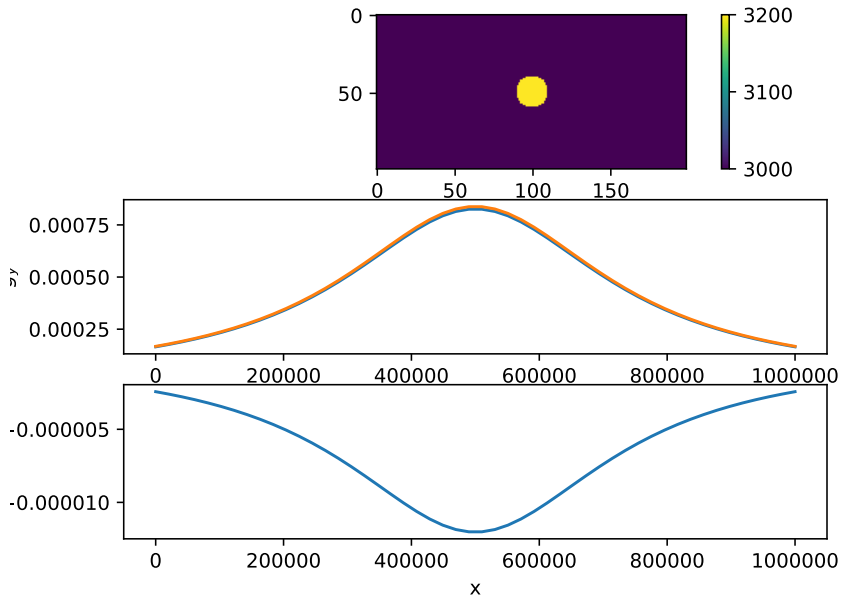
where Ω is the area of the domain on which the integration is to be carried out. Furthermore the density anomaly can be written : $\Delta\rho(x,y) = \rho(x,y) - \rho_0$. We can then carry out the integration for each cell and sum their contributions:

$$g(x_i,y_i) = 2\mathcal{G} \sum_{ic=1}^{nel} \int\int_{\Omega_e} \frac{(\rho(x,y) - \rho_0)(y - y_i)}{(x - x_i)^2 + (y - y_i)^2} dxdy \tag{137}$$

where Ω_e is now the area of a single cell. Finally, one can assume the density to be constant within each cell so that $\rho(x,y) \to \rho(ic)$ and $\int\int_{\Omega_e} dxdy \to hx \times hy$ and then

$$g(x_i,y_i) = 2\mathcal{G} \sum_{ic=1}^{nel} \frac{(\rho(ic) - \rho_0)(y(ic) - y_i)}{(x(ic) - x_i)^2 + (y(ic) - y_i)^2} s_x s_y \tag{138}$$

We will then use the array $gsurf$ to store the value of the gravity anomaly measured at each gravimeter at the surface.



To go further

- explore the effect of the size of the inclusion on the gravity profile.
- explore the effect of the ρ_0 value.
- explore the effect of the grid resolution.
- measure the time that is required to compute the gravity. How does this time vary with nsurf ? how does it vary when the grid resolution is doubled ?
- Assume now that $\rho_2 < \rho_1$. What does the gravity profile look like ?
- what happens when the gravimeters are no more at the surface of the Earth but in a satellite ?
- if you feel brave, redo the whole exercise in 3D...

References

[1] M. Albers. A local mesh refinement multigrid method for 3D convection problems with strongly variable viscosity. *J. Comp. Phys.*, 160:126–150, 2000.

[2] V. Allken, R. Huismans, and C. Thieulot. Three dimensional numerical modelling of upper crustal extensional systems. *J. Geophys. Res.*, 116:B10409, 2011.

[3] V. Allken, R. Huismans, and C. Thieulot. Factors controlling the mode of rift interaction in brittle-ductile coupled systems: a 3d numerical study. *Geochem. Geophys. Geosyst.*, 13(5):Q05010, 2012.

[4] V. Allken, R.S. Huismans, H. Fossen, and C. Thieulot. 3D numerical modelling of graben interaction and linkage: a case study of the Canyonlands grabens, Utah. *Basin Research*, 25:1–14, 2013.

[5] J.D. Anderson. *Computational Fluid Dynamics*. McGraw-Hill, 1995.

[6] K.-J. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, 1982.

[7] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[8] D. Bercovici, G. Schubert, and G.A. Glatzmaier. Three-dimensional convection of an infinite Prandtl-number compressible fluid in a basally heated spherical shell. *J. Fluid Mech.*, 239:683–719, 1992.

[9] B. Blankenbach, F. Busse, U. Christensen, L. Cserepes, D. Gunkel, U. Hansen, H. Harder, G. Jarvis, M. Koch, G. Marquart, D. Moore, P. Olson, H. Schmeling, and T. Schnaubelt. A benchmark comparison for mantle convection codes. *Geophys. J. Int.*, 98:23–38, 1989.

[10] P. B. Bochev, C. R. Dohrmann, and M. D. Gunzburger. Stabilization of low-order mixed finite elements for the stokes equations. *SIAM J. Numer. Anal.*, 44(1):82–101, 2006.

[11] J. Braun, C. Thieulot, P. Fullsack, M. DeKool, and R.S. Huismans. DOUAR: a new three-dimensional creeping flow model for the solution of geological problems. *Phys. Earth. Planet. Inter.*, 171:76–91, 2008.

[12] J. Braun and P. Yamato. Structural evolution of a three-dimensional, finite-width crustal wedge. *Tectonophysics*, 484:181–192, doi:10.1016/j.tecto.2009.08.032, 2009.

[13] H.H. Bui, R. Fukugawa, K. Sako, and S. Ohno. Lagrangian meshfree particles method (SPH) for large deformation and failure flows of geomaterial using elasticplastic soil constitutive model. *Int. J. Numer. Anal. Geomech.*, 32(12):1537–1570, 2008.

[14] P.S. Bullen. *Handbook of Means and Their Inequalities*. Springer; 2nd edition, 2003.

[15] C. Burstedde, G. Stadler, L. Alisic, L.C. Wilcox, E. Tan, M. Gurnis, and O. Ghattas. Large-scale adaptive mantle convection simulation. *Geophy. J. Int.*, 192:889–906, 2013.

[16] F.H. Busse, U. Christensen, R. Clever, L. Cserepes, C. Gable, E. Giannandrea, L. Guillou, G. Houseman, H.-C. Nataf, M. Ogawa, M. Parmentier, C. Sotin, and B. Travis. 3D convection at infinite Prandtl number in Cartesian geometry - a benchmark comparison. *Geophys. Astrophys. Fluid Dynamics*, 75:39–59, 1993.

[17] J.S. Chen, C. Pan, and T.Y.P. Chang. On the control of pressure oscillation in bilinear-displacement constant-pressure element. *Comput. Methods Appl. Mech. Engrg.*, 128:137–152, 1995.

[18] M.V. Chertova, T. Geenen, A. van den Berg, and W. Spakman. Using open sidewalls for modelling self-consistent lithosphere subduction dynamics . *Solid Earth*, 3:313–326, 2012.

[19] Edmund Christiansen and Knud D. Andersen. Computation of collapse states with von mises type yield condition. *International Journal for Numerical Methods in Engineering*, 46:1185–1202, 1999.

[20] Edmund Christiansen and Ole S. Pedersen. Automatic mesh refinement in limit analysis. *International Journal for Numerical Methods in Engineering*, 50:1331–1346, 2001.

[21] M. crouzeix and P.A. Raviart. Conforming and non-conforming finite element methods for solving the stationary Stokes equations. *RAIRO*, 7:33–76, 1973.

[22] C. Cuvelier, A. Segal, and A.A. van Steenhoven. *Finite Element Methods and Navier-Stokes Equations*. D. Reidel Publishing Company, 1986.

[23] D.R. Davies, C.R. Wilson, and S.C. Kramer. Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics. *Geochem. Geophys. Geosyst.*, 12(6), 2011.

[24] P. Davy and P. Cobbold. Indentation tectonics in nature and experiment. 1. experiments scaled for gravity. *Bulletin of the Geological Institutions of Uppsala*, 14:129–141, 1988.

[25] J. de Frutos, V. John, and J. Novo. Projection methods for incompressible ow problems with WENO nite difference schemes. *J. Comp. Phys.*, 309:368–386, 2016.

[26] Y. Deubelbeiss and B.J.P. Kaus. Comparison of Eulerian and Lagrangian numerical techniques for the Stokes equations in the presence of strongly varying viscosity. *Phys. Earth Planet. Interiors*, 171:92–111, 2008.

[27] C.R. Dohrmann and P.B. Bochev. A stabilized finite element method for the Stokes problem based on polynomial pressure projections. *Int. J. Num. Meth. Fluids*, 46:183–201, 2004.

[28] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. 2003.

[29] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons, 2003.

[30] Jean Donea and Antonio Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons, 2003.

[31] T. Duretz, D.A. May, T.V. Gerya, and P.J. Tackley. Discretization errors and free surface stabilisation in the finite difference and marker-in-cell method for applied geodynamics: A numerical study. *Geochem. Geophys. Geosyst.*, 12(Q07004), 2011.

[32] R. Eid. Higher order isoparametric finite element solution of Stokes flow . *Applied Mathematics and Computation*, 162:1083–1101, 2005.

[33] E. Erturk. Discussions on Driven Cavity Flow. *Int. J. Num. Meth. Fluids*, 60:275–294, 2009.

[34] P.J. Frey and P.-L. George. *Mesh generation*. Hermes Science, 2000.

[35] P. Fullsack. An arbitrary Lagrangian-Eulerian formulation for creeping flows and its application in tectonic models. *Geophy. J. Int.*, 120:1–23, 1995.

[36] M. Gerbault, A.N.B. Poliakov, and M. Daignieres. Prediction of faulting from the theories of elasticity and plasticity: what are the limits? *Journal of Structural Geology*, 20:301–320, 1998.

[37] Taras Gerya. *Numerical Geodynamic Modelling*. Cambridge University Press, 2010.

[38] T.V. Gerya, D.A. May, and T. Duretz. An adaptive staggered grid finite difference method for modeling geodynamic Stokes flows with strongly variable viscosity. *Geochem. Geophys. Geosyst.*, 14(4), 2013.

[39] G.H. Golub and C.F. van Loan. *Matrix Computations, 4th edition*. John Hopkins University Press, 2013.

[40] P.M. Gresho and R.L. Sani. *Incompressible flow and the Finite Element Method, vol II*. John Wiley and Sons, Ltd, 2000.

[41] D. Griffiths and D. Silvester. Unstable modes of the q1-p0 element. Technical Report 257, University of MAnchester/UMIST, 1994.

[42] M. Gunzburger. *Finite Element Methods for Viscous Incompressible Flows: A Guide to Theory, Practice and Algorithms*. Academic, Boston, 1989.

[43] T. Heister, J. Dannberg, R. Gassmöller, and W. Bangerth. High Accuracy Mantle Convection Simulation through Modern Numerical Methods. II: Realistic Models and Problems. *Geophy. J. Int.*, 210(2):833–851, 2017.

[44] T.J.R. Hughes. *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Inc., 2000.

[45] T.J.R. Hughes, W.K. Liu, and A. Brooks. Finite element analysis of incompressible viscous flows by the penalty function formulation. *J. Comp. Phys.*, 30:1–60, 1979.

[46] Hoon Huh, Choong Ho Lee, and Wei H. Yang. A general algorithm for plastic flow simulation by finite element limit analysis. *International Journal of Solids and Structures*, 36:1193–1207, 1999.

[47] Alik Ismail-Zadeh and Paul Tackley. *Computational Methods for Geodynamics*. Cambridge University Press, 2010.

[48] J. Ita and S.D. King. Sensitivity of convection with an endothermic phase change to the form of governing equations, initial conditions, boundary conditions, and equation of state. *J. Geophys. Res.*, 99(B8):15,919–15,938, 1994.

[49] L. Jolivet, P. Davy, and P. Cobbold. Right-lateral shear along the Northwest Pacific margin and the India-Eurasia collision. *Tectonics*, 9(6):1409–1419, 1990.

[50] L.M. Kachanov. *Fundamentals of the Theory of Plasticity*. Dover Publications, Inc., 2004.

[51] M. Kimmritz and M. Braack. iDiscretization of the hydrostatic Stokes system by stabilized finite elements of equal order.

[52] S. King, C. Lee, P. van Keken, W. Leng, S. Zhong, E. Tan, N. Tosi, and M. Kameyama. A community benchmark for 2D Cartesian com- pressible convection in the Earths mantle. *Geophy. J. Int.*, 180:7387, 2010.

[53] J.R. Koseff and R.L. Street. The Lid-Driven Cavity Flow: A Synthesis of Qualitative and Quantitative Observations. *J. Fluids Eng*, 106:390–398, 1984.

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

[55] R. Lee, P. Gresho, and R. Sani. Smooting techniques for certain primitive variable solutions of the Navier-Stokes equations. . *Int. Journal for Numerical Methods in Engineering*, 14:1785–1804, 1979.

[56] W. Leng and S. Zhong. Viscous heating, adiabatic heating and energetic consistency in compressible mantle convection. *Geophy. J. Int.*, 173:693–702, 2008.

[57] W. Leng and S. Zhong. Implementation and application of adaptive mesh refinement for thermochemical mantle convection studies. *Geochem. Geophys. Geosyst.*, 12(4), 2011.

[58] J. Li, Y. He, and Z. Chen. Performance of several stabilized finite element methods for the Stokes equations based on the lowest equal-order pairs. *Computing*, 86:37–51, 2009.

[59] X. Liu and S. Zhong. Analyses of marginal stability, heat transfer and boundary layer properties for thermal convection in a compressible fluid with infinite Prandtl number. *Geophy. J. Int.*, 194:125–144, 2013.

[60] C. Loiselet, J. Braun, L. Husson, C. Le Carlier de Veslud, C. Thieulot, P. Yamato, and D. Grujic. Subducting slabs: Jellyfishes in the Earth's mantle. *Geochem. Geophys. Geosyst.*, 11(8):doi:10.1029/2010GC003172, 2010.

[61] D.S. Malkus and T.J.R. Hughes. Mixed finite element methods - reduced and selective integration techniques: a unification of concepts. *Comput. Meth. Appl. Mech. Eng.*, 15:63–81, 1978.

[62] L.E. Malvern. *Introduction to the mechanics of a continuous medium*. Prentice-Hall, Inc., 1969.

[63] A. Mizukami. A mixed finite element method for boundary flux computation. *Computer Methods in Applied Mechanics and Engineering*, 57:239–243, 1986.

[64] P. Molnar and P. Tapponnier. Relation of the tectonics of eastern China to the India-Eurasia collision: Application of the slip-line field theory to large-scale continental tectonics. *Geology*, 5:212–216, 1977.

[65] L. Moresi, S. Quenette, V. Lemiale, C. Mériaux, B. Appelbe, and H.-B. Mühlhaus. Computational approaches to studying non-linear dynamics of the crust and mantle. *Phys. Earth. Planet. Inter.*, 163:69–82, 2007.

[66] M. Nettesheim, T.A. Ehlers, D.M. Whipp, and A. Koptev. The influence of upper-plate advance and erosion on overriding plate deformation in orogen syntaxes. *Solid Earth*, 9:1207–1224, 2018.

[67] S. Norburn and D. Silvester. Fourier analysis of stabilized Q1-Q1 mixed finite element approximation. *SIAM J. Numer. Anal.*, 39:817–833, 2001.

[68] C. O'Neill, L. Moresi, D. Müller, R. Albert, and F. Dufour. Ellipsis 3D: a particle-in-cell finite element hybrid code for modelling mantle convection and lithospheric deformation. *Computers and Geosciences*, 32:1769–1779, 2006.

[69] G. Peltzer and P. Tapponnier. Formation and evolution of strike-slip faults, rifts, and basins during the india-asia collision: an experimental approach. *J. Geophys. Res.*, 93(B12):15085–15177, 1988.

[70] A. Pinelli and A. Vacca. Chebyshev collocation method and multidomain decomposition for the incompressible Navier-Stokes equations. *International Journal for numerical methods in fluids*, 18:781–799, 1994.

[71] A.E. Pusok, B.J.P. Kaus, and A.A. Popov. On the Quality of Velocity Interpolation Schemes for Marker-in-Cell Method and Staggered Grids. *Pure and Applied Geophysics*, pages doi:10.1007/s00024–016–1431–8, 2016.

[72] T. Rabczuk, P.M.A. Areias, and T. Belytschko. A simplified mesh-free method for shear bands with cohesive surfaces . *Int. J. Num. Meth. Eng.*, 69:993–1021, 2007.

[73] J.N. Reddy. On penalty function methods in the finite element analysis of flow problems. *Int. J. Num. Meth. Fluids*, 2:151–171, 1982.

[74] J. Revenaugh and B. Parsons. Dynamic topography and gravity anomalies for fluid layers whose viscosity varies exponentially with depth. *Geophysical Journal of the Royal Astronomical Society*, 90(2):349–368, 1987.

[75] J.G. Rice and R.J. Schnipke. An equal-order velocity-pressure formulation that does not exhibit spurious pressure modes. *Computer Methods in Applied Mechanics and Engineering*, 58:135–149, 1986.

[76] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[77] R.L. Sani, P.M. Gresho, R.L. Lee, and D.F. Griffiths. The cause and cure (?) of the spurious pressures generated by certain FEM solutions of the incompressible Navier-Stokes equations: part 1. *Int. J. Num. Meth. Fluids*, 1:17–43, 1981.

[78] R.L. Sani, P.M. Gresho, R.L. Lee, D.F. Griffiths, and M. Engelman. The cause and cure (?) of the spurious pressures generated by certain fem solutions of the incompressible navier-stokes equations: part 2. *Int. J. Num. Meth. Fluids*, 1:171–204, 1981.

[79] S.M. Schmalholz. A simple analytical solution for slab detachment. *Earth Planet. Sci. Lett.*, 304:45–54, 2011.

[80] H. Schmeling, A.Y. Babeyko, A. Enns, C. Faccenna, F. Funiciello, T. Gerya, G.J. Golabek, S. Grigull, B.J.P. Kaus, G. Morra, S.M. Schmalholz, and J. van Hunen. A benchmark comparison of spontaneous subduction models - Towards a free surface. *Phys. Earth. Planet. Inter.*, 171:198–223, 2008.

[81] D.W. Schmid and Y.Y. Podlachikov. Analytical solutions for deformable elliptical inclusions in general shear. *Geophy. J. Int.*, 155:269–288, 2003.

[82] G. Schubert, D.L. Turcotte, and P. Olson. *Mantle Convection in the Earth and Planets.* Cambridge University Press, 2001.

[83] M. Spiegelman, D.A. May, and C. Wilson. On the solvability of incompressible Stokes with viscoplastic rheologies in geodynamics. *Geochem. Geophys. Geosyst.*, 17:2213–2238, 2016.

[84] G. Stadler, M. Gurnis, C. Burstedde, L.C. Wilcox, L. Alisic, and O. Ghattas. The dynamics of plate tectonics and mantle flow: from local to global scales. *Science*, 329:1033–1038, 2010.

[85] J. Suckale, J.-C. Nave, and B.H. Hager. It takes three to tango: 1. Simulating buoyancy-driven flow in the presence of large viscosity contrasts. *J. Geophys. Res.*, 115(B07409), 2010.

[86] P. Tackley. *Three-dimensional models of mantle convection: Influence of phase transitions and temperature-dependent viscosity.* PhD thesis, California Institute of Technology, 1994.

[87] E. Tan and M. Gurnis. Compressible thermochemical convection and application to lower mantle structures. *J. Geophys. Res.*, 112(B06304), 2007.

[88] Paul Tapponnier and Peter Molnar. Slip-line field theory and large-scale continental tectonics. *Nature*, 264:319–324, November 1976.

[89] M. Thielmann, D.A. May, and B.J.P. Kaus. Discretization errors in the Hybrid Finite Element Particle-In-Cell Method. *Pure and Applied Geophysics*, 2014.

[90] C. Thieulot. FANTOM: two- and three-dimensional numerical modelling of creeping flows for the solution of geological problems. *Phys. Earth. Planet. Inter.*, 188:47–68, 2011.

[91] C. Thieulot. GHOST: Geoscientific Hollow Sphere Tesselation. *Solid Earth*, 9(1–9), 2018.

[92] C. Thieulot, P. Fullsack, and J. Braun. Adaptive octree-based finite element analysis of two- and three-dimensional indentation problems. *J. Geophys. Res.*, 113:B12207, 2008.

[93] J.F. Thompson, B.K. Soni, and N.P. Weatherill. *Handbook of grid generation.* CRC press, 1998.

[94] R.A. Trompert and U. Hansen. On the Rayleigh number dependence of convection with a strongly temperature-dependent viscosity. *Physics of Fluids*, 10(2):351–360, 1998.

[95] D.L. Turcotte and G. Schubert. *Geodynamics, 2nd edition.* Cambridge, 2012.

[96] P.E. van Keken, S.D. King, H. Schmeling, U.R. Christensen, D. Neumeister, and M.-P. Doin. A comparison of methods for the modeling of thermochemical convection. *J. Geophys. Res.*, 102(B10):22,477–22,495, 1997.

[97] D.M. Whipp, C. Beaumont, and J. Braun. Feeding the aneurysm: Orogen-parallel mass transport into Nanga Parbat and the western Himalayan syntaxis. *J. Geophys. Res.*, 119:doi:10.1002/2013JB010929, 2014.

[98] S.D. Willett. Dynamic and kinematic growth and change of a coulomb wedge. In K.R. McClay, editor, *Thrust Tectonics*, pages 19–31. Chapman and Hall, 1992.

[99] H. Xing, W. Yu, and J. Zhang. In *Advances in Geocomputing, Lecture Notes in Earth Sciences.* Springer-Verlag, Berlin Heidelberg, 2009.

[100] P. Yamato, L. Husson, J. Braun, C. Loiselet, and C. Thieulot. Influence of surrounding plates on 3D subduction dynamics. *Geophys. Res. Lett.*, 36(L07303):doi:10.1029/2008GL036942, 2009.

[101] X. Yu and F. Tin-Loi. A simple mixed finite element for static limit analysis. *Computers and Structures*, 84:1906–1917, 2006.

[102] S. Zhong. Analytic solutions for Stokes flow with lateral variations in viscosity. *Geophys. J. Int.*, 124:18–28, 1996.

[103] S. Zhong, M. Gurnis, and G. Hulbert. Accurate determination of surface normal stress in viscous flow from a consistent boundary flux method. *Phys. Earth. Planet. Inter.*, 78:1–8, 1993.

[104] S. Zhong, A. McNamara, E. Tan, L. Moresi, and M. Gurnis. A benchmark study on mantle convection in a 3-D spherical shell using CITCOMS. *Geochem. Geophys. Geosyst.*, 9(10), 2008.

[105] S.J. Zhong, D.A. Yuen, and L.N. Moresi. *Treatise on Geophysics Volume 7 : Mantle Dynamics*. Elsevier B.V., 2007.

[106] O. Zienkiewicz and S. Nakazawa. The penalty function method and its application to the numerical solution of boundary value problems. *The American Society of Mechanical Engineers*, 51, 1982.

[107] O.C. Zienkiewicz, M. Huang, and M. Pastor. Localization problems in plasticity using finite elements with adaptive remeshing. *International Journal for Numerical and Analytical Methods in Geomechanics*, 19:127–148, 1995.

[108] O.C. Zienkiewicz, C. Humpheson, and R.W. Lewis. Associated and non-associated visco-plasticity and plasticity in soil mechanics . *Géotechnique*, 25(4):671–689, 1975.

Index

Notes