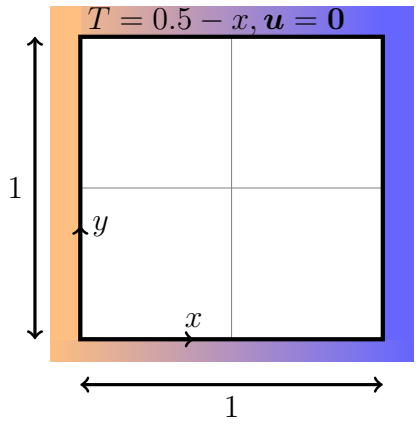

Figure 1: Problem setup and boundary conditions

Project 2: Natural convection in a differentially heated square cavity

Due Date: Friday 24.01.2020 or before

The aim of that project is to investigate the linear stability of the flow in a square of length L, subject to a temperature gradient imposed at its boundary. Using the Boussinesq equation the flow field will be computed by the mean of finite element method. After completing the partially blank codes, results will be compared with literature for different parameters.

1 Preliminary Questions

We recall the Navier-Stokes equation, with the Oberbeck-Boussinesq approximation, written in dimensionless parameters

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} &= \mathrm{Pr}\left(-\nabla p + \frac{1}{\sqrt{\mathrm{Ra}}}\Delta \boldsymbol{u} + \boldsymbol{e}_y T\right) \\ \nabla \cdot \boldsymbol{u} &= 0 \\ \frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T &= \frac{1}{\sqrt{Ra}}\Delta T \end{aligned} \tag{1}$$

with the boundary conditions

$$T = x - 1/2, \tag{2}$$

$$\boldsymbol{u} = \mathbf{0}. \tag{3}$$

1.1 Linearization

Let us define the baseflow $\boldsymbol{q}_0 = (\boldsymbol{u}_0, p_0, T_0)^T$ which is the solution of the steady system of equation:

$$\begin{aligned} (\boldsymbol{u}_0 \cdot \nabla)\, \boldsymbol{u}_0 &= \mathrm{Pr}\left(-\nabla p_0 + \frac{1}{\sqrt{\mathrm{Ra}}}\Delta \boldsymbol{u}_0 + \boldsymbol{e}_y T_0\right) \\ \nabla \cdot \boldsymbol{u}_0 &= 0 \end{aligned} \tag{4}$$

$$\boldsymbol{u}_0 \cdot \nabla T_0 = \frac{1}{\sqrt{Ra}} \Delta T_0$$

Consider the total flow as sum of the baseflow and a perturbation: $\boldsymbol{q} = \boldsymbol{q}_0 + \epsilon \boldsymbol{q}'$ where $\epsilon \ll 1$ and q' is of order 1.

- Insert the decomposition above in (1) and using that $\epsilon \ll 1$, show that the linearised system of equation reads

$$\begin{aligned} \frac{\partial \boldsymbol{u}'}{\partial t} + (\boldsymbol{u}' \cdot \nabla)\, \boldsymbol{u}_0 + (\boldsymbol{u}_0 \cdot \nabla)\, \boldsymbol{u}' &= \mathrm{Pr} \left(-\nabla p' + \frac{1}{\sqrt{\mathrm{Ra}}} \Delta \boldsymbol{u}' + \boldsymbol{e}_y T' \right) \\ \nabla \cdot \boldsymbol{u} &= 0 \\ \frac{\partial T'}{\partial t} + \boldsymbol{u}_0 \cdot \nabla T' + \boldsymbol{u}' \cdot \nabla T_0 &= \frac{1}{\sqrt{Ra}} \Delta T' \end{aligned} \tag{5}$$

  This system of equations correspond to the Jacobian of the system (4)

- What are the boundary conditions of the linearized system?

- Show that the variational form of the linearized **Steady** Navier–Stokes equations reads

$$\begin{aligned} &\int_V (\boldsymbol{u}_0 \cdot \nabla)\, \boldsymbol{u}' \cdot \boldsymbol{w}\ \mathrm{d}V + \int_V (\boldsymbol{u}' \cdot \nabla)\, \boldsymbol{u}_0 \cdot \boldsymbol{w}\ \mathrm{d}V + \frac{\mathrm{Pr}}{\sqrt{\mathrm{Ra}}} \int_V \nabla \boldsymbol{u}' \cdot \nabla \boldsymbol{w}\ \mathrm{d}V \\ &\qquad -\mathrm{Pr} \int_V T' \boldsymbol{e}_y \cdot \boldsymbol{w}\ \mathrm{d}V - \mathrm{Pr} \int_V p'\ \mathrm{div} \boldsymbol{w}\ \mathrm{d}V - \mathrm{Pr} \int_V q\ \mathrm{div} \boldsymbol{u}\ \mathrm{d}V \\ &+ \int_V (\boldsymbol{u}_0 \nabla)\, \boldsymbol{T}' \cdot \theta\ \mathrm{d}V + \int_V (\boldsymbol{u}' \nabla)\, \boldsymbol{T}_0 \cdot \theta\ \mathrm{d}V + \frac{1}{\sqrt{\mathrm{Ra}}} \int_V \nabla \boldsymbol{T}' \cdot \nabla \theta\ \mathrm{d}V = 0 \end{aligned} \tag{6}$$

1.2 Newton Method

In the UE, the non linear problem was solved using the Newton method implemented in FEniCS. Here we propose to create our own Newton method algorithm. Recall that the Newton method reads

$$\boldsymbol{q}_0^{k+1} = \boldsymbol{q}_0^k + \delta \boldsymbol{q} \tag{7}$$

where $\delta \boldsymbol{q}$ is the solution of the linear system of equation

$$\mathcal{J}(\boldsymbol{q}_0) \delta \boldsymbol{q} = -\mathcal{F}(\boldsymbol{q}_0) \tag{8}$$

The Jacobian $\mathcal{J}$ is already known: it the linearized Navier–Stokes equations (6) without the time dependence terms, and $\mathcal{F}(\boldsymbol{q}_0)$ is the variational formulation of the Navier–Stokes problem, *i.e.*

$$\begin{aligned} &\int_V (\boldsymbol{u}_0 \cdot \nabla)\, \boldsymbol{u}_0 \cdot \boldsymbol{w}\ \mathrm{d}V + \frac{\mathrm{Pr}}{\sqrt{\mathrm{Ra}}} \int_V \nabla \boldsymbol{u}_0 \cdot \nabla \boldsymbol{w}\ \mathrm{d}V \\ &-\mathrm{Pr} \int_V T_0 \boldsymbol{e}_y \cdot \boldsymbol{w}\ \mathrm{d}V - \mathrm{Pr} \int_V p_0\ \mathrm{div} \boldsymbol{w}\ \mathrm{d}V - \mathrm{Pr} \int_V q\ \mathrm{div} \boldsymbol{u}_0\ \mathrm{d}V \\ &+ \int_V (\boldsymbol{u}_0 \cdot \nabla)\, \boldsymbol{T}_0 \cdot \theta\ \mathrm{d}V + \frac{1}{\sqrt{\mathrm{Ra}}} \int_V \nabla \boldsymbol{T}_0 \cdot \nabla \theta\ \mathrm{d}V = 0 \end{aligned} \tag{9}$$

The general algorithm is then

1. solve the linear problem (8) to get $\delta \boldsymbol{q}$
2. evaluate $\epsilon_N = ||\delta \boldsymbol{q}||_{L^2}$
3. increment $\boldsymbol{q}_0$:
$$\boldsymbol{q}_0^{k+1} = \boldsymbol{q}_0^k + \alpha \delta \boldsymbol{q}_0 \tag{10}$$
where $\alpha \in [0, 1]$ is a relaxation parameter: if $\alpha = 1$ this amount to the normal Newton method, whereas a very low but non zero value will diminish the impact the of method (lower convergence rate, but potentially more stable).
4. if $\epsilon_N < tol$ stop, otherewise go back to 1, where tol is a chosen absolute tolerance, typically 10^{-6}.

1.3 Linear Stability Analysis

In this section we want, like in the UE to quantify how stable is the flow. To that end we decompose the perturbation of Eq. 5 in normal modes:

$$\boldsymbol{q}' = \sum_{i=1}^{\infty} \hat{\boldsymbol{q}}_i \mathrm{e}^{\lambda t} + \text{complex conjugate} \tag{11}$$

where the $\hat{\boldsymbol{q}}_i$ denotes the ith mode associated with the growth rate λ_i. Injecting this decomposition in the linearized Navier–Stokes equations gives an eigenvalue problem

$$\mathcal{J}(\boldsymbol{q}_0)\, \hat{\boldsymbol{q}}_i = -\mathbf{M}\, \hat{\boldsymbol{q}}_i \tag{12}$$

where $\mathcal{J}(\boldsymbol{q}_0)\, \hat{\boldsymbol{q}}_i$ is the same matrix as in the first part on Newton method, $\mathbf{M}$ is a mass matrix corresponding to the variable that are time dependent, *i.e.* all except the pressure

$$-\mathbf{M}\, \hat{\boldsymbol{q}}_i = -\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{u}}_i \\ \hat{p}_i \\ \hat{T} \end{pmatrix} \tag{13}$$

The variational form associated to this matrix vector product is then nothing else than

$$-\int_V \hat{\boldsymbol{u}}_i \cdot \boldsymbol{w} + \hat{T}_i\, \theta \, \mathrm{d}V \tag{14}$$

2 Coding Aspects

2.1 Mesh Generation

Using the codes given in the UE (3rd one with FE), implement the function `generate_mesh(N)` that generates a 20×20 unitary mesh, and then refine it at 5%, 1%, and 0.5% of the cavity length away from the wall. Then plot the mesh with the command `plot(mesh)`.

Table 1: Critical Rayleigh numbers for Prandtl numbers Pr = 0.71,Pr = 0.015 (water and mercury).

Method	mesh	Pr	Ra_c
Finite Difference second order in time	stretched 80×80	0.71	2.101×10^6
Spectral Method	20×20	0.71	2.110×10^6
Spectral Method	30×30	0.71	2.108×10^6
Spectral Method	20×20	0.015	44580
Spectral Method	30×30	0.015	40695

2.2 Newton Method

1. Complete the function called `Jacobian_variational_formulation` . It should output the variational form of the **Linearized** Navier–Stokes equations.

2. Complete the function called `NavierStokes`. It should out the variational form of the Navier–Stokes equations.

3. Complete the function called `solve_newton_step`, which call the two function implemented in the first and second steps, then solve the linear problem (8) and add the increment to the solution vector (eventually with a relaxation).

4. Complete the function called `solve_newton`, which calls the function implemented in the third step, computes the ϵ_N and loops while it is larger than 10^{-6}

To reach high Rayleigh number values (Pr = 0.71, Ra $> 10^5$) one needs to ramp up progressively: for that case you can for instance run Ra $= [10^4, 10^5, 10^6, 2 \times 10^6]$.

2.3 Linear Stability Analysis

1. Complete the function called `Compute_eigenvalues`: you have to write the variational forms and one of them has already been implemented in 1.

2. The stability boundary is characterised by having the largest eigenvalue having a zero real part. Using `N = 20`, `order = 2` as parameters, bracket the critical Rayleigh number Ra_c at which the eigenvalue is zero up to the second digit, for Pr = 0.71 and Pr = 0.015, and compare with the result from litterature which are summed up in the table 1.

3. How does the stability boundary changes with increase of the polynomial order?

4. Compare with the results you had in the first project (The case with the higher Prandtl number can still have a mismatch).

2.4 General questions

1. Is the implementation stable is the polynomial order for pressure is the same as the polynomial order for the velocity? What is the condition for numerical stability called?

2. On which functional space do the velocity live?

3. Which other kind of algorithm could we have used to compute the Steady flow? Give an example.

4. How is the time required by your computer evolving as the polynomial order increases?

5. Was it longer with you Finite Difference code to see whether the flow was stable/instable for a given Rayleigh number?

6. Do you think it would be possible to solve the eigenvalue problem for a 3D flow with `scipy`'s `eigs` (Do not do it!)?