Centre Tecnològic de Transferència de Calor
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Part 6 : Numerical solution of the Navier-Stokes Equations

CTTC-UPC

Version date : April 2022

In this part, a way of numerically solving the discrete Navier-Stokes equations called the Fractional Step Method is presented. First some details on the mathematical background are given, then the algorithm is presented in detail step by step. The method presented here derives from a more general to a very simplified academic version, however the same steps can be used to derive the equations that approach cases with a more complex geometry. Finally the Lid Driven Cavity benchmark case is presented.

6.1 Theoretical introduction

Navier-Stokes equations can describe the movement of some common fluids, we will focus in incompressible fluids, like water or air at low speeds ($v < 100\ m/s$). The first one basically says that mass is conserved, if density is constant then the mass that enters has to be the same one that goes out, we don't have mass sources and sinks.

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

The second one is a fancy way of writing Newton's Second Law $\boldsymbol{F} = m\boldsymbol{a}$. Remember that momentum is $\boldsymbol{p} = m\boldsymbol{v}$, if we derive it over time, we get the force. So the rate of change of momentum over time in a system (momentum that enters, that exits, and internal change) is related to the forces of that system (viscous forces, pressure forces, buoyancy etc). If we consider that free convection is negligible (If not, we should add the boussinesq approximation $-\rho\beta(T - T_\infty)\boldsymbol{g}$ at the left hand side of the equation)

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} = -\nabla P - \rho(\boldsymbol{v}\cdot\nabla)\boldsymbol{v} + \mu\nabla^2\boldsymbol{v}$$

We can have a third simplified energy equation that gives the temperature field.

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

6.2 Fractional Step Method

We want to solve the velocity field from the momentum equation, we can see we have a convection-difussion equation, and we know how to solve them, right? Well we have a problem, the pressure gradient at the source term $-\nabla P$ is linked to the velocity field and viceversa, so is not as straightforward as it seemed. We need to decouple the pressure gradient.

6.2.1 Helmholtz-Hodge theorem

We will use the Helmholtz-Hodge theorem that allows us to decompose a vectorial field $\boldsymbol{\omega}$ in a vector $\boldsymbol{A}$ that is divergence-free ($\nabla\cdot\boldsymbol{A} = 0$), and the gradient of an scalar field $\nabla\phi$

$$\boldsymbol{\omega} = \boldsymbol{A} + \nabla\phi$$

Notice that the velocity $\boldsymbol{v}$ is a divergence-free vector and the pressure gradient ∇P is the gradient of an scalar field. Now what would our $\boldsymbol{\omega}$ be?

6.2.2 Temporal discretization

First for simplicity we will call the convective and diffusive terms of the equation as $\boldsymbol{R}(\boldsymbol{v})$

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} = -\nabla P \underbrace{-\rho(\boldsymbol{v}\cdot\nabla)\boldsymbol{v} + \mu\nabla^2\boldsymbol{v}}_{\boldsymbol{R}(\boldsymbol{v})}$$

Then we will discretize the equation in time, for simplicity $\boldsymbol{R}(\boldsymbol{v})$ will be discretized as using a forward Euler method and the pressure term will be discretized implicitly

$$\rho\frac{\boldsymbol{v}^{n+1} - \boldsymbol{v}^n}{\Delta t} = -\nabla P^{n+1} + \boldsymbol{R}(\boldsymbol{v}^n) \tag{1}$$

⚠ We used a forward Euler to make this notes simpler but it's very uncommon to use this approach. In general a better approach is use an Adams-Bashforth method (also explicit), where the convective schemes are evaluated at $n+1/2$ using an interpolation from the previous and pre-previous timesteps

$$\rho\frac{\boldsymbol{v}^{n+1} - \boldsymbol{v}^n}{\Delta t} = -\nabla P^{n+1} + \frac{3}{2}\boldsymbol{R}(\boldsymbol{v}^n) - \frac{1}{2}\boldsymbol{R}(\boldsymbol{v}^{n-1})$$

Now is when we will use the previously mentioned theorem, we define a predictor velocity $\boldsymbol{v}^p$ as

$$\boldsymbol{v}^p \equiv \boldsymbol{v}^{n+1} + \frac{\Delta t}{\rho}\nabla P^{n+1} \tag{2}$$

This predictor velocity is used as an intermediate step, don't worry too much about its physical meaning. Here is where the magic begins. First we isolate v^{n+1} from (2)

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^p - \frac{\Delta t}{\rho}\nabla P^{n+1} \tag{3}$$

Then we substitute the result in (1). The pressure gradient disappears and we get

$$\boldsymbol{v}^p = \boldsymbol{v}^n + \frac{\Delta t}{\rho}\boldsymbol{R}(\boldsymbol{v}^n)$$

So what do we have here? We can get this predictor velocity $\boldsymbol{v}^p$ directly with previous timesteps. Now let's apply the divergence operator at both sides of Equation (2)

$$\nabla \cdot \boldsymbol{v}^p = \cancel{\nabla \cdot \boldsymbol{v}^{n+1}} + \frac{\Delta t}{\rho}\nabla \cdot \nabla P^{n+1}$$

So we get a Poisson equation (Remember that $\nabla \cdot \nabla\phi = \nabla^2\phi$), where we can get the pressure at the next timestep P^{n+1} and with it, from (3) we can get the velocity at the next timestep $\boldsymbol{v}^{n+1}$ So from these equations we can write our 3-step algorithm.

6.3 Temporal discretization summary

The algorithm can be simplified as calculate a predictor velocity in an explicit way using (4), then solving the pressure field at P^{n+1} using (5) (Note, we will need to solve a linear system in this step, for example using a Gauss-Seidel solver), and finally calculating the velocity at the next timestep with (6). Then repeat every timestep.

Step 1: $$\boldsymbol{v}^p = \boldsymbol{v}^n + \frac{\Delta t}{\rho}\boldsymbol{R}(\boldsymbol{v}^n) \tag{4}$$

Step 2: $$\nabla^2 P^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \boldsymbol{v}^p \tag{5}$$

Step 3: $$\boldsymbol{v}^{n+1} = \boldsymbol{v}^p - \frac{\Delta t}{\rho}\nabla P^{n+1} \tag{6}$$

6.4 Staggered meshes

Due to numerical limitations, evaluating the 3 quantities u, v, P in the same mesh can be unstable. If we evaluate the pressure gradient, its value does not depend on the value of the pressure at the cell. Let's look at a 1D Case.

$$u^{n+1} = u^p - \frac{\Delta t}{\rho}\frac{\partial P^{n+1}}{\partial x} \simeq u^p - \frac{\Delta t}{\rho}\frac{P_E^{n+1} - P_W^{n+1}}{2\Delta x}$$

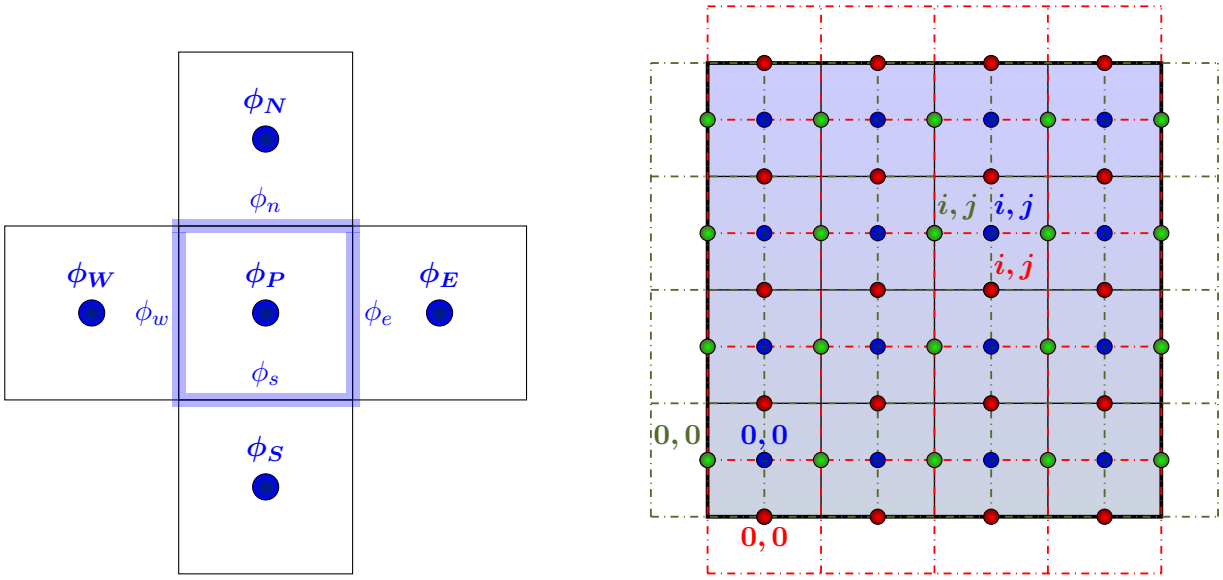
We could get physically incoherent pressure distributions that the equations see as valid For example a pressure field: $P = (..., 100, 0, 100, 0, 100, 0, ...)$ gives that $\nabla P^{n+1} = 0$ at all the points although we know is not a valid field. This problem is the one known as the checkerboard problem, because the pressure field can ressamble a checkerboard.

In order to avoid this problem, we segregate the meshes in 3 different ones (Staggered meshes). This three meshes are shown in Figure 1b.

- The pressure mesh (P) (blue) will have its nodes and control volumes at the center, and its dimension will be $P[N_x][N_y]$.
- The velocity mesh for x (u) (green) will have its nodes at the horizontal faces of mesh P and will have dimensions $u[N_x + 1][N_y]$.

- The velocity mesh for y (v) (red) will have its nodes at the vertical faces of mesh P and will have dimensions $v[N_x][N_y + 1]$

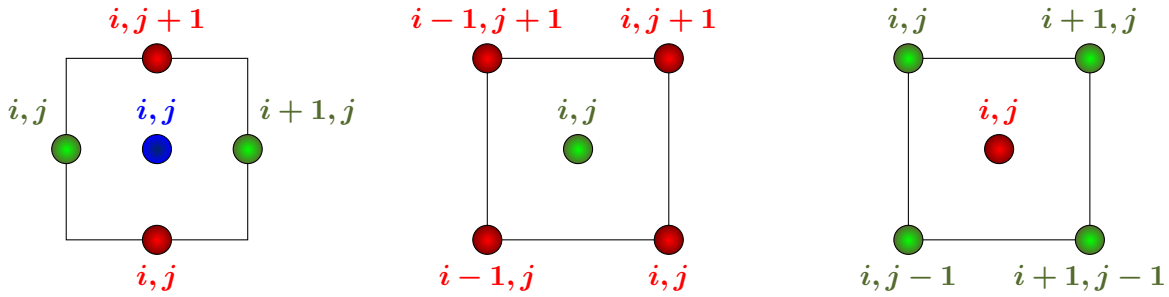
Capital letters N, S, E, W (Nord, Sud, Est, West) reffere to the neighbour nodes, P is the current node and lower case letters n, s, e, w are the magnitudes at the cell face. Neighbour nodes can also be refered with index: $\phi_{i,j}$ for the current node, and neighbour nodes as i.e $\phi_E = \phi_{i+1,j}$.



(a) Mesh notation for a magnitude ϕ. *(b)* 3 different meshes (P blue, u green, v red).

Figure 1: Staggered mesh geometry.

In Figure 2 we show a possible way on how to refer the index relatively to the other ones when we are in a certain mesh, i.e. if we are in the pressure mesh and we want the value of u at the east face, it will be $u_{i+1,j}$ and at the west face it will be $u_{i,j}$ (not $u_{i-1,j}$)



(a) Mesh P. *(b)* Mesh u. *(c)* Mesh v.

Figure 2: Relative index of surrounding nodes for different meshes.

6.5 Spatial discretization

⚠ For simplicity we will assume that we have a uniform 2D mesh ($\Delta x = \Delta y = \text{cte}\ \forall$ mesh), with constant ρ and μ, just so we can have easier equations, cancel some terms, etc. Note that if you want to use a non uniform mesh you could derive the equations the same way as presented here, but without the simplifications

6.5.1 Convective and Diffusive term discretization

We have already seen the discretization for the convective and difussive term, the discretization will be almost identical. We will evaluate the $\boldsymbol{R}(\boldsymbol{v})$ term discretizing it using finite volumes. This term is

a vector with 3 components, we will see the steps for the x component, for the other components the process is the same. First we integrate the equations over a control volume $\mathcal{V}$ placed in the u mesh and apply the divergence theorem at the convective and difusive terms.

$$\int_{\mathcal{V}} \rho \frac{\boldsymbol{v}^p - \boldsymbol{v}^n}{\Delta t}\, d\mathcal{V} = \underbrace{\int_{\mathcal{V}} \mu \nabla^2 \boldsymbol{v}\, d\mathcal{V} - \int_{\mathcal{V}} \rho(\boldsymbol{v}\cdot\nabla)\boldsymbol{v}\, d\mathcal{V}}_{\int_{\mathcal{V}} \boldsymbol{R}(\boldsymbol{v}^n)\, d\mathcal{V}}$$

For the **diffusive term** (remember that $\nabla^2 \boldsymbol{v} = \nabla\cdot\nabla\boldsymbol{v}$), in the case of the x component of the velocity (u), if we consider a constant viscosity μ:

$$\int_{\mathcal{V}} \mu\nabla^2 u\, d\mathcal{V} = \int_{\mathcal{V}} \mu\nabla\cdot\nabla u\, d\mathcal{V} = \int_S \mu\nabla u\cdot d\boldsymbol{S} \simeq \mu \sum_{nb} \frac{u_{nb} - u_p}{d_{nb}}\, \Delta S$$

Because of the uniform 2D mesh we can simplify $\Delta S/d_{nb} = 1$ so:

$$\text{diff}_x = \mu \sum_{nb} \frac{u_{nb} - u_p}{d_{nb}}\, \Delta S = \mu\left((u_N - u_P) + (u_S - u_P) + (u_E - u_P) + (u_W - u_P)\right)$$

For the **convective term**, first we will introduce a more general way of writing Navier-Stokes equations in terms of the outer product, which has the following identity:

$$-\nabla\cdot(\boldsymbol{v}\otimes\boldsymbol{v}) = \boldsymbol{v}(\nabla\cdot\boldsymbol{v}) + (\boldsymbol{v}\cdot\nabla)\boldsymbol{v}$$

where $(\boldsymbol{v}\otimes\boldsymbol{v})$ is the outter product and $u\ v\ w$ are the $x\ y\ z$ components of the velocity vector $\boldsymbol{v}$:

$$(\boldsymbol{v}\otimes\boldsymbol{v}) = \begin{pmatrix} uu & uv & uw \\ vu & vv & vw \\ wu & wv & ww \end{pmatrix}$$

We can see that the first term of the identity is zero in case of incompressible flows ($\nabla\cdot\boldsymbol{v} = 0$), and the second term is the one we have for the momentum equation, so both terms are equivalent. However we will use the general form because it will be more convenient when we apply the divergence theorem.

$$-\rho\int_{\mathcal{V}} \nabla\cdot(\boldsymbol{v}\otimes\boldsymbol{v})\, d\mathcal{V} = -\rho\int_S \boldsymbol{v}\otimes\boldsymbol{v}\cdot d\boldsymbol{S}$$

For the x component, considering that z velocity is zero ($w = 0$), and uniform 2D mesh we have that:

$$\text{conv}_x = -\rho\int_{\mathcal{V}} (\boldsymbol{v}\cdot\nabla)u\, d\mathcal{V} = -\rho\Delta x(v_n u_n - v_s u_s + u_e u_e - u_w u_w)$$

Remember the convection-difussion discretization of the previous part, we transported a property ϕ that now will be $\phi \equiv u$. For the north and south faces, this property was transported by the v velocity, while for the east and west faces this property was transported by the u velocity, hence the combination of u and v we have in this equation. Now we can write the equation for the predictor velocity.

$$\rho\int_{\mathcal{V}} \frac{u^p - u^n}{\Delta t}\, d\mathcal{V} = \int_{\mathcal{V}} R(u^n)\, d\mathcal{V} \qquad \rightarrow \qquad u^p = u^n + \frac{\Delta t}{\rho\Delta\mathcal{V}}\left(-\text{conv}_x + \text{diff}_x\right)^n$$

The integration for the other component y is almost the same.

6.5.2 Pressure discretization

To discretize the Poisson equation we do it the following way. If we consider a non-boundary node, and a uniform 2D mesh:

$$\int_{\mathcal{V}} \nabla \cdot \nabla P \; d\mathcal{V} = \int_{\mathcal{V}} \frac{\rho}{\Delta t} \nabla \cdot \boldsymbol{v}^p \; d\mathcal{V}$$

$$\int_S \nabla P \cdot d\boldsymbol{S} = \frac{\rho}{\Delta t} \int_S \boldsymbol{v}^p \cdot d\boldsymbol{S}$$

$$\sum \frac{P_{nb} - P_P}{d_{nb}} \; \Delta S = \frac{\rho}{\Delta t} \sum \boldsymbol{v}^p_{nb} \cdot \Delta \boldsymbol{S}$$

$$P_N + P_S + P_E + P_W - 4P_P = \frac{\rho \Delta S}{\Delta t} (v^p_n - v^p_s + u^p_e - u^p_w)$$

6.6 Step by step algorithm

We have seen the how we and up with the discrete equations for a 2D uniform mesh. Now we will do a summary of the steps you would have to follow to solve the case:

⚠ Here we used a CDS convective scheme for the mass transport terms ($\rho \boldsymbol{v} \cdot \boldsymbol{S}$) as an example and to simplify, however we could have used other schemes. Also note that some terms have 1/2 not because a convective scheme is used, but because we do an interpolation of values from the vertex to the center of the face.

6.6.1 Step 1a Calculation of u^p (in mesh u^x)

The term $R(u)$ integrated in a control volume $\mathcal{V}$ gives:

$$R(u) = -\rho \Delta x (v_n u_n - v_s u_s + u_e u_e - u_w u_w) + \mu (u_N + u_S + u_E + u_W - 4u_P) \tag{7}$$

Where u is the x component of the velocity, v is the y component. To evaluate the face values, we will use interpolations for the v values and a CDS convective scheme for the u values. Using the mesh index shown in Figure 2b we have that:

$$u_E = u_{i+1,j} \qquad u_W = u_{i-1,j} \qquad u_N = u_{i,j+1} \qquad u_S = u_{i,j-1}$$
$$u_e = \frac{1}{2}(u_{i+1,j} + u_{i,j}) \qquad u_w = \frac{1}{2}(u_{i-1,j} + u_{i,j}) \qquad u_n = \frac{1}{2}(u_{i,j+1} + u_{i,j}) \qquad u_s = \frac{1}{2}(u_{i,j-1} + u_{i,j})$$
$$v_n = \frac{1}{2}(v_{i-1,j+1} + v_{i,j+1}) \qquad v_s = \frac{1}{2}(v_{i-1,j} + v_{i,j})$$

With all the data, the can calculate the x component of the predictor velocity with Equation (4) integrated over a control volume $\mathcal{V} = (\Delta x)^2$.

$$u^p = u^n + \frac{\Delta t}{\rho (\Delta x)^2} R(u^n) \tag{8}$$

6.6.2 Step 1b Calculation of v^p (in mesh v_y)

The same way we have done in section 6.6.1, we have that:

$$R(v) = -\rho \Delta x (v_n v_n - v_s v_s + u_e v_e - u_w v_w) + \mu (v_N + v_S + v_E + v_W - 4v_P) \tag{9}$$

Using the index of Figure 2c:

$$v_E = v_{i+1,j} \qquad v_W = v_{i-1,j} \qquad v_N = v_{i,j+1} \qquad v_S = v_{i,j-1}$$
$$v_e = \frac{1}{2}(v_{i+1,j} + v_{i,j}) \qquad v_w = \frac{1}{2}(v_{i-1,j} + v_{i,j}) \qquad v_n = \frac{1}{2}(v_{i,j+1} + v_{i,j}) \qquad v_s = \frac{1}{2}(v_{i,j-1} + v_{i,j})$$
$$u_w = \frac{1}{2}(u_{i,j} + u_{i,j-1}) \qquad u_e = \frac{1}{2}(u_{i+1,j} + u_{i+1,j-1})$$

The y component of Equation (4) once spatially discretized is.

$$v^p = v^n + \frac{\Delta t}{\rho (\Delta x)^2} R(v^n) \tag{10}$$

6.6.3 Step 2 Calculation of P^{n+1} (in mesh P)

We have to solve the Poisson equation (5) implicitly, so we will need to solve a system of linear equations. We write the Poisson discrete equation isolating P_P ($P_{i,j}$) to get the equation we will use to solve the system with the Gauss-Seidel method (we could use other linear solver approaches):

$$P_{i,j}^{n+1} = \frac{1}{a_p} \left(a_n P_{i,j+1}^{n+1} + a_s P_{i,j-1}^{n+1} + a_e P_{i+1,j}^{n+1} + a_w P_{i-1,j}^{n+1} - \frac{\rho \Delta S}{\Delta t} (v_n^p - v_s^p + u_e^p - u_w^p) \right) \tag{11}$$

For the mesh distribution, as we can see for index of Figure 2a we can use the previously calculated predictor velocities as:

$$v_n^p = v_{i,j+1}^p \qquad v_s^p = v_{i,j}^p \qquad u_e^p = u_{i+1,j}^p \qquad u_w^p = u_{i,j}^p$$

If we consider the inner nodes (non-boundary nodes), then we have the following a_{nb} coefficients.

$$a_n = 1 \qquad a_s = 1 \qquad a_e = 1 \qquad a_w = 1 \qquad a_p = 4$$

In our cases, if we have nodes at the boundary (inlets, outlets, walls), for the pressure we will generally have a Neumann boundary condition $\partial P/\partial n = 0$ so the wall nodes will have some $a_{nb} = 0$ (one for edge nodes, two for vertex nodes) You can write this boundary conditions in a general way as:

$a_{nb} = 0$ If the wall is a boundary where $\nabla P = 0$

$a_{nb} = 1$ If the wall is not a boundary

$$a_p = \sum a_{nb} = a_n + a_s + a_e + a_w$$

6.6.4 Step 3, calculation of u^{n+1}

With Equation (6) we calculate the x and y components of velocity $\boldsymbol{v}$ for the next timestep. We can discretize the gradient with the distance difference. The pressure index is still coherent with the ones described in Figure 1.

$$u_{i,j}^{n+1} = u_{i,j}^p - \frac{\Delta t}{\rho}\frac{\partial P}{\partial x} = u^p - \frac{\Delta t}{\rho}\frac{P_{i,j} - P_{i-1,j}}{\Delta x}$$
$$v_{i,j}^{n+1} = v_{i,j}^p - \frac{\Delta t}{\rho}\frac{\partial P}{\partial y} = v^p - \frac{\Delta t}{\rho}\frac{P_{i,j} - P_{i,j-1}}{\Delta y}$$

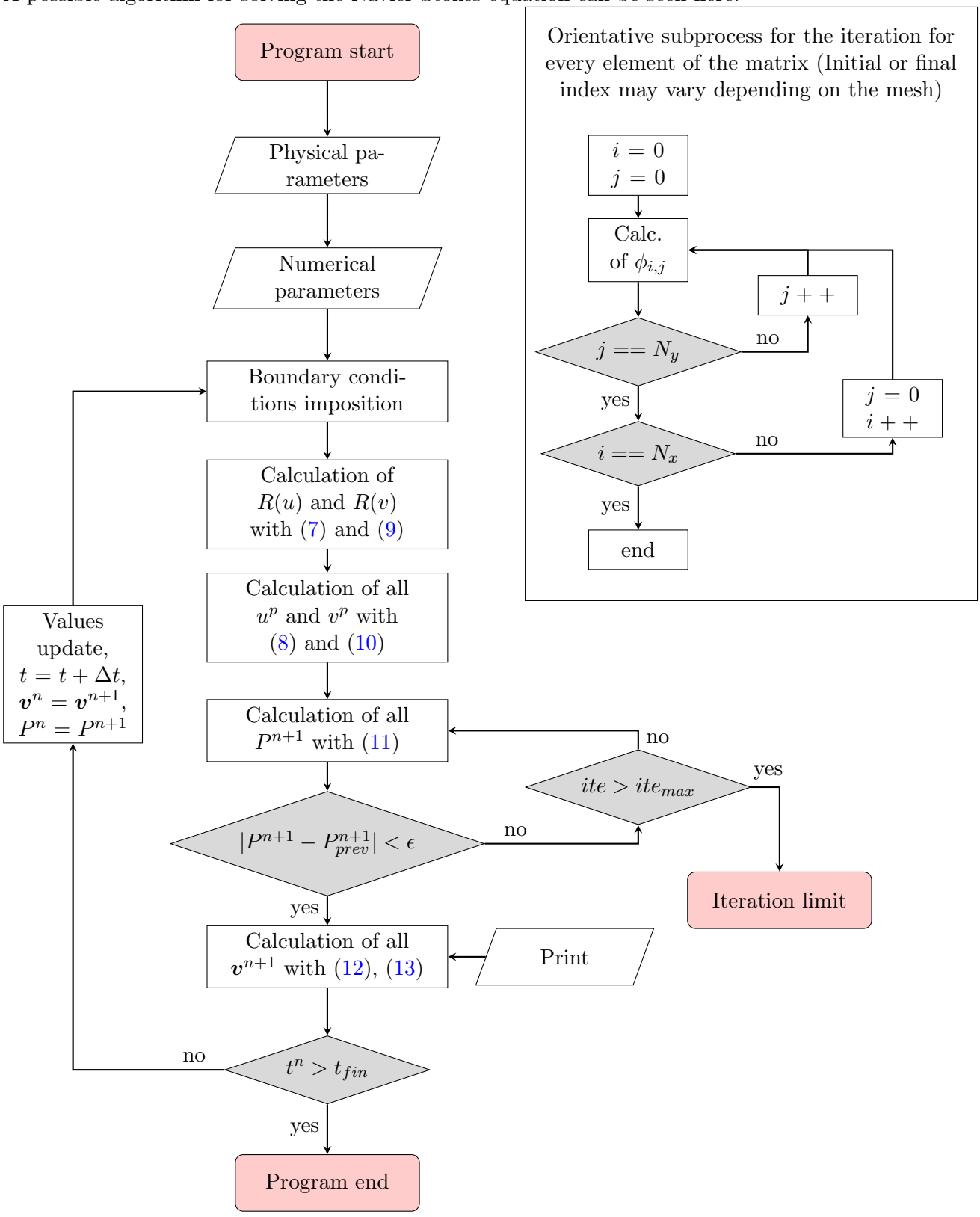
So the velocity at the next timestep will be:

$$u_{i,j}^{n+1} = u_{i,j}^p - \frac{\Delta t}{\rho}\frac{P_{i,j} - P_{i-1,j}}{\Delta x} \tag{12}$$
$$v_{i,j}^{n+1} = v_{i,j}^p - \frac{\Delta t}{\rho}\frac{P_{i,j} - P_{i,j-1}}{\Delta y} \tag{13}$$

Once we have the new velocities, we can update the time, return to section 6.6.1 for the new calculations and so on until we reach the final time.

6.7 Algorithm summary

A possible algorithm for solving the Navier-Stokes equation can be seen here:



6.8 Computing limitations and turbulence models

We have seen how to discretize and solve Navier-Stokes equations. As we saw in the previous parts, we might have some limitations with the Δt to ensure convergence. We can use a criteria known as the Courant-Friedrichs-Levy that says:

$$C_{conv} = \frac{|\boldsymbol{v}|\Delta t}{\Delta x} < 0.35 \qquad C_{diff} = \frac{\mu}{\rho}\frac{\Delta t}{\Delta x^2} < 0.2$$

Basically the convective term says that the fluid shouldn't cross an entire cell within the same timestep. Normally for security we take $C_{conv} < 0.35$. The diffusive term is similar to the criteria we saw in the 1D case. These criteria is not exact, it is just orientative. For example, we could have a bigger timestep and converge in one case, while in another case we could comply with the criteria but Δt may not be small enough so we would have to use a smaller one.

Apart from temporal discretization, we are also limited by the spatial discretization. For laminar cases (low Re numbers [1]) the spatial discretization can be coarser, but as the Re number grows and we enter turbulent regimes, the mesh has to be refined more and more. One possible approach to refine the mesh is using a non uniform one and focus on areas with steeper velocity gradients (For example walls) leaving coarser control volumes in places where we don't need much precision. However, even when using non-uniform meshes, computational cost still increases a lot as we refine the mesh.

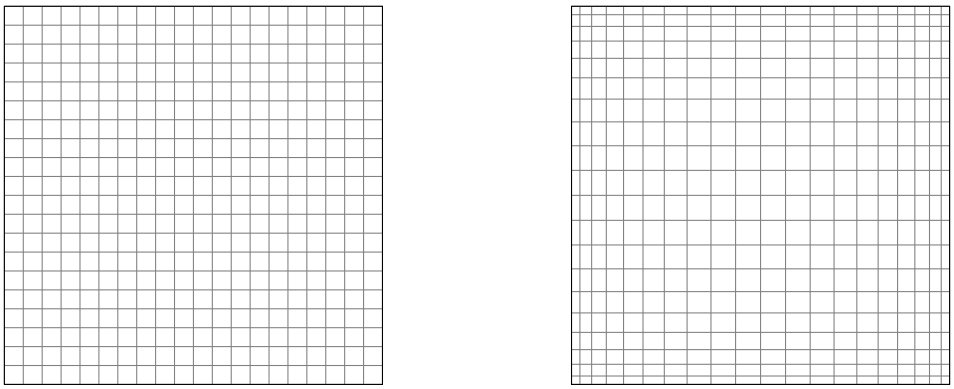


Figure 3: Uniform mesh (left) vs wall refined mesh (right), both 20×20

In addition, as the mesh is refined, our Δt needs to be reduced. Furthermore, turbulence is a 3D phenomena, we cannot simplify a case to a 2D one even if it is symmetric and has nice boundary conditions. In general the total computational cost of a case scales as $\sim Re^{11/4}$, so a case with a $Re = 100$ will take almost 600 times more computing resources as the same case with $Re = 10$. Some cases may be very expensive or even impossible to simulate because of computing limitations. Even a simple case like the temperature distribution of a classroom is currently impossible to solve this way. There are some approaches to tackle this issue, and are called turbulence models. By doing some mathematical approximations and modifying some equations, one can model a "similar" behaviour of the turbulence in a case and this allows us to use coarser meshes. You may have heard of the LES (Large Eddy Simulation) or RANS (Reynolds-averaged Navier-Stokes) turbulence models. If we use no turbulence model at all (What we would call a DNS (Direct Numerical Simulation) approach) and we solve the equations with a very refined mesh, we get more accurate results but it may be more costly or even unfeasible with today's current computing power, so we would need to use turbulence models.

The takeaway from this subsection is that mathematically, turbulence is a phenomena already described by N-S equations and also in their discrete version if we have a fine enough mesh, however when computing power needs exceeds our capabilities, we have to use turbulence models (That won't give results as accurate as the ones we would get if we had infinite computing power).

[1] Remember that the Reynolds number is a dimensionless number that characterizes the ratio of inertial forces respect to viscous ones in an specific case $Re = \frac{\rho v L}{\mu}$.

6.9 Lid Driven cavity

Exercise 6a (optional) : Lid Driven cavity case

We have a cavity of size $L \times L$ with a fluid inside, with viscosity μ and density ρ, as shown in Figure 4. The inferior, left and right walls are static ($u = 0$ and $v = 0$), while the top wall has a constant velocity of $u(x, y = L) = U_{ref}$. The pressure boundary condition for all walls is $\partial P/\partial n = 0$

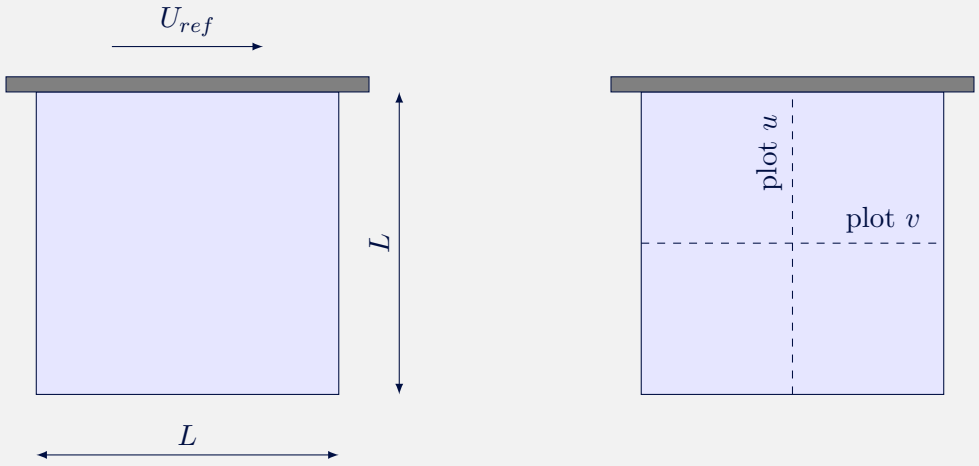


Figure 4: Lid driven cavity geometry and plot data

The case is characterized by the Reynolds number

$$Re = \frac{\rho U_{ref} L}{\mu}$$

More details can be seen at the additional material attached document.

a) Solve the case for $Re = 100$ and $Re = 400$ until steady state is reached.

b) Give the plots of $u(y)$ at $x = 0.5$ comparing them to the reference case.

c) Give the plots of $v(x)$ at $y = 0.5$ comparing them to the reference case.

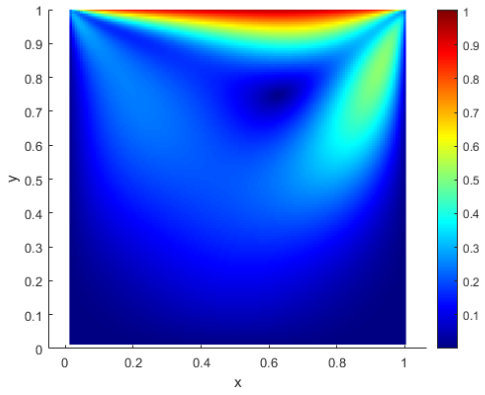
d) Plot the color maps of the velocity magnitude.

6.9.1 Hints for the Lid Driven cavity case

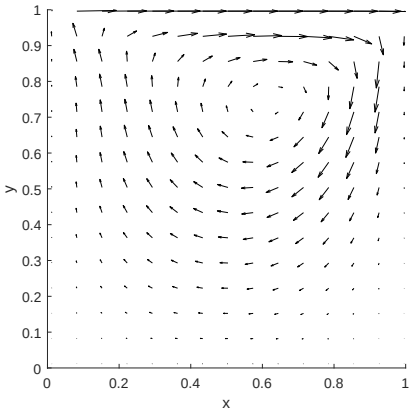
For comparing the results, you can use the reference data [2] shown in Tables 1 and 2. We plot the velocity u at $x = L/2$ as a function of y and the velocity v at $y = L/2$ as a function of x.
As orientative parameters, in this case for $Re = 100$ I used a 128×128 mesh, with a $\Delta t = 0.001\ s$. The solver for the Poisson equation was a Gauss-Seidel solver. If the initial velocity is $\boldsymbol{v} = 0$, then steady state is reached around $t = 30\ s$, the computing time it took was more or less 3 minutes without any optimization, so it can be much faster.
Things that you may need to consider:

- The pressure field solution is relative because we only solve the gradient, so $\nabla(P + \text{cte}) = \nabla P$.
- Some mesh nodes may not be perfectly coincident with the wall boundary conditions (For example the upper wall and U_{ref} at the u mesh). You could approach this with interpolations for more precision, but it is ∼OK to consider for example $u(i, N) \approx U_{ref}$
- Be careful with mesh index and sizes, the u mesh is an $(N + 1) \times N$ mesh, the v mesh is $N \times (N + 1)$ and the pressure mesh is $N \times N$
- Be careful when using non-existing neighbour data if you are in a boundary, you might need to use some conditionals.

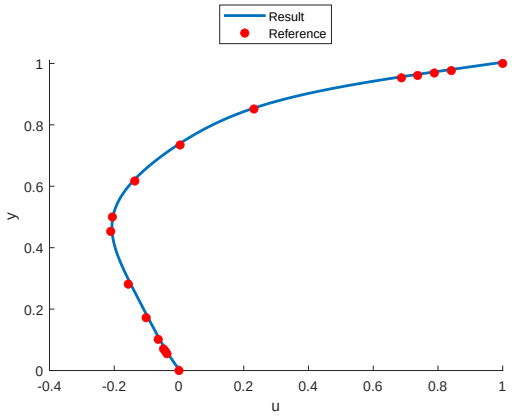
Here i show some of the results i got for the $Re = 100$ case so you can compare



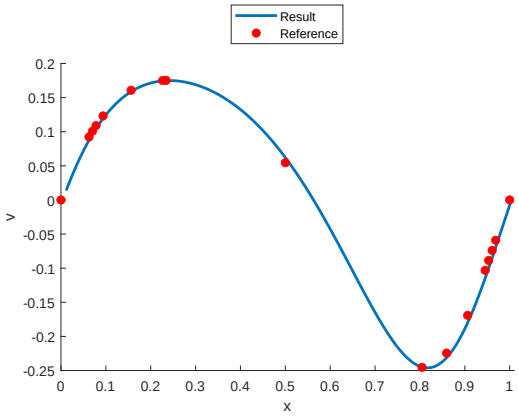
(a) Velocity magnitude colormap



(b) Velocity direction



(c) $u(y)$ velocity at $x = L/2$



(d) $v(x)$ velocity at $y = L/2$

Figure 5: Results for $Re = 100$

[2] GHIA, U. K. N. G.; GHIA, Kirti N.; SHIN, C. T. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. Journal of computational physics, 1982, 48.3: 387-411.

Reference data (Guia et. al) of $u(y)$ velocity at $x = L/2$ and $v(x)$ velocity at $y = L/2$

Table 1: Values of $u(y)$ at $x = L/2$ for different Re numbers

y/L	$Re = 100$	$Re = 400$	$Re = 1000$	$Re = 3200$	$Re = 5000$	$Re = 7500$	$Re = 10000$
1.0000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
0.9766	0.84123	0.75837	0.65928	0.53236	0.48223	0.47244	0.47221
0.9688	0.78871	0.68439	0.57492	0.48296	0.46120	0.47048	0.47783
0.9609	0.73722	0.61756	0.51117	0.46547	0.45992	0.47323	0.48070
0.9531	0.68717	0.55892	0.46604	0.46101	0.46036	0.47167	0.47804
0.8516	0.23151	0.29093	0.33304	0.34682	0.33556	0.34228	0.34635
0.7344	0.00332	0.16256	0.18719	0.19791	0.20087	0.20591	0.20673
0.6172	-0.13641	0.02135	0.05702	0.07156	0.08183	0.08342	0.08344
0.5000	-0.20581	-0.11477	-0.06080	-0.04272	-0.03039	-0.03800	0.03111
0.4531	-0.21090	-0.17119	-0.10648	-0.08664	-0.07404	-0.07503	-0.07540
0.2813	-0.15662	-0.32726	-0.27805	-0.24427	-0.22855	-0.23176	-0.23186
0.1719	-0.10150	-0.24299	-0.38289	-0.34323	-0.33050	-0.32393	-0.32709
0.1016	-0.06434	-0.14612	-0.29730	-0.41933	-0.40435	-0.38324	-0.38000
0.0703	-0.04775	-0.10338	-0.22220	-0.37827	-0.43643	-0.43025	-0.41657
0.0625	-0.04192	-0.09266	-0.20196	-0.35344	-0.42901	-0.43590	-0.42537
0.0547	-0.03717	-0.08186	-0.18109	-0.32407	-0.41165	-0.43154	-0.42735
0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Table 2: Values of $v(x)$ at $y = L/2$ for different Re numbers

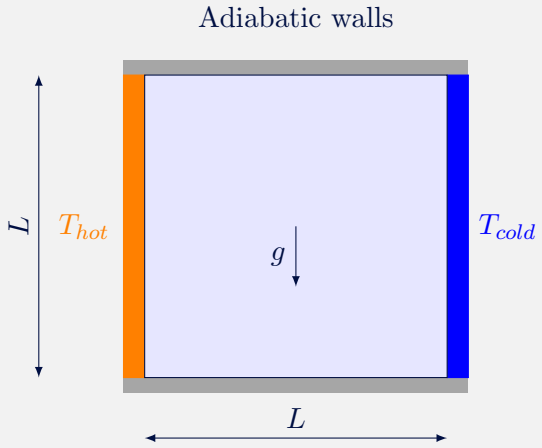
x/L	$Re = 100$	$Re = 400$	$Re = 1000$	$Re = 3200$	$Re = 5000$	$Re = 7500$	$Re = 10000$
0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.0625	0.09233	0.18360	0.27485	0.39560	0.42447	0.43979	0.43983
0.0703	0.10091	0.19713	0.29012	0.40917	0.43329	0.44030	0.43733
0.0781	0.10890	0.20920	0.30353	0.41906	0.43648	0.43564	0.43124
0.0938	0.12317	0.22965	0.32627	0.42768	0.42951	0.41824	0.41487
0.1563	0.16077	0.28124	0.37095	0.37119	0.35368	0.35060	0.35070
0.2266	0.17507	0.30203	0.33075	0.29030	0.28066	0.28117	0.28003
0.2344	0.17527	0.30174	0.32235	0.28188	0.27280	0.27348	0.27224
0.5000	0.05454	0.05186	0.02526	0.00999	0.00945	0.00824	0.00831
0.8047	-0.24533	-0.38598	-0.31966	-0.31184	-0.30018	-0.30448	-0.30719
0.8594	-0.22445	-0.44993	-0.42665	-0.37401	-0.36214	-0.36213	-0.36737
0.9063	-0.16914	-0.23827	-0.51550	-0.44307	-0.41442	-0.41050	-0.41496
0.9453	-0.10313	-0.22847	-0.39188	-0.54053	-0.52876	-0.48590	-0.45863
0.9531	-0.08864	-0.19254	-0.33714	-0.52357	-0.55408	-0.52347	-0.49099
0.9609	-0.07391	-0.15663	-0.27669	-0.47425	-0.55069	-0.55216	-0.52987
0.9688	-0.05906	-0.12146	-0.21388	-0.39017	-0.49774	-0.53858	-0.54302
1.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

6.10 Differentially heated cavity

This case will consist on solving a natural (free) convection case. As a previous step you will need to add the Boussinesq term to the differential momentum equation, modify the discrete equations with this new term and add the solution of the energy equation to the algorithm. You can check the results of this case on the reference paper [3]

Exercise 6b (optional) : Differentially heated cavity

We have a fluid enclosed in a cavity of size $L \times L$ with a hot and a cold wall. All walls are static and the fluid inside moves because of natural convection. All the properties of the fluid and the environment are known.



Propperty	symbol	Units
Specific heat	c_p	J/kgK
Dynamic viscosity	μ	kg/ms
Density at T_∞	ρ	kg/m^3
Reference temperature	T_∞	K
Th. conductivity	λ	W/mK
Th. expansion coeff.	β	$1/K$
Gravity	g	m/s^2

All four walls of the domain are static ($u = 0$ and $v = 0$), the pressure boundary condition for all walls is $\partial P/\partial n = 0$. The top and bottom walls are adiabatic ($\nabla T = 0$), the left wall is at a temperature T_{hot} and the right wall is at a temperature T_{cold}. The case is characterized by the Rayleigh number and the Prandtl number:

$$Ra = \frac{c_p\ g\ \beta\ \rho^2\ (T_{hot} - T_{cold})\ L^3}{\mu\ \lambda} \qquad Pr = \frac{\mu\ c_p}{\lambda}$$

More details on this case can be seen at the reference paper. Chose a Prandtl number of $Pr = 0.71$ (air) and compute the case until steady state is reached for $Ra = 10^3$, $Ra = 10^4$, $Ra = 10^5$ and $Ra = 10^6$. Show the following results, comparing them to the reference if possible.

a) Plot the isotherms and a colormap for the temperature T and u, v velocities

b) Calculate the average Nusselt number Nu.

c) Plot $u(y)$ at $x = 0.5$ and $v(x)$ at $y = 0.5$.

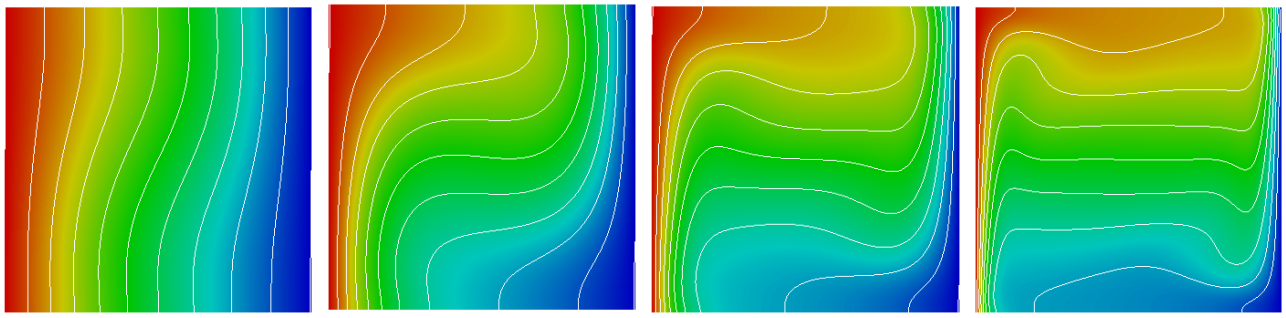
Note: You might need to use dimensionless variables (u^*, v^*, T^*) for comparing to the reference paper.

[3] DE VAHL DAVIS, G. Natural convection of air in a square cavity: a bench mark numerical solution. International Journal for numerical methods in fluids, 1983, vol. 3, no 3, p. 249-264.

6.10.1 Hints for the Diferentially Heated Cavity case

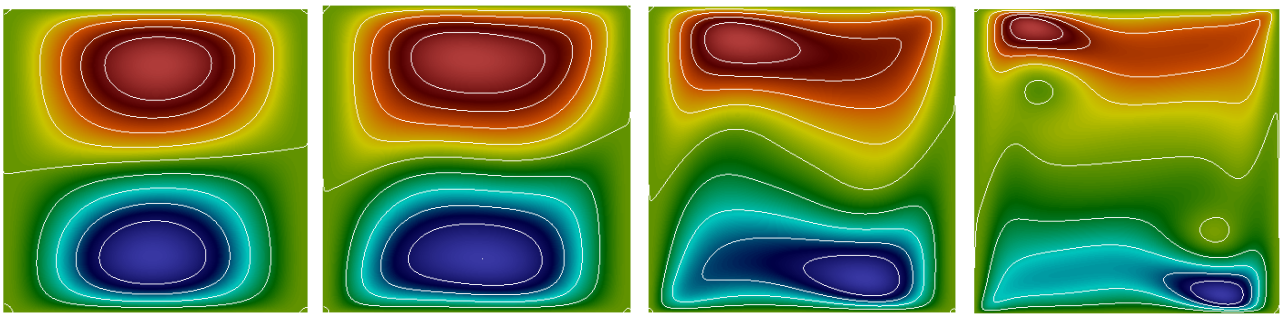
You can take $T_{hot} = 1$, $T_{cold} = 0$, $T_\infty = 0.5$, $c_p = 0.71$, the rest variables to 1, and isolate and impose gravity as a function of Ra. For comparing the results you can use the reference data. Some of it is summarized here.

		Max u^* velocity (at $x = 0.5L$)		Max v^* velocity (at $y = 0.5L$)	
Ra	$\overline{Nu}$	u^*_{max}	y/L	v^*_{max}	x/L
10^3	1.118	3.649	0.813	3.697	0.178
10^4	2.243	16.178	0.823	19.617	0.119
10^5	4.519	34.73	0.855	68.59	0.066
10^6	8.800	64.63	0.850	219.36	0.0379



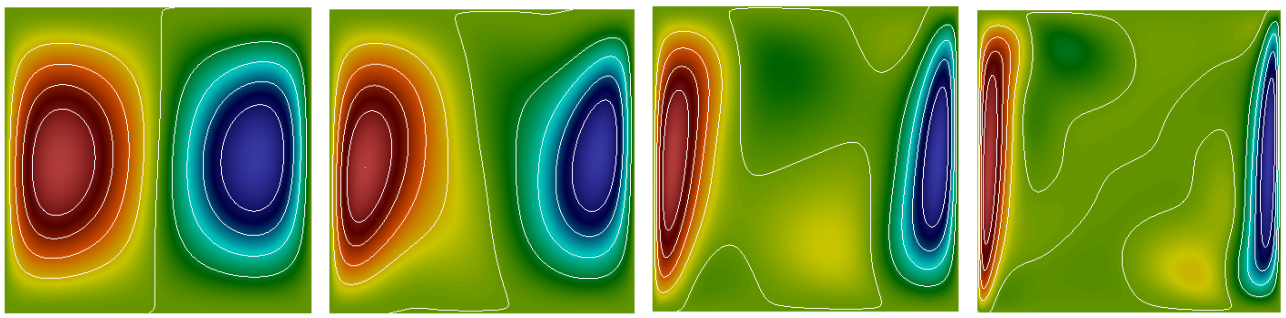
(a) $Ra = 10^3$ *(b)* $Ra = 10^4$ *(c)* $Ra = 10^5$ *(d)* $Ra = 10^6$

Figure 7: Temperature maps for different Rayleigh numbers



(a) $Ra = 10^3$ *(b)* $Ra = 10^4$ *(c)* $Ra = 10^5$ *(d)* $Ra = 10^6$

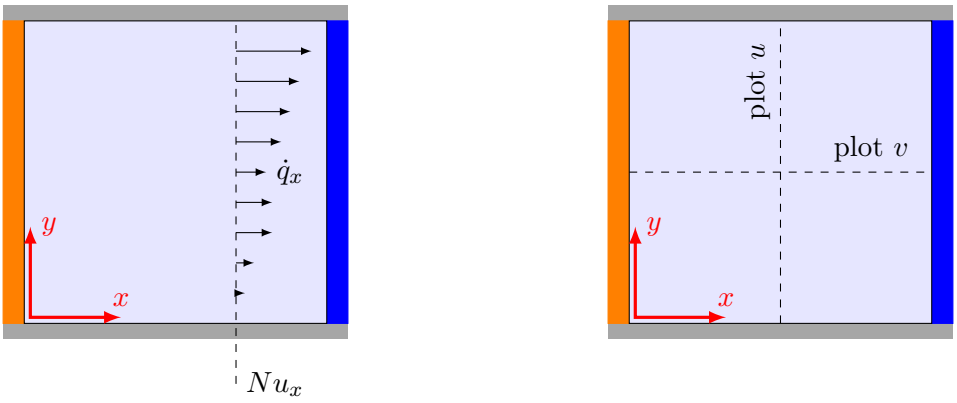
Figure 8: u velocity maps for different Rayleigh numbers



(a) $Ra = 10^3$ *(b)* $Ra = 10^4$ *(c)* $Ra = 10^5$ *(d)* $Ra = 10^6$

Figure 9: v velocity maps for different Rayleigh numbers

In order to calculate the Nusselt number you will integrate the heat flux over a vertical line at a certain position x.



First we will work with dimentionless variables:

$$x^* = \frac{x}{L} \qquad y^* = \frac{y}{L} \qquad T^* = \frac{T - T_{cold}}{T_{hot} - T_{cold}} \qquad u^* = \frac{uL}{\alpha} \qquad v^* = \frac{vL}{\alpha} \qquad \text{where } \alpha = \frac{\lambda}{\rho c_p}$$

The heat flux in the x direction can be written as:

$$\dot{q}_x(x, y) = u^* T^* - \frac{\partial T^*}{\partial x^*}$$

The overall heat flux along a vertical line will be:

$$Nu_x = \int_0^1 \dot{q}_x(x, y)\ dy^* = \int_0^1 \left(u^* T^* - \frac{\partial T^*}{\partial x^*} \right)\ dy^*$$

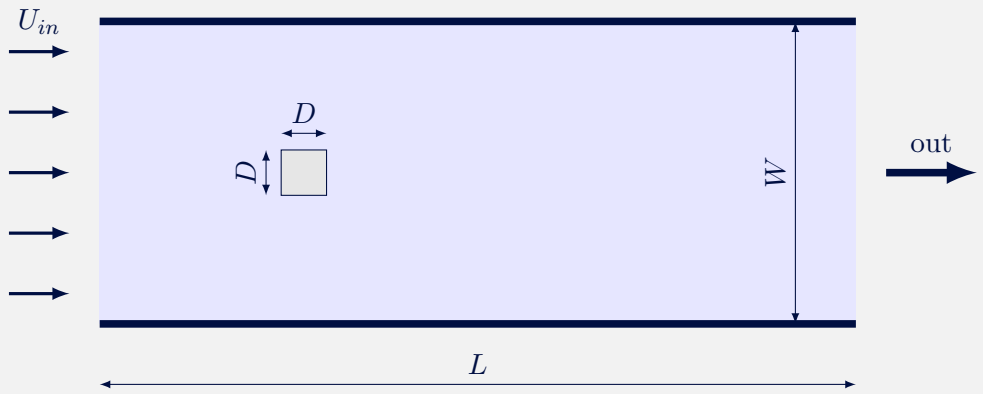
In theory, the heat flux should be constant regardless of the selected vertical line x, however some numerical errors might give small differences. To compute the average nusselt number we can average all the x lines:

$$\overline{Nu} = \int_0^1 Nu_x\ dx^*$$

6.11 Square cylinder

Exercise 6c (optional) : Square cylinder

We want to study the behaviour of an external flow around a 2D square cylinder. We insert this cylinder in a large enough domain ($L \times W$) so that the effect of the external walls is negligible.



The case is characterized by the cylinder Reynolds number:

$$Re = \frac{\rho U_{in} D}{\mu}$$

Solve the case for a low Reynolds number $Re \sim (50 - 250)$. You can search in the literature in order to chose the size of the domain, appropriate mesh and boundary conditions and validate your results. If you plot the velocity colour map over time, you will see what is known as Von Kármán vortex street.

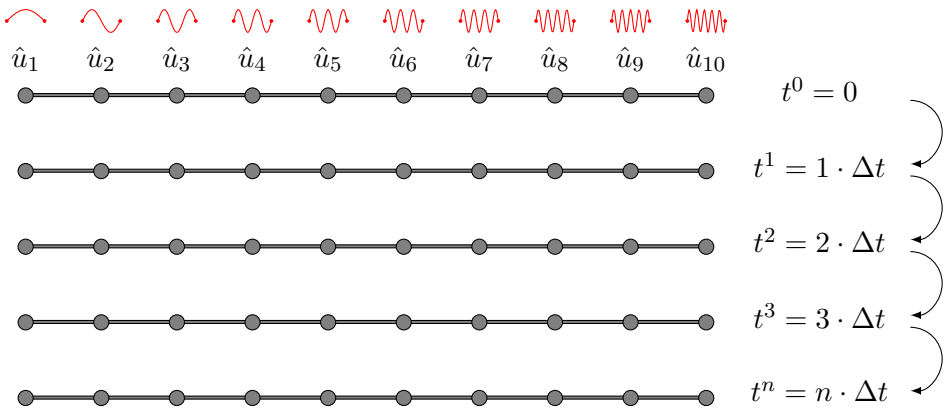
6.12 Introduction to turbulence

6.12.1 Burgers equation

You are refered to the additional notes on LES and turbulence modelling for the Burgers equation [4]. Try using those notes instead of the following ones. Use these ones only if you have further doubts or you are stuck. We will solve the Burgers equation, that can be written as follows:

$$\begin{aligned} &\frac{\partial \hat{u}_k}{\partial t} = 0 && k = 1 \\ &\frac{\partial \hat{u}_k}{\partial t} + \sum_{k=p+q} \hat{u}_p i q \hat{u}_q = -\frac{k^2}{Re}\hat{u}_k && k \in [2, N] \end{aligned}$$

Note that we have done a mode discretization, each k corresponds to a mode or frequency, and we truncate up to N frequencies (ideally we would solve ∞).



We solve the set of equations, we can discretize them using an explicit scheme so :

$$\begin{aligned} \hat{u}_k^{n+1} &= \hat{u}_k^n && k = 1 \\ \frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\Delta t} &= -\sum_{k=p+q} \hat{u}_p^n i q \hat{u}_q^n - \frac{k^2}{Re}\hat{u}_k^n && k \in [2, N] \end{aligned}$$

Now it's very easy to isolate $\hat{u}_k^{n+1}$ as a function of the previous known values. Note that $\hat{u}_k$ is a complex number, so it has a real and an imaginary part:

$$\hat{u}_k = a_k + i \cdot b_k$$

As initial conditions we have that:

$$\hat{u}_k^0 = \frac{1}{k} \qquad \forall k \geq 1$$

Also note that we start our value of $\hat{u}_k = 1/k$ only with real part, then over the iterations it will evolve to a complex number. Another consideration, the term:

$$\sum_{k=p+q} \hat{u}_p i q \hat{u}_q$$

Corresponds to the sum of all the possible combinations of $p \in [-N, N]$ and $q \in [-N, N]$ that added give a particular k (including negative numbers). For example for $N = 3$ and $k = 2$

[4] Burgers equation in Fourier space (by CTTC), 2014

p	q	$p+q=k$
-1	3	2
0	2	2
1	1	2
2	0	2
3	-1	2

i is the imaginary number, and p and q are index ($\hat{u}_k = \hat{u}_q$ when $k = q$). We will have to evaluate $\hat{u}_k$ for k with negative numbers, and we are only solving for $k \in [1, N]$, however we have the property that:

$$\hat{u}_{-k} = \overline{\hat{u}_k}$$

Where $\overline{\hat{u}_k}$ is the complex conjugate [5]
Finally, once you solve the spectrum of velocities to a steady state, you can calculate the energy density of the spectrum using:

$$E_k = \hat{u}_k \cdot \overline{\hat{u}_k}$$

Note that this energy should give a real number. You can also recover the velocity profile with

$$u(x) = \sum_{k=-N}^{N} \hat{u}_k \cdot e^{ikx}$$

Finally, as we are working with complex numbers, your code should be able to operate with them as well. In Matlab this is more straightforward, but in C++ you might want to use operator overloading and classes. With this information you can already plot the black and red lines of Figure 10

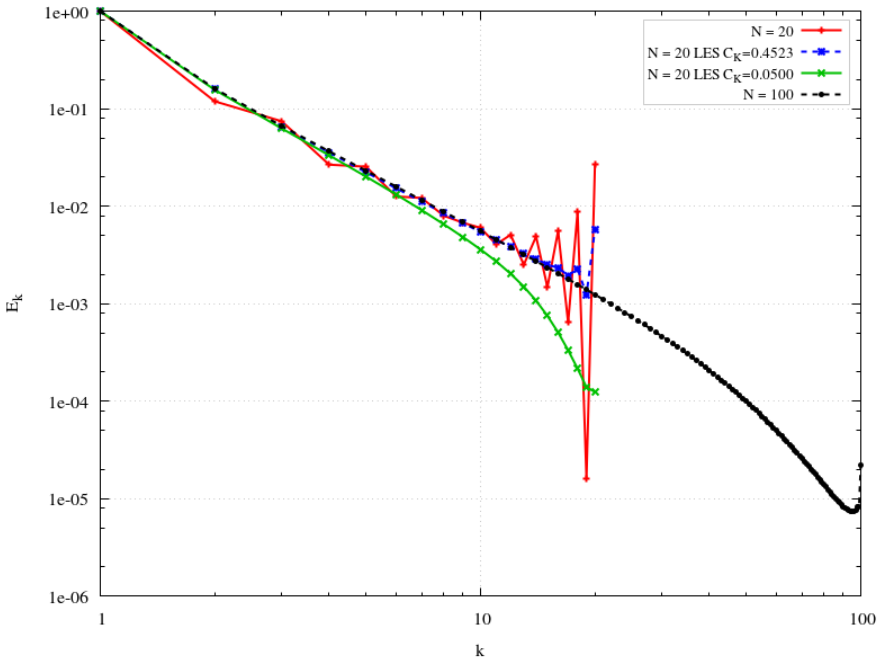


Figure 10: Energy density for different modes

[5] The complex conjugate of a complex number $z = a + bi$ is basically changing the sign for the imaginary part, i.e. $\overline{z} = a - bi$

6.12.2 LES modelling

Now if we want to introduce a LES model, we will modify the viscosity that we imposed with $\nu = 1/Re$ and substitute it with an effective viscosity that will depend on the mode $\nu_{eff}(k)$

$$\nu_{eff}(k) = \nu + \nu_t(k)$$

To find $\nu_t(k)$ for each mode k we first calculate some auxiliary variables $\nu_t^{+\infty}$ and $\nu_t^*(k)$. We also impose some constant C_k that is given and can be seen in Figure 10 for the different plots.

$$\nu_t^{+\infty} = 0.31\frac{5-m}{m+1}\sqrt{3-m}\; C_k^{-3/2} \qquad \text{where} \qquad m = 2$$
$$\nu_t^*(k) = 1 + 34.5e^{-3.03(N/k)}$$

Finally we have that $\nu_t(k)$

$$\nu_t(k) = \nu_t^{+\infty}\sqrt{\frac{E_N}{N}}\nu_t^*(k) \qquad \text{where} \qquad E_N = \hat{u}_N \cdot \overline{\hat{u}_N}$$

6.12.3 Complex numbers operations

Just as a reminder of complex operations that you should already know: The imaginary number is defined as:

$$i = \sqrt{-1} \qquad\qquad i^2 = -1$$

Let $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$ be complex numbers and let s be a real (a non-complex) number.

$$\boldsymbol{z_1} = a + bi \qquad\qquad \boldsymbol{z_2} = c + di$$

The basic operations we can perform are.
Multiplication by a scalar (note that a division by a scalar is basically a multiplication to s^{-1}):

$$s \cdot \boldsymbol{z_1} = (s \cdot a) + (s \cdot b)i$$

Summation and subtraction

$$\boldsymbol{z_1} + \boldsymbol{z_2} = (a + c) + (b + d)i \qquad\qquad \boldsymbol{z_1} - \boldsymbol{z_2} = (a - c) + (b - d)i$$

Multiplication of two complex numbers:

$$\boldsymbol{z_1} \cdot \boldsymbol{z_2} = (ac - bd) + (bc + ad)i$$

Complex conjugation $\overline{\boldsymbol{z_1}}$ (sometimes seen as $\boldsymbol{z_1^*}$)

$$\overline{\boldsymbol{z_1}} = a - bi$$

Magnitude $|\boldsymbol{z_1}|$

$$|\boldsymbol{z_1}| = \sqrt{a^2 + b^2}$$

Transformation to polar form:

$$\boldsymbol{z_1} = |z_1|e^{i\phi} \qquad \text{where} \qquad \phi = \arctan\frac{b}{a}$$

Inverse transform:

$$\boldsymbol{z_1} = |z_1|e^{i\phi} = |z_1|\left(\cos\phi + i\sin\phi\right)$$

Annex : Non uniform meshes

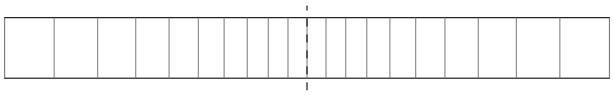
We have already discussed the motivation of using non uniform meshes refined at certain zones of the domain, for example the walls. Note that if you use non uniform meshes, you will have to deduce a more general form of the previously introduced equations.

There are multiple ways of generating non uniform grids, here we will introduce a formula you can use, but it's not the best nor the only one:

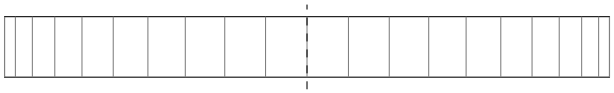
$$x = \frac{i}{N} \cdot L + A\left(X_c - \frac{i}{N} \cdot L\right)\left(1 - \frac{i}{N}\right)\frac{i}{N}$$

x	Position of the node
i	Integer coordinate of the node
N	Number of nodes
L	Length of the mesh
A	Strenght of the refinement $[-2, 2]$
X_c	Centering of the refinement $(0, L)$

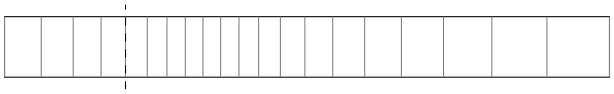
If A is positive, the region near X_c will be refined, however if A is negative, the region far from X_c will be refined.



$A = 1.5 \quad X_c = 0.5$



$A = -1.5 \quad X_c = 0.5$

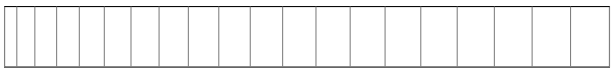


$A = 1.5 \quad X_c = 0.2$

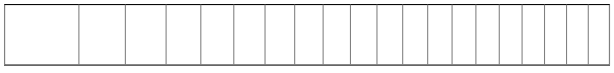
Note that this is bidirectional an does not work well with X_c at the boundaries. If we want only one direction refinement we will need another formula. For example we can use for example:

$$x = \left(\frac{i}{N} \cdot L\right)^{\kappa}$$

κ	Strength of the refinement



$\kappa = 1.3$



$\kappa = 0.7$

Here you can see an example of a mesh refinement in 2D, with $X_c = 0.5L$, $Y_c = 0.5L$ and $A = -1.0$