ME663 - Computational Fluid Dynamics
Algorithms Derivation and application to a 2D Lid-Driven Cavity Flow

Tommaso Bocchietti

A.Y. 2023/24 - W24

UNIVERSITY OF
WATERLOO

Contents

List of Figures

List of Tables

Listings

1 Derivation of discretized governing equations

In this section, we will derive the discretized governing equations for the incompressible Navier-Stokes equations, which will be used to solve the problem at hand.

The set of incompressible Navier-Stokes equations is given by:

$$\nabla \cdot \mathbf{u} = 0$$

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}$$

In the rest of the document, the following hypotheses will be considered:

- Steady-state problem: $\frac{\partial \mathbf{u}}{\partial t} = 0$

- Constant density: $\rho = \text{const}$

- Constant dynamic viscosity: $\mu = \text{const}$

- Zero body forces: $\mathbf{f} = 0$

Based on these hypotheses, the incompressible Navier-Stokes equations can be simplified and expanded as follows:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

$$\frac{\partial uu}{\partial x} + \frac{\partial vu}{\partial y} = -\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{2}$$

$$\frac{\partial uv}{\partial x} + \frac{\partial vv}{\partial y} = -\frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \tag{3}$$

Where $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity and $p = \frac{p}{\rho}$ is the non-dimensional pressure.

Obviously, to solve the problem using a discrete calculator, the equations must be therefore discretized.

1.1 Finite Volume Method

The Finite Volume Method (FVM) is a numerical technique used to discretize partial differential equations, and is particularly well suited for the discretization of the Navier-Stokes equations.

The idea here is to divide the domain into a set of control volumes, and then integrate the governing equations over each control volume. The resulting set of equations will be a set of algebraic equations, which can be solved using a discrete calculator.

1.1.1 Control volumes

Before proceeding with the discretization of the governing equations, we need to define what a control volume is and the notations used in the rest of the document.

In particular, we will assume from now on to have a Cartesian grid, with a uniform mesh spacing in both the x and y directions.

From the Figure 1, we can appreciate graphically how the domain is divided.

Figure 1: Control volumes and control volume faces.

In particular, Figure 1 shows:

- A grid of control volumes, with the subscript (i, j) indicating the position of the control volume in the x and y directions, respectively. For example, the control volume in the center of the grid has indices $(i, j) = (3, 3)$.

- The control volume centers with capital letters, $P, N, S, E, W, \ldots$.

- The control volume faces with lowercase letters, n, s, e, w.

Notice also that the capital letters always refers to a relative position with respect to the control volume in consideration. For example, P refers to the control volume in consideration, N refers to the control volume to the north of P, and so on. For this reason, in Figure 1, the control volume P is highlighted in yellow so to indicate that is the control volume in consideration.

1.1.2 Staggered grid and L_{shape}

As we will see later during the formulation of the solving solution, for the purpose of this work, we will use a specific type of grid, called staggered grid.

We can also give a brief definition of the two types of grids available in the literature, which are:

- **Collocated grid**: all the variables are located at the same point in the control volume (e.g. the center of the control volume).

- **Staggered grid**: the variables are located at different points in the control volume (e.g. the velocity components are located at the center of the faces of the control volume, and the pressure is located at the center of the control volume).

Given the formulation of the staggered grid, it's now useful to define the so called L_{shape}, which is a frame used to define in a compact and clear way the position of the variables in the control volume.

The L_{shape} has been reported for control volume P in Figure 2.
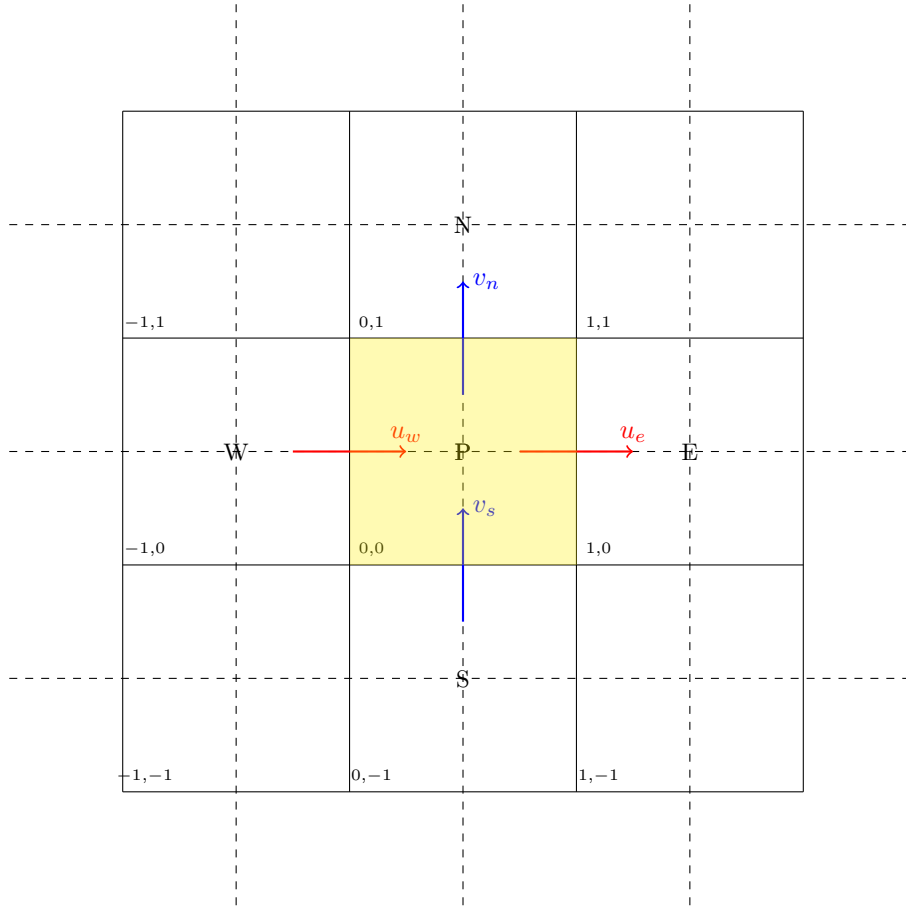
Figure 2: L_{shape} for control volume P.

Basically, the L_{shape} for control volume P links the velocity components to the control volume P itself, and it's used to define the indexes of the system.

In particular, the same Figure 2 can be represented using the index notations, as shown in Figure 3.
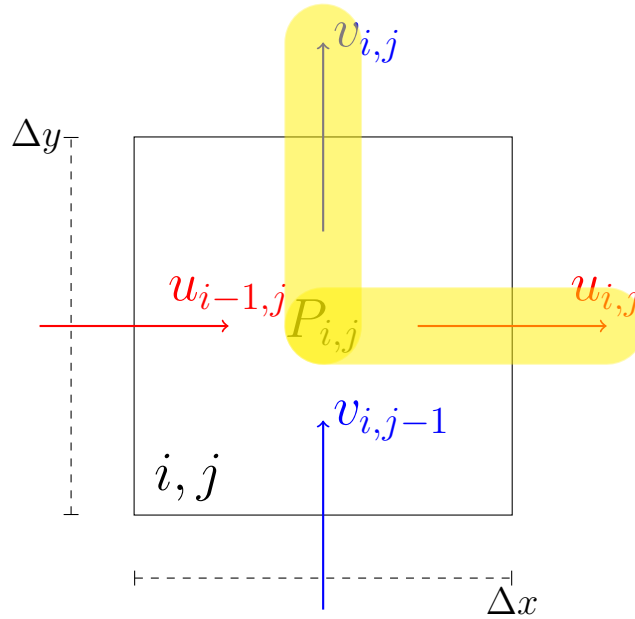
Figure 3: L_{shape} for control volume P using index notations.

From Figure 3, we can appreciate how the L_{shape} works well with index notations, and can be useful when working purely with indexes to refer to the variables.

Finally, from Figures 4 5 & 6, we can appreciate how the staggered grid is used to define different control

volumes for different variables.

During the rest of the document, we will avoid the specification of the control volume taken in consideration, because it will be chosen according to the variable in consideration. If we are working with the pressure, we will refer to the control volume as P_{CV}, if we are working with the velocity components, we will refer to the control volume as U_{CV} or V_{CV}, and so on.

So for example, when dealing with the u velocity component, we will refer to the control volume as U_{CV}, so that:

- u_P will refer to the velocity component centered in the control volume U_{CV}. Equivalent to $u_{i,j}$ in the index notation.

- u_e instead, will be the east component with respect to the control volume U_{CV}.

- u_w will be the west component with respect to the control volume U_{CV}. Spatially, u_w will be centered in the P_{CV}.
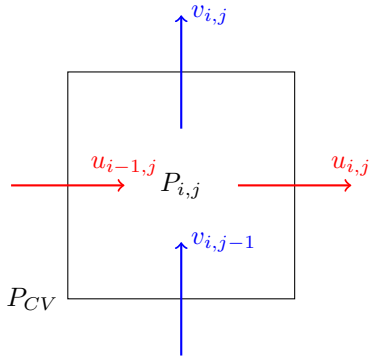
Figure 4: $P_{ControlVolume}$ (same as in Figure 3)
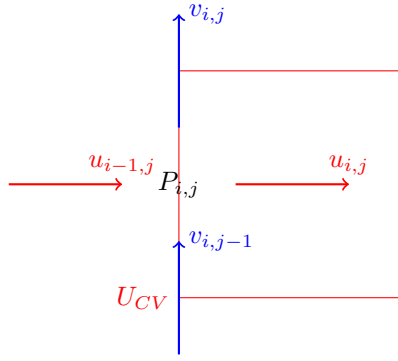
Figure 5: $U_{ControlVolume}$
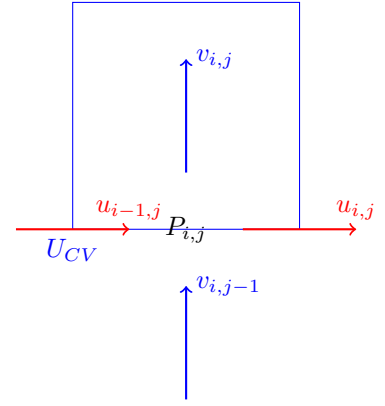
Figure 6: $V_{ControlVolume}$

1.2 Application of the Finite Volume Method

Having defined our working framework, we can now proceed with the application of the Finite Volume Method (FVM) to the incompressible Navier-Stokes equations. We know that the definition of the FVM is based on the integration of the given PDE equation over the control volume, such as:

$$\text{FVM} := \int_V \text{PDE} dV = \int_s^n \int_w^e \text{PDE} dx dy \tag{4}$$

Where s, n, w and e are the south, north, west and east faces of the control volume, respectively.

For our case, we have a set of three equations, the continuity equation 1, and the momentum equations 2 and 3.

We can now proceed by applying the FVM to each of these equations separately.

Notice that in the followings, we will assume:

$$u = u(x) \tag{5}$$
$$v = v(y) \tag{6}$$

Which underline the independence of the velocity components from the y and x coordinates, respectively.

1.2.1 Continuity Equation (FVM)

The FVM for the continuity equation 1 is given by:

$$\int_V \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} dV = 0 \tag{7}$$

Evaluating the integral based on the assumption 6, we obtain:

$$\int_s^n \int_w^e \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} dxdy = 0 \tag{8}$$

$$\int_s^n \left[\int_w^e \frac{\partial u}{\partial x} dx \right] dy + \int_w^e \left[\int_s^n \frac{\partial v}{\partial y} dy \right] dx = 0 \tag{9}$$

$$\int_s^n \left[u_e - u_w \right] dy + \int_w^e \left[v_n - v_s \right] dx = 0 \tag{10}$$

$$(u_e - u_w)\Delta y + (v_n - v_s)\Delta x = 0 \tag{11}$$

Where Δx and Δy are the dimensions of the control volume in the x and y directions, respectively.
Since we will be dealing with indexes for the implementation of the code, we can rewrite the equation as:

$$(u_i - u_{i-1})\Delta y + (v_j - v_{j-1})\Delta x = 0 \tag{12}$$

1.2.2 Momentum Equations (FVM)

The FVM for the momentum equations 2 and 3 is given by:

$$\int_V \frac{\partial uu}{\partial x} + \frac{\partial vu}{\partial y} + \frac{\partial p}{\partial x} - \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) dV = 0 \tag{13}$$

$$\int_V \frac{\partial uv}{\partial x} + \frac{\partial vv}{\partial y} + \frac{\partial p}{\partial y} - \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) dV = 0 \tag{14}$$

Since the equations are very similar, we will only derive the FVM for the first momentum equation (along the x direction), and then we will present the final form for both equations.
Evaluating the integral based on the assumption 6, we obtain:

$$\int_V \underbrace{\frac{\partial uu}{\partial x} + \frac{\partial vu}{\partial y}}_{\text{Convection terms}} + \underbrace{\frac{\partial p}{\partial x}}_{\text{Source term}} - \underbrace{\nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)}_{\text{Diffusion term}} dV = 0 \tag{15}$$

We can now proceed by discretizing the convection, diffusion and source terms separately.

Convection Term The finite volume discretization of the convection term is given by:

$$\int_V \frac{\partial uu}{\partial x} + \frac{\partial vu}{\partial y} dV = 0 \tag{16}$$

We further split the convection term into its directional components, so to solve them separately:

$$\int_s^n \int_w^e \frac{\partial uu}{\partial x} dxdy = \tag{17}$$

$$\int_s^n \left[\int_w^e \frac{\partial uu}{\partial x} dx \right] dy = \tag{18}$$

$$\int_s^n \left[u_e u_e - u_w u_w \right] dy = \tag{19}$$

$$(u_e u_e - u_w u_w)\Delta y \tag{20}$$

Where Δy is the dimension of the control volume in the y direction.
Since our equation is highly non-linear, we can try to linearize it by introducing the concept of:

- Advecting velocity: previous step velocity $\hat{u}_e = \frac{1}{2}(u_P + u_E)$, $\hat{u}_w = \frac{1}{2}(u_W + u_P) \to$ Known.

- Advected velocity: current step velocity $u_e, u_w \to$ Unknown, evaluated using the Convection schemes (following chapter).

We can then rewrite the convection term for the x direction as:

$$\int_V \frac{\partial uu}{\partial x} dV = (\hat{u}_e u_e - \hat{u}_w u_w) \Delta y \tag{21}$$

The same procedure can be applied to the convection term for the y direction, and the final form for the convection term is given by:

$$\int_V \frac{\partial uu}{\partial x} + \frac{\partial vu}{\partial y} dV = (\hat{u}_e u_e - \hat{u}_w u_w) \Delta y + (\hat{v}_n u_n - \hat{v}_s u_s) \Delta x \tag{22}$$

Diffusion Term   The finite volume discretization of the diffusion term is given by:

$$\int_V \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) dV = 0 \tag{23}$$

We further split the diffusion term into its directional components, so to solve them separately:

$$\int_s^n \int_w^e \nu \left(\frac{\partial^2 u}{\partial x^2} \right) dx dy = \tag{24}$$

$$\int_s^n \left[\int_w^e \frac{\partial}{\partial x} \left(\nu \frac{\partial u}{\partial x} \right) dx \right] dy = \tag{25}$$

$$\int_s^n \left[\nu \frac{\partial u}{\partial x} \bigg|_e - \nu \frac{\partial u}{\partial x} \bigg|_w \right] dy = \tag{26}$$

$$\nu \left(\frac{\partial u}{\partial x} \bigg|_e - \frac{\partial u}{\partial x} \bigg|_w \right) \Delta y \tag{27}$$

Where the terms $\left(\frac{\partial u}{\partial x} \bigg|_{Position} \right)$ will be approximated using a Taylor expansion.

In particular, our final form for the diffusion term for the x direction is given by:

$$\int_V \nu \left(\frac{\partial^2 u}{\partial x^2} \right) dV = \nu \left(\approx \frac{\partial^2 u}{\partial x^2} \bigg|_{x \to 0} \right) \Delta x \Delta y \tag{28}$$

The same procedure can be applied to the diffusion term for the y direction, and the final form for the diffusion term is given by:

$$\int_V \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) dV = \nu \left(\approx \frac{\partial^2 u}{\partial x^2} \bigg|_{x \to 0} \right) \Delta x \Delta y + \nu \left(\approx \frac{\partial^2 u}{\partial y^2} \bigg|_{y \to 0} \right) \Delta x \Delta y \tag{29}$$

More details on the approximation of the second order derivatives will be reported in the following chapters.

Source Term   The finite volume discretization of the source term is given by:

$$\int_V -\frac{\partial p}{\partial x} dV = 0 \tag{30}$$

Evaluating the integral based on the assumption 6, we obtain:

$$\int_s^n \int_w^e -\frac{\partial p}{\partial x} dx dy = \tag{31}$$

$$\int_s^n \left[-\int_w^e \frac{\partial p}{\partial x} dx \right] dy = \tag{32}$$

$$\int_s^n \left[-p_e + p_w \right] dy = \tag{33}$$

$$-(p_e - p_w) \Delta y \tag{34}$$

Final Form We can now substitute the discretized convection, diffusion and source terms into the momentum equations for both the x and y directions (13 14), and obtain the final form for the discretized momentum equations:

$$(\hat{u_e}u_e - \hat{u_w}u_w)\Delta y + (\hat{v_n}u_n - \hat{v_s}u_s)\Delta x - \nu \left(\approx \frac{\partial^2 u}{\partial x^2}\bigg|_{x\to 0} + \approx \frac{\partial^2 u}{\partial y^2}\bigg|_{y\to 0} \right) \Delta x \Delta y - (p_e - p_w)\Delta y = 0 \qquad (35)$$

$$(\hat{u_e}v_e - \hat{u_w}v_w)\Delta y + (\hat{v_n}v_n - \hat{v_s}v_s)\Delta x - \nu \left(\approx \frac{\partial^2 v}{\partial x^2}\bigg|_{x\to 0} + \approx \frac{\partial^2 v}{\partial y^2}\bigg|_{y\to 0} \right) \Delta x \Delta y - (p_n - p_s)\Delta x = 0 \qquad (36)$$

2 Schemes

In this section, we will present the schemes used to solve the previously quasi-discretized equations 22 and 29. Those schemes are the "tools" that will allow us to bind the currently considered cell values with its neighbor's cell values. This means that at the end of this section, we will have a set of coefficients (based on the scheme adopted) that will be used to assemble the solving matrix of the system.

To further clarify, what we want to achieve is to compute the following functions (nb indicates a generic neighbor cell, one or more cells away from the current cell P):

$$u_P = f(u_{nb}, v_{nb}, p_{nb}) \qquad (37)$$

$$v_P = f(u_{nb}, v_{nb}, p_{nb}) \qquad (38)$$

As we will see, the result of the schemes will be a set of coefficients that will be used to compute the values of u_P and v_P based on the values of the neighbor cells. In particular, the final form of the system will be:

$$A_P^u u_P = \sum_{nb} A_{nb}^u u_{nb} + b_P^u \qquad (39)$$

$$A_P^v v_P = \sum_{nb} A_{nb}^v v_{nb} + b_P^v \qquad (40)$$

Notice how there is no direct correlation between the u and v equations, that are instead coupled through the pressure term p.

Using the indices' notation system (i, j) becomes:

$$(A_P^u)_{i,j} u_{i,j} = \sum_{nb}(A_{nb}^u)_{i,j} u_{i,j} + (p_{i,j} - p_{i+1,j})\Delta y \qquad (41)$$

$$(A_P^v)_{i,j} v_{i,j} = \sum_{nb}(A_{nb}^v)_{i,j} v_{i,j} + (p_{i,j} - p_{i,j+1})\Delta x \qquad (42)$$

Similar to the approach used in the previous section, we will treat separately the convection and diffusion terms. In this way we will obtain two different sets of coefficients, one for the convection term and one for the diffusion term. Those coefficients will be then reassembled together to reduce the equations to the final form presented above.

In particular, we will have that the coefficients A_P^ϕ and A_{nb}^ϕ will be the difference between the convection and diffusion coefficients, given that from Equation 15, we know:

$$FVM_{\text{Convection}} - FVM_{\text{Diffusion}} = 0 \to (A^\phi) = (A^\phi)_{\text{Convection}} - (A^\phi)_{\text{Diffusion}} \qquad (43)$$

Note: In the following sections, we will refer to the use of `Mathematica`. The complete notebook used to obtain the final form of the coefficients is left in the 6.1 section of this document.

2.1 Convection schemes

In this section, we will present the schemes related to the convection terms of the discretized governing equations, which were derived in Section 1.2.

2.1.1 Upwind Differencing Scheme (UDS)

The Upwind Differencing Scheme (UDS) is the simplest convection scheme, and it is a $1^{th} - order$ scheme.
The idea behind the UDS is to consider the velocity at the face of the U_{CV} equal to the upwind value between the current cell center value and the neighbor cell center value.
This means that UDS scheme can be visualized as:

Figure 7: Example of the UDS scheme applied to the u velocity component.

We can obtain a formula for the UDS that will be used in the implementation of the code. In particular, by defining the Volume Fluxes as: $F_e = \hat{u_e}\Delta y$ & $F_w = \hat{u_w}\Delta y$, we can write that:

$$u_e = \begin{cases} u_P & \text{if } F_e > 0 \\ u_E & \text{if } F_e < 0 \end{cases} \tag{44}$$

$$u_w = \begin{cases} u_W & \text{if } F_w > 0 \\ u_P & \text{if } F_w < 0 \end{cases} \tag{45}$$

The same apply for the v velocity component, but in this case, we will have $F_n = \hat{v_n}\Delta x$, $F_s = \hat{v_s}\Delta x$.
In the end, our Upwind Differencing Scheme (UDS) scheme applied to the convection term will be:

$$\left(\hat{u_e}u_e - \hat{u_w}u_w\right)\Delta y + \left(\hat{v_n}u_n - \hat{v_s}u_s\right)\Delta x =$$
$$\left(F_e u_e - F_w u_w\right) + \left(F_n u_n - F_s u_s\right) =$$
$$u_P * max(F_e, 0) + u_E * min(F_e, 0) + u_W * max(F_w, 0) + u_P * min(F_w, 0) +$$
$$u_P * max(F_n, 0) + u_N * min(F_n, 0) + u_S * max(F_s, 0) + u_P * min(F_s, 0)$$

Since we are interested in the A_P^ϕ and A_{nb}^ϕ coefficients as written in the form of Equation 40, with the help of `Mathematica`, we can regroup the terms based on the velocity components, perform some sign manipulations and obtain the coefficients for the convection term using the UDS scheme for both the u and v momentum equations.

$$\text{Convection Coefficients UDS:} = \textbf{See table 1} \tag{46}$$

2.1.2 Central Differencing Scheme (CDS)

The Central Differencing Scheme (CDS) is a $2^{nd} - order$ scheme.
The idea behind the CDS is to consider the velocity at the face of the control volume as the average between the current cell center value and the neighbor cell center value.
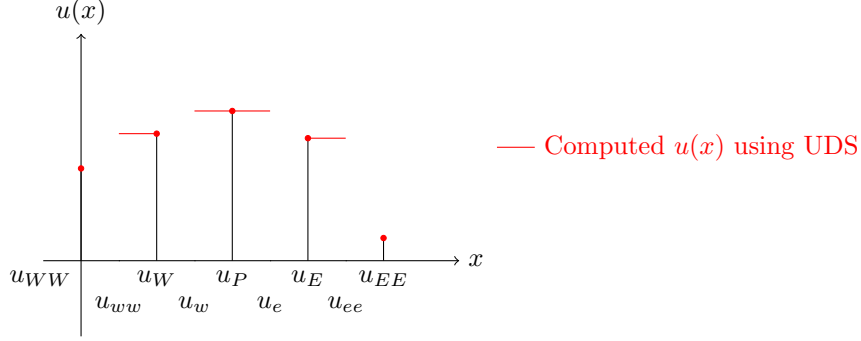This means that CDS scheme can be visualized as:

Figure 8: Example of the CDS scheme applied to the u velocity component.

We can obtain a formula for the CDS that will be used in the implementation of the code. In particular, by using the same definition of the Volume Fluxes as before, we can write that:

$$u_e = \frac{u_P + u_E}{2} \tag{47}$$

$$u_w = \frac{u_W + u_P}{2} \tag{48}$$

The same apply for the v velocity component.
In the end, our CDS scheme applied to the convection term will be:

$$(\hat{u_e} u_e - \hat{u_w} u_w) \Delta y + (\hat{v_n} u_n - \hat{v_s} u_s) \Delta x =$$
$$(F_e u_e - F_w u_w) + (F_n u_n - F_s u_s) =$$
$$F_e \frac{u_P + u_E}{2} - F_w \frac{u_W + u_P}{2} + F_n \frac{u_P + u_N}{2} - F_s \frac{u_S + u_P}{2}$$

Since we are interested in the A_P^{ϕ} and A_{nb}^{ϕ} coefficients as written in the form of Equation 40, with the help of `Mathematica`, we can regroup the terms based on the velocity components, perform some sign manipulations and obtain the coefficients for the convection term using the CDS scheme for both the u and v momentum equations.

Convection Coefficients CDS: = **See table 1**

2.1.3 Quadratic Upstream Interpolation for Convective Kinematics (QUICK)

The Quadratic Upstream Interpolation for Convective Kinematics (QUICK) is a $3^{rd} - order$ scheme.
The idea behind the QUICK scheme is to interpolate the velocity at the center of 3 cells to then compute the velocity at the face of the cell. The choice of the cells to interpolate is based on the direction of the velocity at previous step ($\hat{u}$ or $\hat{v}$) similarly to the UDS scheme. For example, if we are computing the u_e velocity, we will pick as interpolations points the u_P, u_E velocity and u_{EE} or u_W based on the direction of the velocity at the face.
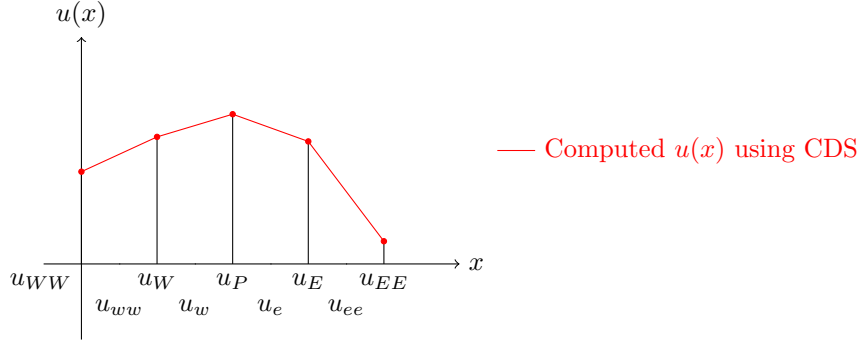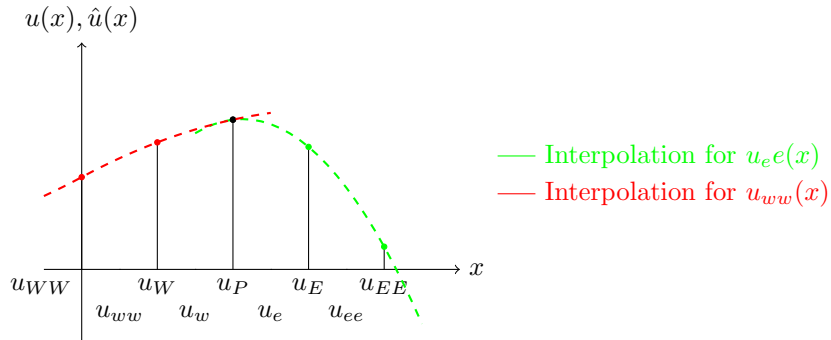The QUICK scheme can be visualized as:

Figure 9: Example of the QUICK scheme applied to the u velocity component.

We can obtain a formula for the QUICK that will be used in the implementation of the code. In particular, we can recall the definition of the Lagrange interpolation polynomial of degree 2:

$$P(x) = \sum_{i=0}^{2} y_i \prod_{j=0, j \neq i}^{2} \frac{x - x_j}{x_i - x_j} \tag{49}$$

In our case, it's convenient to fix the reference system in correspondence of the center of the control volume considered. If that is the case, then for example the evaluation of the velocity at the face u_e will be:

$$u_e = P(x) \Big|_{x = \Delta x / 2} \tag{50}$$

As we said before, the choice of the interpolation points is based on the direction of the velocity at the face of the control volume. In particular, if we are computing the u_e velocity, the possible polynomials ($P_e(x)$) will be:

$$P_e(x) = \begin{cases} P(u_W, u_P, u_E) & \text{if } \hat{u}_e > 0 \\ P(u_P, u_E, u_{EE}) & \text{if } \hat{u}_e < 0 \end{cases} \tag{51}$$

By using `Mathematica`, we can compute the polynomials for the u velocity component in both cases. The results of the symbolic analysis are:

$$u_e = \begin{cases} -\frac{1}{8} u_W + \frac{3}{4} u_P + \frac{3}{8} u_E & \text{if } \hat{u}_e > 0 \\ +\frac{3}{8} u_P + \frac{3}{4} u_E - \frac{1}{8} u_{EE} & \text{if } \hat{u}_e < 0 \end{cases} \tag{52}$$

$$u_w = \begin{cases} -\frac{1}{8} u_{WW} + \frac{3}{4} u_W + \frac{3}{8} u_P & \text{if } \hat{u}_w > 0 \\ +\frac{3}{8} u_W + \frac{3}{4} u_P - \frac{1}{8} u_E & \text{if } \hat{u}_w < 0 \end{cases} \tag{53}$$

The same apply for the v velocity component, but in this case, the variables that decide the interpolation points are the $\hat{v}_n$ and $\hat{v}_s$.

In the end, our QUICK scheme applied to the convection term will be:

$$(\hat{u}_e u_e - \hat{u}_w u_w) \Delta y + (\hat{v}_n u_n - \hat{v}_s u_s) \Delta x = \tag{54}$$

$$(F_e u_e - F_w u_w) + (F_n u_n - F_s u_s) = \tag{55}$$

$$\frac{1}{8}((3u_E + 6u_P - u_W)max(0, F_e) + (-u_{EE} + 6u_E + 3u_P)min(0, F_e)+ \tag{56}$$

$$(-3u_P - 6u_W + u_{WW})max(0, F_w) + (u_E - 6u_P - 3u_W)min(0, F_w)+ \tag{57}$$

$$(3u_N + 6u_P - u_S)max(0, F_n) + (6u_N - u_{NN} + 3u_P)min(0, F_n)+ \tag{58}$$

$$(-3u_P - 6u_S + u_{SS})max(0, F_s) + (u_N - 6u_P - 3u_S)min(0, F_s)) \tag{59}$$

Since we are interested in the A_P^ϕ and A_{nb}^ϕ coefficients as written in the form of Equation 40, with the help of `Mathematica`, we can regroup the terms based on the velocity components, perform some sign manipulations and obtain the coefficients for the convection term using the QUICK scheme for both the u and v momentum equations.

<div align="center">Convection Coefficients QUICK: = See table 1</div>

2.2 Diffusion Schemes

In this section, we will present the schemes related to the diffusion terms of the discretized governing equations, which were derived in Section 1.2.

The general idea here is to approximate the diffusion term using its Taylor expansion and then evaluate the derivative at the cell faces. As a recall, the definition of the Taylor expansion of order n is:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^n(x_0)}{n!}(x - x_0)^n + \mathbf{O}((x - x_0)^{n+1}) \tag{60}$$

2.2.1 2nd-order scheme

The 2nd-order scheme approximate the diffusion term using the Taylor expansion up to the second order.
We can start by writing the polynomial that approximates by interpolation the values of the variable u at the cell faces. To do so, we can use the Lagrange interpolation polynomial, previously defined in Equation 49. Since we are now analyzing the 2nd-order scheme, we will use $n = 2 + 1 = 3$ points to interpolate.
As before, we can use `Mathematica` to obtain the polynomial that approximates function $u(x)$ as:

$$u(x) \approx \frac{2(\Delta x)^2 u_P + x^2(u_E - 2u_P + u_W) + \Delta x x(u_E - u_W)}{2(\Delta x)^2} \tag{61}$$

We can now proceed by evaluating the second derivative of the polynomial as:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_E - 2u_P + u_W}{(\Delta x)^2} \tag{62}$$

And finally, we can the approximated second derivative at the point of interest $x = 0$ as:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_P \approx \frac{u_E - 2u_P + u_W}{(\Delta x)^2} \tag{63}$$

Given the definition of the diffusion term of u along x in Equation 29, we can write the 2nd-order scheme as:

$$\nu\frac{\partial^2 u}{\partial x^2} = \nu\frac{u_E - 2u_P + u_W}{(\Delta x)^2} \tag{64}$$

With a similar approach, we can obtain the 2nd-order scheme for the y direction as:

$$\nu\frac{\partial^2 u}{\partial y^2} = \nu\frac{u_N - 2u_P + u_S}{(\Delta y)^2} \tag{65}$$

The same procedure can be applied to the v momentum equation, obtaining the following results:

$$\nu\frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^2 u}{\partial y^2} = \tag{66}$$

$$\frac{\Delta y \nu(u_E - 2u_P + u_W)}{\Delta x} + \tag{67}$$

$$\frac{\Delta x \nu(u_N - 2u_P + u_S)}{\Delta y} \tag{68}$$

Since we are interested in the A_P^ϕ and A_{nb}^ϕ coefficients as written in the form of Equation 40, with the help of `Mathematica`, we can regroup the terms based on the velocity components, perform some sign manipulations and obtain the coefficients for the diffusion term using the 2nd-order scheme for both the u and v momentum equations.

$$\text{Diffusion Coefficients order} 2^{nd} = \textbf{See table 1} \tag{69}$$

2.2.2 4th-order scheme

The 4th-order scheme approximate the diffusion term using the Taylor expansion up to the fourth order.
Since the reasoning is similar to the 2nd-order scheme, we can directly write the approximated second derivative at the point of interest $x = 0$ as:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_P \approx -\frac{u_{WW} - 16u_W + 30u_P - 16u_E + u_{EE}}{12(\Delta x)^2} \tag{70}$$

Given the definition of the diffusion term of u along x in Equation 29, we can write the 4th-order scheme as:

$$\nu\frac{\partial^2 u}{\partial x^2} = \nu - \frac{u_{WW} - 16u_W + 30u_P - 16u_E + u_{EE}}{12(\Delta x)^2} \tag{71}$$

With a similar approach, we can obtain the 4th-order scheme for the y direction as:

$$\nu\frac{\partial^2 u}{\partial y^2} = \nu - \frac{u_{WW} - 16u_W + 30u_P - 16u_E + u_{EE}}{12(\Delta y)^2} \tag{72}$$

The same procedure can be applied to the v momentum equation, obtaining the following results:

$$\int_V \nu \frac{\partial^2 u}{\partial x^2} + \nu \frac{\partial^2 u}{\partial y^2} dV = \tag{73}$$

$$-\frac{\nu}{12} \left(\frac{u_{WW} - 16u_W + 30u_P - 16u_E + u_{EE}}{(\Delta x)^2} + \frac{u_{WW} - 16u_W + 30u_P - 16u_E + u_{EE}}{(\Delta y)^2} \right) \Delta x \Delta y \tag{74}$$

Since we are interested in the A_P^ϕ and A_{nb}^ϕ coefficients as written in the form of Equation 40, with the help of `Mathematica`, we can regroup the terms based on the velocity components, perform some sign manipulations and obtain the coefficients for the diffusion term using the 4th-order scheme for both the u and v momentum equations.

$$\text{Diffusion Coefficients order } 4^{th} = \textbf{See table 1} \tag{75}$$

2.3   Final coefficients

The final coefficients are obtained by reassembling the convection and diffusion coefficients, based on the FVM result:

$$FVM_{\text{Convection}} - FVM_{\text{Diffusion}} = 0 \rightarrow (A^\phi) = (A^\phi)_{\text{Convection}} - (A^\phi)_{\text{Diffusion}} \tag{76}$$

Table 1: Final Ap coefficients for convection and diffusion using different schemes

Ap	Convection UDS	Convection CDS	Convection QUICK	Diffusion 2° order	Diffusion 4° order
$Ap\phi WWSS$	0	0	0	0	0
$Ap\phi WWS$	0	0	0	0	0
$Ap\phi WWP$	0	0	$-\frac{1}{8}\max(0,Fw\phi)$	0	$-\frac{\Delta y\nu}{12\Delta x}$
$Ap\phi WWN$	0	0	0	0	0
$Ap\phi WWNN$	0	0	0	0	0
$Ap\phi WSS$	0	0	0	0	0
$Ap\phi WS$	0	0	0	0	0
$Ap\phi WP$	$\max(0,Fw\phi)$	$\frac{Fw\phi}{2}$	$\frac{1}{8}\max(0,Fe\phi)+\frac{3}{4}\max(0,Fw\phi)+\frac{3}{8}\min(0,Fw\phi)$	$\frac{\Delta y\nu}{\Delta x}$	$\frac{4\Delta y\nu}{3\Delta x}$
$Ap\phi WN$	0	0	0	0	0
$Ap\phi WNN$	0	0	0	0	0
$Ap\phi PSS$	0	0	$-\frac{1}{8}\max(0,Fs\phi)$	0	$-\frac{\Delta x\nu}{12\Delta y}$
$Ap\phi PS$	$\max(0,Fs\phi)$	$\frac{Fs\phi}{2}$	$\frac{1}{8}\max(0,Fn\phi)+\frac{3}{4}\max(0,Fs\phi)+\frac{3}{8}\min(0,Fs\phi)$	$\frac{\Delta x\nu}{\Delta y}$	$\frac{4\Delta x\nu}{3\Delta y}$
$Ap\phi PP$	$\sum_{nb}Ap\phi$	$\sum_{nb}Ap\phi$	$\sum_{nb}Ap\phi$	$\sum_{nb}Ap\phi$	$\sum_{nb}Ap\phi$
$Ap\phi PN$	$-\min(0,Fn\phi)$	$-\frac{Fn\phi}{2}$	$\frac{1}{8}(-3)\max(0,Fn\phi)-\frac{3}{4}\min(0,Fn\phi)-\frac{1}{8}\min(0,Fs\phi)$	$\frac{\Delta x\nu}{\Delta y}$	$\frac{4\Delta x\nu}{3\Delta y}$
$Ap\phi PNN$	0	0	$\frac{1}{8}\min(0,Fn\phi)$	0	$-\frac{\Delta x\nu}{12\Delta y}$
$Ap\phi ESS$	0	0	0	0	0
$Ap\phi ES$	0	0	0	0	0
$Ap\phi EP$	$-\min(0,Fe\phi)$	$-\frac{Fe\phi}{2}$	$\frac{1}{8}(-3)\max(0,Fe\phi)-\frac{3}{4}\min(0,Fe\phi)-\frac{1}{8}\min(0,Fw\phi)$	$\frac{\Delta y\nu}{\Delta x}$	$\frac{4\Delta y\nu}{3\Delta x}$
$Ap\phi EN$	0	0	0	0	0
$Ap\phi ENN$	0	0	0	0	0
$Ap\phi EESS$	0	0	0	0	0
$Ap\phi EES$	0	0	0	0	0
$Ap\phi EEP$	0	0	$\frac{1}{8}\min(0,Fe\phi)$	0	$-\frac{\Delta y\nu}{12\Delta x}$
$Ap\phi EEN$	0	0	0	0	0
$Ap\phi EENN$	0	0	0	0	0

3 Symmetric Coupled Gauss-Seidel (SCGS)

So far we have presented the set of differential equations that govern the fluid flow, and we have also discretized them using the FVM and applying different schemes (for the convection and diffusion terms).

In the following we will discuss the Symmetric Coupled Gauss-Seidel (SCGS) method by [3, Vanka (1986)], which combine all the information we have presented so far, and will allow us to solve the discretized equations iteratively.

Notice that when we say *solve the equations*, we mean to find the values of the velocity components and the pressure at each cell of the domain, namely $U(x_i, y_j)$ & $V(x_i, y_j)$ & $P(x_i, y_j)$, so to have a complete description of the flow field.

3.1 Variable Correction Concept

Before presenting the SCGS method itself, it's now useful to introduce the concept of variable correction, which is crucial for the understanding of the method.

The idea here is to observe a generic current state of the system $\phi^{(n)}$, as the sum of a known part $\phi^{(n-1)}$ and a correction $\delta\phi^{(n)}$:

$$\phi^{(n)} = \phi^{(n-1)} + \delta\phi^{(n)} \tag{77}$$

Where ϕ is a generic variable, and n is the iteration index.

For simplicity of notation, we will drop the iteration index in the following, and we will rewrite Equation 77 as:

$$\phi = \phi^* + \phi' \tag{78}$$

Where ϕ^* is the known part, and ϕ' is the correction.

By doing so, we can rewrite the state of the system at the current iteration as:

$$u_{i,j} = u_{i,j}^* + u_{i,j}' \tag{79}$$

$$v_{i,j} = v_{i,j}^* + v_{i,j}' \tag{80}$$

$$p_{i,j} = p_{i,j}^* + p_{i,j}' \tag{81}$$

3.2 Equations Coupling

At this point, one could ask how the three equations we have presented so far are related to each other. The answer is in the continuity equation, which is the key to couple the velocity and pressure fields (and indeed, have a close form of the system with 3 equations 12 35 36 and 3 unknowns).

Given that the continuity equation is defined for a given $p - CV$ and that involve the all the four velocity components at the faces of the control volume, it should be intuitive that in order to solve the equilibrium across one cell, we need to have a system of equations that involve all the four velocity components.

By recalling the representation with index notation for the $p - CV$, we can compose our system of equations.
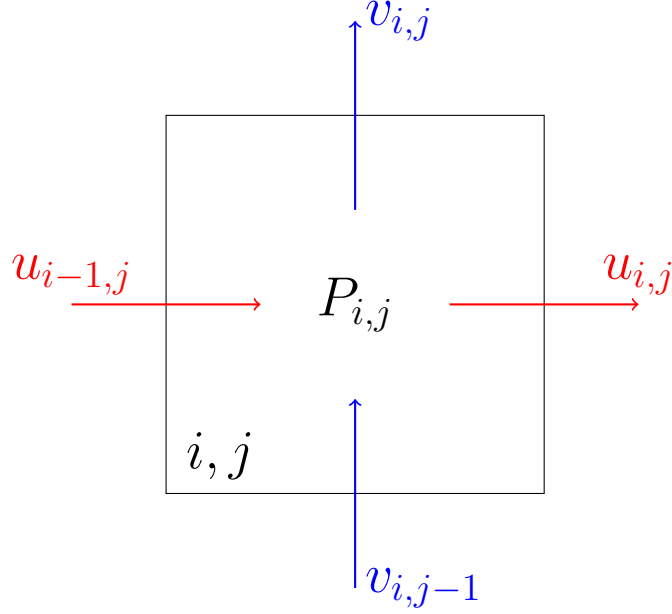
Figure 10: Generic $p - CV$ with the velocity components at the faces.

$$\begin{cases} \text{Mom. } u_{i-1,j} \\ \text{Mom. } u_{i,j} \\ \text{Mom. } v_{i,j-1} \quad = \\ \text{Mom. } v_{i,j} \\ \text{Con.} \end{cases} \begin{cases} (A_P^u)_{i-1,j} u_{i-1,j} & = \sum_{nb} (A_{nb}^u)_{i-1,j} (u_{nb})_{i-1,j} + (p_{i-1,j} - p_{i,j}) \Delta y \\ (A_P^u)_{i,j} u_{i,j} & = \sum_{nb} (A_{nb}^u)_{i,j} (u_{nb})_{i,j} + (p_{i,j} - p_{i+1,j}) \Delta y \\ (A_P^v)_{i,j-1} v_{i,j-1} & = \sum_{nb} (A_{nb}^v)_{i,j-1} (v_{nb})_{i,j-1} + (p_{i,j-1} - p_{i,j}) \Delta x \\ (A_P^v)_{i,j} v_{i,j} & = \sum_{nb} (A_{nb}^v)_{i,j} (v_{nb})_{i,j} + (p_{i,j} - p_{i,j+1}) \Delta x \\ (u_{i,j} - u_{i-1,j}) \Delta y + (v_{i,j} - v_{i,j-1}) \Delta x & = 0 \end{cases}$$

(82)

The system of equations is composed by 5 equations and 5 unknowns $(u_{i-1,j}, u_{i,j}, v_{i,j-1}, v_{i,j}, p_{i,j})$, and it's the result of the coupling of the momentum and continuity equations across a generic $p - CV$.

3.3 Residual Concept

Being the SCGS an iterative method, it's crucial to introduce the concept of residual, which is the key to understand the convergence of the method.

The residual is the difference between the left-hand side and the right-hand side of the discretized equations, and it's a measure of the error of the current state of the system.

For our system of 5 equations, we can define 5 residuals, one for each equation.

To simplify the set of equations (and have a simpler matrix resolution afterwards), we can choose to apply the correction to only some variables, and not to all of them.

In particular, we can choose to neglect the correction for the velocity and pressure components regarding the neighbor cells, and apply the correction only to the pressure and the velocity components at the current cell.

By doing so, we can rewrite the system of equations as:

$$(A_P^u)_{i-1,j}(u_{i-1,j}^* + u_{i-1,j}') = \sum_{nb}(A_{nb}^u)_{i-1,j}(u_{nb}^*)_{i-1,j} + (p_{i-1,j}^* - (p_{i,j}^* + p_{i,j}'))\Delta y \tag{83}$$

$$(A_P^u)_{i,j}(u_{i,j}^* + u_{i,j}') = \sum_{nb}(A_{nb}^u)_{i,j}(u_{nb}^*)_{i,j} + ((p_{i,j}^* + p_{i,j}') - p_{i+1,j}^*)\Delta y \tag{84}$$

$$(A_P^v)_{i,j-1}(v_{i,j-1}^* + v_{i,j-1}') = \sum_{nb}(A_{nb}^v)_{i,j-1}(v_{nb}^*)_{i,j-1} + (p_{i,j-1}^* - (p_{i,j}^* + p_{i,j}'))\Delta x \tag{85}$$

$$(A_P^v)_{i,j}(v_{i,j}^* + v_{i,j}') = \sum_{nb}(A_{nb}^v)_{i,j}(v_{nb}^*)_{i,j} + ((p_{i,j}^* + p_{i,j}') - p_{i,j+1}^*)\Delta x \tag{86}$$

$$((u_{i,j}^* + u_{i,j}') - (u_{i-1,j}^* + u_{i-1,j}'))\Delta y + ((v_{i,j}^* + v_{i,j}') - (v_{i,j-1}^* + v_{i,j-1}'))\Delta x = 0 \tag{87}$$

As explained in Subsection 3.1, the * superscript is used to indicate the value of the variable at the previous iteration, and the $'$ superscript is used to indicate the correction to apply to the variable. This means that the ϕ^* are known values, and the ϕ' are the unknowns to solve for.

We can now rearrange each equation to have the unknowns on the left-hand side and the known on the right-hand side:

$$(A_P^u)_{i-1,j}u'_{i-1,j} + p'_{i,j}\Delta y = R_{i-1,j}^u \tag{88}$$

$$(A_P^u)_{i,j}u'_{i,j} - p'_{i,j}\Delta y = R_{i,j}^u \tag{89}$$

$$(A_P^v)_{i,j-1}v'_{i,j-1} + p'_{i,j}\Delta x = R_{i,j-1}^v \tag{90}$$

$$(A_P^v)_{i,j}v'_{i,j} - p'_{i,j}\Delta x = R_{i,j}^v \tag{91}$$

$$(u'_{i,j} - u'_{i-1,j})\Delta y + (v'_{i,j} - v'_{i,j-1})\Delta x = R_{i,j}^c \tag{92}$$

Where $R_{i,j}^\phi$ is the residual of the ϕ equation at the i, j cell.

The residuals, computed from the previous system by moving the known terms to the right-hand side, are defined as:

$$R_{i-1,j}^u = \sum_{nb}(A_{nb}^u)_{i-1,j}(u_{nb}^*)_{i-1,j} - (A_P^u)_{i-1,j}u_{i-1,j}^* + (p_{i-1,j}^* - p_{i,j}^*)\Delta y \tag{93}$$

$$R_{i,j}^u = \sum_{nb}(A_{nb}^u)_{i,j}(u_{nb}^*)_{i,j} - (A_P^u)_{i,j}u_{i,j}^* + (p_{i,j}^* - p_{i+1,j}^*)\Delta y \tag{94}$$

$$R_{i,j-1}^v = \sum_{nb}(A_{nb}^v)_{i,j-1}(v_{nb}^*)_{i,j-1} - (A_P^v)_{i,j-1}v_{i,j-1}^* + (p_{i,j-1}^* - p_{i,j}^*)\Delta x \tag{95}$$

$$R_{i,j}^v = \sum_{nb}(A_{nb}^v)_{i,j}(v_{nb}^*)_{i,j} - (A_P^v)_{i,j}v_{i,j}^* + (p_{i,j}^* - p_{i,j+1}^*)\Delta x \tag{96}$$

$$R_{i,j}^c = -\left[(u_{i,j}^* - u_{i-1,j}^*)\Delta y + (v_{i,j}^* - v_{i,j-1}^*)\Delta x\right] \tag{97}$$

3.4   Gauss-Seidel Iterative Method

Before, proceeding with the solution of the SCGS method, we need to introduce the Gauss-Seidel Iterative Method (GS) method.

The GS method is an iterative method used to solve a system of linear equations, and it's based on the idea of solving one equation at a time, and using the updated values of the variables to solve the next equation.

For our purpose, is interesting to observe how the introduction of the 'under-relaxation' factor α can help to improve the convergence of the method.

In particular, we can apply the GS method to solve a similar equation of our system, as for example:

$$(A_P^\phi)\phi_P = \sum_{nb}(A_{nb}^\phi)\phi_{nb} + S^\phi \tag{98}$$

Where ϕ is a generic variable, and S^ϕ is the source term of the equation.

The GS method can be written as:

$$\phi_P = \frac{\sum_{nb}(A_{nb}^\phi)\phi_{nb}^* + S^\phi}{A_P^\phi} + \phi_P^* \tag{99}$$

$$\phi_P = \alpha\left(\frac{\sum_{nb}(A_{nb}^\phi)\phi_{nb}^* + S^\phi}{A_P^\phi} - \phi_P^*\right) + \phi_P^* \tag{100}$$

Where ϕ_P^* is the value of the variable at the previous iteration, and α is the under-relaxation factor.

By rearranging the last equation, we have:

$$\phi_P' = \frac{R_P^\phi}{\left(\frac{A_P^\phi}{\alpha_\phi}\right)} \tag{101}$$

Where ϕ_P' is the correction to apply to the variable, and R_P^ϕ is the residual of the equation at the P cell.

This result will be useful in the next section.

3.5 Vanka's approach

The SCGS method is based on the idea of solving the equations in a coupled way.
This means that for each iteration, we will obtain the correction for each of the 5 variables simultaneously. To make this possible, we need to introduce a new matrix, called the **Vanka matrix [3]**.
Considering the system of equations 92, we can write them in a matrix form as:

$$
\begin{bmatrix}
(A_P^u)_{i-1,j} & 0 & 0 & 0 & \Delta y \\
0 & (A_P^u)_{i,j} & 0 & 0 & -\Delta y \\
0 & 0 & (A_P^v)_{i,j-1} & 0 & \Delta x \\
0 & 0 & 0 & (A_P^v)_{i,j} & -\Delta x \\
-\Delta y & \Delta y & -\Delta x & \Delta x & 0
\end{bmatrix}
\begin{bmatrix}
u'_{i-1,j} \\
u'_{i,j} \\
v'_{i,j-1} \\
v'_{i,j} \\
p'_{i,j}
\end{bmatrix}
=
\begin{bmatrix}
R^u_{i-1,j} \\
R^u_{i,j} \\
R^v_{i,j-1} \\
R^v_{i,j} \\
R^c_{i,j}
\end{bmatrix}
\tag{102}
$$

As we can see, the Vanka matrix is a 5×5 matrix, and it's a function of the coefficients of the discretized equations, and of the grid spacing.
Our unknown is the vector of the corrections, being the A_P^ϕ coefficients and the R^ϕ residuals known from the previous iteration.
Given that, we can solve the system of equations 102 to obtain the corrections for the velocity and pressure components at each cell of the domain.
To do so, we must invert the Vanka matrix, and multiply it by the residuals vector:

$$
\left[\phi'\right] = \left[A^\phi\right]^{-1}\left[R^\phi\right]
\tag{103}
$$

As stated in [3], the inverse of the Vanka matrix can be computed analytically using the following algorithm:

$$
r_i = \frac{a_{5i}}{a_{ii}}, \quad i = 1,2,3,4
\tag{104}
$$

$$
DEN = \sum_{i=1}^{4} a_{5i} r_i
\tag{105}
$$

$$
x_5 = \frac{\left[\sum_{i=1}^{4} r_i b_i - b_5\right]}{DEN}
\tag{106}
$$

$$
x_i = \frac{b_i - a_{i5} x_5}{a_{ii}}, \quad i = 1,2,3,4
\tag{107}
$$

Where a_{ij} are the elements of the Vanka matrix A^ϕ, b_i are the elements of the residuals vector R^ϕ, and x_i are the elements of the corrections vector ϕ'.
In case we want to apply an under-relaxation factor to the corrections, we can modify Vanka's Matrix as follows:

$$
\begin{bmatrix}
\frac{(A_P^u)_{i-1,j}}{\alpha_u} & 0 & 0 & 0 & \Delta y \\
0 & \frac{(A_P^u)_{i,j}}{\alpha_u} & 0 & 0 & -\Delta y \\
0 & 0 & \frac{(A_P^v)_{i,j-1}}{\alpha_v} & 0 & \Delta x \\
0 & 0 & 0 & \frac{(A_P^v)_{i,j}}{\alpha_v} & -\Delta x \\
-\Delta y & \Delta y & -\Delta x & \Delta x & 0
\end{bmatrix}
\tag{108}
$$

Where α_u and α_v are the under-relaxation factors for the velocity components u and v respectively.

3.6 Boundary conditions for Lid-Driven Cavity problem

So far, we have presented all the tools and concepts required to solve a generic fluid flow problem (except for the algorithm itself) that respect the hypothesis and conditions stated at the beginning of Section 1.
In this section we will present the boundary conditions that will be used to solve the Lid-Driven Cavity problem and their implementation inside the SCGS method.

3.6.1 Ghosts cells

Consider that our domain has already been discretized (subdivided) into $N_x \times N_y$ cells.
If we think of the scheme for both the convection and diffusion presented before, we can see that at the boundary of our domain, the coefficients $A_p \phi$ 1 involve cells that are outside the domain. For this reason, we need to add a layer of cells outside the physical domain, called ghost cells, to calculate the coefficients at the boundary.

The number of ghost cells required depends on the order of the scheme used to calculate the coefficients. For a second-order scheme, we need at least one ghost cell, and for a fourth-order scheme, we need at least two ghost cells.

Here, and also in the implementation of the SCGS method in the code, we will consider the possibility to adopt a mixed strategy scheme and use:

- Away from boundaries: higher-order schemes

- Near boundaries: UDS scheme for convection and 2nd-order scheme for diffusion

By doing so, we ensure that at the boundaries both the schemes will be applicable with just a single layer of ghost cells.
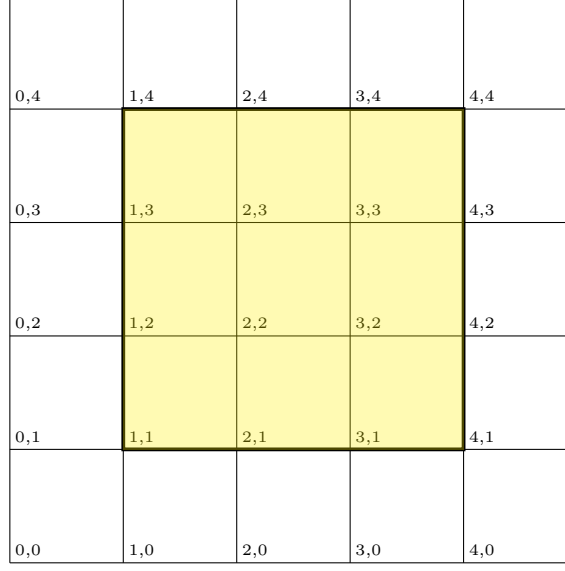

Figure 11: Physical domain and ghost cells

In Figure 11, the yellow area represents the physical domain, while the frame surrounding it represents the ghost cells.

As we can see, the indexes of the cells stick to the physical domain, while the ghost cells may have negative indexes.

In the case above, we can declare some of the problem that will be recalled during the following sections:

- Physical domain: $N_x \times N_y$ grid. In the example above, $N_x = N_y = 3$

- Physical domain indexes: from $(1, 1)$ to $(3, 3)$. In general, from $(1, 1)$ to (N_x, N_y).

- Ghost cells layer: may have different sizes along the x and y directions. In the example above, the ghost cells layer has a size of $N_g = 1$ cell in both directions. We will stick with this size for the rest of the document.

- Ghost cells indexes: may have negative indexes. In the example above, the ghost cells have indexes from $(0, 0)$ to $(4, 4)$, that is equivalent to $(1 - N_g, 1 - N_g)$ to $(N_x + 2 * N_g - 1, N_y + 2 * N_g - 1)$.

We can now proceed to identify the boundary conditions for the Lid-Driven Cavity problem.

3.6.2 No-slip condition

As boundary conditions for the Lid-Driven Cavity problem (and for most of the fluid flow problems), we have the no-slip condition, which gives us the constraint for two direction of the velocity field:

- Component normal to the wall: given a condition of impenetrability, the velocity component normal to the wall must be zero (no-penetration)

- Component tangential to the wall: given a condition of no-slip, the velocity component tangential to the wall must equal to the wall velocity

As an example, we can consider the cell in position $(1, N_y)$, which is the top-left corner of the physical domain (Figure 11 for reference).
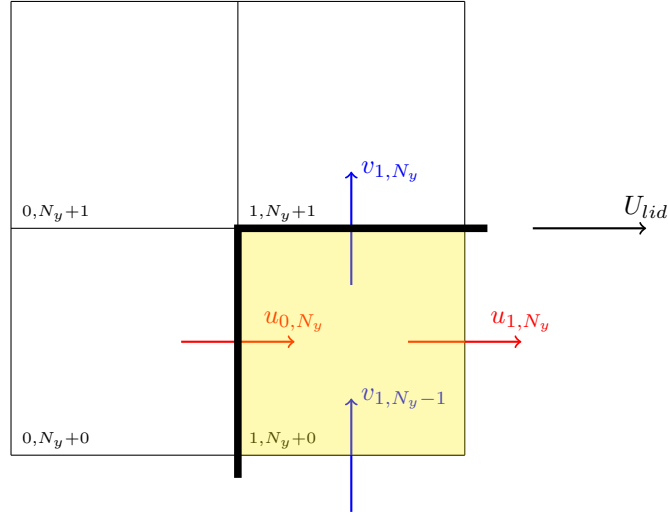

Figure 12: Boundary conditions for the cell $(1, N_y)$

In this case, we must accomplish the following conditions:

- $u_{0,N_y} = 0$ (no-penetration)

- $v_{1,N_y} = 0$ (no-penetration)

- $u_{1,N_y} = U_{lid}$ (no-slip)

Intuitively, we can understand that similar conditions must be applied to the other walls of the domain given the correction of the indexes and the direction of the velocity field.
In general, we can state that the boundary conditions for the Lid-Driven Cavity problem are:

- Bottom wall: $v_{i,0} = 0$ (no-penetration) + $u_{i,1} = 0$ (no-slip)

- Left wall: $u_{0,j} = 0$ (no-penetration) + $v_{1,j} = 0$ (no-slip)

- Top wall: $v_{i,N_y} = 0$ (no-penetration) + $u_{i,N_y} = U_{lid}$ (no-slip)

- Right wall: $u_{N_x,j} = 0$ (no-penetration) + $v_{N_x,j} = 0$ (no-slip)

We can now proceed to see how these conditions are accounted for in the SCGS method.

3.6.3 Boundary conditions inside SCGS method

By adopting the SCGS method, we can impose the boundary conditions in the following way:

- No-penetration condition: impose specific coefficients in the Vanka matrix to be zero

- No-slip condition: impose specific velocity values in the ghost cells

Moreover, as we are going to see, the no-penetration condition imply the velocity to be null and can be imposed to the system before solving the linear system of equations. Instead, given the mathematical formulation of the no-slip condition, we can impose the velocity values in the ghost cells only after solving the linear system of equations (and will affect the next iteration).

No-penetration condition The no-penetration condition imply the normal velocity component at the boundary to be null. This means that if we start from an initial guess of the velocity field equal to zero everywhere, we must ensure that correction of the velocity field at the boundary is null for each iteration.
Taken for example the cell $(1, N_y)$ as in Figure 12, we know that we must impose $u_{0,N_y} = 0$ & $v_{1,N_y} = 0$, which means that the correction of the velocity field at the boundary must be null as well $u'_{0,N_y} = 0$ & $v'_{1,N_y} = 0$.
By recalling the formulation for both the continuity and momentum equations, we can see that multiple terms must be canceled out (known null solution).

For the **continuity equation** written at the cell $(1, N_y)$, we have:

$$-u'_{0,N_y}\Delta y + u'_{1,N_y}\Delta y - v'_{1,N_y-1}\Delta x + v'_{1,N_y}\Delta x = R^c_{1,N_y} \tag{109}$$

By imposing $[A^\phi]_{5,1} = \Delta y = 0$, we can cancel out the first term, and by imposing $[A^\phi]_{5,4} = \Delta x = 0$, we can cancel out the third term.

In this way, even if we are not imposing the no-penetration condition directly, we are imposing an equivalent condition that will lead to the same result.

More in general, given the physical domain of size $N_x \times N_y$, we can impose the no-penetration condition for the continuity equation by setting:

$$@i = 1, u'_{i-1,j} = 0 \rightarrow [A^\phi]_{5,1} = -\Delta y = 0 \quad \forall j = 1, 2, ..., N_y \tag{110}$$

$$@i = N_x, u'_{i,j} = 0 \rightarrow [A^\phi]_{5,2} = \Delta y = 0 \quad \forall j = 1, 2, ..., N_y \tag{111}$$

$$@j = 1, v'_{i,j-1} = 0 \rightarrow [A^\phi]_{5,3} = -\Delta x = 0 \quad \forall i = 1, 2, ..., N_x \tag{112}$$

$$@j = N_y, v'_{i,j} = 0 \rightarrow [A^\phi]_{5,4} = \Delta x = 0 \quad \forall i = 1, 2, ..., N_x \tag{113}$$

For the **momentum equations**, written at the cell $(1, N_y)$ for the two components that are controlled by the no-penetration condition, we have:

$$(A^u_P)_{0,N_y} u'_{0,N_y} + p'_{0,N_y}\Delta y = R^u_{0,N_y} \tag{114}$$

$$(A^v_P)_{1,N_y} v'_{1,N_y} - p'_{1,N_y}\Delta x = R^v_{1,N_y} \tag{115}$$

By imposing $[A^\phi]_{1,5} = \Delta y = 0$ and $[R^\phi]_1 = 0$, we can cancel out the first equation, and by imposing $[A^\phi]_{4,5} = -\Delta x = 0$ and $[R^\phi]_4 = 0$, we can cancel out the second equation.

More in general, given the physical domain of size $N_x \times N_y$, we can impose the no-penetration condition for the momentum equations by setting:

$$@i = 1, u'_{i-1,j} = 0 \rightarrow [A^\phi]_{1,5} = \Delta y = 0 \quad \forall j = 1, 2, ..., N_y \tag{116}$$

$$@i = N_x, u'_{i,j} = 0 \rightarrow [A^\phi]_{2,5} = -\Delta y = 0 \quad \forall j = 1, 2, ..., N_y \tag{117}$$

$$@j = 1, v'_{i,j-1} = 0 \rightarrow [A^\phi]_{3,5} = \Delta x = 0 \quad \forall i = 1, 2, ..., N_x \tag{118}$$

$$@j = N_y, v'_{i,j} = 0 \rightarrow [A^\phi]_{4,5} = -\Delta x = 0 \quad \forall i = 1, 2, ..., N_x \tag{119}$$

No-slip condition The idea behind the no-slip condition for the tangential velocity component is to impose the velocity field in the ghost cells so that the interpolation with the velocity field in the physical domain gives the correct value at the boundary.

As an example, we can consider the cell $(1, N_y)$ as in Figure 12.

We know that we must impose $u_{1,N_y} = U_{lid}$. Given that the condition can't be imposed directly to the correction term as for the no-penetration condition, here we have to solve the system and then impose the condition over the ghost cells velocity based on the current solution of the velocity field.

In particular, we can think of the wall velocity as the average of the velocity in the ghost cell and the velocity in the physical domain. This implies to impose the following condition for the cell $(1, N_y)$:

$$U_{lid} = \frac{1}{2}(u_{1,N_y} + u_{1,N_y+1}) \rightarrow u_{1,N_y+1} = 2U_{lid} - u_{1,N_y} \tag{120}$$

More in general, given the physical domain of size $N_x \times N_y$, we can impose the no-slip condition for the velocity field by setting:

$$@i = 1, v_{0,j} = -v_{1,j} \quad \forall j = 1, 2, ..., N_y \tag{121}$$

$$@i = N_x, v_{N_x+1,j} = -v_{N_x,j} \quad \forall j = 1, 2, ..., N_y \tag{122}$$

$$@j = 1, u_{i,0} = -u_{i,1} \quad \forall i = 1, 2, ..., N_x \tag{123}$$

$$@j = N_y, u_{i,N_y+1} = 2U_{lid} - u_{i,N_y} \quad \forall i = 1, 2, ..., N_x \tag{124}$$

3.7 Convergence criterion

Being the SCGS method an iterative method, we need to define a criterion to stop the iterations, and to consider the solution as converged.

We could have adopted many criteria, but probably the most intuitive one is to monitor the residuals of the equations over the entire domain for each iteration, and to stop the cycle when the maximum of the residual is below a certain threshold.

So far we have defined 5 residuals, one for each equation, and we can define other 3 residuals derived from the previous ones, as follows:

$$\text{Continuity residual}: R^p = |R^c|$$
$$\text{Momentum residual u}: R^u = \frac{\left(|R_{i-1}^u| + |R_i^u|\right)}{2} \tag{125}$$
$$\text{Momentum residual v}: R^v = \frac{\left(|R_{j-1}^v| + |R_j^v|\right)}{2}$$

By doing so, our convergence criterion will be:

$$\max\left(R^p, R^u, R^v\right) < \varepsilon \tag{126}$$

Where ε is the threshold and may vary between 10^{-3} and 10^{-6}, depending on the problem and the computational resources available.

4 Semi-Implicit Method for Pressure Linked Equations (SIMPLE)

After having presented the SCGS algorithm, we will now discuss the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) method.

The SIMPLE algorithm has been proposed by Patankar and Spalding in 1972 [2] and is one of the most used algorithms for solving the Navier-Stokes equations in the field of CFD.

With respect to the coupled approach of the SCGS method, the SIMPLE method decouples the momentum and continuity equations, solving them sequentially. This approach makes also easier to possibly add more equations to the system, such as the energy equation or the turbulence model, given that would be solved independently of the momentum and continuity equations.

Many of the concepts discussed for the SCGS methods, such as variable correction (Section 3.1) or the residual (Section 3.3), or again the Guass-Seide iterative method (Section 3.4), are still valid and we will report in following just their application to the SIMPLE algorithm without repeating the core idea behind them.

The major and fundamental difference between the SCGS and the SIMPLE methods is in the way the equation are coupled together and solved. In the SCGS method, we have seen in Section 3.2 and Section 3.5 that the momentum and continuity equations are coupled together and solved simultaneously. In the SIMPLE method, instead, the momentum and continuity equations are decoupled and solved sequentially.

4.1 Derivation of the pressure correction equation

As we have said in the introduction, the SIMPLE method decouples the momentum and continuity equations, solving them sequentially. The system of equations is now coupled through the pressure correction term, which can be computed from the residuals of the continuity equation.

In the following, we will derive the pressure correction equation starting from the discretized momentum and continuity equations.

Considering of applying correction to every velocity and pressure term, we obtain the following equations:

$$(A_P^u)_{i-1,j}(u_{i-1,j}^* + u_{i-1,j}') = \sum_{nb}(A_{nb}^u)_{i-1,j}(u_{nb}^* + u_{nb}')_{i-1,j} + ((p_{i-1,j}^* + p_{i-1,j}') - (p_{i,j}^* + p_{i,j}'))\Delta y \tag{127}$$

$$(A_P^u)_{i,j}(u_{i,j}^* + u_{i,j}') = \sum_{nb}(A_{nb}^u)_{i,j}(u_{nb}^* + u_{nb}')_{i,j} + ((p_{i,j}^* + p_{i,j}') - (p_{i+1,j}^* + p_{i+1,j}'))\Delta y \tag{128}$$

$$(A_P^v)_{i,j-1}(v_{i,j-1}^* + v_{i,j-1}') = \sum_{nb}(A_{nb}^v)_{i,j-1}(v_{nb}^* + v_{nb}')_{i,j-1} + ((p_{i,j-1}^* + p_{i,j-1}') - (p_{i,j}^* + p_{i,j}'))\Delta x \tag{129}$$

$$(A_P^v)_{i,j}(v_{i,j}^* + v_{i,j}') = \sum_{nb}(A_{nb}^v)_{i,j}(v_{nb}^* + v_{nb}')_{i,j} + ((p_{i,j}^* + p_{i,j}') - (p_{i,j+1}^* + p_{i,j+1}'))\Delta x \tag{130}$$

$$((u_{i,j}^* + u_{i,j}') - (u_{i-1,j}^* + u_{i-1,j}'))\Delta y + ((v_{i,j}^* + v_{i,j}') - (v_{i,j-1}^* + v_{i,j-1}'))\Delta x = 0 \tag{131}$$

Supposing now to be at the converged solution, and so when the correction terms are approximately zero, we subtract the system of converged equations from the above system, obtaining the following equations:

$$(A_P^u)_{i-1,j}(u'_{i-1,j}) = \sum_{nb}(A_{nb}^u)_{i-1,j}(u'_{nb})_{i-1,j} + ((p'_{i-1,j}) - (p'_{i,j}))\Delta y + R_{i-1,j}^u \tag{132}$$

$$(A_P^u)_{i,j}(u'_{i,j}) = \sum_{nb}(A_{nb}^u)_{i,j}(u'_{nb})_{i,j} + ((p'_{i,j}) - (p'_{i+1,j}))\Delta y + R_{i,j}^u \tag{133}$$

$$(A_P^v)_{i,j-1}(v'_{i,j-1}) = \sum_{nb}(A_{nb}^v)_{i,j-1}(v'_{nb})_{i,j-1} + ((p'_{i,j-1}) - (p'_{i,j}))\Delta x + R_{i,j-1}^v \tag{134}$$

$$(A_P^v)_{i,j}(v'_{i,j}) = \sum_{nb}(A_{nb}^v)_{i,j}(v'_{nb})_{i,j} + ((p'_{i,j}) - (p'_{i,j+1}))\Delta x + R_{i,j}^v \tag{135}$$

$$(u'_{i,j} - u'_{i-1,j})\Delta y + (v'_{i,j} - v'_{i,j-1})\Delta x = R_{i,j}^c \tag{136}$$

Moreover, we can neglect the residual terms R^u and R^v, and the correction terms u'_{nb} and v'_{nb}, given that at the convergence of the solution they are approximately zero.

$$(A_P^u)_{i-1,j}(u'_{i-1,j}) = ((p'_{i-1,j}) - (p'_{i,j}))\Delta y \tag{137}$$
$$(A_P^u)_{i,j}(u'_{i,j}) = ((p'_{i,j}) - (p'_{i+1,j}))\Delta y \tag{138}$$
$$(A_P^v)_{i,j-1}(v'_{i,j-1}) = ((p'_{i,j-1}) - (p'_{i,j}))\Delta x \tag{139}$$
$$(A_P^v)_{i,j}(v'_{i,j}) = ((p'_{i,j}) - (p'_{i,j+1}))\Delta x \tag{140}$$
$$(u'_{i,j} - u'_{i-1,j})\Delta y + (v'_{i,j} - v'_{i,j-1})\Delta x = R_{i,j}^c \tag{141}$$

By doing so, we can now rearrange the equations so to write every velocity correction as a function of the pressure correction term:

$$u'_{i,j} = \frac{\Delta y}{(A_P^u)_{i,j}}(p'_{i,j} - p'_{i+1,j}) \tag{142}$$

$$u'_{i-1,j} = \frac{\Delta y}{(A_P^u)_{i-1,j}}(p'_{i-1,j} - p'_{i,j}) \tag{143}$$

$$v'_{i,j} = \frac{\Delta x}{(A_P^v)_{i,j}}(p'_{i,j} - p'_{i,j+1}) \tag{144}$$

$$v'_{i,j-1} = \frac{\Delta x}{(A_P^v)_{i,j-1}}(p'_{i,j-1} - p'_{i,j}) \tag{145}$$

Similarly, we can rearrange the continuity equation as a function of the pressure correction term only:

$$\left(\frac{\Delta y}{(A_P^u)_{i,j}}(p'_{i,j} - p'_{i+1,j}) - \frac{\Delta y}{(A_P^u)_{i-1,j}}(p'_{i-1,j} - p'_{i,j})\right)\Delta y + \ldots \tag{146}$$

$$+ \left(\frac{\Delta x}{(A_P^v)_{i,j}}(p'_{i,j} - p'_{i,j+1}) - \frac{\Delta x}{(A_P^v)_{i,j-1}}(p'_{i,j-1} - p'_{i,j})\right)\Delta x = R_{i,j}^c \tag{147}$$

We can now explicitly write the pressure coefficients by rearranging the equations above:

$$(A_E^p)_{i,j} = \frac{(\Delta y)^2}{(A_P^u)_{i,j}} \tag{148}$$

$$(A_W^p)_{i,j} = \frac{(\Delta y)^2}{(A_P^u)_{i-1,j}} \tag{149}$$

$$(A_N^p)_{i,j} = \frac{(\Delta x)^2}{(A_P^v)_{i,j}} \tag{150}$$

$$(A_S^p)_{i,j} = \frac{(\Delta x)^2}{(A_P^v)_{i,j-1}} \tag{151}$$

$$(A_P^p)_{i,j} = (A_E^p)_{i,j} + (A_W^p)_{i,j} + (A_N^p)_{i,j} + (A_S^p)_{i,j} \tag{152}$$

The pressure correction equation then becomes:

$$(A_P^p)_{i,j}p'_{i,j} = \sum_{nb}(A_{nb}^p)_{i,j}p'_{nb} + R_{i,j}^c \tag{153}$$

25

4.2   Algorithm

As we have said at the beginning of this section, the SIMPLE method decouples the momentum and continuity equations, solving them sequentially.

4.2.1   Initialization

At first, all the states $(u*, v*, p*)$ are initialized to zero.

4.2.2 u Momentum equation

Compute the coefficients   The coefficients A_P^u, A_{nb}^u are computed for the entire domain based on the convection and diffusion schemes adopted (see Section 2.3).

Apply the boundary conditions   The boundary conditions for the velocity field are applied. See section 4.5 for more details.

Application of the Gauss-Seidel method   In order to compute the pressure correction, we need at first to compute the continuity residual $R_{i,j}^c$. To do so, we need to obtain a first approximation of the velocity field by solving the momentum equations.
Here, we apply iteratively the Gauss-Seidel method to solve the momentum equations for all the cells in the domain.

$$u_{i,j}^* = \alpha_u \left(\frac{\sum_{nb}(A_{nb}^u)_{i,j} u_{nb}^* - (p_{i,j}^* - p_{i+1,j}^*)\Delta y}{(A_P^u)_{i,j}} \right) + (1 - \alpha_u)u_{i,j}^* \tag{154}$$

4.3 v Momentum equation

Compute the coefficients   The coefficients A_P^v, A_{nb}^v are computed for the entire domain based on the convection and diffusion schemes adopted (see Section 2.3).

Apply the boundary conditions   The boundary conditions for the velocity field are applied. See section 4.5 for more details.

Application of the Gauss-Seidel method   Similarly to what we have done for the u momentum equation, we now have to apply iteratively the Gauss-Seidel method to solve the momentum equations for all the cells in the domain.

$$v_{i,j}^* = \alpha_v \left(\frac{\sum_{nb}(A_{nb}^v)_{i,j} v_{nb}^* - (p_{i,j}^* - p_{i,j+1}^*)\Delta x}{(A_P^v)_{i,j}} \right) + (1 - \alpha_v)v_{i,j}^* \tag{155}$$

4.4   Pressure correction

Compute the coefficients   The coefficients A_P^p, A_{nb}^p are computed for the entire domain based on A_P^u and A_P^v. See Equations 152 for more details.

Apply the boundary conditions   The boundary conditions for the pressure field are applied. See section 4.5 for more details.

Compute the continuity residual   The continuity residual $R_{i,j}^c$ is computed for all the cells in the domain based on the solution of the momentum equations obtained during the Gauss-Seidel iterations.

$$R_{i,j}^c = -((u_{i,j}^* - u_{i-1,j}^*)\Delta y + (v_{i,j}^* - v_{i,j-1}^*)\Delta x) \tag{156}$$

Application of the Gauss-Seidel method   As we have done for the momentum equations, we now apply iteratively the Gauss-Seidel method to solve the pressure correction equation for all the cells in the domain.

$$p_{i,j}' = \frac{\sum_{nb}(A_{nb}^p)_{i,j} p_{nb}' + R_{i,j}^c}{(A_P^p)_{i,j}} \tag{157}$$

4.4.1 Update the velocity and pressure fields

Finally, based on the pressure correction term, we can update the velocity and pressure fields.

$$p^*_{i,j} = p^*_{i,j} + \alpha_p p'_{i,j} \tag{158}$$

$$u^*_{i,j} = u^*_{i,j} - \alpha_u \frac{\Delta y}{(A^u_P)_{i,j}}(p'_{i,j} - p'_{i+1,j}) \tag{159}$$

$$v^*_{i,j} = v^*_{i,j} - \alpha_v \frac{\Delta x}{(A^v_P)_{i,j}}(p'_{i,j} - p'_{i,j+1}) \tag{160}$$

4.4.2 Convergence criterion

The convergence criterion for the SIMPLE algorithm is based on the residuals of the continuity and pressure correction equations.

In particular, the convergence criterion is satisfied when the maximum value of the sum of all the residuals is below a certain threshold, which is:

$$\max\left(\sum_{i,j}|R^u_{i,j}|, \sum_{i,j}|R^v_{i,j}|, \sum_{i,j}|R^{pp}_{i,j}|\right) < \epsilon \tag{161}$$

Until the convergence criterion is not satisfied, the algorithm will continue to iterate.

4.5 Boundary conditions for Lid-Driven Cavity problem

The boundary conditions for the SIMPLE algorithm are based on the same idea of No-Slip and No-Penetration conditions as for the SCGS algorithm. Their implementation, however, is slightly different.

As we have done for the SCGS algorithm, we will consider the Lid-Driven Cavity problem to explain how the boundary conditions are applied. In particular, we can consider the top-left corner of the domain as represented in Figure 13.
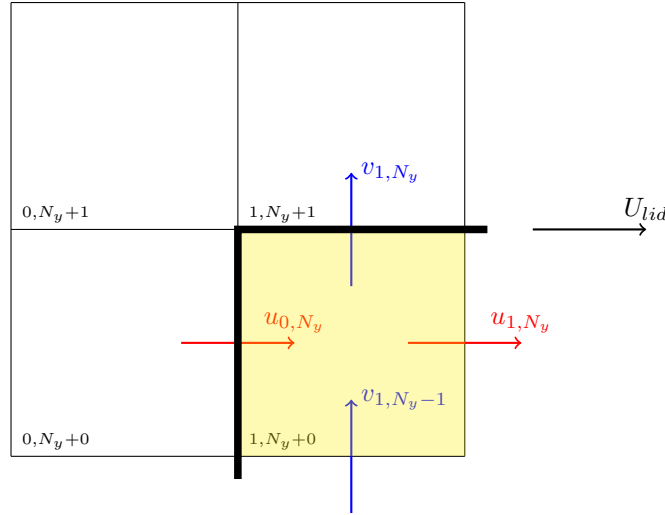

Figure 13: Boundary conditions for the cell $(1, N_y)$

Considering Figure 13, and in particular the cell $(1, N_y)$, we can write the boundary conditions for the velocity field as follows:

$$u_{0,N_y} = 0 \tag{162}$$
$$u_{1,N_y} = U_{lid} \tag{163}$$
$$v_{1,N_y-1} = 0 \tag{164}$$
$$v_{1,N_y} = 0 \tag{165}$$

The boundary conditions for the pressure field are instead:

$$p_{0,N_y} = p_{1,N_y} \tag{166}$$
$$p_{1,N_y} = p_{0,N_y} \tag{167}$$

Similar consideration can be done for the other boundaries of the domain.

4.5.1 Application of the boundary conditions

When it comes to apply the boundary conditions for the SIMPLE algorithm, we need to either modify directly the velocity fields or, for pressure, we need to modify the coefficients of the pressure correction equation.

u **Boundary conditions** In order to satisfy Equations 165, for the u velocity field we can directly modify the velocity field as:

$$u_{0,N_y} = 0 \tag{168}$$
$$U_{lid} = \frac{u_{1,N_y} + u_{1,N_y+1}}{2} \rightarrow u_{1,N_y+1} = 2U_{lid} - u_{1,N_y} \tag{169}$$

v **Boundary conditions** Similarly to what we have done for the u velocity field, we can directly modify the velocity field as:

$$v_{1,N_y} = 0 \tag{170}$$
$$0 = \frac{v_{1,N_y-1} + v_{0,N_y-1}}{2} \rightarrow v_{0,N_y-1} = -v_{1,N_y-1} \tag{171}$$

Pressure Boundary conditions For the pressure field, we need to modify the coefficients of the pressure correction equation.

$$(A_W^p)_{1,N_y} = 0 \tag{172}$$
$$(A_N^p)_{1,N_y} = 0 \tag{173}$$

By doing so, the pressure correction equation for the cell $(1, N_y)$ becomes:

$$(A_P^p)_{1,N_y} p'_{1,N_y} = ((A_E^p)_{1,N_y} + (A_S^p)_{1,N_y}) p'_{1,N_y} = (A_E^p)_{1,N_y} p'_{2,N_y} + (A_S^p)_{1,N_y} p'_{1,N_y-1} + R^c_{1,N_y} \tag{174}$$

5 Code implementation

When it comes to the implementation of the SCGS and SIMPLE algorithms, lots of details need to be taken into account, such as the data structures, the memory allocation, the data exchange between the different modules of the code, and so on.
The complete codebase is available on GitHub at `https://github.com/Bocchio01/CFD_Simulation_Engine`.

6 Results

Despite the effort made in the implementation of the SCGS and SIMPLE algorithms, the results are not satisfactory.
Both the SIMPLE and SCGS are not stable and quickly diverge if the following conditions are met:

- Too high under relaxation factors are chosen (¿0.1)

- High Reynolds number (¿100)

- Coarse mesh

The combination of the above conditions leads to a divergence of the solution.
For SCGS the problem can be spotted as a non-diagonal dominance of Vanka's matrix.
For SIMPLE, an analysis of the residual between iteration, shows that the code starts converging (up to 200 outer iterations) and the solution of the problem seems reasonable. Suddenly some issue in the corner top-right of the domain brings to a rapid divergence of the system.
At this point we believe that the issue is somewhere in the coefficients used for convection, and in particular about the computation of fluxes. However, after several attempts, we were not able to spot the issue.

6.1 Results for the Lid-Driven Cavity Flow

Here we present the results for the Lid-Driven Cavity Flow, computed using both the SCGS and SIMPLE algorithms.

In Figure 14, we leave our reference for the solution of the Lid-Driven Cavity Flow, as computed by [1] for a Reynolds number of $Re = 1000$.
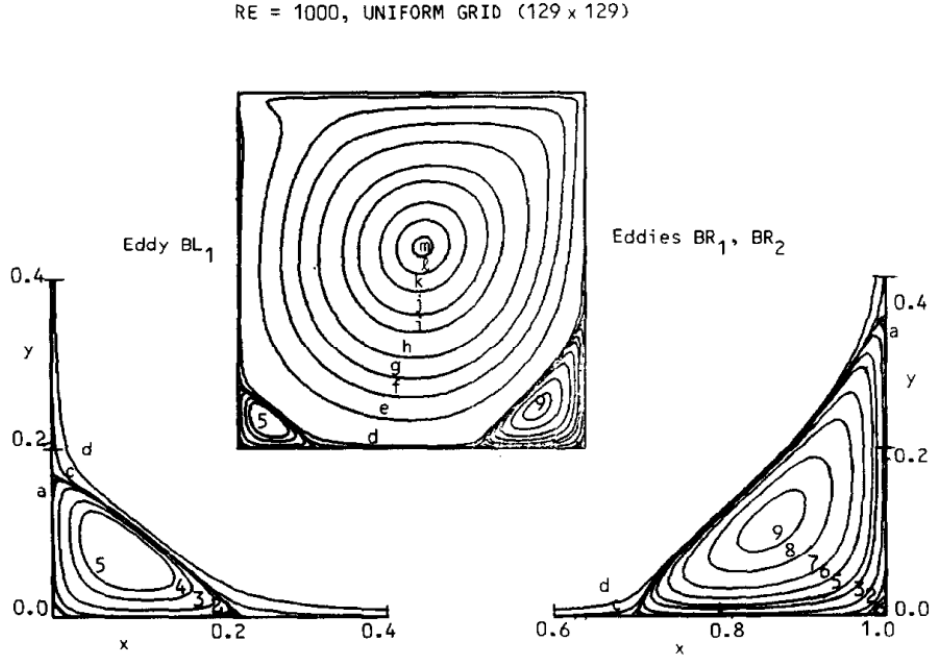

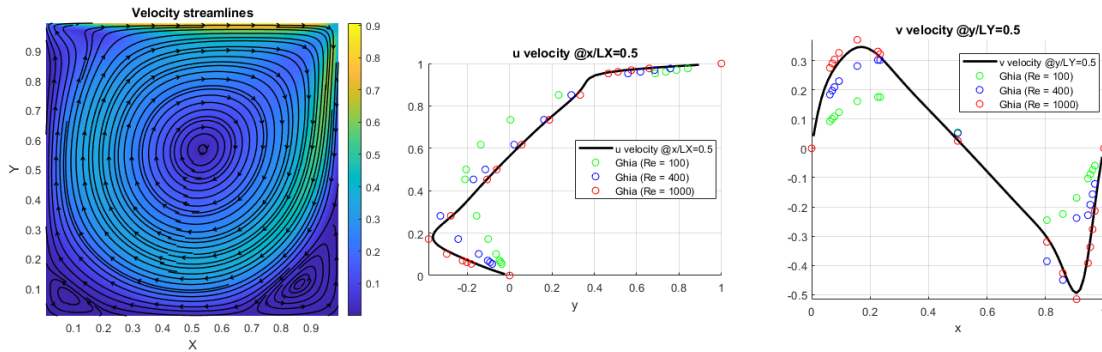Figure 14: Ghia's solution for the lid-driven cavity flow at $Re = 1000$.

Figure 15: SCGS, 80×80 mesh, $Re = 1000$, $URF_{u,v} = 0.08$, QUICK and Second Order schemes for convection and diffusion. Convergence criteria: 10^{-5}. Iterations: 5000, last residual: 0.003421.
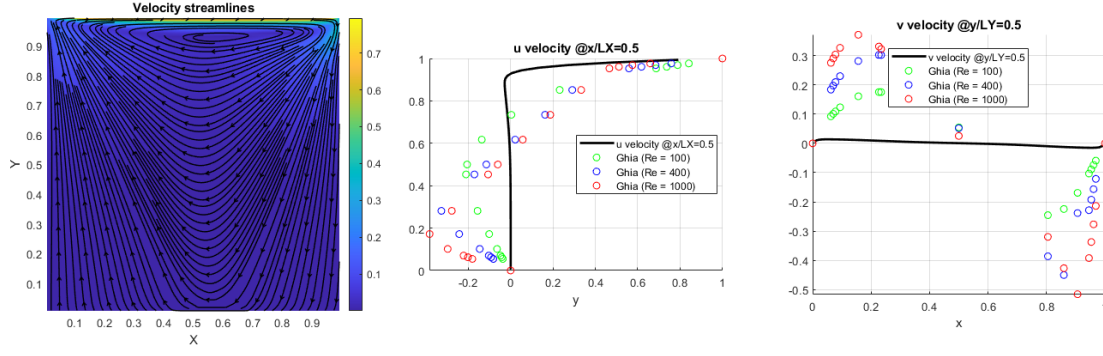
Figure 16: SIMPLE, 80×80 mesh, $Re = 1000$, $URF_{u,v} = 0.08$, $URF_p = 0.03$, QUICK and Second Order schemes for convection and diffusion. Convergence criteria: 10^{-5}. Iterations: 50, last residual: 0.001937.

Figure 17: SIMPLE, 80×80 mesh, $Re = 1000$, $URF_{u,v} = 0.08$, $URF_p = 0.03$, QUICK and Second Order schemes for convection and diffusion. Convergence criteria: 10^{-5}. Iterations: 100, last residual: 0.000574.

Figure 18: SIMPLE, 80×80 mesh, $Re = 1000$, $URF_{u,v} = 0.08$, $URF_p = 0.03$, QUICK and Second Order schemes for convection and diffusion. Convergence criteria: 10^{-5}. Iterations: 200, last residual: 0.001514.
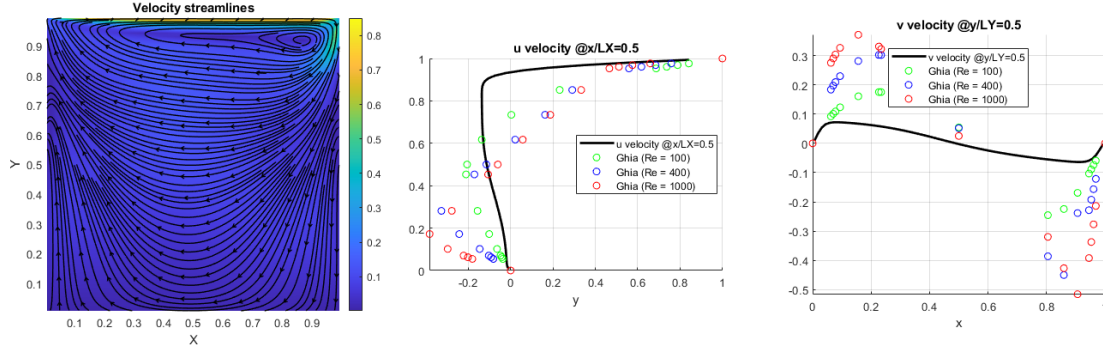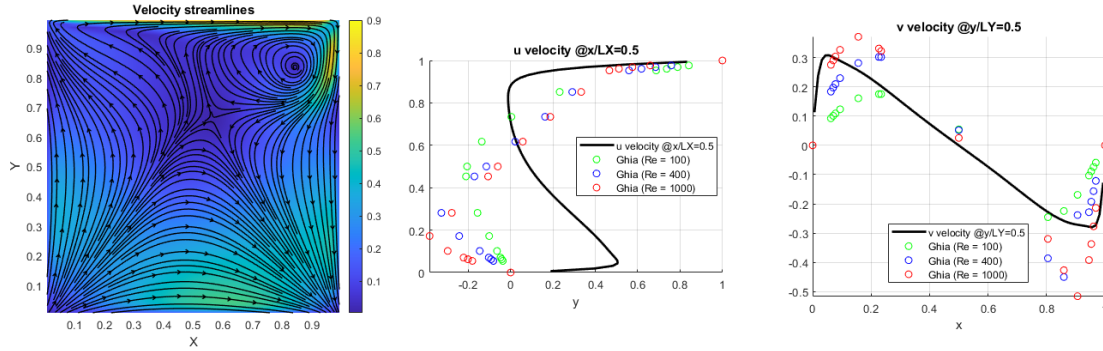
While the SCGS algorithm at least for given conditions seems to converge even if with excessive number of iterations, the SIMPLE algorithm diverges at iteration 354, when the states of the system force the overflows of the variables.

References

[1] Urmila Ghia, Karman Ghia, and C. T. Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

[2] Suhas V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15:1787–1806, 1972.

[3] S.P Vanka. Block-implicit multigrid solution of navier-stokes equations in primitive variables. *Journal of Computational Physics*, 65(1):138–158, 1986.

A Mathematica code

Here follows the `Mathematica` notebook used for symbolic analysis of the discretized schemes.

```mathematica
1  (*Tommaso Bocchietti*)
2  (*ME663 - Computational Fluid Dynamics*)
3  (*Ap coefficient calculations*)
4  Clear["Global`*"];
5
6  (*Define the Uvelocities and Vvelocities lists*)
7  Hlabels={"WW","W","P","E","EE"};
8  Vlabels={"SS","S","P","N","NN"};
9
10 (*Initialize empty matrices*)
11 \[Phi]=ConstantArray[0,{Length[Hlabels],Length[Vlabels]}];
12
13 (*Populate the matrices using for loops*)
14 For[i=1,i<=Length[Hlabels],i++,
15 For[j=1,j<=Length[Vlabels],j++,
16 \[Phi][[i,j]]=Symbol["\[Phi]"<>Hlabels[[i]]<>Vlabels[[j]]];
17 ];
18 ];
19
20 (*Extrapolate velocity at (i,j)=(i,0) and (i,j)=(0,j)*)
21 velWWtoEE=\[Phi][[All,Ceiling[Length[Vlabels]/2]]];
22 velSStoNN=\[Phi][[Ceiling[Length[Hlabels]/2]]];
23
24 (*Fix lenght as adapting*)
25 ptsWWtoEE=Table[{i*\[CapitalDelta]x,velWWtoEE[[i+3]]},{i,-2,2}];
26 ptsSStoNN = Table[{i*\[CapitalDelta]y,velSStoNN[[i+3]]},{i,-2,2}];
27
28 (*phi is an alias to access the element of \[Phi] more easly*)
29 phi[i_,j_]:= \[Phi][[i-3,j-3]];
30 (*phi[i_,j_]:= 0;*)
31
32 (*Define a list of labels for final Ap coefficients*)
33 ApLabels="Ap"<>ToString[#]&/@Flatten[\[Phi]];
34
35 (*General definition of convection and diffusion discretized equations*)
36 Convection[\[Phi]e_, \[Phi]w_, \[Phi]n_, \[Phi]s_] :=(
37    (\[Phi]e[[1]]*Max[Fe\[Phi], 0] + \[Phi]e[[2]]*Min[Fe\[Phi], 0])-
38 (\[Phi]w[[1]]*Max[Fw\[Phi], 0] + \[Phi]w[[2]]*Min[Fw\[Phi], 0]) +
39 (\[Phi]n[[1]]*Max[Fn\[Phi], 0] + \[Phi]n[[2]]*Min[Fn\[Phi], 0])-
40 (\[Phi]s[[1]]*Max[Fs\[Phi], 0] + \[Phi]s[[2]]*Min[Fs\[Phi], 0])
41 );
42
43 ConvectionCDS[\[Phi]e_, \[Phi]w_, \[Phi]n_, \[Phi]s_] :=(
44    (\[Phi]e*Fe\[Phi] )-(\[Phi]w*Fw\[Phi]) +
45 (\[Phi]n*Fn\[Phi])- (\[Phi]s*Fs\[Phi])
46 );
47
48 Diffusion[dd\[Phi]x_ , dd\[Phi]y_]:=\[Nu]*(dd\[Phi]x+dd\[Phi]y)*\[CapitalDelta]x*\[
       CapitalDelta]y;
49
50 (*Convection schemes*)
51 (*UDS Scheme for convection*)
52 UDS\[Phi]e = {phi[+0,+0], phi[+1,+0]};
53 UDS\[Phi]w = {phi[-1,+0], phi[+0,+0]};
54 UDS\[Phi]n = {phi[+0,+0], phi[+0,+1]};
55 UDS\[Phi]s = {phi[+0,-1], phi[+0,+0]};
56
```

```mathematica
57 (*USD computing and sign correction (change sign to all elements except the ApPP one)*)
58 UDSAp=Table[D[Convection[UDS\[Phi]e, UDS\[Phi]w, UDS\[Phi]n, UDS\[Phi]s],\[Phi]Vel],{\[Phi]
      Vel,Flatten[\[Phi]]}];
59 UDSAp=MapIndexed[If[First[#2]!=Ceiling[Length[UDSAp]/2],-#1,#1]&,UDSAp];

60
61 (*UDS Scheme for convection*)
62 CDS\[Phi]e = 1/2*(phi[+0,+0]+phi[+1,+0]);
63 CDS\[Phi]w = 1/2*(phi[-1,+0]+phi[+0,+0]);
64 CDS\[Phi]n = 1/2*(phi[+0,+0]+phi[+0,+1]);
65 CDS\[Phi]s = 1/2*(phi[+0,-1]+phi[+0,+0]);

66
67 (*CDS computing and sign correction (change sign to all elements except the ApPP one)*)
68 CDSAp=Table[D[ConvectionCDS[CDS\[Phi]e, CDS\[Phi]w, CDS\[Phi]n, CDS\[Phi]s],\[Phi]Vel],{\[
      Phi]Vel,Flatten[\[Phi]]}];
69 CDSAp=MapIndexed[If[First[#2]!=Ceiling[Length[CDSAp]/2],-#1,#1]&,CDSAp];

70
71 (*QUICK Scheme for convection*)
72 (*Partition[points[[1]],3,1] -> Sliding windows of 3 points amplitude*)
73 (*InterpolatingPolynomial[#,x]&/@ -> Apply interpolation over each set of points*)
74 QUICK\[Phi]e=InterpolatingPolynomial[#,x]&/@Partition[ptsWWtoEE,3,1][[2;;3]]/. x->\[
      CapitalDelta]x/2;
75 QUICK\[Phi]w=InterpolatingPolynomial[#,x]&/@Partition[ptsWWtoEE,3,1][[1;;2]]/. x->-\[
      CapitalDelta]x/2;
76 QUICK\[Phi]n=InterpolatingPolynomial[#,y]&/@Partition[ptsSStoNN,3,1][[2;;3]]/. y->\[
      CapitalDelta]y/2;
77 QUICK\[Phi]s=InterpolatingPolynomial[#,y]&/@Partition[ptsSStoNN,3,1][[1;;2]]/. y->-\[
      CapitalDelta]y/2;

78
79 (*QUICK computing and sign correction (change sign to all elements except the ApPP one)*)
80 QUICKAp=Table[D[Convection[QUICK\[Phi]e, QUICK\[Phi]w, QUICK\[Phi]n, QUICK\[Phi]s],\[Phi]Vel
      ],{\[Phi]Vel,Flatten[\[Phi]]}];
81 QUICKAp=MapIndexed[If[First[#2]!=Ceiling[Length[QUICKAp]/2],-#1,#1]&,QUICKAp];

82
83 (*Diffusion schemes*)
84 (*Second order scheme for diffusion*)
85 dd\[Phi]x2=D[InterpolatingPolynomial[ptsWWtoEE[[2;;4]],x], {x,2}]/. x->0;
86 dd\[Phi]y2=D[InterpolatingPolynomial[ptsSStoNN[[2;;4]],y], {y,2}]/. y->0;

87
88 (*DIF2 computing and sign correction (change sign to the ApPP element only)*)
89 DIF2Ap=Table[D[Diffusion[dd\[Phi]x2,dd\[Phi]y2],\[Phi]Vel],{\[Phi]Vel,Flatten[\[Phi]]}];
90 DIF2Ap=MapIndexed[If[First[#2]==Ceiling[Length[DIF2Ap]/2],-#1,#1]&,DIF2Ap];

91
92 (*Fourth order scheme for diffusion*)
93 dd\[Phi]x4=D[InterpolatingPolynomial[ptsWWtoEE[[1;;5]],x], {x,2}]/. x->0;
94 dd\[Phi]y4=D[InterpolatingPolynomial[ptsSStoNN[[1;;5]],y], {y,2}]/. y->0;

95
96 (*DIF4 computing and sign correction (change sign to the ApPP element only)*)
97 DIF4Ap=Table[D[Diffusion[dd\[Phi]x4,dd\[Phi]y4],\[Phi]Vel],{\[Phi]Vel,Flatten[\[Phi]]}];
98 DIF4Ap=MapIndexed[If[First[#2]==Ceiling[Length[DIF4Ap]/2],-#1,#1]&,DIF4Ap];

99
100 (*Final coefficients table*)
101 TableForm[Transpose[{ApLabels,UDSAp,CDSAp, QUICKAp, DIF2Ap, DIF4Ap}], TableHeadings->{None,{
      "Ap\[Phi]HV","Convection UDS","Convection CDS", "Convection QUICK", "Diffusion 2", "
      DIffusion 4"}}]
```

Listing 1: Mathematica notebook used for symbolic analysis of the discretized schemes.