2D Euler Solver

Maj. Eng. **Ney** Rafael Secco, ney@ita.br

August 27, 2025

1 Introduction

This report shows the main concepts used to implement a structured cell-centered finite volume solver for 2D Euler equations.

2 Euler Equations

The unsteady Euler equations for two dimensions are:

$$\begin{aligned}
&\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \\
&\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(\rho u^2 + p) + \frac{\partial}{\partial y}(\rho uv) = 0 \\
&\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho v^2 + p) = 0 \\
&\frac{\partial}{\partial t}(E) + \frac{\partial}{\partial x}((p+E)u) + \frac{\partial}{\partial y}((p+E)v) = 0
\end{aligned} \tag{1}$$

where ρ is density, u is the flow velocity along the x direction, v is the flow velocity along the y direction, E is energy, and p is pressure, which is correlated to the other variables through the state equation:

$$p = (\gamma - 1)\left(E - \frac{\rho(u^2 + v^2)}{2}\right) \tag{2}$$

where γ is the ratio of specific heats of the fluid. Other important relationships are given below:

$$p = \rho \cdot R \cdot T \qquad a = \sqrt{\gamma \cdot R \cdot T} \qquad M = \frac{V}{a} = \frac{\sqrt{u^2 + v^2}}{a} \tag{3}$$

with R being the fluid gas constant, T is the local temperature, a is the local speed of sound, V is the local flow velocity, and M is the local Mach number. Let us define the vector of conserved variables $\mathbf{U} = [\rho\ \rho u\ \rho v\ E]^T$ and fluxes:

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix} \quad \mathbf{F}_x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (p+E)u \end{bmatrix} \quad \mathbf{F}_y = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (p+E)v \end{bmatrix} \tag{4}$$

Then we can cast Eq. (1) as:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_x}{\partial x} + \frac{\partial \mathbf{F}_y}{\partial y} = 0 \tag{5}$$

In a more compact form:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0 \tag{6}$$

If we define a control volume A as shown in Fig. 1 (actually a control "area" in a 2D case), we can integrate the equation above to express it as a conservation law:

$$\int_A \left(\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} \right) \mathrm{d}A = 0 \tag{7}$$

$$\int_A \frac{\partial \mathbf{U}}{\partial t} \mathrm{d}A + \int_A (\nabla \cdot \mathbf{F}) \, \mathrm{d}A = 0 \tag{8}$$

$$\frac{\partial}{\partial t} \int_A \mathbf{U} \mathrm{d}A + \int_A (\nabla \cdot \mathbf{F}) \, \mathrm{d}A = 0 \tag{9}$$

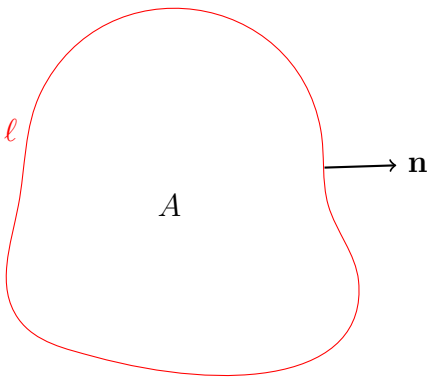

Figure 1: Two dimensional control volume.

We can apply Gauss's theorem at the second term to obtain:

$$\frac{\partial}{\partial t} \int_A \mathbf{U} \mathrm{d}A + \oint_\ell (\mathbf{F} \cdot \mathbf{n}) \, \mathrm{d}\ell = 0 \tag{10}$$

where $\mathbf{n} = [n_x \quad n_y]^T$ is a unitary vector normal to the control volume boundaries, and $\mathrm{d}\ell$ is a infinitesimal length along the boundary.

2.1 Finite Volume Approach

Assume that we discretize the domain in a mesh and then take one cell as a control volume, as shown in Fig. 2.

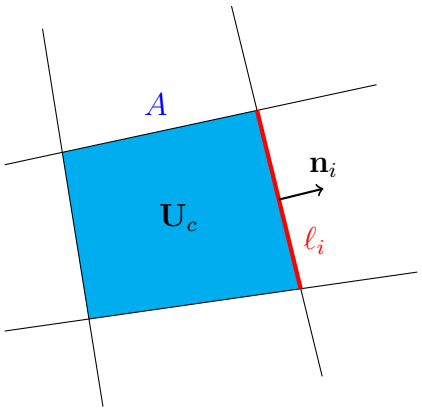

Figure 2: Cell and edge parameters.

where A is the cell area. In addition, consider that the conserved variables are represented by an average value inside this cell:

$$\mathbf{U}_c = \frac{1}{A}\int_A \mathbf{U}\mathrm{d}A \tag{11}$$

We will also consider that the fluxes have a constant value along each boundary (edge) of the cell. Thus we can approximate the second term of Eq. (10) as:

$$\oint_\ell (\mathbf{F}\cdot\mathbf{n})\,\mathrm{d}\ell = \sum_i \mathbf{F}_i\cdot\mathbf{n}_i\cdot\ell_i \tag{12}$$

with i representing the index of each edge of the cell, $\mathbf{n}_i$ a unitary vector normal to the edge, and ℓ_i the edge length. Then Eq. (10) becomes:

$$\frac{\partial \mathbf{U}_c}{\partial t} + \frac{1}{A}\sum_i \mathbf{F}_i\cdot\mathbf{n}_i\cdot\ell_i = 0 \tag{13}$$

$$\frac{\partial \mathbf{U}_c}{\partial t} = -\frac{1}{A}\sum_i \mathbf{F}_i\cdot\mathbf{n}_i\cdot\ell_i \tag{14}$$

This equation is the basis of the Cell-Centered Finite Volume Method: for every cell, we compute the sum of edge fluxes, and then use the result to update conserved variables with a time-marching method (such as Runge-Kutta).

We can also use the right-hand side of Eq. (14) to define residuals $\mathbf{r}_c$ of the cell as:

$$\mathbf{r}_c = \frac{1}{A} \sum_i \mathbf{F}_i \cdot \mathbf{n}_i \cdot \ell_i \tag{15}$$

In a steady state situation the residuals should be zero, since they are both related:

$$\frac{\partial \mathbf{U}_c}{\partial t} = -\mathbf{r}_c \tag{16}$$

2.2 Flux computation

The fluxes $\mathbf{F}_x$ and $\mathbf{F}_y$ depend on the conserved variables $\mathbf{U}$. Therefore, we need to select which values of conserved variables are attributed for each edge when evaluating Eq. (14).

Let us analyze the edge and cells represented in Fig. 3.

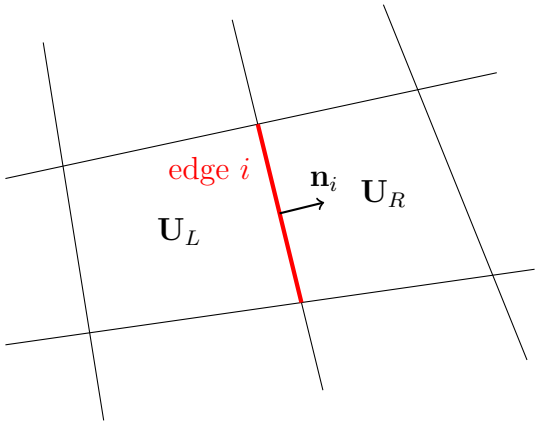

Figure 3: Definition of left and right cells sharing an edge.

At a first glance, we may be inclined to take the average of the conserved variables of the two cells sharing the edge. However, this would be analogous to a central-differences method, which is unstable for hyperbolic equations. Fortunately, there are ways to compute the flux taking into account the direction which information is propagated through the mesh, leading to stable methods. These are also called approximate Riemann solvers.

For instance, we can use Roe's flux to find the flux along the edge based on the states of the neighboring cells:

$$\mathbf{F}_i = \mathbf{F}_{\text{Roe}}(\mathbf{U}_L, \mathbf{U}_R) \tag{17}$$

The derivation of Roe's flux is out of scope of this manuscript.

2.3 Second-order approximation

We need to define the state variable values on the left and right sides of the edge when computing the fluxes of Eq. (17). If we choose the cell-centered values, we get a first order

method, which has its corresponding numerical errors. We can increase the numerical accuracy of the method if we use assume that state variables vary linearly within the cell, and then use this linear trend to compute edge values. For instance, with U being one of the state variables:

$$U(\mathbf{x}) = U_c + \nabla U \cdot (\mathbf{x} - \mathbf{x}_c) \tag{18}$$

where, U is one state variable of $\mathbf{U}$, $\mathbf{x}_c$ is the cell centroid, and $\mathbf{x}$ is another point within the cell. The gradient $\nabla U = [\partial U/\partial x \ \partial U/\partial y]$, which indicates the linear variation, can be estimated using Gauss's theorem:

$$\frac{\partial U}{\partial x} = \frac{1}{A} \sum_i U_i \cdot n_{x,i} \cdot \ell_i \tag{19}$$

$$\frac{\partial U}{\partial y} = \frac{1}{A} \sum_i U_i \cdot n_{y,i} \cdot \ell_i \tag{20}$$

where $n_{x,i}$ and $n_{y,i}$ are the components of the unitary vector normal to the edge, ℓ_i is the edge length, and U_i is the average of the state variables of the cells that share face i. The derivation of these relationships can be seen in Sec. 6.1.

Let $\mathbf{x}_i$ be the center of edge i, which is shared by cells j on the left and $j+1$ on the right, as shown in Fig. 4.

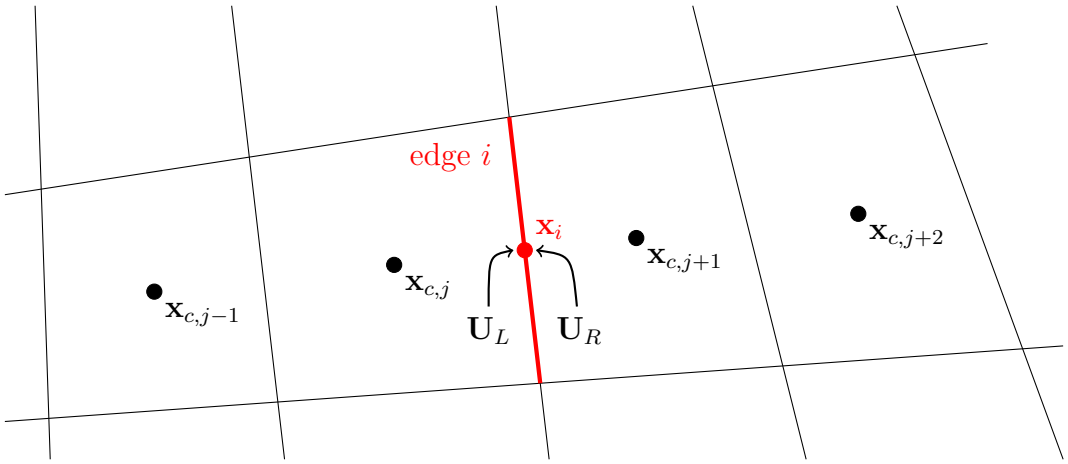

Figure 4: Opposite neighbors for limiter computation.

The state variables at each side of the edge can be computed as:

$$U_L = U_j + \nabla U_j \cdot (\mathbf{x}_i - \mathbf{x}_{c,j}) \qquad U_R = U_{j+1} + \nabla U_{j+1} \cdot (\mathbf{x}_i - \mathbf{x}_{c,j+1}) \tag{21}$$

The approximation shown above may generate oscillations near shocks. Therefore, we need to add limiters (ϕ) to avoid using downstream information in these situations:

$$U_L = U_j + \phi_j \cdot \nabla U_j \cdot (\mathbf{x}_i - \mathbf{x}_{c,j}) \qquad U_R = U_{j+1} + \phi_{j+1} \cdot \nabla U_{j+1} \cdot (\mathbf{x}_i - \mathbf{x}_{c,j+1}) \tag{22}$$

There are several limiters recommended in the literature, and they usually observe how state variables change in a sequence of cells. In a structured mesh, these relative changes (r) for cells j and $j+1$ can be defined as:

$$r_j = \frac{U_j - U_{j-1}}{U_{j+1} - U_j + \epsilon} \qquad r_{j+1} = \frac{U_{j+1} - U_{j+2}}{U_j - U_{j+1} + \epsilon} \tag{23}$$

where U_{j-1} and U_{j+2} are the neighbor cells on the opposite side of edge i (see Fig. 4). The ϵ value is a parameter used to avoid division by zero, and we can set it as $\epsilon = 1 \cdot 10^{-10}$. Then, the Van Albada limiter can be computed as:

$$\phi_j = \phi(r_j) = \frac{r_j^2 + r_j}{r_j^2 + 1} \qquad \phi_{j+1} = \phi(r_{j+1}) = \frac{r_{j+1}^2 + r_{j+1}}{r_{j+1}^2 + 1} \tag{24}$$

ATTENTION: The limiter values will be different for each cell, edge, and state variable.

The values of U_L and U_R computed with Eq. (22) can be assigned to the flux computation of Eq. (17).

2.4 Time-stepping

We can discretize time in steps of Δt, as follows:

$$t^{(n+1)} = t^{(n)} + \Delta t \tag{25}$$

where n represents the number of steps.

Let $\mathbf{U}_c^{(n)}$ represent the state variables of the entire mesh at instant $t^{(n)}$. Once we evaluate the fluxes on all edges of the mesh and have the residual values for every cell, we can use $\partial \mathbf{U}_c^{(n)} / \partial t = -\mathbf{r}_c(\mathbf{U}_c^{(n)})$ to update the conserved variables, what corresponds to a march in time. We can use, for example, the five-stage Runge-Kutta scheme (RK5):

$$\mathbf{U}_1 = \mathbf{U}_c^{(n)} - \frac{1}{4} \cdot \Delta t \cdot \mathbf{r}_c(\mathbf{U}_c^{(n)}) \tag{26}$$

$$\mathbf{U}_2 = \mathbf{U}_c^{(n)} - \frac{1}{6} \cdot \Delta t \cdot \mathbf{r}_c(\mathbf{U}_1) \tag{27}$$

$$\mathbf{U}_3 = \mathbf{U}_c^{(n)} - \frac{3}{8} \cdot \Delta t \cdot \mathbf{r}_c(\mathbf{U}_2) \tag{28}$$

$$\mathbf{U}_4 = \mathbf{U}_c^{(n)} - \frac{1}{2} \cdot \Delta t \cdot \mathbf{r}_c(\mathbf{U}_3) \tag{29}$$

$$\mathbf{U}_c^{(n+1)} = \mathbf{U}_c^{(n)} - \frac{1}{1} \cdot \Delta t \cdot \mathbf{r}_c(\mathbf{U}_4) \tag{30}$$

ATTENTION: to save memory, you can overwrite the matrix $\mathbf{U}_c^{(n)}$ on every step instead of storing $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{U}_3$, and $\mathbf{U}_4$.

2.5 Normalization

We can normalize the Euler equations with respect to the freestream density (ρ_∞), freestream velocity ($V_\infty = \sqrt{u_\infty^2 + v_\infty^2}$), and a reference length of the geometry (L_{ref}). This facilitates the definition of boundary conditions and improves the numerical consistency of the problem since all variables have reasonable orders of magnitude. The normalized variables are defined as follows:

$$\begin{aligned} \bar{x} &= \frac{x}{L_{\text{ref}}} \qquad \bar{y} = \frac{y}{L_{\text{ref}}} \qquad \bar{t} = \frac{V_\infty}{L_{\text{ref}}} t \\ \bar{\rho} &= \frac{\rho}{\rho_\infty} \qquad \bar{u} = \frac{u}{V_\infty} \qquad \bar{v} = \frac{v}{V_\infty} \\ \bar{p} &= \frac{p}{\rho_\infty V_\infty^2} \quad \bar{E} = \frac{E}{\rho_\infty V_\infty^2} \end{aligned} \tag{31}$$

If we apply these normalizations into Eqs. (1) and (2) we get:

$$\begin{aligned} &\frac{\partial}{\partial \bar{t}}(\bar{\rho}) + \frac{\partial}{\partial \bar{x}}(\bar{\rho}\bar{u}) + \frac{\partial}{\partial \bar{y}}(\bar{\rho}\bar{v}) = 0 \\ &\frac{\partial}{\partial \bar{t}}(\bar{\rho}\bar{u}) + \frac{\partial}{\partial \bar{x}}(\bar{\rho}\bar{u}^2 + \bar{p}) + \frac{\partial}{\partial \bar{y}}(\bar{\rho}\bar{u}\bar{v}) = 0 \\ &\frac{\partial}{\partial \bar{t}}(\bar{\rho}\bar{v}) + \frac{\partial}{\partial \bar{x}}(\bar{\rho}\bar{u}\bar{v}) + \frac{\partial}{\partial \bar{y}}(\bar{\rho}\bar{v}^2 + \bar{p}) = 0 \\ &\frac{\partial}{\partial \bar{t}}\left(\bar{E}\right) + \frac{\partial}{\partial \bar{x}}\left((\bar{p} + \bar{E})\bar{u}\right) + \frac{\partial}{\partial \bar{y}}\left((\bar{p} + \bar{E})\bar{v}\right) = 0 \end{aligned} \tag{32}$$

$$\bar{p} = (\gamma - 1)\left(\bar{E} - \frac{\bar{\rho}(\bar{u}^2 + \bar{v}^2)}{2}\right) \tag{33}$$

Note that these equations have the same structure as their dimensional counterparts. Therefore, we can use the same fluxes and time march derived for the dimensional Euler equations.

If the mesh dimensions are not normalized by the reference length (e.g. they are given in physical coordinates), we can set $L_{\text{ref}} = 1$. Then each time step of Sec. 2.4 taken in intervals of $\Delta\bar{t}$ will correspond to intervals of $V_\infty \cdot \Delta\bar{t}$ in dimensional time.

When dealing with external flow problems, we usually define the freestream Mach number (M_∞), angle of attack (α), and altitude (in terms of the air density ρ_∞ and pressure p_∞). The Mach number is related to the airspeed V_∞ by:

$$M_\infty = \frac{V_\infty}{a_\infty} \tag{34}$$

with a_∞ being the speed of sound, which is given by:

$$a_\infty = \sqrt{\gamma \cdot R \cdot T_\infty} = \sqrt{\frac{\gamma \cdot p_\infty}{\rho_\infty}} \tag{35}$$

Thus, the freestream velocity can be computed as:

$$V_\infty = M_\infty \sqrt{\frac{\gamma \cdot p_\infty}{\rho_\infty}} \tag{36}$$

The expression above help us define the normalized freestream parameters as:

$$\begin{aligned} \bar{\rho}_\infty &= 1 \qquad \bar{u}_\infty = \cos\alpha \qquad\qquad \bar{v}_\infty = \sin\alpha \\ \bar{p}_\infty &= \frac{1}{\gamma M_\infty^2} \qquad\qquad \bar{E}_\infty = \frac{1}{2} + \frac{1}{\gamma(\gamma-1)M_\infty^2} \end{aligned} \tag{37}$$

2.6 Algorithm

The overall algorithm to solve Euler equations is:

1. Read the mesh file;
2. Extrapolate the mesh to generate ghost nodes;
3. Compute cell metrics (areas, normal vectors, ...);
4. Initialize state variables at all cells (for instance, using the freestream conditions);
5. Perform timesteps until the solution converges:
   (a) Set state variables at ghost cells to enforce boundary conditions;
   (b) Loop over all edges to compute fluxes;
   (c) Use the accumulated fluxes (residuals) to perform the timestep and update state variables;
   (d) Check convergence (for example, with the MSE of residuals);
6. Postprocess the results.

3 Boundary Conditions

3.1 Ghost nodes

The application of boundary conditions requires an additional layer of cells at the border of the mesh (see Fig. 5). We can generate this extra layer by extrapolating the interior edges

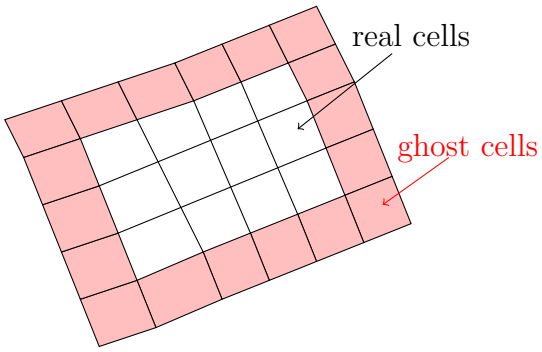

Figure 5: Definition of ghost cells.

of the original mesh. These cells are not physical, and they only exist so that we can use the same flux computation subroutine for every edge in the mesh.

At every timestep, we need to set state variables at the ghost nodes that yields the desired fluxes, as outlined in Sec. 2.6. For instance, there should be no mass flux at walls.

3.2 Farfield

For the farfield boundary conditions, we just have to assign the normalized freestream state variables to the ghost cells:

$$\mathbf{U}_{\text{ghost}} = \mathbf{U}_\infty = \begin{bmatrix} \bar{\rho}_\infty \\ \bar{\rho}_\infty \bar{u}_\infty \\ \bar{\rho}_\infty \bar{v}_\infty \\ \bar{E}_\infty \end{bmatrix} = \begin{bmatrix} 1 \\ \cos\alpha \\ \sin\alpha \\ \dfrac{1}{2} + \dfrac{1}{\gamma(\gamma-1)M_\infty^2} \end{bmatrix} \tag{38}$$

3.3 Periodic

The periodic boundary condition to simulate symmetries or when two mesh borders make contact (as in an O-mesh). Here, we take the state variables of real cells in one border and assign them to the ghost cells of the opposite border.

As an example, let us apply a periodic boundary condition between the i-borders of a structured mesh. The indices $i = 1$ and $i = 2$ refer, respectively, to the layers of ghost and physical cells at the left side of the parametric domain, and the indices $i = n_i - 1$ and $i = n_i - 2$ refer, respectively, to the layers of ghost and physical cells at the right side of the parametric domain.

The ghost cell values for the periodic boundary condition are:

$$\mathbf{U}_{1,j} = \mathbf{U}_{n_i-2,j} \quad \text{for} \quad j = 1, n_j - 1 \tag{39}$$

$$\mathbf{U}_{n_i-1,j} = \mathbf{U}_{2,j} \quad \text{for} \quad j = 1, n_j - 1 \tag{40}$$

Similarly, if the boundary condition is applied along the j-borders:

$$\mathbf{U}_{i,1} = \mathbf{U}_{i,n_j-2} \quad \text{for} \quad i = 1, n_i - 1 \tag{41}$$

$$\mathbf{U}_{i,n_j-1} = \mathbf{U}_{i,2} \quad \text{for} \quad i = 1, n_i - 1 \tag{42}$$

3.4 Inviscid walls

The inviscid wall boundary condition should prevent any mass and energy flux through the wall. In addition, the flow velocity should be tangent to the wall. We can achieve these conditions by mirroring the state variables of the physical cell using the wall as a symmetry plane. Then we assign this mirrored state to the ghost cells.

Let $\mathbf{V}_{\text{real}} = [u_{\text{real}}\ v_{\text{real}}]$ be the velocity vector at one of the physical cells at the border, $\mathbf{V}_{\text{ghost}} = [u_{\text{ghost}}\ v_{\text{ghost}}]$ the velocity vector at one of the ghost cells at the border, and $\mathbf{n} = [n_x\ n_y]$ a unitary vector normal to the edge between the real and ghost cells, pointing towards the ghost cell (see Fig. 6). If we mirror the real cell velocity using the edge as the symmetry plane, we get:

$$\mathbf{V}_{\text{ghost}} = \mathbf{V}_{\text{real}} - 2(\mathbf{V}_{\text{real}} \cdot \mathbf{n})\mathbf{n} \tag{43}$$

where the dot product $(\mathbf{V}_{\text{real}} \cdot \mathbf{n})$ represents the component of $\mathbf{V}_{\text{real}}$ that is perpendicular to the edge.

The other state variables can remain constant at the ghost cells:

$$\bar{\rho}_{\text{ghost}} = \bar{\rho}_{\text{real}} \qquad \bar{E}_{\text{ghost}} = \bar{E}_{\text{real}} \tag{44}$$

Therefore, the state variables at the ghost cell at a wall can be summarized as:

$$\mathbf{U}_{\text{ghost}} = \begin{bmatrix} \bar{\rho}_{\text{real}} \\ \bar{\rho}_{\text{real}}(\bar{u}_{\text{real}} - 2(\bar{u}_{\text{real}} n_x + \bar{v}_{\text{real}} n_y) n_x \\ \bar{\rho}_{\text{real}}(\bar{v}_{\text{real}} - 2(\bar{u}_{\text{real}} n_x + \bar{v}_{\text{real}} n_y) n_y \\ \bar{E}_{\text{real}} \end{bmatrix} \tag{45}$$

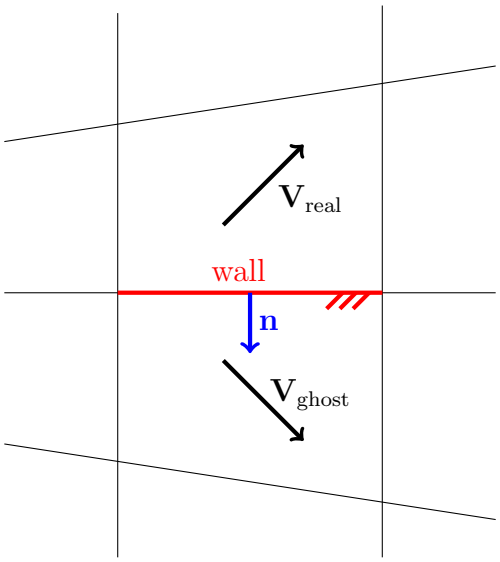

Figure 6: Velocities at wall boundary conditions.

4 Mesh

We need to discretize the domain to solve Euler equations numerically. A mesh consists of a set of points connected by edges. In a structured two dimensional mesh, every point is connected to four neighboring points in a regular manner. The mesh has a representation in physical space, with x and y coordinates, and in a parametric space, with i and j coordinates, as shown in Fig.7.

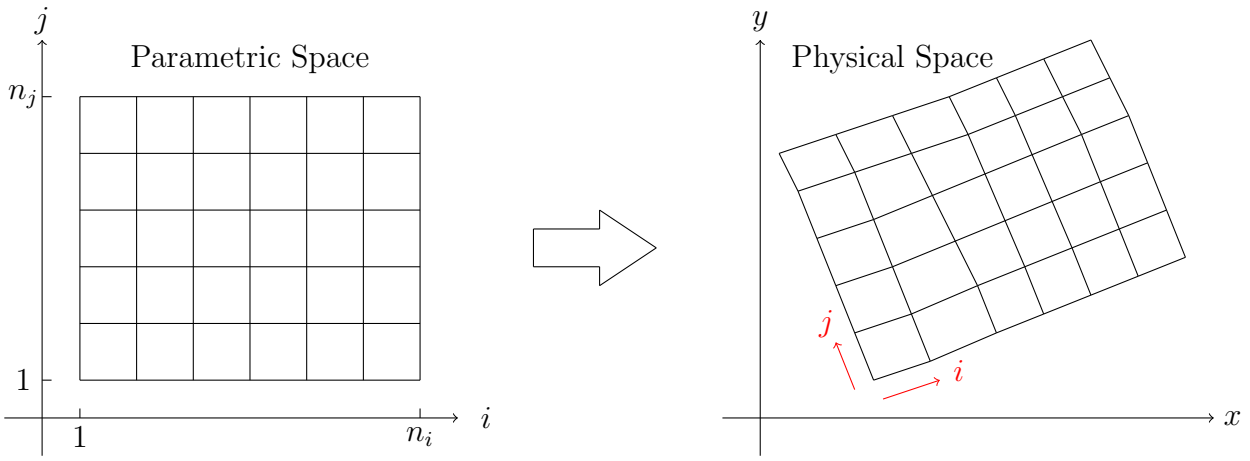

Figure 7: Structured mesh representation in parametric and physical spaces.

The points of this 2D mesh can be stored as two matrices, $\mathbf{X}$ and $\mathbf{Y}$, both of dimensions $n_i \times n_j$, with n_i representing the number of points along i coordinate and n_j representing the number of points along coordinate j. Therefore, the number of cells in the mesh is

$(n_i - 1) \cdot (n_j - 1)$.

ATTENTION: It is important to define the mesh following the right-hand rule, that is, the dot product between the i and j vectors of each cell should give a positive result along the z axis. Otherwise, nasty bugs will appear!

4.1 Ghost nodes

The application of boundary conditions requires an additional layer of cells at each border of the mesh. We can generate this extra layer by extrapolating the interior edges of the original mesh.

ATTENTION: The matrices $\mathbf{X}$ and $\mathbf{Y}$ will store the ghost nodes at their borders. Therefore, the physical nodes range from indices 2 to $n_i - 1$, and from 2 to $n_j - 1$ in these matrices.

We can compute the ghost nodes with:

$$\begin{array}{ll} x_{1,j} = 2x_{2,j} - x_{3,j} & \text{for} \quad j = 2, n_j - 1 \\ x_{n_i,j} = 2x_{n_i-1,j} - x_{n_i-2,j} & \text{for} \quad j = 2, n_j - 1 \\ x_{i,1} = 2x_{i,2} - x_{i,3} & \text{for} \quad i = 2, n_i - 1 \\ x_{i,n_j} = 2x_{i,n_j-1} - x_{i,n_j-2} & \text{for} \quad i = 2, n_i - 1 \end{array} \tag{46}$$

$$\begin{array}{ll} y_{1,j} = 2y_{2,j} - y_{3,j} & \text{for} \quad j = 2, n_j - 1 \\ y_{n_i,j} = 2y_{n_i-1,j} - y_{n_i-2,j} & \text{for} \quad j = 2, n_j - 1 \\ y_{i,1} = 2y_{i,2} - y_{i,3} & \text{for} \quad i = 2, n_i - 1 \\ y_{i,n_j} = 2y_{i,n_j-1} - y_{i,n_j-2} & \text{for} \quad i = 2, n_i - 1 \end{array} \tag{47}$$

The corner nodes can be obtained with quadratic interpolations:

$$\begin{array}{l} x_{1,1} = x_{1,2} + x_{2,1} - x_{2,3} - x_{3,2} + x_{3,3} \\ x_{n_i,1} = x_{n_i-1,1} + x_{n_i,2} - x_{n_i-2,2} - x_{n_i-1,3} + x_{n_i-2,3} \\ x_{n_i,n_j} = x_{n_i-1,n_j} + x_{n_i,n_j-1} - x_{n_i-2,n_j-1} - x_{n_i-1,n_j-2} + x_{n_i-2,n_j-2} \\ x_{1,n_j} = x_{2,n_j} + x_{1,n_j-1} - x_{3,n_j-1} - x_{2,n_j-2} + x_{3,n_j-2} \end{array} \tag{48}$$

$$\begin{array}{l} y_{1,1} = y_{1,2} + y_{2,1} - y_{2,3} - y_{3,2} + y_{3,3} \\ y_{n_i,1} = y_{n_i-1,1} + y_{n_i,2} - y_{n_i-2,2} - y_{n_i-1,3} + y_{n_i-2,3} \\ y_{n_i,n_j} = y_{n_i-1,n_j} + y_{n_i,n_j-1} - y_{n_i-2,n_j-1} - y_{n_i-1,n_j-2} + y_{n_i-2,n_j-2} \\ y_{1,n_j} = x_{2,n_j} + y_{1,n_j-1} - y_{3,n_j-1} - y_{2,n_j-2} + y_{3,n_j-2} \end{array} \tag{49}$$

See Sec. 6.2 for the derivation of these expressions.

4.2 Cell centers

The coordinates of each cell center (x_c, y_c) can be computed by averaging the four corner nodes:

$$x_{c,i,j} = \frac{x_{i,j} + x_{i+1,j} + x_{i,j+1} + x_{i+1,j+1}}{4} \quad \text{for} \quad i = 1, n_1 - 1 \quad \text{and} \quad j = 1, n_j - 1 \tag{50}$$

$$y_{c,i,j} = \frac{y_{i,j} + y_{i+1,j} + y_{i,j+1} + y_{i+1,j+1}}{4} \quad \text{for} \quad i = 1, n_1 - 1 \quad \text{and} \quad j = 1, n_j - 1 \tag{51}$$

4.3 Cell areas

The area of each cell can be computed based on the cross product of cell diagonals:

$$A_{i,j} = \frac{(x_{i+1,j+1} - x_{i,j}) \cdot (y_{i,j+1} - y_{i+1,j}) - (x_{i,j+1} - x_{i+1,j}) \cdot (y_{i+1,j+1} - y_{i,j})}{2} \tag{52}$$

$$\text{for } i = 1, n_1 - 1 \text{ and } j = 1, n_j - 1$$

4.4 Normal and tangent vectors

We need to compute normal vectors of every edge (face) of the mesh, as shown in Fig. 8.

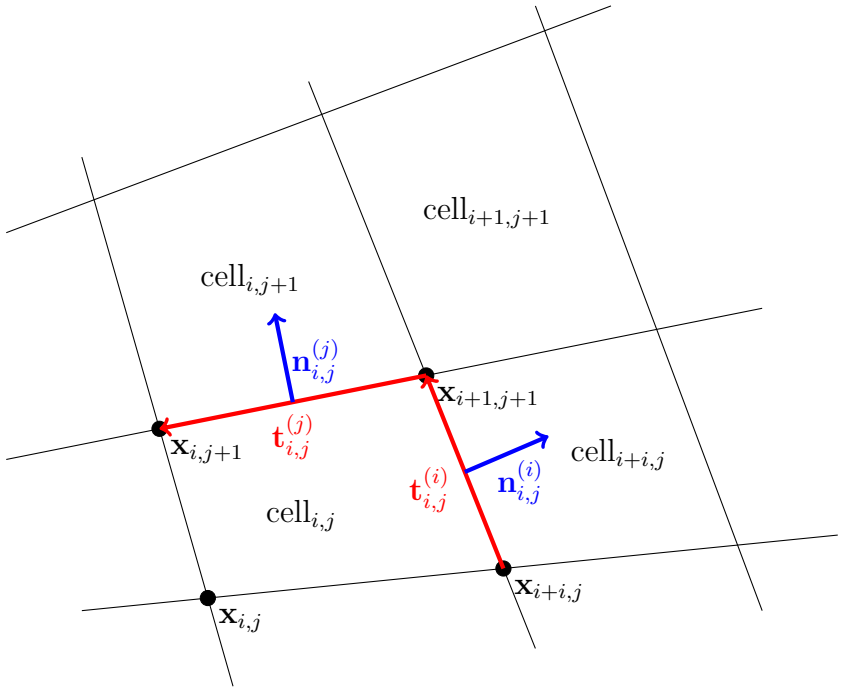

Figure 8: Face normal definition.

The vectors associated with the edge between cells (i,j) and $(i+1,j)$ are:

$$\mathbf{t}_{i,j}^{(i)} = \mathbf{x}_{i+1,j+1} - \mathbf{x}_{i+1,j} \tag{53}$$

$$\mathbf{n}_{i,j}^{(i)} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \frac{\mathbf{t}_{i,j}^{(i)}}{\left|\mathbf{t}_{i,j}^{(i)}\right|} \tag{54}$$

Note that we use a rotation matrix to obtain the normal vector based on the tangent vector. The magnitude of both vectors is equal to the edge length.

The vectors associated with the edge between cells (i,j) and $(i,j+1)$ are:

$$\mathbf{t}_{i,j}^{(j)} = \mathbf{x}_{i,j+1} - \mathbf{x}_{i+1,j+1} \tag{55}$$

$$\mathbf{n}_{i,j}^{(j)} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \frac{\mathbf{t}_{i,j}^{(j)}}{\left|\mathbf{t}_{i,j}^{(j)}\right|} \tag{56}$$

We also store the edge lengths as:

$$\ell_{i,j}^{(i)} = \left|\mathbf{t}_{i,j}^{(i)}\right| \tag{57}$$

$$\ell_{i,j}^{(j)} = \left|\mathbf{t}_{i,j}^{(j)}\right| \tag{58}$$

5 Postprocessing

Once the solution is converged, we can compute flow properties and integrate forces.

5.1 Secondary variables

The secondary state variables at each cell can be obtained from the primary state variables ($\bar{\rho}$, $\bar{\rho}\bar{u}$, $\bar{\rho}\bar{v}$, and $\bar{E}$) with:

$$\bar{u} = \frac{\bar{\rho}\bar{u}}{\bar{\rho}} \tag{59}$$

$$\bar{v} = \frac{\bar{\rho}\bar{v}}{\bar{\rho}} \tag{60}$$

$$\bar{p} = (\gamma - 1)\left(\bar{E} - \frac{\bar{\rho}(\bar{u}^2 + \bar{v}^2)}{2}\right) \tag{61}$$

$$\bar{a} = \sqrt{\frac{\gamma \cdot \bar{p}}{\bar{\rho}}} \tag{62}$$

$$M = \frac{\sqrt{\bar{u}^2 + \bar{v}^2}}{\bar{a}} \tag{63}$$

5.2 Force integration

We usually want to compute integrated forces along the wall boundaries to find, lift, drag, and moments around the bodies of interest.

We assume that the state variables at the wall are the average between the real and ghost cells that share the wall:

$$\begin{bmatrix} \bar{\rho}_{\text{wall}} \\ \bar{\rho}_{\text{wall}}\bar{u}_{\text{wall}} \\ \bar{\rho}_{\text{wall}}\bar{v}_{\text{wall}} \\ \bar{E}_{\text{wall}} \end{bmatrix} = \mathbf{U}_{\text{wall}} = \frac{\mathbf{U}_{\text{real}} + \mathbf{U}_{\text{ghost}}}{2} \tag{64}$$

This will give a flow velocity tangent to the wall due to how we assign the ghost cell states. Once we have the wall state, we find the corresponding pressure, sound speed, and Mach number with:

$$\bar{p}_{\text{wall}} = (\gamma - 1)\left(\bar{E}_{\text{wall}} - \frac{\bar{\rho}_{\text{wall}}(\bar{u}^2_{\text{wall}} + \bar{v}^2_{\text{wall}})}{2}\right) \tag{65}$$

$$\bar{a}_{\text{wall}} = \sqrt{\frac{\gamma \cdot \bar{p}_{\text{wall}}}{\bar{\rho}_{\text{wall}}}} \tag{66}$$

$$M_{\text{wall}} = \frac{\sqrt{\bar{u}^2_{\text{wall}} + \bar{v}^2_{\text{wall}}}}{\bar{a}_{\text{wall}}} \tag{67}$$

Let i represent the index of a given cell near a wall boundary. The pressure coefficient at the wall edge of this cell is:

$$C_{p,i} = \frac{p_{\text{wall,i}} - p_\infty}{\frac{\rho V_\infty^2}{2}} = 2\left(\frac{p_{\text{wall,i}}}{\rho V_\infty^2} - \frac{p_\infty}{\rho V_\infty^2}\right) \tag{68}$$

If we use the pressure normalization explained in Sec. 2.5 we get:

$$C_{p,i} = 2\left(\bar{p}_{\text{wall,i}} - \bar{p}_\infty\right) = 2\left(\bar{p}_{\text{wall,i}} - \frac{1}{\gamma M_\infty^2}\right) \tag{69}$$

The force components at the wall are:

$$F_{x,i} = p_{\text{wall,i}} \cdot \ell_{\text{wall,i}} \cdot n_{x,i} \tag{70}$$

$$F_{y,i} = p_{\text{wall,i}} \cdot \ell_{\text{wall,i}} \cdot n_{y,i} \tag{71}$$

with $\ell_{\text{wall,i}}$ being the edge length and $\mathbf{n}_i = [n_{x,i}\ n_{y,i}]$ a unitary vector normal to the wall, pointing towards the ghost cell (out of the domain).

The moment generated by these forces with respect to a reference point $(x_{\text{ref}}, y_{\text{ref}})$ is:

$$M_i = (x_{\text{wall}} - x_{\text{ref}}) \cdot F_{y,i} - (y_{\text{wall}} - y_{\text{ref}}) \cdot F_{x,i} \tag{72}$$

with x_{wall} and y_{wall} being the center of the wall edge.

These are force and moment contributions of a single wall edge. To get the total force, we need to add the contributions of the entire wall boundary:

$$F_x = \sum_i F_{x,i} \tag{73}$$

$$F_y = \sum_i F_{y,i} \tag{74}$$

$$M = \sum_i M_i \tag{75}$$

We can compute the corresponding non-dimensional coefficients with:

$$C_x = \frac{F_x}{\frac{\rho_\infty V_\infty^2}{2} \cdot L_{\text{ref}}} \tag{76}$$

$$C_y = \frac{F_y}{\frac{\rho_\infty V_\infty^2}{2} \cdot L_{\text{ref}}} \tag{77}$$

$$C_M = \frac{M}{\frac{\rho_\infty V_\infty^2}{2} \cdot L_{\text{ref}}^2} \tag{78}$$

We can use the normalizations from Sec. 2.5 we get:

$$C_x = \frac{2}{L_{\text{ref}}} \sum_i (\bar{p}_{\text{wall,i}} \cdot \ell_{\text{wall,i}} \cdot n_{x,i}) \tag{79}$$

$$C_y = \frac{2}{L_{\text{ref}}} \sum_i (\bar{p}_{\text{wall,i}} \cdot \ell_{\text{wall,i}} \cdot n_{y,i}) \tag{80}$$

$$C_m = \frac{2}{L_{\text{ref}}^2} \sum_i \bar{p}_{\text{wall,i}} \cdot \ell_{\text{wall,i}} \cdot ((x_{\text{wall}} - x_{\text{ref}}) \cdot n_{y,i} - (y_{\text{wall}} - y_{\text{ref}}) \cdot n_{x,i}) \tag{81}$$

We can obtain the usual aerodynamic coefficients with:

$$C_L = C_y \cdot \cos\alpha - C_x \cdot \sin\alpha \tag{82}$$

$$C_D = C_y \cdot \sin\alpha + C_x \cdot \cos\alpha \tag{83}$$

$$C_M = -C_m \tag{84}$$

The minus sign is to apply the convention where positive pitching moment increases the angle of attack.

6 Theoretical background

6.1 Gradients of scalar fields

Suppose we want to estimate the gradient of a scalar field $U = U(x, y)$. In other words, we want to compute:

$$\nabla U = \begin{bmatrix} \dfrac{\partial U}{\partial x} & \dfrac{\partial U}{\partial y} \end{bmatrix} \tag{85}$$

Let us define a vector field $\mathbf{V} = [V_x \quad V_y]$ defined over the same space, with $V_x = V_x(x, y)$ and $V_y = V_y(x, y)$. According to Gauss's theorem, the following relationship is valid for this vector field within a control volume A, defined by the boundary ℓ:

$$\int_A (\nabla \cdot \mathbf{V}) \, \mathrm{d}A = \oint_\ell (\mathbf{V} \cdot \mathbf{n}) \, \mathrm{d}\ell \tag{86}$$

$$\int_A \left(\frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} \right) \mathrm{d}A = \oint_\ell (V_x \cdot n_x + V_y \cdot n_y) \cdot \mathrm{d}\ell \tag{87}$$

with n_x and n_y being components of unitary vectors normal to the boundary, and pointing to the outside of the control volume A. We can assign the desired scalar field U to one of the components of the vector field $\mathbf{V}$. For example:

$$\mathbf{V} = [V_x \quad V_y] = [U \quad 0] \tag{88}$$

For this particular case, Eq. (87) becomes:

$$\int_A \frac{\partial U}{\partial x} \mathrm{d}A = \oint_\ell U \cdot n_x \cdot \mathrm{d}\ell \tag{89}$$

Now consider that the control volume A is a cell of the mesh. We can define that the average value of $\partial U/\partial x$ within this cell is:

$$\left(\frac{\partial U}{\partial x} \right)_c = \frac{1}{A} \int_A \frac{\partial U}{\partial x} \mathrm{d}A \tag{90}$$

In addition, if we consider that the scalar field U is constant along each edge of the cell, and that these edges are straight, we have:

$$\oint_{\ell} U \cdot n_x \cdot \mathrm{d}\ell = \sum_i U_i \cdot n_{x,i} \cdot \ell_i \tag{91}$$

where i represents the indices of each edge and ℓ_i is the edge length. Since each edge is shared between two cells, we can assign U_i as the average of the scalar fields at these cells:

$$U_i = \frac{U_{L,i} + U_{R,i}}{2} \tag{92}$$

Generally, $U_{L,i} = U_c$, which is the average scalar field value at the cell we are computing the derivative. Then Eq. (89) becomes:

$$\left(\frac{\partial U}{\partial x}\right)_c = \frac{1}{A} \sum_i U_i \cdot n_{x,i} \cdot \ell_i \tag{93}$$

We can do a similar derivation for the y derivative to get:

$$\left(\frac{\partial U}{\partial y}\right)_c = \frac{1}{A} \sum_i U_i \cdot n_{y,i} \cdot \ell_i \tag{94}$$

6.2 Quadratic interpolation of corner nodes

Assume that the x coordinate of the nodes follows a quadratic function of the parametric coordinates i and j:

$$x_{i,j} = a_0 + a_1 \cdot i + a_2 \cdot j + a_3 \cdot i^2 + a_4 \cdot i \cdot j + a_5 \cdot j^2 \tag{95}$$

The a coefficients can be determined by sampling this function at 6 nodes. We can choose the nearest nodes to the ($i = 1$, $j = 1$) corner:

$$\begin{aligned}
x_{1,2} &= a_0 + a_1 \cdot 1 + a_2 \cdot 2 + a_3 \cdot 1 + a_4 \cdot 2 + a_5 \cdot 4 \\
x_{2,1} &= a_0 + a_1 \cdot 2 + a_2 \cdot 1 + a_3 \cdot 4 + a_4 \cdot 2 + a_5 \cdot 1 \\
x_{2,2} &= a_0 + a_1 \cdot 2 + a_2 \cdot 2 + a_3 \cdot 4 + a_4 \cdot 4 + a_5 \cdot 4 \\
x_{2,3} &= a_0 + a_1 \cdot 2 + a_2 \cdot 3 + a_3 \cdot 4 + a_4 \cdot 6 + a_5 \cdot 9 \\
x_{3,2} &= a_0 + a_1 \cdot 3 + a_2 \cdot 2 + a_3 \cdot 9 + a_4 \cdot 6 + a_5 \cdot 4 \\
x_{3,3} &= a_0 + a_1 \cdot 3 + a_2 \cdot 3 + a_3 \cdot 9 + a_4 \cdot 9 + a_5 \cdot 9
\end{aligned} \tag{96}$$

In matrix form:

$$\begin{bmatrix} x_{1,2} \\ x_{2,1} \\ x_{2,2} \\ x_{2,3} \\ x_{3,2} \\ x_{3,3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 2 & 1 & 2 & 4 \\ 1 & 2 & 1 & 4 & 2 & 1 \\ 1 & 2 & 2 & 4 & 4 & 4 \\ 1 & 2 & 3 & 4 & 6 & 9 \\ 1 & 3 & 2 & 9 & 6 & 4 \\ 1 & 3 & 3 & 9 & 9 & 9 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \tag{97}$$

Solving for the coefficients (inverting the matrix):

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 3 & 3 & -3 & -3 & -3 & 4 \\ -2.5 & 0 & 2 & 2 & 0.5 & -2 \\ 0 & -2.5 & 2 & 0.5 & 2 & -2 \\ 0.5 & 0 & -1 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0.5 & -1 & 0.5 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_{1,2} \\ x_{2,1} \\ x_{2,2} \\ x_{2,3} \\ x_{3,2} \\ x_{3,3} \end{bmatrix} \tag{98}$$

If we sample Eq. (95) at the ($i = 1$, $j = 1$) corner:

$$x_{1,1} = a_0 + a_1 + a_2 + a_3 + a_4 + a_5 \tag{99}$$

Substituting the values of the coefficients yields:

$$x_{1,1} = x_{1,2} + x_{2,1} - x_{2,3} - x_{3,2} + x_{3,3} \tag{100}$$

The same interpolation also works for the y coordinate, thus:

$$y_{1,1} = y_{1,2} + y_{2,1} - y_{2,3} - y_{3,2} + y_{3,3} \tag{101}$$

Similar approaches can be done to derive equations for the remaining corners, as presented in Eq. (48) and Eq. (49).