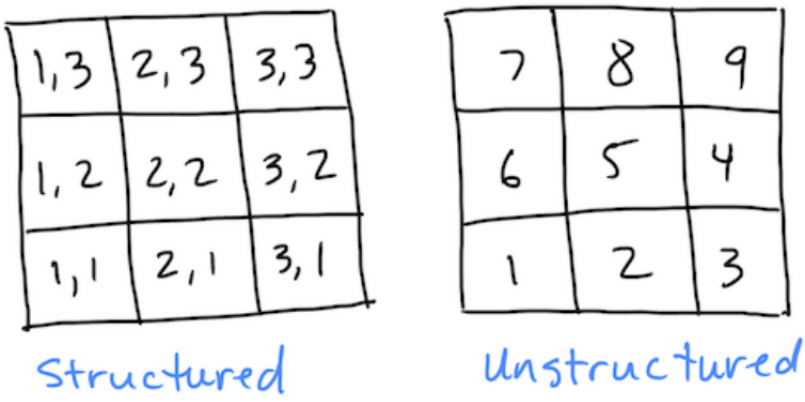
Contents

1 Two and Three Dimensional Grids

1.1 General introduction

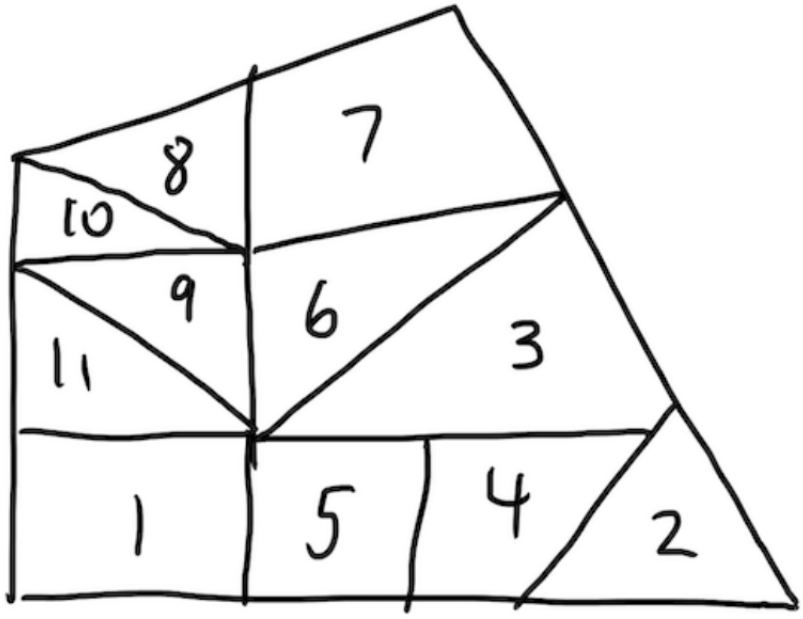
If the grids have an ordered, orthogonal structure, then it is easy to extend what we have done so far to higher dimensional grids.
We will discuss **structured** vs **unstructured** grid. The main idea is that the labels are associated with how the grid cells are labeled, and not their topologies.



We can see from the image above that:

- **structured grid** uses **ordered** row and column labelling
- **unstructutured grid** uses **arbitrary** cell labelling

As a result, for grid with arbitrary polyhedra with no inherent structure, it is better to store the grid with an unstructured labelling scheme, as shown below.



1.2 Structured Grid

1.2.1 General Discretization

The goal is to discretize a generic transport equation over a 2D structured grid. Then we will extend from 2D to 3D. Recall our generic transport equation:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) + \nabla \cdot \mathbf{J}_\phi = S_\phi$$

Integrate through space and time, we arrive at:

$$\frac{\phi^{t+\Delta t/2} - \phi^{t-\Delta t/2}}{\Delta t} + \sum_{i=0}^{N_{ip}-1} \mathbf{u}_{ip} \cdot \mathbf{n}_{ip} \phi_{ip} A_{ip} = \sum_{i=0}^{N_{ip}-1} \mathbf{J}_{\phi,ip} \cdot \mathbf{n}_{ip} A_{ip} + S_\phi V_P$$

1. Mass Equation

   Say we want to make this general form to represent the conservation of mass equation, we set $\phi = \rho$, $\mathbf{J}_\phi = 0$ and $S_\phi = 0$. Then, using the

integration points as w (west), e (east), s (south) and n (north), we get:

$$\frac{\rho^{t+\Delta t/2} - \rho^{t-\Delta t/2}}{\Delta t} + \dot{m}_e - \dot{m}_w + \dot{m}_n - \dot{m}_s = 0$$

For the case of constant density:

$$\dot{m}_e - \dot{m}_w + \dot{m}_n - \dot{m}_s = 0$$

2. Momentum Equation

Likewise, to make the general form representing the conservation of momentum in the x-direction, we simply set $\phi = \rho u$, $\mathbf{J}_\phi = \mu \nabla u$ and $S_\phi = -\partial p / \partial x$.

$$\frac{\rho^{t+\Delta t/2} - \rho^{t-\Delta t/2}}{\Delta t} + \dot{m}_e u_e - \dot{m}_w u_w + \dot{m}_n u_n - \dot{m}_s u_s = \mu \left.\frac{\partial u}{\partial x}\right|_e - \mu \left.\frac{\partial u}{\partial x}\right|_w + \mu \left.\frac{\partial u}{\partial y}\right|_n - \mu \left.\frac{\partial u}{\partial y}\right|_s - \frac{\partial p}{\partial y} V_P$$

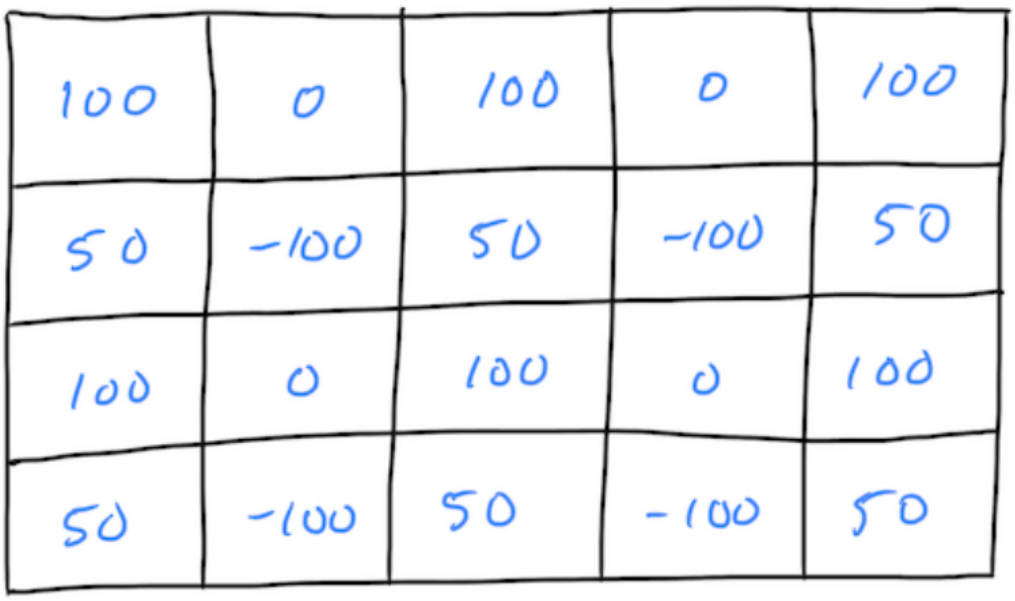
The discretization procedure for the momentum equation is as follow:

- appropriate momentum equation - (mass equation * appropriate velocity component at cell P)
- choose time integration scheme for the transient term
- choose advection scheme for the advection term
- approximate the derivatives using finite differences.

The momentum equations in x and y can be written in terms of their linearization coefficients:

$$a_p u_p = -a_W u_W - a_E u_E - a_S u_S - a_N u_N + b_u - \frac{p_E - p_W}{2\Delta x} V_P$$
$$a_p v_p = -a_W v_W - a_E v_E - a_S v_S - a_N v_N + b_v - \frac{p_N - p_S}{2\Delta y} V_P$$

Recall back to our 1D case, the pressure can field can oscillate if we are not careful. In 2D, this is even worse because the oscillations can happen in both directions. An example of an oscillated pressure field but would be accepted by the solver as a smooth pressure field is shown below:



Same as the 1D case, this oscillations can be solved by

- using staggered grid
- using collocated grid with different advected and advecting velocities.

because the staggered grid becomes more complicated at higher dimensions, we only consider collocated grid. The resulting expressions for advecting velocities in each direction are as follow:

$$\hat{u}_e = \frac{1}{2}(u_P + u_E) - \hat{d}_e^u \left[\left.\frac{dp}{dx}\right|_e - \frac{1}{2}\left(\left.\frac{dp}{dx}\right|_P + \left.\frac{dp}{dx}\right|_E \right) \right]$$
$$\hat{v}_n = \frac{1}{2}(v_P + v_N) - \hat{d}_n^v \left[\left.\frac{dp}{dy}\right|_n - \frac{1}{2}\left(\left.\frac{dp}{dy}\right|_P + \left.\frac{dp}{dy}\right|_N \right) \right]$$

with the superscript on $\hat{d}$ represents the equation it is associated with. Similar to 1D, the coupling can be done direct or seggregated method, e.g. SIMPLE or SIMPLEC

1.2.2 False Diffusion

Recall in 1D, Taylor series analysis shows some serious problem, but they are not as bad. Using UDS for linearization and correcting the advective

fluxes with a higher order method, we can get good results in these cases.

In 2D and 3D, false diffusion comes from a different source than 1D. Usually, this associates with cases where flow streamlines are not well aligned with the grid lines.

Consider a steady advection of scalar quantity with **no sources**, real diffusion is **negligible** compare to advection.

$$\dot{m}_e\phi_e - \dot{m}_w\phi_w + \dot{m}_n\phi_n - \dot{m}_s\phi_s = 0$$

Assume positive flow in x and y and using UDS for advection:

$$\dot{m}_e\phi_P - \dot{m}_w\phi_W + \dot{m}_n\phi_P - \dot{m}_s\phi_S = 0$$

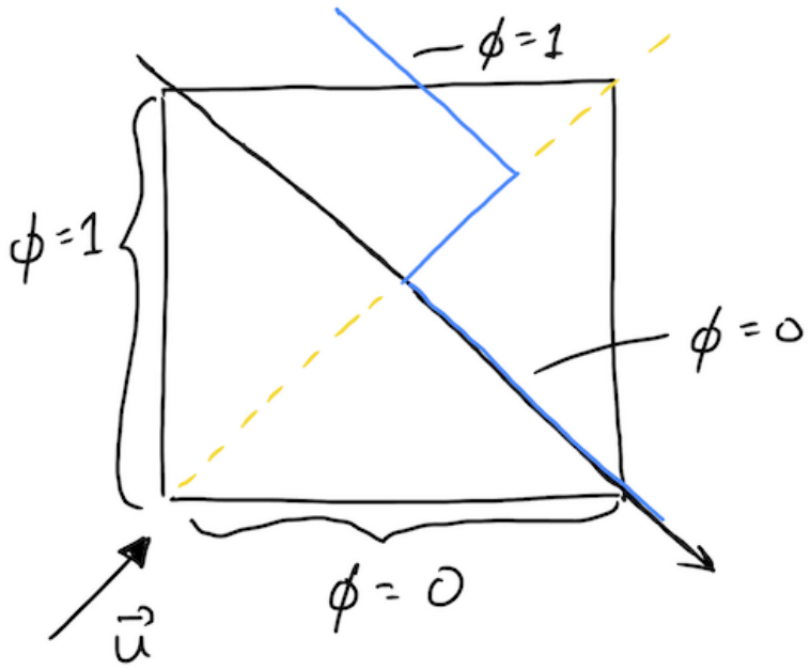
Solving for ϕ_P, we get:

$$\phi_P = \frac{\dot{m}_w}{\dot{m}_e + \dot{m}_n}\phi_W + \frac{\dot{m}_s}{\dot{m}_e + \dot{m}_n}\phi_S$$

Consider flow at 45 degrees to the x axis, meaning $\dot{m}_e = \dot{m}_w = \dot{m}_n = \dot{m}_s$, we have:

$$\phi_P = \frac{1}{2}\phi_W + \frac{1}{2}\phi_S$$

If $\phi = 0$ is advected from the bottom, and $\phi = 1$ is advected from the left. Then the exact solution is a step profile at any cross-secion perpendicular to the flow direction:



Below is the code to solve this problem: language=Python,label= ,caption= ,captionpos=b,numbers=none,style=mystyle import matplotlib import matplotlib.pyplot as plt import numpy as np matplotlib.use('Agg') fname = '../pic/45degreeStep.png'

create array to hold the solution phi = np.zeros((7,7))

set left advected value phi[:,0] = 1

print('Initial matrix:') print(np.vectorize(" compute the solution starting from the bottom left for j in reversed(range(phi.shape[0]-1)): for i in range(1,phi.shape[1]): phi[j,i] = 0.5*phi[j,i-1] + 0.5*phi[j+1,i]

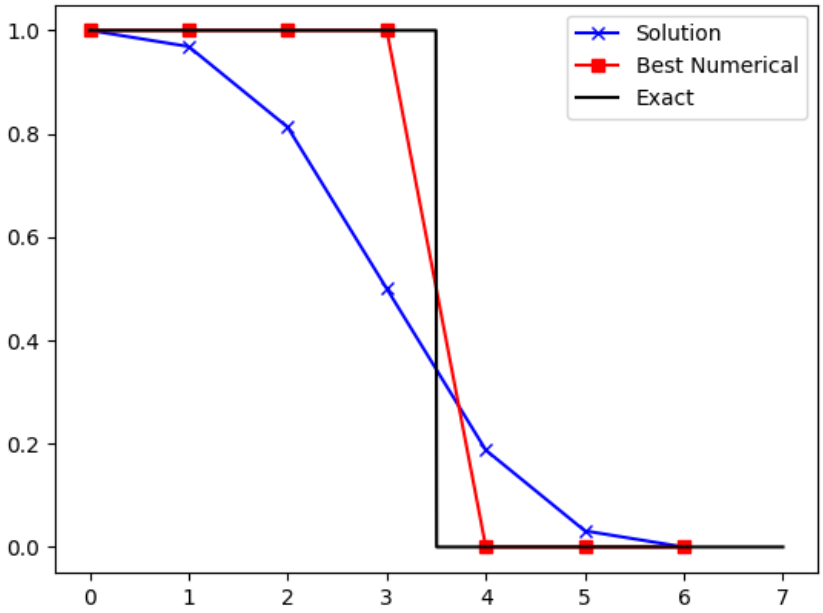
print the solution matrix print('matrix:') print(np.vectorize("

plot solution along the diagonal cross-section sol = np.diag(phi) x = np.array([i for i in range(sol.size)]) plt.plot(x,sol,'-bx', label='Solution')

plot the best possible numerical solution based on this grid best = np.where(x < x.size/2.0,1,0) plt.plot(x , best, '-rs',label = 'Best Numerical')

plot exact solution on fine grid x$_e$$xact = np.linspace(0, x.size, 1000)exact = np.where(x_exact < x.size/2.0, 1, 0)plt.plot(x_exact, exact,' k-', label =' Exact')$

plt.legend() plt.savefig(fname)



We can see that the solution looks quite diffusive. Because there is no actual diffusion, all of this represents false diffusion. To get a good solution, the false diffusion coefficient Γ_{false} should much less than the real diffusion coefficient Γ_{real}.

In 2D, our false diffusion coefficient looks like this:

$$\Gamma_{false} = \frac{\rho|\mathbf{u}|\Delta x \Delta y sin(2\Theta)}{4(\Delta y sin^3\Theta + \Delta x cos^3(\Theta))}$$

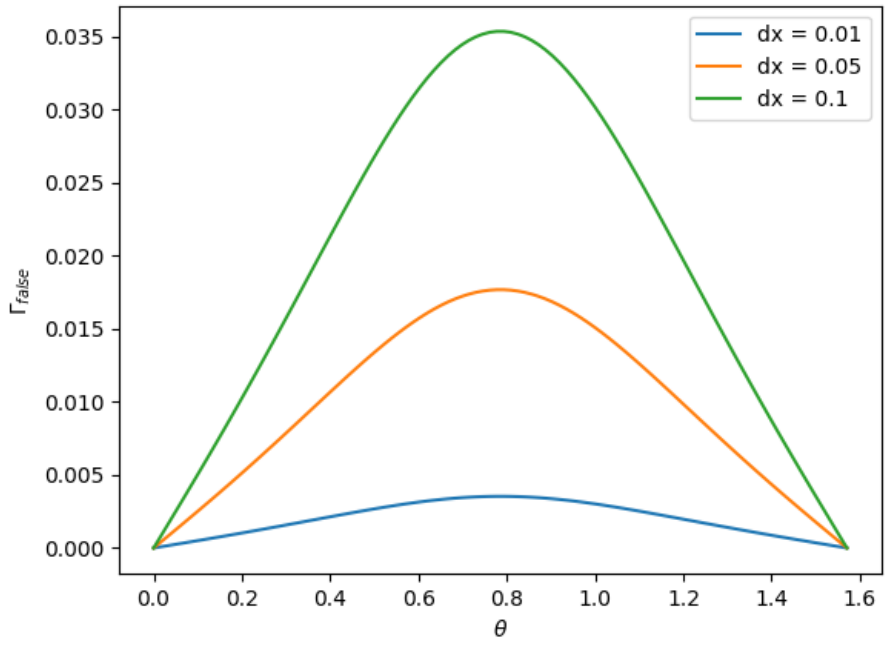
where $\Delta x, \Delta y$ are grid spacing in each direction and Θ is the angle the velocity makes with the x-axis. Here, we plot the values of Γ_{false} against different values of grid spacing and Θ. language=Python,label= ,caption= ,captionpos=b,numbers=none import matplotlib import matplotlib.pyplot as plt import numpy as np matplotlib.use('Agg') fname = '../pic/false$_d iffusion2D.png' assumevelocity, dens$ $[0.01, 0.05, 0.1] theta = np.linspace(0, np.pi/2, 100)$

for tables headers = ["delta", "theta", "gamma"]

calculate the false diffusion for d in delta: gamma = d*d*np.sin(2*theta)/4/(d*np.power(np.sin(theta
+ d*np.power(np.cos(theta),3)) plt.plot(theta,gamma,label = "dx = " +
str(d))

print("THETA GAMMA") print("THETA GAMMA") for i,theta$_i$*inenumerate*(*theta*) : *print*(”

show the plot plt.xlabel(r"θ") plt.ylabel(r"Γ_{false}") plt.legend() plt.savefig(fname)



Note:

- when flow is essentially parallel to the grid lines, the false diffusion is zero.
- False diffusion is most severe (highest) when the flow is at 45 degrees ($\frac{pi}{2} \approx 0.8$)
- Refining the grid spacing reduces the diffusion.
- To improve accuracy: use higher order advection schemes (reduce effects of false diffusion) and ensure grid is fine enough.

1.3 Non-Orthogonal Unstructured Grid

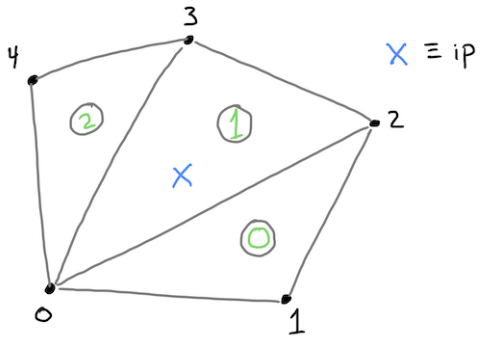
- There is no natural ordering available in an unstructured grid.
- In structured grid, the next neighboring control volume can be found by shifting the current cell index. In unstructured grid, we need a map that stores the cell connnectivity. In other words, each cell needs a list of all of its neighboring cells.
- Problems include grid geometry, interpolation and gradient reconstruction

1.3.1 Grid Geometry

We have a set of points representing the corners of the control volume. These points are connected by edges, defining a set of faces. Each face belongs to 2 control volumes, one on each side.

To calculate the grid geometry, we start with the faces and build the volumes. Assume that the faces are arbitrary polygons, these combine to make arbitrary polyhedral control volumes.

First, we start by choosing an arbitray corner node, then we connect it with the other corner nodes, creating a set of triangular faces. The area of each triangle can be calculated using cross products.



$$A_0 = \frac{1}{2}||(\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0)||$$
$$A_1 = \frac{1}{2}||(\mathbf{x}_2 - \mathbf{x}_0) \times (\mathbf{x}_3 - \mathbf{x}_0)||$$
$$A_2 = \frac{1}{2}||(\mathbf{x}_3 - \mathbf{x}_0) \times (\mathbf{x}_4 - \mathbf{x}_0)||$$

In general, for triangle with index i:

$$A_i = \frac{1}{2}||(\mathbf{x}_{i+1} - \mathbf{x}_0) \times (\mathbf{x}_{i+2} - \mathbf{x}_0)||$$

For even more general case with N_c corner nodes, the total area associated with the integration point ip is calculated as:

$$A_{ip} = \sum_{i=0}^{N_c-2} A_i = \frac{1}{2} \sum_{i=0}^{N_c-2} ||(\mathbf{x}_{i+1} - x_0) \times (\mathbf{x}_{i+2} - \mathbf{x}_0)||$$

Recall the centroid of the face, which is the location of the integration point. To find this centroid, we use the **area-weighted average** of the centroid for **each sub-divided** triangular faces, defined above. The centroid of a triangle is defined by the average of its corner positions:

$$\mathbf{x}_{c,i} = \frac{1}{3}(\mathbf{x}_i + \mathbf{x}_{i+1} + \mathbf{x}_{i+2})$$

with $i = 0, 1, ..., N_c - 2$. It follows that the integration point is:

$$\mathbf{x}_{ip} = \frac{1}{A_{ip}} \sum_{i=0}^{N_c-2} A_i \mathbf{x}_{c,i}$$

For the normal vector from he face, we can calculate similarly to the face area. This is because the cross product define vectors normal to each triangular sub-face. The unit normal vector can be obtained by:

$$\mathbf{n}_i = \frac{(\mathbf{x}_{i+1} - \mathbf{x}_0) \times (\mathbf{x}_{i+2} - \mathbf{x}_0)}{||(\mathbf{x}_{i+1} - \mathbf{x}_0) \times (\mathbf{x}_{i+2} - \mathbf{x}_0)||}$$

The normal vector at the integration point is calculated usin the an area-weighted average:

$$\mathbf{n}_{ip} = \frac{1}{A_{ip}} \sum_{i=0}^{N_c-2} A_i \mathbf{n}_i$$

Note: This is assuming the faces are nearly planar. If they are warped, we need to repeat this process (calculate $\mathbf{n}_i$ then calculate $\mathbf{n}_{ip}$) for different choice of x_0 until we get the best possible estimate of the normal vector.

Next, recall the volume of the cell is calculated as:

$$V_P = \int_V dV$$

We want to relate this integral to the face geometry. Using the following relation:

$$\nabla \cdot \mathbf{x} = \frac{\partial x}{\partial x} + \frac{\partial y}{\partial y} + \frac{\partial z}{\partial z} = 3$$

so the volume integral can be re-written as:

$$V_P = \frac{1}{3} \int_V \nabla \cdot \mathbf{x} dV$$

By doing it this way, we introduce a divergence operator into the volume integral. This can then be transformed into a surface integral by Gauss' theorem:

$$V_P = \frac{1}{3} \int_S \mathbf{x} \cdot \mathbf{n} dS$$

or as a discrete summation over all of the integration points:

$$V_P = \frac{1}{3} \sum_{ip=0}^{N_{ip}-1} \mathbf{x}_{ip,i} \cdot \mathbf{n}_{ip,i} A_{ip,i}$$

Recall the centroid of a volume P is given as:

$$\mathbf{x}_P = \frac{1}{V_P} \int_V \mathbf{x} dV$$

Similar to before, we like to express quantity under integral to involve some divergence operator, so that we can use Gauss' theorem to transform it into an area integral. This time, we rely on this trick:

$$\nabla \cdot (\mathbf{x}\mathbf{x}) = \mathbf{x} \nabla \cdot \mathbf{x} + \mathbf{x} \cdot \nabla \mathbf{x}$$

The first term on the RHS involves $\nabla \cdot \mathbf{x}$ which we already showed to equal to 3. So, we can write:

$$\nabla \cdot (\mathbf{x}\mathbf{x}) = 3\mathbf{x} + \mathbf{x} \cdot \nabla \mathbf{x}$$

The second term on the RHS can be expanded as follow:

$$\begin{aligned}\mathbf{x}\cdot\nabla\mathbf{x} &= \left(x\frac{\partial}{\partial x}+y\frac{\partial}{\partial y}+z\frac{\partial}{\partial z}\right)\mathbf{x}\\ &= \left(x\frac{\partial\mathbf{x}}{\partial x}+y\frac{\partial\mathbf{x}}{\partial y}+z\frac{\partial\mathbf{x}}{\partial z}\right)\\ &= x\left(\begin{bmatrix}1\\0\\0\end{bmatrix}+y\begin{bmatrix}0\\1\\0\end{bmatrix}+z\begin{bmatrix}0\\0\\1\end{bmatrix}\right)=\begin{bmatrix}x\\y\\z\end{bmatrix}\\ &= \mathbf{x}\end{aligned}$$

As a result:

$$\nabla\cdot(\mathbf{xx}) = 4\mathbf{x}$$

Now, we can rewrite the expression for the cell centroid:

$$\mathbf{x}_P = \frac{1}{4V_P}\int_V \nabla\cdot(\mathbf{xx})dV = \frac{1}{4V_P}\int_S(\mathbf{xx})\cdot\mathbf{n}dS$$

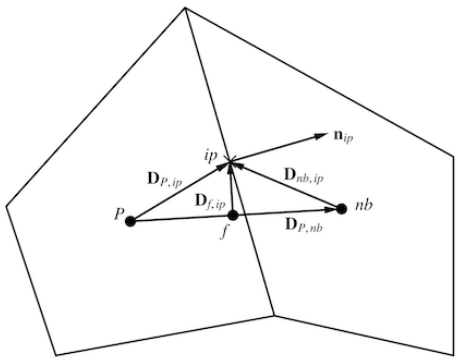
Same as before, we express as a discrete summation over the integration points:

$$\mathbf{x}_P = \frac{1}{4V_P}\sum_{ip=0}^{N_{ip}-1}\mathbf{x}_{ip,i}\mathbf{x}_{ip,i}\cdot\mathbf{n}_{ip,i}A_{ip,i}$$

Thus, we can define all the required face and cell geometry for unstructured grid calculations.

1.3.2 Interpolation

To do interpolations, we define the following variables for a particular control volume face (diagram):



Label	Description
P	control volume under consideration
nb	Neighboring control volume sharing the face containing ip
ip	Integration point location (face centroid)
f	Point along the vector connecting P to nb
$\mathbf{D}_{P,nb}$	Displacement vector from P to nb
$\mathbf{D}_{f,ip}$	Displacement vector from f to ip

Note: f could be anywhere along $\mathbf{D}_{P,nb}$ but best to make it so $\mathbf{D}_{P,nb}$ perpendicular to $\mathbf{D}_{f,ip}$, i.e. $\mathbf{D}_{f,ip} \cdot \mathbf{D}_{P,nb} = 0$. This minimizes $\mathbf{D}_{f,ip}$ which will also minimize the gradient correction term. In addition, if the grid is orthogonal, then this makes sure that $\mathbf{D}_{f,ip}$ is exactly zero.

We define the quantity f_{ip} to represent f as a function of $\mathbf{D}_{P,nb}$:

$$\mathbf{x}_f = \mathbf{x}_P + f_{ip}\mathbf{D}_{P,nb}$$

A general second order interpolation of a value ϕ to the integration point can be formulated as:

$$\phi_{ip} = (1 - f_{ip})\phi_P + f_{ip}\phi_{nb} + \mathbf{D}_{f,ip} \cdot \left[(1 - f_{ip})\nabla\phi\Big|_P + f_{ip}\nabla\phi\Big|_{nb}\right]$$

Note:

- 1st term on RHS is an **inverse distance interpolation** to the point f.
- 2nd term on RHS is **non-orthogonal correction** from f to ip. Here, f is estimated based on an inverse distance interpolation along $\mathbf{D}_{P,nb}$.

1.3.3 Gradient Reconstruction

We focus on the Gauss-based method because they are simple to explain. This method is based on the Gauss' theorem, which allows us to write:

$$\int_V \nabla\phi dV = \int_S \phi \mathbf{n} dS$$

Assume $\nabla\phi$ to be piecewise constant in each cell, we can rewrite the above equation for cell P as:

$$\nabla\phi\Big|_P V_P = \sum_{ip=0}^{Nip-1} \phi_{ip}\mathbf{n}_{ip}A_{ip}$$

where we have replaced the surface integral with a discrete summation. Solving for the gradient:

$$\nabla\phi\Big|_P = \frac{1}{V_P}\sum_{ip=0}^{Nip-1}\phi_{ip}\mathbf{n}_{ip}A_{ip}$$

Note: This interpolation of ϕ_{ip} depends on the gradient, making this a non linear system. A few solutions:

- ignore the gradient in the calculation of ϕ_{ip}, only use the inverse distance approximation based on ϕ_P and ϕ_{nb}. Not very accurate because gradients are at most 1st order accurate. ANSYS Fluent "Green Gauss Cell-Based" uses this. In fact, $f_{ip} = 0.5$ is used so values are not even accurate on orthogonal grid.

- Use the most recent value of ϕ_{ip} and solve the system iteratively until it converges. Need to update the gradients many time, so this method is computational expensive. Convergence is also quite poor, so it is not recommended.

- Startout not using the gradient corrections, then calculate the gradient in multiple stages. This method is efficient if the number of gradient updates is low. The accuracy is much better than ignoring the gradient correction alltogether.

- Create a linear system which solves all of the gradients simulataneously. This is expensive, requires coding complexity and can be expensive.

1.3.4 Discretization Details

The transient and source terms are **volume-based terms**, so they can be treated just similarly to structured grids. Advection and diffusion terms are **surface-based terms**, so they need to be treated differently for unstructured grids.

1. Advection Terms The discretized advection term is given as:

$$\int_V \nabla\cdot(\mathbf{u}\phi)dV = \sum_{i=0}^{N_{ip}-1}\mathbf{u}_{ip}\cdot\mathbf{n}_{ip}\phi_{ip}A_{ip}$$

   Generally, $\mathbf{u}_{ip}$ is replaced by the advecting velocity $\hat{\mathbf{u}}_{ip}$. A similar definition for $\hat{\mathbf{u}}_{ip}$ can be developed for unstructured grid. We use a

2nd order upwind interpolation:

$$\phi_{ip} = \phi_u + \nabla\phi\Big|_u \cdot \mathbf{D}_{ip,u}$$

where u represents the upwind cell, i.e.:

$$u = \begin{cases} P & \text{if } \dot{m}_{ip} \geq 0 \\ nb & \text{if } \dot{m}_{ip} < 0 \end{cases}$$

Note:

- linearization is carried out with respect to ϕ_u, ensuring the linearization coefficients have correct signs, similar to UDS.
- CDS and QUICK can also be adapted for unstructured grids.
- Sometimes a flux limiter is applied to ensure the integration point value is bounded by the the surrounding cell values. This is important in flow with discontinuities (shocks)

2. Diffusion Terms The discretization diffusion term is given as:

$$\int_V \nabla \cdot \mathbf{J}_\phi dV = \sum_{i=0}^{N_{ip}-1} \mathbf{J}_{\phi,ip} \cdot \mathbf{n}_{ip} A_{ip}$$

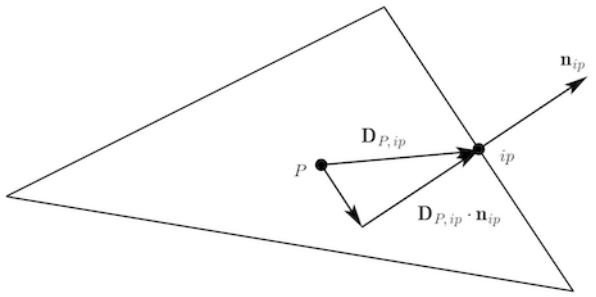
In Fourier's or Fick's law, the diffusive flux $\mathbf{J}$ is proportional to the gradient of ϕ

$$\mathbf{J}_{\phi,ip} = -\Gamma_P \nabla\phi\Big|_{ip}$$

To calculate the gradient at the integration point, we need to enforce a continuous flux across all faces. Mathematically, this continuity of the diffusive flux is expressed as:

$$\Gamma_P \nabla\phi\Big|_{ip} \cdot \mathbf{n}_{ip} = \Gamma_{nb} \nabla\phi\Big|_{ip,nb} \cdot \mathbf{n}_{ip}$$

The derivatives normal to the integration point are calculated by extrapolating from the cell center to a point on a line that intersects ip and is normal to the control surface:



We then use a finite difference approximation to evaluate the normal derivative along this line:

$$\nabla\phi\Big|_{ip,P} \cdot \mathbf{n}_{ip} = \frac{\phi_i p - [\phi_P + \nabla\phi|_P \cdot (\mathbf{D}_{P,ip} - (\mathbf{D}_{P,ip} \cdot \mathbf{n}_i p)\mathbf{n}_i p)]}{\mathbf{D}_{P,ip} \cdot \mathbf{n}_i p}$$

We form a similar expression for control volume nb, then equating the two through flux balance expression. We then get the following expression for the integration point value, satisfying the heat flux from both sides of the control surface:

$$\begin{aligned}\phi_{ip} = &\frac{\Gamma_{nb}(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip})}{\Gamma_{nb}(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}) - \Gamma_P(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})}\phi_{nb} - \frac{\Gamma_P(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})}{\Gamma_{nb}(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}) - \Gamma_P(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})}\phi_P \\ &+ \frac{\Gamma_{nb}(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip})(\mathbf{D}_{nb,ip} - (\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})\mathbf{n}_{ip})}{\Gamma_{nb}(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}) - \Gamma_P(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})} \cdot \nabla\phi|_{nb} \\ &- \frac{\Gamma_P(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})(\mathbf{D}_{P,ip} - (\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip})\mathbf{n}_{ip})}{\Gamma_{nb}(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}) - \Gamma_P(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})} \cdot \nabla\phi|_P\end{aligned}$$

We then substitute this ϕ_{ip} into the equation for the normal derivative to get an expression in terms of the cell-centered value.

$$\begin{aligned}\nabla\phi|_{ip,P} \cdot \mathbf{n}_{ip} = &\frac{\phi_{nb} - \phi_P}{(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}) - \frac{\Gamma_P}{\Gamma_{nb}}(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})} + \frac{(\mathbf{D}_{nb,ip} - (\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})\mathbf{n}_{ip})}{(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}) - \frac{\Gamma_P}{\Gamma_{nb}}(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})} \cdot \nabla\phi|_{nb} \\ &- \frac{(\mathbf{D}_{P,ip} - (\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip})\mathbf{n}_{ip})}{(\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}) - \frac{\Gamma_P}{\Gamma_{nb}}(\mathbf{D}_{nb,ip} \cdot \mathbf{n}_{ip})} \cdot \nabla\phi|_P\end{aligned}$$

This expression can then be used to calculate the diffusive flux from the P cell, with a similar expression being used for nb, which ensures the flux is consistent. The two gradient terms in the equation above

account for non-orthogonality in the grid. In the limit of a completely orthogonal grid the equation above reduces to the harmonic mean formulation of Patankar. In the limit of an orthogal grid with constant diffusivity, Γ, this expression is identical to that used in our one-dimensional codes.