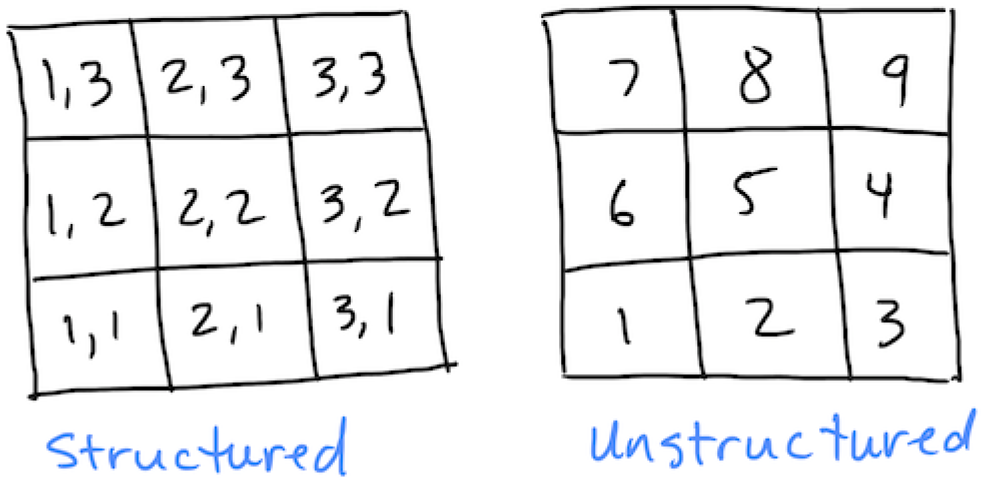
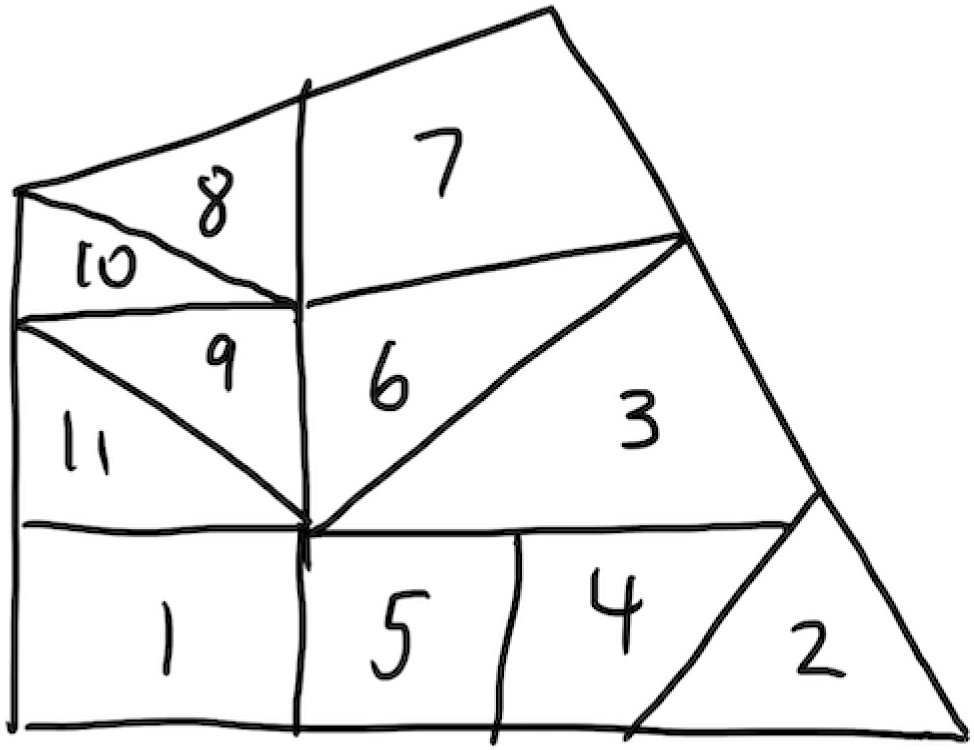
Two and Three Dimensional Grids

In this lesson, we discuss the details of how to extend what we have learned to two and three dimensional grids. Throughout the lessons so far, we have been careful to avoid making too many assumptions that were specific to the fact that the cases were one-dimensional. It is therefore quite straightforward to exend what we have done so far to higher dimensional grids, provided they have an ordered, orthogonal structure.

This leads us to a distinction between what are called "structured" and "unstructured" grids. It is important to note that these labels are primarily associated with how the grid cells are labelled, not their actual topologies (although structured grids are restricted by the topologies that can be represented with structured storage schemes). The figure below shows two identical grids where one is stored as a structured grid (i.e. with ordered row and column labelling) and one that is stored as an unstructured grid (i.e. with arbitrary cell labelling).



Clearly, for grids that are composed of abitrary polyhedra with no inherent structure, there is no option other than storing the grid with an unstructured labelling scheme, as shown below.



In this lesson, we will first look at how to work with higher dimensional structured grids, then we will discuss some of the details of unstructured grids.

Structured Grids

General Discretization

We will now discretize a generic transport equation over a two-dimensional structured grid. The extension to three dimensions from this is then almost trivial. From Lesson 1, recall the generic transport equation:

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

Carrying through the space-time integration, we arrive at:

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

Suppose we want to make this general form represent the conservation of mass equation, then we set $\phi = \rho, \mathbf{J}_\phi = 0$ and $S_\phi = 0$. If we then label the integration points as w (west), e (east), s (south), and n (north), we arrive at:

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

Assuming density is constant we arrive at:

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

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

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

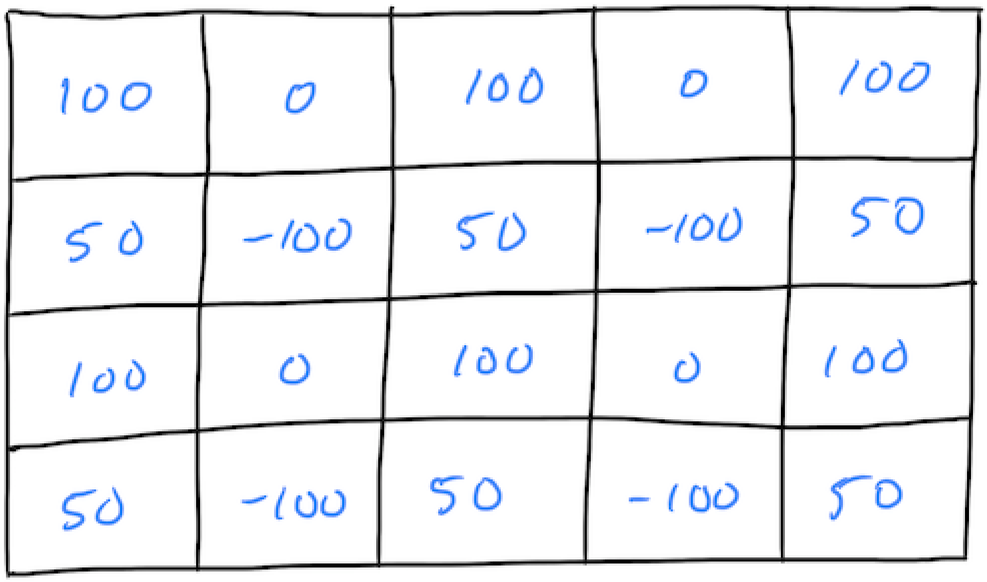
Similarly, we can derive the momentum equation for the y component of velocity. To complete the discretization of the momentum equation, we should subtract the mass equation, multipled by the appropriate velocity component at cell P, from each momentum equation. Then, we choose a time integration scheme to complete the transient term, choose an advection scheme to complete the advection term (although we will still linearize based on UDS), and approximate the derivative terms using finite differences.

Without going through all of the details, the momentum equations in the x and y directions can be written in terms of their linearization coefficients (similar to Lesson 5):

$$a_P u_P = -a_W u_W - a_E u_E - a_S u_S - a_N u_N + b_u - \frac{p_E - p_W}{2\Delta x} V_P$$

$$a_P v_P = -a_W v_W - a_E v_E - a_S v_S - a_N v_N + b_v - \frac{p_N - p_S}{2\Delta y} V_P$$

Similar to the one-dimensional case, an oscillatory pressure field can be detected as smooth if we are not careful. In two dimensions, the situation is actually worse because oscillatory modes can develop in both directions. The diagram below shows an example that would be accepted by the solver as a smooth pressure field:



100	0	100	0	100
50	-100	50	-100	50
100	0	100	0	100
50	-100	50	-100	50

As in one-dimension, this problem can be overcome by (i) using a staggered grid or (ii) using a collocated grid with different advected and advecting velocities. Since the staggered grid become more complicated as the number of dimensions increases, we will only consider the collocated approach. The derivation follows a very similar pattern to what was shown in Lesson 5. The resulting expressions for the advecting velocities in each direction are:

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

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

where the superscript on $\hat{d}$ denotes the equation with which it is associated. Similar to one-dimension, the coupling can either be done in a direct or segregated method (e.g. SIMPLE or SIMPLEC).

False Diffusion

We have already discussed false diffusion in one dimension and found that although a Taylor series analysis shows it is a serious problem, it is not as bad as the analysis would indicate. We found that using UDS for linearization and correcting the advective fluxes with a higher order method was an effective method of getting good results in these cases.

In two (and three) dimensions the problem of false diffusion comes from a different source than in one dimension, and is associated with cases where the flow streamlines are not well aligned with the grid lines. For steady advection of a scalar quantity with no sources and negligible real diffusion in comparison to advection, the discrete transport equation is given as:

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

If we use UDS for advection, and we assume a positive flow in both the x and y directions, we get:

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

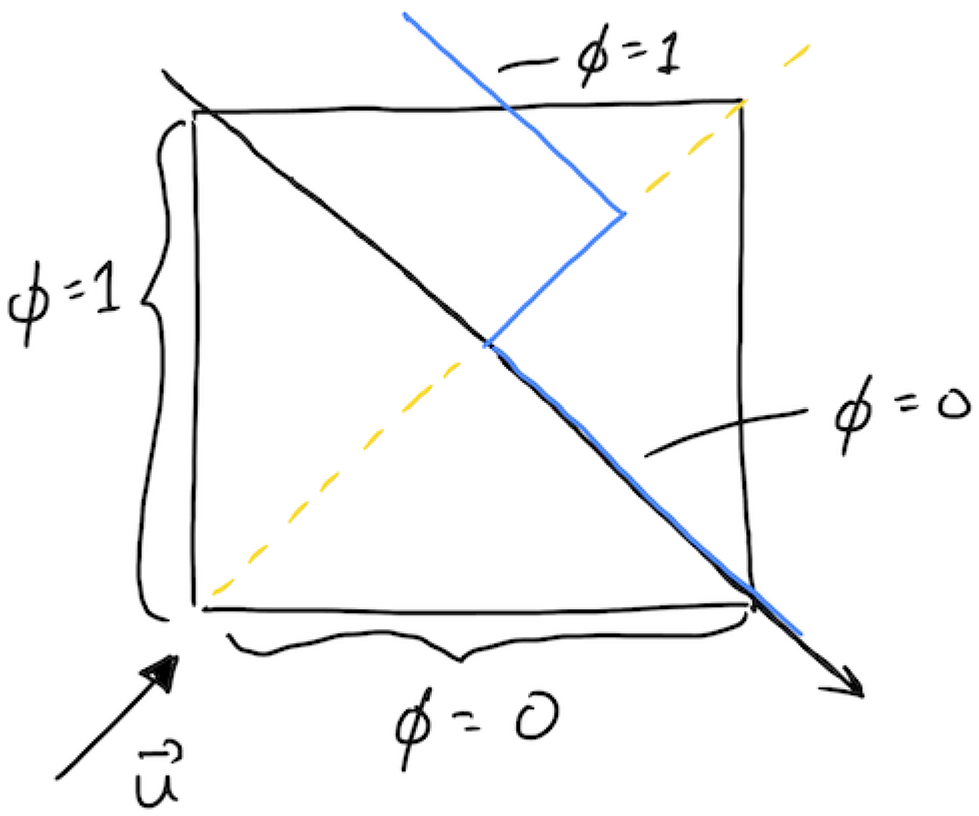
Solving for ϕ_P we get:

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

If we consider a flow at 45 degrees to the x axis, then $\dot{m}_e = \dot{m}_w = \dot{m}_n = \dot{m}_s$ and the solution becomes:

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

If a value of 0 is advected from the bottom surface of the domain and a value of 1 is advected from the left surface of the domain, the exact solution is a step profile at any cross-section perpendicular to the flow direction, as shown below:



The code cell below shows the actual solution to the problem:

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Create an array to hold the solution
phi = np.zeros((7, 7))

# Fill in the advected values on the left
phi[:,0] = 1

# 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(phi)

# Plot the solution along the diagonal cross-section
sol = np.diag(phi)
x = np.array([_ for _ in range(sol.size)]) # Scale of axis is arbitrary
plt.plot(x, sol, 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, label='Best Numerical')

# Plot the exact solution based on a fine grid
x_exact = np.linspace(0, x.size, 1000)
exact = np.where(x_exact < x.size/2.0, 1, 0)
plt.plot(x_exact, exact, label='Exact')

# Show the plot with legend
plt.legend()
plt.show()
```

It can clearly be seen that the solution looks quite diffusive. Since there is no actual diffusion, all of this represents false diffusion. In order to get a good solution the false diffusion coefficient Γ_{false} should be much less than the real diffusion coefficient, Γ_{real}.

An approximate expression for the false diffusion coefficient in two dimensions can be found to be:

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

where Δx and Δy are the grid spacings in each direction and θ is the angle that the velocity makes with the x axis.

Below, we plot the value of Γ_{false} for different angles and grid spacings (assuming equal grid spacings in x and y).

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Assume unit values of density and velocity magnitudes

# Set the parameters of study
delta = [0.01, 0.05, 0.1]
theta = np.linspace(0, np.pi/2, 100)

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


# Show the plot
plt.xlabel(r"$\theta$")
plt.ylabel(r"$\Gamma_{false}$")
plt.legend()
plt.show()
```

It can be seen in the plot above that false diffusion is most severe when the flow is at 45 degrees to the grid lines and is essentially zero when the flow is parallel to the grid lines. It is also shown that refining the grid spacing reduces false diffusion.

In order to improve accuracy we can use higher order advection schemes to reduce the effects of false diffusion as much as possible, while also ensuring the grid is fine enough.

Non-Orthogonal Unstructured Grids

One of the differences between structured and unstructured grids is that there is no natural ordering available in an unstructured grid. Unlike a structured grid, where the neighbouring control volumes can be found by an appropriate shifting of the given cell index, unstructured grids must store a map of the cell connectivity. That is to say, for each cell, there must be a list of all of its neighbouring cells.

Other issues that are faced in non-orthogonal unstructured grids are the calculation of grid geometry, interpolation, and gradient reconstruction. Those topics will be discussed below.

Grid Geometry

In general, an unstructured grid will be defined by a set of points representing the corners of the control volume. These points are connected by edges which define a set of faces. Each face then belongs to two control volumes, one on each side. To calculate the grid geometry, we start with the faces and then build the volumes. Here we will assume that the faces are arbitrary polygons that combine to make arbitrary polyhedral control volumes.

To calculate the face geometry, we start by choosing a single (arbitrary) corner node and connect it with each of the other corner nodes, creating a set of triangular faces. Using cross products, we can calculate the area of each triangle.

Subfaces

For the example above:

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

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

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

Or, in general for the triangle with index i:

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

In a general case with N_c corner nodes, the total area of the face associated with the integration point ip can be calculated as:

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

We also need to find the centroid of the face, which is the position of the integration point (recall Lesson 1, where we showed this positioning is critical for second order accuracy). To find the centroid, we use the area-weighted average of the centroids of each of the sub-divided triangular faces, defined above. The centroid of a triangle is defined by the average of its corner positions:

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

where $i = 0, 1, \ldots, N_c - 2$.

The integration point location is therefore:

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

We will also need the normal vector from the face, which can be calculated similarly to the face area, since the cross products define vectors normal to each triangular sub-face. To obtain a unit normal vector, each vector is scaled by its magnidude, i.e.:

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

Then, similar to the centroid, an area-weighted average is used to compute the normal vector at the integration point:

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

Note that this assumes the face is nearly planar. If there is a chance that the face is highly warped, it may be a good idea to repeat this process for each possible choice of $\mathbf{x}_0$ and average all of the results to get the best possible estimate of the normal vector.

Now that all of the face geometry is defined, we can calculate the geometry of the cell. The volume of the cell is defined as:

$$V_P = \int_V dV$$

Ideally, we want to relate this integral to the face geometry. An interesting trick can be applied to this integral by noting the following:

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

As a result, the volume integral can be re-written as:

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

Essentially, we have just multiplied and divided the equation by the value 3, which has no effect. However, this has introduced a divergence operator into the volume integral that can be transformed into a surface integral by Gauss' theorem:

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

where $\mathbf{n}$ is the unit normal vector directed away from the surface of the control volume. This can then be replaced by a discrete summation over all of the integration points:

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

All of the quantites in the summation are properties of the face geometry that are known. Therefore, the volume of the cell can be calculated in this way.

The definition of the centroid of the volume P is given as:

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

Once again, we would like to express this as the divergence of a vector such that we can convert it into a surface integral depending on quantities at the integration point. Once again, we perform a particular manipulation to the equation to accomplish this. In this case, consider the following:

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

In the first term on the right side of the equation above, we have $\nabla \cdot \mathbf{x}$, which we have already shown is equal to 3. Therefore, we can say:

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

Expanding the second term on the right side of the equation above:

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

Therefore:

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

We can then rewrite the expression for the cell centroid as:

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

Expressing as a discrete summation over the integration point faces:

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

This defines all of the required face and cell geometry required for unstructured grid calculations.

Interpolations

In order to perform interpolations on the grid, we define the following points associated with a particular control volume face:

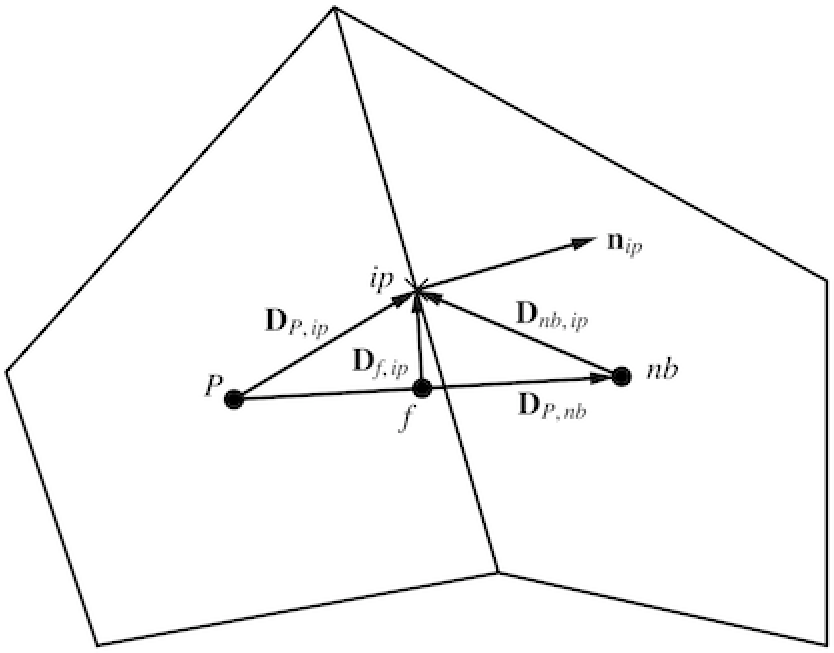
Label	Description
P	Control volume under consideration
nb	Neighbouring control volume sharing the face containing ip
ip	Integration point location (face centroid)
f	Point along the vector connecting P to nb

Then, we define the following displacement vectors:

Label	Description
$\mathbf{D}_{P,nb}$	Displacement vector from P to nb

Label	Description
$\mathbf{D}_{f,ip}$	Displacement vector from f to ip

These are illustrated further in the diagram below:



In practice, f could be located anywhere along the vector $\mathbf{D}_{P,nb}$, but the best practice is to place it such that $\mathbf{D}_{f,ip}$ is perpendicular to $\mathbf{D}_{P,nb}$, i.e. $\mathbf{D}_{f,ip} \cdot \mathbf{D}_{P,nb} = 0$. This minimizes the size of $\mathbf{D}_{f,ip}$, which minimizes the size of the gradient correction term, which will be shown below. Also, if the grid happens to be orthogonal, this ensures that $\mathbf{D}_{f,ip}$ is exactly zero.

Based on the placement of f we define the quantity f_{ip} to represent the location of 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:

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

where the first term is an inverse distance interpolation to the point f and the second term is a non-orthogonal correction from f to ip. In the non-orthogonal correction term, the gradient at f is estimated based on an inverse distance interpolation along $\mathbf{D}_{P,nb}$.

It is now clear that we need to know the gradients of all variables in order to perform interpolations on the grid. This will be considered next.

Gradient Reconstruction

There are several different ways that the gradient can be reconstructed. Here we will focus on Gauss-based methods, since they are relatively simple to explain. However, it is worth noting that there are methods based on least-squares that are both popular and effective.

The Gauss-based gradient reconstruction method is based on Gauss' theorem, which allows us to write:

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

If we assume $\nabla\phi$ to be piecewise constant in each cell, the equation above can be re-written for the cell P as:

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

where the surface integral has been replaced with a discrete summation. Solving for the gradient:

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

The problem with the expression above is that the interpolation of ϕ_{ip} depends on the gradient, making this a non-linear system. There are a few solutions to this problem:

- Ignore the gradients in the calculation of ϕ_{ip} and simply use the inverse distance approximation based on ϕ_P and ϕ_{nb}. This is not very accurate, since the gradients will be at most first order accurate. This is the current default behaviour in ANSYS Fluent's "Green-Gauss Cell-Based" method (in fact they assume $f_{ip} = 0.5$, so the values are not even accurate on an orthogonal grid).
- Include the most recent gradients in the calculation of ϕ_{ip}, and solve the system iteratively until it converges. This method can be computationally expensive, since the gradients may need to be updated many times. The convergence of this method has been observed to be quite poor (it may diverge easily), so it is not recommended.
- Calculate the gradients in multiple stages, first using no gradient corrections. This will give a low-order estimate of the gradients. Then, repeat the process using the first-pass gradients in the correction term. Optionally, the process can be repeated a few times with the previous gradient solution being used in the gradient correction term. For convergence of this method, it is important to always use the gradients from the last complete gradient evaluation. Since the order of update can be quite random, it is better to always use the older values that use some new and some old, which causes instability in the solution. This method is fairly efficient if the number of gradient updates is kept low (e.g. a maximum of two or three updates afer the initial pass). The accuracy is much better than the option of ignoring the gradient correction altogether.
- Create a linear system so that all of the gradients can be solved simultaneously. This method is effective, but requires a significant amount of additional memory, additional coding complexity, and can be computationally expensive.

Discretization Details

The transient and source terms (including pressure source terms) can be treated similarly to structured grids, since they are volume-based terms. Advection and diffusion terms are surface terms, and therefore must be considered differently for non-orthogonal unstructured grids. These details will be discussed briefly below.

Advection Terms

The discretized advection term from Lesson 1 was given as:

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

In general, the value $\mathbf{u}_{ip}$ is replaced by the advecting velocity $\hat{\mathbf{u}}_{ip}$, as discussed Lesson 5. Although it won't be discussed here, a similar definition for $\hat{\mathbf{u}}_{ip}$ can be developed for unstructured grids. Interpolation of ϕ_{ip} carries some of the same stability constraints as in one-dimension. Therefore is is common to use a second-order upwind interpolation, defined by:

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

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

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

Linearization is carried out with respect to ϕ_u, which ensures the linearization coefficients have the correct sign, similar to the UDS scheme discussed previously. There are of course many other possible advection schemes, including CDS and QUICK which can be adapted to general unstructured grids. It should be noted that sometimes a "flux limiter" needs to be applied to ensure the integration point value is bounded by the surrounding cell values. This is particularly important for flows with discontinities (e.g. shocks), but is also useful to compensate for the fact that gradients may not always be accurate and could cause unbounded face values to occur.

Diffusion Terms

From Lesson 1, the discretized diffusion term was given as:

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

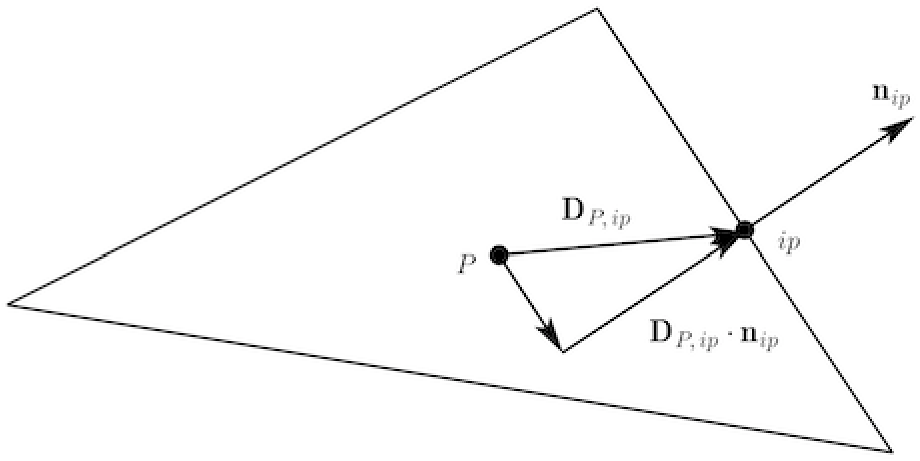
Typically, the diffusive flux $\mathbf{J}$ is proportional to the gradient of ϕ, e.g. Fourier's law or Fick's law. Therefore we assume:

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

Calculation of the gradient at the integration point is based on the enforcement of a continuous flux across all faces. The continuity of diffusive flux at a face is expressed mathematically as

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

The derivatives normal to the integration point are computed by extrapolating from the cell-centers to a point located on a line that intersects ip and is normal to the control surface, as shown in the figure below.



A finite-difference approximation can then be used to evaluate the normal derivative along this line as:

$$\nabla\phi|_{ip,P} \cdot \mathbf{n}_{ip} = \frac{\phi_{ip} - \left[\phi_P + \nabla\phi|_P \cdot (\mathbf{D}_{P,ip} - (\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip})\mathbf{n}_{ip})\right]}{\mathbf{D}_{P,ip} \cdot \mathbf{n}_{ip}}$$

Forming a similar expression for the control volume nb and equating the two through the flux balance expression, results in the following expression for the integration point value that satisfies the heat flux from both sides of the control surface:

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

Substituting the expression above back into the equatino for the normal derivative results in the following expression for the normal derivative, in terms of the cell-centered values, which ensures a flux balance across the control surface:

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

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.

Summary

With this lesson, you now have most of the information required if you were to extend our one dimensional code to higher dimensions. You also now have an understanding of the additional considerations that come into play in unstructured CFD codes.