One-Dimensional Convection of a Scalar

Problem Definition

In this lesson, we explore the solution for convection of a scalar in one-dimension. So far, we have only considered diffusion processes, and we will now include the advection term. For thermal convection, the advection-diffusion equation is given by:

$$\frac{\partial\left(\rho c_p T\right)}{\partial t} + \nabla\cdot\left(\rho c_p \mathbf{u} T\right) = k\nabla^2 T + S$$

For the current lesson, as in the previous one, we shall consider the density, ρ and specific heat capacity, c_p, to be constants.

At this point, we will assume that the flow field, $\mathbf{u}$ is already known, and that we are simply solving for the resulting temperature field. Later, we will discuss how to solve the flow field.

We will also define the equation for conservation of mass (with no mass sources), as this will be required for the subsequent discretization:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot\left(\rho\mathbf{u}\right) = 0$$

Discretization

As we have done previously, the energy equation is discretized by integrating the governing equation over space and also over time.

$$\int_{t_0}^{t_1}\int_V \frac{\partial\left(\rho c_p T\right)}{\partial t} dt dV + \int_{t_0}^{t_1}\int_V \nabla\cdot\left(\rho c_p \mathbf{u} T\right) dt dV = \int_{t_0}^{t_1}\int_V k\nabla^2 T dV dt + \int_{t_0}^{t_1}\int_V S dV dt$$

Integration of the transient, diffusive, and source terms has already been covered in the previous lessons, so will not be shown directly here.

It is noted that integration over the time interval Δt proceeds similarly as for the diffusion and source terms. Since the integrated equation will eventually be divided by Δt (as before) we will simply look at the volume integraion. The volume integral within the advection term is converted to a surface integral using Gauss' divergence theorem, according to:

$$\int_V \nabla\cdot\left(\rho c_p \mathbf{u} T\right) dV = \int_S \left(\rho c_p \mathbf{u} T\right)\cdot\mathbf{n} dS$$

The surface integral is then appoximated as a discrete summation over the integration points:

$$\int_V \nabla\cdot\left(\rho c_p \mathbf{u} T\right) dV = \sum_{i=0}^{N_{ip}-1}\left(\rho c_p \mathbf{u} T\right)\cdot\mathbf{n}_{ip} A_{ip}$$

For the one-dimensional control volume, P, this results in

$$\int_V \nabla\cdot\left(\rho c_p \mathbf{u} T\right) dV = \rho c_p u_e T_e A_e - \rho c_p u_w T_w A_w$$

Defining the mass flux as:

$$\dot{m} = \rho u A$$

the discretized advection term can then be re-written as:

$$\int_V \nabla\cdot\left(\rho c_p \mathbf{u} T\right) dV = \dot{m}_e c_p T_e - \dot{m}_w c_p T_w$$

This term can be considered to represent the difference in energy between a parcel of fluid that enters a control volume at T_w and leaves at T_e, as shown below.

AdvectionBalance

The discretized energy equation is then:

$$\frac{\left(\rho c_p T_P V_P\right)^{t+\Delta t/2} - \left(\rho c_p T_P V_P\right)^{t-\Delta t/2}}{\Delta t} + \dot{m}_e c_p T_e - \dot{m}_w c_p T_w = -D_w\left(T_P - T_W\right) + D_e\left(T_E - T_P\right) + S_P V_P$$

In the equation above, the transient term is treated as before, where the interpolation of the values at the times $t+\Delta t/2$ and $t-\Delta t/2$ define the time integration scheme. It should be noted that the discretization is not yet complete, since it has not yet been specifed how to determine the mass flux and temperature values at the east and west integration points. The correct form of the temperature interpolation will be considered in detail shortly, and calculation of the mass flux will be considered in the next lesson (for now we assume it is known).

Before moving on to the interpolation of the face values, we must consider whether or not the given equation is independent of temperature level (in the absence of source terms) according to Rule 4 given in Lesson 1. It can be argued that it should should be, since the transient, advection, and diffusion terms involve only derivatives of temperature. It can be seen that this should be true, provided mass is conserved (i.e. $\dot{m}_e = \dot{m}_w$). This is easy to ensure in one dimension, but more difficult in multidimensional problems. In general, we cannot assure that the numerically computed mass fluxes will always be conservative while the energy equation is being solved. This can cause major problems with the solution process, since the effect of failing to conserve mass would be seen as an apparent energy source (or sink) within the domain.

We can get around this problem by subtracting the discretized conservation of mass equation from the energy equation. Since we have assumed the density to be constant, the conservation of mass equation can be discretized in quite a simple manner, resulting in:

$$\dot{m}_e - \dot{m}_w = 0$$

Exercise: Derive the discretized conservation of mass equation shown above.

Multiplying the equation above by c_p and a reference temperature (let's choose this to be T_P) and subtracting from the discretized energy equation results in:

$$\frac{\left(\rho c_p T_P V_P\right)^{t+\Delta t/2} - \left(\rho c_p T_P V_P\right)^{t-\Delta t/2}}{\Delta t} + \dot{m}_e c_p\left(T_e - T_P\right) - \dot{m}_w c_p\left(T_w - T_P\right) = -D_w\left(T_P - T_W\right) + D_e\left(T_E - T_P\right) + S_P V_P$$

This means that if there is a positive imbalance of mass ($\dot{m}_e > \dot{m}_w$), there will be a negative source in the energy equation to conterbalance the positive source created by the imbalance itself. If there is a negative imbalance, the opposite is true. Therefore, this step in the discretization can help with the stability of the numerical method by counteracting the effects of erroneous energy souces, such that the equations are again independent of the temperature level.

Analysis of the Advection Term with Explicit Time Integration

Let us first assume that we can interpolate the integration point values in the advection term using a simple central difference (i.e. piecewise linear) approximation:

$$T_e = \frac{1}{2}\left(T_P + T_E\right)$$

$$T_w = \frac{1}{2}\left(T_W + T_P\right)$$

We will also assume an explicit time integration scheme with no source term at this point. Keeping the T_P terms arising from the subtraction of the mass equation as implicit (i.e. current timestep) terms, results in the discrete equation:

$$\frac{\rho c_p V_P\left(T_P - T_P^o\right)}{\Delta t} + \dot{m}_e c_p\left[\frac{1}{2}\left(T_P^o + T_E^o\right) - T_P\right] - \dot{m}_w c_p\left[\frac{1}{2}\left(T_W^o + T_P^o\right) - T_P\right] = -D_w\left(T_P^o - T_W^o\right) + D_e\left(T_E^o - T_P^o\right)$$

where the temperatures with the superscript 'o' are evaluated at the previous timestep, while those without a superscript are those for the current timestep (i.e. those that are being solved). Grouping together all of the terms that depend on each temperature, results in:

$$\left(\frac{\rho c_p V_P}{\Delta t} + c_p\dot{m}_w - c_p\dot{m}_e\right)T_P = \left(\frac{\rho c_p V_P}{\Delta t} + \frac{c_p\dot{m}_w}{2} - \frac{c_p\dot{m}_e}{2} - D_e - D_w\right)T_P^o + \left(D_e - \frac{c_p\dot{m}_e}{2}\right)T_E^o + \left(D_w - \frac{c_p\dot{m}_w}{2}\right)T_W^o$$

If we assume that mass is conserved, i.e. $\dot{m}_e = \dot{m}_w$, then:

$$\frac{\rho c_p V_P}{\Delta t}T_P - \left(\frac{\rho c_p V_P}{\Delta t} - D_e - D_w\right)T_P^o - \left(D_e - \frac{c_p\dot{m}_e}{2}\right)T_E^o - \left(D_w - \frac{c_p\dot{m}_w}{2}\right)T_W^o = 0$$

According to Rule 2 from Lesson 1, the coefficient on T_P must be positive and the coefficients on the remaining terms must be negative. For the coefficient on T_P^o, this requires

$$D_e + D_w \le \frac{\rho c_p V_P}{\Delta t}$$

or:

$$\Delta t \le \frac{\rho c_p V_P}{D_e + D_w}$$

This is the same timestep restriction that was found for the explicit time integration scheme in Lesson 3, i.e.:

$$\frac{\alpha\Delta t}{\Delta x^2} \le \frac{1}{2}$$

Therefore, the advection term has not changed the timestep restriction thus far.

It is noted, however, that the coefficients on T_E^o and T_P^o could become positive for certain mass flow rates. For the east face, preventing this coefficient from becoming positive would require:

$$D_e \ge \frac{c_p\dot{m}_e}{2}$$

Expanding each of these terms and simplifying:

$$\frac{kA}{\Delta x} > \frac{c_p\rho u A}{2}$$

$$\Delta x < \frac{2k}{\rho c_p u}$$

In terms of the thermal diffusivity, α, it is therefore required that:

$$\frac{u\Delta x}{\alpha} < 2$$

This places a restriction on the spatial grid size, Δx, that can be used to compute a solution to an advection-diffusion equation. In order to have a stable, oscillation-free solution, both the spatial and temporal restrictions given above must be satisfied. Multiplying the left and right sides of both conditions together:

$$\frac{\alpha\Delta t}{\Delta x^2}\cdot\frac{u\Delta x}{\alpha} < \frac{1}{2}\cdot 2$$

$$\frac{u\Delta t}{\Delta x} < 1$$

The quantity on the left side of the equation above is known as the Courant number, Co. The time and space step restriction then can be written as:

$$\mathrm{Co} < 1$$

This condition is commonly known as the Courant-Friedrichs-Lewy (CFL) Condition. Physically, this implies that the flow field is not allowed to advect a parcel of fluid more than a distance Δx in the time Δt.

The next question to ask is whether or not this restriction is serious. To answer this question, we will consider the example of flow in a tube with constant wall temperature, T_w. This problem has an exact solution given as:

$$\frac{T_w - T(x)}{T_w - T_{in}} = \exp\left(-\frac{hP}{\dot{m}c_p}x\right)$$

where T_{in} is the inlet temperature, h is the convection coefficient associated with the wall heat transfer, and P is the perimeter of the tube. Let us consider the solution of this problem up until the point there the bulk temperature difference ($T_w - T(x)$) has reached 5% of the temperature difference between the wall and the inlet ($T_w - T_{in}$), i.e.

$$\frac{T_w - T(x_L)}{T_w - T_{in}} = 0.05$$

where x_L represents the axial location where this condition is reached. Let us further assume that the heat transfer coefficient is given in terms of the Nusselt number, i.e.:

$$\mathrm{Nu} = \frac{hD}{k}$$

where D is the diameter of the tube. Expressing the equation for the general solution in terms of the given geometry and parameters, we have:

$$\frac{T_w - T(x)}{T_w - T_{in}} = \exp\left(-\frac{\frac{\mathrm{Nu}k}{D}\pi D}{\rho u\pi\frac{D^2}{4}c_p}x_L\right) = \exp\left(-\frac{4\mathrm{Nu}\alpha}{uD^2}x_L\right)$$

Noting that $uD/\alpha = \mathrm{RePr}$, where $\mathrm{Re} = uD/\nu$, $\mathrm{Pr} = \nu/\alpha$, and ν is the kinematic viscosity, we have:

$$\frac{T_w - T(x)}{T_w - T_{in}} = \exp\left(-\frac{4\mathrm{Nu}}{\mathrm{RePr}}\frac{x_L}{D}\right)$$

If the left side of the equation equals 0.05, then:

$$\frac{4\mathrm{Nu}}{\mathrm{RePr}}\frac{x_L}{D} = 3$$

$$\frac{x_L}{D} = \frac{3}{4}\frac{\mathrm{RePr}}{\mathrm{Nu}}$$

Then, if we use the restriction $\Delta x \le 2\alpha/u$, we can find the number of control volumes required to discretize the domain:

$$N_{CV} \ge \frac{x_L}{\Delta x} = \frac{\frac{3}{4}\frac{\mathrm{RePr}}{\mathrm{Nu}}D}{\frac{2\alpha}{u}} = \frac{3}{8}\frac{\mathrm{Re}^2\mathrm{Pr}^2}{\mathrm{Nu}}$$

If we take $\mathrm{Nu} = 5, \mathrm{Re} = 1000, \mathrm{Pr} = 1$, then $N_{CV} \approx 10^5$. If Pr were to increase to 10, then $N_{CV} \approx 10^7$. In this case, the solution at each timestep would require the solution of 10^7 equations, which is impractical for a simple one-dimensional problem.

To calculate the minimum number of timesteps required, let us calculate the length of time it takes for a parcel of fluid entering the duct to make its way to the exit. In reality, even more timeteps than this would be required to reach a steady-state solution. If the length of time to traverse the duct is x_L/u, and the timestep size is based on the timestep restriction derived previously, then the number of timesteps is:

$$N_t = \frac{x_L/u}{\Delta t} = \frac{\frac{3}{4}\frac{\mathrm{RePr}}{\mathrm{Nu}}\frac{D}{u}}{\frac{1}{2}\frac{\Delta x^2}{\alpha}} = \frac{\frac{3}{4}\frac{\mathrm{RePr}}{\mathrm{Nu}}\frac{D}{u}}{\frac{1}{2}\frac{\frac{4\alpha^2}{u^2}}{\alpha}} = \frac{3}{8}\frac{\mathrm{RePr}}{\mathrm{Nu}}\frac{Du}{\alpha} = \frac{3}{8}\frac{\mathrm{Re}^2\mathrm{Pr}^2}{\mathrm{Nu}}$$

Again, assuming $\mathrm{Nu} = 5, \mathrm{Re} = 1000, \mathrm{Pr} = 1$, then the number of timesteps is around 10^5. If Pr is increase to 10, then the number of timesteps increases to 10^7. Again, this is quite an impractical situation.

Now, we must ask ourselves what created the restrictions discussed above on both the timestep and spatial resolution.

Discussion of the Restrictions on Timestep

Since we have already seen that the use of an explicit time integration scheme results in a timestep restriction, let us consider a first-order implicit time integration scheme, which results in the discrete equation:

$$\left(\frac{\rho c_p V_P}{\Delta t} + D_e + D_w + c_p\dot{m}_w - c_p\dot{m}_e\right)T_P - \left(D_e - \frac{c_p\dot{m}_e}{2}\right)T_E - \left(D_w + \frac{c_p\dot{m}_w}{2}\right)T_W - \frac{\rho c_p V_P}{\Delta t}T_P^o = 0$$

In this case there is no timestep restriction since the coefficient on T_P cannot become negative, assuming mass is reasonably well conserved. There is still a restriction on grid resolution to keep the coefficients on T_E and T_W negative. This can be expressed as:

$$D_e - \frac{c_p\dot{m}_e}{2} \le 0 \rightarrow \frac{u\Delta x}{\alpha} \le 2$$

So, while there is no formal restriction on the timestep size, the number of control volumes is still large, since the previous analysis still applies in this regard.

Discussion of the Restrictions on Spatial Resolution

Having resolved the timestep restriction by moving to an implicit time integration scheme, we conclude that the restriction on the grid size must result from the interpolation method chosen for the integration point temperatures in the advection term.

Let us consider again the flow in a duct, where the analytical solution is known and can be applied between the P and E locations. The solution for the temperature profile between the cell centres is:

$$\frac{T - T_P}{T_E - T_P} = \frac{\exp\left[\mathrm{Pe}_\Delta\left(\frac{x - x_P}{x_E - x_P}\right)\right] - 1}{\exp(\mathrm{Pe}_\Delta) - 1}$$

where Pe_Δ is the Péclet number, which represents the ratio of convection to diffusion.

$$\mathrm{Pe}_\Delta = \frac{u\Delta x}{\alpha} = \mathrm{Re}_\Delta\mathrm{Pr}$$

Based on the magnitude of Pe_Δ we can identify different flow regimes:

- $\mathrm{Pe}_\Delta \approx 0$: diffusion dominated
- $|\mathrm{Pe}_\Delta| \approx 1$: convection and diffusion
- $|\mathrm{Pe}_\Delta| \gg 1$: convection dominated

Below, we visualize the solution for various values of Pe_Δ, where negative values correspond to flow in the negative direction.

```python
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# Assign the Pe values to be plotted (note that zero will result in divide by zero)
Pe_vals = [-50, -5, 1e-6, 5, 50]

# Assign the x values (assume normalized by dx)
x = np.linspace(0,1)

# Assign arbitrary values for TP and TE
TP = 1
TE = 0

for Pe in Pe_vals:
    T = (TE - TP)*(np.exp(Pe*(x-x[0])/(x[-1]-x[0])) - 1)/(np.exp(Pe) - 1) + TP
    plt.plot(x, T, label=str(int(Pe)))

plt.xlabel(r"$x/\Delta x$")
plt.ylabel(r"$T$")
plt.legend()
plt.show()
```

It can be seen from the figure above that the assumption of a piecewise linear temperature profile, using the central difference scheme (CDS), is only valid for $\mathrm{Pe}_\Delta \approx 0$. In any practical situation Pe_Δ will be large. Therefore, we need to consider a differnt interpolation scheme.

The Upwind Difference Scheme (UDS)

We now introduce a new interpolation scheme, called the upwind difference scheme (UDS), which is given for the east integration point as:

$$T_e = \frac{1+\alpha_e}{2}T_P + \frac{1-\alpha_e}{2}T_E$$

where α_e is a weighting factor (not to be confused with α, the thermal diffusivity). Ideally we would have:

- $\mathrm{Pe}_\Delta \approx 0; \alpha_e = 0;$ CDS is recovered
- $\mathrm{Pe}_\Delta \rightarrow \infty; \alpha_e = 1; T_e = T_P$
- $\mathrm{Pe}_\Delta \rightarrow -\infty; \alpha_e = -1; T_e = T_E$

It can be seen that this is consistent with what is being observed in the plot above. It is called an "upwind" scheme because the integration point value depends on which way the fluid is flowing. In this case, the discrete equation, in terms of the cell residual, becomes:

$$r_P = \left(\frac{\rho c_p V_P}{\Delta t} + D_e + D_w + \frac{1}{2}c_p\dot{m}_w\left(1+\alpha_w\right) - \frac{1}{2}c_p\dot{m}_e\left(1-\alpha_e\right)\right)T_P - \left[D_e - \frac{1}{2}c_p\dot{m}_e\left(1-\alpha_e\right)\right]T_E - \left[D_w + \frac{1}{2}c_p\dot{m}_w\left(1+\alpha_w\right)\right]T_W - \frac{\rho c_p V_P}{\Delta t}T_P^o$$

The linearization coefficients are then:

$$a_W = -D_w - \frac{1}{2}c_p\dot{m}_w\left(1+\alpha_w\right)$$

$$a_E = -D_e + \frac{1}{2}c_p\dot{m}_e\left(1-\alpha_e\right)$$

$$a_P = \frac{\rho c_p V_P}{\Delta t} - a_W - a_E$$

For the case of a fast-flowing fluid in the postive direction, $\alpha_w = \alpha_e = 1$. The linearization coefficients on the west and east cells are then:

$$a_W = -D_w - c_p\dot{m}_w$$

$$a_E = -D_e$$

These values cannot become positive (since the fluid is flowing in the positive direction and there $\dot{m}_w$ is positive).

For the case of a fast-flowing fluid in the negative direction, $\alpha_w = \alpha_e = -1$. The linearization coefficients are then:

$$a_W = -D_w$$

$$a_E = -D_e + c_p\dot{m}_e$$

Again, these values cannot become positive (since $\dot{m}_e$ is negative in this case). It can also be confirmed that for the case of pure diffusion, the coefficients cannot take on the incorrect sign.

Therefore, using UDS ensures that the solution is stable for any Δx. Combining UDS with an implicit time integration scheme means that there are no formal restrictions on timestep or grid size. It is then up to the analyst to choose values that give sufficient accuracy and attain a solution in an efficient manner.

False Diffusion

While it may seem like UDS has solved all of our problems, it turns out that it is not very accurate due to the fact that it is only a first-order scheme when $\alpha_e = \pm 1$ (since the interpolation only involves one cell value). It is still useful for its stability properties, but typically requires some improvements (to be discussed later). But first, let us try to estimate the accuracy of UDS in comparison to CDS using a Taylor sieries analysis about the east face integration point, based on the diagram below.

FalseDiffusionTaylorDomain

Expanding about this point results in the following estimates of the cell values:

$$T_E = T_e + \frac{\Delta}{2}\left.\frac{dT}{dx}\right|_e + \frac{(\Delta/2)^2}{2}\left.\frac{d^2T}{dx^2}\right|_e + \ldots$$

$$T_P = T_e - \frac{\Delta}{2}\left.\frac{dT}{dx}\right|_e + \frac{(\Delta/2)^2}{2}\left.\frac{d^2T}{dx^2}\right|_e - \ldots$$

Using the CDS interpolation, the value of T_e can be represented by substituting the above estimates:

$$T_e^{CDS} = \frac{T_P + T_E}{2} = T_e + \frac{(\Delta/2)^2}{2}\left.\frac{d^2T}{dx^2}\right|_e + O(\Delta^4)$$

The final term, $O(\Delta^4)$ represents the fact that all terms with odd powers of Δ will cancel, so the magnitude of the first omitted term is proportional to Δ^4.

For UDS (assuming flow in the positive direction), a similar exercise results in:

$$T_e^{UDS} = T_P = T_e - \frac{\Delta}{2}\left.\frac{dT}{dx}\right|_e + O(\Delta^2)$$

For each of the schemes, the error is estimated based on the first truncated term. For both schemes, the leading term is T_e (the value being computed), and the next term will be truncated.

For CDS, the magnitude of the error is estimated as:

$$e^{CDS} \sim \dot{m}c_p\frac{(\Delta/2)^2}{2}\left.\frac{d^2T}{dx^2}\right|_e \sim O(\Delta^2)$$

For UDS, the magnitude of the error is estimated as:

$$e^{UDS} \sim -\dot{m}c_p\frac{\Delta}{2}\left.\frac{dT}{dx}\right|_e \sim O(\Delta)$$

Note that the error estimate inlcudes the $\dot{m}c_p$ term, since the integration point temperatures are multiplied by this value in the energy equation, so this gives a full estimate of the error in this term, not just the error in the interpolated value. On this basis, it can be said that CDS is second-order accurate in space, while UDS is only first order accurate. The interpretation of this is that halving the grid size will result in a reduction of error by a factor of two for UDS, but a factor of 4 for CDS.

The first order error term for UDS is proportional to the temperature gradient, making it appear very much like a diffusion term. In this way, we may call this error "false diffusion", i.e.

$$e^{UDS} = -\dot{m}c_p\frac{\Delta}{2}\left.\frac{dT}{dx}\right|_e = -\frac{\rho c_p u_e A_e\Delta}{2}\left.\frac{dT}{dx}\right|_e = -\Gamma^{false}\left.\frac{dT}{dx}\right|_e A_e$$

where

$$\Gamma^{false} = \frac{\rho c_p u_e\Delta}{2}$$

Let us now take the ratio of Γ^{false} to $\Gamma^{real} = k$

$$\frac{\Gamma^{false}}{\Gamma^{real}} = \frac{\rho c_p u\Delta}{2k} = \frac{1}{2}\frac{u\Delta}{\nu}\frac{\nu\rho c_p}{k} = \frac{1}{2}\frac{u\Delta}{\nu}\frac{\nu}{\alpha} = \frac{1}{2}\mathrm{Re}_\Delta\mathrm{Pr} = \frac{1}{2}\mathrm{Pe}_\Delta$$

Therefore, for large Pe, false diffusion appears to completely dominate over real diffusion. This would be a very bad thing, since we would be unable to model real diffusion. However, the situation is not quite as bad as it seems. In our analysis, we have made the assumption that the leading term in the Taylor series is a good estimate of the error. However, for such convection problems, that may not always be the case. To test this, let us consider again the exact solution between the points P and E:

$$\frac{T - T_P}{T_E - T_P} = \frac{\exp(\mathrm{Pe}\,x^*) - 1}{\exp(\mathrm{Pe}) - 1}$$

where:

$$\mathrm{Pe} = \mathrm{Pe}_\Delta$$

$$x^* = \left(\frac{x - x_P}{x_E - x_P}\right)$$

Another way to express this is:

$$T - T_P = \left(T_E - T_P\right)\frac{\exp(\mathrm{Pe}\,x^*) - 1}{\exp(\mathrm{Pe}) - 1} = A\left[\exp(\mathrm{Pe}\,x^*) - 1\right]$$

The first derivative term in the Taylor series, for this particular solution is:

$$\left.\frac{dT}{dx}\right|_e = \left.\frac{dT}{dx^*}\right|_e\frac{dx^*}{dx}$$

where

$$\frac{dx^*}{dx} = \frac{1}{x_E - x_P} = \frac{1}{\Delta}$$

and

$$\left.\frac{dT}{dx^*}\right|_e = A\mathrm{Pe}\exp(\mathrm{Pe}\,x^*) = A\mathrm{Pe}\exp\left(\frac{\mathrm{Pe}}{2}\right)$$

since $x^* = 1/2$ when $x = x_e$. Combining the above expressions together results in:

$$\left.\frac{dT}{dx}\right|_e = \frac{A\mathrm{Pe}}{\Delta}\exp\left(\frac{\mathrm{Pe}}{2}\right)$$

Following the same procedure, the higher order derivatives can be computed as:

$$\left.\frac{d^2T}{dx^2}\right|_e = \frac{A\mathrm{Pe}^2}{\Delta^2}\exp\left(\frac{\mathrm{Pe}}{2}\right)$$

$$\left.\frac{d^3T}{dx^3}\right|_e = \frac{A\mathrm{Pe}^3}{\Delta^3}\exp\left(\frac{\mathrm{Pe}}{2}\right)$$

and so forth. Substituting these into the Taylor series for T_P:

$$\begin{aligned}T_P &= T_e - \frac{\Delta}{2}\left.\frac{dT}{dx}\right|_e + \frac{(\Delta/2)^2}{2}\left.\frac{d^2T}{dx^2}\right|_e - \frac{(\Delta/2)^3}{6}\left.\frac{d^2T}{dx^2}\right|_e \\ &= T_e - \frac{\Delta}{2}\frac{A\mathrm{Pe}}{\Delta}\exp\left(\frac{\mathrm{Pe}}{2}\right) + \frac{(\Delta/2)^2}{2}\frac{A\mathrm{Pe}^2}{\Delta^2}\exp\left(\frac{\mathrm{Pe}}{2}\right) - \frac{(\Delta/2)^3}{6}\frac{A\mathrm{Pe}^3}{\Delta^3}\exp\left(\frac{\mathrm{Pe}}{2}\right) \\ &= T_e - \frac{A\mathrm{Pe}}{2}\exp\left(\frac{\mathrm{Pe}}{2}\right) + \frac{A\mathrm{Pe}^2}{8}\exp\left(\frac{\mathrm{Pe}}{2}\right) - \frac{A\mathrm{Pe}^3}{48}\exp\left(\frac{\mathrm{Pe}}{2}\right) \\ &= T_e - \frac{A\mathrm{Pe}\exp\left(\frac{\mathrm{Pe}}{2}\right)}{2}\left[1 - \frac{\mathrm{Pe}}{4} + \frac{\mathrm{Pe}}{24}\right]\end{aligned}$$

Now, recall that the first term in the square brackets was considered representative of the error for UDS.

If we let:

$$S = \left[1 - \frac{\mathrm{Pe}}{4} + \frac{\mathrm{Pe}}{24} - \ldots\right]$$

then:

- For $\mathrm{Pe} = 0.01$, $S = 1 - 0.0025 + 0.000004 - \ldots$
- For $\mathrm{Pe} = 1$, $S = 1 - 0.025 + 0.0416 - \ldots$
- For $\mathrm{Pe} = 100$, $S = 1 - 25 + 416.6 - \ldots$
- For $\mathrm{Pe} = 1000$, $S = 1 - 250 + 4166.6 - \ldots$

Therefore, it is clear that the series only converges for $\mathrm{Pe} \lesssim 1$ (for this particular problem), so the first term is really only representative of the error in this case. If the profile being approximated is close to being linear, then the series converges and gives us a good error estimate (and also a good estimate of false diffusion). If the profile is highly non-linear (as in the example above) the Taylor series does not give us any useful information. Therefore, the false diffusion induced by UDS is not as bad as it may have looked for high Pe. However, UDS is a first-order scheme and its accuracy is therefore limited. As a result we need to look for ways to improve the accuracy of UDS while preserving its stability properties.

Improvements to the Advection Scheme

Power Law Scheme

One way of improving the UDS scheme is by appropriately selecting the weighting coefficients, α_e such that the linearization coefficients cannot take on the incorrect sign. Based on the generalized UDS expression

$$T_e = \frac{1+\alpha_e}{2}T_P + \frac{1-\alpha_e}{2}T_E$$

the Power Law Scheme selects α_e as:

$$\alpha_e = \frac{\mathrm{Pe}^2}{5 + \mathrm{Pe}^2}$$

For $\mathrm{Pe} \approx 1$, $\alpha_e \approx 1/2$, so the scheme is second order accurate. For large Pe, the scheme approaches UDS and the scheme can only be considered first order accurate. Therefore, the power law scheme can only be considered as a partial solution. We will not use the power law scheme.

Deferred Correction Approach

An alternative approach for improving an advection scheme is based on the idea of a "deferred correction" where UDS is used as the main advection scheme and linearized accordingly. However, the UDS terms are then also subtracted from the discretized equation and the terms for a higher order scheme are added explicitly. The subtracted UDS terms and the higher order terms are not linearized. Therefore, the linearization maintains the stability of UDS, but the accuracy of the higher order scheme. Once the non-linear iteration is converged, only the higher order terms remain.

Using this idea, the advective flux through the east face of a control volume can be written as:

$$F_e = F_e^{UDS} + \left(F_e^{HOS} - F_e^{UDS}\right)$$

where F_e^{UDS} and F_e^{HOS} refer to the flux computed by UDS and the higher order scheme, respectively. As mentioned, linearization is only carried out on the first term in the equation above, since this will guarantee stability of the numerical method. This does, however, make the linearization inexact (since it is based on the UDS scheme not the higher order scheme). Therefore, iteration is required to arrive at the solution.

Central Difference Scheme (CDS)

The CDS scheme, described earlier, can be used to improve the accuracy of the advection discretization, provided it is implmented using the deferred correction approach. Recall that the CDS scheme is given as:

$$T_e = \frac{1}{2}\left(T_P + T_E\right)$$

Quadratic Upwind Interpolation for Convective Kinematics (QUICK)

The QUICK scheme is derived by passing a parabola through cell values, biased towards the upwind direction. For flow in the positive direction, interpolations for the east integration point will involve cells W, P, and E. For flow in the negative direction, interpolations will involve cells P, E, and EE.

The resulting general expression for $T(x)$ is:

$$T(x) = \frac{(x - x_P)(x - x_E)}{(x_W - x_P)(x_W - x_E)}T_W + \frac{(x - x_W)(x - x_E)}{(x_P - x_W)(x_P - x_E)}T_P + \frac{(x - x_W)(x - x_P)}{(x_E - x_W)(x_E - x_P)}T_E$$

For a uniform grid with spacing Δ:

$$T_e = -\frac{1}{8}T_W + \frac{3}{4}T_P + \frac{3}{8}T_E$$

$$T_w = -\frac{1}{8}T_{WW} + \frac{3}{4}T_W + \frac{3}{8}T_P$$

Note that due to the negative sign in the leading term, this scheme could result in linearization coefficients taking on the wrong sign if implemented directly. When implemented using deferred corrections, the QUICK scheme can be an effective higher order scheme.

Exercise: Derive the expressions for the QUICK scheme where the flow is in the negative direction.

Implementation

The code below demonstrates the implementation of the pure UDS advection scheme using a class called `UpwindAdvectionModel`. The structure of the class is based on the `DiffusionModelClass`.

```python
import numpy as np

class UpwindAdvectionModel:
    """Class defining an upwind advection model"""

    def __init__(self, grid, phi, Uhe, rho, cp, west_bc, east_bc):
        """Constructor"""
        self._grid = grid
        self._phi = phi
        self._Uhe = Uhe
        self._rho = rho
        self._cp = cp
        self._west_bc = west_bc
        self._east_bc = east_bc
        self._alphae = np.zeros(self._grid.ncv+1)
        self._phie = np.zeros(self._grid.ncv+1)

    def add(self, coeffs):
        """Function to add diffusion terms to coefficient arrays"""

        # Calculate the weighting factors
        for i in range(self._grid.ncv+1):
            if self._Uhe[i] >= 0:
                self._alphae[i] = 1
            else:
                self._alphae[i] = -1

        # Calculate the east integration point values (including both boundaries)
        self._phie = (1 + self._alphae)/2*self._phi[0:-1] + (1 - self._alphae)/2*self._phi[1:]

        # Calculate the face mass fluxes
        mdote = self._rho*self._Uhe*self._grid.Af

        # Calculate the west and east face advection flux terms for each face
        flux_w = self._cp*mdote[:-1]*self._phie[:-1]
        flux_e = self._cp*mdote[1:]*self._phie[1:]

        # Calculate mass imbalance term
        imbalance = - self._cp*mdote[1:]*self._phi[1:-1] + self._cp*mdote[:-1]*self._phi[1:-1]

        # Calculate the linearization coefficients
        coeffW = - self._cp*mdote[:-1]*(1 + self._alphae[:-1])/2
        coeffE = self._cp*mdote[1:]*(1 - self._alphae[1:])/2
        coeffP = - coeffW - coeffE

        # Modify the linearization coefficients on the boundaries
        coeffP[0] += coeffW[0]*self._west_bc.coeff()
        coeffP[-1] += coeffE[-1]*self._east_bc.coeff()

        # Zero the boundary coefficients that are not used
        coeffW[0] = 0.0
        coeffE[-1] = 0.0

        # Calculate the net flux from each cell
        flux = flux_e - flux_w

        # Add to coefficient arrays
        coeffs.accumulate_aP(coeffP)
        coeffs.accumulate_aW(coeffW)
        coeffs.accumulate_aE(coeffE)
        coeffs.accumulate_rP(flux)
        coeffs.accumulate_rP(imbalance)

        # Return the modified coefficient array
        return coeffs
```

Advection-Diffusion Problem with External Convection

The code below solves the advection-diffusion equation for flow in a square duct with an external heat flux speficied by a heat transfer coefficient and ambient temperature.

Here we have defined a new variable called `Uhe`, which stores the velocity used to calculate the mass fluxes through the faces. In this case, we just set it equal to a constant value, since the flow is incompressible and the duct has a constant cross-sectional area. This approach will be used further in the next lesson, and will become more clear once we begin solving coupled mass and momentum equations.

In [ ]:

```python
from Lesson4.Grid import Grid
from Lesson4.ScalarCoeffs import ScalarCoeffs
from Lesson4.BoundaryConditions import BoundaryLocation, DirichletBc, NeumannBc
from Lesson4.Models import DiffusionModel, SurfaceConvectionModel, FirstOrderTransientModel
from Lesson4.LinearSolver import solve

import numpy as np
from numpy.linalg import norm

# Define the grid
lx = 1.0
ly = 0.1
lz = 0.1
ncv = 50
grid = Grid(lx, ly, lz, ncv)

# Set the timestep information
nTime = 1
dt = 1e9
time = 0

# Set the maximum number of iterations and convergence criterion
maxIter = 10
converged = 1e-6

# Define thermophysical properties
rho = 1000
cp = 4000
k = 0.5

# Define the surface convection parameters
ho = 50
To = 200

# Define the coefficients
coeffs = ScalarCoeffs(grid.ncv)

# Initial conditions
T0 = 300
U0 = 0.01

# Initialize field variable arrays
T = T0*np.ones(grid.ncv+2)
Uhe = U0*np.ones(grid.ncv+1)

# Define boundary conditions
west_bc = DirichletBc(T, grid, 400, BoundaryLocation.WEST)
east_bc = NeumannBc(T, grid, 0, BoundaryLocation.EAST)

# Apply boundary conditions
west_bc.apply()
east_bc.apply()

# Define the transient model
Told = np.copy(T)
transient = FirstOrderTransientModel(grid, T, Told, rho, cp, dt)

# Define the diffusion model
diffusion = DiffusionModel(grid, T, k, west_bc, east_bc)

# Define the surface convection model
surfaceConvection = SurfaceConvectionModel(grid, T, ho, To)

# Define the advection model
advection = UpwindAdvectionModel(grid, T, Uhe, rho, cp, west_bc, east_bc)

# Loop through all timesteps
for tStep in range(nTime):
    # Update the time information
    time += dt

    # Print the timestep information
    print("Timestep = {}; Time = {}".format(tStep, time))

    # Store the "old" temperature field
    Told[:] = T[:]

    # Iterate until the solution is converged
    for i in range(maxIter):
        # Zero the coefficients and add each influence
        coeffs.zero()
        coeffs = diffusion.add(coeffs)
        coeffs = surfaceConvection.add(coeffs)
        coeffs = advection.add(coeffs)
        coeffs = transient.add(coeffs)

        # Compute residual and check for convergence
        maxResid = norm(coeffs.rP, np.inf)
        avgResid = np.mean(np.absolute(coeffs.rP))
        print("Iteration = {}; Max. Resid. = {}; Avg. Resid. = {}".format(i, maxResid, avgResid))
        if maxResid < converged:
            break

        # Solve the sparse matrix system
        dT = solve(coeffs)

        # Update the solution and boundary conditions
        T[1:-1] += dT
        west_bc.apply()
        east_bc.apply()
```

In [ ]:

```python
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(grid.xP, T)

plt.xlabel("x")
plt.ylabel("T")
plt.show()
```

Exercise: Explore the effect of the mass flow rate in the problem above. How is the solution different when there is no mass flow?

Now that you have completed this lesson on convection of a scalar, you are ready to move on the next lesson on Solution of Mass and Momentum Equations where we will implement more code to solve both the mass and momentum equations.