Lecture 8

Finite Difference Methods for Convection-Diffusion

In this lecture, we introduce the finite difference method for the solution of PDE's. We will limit our discussion of PDE's to convection-diffusion. Recall from Equation 7.6 that the convection equation is,

$$\frac{\partial U}{\partial t} + u\frac{\partial U}{\partial x} + v\frac{\partial U}{\partial y} = 0,$$

where U is the scalar quantity which is convected. Adding diffusion to this equation gives the convection-diffusion equation,

$$\frac{\partial U}{\partial t} + u\frac{\partial U}{\partial x} + v\frac{\partial U}{\partial y} = \mu\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right), \tag{8.1}$$

where μ is the diffusion coefficient.

8.1 Finite Difference Approximations

The finite difference approximation of a PDE is constructed using a grid over the domain of interest. The finite difference approximations are easiest to derive using a structured, rectangular as shown in Figure 8.1. For a finite difference approximation of a PDE, the solution is sought at the nodes of the mesh. We use the notation that $U_{i,j}$ is the value of U at the (i,j) node.

A common finite difference approximation of $\partial U/\partial x$ at node (i,j) is,

$$\left.\frac{\partial U}{\partial x}\right|_{i,j} \approx \delta_{2x}U_{i,j} \equiv \frac{1}{2\Delta x}\left(U_{i+1,j} - U_{i-1,j}\right). \tag{8.2}$$

The finite difference operator δ_{2x} is called a central difference operator. Finite difference approximations can also be one-sided. For example, a backward difference approximation is,

$$\left.\frac{\partial U}{\partial x}\right|_{i,j} \approx \delta_x^- U_{i,j} \equiv \frac{1}{\Delta x}\left(U_{i,j} - U_{i-1,j}\right), \tag{8.3}$$

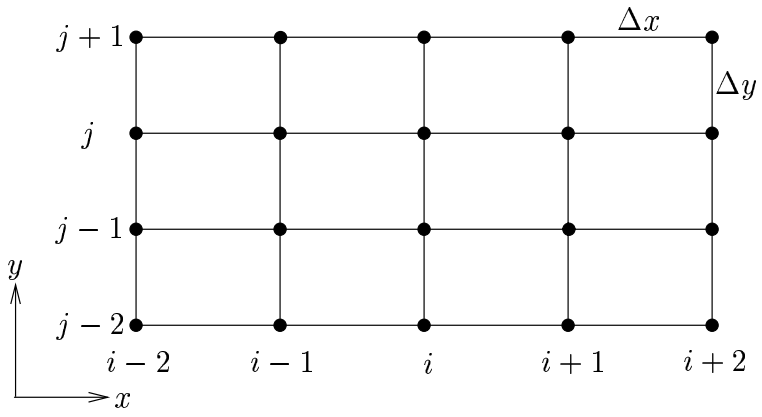

Figure 8.1: Two-dimensional structured mesh for finite difference approximations.

and a forward difference approximation is,

$$\left.\frac{\partial U}{\partial x}\right|_{i,j} \approx \delta_x^+ U_{i,j} \equiv \frac{1}{\Delta x}\left(U_{i+1,j} - U_{i,j}\right), \tag{8.4}$$

A very common finite difference approximation for a second derivative is,

$$\left.\frac{\partial^2 U}{\partial x^2}\right|_{i,j} \approx \delta_x^2 U_{i,j} \equiv \frac{1}{\Delta x^2}\left(U_{i+1,j} - 2U_{i,j} + U_{i-1,j}\right). \tag{8.5}$$

As in the first derivative case, since this derivative approximation uses both $i \pm 1$ values, it is known as a central difference approximation of the second derivative. This approximation can be motivated by approximating the second derivatives as a difference of first derivatives,

$$\begin{aligned}
\frac{\partial^2 U}{\partial x^2}(x) &= \lim_{\Delta x \to 0} \frac{\partial U/\partial x(x + \Delta x/2) - \partial U/\partial x(x - \Delta x/2)}{\Delta x} \\
&= \lim_{\Delta x \to 0} \frac{1}{\Delta x}\left[\frac{U(x+\Delta x) - U(x)}{\Delta x} - \frac{U(x) - U(x-\Delta x)}{\Delta x}\right] \\
&= \lim_{\Delta x \to 0} \frac{1}{\Delta x^2}\left[U(x+\Delta x) - 2U(x) + U(x-\Delta x)\right].
\end{aligned}$$

To approximate the convection-diffusion equation we can combine various finite difference derivative approximations. For example, consider the one-dimensional convection-diffusion equation,

$$\frac{\partial U}{\partial t} + u\frac{\partial U}{\partial x} = \mu\frac{\partial^2 U}{\partial x^2}.$$

Approximating this using central differences for all derivatives, the convection-diffusion equation at node i can be approximated as,

$$\frac{dU_i}{dt} + u_i \delta_{2x} U_i = \mu \delta_x^2 U_i. \tag{8.6}$$

This is an ordinary differential equation for U_i which is coupled to the nodal values at $U_{i\pm1}$. To make this a fully discrete approximation, we need to discretize in time. To do this, we could apply any of the ODE integration methods that we discussed previously. For example, the simple forward Euler integration method would give,

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + u_i^n \delta_{2x} U_i^n = \mu \delta_x^2 U_i^n. \tag{8.7}$$

In finite differences, we often make use of the spatial stencil of a discretization. The spatial stencil is simply the subset of nodes which are used to discretize the problem in space. The central difference approximation in Equation 8.6 gives a three-point spatial stencil since dU/dt at node i depends on three nodal values of U (including itself), specifically U_{i-1}, U_i, and U_{i+1}. To illustrate this, note that Equation (8.6) can be written as,

$$\frac{dU_i}{dt} = a_{i-1}U_{i-1} + a_i U_i + a_{i+1}U_{i+1}$$

where

$$\begin{aligned} a_{i-1} &= \frac{u_i}{2\Delta x} + \frac{\mu}{\Delta x^2}, \\ a_i &= -2\frac{\mu}{\Delta x^2}, \\ a_{i+1} &= -\frac{u_i}{2\Delta x} + \frac{\mu}{\Delta x^2}, \end{aligned}$$

We can also place all of the nodal states into a vector,

$$U = (U_1, U_2, \ldots, U_{i-1}, U_i, U_{i+1}, \ldots, U_{N_x-1}, U_{N_x})^T,$$

where N_x is the total number of points in the x-direction. Then, the finite difference approximation of convection-diffusion can be written in the following form,

$$\frac{dU}{dt} = AU + b, \tag{8.8}$$

where b will contain boundary condition related data (boundary conditions are discussed in Section 8.2) and the matrix A is,

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} & \ldots & A_{1,N_x} \\ A_{2,1} & A_{2,2} & A_{2,3} & \ldots & A_{2,N_x} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{Nx,1} & A_{N_x,2} & A_{N_x,3} & \ldots & A_{N_x,N_x} \end{pmatrix}$$

Note that row i of this matrix contains the coefficients of the nodal values for the ODE governing node i. Except for rows requiring boundary condition data, the values of $A_{i,j}$ are related to the coefficients $a_{i\pm1}$ and a_i, specifically,

$$A_{i,i-1} = a_{i-1}, \quad A_{i,i} = a_i, \quad A_{i,i+1} = a_{i+1},$$

and all other entries in row i are zero.

In-class Discussion 8.1 (Finite Volume vs. Finite Difference Discretizations)

8.2 Boundary Conditions

In this section, we discuss the implementation of finite difference methods at boundaries. This discussion is not meant to be comprehensive, as the issues are many and often subtle.

8.2.1 Dirichlet Boundary Conditions

A Dirichlet boundary condition is one in which the state is specified at the boundary. For example, in a heat transfer problem, the temperature might be known at the boundary. Dirichlet boundary conditions can be implemented in a relatively straightforward manner. For example, suppose that we are solving a one-dimensional convection-diffusion problem and we want the value of U at $i = 1$, to be U_{inlet},

$$U_1 = U_{inlet}.$$

To implement this, we fix $U_1 = U_{inlet}$ and apply the finite difference discretization only over the interior of the computational domain accounting for the known value of U_1 at any place where the interior discretization depends on it. For example, at the first interior node (i.e. $i = 2$), the central difference discretization of 1-D convection-diffusion gives,

$$\frac{dU_2}{dt} + u_2 \frac{U_3 - U_1}{2\Delta x} = \mu \frac{U_3 - 2U_2 + U_1}{\Delta x^2}.$$

Accounting for the known value of U_1, this becomes,

$$\frac{dU_2}{dt} + u_2 \frac{U_3 - U_{inlet}}{2\Delta x} = \mu \frac{U_3 - 2U_2 + U_{inlet}}{\Delta x^2}. \tag{8.9}$$

In terms of the vector notation, when a Dirichlet boundary condition is applied we usually remove that state from the vector U. So, in the situation where U_1 is known, the state vector is defined as,

$$U = (U_2, U_3, \ldots, U_{i-1}, U_i, U_{i+1}, \ldots, U_{N_x-1}, U_{N_x})^T,$$

The b vector then will contain the contributions from the known boundary values. For example, by re-arranging Equation (8.9), the first row of b contains,

$$b_1 = u_2 \frac{U_{inlet}}{2\Delta x} + \mu \frac{U_{inlet}}{\Delta x^2}.$$

Since U_{inlet} does not enter any of the other node's stencils, the remaining rows of b will be zero (unless the are altered by the other boundary). Note, the first row of A is,

$$A_{1,1} = -2\frac{\mu}{\Delta x^2}, \qquad A_{1,2} = -\frac{u_2}{2\Delta x} + \frac{\mu}{\Delta x^2},$$

and $A_{1,j} = 0$ for $j > 2$.

8.2.2 Neumann Boundary Conditions

Notes will be coming on this issue.

Example 8.1 (Finite Difference Method applied to 1-D Convection) *In this example, we apply the central difference approximation to solve 1-D convection, i.e.,*

$$\frac{\partial U}{\partial t} + u\frac{\partial U}{\partial x} = 0.$$

We will solve a problem that is nearly the same as that in Example 7.4. Specifically, the initial condition is

$$U_0(x) = e^{-x^2}.$$

We let the velocity, $u = 1$. Instead of solving a problem with periodicity, we enforce the inflow boundary (which is at $x = x_L$ since $u > 0$),

$$U(t, x_L) = e^{-x_L^2}.$$

Physically, since $u > 0$, no boundary condition is needed at $x = x_R$ (i.e. the scalar just convects out the domain at this location). But, the central difference discretization cannot be applied at the outlet node (i.e. at $i = N_x$) because it would require the $N_x + 1$ nodal value (which does not exist). To handle this problem, we switch the discretization at $i = N_x$ to a one-sided backwards difference formula.

```
% This Matlab script solves the one-dimensional convection
% equation using a finite difference algorithm.  The
% discretization uses central differences in space and forward
% Euler in time.  An inflow bc is set and at the outflow a
% one-sided (backwards) difference is used.  The initial condition
% is a Gaussian distribution.
%

clear all;
```

```
% Set-up grid
xL = -4;
xR =  4;
Nx = 51; % number of control volumes
x = linspace(xL,xR,Nx);

% Calculate cell size in control volumes (assumed equal)
dx = x(2) - x(1);

% Set velocity
u = 1;

% Set final time
tfinal = 100;

% Set timestep
CFL = 0.1;
dt = CFL*dx/abs(u);

% Set initial condition
U = exp(-x.^2);
t = 0;

% Set bc state at left (assumes u>0)
UL = exp(-xL^2);

% Loop until t > tfinal
while (t < tfinal),

  % Copy old state vector
  U0 = U;

  % Inflow boundary
  U(1) = UL;

  % Interior nodes
  for i = 2:Nx-1,
    U(i) = U0(i) - dt*(U0(i+1)-U0(i-1))/(2*dx);
  end

  % Outflow boundary uses backward difference
  i = Nx;
  U(i) = U0(i) - dt*(U0(i)-U0(i-1))/(dx);
```

```
  % Increment time
  t = t + dt;

  % Plot current solution
  plot(x,U,'*');
  xlabel('x'); ylabel('U');
  title(sprintf('time = %f\n',t));
  axis([xL, xR, -0.5, 1.5]);
  grid on;
  drawnow;

end
```

8.3 Truncation Error Analysis

These notes are still to come.