

Chapter 13

Finite Difference Methods

In the previous chapter we developed finite difference approximations for partial derivatives. In this chapter we will use these finite difference approximations to solve partial differential equations (PDEs) arising from conservation law presented in Chapter 11.

48 Self-Assessment

Before reading this chapter, you may wish to review...

- Conservation Laws 11
- Finite Difference Approximations 12

After reading this chapter you should be able to...

- implement a finite difference method to solve a PDE
- compute the order of accuracy of a finite difference method
- develop upwind schemes for hyperbolic equations

Relevant self-assessment exercises: 4 - 6

49 Finite Difference Methods

Consider the one-dimensional convection-diffusion equation,

$$\frac{\partial U}{\partial t} + u \frac{\partial U}{\partial x} - \mu \frac{\partial^2 U}{\partial x^2} = 0. \quad (101)$$

Approximating the spatial derivative using the central difference operators gives the following approximation at node i ,

$$\frac{dU_i}{dt} + u_i \delta_{2x} U_i - \mu \delta_x^2 U_i = 0 \quad (102)$$

This is an ordinary differential equation for U_i which is coupled to the nodal values at $U_{i\pm 1}$. Assembling all of the nodal states into a single vector

$$U = (U_0, U_1, \dots, U_{i-1}, U_i, U_{i+1}, \dots, U_{N_x-1}, U_{N_x})^T,$$

gives a system of coupled ODEs of the form:

$$\frac{dU}{dt} = AU + b \quad (103)$$

where b will contain boundary condition related data (boundary conditions are discussed in Section 49.1). The matrix A has the form:

$$A = \begin{bmatrix} A_{0,0} & A_{0,1} & \dots & A_{0,N_x} \\ A_{1,0} & A_{1,1} & \dots & A_{1,N_x} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N_x,0} & A_{N_x,1} & \dots & A_{N_x,N_x} \end{bmatrix}.$$

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 α_j of the finite difference approximation. Specifically, for our central difference approximation

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

and all other entries in row i are zero. In general, the number of non-zero entries in row i will correspond to the size of the stencil of the finite difference approximations used.

We refer to Equation 103 as being semi-discrete, since we have discretized the PDE in space but not in time. To make this a fully discrete approximation, we could apply any of the ODE integration methods that we discussed previously. For example, the simple forward Euler integration method would give,

$$\frac{U^{n+1} - U^n}{\Delta t} = AU^n + b. \quad (104)$$

Using central difference operators for the spatial derivatives and forward Euler integration gives the method widely known as a Forward Time-Central Space (FTCS) approximation. Since this is an explicit method A does not need to be formed explicitly. Instead we may simply update the solution at node i as:

$$U_i^{n+1} = U_i^n - \frac{1}{\Delta t}(u_i \delta_{2x} U_i^n - \mu \delta_x^2 U_i^n) \quad (105)$$

Example 1. Finite Difference Method applied to 1-D Convection

In this example, we solve the 1-D convection equation,

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

using a central difference spatial approximation with a forward Euler time integration,

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + u_i^n \delta_{2x} U_i^n = 0.$$

Note: this approximation is the Forward Time-Central Space method from Equation 111 with the diffusion terms removed. We will solve a problem that is nearly the same as that in Example 3. Specifically, we use a constant velocity, $u = 1$ and set the initial condition to be

$$U_0(x) = 0.75e^{-(\frac{x-0.5}{0.1})^2}.$$

We consider the domain $\Omega = [0, 1]$ with periodic boundary conditions and we will make use of the central difference approximation developed in Exercise 1. The matlab script which implements this algorithm is:

```

1 % This Matlab script solves the one-dimensional convection
2 % equation using a finite difference algorithm. The
3 % discretization uses central differences in space and forward
4 % Euler in time.
5
6 clear all;
7 close all;
8
9 % Number of points
10 Nx = 50;
11 x = linspace(0,1,Nx+1);
12 dx = 1/Nx;
13
14 % velocity
15 u = 1;
16
17 % Set final time
18 tfinal = 10.0;
19
20 % Set timestep
21 dt = 0.001;
22
23 % Set initial condition
24 Uo = 0.75*exp(-(x-0.5)/0.1).^2);
25 t = 0;
26
27 U = Uo;
28
29 % Loop until t > tfinal
30 while (t < tfinal),
31     % Forward Euler step
32     U(2:end) = U(2:end) - dt*u*centraldiff(U(2:end));
33     U(1) = U(end); % enforce periodicity
34
35     % Increment time
36     t = t + dt;
37
38     % Plot current solution
39     clf
40     plot(x,Uo,'b*');
41     hold on;
42     plot(x,U,'*','color',[0 0.5 0]);
43     xlabel('x','fontsize',16); ylabel('U','fontsize',16);
44     title(sprintf('t = %f\n',t));
45     axis([0, 1, -0.5, 1.5]);
46     grid on;
47     drawnow;
48 end

```

Figure 20 plots the finite difference solution at time $t = 0.25$, $t = 0.5$, and $t = 1.0$. The exact solution for this problem has $U(x, t) = U_o(x)$ for any integer time ($t = 1, 2, \dots$). When the numerical method is run, the Gaussian disturbance is convected across the domain, however small oscillations are observed at $t = 0.5$ which begin to pollute the numerical solution. Eventually, these oscillations grow until the entire solution is contaminated. In Chapter 14 we will show that the FTCS algorithm is unstable for any Δt for pure convection. Thus, what we are observing is an instability that can be predicted through some analysis.

Exercise 1. Download the matlab code from Example 1 and modify the code to use the backward difference formula δ_x^- . This method known, as the Forward Time-Backward Space (FTBS) method. Using the same $u = 1$, $\Delta t = \frac{1}{1000}$ and $\Delta x = \frac{1}{50}$ does the FTBS method exhibit the same instability as the FTCS method?

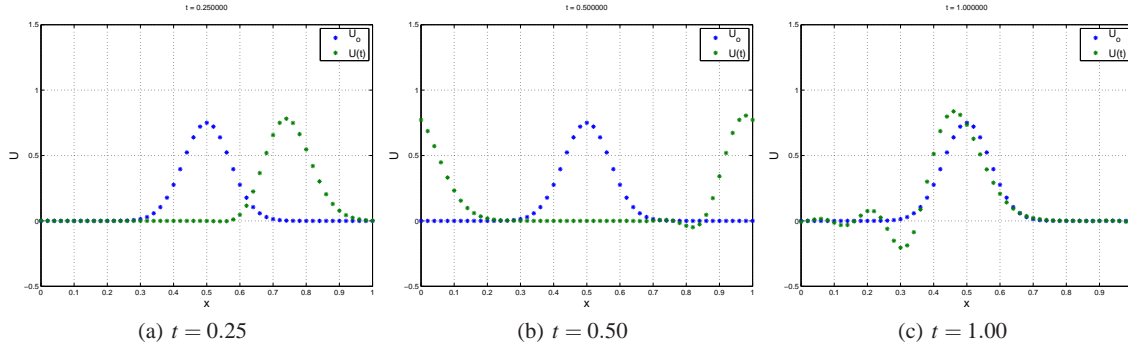


Fig. 20 Forward Time-Central Space method for 1-D convection.

- (a) Yes
- (b) No
- (c) Sometimes
- (d) I don't know

49.1 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. In particular, we only focus on 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 may be known at the domain boundaries. 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 $i = 0$, to be U_{inlet} ,

$$U_0 = U_{inlet}.$$

To implement this, we fix $U_0 = U_{inlet}$ and apply the finite difference discretization only over the interior of the computational domain accounting for the known value of U_0 at any place where the interior discretization depends on it. For example, at the first interior node (i.e. $i = 1$), the central difference discretization of for the 1-D convection-diffusion equation gives,

$$\frac{dU_1}{dt} + u_1 \frac{U_2 - U_0}{2\Delta x} = \mu \frac{U_2 - 2U_1 + U_0}{\Delta x^2}.$$

Accounting for the known value of U_0 , this becomes,

$$\frac{dU_1}{dt} + u_1 \frac{U_2 - U_{inlet}}{2\Delta x} = \mu \frac{U_2 - 2U_1 + U_{inlet}}{\Delta x^2}. \quad (106)$$

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_0 is known, the state vector is defined as,

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

The b vector then will contain the contributions from the known boundary values. For example, by re-arranging Equation (106), 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 they are altered by the other boundary).

Exercise 2. Download the matlab code from Example 1 and modify the code to use a Dirichlet boundary condition on the inflow and the backwards difference formula δ_x^- on the outflow. Using the same $u = 1$, $\Delta t = \frac{1}{1000}$ and $\Delta x = \frac{1}{50}$ is the FTCS method with Dirichlet boundary condition stable?

- (a) Yes
- (b) No
- (c) Sometimes
- (d) I don't know

49.2 Truncation Error for a PDE

In the discussion of ODE integration, we used the ideas of consistency and stability to prove convergence through the Dahlquist Equivalence Theorem. Similar concepts also exist for PDE discretizations, however, we cannot cover these here. We will briefly look at the truncation error for a partial differential equation. We have already discussed the local truncation error for finite difference approximation of derivatives in Section 47.1.

Similar to the ODE case, the truncation error is defined as the remainder after an exact solution to the governing equation is substituted into the finite difference approximation. For example, suppose we are using the FTCS algorithm in Equation 111 to approximate the one-dimensional convection-diffusion equation. Then the local truncation error for the PDE approximation is defined as,

$$\tau \equiv \frac{U_i^{n+1} - U_i^n}{\Delta t} + u_i^n \delta_{2x} U_i^n - \mu \delta_x^2 U_i^n, \quad (107)$$

where $U(x, t)$ is an exact solution to Equation (94). Note that the truncation error defined here for PDE's is not quite a direct analogy with the standard definition of local truncation error used in ODE integration, specifically in Equation (16). In particular, in the ODE case, the truncation error is defined as the difference between the numerical solution and the exact solution after one step of the method (starting from exact values). However, in the PDE case, we have defined the truncation error as the remainder after an exact solution is substituted into the numerical method when the numerical method is written in a form that approximates the governing PDE.

Except for this difference in the definition, the calculation of the local truncation follows the same procedure as in the ODE case in which Taylor series substitutions are used to expand the error in powers of Δt . Continuing on with our example, we use Taylor series of U about $t = t^n$ and $x = x_i$. However, we can use our previous results from the analysis of the truncation error of spatial derivatives. Specifically, we have

$$\begin{aligned} \delta_{2x} U_i &= U_{xi} + \frac{1}{6} \Delta x^2 U_{xxx} + O(\Delta x^4) \\ \delta_x^2 U_i &= U_{xx} + \frac{1}{12} \Delta x^2 U_{xxxx} + O(\Delta x^4) \end{aligned}$$

Using these results,

$$\tau = \frac{U_i^{n+1} - U_i^n}{\Delta t} + u_i^n \left[U_{xi}^n + \frac{1}{6} \Delta x^2 U_{xxx_i}^n + O(\Delta x^4) \right] - \mu \left[U_{xx_i}^n + \frac{1}{12} \Delta x^2 U_{xxxx_i}^n + O(\Delta x^4) \right]$$

Then, performing a similar Taylor series of the time derivative approximation gives,

$$\frac{U^{n+1} - U^n}{\Delta t} = U_t^n + \frac{1}{2} \Delta t U_{tt}^n + O(\Delta t^2).$$

Substituting this into τ and collecting terms in powers of Δt and Δx gives,

$$\begin{aligned} \tau = & U_{ti}^n + u_i^n U_{xi}^n - \mu U_{xx_i}^n + \\ & \frac{1}{2} \Delta t U_{tti}^n + O(\Delta t^2) + \\ & \frac{1}{6} \Delta x^2 u_i^n U_{xxx_i}^n - \frac{1}{12} \Delta x^2 \mu U_{xxxx_i}^n + O(\Delta x^4) \end{aligned}$$

The first line of this equation is actually just the PDE, evaluated at $t = t^n$ and $x = x_i$. Since U is an exact solution to the PDE, this is zero. The second line shows that the time discretization introduces an $O(\Delta t)$. The third line shows that the spatial discretization introduces an $O(\Delta x^2)$ error. Thus, this numerical method is first-order accurate in time and second-order accurate in space.

Exercise 3. Download the matlab code from Example 1. Verify the this method is indeed second order accurate in space.

50 Upwinding

Consider the one-dimensional convection equation,

$$\frac{\partial U}{\partial t} + u \frac{\partial U}{\partial x} = 0. \quad (108)$$

Recall from Chapter 11 that for the convection equation, the solution $U(x, t)$ is obtained by following the characteristic line back to the initial condition and evaluating $U_o(\xi)$ where $\xi = x - ut$. The convection equation has some inherent directionality as the solution at (x, t) depends only upon previous values of U upstream of x . As the physical problem has inherent directionality, it is natural for our numerical scheme to also have some sort of directional bias.

First consider the FTCS method from example 1. The solution U_i^{n+1} is updated as:

$$U_i^{n+1} = U_i^n - \frac{u}{2\Delta t} (U_{i+1}^n - U_{i-1}^n)$$

This is graphically depicted in Figure 21(a). This numerical scheme has no inherent directionality.

Now consider the Forward Time-Backward Space (FTBS) method from Exercise 4 which uses the backwards difference formula δ_x^- , and gives the update

$$U_i^{n+1} = U_i^n - \frac{u}{\Delta t} (U_i^n - U_{i-1}^n).$$

This method is depicted graphically in Figure 21(b). Notice that the numerical scheme is now biased to upstream values. Numerical schemes which exhibit such an upstream bias are called **upwind** schemes. Thus, the FTBS method is also know as the first-order upwind scheme (since it is first-order accurate in both space and time).

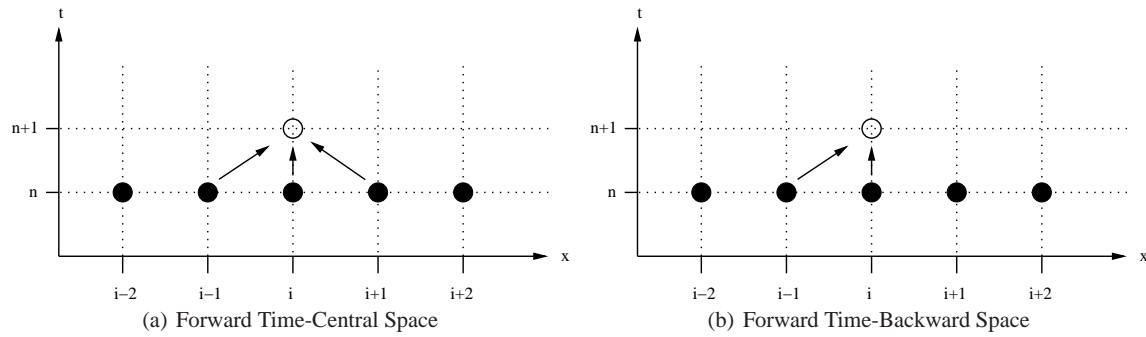


Fig. 21 Solution dependence for finite difference method.

Exercise 4. Download the matlab code from Example 1 and modify the code to use the backward difference formula δ_x^- . Using $u = 1$, $\Delta t = \frac{1}{1000}$ and $\Delta x = \frac{1}{50}$ the FTBS method does not exhibit the same instability as the FTCS method. Fixing $u =$, $\Delta x = \frac{1}{50}$ what is the corresponding value of Δt above which the FTBS method begin exhibit instabilities.

- (a) $\Delta t = \frac{1}{500}$
- (b) $\Delta t = \frac{1}{100}$
- (c) $\Delta t = \frac{1}{50}$
- (d) $\Delta t = \frac{1}{10}$

Exercise 5. Fixing $u =$, $\Delta t = \frac{1}{100}$ what is the corresponding value of Δx above which the FTBS method begin exhibit instabilities.

- (a) $\Delta x = \frac{1}{500}$
- (b) $\Delta x = \frac{1}{100}$
- (c) $\Delta x = \frac{1}{50}$
- (d) $\Delta x = \frac{1}{10}$

Exercise 6. Consider the first-order upwind scheme applied to the convection equation. For what values of Δt , Δx and u will the method return the exact solution at the nodes?

- (a) $\Delta t = \frac{1}{50}$, $\Delta x = \frac{1}{50}$, $u = 1$
- (b) $\Delta t = \frac{1}{50}$, $\Delta x = \frac{1}{50}$, $u > 0$
- (c) $\Delta t = \frac{\Delta x}{u}$, $u > 0$
- (d) both a) and c)