Chapter 4

Partial Differential Equations

In this chapter we will consider continuous systems described by *partial differential equations* (PDE's), e.g. fluids (Navier-Stokes' equation), Schrödinger equation, Laplace's and Poisson's equation.

We will consider the solution using **Finite Differences**: discretising the solution space, i.e. replacing continuous space by a finite grid. Such problems are of two classes **Boundary Value** and **Initial Value**.

4.1 Boundary-Value Problems

The paradigm is *Poisson's equation*:

$$\nabla^2 \phi = \rho. \tag{4.1}$$

It is an example of an **Elliptic Equation**.

We aim to solve this equation in a volume V, with boundary conditions imposed on the surface ∂V. We begin by discretising the equation, and for completeness consider the two-dimensional problem:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi = \rho. \tag{4.2}$$

Setting

$$\begin{aligned} x &= i\Delta x \\ y &= j\Delta x \end{aligned}$$

we have

$$\begin{aligned} \frac{\partial \phi}{\partial x} &\to \frac{1}{2\Delta x} \{\phi(i+1,j) - \phi(i-1,j)\} & (4.3) \\ \nabla^2 \phi &\to \left(\frac{1}{\Delta x}\right)^2 \{\phi(i+1,j) + \phi(i-1,j) + \phi(i,j+1) + \\ &\quad \phi(i,j-1) - 4\phi(i,j)\}. & (4.4) \end{aligned}$$

Discretising reduces a continuous, linear partial differential equation to a set of linear, simultaneous equations, of the form

$$\mathbf{Au} = \mathbf{b} \tag{4.5}$$

where

- $\mathbf{b}$ is the source vector
- $\mathbf{A}$ is a sparse finite-difference matrix
- $\mathbf{u}$ is the solution vector

$$\boxed{\begin{array}{c}\text{Solution of Boundary-Value Problem}\\ \leftrightarrow \\ \text{Solution of large, sparse system of linear equations}\end{array}}$$

There are two broad methods for solving these equations

- **Relaxation Methods**, where we employ the sparseness of $\mathbf{A}$ directly.
- **Matrix Methods**, where we look at more general methods at inverting the matrix $\mathbf{A}$.

4.1.1 Relaxation Methods

These rely on separating $\mathbf{A}$ into two parts, one of which is easily invertible, $\mathbf{E}$, and a remainder, $\mathbf{F}$:

$$\mathbf{A} = \mathbf{E} - \mathbf{F}. \tag{4.6}$$

The algorithm is then

```
Make initial guess $\mathbf{u}^{(0)}$
Iterate until convergence:
    $\mathbf{u}^{(n)} = \mathbf{E}^{-1}(\mathbf{F}\mathbf{u}^{(n-1)} + \mathbf{b})$
```

There are several well-known examples:

- **Jacobi**

  Let us consider the paradigm problem of equation (4.1). We will label the grid-points in "typewriter order", with the address of point (i, j) given by $n = i + L_x(j-1)$, for $i = 1, L_x$ and $j = 1, L_y$, where L_x and L_y are the number of grid points in the x and y directions respectively. We write the matrix $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U} \tag{4.7}$$

where

$$\begin{aligned} \mathbf{L} &= \text{lower triangular} \\ \mathbf{U} &= \text{upper triangular} \\ \mathbf{D} &= \text{diagonal} \end{aligned}$$

Then the iterative scheme is

$$\mathbf{D}\mathbf{u}^{(n)} = -(\mathbf{L}+\mathbf{U})\,\mathbf{u}^{(n-1)} + \mathbf{b}$$

- **Gauss-Seidel**

  Here our iteration step is

  $$(\mathbf{D}+\mathbf{L})\,\mathbf{u}^{(n)} = -\mathbf{U}\mathbf{u}^{(n-1)} + \mathbf{b}$$

  It is easy to check that this corresponds to sweeping through the grid in typewriter order, and updating in place. We will see below that it is more efficient that the Jacobi algorithm.

- ***Parallel*** **Gauss-Seidel**

  Whilst it is straightforward to implement the Jacobi algorithm on a parallel machine, it is easy to see that this is not the case with the Gauss-Seidel algorithm. There are versions, however, that can be implemented on a parallel machine: the best-known example is the *checkerboard* or *red-black* decomposition.

  We begin by labelling sites according to whether $(i+j)$ is *even* or *odd*, known as the **parity**. We will now specialise to the case where the *off-diagonal* elements of **A** only connect sites of **opposite parity** (true for Poisson's equation above), and write

  $$\mathbf{A} = \mathbf{D} + \mathbf{A}^{\mathrm{oe}} + \mathbf{A}^{\mathrm{eo}}$$

  where $\mathbf{A}^{\mathrm{oe}}$ and $\mathbf{A}^{\mathrm{eo}}$ connect even-to-odd and odd-to-even sites respectively. Our iteration scheme is now

  $$(\mathbf{D}+\mathbf{A}^{\mathrm{oe}})\,\mathbf{u}^{(n)} = -\mathbf{A}^{\mathrm{eo}}\mathbf{u}^{(n-1)}$$

  This corresponds to updating the *even* sites, following by updating the *odd* sites using the newly computed values on the even sites. If the off-diagonal elements connect sites of the *same* parity, the red-black decomposition is not, in general, useful.

4.1.2 Convergence of Relaxation Methods

All of the above methods are of the form

$$\mathbf{E}\mathbf{u}^{(n)} = \mathbf{F}\mathbf{u}^{(n-1)} + \mathbf{b}. \tag{4.8}$$

$\mathbf{E}^{-1}\mathbf{F}$ is the **iteration matrix** and its eigenvalues determine the speed of convergence.

To see this, we introduce the **residual vector**

$$\mathbf{r}^{(n)} = \mathbf{u}^{(n)} - \mathbf{u}, \tag{4.9}$$

where $\mathbf{u}$ is the exact solution, i.e.

$$\mathbf{E}\mathbf{u} = \mathbf{F}\mathbf{u} + \mathbf{b}. \tag{4.10}$$

Then

$$\mathbf{E}\mathbf{r}^{(n)} = \mathbf{F}\mathbf{r}^{(n-1)}, \tag{4.11}$$

and

$$\mathbf{r}^{(n)} = (\mathbf{E}^{-1}\mathbf{F})^n\mathbf{r}^{(0)} \tag{4.12}$$

The **spectral radius**, $\rho(\mathbf{M})$ of $\mathbf{M}$ is the *absolute value of the eigenvalue of* $\mathbf{M}$ *with the largest absolute value.*

$$\boxed{\text{For convergence we require } \rho(\mathbf{E}^{-1}\mathbf{F}) < 1.}$$

To estimate $\rho(\mathbf{M})$ we introduce the concept of the matrix norm $||M||$. By the definition of norms on matrices and vectors require:

$$||Mx|| \le ||M||\ ||x||(Schwarzinequality) \tag{4.13}$$

and

$$||\alpha M|| \le |\alpha|\ ||M|| \tag{4.14}$$

for all complex numbers α.

If we select an eigenvector x of M with eigenvalue λ such that

$$Mx = \lambda x \tag{4.15}$$

then

$$||Mx|| = ||\lambda x|| = |\lambda|\ ||x|| \le ||M||\ ||x|| \tag{4.16}$$

so

$$|\lambda| \le ||M|| \qquad \forall\lambda \tag{4.17}$$

If we choose λ to be the eigenvalue with largest maginitude then the LHS is just $\rho(M)$, ie

$$\rho(M) \le ||M|| \tag{4.18}$$

This is true for all norms.

Thus for the Jacobi algorithm:

$$\rho_J \leq \|\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\|_\infty = \max_i \sum_{j\neq i} \left|\frac{a_{ij}}{a_{ii}}\right|, \tag{4.19}$$

so the solution converges provided $\mathbf{A}$ is **diagonally dominant**, i.e.

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}|.$$

Let us now evaluate the spectral radius for the Jacobi algorithm for the Poisson equation on an N^d grid with **Dirichlet** boundary conditions ($\phi = 0$ on boundary):

$\phi_{\mathbf{x}} = 0$ for $x_j = 0$ or N.

$$\phi_{\mathbf{x}}^{(n)} = \sum_{\mathbf{k}} \left(\sin \frac{\pi \mathbf{k}.\mathbf{x}}{N}\right) \tilde{\phi}_{\mathbf{k}}^{(n)}, \qquad k_j = 1, \ldots, N-1. \tag{4.20}$$

The Jacobi algorithm is

$$\tilde{\phi}_{\mathbf{k}}^{(n+1)} = \frac{1}{d}\sum_{j=1}^{d} \left(\cos\frac{\pi k_j}{N}\right) \tilde{\phi}_{\mathbf{k}}^{(n)} - \frac{a^2}{4}\tilde{\rho}_{\mathbf{k}}. \tag{4.21}$$

Hence,

$$\begin{aligned} \rho_J &= \frac{1}{d}\sum_{j=1}^{d} \cos\frac{\pi}{N} & (4.22)\\ &\approx 1 - \frac{\pi^2}{2N^2} \qquad N \text{ large.} & (4.23) \end{aligned}$$

The number of iterations, $n_J(p)$, required to reduce the error by a factor 10^{-p} is

$$n_J(p) \sim -\frac{p \ln 10}{\ln \rho_J} \sim pN^2. \tag{4.24}$$

For Gauss-Seidel we have

$$\rho_{GS} \sim 1 - \frac{\pi^2}{N^2}$$

so that

$$n_{GS} \sim n_J/2.$$

Successive Over-Relaxation (SOR)

This is obtained from Gauss-Seidel by writing it as

$$\begin{aligned} \mathbf{u}^{(n)} &= (\mathbf{L}+\mathbf{D})^{-1}(-\mathbf{U}\mathbf{u}^{(n-1)} + \mathbf{b}) \\ &= (\mathbf{L}+\mathbf{D})^{-1}\left[(\mathbf{L}+\mathbf{D})\mathbf{u}^{(n-1)} + \mathbf{b} - \mathbf{A}\mathbf{u}^{(n-1)}\right] \\ &= \mathbf{u}^{(n-1)} - (\mathbf{L}+\mathbf{D})^{-1}(\mathbf{A}\mathbf{u}^{(n-1)} - \mathbf{b}) & (4.25) \end{aligned}$$

and then **overcorrecting**:

$$\mathbf{u}^{(n)} = \mathbf{u}^{(n-1)} - \omega(\mathbf{L}+\mathbf{D})^{-1}(\mathbf{A}\mathbf{u}^{(n-1)} - \mathbf{b}). \tag{4.26}$$

- SOR converges for $0 < \omega < 2$;
- for finite difference equations, **over-relaxation**, $1 < \omega < 2$, is usually faster than Gauss-Seidel ($\omega = 1$);
- optimal ω is

  $$\omega_{opt} = \frac{2}{1+\sqrt{1-\rho_J^2}} \tag{4.27}$$

  then

  $$\rho_{SOR} = \left(\frac{\rho_J}{1+\sqrt{1-\rho_J^2}}\right)^2. \tag{4.28}$$

  Thus for the Poisson equation,

  $$\omega_{opt} \approx \frac{2}{1+\pi/N} \tag{4.29}$$

  $$\rho_{SOR} \approx 1 - \frac{2\pi}{N} \tag{4.30}$$

  and the number of iterations required to reduce the error by 10^{-p} is

  $$n_{SOR}(p) \sim pN. \tag{4.31}$$

- There are cases where it is necessary to **under-relax**. This is particularly true for non-linear equations where under-relaxation is necessary to stabilize the solution.

Generally ω must be chosen by **trial and error.**

4.1.3 Matrix Methods

The relaxation methods discussed above are neither particularly efficient nor particularly robust. To derive more powerful techniques, we look at more general methods of inverting a matrix. The most widely used methods belong to a class of **conjugate gradient** algorithms, which are **exact**, in a sense we will see below, *in exact arithmetic.*

Conjugate Gradient Algorithm

Consider the solution of

$$\mathbf{A}\mathbf{u} = \mathbf{b}$$

where $\mathbf{A}$ is an $M \times M$, **symmetric, positive definite** matrix. The conjugate gradient method is motivated by the observation that the function

$$\begin{aligned} F(\mathbf{u}) &= \frac{1}{2}(\mathbf{b} - \mathbf{A}\mathbf{u})^T \mathbf{A}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{u}) \\ &= \frac{1}{2}\mathbf{u}^T \mathbf{A}\mathbf{u} - \mathbf{u}^T \mathbf{b} + \text{const} \end{aligned} \tag{4.32}$$

attains its minimum uniquely at

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{b}.$$

To understand the method, it is useful to consider first the "method of steepest descent", where a sequence $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \ldots$ is generated by finding, at each stage, an α_k which minimises $F(\mathbf{u}_k + \alpha_k \mathbf{r}_k)$ with $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{u}_k$, and setting

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{r}_k. \tag{4.33}$$

At each stage we are going down the direction of steepest descent, since

$$\nabla F(\mathbf{u}_k) = \mathbf{A}\mathbf{u}_k - \mathbf{b} = -\mathbf{r}_k.$$

The **conjugate gradient** method ensures that, at each stage, we are going down a direction $\mathbf{p}_k$ *conjugate to all previous directions.* Thus after M iterations, the whole space is spanned and u_{M+1} is the **exact solution** to

$$\nabla F(\mathbf{u}_{M+1}) = \mathbf{A}\mathbf{u}_{M+1} - \mathbf{b} = 0. \tag{4.34}$$

The algorithm is:

Initial condition:

$$\mathbf{r}_0 = \mathbf{p}_0 = \mathbf{b} - \mathbf{A}\mathbf{u}$$

For $k = 0, 1, \ldots$:

$$\begin{aligned} \alpha_k &= \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k} \\ \mathbf{u}_{k+1} &= \mathbf{u}_k + \alpha_k \mathbf{p}_k \\ \mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k \end{aligned}$$

If $\mathbf{r}_{k+1}^T \mathbf{r}_{k+1} <$ *tolerance,* ***stop***

$$\begin{aligned} \beta_k &= \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} \\ \mathbf{p}_{k+1} &= \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k. \end{aligned}$$

Note that the algorithm only involves $\mathbf{A}$ multiplicatively, and usually converges in much less than M iterations. Only four vectors need to be stored ($\mathbf{u}$, $\mathbf{r}$, $\mathbf{p}$, $\mathbf{Ap}$). The name arises because $\mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0$ for $i \neq j$. At each stage, $\mathbf{r}_{k+1}$ is an estimate of the residual $\mathbf{b} - \mathbf{A}\mathbf{u}_{k+1}$.

4.2 Initial-Value Problems

There are two classes: **Hyperbolic Equations**, e.g. the wave equation

$$\frac{\partial^2 \phi}{\partial t^2} = c^2 \nabla^2 \phi \tag{4.35}$$

and **Parabolic Equations**, e.g. the diffusion equation

$$\frac{\partial \phi}{\partial t} = D \nabla^2 \phi \tag{4.36}$$

In contrast to the boundary-value problems, we are given the solution ϕ *everywhere* at $t = 0$, and wish to see how ϕ evolves with time.

In the case of initial-value problems, the potential pitfalls of the numerical simulation are concerned with the **stability** of the solution. We will illustrate this for the case of the one-dimensional diffusion equation, using the Euler algorithm introduced in *Computer Simulation* for the time evolution.

4.2.1 One-dimensional Diffusion Equation

The one-dimensional diffusion equation is

$$\frac{\partial \phi}{\partial t} = D \frac{\partial^2 \phi}{\partial x^2} = -H\phi \tag{4.37}$$

Forward Time, Centred Space (FTCS)

We will discretise the spatial coordinate on a grid with spacing Δx using the usual three-point stencil ("centred space") , and discretise the temporal coordinate on a grid of spacing Δt using the "forward-time" Euler method. Introducing

$$\phi^{(n)} = \phi(t = n\Delta t),$$

the FTCS form for the discretised diffusion equation is

$$\frac{\phi^{(n+1)}(j)-\phi^{(n)}(j)}{\Delta t}=\frac{D}{\Delta x^2}\left[\phi^{(n)}(j+1)+\phi^{(n)}(j-1)-2\phi^{(n)}(j)\right], \tag{4.38}$$

yielding

$$\phi^{(n+1)}=(1-\Delta tH)\phi^{(n)} \tag{4.39}$$

where

$$H\phi(i)=-\frac{D}{\Delta x^2}[\phi(j+1)+\phi(j-1)-2\phi(j)]. \tag{4.40}$$

The FTCS method is an **explicit** scheme: $\phi^{(n+1)}$ *only depends on quantities already known.*

To investigate the stability of this method, let ψ_λ be the eigenvectors of the discrete operator H, with eigenvalues ϵ_λ. Since H is Hermitian, the eigenvalues are real and the eigenvectors can be chosen orthonormal. Thus we can expand the solution at any time in this basis:

$$\phi^{(n)}=\sum_\lambda \tilde{\phi}_\lambda^{(n)}\psi_\lambda \tag{4.41}$$

Formally, the solution of equation (4.37) is

$$\phi^{(n)}=e^{-nH\Delta t}\phi^{(0)}=\sum_\lambda e^{-n\epsilon_\lambda\Delta t}\tilde{\phi}_\lambda^{(0)}\psi_\lambda, \tag{4.42}$$

yielding

$$\tilde{\phi}_\lambda^{(n)}=e^{-\epsilon_\lambda\Delta t}\tilde{\phi}_\lambda^{(n-1)}. \tag{4.43}$$

The modes are attenuated with increasing n.

The FTCS method yields a solution

$$\tilde{\phi}_\lambda^{(n)}=(1-\epsilon_\lambda\Delta t)\tilde{\phi}_\lambda^{(n-1)}. \tag{4.44}$$

The **amplification factor** ξ_λ is given by

$$\xi_\lambda=1-\epsilon_\lambda\Delta t \tag{4.45}$$

The time dependence is ξ_λ^n, and will be **unstable** if $|\xi_\lambda|>1$.

Let us now apply these ideas to the one-dimensional diffusion equation on a spatial grid of N points. It is easy to see that the eigenfunctions are given by

$$\psi_\lambda(j)=e^{\lambda\pi ij/N} \tag{4.46}$$

Then the *eigenvalue equation* is

$$\epsilon_\lambda \psi_\lambda = -\frac{D}{2\Delta x^2}\left\{e^{\lambda\pi i/N} + e^{-\lambda\pi i/N} - 2\right\}\psi_\lambda \tag{4.47}$$

yielding

$$\epsilon_\lambda = \frac{4D}{\Delta x^2}\sin^2\frac{\lambda\pi}{2N} \tag{4.48}$$

Thus the amplification amplitude of wave number λ is

$$\xi_\lambda = 1 - \frac{4D\Delta t}{\Delta x^2}\sin^2\frac{\lambda\pi}{2N} \tag{4.49}$$

and $|\xi| < 1$ provided

$$\frac{2D\Delta t}{\Delta x^2} \leq 1 \tag{4.50}$$

$\Delta x^2/D$ is the *diffusion time across* Δx, and thus this condition requires that the *time step* Δt *is less than the diffusion time across* Δx.

This method is highly unsatisfactory. The *instability* requires us to employ a time step determined not by the characteristic time scale of the evolution (l^2/D where l is the system size), but rather by the spatial step size we are using! Note that this is distinct from the **accuracy** of the solution.

Fully Implicit Scheme

A way round the instability is to replace the second spatial derivative by the value at the new time:

$$\frac{\phi^{(n+1)}(j) - \phi^{(n)}(j)}{\Delta t} = \frac{D}{\Delta x^2}\left[\phi^{(n+1)}(j+1) + \phi^{(n+1)}(j-1) - 2\phi^{(n+1)}(j)\right], \tag{4.51}$$

This is an *implicit* scheme since $\phi^{(n+1)}$ appears on *both* sides of the equation. This has an amplification factor

$$\xi_\lambda = \frac{1}{1+\epsilon_\lambda\Delta t} < 1, \tag{4.52}$$

but requires the solving a tridiagonal system of equations, which is tractable.

If we wish to construct a scheme which is both **stable** and **accurate**, we can average the *fully implicit* and *fully explicit* schemes to give the **Crank-Nicholson scheme**:

$$\begin{aligned}\frac{\phi^{(n+1)}(j) - \phi^{(n)}(j)}{\Delta t} &= \frac{D}{2\Delta x^2}[(\phi^{(n+1)}(j+1) + \phi^{(n+1)}(j-1) - 2\phi^{(n+1)}(j)) \\ &\quad +(\phi^{(n)}(j+1) + \phi^{(n)}(j-1) - 2\phi^{(n)}(j))],\end{aligned} \tag{4.53}$$

where is *second order* in Δt and *stable.* The cost is the use of a sparse linear equation solver.

4.2.2 Wave Equation

Many initial-value problems in one dimension can be written as *flux-conservation* equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{\partial}{\partial x}\mathbf{F}(\mathbf{u}) \tag{4.54}$$

where **F** is the **conserved flux**. For the case of the one-dimensional wave equation,

$$\frac{\partial^2 \phi}{\partial t^2} = c^2 \frac{\partial^2 \phi}{\partial x^2}$$

we set

$$r = c\frac{\partial \phi}{\partial x}; \quad s = \frac{\partial \phi}{\partial t}, \tag{4.55}$$

and obtain equation (4.54) with

$$\mathbf{u} = \begin{pmatrix} r \\ s \end{pmatrix}; \quad \mathbf{F} = \begin{pmatrix} 0 & -c \\ -c & 0 \end{pmatrix}\mathbf{u}. \tag{4.56}$$

Thus we will consider the simple first-order PDE

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}. \tag{4.57}$$

This has a solution of the form $u = f(x - ct)$; u is transported by flow with velocity c.

We will discretise equation (4.57) using the FTCS algorithm

$$\frac{u^{(n+1)}(j) - u^{(n)}(j)}{\Delta t} = -c\left(\frac{u^{(n)}(j+1) - u^{(n)}(j-1)}{2\,\Delta x}\right). \tag{4.58}$$

Once again, we write the solution in terms of eigenmodes with eigenvalue ϵ_λ. A stability analysis yields

$$\xi_\lambda = 1 - i\frac{c\,\Delta t}{\Delta x}\sin\frac{\lambda\pi}{N}, \tag{4.59}$$

for the $|\xi_\lambda| \geq 1$; the method is **unconditionally unstable**!.

Lax Method

In the time derivative, make the substition

$$u^{(n)}(j) = \frac{1}{2}[u^{(n)}(j+1) + u^{(n)}(j)]. \tag{4.60}$$

Thus

$$u^{(n+1)}(j) = \frac{1}{2}[u^{(n)}(j+1) + u^{(n)}(j)] - c\Delta t\left[\frac{u^{(n)}(j+1) - u^{(n)}(j-1)}{2\Delta x}\right], \tag{4.61}$$

for which the amplification factor is

$$\xi_\lambda = \cos\frac{\lambda\pi}{N} - i\frac{c\Delta t}{2\Delta x}\sin\frac{\lambda\pi}{N} \tag{4.62}$$

which is stable, $|\xi| < 1$, provided

$$\frac{c\Delta t}{\Delta x} \leq 1.$$

This is the **Courant Condition**: $u^{(n+1)}(j)$ depends on information at previous times *within the domain of dependence.* If Δt is too large, some information is missing and an instability is created.

Higher Dimensions

The *flux conservative* equations become

$$\frac{\partial u}{\partial t} = -\nabla.\mathbf{F} \tag{4.63}$$

Lax scheme in d spatial dimensions:

$$u^{(n+1)}(\mathbf{x}) = \frac{1}{2d}\sum_{j=1}^{d}\left(u^{(n)}(\mathbf{x}+\hat{\mathbf{j}}) + u^{(n)}(\mathbf{x}-\hat{\mathbf{j}})\right) - \frac{\Delta t}{2\Delta x}\sum_{j=1}^{d}\left(F_j^{(n)}(\mathbf{x}+\hat{\mathbf{j}}) - F_j^{(n)}(\mathbf{x}-\hat{\mathbf{j}})\right). \tag{4.64}$$

In this case, the **Courant Condition** is

$$\frac{|\mathbf{v}|\Delta t}{\Delta x} \leq \frac{1}{\sqrt{d}} \tag{4.65}$$

where $\mathbf{F} = \mathbf{v}u$ with $\mathbf{v}$ constant.