Numerical Methods for Partial Differential Equations

Seongjai Kim

Department of Mathematics and Statistics
Mississippi State University
Mississippi State, MS 39762 USA
Email: skim@math.msstate.edu

October 19, 2026

Seongjai Kim, Department of Mathematics and Statistics, Mississippi State University, Mississippi State, MS 39762-5921 USA Email: skim@math.msstate.edu. The work of the author is supported in part by NSF grant DMS-1228337.

Prologue

In the area of “Numerical Methods for Differential Equations", it seems very hard to find a textbook incorporating mathematical, physical, and engineering issues of numerical methods in a synergistic fashion. So the first goal of this lecture note is to provide students a convenient textbook that addresses both physical and mathematical aspects of numerical methods for partial differential equations (PDEs).

In solving PDEs numerically, the following are essential to consider:

- physical laws governing the differential equations (physical understanding),
- stability/accuracy analysis of numerical methods (mathematical understanding),
- issues/difficulties in realistic applications, and
- implementation techniques (efficiency of human efforts).

In organizing the lecture note, I am indebted by Ferziger and Peric [23], Johnson [32], Strikwerda [64], and Varga [68], among others. Currently the lecture note is not fully grown up; other useful techniques would be soon incorporated. Any questions, suggestions, comments will be deeply appreciated.

Contents

Title **2**

Prologue **3**

Table of Contents **10**

1 Mathematical Preliminaries **1**

1.1. Taylor's Theorem & Polynomial Fitting . . . 2
1.2. Finite Differences . . . 6
1.2.1. Uniformly spaced grids . . . 6
1.2.2. General Grids . . . 8
1.3. Overview of PDEs . . . 12
1.4. Calculus of Variations: Variational Calculus . . . 17
1.5. Difference Equations . . . 20
1.6. Homework . . . 23

2 Numerical Solution of Ordinary Differential Equations **25**

2.1. Elementary Theory of Initial-Value Problems . . . 26
2.2. Taylor-Series Methods . . . 28
2.2.1. The Euler Method . . . 28
2.2.2. Higher-order Taylor Methods . . . 34
2.3. Runge-Kutta Methods . . . 37
2.3.1. Second-order Runge-Kutta method . . . 38
2.3.2. Fourth-order Runge-Kutta method . . . 40
2.3.3. Adaptive methods . . . 42
2.4. One-Step Methods: Accuracy Comparison . . . 43
2.5. Multi-step Methods . . . 46
2.6. High-Order Equations & Systems of Differential Equations . . . 50
2.7. Implicit Methods and Implicit Differential Equations . . . 57
2.7.1. Revisit of the Euler Method . . . 57
2.7.2. Implicit Differential Equations . . . 61

2.8. Homework . . . 63

3 Properties of Numerical Methods **65**
3.1. General Parabolic Problems: Maximum Principle . . . 66
3.2. A Model Problem: Heat Conduction in 1D . . . 68
3.3. Consistency . . . 70
3.4. Convergence . . . 72
3.5. Stability . . . 77
3.5.1. Approaches for proving stability . . . 77
3.5.2. The von Neumann analysis . . . 79
3.5.3. Influence of lower-order terms . . . 82
3.6. Boundedness – Maximum Principle . . . 83
3.6.1. Convection-dominated fluid flows . . . 83
3.6.2. Stability vs. boundedness . . . 84
3.7. Conservation . . . 85
3.8. A Multi-Step Central Scheme . . . 86
3.9. The θ-Method . . . 87
3.9.1. Stability analysis for the θ-Method . . . 88
3.9.2. Accuracy order . . . 89
3.9.3. Maximum principle . . . 91
3.9.4. Error analysis . . . 92
3.10. Homework . . . 93

4 Elliptic Partial Differential Equations **95**
4.1. Regularity Estimates . . . 96
4.2. Maximum and Minimum Principles . . . 98
4.3. Discrete Maximum and Minimum Principles . . . 101
4.4. Coordinate Changes . . . 103
4.5. Cylindrical and Spherical Coordinates . . . 105
4.6. Homework . . . 108

5 Finite Difference Methods for Elliptic Equations **109**
5.1. Finite Difference (FD) Methods . . . 110
5.1.1. Constant-coefficient problems . . . 110
5.1.2. General diffusion coefficients . . . 113
5.1.3. FD schemes for mixed derivatives . . . 114
5.1.4. L^∞-norm error estimates for FD schemes . . . 115
5.1.5. The Algebraic System for FDM . . . 119

5.2. Solution of Linear Algebraic Systems . . . 122
5.2.1. Direct method: the LU factorization . . . 122
5.2.2. Linear iterative methods . . . 126
5.2.3. Convergence theory . . . 127
5.2.4. Graph theory for the estimation of the spectral radius . . . 128
5.3. Relaxation methods . . . 133
5.3.1. Jacobi method . . . 134
5.3.2. Gauss-Seidel method . . . 134
5.3.3. Successive over-relaxation (SOR) method . . . 135
5.3.4. Convergence of relaxation methods . . . 136
5.3.5. Line relaxation methods . . . 137
5.4. Krylov Subspace Methods . . . 140
5.4.1. Steepest descent method . . . 140
5.4.2. Conjugate gradient (CG) method . . . 143
5.4.3. Preconditioned CG method . . . 146
5.5. Other Iterative Methods . . . 147
5.5.1. Incomplete LU-factorization . . . 147
5.6. Numerical Examples with Python . . . 150
5.7. Homework . . . 155

The 15% Rule: Turning Curiosity into Financial Freedom **158**

6 Finite Element and Finite Volume Methods for Elliptic Equations **163**
6.1. The Hilbert spaces . . . 164
6.2. Abstract Variational Problem . . . 166
6.3. Finite Element (FE) Methods in 1D Space . . . 168
6.3.1. Variational formulation . . . 168
6.3.2. Formulation of FEMs . . . 172
6.4. An error estimate for FEM in 1D . . . 183
6.5. Other Variational Principles . . . 187
6.6. FEM for the Poisson Equation . . . 188
6.6.1. Integration by Parts . . . 188
6.6.2. The Formulation of FEMs . . . 191
6.6.3. Assembly: Element stiffness matrices . . . 195
6.6.4. Extension to Neumann boundary conditions . . . 197
6.7. Finite Volume (FV) Method . . . 198
6.8. Average of The Diffusion Coefficient . . . 202
6.9. Numerical Examples with Python . . . 203

6.10. Homework . . . 206

7 FD Methods for Hyperbolic Equations 209

7.1. Introduction . . . 210
7.2. Basic Difference Schemes . . . 213
7.2.1. Consistency . . . 215
7.2.2. Convergence . . . 217
7.2.3. Stability . . . 220
7.2.4. Accuracy . . . 224
7.3. Conservation Laws . . . 227
7.3.1. Euler equations of gas dynamics . . . 227
7.4. Shocks and Rarefaction . . . 234
7.4.1. Characteristics . . . 234
7.4.2. Weak solutions . . . 236
7.5. Numerical Methods . . . 238
7.5.1. Modified equations . . . 238
7.5.2. Conservative methods . . . 245
7.5.3. Consistency . . . 249
7.5.4. Godunov's method . . . 250
7.6. Nonlinear Stability . . . 251
7.6.1. Total variation stability (TV-stability) . . . 252
7.6.2. Total variation diminishing (TVD) methods . . . 254
7.6.3. Other nonoscillatory methods . . . 255
7.7. Numerical Examples with Python . . . 260
7.8. Homework . . . 262

8 Locally One-Dimensional Methods 263

8.1. Heat Conduction in 1D Space: Revisited . . . 264
8.2. Heat Equation in Two and Three Variables . . . 270
8.2.1. The θ-method . . . 271
8.2.2. Convergence analysis for θ-method . . . 273
8.3. LOD Methods for the Heat Equation . . . 276
8.3.1. The ADI method . . . 277
8.3.2. Accuracy of the ADI: Two examples . . . 283
8.3.3. The general fractional step (FS) procedure . . . 286
8.3.4. Improved accuracy for LOD procedures . . . 288
8.3.5. A convergence proof for the ADI-II . . . 295
8.3.6. Accuracy and efficiency of ADI-II . . . 297

8.4. Homework . . . 299

9 Domain Decomposition Methods **301**
9.1. Introduction to DDMs . . . 302
9.2. Overlapping Schwarz Alternating Methods (SAMs) . . . 305
9.2.1. Variational formulation . . . 305
9.2.2. SAM with two subdomains . . . 306
9.2.3. Convergence analysis . . . 307
9.2.4. Coarse subspace correction . . . 310
9.3. Nonoverlapping DDMs . . . 313
9.3.1. Multi-domain formulation . . . 313
9.3.2. The Steklov-Poincaré operator . . . 315
9.3.3. The Schur complement matrix . . . 317
9.4. Iterative DDMs Based on Transmission Conditions . . . 320
9.4.1. The Dirichlet-Neumann method . . . 320
9.4.2. The Neumann-Neumann method . . . 322
9.4.3. The Robin method . . . 323
9.4.4. Remarks on DDMs of transmission conditions . . . 324
9.5. Homework . . . 330

P Projects **331**
P.1. Coordinate Transformations and FDMs . . . 332
P.2. High-order FEMs for PDEs of One Spacial Variable . . . 334

A Basic Concepts in Fluid Dynamics **337**
A.1. Conservation Principles . . . 337
A.2. Conservation of Mass . . . 339
A.3. Conservation of Momentum . . . 339
A.4. Non-dimensionalization of the Navier-Stokes Equations . . . 342
A.5. Generic Transport Equations . . . 344
A.6. Homework . . . 345

B Orthogonal Polynomials and Quadratures **347**
B.1. Orthogonal Polynomials . . . 347
B.2. Gauss-Type Quadratures . . . 349

C Some Mathematical Formulas **353**
C.1. Trigonometric Formulas . . . 353
C.2. Vector Identities . . . 353

D Finite Difference Formulas **355**

Bibliography **357**

Index **363**

CHAPTER 1

Mathematical Preliminaries

In the approximation of derivatives, we consider the ***Taylor series expansion*** and the ***curve fitting*** as two of most popular tools. This chapter begins with a brief review for these introductory techniques, followed by finite difference schemes, and an overview of partial differential equations (PDEs).

In the study of numerical methods for PDEs, implementation and computational experimentation are essential to understanding algorithmic behavior. However, these tasks are often too time-consuming to complete within strict deadlines. This bottleneck is particularly challenging for graduate students enrolled in numerical PDE courses. You will receive a basic software tool, **modelcode**, to practice numerical methods effectively.

Contents of Chapter 1

1.1. Taylor's Theorem & Polynomial Fitting 2
1.2. Finite Differences 6
1.3. Overview of PDEs 12
1.4. Calculus of Variations: Variational Calculus 17
1.5. Difference Equations 20
1.6. Homework 23

1.1. Taylor's Theorem & Polynomial Fitting

While the differential equations are defined on continuous variables, their numerical solutions must be computed on a finite number of discrete points. The derivatives should be approximated appropriately to simulate the physical phenomena accurately and efficiently. Such approximations require various mathematical and computational tools. In this section we present a brief review for the Taylor's series and the curve fitting.

Theorem 1.1. (Taylor's Theorem). *Assume that $u \in C^{n+1}[a,b]$ and let $c \in [a,b]$. Then, for every $x \in (a,b)$, there is a point ξ that lies between x and c such that*

$$u(x) = p_n(x) + E_{n+1}(x), \tag{1.1}$$

where p_n is a polynomial of degree $\leq n$ and E_{n+1} denotes the remainder defined as

$$p_n(x) = \sum_{k=0}^{n} \frac{u^{(k)}(c)}{k!}(x-c)^k, \quad E_{n+1}(x) = \frac{u^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}.$$

The formula (1.1) can be rewritten for $u(x+h)$ (about x) as follows:
For $x,\ x+h \in (a,b)$,

$$\begin{aligned} u(x+h) &= \sum_{k=0}^{n} \frac{u^{(k)}(x)}{k!}h^k + \frac{u^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \\ &= u(x) + \frac{u'(x)}{1!}h + \frac{u''(x)}{2!}h^2 + \cdots + \frac{u^{(n)}(x)}{n!}h^n + \frac{u^{(n+1)}(\xi)}{(n+1)!}h^{n+1}. \end{aligned} \tag{1.2}$$

Curve Fitting

Another useful tool in numerical analysis is the *curve fitting*. It is often the case that the solution must be represented as a continuous function rather than a collection of discrete values. For example, when the function is to be evaluated at a point which is not a grid point, the function must be interpolated near the point before the evaluation.

First, we introduce the existence theorem for interpolating polynomials.

Theorem 1.2. *Let $x_0, x_1, \cdots, x_N$ be a set of distinct points. Then, for arbitrary real values $y_0, y_1, \cdots, y_N$, **there is a unique polynomial** p_N of degree $\leq N$ such that*

$$p_N(x_i) = y_i, \quad i = 0, 1, \cdots, N. \tag{1.3}$$

Lagrange interpolating polynomial

- Let $\{a = x_0 < x_1 < \cdots < x_N = b\}$ be a partition of the interval $[a, b]$.
- Then, the **Lagrange form of interpolating polynomial** is formulated as a linear combination of the so-called *cardinal functions*:

$$p_N(x) = \sum_{i=0}^{N} L_{N,i}(x) u(x_i). \tag{1.4}$$

- Here the **cardinal functions** are defined as

$$L_{N,i}(x) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \left(\frac{x - x_j}{x_i - x_j} \right) \in \mathbf{P}_N, \tag{1.5}$$

  where $\mathbf{P}_N$ is the set of polynomials of degree $\leq N$, which satisfy

$$L_{N,i}(x_j) = \delta_{ij}, \quad i, j = 0, 1, \cdots, N.$$

Newton polynomial

The **Newton form of the interpolating polynomial** that interpolates u at $\{x_0, x_1, \cdots, x_N\}$ is given as

$$p_N(x) = \sum_{k=0}^{N} \left[a_k \prod_{j=0}^{k-1} (x - x_j) \right], \tag{1.6}$$

where the coefficients a_k, $k = 0, 1, \cdots, N$, can be computed as divided differences

$$a_k = u[x_0, x_1, \cdots, x_k]. \tag{1.7}$$

Definition 1.3. (Divided Differences) The **divided differences** for the function $u(x)$ are defined as

$$\begin{aligned} u[x_j] &= u(x_j), \\ u[x_j, x_{j+1}] &= \frac{u[x_{j+1}] - u[x_j]}{x_{j+1} - x_j}, \\ u[x_j, x_{j+1}, x_{j+2}] &= \frac{u[x_{j+1}, x_{j+2}] - u[x_j, x_{j+1}]}{x_{j+2} - x_j}, \end{aligned} \tag{1.8}$$

and the recursive rule for higher-order divided differences is

$$\begin{aligned} &u[x_j, x_{j+1}, \cdots, x_m] \\ &\quad = \frac{u[x_{j+1}, x_{j+2}, \cdots, x_m] - u[x_j, x_{j+1}, \cdots, x_{m-1}]}{x_m - x_j}, \end{aligned} \tag{1.9}$$

for $j < m$.

Table 1.1: Divided-difference table for $u(x)$.

x_j	$u[x_j]$	$u[\ ,\ ]$	$u[\ ,\ ,\ ]$	$u[\ ,\ ,\ ,\ ]$	$u[\ ,\ ,\ ,\ ,\ ]$
x_0	$u[x_0]$				
x_1	$u[x_1]$	$u[x_0, x_1]$			
x_2	$u[x_2]$	$u[x_1, x_2]$	$u[x_0, x_1, x_2]$		
x_3	$u[x_3]$	$u[x_2, x_3]$	$u[x_1, x_2, x_3]$	$u[x_0, x_1, x_2, x_3]$	
x_4	$u[x_4]$	$u[x_3, x_4]$	$u[x_2, x_3, x_4]$	$u[x_1, x_2, x_3, x_4]$	$u[x_0, x_1, x_2, x_3, x_4]$

Example

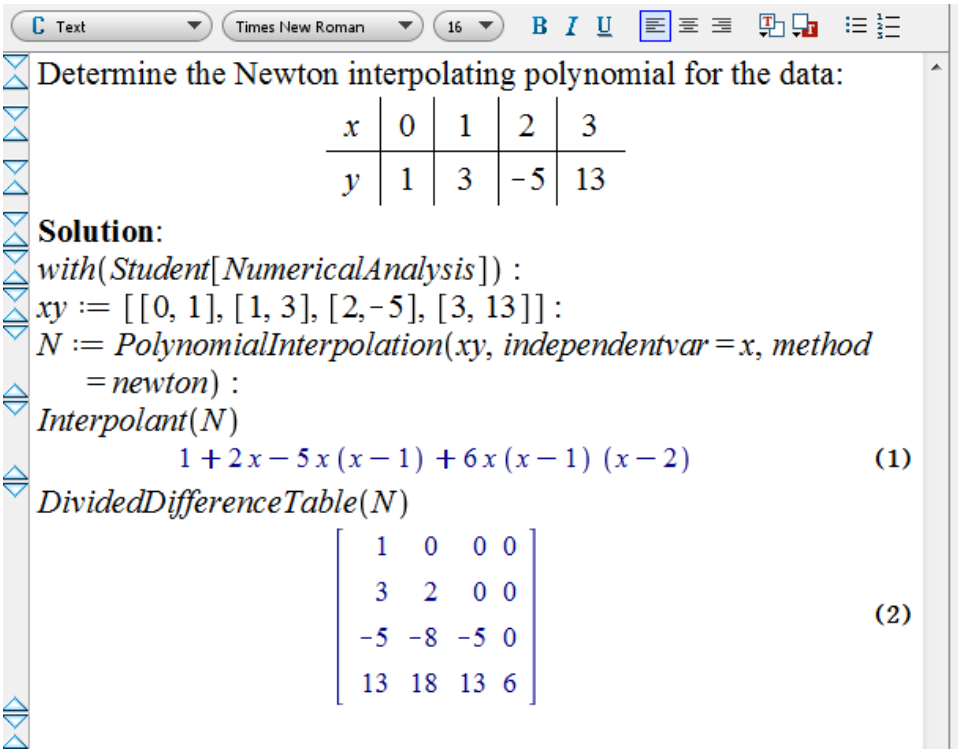


Figure 1.1: A Maple program

Interpolation Error Theorem

Theorem 1.4. (Interpolation Error Theorem). *Let the interval be partitioned into* $\{a = x_0 < x_1 < \cdots < x_N = b\}$ *and* p_N *interpolate* u *at the nodal points of the partitioning. Assume that* $u^{(N+1)}(x)$ *exists for each* $x \in [a, b]$. *Then, there is a point* $\xi \in [a, b]$ *such that*

$$u(x) = p_N(x) + \frac{u^{(N+1)}(\xi)}{(N+1)!} \prod_{j=0}^{N} (x - x_j), \quad \forall\, x \in [a, b]. \tag{1.10}$$

Further, assume that the points are uniformly spaced and $\max_{x\in[a,b]} |u^{(N+1)}(x)| \leq M$, *for some* $M > 0$. *Then,*

$$\max_{x\in[a,b]} |u(x) - p_N(x)| \leq \frac{M}{4(N+1)} \left(\frac{b-a}{N}\right)^{N+1}. \tag{1.11}$$

1.2. Finite Differences

Here are the bases for **finite difference** (FD) approximations. **Taylor series methods** are more common than **curve-fitting methods**. However, curve-fitting methods easily yield higher-order FD schemes for non-uniformly spaced grid points.

1.2.1. Uniformly spaced grids

- Let $h = (b-a)/N$, for some positive integer N, and

$$x_i = a + ih, \quad i = 0, 1, \cdots, N.$$

- Define $u_i = u(x_i)$, $i = 0, 1, \cdots, N$.

Then, it follows from (1.2) that

$$\begin{aligned}
\text{(a)}\quad u_{i+1} &= u_i + u_x(x_i)h + \frac{u_{xx}(x_i)}{2!}h^2 + \frac{u_{xxx}(x_i)}{3!}h^3 \\
&\quad + \frac{u_{xxxx}(x_i)}{4!}h^4 + \frac{u_{xxxxx}(x_i)}{5!}h^5 + \cdots, \\
\text{(b)}\quad u_{i-1} &= u_i - u_x(x_i)h + \frac{u_{xx}(x_i)}{2!}h^2 - \frac{u_{xxx}(x_i)}{3!}h^3 \\
&\quad + \frac{u_{xxxx}(x_i)}{4!}h^4 - \frac{u_{xxxxx}(x_i)}{5!}h^5 + \cdots.
\end{aligned} \tag{1.12}$$

One-sided FD operators

Solve the above equations for $u_x(x_i)$ to have

$$\begin{aligned}
u_x(x_i) &= \frac{u_{i+1} - u_i}{h} - \frac{u_{xx}(x_i)}{2!}h - \frac{u_{xxx}(x_i)}{3!}h^2 \\
&\quad - \frac{u_{xxxx}(x_i)}{4!}h^3 + \cdots, \\
u_x(x_i) &= \frac{u_i - u_{i-1}}{h} + \frac{u_{xx}(x_i)}{2!}h - \frac{u_{xxx}(x_i)}{3!}h^2 \\
&\quad + \frac{u_{xxxx}(x_i)}{4!}h^3 - \cdots.
\end{aligned} \tag{1.13}$$

By truncating the terms including h^k, $k = 1, 2, \cdots$, we define the first-order FD schemes

$$\begin{aligned} u_x(x_i) &\approx D_x^+ u_i := \frac{u_{i+1} - u_i}{h}, \quad \text{(forward)} \\ u_x(x_i) &\approx D_x^- u_i := \frac{u_i - u_{i-1}}{h}, \quad \text{(backward)} \end{aligned} \tag{1.14}$$

where D_x^+ and D_x^- are called the **forward difference operator** and **backward difference operator**, respectively.

Central FD operators

The central second-order FD scheme for u_x: Subtract (1.12.b) from (1.12.a) and divide the resulting equation by $2h$.

$$u_x(x_i) = \frac{u_{i+1} - u_{i-1}}{2h} - \frac{u_{xxx}(x_i)}{3!}h^2 - \frac{u_{xxxxx}(x_i)}{5!}h^4 - \cdots . \tag{1.15}$$

Thus the central second-order FD scheme reads

$$u_x(x_i) \approx D_x^1 u_i := \frac{u_{i+1} - u_{i-1}}{2h}. \quad \text{(central)} \tag{1.16}$$

Note that the central difference operator D_x^1 is the average of the forward and backward operators, i.e.,

$$D_x^1 = \frac{D_x^+ + D_x^-}{2}.$$

The FD scheme for $u_{xx}(x_i)$: Add the two equations in (1.12) and divide the resulting equation by h^2.

$$u_{xx}(x_i) = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} - 2\frac{u_{xxxx}(x_i)}{4!}h^2 - 2\frac{u_{xxxxxx}(x_i)}{6!}h^4 - \cdots . \tag{1.17}$$

Thus the central second-order FD scheme for u_{xx} at x_i reads

$$u_{xx}(x_i) \approx D_x^2 u_i := \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}. \tag{1.18}$$

Note that

$$D_x^2 = D_x^- D_x^+ = D_x^+ D_x^-. \tag{1.19}$$

1.2.2. General Grids

Taylor series approaches

For $\{a = x_0 < x_1 < \cdots < x_N = b\}$, a partition of the interval $[a, b]$, let

$$h_i = x_i - x_{i-1}, \quad i = 1, 2, \cdots, N.$$

Recall: The Taylor series expansions for u_{i+1} and u_{i-1} (about x_i) read

$$\begin{aligned} \text{(a)} \quad u_{i+1} &= u_i + u_x(x_i)h_{i+1} + \frac{u_{xx}(x_i)}{2!}h_{i+1}^2 + \frac{u_{xxx}(x_i)}{3!}h_{i+1}^3 + \cdots, \\ \text{(b)} \quad u_{i-1} &= u_i - u_x(x_i)h_i + \frac{u_{xx}(x_i)}{2!}h_i^2 - \frac{u_{xxx}(x_i)}{3!}h_i^3 + \cdots. \end{aligned} \tag{1.20}$$

which correspond to (1.12).

The second-order FD scheme for u_x

Multiply (1.20.b) by r_i^2 $(:= (h_{i+1}/h_i)^2)$ and subtract the resulting equation from (1.20.a) to have

$$\begin{aligned} u_x(x_i) &= \frac{u_{i+1} - (1 - r_i^2)u_i - r_i^2 u_{i-1}}{h_{i+1} + r_i^2 h_i} - \frac{h_{i+1}^3 + r_i^2 h_i^3}{6(h_{i+1} + r_i^2 h_i)}u_{xxx}(x_i) - \cdots \\ &= \frac{h_i^2 u_{i+1} + (h_{i+1}^2 - h_i^2)u_i - h_{i+1}^2 u_{i-1}}{h_i h_{i+1}(h_i + h_{i+1})} - \frac{h_i h_{i+1}}{6}u_{xxx}(x_i) - \cdots. \end{aligned}$$

Thus the second-order approximation for $u_x(x_i)$ becomes

$$u_x(x_i) \approx \frac{h_i^2 u_{i+1} + (h_{i+1}^2 - h_i^2)u_i - h_{i+1}^2 u_{i-1}}{h_i h_{i+1}(h_i + h_{i+1})}. \tag{1.21}$$

Note: It is relatively easy to find the second-order FD scheme for u_x in nonuniform grids, as just shown, using the Taylor series approach. However, for higher-order schemes, it requires a tedious work for the derivation. The curve fitting approached can be applied for the approximation of both u_x and u_{xx} more conveniently.

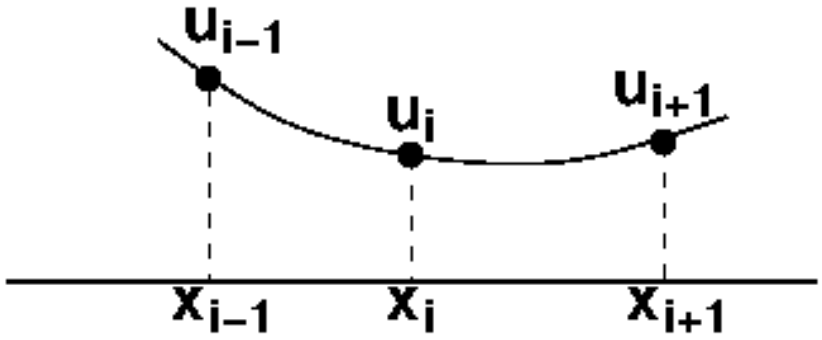


Figure 1.2: The curve fitting by the interpolating quadratic polynomial.

Curve fitting approaches

An alternative way of obtaining FD approximations is to

- fit the function to an interpolating polynomial &
- differentiate the resulting polynomial.

For example, the quadratic polynomial that interpolates u at $\{x_{i-1}, x_i, x_{i+1}\}$ can be constructed as (see Figure 1.2)

$$p_2(x) = a_0 + a_1(x - x_{i-1}) + a_2(x - x_{i-1})(x - x_i), \tag{1.22}$$

where the coefficients a_k, $k = 0, 1, 2$, are determined by e.g. the divided differences:

$$\begin{aligned} a_0 &= u_{i-1}, \quad a_1 = \frac{u_i - u_{i-1}}{h_i}, \\ a_2 &= \frac{h_i(u_{i+1} - u_i) - h_{i+1}(u_i - u_{i-1})}{h_i h_{i+1}(h_i + h_{i+1})}. \end{aligned}$$

Thus

$$\begin{aligned} u_x(x_i) &\approx p_2'(x_i) = a_1 + a_2 h_i \\ &= \frac{h_i^2 u_{i+1} + (h_{i+1}^2 - h_i^2)u_i - h_{i+1}^2 u_{i-1}}{h_i h_{i+1}(h_i + h_{i+1})}, \end{aligned} \tag{1.23}$$

which is second-order and identical to (1.21).

Higher-order FDs for $u_x(x_i)$

- For higher-order approximations for $u_x(x_i)$, the function must be fit to higher-degree polynomials that interpolate u at a larger set of grid points including x_i.
- For a fourth-order approximation, for example, we should construct a fourth-degree polynomial.
  - Let $p_{i-2,4}(x)$ be the fourth-order Newton polynomial that interpolates u at $\{x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$, i.e.,

$$p_{i-2,4}(x) = \sum_{k=0}^{4} \left[a_{i-2,k} \prod_{j=0}^{k-1} (x - x_{i-2+j}) \right], \tag{1.24}$$

    where

$$a_{i-2,k} = u[x_{i-2}, x_{i-1}, \cdots, x_{i-2+k}], \quad k = 0, \cdots, 4.$$

  - Then it follows from the Interpolation Error Theorem (1.10) that

$$u_x(x_i) = p'_{i-2,4}(x_i)$$
$$+\frac{u^{(5)}(\xi)}{5!}(x_i - x_{i-2})(x_i - x_{i-1})(x_i - x_{i+1})(x_i - x_{i+2}).$$

  - Therefore, under the assumption that $u^{(5)}(x)$ exists, $p'_{i-2,4}(x_i)$ approximates $u_x(x_i)$ with a fourth-order truncation error.

FDs for $u_{xx}(x_i)$

- The second-derivative u_{xx} can be approximated by differentiating the interpolating polynomial twice. For example, from p_2 in (1.22), we have

$$\begin{aligned} u_{xx}(x_i) \approx p_2''(x_i) &= 2\frac{h_i(u_{i+1}-u_i)-h_{i+1}(u_i-u_{i-1})}{h_ih_{i+1}(h_i+h_{i+1})} \\ &= \frac{h_{i+1}u_{i-1}-(h_i+h_{i+1})u_i+h_iu_{i+1}}{\frac{1}{2}h_ih_{i+1}(h_i+h_{i+1})}. \end{aligned} \tag{1.25}$$

  The above approximation has a ***first-order accuracy for general grids.*** However, it turns out to be second-order accurate when $h_i = h_{i+1}$; compare it with the one in (1.18).

- A higher-order FD scheme for u_{xx} can be obtained from the twice differentiation of $p_{i-2,4}$ in (1.24):

$$u_{xx}(x_i) \approx p_{i-2,4}''(x_i), \tag{1.26}$$

  which is a third-order approximation and becomes fourth-order for uniform grids.

- The thumb of rule is to utilize higher-order interpolating polynomials for higher-order FD approximations.

1.3. Overview of PDEs

Parabolic Equations

- The one-dimensional (1D) differential equation

$$u_t - \alpha^2 u_{xx} = f(x,t), \quad x \in (0,L), \tag{1.27}$$

  is a standard 1D parabolic equation, which is often called the **heat/diffusion equation**.
- The equation models many physical phenomena such as ***heat distribution on a rod***: $u(x,t)$ ***represents the temperature at the position*** x ***and time*** t, α^2 is the **thermal diffusivity** of the material, and $f(x,t)$ denotes a source/sink along the rod.
- When the material property is **nonuniform** along the rod, the coefficient α is a function of x. In this case, the thermal conductivity K depends on the position x and the heat equation becomes

$$u_t - \nabla \cdot (K(x)\, u_x)_x = f(x,t). \tag{1.28}$$

Note: To make the heat equation *well-posed* (existence, uniqueness, and stability), we have to supply an initial condition and appropriate boundary conditions on the both ends of the rod.

Heat equation in 2D/3D

In 2D or 3D, the heat equations can be formulated as

$$\begin{aligned} &u_t - \nabla \cdot (K \nabla u) = f, && (\mathbf{x},t) \in \Omega \times [0,J] && \\ &\quad u(\mathbf{x},0) = u_0(\mathbf{x}), && \mathbf{x} \in \Omega && \text{(IC)} \\ &\quad u(\mathbf{x},t) = g(\mathbf{x},t), && (\mathbf{x},t) \in \Gamma \times [0,J] && \text{(BC)} \end{aligned} \tag{1.29}$$

where $\Gamma = \partial\Omega$, the boundary of Ω.

Hyperbolic Equations

The **first-order hyperbolic differential equation** is formulated as

$$u_t + au_x = f, \tag{1.30}$$

which represents the **convection** (movement) of certain materials. Here a denotes the velocity.

The **second-order hyperbolic differential equation**

$$\frac{1}{v^2}u_{tt} - u_{xx} = f(x,t), \quad x \in (0, L) \tag{1.31}$$

is often called the wave equation. The coefficient v is the wave velocity, while f represents a source. The equation can be used to describe the vibration of a flexible string, for which u denotes the displacement of the string.

In higher dimensions, the wave equation can be formulated similarly.

Elliptic Equations

The second-order elliptic equations are obtained as the steady-state solutions (as $t \to \infty$) of the parabolic and hyperbolic equations. For example,

$$\begin{aligned} -\nabla \cdot (K\nabla u) &= f, \ \ \mathbf{x} \in \Omega \\ u(\mathbf{x}) &= g(\mathbf{x}), \ \ \mathbf{x} \in \Gamma \end{aligned} \tag{1.32}$$

represents a steady-state heat distribution for the given heat source f and the boundary condition g.

Fluid Mechanics

The 2D **Navier-Stokes (NS) equations** for viscous **incompressible fluid flows**:

Momentum equations

$$\begin{aligned} &u_t + p_x - \tfrac{1}{R}\Delta u + (u^2)_x + (uv)_y = g_1 \\ &v_t + p_y - \tfrac{1}{R}\Delta v + (uv)_x + (v^2)_y = g_2 \end{aligned} \tag{1.33}$$

Continuity equation

$$u_x + v_y = 0$$

Here (u, v) denote the velocity fields in (x, y)-directions, respectively, p is the pressure, R is the (dimensionless) *Reynolds number*, and (g_1, g_2) are body forces. See e.g. [23] for computational methods for fluid dynamics.

Finance Modeling

In option pricing, the most popular model is the **Black-Scholes (BS) differential equation**

$$u_t + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial S^2} + rS\frac{\partial u}{\partial S} - ru = 0 \tag{1.34}$$

Here

- $S(t)$ is the stock price at time t
- $u = u(S(t), t)$ denotes the price of an option on the stock
- σ is the volatility of the stock
- r is the (risk-free) interest rate

Note that the BS model is a backward parabolic equation, which needs a final condition at time T. For European calls, for example, we have the condition

$$u(S, T) = \max(S - X, 0),$$

while for a put option, the condition reads

$$u(S, T) = \max(X - S, 0),$$

where X is the exercise price at the expiration date T.

- Call option: the right to buy the stock
- Put option: the right to sell the stock

Image Processing: A PDE Application

- As higher reliability and efficiency are required, PDE-based mathematical techniques have become important components of many research and processing areas, including image processing.
- PDE-based methods have been applied for various image processing tasks such as image denoising, interpolation, inpainting, segmentation, and object detection.

Example: Image denoising

- **Noise model:**

$$f = u + \eta \tag{1.35}$$

where f is the observed (noisy) image, u denotes the desired image, and η is the noise.

- **Optimization problem:**
Minimize the total variation (TV) with the constraint

$$\min_u \int_\Omega |\nabla u| d\mathbf{x} \quad \text{subj. to} \|f - u\|^2 = \sigma^2. \tag{1.36}$$

Using a Lagrange multiplier, the above minimization problem can be rewritten as

$$\min_u \Big(\int_\Omega |\nabla u| d\mathbf{x} + \frac{\lambda}{2} \int_\Omega (f-u)^2 d\mathbf{x} \Big), \tag{1.37}$$

from which employing **variational calculus**, we can derive the corresponding **Euler-Lagrange equation**

$$-\nabla \cdot \Big(\frac{\nabla u}{|\nabla u|} \Big) = \lambda (f - u), \tag{1.38}$$

which is called the **total variation model** in image denoising [58].

Remark 1.5.

- Many other image processing tasks (such as interpolation and inpainting) can be considered as “generalized denoising." For example, the main issue in interpolation is to remove or significantly reduce artifacts of traditional interpolation methods, and the artifacts can be viewed as noise [8, 34].
- Variants of the TV model can be applied for various image processing tasks.

Numerical methods for PDEs

- **Finite difference method**: Simple, easiest technique. It becomes quite complex for irregular domains
- **Finite element method**: Most popular, due to most flexible over complex domains
- **Finite volume method**: Very popular in computational fluid dynamics (CFD).
  - Surface integral over control volumes
  - Locally conservative
- **Spectral method**: Powerful if the domain is simple and the solution is smooth.
- **Boundary element method**: Useful for PDEs which can be formulated as integral equations; it solves the problem on the boundary to find the solution over the whole domain.
  - The algebraic system is often full
  - Not many problems can be written as integral equations, for example, nonlinear equations.
- **Meshless/mesh-free method**: Developed to overcome drawbacks of meshing and re-meshing, for example, in crack propagation problems and large deformation simulations.

1.4. Calculus of Variations: Variational Calculus

- Consider the following minimization problem

$$\min_u \int_\Omega |\nabla u|^p d\mathbf{x} \quad \text{subj. to} \quad \|f-u\| = \sigma, \tag{1.39}$$

  where f is the given image, u denotes a desired image to be restored, and $\sigma > 0$ is a (small) constant.

- Introducing a **Lagrange multiplier** λ, the above problem can be equivalently written as

$$\min_u \left\{ \int_\Omega |\nabla u|^p d\mathbf{x} + \frac{\lambda}{2}\|f-u\|^2 \right\}. \tag{1.40}$$

- Employing **calculus of variations**, we can rewrite it as

$$u_t = \begin{cases} 2\Delta u + \lambda(f-u), & p=2, \\ \nabla \cdot \left(\dfrac{\nabla u}{|\nabla u|}\right) + \lambda(f-u), & p=1. \end{cases} \tag{1.41}$$

In this section, we will study the mathematical backgrounds for such derivation.

Derivation

For simplicity, we will derive it for problems in one variable. Consider

$$J = \int_{x_1}^{x_2} g(x,u,u')dx. \tag{1.42}$$

Here the question is: *What is the function u that satisfies*

$$u(x_1) = u_1 \ \textit{and} \ u(x_2) = u_2$$

and renders the integral (1.42) a minimum?

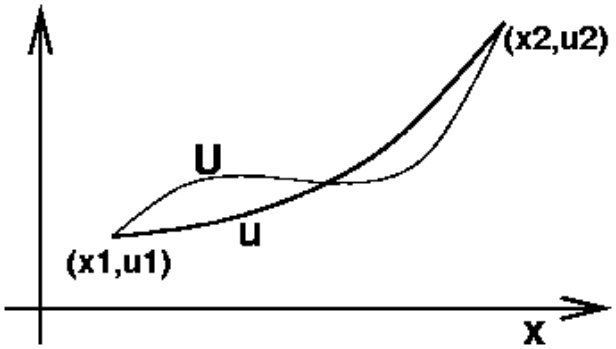


Figure 1.3: The minimizer u and its variation U as a comparison function.

The answer can be found as follows:

1. Let u be the minimizing function of (1.42).
2. Consider the one-parameter family of **comparison functions**:

$$U(x) = u(x) + \varepsilon\eta(x), \tag{1.43}$$

   where η is an arbitrary differentiable function such that

$$\eta(x_1) = \eta(x_2) = 0.$$

   See Figure 1.3.
3. Note

$$U'(x) = u'(x) + \varepsilon\eta'(x).$$

4. Consider

$$J(\varepsilon) = \int_{x_1}^{x_2} g(x, U, U')dx.$$

   Then, since $U = u$ when $\varepsilon = 0$, we have

$$J'(0) = 0. \tag{1.44}$$

Implication of $J'(0) = 0$

- Differentiating J with respect to ε reads

$$\begin{aligned} \frac{dJ}{d\varepsilon} = J'(\varepsilon) &= \int_{x_1}^{x_2} \Big(\frac{\partial g}{\partial U} \frac{\partial U}{\partial \varepsilon} + \frac{\partial g}{\partial U'} \frac{\partial U'}{\partial \varepsilon} \Big) dx \\ &= \int_{x_1}^{x_2} \Big(\frac{\partial g}{\partial U} \eta + \frac{\partial g}{\partial U'} \eta' \Big) dx. \end{aligned}$$

- Thus,

$$J'(0) = \int_{x_1}^{x_2} \Big(\frac{\partial g}{\partial u} \eta + \frac{\partial g}{\partial u'} \eta' \Big) dx = 0. \tag{1.45}$$

- Now, apply **integration by parts** to get

$$\begin{aligned} J'(0) &= \int_{x_1}^{x_2} \frac{\partial g}{\partial u} \eta + \frac{\partial g}{\partial u'} \eta \Big|_{x_1}^{x_2} - \int_{x_1}^{x_2} \frac{d}{dx} \Big(\frac{\partial g}{\partial u'} \Big) \eta \, dx \\ &= \int_{x_1}^{x_2} \left[\frac{\partial g}{\partial u} - \frac{d}{dx} \Big(\frac{\partial g}{\partial u'} \Big) \right] \eta \, dx = 0. \end{aligned} \tag{1.46}$$

  Since the above holds for all η, we must have

$$\frac{\partial g}{\partial u} - \frac{d}{dx} \Big(\frac{\partial g}{\partial u'} \Big) = 0, \tag{1.47}$$

  which is called the **Euler-Lagrange equation**.

- Utilizing the same arguments, one can get the Euler-Lagrange equation for minimization problems in n independent variables:

$$\Big(\frac{\partial}{\partial u} - \sum_{i=1}^{n} \frac{d}{dx_i} \frac{\partial}{\partial u_{x_i}} \Big) g = 0, \tag{1.48}$$

  where $u_{x_i} = \partial u / \partial x_i$.

Note: Certain optimization problems can be cast as PDEs, which facilitates the design of efficient computational algorithms.

See Homework 1.2, p. 23, for the PDE representation of total variation.

1.5. Difference Equations

This section covers solution methods and **stability analysis** for difference equations as an introductory exercise.

Example 1.6. Find a general form for y_n by solving the recurrence relation

$$\begin{aligned} &2y_{n+2} - 5y_{n+1} + 2y_n = 0 \\ &y_0 = 2, \;\; y_1 = 1 \end{aligned} \tag{1.49}$$

Solution. Let

$$y_n = \alpha^n. \tag{1.50}$$

and plug it into the first equation of (1.49) to have

$$2\alpha^{n+2} - 5\alpha^{n+1} + 2\alpha^n = 0,$$

which implies

$$2\alpha^2 - 5\alpha + 2 = 0. \tag{1.51}$$

The last equation is called the **characteristic equation** of the difference equation (1.49), of which the two roots are

$$\alpha = 2, \;\; \frac{1}{2}.$$

Thus, the general solution of the difference equation reads

$$y_n = c_1\, 2^n + c_2 \left(\frac{1}{2}\right)^n, \tag{1.52}$$

where c_1 and c_2 are constants. One can determine the constants using the initial conditions in (1.49).

$$y_0 = c_1 + c_2 = 2, \quad y_1 = 2\, c_1 + \frac{c_2}{2} = 1$$

which implies

$$c_1 = 0, \quad c_2 = 2. \tag{1.53}$$

What we have found is that

$$y_n = 2\left(\frac{1}{2}\right)^n = 2^{1-n}. \tag{1.54}$$

A small change in the initial conditions

Example 1.7. Now, consider the difference equation in Example 1.6, with a little bit different initial conditions:

$$\begin{aligned} &2w_{n+2} - 5w_{n+1} + 2w_n = 0 \\ &w_0 = 2, \quad w_1 = 1.01 \end{aligned} \tag{1.55}$$

Solution. The difference equation has the general solution of the form as in (1.52):

$$w_n = c_1\, 2^n + c_2 \left(\frac{1}{2}\right)^n. \tag{1.56}$$

Using the new initial conditions, we have

$$w_0 = c_1 + c_2 = 2, \quad w_1 = 2\, c_1 + \frac{c_2}{2} = 1.01,$$

Thus, the solution becomes

$$w_n = \frac{1}{150}\, 2^n + \frac{299}{150} \left(\frac{1}{2}\right)^n. \tag{1.57}$$

Comparison

y_0	$=$	2	w_0	$=$	2
y_1	$=$	1	w_1	$=$	1.01
	$\vdots$			$\vdots$	
y_{10}	$=$	9.7656×10^{-4}	w_{10}	$=$	6.8286
y_{20}	$=$	9.5367×10^{-7}	w_{20}	$=$	6.9905×10^{3}

Thus, the difference equation in (1.49) or (1.55) is *unstable*.

Stability Theory

Physical Definition: A (FD) scheme is **stable** if a small change in the initial conditions produces a small change in the state of the system.

- Most aspects in nature are stable.
- Some phenomena in nature can be represented by differential equations (ODEs and PDEs), while they may be solved through difference equations.
- Although ODEs and PDEs are stable, their approximations (finite difference equations) may not be stable. In this case, the approximation is a failure.

Definition 1.8. A differential equation is

- **stable** if for every set of initial data, the solution remains bounded as $t \to \infty$.
- **strongly stable** if the solution approaches zero as $t \to \infty$.

Stability of difference equations

Theorem 1.9. *A finite difference equation is stable if and only if*

(a) $|\alpha| \leq 1$ *for all roots of the characteristic equation, and*

(b) if $|\alpha| = 1$ *for some root, then the root is simple.*

Theorem 1.10. *A finite difference equation is strongly stable if and only if* $|\alpha| < 1$ *for all roots of the characteristic equation.*

1.6. Homework

1. For an interval $[a, b]$, let the grid be uniform:

$$x_i = ih + a; \quad i = 0, 1, \cdots, N, \quad h = \frac{b-a}{N}. \tag{1.58}$$

Second-order schemes for u_x and u_{xx}, on the uniform grid given as in (1.58), respectively read

$$\begin{aligned} u_x(x_i) &\approx D_x^1 u_i = \frac{u_{i+1} - u_{i-1}}{2h}, \\ u_{xx}(x_i) &\approx D_x^2 u_i = D_x^+ D_x^- u_i = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}. \end{aligned} \tag{1.59}$$

   (a) Use Divided Differences to construct the second-order Newton polynomial $p_2(x)$ which passes (x_{i-1}, u_{i-1}), (x_i, u_i), and (x_{i+1}, u_{i+1}).

   (b) Evaluate $p_2'(x_i)$ and $p_2''(x_i)$ to compare with the FD schemes in (1.59).

2. For the minimization problem (1.40) in 2D, the functional J is expressed as ($p = 1$)

$$J = \int_\Omega |\nabla u| d\mathbf{x} + \frac{\lambda}{2}\|f - u\|^2 = \int_\Omega \left(\sqrt{u_x^2 + u_y^2} + \frac{\lambda}{2}(f-u)^2\right) d\mathbf{x}. \tag{1.60}$$

Employ variational calculus to show that its minimizer u satisfies the following PDE:

$$\nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) + \lambda(f - u) = 0. \tag{1.61}$$

3. Find the general solution of each of the following difference equations:

   (a) $y_{n+1} = 3y_n$

   (b) $y_{n+1} = 3y_n + 2$

   (c) $y_{n+2} - 8y_{n+1} + 12y_n = 0$

   (d) $y_{n+2} - 6y_{n+1} + 9y_n = 1$

4. Determine, for each of the following difference equations, whether it is stable or unstable.

   (a) $y_{n+2} - 5y_{n+1} + 6y_n = 0$

(b) $8y_{n+2} + 2y_{n+1} - 3y_n = 0$

(c) $3y_{n+2} + y_n = 0$

(d) $4y_{n+4} + 5y_{n+2} + y_n = 0$

CHAPTER 2

Numerical Solution of Ordinary Differential Equations

The first-order initial value problem (IVP) is formulated as follows: find $\{y_i(x) : i = 1, 2, \cdots, M\}$ satisfying

$$\begin{aligned} \frac{dy_i}{dx} &= f_i(x, y_1, y_2, \cdots, y_M), \quad i = 1, 2, \cdots, M, \\ y_i(x_0) &= y_{i0}, \end{aligned} \tag{2.1}$$

for a prescribed initial values $\{y_{i0} : i = 1, 2, \cdots, M\}$.

We assume that (2.1) admits a unique solution in a neighborhood of x_0.

For simplicity, we consider the case $M = 1$:

$$\begin{aligned} \frac{dy}{dx} &= f(x, y), \\ y(x_0) &= y_0. \end{aligned} \tag{2.2}$$

It is known that if f and $\partial f/\partial y$ are continuous in a strip $(a, b) \times \mathbb{R}$ containing (x_0, y_0), then (2.2) has a unique solution in an interval I, where $x_0 \in I \subset (a, b)$.

Contents of Chapter 2

2.1. Elementary Theory of Initial-Value Problems 26
2.2. Taylor-Series Methods 28
2.3. Runge-Kutta Methods 37
2.4. One-Step Methods: Accuracy Comparison 43
2.5. Multi-step Methods 46
2.6. High-Order Equations & Systems of Differential Equations 50
2.7. Implicit Methods and Implicit Differential Equations 57
2.8. Homework 63

2.1. Elementary Theory of Initial-Value Problems

Definition 2.1. The first-order **initial value problem (IVP)** is formulated as follows: find $\{y_i(x) : i = 1, 2, \cdots, M\}$ satisfying

$$\begin{aligned} \frac{dy_i}{dx} &= f_i(x, y_1, y_2, \cdots, y_M), \quad i = 1, 2, \cdots, M, \\ y_i(x_0) &= y_{i0}, \end{aligned} \tag{2.3}$$

for a prescribed initial values $\{y_{i0} : i = 1, 2, \cdots, M\}$.

- We assume that (2.3) admits a unique solution in a neighborhood of x_0.
- For simplicity, we consider the case $M = 1$:

$$\begin{aligned} \frac{dy}{dx} &= f(x, y), \\ y(x_0) &= y_0. \end{aligned} \tag{2.4}$$

Theorem 2.2. (Existence and Uniqueness of the Solution): *Suppose that $R = \{(x, y) \mid a \le x \le b, -\infty < y < \infty\}$, f is continuous on R, and $x_0 \in [a, b]$. If f satisfies a Lipschitz condition on R in the variable y, i.e., there is a constant $L > 0$ such that*

$$|f(x, y_1) - f(x, y_2)| \le L|y_1 - y_2|, \quad \forall\, y_1, y_2, \tag{2.5}$$

then the IVP (2.4) has a unique solution $y(x)$ in an interval I, where $x_0 \in I \subset (a, b)$.

Theorem 2.3. *Suppose that $f(x, y)$ is defined on $R \subset \mathbb{R}^2$. If a constant $L > 0$ exists with*

$$\left|\frac{\partial f(x, y)}{\partial y}\right| \le L, \quad \forall\, (x, y) \in R, \tag{2.6}$$

then f satisfies a Lipschitz condition on R in the variable y with the same Lipschitz constant L.

Example 2.4. Show that each of the initial-value problems has a unique solution and find the solution.

(a) $y' = e^{x-y}, \quad 0 \le x \le 1, \quad y(0) = 1$

(b) $y' = (1 + x^2)y, \quad 3 \le x \le 5, \quad y(3) = 1$

Solution. *(Existence and uniqueness)*:

(Find the solution): Here, we will find the solution for (b), using Maple.

Maple-code

```
DE := diff(y(x), x) = y(x)*(x^2 + 1);
                       d              / 2    \
                      --- y(x) = y(x) \x  + 1/
                       dx
IC := y(3) = 1;
                                y(3) = 1
dsolve({DE, IC}, y(x));
                                  /1   / 2    \\
                                exp|- x \x  + 3/|
                                  \3           /
                      y(x) = -----------------
                                    exp(12)
```

Strategy 2.5. (Numerical Solution): In the following, we describe **step-by-step methods** for (2.4); that is, we start from $y_0 = y(x_0)$ and proceed stepwise.

- In the first step, we compute y_1 which approximate the solution y of (2.4) at $x = x_1 = x_0 + h$, where h is the step size.
- The second step computes an approximate value y_2 of the solution at $x = x_2 = x_0 + 2h$, etc..

Note: We first introduce the Taylor-series methods for (2.4), followed by Runge-Kutta methods and multi-step methods. All of these methods are applicable straightforwardly to (2.3).

2.2. Taylor-Series Methods

Preliminary **2.6.** Here we rewrite the initial value problem (IVP):

$$\left\{ \begin{array}{rcl} y' & = & f(x,y), \\ y(x_0) & = & y_0. \end{array} \right. \quad \text{(IVP)} \tag{2.7}$$

For the problem, a continuous approximation to the solution $y(x)$ will not be obtained; instead, approximations to y will be generated at various points, called **mesh points** in the interval $[x_0, T]$, for some $T > x_0$. Let

- $h = (T - x_0)/N_t$, for an integer $N_t \geq 1$
- $x_n = x_0 + nh$, $n = 0, 1, 2, \cdots, N_t$
- y_n be the approximate solution of y at x_n, i.e., $y_n \approx y(x_n)$.

2.2.1. The Euler Method

Step 1

- It is to find an approximation of $y(x_1)$, marching through the first subinterval $[x_0, x_1]$ and using a Taylor-series involving only up to the first-derivative of y.
- Consider the *Taylor series*

$$y(x_1) \;=\; y(x_0 + h) \;=\; y(x_0) + hy'(x_0) + \frac{h^2}{2} y''(x_0) + \cdots . \tag{2.8}$$

- Utilizing $y(x_0) = y_0$ and $y'(x_0) = f(x_0, y_0)$, the value $y(x_1)$ can be approximated by

$$y_1 = y_0 + h\, f(x_0, y_0), \tag{2.9}$$

where the second- and higher-order terms of h are ignored.

Such an idea can be applied recursively for the computation of solution on later subintervals. Indeed, since

$$y(x_2) = y(x_1) + hy'(x_1) + \frac{h^2}{2}y''(x_1) + \cdots,$$

by replacing $y(x_1)$ and $y'(x_1)$ with y_1 and $f(x_1, y_1)$, respectively, we obtain

$$y_2 = y_1 + h\, f(x_1, y_1), \tag{2.10}$$

which approximates the solution at $x_2 = x_0 + 2h$.

Algorithm 2.7. Summarizing the above, the **Euler method** solving the first-order IVP (2.7) is formulated as

$$y_{n+1} = y_n + h\, f(x_n, y_n), \quad n \geq 0. \tag{2.11}$$

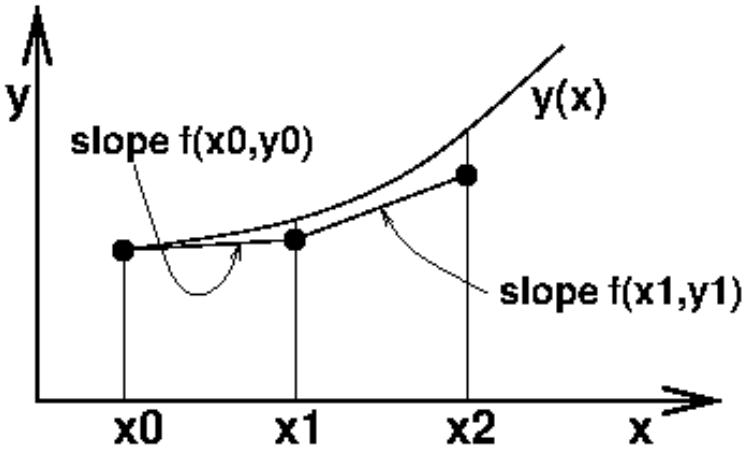


Figure 2.1: The Euler method.

Geometrically it is an approximation of the curve $\{x, y(x)\}$ by a polygon of which **the first segment is tangent to the curve at x_0**, as shown in the above figure.

- For example, y_1 is determined by moving the point (x_0, y_0) by the length of h with the slope $f(x_0, y_0)$, **the evaluation of f ad the left edge point**.

Algegraic Interpretation of the Euler Method

Remark 2.8. Applying the **Fundamental Theorem of Calculus** (Part 1), the solution for the IVP (2.7) can be *implicitly* formulated as

$$y(x) \;=\; y_0 + \int_{x_0}^{x} f(t, y(t))\, dt. \tag{2.12}$$

- **Check**:

$$y'(x) = f(x, y(x)) \;\text{ and }\; y(x_0) = y_0 + \int_{x_0}^{x_0} f(t, y(t))\, dt = y_0.$$

- When $x = x_1$, Equation (2.12) reads

$$y(x_1) \;=\; y_0 + \int_{x_0}^{x_1} f(t, y(t))\, dt. \tag{2.13}$$

- Let us **approximate the integral using the left edge value**:

$$\int_{x_0}^{x_1} f(t, y(t))\, dt \approx \int_{x_0}^{x_1} f(x_0, y(x_0))\, dt = (x_1 - x_0) f(x_0, y_0). \tag{2.14}$$

- Now, it follows from (2.12) and (2.14) that

$$y(x_1) \;\approx\; y_0 + h f(x_0, y_0) \quad (=: \mathbf{y_1}), \tag{2.15}$$

which represents the Euler method. □

Convergence of the Euler Method

Theorem 2.9. *Let f satisfy the Lipschitz condition in its second variable, i.e., there is $\lambda > 0$ such that*

$$|f(x, y_1) - f(x, y_2)| \le \lambda |y_1 - y_2|, \quad \forall\, y_1, y_2. \tag{2.16}$$

Then, the Euler method is convergent; more precisely,

$$|y_n - y(x_n)| \le \frac{C}{\lambda} h[(1 + \lambda h)^n - 1], \quad n = 0, 1, 2, \cdots, N_t. \tag{2.17}$$

Note: The term $(1 + \lambda h)^n$ is bounded by a constant:

$$(1 + \lambda h)^n \le (1 + \lambda h)^{N_t} \le e^{\lambda (T - x_0)}, \quad n = 0, 1, 2, \cdots, N_t. \tag{2.18}$$

See Exercise 2.2.2.

Proof. The true solution y satisfies

$$y(x_{n+1}) = y(x_n) + h f(x_n, y(x_n)) + \mathcal{O}(h^2). \tag{2.19}$$

Thus it follows from (2.11) and (2.19) that

$$\begin{aligned} e_{n+1} &= e_n + h[f(x_n, y_n) - f(x_n, y(x_n))] + \mathcal{O}(h^2) \\ &= e_n + h[f(x_n, y(x_n) + e_n) - f(x_n, y(x_n))] + \mathcal{O}(h^2), \end{aligned}$$

where $e_n = y_n - y(x_n)$. Utilizing (2.16), we have

$$|e_{n+1}| \le (1 + \lambda h)|e_n| + Ch^2. \tag{2.20}$$

Here we will prove (2.17) by using (2.20) and induction. It holds trivially when $n = 0$. Suppose it holds for n. Then,

$$\begin{aligned} |e_{n+1}| &\le (1 + \lambda h)|e_n| + Ch^2 \\ &\le (1 + \lambda h) \cdot \frac{C}{\lambda} h[(1 + \lambda h)^n - 1] + Ch^2 \\ &= \frac{C}{\lambda} h[(1 + \lambda h)^{n+1} - (1 + \lambda h)] + Ch^2 \\ &= \frac{C}{\lambda} h[(1 + \lambda h)^{n+1} - 1], \end{aligned}$$

which completes the proof. □

Example 2.10. Consider

$$\begin{aligned} y' &= y - x^3 + x + 1, \quad 0 \le x \le 3, \\ y(0) &= 0.5. \end{aligned} \tag{2.21}$$

As the step lengths become smaller, $h = 1 \to \frac{1}{4} \to \frac{1}{16}$, the numerical solutions represent the exact solution better, as shown in the following figures:

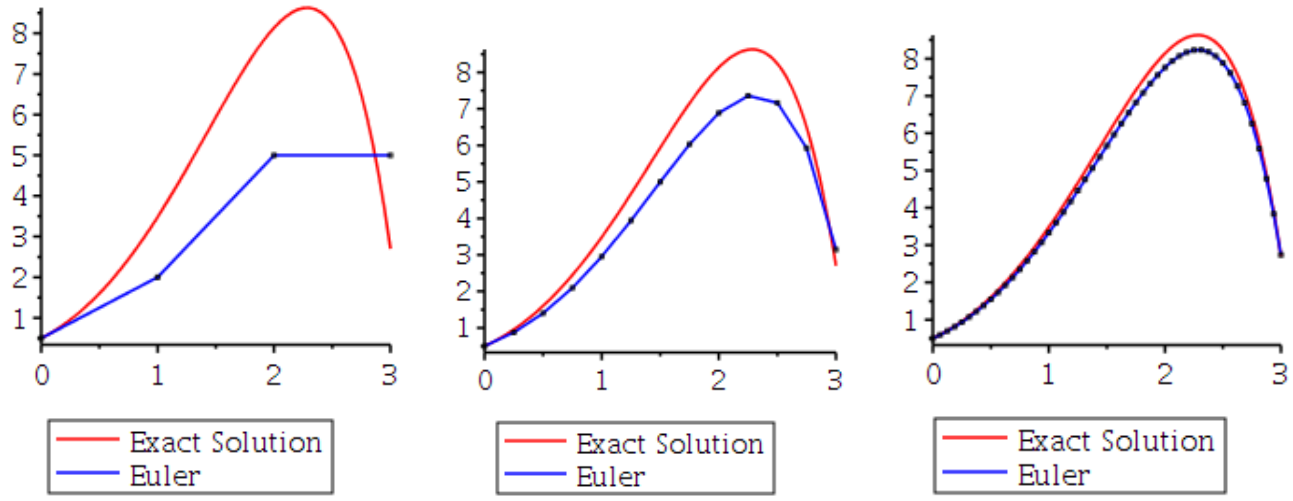


Figure 2.2: The Euler method, with $h = 1 \to \frac{1}{4} \to \frac{1}{16}$.

Example 2.11. Implement a code for the Euler method to solve

$$y' = y - x^3 + x + 1, \quad 0 \le x \le 3, \quad y(0) = 0.5, \quad \text{with } h = \frac{1}{16}.$$

Solution.

Euler.mw

```
Euler := proc(f, x0, b, nx, y0, Y)
    local h, t, w, n:
    h := (b - x0)/nx:
    t := x0; w := y0:
    Y[0] := w:
    for n to nx do
        w := w + h*eval(f, [x = t, y = w]);
        Y[n] := w;
        t := t + h;
    end do:
end proc:

# Now, solve it using "Euler"
f := -x^3 + x + y + 1:
x0 := 0: b := 3: y0 := 0.5:

nx := 48:
YEuler := Array(0..nx):
Euler(f, x0, b, nx, y0, YEuler):

# Check the maximum error
DE := diff(y(x), x) = y(x) - x^3 + x + 1:
dsolve([DE, y(x0) = y0], y(x))
                                    2    3   7
             y(x) = 4 + 5 x + 3 x  + x  - - exp(x)
                                         2
exacty := x -> 4 + 5*x + 3*x^2 + x^3 - 7/2*exp(x):
maxerr := 0:
h := (b - x0)/nx:
for n from 0 to nx do
    maxerr := max(maxerr,abs(exacty(n*h)-YEuler[n]));
end do:
evalf(maxerr)
                              0.39169859
```

2.2.2. Higher-order Taylor Methods

Preliminary **2.12. Higher-order Taylor methods** are based on Taylor series expansion.

- If we expand the solution $y(x)$, in terms of its mth-order Taylor polynomial about x_n and evaluated at x_{n+1}, we obtain

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hy'(x_n) + \frac{h^2}{2!}y''(x_n) + \cdots + \frac{h^m}{m!}y^{(m)}(x_n) \\ &\quad + \frac{h^{m+1}}{(m+1)!}y^{(m+1)}(\xi_n). \end{aligned} \tag{2.22}$$

- Successive differentiation of the solution, $y(x)$, gives

$$y'(x) = f(x, y(x)), \;\; y''(x) = f'(x, y(x)), \;\; \cdots,$$

and generally,

$$y^{(k)}(x) = f^{(k-1)}(x, y(x)). \tag{2.23}$$

- Thus, we have

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hf(x_n, y(x_n)) + \frac{h^2}{2!}f'(x_n, y(x_n)) + \cdots + \frac{h^m}{m!}f^{(m-1)}(x_n, y(x_n)) \\ &\quad + \frac{h^{m+1}}{(m+1)!}f^{(m)}(\xi_n, y(\xi_n)) \end{aligned} \tag{2.24}$$

Algorithm **2.13.** The **Taylor method of order** m corresponding to (2.24) is obtained by deleting the remainder term involving ξ_n:

$$y_{n+1} = y_n + h\,T_m(x_n, y_n), \tag{2.25}$$

where

$$T_m(x_n, y_n) = f(x_n, y_n) + \frac{h}{2!}f'(x_n, y_n) + \cdots + \frac{h^{m-1}}{m!}f^{(m-1)}(x_n, y_n). \tag{2.26}$$

Remark 2.14.

- $m = 1 \Rightarrow y_{n+1} = y_n + hf(x_n, y_n)$
  which is the Euler method.
- $m = 2 \Rightarrow y_{n+1} = y_n + h\left[f(x_n, y_n) + \frac{h}{2}f'(x_n, y_n)\right]$
- As m increases, the method achieves higher-order accuracy; however, **it requires to compute derivatives of $f(x, y(x))$.**

Example 2.15. Consider the initial-value problem

$$y' = y - x^3 + x + 1, \quad y(0) = 0.5. \tag{2.27}$$

(a) Find $T_3(x, y)$.

(b) Perform two iterations to find y_2, with $h = 1/2$.

Solution. Part (a): Since $y' = f(x, y) = y - x^3 + x + 1$,

$$\begin{aligned} f'(x, y) &= y' - 3x^2 + 1 \\ &= (y - x^3 + x + 1) - 3x^2 + 1 \\ &= y - x^3 - 3x^2 + x + 2 \end{aligned}$$

and

$$\begin{aligned} f''(x, y) &= y' - 3x^2 - 6x + 1 \\ &= (y - x^3 + x + 1) - 3x^2 - 6x + 1 \\ &= y - x^3 - 3x^2 - 5x + 2 \end{aligned}$$

Thus

$$\begin{aligned} T_3(x, y) &= f(x, y) + \frac{h}{2}f'(x, y) + \frac{h^2}{6}f''(x, y) \\ &= y - x^3 + x + 1 + \frac{h}{2}(y - x^3 - 3x^2 + x + 2) \\ &\quad + \frac{h^2}{6}(y - x^3 - 3x^2 - 5x + 2) \end{aligned} \tag{2.28}$$

For m large, the computation of T_m is time-consuming and cumbersome.

Part (b):

Euler_T3.mw

```
T3 := y - x^3 + x + 1 + 1/2*h*(-x^3 - 3*x^2 + x + y + 2)
      + 1/6*h^2*(-x^3 - 3*x^2 - 5*x + y + 2):
h := 1/2:
x0 := 0: y0 := 1/2:
y1 := y0 + h*eval(T3, [x = x0, y = y0])
                                155
                                ---
                                96
y2 := y1 + h*eval(T3, [x = x0 + h, y = y1])
                               16217
                               -----
                               4608
evalf(%)
                           3.519314236

exacty := x -> 4 + 5*x + 3*x^2 + x^3 - 7/2*exp(x):
exacty(1)
                                7
                          13 - - exp(1)
                                2
evalf(%)
                           3.486013602
#The absolute error:
evalf(abs(exacty(1) - y2))
                           0.033300634
```

2.3. Runge-Kutta Methods

Note: What we are going to do is to solve the *initial value problem* (IVP):

$$\begin{cases} y' &= f(x,y), \\ y(x_0) &= y_0. \end{cases} \quad \text{(IVP)} \tag{2.29}$$

- The **Taylor-series methods** of the preceding section have the drawback of requiring the **computation of derivatives of $f(x,y)$**.
  - It is a tedious and time-consuming procedure for most cases,
  - which makes the Taylor methods seldom used in practice.

Definition 2.16. Runge-Kutta methods

- have **high-order local truncation error of the Taylor methods**,
- but **eliminate the need to compute the derivatives of $f(x,y)$**.

That is, the Runge-Kutta methods are formulated, incorporating a **weighted average of the slope**, as follows:

$$y_{n+1} = y_n + h\left(w_1K_1 + w_2K_2 + \cdots + w_mK_m\right), \tag{2.30}$$

where

(a) $w_j \geq 0$ and $w_1 + w_2 + \cdots + w_m = 1$

(b) K_j are recursive evaluations of the slope $f(x,y)$

(c) Need to determine w_j and other parameters to satisfy

$$w_1K_1 + w_2K_2 + \cdots + w_mK_m \approx T_m(x_n, y_n) + \mathcal{O}(h^m) \tag{2.31}$$

That is, Runge-Kutta methods evaluate an *average slope* of $f(x,y)$ on the interval $[x_n, x_{n+1}]$ in the same order of accuracy as the mth-order Taylor method.

2.3.1. Second-order Runge-Kutta method

Formulation:

$$y_{n+1} = y_n + h\,(w_1 K_1 + w_2 K_2) \tag{2.32}$$

where

$$\begin{aligned} K_1 &= f(x_n, y_n) \\ K_2 &= f(x_n + \alpha h, y_n + \beta h K_1) \end{aligned}$$

Requirement: Determine w_1, w_2, α, β such that

$$w_1 K_1 + w_2 K_2 \;=\; T_2(x_n, y_n) + \mathcal{O}(h^2) \;=\; f(x_n, y_n) + \frac{h}{2} f'(x_n, y_n) + \mathcal{O}(h^2). \tag{2.33}$$

Derivation: For the left-hand side of (2.32), the Taylor series reads

$$y(x+h) = y(x) + h\,y'(x) + \frac{h^2}{2} y''(x) + \mathcal{O}(h^3).$$

Since $y' = f$ and $y'' = f_x + f_y y' = f_x + f_y f$,

$$y(x+h) = y(x) + h\,f + \frac{h^2}{2}(f_x + f_y f) + \mathcal{O}(h^3). \tag{2.34}$$

On the other hand, the right-side of (2.32) can be reformulated as

$$\begin{aligned} & y + h(w_1 K_1 + w_2 K_2) \\ &= y + w_1 h\, f(x,y) + w_2 h\, f(x + \alpha h, y + \beta h\, K_1) \\ &= y + w_1 h\, f + w_2 h\,(f + \alpha h\, f_x + \beta h\, f_y f) + \mathcal{O}(h^3), \end{aligned}$$

which reads

$$y + h(w_1 K_1 + w_2 K_2) \;=\; y + (w_1 + w_2) h\, f + h^2 (w_2 \alpha f_x + w_2 \beta f_y f) + \mathcal{O}(h^3). \tag{2.35}$$

The comparison of (2.34) and (2.35) drives the following result, for the second-order Runge-Kutta methods.

Results:

$$w_1 + w_2 = 1, \quad w_2\,\alpha = \frac{1}{2}, \quad w_2\,\beta = \frac{1}{2}. \tag{2.36}$$

Common Choices:

Algorithm 2.17.

I. $w_1 = w_2 = \dfrac{1}{2}$, $\alpha = \beta = 1$:
Then, the algorithm (2.32) becomes

$$y_{n+1} = y_n + \frac{h}{2}(K_1 + K_2), \tag{2.37}$$

where

$$\begin{aligned} K_1 &= f(x_n, y_n) \\ K_2 &= f(x_n + h, y_n + h\,K_1) \end{aligned}$$

This algorithm is the **Second-order Runge-Kutta (RK2) method**, which is also known as the **Heun's method**.

II. $w_1 = 0$, $w_2 = 1$, $\alpha = \beta = \dfrac{1}{2}$:
For the choices, the algorithm (2.32) reads

$$y_{n+1} = y_n + h\, f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(x_n, y_n)\right) \tag{2.38}$$

which is also known as the **Modified Euler method**.

It follows from (2.34) and (2.35) that the **local truncation error** for the above Runge-Kutta methods are $\mathcal{O}(h^3)$. Thus the **global error** becomes

$$\mathcal{O}(h^2). \tag{2.39}$$

2.3.2. Fourth-order Runge-Kutta method

Formulation:

$$y_{n+1} = y_n + h\left(w_1 K_1 + w_2 K_2 + w_3 K_3 + w_4 K_4\right) \tag{2.40}$$

where

$$\begin{aligned} K_1 &= f(x_n, y_n) \\ K_2 &= f(x_n + \alpha_1 h, y_n + \beta_1 h\, K_1) \\ K_3 &= f(x_n + \alpha_2 h, y_n + \beta_2 h\, K_1 + \beta_3 h\, K_2) \\ K_4 &= f(x_n + \alpha_3 h, y_n + \beta_4 h\, K_1 + \beta_5 h\, K_2 + \beta_6 h\, K_3) \end{aligned}$$

Requirement: Determine w_j, α_j, β_j such that

$$w_1 K_1 + w_2 K_2 + w_3 K_3 + w_4 K_4 \;=\; T_4(x_n, y_n) + \mathcal{O}(h^4)$$

The most common choice:

Algorithm 2.18. Fourth-order Runge-Kutta method (RK4): The most commonly used set of parameter values yields

$$y_{n+1} = y_n + \frac{h}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right) \tag{2.41}$$

where

$$\begin{aligned} K_1 &= f(x_n, y_n) \\ K_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_1) \\ K_3 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_2) \\ K_4 &= f(x_n + h, y_n + hK_3) \end{aligned}$$

The **local truncation error** for the above RK4 can be derived as

$$\frac{h^5}{5!} y^{(5)}(\xi_n), \tag{2.42}$$

for some $\xi_n \in [x_n, x_{n+1}]$. Thus the **global error** reads, for some $\xi \in [x_0, T]$,

$$\frac{(T - x_0)h^4}{5!} y^{(5)}(\xi). \tag{2.43}$$

Example 2.19. Use RK4 to solve the initial-value problem

$$y' = y - x^3 + x + 1, \quad y(0) = 0.5. \tag{2.44}$$

RK4.mw

```
f := proc(x,w)
        w-x^3+x+1
   end proc:
RK4 := proc(x0,xt,nt,y0,y)
        local h,x,w,n,K1,K2,K3,K4;
        h:=(xt-x0)/nt:
        x:=x0: w:=y0:
        y[0]:=w;
        for n from 1 by 1 to nt do
            K1:=f(x,w);
            K2:=f(x+h/2,w+(h/2)*K1);
            K3:=f(x+h/2,w+(h/2)*K2);
            x:=x+h;
            K4:=f(x,w+h*K3);
            w:=w+(h/6.)*(K1+2*K2+2*K3+K4);
            y[n]:=w;
        end do
end proc:

x0 := 0: xt := 3: nt := 48: y0 := 0.5:
yRK4 := Array(0..nt);
RK4(x0,xt,nt,y0,yRK4):

exacty := x -> 4 + 5*x + 3*x^2 + x^3 - 7/2*exp(x):
h := (xt - x0)/nt:
maxerr := 0:
for n from 0 by 1 to nt do
    maxerr:=max(maxerr,abs(exacty(n*h)-yRK4[n]));
end do:
evalf[16](maxerr)
                          0.00000184873274
```

Convergence Test for RK4, with (2.44):

h	Max-error	Error-ratio
$1/4$	$4.61 \cdot 10^{-4}$	
$1/8$	$2.93 \cdot 10^{-5}$	$\frac{4.61 \cdot 10^{-4}}{2.93 \cdot 10^{-5}} = 15.73378840$
$1/16$	$1.85 \cdot 10^{-6}$	$\frac{2.93 \cdot 10^{-5}}{1.85 \cdot 10^{-6}} = 15.83783784$
$1/32$	$1.01 \cdot 10^{-7}$	$\frac{1.85 \cdot 10^{-6}}{1.01 \cdot 10^{-7}} = 18.31683168$

Thus, the global truncation error of RK4 is in $\mathcal{O}(h^4)$.

2.3.3. Adaptive methods

Remark 2.20.

- **Accuracy** of numerical methods can be improved **by decreasing the step size**.
- Decreasing the step size $\approx$ Increasing the computational cost
- There may be subintervals where a relatively large step size suffices and other subintervals where a small step is necessary to keep the truncation error within a desired limit.
- An **adaptive method** is a numerical method which uses a variable step size.
- Example: **Runge-Kutta-Fehlberg method** (**RKF45**), which uses RK5 to estimate local truncation error of RK4.

2.4. One-Step Methods: Accuracy Comparison

For an accuracy comparison among the one-step methods presented in the previous sections, consider the motion of the **spring-mass system**:

$$\begin{aligned} &y''(t) + \frac{\kappa}{m} y = \frac{F_0}{m}\cos(\mu t), \\ &y(0) = c_0, \quad y'(0) = 0, \end{aligned} \tag{2.45}$$

where m is the mass attached at the end of a spring of the spring constant κ, the term $F_0\cos(\mu t)$ is a periodic driving force of frequency μ, and c_0 is the initial displacement from the equilibrium position.

- It is not difficult to find the analytic solution of (2.45):

$$y(t) = A\cos(\omega t) + \frac{F_0}{m(\omega^2 - \mu^2)}\cos(\mu t), \tag{2.46}$$

  where $\omega = \sqrt{\kappa/m}$ is the angular frequency and the coefficient A is determined corresponding to c_0.

- Let $y_1 = y$ and $y_2 = -y_1'/\omega$. Then, we can reformulate (2.45) as

$$\begin{aligned} y_1' &= -\omega y_2, & y_0(0) &= c_0, \\ y_2' &= \omega y_1 - \frac{F_0}{m\omega}\cos(\mu t), & y_2(0) &= 0. \end{aligned} \tag{2.47}$$

  We will deal with details of *High-Order Equations & Systems of Differential Equations* in § 2.6 on page 50.

- The motion is periodic only if μ/ω is a rational number. We choose

$$m = 1, \;\; F_0 = 40, \;\; A = 1, \;\; \omega = 4\pi, \;\; \mu = 2\pi. \quad (\Rightarrow c_0 \approx 1.33774) \tag{2.48}$$

  Thus the **fundamental period of the motion**

$$T = \frac{2\pi q}{\omega} = \frac{2\pi p}{\mu} = 1.$$

  See Figure 2.3 for the trajectory of the mass satisfying (2.47)-(2.48).

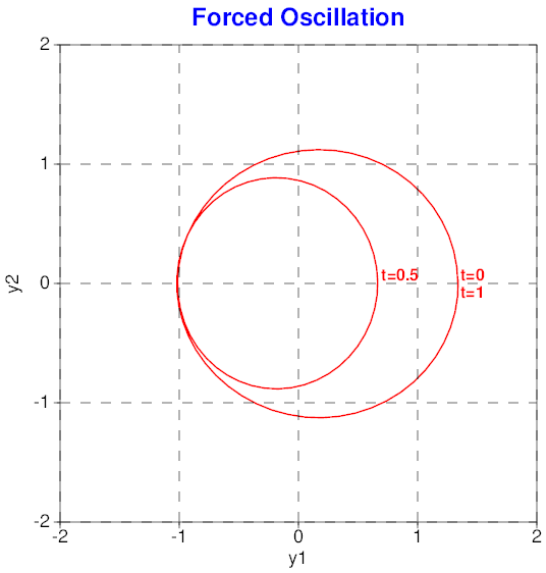


Figure 2.3: The trajectory of the mass satisfying (2.47)-(2.48).

Accuracy comparison

Table 2.1: The ℓ^2-error at $t = 1$ for various time step sizes.

$1/h$	Euler	RK2	RK4
100	1.19	3.31E-2	2.61E-5
200	4.83E-1 (1.3)	8.27E-3 (2.0)	1.63E-6 (4.0)
400	2.18E-1 (1.1)	2.07E-3 (2.0)	1.02E-7 (4.0)
800	1.04E-1 (1.1)	5.17E-4 (2.0)	6.38E-9 (4.0)

- Table 2.1 presents the ℓ^2-error at $t = 1$ for various time step sizes h, defined as

$$|\mathbf{y}^h_{N_t} - \mathbf{y}(1)| = \left(\left[y^h_{1,N_t} - y_1(1) \right]^2 + \left[y^h_{2,N_t} - y_2(1) \right]^2 \right)^{1/2}, \tag{2.49}$$

  where $\mathbf{y}^h_{N_t}$ denotes the computed solution at the N_t-th time step with $h = 1/N_t$.

- The numbers in parenthesis indicate the **order of convergence** α, defined as

$$\alpha := \frac{\ln(E(2h)/E(h))}{\ln 2}, \tag{2.50}$$

  where $E(h)$ and $E(2h)$ denote the errors obtained with the grid spacing to be h and $2h$, respectively.

- As one can see from the table, the one-step methods exhibit the expected accuracy.
- RK4 shows a much better accuracy than the lower-order methods, which explains its popularity.

Definition 2.21. (Order of Convergence):

- Let's assume that the algorithm under consideration produces error in $\mathcal{O}(h^\alpha)$. Then, we may write

$$E(h) = C\,h^\alpha, \tag{2.51}$$

  where h is the grid size.
- When the grid size is ph, the error will become

$$E(ph) = C\,(ph)^\alpha. \tag{2.52}$$

- It follows from (2.51) and (2.52) that

$$\frac{E(ph)}{E(h)} = \frac{C\,(ph)^\alpha}{C\,h^\alpha} = p^\alpha. \tag{2.53}$$

  Note that the values on the left side are determined by your experimental results.
- By taking a logarithm, one can solve the above equation for the **order of convergence** α:

$$\alpha = \frac{\ln(E(ph)/E(h))}{\ln p}. \tag{2.54}$$

  When $p = 2$, the above becomes (2.50).

2.5. Multi-step Methods

The problem: The first-order initial value problem (IVP)

$$\begin{cases} y' &= f(x,y), \\ y(x_0) &= y_0. \end{cases} \quad \text{(IVP)} \tag{2.55}$$

Numerical Methods:

- **Single-step/Starting methods**: Euler's method, Modified Euler's, Runge-Kutta methods
- **Multi-step/Continuing methods**: Adams-Bashforth-Moulton

Definition 2.22. An ***m*-step method**, $m \geq 2$, for solving the IVP, is a difference equation for finding the approximation y_{n+1} at $x = x_{n+1}$, given by

$$\begin{aligned} y_{n+1} &= a_1 y_n + a_2 y_{n-1} + \cdots + a_m y_{n+1-m} \\ &\quad + h[b_0 f(x_{n+1}, y_{n+1}) + b_1 f(x_n, y_n) + \cdots + b_m f(x_{n+1-m}, y_{n+1-m})]. \end{aligned} \tag{2.56}$$

The m-step method is said to be

$$\begin{cases} \text{explicit or open}, & \text{if } b_0 = 0 \\ \text{implicit or closed}, & \text{if } b_0 \neq 0 \end{cases}$$

Algorithm 2.23. (Fourth-order multi-step methods):

Let $y_i' = f(x_i, y_i)$.

- **Adams-Bashforth method** (explicit)

$$y_{n+1} = y_n + \frac{h}{24}(55y_n' - 59y_{n-1}' + 37y_{n-2}' - 9y_{n-3}') \tag{2.57}$$

- **Adams-Moulton method** (implicit)

$$y_{n+1} = y_n + \frac{h}{24}(9y_{n+1}' + 19y_n' - 5y_{n-1}' + y_{n-2}') \tag{2.58}$$

- **Adams-Bashforth-Moulton method** (predictor-corrector)

$$\begin{aligned} y_{n+1}^* &= y_n + \frac{h}{24}(55y_n' - 59y_{n-1}' + 37y_{n-2}' - 9y_{n-3}') \\ y_{n+1} &= y_n + \frac{h}{24}(9y'^*_{n+1} + 19y_n' - 5y_{n-1}' + y_{n-2}') \end{aligned} \tag{2.59}$$

where $y'^*_{n+1} = f(x_{n+1}, y^*_{n+1})$

Remark 2.24.

- y_1, y_2, y_3 can be computed by RK4.
- Multi-step methods may save evaluations of $f(x, y)$ such that in each step, they require only **one or two** new evaluations of $f(x, y)$ to fulfill the step.
- RK methods are accurate enough and easy to implement, so that multi-step methods are rarely applied in practice.
- ABM shows a **strong stability** for special cases, occasionally but not often [11].

ABM.mw

```
## Maple code: Adams-Bashforth-Moulton (ABM) Method
## Model Problem: y'= y- x^3 + x + 1, y(0) = 0.5, 0 <= x <= 3

f := proc(x,w)
        w-x^3+x+1
    end proc:
RK4 := proc(x0,xt,nt,y0,y)
        local h,x,w,n,K1,K2,K3,K4;
        h:=(xt-x0)/nt:
        x:=x0: w:=y0:
        y[0]:=w;
        for n from 1 by 1 to nt do
            K1:=f(x,w);
            K2:=f(x+h/2,w+(h/2)*K1);
            K3:=f(x+h/2,w+(h/2)*K2);
            x:=x+h;
            K4:=f(x,w+h*K3);
            w:=w+(h/6.)*(K1+2*K2+2*K3+K4);
            y[n]:=w;
        end do
end proc:

ABM:= proc(x0,xt,nt,y0,y)
        local h,x,w,n,ystar;
        h:=(xt-x0)/nt:
         ### Initialization with RK4
        RK4(x0,x0+3*h,3,y0,y);
        w:=y[3];
        ### Now, ABM steps
        for n from 4 by 1 to nt do
           x:=x0+n*h;
           ystar:=w +(h/24)*(55*f(x-h,y[n-1])-59*f(x-2*h,y[n-2])
                        +37*f(x-3*h,y[n-3])-9*f(x-4*h,y[n-4]));
           w:=w +(h/24)*(9*f(x,ystar)+19*f(x-h,y[n-1])
                        -5*f(x-2*h,y[n-2])+f(x-3*h,y[n-3]));
```

```
            y[n]:=w;
        end do;
end proc:

x0 := 0: xt := 3: nt := 48: y0 := 0.5:
yABM := Array(0..nt);
ABM(x0,xt,nt,y0,yABM):

exacty := x -> 4 + 5*x + 3*x^2 + x^3 - 7/2*exp(x):
h := (xt - x0)/nt:
maxerr := 0:
for n from 0 by 1 to nt do
    maxerr:=max(maxerr,abs(exacty(n*h)-yABM[n]));
end do:
evalf[16](maxerr)
                          0.00005294884316
```

Note: The maximum error for RK4 $= 1.85 \cdot 10^{-6}$.

2.6. High-Order Equations & Systems of Differential Equations

The problem: 2nd-order initial value problem (IVP)

$$\begin{cases} y'' = f(x, y, y'), & x \in [x_0, T] \\ y(x_0) = y_0, \quad y'(x_0) = u_0, \end{cases} \tag{2.60}$$

An equivalent problem: Let $u = y'$. Then,

$$u' = y'' = f(x, y, y') = f(x, y, u)$$

Thus, the above 2nd-order IVP can be equivalently written as the following system of first-order DEs:

$$\begin{cases} y' = u, & y(x_0) = y_0, \\ u' = f(x, y, u), & u(x_0) = u_0, \end{cases} \quad x \in [x_0, T]. \tag{2.61}$$

The right-side of the DEs involves no derivatives. □

Remark 2.25. The system (2.61) can be solved by one of the numerical methods (we have studied), after modifying it for vector functions.

- Let
$$Y = \begin{bmatrix} y \\ u \end{bmatrix}, \quad F(x, Y) = \begin{bmatrix} u \\ f(x, y, u) \end{bmatrix}, \quad Y_0 := \begin{bmatrix} y_0 \\ u_0 \end{bmatrix}.$$
- Then (2.61) reads
$$Y' = F(x, Y), \quad Y(x_0) = Y_0. \tag{2.62}$$

Example 2.26. Write the following DEs as a system of first-order differential equations.

(a) $\begin{cases} y'' + xy' + y = 0, \\ y(0) = 1, \quad y'(0) = 2. \end{cases}$ (b) $\begin{cases} x'' - x' + 2y'' = e^t, \\ -2x + y'' + 2y = 3t^2. \end{cases}$

Hint: For (b), you should first rewrite ***the first equation*** as $x'' = F(t, x, x')$ and introduce $x' = u$ and $y' = v$.

Solution.

RK4SYS.mw

```
## Ex) IVP of 2 equations:
##  x' = 2x+4y,  x(0)=-1
##  y' = -x+6y,  y(0)= 6, 0 <= t <= 1

ef := proc(t,w,f)
        f(1):=2*w(1)+4*w(2);
        f(2):=-w(1)+6*w(2);
end proc:

RK4SYS := proc(t0,tt,nt,m,x0,x)
        local h,t,w,n,j,K1,K2,K3,K4;
        #### initial setting
        w:=Vector(m):
        K1:=Vector(m):
        K2:=Vector(m):
        K3:=Vector(m):
        K4:=Vector(m):
        h:=(tt-t0)/nt:
        t:=t0;
        w:=x0;
        for j from 1 by 1 to m do
               x[0,j]:=x0(j);
        end do;
        #### RK4 marching
        for n from 1 by 1 to nt do
             ef(t,w,K1);
             ef(t+h/2,w+(h/2)*K1,K2);
             ef(t+h/2,w+(h/2)*K2,K3);
             ef(t+h,w+h*K3,K4);
             w:=w+(h/6.)*(K1+2*K2+2*K3+K4);
             for j from 1 by 1 to m do
                    x[n,j]:=w(j);
             end do
        end do
end proc:
```

```
36
37  m := 2:
38  x0 := Vector(m):
39
40  t0 := 0: tt := 1.: nt := 40:
41  x0(1) := -1:
42  x0(2) := 6:
43
44  xRK4 := Array(0..nt, 1..m):
45  RK4SYS(t0,tt,nt,m,x0,xRK4):
46
47  # Compute the analytic solution
48  #-------------------------------
49  ODE := diff(x(t),t)=2*x(t)+4*y(t), diff(y(t),t)=-x(t)+6*y(t):
50  ivs := x(0) = -1, y(0) = 6;
51  dsolve([ODE, ivs]);
52    /                                      1                        \
53   { x(t) = exp(4 t) (-1 + 26 t), y(t) = - exp(4 t) (24 + 52 t) }
54    \                                      4                        /
55  ex := t -> exp(4*t)*(-1 + 26*t):
56  ey := t -> 1/4*exp(4*t)*(24 + 52*t):
57
58  # Check error
59  #-------------------------------
60  h := (tt - t0)/nt:
61  printf("     n     x(n)      y(n)     error(x)  error(y)\n");
62  printf(" ----------------------------------------------\n");
63  for n from 0 by 2 to nt do
64      printf(" \t %5d %10.3f  %10.3f     %-10.3g %-10.3g\n",
65             n, xRK4[n,1], xRK4[n,2], abs(xRK4[n,1]-ex(n*h)),
66              abs(xRK4[n,2]-ey(n*h)) );
67  end do;
```

Result

n	x(n)	y(n)	error(x)	error(y)
0	-1.000	6.000	0	0
2	0.366	8.122	6.04e-06	4.24e-06
4	2.387	10.890	1.54e-05	1.07e-05
6	5.284	14.486	2.92e-05	2.01e-05
8	9.347	19.140	4.94e-05	3.35e-05
10	14.950	25.144	7.81e-05	5.26e-05
12	22.577	32.869	0.000118	7.91e-05
14	32.847	42.782	0.000174	0.000115
16	46.558	55.474	0.000251	0.000165
18	64.731	71.688	0.000356	0.000232
20	88.668	92.363	0.000498	0.000323
22	120.032	118.678	0.000689	0.000443
24	160.937	152.119	0.000944	0.000604
26	214.072	194.550	0.00128	0.000817
28	282.846	248.313	0.00174	0.0011
30	371.580	316.346	0.00233	0.00147
32	485.741	402.332	0.00312	0.00195
34	632.238	510.885	0.00414	0.00258
36	819.795	647.785	0.00549	0.0034
38	1059.411	820.262	0.00725	0.00447
40	1364.944	1037.359	0.00954	0.00586

RK4SYSTEM.mw

```
## Ex) y''-2*y'+2*y = exp(2*x)*sin(x), 0 <= x <= 1,
##     y(0) = -0.4, y'(0) = -0.6
Digits := 20:
RK4SYSTEM := proc(a,b,nt,X,F,x0,xn)
   local h,hh,t,m,n,j,w,K1,K2,K3,K4;
   #### initial setting
   with(LinearAlgebra):
   m := Dimension(Vector(F));
   w  :=Vector(m);
   K1:=Vector(m);
   K2:=Vector(m);
   K3:=Vector(m);
   K4:=Vector(m);
   h:=(b-a)/nt;  hh:=h/2;
   t :=a;
   w:=x0;
   for j from 1 by 1 to m do
          xn[0,j]:=x0[j];
   end do;
   #### RK4 marching
   for n from 1 by 1 to nt do
      K1:=Vector(eval(F,[x=t,seq(X[i+1]=xn[n-1,i], i = 1..m)]));
      K2:=Vector(eval(F,[x=t+hh,seq(X[i+1]=xn[n-1,i]+hh*K1[i], i = 1..m)]));
      K3:=Vector(eval(F,[x=t+hh,seq(X[i+1]=xn[n-1,i]+hh*K2[i], i = 1..m)]));
      t:=t+h;
      K4:=Vector(eval(F,[x=t,seq(X[i+1]=xn[n-1,i]+h*K3[i], i = 1..m)]));
      w:=w+(h/6)*(K1+2*K2+2*K3+K4);
      for j from 1 by 1 to m do
          xn[n,j]:=evalf(w[j]);
      end do
   end do
end proc:

# Call RK4SYSTEM.mw
#-------------------------------
with(LinearAlgebra):
m := 2:
F := [yp, exp(2*x)*sin(x) - 2*y + 2*yp]:
X := [x, y, yp]:
X0 := <-0.4, -0.6>:
a := 0: b := 1: nt := 10:
Xn := Array(0..nt, 1..m):
RK4SYSTEM(a, b, nt, X, F, X0, Xn):
```

```
44
45  # Compute the analytic solution
46  #-------------------------------
47  DE := diff(y(x), x, x) - 2*diff(y(x), x) + 2*y(x) = exp(2*x)*sin(x):
48  IC := y(0) = -0.4, D(y)(0) = -0.6:
49  dsolve({DE, IC}, y(x))
50                        1
51              y(x) = - exp(2 x) (sin(x) - 2 cos(x))
52                        5
53  ey := x -> 1/5*exp(2*x)*(sin(x) - 2*cos(x))
54  diff(ey(x), x)
55       2                                    1
56       - exp(2 x) (sin(x) - 2 cos(x)) + - exp(2 x) (2 sin(x) + cos(x))
57       5                                    5
58  eyp:=x->2/5*exp(2*x)*(sin(x)-2*cos(x))+1/5*exp(2*x)*(2*sin(x) + cos(x)):
59
60  # Check error
61  #-------------------------------
62  printf(" n   y_n    y(x_n)    y'_n    y'(x_n)   err(y)   err(y')\n");
63  printf("-------------------------------------------------------\n");
64  for n from 0 to nt do
65      xp := h*n + a;
66      printf("  %2d   %12.8f   %12.8f   %12.8f   %12.8f    %.3g    %.3g\n",
67          n, Xn[n, 1], ey(xp), Xn[n, 2], eyp(xp),
68          abs(Xn[n, 1] - ey(xp)), abs(Xn[n, 2] - eyp(xp)));
69  end do:
```

Result

```
1    n       y_n           y(x_n)        y'_n          y'(x_n)      err(y)      err(y')
2   ---------------------------------------------------------------------------------
3     0    -0.40000000   -0.40000000   -0.60000000   -0.60000000   0           0
4     1    -0.46173334   -0.46173297   -0.63163124   -0.63163105   3.72e-07    1.91e-07
5     2    -0.52555988   -0.52555905   -0.64014895   -0.64014866   8.36e-07    2.84e-07
6     3    -0.58860144   -0.58860005   -0.61366381   -0.61366361   1.39e-06    1.99e-07
7     4    -0.64661231   -0.64661028   -0.53658203   -0.53658220   2.02e-06    1.68e-07
8     5    -0.69356666   -0.69356395   -0.38873810   -0.38873905   2.71e-06    9.58e-07
9     6    -0.72115190   -0.72114849   -0.14438087   -0.14438322   3.41e-06    2.35e-06
10    7    -0.71815295   -0.71814890    0.22899702    0.22899243   4.06e-06    4.59e-06
11    8    -0.66971133   -0.66970677    0.77199180    0.77198383   4.55e-06    7.97e-06
12    9    -0.55644290   -0.55643814    1.53478148    1.53476862   4.77e-06    1.29e-05
13   10    -0.35339886   -0.35339436    2.57876634    2.57874662   4.50e-06    1.97e-05
```

2.7. Implicit Methods and Implicit Differential Equations

2.7.1. Revisit of the Euler Method

The problem: The first-order initial value problem (IVP)

$$\begin{cases} y' &= f(x,y), \\ y(x_0) &= y_0. \end{cases} \quad \text{(IVP)} \tag{2.63}$$

The Euler method $y_{n+1} = y_n + hf(x_n, y_n)$ can be rewritten as

$$\frac{y_{n+1} - y_n}{h} = f(x_n, y_n), \quad \text{(Forward Euler)} \tag{2.64}$$

in which the left side is a **forward approximation of** $y'(x_n)$. The formula (2.64) is called the **forward Euler method** for the IVP (2.63).

Definition 2.27. The **backward Euler method** for the IVP (2.63) is defined as

$$\frac{y_{n+1} - y_n}{h} = f(x_{n+1}, y_{n+1}), \quad \text{(Backward Euler)} \tag{2.65}$$

in which the left side is a **backward approximation of** $y'(x_{n+1})$.

Remark 2.28. **The Euler Methods**

- The forward Euler method (2.64) is an explicit method.
- The backward Euler method (2.65) is an implicit method, which requires to solve the following equation for y_{n+1}:

$$y_{n+1} = y_n + h\, f(x_{n+1}, y_{n+1}). \tag{2.66}$$

Claim 2.29. Let f satisfy a Lipschitz condition in y with a Lipschitz constant L. Then the fixed-point iteration applied for (2.66) converges when $h < 1/L$.

Example 2.30. Apply the Euler methods to solve the following IVP

$$y' + 2y = 2 - e^{-4x}, \quad y(0) = 1, \quad 0 \le x \le 1, \tag{2.67}$$

whose theoretical solution is $y(x) = 1 + \frac{1}{2}(e^{-4x} - e^{-2x})$.

Euler.py

```
#!/usr/bin/python3

import numpy as np
from numpy.linalg import norm

def f(x,y): return -2*y -np.exp(-4*x)+2

def exact_y(x):
    return 1+0.5*(np.exp(-4*x)-np.exp(-2*x))

#-----------------------------------------
def forward_Euler(x0,T,Nt,y0):
    x=x0; h =(T-x0)/Nt;
    y = np.zeros([Nt+1,]); y[0]=y0;
    for n in range(Nt):
         y[n+1] = y[n]+h*f(x,y[n])
         x += h;
    return y

#-----------------------------------------
def fixed_point(x,p,yn,h,tol):
    pp=p; err=10.; it0=0
    while(err>tol):
        p = yn+h*f(x,p)
        err = abs(p-pp);
        pp=p; it0+=1
    return p,it0

#-----------------------------------------
def backward_Euler(x0,T,Nt,y0,tol):
    x=x0; h =(T-x0)/Nt;
    y = np.zeros([Nt+1,]); y[0]=y0; it=0;
    for n in range(Nt):
         x += h;
         p,it0 = fixed_point(x,y[n],y[n],h,tol)
         y[n+1] = p; it += it0
    return y,it

```

```
#-----------------------------------------
def Crank_Nicolson(x0,T,Nt,y0,tol):
    x=x0; h =(T-x0)/Nt;
    y = np.zeros([Nt+1,]); y[0]=y0; it=0;
    for n in range(Nt):
         fn = f(x,y[n])
         x += h;
         #fixed-point iteration: included
         p = y[n]; pp=p; err=10.
         while(err>tol):
            p = y[n]+(h/2)*(f(x,p) + fn);
            err = abs(p-pp);
            pp=p; it += 1
         y[n+1] = p
    return y,it

#=========================================
if __name__ == "__main__":
    x0=0.0; T=1; y0=1.0; Nt = 40
    h = (T-x0)/Nt;
    tol1 = h/5
    tol2 = h**2/5

    X = np.linspace(x0,T,Nt+1)
    Y = exact_y(X.T)

    yf = forward_Euler(x0,T,Nt,y0)
    print('forward_Euler:  max-error = %g' %(norm(Y-yf,np.inf)))

    yb,it = backward_Euler(x0,T,Nt,y0,tol1)
    print('backward_Euler: max-error = %g' %(norm(Y-yb,np.inf)))
    print('   average fixed-point iter = %g' %(it/Nt))

    yc,it = Crank_Nicolson(x0,T,Nt,y0,tol2)
    print('Crank_Nicolson: max-error = %g' %(norm(Y-yc,np.inf)))
    print('   average fixed-point iter = %g' %(it/Nt))
```

Output

```
[Fri Jun.23] Euler.py
forward_Euler:  max-error = 0.0085829
backward_Euler: max-error = 0.00787605
   average fixed-point iter = 1.225
Crank_Nicolson: max-error = 0.000168574
   average fixed-point iter = 2.225
```

Algorithm 2.31. **The Crank-Nicolson method** For the IVP (2.63), the **Crank-Nicolson (CN) method** is formulated as

$$\frac{y_{n+1}-y_n}{h} = \frac{f(x_n,y_n)+f(x_{n+1},y_{n+1})}{2}, \quad \text{(CN)} \tag{2.68}$$

of which the global truncation error is $\mathcal{O}(h^2)$:

$$\frac{f(x_n,y_n)+f(x_{n+1},y_{n+1})}{2} = f(x_{n+1/2},y_{n+1/2}) + \mathcal{O}(h^2).$$

Note: Equation (2.68) can be rewritten as

$$y_{n+1} = y_n + \frac{h}{2}\big(f_n + f(x_{n+1},y_{n+1})\big), \quad f_n = f(x_n,y_n). \tag{2.69}$$

- Equation (2.69) is implemented in `Euler.py`.
- The CN method is also called the **Trapezoid method**.
- When $f(x_{n+1},y_{n+1})$ is evaluated by using a predicted value:

$$f(x_{n+1},y_{n+1}) \to f(x_{n+1},y^*_{n+1}), \;\; y^*_{n+1} = y_n + hf(x_n,y_n),$$

  the resulting algorithm becomes the Heun's method (RK2).

Remark 2.32. **Implicit Methods**
The disadvantage to using implicit methods is the cost to solve (2.66) or (2.69). However, **advantages** include that

- they are usually **more numerically stable**, and
- **a larger step size h** can be used, in particular for the numerical solution of stiff equations.

2.7.2. Implicit Differential Equations

Definition 2.33. An equation of the type

$$F(x, y, y') = 0, \quad y(x_0) = y_0, \tag{2.70}$$

where F is continuous, is called the **first-order implicit ordinary differential equation** (**IODE-1**).

- Under a suitable assumption on F, (2.70) can be reduced to

$$y' = f(x, y, y'), \quad y(x_0) = y_0. \tag{2.71}$$

Algorithm 2.34. The **third-order Runge-Kutta method** (**RK3**) for solving (2.71) is defined as

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{6}(K_1 + 4K_2 + K_3), \quad \text{where} \\ & K_1 = f(x_n, y_n, K_1) \\ & K_2 = f(x_n + h/2, y_n + (h/2)K_1, K_2) \\ & K_3 = f(x_{n+1}, y_n - hK_1 + 2hK_2, K_3) \end{aligned} \tag{2.72}$$

Note: The terms K_i, $i = 1, 2, 3$, are defined implicitly.

- They can be estimated by applying an iterative method, e.g., the **fixed-point iteration**.
- Each estimation of K_i is of the form $\boldsymbol{d} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})$.
  - It can be initialized by the last K value.
  - When $n = 0$, $K_1 = y'(x_0)$ must be initialized appropriately.
- The iteration may stop with a *reasonable* stopping tolerance:

$$\texttt{tol} = \mathcal{O}(h^\alpha), \quad \alpha = 3. \tag{2.73}$$

Example 2.35. Consider the problem

$$y' = \frac{1}{10}\big(\cos(x^2y') - \cos(e^y)\big) + \frac{1}{x}, \quad 1 \le x \le 3, \qquad y(1) = 0. \tag{2.74}$$

The exact solution is $y(x) = \ln x$. Apply the RK3 and the fixed-point iteration to solve the problem, with $N_t = 20,\ 40,\ 80$ ($h = 0.1,\ 0.05,\ 0.025$).

Hint: You may use some portions/ideas in `Euler.py`, implemented in Example 2.30, p. 58. For example, you may start with

IODE1.py

```
#!/usr/bin/python3

import numpy as np
from numpy.linalg import norm

def f(x,y,d): return (np.cos(x**2*d)-np.cos(np.exp(y)))/10+1/x
def exact_y(x): return np.log(x)

#-----------------------------------------
def fixed_pointF3(x,y,d,tol):
    err=1.; pd=d;
    while(err>tol):
        d = f(x,y,d)
        err = abs(d-pd); pd=d
    return d

#-----------------------------------------
# Here, implement "a function for the RK3"
#       that uses fixed_pointF3
```

Ans: Set `tol` = $h^3/5$ and for the fixed-point iteration, $y'(x_0)$ is initialized by 1.
Max.Error= [1.55e-05, 3.38e-06, 9.60e-08], for $h = [0.1,\ 0.05,\ 0.025]$.

2.8. Homework

1. For the IVP in (2.27),
   (a) Find $T_4(x, y)$.
   (b) Perform two steps of the 3rd and 4th-order Taylor methods, with $h = 1/2$, to find an approximate solutions of y at $x = 1$.
   (c) Compare the errors, given that the exact solution
   $$y(x) = 4 + 5x + 3x^2 + x^3 - \frac{7}{2}e^x$$
2. Verify (2.18).

   Hint: Use ① $\lim_{x\to 0^+}(1+x)^{1/x} = e$ and ② the limit is obtained from an increasing sequence, i.e., $(1+x)^{1/x} \nearrow e$ as $x \searrow 0$.
3. Derive the global error of RK4 in (2.43), given the local truncation error (2.42).
4. Write the following DE as a system of first-order differential equations.
   $$\begin{aligned} x'' + x'y - 2y'' &= t, \\ -2y + y'' + x &= e^{-t}, \end{aligned}$$
   where the derivative denotes d/dt.
5. Complete an implementation for the problem in Example 2.35, p. 62.

CHAPTER 3

Properties of Numerical Methods

- Numerical methods compute approximate solutions for differential equations (DEs).
- In order for the numerical solution to be a reliable approximation of the given problem, the numerical method should satisfy certain properties.
- In this chapter, we consider properties of numerical methods that are most common in numerical analysis such as *consistency, convergence, stability, accuracy order, boundedness / maximum principle, and conservation*.

Contents of Chapter 3

3.1. General Parabolic Problems: Maximum Principle 66
3.2. A Model Problem: Heat Conduction in 1D 68
3.3. Consistency 70
3.4. Convergence 72
3.5. Stability 77
3.6. Boundedness – Maximum Principle 83
3.7. Conservation 85
3.8. A Multi-Step Central Scheme 86
3.9. The θ-Method 87
3.10. Homework 93

3.1. General Parabolic Problems: Maximum Principle

Let Ω be a bounded domain in $\mathbb{R}^d$, $I = (0, T)$, finite interval in time, and $\Gamma = \partial\Omega$. Consider a **general parabolic problem** of the form

$$\begin{aligned} u_t - \Delta u &= f, && \text{in } \Omega \times I, \\ u &= g, && \text{on } \Gamma \times I, \\ u(\cdot, 0) &= u_0, && \text{in } \Omega, \end{aligned} \tag{3.1}$$

where $\Delta = \sum \partial_{x_i, x_i}$, the **Laplacian operator**.

In order to show a **maximum principle** for this problem, it is convenient to introduce the **parabolic boundary** of $\Omega \times I$:

$$\Gamma_p \stackrel{\text{def}}{=\!=} (\Gamma \times \bar{I}) \cup (\Omega \times \{t = 0\}).$$

Theorem 3.1. (Maximum Principle)
Let u be smooth and assume that $u_t - \Delta u \leq 0$ in $\Omega \times I$. Then u attains its maximum on the parabolic boundary Γ_p.

Proof. **Assume that the conclusion is not true.** Then the maximum would be attained either at an interior point of $\Omega \times I$ or at a point of $\Omega \times \{t = T\}$. That is, at a point $(\bar{\mathbf{x}}, \bar{t}) \in \Omega \times (0, T]$, we have

$$u(\bar{\mathbf{x}}, \bar{t}) = \max_{\bar{\Omega} \times \bar{I}} u = M > m = \max_{\Gamma_p} u. \tag{3.2}$$

Select $\varepsilon > 0$, sufficiently small, such that

$$m + \varepsilon \max_{\Gamma_p} |\mathbf{x}|^2 < M,$$

and define

$$w(\mathbf{x}, t) = u(\mathbf{x}, t) + \varepsilon |\mathbf{x}|^2. \tag{3.3}$$

Then

$$\max_{\Gamma_p} w \leq m + \varepsilon \max_{\Gamma_p} |\mathbf{x}|^2 < M \leq \max_{\bar{\Omega} \times \bar{I}} w.$$

Thus w must take its maximum at some point $(\widetilde{\mathbf{x}}, \widetilde{t}) \notin \Gamma_p$, where

$$
\begin{aligned}
-\Delta w(\widetilde{\mathbf{x}},\widetilde{t}) &= -\sum_{i=1}^{d} w_{x_i,x_i}(\widetilde{\mathbf{x}},\widetilde{t}) \;\geq\; 0,\\
w_t(\widetilde{\mathbf{x}},\widetilde{t}) &\begin{cases} = 0 & \text{if } \widetilde{t} < T,\\ \geq 0 & \text{if } \widetilde{t} = T, \end{cases}
\end{aligned}
\tag{3.4}
$$

from which we have

$$
w_t(\widetilde{\mathbf{x}},\widetilde{t}) - \Delta w(\widetilde{\mathbf{x}},\widetilde{t}) \geq 0. \tag{3.5}
$$

On the other hand, it follows from our assumption that

$$
w_t - \Delta w \;=\; u_t - \Delta u - 2d\varepsilon \leq -2d\varepsilon, \quad \text{in } \Omega \times I, \tag{3.6}
$$

where we have used $\Delta|\mathbf{x}|^2 = \Delta(x_1^2 + x_2^2 + \cdots + x_d^2) = 2d$. This is a contradiction to (3.5) and thus completes the proof. □

Remark 3.2. By considering $\pm u$, it follows particularly that the solution of the **homogeneous heat equation** ($f = 0$) attains both its maximum and its minimum on Γ_p.

Note: The maximum principle prevents the solution from **developing local extrema or oscillating in the interior of the domain.**

- In the mathematical fields of differential equations (DEs) and geometric analysis, the maximum principle is one of the most useful and best known tools of study.
- The maximum principle enables one to obtain information about solutions of DEs without any explicit knowledge of the solutions themselves.
- In particular, the maximum principle is a useful tool in the numerical approximation of solutions of ODEs and PDEs and in **the determination of error bounds in numerical approximations.**

One can easily prove the following; see Homework 3.1.

Theorem 3.3. *Let* $u \in C^2(\Omega) \cap C(\bar{\Omega})$ *and* $-\Delta u \leq 0$. *Then*

$$
\sup_{\Omega} u = \max_{\partial\Omega} u. \tag{3.7}
$$

3.2. A Model Problem: Heat Conduction in 1D

Let $\Omega = (0,1)$ and $J = (0,T]$, for some $T > 0$. Consider the following simplest model problem for parabolic equations in one-dimensional (1D) space:

$$\begin{aligned} u_t - u_{xx} &= f, \quad (x,t) \in \Omega \times J, \\ u &= 0, \qquad (x,t) \in \Gamma \times J, \\ u &= u_0, \qquad x \in \Omega, \ \ t = 0, \end{aligned} \tag{3.8}$$

where f is a heat source, Γ denotes the boundary of Ω, i.e., $\Gamma = \{0,1\}$, and u_0 is the prescribed initial value of the solution at $t = 0$.

Finite difference methods

We begin with our discussion of finite difference (FD) methods for (3.8) by partitioning the domain. Let

$$\begin{aligned} &\Delta t = T/n_t, \quad t^n = n\Delta t, \ \ n = 0, 1, \cdots, n_t; \\ &\Delta x = 1/n_x, \quad x_j = j\Delta x, \ \ j = 0, 1, \cdots, n_x; \end{aligned}$$

for some positive integers n_t and n_x. Define $u_j^n = u(x_j, t^n)$.

Let

$$\mathcal{S}^n := \Omega \times (t^{n-1}, t^n] \tag{3.9}$$

be the nth **space-time slice**.

- Suppose that the computation has been performed for $u^k = \{u_j^k\}$, $0 \le k \le n-1$.
- Then, the task is to compute u^n by integrating the equation on the space-time slice $\mathcal{S}^n$, utilizing FD schemes.

The basic idea of FD schemes is to replace derivatives by FD approximations. It can be done in various ways; here we consider most common ways that are based on the **Taylor's formula**.

Recall: The central second-order FD formula for u_{xx} presented in (1.17):

$$\begin{aligned} u_{xx}(x_i) &= \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} - 2\frac{u_{xxxx}(x_i)}{4!}h^2 \\ &\quad -2\frac{u_{xxxxxx}(x_i)}{6!}h^4 - \cdots . \end{aligned} \tag{3.10}$$

Apply the above to have

$$\begin{aligned} u_{xx}(x_j, t^n) &= \frac{u^n_{j-1} - 2u^n_j + u^n_{j+1}}{\Delta x^2} \\ &\quad -2\frac{u_{xxxx}(x_j, t^n)}{4!}\Delta x^2 + \mathcal{O}(\Delta x^4). \end{aligned} \tag{3.11}$$

- For the temporal direction, one can also apply a difference formula for **the approximation of the time-derivative u_t.**
- Depending on the way of combining the spatial and temporal differences, the resulting scheme can behave quite differently.

Explicit Scheme for $u_t - u_{xx} = f$

The following presents the simplest scheme:

$$\frac{v^n_j - v^{n-1}_j}{\Delta t} - \frac{v^{n-1}_{j-1} - 2v^{n-1}_j + v^{n-1}_{j+1}}{\Delta x^2} = f^{n-1}_j, \tag{3.12}$$

which is an explicit scheme for (3.8), called the **forward Euler method**. Here v^n_j is an approximation of u^n_j.

The above scheme can be rewritten as

$$v^n_j = \mu\, v^{n-1}_{j-1} + (1 - 2\mu)\, v^{n-1}_j + \mu\, v^{n-1}_{j+1}, + \Delta t f^{n-1}_j \tag{3.13}$$

where

$$\mu = \frac{\Delta t}{\Delta x^2}$$

3.3. Consistency

The bottom line for an accurate numerical method is that the discretization becomes exact as the grid spacing tends to zero, which is the basis of *consistency*.

Definition 3.4. Given a PDE $Pu = f$ and a FD scheme $P_{\Delta x,\Delta t} v = f$, the FD scheme is said to be **consistent** with the PDE if for every smooth function $\phi(x,t)$

$$P\phi - P_{\Delta x,\Delta t}\phi \to 0 \quad \text{as} \quad (\Delta x, \Delta t) \to 0,$$

with the convergence being pointwise at each grid point.

Note: Not all numerical methods based on Taylor series expansions are consistent; sometimes, we may have to restrict the manner in which Δx and Δt approach zero in order for them to be consistent.

Example 3.5. *The forward Euler scheme (3.12) is consistent.*

Proof. For the heat equation in 1D,

$$P\phi \equiv \Big(\frac{\partial}{\partial t} - \frac{\partial^2}{\partial x^2}\Big)\phi = \phi_t - \phi_{xx}.$$

The forward Euler scheme (3.12) reads

$$P_{\Delta x,\Delta t}\phi = \frac{\phi_j^n - \phi_j^{n-1}}{\Delta t} - \frac{\phi_{j-1}^{n-1} - 2\phi_j^{n-1} + \phi_{j+1}^{n-1}}{\Delta x^2}$$

The truncation error for the temporal discretization can be obtained applying the one-sided FD formula:

$$\phi_t(x_j, t^{n-1}) = \frac{\phi_j^i - \phi_j^{n-1}}{\Delta t} - \frac{\phi_{tt}(x_j, t^{n-1})}{2!}\Delta t + \mathcal{O}(\Delta t^2). \tag{3.14}$$

It follows from (3.11) and (3.14) that the truncation error of the forward Euler scheme evaluated at (x_j, t^{n-1}) becomes

$$\begin{aligned} &\left(P\phi - P_{\Delta x,\Delta t}\phi\right)(x_j, t^{n-1}) \\ &\quad = -\frac{\phi_{tt}(x_j, t^{n-1})}{2!}\Delta t + 2\frac{\phi_{xxxx}(x_j, t^{n-1})}{4!}\Delta x^2 + \mathcal{O}(\Delta t^2 + \Delta x^4), \end{aligned} \tag{3.15}$$

which clearly approaches zero as $(\Delta x, \Delta t) \to 0$. □

Truncation Error

Definition 3.6. Let u be smooth and

$$P\,u(x_j, t^n) \;=\; P_{\Delta x,\Delta t}\,u_j^n + \mathcal{T}u_j^n, \tag{3.16}$$

Then, $\mathcal{T}u_j^n$ is called the **truncation error** of the FD scheme $P_{\Delta x,\Delta t}v = f$ evaluated at (x_j, t^n).

It follows from (3.15) that the truncation error of the forward Euler scheme (3.12) is

$$\mathcal{O}(\Delta t + \Delta x^2)$$

for all grid points (x_j, t^n).

3.4. Convergence

A numerical method is said to be *convergent* if the solution of the FD scheme tends to the exact solution of the PDE as the grid spacing tends to zero. We define convergence in a formal way as follows:

Definition 3.7. A FD scheme approximating a PDE is said to be convergent if

$$u(x,t) - v_j^n \to 0, \text{ as } (x_j, t^n) \to (x,t) \text{ and } (\Delta x, \Delta t) \to 0, \tag{3.17}$$

where $u(x,t)$ is the exact solution of PDE and v_j^n denotes the the solution of the FD scheme.

Note: Consistency implies that the truncation error

$$(Pu - P_{\Delta x,\Delta t}u) \to 0, \text{ as } (\Delta x, \Delta t) \to 0.$$

So consistency is certainly necessary for convergence, but may not be sufficient.

Example 3.8. *The forward Euler scheme (3.12) is convergent, when*

$$\mu = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}. \tag{3.18}$$

Proof. **(The scheme)** Recall the explicit scheme (3.12):

$$\frac{v_j^n - v_j^{n-1}}{\Delta t} - \frac{v_{j-1}^{n-1} - 2v_j^{n-1} + v_{j+1}^{n-1}}{\Delta x^2} = f_j^{n-1} \tag{3.19}$$

which can be expressed as

$$P_{\Delta x,\Delta t}\, v_j^{n-1} = f_j^{n-1} \tag{3.20}$$

On the other hand, for the exact solution u,

$$P_{\Delta x,\Delta t}\, u_j^{n-1} + \mathcal{T} u_j^{n-1} = f_j^{n-1} \tag{3.21}$$

(Error equation) Let

$$e_j^n = u_j^n - v_j^n,$$

where u is the exact solution of (3.8). Then, from (3.20) and (3.21), the *error equation* becomes

$$P_{\Delta x,\Delta t}\, e_j^{n-1} \;=\; -\mathcal{T} u_j^{n-1},$$

which in detail reads

$$\frac{e_j^n - e_j^{n-1}}{\Delta t} - \frac{e_{j-1}^{n-1} - 2e_j^{n-1} + e_{j+1}^{n-1}}{\Delta x^2} \;=\; -\mathcal{T} u_j^{n-1}. \tag{3.22}$$

In order to control the error more conveniently, we reformulate the error equation

$$e_j^n \;=\; \mu\, e_{j-1}^{n-1} + (1-2\mu)\, e_j^{n-1} + \mu\, e_{j+1}^{n-1} - \Delta t\, \mathcal{T} u_j^{n-1}. \tag{3.23}$$

(Error analysis with ℓ_∞-norm) Now, define

$$\mathcal{E}^n = \max_j |e_j^n|, \quad \mathcal{T}^n = \max_j |\mathcal{T} u_j^n|, \quad \widehat{\mathcal{T}} = \max_n \mathcal{T}^n.$$

Note that $v_j^0 = u_j^0$ for all j and therefore $\mathcal{E}^0 = 0$.

It follows from (3.23) and the assumption (3.18) that

$$\begin{aligned} |e_j^n| \;&\le\; \mu\, |e_{j-1}^{n-1}| + (1-2\mu)\, |e_j^{n-1}| + \mu\, |e_{j+1}^{n-1}| \\ &\quad\; + \Delta t\, |\mathcal{T} u_j^{n-1}| \\ &\le\; \mu\, \mathcal{E}^{n-1} + (1-2\mu)\, \mathcal{E}^{n-1} + \mu\, \mathcal{E}^{n-1} \\ &\quad\; + \Delta t\, \mathcal{T}^{n-1} \\ &=\; \mathcal{E}^{n-1} + \Delta t\, \mathcal{T}^{n-1}. \end{aligned} \tag{3.24}$$

Since the above inequality holds for all j, we have

$$\mathcal{E}^n \le \mathcal{E}^{n-1} + \Delta t\, \mathcal{T}^{n-1}, \tag{3.25}$$

and therefore

$$\begin{aligned} \mathcal{E}^n \;&\le\; \mathcal{E}^{n-1} + \Delta t\, \mathcal{T}^{n-1} \\ &\le\; \mathcal{E}^{n-2} + \Delta t\, \mathcal{T}^{n-1} + \Delta t\, \mathcal{T}^{n-2} \\ &\le\; \quad \cdots \\ &\le\; \mathcal{E}^0 + \sum_{k=1}^{n-1} \Delta t\, \mathcal{T}^k. \end{aligned} \tag{3.26}$$

Since $\mathcal{E}^0 = 0$,

$$\mathcal{E}^n \leq (n-1)\Delta t\, \widehat{\mathcal{T}} \leq T\, \widehat{\mathcal{T}}, \tag{3.27}$$

where T is the upper bound of the time available. Since $\widehat{\mathcal{T}} = \mathcal{O}(\Delta t + \Delta x^2)$, the maximum norm of the error approaches zero as $(\Delta x, \Delta t) \to 0$. □

Remark 3.9.

- The assumption $\mu \leq 1/2$ makes coefficients in the forward Euler scheme (3.13) nonnegative, which in turn makes v_j^n a weighted average of $\{v_{j-1}^{n-1}, v_j^{n-1}, v_{j+1}^{n-1}\}$.
- The analysis can often conclude

$$\mathcal{E}^n = \mathcal{O}(\widehat{\mathcal{T}}), \ \forall\, n$$

- Convergence is what a numerical scheme must satisfy.

- However, showing convergence is not easy in general, if attempted in a direct manner as in the previous example.
- There is a related concept, stability, that is easier to check.

An Example: $\mu \leq 1/2$

```
ForwardEuler := proc(a, b, T, nx, nt, α, f, u0)
    local j, n, h, k, xj, tn, id1, id2, mu, w, wT;
    with(LinearAlgebra) :
    h := (b − a)/nx;
    k := T/nt;
    mu := α^2 · k/h^2;
    print(`mu=`, mu);

    w := Matrix(nx + 1, 2);
    wT := Vector(nx + 1);
    for j to nx + 1 do
        w[j, 1] := eval(u0, x = a + (j − 1)·h);
    end do;

    for n from 1 to nt do
        tn := (n − 1)·k;
        id1 := modp(n + 1, 2) + 1;
        id2 := modp(n, 2) + 1;
        for j from 2 to nx do
            xj := a + (j − 1)·h;
            w[j, id2] := (1 − 2·mu)·w[j, id1]
                + mu·(w[j − 1, id1] + w[j + 1, id1])
                + k·eval(f, [x = xj, t = tn]);
        end do;
    end do;
    for j to nx + 1 do wT[j] := w[j, id2]; end do;
    return wT;
end proc:
```

Figure 3.1: The explicit scheme (forward Euler) in Maple.

The problem:

$$\begin{aligned} u_t - \alpha^2 u_{xx} &= 0, \quad (x,t) \in [0,1] \times [0,1], \\ u &= 0, \qquad (x,t) \in \{0,1\} \times [0,1], \\ u &= \sin(\pi x), \quad x \in [0,1], \;\; t = 0, \end{aligned} \tag{3.28}$$

The exact solution:

$$u(x,t) = e^{-\pi^2 t} \sin(\pi x)$$

Parameter setting:

$$a := 0;\ \ b := 1;\ \ T := 1;\ \ \alpha := 1;\ \ f := 0;$$
$$nx := 10;$$

Numerical results:

$nt := 200$ ($\mu = 1/2$)	$\|u^{n_t} - v^{n_t}\|_\infty = 7.94 \times 10^{-6}$
$nt := 170$ ($\mu \approx 0.588$)	$\|u^{n_t} - v^{n_t}\|_\infty = 1.31 \times 10^{9}$

- For the case $\mu \approx 0.588$, the numerical solution becomes oscillatory and blows up.

3.5. Stability

- The example with Figure 3.1 shows that consistency of a numerical method is not enough to guarantee convergence of its solution to the exact solution.
- In order for a consistent numerical scheme to be convergent, a required property is stability.
- Note that if a scheme is convergent, it produces a bounded solution whenever the exact solution is bounded.
- This is the basis of stability.
- We first define the L^2-norm of grid function v:

$$\|v\|_{\Delta x} = \left(\Delta x \sum_j |v_j|^2\right)^{1/2}.$$

Definition 3.10. A FD scheme $P_{\Delta x,\Delta t} v = 0$ for a homogeneous PDE $Pu = 0$ is stable if for any positive T, there is a constant C_T such that

$$\|v^n\|_{\Delta x} \le C_T \sum_{m=0}^{M} \|u^m\|_{\Delta x}, \tag{3.29}$$

for $0 \le t^n \le T$ and for Δx and Δt sufficiently small. Here M is chosen to incorporate the data initialized on the first $M+1$ levels.

3.5.1. Approaches for proving stability

There are two fundamental approaches for proving stability:

- **The Fourier analysis (von Neumann analysis)**
  It applies only to linear constant coefficient problems.
- **The energy method**
  It can be used for more general problems with variable coefficients and nonlinear terms. But it is quite complicated and the proof is problem dependent.

Theorem 3.11. (Lax-Richtmyer Equivalence Theorem). *Given a well-posed linear initial value problem and its FD approximation that satisfies the* ***consistency*** *condition,* ***stability is a necessary and sufficient condition for convergence.***

Note: The above theorem is very useful and important.

- Proving convergence is difficult for most problems.
- However, the determination of consistency of a scheme is quite easy as shown in §3.3, and determining stability is also easier than showing convergence.
- Later, we will introduce the von Neumann analysis of stability of FD schemes, which allows one to analyze stability much simpler than a direct verification of (3.29).

Now, let us consider a sufficient condition for stability.

Theorem 3.12. *A FD scheme* $P_{\Delta x,\Delta t}v = 0$ *for a homogeneous PDE* $Pu = 0$ *is stable if*

$$\|v^n\|_{\Delta x} \le (1 + C\Delta t)\|v^{n-1}\|_{\Delta x}, \tag{3.30}$$

for some $C \ge 0$ *independent on* Δt

Proof. Recall $\Delta t = T/n_t$, for some positive integer n_t. A recursive application of (3.30) reads

$$\begin{aligned} \|v^n\|_{\Delta x} &\le (1 + C\Delta t)\|v^{n-1}\|_{\Delta x} \le (1 + C\Delta t)^2\|v^{n-2}\|_{\Delta x} \\ &\le \cdots \le (1 + C\Delta t)^n\|v^0 (= u^0)\|_{\Delta x}. \end{aligned} \tag{3.31}$$

Here the task is to show $(1 + C\Delta t)^n$ is bounded by some positive number C_T for $n = 1, \cdots, n_t$, independently on Δt. Since $\Delta t = T/n_t$, we have

$$\begin{aligned} (1 + C\Delta t)^n &= (1 + CT/n_t)^n \\ &\le (1 + CT/n_t)^{n_t} = \left[(1 + CT/n_t)^{n_t/CT}\right]^{CT} \\ &\le e^{CT}, \end{aligned}$$

which proves (3.29) with by $C_T := e^{CT}$. □

3.5.2. The von Neumann analysis

- Let ϕ be a grid function defined on grid points of spacing Δx and $\phi_j = \phi(j\Delta x)$. Then, its Fourier transform is given by, for $\xi \in [-\pi/\Delta x, \pi/\Delta x]$,

$$\widehat{\phi}(\xi) = \frac{1}{\sqrt{2\pi}} \sum_{j=-\infty}^{\infty} e^{-ij\Delta x\xi}\, \phi_j, \tag{3.32}$$

  and the inverse formula is

$$\phi_j = \frac{1}{\sqrt{2\pi}} \int_{-\pi/\Delta x}^{\pi/\Delta x} e^{ij\Delta x\xi}\, \widehat{\phi}(\xi) d\xi. \tag{3.33}$$

- Parseval's identity

$$\|\phi^n\|_{\Delta x} = \|\widehat{\phi^n}\|_{\Delta x}, \tag{3.34}$$

  where

$$\|\phi^n\|_{\Delta x} = \Big(\sum_{j=-\infty}^{\infty} |\phi_j|^2 \Delta x \Big)^{1/2},$$
$$\|\widehat{\phi^n}\|_{\Delta x} = \Big(\int_{-\pi/\Delta x}^{\pi/\Delta x} |\widehat{\phi}(\xi)|^2 d\xi \Big)^{1/2}$$

- The stability inequality (3.29) can be replaced by

$$\|\widehat{v^n}\|_{\Delta x} \le C_T \sum_{m=0}^{M} \|\widehat{v^m}\|_{\Delta x}. \tag{3.35}$$

- Thus stability can be determined by providing (3.35) in the frequency domain.

Example

To show how one can use the above analysis, we exemplify the forward Euler scheme (3.13), with $f = 0$:

$$v_j^n = \mu\, v_{j-1}^{n-1} + (1-2\mu)\, v_j^{n-1} + \mu\, v_{j+1}^{n-1} \tag{3.36}$$

- The inversion formula implies

$$v_j^n = \frac{1}{\sqrt{2\pi}} \int_{-\pi/\Delta x}^{\pi/\Delta x} e^{ij\Delta x\xi}\, \widehat{v}^n(\xi)\, d\xi. \tag{3.37}$$

  Thus it follows from (3.36) and (3.37) that

$$v_j^n = \frac{1}{\sqrt{2\pi}} \int_{-\pi/\Delta x}^{\pi/\Delta x} \mathcal{F}_{\Delta x,j}(\xi)\, d\xi, \tag{3.38}$$

  where

$$\begin{aligned} \mathcal{F}_{\Delta x,j}(\xi) &= \mu e^{i(j-1)\Delta x\xi}\, \widehat{v}^{n-1}(\xi) \\ &\quad +(1-2\mu) e^{ij\Delta x\xi}\, \widehat{v}^{n-1}(\xi) \\ &\quad +\mu e^{i(j+1)\Delta x\xi}\, \widehat{v}^{n-1}(\xi) \\ &= e^{ij\Delta x\xi} \left[\mu\, e^{-i\Delta x\xi} + (1-2\mu) + \mu\, e^{i\Delta x\xi}\right] \widehat{v}^{n-1}(\xi) \end{aligned}$$

- Comparing (3.37) with (3.38), we obtain

$$\widehat{v}^n(\xi) = \left[\mu\, e^{-i\Delta x\xi} + (1-2\mu) + \mu\, e^{i\Delta x\xi}\right] \widehat{v}^{n-1}(\xi) \tag{3.39}$$

- Letting $\vartheta = \Delta x\xi$, we define the **amplification factor** for the scheme (3.13) by

$$\begin{aligned} g(\vartheta) &= \mu\, e^{-i\Delta x\xi} + (1-2\mu) + \mu\, e^{i\Delta x\xi} \\ &= \mu\, e^{-i\vartheta} + (1-2\mu) + \mu\, e^{i\vartheta} \\ &= (1-2\mu) + 2\mu\cos(\vartheta) \\ &= 1 - 2\mu(1-\cos(\vartheta)) = 1 - 4\mu\sin^2(\vartheta/2) \end{aligned} \tag{3.40}$$

- Equation (3.39) can be rewritten as

$$\widehat{v}^n(\xi) = g(\vartheta)\, \widehat{v}^{n-1}(\xi) = g(\vartheta)^2\, \widehat{v}^{n-2}(\xi) = \cdots = g(\vartheta)^n\, \widehat{v}^0(\xi). \tag{3.41}$$

  Therefore, when $g(\vartheta)^n$ is suitably bounded, the scheme is stable. In fact, $g(\vartheta)^n$ would be uniformly bounded only if $|g(\vartheta)| \leq 1 + C\Delta t$.

- It is not difficult to see

$$|g(\vartheta)| = |1 - 2\mu(1 - \cos(\vartheta))| \leq 1$$

only if

$$0 \leq \mu \leq 1/2 \tag{3.42}$$

which is the **stability condition** of the scheme (3.13).

The von Neumann analysis: Is it complicated?

A simpler and equivalent procedure of the von Neumann analysis can be summarized as follows:

- Replace v_j^n by $g^n e^{ij\vartheta}$ for each value of j and n.
- Find conditions on coefficients and grid spacings which would satisfy $|g| \leq 1 + C\Delta t$, for some $C \geq 0$.

The forward Euler scheme (3.13):

$$v_j^n = \mu\, v_{j-1}^{n-1} + (1 - 2\mu)\, v_j^{n-1} + \mu\, v_{j+1}^{n-1}$$

Replacing v_j^n with $g^n e^{ij\vartheta}$ gives

$$g^n e^{ij\vartheta} = \mu\, g^{n-1} e^{i(j-1)\vartheta} + (1 - 2\mu)\, g^{n-1} e^{ij\vartheta} + \mu\, g^{n-1} e^{i(j+1)\vartheta}$$

Dividing both sides of the above by $g^{n-1} e^{ij\vartheta}$, we obtain

$$g = \mu\, e^{-i\vartheta} + (1 - 2\mu) + \mu\, e^{i\vartheta}$$

which is exactly the same as in (3.40)

3.5.3. Influence of lower-order terms

- Let us consider the model problem (3.8) augmented by lower-order terms

$$u_t = u_{xx} + a u_x + b u \tag{3.43}$$

  where a and b are constants.

- We can construct an explicit scheme

$$\frac{v_j^n - v_j^{n-1}}{\Delta t} = \frac{v_{j-1}^{n-1} - 2v_j^{n-1} + v_{j+1}^{n-1}}{\Delta x^2} + a\frac{v_{j+1}^{n-1} - v_{j-1}^{n-1}}{2\Delta x} + b\, v_j^{n-1} \tag{3.44}$$

- From the von Neumann analysis, we can obtain the amplification factor

$$g(\vartheta) = 1 - 4\mu \sin^2(\vartheta/2) + i\frac{a\Delta t}{\Delta x}\sin(\vartheta) + b\Delta t, \tag{3.45}$$

  which gives

$$\begin{aligned} |g(\vartheta)|^2 &= \big(1 - 4\mu\sin^2(\vartheta/2) + b\Delta t\big)^2 + \Big(\frac{a\Delta t}{\Delta x}\sin(\vartheta)\Big)^2 \\ &= \big(1 - 4\mu\sin^2(\vartheta/2)\big)^2 + 2\big(1 - 4\mu\sin^2(\vartheta/2)\big)b\Delta t \\ &\quad + (b\Delta t)^2 + \Big(\frac{a\Delta t}{\Delta x}\sin(\vartheta)\Big)^2 \end{aligned}$$

- Hence, under the condition $0 < \mu = \Delta t/\Delta x^2 \le 1/2$,

$$\begin{aligned} |g(\vartheta)|^2 &\le 1 + 2|b|\Delta t + (b\Delta t)^2 + \frac{|a|^2}{2}\Delta t \\ &\le \big(1 + (|b| + |a|^2/4)\,\Delta t\big)^2. \end{aligned} \tag{3.46}$$

- Thus, lower-order terms do not change the stability condition. (Homework for details.)

3.6. Boundedness – Maximum Principle

- Numerical solutions should lie between proper bounds.
- For example, physical quantities such as density and kinetic energy of turbulence must be positive, while concentration should be between 0 and 1.

In the absence of sources and sinks, some variables are required to have maximum and minimum values on the boundary of the domain. The above property is call the **maximum principle**, which should be inherited by the numerical approximation.

3.6.1. Convection-dominated fluid flows

To illustrate **boundedness** of the numerical solution of PDEs, we consider the **convection-diffusion problem**:

$$u_t - \varepsilon u_{xx} + a u_x = 0. \tag{3.47}$$

where $\varepsilon > 0$. When the spatial derivatives are approximated by central differences, the algebraic equation for u_j^n reads

$$u_j^n = u_j^{n-1} - \Big[\varepsilon \frac{-u_{j-1}^{n-1} + 2u_j^{n-1} - u_{j+1}^{n-1}}{\Delta x^2} + a \frac{u_{j+1}^{n-1} - u_{j-1}^{n-1}}{2\Delta x}\Big]\Delta t,$$

or

$$u_j^n = \Big(d + \frac{\sigma}{2}\Big) u_{j-1}^{n-1} + (1 - 2d) u_j^{n-1} + \Big(d - \frac{\sigma}{2}\Big) u_{j+1}^{n-1}, \tag{3.48}$$

where the dimensionless parameters are defined as

$$d = \frac{\varepsilon \Delta t}{\Delta x^2} \quad \text{and} \quad \sigma = \frac{a \Delta t}{\Delta x}.$$

- σ: the **Courant number**
- $\Delta x / a$: the characteristic convection time
- $\Delta x^2 / \varepsilon$: the characteristic diffusion time

  These are the time required for a disturbance to be transmitted by convection and diffusion over a distance Δx.

3.6.2. Stability vs. boundedness

The requirement that the coefficients of the old nodal values be nonnegative leads to

$$(1 - 2d) \geq 0, \qquad \frac{|\sigma|}{2} \leq d. \tag{3.49}$$

- The first condition leads to the limit on Δt as

  $$\Delta t \leq \frac{\Delta x^2}{2\varepsilon},$$

  which guarantees **stability** of (3.48). Recall that lower-order terms do not change the stability condition (§3.5.3).

- The second condition imposes no limit on the time step. But it gives a relation between convection and diffusion coefficients.

- The **cell Peclet number** is defined and bounded as

  $$\mathrm{Pe}_{\mathrm{cell}} := \frac{|\sigma|}{d} = \frac{|a|\Delta x}{\varepsilon} \leq 2. \tag{3.50}$$

  which is a sufficient (but not necessary) condition for **boundedness** of the solution of (3.48).

3.7. Conservation

- When the equations to be solved are from conservation laws, the numerical scheme should respect these laws both locally and globally. This means that the amount of a conserved quantity leaving a control volume is equal to the amount entering to adjacent control volumes.
- If divergence form of equations and a finite volume method is used, this is readily guaranteed for each individual control volume and for the solution domain as a whole.
- For other discretization methods, conservation can be achieved if care is taken in the choice of approximations. Sources and sinks should be carefully treated so that the net flux for each individual control volume is conservative.
- Conservation is a very important property of numerical schemes. Once conservation of mass, momentum, and energy is guaranteed, the error of conservative schemes is only due to an improper distribution of these quantities over the solution domain.
- Non-conservative schemes can produce artificial sources or sinks, changing the balance locally or globally. However, non-conservative schemes can be consistent and stable and therefore lead to correct solutions in the limit of mesh refinement; error due to non-conservation is appreciable in most cases only when the mesh is not fine enough.
- The problem is that it is difficult to know on which mesh the non-conservation error is small enough. Conservative schemes are thus preferred.

3.8. A Multi-Step Central Scheme

- Before we begin considering general implicit methods, we would like to mention an interesting scheme for solving (3.8):

$$\frac{v_j^{n+1} - v_j^{n-1}}{2\Delta t} - \frac{v_{j-1}^n - 2v_j^n + v_{j+1}^n}{\Delta x^2} = f_j^n, \tag{3.51}$$

  of which the truncation error

$$\textbf{Trunc.Err} = \mathcal{O}(\Delta t^2 + \Delta x^2). \tag{3.52}$$

- To study its stability, we set $f \equiv 0$ and substitute $v_j^n = g^n e^{ij\vartheta}$ into (3.51) to obtain

$$\frac{g - 1/g}{2\Delta t} - \frac{e^{-i\vartheta} - 2 + e^{i\vartheta}}{\Delta x^2} = 0,$$

  or

$$g^2 + (8\mu \sin^2(\vartheta/2))g - 1 = 0. \tag{3.53}$$

- We see that (3.53) has two distinct real roots g_1 and g_2 which should satisfy

$$g_1 \cdot g_2 = -1. \tag{3.54}$$

- Hence the magnitude of one root must be greater than one, for some modes and for all $\mu > 0$, for which we say that the scheme is **unconditionally unstable.**
- This example warns us that we need be careful when developing a FD scheme. We cannot simply put combinations of difference approximations together.

3.9. The θ-Method

Let $\mathcal{A}_1$ be the central second-order approximation of $-\partial_{xx}$, defined as

$$\mathcal{A}_1 v_j^n := -\frac{v_{j-1}^n - 2v_j^n + v_{j+1}^n}{\Delta x^2}.$$

Then the θ-method for (3.8), $u_t - u_{xx} = f$, is

$$\frac{v^n - v^{n-1}}{\Delta t} + \mathcal{A}_1 \left[\theta v^n + (1-\theta)v^{n-1}\right] = f^{n-1+\theta}, \tag{3.55}$$

for $\theta \in [0, 1]$, or equivalently

$$(I + \theta\Delta t\mathcal{A}_1)v^n = [I - (1-\theta)\Delta t\mathcal{A}_1]v^{n-1} + \Delta t f^{n-1+\theta}. \tag{3.56}$$

The following three choices of θ are popular.

- **Forward Euler method ($\theta = 0$)**: The algorithm (3.56) is reduced to

  $$v^n = (I - \Delta t\mathcal{A}_1)v^{n-1} + \Delta t f^{n-1}, \tag{3.57}$$

  which is the explicit scheme in (3.13), requiring the stability condition

  $$\mu = \frac{\Delta t}{\Delta x^2} \le \frac{1}{2}.$$

- **Backward Euler method ($\theta = 1$)**: This is an implicit method written as

  $$(I + \Delta t\mathcal{A}_1)v^n = v^{n-1} + \Delta t f^n. \tag{3.58}$$

  - The method must invert a tridiagonal matrix to get the solution in each time level.
  - But it is **unconditionally stable**, stable independently on the choice of Δt.

- **Crank-Nicolson method ($\theta = 1/2$)**:

$$\left(I + \frac{\Delta t}{2}\mathcal{A}_1\right)v^n = \left(I - \frac{\Delta t}{2}\mathcal{A}_1\right)v^{n-1} + \Delta t f^{n-1/2}. \tag{3.59}$$

  - It requires to solve a tridiagonal system in each time level, as in the backward Euler method.
  - However, the Crank-Nicolson method is most popular, because it is second-order in both space and time and unconditionally stable.
  - The Crank-Nicolson method can be viewed as an explicit method in the first half of the space-time slice $\mathcal{S}^n (:= \Omega \times (t^{n-1}, t^n])$ and an implicit method in the second half of $\mathcal{S}^n$. Hence it is often called a **semi-implicit** method.

3.9.1. Stability analysis for the θ-Method

Setting $f \equiv 0$, the algebraic system (3.56) reads pointwisely

$$\begin{aligned} &-\theta\mu\, v_{j-1}^n + (1 + 2\theta\mu)v_j^n - \theta\mu\, v_{j+1}^n \\ &\quad = (1-\theta)\mu\, v_{j-1}^{n-1} + [1 - 2(1-\theta)\mu]v_j^{n-1} + (1-\theta)\mu\, v_{j+1}^{n-1}, \end{aligned} \tag{3.60}$$

where $\mu = \Delta t/\Delta x^2$.

For an stability analysis for this one-parameter family of systems by utilizing the von Neumann analysis in §3.5.2, substitute $g^n e^{ij\vartheta}$ for v_j^n in (3.60) to have

$$\begin{aligned} &g\left[-\theta\mu\, e^{-i\vartheta} + (1 + 2\theta\mu) - \theta\mu\, e^{i\vartheta}\right] \\ &\quad = (1-\theta)\mu\, e^{-i\vartheta} + [1 - 2(1-\theta)\mu] + (1-\theta)\mu\, e^{i\vartheta}. \end{aligned}$$

That is,

$$\begin{aligned} g &= \frac{1 - 2(1-\theta)\mu\,(1 - \cos\vartheta)}{1 + 2\theta\mu\,(1 - \cos\vartheta)} \\ &= \frac{1 - 4(1-\theta)\mu\, \sin^2(\vartheta/2)}{1 + 4\theta\mu\, \sin^2(\vartheta/2)}. \end{aligned} \tag{3.61}$$

Because $\mu > 0$ and $\theta \in [0, 1]$, **the amplification factor g cannot be larger than one.**

The condition $g \geq -1$ is equivalent to

$$1 - 4(1-\theta)\mu \sin^2(\vartheta/2) \;\geq\; -\big[1 + 4\theta\mu \sin^2(\vartheta/2)\big],$$

or

$$(1-2\theta)\mu \sin^2 \frac{\vartheta}{2} \;\leq\; \frac{1}{2}.$$

Thus the θ-method (3.56) is stable if

$$(1-2\theta)\mu \;\leq\; \frac{1}{2}. \tag{3.62}$$

In conclusion:

- The θ-method is unconditionally stable for $\theta \geq 1/2$
- When $\theta < 1/2$, the method is stable only if

$$\mu = \frac{\Delta t}{\Delta x^2} \;\leq\; \frac{1}{2(1-2\theta)}, \quad \theta \in [0, 1/2). \tag{3.63}$$

3.9.2. Accuracy order

We shall choose $(x_j, t^{n-1/2})$ for the expansion point in the following derivation for the truncation error of the θ-method.

The arguments in §1.2 give

$$\frac{u_j^n - u_j^{n-1}}{\Delta t} = \left[u_t + \frac{u_{ttt}}{6}\left(\frac{\Delta t}{2}\right)^2 + \cdots\right]_j^{n-1/2}. \tag{3.64}$$

Also from the section, we have

$$\mathcal{A}_1 u_j^\ell = -\left[u_{xx} + \frac{u_{xxxx}}{12}\Delta x^2 + 2\frac{u_{xxxxxx}}{6!}\Delta x^4 + \cdots\right]_j^\ell, \quad \ell = n-1, n.$$

We now expand each term in the right side of the above equation in powers of Δt, about $(x_j, t^{n-1/2})$, to have

$$\begin{aligned} \mathcal{A}_1 u_j^{(n-\frac{1}{2})\pm\frac{1}{2}} \;=\; & -\left[u_{xx} + \frac{u_{xxxx}}{12}\Delta x^2 + 2\frac{u_{xxxxxx}}{6!}\Delta x^4 + \cdots\right]_j^{n-1/2} \\ & \mp\frac{\Delta t}{2}\left[u_{xxt} + \frac{u_{xxxxt}}{12}\Delta x^2 + 2\frac{u_{xxxxxxt}}{6!}\Delta x^4 + \cdots\right]_j^{n-1/2} \\ & -\frac{1}{2}\left(\frac{\Delta t}{2}\right)^2\left[u_{xxtt} + \frac{u_{xxxxtt}}{12}\Delta x^2 + \cdots\right]_j^{n-1/2} - \cdots. \end{aligned} \tag{3.65}$$

It follows from (3.64) and (3.65) that

$$\begin{aligned}\frac{u_j^n - u_j^{n-1}}{\Delta t} + \mathcal{A}_1\left[\theta u_j^n + (1-\theta)u_j^{n-1}\right] \;=\; & u_t + \frac{u_{ttt}}{6}\Big(\frac{\Delta t}{2}\Big)^2 + \mathcal{O}(\Delta t^4) \\ & -\Big(u_{xx} + \frac{u_{xxxx}}{12}\Delta x^2 + 2\frac{u_{xxxxxx}}{6!}\Delta x^4 + \cdots\Big) \\ & -\frac{\Delta t}{2}(2\theta-1)\Big(u_{xxt} + \frac{u_{xxxxt}}{12}\Delta x^2 + 2\frac{u_{xxxxxxt}}{6!}\Delta x^4 + \cdots\Big) \\ & -\frac{1}{2}\Big(\frac{\Delta t}{2}\Big)^2\Big(u_{xxtt} + \frac{u_{xxxxtt}}{12}\Delta x^2 + \cdots\Big) - \cdots,\end{aligned} \tag{3.66}$$

of which the right side is evaluated at $(x_j, t^{n-1/2})$.

So the truncation error $\mathcal{T}u(:= Pu - P_{\Delta x,\Delta t}u)$ turns out to be

$$\begin{aligned}\mathcal{T}u_j^{n-1/2} \;=\; & \Big(\theta - \frac{1}{2}\Big)u_{xxt}\Delta t + \frac{u_{xxxx}}{12}\Delta x^2 - \frac{u_{ttt}}{24}\Delta t^2 + \frac{u_{xxtt}}{8}\Delta t^2 \\ & +\Big(\theta - \frac{1}{2}\Big)\frac{u_{xxxxt}}{12}\Delta t\Delta x^2 + 2\frac{u_{xxxxxx}}{6!}\Delta x^4 + \cdots \\ =\; & \Big[\Big(\theta - \frac{1}{2}\Big)\Delta t + \frac{\Delta x^2}{12}\Big]u_{xxt} + \frac{\Delta t^2}{12}u_{ttt} \\ & +\Big[\Big(\theta - \frac{1}{2}\Big)\Delta t + \frac{\Delta x^2}{12}\Big]\frac{\Delta x^2}{12}u_{xxxxt} - \Big(\frac{1}{12^2} - \frac{2}{6!}\Big)u_{xxxxxx}\Delta x^4 + \cdots,\end{aligned} \tag{3.67}$$

where we have utilized $u_t = u_{xx} + f$.

Thus the accuracy order reads

$$\begin{cases} \mathcal{O}(\Delta t^2 + \Delta x^2) & \text{when } \theta = \dfrac{1}{2}, \\ \mathcal{O}(\Delta t^2 + \Delta x^4) & \text{when } \theta = \dfrac{1}{2} - \dfrac{\Delta x^2}{12\Delta t}, \\ \mathcal{O}(\Delta t + \Delta x^2) & \text{otherwise.} \end{cases} \tag{3.68}$$

Note: The second choice of θ in (3.68) is less than 1/2, which is **very special** and equivalent to

$$\frac{\Delta t}{\Delta x^2} = \frac{1}{6(1-2\theta)}.$$

Hence it satisfies (3.63); the method is stable and we can take large time steps while maintaining accuracy and stability. For example, when $\Delta x = \Delta t = 0.01$, we have $\theta = \frac{1}{2} - \frac{1}{1200}$ for the $(2,4)$-accuracy scheme in time-space.

3.9.3. Maximum principle

For heat conduction without interior sources/sinks, it is known mathematically and physically that the extreme values of the solution appear either in the initial data or on the boundary. This property is called the **maximum principle**.

- It is quite natural and sometimes very important to examine if the numerical solution satisfies the maximum principle.
- Once the scheme satisfies the maximum principle, the solution will never involve interior local extrema.

Theorem 3.13. (Maximum principle for the θ-method) *Let $f = 0$ and the θ-method be set satisfying $\theta \in [0, 1]$ and*

$$(1 - \theta)\mu \leq \frac{1}{2}. \tag{3.69}$$

If the computed solution v has an interior maximum or minimum, then v is constant.

Proof. We rewrite the component-wise expression of the θ-method, (3.60), in the form

$$\begin{aligned}(1 + 2\theta\mu)v_j^n = \theta\mu(v_{j-1}^n + v_{j+1}^n) + (1 - \theta)\mu(v_{j-1}^{n-1} + v_{j+1}^{n-1}) \\ +[1 - 2(1 - \theta)\mu]v_j^{n-1}.\end{aligned} \tag{3.70}$$

Under the hypotheses of the theorem all coefficients in the right side of the above equation are nonnegative and sum to $(1 + 2\theta\mu)$. Hence this leads to the conclusion that the interior point (x_j, t^n) can have a local maximum or minimum only if all five neighboring points, related to the right side of (3.70), have the same maximum or minimum value. The argument then implies that v has the same value at all grid points including those on the boundary. This completes the proof. □

3.9.4. Error analysis

Let

$$e_j^n = u_j^n - v_j^n,$$

where $u_j^n = u(x_j, t^n)$ with u being the exact solution of (3.8). Define

$$\mathcal{E}^n = \max_j |e_j^n|, \quad \mathcal{T}^{n-1/2} = \max_j |\mathcal{T} u_j^{n-1/2}|,$$

where $\mathcal{T} u_j^{n-1/2}$ is the truncation error at $(x_j, t^{n-1/2})$ defined in (3.67).

Theorem 3.14. *Let $\theta \in [0,1]$ and $(1-\theta)\mu \le \frac{1}{2}$ for the θ-method. Then,*

$$\mathcal{E}^n \le \Delta t \sum_{k=1}^{n} \mathcal{T}^{k-1/2}. \tag{3.71}$$

Proof. It follows from (3.71) that

$$\mathcal{E}^n \le n\Delta t \max_k \mathcal{T}^{k-1/2} \le T \max_k \mathcal{T}^{k-1/2}, \tag{3.72}$$

where T is the upper limit of the time variable. □

3.10. Homework

1. Prove Theorem 3.3. *Hint*: Use the proof of Theorem 3.1.
2. The energy method can be utilized to prove stability of the forward Euler scheme for $u_t - u_{xx} = 0$:
$$v_j^n = \mu\, v_{j-1}^{n-1} + (1-2\mu)\, v_j^{n-1} + \mu\, v_{j+1}^{n-1} \tag{3.73}$$
The analysis requires you to prove
$$\|v^n\|_{\Delta x}^2 \le (1 + C\Delta t)^2 \|v^{n-1}\|_{\Delta x}^2, \tag{3.74}$$
for some $C \ge 0$. Prove it, assuming $1 - 2\mu \ge 0$ and using the following hint
   - Start with squaring (3.73).
   - Apply the inequality $|ab| \le \dfrac{a^2 + b^2}{2}$.
   - Use the observation
$$\sum_j |v_{j-1}^{n-1}|^2 = \sum_j |v_j^{n-1}|^2 = \sum_j |v_{j+1}^{n-1}|^2$$
3. Verify (3.45) and (3.46).
4. Use the arguments in the proof of Example 3.8 on page 72 to prove Theorem 3.14.
5. This problem shows a different way of maximum principle for FD methods. Prove that the solution of the forward Euler method (3.12) satisfies
$$\min_j v_j^{n-1} \le v_j^n \le \max_j v_j^{n-1} \tag{3.75}$$
when $f \equiv 0$ and $\mu \le 1/2$.
6. Consider the problem in (3.28):
$$\begin{aligned} u_t - u_{xx} &= 0, && (x,t) \in [0,1] \times [0,1], \\ u &= 0, && (x,t) \in \{0,1\} \times [0,1], \\ u &= \sin(\pi x), && x \in [0,1], \ \ t = 0 \end{aligned} \tag{3.76}$$
   (a) Implement a code for the θ-method.
   (b) Compare its performances for $\theta = 0, 1, 1/2$.
   Choose $\Delta x = 1/10,\ 1/20$; set either $\Delta t = \Delta x$ or Δt to satisfy the stability limit.

CHAPTER 4

Elliptic Partial Differential Equations

Contents of Chapter 4

4.1. Regularity Estimates 96
4.2. Maximum and Minimum Principles 98
4.3. Discrete Maximum and Minimum Principles 101
4.4. Coordinate Changes 103
4.5. Cylindrical and Spherical Coordinates 105
4.6. Homework 108

4.1. Regularity Estimates

The **quasilinear second-order elliptic equation** in 2D is defined as

$$-\nabla \cdot (A(\mathbf{x})\nabla u) + b(\mathbf{x}, u, \nabla u) = f(\mathbf{x}), \tag{4.1}$$

where A is symmetric positive definite, i.e.,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}, \quad a_{11} > 0, \;\; a_{22} > 0, \quad a_{11}a_{22} > a_{12}^2.$$

- For simplicity, we begin with the constant coefficient linear equation

  $$-\nabla \cdot (A\nabla u) + \mathbf{b} \cdot \nabla u + cu = f, \tag{4.2}$$

  where $\mathbf{b} = (b_1, b_2)$.
- The **Fourier transform** in 2D reads

  $$\widehat{u}(\boldsymbol{\xi}) = \frac{1}{2\pi} \int_{\mathbb{R}^2} u(\mathbf{x}) e^{-i\mathbf{x}\cdot\boldsymbol{\xi}} d\mathbf{x};$$

  its inverse formula is

  $$u(\mathbf{x}) = \frac{1}{2\pi} \int_{\mathbb{R}^2} u(\boldsymbol{\xi}) e^{i\mathbf{x}\cdot\boldsymbol{\xi}} d\boldsymbol{\xi}.$$

- The Fourier transform satisfies the **Parseval's identity**

  $$\int_{\mathbb{R}^2} |u(\mathbf{x})|^2 d\mathbf{x} = \int_{\mathbb{R}^2} |\widehat{u}(\boldsymbol{\xi})|^2 d\boldsymbol{\xi}. \tag{4.3}$$

- Let $\partial_{\mathbf{x}} = (\partial_{x_1}, \partial_{x_2})$, where $\partial_{x_i} = \partial/\partial x_i$, $i = 1, 2$. For $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$, a pair of nonnegative integers, define

  $$|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2, \quad \boldsymbol{\xi}^{\boldsymbol{\alpha}} = \xi_1^{\alpha_1}\xi_2^{\alpha_2}, \quad \partial_{\mathbf{x}}^{\boldsymbol{\alpha}} = (\partial_{x_1}^{\alpha_1}, \partial_{x_2}^{\alpha_2}).$$

- Since

  $$\widehat{\partial_{\mathbf{x}}^{\boldsymbol{\alpha}} u} = i^{|\boldsymbol{\alpha}|}\, \boldsymbol{\xi}^{\boldsymbol{\alpha}}\, \widehat{u}, \tag{4.4}$$

  Equation (4.2) in its Fourier transform becomes

  $$P(\boldsymbol{\xi})\, \widehat{u}(\boldsymbol{\xi}) = \widehat{f}(\boldsymbol{\xi}), \tag{4.5}$$

  where

  $$P(\boldsymbol{\xi}) = \boldsymbol{\xi} \cdot A\boldsymbol{\xi} + i\mathbf{b} \cdot \boldsymbol{\xi} + c.$$

- From the ellipticity requirements: $a_{11} > 0$, $a_{22} > 0$, and $a_{11}a_{22} > a_{12}^2$, we see
$$\boldsymbol{\xi} \cdot A\boldsymbol{\xi} \geq C_0|\boldsymbol{\xi}|^2,$$
for some $C_0 > 0$. Thus there are $C_1 > 0$ and $R_0 \geq 0$ such that
$$|P(\boldsymbol{\xi})| \geq C_1|\boldsymbol{\xi}|^2, \quad \text{if } |\boldsymbol{\xi}| \geq R_0, \tag{4.6}$$
and therefore we have
$$|\widehat{u}(\boldsymbol{\xi})| \leq C_2\frac{|\widehat{f}(\boldsymbol{\xi})|}{|\boldsymbol{\xi}|^2}, \quad \text{if } |\boldsymbol{\xi}| \geq R_0, \tag{4.7}$$
for some $C_2 > 0$.
- Thus, from (4.3), (4.4), and (4.7),
$$\begin{aligned}\int_{\mathbb{R}^2} |\partial_{\mathbf{x}}^{\boldsymbol{\alpha}} u|^2 d\mathbf{x} &= \int_{\mathbb{R}^2} |\boldsymbol{\xi}^{\boldsymbol{\alpha}}\widehat{u}|^2 d\boldsymbol{\xi} \\ &\leq \int_{|\boldsymbol{\xi}|\leq R_0} |\boldsymbol{\xi}|^{2|\boldsymbol{\alpha}|}\,|\widehat{u}|^2 d\boldsymbol{\xi} + \int_{|\boldsymbol{\xi}|\geq R_0} C_2|\boldsymbol{\xi}|^{2|\boldsymbol{\alpha}|}\,\frac{|\widehat{f}|^2}{|\boldsymbol{\xi}|^2} d\boldsymbol{\xi} \\ &\leq R_0^{2|\boldsymbol{\alpha}|}\int_{\mathbb{R}^2} |\widehat{u}|^2 d\boldsymbol{\xi} + C_2\int_{\mathbb{R}^2} |\boldsymbol{\xi}|^{2|\boldsymbol{\alpha}|-2}\,|\widehat{f}|^2 d\boldsymbol{\xi}.\end{aligned} \tag{4.8}$$
- For nonnegative integer s, the $H^s(\mathbb{R}^2)$-norm is defined as
$$\|u\|_s^2 = \sum_{|\boldsymbol{\alpha}|\leq s}\int_{\mathbb{R}^2} |\partial_{\mathbf{x}}^{\boldsymbol{\alpha}} u|^2 d\mathbf{x}.$$
Then, it follows from (4.8) and the Parseval's identity that
$$\|u\|_{s+2}^2 \leq C(\|f\|_s^2 + \|u\|_0^2), \quad s \geq 0, \tag{4.9}$$
for some $C = C(s, A, \mathbf{b}, c) > 0$. The inequality (4.9) is called a **regularity estimate**.

Note: When $\mathbf{b} = 0$ and $c \geq 0$, (4.6) holds with $R_0 = 0$. Thus the regularity estimate reads
$$\|u\|_{s+2} \leq C\|f\|_s, \quad s \geq 0, \quad \text{if } \mathbf{b} = 0 \text{ and } c \geq 0. \tag{4.10}$$

When (4.2) is defined on bounded domain $\Omega \subset \mathbb{R}^2$ whose boundary is sufficiently smooth, one can obtain an **interior regularity estimate** of the form
$$\|u\|_{s+2,\Omega_1}^2 \leq C(\|f\|_{s,\Omega}^2 + \|u\|_{0,\Omega}^2), \quad s \geq 0, \tag{4.11}$$
where $\Omega_1 \subset \Omega$ is such that its boundary is contained in the interior of Ω, and the constant $C = C(s, A, \mathbf{b}, c, \Omega, \Omega_1) > 0$.

4.2. Maximum and Minimum Principles

This section presents the maximum and minimum principles for subharmonic and superharmonic functions, respectively, following Gilberg and Trudinger [26, Ch.2].

Definition 4.1. The function u is called *harmonic* (*subharmonic, superharmonic*) in $\Omega \subset \mathbb{R}^n$ if it satisfies

$$-\Delta u = 0 \ \ (\leq 0, \geq 0), \quad \mathbf{x} \in \Omega. \tag{4.12}$$

The following is known as the **mean value theorems**, which characterize harmonic functions.

Theorem 4.2. *Let $u \in C^2(\Omega)$ satisfy $-\Delta u = 0 \ \ (\leq 0, \geq 0)$ in Ω. Then, for any ball $B = B_R(\mathbf{y}) \Subset \Omega$, we have*

$$\begin{aligned} u(\mathbf{y}) &= \ (\leq, \geq) \frac{1}{|\partial B|} \int_{\partial B} u \, ds, \\ u(\mathbf{y}) &= \ (\leq, \geq) \frac{1}{|B|} \int_{B} u \, d\mathbf{x}. \end{aligned} \tag{4.13}$$

With the aid of Theorem 4.2, the **strong maximum principle** for subharmonic functions and the **strong minimum principle** for superharmonic functions can be derived as follows.

Theorem 4.3. *Let $-\Delta u \leq 0 \ \ (\geq 0)$ in Ω and suppose there is a point $\mathbf{y} \in \Omega$ such that*

$$u(\mathbf{y}) = \sup_{\Omega} u \ \ (\inf_{\Omega} u). \tag{4.14}$$

Then u is constant. Therefore a harmonic function cannot assume an interior maximum or minimum value unless it is constant.

Proof. Let $-\Delta u \leq 0$ in Ω, $M = \sup_\Omega u$ and $\Omega_M = \{\mathbf{x} \in \Omega : u(\mathbf{x}) = M\}$. By assumption, $\Omega_M \neq \emptyset$. Furthermore since u is continuous, Ω_M is closed relative to Ω. We are going to show Ω_M is also open relative to Ω to conclude $\Omega_M = \Omega$. Let $\mathbf{z}$ is a point in Ω_M. Apply the mean value inequality (4.13) to the

subharmonic function $u - M$ in a ball $B = B_R(\mathbf{z}) \subset\subset \Omega$ to get

$$0 = u(\mathbf{z}) - M \leq \frac{1}{|B|} \int_B (u - M)\, d\mathbf{x} \leq 0.$$

Since $u - M \leq 0$ in in $B_R(\mathbf{z})$, we must have $u = M$ in $B_R(\mathbf{z})$, which implies Ω_M is open. The result for superharmonic functions follows by replacing u by $-u$. □

Theorem 4.3 implies the following **weak maximum principle** and **weak minimum principle**.

Theorem 4.4. *Let $u \in C^2(\Omega) \cap C^0(\overline{\Omega})$ with $-\Delta u \leq 0$ (≥ 0) in Ω. Then, provided that Ω is bounded,*

$$\sup_{\Omega} u = \sup_{\partial\Omega} u \quad (\inf_{\Omega} u = \inf_{\partial\Omega} u).$$

Therefore, for a harmonic function u,

$$\inf_{\partial\Omega} u \leq u(\mathbf{x}) \leq \sup_{\partial\Omega} u, \quad \mathbf{x} \in \Omega. \tag{4.15}$$

The uniqueness theorem for the classical Dirichlet problem for the Poisson equation in bounded domains follows from Theorem 4.4.

Theorem 4.5. *Let $u, v \in C^2(\Omega) \cap C^0(\overline{\Omega})$ satisfy $-\Delta u = -\Delta v$ in Ω and $u = v$ on $\partial\Omega$. Then $u = v$ in Ω.*

Proof. See Homework 4.1. □

Now, consider the linear elliptic operator of the form

$$Lu = -\nabla \cdot (A(\mathbf{x})\nabla u) + \mathbf{b}(\mathbf{x}) \cdot \nabla u + c(\mathbf{x})u. \tag{4.16}$$

Definition 4.6. A function u satisfying $Lu = 0 \ (\leq 0, \geq 0)$ in Ω is called a **solution** (**subsolution**, **supersolution**) of $Lu = 0$ in Ω.

Analogues to Theorems 4.4 and 4.5 can be proved for L. See [26, §3.1] for proofs.

Theorem 4.7. *Let L be elliptic in a bounded domain Ω with $c = 0$. Suppose $u \in C^2(\Omega) \cap C^0(\overline{\Omega})$ with $Lu \leq 0 \ (\geq 0)$ in Ω. Then*

$$\sup_{\Omega} u = \sup_{\partial\Omega} u \ \ (\inf_{\Omega} u = \inf_{\partial\Omega} u).$$

Theorem 4.8. *Let L be elliptic with $c \geq 0$. Suppose $u, v \in C^2(\Omega) \cap C^0(\overline{\Omega})$ satisfy $Lu = Lv$ in Ω and $u = v$ on $\partial\Omega$. Then $u = v$ in Ω. If $Lu \leq Lv$ in Ω and $u \leq v$ on $\partial\Omega$, then $u \leq v$ in Ω.*

4.3. Discrete Maximum and Minimum Principles

Let Δ_h be the discrete five-point Laplacian defined on grid points $\Omega_h = \{\mathbf{x}_{pq} \in \overline{\Omega}\}$, where h is the grid size and Ω is a bounded region in 2D.

Theorem 4.9. *Let Ω be a rectangular region and $-\Delta_h u \leq 0$ (≥ 0) on Ω_h. If u has an interior maximum (minimum), then u is constant on Ω_h. Therefore*

$$\max_{\Omega_h} u = \max_{\partial\Omega_h} u \quad (\min_{\Omega_h} u = \min_{\partial\Omega_h} u). \tag{4.17}$$

Proof. First, consider the case $-\Delta_h u \leq 0$; let u have a maximum value at an interior point $\mathbf{x}_{pq}$. The condition $-\Delta_h u \leq 0$ is equivalent to

$$u_{pq} \leq \frac{1}{2+2r^2}(u_{p-1,q} + u_{p+1,q} + r^2 u_{p,q-1} + r^2 u_{p,q+1}), \tag{4.18}$$

where $r = h_x/h_y$. Hence this easily leads to the conclusion that the interior point $\mathbf{x}_{pq}$ can have a (local) maximum only if all neighboring points have the same maximum value and that the inequality is actually an equality.

The argument then implies that u has the same value at all grid points including those on the boundary. This proves the discrete maximum principle for $-\Delta_h u \leq 0$.

Now, the discrete minimum principle for the superharmonic functions can be proved by replacing u by $-u$ and following the same argument. □

The following generalizes Theorem 4.9.

Theorem 4.10. *Let $L = -\nabla \cdot A(\mathbf{x})\nabla + \mathbf{b}(\mathbf{x}) \cdot \nabla$ be an elliptic operator defined in a rectangular region Ω, where $A(\mathbf{x}) = \operatorname{diag}(a_{11}(\mathbf{x}), a_{22}(\mathbf{x}))$, and L_h be the a five-point FD discretization of L. Assume that h is sufficiently small when $\mathbf{b} \neq 0$. Suppose a function u satisfies $L_h u \leq 0$ (≥ 0) on Ω_h and has an interior maximum (minimum), then u is constant on Ω_h. Thus*

$$\max_{\Omega_h} u = \max_{\partial\Omega_h} u \quad (\min_{\Omega_h} u = \min_{\partial\Omega_h} u) \tag{4.19}$$

and therefore, for a solution u of $L_h u = 0$,

$$\inf_{\partial\Omega_h} u \leq u(\mathbf{x}) \leq \sup_{\partial\Omega_h} u, \quad \mathbf{x} \in \Omega_h. \tag{4.20}$$

Proof. Let u have a maximum at an interior point $\mathbf{x}_{pq}$. The condition $L_h u \leq 0$ is equivalent to

$$u_{pq} \leq \frac{1}{a_{pq}^{pq}} \left(-a_{p-1,q}^{pq}\, u_{p-1,q} - a_{p+1,q}^{pq}\, u_{p+1,q} - a_{p,q-1}^{pq}\, u_{p,q-1} - a_{p,q+1}^{pq}\, u_{p,q+1} \right), \tag{4.21}$$

where a_{rs}^{pq} is the matrix entry corresponding to the relationship of L_h from u_{pq} to u_{rs}. Note that for five-point FD schemes,

$$a_{pq}^{pq} = -(a_{p-1,q}^{pq} + a_{p+1,q}^{pq} + a_{p,q-1}^{pq} + a_{p,q+1}^{pq}) > 0. \tag{4.22}$$

When $\mathbf{b} = 0$, it is easy to see that the coefficients a_{rs}^{pq}, $(pq) \neq (rs)$, are all strictly negative; for the case $\mathbf{b} \neq 0$, one needs to choose the grid size h sufficiently small in order for the four off-diagonal entries of the algebraic system to remain negative.

Now, let u_{pq} be an interior (local) maximum. Then it follows from (4.21), (4.22), and $a_{rs}^{pq} < 0$, $(pq) \neq (rs)$, that all the neighboring values must be the same as the maximum, which implies u is constant on Ω_h.

This proves the discrete maximum principle for subsolutions. As in the proof of Theorem 4.9, the discrete minimum principle for supersolutions can be proved by replacing u by $-u$ and following the same argument. □

See Exercise 5.7, on page 156, for the maximum principle applied for more general elliptic problems.

4.4. Coordinate Changes

Note: (**Coordinate Change**)
Often we have to solve the PDEs on a domain that is not a rectangle or other easy shape. In the case it is desirable to change coordinates so that the solution can be computed in a convenient coordinate system.

We begin with the elliptic equation

$$-\nabla \cdot (A(\mathbf{x})\nabla u) = f(\mathbf{x}), \tag{4.23}$$

where $A = [a_{ij}]$ is symmetric positive definite. Let $\boldsymbol{\xi}$ be another coordinate system:

$$\boldsymbol{\xi} = \boldsymbol{\xi}(\mathbf{x}). \tag{4.24}$$

Then, since

$$\frac{\partial u}{\partial x_j} = \sum_i \frac{\partial u}{\partial \xi_i}\frac{\partial \xi_i}{\partial x_j} = \begin{bmatrix} \frac{\partial \xi_1}{\partial x_j} & \frac{\partial \xi_2}{\partial x_j} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial \xi_1} \\ \frac{\partial u}{\partial \xi_2} \end{bmatrix}, \tag{4.25}$$

we see

$$\nabla_{\mathbf{x}} = J^T \nabla_{\boldsymbol{\xi}}, \quad J = \left[\frac{\partial \xi_i}{\partial x_j}\right], \tag{4.26}$$

where J is the **Jacobian** to be expressed in terms of $\boldsymbol{\xi}$. Thus

$$\nabla_{\mathbf{x}} \cdot A \nabla_{\mathbf{x}} = \nabla_{\boldsymbol{\xi}} \cdot JAJ^T \nabla_{\boldsymbol{\xi}}. \tag{4.27}$$

Note:

- Note that the Jacobian J must be expressed in terms of $\boldsymbol{\xi}$.
- The new diffusivity tensor $B(:= JAJ^T)$ is symmetric; its positiveness can be shown for certain cases.

Example 4.11. Consider the **Poisson equation** defined on a **trapezoidal domain**:

$$\Omega = \{(x_1, x_2) : 0 < x_1 < 1,\ \ 0 < x_2 < (1 + x_1)/2\}.$$

Define a new coordinate system $\boldsymbol{\xi} \in (0, 1)^2$,

$$\xi_1 = x_1, \quad \xi_2 = \frac{2x_2}{1 + x_1}.$$

Then the Jacobian reads

$$J = \begin{bmatrix} 1 & 0 \\ -2x_2/(1 + x_1)^2 & 2/(1 + x_1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\xi_2/(1 + \xi_1) & 2/(1 + \xi_1) \end{bmatrix} \tag{4.28}$$

and

$$B = JAJ^T = JJ^T = \begin{bmatrix} 1 & -\dfrac{\xi_2}{1 + \xi_1} \\ -\dfrac{\xi_2}{1 + \xi_1} & \dfrac{\xi_2^2 + 4}{(1 + \xi_1)^2} \end{bmatrix}. \tag{4.29}$$

The matrix $B(\boldsymbol{\xi})$ is clearly symmetric and positive definite on the unit square. The problem

$$-\nabla \cdot B(\boldsymbol{\xi})\nabla\, u = f(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in (0, 1)^2, \tag{4.30}$$

can be approximated by the standard second-order FD method.

4.5. Cylindrical and Spherical Coordinates

Definition 4.12. *In the **cylindrical coordinate system**, a **point** P in the 3D space is represented as an ordered triple (r, θ, z), where r and θ are polar coordinates of the projection of P onto the xy-plane and z is the directed distance from the xy-plane to P.*

Definition 4.13. *The conversion between the **Cylindrical Coordinates** and the Rectangular Coordinate system gives*

$(x, y, z)_R \leftarrow (r, \theta, z)_C$	$(r, \theta, z)_C \leftarrow (x, y, z)_R$
$x = r\cos\theta$	$r^2 = x^2 + y^2$
$y = r\sin\theta$	$\tan\theta = \dfrac{y}{x}$
$z = z$	$z = z$

(4.31)

It follows from (4.31) that

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial u}{\partial \theta}\frac{\partial \theta}{\partial x} = \frac{x}{r}\frac{\partial u}{\partial r} - \frac{y}{r^2}\frac{\partial u}{\partial \theta} = \cos\theta\,\frac{\partial u}{\partial r} - \frac{\sin\theta}{r}\,\frac{\partial u}{\partial \theta},$$

where, as in Example 4.11, (x, y) is expressed in terms of (r, θ).

Replacing the function u in the above equation by $\dfrac{\partial u}{\partial x}$, we see

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &= \cos\theta\,\frac{\partial}{\partial r}\left(\frac{\partial u}{\partial x}\right) - \frac{\sin\theta}{r}\,\frac{\partial}{\partial \theta}\left(\frac{\partial u}{\partial x}\right) \\
&= \cos\theta\,\frac{\partial}{\partial r}\left(\cos\theta\,\frac{\partial u}{\partial r} - \frac{\sin\theta}{r}\,\frac{\partial u}{\partial \theta}\right) - \frac{\sin\theta}{r}\,\frac{\partial}{\partial \theta}\left(\cos\theta\,\frac{\partial u}{\partial r} - \frac{\sin\theta}{r}\,\frac{\partial u}{\partial \theta}\right) \\
&= \cos^2\theta\,\frac{\partial^2 u}{\partial r^2} - \frac{2\sin\theta\,\cos\theta}{r}\,\frac{\partial^2 u}{\partial \theta \partial r} + \frac{\sin^2\theta}{r^2}\,\frac{\partial^2 u}{\partial \theta^2} \\
&\quad + \frac{\sin^2\theta}{r}\,\frac{\partial u}{\partial r} + \frac{2\sin\theta\,\cos\theta}{r^2}\,\frac{\partial u}{\partial \theta}.
\end{aligned} \tag{4.32}$$

In the same way, one can show that

$$\frac{\partial u}{\partial y} = \sin\theta\,\frac{\partial u}{\partial r} + \frac{\cos\theta}{r}\,\frac{\partial u}{\partial \theta}$$

and

$$\begin{aligned}\frac{\partial^2 u}{\partial y^2} &= \sin^2\theta\,\frac{\partial^2 u}{\partial r^2} + \frac{2\sin\theta\,\cos\theta}{r}\,\frac{\partial^2 u}{\partial\,\theta\partial r} + \frac{\cos^2\theta}{r^2}\,\frac{\partial^2 u}{\partial\theta^2} \\ &\quad + \frac{\cos^2\theta}{r}\,\frac{\partial u}{\partial r} - \frac{2\sin\theta\,\cos\theta}{r^2}\,\frac{\partial u}{\partial\theta}.\end{aligned} \tag{4.33}$$

Claim 4.14. From (4.32) and (4.33), the Laplacian of u in **cylindrical coordinates** is

$$\begin{aligned}\Delta u &= \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial\theta^2} + \frac{\partial^2 u}{\partial z^2} \\ &= \frac{1}{r}(ru_r)_r + \frac{1}{r^2}u_{\theta\theta} + u_{zz}.\end{aligned} \tag{4.34}$$

Spherical Coordinates

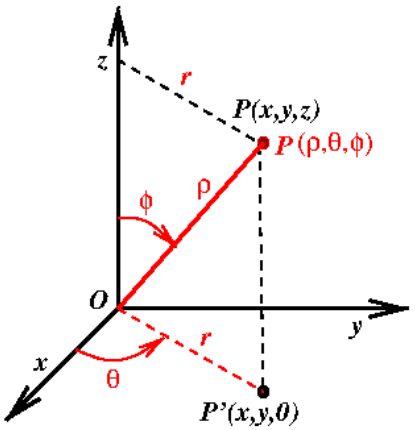


Figure 4.1: Spherical coordinates of P.

Definition 4.15. *The **spherical coordinates** (ρ, θ, ϕ) of a point P is shown in Figure 4.1, where $\rho = |\overline{OP}| = \sqrt{x^2+y^2+z^2}$, θ is the angle from the x-axis to the line segment $\overline{OP'}$, and ϕ is the angle between the positive z-axis and the line segment $\overline{OP}$.*

Note: By observing the definition, we can see the following inequalities:

$$\rho \geq 0 \qquad 0 \leq \theta \leq 2\pi, \qquad 0 \leq \phi \leq \pi.$$

For a **convenient conversion formula**, consider first

$$\mathbf{z = \rho \cos \phi, \quad r = \rho \sin \phi,} \tag{4.35}$$

then use

$$\boldsymbol{x = r \cos \theta, \quad y = r \sin \theta.}$$

Definition 4.16. *The conversion between the **Spherical Coordinates** and the Rectangular Coordinate system gives*

$(x,y,z)_R \leftarrow (\rho,\theta,\phi)_S$	$(\rho,\theta,\phi)_S \leftarrow (x,y,z)_R$
$\boldsymbol{x = r \cos \theta} = \rho \sin\phi\cos\theta$	$\rho^2 = x^2 + y^2 + z^2$
$\boldsymbol{y = r \sin \theta} = \rho \sin\phi\sin\theta$	$\cos\phi = \dfrac{z}{\rho}$
$z = \rho \cos\phi$	$\cos\theta = \dfrac{x}{\rho \sin\phi}$

(4.36)

Claim 4.17. Using the arguments for the cylindrical coordinates, one can see that the Laplacian of u in **spherical coordinates** is

$$\begin{aligned} \Delta u &= \frac{\partial^2 u}{\partial \rho^2} + \frac{2}{\rho}\frac{\partial u}{\partial \rho} + \frac{1}{\rho^2}\frac{\partial^2 u}{\partial \phi^2} + \frac{\cot\phi}{\rho^2}\frac{\partial u}{\partial \phi} + \frac{1}{\rho^2 \sin^2\phi}\frac{\partial^2 u}{\partial \theta^2} \\ &= \frac{1}{\rho^2}(\rho^2 u_\rho)_\rho + \frac{1}{\rho^2 \sin\phi}(u_\phi \sin\phi)_\phi + \frac{1}{\rho^2 \sin^2\phi}\, u_{\theta\theta}. \end{aligned} \tag{4.37}$$

Proof. See Homework 4.3. □

4.6. Homework

1. Prove Theorem 4.5.
2. Consider the **Poisson equation** defined on a region:

$$\Omega = \{(x_1, x_2) : 0 < x_1 < 2,\ \ 0 < x_2 < e^{-x_1/2}\}.$$

   Define a change of variables $\boldsymbol{\xi} = \boldsymbol{\xi}(\mathrm{x})$ to derive a new coordinate system for the Poisson equation to be defined on a rectangle in the new coordinates.
3. Prove Claim 4.17.

CHAPTER 5

Finite Difference Methods for Elliptic Equations

This chapter introduces finite difference methods for elliptic PDEs defined on 1-dimensional (1D), 2-dimensional (2D), or 3-dimensional (3D) regions.

Contents of Chapter 5

5.1. Finite Difference (FD) Methods . . . 110
5.2. Solution of Linear Algebraic Systems . . . 122
5.3. Relaxation methods . . . 133
5.4. Krylov Subspace Methods . . . 140
5.5. Other Iterative Methods . . . 147
5.6. Numerical Examples with Python . . . 150
5.7. Homework . . . 155

5.1. Finite Difference (FD) Methods

Let $\Omega = (a_x, b_x) \times (a_y, b_y)$ in 2D space. Consider the model problem

$$\begin{array}{ll} \text{(a)} & -\nabla \cdot (a\nabla u) + cu = f, \quad \mathbf{x} \in \Omega \\ \text{(b)} & au_\nu + \beta u = g, \qquad\quad \mathbf{x} \in \Gamma, \end{array} \tag{5.1}$$

where the diffusivity $a(\mathbf{x}) > 0$ and the coefficient $c(\mathbf{x}) \geq 0$.

- When $c \equiv 0$ and $\beta \equiv 0$, the problem (5.1) has infinitely many solutions.
  - If $u(\mathbf{x})$ is a solution, so is $u(\mathbf{x}) + C$, for $\forall\, C \in \mathbb{R}$.
  - Also we can see that the corresponding algebraic system is singular.
  - The singularity is not a big issue in numerical simulation; one may impose a Dirichlet condition at a grid point on the boundary.
- We may assume that (5.1) admits a unique solution.

To explain the main feature of the central FD method, we may start with the problem (5.1) with the constant diffusivity, i.e., $a = 1$.

5.1.1. Constant-coefficient problems

Consider the following simplified problem ($a \equiv 1$):

$$\begin{aligned} -u_{xx} - u_{yy} + cu &= f(x,y), \quad (x,y) \in \Omega, \\ u_\nu + \beta u &= g(x,y), \quad (x,y) \in \Gamma, \end{aligned} \tag{5.2}$$

To make it even simpler, we may start with the 1D problem:

$$\begin{array}{lrcll} \text{(a)} & -u_{xx} + cu & = & f, & x \in (a_x, b_x), \\ \text{(b)} & -u_x + \beta u & = & g, & x = a_x, \\ \text{(c)} & u_x + \beta u & = & g, & x = b_x. \end{array} \tag{5.3}$$

Finite Difference Approximation

- Select n_x equally spaced grid points on the interval $[a_x, b_x]$:

$$x_i = a_x + ih_x, \quad i = 0, 1, \cdots, n_x, \quad h_x = \frac{b_x - a_x}{n_x}.$$

- Let $u_i = u(x_i)$ and recall (1.17) on page 7:

$$-u_{xx}(x_i) \approx \frac{-u_{i-1} + 2u_i - u_{i+1}}{h_x^2} + \frac{u_{xxxx}(x_i)}{12} h_x^2 + \cdots . \tag{5.4}$$

- Apply the FD scheme for (5.3.a) to have

$$-u_{i-1} + (2 + h_x^2 c)u_i - u_{i+1} = h_x^2 f_i. \tag{5.5}$$

- However, we will meet ghost grid values at the end points. For example, at the point $a_x = x_0$, the formula becomes

$$-u_{-1} + (2 + h_x^2 c)u_0 - u_1 = h_x^2 f_0. \tag{5.6}$$

  Here the value u_{-1} is not defined and we call it a **ghost grid value**.

- Now, let's replace the value by using the boundary condition (5.3.b). Recall the central FD scheme (1.16) for u_x at x_0:

$$u_x(x_0) \approx \frac{u_1 - u_{-1}}{2h_x}, \quad \textbf{Trunc.Err} = -\frac{u_{xxx}(x_0)}{6} h_x^2 + \cdots . \tag{5.7}$$

  Thus he equation (5.3.b) can be approximated (at x_0)

$$u_{-1} + 2h_x \beta u_0 - u_1 = 2h_x g_0. \tag{5.8}$$

- Hence it follows from (5.6) and (5.8) that

$$(2 + h_x^2 c + 2h_x \beta)u_0 - 2u_1 = h_x^2 f_0 + 2h_x g_0. \tag{5.9}$$

  The same can be considered for the algebraic equation at the point x_n.

The problem (5.3) is reduced to finding the solution $\mathbf{u}_1$ satisfying

$$A_1\mathbf{u}_1 = \mathbf{b}_1, \tag{5.10}$$

where

$$A_1 = \begin{bmatrix} 2+h_x^2c+2h_x\beta & -2 & & & & \\ -1 & 2+h_x^2c & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2+h_x^2c & -1 & \\ & & & -2 & 2+h_x^2c+2h_x\beta & \end{bmatrix},$$

and

$$\mathbf{b}_1 = \begin{bmatrix} h_x^2f_0 \\ h_x^2f_1 \\ \vdots \\ h_x^2f_{n_x-1} \\ h_x^2f_{n_x} \end{bmatrix} + \begin{bmatrix} 2h_xg_0 \\ 0 \\ \vdots \\ 0 \\ 2h_xg_{n_x} \end{bmatrix}.$$

Such a technique of removing ghost grid values is called **outer bordering**. We can use it for the 2D problem (5.2) along the boundary grid points.

Note: **Symmetrization**: The matrix A_1 is not symmetric! You can symmetrize it by dividing the first and the last rows of $[A_1|\mathbf{b}_1]$ by 2. For the 2D problem, you have to apply "division by 2" along each side of boundaries. (So, the algebraic equations corresponding to the corner points would be divided by a total factor of 4, for a symmetric algebraic system.)

5.1.2. General diffusion coefficients

Let the 1D problem read

$$\begin{array}{lrcll} \text{(a)} & -(au_x)_x + cu & = & f, & x \in (a_x, b_x), \\ \text{(b)} & -au_x + \beta u & = & g, & x = a_x, \\ \text{(c)} & au_x + \beta u & = & g, & x = b_x. \end{array} \tag{5.11}$$

The central FD scheme for $(au_x)_x$ can be obtained as follows.

- The term (au_x) can be viewed as a function and approximated as

$$(au_x)_x(x_i) \approx \frac{(au_x)_{i+1/2} - (au_x)_{i-1/2}}{h_x} + \mathcal{O}(h_x^2), \tag{5.12}$$

  where $(au_x)_{i+1/2}$ denotes the value of (au_x) evaluated at $x_{i+1/2} := (x_i + x_{i+1})/2$.
- The terms $(au_x)_{i+1/2}$ and $(au_x)_{i-1/2}$ can be again approximated as

$$\begin{aligned} (au_x)_{i+1/2} &\approx a_{i+1/2}\frac{u_{i+1} - u_i}{h_x} - a_{i+1/2}\frac{u_{xxx}(x_{i+1/2})}{3!}\left(\frac{h_x}{2}\right)^2 + \cdots, \\ (au_x)_{i-1/2} &\approx a_{i-1/2}\frac{u_i - u_{i-1}}{h_x} - a_{i-1/2}\frac{u_{xxx}(x_{i-1/2})}{3!}\left(\frac{h_x}{2}\right)^2 + \cdots. \end{aligned} \tag{5.13}$$

- Combine the above two equations to have

$$-(au_x)_x(x_i) \approx \frac{-a_{i-1/2}u_{i-1} + (a_{i-1/2} + a_{i+1/2})u_i - a_{i+1/2}u_{i+1}}{h_x^2}, \tag{5.14}$$

  of which the overall truncation error becomes $\mathcal{O}(h_x^2)$. See Homework 5.1 on page 155.

Note:

- The y-directional approximation can be done in the same fashion.
- The reader should also notice that the quantities $a_{i+1/2}$ evaluated at mid-points are not available in general.
- We may replace it by the arithmetic/harmonic average of a_i and a_{i+1}:

$$a_{i+1/2} \approx \frac{a_i + a_{i+1}}{2} \quad \text{or} \quad \left[\frac{1}{2}\left(\frac{1}{a_i} + \frac{1}{a_{i+1}}\right)\right]^{-1}. \tag{5.15}$$

- The harmonic average is preferred; the resulting system holds the conservation property. See §6.8.

5.1.3. FD schemes for mixed derivatives

The linear elliptic equation in its general form is given as

$$-\nabla \cdot (A(\mathbf{x})\nabla u) + \mathbf{b} \cdot \nabla u + cu = f, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \tag{5.16}$$

where $1 \le d \le 3$ and

$$-\nabla \cdot (A(\mathbf{x})\nabla u) = -\sum_{i,j} \frac{\partial}{\partial x_i}\left(a_{ij}(\mathbf{x})\frac{\partial u}{\partial x_j}\right).$$

Thus we must approximate the mixed derives whenever they appear.

As an example, we consider a second-order FD scheme for $(au_x)_y$ on a mesh of grid size $h_x \times h_y$:

$$\begin{aligned}(au_x)_y(\mathbf{x}_{pq}) &\approx \frac{au_x(\mathbf{x}_{p,q+1}) - au_x(\mathbf{x}_{p,q-1})}{2h_y} + \mathcal{O}(h_y^2) \\ &\approx \frac{a_{p,q+1}(u_{p+1,q+1} - u_{p-1,q+1}) - a_{p,q-1}(u_{p+1,q-1} - u_{p-1,q-1})}{4h_x h_y} \\ &\quad + \mathcal{O}(h_x^2) + \mathcal{O}(h_y^2).\end{aligned} \tag{5.17}$$

- There may involve difficulties in FD approximations when the diffusion coefficient A is a full tensor.
- Scalar coefficients can also become a full tensor when coordinates are changed.

5.1.4. L^∞-norm error estimates for FD schemes

Let Ω be a rectangular domain in 2D and $\Gamma = \partial\Omega$. Consider

$$\begin{aligned} -\Delta u &= f, \quad \mathbf{x} \in \Omega, \\ u &= g, \quad \mathbf{x} \in \Gamma, \end{aligned} \tag{5.18}$$

where $\mathbf{x} = (x, y) = (x_1, x_2)$ and

$$\Delta = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}.$$

Let Δ_h be the discrete five-point Laplacian:

$$\begin{aligned} \Delta_h u_{pq} &= (\delta_x^2 + \delta_y^2) u_{pq} \\ &:= \frac{u_{p-1,q} - 2u_{pq} + u_{p+1,q}}{h_x^2} + \frac{u_{p,q-1} - 2u_{pq} + u_{p,q+1}}{h_y^2}. \end{aligned} \tag{5.19}$$

Consistency: Let u_h be the FD solution of (5.18), i.e.,

$$\begin{aligned} -\Delta_h u_h &= f, \quad \mathbf{x} \in \Omega_h, \\ u_h &= g, \quad \mathbf{x} \in \Gamma_h, \end{aligned} \tag{5.20}$$

where Ω_h and Γ_h are the sets of grid points on Ω° and Γ, respectively. Note that the exact solution u of (5.18) satisfies

$$-\Delta_h u = f + \mathcal{O}(h^2 \partial^4 u), \quad \mathbf{x} \in \Omega_h. \tag{5.21}$$

Thus it follows from (5.20) and (5.21) that for some $C > 0$ independent of h,

$$\|\Delta_h (u - u_h)\|_{\infty,\Omega_h} \le C h^2 \|\partial^4 u\|_{\infty,\Omega_h}, \tag{5.22}$$

where $\|\cdot\|_{\infty,\Omega_h}$ denotes the maximum norm measured on the grid points Ω_h.

Note: We will soon use the inequality in (5.22) for convergence analysis.

Convergence: We are more interested in an error estimate for $(u - u_h)$ rather than for $\Delta_h(u - u_h)$. We begin with the following lemma.

Lemma 5.1. *Let Ω is a rectangular domain and v_h be a discrete function defined on a grid Ω_h of Ω with $v_h = 0$ on the boundary Γ_h. Then*

$$\|v_h\|_{\infty,\Omega_h} \le C\|\Delta_h v_h\|_{\infty,\Omega_h}, \tag{5.23}$$

for some $C > 0$ independent on h.

Proof. Let the function f_h be defined as

$$f_h := -\Delta_h v_h, \quad \mathbf{x} \in \Omega_h.$$

Then obviously

$$\begin{aligned} &\text{(a)} \quad \|f_h\|_{\infty,\Omega_h} = \|\Delta_h v_h\|_{\infty,\Omega_h}, \\ &\text{(b)} \quad -\|f_h\|_{\infty,\Omega_h} \le -\Delta_h v_h \le \|f_h\|_{\infty,\Omega_h}. \end{aligned} \tag{5.24}$$

Let $\widehat{\mathbf{x}} = (\hat{x}, \hat{y})$ be the centroid of Ω and consider

$$w_h(\mathbf{x}) = \frac{1}{4}\,|\mathbf{x} - \widehat{\mathbf{x}}|^2 = \frac{1}{4}\left((x - \hat{x})^2 + (y - \hat{y})^2\right), \quad \mathbf{x} \in \Omega_h.$$

Then w_h has its maximum on the boundary, bounded by a constant $C > 0$ independent on h, and

$$-\Delta_h w_h = -1, \quad \mathbf{x} \in \Omega_h.$$

So from (5.24.b) we have

$$-\Delta_h(v_h + \|f_h\|_{\infty,\Omega_h}\, w_h) = -\Delta_h v_h - \|f_h\|_{\infty,\Omega_h} \le 0$$

and therefore from the discrete maximum principle for subharmonic functions, Theorem 4.9 on page 101,

$$v_h + \|f_h\|_{\infty,\Omega_h}\, w_h \le \|f_h\|_{\infty,\Omega_h}\, \|w_h\|_{\infty,\Gamma_h} \le C\, \|f_h\|_{\infty,\Omega_h}.$$

Since $w_h \ge 0$,

$$v_h \le C\, \|f_h\|_{\infty,\Omega_h}. \tag{5.25}$$

The argument in the proof can be applied for the same conclusion, when v_h is replaced by $-v_h$. Thus, (5.23) follows from (5.24.a) and (5.25). □

Clearly, $(u - u_h)$ in (5.22) can be considered as a discrete function on the unit square with $u - u_h = 0$ on Γ_h. Therefore, with a aid of Lemma 5.1, one can conclude

Theorem 5.2. *Let u and u_h be the solutions of (5.18) and (5.20), respectively. Then*

$$\|u - u_h\|_{\infty,\Omega_h} \leq Ch^2 \|\partial^4 u\|_{\infty,\Omega_h}, \tag{5.26}$$

for some $C > 0$ independent on the grid size h.

Generalization

The above theorem, Theorem 5.2, can be expanded for more general elliptic problems of the form

$$\begin{aligned} Lu := -\nabla \cdot (A(\mathbf{x})\nabla u) + \mathbf{b}(\mathbf{x}) \cdot \nabla u = f, & \quad \mathbf{x} \in \Omega, \\ u = g, & \quad \mathbf{x} \in \Gamma, \end{aligned} \tag{5.27}$$

where $A(\mathbf{x}) = \text{diag}(a_{11}(\mathbf{x}), a_{22}(\mathbf{x}))$.

Let L_h be the five-point central discretization of L and u_h be the solution of

$$\begin{aligned} L_h u_h = f, & \quad \mathbf{x} \in \Omega_h, \\ u_h = g, & \quad \mathbf{x} \in \Gamma_h. \end{aligned} \tag{5.28}$$

Theorem 5.3. *Let u and u_h be the solutions of (5.27) and (5.28), respectively. Assume h is sufficiently small for the case $\mathbf{b} \neq 0$. Then*

$$\|u - u_h\|_{\infty,\Omega_h} \leq Ch^2, \tag{5.29}$$

for some $C = C(\Omega, \partial^3 u, \partial^4 u) > 0$ independent on the grid size h.

Proof. Note that

$$\begin{aligned} L_h u &= f + \mathcal{O}(h^2), \\ L_h u_h &= f, \end{aligned} \qquad \mathbf{x} \in \Omega_h.$$

Thus, we have

$$\|L_h(u - u_h)\|_{\infty,\Omega_h} \leq Ch^2, \tag{5.30}$$

for some $C > 0$ independent on h. Now, follow the same arguments utilized in Lemma 5.1, with Theorem 4.9 replaced by Theorem 4.10, to get

$$\|v_h\|_{\infty,\Omega_h} \leq C\|L_h v_h\|_{\infty,\Omega_h}, \tag{5.31}$$

for discrete functions v_h such that $v_h = 0$ on Γ_h. The inequality (5.29) follows from (5.30) and (5.31) with $v_h = u - u_h$. □

5.1.5. The Algebraic System for FDM

Let $\Omega = [a_x, b_x] \times [a_y, b_y]$ and $\Gamma = \partial\Omega$. Consider (5.18):

$$\begin{aligned} -\Delta u &= f, \quad \mathbf{x} \in \Omega, \\ u &= g, \quad \mathbf{x} \in \Gamma. \end{aligned} \tag{5.32}$$

Definition 5.4. Define, for some positive integers n_x, n_y,

$$h_x = \frac{b_x - a_x}{n_x}, \quad h_y = \frac{b_y - a_y}{n_y}$$

and

$$\begin{aligned} x_p &= a_x + p\,h_x, \quad p = 0, 1, \cdots, n_x \\ y_q &= a_y + q\,h_y, \quad q = 0, 1, \cdots, n_y \end{aligned}$$

Let Δ_h be the **discrete five-point Laplacian** (5.19):

$$\begin{aligned} \Delta_h u_{pq} &= (\delta_x^2 + \delta_y^2) u_{pq} \\ &:= \frac{u_{p-1,q} - 2u_{pq} + u_{p+1,q}}{h_x^2} + \frac{u_{p,q-1} - 2u_{pq} + u_{p,q+1}}{h_y^2}. \end{aligned} \tag{5.33}$$

Then, when the grid points are ordered row-wise, the algebraic system for the FDM reads

$$A\mathbf{u} = \mathbf{b}, \tag{5.34}$$

where

$$A = \begin{bmatrix} B & -I/h_y^2 & & & 0 \\ -I/h_y^2 & B & -I/h_y^2 & & \\ & \ddots & \ddots & \ddots & \\ & & -I/h_y^2 & B & -I/h_y^2 \\ 0 & & & -I/h_y^2 & B \end{bmatrix} \tag{5.35}$$

with I, the identity matrix of dimension $n_x - 1$; B with $d = \frac{2}{h_x^2} + \frac{2}{h_y^2}$ is given by

$$B = \begin{bmatrix} d & -1/h_x^2 & & & 0 \\ -1/h_x^2 & d & -1/h_x^2 & & \\ & \ddots & \ddots & \ddots & \\ & & -1/h_x^2 & d & -1/h_x^2 \\ 0 & & & -1/h_x^2 & d \end{bmatrix} \in \mathbb{R}^{(n_x-1)\times(n_x-1)}. \tag{5.36}$$

On the other hand,

$$\begin{aligned} b_{pq} &= f_{pq} + \frac{g_{p-1,q}}{h_x^2}\delta_{p-1,0} + \frac{g_{p+1,q}}{h_x^2}\delta_{p+1,n_x} \\ &\quad + \frac{g_{p,q-1}}{h_y^2}\delta_{q-1,0} + \frac{g_{p,q+1}}{h_y^2}\delta_{q+1,n_y} \end{aligned} \tag{5.37}$$

Note: The **global point index** for the row-wise ordering of the interior points, $i = 0, 1, 2, \cdots$, becomes

$$i = (q-1) * (n_x - 1) + p - 1 \tag{5.38}$$

Saving and managing the algebraic system

- For the FDM we just considered, the total number of interior nodal points is

  $$(n_x - 1) * (n_y - 1)$$

  Thus, you may try to open the matrix and other arrays based on this number.
- Saving nonzero entries only, the matrix A can be stored in an array of the form

  $$A[M][5] \text{ or } A[n_y - 1][n_x - 1][5], \tag{5.39}$$

  where $M = (n_x - 1) * (n_y - 1)$.
- However, it is often more convenient when the memory objects are opened incorporating all the nodal points (including those on boundaries). You may open the matrix as

  $$A[n_y + 1][n_x + 1][5]. \tag{5.40}$$

Continued on the next page

- The matrix A in (5.35) can be saved, in Python, as

```
rx, ry = 1/hx**2, 1/hy**2
d = 2*(rx+ry)
for q in range(1,ny):
    for p in range(1,nx):
        A[q][p][0] = -ry
        A[q][p][1] = -rx
        A[q][p][2] =  d
        A[q][p][3] = -rx
        A[q][p][4] = -ry
```

- Let the solution vector u be opened in `u[ny+1][nx+1]` and initialized along the boundaries. Then, the Gauss-Seidel iteration can be carried out as

```
import numpy as np; import copy
from numpy import abs,sqrt,pi,sin,cos

# the Jacobi matrix
T = copy.deepcopy(A) # np.ndarray((ny+1,nx+1,5),float)
for q in range(1,ny):
    for p in range(1,nx):
        for c in [0,1,3,4]:
            T[q][p][c] = -T[q][p][c]/T[q][p][2]

# A function for the Gauss-Seidel iteration
def Gauss_Seidel(T,u,itmax=1):
    ny,nx = leng(u)-1, len(u[0])-1
    for it in range(0,itmax):
        for q in range(1,ny):
            for p in range(1,nx):
                u[q][p] = T[q][p][0]*u[q-1][p] \
                         +T[q][p][1]*u[q][p-1] \
                         +T[q][p][3]*u[q][p+1] \
                         +T[q][p][4]*u[q+1][p]
```

5.2. Solution of Linear Algebraic Systems

In this section, we consider solution methods for the following linear system

$$A\mathbf{x} = \mathbf{b}, \tag{5.41}$$

where $A \in \mathbb{C}^{n\times n}$ and $\mathbf{b} \in \mathbb{C}^n$.

In most applications of PDEs, the matrix A is real-valued and sparse. By being sparse we mean that a large portion of entries in A is zero. For example, the maximum number of nonzero entries in a row is five for the central FD application to the Poisson equation in 2D.

5.2.1. Direct method: the LU factorization

Let the matrix

$$A = [a_{ij}]$$

be factorized into LU, where

$$L = [l_{ij}], \quad U = [u_{ij}]$$

are respectively lower and upper triangular matrices with $l_{ii} = 1$.

Then (5.41) reads

$$A\mathbf{x} = LU\mathbf{x} = \mathbf{b}, \tag{5.42}$$

which can be solved by

$$\begin{aligned} L\mathbf{y} &= \mathbf{b}, \\ U\mathbf{x} &= \mathbf{y}, \end{aligned} \tag{5.43}$$

by the **forward elimination** and **backward substitution**.

The **LU factorization** can be carried out by the **Gauss elimination** procedure. Define $A^{(1)} = [a_{ij}^{(1)}] = [a_{ij}]$ and

$$A^{(k)} = \begin{bmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \cdots & \cdots & \cdots & a_{1n}^{(k)} \\ & a_{22}^{(k)} & \cdots & \cdots & \cdots & a_{2n}^{(k)} \\ & & \ddots & \cdots & \cdots & \vdots \\ & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & 0 & & a_{k+1,k}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & & \vdots & \ddots & \vdots \\ & & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}. \tag{5.44}$$

Using the Gauss elimination procedure, $A^{(k+1)}$ and the entries of L can be determined as

$$\begin{aligned} a_{ij}^{(k+1)} &= \begin{cases} a_{ij}^{(k)} - \left(a_{ik}^{(k)} / a_{kk}^{(k)}\right) a_{kj}^{(k)}, & \text{for } i = k+1, \cdots, n, \;\; j = k, \cdots, n, \\ a_{ij}^{(k)}, & \text{else}, \end{cases} \\ l_{kk} &= 1, \\ l_{ik} &= a_{ik}^{(k)} / a_{kk}^{(k)}, \quad i = k+1, \cdots, n. \end{aligned} \tag{5.45}$$

Then, finally

$$U = A^{(n)} = [a_{ij}^{(n)}]. \tag{5.46}$$

The above procedure can be summarized into the following pseudocode:

$$\begin{array}{l} \textbf{For } k = 1 \textbf{ to } n-1 \\ \quad \left[\begin{array}{l} \textbf{For } i = k+1 \textbf{ to } n \\ \quad \left[\begin{array}{l} m_i \leftarrow a_{ik}/a_{kk}\,; \\ \textbf{if } m_i = 0, \textbf{ continue}\,; \\ a_{ik} \leftarrow m_i\,; \\ \textbf{For } j = k+1 \textbf{ to } n \\ \quad \left[a_{ij} \leftarrow a_{ij} - m_i\, a_{kj}\,; \right. \end{array} \right. \end{array} \right. \end{array} \tag{5.47}$$

In the output of the algorithm, the upper part including the main diagonal becomes U, while its strictly lower part is the corresponding part of L.

Algorithm (5.47) should be modified to incorporate the so-called bf partial pivoting when a pivot a_{kk} is expected to be zero or small in modulus.

The **LU factorization with partial pivoting** must look like the following:

```
For k = 1 to n − 1
    a_max ← 0 ;  i_max ← 0 ;                    /*find pivot*/
    For i = k to n
        if (|a_ik| > a_max)
            a_max ← |a_ik| ;  i_max ← i ;
    if (i_max = 0) stop ;                       /*A is singular*/
    if (i_max ≠ k)
        for j = 1 to n                          /*row interchange*/
            tmp ← a_kj ;
            a_kj ← a_{i_max,j} ;
            a_{i_max,j} ← tmp ;
        itmp ← intch[k] ;                       /*save interchange*/
        intch[k] ← intch[i_max] ;
        intch[i_max] ← itmp ;
    For i = k + 1 to n                          /*row operations*/
        m_i ← a_ik/a_kk ;
        if m_i = 0, continue ;
        a_ik ← m_i ;
        For j = k + 1 to n
            a_ij ← a_ij − m_i a_kj ;
```

(5.48)

Remark 5.5. In the above algorithm, the array "intch" must be initialized in advance `intch[i]=i`. You can use the array resulting from (5.48) to reorder the entries of the right-hand side b. That is,

$$\text{b[i]} \leftarrow \text{b[intch[i]]}, \quad i = 1, \cdots, n$$

Banded matrices: For a square matrix $A = [a_{ij}]$, if

$$a_{ij} = 0 \quad \text{for} \quad |i - j| > d, \quad \forall\, i, j,$$

the matrix is called to be *banded* with the **bandwidth** d.

- In most applications with the numerical solution of PDEs, the algebraic system is banded.
- For banded matrices, the LU factorization algorithms presented in (5.47) and (5.48) can be easily modified. For example, for the algorithm (5.47), simply replace the integers n appeared as the last indices of the i- and j-loops by $\min(n, k + d)$.

5.2.2. Linear iterative methods

Basic Concepts

- For solving linear algebraic systems, linear iterative methods begin with splitting the matrix A by

$$A = M - N, \tag{5.49}$$

  for some invertible matrix M.

- Then, the linear system equivalently reads

$$M\mathbf{x} = N\mathbf{x} + \mathbf{b}. \tag{5.50}$$

- Associated with the splitting is an iterative method

$$M\mathbf{x}^k = N\mathbf{x}^{k-1} + \mathbf{b}, \tag{5.51}$$

  or, equivalently,

$$\mathbf{x}^k = M^{-1}(N\mathbf{x}^{k-1} + \mathbf{b}) = \mathbf{x}^{k-1} + M^{-1}(\mathbf{b} - A\mathbf{x}^{k-1}), \tag{5.52}$$

  for an initial value $\mathbf{x}^0$.

Note:

- Methods differ for different choices of M.
- M must be easy to invert (efficiency) and $M^{-1} \approx A^{-1}$ (convergence).

5.2.3. Convergence theory

Let

$$\mathbf{e}^k = \mathbf{x} - \mathbf{x}^k;$$

from (5.50) and (5.51), we obtain the error equation

$$M\mathbf{e}^k = N\mathbf{e}^{k-1}$$

or, equivalently,

$$\mathbf{e}^k = M^{-1}N\mathbf{e}^{k-1}. \tag{5.53}$$

Thus

$$\begin{aligned} \|\mathbf{e}^k\| &\leq \|M^{-1}N\| \cdot \|\mathbf{e}^{k-1}\| \\ &\leq \|M^{-1}N\|^2 \cdot \|\mathbf{e}^{k-2}\| \\ &\vdots \\ &\leq \|M^{-1}N\|^k \cdot \|\mathbf{e}^0\|. \end{aligned} \tag{5.54}$$

Claim 5.6. A sufficient condition for the convergence is

$$\|M^{-1}N\| < 1. \tag{5.55}$$

Let $\sigma(B)$ be the spectrum, the set of eigenvalues of the matrix B, and $\rho(B)$ denote the spectral radius defined by

$$\rho(B) = \max_{\lambda_i \in \sigma(B)} |\lambda_i|.$$

Theorem 5.7. *The iteration converges if and only if*

$$\rho(M^{-1}N) < 1. \tag{5.56}$$

5.2.4. Graph theory for the estimation of the spectral radius

Definition 5.8. *A* **permutation matrix** *is a square matrix in which each row and each column has one entry of unity, all others zero.*

Definition 5.9. *For $n \geq 2$, an $n \times n$ complex-valued matrix A is* **reducible** *if there is a permutation matrix P such that*

$$PAP^T = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

where A_{11} and A_{22} are respectively $r \times r$ and $(n-r) \times (n-r)$ submatrices, $0 < r < n$. If no such permutation matrix exists, then A is **irreducible**.

The geometrical interpretation of the concept of the irreducibility by means of graph theory is useful.

Geometrical interpretation of irreducibility

- Given $A = (a_{ij}) \in \mathbb{C}^{n \times n}$, consider n distinct points

$$P_1, P_2, \cdots, P_n$$

in the plane, which we will call **nodes** or **nodal points**.

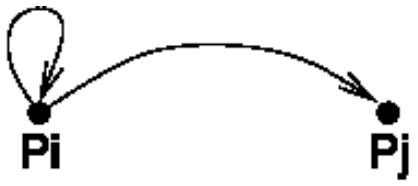


Figure 5.1: The directed paths for nonzero a_{ii} and a_{ij}.

- For any nonzero entry a_{ij} of A, we connect P_i to P_j by a path $\overrightarrow{P_iP_j}$, directed from the node P_i to the node P_j; a nonzero a_{ii} is joined to itself by a directed loop, as shown in Figure 5.1.

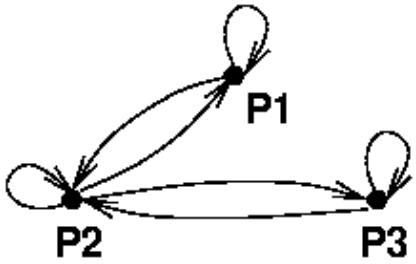


Figure 5.2: The directed graph $G(A)$ for A in (5.57).

- In this way, every $n \times n$ matrix A can be associated a **directed graph** $G(A)$. For example, the matrix

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \tag{5.57}$$

  has a directed graph shown in Figure 5.2.

Definition 5.10. A directed graph is **strongly connected** if, for any ordered pair of nodes (P_i, P_j), there is a directed path of a finite length

$$\overrightarrow{P_i P_{k_1}},\ \overrightarrow{P_{k_1} P_{k_2}},\ \cdots,\ \overrightarrow{P_{k_{r-1}} P_{k_r = j}},$$

connecting from P_i to P_j.

The theorems to be presented in this subsection can be found in [68] along with their proofs.

Theorem 5.11. *An $n \times n$ complex-valued matrix A is irreducible if and only if its directed graph $G(A)$ is strongly connected.*

It is obvious that the matrices obtained from FD/FE methods of the Poisson equation are strongly connected. Therefore the matrices are irreducible.

Eigenvalue Locus Theorem

For $A = [a_{ij}] \in \mathbb{C}^{n\times n}$, let

$$\Lambda_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|$$

Theorem 5.12. (Eigenvalue locus theorem) *Let $A = [a_{ij}]$ be an irreducible $n \times n$ complex matrix. Then,*

1. **(Gerschgorin [25])** *All eigenvalues of A lie in the union of the disks in the complex plane*

$$|z - a_{ii}| \leq \Lambda_i, \quad 1 \leq i \leq n. \tag{5.58}$$

2. **(Taussky [65])** *In addition, assume that λ, an eigenvalue of A, is a boundary point of the union of the disks $|z - a_{ii}| \leq \Lambda_i$. Then, all the n circles $|z - a_{ii}| = \Lambda_i$ must pass through the point λ, i.e., $|\lambda - a_{ii}| = \Lambda_i$ for all $1 \leq i \leq n$.*

For example, for

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

$\Lambda_1 = 1$, $\Lambda_2 = 2$, and $\Lambda_3 = 1$. Since $a_{ii} = 2$, for $i = 1, 2, 3$,

$$|\lambda - 2| < 2$$

for all eigenvalues λ of A.

Positiveness

Definition 5.13. An $n \times n$ complex-valued matrix $A = [a_{ij}]$ is **diagonally dominant** if

$$|a_{ii}| \geq \Lambda_i := \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \tag{5.59}$$

for all $1 \leq i \leq n$. An $n \times n$ matrix A is *irreducibly diagonally dominant* if A is irreducible and diagonally dominant, with strict inequality holding in (5.59) for at least one i.

Theorem 5.14. *Let A be an $n \times n$ strictly or irreducibly diagonally dominant complex-valued matrix. Then, A is nonsingular. If all the diagonal entries of A are in addition positive real, then the real parts of all eigenvalues of A are positive.*

Corollary 5.15. *A Hermitian matrix satisfying the conditions in Theorem 5.14 is positive definite.*

Corollary 5.16. *The FD/FE matrices from diffusion equations (including the Poisson equation) are positive definite, when it is symmetric.*

Regular splitting and M-matrices

Definition 5.17. For $n \times n$ real matrices, A, M, and N, $A = M - N$ is a **regular splitting** of A if M is nonsingular with $M^{-1} \geq 0$, and $N \geq 0$.

Theorem 5.18. *If $A = M - N$ is a regular splitting of A and $A^{-1} \geq 0$, then*

$$\rho(M^{-1}N) = \frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N)} < 1. \tag{5.60}$$

Thus, the matrix $M^{-1}N$ is convergent and the iterative method of (5.51) converges for any initial value $\mathbf{x}^0$.

Definition 5.19. *An $n \times n$ real matrix $A = [a_{ij}]$ with $a_{ij} \leq 0$ for all $i \neq j$ is an* **M-matrix** *if A is nonsingular and $A^{-1} \geq 0$.*

Theorem 5.20. *Let $A = (a_{ij})$ be an $n \times n$ M-matrix. If M is any $n \times n$ matrix obtained by setting certain off-diagonal entries of A to zero, then $A = M - N$ is a regular splitting of A and $\rho(M^{-1}N) < 1$.*

Theorem 5.21. *Let A be an $n \times n$ real matrix with $A^{-1} > 0$, and $A = M_1 - N_1 = M_2 - N_2$ be two regular splittings of A. If $N_2 \geq N_1 \geq 0$, where neither $N_2 - N_1$ nor N_1 is null, then*

$$1 > \rho(M_2^{-1}N_2) > \rho(M_1^{-1}N_1) > 0. \tag{5.61}$$

5.3. Relaxation methods

- We first express $A = (a_{ij})$ as the matrix sum

$$A = D - E - F, \tag{5.62}$$

where

$$\begin{aligned} &D = \mathbf{diag}(a_{11}, a_{22}, \cdots, a_{nn}), \\ &E = (e_{ij}), \quad e_{ij} = \begin{cases} -a_{ij}, & \text{if } i > j, \\ 0, & \text{else}, \end{cases} \\ &F = (f_{ij}), \quad f_{ij} = \begin{cases} -a_{ij}, & \text{if } i < j, \\ 0, & \text{else}. \end{cases} \end{aligned}$$

- Then, a relaxation method can be formulated by selecting M and N for a regular splitting:

$$A = M - N \tag{5.63}$$

- Popular examples are

Table 5.1: Relaxation methods

Methods	M	N
Jacobi method	D	$E + F$
Gauss-Seidel method	$D - E$	F
SOR method	$\frac{1}{\omega}D - E$	$\frac{1-\omega}{\omega}D + F$
Richardson method	I	$I - A$

SOR stands for **Successive Over Relaxation**.

5.3.1. Jacobi method

It is formulated as

$$D\mathbf{x}^k = (E+F)\mathbf{x}^{k-1} + \mathbf{b}, \tag{5.64}$$

which is the same as choosing

$$M = D, \quad N = E + F$$

The i-th component of (5.64) reads

$$a_{ii}\, x_i^k = -\sum_{j=1}^{i-1} a_{ij} x_j^{k-1} - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1} + b_i$$

or, equivalently,

$$x_i^k = \Big(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k-1} - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1}\Big) \Big/ a_{ii}, \tag{5.65}$$

for $i = 1, \cdots, n$.

5.3.2. Gauss-Seidel method

For the choice

$$M = D - E, \quad N = F,$$

we obtain the **Gauss-Seidel method**:

$$(D-E)\mathbf{x}^k = F\mathbf{x}^{k-1} + \mathbf{b}. \tag{5.66}$$

Its i-th component reads

$$\sum_{j=1}^{i} a_{ij} x_j^k = \sum_{j=i+1}^{n} -a_{ij} x_j^{k-1} + b_i,$$

which is equivalent to

$$x_i^k = \Big(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1}\Big) \Big/ a_{ii}, \quad i = 1, \cdots, n. \tag{5.67}$$

Note:

- The difference of the Gauss-Seidel method (5.67) out of the Jacobi method (5.65) is to utilize the updated values x_j^k, $j = 1, \cdots, i-1$.s
- It makes the method converge or diverge twice faster asymptotically.

5.3.3. Successive over-relaxation (SOR) method

Now, we consider the third basic linear iterative method for solving $A\mathbf{x} = \mathbf{b}$. Choose

$$M = \frac{1}{\omega}D - E, \quad N = \frac{1-\omega}{\omega}D + F, \quad \omega \in (0, 2),$$

where ω is called the **relaxation parameter** which is often set larger than one.

With the splitting, the SOR method can be formulated as

$$(D - \omega E)\mathbf{x}^k = \left[(1-\omega)D + \omega F\right]\mathbf{x}^{k-1} + \omega\mathbf{b}. \tag{5.68}$$

Since the above equation equivalently reads

$$D\mathbf{x}^k = (1-\omega)D\mathbf{x}^{k-1} + \omega\left(\mathbf{b} + E\mathbf{x}^k + F\mathbf{x}^{k-1}\right),$$

the i-th component of SOR becomes

$$\begin{aligned} x_{GS,i}^k &= \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^k - \sum_{j=i+1}^{n} a_{ij}x_j^{k-1}\right)\Big/ a_{ii}\,, \\ x_i^k &= (1-\omega)\, x_i^{k-1} + \omega\, x_{GS,i}^k. \end{aligned} \tag{5.69}$$

for $i = 1, \cdots, n$. Note that SOR turns out to be the Gauss-Seidel method when $\omega = 1$.

5.3.4. Convergence of relaxation methods

Let B, $\mathcal{L}_1$, and $\mathcal{L}_\omega$ be respectively the iteration matrices of the Jacobi, Gauss-Seidel, and SOR methods. That is,

$$B = D^{-1}(E+F), \quad \mathcal{L}_1 = (D-E)^{-1}F,$$
$$\mathcal{L}_\omega = (D-\omega E)^{-1}\big[(1-\omega)D + \omega F\big].$$

Theorem 5.22. (Stein and Rosenberg [62]) *On and only one of the following mutually exclusive relations is valid:*

$$\begin{aligned} &1.\ \ \rho(B) = \rho(\mathcal{L}_1) = 0, \\ &2.\ \ 0 < \rho(\mathcal{L}_1) < \rho(B) < 1, \\ &3.\ \ \rho(B) = \rho(\mathcal{L}_1) = 1, \\ &4.\ \ 1 < \rho(B) < \rho(\mathcal{L}_1). \end{aligned} \tag{5.70}$$

Thus the Jacobi and Gauss-Seidel methods are either both convergent or both divergent.

Theorem 5.23. (Ostrowski [55]) *Let $A = D - E - E^*$ be an $n \times n$ Hermitian matrix, where D is Hermitian and positive definite and $D - \omega E$ is nonsingular for $0 \le \omega \le 2$. Then,*

$$\rho(\mathcal{L}_\omega) < 1 \iff A \text{ is positive definite } \& \ 0 < \omega < 2. \tag{5.71}$$

Note that the matrices D and E in **Ostrowski's theorem** need not to be diagonal and strictly lower triangular matrices.

Optimal parameter for SOR: For algebraic systems of **good** properties, it is theoretically known that the convergence of SOR can be optimized when

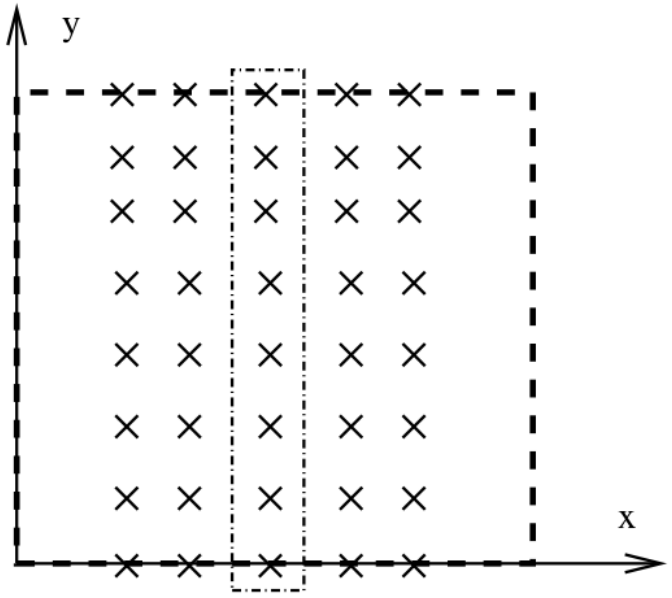
$$\omega = \frac{2}{1+\sqrt{1-\rho(B)}}, \tag{5.72}$$

where B is the Jacobi iteration matrix.

However, in most cases you can find a better parameter for a given algebraic system.

5.3.5. Line relaxation methods

- The standard Jacobi, Gauss-Seidel, and SOR schemes are called **point relaxation methods**.
- We can compute a whole line of new values using a direct method, e.g., Gauss elimination.
- this leads to **line relaxation methods**.



Algebraic interpretation: As in §5.1.5, consider

$$\begin{aligned} -\Delta u &= f, \quad \mathbf{x} \in \Omega, \\ u &= g, \quad \mathbf{x} \in \Gamma, \end{aligned} \tag{5.73}$$

where Ω is a rectangular domain in $\mathbb{R}^2$, and its discrete five-point Laplacian

$$\begin{aligned} \Delta_h u_{pq} &= (\delta_x^2 + \delta_y^2) u_{pq} \\ &:= \frac{u_{p-1,q} - 2u_{pq} + u_{p+1,q}}{h_x^2} + \frac{u_{p,q-1} - 2u_{pq} + u_{p,q+1}}{h_y^2}. \end{aligned} \tag{5.74}$$

Then, for the **column-wise point ordering**, the algebraic system for the FDM reads

$$A\mathbf{u} = \mathbf{b}, \tag{5.75}$$

where

$$A = \begin{bmatrix} C & -I/h_x^2 & & & 0 \\ -I/h_x^2 & C & -I/h_x^2 & & \\ & \ddots & \ddots & \ddots & \\ & & -I/h_x^2 & C & -I/h_x^2 \\ 0 & & & -I/h_x^2 & C \end{bmatrix} \tag{5.76}$$

with I being the identity matrix of dimension $n_y - 1$ and C being a matrix of order $n_x - 1$ given by

$$C = \begin{bmatrix} d & -1/h_y^2 & & & 0 \\ -1/h_y^2 & d & -1/h_y^2 & & \\ & \ddots & \ddots & \ddots & \\ & & -1/h_y^2 & d & -1/h_y^2 \\ 0 & & & -1/h_y^2 & d \end{bmatrix} \tag{5.77}$$

where $d = \dfrac{2}{h_x^2} + \dfrac{2}{h_y^2}$.

Note:

- A line relaxation method can be viewed as a (standard) relaxation method which deals with the matrix C like a single entry of a tridiagonal matrix.
- Once a point relaxation method converges, its line method converges **twice faster asymptotically**.
- Line methods can employ the line solver in alternating directions of (x, y).

Convergence Comparison

For (5.73) on p.137, we choose

$$\Omega = (0,1)^2, \quad n = n_x = n_y.$$

The following table includes the spectral radii of iteration matrices $\rho(T)$ and the required iteration counts k for the convergence to satisfy the tolerance $\|e^k\|/\|e^0\| < 10^{-6}$.

Table 5.2: Convergence comparison

	Point Jacobi		Line Jacobi		Point GS		Line GS	
n	$\rho(T)$	k	$\rho(T)$	k	$\rho(T)$	k	$\rho(T)$	k
5	0.8090	66	0.6793	36	0.6545	33	0.4614	18
10	0.9511	276	0.9067	142	0.9045	138	0.8221	71
20	0.9877	1116	0.9757	562	0.9755	558	0.9519	281
40	0.9969	4475	0.9939	2241	0.9938	2238	0.9877	1121

Remark 5.24. Final remarks for Relaxation Methods

- GS methods converge *asymptotically* twice faster than Jacobi methods, in either point or line iterations. SOR is yet faster and the line SOR is again twice faster.
- Relaxation methods sweep over either points or groups of points. For a faster convergence, you may let them visit the points in an order followed by the opposite order.
- For line methods, the tridiagonal matrix can be stored in a 3-column array, instead of a square big-fat array.

5.4. Krylov Subspace Methods

We consider Krylov subspace methods for solving

$$A\mathbf{x} = \mathbf{b}, \tag{5.78}$$

when A is **symmetric positive definite**.

Given an initial guess $\mathbf{x}_0 \in \mathbb{R}^n$, find successive approximations $\mathbf{x}_k \in \mathbb{R}^n$ of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \quad k = 0, 1, \cdots, \tag{5.79}$$

where $\mathbf{p}_k$ is the **search direction** and $\alpha_k > 0$ is the **step length**. Different methods differ in the choice of the search direction and the step length.

In this section, we consider the **gradient method** (also known as the steepest descent method, or the Richardson's method), the **conjugate gradient (CG) method**, and **preconditioned CG method**. For other Krylov subspace methods, see e.g. [3, 33].

Note that (5.78) admits a unique solution $x \in \mathbb{R}^n$, which is equivalently characterized by

$$\min_{\boldsymbol{\eta} \in \mathbb{R}^n} f(\boldsymbol{\eta}), \quad f(\boldsymbol{\eta}) = \frac{1}{2}\boldsymbol{\eta} \cdot A\boldsymbol{\eta} - \mathbf{b} \cdot \boldsymbol{\eta}, \tag{5.80}$$

where $\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$.

5.4.1. Steepest descent method

We denote the **gradient** and **Hessian** of f by f' and f'', respectively:

$$f'(\boldsymbol{\eta}) = A\boldsymbol{\eta} - \mathbf{b}, \qquad f''(\boldsymbol{\eta}) = A.$$

Given $\mathbf{x}_{k+1}$ as in (5.79), we have by Taylor's formula

$$\begin{aligned} f(\mathbf{x}_{k+1}) &= f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \\ &= f(\mathbf{x}_k) + \alpha_k f'(\mathbf{x}_k) \cdot \mathbf{p}_k + \frac{\alpha_k^2}{2} \mathbf{p}_k \cdot f''(\boldsymbol{\xi}) \mathbf{p}_k, \end{aligned}$$

for some $\boldsymbol{\xi}$.

Since the element of $f''(=A)$ is bounded,

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) + \alpha_k f'(\mathbf{x}_k)\cdot\mathbf{p}_k + \mathcal{O}(\alpha_k^2), \quad \text{as } \alpha_k \to 0.$$

The goal: to find $\mathbf{p}_k$ and α_k such that

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k),$$

which can be achieved if

$$f'(\mathbf{x}_k)\cdot\mathbf{p}_k < 0 \tag{5.81}$$

and α_k is sufficiently small.

Choice: (5.81) holds if we choose, when $f'(\mathbf{x}_k)\neq 0$,

$$\mathbf{p}_k = -f'(\mathbf{x}_k) = \mathbf{b} - A\mathbf{x}_k =: \mathbf{r}_k \tag{5.82}$$

Optimal Step Length

The following applies widely to most Krylov subspace methods.

Remark 5.25. **Optimal Step Length**: We may determine α_k such that

$$f(\mathbf{x}_k + \alpha_k\mathbf{p}_k) = \min_{\alpha} f(\mathbf{x}_k + \alpha\mathbf{p}_k),$$

in which case α_k is said to be **optimal**. If α_k is optimal, then

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}\alpha} f(\mathbf{x}_k + \alpha\mathbf{p}_k)\Big|_{\alpha=\alpha_k} = f'(\mathbf{x}_k + \alpha_k\mathbf{p}_k)\cdot\mathbf{p}_k \\ &= (A(\mathbf{x}_k + \alpha_k\mathbf{p}_k) - \mathbf{b})\cdot\mathbf{p}_k \\ &= (A\mathbf{x}_k - \mathbf{b})\cdot\mathbf{p}_k + \alpha_k\mathbf{p}_k\cdot A\mathbf{p}_k. \end{aligned}$$

So,

$$\alpha_k = \frac{\mathbf{r}_k\cdot\mathbf{p}_k}{\mathbf{p}_k\cdot A\mathbf{p}_k}. \tag{5.83}$$

Convergence of the steepest descent method

For the steepest descent method, the following is known

$$\| \mathbf{x} - \mathbf{x}_k \|_2 \leq \left(1 - \frac{1}{\kappa(A)} \right)^k \| \mathbf{x} - \mathbf{x}_0 \|_2. \tag{5.84}$$

Thus, the number of iterations required to reduce the error by a factor of ε is in the order of the condition number of A:

$$k \ \geq \ \kappa(A) \log \frac{1}{\varepsilon}. \tag{5.85}$$

Definition 5.26. The condition number of a matrix A is

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|, \tag{5.86}$$

for a matrix norm.

5.4.2. Conjugate gradient (CG) method

In this method the search directions $\mathbf{p}_k$ are conjugate, i.e.,

$$\mathbf{p}_i \cdot A\mathbf{p}_j = 0, \quad i \neq j, \tag{5.87}$$

and the step length α_k is chosen to be optimal. The following is the original version of the CG method.

Algorithm 5.27. **(CG Algorithm, V.1)**

$$\begin{array}{l}
\text{Select } \mathbf{x}_0,\ \varepsilon; \\
\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0,\ \ \mathbf{p}_0 = \mathbf{r}_0; \\
\text{Do } k = 0, 1, \cdots \\
\qquad \alpha_k = \mathbf{r}_k \cdot \mathbf{p}_k / \mathbf{p}_k \cdot A\mathbf{p}_k; \qquad \text{(CG1)} \\
\qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k; \qquad \text{(CG2)} \\
\qquad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k; \qquad \text{(CG3)} \\
\qquad \text{if } \| \mathbf{r}_{k+1} \|_2 < \varepsilon \| \mathbf{r}_0 \|_2, \ \text{stop}; \\
\qquad \beta_k = -\mathbf{r}_{k+1} \cdot A\mathbf{p}_k / \mathbf{p}_k \cdot A\mathbf{p}_k; \qquad \text{(CG4)} \\
\qquad \mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k; \qquad \text{(CG5)} \\
\text{End Do}
\end{array} \tag{5.88}$$

Remark 5.28.

- α_k in (CG1) is designed such that $\mathbf{r}_{k+1} \cdot \mathbf{p}_k = 0$.
  You may easily verify it using $\mathbf{r}_{k+1}$ in (CG3).
- $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$, by definition. So,

$$\begin{aligned}
\mathbf{r}_{k+1} &= \mathbf{b} - A\mathbf{x}_{k+1} = \mathbf{b} - A(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \\
&= \mathbf{b} - A\mathbf{x}_k - \alpha_k A\mathbf{p}_k = \mathbf{r}_k - \alpha_k A\mathbf{p}_k,
\end{aligned}$$

  which is (CG3).
- β_k in (CG4) is determined such that $\mathbf{p}_{k+1} \cdot A\mathbf{p}_k = 0$.
  Verify it using $\mathbf{p}_{k+1}$ in (CG5).
- The CG method finds the iterate

$$\mathbf{x}_k \in \mathbf{x}_0 + \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \cdots, A^{k-1}\mathbf{r}_0\}$$

  so that $(\mathbf{x} - \mathbf{x}_k) \cdot A(\mathbf{x} - \mathbf{x}_k)$ is minimized.

Theorem 5.29. *For* $m = 0, 1, \cdots,$

$$\begin{aligned} \text{span}\{\mathbf{p}_0, \cdots, \mathbf{p}_m\} &= \text{span}\{\mathbf{r}_0, \cdots, \mathbf{r}_m\} \\ &= \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \cdots, A^m\mathbf{r}_0\}. \end{aligned} \tag{5.89}$$

Theorem 5.30. *The search directions and the residuals satisfy the orthogonality,*

$$\mathbf{p}_i \cdot A\mathbf{p}_j = 0; \quad \mathbf{r}_i \cdot \mathbf{r}_j = 0, \quad i \neq j. \tag{5.90}$$

Theorem 5.31. *For some* $m \leq n$*, we have* $A\mathbf{x}_m = b$ *and*

$$\| \mathbf{x} - \mathbf{x}_k \|_A \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k \| \mathbf{x} - \mathbf{x}_0 \|_A. \tag{5.91}$$

So the required iteration number to reduce the error by a factor of ε *is*

$$k \geq \frac{1}{2}\sqrt{\kappa(A)}\, \log \frac{2}{\varepsilon}. \tag{5.92}$$

Proofs of the above theorems can be found in e.g. [32].

Simplification of the CG method

Using the properties and identities involved in the method, one can derive a more popular form of the CG method.

Algorithm 5.32. (CG Algorithm, V.2)

$$
\begin{aligned}
&\text{Select } \mathbf{x}_0,\ \varepsilon;\\
&\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0,\ \ \mathbf{p}_0 = \mathbf{r}_0;\\
&\text{Compute } \rho_0 = \mathbf{r}_0 \cdot \mathbf{r}_0;\\
&\text{Do } k = 0, 1, \cdots\\
&\qquad \alpha_k = \rho_k / \mathbf{p}_k \cdot A\mathbf{p}_k;\\
&\qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k;\\
&\qquad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k;\\
&\qquad \text{if } \| \mathbf{r}_{k+1} \|_2 < \varepsilon \| \mathbf{r}_0 \|_2,\ \text{stop};\\
&\qquad \rho_{k+1} = \mathbf{r}_{k+1} \cdot \mathbf{r}_{k+1};\\
&\qquad \beta_k = \rho_{k+1} / \rho_k;\\
&\qquad \mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k;\\
&\text{End Do}
\end{aligned}
\tag{5.93}
$$

Note:

$$
\begin{aligned}
\mathbf{r}_k \cdot \mathbf{p}_k &= \mathbf{r}_k \cdot (\mathbf{r}_k + \beta_{k-1}\mathbf{p}_{k-1}) = \mathbf{r}_k \cdot \mathbf{r}_k,\\
\beta_k &= -\mathbf{r}_{k+1} \cdot A\mathbf{p}_k / \mathbf{p}_k \cdot A\mathbf{p}_k = -\mathbf{r}_{k+1} \cdot A\mathbf{p}_k \frac{\alpha_k}{\rho_k}\\
&= \mathbf{r}_{k+1} \cdot (\mathbf{r}_{k+1} - \mathbf{r}_k)/\rho_k = \rho_{k+1}/\rho_k.
\end{aligned}
$$

5.4.3. Preconditioned CG method

The condition number of A is the critical point for the convergence of the CG method. If we can find a matrix M such that

$$M \approx A$$

and it is easy to invert, we may try to apply the CG algorithm to the following system

$$M^{-1}Ax = M^{-1}\mathbf{b}. \tag{5.94}$$

Since

$$\kappa(M^{-1}A) \ll \kappa(A) \tag{5.95}$$

(hopefully, $\kappa(M^{-1}A) \approx 1$), the CG algorithm will converge much faster.

In practice, we do not have to multiply M^{-1} to the original algebraic system and the algorithm can be implemented as

Algorithm 5.33. **(Preconditioned CG)**

$$\begin{aligned}
&\text{Select } \mathbf{x}_0,\ \varepsilon;\\
&\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0,\ \ M\mathbf{z}_0 = \mathbf{r}_0;\\
&\mathbf{p}_0 = \mathbf{z}_0,\ \text{compute } \rho_0 = \mathbf{z}_0^*\mathbf{r}_0;\\
&\text{Do } k = 0, 1, \cdots\\
&\qquad \alpha_k = \rho_k / \mathbf{p}_k^* A\mathbf{p}_k;\\
&\qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k;\\
&\qquad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k;\\
&\qquad \text{if } \| \mathbf{r}_{k+1} \|_2 < \varepsilon \| \mathbf{r}_0 \|_2,\ \text{stop};\\
&\qquad M\mathbf{z}_{k+1} = \mathbf{r}_{k+1};\\
&\qquad \rho_{k+1} = \mathbf{z}_{k+1}^* \mathbf{r}_{k+1};\\
&\qquad \beta_k = \rho_{k+1} / \rho_k;\\
&\qquad \mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k;\\
&\text{End Do}
\end{aligned} \tag{5.96}$$

Here the superscript * indicates the transpose complex-conjugate; it is the transpose for real-valued systems.

5.5. Other Iterative Methods

5.5.1. Incomplete LU-factorization

Here, we introduce Stone's **strongly implicit procedure** (**SIP**) [63] to solve the following linear system

$$A\mathbf{x} = \mathbf{b}. \tag{5.97}$$

- As for other iterative methods, SIP is based on a **regular splitting**, $A = M - N$, with M being an **incomplete** LU **factorization** (**ILU**);

$$M = L_I U_I = A + N, \tag{5.98}$$

  where L_I and U_I are respectively the lower and upper triangular components of the ILU factorization of A, where the entries of the main diagonal of U_I are all one.
- The iteration corresponding to the splitting (5.98) is formulated as

$$L_I U_I \mathbf{x}^k = N\mathbf{x}^{k-1} + \mathbf{b}, \tag{5.99}$$

  or, since $N = L_I U_I - A$,

$$\begin{aligned} &\text{(a)} \quad \mathbf{r}^{k-1} = \mathbf{b} - A\mathbf{x}^{k-1}, \\ &\text{(b)} \quad L_I U_I \boldsymbol{\delta}^k = \mathbf{r}^{k-1}, \\ &\text{(c)} \quad \mathbf{x}^k = \mathbf{x}^{k-1} + \boldsymbol{\delta}^k. \end{aligned} \tag{5.100}$$

- The iteration (5.99) converges fast, when we choose elements of L_I and U_I in a way that N *is as small as possible*.

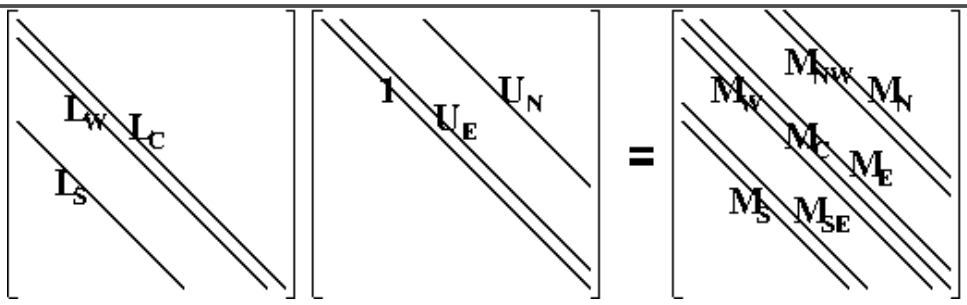


Figure 5.3: Systematic presentation of $L_I U_I = M$. The subscripts S, W, E, N, and C denote respectively south, west, east, north, and center. Note that diagonals of M marked by subscripts SE and NW are not found in A.

Derivation of SIP

- For a 2D problem in a rectangular mesh where the grid points are ordered in the row-wise manner, the ILU factorization is in the form as in Figure 5.3 and the row of M corresponding to the (ℓ, m)-th grid point is given as

$$\begin{aligned} M_S^{\ell,m} &= L_S^{\ell,m}, \\ M_{SE}^{\ell,m} &= L_S^{\ell,m} U_E^{\ell,m-1}, \\ M_W^{\ell,m} &= L_W^{\ell,m}, \\ M_C^{\ell,m} &= L_S^{\ell,m} U_N^{\ell,m-1} + L_W^{\ell,m} U_E^{\ell-1,m} + L_C^{\ell,m}, \\ M_E^{\ell,m} &= L_C^{\ell,m} U_E^{\ell,m}, \\ M_{NW}^{\ell,m} &= L_W^{\ell,m} U_N^{\ell-1,m}, \\ M_N^{\ell,m} &= L_C^{\ell,m} U_N^{\ell,m}. \end{aligned} \tag{5.101}$$

- The (ℓ, m)-th component of $N\mathbf{x}$ is

$$\begin{aligned} (N\mathbf{x})_{\ell,m} &= N_C^{\ell,m} x_{\ell,m} + N_S^{\ell,m} x_{\ell,m-1} + N_W^{\ell,m} x_{\ell-1,m} + N_E^{\ell,m} x_{\ell+1,m} \\ &\quad + N_N^{\ell,m} x_{\ell,m+1} + M_{SE}^{\ell,m} x_{\ell+1,m-1} + M_{NW}^{\ell,m} x_{\ell-1,m+1}. \end{aligned} \tag{5.102}$$

- Due to occasional instabilities, we introduce the **approximations**

$$\begin{aligned} x_{\ell+1,m-1} &\approx \alpha(x_{\ell,m-1} + x_{\ell+1,m} - x_{\ell,m}), \\ x_{\ell-1,m+1} &\approx \alpha(x_{\ell,m+1} + x_{\ell-1,m} - x_{\ell,m}), \end{aligned} \quad 0 < \alpha \le 1, \tag{5.103}$$

  and rewrite (5.102) as

$$\begin{aligned} (N\mathbf{x})_{\ell,m} &\approx (N_C^{\ell,m} - \alpha M_{SE}^{\ell,m} - \alpha M_{NW}^{\ell,m}) x_{\ell,m} \\ &\quad + (N_S^{\ell,m} + \alpha M_{SE}^{\ell,m}) x_{\ell,m-1} + (N_W^{\ell,m} + \alpha M_{NW}^{\ell,m}) x_{\ell-1,m} \\ &\quad + (N_E^{\ell,m} + \alpha M_{SE}^{\ell,m}) x_{\ell+1,m} + (N_N^{\ell,m} + \alpha M_{NW}^{\ell,m}) x_{\ell,m+1}. \end{aligned} \tag{5.104}$$

- **Set each of coefficients in the right-side of (5.104) to be zero.** Then, it follows from (5.101) that entries of N are presented by those of L_I and U_I:

$$\begin{aligned} N_S^{\ell,m} &= -\alpha M_{SE}^{\ell,m} = -\alpha L_S^{\ell,m} U_E^{\ell,m-1}, \\ N_W^{\ell,m} &= -\alpha M_{NW}^{\ell,m} = -\alpha L_W^{\ell,m} U_N^{\ell-1,m}, \\ N_C^{\ell,m} &= \alpha(M_{SE}^{\ell,m} + M_{NW}^{\ell,m}) = \alpha(L_S^{\ell,m} U_E^{\ell,m-1} + L_W^{\ell,m} U_N^{\ell-1,m}), \\ N_E^{\ell,m} &= -\alpha M_{SE}^{\ell,m} = -\alpha L_S^{\ell,m} U_E^{\ell,m-1}, \\ N_N^{\ell,m} &= -\alpha M_{NW}^{\ell,m} = -\alpha L_W^{\ell,m} U_N^{\ell-1,m}. \end{aligned} \tag{5.105}$$

- Now, utilizing $M = A+N$, (5.101), and (5.105), one can obtain Stone's SIP [63]:

$$\begin{aligned} L_S^{\ell,m} &= A_S^{\ell,m}/(1+\alpha U_E^{\ell,m-1}), \\ L_W^{\ell,m} &= A_W^{\ell,m}/(1+\alpha U_N^{\ell-1,m}), \\ L_C^{\ell,m} &= A_C^{\ell,m} + \alpha(L_S^{\ell,m}U_E^{\ell,m-1} + L_W^{\ell,m}U_N^{\ell-1,m}) \\ &\quad -L_S^{\ell,m}U_N^{\ell,m-1} - L_W^{\ell,m}U_E^{\ell-1,m}, \\ U_E^{\ell,m} &= (A_E^{\ell,m} - \alpha L_S^{\ell,m}U_E^{\ell,m-1})/L_C^{\ell,m}, \\ U_N^{\ell,m} &= (A_N^{\ell,m} - \alpha L_W^{\ell,m}U_N^{\ell-1,m})/L_C^{\ell,m}. \end{aligned} \tag{5.106}$$

Remark 5.34.

- The approximations in (5.103) are second-order accurate when $\alpha = 1$.
- But the algorithm (5.106) can be unstable for some problems; the parameter α is often chosen between 0.92 and 0.96 [23].
- Entries of L_I and U_I used in (5.106) whose indices are outside the index boundaries should be set zero.

5.6. Numerical Examples with Python

Here we demonstrate a Python code for solving

$$\begin{aligned} -\Delta u &= f, \quad \mathbf{x} \in \Omega = (0,1)^2 \\ u &= g, \qquad \mathbf{x} \in \partial\Omega \end{aligned} \tag{5.107}$$

The exact solution is chosen as

$$u(x,y) = \sin(\pi x)\sin(\pi y) \tag{5.108}$$

so that the right-hand side becomes

$$f(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y)$$

With the number of grid points $n = n_x = n_y$, the maximum errors are as follows

Table 5.3: The maximum error $\|u - u_h\|_\infty$.

n	10	20	40	80
$\|u - u_h\|_\infty$	0.00827	0.00206	0.00050	6.42e-05

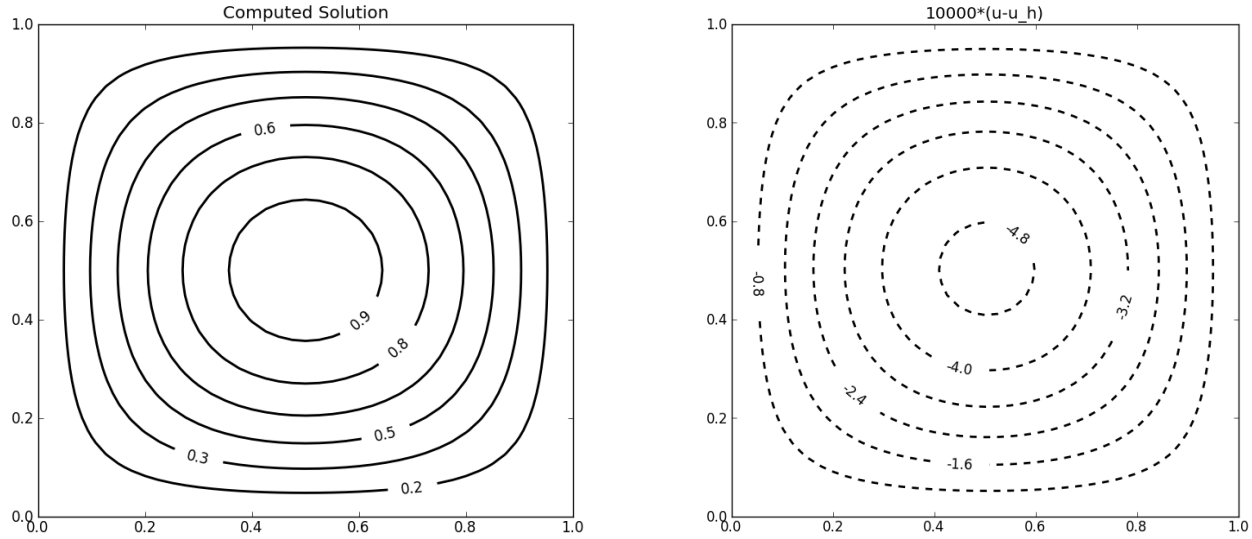


Figure 5.4: Contour plots of computed solution with $n = 40$ (left) and the 10000-times magnified error (right)

The whole code is attached below.

```
#=======================================================
# Elliptic_2D.py
# This module solves, by the 2nd-order FD method & SOR
#     -(u_xx+u_yy)=f, (x,y) in (ax,bx)x(ay,by)
#         u=g,        (x,y) on its boundary
# Supporting functions are built in "util_ellip2D.py"
#=======================================================
from util_ellip2D import *

##----------------------
## User Input
##----------------------
ax,bx = 0., 1.
ay,by = 0., 1.
nx= 40; ny=nx

itmax = 1000
tol   = 1.e-6
omega = 1.8

level = 2
##----------------------
## End of "User Input"
##----------------------

print 'Elliptic_2D: (ax,bx)x(ay,by)=(%g,%g)x(%g,%g),\
 (nx,ny)=(%d,%d)' % (ax,bx,ay,by, nx,ny)

## build up coefficient matrix & others
A = coeff_matrix(ax,bx,ay,by,nx,ny,level)
b = get_rhs(ax,bx,ay,by,nx,ny,level)
U = get_exact_sol(ax,bx,ay,by,nx,ny,level)
X = init_X(U)

## solve with SOR
sol_SOR(A,X,b,omega,tol,itmax,level)

## Checking error
if level:
    print "  Max-error=%g" % (error8(U,X,level))

## Want to see the figure?
if level>=3:
    contourplot(U,ax,bx,ay,by,'Exact Solution',2)
    contourplot(X,ax,bx,ay,by,'Computed Solution',2)
```

```
##=================================================
## util_ellip2D.py
##=================================================
import numpy as np
from numpy import abs,sqrt,pi,sin,cos
import matplotlib.pyplot as plt
from matplotlib.mlab import griddata
from copy import deepcopy

def coeff_matrix(ax,bx,ay,by,nx,ny,level=0):
    matA = np.ndarray((ny+1,nx+1,5),float)
    hx,hy= (bx-ax)/nx, (by-ay)/ny
    for p in range(0,nx+1):
        matA[0][p]=[0,0,1,0,0]; matA[ny][p]=[0,0,1,0,0]
    for q in range(0,ny+1):
        matA[q][0]=[0,0,1,0,0]; matA[q][nx]=[0,0,1,0,0]
    rx,ry = 1./hx**2, 1./hy**2
    d     = 2*(rx+ry)
    for q in range(1,ny):
        for p in range(1,nx):
            matA[q][p][0] = -ry
            matA[q][p][1] = -rx
            matA[q][p][2] = d
            matA[q][p][3] = -rx
            matA[q][p][4] = -ry
    return matA

def get_rhs(ax,bx,ay,by,nx,ny,level=0):
    vec_b = np.ndarray((ny+1,nx+1),float)
    hx,hy = (bx-ax)/nx, (by-ay)/ny
    for q in range(0,ny+1):
        y = ay+q*hy
        for p in range(0,nx+1):
            x = ax+p*hx
            vec_b[q][p] = funct_f(x,y)
    return vec_b

def get_exact_sol(ax,bx,ay,by,nx,ny,level=0):
    vec_u = np.ndarray((ny+1,nx+1),float)
    hx,hy = (bx-ax)/nx, (by-ay)/ny
    for q in range(0,ny+1):
        y = ay+q*hy
        for p in range(0,nx+1):
            x = ax+p*hx
```

```
            vec_u[q][p] = funct_u(x,y)
    return vec_u

def funct_f(x,y):
    return 2*pi**2*sin(pi*x)*sin(pi*y)

def funct_u(x,y):
    return sin(pi*x)*sin(pi*y)

def contourplot(XX,ax,bx,ay,by,title,level=0):
    ny,nx = len(XX),len(XX[0])
    xi = np.linspace(ax,bx,nx)
    yi = np.linspace(ay,by,ny)
    X,Y= np.meshgrid(xi, yi)
    Z = griddata(X.ravel(),Y.ravel(),XX.ravel(),xi,yi)
    CS = plt.contour(X, Y, Z, linewidths=2,colors='k')
    plt.clabel(CS, inline=2, fmt='%1.1f', fontsize=12)
    plt.title(title)
    plt.show()

def init_X(U,level=0):
    X = deepcopy(U)
    ny,nx = len(U),len(U[0])
    for q in range(1,ny-1):
        for p in range(1,nx-1):
            X[q][p] = 0.
    return X

def sol_SOR(A,X,b,omega,tol,itmax,level=0):
    ny,nx = len(X),len(X[0])
    for it in range(0,itmax):
        err=0.
        for j in range(1,ny-1):
          for i in range(1,nx-1):
            gs =( b[j][i]-(A[j][i][0]*X[j-1][i]\
                          +A[j][i][1]*X[j][i-1]\
                          +A[j][i][3]*X[j][i+1]\
                          +A[j][i][4]*X[j+1][i]) )\
                / A[j][i][2]
            xnew = (1.-omega)*X[j][i]+omega*gs
            err = max(err, abs(X[j][i]-xnew))
            X[j][i] = xnew
        if err<tol:
            if level>=1:
                print "sol_SOR: converged it= %d" %(it+1)
```

```
            break

def error8(X,Y,level=0):
    ny,nx = len(X),len(X[0])
    err8=0.
    for q in range(0,ny):
        for p in range(0,nx):
            err8=max(err8,abs(X[q][p]-Y[q][p]))
    return err8
```

5.7. Homework

1. Verify that the overall truncation error for the FD scheme (5.14) is second-order in h_x. *Hint*: Define
$$K(x) = a(x)\frac{u_{xxx}(x)}{3!}\left(\frac{h_x}{2}\right)^2 + \cdots,$$
for the truncation errors appeared in (5.13). Then the truncation error for the approximation of $(au_x)_{i+1/2} - (au_x)_{i-1/2}$ becomes $K(x_{i+1/2}) - K(x_{i-1/2}) = h_x K'(x_i) + \cdots$.

2. Implement a code to solve
$$\begin{cases} -(uu_x)_x = 0, & x \in (0,2), \\ u(0) = g_L, \quad u(2) = g_R, \end{cases} \tag{5.109}$$
utilizing the second-order FD scheme (5.14) on a uniform grid. At the grid point x_i, your approximation will read
$$\frac{-u_{i-1}^2 + 2u_i^2 - u_{i+1}^2}{h_x^2} = 0. \tag{5.110}$$
For the solver, you may use the simplest method (the Jacobi!) and its variant. For the number of grid points, you may choose a convenient number, e.g., $n_x = 20$.

   (a) Derive (5.110).
   (b) Solve to plot the FD solution for $g_L = 0$ and $g_R = 2$.
   (The exact solution $u = \sqrt{2x}$ and you may assume that the numerical solution is nonnegative.)
   (c) Solve to plot the FD solution for $g_L = -1$ and $g_R = 1$.
   $\left(\text{The exact solution } u = \begin{cases} \sqrt{x-1}, & x \geq 1, \\ -\sqrt{1-x}, & x < 1. \end{cases}\right)$ The FD equation (5.110) reads $u_i = \pm\sqrt{(u_{i-1}^2 + u_{i+1}^2)/2}$. You have to modify the iterative algorithm to choose the right one. This step will be so hard, but I believe it is fun to conquer.
   (d) *(Optional)* Do you have any idea overcoming the difficulty involved in (5.2c)?

3. For the 3D Poisson equation
$$\begin{aligned} -(u_{xx} + u_{yy} + u_{zz}) &= f, \quad \mathbf{x} = (x,y,z) \in \Omega = (0,1)^3, \\ u &= 0, \qquad\quad \mathbf{x} = (x,y,z) \in \partial\Omega \end{aligned} \tag{5.111}$$

   (a) Apply the central second-order FD method, with a uniform grid size $h = h_x = h_y = h_z$, to get difference equations.
   (b) Show that the maximum principle still applies.
   (c) Prove that
   $$\|u - u_h\|_\infty \leq \frac{h^2}{24} \max_{\mathbf{x}\in\Omega}(|u_{xxxx}| + |u_{yyyy}| + |u_{zzzz}|), \tag{5.112}$$
   where u_h is the finite difference solution.

4. Consider the eigenvalue problem

$$\begin{aligned} -\Delta u &= \lambda u, \quad (x,y) \in \Omega = (0,1)^2, \\ u &= 0, \qquad (x,y) \in \partial\Omega, \end{aligned} \tag{5.113}$$

where the eigenfunction $u(x,y) \neq 0$. Prove that the eigenvalues and the corresponding eigenfunctions are

$$\begin{aligned} \lambda_{mn} &= (m^2+n^2)\pi^2, \\ u_{mn}(x,y) &= \sin(m\pi x)\sin(n\pi y), \end{aligned} \tag{5.114}$$

for $m, n = 1, 2, \cdots$. (*Hint*: Set $u(x,y) = X(x)Y(y)$ to plug it in (5.113).)

5. Modify the Python code in §5.6 to add a line SOR method, for the line either in the x-direction or in the y-direction. Provide a convergence analysis comparing convergence speeds between the point SOR and the line SOR.

6. Edit once more the Python code you just modified for Homework 5.5 to solve more general elliptic problem of the form

$$\begin{aligned} -[d_1(x,y)u_x]_x - [d_2(x,y)u_y]_y + r(x,y)u &= f, \quad \mathbf{x} \in \Omega = (0,1)^2 \\ u &= g, \qquad \mathbf{x} \in \partial\Omega. \end{aligned} \tag{5.115}$$

   (a) Choose f and g accordingly such that the exact solution

$$u(x,y) = (1-x^2)(y^3-y) \tag{5.116}$$

   and the coefficients

$$d_1(x,y) = 2 + x^2 - y^2, \quad d_2(x,y) = e^{xy}, \quad r(x,y) = x + 2y.$$

   (b) Estimate the convergence rate by running different mesh sizes, for example, $n = 10, 20, 40, 80$.

   (c) Visualize computed solutions with 3D mesh/surface plots in Python.

7. **(Optional)** Let $A = (a_{ij})$ be a nonsingular square matrix, obtained from a FD/FE approximation of an elliptic problem of the form

$$\begin{aligned} -\nabla\cdot(a(\mathbf{x})\nabla u) + \mathbf{b}(\mathbf{x})\cdot\nabla u + c(\mathbf{x})u &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ \alpha(\mathbf{x})u_\nu + \beta(\mathbf{x})u &= g(\mathbf{x}), \qquad \mathbf{x} \in \Gamma, \end{aligned} \tag{5.117}$$

where $a > 0$, $c \geq 0$, $\alpha \geq 0$, and Ω is a bounded domain in $\mathbb{R}^d$, $1 \leq d \leq 3$, with its boundary $\Gamma = \partial\Omega$. Assume that

   (i) The elements in the main diagonal of A are positive and the other elements are nonpositive, i.e., for each i,

$$a_{ii} > 0; \quad a_{ij} \leq 0, \ \ i \neq j.$$

(ii) A is *diagonally dominant*, i.e., for each i,

$$a_{ii} \geq \sum_{j \neq i} |a_{ij}|,$$

and at least one of the inequalities is strict.

(iii) The directed graph of A is *strongly connected*. (The standard FD/FE methods always satisfy this condition.)

(a) Prove the following *generalized maximum principle*:

Theorem 5.35. (Maximum Principle) *Suppose that A satisfies all the above assumptions and that*

$$A\mathbf{u} \leq 0 \ \ (A\mathbf{u} \geq 0).$$

Then, the solution $\mathbf{u}$ *has its maximum* (*minimum*) *on the boundary.*

(b) Let $\Omega = (0,1)^3$ and consider the 7-point FD method for the problem in (5.117). Find conditions on the coefficients and the mesh size h with which the numerical solution of (5.117) satisfies the maximum principle.

The 15% Rule: Turning Curiosity into Financial Freedom

The 15% Rule

- Use 85% of your effort to become excellent at what you do.
- Use 15% to discover what else you could become.

Invest 15% of your discretionary/optional time in something outside your field: One day a week–or one or two hours a day:

- Put your phone away.
- Stop scrolling.
- Pick a field you know little about.
- Study it as if you were going to become the CEO of a company in that field.

Not for a grade. Not for your résumé.
Study it because someday it may become an opportunity.

Why Think Like a CEO?

Don't just ask:

How does renewable energy work?

Ask:

Where is the money made?

What problems remain unsolved?

Who are the customers?

What technologies are changing the industry?

Where could mathematics or computing create value?

If I were the CEO, what would I do?

That changes learning from

$$\begin{array}{c} \text{Information} \longrightarrow \text{Knowledge} \\ \downarrow \\ \text{Information} \longrightarrow \text{Knowledge} \longrightarrow \text{Opportunity} \end{array}$$

Three hundred years ago, **land** was wealth.
Later, the **industrialists** came to possess wealth.
Today, wealth lies in **information**.

Your Learning Speed Is Trainable

Let T_n be the time required to understand the n-th field. Then

$$T_1 \approx \textbf{1–2 years}$$

- However, as you keep learning new fields, you start to see **common structures**:

  Technology → Market → Cost → Risk → Competition → Opportunity

- So, conceptually: $T_{k+1} < T_k$
- With practice, you will eventually reach a point where you can grasp the essentials of a new subject **in just one or two months**:

$$T_n \approx \textbf{1–2 months, for a large } n$$

I call it the **learning compound interest**. It's not just money that compounds; the ability to learn compounds as well.

Here are some **topic examples** for the 15% Rule:

- Renewable Energy, AI for Healthcare
- Robotics, Cybersecurity
- Agriculture Technology, Stock Market

Every business or industrial R&D project focuses on **optimizing five key variables: scale, shape, direction, interface, and flow.**

Beyond Equations: Building True

Core Lessons from Robert Kiyosaki's ***Rich Dad Poor Dad***

- Your career earns money. Your financial system buys freedom.
- Don't measure wealth by salary. Measure it by freedom.

High Income $\neq$ Financial Freedom

The Poor Dad Mindset: The Lifestyle Trap

Imagine your salary grows:

$\$60K \rightarrow \$80K \rightarrow \$100K \rightarrow \$130K$

But your spending grows with it:

$\$50K \rightarrow \$70K \rightarrow \ \$90K \rightarrow \$120K.$

Your career improved enormously.
Your freedom did not.

A raise should increase your assets before it increases your lifestyle.

The Rich Dad Idea: Assets First

Robert Kiyosaki's useful simplification:

Assets put money into your pocket.

Liabilities take money out of your pocket.

Financial Thinking for Scientists

- Your Greatest Asset is Time
- Avoid the Three Delays:

  "I'll start when I make more money."
  "I'll invest after I buy everything I need."
  "I'll learn about finance later."

  $$\boxed{\text{Earn} \rightarrow \text{Save/Invest} \rightarrow \text{Spend}}$$

- Build a Machine, Not a Willpower Test

  Good scientists automate repetitive processes.
  Do the same with money.
  When income arrives: automatically
  save $\rightarrow$ invest $\rightarrow$ live on the remainder
  Don't make the same financial decision twelve times a year.
  Design the system once.
  This is essentially an algorithm for personal finance.

The poor and the middle class work for money
The rich have money work for them

- If you work for money, you give the power to your employer.
- If money works for you, you keep the power and control it.

One Equation to Remember

I might finish with a deliberately informal equation:

$$\text{Freedom} \approx \frac{\text{Assets + Skills + Time}}{\text{Lifestyle Demand}}$$

Invest time to get skills. (*The 15% Rule*)
Earn with your skills.
Build assets with your income.
Buy back your time.

CHAPTER 6

Finite Element and Finite Volume Methods for Elliptic Equations

This chapter consideres finite element and finite volume methods for elliptic PDEs defined on 1D and 2D regions.

Contents of Chapter 6

6.1. The Hilbert spaces . . . 164
6.2. Abstract Variational Problem . . . 166
6.3. Finite Element (FE) Methods in 1D Space . . . 168
6.4. An error estimate for FEM in 1D . . . 183
6.5. Other Variational Principles . . . 187
6.6. FEM for the Poisson Equation . . . 188
6.7. Finite Volume (FV) Method . . . 198
6.8. Average of The Diffusion Coefficient . . . 202
6.9. Numerical Examples with Python . . . 203
6.10. Homework . . . 206

6.1. The Hilbert spaces

- We first define the space of **square integrable functions** on I:

$$L^2(I) = \{v : v \text{ is defined on } I \text{ and } \int_I v^2 dx < \infty\}.$$

- The space $L^2(I)$ is a Hilbert space with the scalar product

$$(v, w) = \int_I v(x)w(x)dx$$

and the corresponding norm (the L^2-norm)

$$\|v\| = (v, v)^{1/2} = \Big(\int_I [v(x)]^2 \, dx \Big)^{1/2}.$$

- In general, for an integer $r \geq 0$, we define a Hilbert space

$$H^r(I) = \{v \in L^2(I) : v^{(k)} \in L^2(I), \;\; k = 1, \cdots, r\}$$

with the corresponding norm (the $H^r(I)$-norm)

$$\|v\|_r = \Big(\int_I \sum_{k=0}^{r} \Big[v^{(k)}(x) \Big]^2 \, dx \Big)^{1/2},$$

where $v^{(k)}$ denotes the k-th derivative of v. It is often convenient to define

$$|v|_r = \Big(\int_I \Big[v^{(r)}(x) \Big]^2 \, dx \Big)^{1/2}, \quad v \in H^r(I).$$

- Note that $L^2(I) = H^0(I)$ and $\| \cdot \| = \| \cdot \|_0 = | \cdot |_0$.

The following shall be useful for the error estimate to be presented in §6.4.

- The **Cauchy-Schwarz inequality** reads

$$|(v, w)| \leq \|v\| \cdot \|w\|. \tag{6.1}$$

- Consider the problem (D) in (6.9). Then, it is well known that

$$\|u\|_{s+2} \leq C\|f\|_s, \quad s = 0, 1, \cdots, \tag{6.2}$$

  for some $C > 0$, independent of u and f.

  The above **regularity estimate** holds for higher-dimensional problems (the Poisson equation in 2D and 3D) when the boundary is smooth enough. See Section 4.1 for the details.

6.2. Abstract Variational Problem

Let V be a normed space and consider the following **abstract variational problem**.

Find $u \in V$ such that

$$a(u, v) = f(v), \quad \forall v \in V, \tag{6.3}$$

where $a(\cdot, \cdot) : V \times V \to \mathbb{R}$ is a continuous bilinear form and $f : V \to \mathbb{R}$ is a continuous linear form.

Theorem 6.1. (Lax-Milgram Lemma) *Suppose that V is a Hilbert space with norm $\|\cdot\|$. Let $a(\cdot, \cdot) : V \times V \to \mathbb{R}$ is a continuous **V-elliptic bilinear form** in the sense that*

$$\exists\, \alpha \ \text{ s.t. } \ \alpha \|v\|^2 \leq a(v, v), \ \ \forall v \in V, \tag{6.4}$$

and $f : V \to \mathbb{R}$, a continuous linear form. Then, the abstract variational problem (6.3) has one and only one solution.

Existence and uniqueness of the solution:

Consider the Laplace equation

$$\begin{aligned} -\Delta u &= f, \quad \mathbf{x} \in \Omega, \\ u &= 0, \qquad \mathbf{x} \in \Gamma = \partial\Omega. \end{aligned} \tag{6.5}$$

Then, using the **Green's formula**, its variational problem is formulated as follows:

Find $u \in V = H_0^1(\Omega)$ such that

$$a(u, v) \equiv (\nabla u, \nabla v) = (f, v) \equiv f(v), \ \ \forall v \in V. \tag{6.6}$$

Here the Hilbert space

$$H_0^1(\Omega) = \{v : \ v \text{ and } \nabla v \text{ are square-integrable and } \ v|_\Gamma = 0\}$$

equipped with the norm $\|\cdot\|_1$ defined as

$$\|v\|_1^2 = \|v\|_0^2 + \|\nabla v\|_0^2$$

Theorem 6.2. *The variational problem (6.6) has a unique solution.*

Proof. Application of the **Cauchy-Schwarz inequality** shows that

$$|(\nabla u, \nabla v)| \leq \|\nabla u\|_0 \cdot \|\nabla v\|_0 \leq \|\nabla u\|_1 \cdot \|\nabla v\|_1,$$

which implies that $a(\cdot, \cdot)$ is continuous on $H_0^1(\Omega) \times H_0^1(\Omega)$.

Using the **Poincaré inequality**,

$$\int_\Omega u^2 d\mathbf{x} \leq C \int_\Omega |\nabla u|^2 d\mathbf{x}, \ \ \forall v \in H_0^1(\Omega), \tag{6.7}$$

or

$$\|v\|_0^2 \leq C\|\nabla v\|_0^2 = Ca(v, v),$$

we obtain

$$\|v\|_0^2 + \|\nabla v\|_0^2 \leq (1 + C)\|\nabla v\|_0^2 = (1 + C)a(v, v).$$

That is,

$$\frac{1}{1 + C}\|v\|_1^2 \leq a(v, v) \tag{6.8}$$

which shows that $a(\cdot, \cdot)$ is V-elliptic. Hence, by the Lax-Milgram lemma, the variational problem has a unique solution. □

The V-ellipticity is sometimes said to be **coercive**.

6.3. Finite Element (FE) Methods in 1D Space

Consider the model problem formulated in 1D space:

$$\text{(D)} \quad \begin{cases} -u'' = f, & x \in I = (0,1), \\ u = 0, & x = 0,\ 1, \end{cases} \tag{6.9}$$

which we call the **differential problem (D)**.

- FEM begins with a **variational formulation** for the given differential problem.
- The variational formulation is sometimes called the **weak formulation**.

6.3.1. Variational formulation

Define the product

$$(v, w) = \int_I v(x)w(x)dx \tag{6.10}$$

and the **linear space**

$$\begin{aligned} V = \{v : v \in C^0[0,1];\ \ v' \text{ is piecewise continuous} \\ \text{and bounded on } [0,1];\ \ v(0) = v(1) = 0\}. \end{aligned} \tag{6.11}$$

Variational Problem:

Use the **integration by parts** to have

$$\int_I -u''v \ = \ -u'v\Big|_0^1 + \int_I u'v' \ = \ \int_I u'v', \quad \forall\, v \in V.$$

Then, (6.9) can be written as

$$(u', v') = (f, v), \quad \forall\, v \in V. \tag{6.12}$$

Now, we define the **variational problem (V)** corresponding to the differential problem (6.9):

$$\text{(V)} \quad \begin{cases} \text{Find } u \in V \text{ such that} \\ (u', v') = (f, v), \ \ \forall\, v \in V. \end{cases} \tag{6.13}$$

Claim 6.3. The problem (D) is equivalent to the problem (V), when solutions are sufficiently smooth.

Proof. ((D) $\Rightarrow$ (V)): Clear.

((D) $\Leftarrow$ (V)): Let u be a solution of (V). Then,

$$(u', v') = (f, v), \ \ \forall\, v \in V. \tag{6.14}$$

Now, assume that u'' exists. Then, because

$$(u', v') = \int_I u'v' = u'v\Big|_0^1 - \int_I u''v = (-u'', v),$$

Equation (6.14) reads

$$(u'' + f, v) = 0, \ \ \forall\, v \in V.$$

So u should satisfy (6.9). □

Minimization Problem:

Define a functional $F : V \to \mathbb{R}$ as

$$F(v) = \frac{1}{2}(v', v') - (f, v), \ \ v \in V. \tag{6.15}$$

Then, the **minimization problem (M)** is formulated as

$$\text{(M)} \quad \begin{cases} \text{Find } u \in V \text{ such that} \\ F(u) \leq F(v), \ \ \forall\, v \in V. \end{cases} \tag{6.16}$$

Claim 6.4. The minimization problem (M) is equivalent to the variational problem (V).

Proof. ($\Rightarrow$): Let u be a solution of (M). Then,

$$F(u) \leq F(u + \varepsilon v), \ \ \forall\, v \in V, \ \ \forall \varepsilon \in \mathbb{R}. \tag{6.17}$$

Define $g(\varepsilon) = F(u + \varepsilon v)$. Then, $g'(0) = 0$. Since

$$g(\varepsilon) = \frac{1}{2}(u', u') + \varepsilon(u', v') + \frac{\varepsilon^2}{2}(v', v') - (f, u) - \varepsilon(f, v), \tag{6.18}$$

we have

$$g'(\varepsilon)\Big|_{\varepsilon=0} = \left[(u', v') + \varepsilon(v', v') - (f, v)\right]\Big|_{\varepsilon=0} = 0, \ \ \forall\, v \in V.$$

So, we conclude $(u', v') = (f, v), \ \ \forall\, v \in V$.

($\Leftarrow$): Now, let u be a solution of (V). Then, the objective is to show $F(u) \leq F(v), \ \ \forall\, v \in V$. For given $v \in V$, let $w = v - u$. Then, $w \in V$ and

$$\begin{aligned} F(v) &= F(u + w) = \frac{1}{2}(u' + w', u' + w') - (f, u + w) \\ &= \frac{1}{2}(u', u') - (f, u) + \frac{1}{2}(w', w') + (u', w') - (f, w). \end{aligned} \tag{6.19}$$

The last two terms in the right side of the above equation become zero, because u be a solution of (V). So

$$F(v) = F(u) + \frac{1}{2}(w', w') \geq F(u), \ \ \forall\, v \in V, \tag{6.20}$$

which completes the proof. □

Claim 6.5. The problem (V) admits a unique solution.

Proof. Existence and uniqueness can be proved in an abstract mathematical theory for variational problems, using the **Lax-Milgram lemma**, as in Theorem 6.2 on p.167. Here we will consider uniqueness only.

(Uniqueness): Let u_1 and u_2 be two solutions of (V). Then,

$$\begin{aligned}(u_1', v') &= (f, v), \quad \forall\, v \in V, \\ (u_2', v') &= (f, v), \quad \forall\, v \in V,\end{aligned} \tag{6.21}$$

which reads

$$(u_1' - u_2', v') = 0, \quad \forall\, v \in V. \tag{6.22}$$

Thus, by choosing $v = (u_1 - u_2)$, we reach at

$$\int_I (u_1' - u_2')^2 dx = 0, \tag{6.23}$$

which implies $u_1' - u_2' = 0$ and therefore $u_1 - u_2 = c$, a constant. Since $u_1(0) = u_2(0) = 0$, the constant c must be zero. Thus $u_1 \equiv u_2$, which completes the proof. □

Summary 6.6.

- (D) $\Leftrightarrow$ (V) $\Leftrightarrow$ (M). (when u'' exists)
- They admit a unique solution.

6.3.2. Formulation of FEMs

In designing a FEM, the following four steps are to be performed:

- **Partitioning**:
  The domain should be partitioned into **a collection of elements** of the mesh size h.
- **Subspace $V_h \subset V$ and basis functions $\{\varphi_j(x)\}$**:
  A finite-dimensional subspace is set to represent the numerical solution that is a linear combination of basis functions. That is,

$$u_h(x) = \sum_{j=1}^{M} \xi_j \varphi_j(x). \tag{6.24}$$

  For example, $\varphi_j(x)$ are piecewise polynomials (splines).
- **Application of variational principles**:
  Different variational principles produce various FEMs.
  - the minimization principle (Rayleigh-Ritz)
  - weighted residual approaches with the weights being either the basis functions (Galerkin) or different functions (Petrov-Galerkin)
  - least-square approaches
  - collocation method

  Simply put, this step involves **formulating a relationship between the unknown variables** $\{\xi_j\}$.
- **Assembly**:
  The linear system can be assembled for $(\xi_1, \xi_2, \cdots, \xi_M)^T$ with the integrals approximated by numerical quadrature.

Step 1. Partitioning

Let

$$0 = x_0 < x_1 < \cdots < x_M < x_{M+1} = 1$$

be a partition of the unit interval. Define

$$h_j = x_j - x_{j-1}, \;\; I_j = [x_{j-1}, x_j], \quad j = 1, 2, \cdots, M+1$$

and

$$h = \max_{1 \le j \le M+1} h_j. \tag{6.25}$$

Step 2. Subspace and Basis Functions

Define a finite-dimensional subspace of V as

$$\begin{aligned} V_h \;\; = \;\; \{v \in V : v \text{ is a polynomial of} \\ \text{degree} \le k \text{ on each } I_j\}. \end{aligned} \tag{6.26}$$

Note:

- Corresponding basis functions are determined depending on the choice of polynomial degree $k \ge 1$ and therefore on the nodal points.
- Each of basis functions is related to a nodal point.
- Basis functions $\varphi_j \in V_h$ are defined to satisfy

$$\varphi_j(x_i) = \delta_{ij} := \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{else.} \end{cases}$$

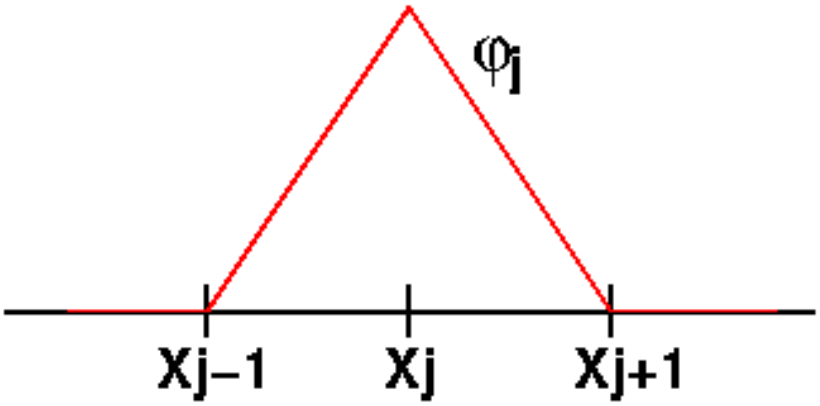


Figure 6.1: The basis function φ_j.

Example 6.7. ***$k = 1$ (the linear FEM)***:
The basis function φ_j is depicted in Figure 6.1:

$$\varphi_j(x) = \begin{cases} \dfrac{1}{h_j}(x - x_{j-1}), & x \in [x_{j-1}, x_j], \\ \dfrac{-1}{h_{j+1}}(x - x_{j+1}), & x \in [x_j, x_{j+1}], \\ 0, & \text{elsewhere.} \end{cases} \tag{6.27}$$

Note:

- The functions $v \in V_h$ can be expressed as a linear combination of the basis functions as

$$v(x) = \sum_{j=1}^{M} \eta_j \varphi_j(x), \quad x \in [0, 1].$$

- The above expression is unique for given $v \in V_h$; in fact,

$$\eta_j = v(x_j), \quad j = 1, 2, \cdots, M.$$

Example **6.8.** ***$k > 1$* (higher-order FEMs)**:

- For each interval $I_j = [x_{j-1}, x_j]$, the degree of freedom of k-th order polynomials is $k + 1$.

  It requires to choose $k + 1$ nodal points in each interval.
- As for the linear FEM, the two endpoints can naturally become nodal points.

  We should select $k - 1$ extra nodal points inside the interval I_j.
- In the literature, a common practice is to select those nodal points in such a way that the numerical quadrature of the integrals is as accurate as possible when the nodal points are used as quadrature points.
- Such selection is related to the family of orthogonal polynomials such as Legendre polynomials and Chebyshev polynomials; see Appendix B for details.

Step 3. Application of variational principles

The most popular FEM is the **Galerkin method**, which is a **weighted residual approach** with the weights being basis functions.

Weighted residual approaches:

- Let $P(u) = -u''$. For the differential problem (6.9), define the residual R as
$$R(v) = P(v) - f \tag{6.28}$$
- Then, we have
$$R(u) = P(u) - f = 0. \tag{6.29}$$
- However, for $u_h(x) = \sum_{j=1}^{M} \xi_j \varphi_j(x)$,
$$R(u_h) = P(u_h) - f \neq 0, \quad \text{in general.} \tag{6.30}$$
- Weighed residual approaches are seeking an approximate solution
$$u_h(x) = \sum_{j=1}^{M} \xi_j \varphi_j(x)$$
which satisfies
$$\int_I R(u_h)\, w(x)\, dx = 0, \tag{6.31}$$
for a sequence of **weight functions** $w(x) \in \{w_i(x)\}$, which is also called **trial functions**.
- When the integration by parts is utilized, (6.31) reads
$$(u_h', w') = (f, w) \tag{6.32}$$

The linear Galerkin method

For the subspace V_h of **linear** basis functions $\{\varphi_j(x)\}$, let

$$w_i(x) = \varphi_i(x) \tag{6.33}$$

Then, the linear Galerkin FEM for the differential problem (6.9) is formulated as

$$\text{Find } u_h \in V_h \text{ s.t. } (u_h', \varphi_i') = (f, \varphi_i),\ \forall\, \varphi_i \in V_h \tag{6.34}$$

As in §6.3.1, one can show that (6.34) admits a unique solution.

Summary 6.9.

- Given basis functions $\{\varphi_j(x)\} \subset V_h$, the numerical solution u_h is uniquely expressed as

$$u_h(x) = \sum_{j=1}^{M} \xi_j \varphi_j(x). \tag{6.35}$$

- The numerical solution must be the solution of a variational formulation. For example, the solution of the **linear Galerkin FEM** satisfies

$$(u_h', \varphi_i') = (f, \varphi_i),\ \forall\, \varphi_i \in V_h \tag{6.36}$$

Step 4. Assembly for a Linear System

The next objective is to assemble a **linear system** for the unknown vector $\boldsymbol{\xi} := (\xi_1, \xi_2, \cdots, \xi_M)^T$. From (6.35) and (6.36),

$$(u_h', \varphi_i') = \sum_{j=1}^{M} \xi_j (\varphi_j', \varphi_i') = (f, \varphi_i), \ \ \forall\, \varphi_i \in V_h. \tag{6.37}$$

Since $u_h = \sum \xi_j \varphi_j$, we can rewrite the above equation

$$\sum_{j=1}^{M} (\varphi_j', \varphi_i') \xi_j = (f, \varphi_i), \ \ i = 1, \cdots, M. \tag{6.38}$$

Define

$$a_{ij} = (\varphi_j', \varphi_i'), \quad b_i = (f, \varphi_i). \tag{6.39}$$

Then, (6.38) equivalently reads the algebraic system of the form

$$A\boldsymbol{\xi} = \mathbf{b}, \tag{6.40}$$

where $A = (a_{ij})$ is an $M \times M$ matrix and $\mathbf{b} = (b_1, b_2, \cdots, b_M)^T$ is the source vector.

Note:

- The matrix A has good properties such as being symmetric and positive definite.
- We will show them later; we first consider details for the computation of a_{ij} and b_i.
- Note that

$$a_{ij} = (\varphi_j', \varphi_i') = \int_I \varphi_j'(x) \varphi_i'(x) dx = 0, \quad \text{if } |i - j| \geq 2,$$

because the support of φ_j is $[x_{j-1}, x_{j+1}]$. Thus, there are only three cases for nonzero entries of A:

$$j = i - 1,\ i,\ i + 1.$$

Computation of a_{ij} and b_i: Recall

$$\varphi_j(x) = \begin{cases} \dfrac{1}{h_j}(x - x_{j-1}), & x \in [x_{j-1}, x_j], \\ \dfrac{-1}{h_{j+1}}(x - x_{j+1}), & x \in [x_j, x_{j+1}], \\ 0, & \text{elsewhere.} \end{cases} \tag{6.41}$$

Case $j = i - 1$: It follows from (6.41) that

$$\begin{aligned} a_{i,i-1} &= (\varphi'_{i-1}, \varphi'_i) = \int_{x_{i-1}}^{x_i} \varphi'_{i-1}(x)\varphi'_i(x)dx \\ &= \int_{x_{i-1}}^{x_i} \frac{-1}{h_i} \cdot \frac{1}{h_i} dx = \frac{-1}{h_i}. \end{aligned}$$

Case $j = i$: Again utilizing (6.41), we have

$$\begin{aligned} a_{i,i} &= (\varphi'_i, \varphi'_i) = \int_{x_{i-1}}^{x_{i+1}} \varphi'_i(x)\varphi'_i(x)dx \\ &= \int_{x_{i-1}}^{x_i} + \int_{x_i}^{x_{i+1}} \varphi'_i(x)\varphi'_i(x)dx = \frac{1}{h_i} + \frac{1}{h_{i+1}}. \end{aligned}$$

Case $j = i + 1$:

$$\begin{aligned} a_{i,i+1} &= (\varphi'_{i+1}, \varphi'_i) = \int_{x_i}^{x_{i+1}} \varphi'_{i+1}(x)\varphi'_i(x)dx \\ &= \int_{x_i}^{x_{i+1}} \frac{1}{h_{i+1}} \cdot \frac{-1}{h_{i+1}} dx = \frac{-1}{h_{i+1}}. \end{aligned}$$

Computation of b_i: Finally, it can be done as

$$b_i = (f, \varphi_i) = \int_{x_{i-1}}^{x_{i+1}} f(x)\varphi_i(x)dx \approx f_i \frac{h_i + h_{i+1}}{2},$$

where f has been approximated by $f_i = f(x_i)$ on $[x_{i-1}, x_{i+1}]$.

Properties of the algebraic system

Definition 6.10. A matrix $S = (s_{ij}) \in \mathbb{R}^{M\times M}$ is said to be **positive definite** if

$$\boldsymbol{\eta} \cdot S\boldsymbol{\eta} = \sum_{i,j=1}^{M} \eta_i s_{ij} \eta_j > 0, \quad \forall\, \boldsymbol{\eta} \in \mathbb{R}^M, \ \ \boldsymbol{\eta} \neq 0.$$

It has been known that a matrix S is symmetric positive definite if and only if all eigenvalues of S are strictly positive.

Lemma 6.11. The matrix A in (6.40) is symmetric positive definite.

Proof. Symmetry is easy to see, because

$$a_{ij} := (\varphi_j', \varphi_i') = (\varphi_i', \varphi_j') =: a_{ji}.$$

Given $\boldsymbol{\eta} \in \mathbb{R}^M$, we define $v(x) = \sum_{j=1}^{M} \eta_j \varphi_j(x)$. Then

$$\begin{aligned} \boldsymbol{\eta} \cdot A\boldsymbol{\eta} = \sum_{i,j=1}^{M} \eta_i a_{ij} \eta_j &= \sum_{i,j=1}^{M} \eta_i (\varphi_i', \varphi_j') \eta_j \\ &= \Big(\sum_{i}^{M} \eta_i \varphi_i', \sum_{j}^{M} \eta_j \varphi_j' \Big) \geq 0, \end{aligned} \tag{6.42}$$

with equality satisfied only if $v' = 0$, and therefore only if $v = 0$ because $v(0) = 0$; which implies that equality holds only if $\boldsymbol{\eta} = 0$. This completes the proof. □

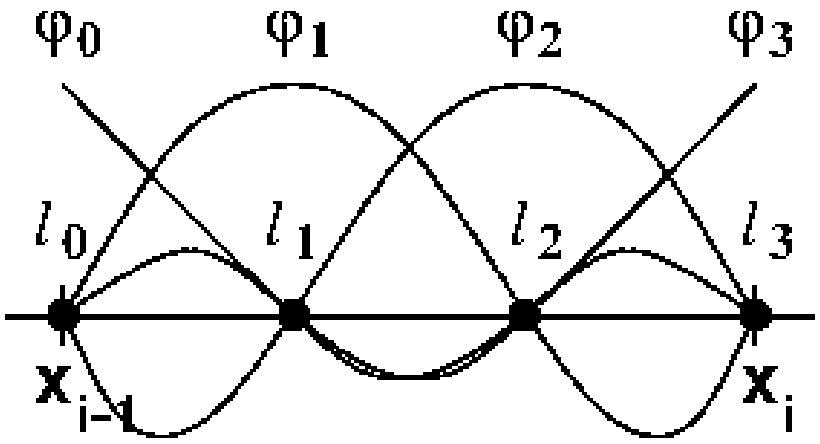


Figure 6.2: The element $I_i = [x_{i-1}, x_i]$ and the basis functions for the cubic FE method.

Higher-order FEMs:

- Higher-order FE methods introduce higher-order basis functions.
- Figure 6.2 presents the element $I_i = [x_{i-1}, x_i]$ and the basis functions each of which is cubic in I_i.
- Since the degree of freedom for cubic polynomials is four, we need to provide four independent information to determine the polynomial uniquely.
- For the purpose, one can choose four distinct points (including two edge points), as shown in Figure 6.2. The points are called the **nodal points**.

Construction of cubic basis functions

- Let the nodal points be given and denoted by ℓ_p, $p = 0, \cdots, 3$.
- Then the local basis functions φ_j on the element I_i must read

$$\varphi_j(\ell_p) = \delta_{jp}, \quad j, p = 0, \cdots, 3.$$

- The above property can be satisfied the **cardinal functions**:

$$\varphi_j(x) = \prod_{\substack{m=0 \\ m \neq j}}^{3} \left(\frac{x - \ell_m}{\ell_j - \ell_m} \right), \quad j = 0, \cdots, 3, \tag{6.43}$$

  and they can serve as basis functions.
- It is often to choose Gauss-Lobatto points for the nodal points; see Appendix B for details.

Construction of general-order basis functions

We generalize the above argument for FE methods utilizing piecewise kth-order polynomials $k \geq 1$, as follows:

- Select extra $(k-1)$ nodal points such that each element I_i has $(k+1)$ nodal points including the two edge points.
- Denote them by ℓ_m, $m = 0, \cdots, k$.
- Define the local basis functions as

$$\varphi_j(x) = \prod_{\substack{m=0 \\ m \neq j}}^{k} \left(\frac{x - \ell_m}{\ell_j - \ell_m} \right), \quad j = 0, \cdots, k.$$

- The basis functions associated with the edge points must be extended both side for the final form of the basis functions.

6.4. An error estimate for FEM in 1D

Let u and u_h be the solutions of Problem (V) in (6.13) and Problem (V_h) in (6.34), respectively. Then,

$$\begin{aligned}(u', v') &= (f, v), \quad \forall\, v \in V,\\ (u_h', v') &= (f, v), \quad \forall\, v \in V_h.\end{aligned} \tag{6.44}$$

Note that $V_h \subset V$. Thus it follows from the above equations that

$$(u' - u_h', v') = 0, \quad \forall\, v \in V_h. \tag{6.45}$$

Theorem 6.12. *For any $v \in V_h$, we have*

$$\|(u - u_h)'\| \le \|(u - v)'\|. \tag{6.46}$$

Proof. Given v, an arbitrary function in V_h, let $w = u_h - v \in V_h$. Then, utilizing (6.45) and the Cauchy-Schwarz inequality, we have

$$\begin{aligned}\|(u - u_h)'\|^2 &= ((u - u_h)', (u - u_h)')\\ &= ((u - u_h)', (u - u_h)') + ((u - u_h)', w')\\ &= ((u - u_h)', (u - u_h + w)')\\ &= ((u - u_h)', (u - v)')\\ &\le \|(u - u_h)'\| \cdot \|(u - v)'\|,\end{aligned}$$

from which (6.46) follows. □

Note:

- The inequality (6.46) allows us to analyze the error $\|(u - u_h)'\|$ *quantitatively*.
- That is, we can choose $v \in V_h$ *suitably* to estimate the right side of (6.46).
- We shall choose v to be the **interpolant** of u, $\pi_h u$, which interpolates u at all the nodal points x_j. See Figure 6.3.

Now, one can prove that for $x \in [0, 1]$,

$$|u(x) - \pi_h u(x)| \le \frac{h^2}{8} \max_{\xi \in (0,1)} |u''(\xi)|, \tag{6.47}$$

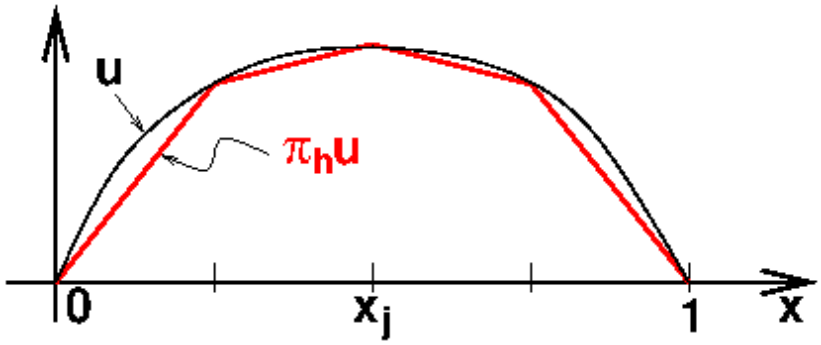


Figure 6.3: The solution u and its interpolant $\pi_h u$.

$$|u'(x) - \pi_h u'(x)| \le h \max_{\xi\in(0,1)} |u''(\xi)|. \tag{6.48}$$

(*See Homework 6.2.*) The above inequalities hold for any (sufficiently smooth) function u and its interpolant $\pi_h u$. The estimates are called the **interpolation estimate**.

- It follows from (6.48) and Theorem 6.12 that

$$\|(u - u_h)'\|_0 \le Ch|u|_2, \tag{6.49}$$

  for some constant $C > 0$, independent of h.

- Since

$$\begin{aligned} |(u - u_h)(x)| &= \Big| \int_0^x (u - u_h)'(t)dt \Big| \\ &\le \|(u - u_h)'\|_0 \cdot \Big(\int_0^x 1^2 dt \Big)^{1/2} \\ &\le \|(u - u_h)'\|_0, \end{aligned}$$

  we have

$$|(u - u_h)(x)| \le Ch|u|_2. \tag{6.50}$$

- Therefore, from (6.49) and (6.50),

$$\|u - u_h\|_1 \le Ch|u|_2. \tag{6.51}$$

Estimation of $\|u - u_h\|_0$

Theorem 6.13. *Let u and u_h be the solutions of Problem* (V) *and Problem* (V_h)*, respectively. Then*

$$\|u - u_h\|_0 \le Ch^2|u|_2, \tag{6.52}$$

where $C > 0$ is independent on h.

Proof.

- Let $e = u - u_h$. Then, we know from (6.45) that

$$(e', v') = 0, \quad \forall v \in V_h. \tag{6.53}$$

- We shall estimate $(e, e) = \|e\|_0^2$ using the so-called **duality argument** which is popular in FEM error analysis.
- Let ϕ be the solution of the following dual problem

$$\begin{aligned} -\phi'' &= e, \quad x \in I, \\ \phi &= 0, \quad x = 0 \text{ or } 1. \end{aligned} \tag{6.54}$$

- Then, from (6.2) with $s = 0$,

$$\|\phi\|_2 \le C\|e\|_0, \tag{6.55}$$

  where $C > 0$ is independent on e.
- Using the integration by parts and the fact that $e(0) = e(1) = 0$,

$$(e, e) = (e, -\phi'') = (e', \phi') = (e', \phi' - \pi_h\phi'),$$

  where $\pi_h\phi \in V_h$ denotes the interpolant of ϕ.
- Now, apply the interpolation estimate (6.48) to ϕ and use the regularity estimate (6.55) to get

$$\|e\|_0^2 \le \|e\|_1 \cdot \|\phi - \pi_h\phi\|_1 \le \|e\|_1 \cdot Ch|\phi|_2 \le Ch\|e\|_1 \cdot \|e\|_0.$$

- Thus dividing by $\|e\|_0$ and utilizing (6.51), we finally reach at

$$\|e\|_0 \le Ch\|e\|_1 \le Ch^2|u|_2$$

  and the proof is complete. □

Remark 6.14.

- **Error estimate for the linear FEM**

  The error estimates in (6.51) and (6.52) can be rewritten as

$$\|u - u_h\|_s \leq Ch^{2-s}|u|_2, \quad s = 0, 1. \tag{6.56}$$

- **Error estimate for general-order FEMs**

  When piecewise k-th order polynomials ($k \geq 1$) are employed for the basis functions, one can use the same arguments presented in this section to show

$$\|u - u_h\|_s \leq Ch^{k+1-s}|u|_{k+1}, \quad s = 0, 1, \cdots, k. \tag{6.57}$$

6.5. Other Variational Principles

The FEM we have consider so far is the Galerkin method, one of weighted residual approaches.

There have been other variational principles such as

- the minimization principle (Rayleigh-Ritz methods),
- least-square approaches,
- collocation methods, and
- weighted residual approaches with the weights being different from the basis functions (Petrov-Galerkin methods).

6.6. FEM for the Poisson Equation

Let $\Omega \subset \mathbb{R}^2$ be bounded domain with its boundary $\Gamma = \partial\Omega$ being smooth. Consider

$$\text{(D)} \quad \begin{cases} -\Delta u = f, & \mathbf{x} \in \Omega, \\ u = 0, & \mathbf{x} \in \Gamma, \end{cases} \tag{6.58}$$

where $\mathbf{x} = (x, y) = (x_1, x_2)$.

6.6.1. Integration by Parts

To derive a variational form for (6.58), we first introduce the divergence theorem. Let $A = (a_1, a_2)$ be a vector-valued function on $\mathbb{R}^2$. Then **divergence** of A is defined as

$$\nabla \cdot A = \frac{\partial a_1}{\partial x_1} + \frac{\partial a_2}{\partial x_2}.$$

Let $\mathbf{n} = (n_1, n_2)$ be the **outward unit normal** to Γ and

$$v_{\mathbf{n}} = \frac{\partial v}{\partial \mathbf{n}} = \nabla v \cdot \mathbf{n} = \frac{\partial v}{\partial x_1} n_1 + \frac{\partial v}{\partial x_2} n_2.$$

Theorem 6.15. (Divergence theorem) *Let $A = (a_1, a_2)$ be a vector-valued differentiable function on a bounded region Ω in $\mathbb{R}^2$, with its boundary Γ. Then*

$$\int_\Omega \nabla \cdot A \, d\mathbf{x} = \int_\Gamma A \cdot \mathbf{n} \, ds, \tag{6.59}$$

where s is the element of arc length.

- Apply the divergence theorem to $A = (vw, 0)$ and $A = (0, vw)$ to read

$$\int_\Omega \frac{\partial}{\partial x_1}(vw)d\mathbf{x} = \int_\Gamma vwn_1 ds,$$
$$\int_\Omega \frac{\partial}{\partial x_2}(vw)d\mathbf{x} = \int_\Gamma vwn_2 ds,$$

  which implies

$$\int_\Omega \frac{\partial v}{\partial x_i} w d\mathbf{x} = \int_\Gamma vwn_i ds - \int_\Omega v\frac{\partial w}{\partial x_i}d\mathbf{x}, \quad i = 1, 2. \tag{6.60}$$

- Thus we have the **Green's formula**

$$\begin{aligned}\int_\Omega \nabla v \cdot \nabla w d\mathbf{x} &\equiv \int_\Omega \left[\frac{\partial v}{\partial x_1}\frac{\partial w}{\partial x_1} + \frac{\partial v}{\partial x_2}\frac{\partial w}{\partial x_2}\right] \\ &= \int_\Gamma v\frac{\partial w}{\partial x_1}n_1 ds - \int_\Omega v\frac{\partial^2 w}{\partial {x_1}^2}d\mathbf{x} \\ &\quad + \int_\Gamma v\frac{\partial w}{\partial x_2}n_2 ds - \int_\Omega v\frac{\partial^2 w}{\partial {x_2}^2}d\mathbf{x} \\ &= \int_\Gamma v\frac{\partial w}{\partial \mathbf{n}}ds - \int_\Omega v\Delta w d\mathbf{x}.\end{aligned}$$

That is,

$$(\nabla v, \nabla w) =< v, w_{\mathbf{n}} > -(v, \Delta w), \tag{6.61}$$

or equivalently

$$(-\Delta w, v) = - < w_{\mathbf{n}}, v > +(\nabla w, \nabla v), \tag{6.62}$$

where $< v, w >= \int_\Gamma vw\, ds$.

The Variational Formulation

- First, define the **linear space**

$$V = \{v : v \in C^0(\Omega);\ \ \nabla v \text{ is piecewise continuous and bounded on } \Omega;\ \ v(\mathbf{x}) = 0,\ \ \mathbf{x} \in \Gamma\}. \tag{6.63}$$

- Let

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v d\mathbf{x}.$$

- Define the **variational problem** (V)

$$\text{(V)} \quad \begin{cases} \text{Find } u \in V \text{ such that} \\ a(u, v) = (f, v),\ \ \forall\, v \in V. \end{cases} \tag{6.64}$$

- The **minimization problem** (M) is formulated as

$$\text{(M)} \quad \begin{cases} \text{Find } u \in V \text{ such that} \\ F(u) \le F(v),\ \ \forall\, v \in V, \end{cases} \tag{6.65}$$

  where

$$F(v) = \frac{1}{2} a(v, v) - (f, v).$$

Then, as for the 1D model problem in §6.3.1, one can prove that

- problems (D), (V), and (M) are equivalent when the solution u is sufficiently smooth, and
- they admit a unique solution.

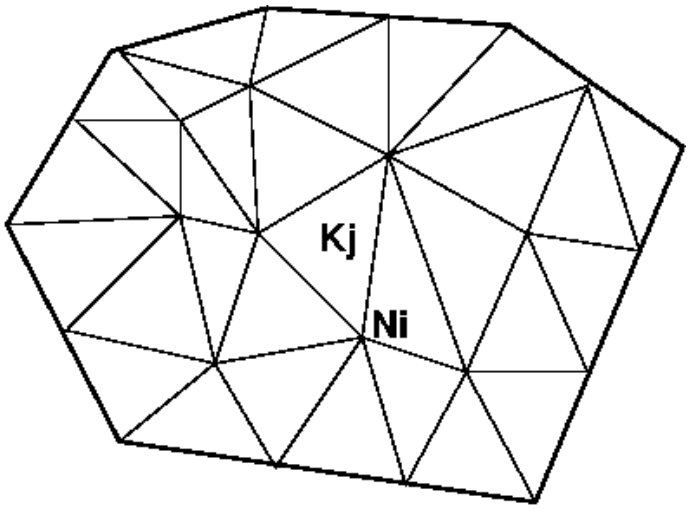


Figure 6.4: Triangulation $\mathcal{T}_h$ of Ω.

6.6.2. The Formulation of FEMs

To define an FEM for the Poisson equation (6.64), we need to follow steps as for the FE method for the 1D problem presented in §6.3.2:

- Triangulation
- Subspace $V_h \subset V$ and basis functions
- Application of variational principles
- Assembly for the linear system

Step 1. Triangulation

- Let K_j, $j = 1, \cdots, m$, be nonoverlapping triangles such that

$$\Omega = \cup_{j=1}^{m} K_j;$$

  we assume that no vertex of a triangle lies on the edge of another triangle as shown in Figure 6.4.

- Let h be the longest side of edges of the triangles, i.e.,

$$h = \max_j \mathbf{diam}(K_j).$$

- Then the collection of such triangles composes the finite elements

$$\mathcal{T}_h = \{K_1, K_2, \cdots, K_m\}.$$

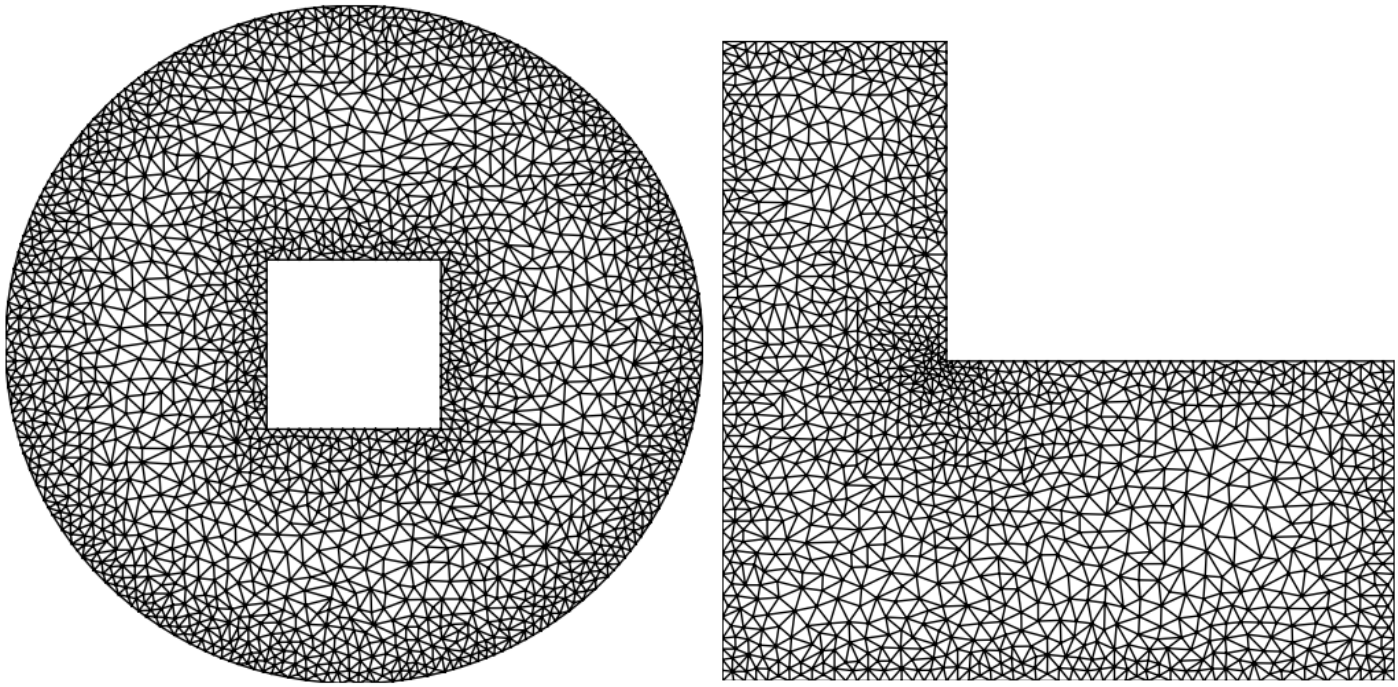


Figure 6.5: Two meshes generated using the Python package MeshPy.

- The `*.mesh` file, which saves an FE mesh, consists of

nPT	the number of vertices (points)
nEL	the number of elements/triangles
$(x, y)_i$	the vertices
$(n_1, n_2, n_3)_j$	the connectivity

Step 2. Subspace $V_h \subset V$ and Basis Functions

- For the linear FE method, for example, we define a subspace of V as

$$V_h = \{v \in V : v \text{ is linear on each } K_j\}. \tag{6.66}$$

- The corresponding basis functions $\{\varphi_j\}$ are as

$$\varphi_j(N_i) = \delta_{ij}, \tag{6.67}$$

  where N_i are the vertices, the nodal points.

- Each basis function φ_i restricted on an element K_j, one vertex of which is N_i, is linear of the form

$$\varphi_i(\mathbf{x}) = ax_1 + bx_2 + c, \quad \mathbf{x} \in K_j. \tag{6.68}$$

- Then the solution of the locally linear FEM can be expressed as

$$u_h(\mathbf{x}) = \sum_{j=1}^{M} \xi_j \varphi_j(\mathbf{x}), \text{ for some } M > 0. \tag{6.69}$$

Step 3. Application of Variational Principles

The **linear Galerkin FEM** for (6.64) can be formulated as

$$(\mathbf{V}_h) \quad \begin{cases} \text{Find } u_h \in V_h \text{ such that} \\ a(u_h, v) = (f, v), \ \forall\, v \in V_h. \end{cases} \tag{6.70}$$

The error analysis for the linear Galerkin method can be carried out following the arguments in §6.4.

Theorem 6.16.
Let u and u_h be the solutions of (6.64) and (6.70), respectively. Then

$$\|u - u_h\|_s \le Ch^{2-s}|u|_2, \quad s = 0, 1, \tag{6.71}$$

where $C > 0$ is a constant independent on h.

It is fun to prove the theorem; challenge it for an extra credit, or more importantly, for your pride!

Step 4. Assembly for the Linear System

As in (6.69), let

$$u_h(\mathbf{x}) = \sum_{j=1}^{M} \xi_j \varphi_j(\mathbf{x}), \quad \text{for some } M > 0.$$

Then, the algebraic system for (6.70), $a(u_h, v) = (f, v)$, can be formulated as

$$A\boldsymbol{\xi} = \mathbf{b}, \tag{6.72}$$

where $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_M)^T$ is the solution vector and

$$\begin{aligned} A &= (a_{ij}), \quad a_{ij} := a(\varphi_j, \varphi_i), \\ \mathbf{b} &= (b_1, \cdots, b_M)^T, \quad b_i := (f, \varphi_i). \end{aligned}$$

Note:

- As for the 1D problem in §6.3.2, the matrix A is symmetric and positive definite.
- Thus the system (6.72) admits a unique solution.

Stiffness matrix A:

Let the **stiffness matrix** be $A = (a_{ij})$. Then,

$$a_{ij} = a(\varphi_j, \varphi_i) = \sum_{K \in \mathcal{T}_h} a_{ij}^K, \tag{6.73}$$

where

$$a_{ij}^K = a_K(\varphi_j, \varphi_i) = \int_K \nabla\varphi_j \cdot \nabla\varphi_i d\mathbf{x}. \tag{6.74}$$

Definition 6.17. The **element stiffness matrix** A_K of the element K is

$$A_K = \begin{bmatrix} a_{11}^K & a_{12}^K & a_{13}^K \\ a_{21}^K & a_{22}^K & a_{23}^K \\ a_{31}^K & a_{32}^K & a_{33}^K \end{bmatrix},$$

where each component can be computed from (6.74).

- The stiffness matrix A can be constructed through the contributions from the element stiffness matrices A_K, $K \in \mathcal{T}_h$.
- Looks complicated? We will deal with an efficient method for the computation of a_{ij}^K in a separate section; see §6.6.3.

6.6.3. Assembly: Element stiffness matrices

- The computation of the element stiffness matrix

$$A^K := \left(a_{ij}^K\right) \in \mathbb{R}^{3\times 3}$$

  is not a simple task for the element $K \in \mathcal{T}_h$ in a general geometry.
- To overcome the complexity, we introduce the reference element $\widehat{K}$ and an affine mapping $F : \widehat{K} \to K$. See Figure 6.6.

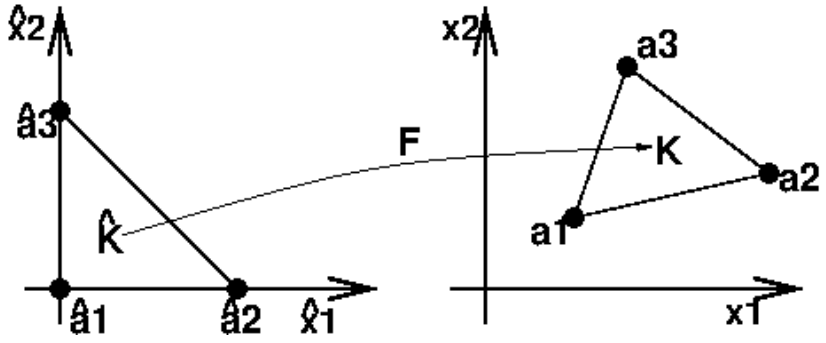


Figure 6.6: The affine mapping $F : \widehat{K} \to K$.

The reference element $\widehat{K}$: It has the following three vertices

$$\widehat{\mathbf{a}}_1 = [0,0]^T,\ \ \widehat{\mathbf{a}}_2 = [1,0]^T,\ \ \widehat{\mathbf{a}}_3 = [0,1]^T, \tag{6.75}$$

and the corresponding reference basis functions are

$$\widehat{\varphi}_1(\widehat{\mathbf{x}}) = 1 - \widehat{x}_1 - \widehat{x}_2,\ \ \widehat{\varphi}_2(\widehat{\mathbf{x}}) = \widehat{x}_1,\ \ \widehat{\varphi}_3(\widehat{\mathbf{x}}) = \widehat{x}_2. \tag{6.76}$$

Affine mapping F:

- The mapping $F : \widehat{K} \to K \ \ (\widehat{\mathbf{x}} \mapsto \mathbf{x})$ must be defined as

$$\mathbf{a}_i = F(\widehat{\mathbf{a}}_i), \quad \varphi_i(\mathbf{x}) = \widehat{\varphi}_i(\widehat{\mathbf{x}}), \quad i = 1, 2, 3. \tag{6.77}$$

  That is, the corners and the basis functions of K are defined as the affine images of those of $\widehat{K}$.

- Let J be the **Jacobian** of the affine mapping F:

$$J := \left[\frac{\partial F_i}{\partial \widehat{x}_j}\right] = \left[\frac{\partial x_i}{\partial \widehat{x}_j}\right] = \begin{bmatrix} \dfrac{\partial x_1}{\partial \widehat{x}_1} & \dfrac{\partial x_1}{\partial \widehat{x}_2} \\ \dfrac{\partial x_2}{\partial \widehat{x}_1} & \dfrac{\partial x_2}{\partial \widehat{x}_2} \end{bmatrix}. \tag{6.78}$$

- Then, it follows from the chain rule that

$$\nabla \varphi_j = J^{-T} \nabla \widehat{\varphi}_j, \quad j = 1, 2, 3, \tag{6.79}$$

  where J^{-T} is the transpose of J^{-1}, which implies

$$\begin{aligned} a_{ij}^K &:= \int_K \nabla \varphi_j \cdot \nabla \varphi_i d\mathbf{x} \\ &= \int_{\widehat{K}} (J^{-T} \nabla \widehat{\varphi}_j) \cdot (J^{-T} \nabla \widehat{\varphi}_i) \, |\det J| \, d\widehat{\mathbf{x}}. \end{aligned} \tag{6.80}$$

Note:

- Every affine mapping in $\mathbb{R}^n$ has the form $B\widehat{\mathbf{x}} + \mathbf{s}$, where $B \in \mathbb{R}^{n \times n}$ and $\mathbf{s} \in \mathbb{R}^n$.
- From some algebra, it can be shown that

$$F(\widehat{\mathbf{x}}) = [\mathbf{a}_2 - \mathbf{a}_1, \mathbf{a}_3 - \mathbf{a}_1] \, \widehat{\mathbf{x}} + \mathbf{a}_1 \tag{6.81}$$

  Thus

$$J = [\mathbf{a}_2 - \mathbf{a}_1, \mathbf{a}_3 - \mathbf{a}_1] \in \mathbb{R}^{2 \times 2}. \tag{6.82}$$

6.6.4. Extension to Neumann boundary conditions

Consider the following problem of Neumann boundary condition

$$\begin{aligned} -\Delta u + u &= f, \quad \mathbf{x} \in \Omega, \\ u_{\mathbf{n}} &= g, \quad \mathbf{x} \in \Gamma. \end{aligned} \tag{6.83}$$

For the problem, it is natural to choose $V = H^1(\Omega)$ for the linear space.

Integration by parts:

- It follows from the Green's formula (6.62) that (6.83) reads

$$(\nabla u, \nabla v) + (u, v) = (f, v) + < g, v >, \quad v \in V. \tag{6.84}$$

- Define

$$\begin{aligned} a(u, v) &= (\nabla u, \nabla v) + (u, v), \\ F(v) &= \frac{1}{2} a(v, v) - (f, v) - < g, v > . \end{aligned}$$

- Then, one can formulate the variational problem

$$\text{(V)} \quad \begin{cases} \text{Find } u \in V \text{ such that} \\ a(u, v) = (f, v) + < g, v >, \ \forall\, v \in V, \end{cases} \tag{6.85}$$

and the minimization problem

$$\text{(M)} \quad \begin{cases} \text{Find } u \in V \text{ such that} \\ F(u) \le F(v), \ \forall\, v \in V. \end{cases} \tag{6.86}$$

Note:

- In (6.85) the boundary condition is implicitly imposed. Such a boundary condition is called a **natural** boundary condition.
- On the other hand, the Dirichlet boundary condition as in (6.58) is called a **essential** boundary condition.
- For the problem (6.85), an FEM can be formulated as for (6.64); a similar error analysis can be obtained.

6.7. Finite Volume (FV) Method

Here we will discuss one of easiest FV methods formulated on a rectangular domain. For problems on more general domains or convection-dominated problems, the FV method can be more complicated. However, the major ideas would be near around the same corner.

Consider the following problem of general diffusion coefficients

$$\begin{aligned} -\nabla \cdot (a \nabla u) &= f, \quad \mathbf{x} \in \Omega, \\ u &= 0, \quad \mathbf{x} \in \Gamma. \end{aligned} \tag{6.87}$$

Formulation of FV methods

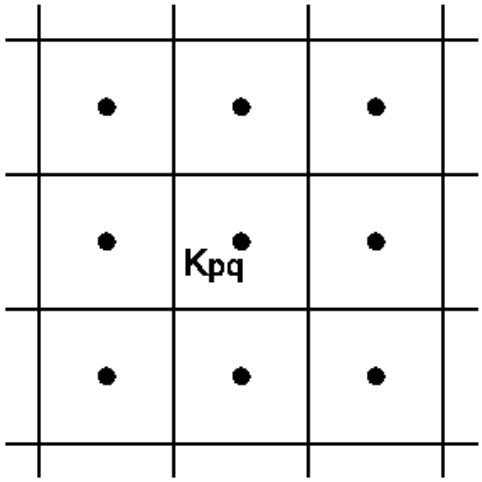


Figure 6.7: Cell-centered FV method on a uniform mesh of grid size $h_x \times h_y$. For this case, each cell is a **control volume**.

1. Triangulation:

Let Ω be a rectangular domain partitioned into elements, called **cells**. For simplicity, we assume all cells are rectangular of size $h_x \times h_y$. See Figure 6.7.

2. Localization:

Let ϕ_{pq} be the characteristic function of the cell K_{pq}, i.e.,

$$\phi_{pq}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in K_{pq}, \\ 0, & \text{else.} \end{cases}$$

3. Variational principle:

Multiplying the first equation of (6.87) by ϕ_{pq} and integrating the result over the domain Ω, we have

$$\int_{\Omega} -\nabla \cdot (a\nabla u)\phi_{pq} d\mathbf{x} = \int_{K_{pq}} -\nabla \cdot (a\nabla u) d\mathbf{x} = \int_{K_{pq}} f \, d\mathbf{x}.$$

Therefore, from the divergence theorem,

$$-\int_{\partial K_{pq}} a u_{\mathbf{n}_{pq}} \, ds = \int_{K_{pq}} f \, d\mathbf{x}, \tag{6.88}$$

where s is the edge element and $\mathbf{n}_{pq}$ denotes the unit out normal to ∂K_{pq}.

4. Approximation and evaluation:

Now we have to evaluate or approximate the quantity $a u_{\mathbf{n}_{pq}}$ along the boundary of the cell K_{pq}.

On $\partial K_{pq} \cap \partial K_{p+1,q}$ (“East", the right vertical edge), for example, it can be approximated as

$$a u_{\mathbf{n}_{pq}}(\mathbf{x}) \;\approx\; a_{p+1/2,q} \frac{u_{p+1,q} - u_{p,q}}{h_x}, \;\; \mathbf{x} \in \partial K_{pq} \cap \partial K_{p+1,q}, \tag{6.89}$$

where the approximation is second-order accurate.

Thus

$$\textbf{(E)} \quad \int_{K_{pq} \cap \partial K_{p+1,q}} a u_{\mathbf{n}_{pq}}(\mathbf{x}) ds \;\approx\; \frac{h_y}{h_x} a_{p+1/2,q} (u_{p+1,q} - u_{p,q}). \tag{6.90}$$

The same can be applied for other edges. That is,

$$\begin{aligned}
\textbf{(W)} \quad & \int_{K_{pq} \cap \partial K_{p-1,q}} a u_{\mathbf{n}_{pq}}(\mathbf{x}) ds \;\approx\; \frac{h_y}{h_x} a_{p-1/2,q} (u_{p-1,q} - u_{p,q}) \\
\textbf{(N)} \quad & \int_{K_{pq} \cap \partial K_{p,q+1}} a u_{\mathbf{n}_{pq}}(\mathbf{x}) ds \;\approx\; \frac{h_x}{h_y} a_{p,q+1/2} (u_{p,q+1} - u_{p,q}) \\
\textbf{(S)} \quad & \int_{K_{pq} \cap \partial K_{p,q-1}} a u_{\mathbf{n}_{pq}}(\mathbf{x}) ds \;\approx\; \frac{h_x}{h_y} a_{p,q-1/2} (u_{p,q-1} - u_{p,q})
\end{aligned} \tag{6.91}$$

The right-hand side term:

The right-hand side term of (6.88) can be integrated by the mass-lumping technique to become $h_x h_y f_{pq}$. That is,

$$\int_{K_{pq}} f \, d\mathbf{x} \approx h_x h_y f_{pq}. \tag{6.92}$$

For (6.88), combine (6.90), (6.91), and (6.92) and divide the resulting equation by $h_x h_y$ to have

$$\begin{aligned} &-\Big[\frac{1}{h_x^2} a_{p+1/2,q}(u_{p+1,q} - u_{p,q}) + \frac{1}{h_x^2} a_{p-1/2,q}(u_{p-1,q} - u_{p,q}) \\ &+ \frac{1}{h_y^2} a_{p,q+1/2}(u_{p,q+1} - u_{p,q}) + \frac{1}{h_y^2} a_{p,q-1/2}(u_{p,q-1} - u_{p,q}) \Big] \\ &= \frac{-a_{p-1/2,q}u_{p-1,q} + (a_{p-1/2,q} + a_{p+1/2,q})u_{p,q} - a_{p+1/2,q}u_{p+1,q}}{h_x^2} \\ &\quad \frac{-a_{p,q-1/2}u_{p,q-1} + (a_{p,q-1/2} + a_{p,q+1/2})u_{p,q} - a_{p,q+1/2}u_{p,q+1}}{h_y^2} \\ &= f_{pq} \end{aligned} \tag{6.93}$$

which is the same as the finite difference equation for interior nodal points.

Convection term:

When a convection term $\mathbf{b} \cdot \nabla u$ appears in the differential equation, the same idea can be applied. For example, since $\mathbf{b} \cdot \nabla u = b_1 u_x + b_2 u_y$ in 2D,

$$\begin{aligned} \int_\Omega \mathbf{b} \cdot \nabla u \phi_{pq} d\mathbf{x} &= \int_{K_{pq}} (b_1 u_x + b_2 u_y) d\mathbf{x} \\ &\approx h_x h_y \Big(b_{1,pq} \frac{u_{p+1,q} - u_{p-1,q}}{2h_x} + b_{2,pq} \frac{u_{p,q+1} - u_{p,q-1}}{2h_y} \Big), \end{aligned} \tag{6.94}$$

which is again the same as the FD method.

Remark 6.18.

- The idea used in the above is the basis for the **finite volume method** defined on **control volumes** (CVs).
- Here we have put the nodal points at the center of the rectangular cells and used the cells for the CVs. Thus the method is sometimes called the **cell-centered finite difference method**.
- At interior points, the algebraic equations obtained from the FV method are equivalent to those of the second-order FD method (on rectangular meshes) or the linear FE method (on triangular meshes).
- Boundary conditions must be treated accurately. See Homework 6.3.
- When the nodal points are set on the corners of the cells, the CV should be determined such that it contains the nodal point in an appropriate way; the CVs are nonoverlapping and their union becomes the whole domain.

6.8. Average of The Diffusion Coefficient

Remark 6.19.

- The conormal flux $au_{\mathbf{n}}$ on a interface denotes the mass or energy movement through the interface.
- Thus it must be continuous (mass/energy conservation), on the interfaces of finite elements or control volumes. That is,

$$au_{\mathbf{n}_{pq}}(\mathbf{x}) = -au_{\mathbf{n}_{p+1,q}}(\mathbf{x}), \quad \mathbf{x} \in \partial K_{pq} \cap \partial K_{p+1,q} \tag{6.95}$$

- Such a physical consideration gives a way of approximating the diffusion coefficient a to get a more physical (and therefor more accurate) numerical solution.

Approximation of the diffusion coefficient

- Let a be locally constant, i.e., constant on each cell.
- Then conormal flux in (6.88) on $\partial K_{pq} \cap \partial K_{p+1,q}$ can be approximated as

$$au_{\mathbf{n}_{pq}}(\mathbf{x}) \approx a_{pq}\frac{u_e - u_{pq}}{h_x/2}, \quad \mathbf{x} \in \partial K_{pq} \cap \partial K_{p+1,q}, \tag{6.96}$$

  where u_e is introduced to represent the solution on the interface $\partial K_{pq} \cap \partial K_{p+1,q}$.
- From the other side of the interface, we have

$$au_{\mathbf{n}_{p+1,q}}(\mathbf{x}) \approx a_{p+1,q}\frac{u_e - u_{p+1,q}}{h_x/2}, \ \ \mathbf{x} \in \partial K_{pq} \cap \partial K_{p+1,q}. \tag{6.97}$$

- Here **the goal** is to find $\widetilde{a}$ such that

$$a_{pq}\frac{u_e - u_{pq}}{h_x/2} = a_{p+1,q}\frac{u_{p+1,q} - u_e}{h_x/2} = \widetilde{a}\frac{u_{p+1,q} - u_{pq}}{h_x}. \tag{6.98}$$

- It can be solved as

$$\widetilde{a} = \left[\frac{1}{2}\left(\frac{1}{a_{pq}} + \frac{1}{a_{p+1,q}}\right)\right]^{-1}, \tag{6.99}$$

  which is the harmonic average of a_{pq} and $a_{p+1,q}$.

6.9. Numerical Examples with Python

- A Python code is implemented for solving

$$\begin{aligned} -u_{xx} &= f, \quad x \in (0,1) \\ u &= g, \qquad x = 0,\, 1, \end{aligned} \tag{6.100}$$

  using high-order Galerkin FE methods.
- The exact solution is chosen as

$$u(x) = \sin(\pi x) \tag{6.101}$$

  so that the right-hand side becomes

$$f(x,y) = \pi^2 \sin(\pi x)$$

- For various number of grid points n_x and the order of basis functions k, the maximum errors are found as in the table below.

Table 6.1: The maximum error $\|u - u_h\|_\infty$.

	k			
n_x	1	2	3	4
2	0.234	0.00739	0.000428	1.67e-05
4	0.053(2.14)	0.000562(3.72)	1.45e-05(4.88)	3.37e-07(5.63)
8	0.013(2.03)	3.67e-05(3.94)	4.61e-07(4.98)	5.58e-09(5.92)
16	0.00322(2.01)	2.31e-06(3.99)	1.45e-08(4.99)	8.84e-11(5.98)

The numbers in parentheses denote convergence rates. Note that **superconvergence** is observed for $k \geq 2$.

The following shows the main routine `FEM_1D_High_Order.py`, the user parameter file `USER_PARS.py`, and the core functions for the construction of the stiffness matrix.

```
## FEM_1D_High_Order.py
##-- read USER_PARS and util ---------
from USER_PARS import *
from util_FEM_1D import *

level = 2
print_USER_PARS(level)
from fem_1d import *

#-----------------------------------
A = stiffness_mtx(level)
b = get_rhs(level)
dirichlet_BC(A)

ALU = mtx_banded_lu(A,level)
mtx_banded_lusol(ALU,b)

U = exact_sol(level)
print "L8-error = %.3g" %(max_difference(U,b))

## USER_PARS.py
##-----------------------
ax,bx = 0.,1.0;
nx = 20
poly_order = 3

## fem_1d.py
##-----------------------
def stiffness_mtx(level=0):
    A = np.ndarray((row,col),float)
    init_array(A)
    for e in range (nx):
        g0,g1 = e*kpoly,(e+1)*kpoly
```

```
        xl,xr = XG[e],XG[e+1]
        E = element_stiffness(xl,xr,kpoly)
        for i in range(kpoly+1):
            for j in range(kpoly+1):
                A[g0+i][kpoly+j-i] += E[i][j]
    return A

def element_stiffness(xl,xr,kpoly):
    m = kpoly+1
    E = np.ndarray((m,m),float)
    init_array(E)
    XL,WT = local_points_weights(xl,xr,kpoly)
    XT = get_XT(XL)
    for i in range(m):
        for j in range(m):
            for l in range(m):
                dphi_i_xl=eval_dphi(i,kpoly,XL[i],XL[l],XT)
                dphi_j_xl=eval_dphi(j,kpoly,XL[j],XL[l],XT)
                E[i][j]+=(dphi_i_xl*dphi_j_xl*WT[l])
    return E
```

6.10. Homework

1. Consider the model problem (6.9). Verify that the algebraic system from the linear Galerkin method is equivalent to that of finite difference method when the mesh is uniform, i.e.,
$$h = h_i, \quad i = 1, \cdots, M+1,$$

2. Prove (6.47) and (6.48). *Hint*: In each subinterval $I_j = [x_{j-1}, x_j]$, the difference between u and its linear interpolant can be expressed as follows: for $x \in I_j$,
$$u(x) - \pi_h u(x) = \frac{u''(\xi_j)}{2!}(x - x_{j-1})(x - x_j), \quad \text{for some } \xi_j \in I_j.$$
(See (1.10)on p.5.)

3. Let $\Omega = (0,1)^2$ and $\Gamma = \partial\Omega$ and consider
$$\begin{array}{rll} -\nabla \cdot (a(\mathbf{x})\nabla u) = f, & \mathbf{x} \in \Omega, \\ u = g_D, & \mathbf{x} \in \Gamma_D, \\ a u_{\mathbf{n}} = g_N, & \mathbf{x} \in \Gamma_N, \end{array} \tag{6.102}$$
where $\Gamma = \Gamma_D \cup \Gamma_N$ and Γ_D and Γ_N are distinct nonempty boundary portions corresponding to the Dirichlet and Neumann boundary conditions, respectively. Consider a FV method on a rectangular cells with cell-centered nodal points, as considered in Section 6.7. Design to suggest numerical methods for an effective treatment for each of the boundary conditions. (You may assume $g_D = g_N \equiv 0$, if you want.)

4. Consider the following 1D elliptic problem of general form
$$\begin{array}{l} -((1+x^2)u_x)_x + 5u_x = f, \quad x \in (0,1) \\ u_x(0) = g_N, \;\; u(1) = g_D \end{array} \tag{6.103}$$
Choose the exact solution as in (6.101):
$$u(x) = \sin(\pi x)$$
and correspondingly the right side f and the boundary data, g_N and g_D.
   (a) Formulate the Galerkin method for (6.103).
   (b) Modify the Python code in §6.9 to solve the above problem.
   (c) Carry out an error analysis as in Table 6.1.

5. Assume that $v(x) \in C^1[a,b]$ and $v(a) = 0$. Prove that the one-dimensional Poincaré inequality
$$\|v\|_0 \le \frac{b-a}{\sqrt{2}}\|v'\|_0. \tag{6.104}$$
Hint: You may begin with
$$v(x) = v(a) + \int_a^x v'(t)dt = \int_a^x v'(t)dt.$$

Thus, by the Cauchy-Schwarz inequality

$$\begin{aligned} |v(x)| &\leq \int_a^x |v'|dt \leq \Big(\int_a^x dt\Big)^{1/2}\Big(\int_a^x (v')^2 dt\Big)^{1/2} \\ &\leq \sqrt{x-a}\,\|v'\|_0 \end{aligned} \tag{6.105}$$

Now, square the inequality and then integrate over the interval.

6. Use the arguments in the proof of Homework 6.5 to prove the Poincaré inequality (6.7) when $\Omega = (0,1)^2$:

$$\int_\Omega u^2 d\mathbf{x} \leq C \int_\Omega |\nabla u|^2 d\mathbf{x}, \;\; \forall v \in H_0^1(\Omega), \tag{6.106}$$

for some $C > 0$. Try to determine the constant C as small as possible. $\Big($Note that $\int_\Omega f(\mathbf{x})\, d\mathbf{x} = \int_0^1 \int_0^1 f(x,y)\, dxdy = \int_0^1 \int_0^1 f(x,y)\, dydx.\Big)$

CHAPTER 7

FD Methods for Hyperbolic Equations

This chapter considers finite difference methods for hyperbolic PDEs. We begin with numerical methods for the linear scalar wave equation. Then, numerical methods for conservation laws are treated along with nonlinear stability. A Python code is included for the Lax-Wendroff scheme to solve the one-way wave equation.

Contents of Chapter 7

7.1. Introduction . . . 210
7.2. Basic Difference Schemes . . . 213
7.3. Conservation Laws . . . 227
7.4. Shocks and Rarefaction . . . 234
7.5. Numerical Methods . . . 238
7.6. Nonlinear Stability . . . 251
7.7. Numerical Examples with Python . . . 260
7.8. Homework . . . 262

7.1. Introduction

Consider the initial value problem

$$\begin{aligned} &\mathbf{u}_t + A\,\mathbf{u}_x = 0 \\ &\mathbf{u}|_{t=0} = \mathbf{u}_0(x), \end{aligned} \tag{7.1}$$

where $A = [a_{ij}] \in \mathbb{R}^{m\times m}$ and $\mathbf{u}$ is a vector function of m components, $m \geq 1$.

- The problem (7.1) is well-posed if and only if all eigenvalues of A are real and there is a complete set of eigenvectors [27].
- Such a system is called **(strongly) hyperbolic**.
- We will restrict our discussions to such hyperbolic problems.

Let $\{\phi_1, \cdots, \phi_m\}$ be the complete set of eigenvectors corresponding to the eigenvalues $\{\lambda_1, \cdots, \lambda_m\}$. Define a matrix

$$S = [\phi_1, \cdots, \phi_m], \quad \Gamma = \mathbf{diag}(\lambda_1, \cdots, \lambda_m).$$

Then, from linear algebra theory, we obtain

$$A = S\Gamma S^{-1}. \tag{7.2}$$

Apply S^{-1} to (7.1) to have

$$\begin{array}{l} S^{-1}\mathbf{u}_t + \Gamma\, S^{-1}\,\mathbf{u}_x \;=\; 0 \\ S^{-1}\mathbf{u}|_{t=0} \;=\; S^{-1}\mathbf{u}_0(x). \end{array} \tag{7.3}$$

Let $\widetilde{\mathbf{u}} = S^{-1}\mathbf{u}$. Then, (7.3) is reduced to the following m scalar equations

$$\begin{array}{l} \widetilde{u}_{i,t} + \lambda_i\, \widetilde{u}_{i,x} \;=\; 0, \quad i = 1, \cdots, m, \\ \widetilde{u}_i|_{t=0} \;=\; \widetilde{u}_{i,0}(x). \end{array} \tag{7.4}$$

Hence the chapter begins with discussions focusing on the scalar equation:

$$\begin{aligned} &u_t + a u_x = 0, \qquad (x,t) \in \Omega \times J, \\ &u(x,0) = u_0(x), \quad x \in \Omega, \;\; t = 0, \end{aligned} \tag{7.5}$$

where $\Omega = (a_x, b_x) \subset \mathbb{R}$ and $J = (0, T]$, $T > 0$, the time interval. Here the boundary condition is ignored for simplicity. (Or, we may assume $\Omega = \mathbb{R}$.)

When a is a constant, (7.5) has the exact solution

$$u(x,t) = u_0(x - at). \tag{7.6}$$

7.2. Basic Difference Schemes

We begin with our discussion of finite difference (FD) schemes for (7.5) by defining grid points in the (x, t) plane.

Let Δx and Δt be the spatial and temporal grid sizes, respectively; then the grid will be the points

$$(x_m, t^n) = (m\Delta x, n\Delta t)$$

for integers m and $n \geq 0$. For a function v defined either on the grid or for continuously varying (x, t), we write v_m^n for the value of v at (x_m, t^n), i.e.,

$$v_m^n = v(x_m, t^n).$$

Let

$$\mathcal{S}^n := \Omega \times (t^{n-1}, t^n]$$

be the nth *space-time* slice. Suppose that the computation has been performed for $u^j = \{u_m^j\}$, $0 \leq j \leq n-1$. Then, the task is to compute u^n by integrating the equation on the space-time slice $\mathcal{S}^n$, utilizing FD schemes.

The following presents examples of the forward-time (*explicit*) schemes for (7.5):

$$
\begin{aligned}
&\text{(a)} \quad \frac{v_m^n - v_m^{n-1}}{\Delta t} + a\frac{v_m^{n-1} - v_{m-1}^{n-1}}{\Delta x} = 0, \\
&\text{(b)} \quad \frac{v_m^n - v_m^{n-1}}{\Delta t} + a\frac{v_{m+1}^{n-1} - v_m^{n-1}}{\Delta x} = 0, \\
&\text{(c)} \quad \frac{v_m^n - v_m^{n-1}}{\Delta t} + a\frac{v_{m+1}^{n-1} - v_{m-1}^{n-1}}{2\Delta x} = 0, \\
&\text{(d)} \quad \frac{v_m^n - v_m^{n-2}}{2\Delta t} + a\frac{v_{m+1}^{n-1} - v_{m-1}^{n-1}}{2\Delta x} = 0, \qquad \text{(leapfrog)} \\
&\text{(e)} \quad \frac{v_m^n - \frac{v_{m+1}^{n-1} + v_{m-1}^{n-1}}{2}}{\Delta t} + a\frac{v_{m+1}^{n-1} - v_{m-1}^{n-1}}{2\Delta x} = 0. \quad \text{(Lax-Friedrichs)}
\end{aligned}
\tag{7.7}
$$

These explicit schemes shall be exemplified in describing properties of numerical methods.

7.2.1. Consistency

The bottom line for accurate numerical methods is that the discretization becomes exact as the grid spacing tends to zero, which is the basis of *consistency*. Recall the definition of consistency.

Definition 7.1. *Given a PDE $Pu = f$ and a FD scheme $P_{\Delta x,\Delta t}u = f$, the FD scheme is said to be consistent with the PDE if for every smooth function $\phi(x,t)$*

$$P\phi - P_{\Delta x,\Delta t}\phi \to 0 \quad \textit{as} \quad (\Delta x, \Delta t) \to 0,$$

with the convergence being pointwise at each grid point.

Not all numerical methods based on Taylor series expansions are consistent.

Example 7.2. *The forward-time forward-space scheme is consistent.*

Proof. For the one-way wave equation (7.5),

$$P\phi \equiv \left(\frac{\partial}{\partial t} + a\frac{\partial}{\partial x}\right)\phi = \phi_t + a\phi_x.$$

For the forward-time forward-space scheme (7.7b),

$$P_{\Delta x,\Delta t}\phi = \frac{\phi_m^n - \phi_m^{n-1}}{\Delta t} + a\frac{\phi_{m+1}^{n-1} - \phi_m^{n-1}}{\Delta x}.$$

To find the truncation error of the numerical scheme, we begin with the Taylor series in x and t about (x_m, t^n):

$$\phi_m^n = \phi_m^{n-1} + \Delta t\, \phi_t(x_m, t^{n-1}) + \frac{\Delta t^2}{2}\phi_{tt}(x_m, t^{n-1}) + \mathcal{O}(\Delta t^3),$$
$$\phi_{m+1}^{n-1} = \phi_m^{n-1} + \Delta x\, \phi_x(x_m, t^{n-1}) + \frac{\Delta x^2}{2}\phi_{xx}(x_m, t^{n-1}) + \mathcal{O}(\Delta x^3).$$

With some algebra, one can obtain

$$P_{\Delta x,\Delta t}\phi = \phi_t + a\phi_x + \frac{\Delta t}{2}\phi_{tt} + a\frac{\Delta x}{2}\phi_{xx} + \mathcal{O}(\Delta x^2 + \Delta t^2).$$

Thus, as $(\Delta x, \Delta t) \to 0$,

$$P\phi - P_{\Delta x,\Delta t}\phi = -\frac{\Delta t}{2}\phi_{tt} - a\frac{\Delta x}{2}\phi_{xx} + \mathcal{O}(\Delta x^2 + \Delta t^2) \to 0.$$

Therefore, the scheme is consistent. □

7.2.2. Convergence

A numerical method is said to be *convergent* if the solution of the FD scheme tends to the exact solution of the PDE as the grid spacing tends to zero. We redefine convergence in a formal way as follows:

Definition 7.3. *A FD scheme approximating a PDE is said to be convergent if*

$$u(x,t) - u_m^n \to 0 \quad \textit{as} \quad (x_m, t^n) \to (x,t) \ \textit{as} \ (\Delta x, \Delta t) \to 0,$$

where $u(x,t)$ is the exact solution of PDE and u_m^n denotes the the solution of the FD scheme.

Consistency implies that the truncation error

$$(Pu - P_{\Delta x, \Delta t} u) \to 0$$

as Δx and Δt approach zero. So consistency is certainly necessary for convergence. But as the following example shows, a numerical scheme may be consistent but not convergent.

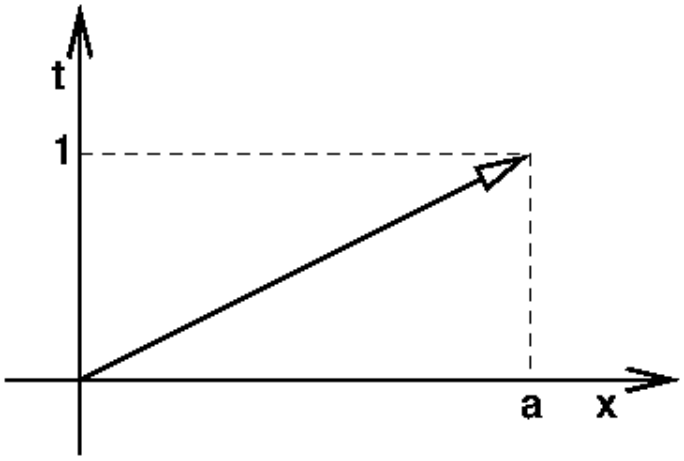


Figure 7.1: The characteristic curve passing the origin of the xt-plane.

Example 7.4. *The forward-time forward-space scheme for* (7.5) *is not convergent, when* $a > 0$.

Proof. The scheme (7.7b) is consistent from Example 7.2. The problem (7.5) has the exact solution

$$u(x,t) = u_0(x - at),$$

a shift of u_0 by at. The lines having the slope $1/a$ in the xt-plane become characteristics of the problem; when $a > 0$, the characteristic curve passing the origin is shown in Figure 7.1.

On the other hand, the scheme (7.7b) can be rewritten as

$$v_m^n = v_m^{n-1} - a\lambda(v_{m+1}^{n-1} - v_m^{n-1}) = (1 + a\lambda)v_m^{n-1} - a\lambda v_{m+1}^{n-1}, \tag{7.8}$$

where $\lambda = \Delta t/\Delta x$. Let the initial data be given

$$u_0(x) = \begin{cases} 1, & \text{if } x \le 0, \\ 0, & \text{else.} \end{cases}$$

Since it is natural for the scheme to take the initial data

$$v_m^0 = \begin{cases} 1, & \text{if } x_m \le 0, \\ 0, & \text{else,} \end{cases}$$

it follows from (7.8) that

$$v_m^n \equiv 0 \quad \forall\, m > 0,\ \ n \ge 0.$$

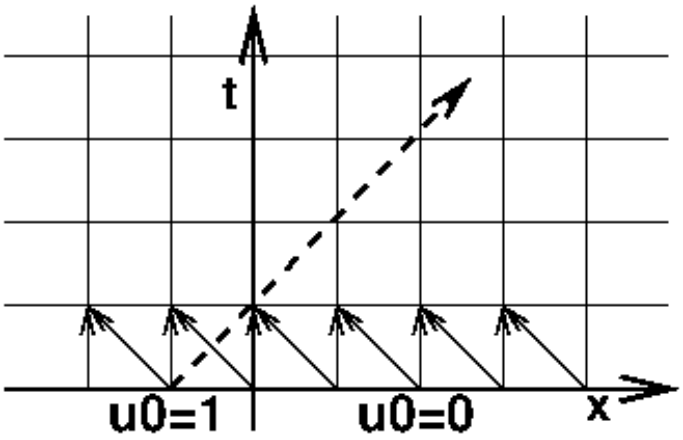


Figure 7.2: The forward-time forward-space scheme for $u_t + au_x = 0$, $a > 0$.

See Figure 7.2. The above holds for any choices of Δx and Δt. Therefore, v_m^n cannot converge to the exact solution $u(x,t)$ in (7.6). □

Showing that a given consistent scheme is convergent is not easy in general, if attempted in a direct manner as in Homework 7.1. However, there is a related concept, stability, that is easier to check.

7.2.3. Stability

Example 7.4 shows that consistency is not enough for a numerical method to guarantee convergence of its solution to the exact solution. In order for a consistent numerical scheme to be convergent, the required property is stability. Recall the L^2-norm of grid function v:

$$\|v\|_{\Delta x} = \left(\Delta x \sum_{m=-\infty}^{\infty} |v_m|^2 \right)^{1/2}.$$

Definition 7.5. *A FD scheme $P_{\Delta x, \Delta t} v = 0$ for a homogeneous PDE $Pu = 0$ is stable if for any positive T, there is a constant C_T such that*

$$\|v^n\|_{\Delta x} \leq C_T \sum_{j=0}^{J} \|v^j\|_{\Delta x}, \tag{7.9}$$

for $0 \leq t^n \leq T$ and for Δx and Δt sufficiently small. Here J is chosen to incorporate the data initialized on the first $J+1$ levels.

Example 7.6. *The schemes* (7.7a) *and* (7.7b) *can be written of the form*

$$v_m^n = \alpha v_m^{n-1} + \beta v_{m\mp1}^{n-1}.$$

Then they are stable if $|\alpha| + |\beta| \le 1$.

Proof. Indeed, for the scheme (7.7a),

$$\begin{aligned}
\sum_{m=-\infty}^{\infty} |v_m^n|^2 &= \sum_{m=-\infty}^{\infty} |\alpha v_m^{n-1} + \beta v_{m-1}^{n-1}|^2 \\
&\le \sum_{m=-\infty}^{\infty} |\alpha v_m^{n-1}|^2 + 2|\alpha\beta v_m^{n-1} v_{m-1}^{n-1}| + |\beta v_{m-1}^{n-1}|^2 \\
&\le \sum_{m=-\infty}^{\infty} |\alpha|^2 |v_m^{n-1}|^2 + |\alpha||\beta|(|v_m^{n-1}|^2 + |v_{m-1}^{n-1}|^2) + |\beta|^2 |v_{m-1}^{n-1}|^2 \\
&= \sum_{m=-\infty}^{\infty} (|\alpha| + |\beta|)^2 |v_m^{n-1}|^2.
\end{aligned}$$

Thus the scheme is stable if $|\alpha| + |\beta| = |1 - a\lambda| + |a\lambda| \le 1$, where $\lambda = \Delta t/\Delta x$. Therefore, a sufficient condition for stability of (7.7a) is $0 \le a\lambda \le 1$. The analysis is similar for (7.7b); it is stable if $-1 \le a\lambda \le 0$. □

The stability inequality (7.9) can be easily satisfied when

$$\|v^n\|_{\Delta x} \le (1 + C\Delta t)\|v^{n-1}\|_{\Delta x}, \tag{7.10}$$

for some $C \ge 0$ independent on Δt.

Theorem 7.7. (Lax-Richtmyer Equivalence Theorem). *Given a well-posed linear initial value problem and its FD approximation that satisfies the consistency condition, stability is the necessary and sufficient condition for convergence.*

The above theorem is very useful and important. Providing convergence is difficult for most problems. However, the determination of consistency of a scheme is quite easy as shown in §7.2.1, and determining stability is also easier than showing convergence. Here we introduce the von Neumann analysis of stability of FD schemes, which allows one to analyze stability much simpler than a direct verification of (7.9).

The von Neumann analysis

A simple procedure of the von Neumann analysis reads

- *Replace* v_m^n *by* $g^n e^{im\vartheta}$ *for each value of* m *and* n.
- *Find conditions on coefficients and grid spacings which would satisfy* $|g| \leq 1 + C\Delta t$, *for some* $C \geq 0$.

The Courant-Friedrichs-Lewy (CFL) condition

The von Neumann analysis is not easy to utilize for rather general problems, in particular, for **nonlinear problems**. In computational fluid dynamics (CFD), a more popular concept is the so-called *CFL condition*.

Theorem 7.8. *Given an explicit scheme for* $u_t + au_x = 0$ *of the form*

$$v_m^n = \alpha v_{m-1}^{n-1} + \beta v_m^{n-1} + \gamma v_{m+1}^{n-1}$$

with $\lambda = \Delta t/\Delta x$ *held constant, a necessary condition for stability is the Courant-Friedrichs-Lewy* (CFL) *condition*

$$|a\lambda| \le 1.$$

Proof. Let $\Delta t = 1/n$, for some $n \ge 1$. Then the physical domain of dependence for the exact solution at the point $(x,t) = (0,1)$ must be $(\pm a, 0)$, i.e.,

$$u(0,1) = u_0(\pm a).$$

On the other hand, it follows from the FD scheme that the numerical solution v_0^n depends on v_m^0, $|m| \le n$. Since

$$m\Delta x = m\Delta t/\lambda \le n\Delta t/\lambda = 1/\lambda,$$

we can see that the numerical solution at $(0,1)$, v_0^n, depends on x for $|x| \le 1/\lambda$. Suppose $|a\lambda| > 1$. Then we have $|a| > 1/\lambda$. So v_0^n depends on x for

$$|x| \le 1/\lambda < |a|.$$

Thus v_0^n cannot converge to the exact value $u(0,1) = u_0(\pm a)$ as $\Delta x \to 0$ with $\lambda = \Delta t/\Delta x$ keeping constant. This proves the theorem. □

One can see from the above theorem and proof that

stability requires the numerical domain of dependence contain the physical domain of dependence.

This physical observation is very useful for stability analysis for certain nonlinear problems [40].

7.2.4. Accuracy

We define the order of accuracy for numerical schemes for PDEs.

Definition 7.9. (Order of accuracy). Let $P_{\Delta x,\Delta t}u = R_{\Delta x,\Delta t}f$ be a numerical scheme for $Pu = f$. Assume that for every smooth function ϕ,

$$P_{\Delta x,\Delta t}\phi = R_{\Delta x,\Delta t}(P\phi) + \mathcal{O}(\Delta x^p) + \mathcal{O}(\Delta t^q).$$

Then, the scheme is said to have *the p-th order accuracy in space and the q-th order accuracy in time*, and denoted by the "accuracy order (p, q) in space-time".

For example, the forward-time forward-space, forward-time central-space, and leapfrog schemes for (7.5) have the accuracy orders $(1, 1)$, $(2, 1)$, and $(2, 2)$ in space-time, respectively.

Crank-Nicolson (CN) scheme: Consider the one-way wave equation with a source term

$$u_t + au_x = f. \tag{7.11}$$

The scheme is based on central differences about $(x, t^{n-1/2})$, where $t^{n-1/2} = (t^{n-1} + t^n)/2$. Since

$$\begin{aligned}
u_t(x_m, t^{n-1/2}) &= \frac{u_m^n - u_m^{n-1}}{\Delta t} + \mathcal{O}(\Delta t^2), \\
u_x(x_m, t^{n-1/2}) &= \frac{u_x(x_m, t^n) + u_x(x_m, t^{n-1})}{2} + \mathcal{O}(\Delta t^2) \\
&= \frac{1}{2}\left[\frac{u_{m+1}^n - u_{m-1}^n}{2\Delta x} + \frac{u_{m+1}^{n-1} - u_{m-1}^{n-1}}{2\Delta x}\right] + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2), \\
f(x_m, t^{n-1/2}) &= \frac{f_m^n + f_m^{n-1}}{2} + \mathcal{O}(\Delta t^2),
\end{aligned}$$

we obtain the CN scheme

$$\frac{v_m^n - v_m^{n-1}}{\Delta t} + \frac{a}{2}\left[\frac{v_{m+1}^n - v_{m-1}^n}{2\Delta x} + \frac{v_{m+1}^{n-1} - v_{m-1}^{n-1}}{2\Delta x}\right] = \frac{f_m^n + f_m^{n-1}}{2}, \tag{7.12}$$

where the truncation error is

$$\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2).$$

Thus the CN scheme has the accuracy order $(2, 2)$.

It follows from the von Neumann analysis presented in §7.2.3 that the amplification factor for the CN scheme is

$$g(\vartheta) = \frac{1 - i\frac{a\lambda}{2}\sin\vartheta}{1 + i\frac{a\lambda}{2}\sin\vartheta}, \quad \lambda = \frac{\Delta t}{\Delta x}.$$

Thus its magnitude is identically one and therefore the CN scheme is stable for every choice of Δx and Δt (*unconditional stability*).

Note: The numerical solution of the CN method (7.12) may involve oscillations when the initial data is nonsmooth.

For a wide range of PDEs, the CN scheme is unconditionally stable and of a second-order accuracy in both space and time. These two advantageous properties have made the scheme quite popular.

7.3. Conservation Laws

The conservation laws in one-dimensional (1D) space have the form

$$\boxed{\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = 0.} \tag{7.13}$$

Here

$$u : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^m$$

and $f : \mathbb{R}^m \to \mathbb{R}^m$ is called the *flux function*. For simplicity, we may consider the pure initial value problem, or *Cauchy problem*, in which (7.13) holds for $-\infty < x < \infty$ and $t \geq 0$. In this case we must specify initial conditions only

$$u(x,0) = u_0(x), \quad -\infty < x < \infty. \tag{7.14}$$

We assume that the system (7.13) is *hyperbolic*. That is, the Jacobian matrix $f'(u)$ of the flux function is

- of real eigenvalues, and
- diagonalizable, i.e., there is a complete set of m linearly independent eigenvectors.

In 2D, a system of conservation laws can be written as

$$u_t + f(u)_x + g(u)_y = 0, \tag{7.15}$$

where

$$u : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^m, \quad f, g : \mathbb{R}^m \to \mathbb{R}^m.$$

7.3.1. Euler equations of gas dynamics

Consider “a tube" where properties of the gas such as density and velocity are assumed to be constant across each cross section of the tube. Let $\rho(x,t)$ and $v(x,t)$ be respectively the density and the velocity at point x and time t. Then

$$\textbf{mass in } [x_1, x_2] \text{ at time } t = \int_{x_1}^{x_2} \rho(x,t)dx.$$

Assume that the walls of the tube are impermeable and that mass is neither created nor destroyed. Then the mass in a section $[x_1, x_2]$ can change only because of gas flowing across the end points x_1 and x_2. The rate of flow, or *flux* of gas at (x, t) is given by

$$\textbf{mass flux} \text{ at } (x,t) = \rho(x,t)\, v(x,t).$$

Thus, the change rate of mass in $[x_1, x_2]$ is

$$\boxed{\frac{d}{dt}\int_{x_1}^{x_2} \rho(x,t)dx = \rho(x_1,t)\, v(x_1,t) - \rho(x_2,t)\, v(x_2,t),} \tag{7.16}$$

which is one *integral form* of conservation law.

Integrate (7.16) in time from t_1 to t_2 to have

$$\boxed{\begin{aligned}\int_{x_1}^{x_2} \rho(x,t_2)dx \;&=\; \int_{x_1}^{x_2} \rho(x,t_1)dx \\ &\quad + \int_{t_1}^{t_2} \rho(x_1,t)\, v(x_1,t)dt - \int_{t_1}^{t_2} \rho(x_2,t)\, v(x_2,t)dt.\end{aligned}} \tag{7.17}$$

This is another *integral form* of conservation law.

Geometric interpretation for (7.17):

Derivation of differential form: Now, assume ρ and v are differentiable. Since

$$\rho(x,t_2) - \rho(x,t_1) = \int_{t_1}^{t_2} \frac{\partial}{\partial t}\rho(x,t)\,dt,$$
$$\rho(x_2,t)\,v(x_2,t) - \rho(x_1,t)\,v(x_1,t) = \int_{x_1}^{x_2} \frac{\partial}{\partial x}(\rho(x,t)\,v(x,t))\,dx,$$

the equation (7.17) reads

$$\int_{t_1}^{t_2}\int_{x_1}^{x_2} \Big[\frac{\partial}{\partial t}\rho(x,t) + \frac{\partial}{\partial x}(\rho(x,t)\,v(x,t))\Big]\,dx\,dt = 0. \tag{7.18}$$

Since this must hold for any section $[x_1,x_2]$ and for any time interval $[t_1,t_2]$, the integrand in (7.18) must be identically zero, i.e.,

$$\boxed{\rho_t + (\rho v)_x = 0. \quad \text{(conservation of mass)}} \tag{7.19}$$

Euler equations of gas dynamics:

$$\boxed{\begin{array}{ll} \rho_t + (\rho v)_x = 0, & \text{(conservation of mass)} \\ (\rho v)_t + (\rho v^2 + p)_x = 0, & \text{(conservation of momentum)} \\ E_t + (v(E+p))_x = 0. & \text{(conservation of energy)} \end{array}} \tag{7.20}$$

The rule of thumb (in the derivation of conservation laws) is that

- For any quantity z which is advected with the flow will have a contribution to the flux of the form zv.
- Besides advection, there are forces on the fluid that cause acceleration due to Newton's laws. Since we assume there is no outside forces, the only force is due to variations in the fluid itself; it is proportional to the pressure gradient for momentum and proportional to the gradient of vp for energy.

The pressure variable can be replaced by additional equations of physics, called the *state equations*. For gases,

$$
\begin{aligned}
E &= \frac{1}{2}\rho v^2 + \rho e, && \text{(total energy)}\\
p &= R\,\rho\,T, && \text{(pressure: ideal gas law)}\\
e &= c_v T, && \text{(specific internal energy: polytropic gas)}\\
h &= e + p/\rho = c_p T, && \text{(enthalpy: polytropic gas)}\\
\gamma &= c_p/c_v, && \text{(ratio of specific heat)}\\
R &= c_p - c_v. && \text{(polytropic gas)}
\end{aligned}
$$

The *polytropic gas* is such that the internal energy is proportional to the temperature, so the coefficients c_v and c_p are constants, called respectively the *specific heat at constant volume* and the *specific heat at constant pressure*. (In general, “specific" means “per unit mass".)

The equation of state for a polytropic gas: Note that $T = p/(R\rho)$ so that

$$e = c_v T = \frac{c_v}{R}\frac{p}{\rho} = \frac{c_v}{c_p - c_v}\frac{p}{\rho} = \frac{1}{\gamma - 1}\frac{p}{\rho}.$$

Thus the equation of state for a polytropic gas is

$$\boxed{E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2.} \tag{7.21}$$

Isothermal flow: Assume the temperature is constant through the tube. Then, from the ideal gas law,

$$p = R\,\rho\,T = a^2\rho,$$

where $a = \sqrt{RT}$ is the sound speed. Thus the *isothermal equations* read

$$\begin{bmatrix} \rho \\ \rho v \end{bmatrix}_t + \begin{bmatrix} \rho v \\ \rho v^2 + a^2\rho \end{bmatrix}_x = 0. \tag{7.22}$$

7.4. Shocks and Rarefaction

7.4.1. Characteristics

Consider the linear advection equation

$$\begin{aligned} &u_t + a u_x = 0, \\ &u(x,0) = u_0(x). \end{aligned} \tag{7.23}$$

The exact solution is simply

$$u(x,t) = u_0(x - at), \quad t \geq 0.$$

The solution is constant along each ray $x - at = x_0$. Such rays are known as the *characteristics* of the equation.

Note that the characteristics are curves in the x-t plane satisfying the ODE $x'(t) = a$, $x(0) = x_0$. Let us differentiate $u(x,t)$ along one of these curves to find the change rate of the solution along the characteristics:

$$\frac{d}{dt}u(x,t) = \frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}u(x,t)x' = u_t + a u_x = 0,$$

which confirms that u is constant along the characteristics.

There is a fundamental property of *linear* hyperbolic equations: singularities propagate only along characteristics.

Nonsmooth data: We consider the so-called *vanishing-viscosity approach*. Let u^ε be the solution of

$$u_t + au_x = \varepsilon u_{xx}. \tag{7.24}$$

Then u^ε is smooth for $t > 0$ even if u_0 is not smooth, because it is the solution of a parabolic equation.

Note that (7.24) simplifies if we make a change of variables to follow the characteristics:

$$v^\varepsilon(x,t) = u^\varepsilon(x + at, t).$$

Then v^ε satisfies the heat equation

$$v_t^\varepsilon(x,t) = \varepsilon v_{xx}^\varepsilon(x,t).$$

Thus, after solving the heat equation, we can compute $u^\varepsilon(x,t) = v^\varepsilon(x - at, t)$ *explicitly*. It is easy to verify that the *vanishing-viscosity* solution is equal to $u_0(x - at)$:

$$\lim_{\varepsilon \to 0} u^\varepsilon(x,t) = u(x,t) = u_0(x - at).$$

7.4.2. Weak solutions

A natural way to define a generalized solution of the inviscid equation that does not require differentiability is to go back to the integral form of the conservation law. We say $u(x,t)$ is a *generalized solution* if (7.17) is satisfied for all x_1, x_2, t_1, and t_2.

There is another approach that results in a different integral formulation that is often more convenient to work with.

Let $\phi \in C_0^1(\mathbb{R} \times \mathbb{R}^+)$. Multiply $u_t + f(u)_x = 0$ by ϕ and integrate over space and time to have

$$\int_0^\infty \int_{-\infty}^\infty [\phi u_t + \phi f(u)_x]\, dx\, dt = 0.$$

Using integration by parts gives

$$\int_0^\infty \int_{-\infty}^\infty [\phi_t u + \phi_x f(u)]\, dx\, dt = -\int_{-\infty}^\infty \phi(x,0)u(x,0)\, dx. \tag{7.25}$$

Definition 7.10. *The function $u(x,t)$ is called a weak solution of $u_t + f(u)_x = 0$ if (7.25) holds for all $\phi \in C_0^1(\mathbb{R} \times \mathbb{R}^+)$.*

Known facts:

- Any weak solution satisfies the original integral conservation law.
- The vanishing-viscosity generalized solution is a weak solution.
- For nonlinear problems, weak solutions are often not unique, and therefore an additional problem is often considered to identify which weak solution is the physically correct vanishing-viscosity solution.
- There are other conditions to avoid working with the viscous equation directly. They are usually called the *entropy conditions*. Thus the vanishing-viscosity solution is also called the entropy solution.

7.5. Numerical Methods

7.5.1. Modified equations

In this subsection, we briefly review accuracy and stability for the Riemann problem of the linear advection equation:

$$\begin{aligned} &u_t + a u_x = 0, \quad x \in \mathbb{R}, \ t \geq 0, \\ &u_0(x) = \begin{cases} 1, & x < 0, \\ 0, & x > 0. \end{cases} \end{aligned} \tag{7.26}$$

The exact solution is given

$$u(x,t) = u_0(x - at). \tag{7.27}$$

Consider the following numerical schemes:

$$\begin{aligned} &\frac{U_j^{n+1} - U_j^n}{k} + a\frac{U_j^n - U_{j-1}^n}{h} = 0, && \text{(explicit one-sided)} \\ &\frac{U_j^{n+1} - \frac{U_{j+1}^n + U_{j-1}^n}{2}}{k} + a\frac{U_{j+1}^n - U_{j-1}^n}{2h} = 0, && \text{(Lax-Friedrichs)} \\ &\frac{U_j^{n+1} - U_j^n}{k} + a\frac{U_{j+1}^n - U_{j-1}^n}{2h} \\ &\qquad -\frac{k}{2}a^2\frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2} = 0. && \text{(Lax-Wendroff)} \end{aligned} \tag{7.28}$$

Lax-Wendroff scheme: Note that

$$u_t(x_j, t^n) = \frac{U_j^{n+1} - U_j^n}{k} - \frac{k}{2}u_{tt} - \frac{k^2}{6}u_{ttt} - \cdots .$$

Since

$$u_t = -au_x,$$

we have

$$\begin{aligned} u_{tt} = (u_t)_t = (-au_x)_t = -au_{xt} = -au_{tx} \\ = -a(u_t)_x = -a(-au_x)_x = a^2u_{xx} \end{aligned}$$

Therefore, the Lax-Wendroff scheme can be obtained by taking care of $u_{tt} = a^2u_{xx}$ by the central scheme; its truncation error is

$$\begin{aligned} -\frac{k^2}{6}u_{ttt} - a\frac{h^2}{6}u_{xxx} + \cdots &= \frac{k^2}{6}a^3u_{xxx} - a\frac{h^2}{6}u_{xxx} + \cdots \\ &= \frac{h^2}{6}a\Big(\frac{k^2}{h^2}a^2 - 1\Big)u_{xxx} + \cdots \end{aligned}$$

Thus, when h and k are sufficiently small, solving (7.26) by the Lax-Wendroff scheme is equivalent to solving the following equation exactly:

$$u_t + au_x = \frac{h^2}{6}a\Big(\frac{k^2}{h^2}a^2 - 1\Big)u_{xxx}. \tag{7.29}$$

Equation (7.29) is called the *modified equation* of (7.26) for the Lax-Wendroff scheme. By analyzing (7.29) in PDE sense, one can understand the Lax-Wendroff scheme.

Finite difference equation was introduced in the first place because it is easier to solve than a PDE; on the other hand, it is often easier to predict qualitative *behavior of a PDE than difference equations.*

Dispersion analysis: Equation (7.29) is a *dispersive equation* of the form

$$u_t + au_x = \mu u_{xxx}. \tag{7.30}$$

To look at a Fourier series solution to this equation, take $u(x,t)$ as

$$u(x,t) = \int_{-\infty}^{\infty} \widehat{u}(\xi,t) e^{i\xi x} d\xi,$$

where ξ is the *wave number*. Here the purpose is to see that the Fourier components with different wave number ξ propagate at different speeds (dispersion).

Due to linearity, it suffices to consider each wave number in isolation, so suppose that we look for solution of (7.30) of the form

$$u(x,t) = e^{i(\xi x - ct)}, \tag{7.31}$$

where $c = c(\xi)$ is called the *frequency*. Plugging this into (7.30) gives

$$c(\xi) = a\xi + \mu\,\xi^3. \tag{7.32}$$

This expression is called the *dispersion relation* for (7.30).

Define

$$c_p(\xi) = c(\xi)/\xi, \quad \textbf{(\textit{phase velocity})}$$
$$c_g(\xi) = c'(\xi). \quad \textbf{(\textit{group velocity})}$$

The phase velocity is the speed of wave peaks or in single frequency, while the group velocity is the speed of energy in wavetrain.

Then, for the modified equation of Lax-Friedrichs scheme in (7.29), we have

$$c_p = a + \mu\,\xi^2, \quad c_g = a + 3\mu\,\xi^2. \tag{7.33}$$

Recall that the CFL condition reads

$$|a\lambda| = |ak/h| \le 1.$$

Thus, when the Lax-Friedrichs scheme is stable, the coefficient μ for (7.29) must be nonpositive, i.e.,

$$\mu = \frac{h^2}{6} a\Big(\frac{k^2}{h^2}a^2 - 1\Big) \le 0, \tag{7.34}$$

which implies from (7.33) that both the phase velocity and the group velocity are *smaller* than the actual velocity a.

Remarks:

- For the step function in (7.26), the Fourier spectrum decays only as

$$\widehat{u_0}(\xi) = \mathcal{O}(1/\xi), \quad \text{as } |\xi| \to \infty.$$

  (For smooth solutions, its Fourier spectrum decays exponentially.)
- Thus for the Lax-Wendroff scheme, dispersion becomes visible near

$$x = c_g t.$$

  (although the scheme satisfies the stability condition.)
- The numerical solution is oscillatory in the upstream (behind).

Beam-Warming scheme: This method is one-sided second-order venison of the Lax-Wendroff scheme:

$$\begin{aligned}\frac{U_j^{n+1}-U_j^n}{k}+a\frac{3U_j^n-4U_{j-1}^n+U_{j-2}^n}{2h}\\ -\frac{k}{2}a^2\frac{U_j^n-2U_{j-1}^n+U_{j-2}^n}{h^2}=0.\end{aligned} \tag{7.35}$$

(Beam-Warming)

Then the associated modified equation reads

$$u_t+au_x=\mu u_{xxx},\quad \mu=\frac{h^2}{6}a\Big(2-\frac{3k}{h}a+\frac{k^2}{h^2}a^2\Big). \tag{7.36}$$

Remarks:

- Since $\mu>0$ for sufficiently small k, the group velocity will be larger than the actual speed a; there must be oscillation propagating faster than the shock speed.
- Here the point is that *a upwind modification is not sufficient enough to cure oscillation*.

Upwind (one-sided) scheme: For the explicit one-sided scheme in (7.28), one can find its modified equation as

$$u_t + au_x = \varepsilon\, u_{xx}, \quad \varepsilon = \frac{1}{2}ha\Big(1 - \frac{k}{h}a\Big). \tag{7.37}$$

Note that the stability requires $\varepsilon \geq 0$. This is a heat equation; the solution must be diffusive.

When the dispersion analysis is applied for (7.37), the dispersion relation is complex-valued as

$$c(\xi) = a\xi - i\varepsilon\xi^2.$$

It is not appropriate to analyze dispersive behavior of the solution. What we can claim is that *the solution is diffusive*.

7.5.2. Conservative methods

Consider the Burgers's equation in *conservation form*:

$$u_t + \left(\frac{u^2}{2}\right)_x = 0. \tag{7.38}$$

It can be rewritten in *advection form*

$$u_t + uu_x = 0. \tag{7.39}$$

When we consider the advection form, a natural (explicit) numerical scheme reads

$$\frac{U_j^{n+1} - U_j^n}{k} + U_j^n \frac{U_j^n - U_{j-1}^n}{h} = 0. \tag{7.40}$$

When e.g. the initial value is given as

$$U_j^0 = \begin{cases} 1, & j < 0, \\ 0, & j \geq 0, \end{cases}$$

one can easily verify that

$$U_j^1 = U_j^0, \quad \forall\, j.$$

For other initial values, the scheme easily involves a large error in the shock speed. Why? Answer: It is not conservative.

Conservative methods: Consider the following conservative form of conservation law

$$u_t + f(u)_x = 0. \tag{7.41}$$

Its simple and natural numerical method can be formulated as

$$\frac{U_j^{n+1} - U_j^n}{k} + \frac{F(U_{j-p}^n, U_{j-p+1}^n, \cdots, U_{j+q}^n) - F(U_{j-p-1}^n, U_{j-p+1}^n, \cdots, U_{j+q-1}^n)}{h} = 0, \tag{7.42}$$

for some F of $p+q+1$ arguments, called the *numerical flux function*.

In the simplest case, $p = 0$ and $q = 1$. Then, (7.42) becomes

$$\boxed{\boldsymbol{U_j^{n+1} = U_j^n - \frac{k}{h}[F(U_j^n, U_{j+1}^n) - F(U_{j-1}^n, U_j^n)].}} \tag{7.43}$$

The above numerical scheme is very natural if we view U_j^n as an approximation of the cell average $\overline{u}_j^n$,

$$\overline{u}_j^n = \frac{1}{h}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t^n)dx.$$

Consider the integral form of the conservation law (7.17),

$$\begin{aligned}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t^{n+1})dx \;&=\; \int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t^n)dx \\ &+\int_{t^n}^{t^{n+1}} f(u(x_{j-1/2},t))dt - \int_{t^n}^{t^{n+1}} f(u(x_{j+1/2},t))dt.\end{aligned} \tag{7.44}$$

Then, dividing by h, we have

$$\boxed{\overline{u}_j^{n+1} \;=\; \overline{u}_j^n - \frac{1}{h}\left(\int_{t^n}^{t^{n+1}} f(u(x_{j+1/2},t))dt - \int_{t^n}^{t^{n+1}} f(u(x_{j-1/2},t))dt\right).} \tag{7.45}$$

Comparing this with (7.43), we can see that the numerical flux $F(U_j^n, U_{j+1}^n)$ plays the role of an average flux at $x = x_{j+1/2}$ over the time interval $[t^n, t^{n+1}]$:

$$\boxed{\boldsymbol{F(U_j^n, U_{j+1}^n) \;\approx\; \frac{1}{k}\int_{t^n}^{t^{n+1}} f(u(x_{j+1/2},t))dt.}} \tag{7.46}$$

The Godunov's method is based on this approximation, assuming that the solution is piecewise constant on each cell $(x_{j-1/2}, x_{j+1/2})$.

Upwind scheme: For the Burgers's equation (7.38), the upwind scheme in conservative form reads

$$U_j^{n+1} = U_j^n - \frac{k}{h}\Big[\frac{1}{2}(U_j^n)^2 - \frac{1}{2}(U_{j-1}^n)^2\Big], \tag{7.47}$$

where

$$F(U_j^n, U_{j+1}^n) = \frac{1}{2}(U_j^n)^2.$$

Lax-Friedrichs scheme: The generalization of the Lax-Friedrichs scheme to the conservation law takes the form

$$U_j^{n+1} = \frac{1}{2}(U_{j-1}^n + U_{j+1}^n) - \frac{k}{2h}\Big[f(U_{j+1}^n) - f(U_{j-1}^n)\Big], \tag{7.48}$$

which can be rewritten in the conservation form by taking

$$F(U_j^n, U_{j+1}^n) = \frac{h}{2k}(U_j^n - U_{j+1}^n) + \frac{1}{2}(f(U_j^n) + f(U_{j+1}^n)). \tag{7.49}$$

7.5.3. Consistency

The numerical method (7.43) is said to be *consistent* with the original conservation law if the numerical flux F reduces to the true flux f for the constant flow. That is, if $u(x,t) \equiv \widehat{u}$, say, then we expect

$$F(\widehat{u},\widehat{u}) = f(\widehat{u}), \quad \forall\, \widehat{u} \in \mathbb{R}. \tag{7.50}$$

We say F is *Lipschitz continuous* at $\widehat{u}$ if there is a constant $K \geq 0$ (which may depend on $\widehat{u}$) such that

$$|F(v,w) - f(\widehat{u})| \leq K \max(|v-\widehat{u}|, |w-\widehat{u}|).$$

Note that the Lipschitz continuity is sufficient for consistency.

7.5.4. Godunov's method

$$U_j^{n+1} = U_j^n - \frac{k}{h}[F(U_j^n, U_{j+1}^n) - F(U_{j-1}^n, U_j^n)], \tag{7.51}$$

where

$$F(U_j^n, U_{j+1}^n) \approx \frac{1}{k}\int_{t^n}^{t^{n+1}} f(\widetilde{u}(x_{j+1/2}, t))dt = f(u^*(U_j^n, U_{j+1}^n)). \tag{7.52}$$

Here

- $\widetilde{u}(x,t)$ is the piecewise constant representation of the solution, over the grid cell $(x_{j-1/2}, x_{j+1/2})$.
- $u^*(U_j^n, U_{j+1}^n)$ is the Riemann solution on $\{x_{j+1/2}\} \times [t^n, t^{n+1}]$.
- The method is consistent.
- Stability of the method requires to choose k small enough to satisfy

  $$\sigma = \frac{k}{h}\max_j |f'(U_j^n)| \leq 1,$$

  where σ is called the *Courant number*.

7.6. Nonlinear Stability

To guarantee convergence, we need some form of stability, just as for linear problems. Unfortunately, the Lax-Richtmyer Equivalence Theorem no longer holds and we cannot use the same approach to prove convergence. In this section, we will consider one form of nonlinear stability that allows us to prove convergence results for a wide class of practical problems. So far, this approach has been completely successful only for scalar problems. For general systems of equations with arbitrary initial data, no numerical method has been prove to be stable or convergent, although convergence results have been obtained in some special cases.

7.6.1. Total variation stability (TV-stability)

We first define the *total variation* (TV) over $[0,T]$ by

$$\begin{aligned} TV_T(u) &= \limsup_{\varepsilon\to 0}\frac{1}{\varepsilon}\int_0^T\int_{-\infty}^{\infty}|u(x+\varepsilon,t)-u(x,t)|\,dx\,dt \\ &\quad+\limsup_{\varepsilon\to 0}\frac{1}{\varepsilon}\int_0^T\int_{-\infty}^{\infty}|u(x,t+\varepsilon)-u(x,t)|\,dx\,dt. \end{aligned} \tag{7.53}$$

Define

$$\|v\|_{1,T}=\int_0^T\|v\|_1\,dt=\int_0^T\int_{-\infty}^{\infty}|v(x,t)|\,dx\,dt$$

and

$$\mathcal{K}=\{u\in L_{1,T}: TV_T(u)\le R \text{ and } \mathbf{Supp}(u(\cdot,t))\subset[-M,M],\ \ \forall\, t\in[0,T]\}. \tag{7.54}$$

When we consider numerical solution $U=\{U_j^n\}$, piecewise constant, then

$$\begin{aligned} TV_T(U) &= \sum_{n=0}^{T/k}\sum_{j=-\infty}^{\infty}\Big[k|U_{j+1}^n-U_j^n|+h|U_j^{n+1}-U_j^n|\Big] \\ &= \sum_{n=0}^{T/k}\Big[k\,TV(U^n)+\|U_j^{n+1}-U_j^n\|_1\Big]. \end{aligned} \tag{7.55}$$

Definition **7.11.** *We will say that a numerical method is* total variation stable *(TV-stable), if all approximations U_k for $k < k_0$ lie in some fixed set of the form (7.54) (where R and M may depend on the initial data u_0 and the flux function $f(u)$, but not on k).*

Theorem **7.12.** *Consider a conservative method with a Lipschitz continuous numerical flux $F(U; j)$. Suppose that for each initial data u_0, there exists some k_0, $R > 0$ such that*

$$TV(U^n) \le R, \qquad \forall\, n, k \text{ with } k < k_0, \;\; nk \le T. \tag{7.56}$$

Then, the method is TV-stable.

Theorem **7.13.** *Suppose U_k is generated by a numerical method in conservation form with Lipschitz continuous numerical flux, consistent with some scalar conservation law. If the method is TV-stable, then it is convergent in the following sense*

$$\text{dist}(U_k, \mathcal{W}) \to 0, \quad \text{as } k \to 0, \tag{7.57}$$

where $\mathcal{W} = \{w : \; w(x,t) \text{ is a weak solution}\}$.

7.6.2. Total variation diminishing (TVD) methods

We have just seen that TV-stability of a consistent and conservative numerical method is enough to guarantee convergence, in the sense in (7.57). One easy way to ensure TV-stability is to require that the TV be nonincreasing as time evolves, so that the TV at any time is uniformly bounded by the TV of the initial data. This requirement gives rise to the very important class of methods.

Definition 7.14. *The numerical method* $U_j^{n+1} = \mathcal{H}(U^n; j)$ *is called* total variation diminishing *(TVD) if*

$$TV(U^{n+1}) \leq TV(U^n) \tag{7.58}$$

for all grid functions U^n*.*

It can be shown that the true solution to the scalar conservation law has this TVD property, i.e., any weak solution $u(x,t)$ satisfies

$$TV(u(\cdot, t_2)) \leq TV(u(\cdot, t_1)) \quad \text{for } t_2 \geq t_1. \tag{7.59}$$

Thus it is reasonable to impose TVD on the numerical solution as well, yielding a TV-stability and hence convergence method.

7.6.3. Other nonoscillatory methods

Monotonicity preserving methods: A method is *monotonicity preserving* if U^n, $n \geq 1$, are monotone for a monotone initial data u_0.

Theorem 7.15. *Any TVD method is monotonicity preserving.*

Another attractive feature of the TVD requirement is that it is possible to derive methods with a high order of accuracy that are TVD. By contrast, if we define “stability" by mimicking certain other properties of the true solution, we find that accuracy is limited to first order. Nevertheless, we introduce some of these other concepts, because they are useful and frequently seen in the literature.

l_1-contracting methods: Any weak solution of a scalar conservation law satisfies

$$\|u(\cdot, t_2)\|_1 \leq \|u(\cdot, t_1)\|_1, \quad \text{for } t_2 \geq t_1. \tag{7.60}$$

More generally: *If u and v are both entropy solutions of the same conservation law (but possibly with different data), and if $u_0 - v_0$ has compact support, then*

$$\|u(\cdot, t_2) - v(\cdot, t_2)\|_1 \leq \|u(\cdot, t_1) - v(\cdot, t_1)\|_1, \quad \textit{for } t_2 \geq t_1. \tag{7.61}$$

This property is called *L_1-contraction*. In discrete space l_1, for grid functions $U = \{U_j\}$ we define the l_1-norm by

$$\|U\|_1 = h \sum_{j=-\infty}^{\infty} |U_j|.$$

In analogy to the L_1-contraction property (7.61) of the true solution operator, we say that a numerical method

$$U_j^{n+1} = \mathcal{H}(U^n; j) \tag{7.62}$$

is *l_1-contracting* if any two grid functions U^n and V^n for which $U^n - V^n$ has compact support satisfy

$$\|U^{n+1} - V^{n+1}\|_1 \leq \|U^n - V^n\|_1. \tag{7.63}$$

Theorem 7.16. *Any l_1-contracting numerical method is TVD.*

Proof. The proof depends on the following important relation between the 1-norm and TV: Given any grid function U, define V by shifting U as

$$V_j = U_{j-1}, \quad \forall\, j.$$

Then

$$TV(U) = \frac{1}{h}\|U - V\|_1.$$

Now, suppose the method (7.62) is l_1-contracting. Define $V_j^n = U_{j-1}^n$. Note that the methods under consideration are translation invariant, i.e.,

$$V_j^{n+1} = \mathcal{H}(V^n; j).$$

Thus l_1-contraction implies

$$\begin{aligned} TV(U^{n+1}) &= \frac{1}{h}\|U^{n+1} - V^{n+1}\|_1 \\ &\leq \frac{1}{h}\|U^n - V^n\|_1 \\ &= TV(U^n) \end{aligned}$$

and hence the method is TVD. □

Example 7.17. *The upwind method is l_1-contracting and therefore TVD, provided the CFL condition is satisfied.*

Monotone methods: Another useful property of the entropy-satisfying weak solution is as following: If we take two sets of initial data u_0 and v_0, with

$$v_0(x) \geq u_0(x), \quad \forall\, x,$$

then the respective entropy solutions u and v satisfy

$$v(x,t) \geq u(x,t), \quad \forall\, x,\, t. \tag{7.64}$$

The numerical method $U_j^{n+1} = \mathcal{H}(U^n; j)$ is called a *monotone method* if

$$V_j^n \geq U_j^n \Rightarrow V_j^{n+1} \geq U_j^{n+1}, \quad \forall\, j. \tag{7.65}$$

To prove that a method is monotone, it suffices to check that

$$\frac{\partial}{\partial U_i^n}\mathcal{H}(U^n; j) \geq 0, \quad \forall\, i,\, j,\, U^n. \tag{7.66}$$

This means that if we increase the value of any U_i^n then the value of U_j^{n+1} cannot decrease as a result.

Example 7.18. *The Lax-Friedrichs scheme (7.48) (See page 248) is monotone provided that the CFL condition is satisfied, because*

$$\mathcal{H}(U^n; j) = \frac{1}{2}(U_{j-1}^n + U_{j+1}^n) - \frac{k}{2h}\Big[f(U_{j+1}^n) - f(U_{j-1}^n)\Big]$$

satisfies

$$\frac{\partial}{\partial U_i^n}\mathcal{H}(U^n; j) = \begin{cases} \frac{1}{2}\Big(1 + \frac{k}{h}f'(U_{j-1}^n)\Big), & i = j-1, \\ \frac{1}{2}\Big(1 - \frac{k}{h}f'(U_{j+1}^n)\Big), & i = j+1, \\ 0, & \textit{otherwise}. \end{cases}$$

Theorem **7.19.** *Any monotone method is l_1-contracting.*

To summarize the relation between the different types of methods considered above, we have

$$\begin{array}{rcl} \text{monotone} & \Rightarrow & l_1\text{-contracting} \Rightarrow \text{TVD} \\ & & \Rightarrow \text{monotonicity preserving} \end{array}$$

Theorem **7.20.** *A monotone method is at most first-order accurate.*

Theorem **7.21.** *The numerical solution computed with a consistent monotone method with k/h fixed converges to the entropy solution as $k \to 0$.*

Note that the numerical solution by a TVD method converges to a weak solution that may not be the entropy solution. However, the notion of TV-stability is much more useful, because it is possible to derive TVD methods that have better than first-order accuracy.

We close the chapter with the following well-known theorem:

Theorem **7.22. (Godunov)**. *A linear, monotonicity preserving method is at most first-order accurate.*

7.7. Numerical Examples with Python

A Python code is implemented for the Lax-Wendroff scheme in (7.28), for solving

$$\begin{aligned} &u_t + a u_x = 0, \quad (x,t) \in (-1,6) \times (0,2] \\ &u(x,0) = \begin{cases} 1, & x \in [0,2] \\ 0, & \text{elsewhere}, \end{cases} \end{aligned} \tag{7.67}$$

where $a = 1$.

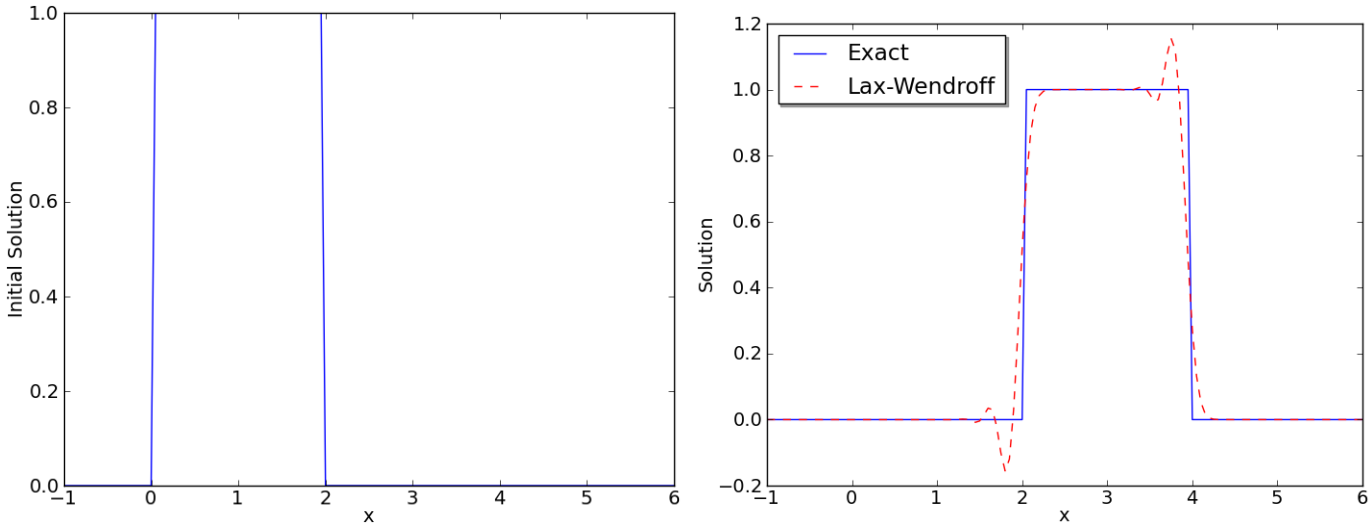


Figure 7.3: The Lax-Wendroff scheme: (left) The initial solution and (right) the solution at $t = 2$.

The following shows the main routine `lax_wendroff.py`:

```
def lax_wendroff(U0,ax,bx,nx,T,nt,a,level=0):
    hx,ht = (bx-ax)/nx, T/nt
    if level>=1:
        print("Lax-Wendroff: a=%g, nx=%d, nt=%d, hx=%g, ht=%g")\
             %(a,nx,nt,hx,ht)

    U =np.ndarray((2,nx+1),float)
    for i in range(nx+1):
        U[0][i]=U0[i];  U[1][i]=0.

    alam = a*ht/hx
    alam2= alam**2
    for n in range(0,nt):
        id0,id1 = n%2,(n+1)%2
        for j in range (1,nx):
            U[id1][j]=U[id0][j]-(alam/2.)*(U[id0][j+1]-U[id0][j-1])\
                 +(alam2/2.)*(U[id0][j+1]-2.*U[id0][j]+U[id0][j-1])
    return U[id1]
```

7.8. Homework

1. Find conditions on a and λ with which the FD schemes in (7.7.a)-(7.7.c) are stable or unstable.
2. Consider the leapfrog scheme (7.7.d).
   (a) Derive the relation
   $$\begin{aligned}&\sum_{m=-\infty}^{\infty} |v_m^{n+1}|^2 + |v_m^n|^2 + a\lambda(v_m^{n+1}v_{m+1}^n - v_{m+1}^{n+1}v_m^n)\\ &= \sum_{m=-\infty}^{\infty} |v_m^n|^2 + |v_m^{n-1}|^2 + a\lambda(v_m^n v_{m+1}^{n-1} - v_{m+1}^n v_m^{n-1})\\ &= \sum_{m=-\infty}^{\infty} |v_m^1|^2 + |v_m^0|^2 + a\lambda(v_m^1 v_{m+1}^0 - v_{m+1}^1 v_m^0)\end{aligned}$$
   (Hint: Multiply the leapfrog scheme by $v_m^{n+1} + v_m^{n-1}$ and sum over all m.)
   (b) Show that
   $$(1 - |a\lambda|)\sum_{m=-\infty}^{\infty} |v_m^{n+1}|^2 + |v_m^n|^2 \le (1 + |a\lambda|)\sum_{m=-\infty}^{\infty} |v_m^1|^2 + |v_m^0|^2.$$
   (Hint: Use the inequality $-\frac{1}{2}(x^2 + y^2) \le xy \le \frac{1}{2}(x^2 + y^2)$.)
   (c) Conclude the scheme is stable if $|a\lambda| < 1$.
3. Consider finite difference schemes of the form
   $$v_m^{n+1} = \alpha v_{m+1}^n + \beta v_{m-1}^n.$$
   (a) Show that they are stable if $|\alpha| + |\beta| \le 1$.
   (Use the arguments as in Example 7.6 rather than the Von Neumann analysis.)
   (b) Conclude that the Lax-Friedrichs scheme (7.7.e) is stable if $|a\lambda| \le 1$, where $\lambda = k/h$.
4. Verify the modified equation of the Beam-Warming scheme presented in (7.36).
5. Derive the conservation form for the Lax-Friedrichs scheme applied to the conservation law and presented in (7.48). (Use (7.49).)
6. Modify the Python code in § 7.7 to solve the one-way wave equation (7.67) by the Beam-Warming scheme (7.35).

CHAPTER 8

Locally One-Dimensional Methods

Explicit schemes for parabolic equations are easy to implement, but they are stable only if the time step size is chosen sufficiently small: $\Delta t = \mathcal{O}(\Delta x^2)$. Implicit methods are often unconditionally stable; however, a large algebraic system must be solved (directly or iteratively) for the time integration on each of the space-time slices. In this chapter, we will introduce the locally one-dimensional (LOD) methods such as the alternating direction implicit (ADI) method and the fractional step (FS) method, in order to solve the algebraic system of equations efficiently. The LOD methods can be viewed as a perturbation of standard implicit methods.

Contents of Chapter 8

8.1. Heat Conduction in 1D Space: Revisited 264
8.2. Heat Equation in Two and Three Variables 270
8.3. LOD Methods for the Heat Equation 276
8.4. Homework 299

8.1. Heat Conduction in 1D Space: Revisited

Let $\Omega = (0,1)$ and $J = (0,T]$, for some $T > 0$. Consider the following simplest model problem for parabolic equations in 1D:

$$\begin{aligned} u_t - u_{xx} &= 0, \quad (x,t) \in \Omega \times J, \\ u &= 0, \qquad (x,t) \in \Gamma \times J, \\ u &= u_0, \qquad x \in \Omega, \ \ t = 0, \end{aligned} \tag{8.1}$$

where Γ is the boundary of Ω, i.e., $\Gamma = \{0,1\}$, and u_0 is the prescribed initial value of the solution at $t = 0$.

Let

$$\begin{aligned} &\Delta t = T/n_t, \quad t^n = n\Delta t, \ \ n = 0, 1, \cdots, n_t; \\ &\Delta x = 1/n_x, \ \ x_j = j\Delta x, \ \ j = 0, 1, \cdots, n_x; \end{aligned}$$

for some positive integers n_t and n_x. Define $u_j^n = u(x_j, t^n)$. Let $\mathcal{A}_1$ be the central second-order approximation of $-\partial_{xx}$, defined as

$$\mathcal{A}_1 u_j^n := \frac{-u_{j-1}^n + 2u_j^n - u_{j+1}^n}{\Delta x^2}.$$

Then the θ-method for (8.1) is

$$\frac{v^n - v^{n-1}}{\Delta t} + \mathcal{A}_1 \left[\theta v^n + (1-\theta) v^{n-1}\right] = 0, \quad \theta \in [0,1], \tag{8.2}$$

or equivalently

$$(I + \theta \Delta t \mathcal{A}_1) v^n = [I - (1-\theta)\Delta t \mathcal{A}_1] v^{n-1}, \quad \theta \in [0,1]. \tag{8.3}$$

Forward Euler method ($\theta = 0$): The algorithm (8.3) is reduced to

$$v^n = (I - \Delta t \mathcal{A}_1) v^{n-1},$$

which is explicit and cheap to compute the solution in each time level. However, we shall see later that its stability requires to choose Δt small enough to satisfy

$$\mu = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}.$$

Backward Euler method ($\theta = 1$): This is an implicit method written as

$$(I + \Delta t \mathcal{A}_1) v^n = v^{n-1}.$$

The method must invert a tridiagonal matrix to get the solution in each time level. But it is stable independently on the choice of Δt.

Crank-Nicolson method ($\theta = 1/2$):

$$\left(I + \frac{\Delta t}{2} \mathcal{A}_1\right) v^n = \left(I - \frac{\Delta t}{2} \mathcal{A}_1\right) v^{n-1}.$$

It requires to solve a tridiagonal system in each time level, as in the backward Euler method. However, the Crank-Nicolson method is most popular, because

- it is unconditionally stable
- its error $= \mathcal{O}(\Delta x^2 + \Delta t^2)$

It is often called a *semi-implicit* method.

Stability analysis

Components of the algebraic system (8.3) are

$$\begin{aligned} &-\theta\mu\, v_{j-1}^n + (1+2\theta\mu)v_j^n - \theta\mu\, v_{j+1}^n \\ &\quad = (1-\theta)\mu\, v_{j-1}^{n-1} + [1-2(1-\theta)\mu]v_j^{n-1} + (1-\theta)\mu\, v_{j+1}^{n-1}, \end{aligned} \tag{8.4}$$

where $\mu = \Delta t/\Delta x^2$.

For an stability analysis for this one-parameter family of systems, substitute $g^n e^{ij\vartheta}$ for v_j^n in (8.4) to have

$$\begin{aligned} &g\left[-\theta\mu\, e^{-ij\vartheta} + (1+2\theta\mu) - \theta\mu\, e^{ij\vartheta}\right] \\ &\quad = (1-\theta)\mu\, e^{-ij\vartheta} + [1-2(1-\theta)\mu] + (1-\theta)\mu\, e^{ij\vartheta}, \end{aligned}$$

i.e.,

$$g = \frac{1-2(1-\theta)\mu\,(1-\cos\vartheta)}{1+2\theta\mu\,(1-\cos\vartheta)} = \frac{1-4(1-\theta)\mu\,\sin^2\frac{\vartheta}{2}}{1+4\theta\mu\,\sin^2\frac{\vartheta}{2}}.$$

Because $\mu > 0$ and $\theta \in [0,1]$, the amplification factor g cannot be larger than one. The condition $g \geq -1$ is equivalent to

$$1-4(1-\theta)\mu\,\sin^2\frac{\vartheta}{2} \;\geq\; -\left[1+4\theta\mu\,\sin^2\frac{\vartheta}{2}\right],$$

or

$$(1-2\theta)\mu\,\sin^2\frac{\vartheta}{2} \;\leq\; \frac{1}{2}.$$

Thus (8.3) is stable if

$$(1-2\theta)\mu \;\leq\; \frac{1}{2}. \tag{8.5}$$

In conclusion:

- The θ-method is unconditionally stable for $\theta \geq 1/2$, because every choice of μ satisfies the above inequality.
- When $\theta < 1/2$, the method is stable only if

$$\mu = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2(1-2\theta)}, \quad \theta \in [0, 1/2). \tag{8.6}$$

- For example, the forward Euler method ($\theta = 0$) is stable only if

$$\Delta t \leq \Delta x^2/2;$$

  Δt must be chosen sufficiently small for stability.

Maximum principle

For heat conduction without interior sources/sinks, it is known mathematically and physically that the extreme values of the solution appear either in the initial data or on the boundary. This property is called the *maximum principle*. It is quite natural and sometimes very important to examine if the numerical solution satisfies the maximum principle, too.

Theorem 8.1. (Maximum principle for the θ-method). *Let the θ-method be set satisfying $\theta \in [0, 1]$ and*

$$(1 - \theta)\mu \leq \frac{1}{2}.$$

If the computed solution v has an interior maximum or minimum, then v is constant.

Error analysis

Let

$$e_j^n = u_j^n - v_j^n,$$

where $u_j^n = u(x_j, t^n)$ with u being the exact solution of (8.1). Define

$$\mathcal{E}^n = \max_j |e_j^n|, \quad \mathcal{T}^{n-1/2} = \max_j |\mathcal{T} u_j^{n-1/2}|,$$

where $\mathcal{T} u_j^{n-1/2}$ is the truncation error expanded at $(x_j, t^{n-1/2})$. Note that $v_j^0 = u_j^0$, $j = 0, \cdots, n_x$, and therefore $\mathcal{E}^0 = 0$.

Theorem 8.2. *Let the θ-method be set satisfying $\theta \in [0, 1]$ and $(1-\theta)\mu \leq \frac{1}{2}$. Then,*

$$\mathcal{E}^n \leq \Delta t \sum_{k=1}^{n} \mathcal{T}^{k-1/2}. \tag{8.7}$$

It follows from (8.7) that

$$\mathcal{E}^n \leq n\Delta t \max_k \mathcal{T}^{k-1/2} \leq T \max_k \mathcal{T}^{k-1/2},$$

where T is the upper limit of the time variable.

8.2. Heat Equation in Two and Three Variables

Let Ω be a bounded domain in $\mathbb{R}^m$, $m = 2$ or 3, with boundary $\Gamma = \partial\Omega$. Consider the parabolic problem

$$\begin{aligned} u_t - \nabla \cdot (a\nabla u) + cu &= f, && (\mathbf{x}, t) \in \Omega \times J, \\ \alpha_1 u_\nu + \alpha_2 u &= g, && (\mathbf{x}, t) \in \Gamma \times J, \\ u &= u_0, && \mathbf{x} \in \Omega, \ \ t = 0, \end{aligned} \tag{8.8}$$

where

- $a > 0$, $c \geq 0$, $\alpha_1 \geq 0$, and $\alpha_2 \geq 0$ are given functions, $\alpha_1 + \alpha_2 > 0$,
- the subscript ν denotes the outer unit normal on Γ,
- u_0 is the prescribed initial value of the solution at $t = 0$, and
- f and g represent external sources and sinks.

8.2.1. The θ-method

Let $\mathcal{T}_h$ be the mesh of Ω consisting of elements of which the maximum edge size is h. Let $\mathcal{A}$ be the approximation of $-\nabla \cdot a\nabla + c$ on the mesh $\mathcal{T}_h$, having the p-th order accuracy, i.e.,

$$\mathcal{A}u \approx -\nabla \cdot (a\nabla u) + cu + \mathcal{O}(h^p).$$

Then, the θ-method for (8.8) reads[1]

$$\frac{v^n - v^{n-1}}{\Delta t} + \mathcal{A}\left[\theta v^n + (1-\theta)v^{n-1}\right] = f^{n-1/2}, \quad \theta \in [0,1], \tag{8.9}$$

and the truncation error for the n-th time level is

$$\delta^{n-1/2} = \mathcal{O}\left((1-2\theta)\Delta t + \Delta t^2 + h^p\right).$$

Note that $\mathcal{A}$ is symmetric and nonnegative; it is positive definite when $c > 0$ or $\alpha_2 > 0$.

Let $\mathbf{v}^n$ be the solution vector in the n-th time level. Then the method (8.9) in its matrix representation reads

$$\left[I + \theta\Delta t\mathcal{A}\right]\mathbf{v}^n = \Delta t f^{n-1/2} + \left[I - (1-\theta)\Delta t\mathcal{A}\right]\mathbf{v}^{n-1}. \tag{8.10}$$

[1] Here we used $f^{n-1/2}$, instead of $f^{n-1+\theta}$, for a simplier presentation.

Notes:

- When $\theta > 0$, it is necessary to invert a matrix, either exactly or approximately, to get the solution in the new time level.
- When the domain is rectangular or cubic, the algebraic system (8.10) can be perturbed to become a series of traditional systems; the resulting problem can be solved very efficiently. This is the basic idea of the locally one-dimensional (LOD) methods to be treated in this chapter later.

8.2.2. Convergence analysis for θ-method

For a simpler presentation, we define

$$\overline{\partial}_t v^n = \frac{v^n - v^{n-1}}{\Delta t}.$$

Let

$$e^n = u^n - v^n,$$

where u^n is the exact solution of (8.8) at the time level t^n. Then, the error equation associated with the θ-method (8.9) is

$$\overline{\partial}_t e^n + \mathcal{A}[\theta e^n + (1-\theta)e^{n-1}] = \delta^{n-1/2}. \tag{8.11}$$

Choose $\overline{\partial}_t e^n$ as a test function. Then, for $n \geq 1$,

$$(\overline{\partial}_t e^n, \overline{\partial}_t e^n) + \left(\mathcal{A}[\theta e^n + (1-\theta)e^{n-1}],\, \overline{\partial}_t e^n\right) = (\delta^{n-1/2}, \overline{\partial}_t e^n). \tag{8.12}$$

Note that

$$\theta e^n + (1-\theta)e^{n-1} = \frac{1}{2}\left((e^n + e^{n-1}) + (2\theta - 1)(e^n - e^{n-1})\right)$$

and therefore

$$\begin{aligned} &\left(\mathcal{A}[\theta e^n + (1-\theta)e^{n-1}],\, \overline{\partial}_t e^n\right)\Delta t \\ &= \frac{1}{2}\Big[(\mathcal{A}e^n, e^n) - (\mathcal{A}e^{n-1}, e^{n-1}) \\ &\quad + (2\theta - 1)(\mathcal{A}\overline{\partial}_t e^n, \overline{\partial}_t e^n)\Delta t^2\Big], \quad n \geq 1. \end{aligned} \tag{8.13}$$

Multiply (8.12) by Δt and utilize (8.13) to have

$$\begin{aligned} \|\overline{\partial}_t e^n\|^2 \Delta t + \frac{2\theta - 1}{2} (\mathcal{A}\overline{\partial}_t e^n, \overline{\partial}_t e^n) \Delta t^2 \\ + \frac{1}{2} \left[(\mathcal{A}e^n, e^n) - (\mathcal{A}e^{n-1}, e^{n-1}) \right] \\ = (\delta^{n-1/2}, \overline{\partial}_t e^n) \Delta t, \quad n \geq 1. \end{aligned} \tag{8.14}$$

Summing (8.14) beginning at $n = 1$ reads

$$\begin{aligned} \sum_{j=1}^{n} \|\overline{\partial}_t e^j\|^2 \Delta t + \frac{2\theta - 1}{2} \sum_{j=1}^{n} (\mathcal{A}\overline{\partial}_t e^j, \overline{\partial}_t e^j) \Delta t^2 + \frac{1}{2} (\mathcal{A}e^n, e^n) \\ = \frac{1}{2} (\mathcal{A}e^0, e^0) + \sum_{j=1}^{n} (\delta^{j-1/2}, \overline{\partial}_t e^j) \Delta t. \end{aligned} \tag{8.15}$$

Now, we apply the inequality ($|ab| \leq (a^2 + b^2)/2$) to the last term in (8.15) to obtain the following inequality:

$$\begin{aligned} \sum_{j=1}^{n} \|\overline{\partial}_t e^j\|^2 \Delta t + (2\theta - 1) \sum_{j=1}^{n} (\mathcal{A}\overline{\partial}_t e^j, \overline{\partial}_t e^j) \Delta t^2 + (\mathcal{A}e^n, e^n) \\ \leq (\mathcal{A}e^0, e^0) + \sum_{j=1}^{n} \|\delta^{j-1/2}\|^2 \Delta t. \end{aligned} \tag{8.16}$$

Thus, the estimation of the error generated by the θ-method is reduced to bounding the errors in v^0 and the truncation error.

Note: The estimate (8.16) also indicates that

- The θ-method is unconditionally stable for $\theta \in [1/2, 1]$.
- When $\theta \in [0, 1/2)$, it is stable if

$$1 + (2\theta - 1)\rho(\mathcal{A})\Delta t \geq 0,$$

where $\rho(\mathcal{A})$ is the spectral radius of $\mathcal{A}$ (the largest eigenvalue of $\mathcal{A}$ in modulus). Since

$$\rho(\mathcal{A}) \approx 4m\|a\|_\infty / h^2,$$

where m is the dimensionality and $\|a\|_\infty = \max_{x\in\Omega} |a(x)|$, the θ-method is stable if

$$\frac{\Delta t}{h^2} \leq \frac{1}{4(1-2\theta)m\|a\|_\infty}, \quad \theta \in [0, 1/2). \tag{8.17}$$

The inequality in (8.17) is compared to the analysis in (8.6).

- The θ-method is particularly interesting when $\theta = 1/2$, because the truncation error becomes second-order in time. This case is called the *Crank-Nicolson* or *semi-implicit* method. The spatial derivatives can be approximated to have a p-th order accuracy, $p \geq 2$, independently on θ or Δt.

8.3. LOD Methods for the Heat Equation

Over the last five decades or so, many time-stepping procedures have been introduced to allow multidimensional parabolic problems to be approximated accurately and *efficiently*. These procedures treat the spatial variables individually in a cyclic fashion; we shall call any such a procedure a *locally one-dimensional* (LOD) method. Here we will be mainly concerned with two families of these methods, namely the alternating direction implicit (ADI) methods [14, 18, 56] and the fractional-step (FS) procedures [20, 51, 71, 72]. These methods can be interpreted as perturbations of some underlying implicit multidimensional numerical method, such as the Crank-Nicolson or the backward Euler method. Recently, a unified approach of these LOD methods, along with strategies for virtual elimination of the splitting error, has been studied by Douglas and Kim [16].

8.3.1. The ADI method

Consider the parabolic problem (8.8) defined on a rectangular domain $\Omega \subset \mathbb{R}^2$. Let $\mathcal{T}_h$ be a uniform mesh of rectangular elements of which the edge lengths are h_x and h_y, $h = \max(h_x, h_y)$. Define

$$\mathcal{A}_1 u \approx -(a u_x)_x + \frac{1}{2} c u, \quad \mathcal{A}_2 u \approx -(a u_y)_y + \frac{1}{2} c u,$$

which are finite difference or finite element approximations on the mesh $\mathcal{T}_h$ having a truncation error of $\mathcal{O}(h^p)$, $p \geq 2$. Let

$$\mathcal{A} = \mathcal{A}_1 + \mathcal{A}_2.$$

Then the Crank-Nicolson difference equation for the heat equation (8.8) reads

$$\frac{v^n - v^{n-1}}{\Delta t} + \frac{1}{2}\mathcal{A}(v^n + v^{n-1}) = f^{n-1/2} + \mathcal{O}(h^p + \Delta t^2), \tag{8.18}$$

where

$$f^{n-1/2} = \frac{1}{2}(f^n + f^{n-1}).$$

The truncation error for the CN procedure (8.18) is

$$\mathcal{O}(\Delta x^2 + \Delta t^2).$$

The original ADI method:

The ADI method of Douglas-Peaceman-Rachford [14, 18, 56] is a perturbation of the Crank-Nicolson difference equation that has a splitting error of $\mathcal{O}(\Delta t^2)$, so that it is second-order correct in time.

Let us formulate it in an equivalent way that will coincide with the general formulation in Douglas-Gunn [15] of ADI methods. Given an approximation w^0 to u_0, find w^n, $n \geq 1$, by solving

$$\begin{aligned} \frac{w^* - w^{n-1}}{\Delta t} + \frac{1}{2}\mathcal{A}_1(w^* + w^{n-1}) + \mathcal{A}_2 w^{n-1} &= f^{n-1/2}, \\ \frac{w^n - w^{n-1}}{\Delta t} + \frac{1}{2}\mathcal{A}_1(w^* + w^{n-1}) + \frac{1}{2}\mathcal{A}_2(w^n + w^{n-1}) &= f^{n-1/2}, \end{aligned} \tag{8.19}$$

or, equivalently,

$$\begin{aligned} \Big(1 + \frac{\Delta t}{2}\mathcal{A}_1\Big) w^* &= \Big(1 - \frac{\Delta t}{2}\mathcal{A}_1 - \Delta t \mathcal{A}_2\Big) w^{n-1} + \Delta t f^{n-1/2}, \\ \Big(1 + \frac{\Delta t}{2}\mathcal{A}_2\Big) w^n &= w^* + \frac{\Delta t}{2}\mathcal{A}_2 w^{n-1}. \end{aligned} \tag{8.20}$$

Here w^* is an intermediate value.

Splitting error of ADI: The intermediate solution w^* can be found (implicitly) as

$$w^* = w^n + \frac{\Delta t}{2}\mathcal{A}_2(w^n - w^{n-1}).$$

Thus, by plugging it into the first equation of (8.20), we have

$$\Big(1 + \frac{\Delta t}{2}\mathcal{A}_1\Big)\Big(1 + \frac{\Delta t}{2}\mathcal{A}_2\Big)w^n = \Big(1 - \frac{\Delta t}{2}\mathcal{A}\Big)w^{n-1} + \frac{\Delta t^2}{4}\mathcal{A}_1\mathcal{A}_2 w^{n-1} + \Delta t f^{n-1/2}.$$

Multiply out the left hand side and rewrite the result as

$$\frac{w^n - w^{n-1}}{\Delta t} + \frac{1}{2}\mathcal{A}(w^n + w^{n-1}) + \frac{\Delta t}{4}\mathcal{A}_1\mathcal{A}_2(w^n - w^{n-1}) = f^{n-1/2}. \tag{8.21}$$

Thus, compared with (8.18), the splitting error is given by

$$\frac{\Delta t}{4}\mathcal{A}_1\mathcal{A}_2(w^n - w^{n-1}), \tag{8.22}$$

which is $\mathcal{O}(\Delta t^2)$ for a smooth solution.

Notes:

- Some theoretical aspects of the method were treated in detail in Douglas [14], while practical aspects of the method were considered in the companion paper by Peaceman-Rachford [56].
- In each half of the calculation, the matrix to be inverted is tridiagonal, so that the algorithm requires $\mathcal{O}(N := n_t n_x n_y)$ flops.
- The ADI (8.19) can be equivalently formulated in many different ways. The modelcode `ADI_HEAT.CF.tar` in GRADE [35] is implemented based on the following formulation:

$$\begin{aligned} \left(1+\frac{\Delta t}{2}\mathcal{A}_1\right)w^* &= \left(1-\frac{\Delta t}{2}\mathcal{A}_2\right)w^{n-1}+\frac{\Delta t}{2}f^{n-1/2} \\ \left(1+\frac{\Delta t}{2}\mathcal{A}_2\right)w^n &= \left(1-\frac{\Delta t}{2}\mathcal{A}_1\right)w^*+\frac{\Delta t}{2}f^{n-1/2}. \end{aligned} \tag{8.23}$$

General ADI procedure

Consider a parabolic problem of the form

$$u_t + \sum_{i=1}^{m} A_i u = f, \quad (\mathbf{x}, t) \in \Omega \times J, \tag{8.24}$$

with an appropriate initial data and boundary condition. If $A = A_1 + \cdots + A_m$, then the basic Crank-Nicolson approximation to (8.24) is given by

$$\frac{w^n - w^{n-1}}{\Delta t} + \frac{1}{2} A(w^n + w^{n-1}) = f^{n-1/2}, \quad n \geq 1. \tag{8.25}$$

(Here, we are interested in the time discretization of (8.24); consequently, we shall ignore spatial discretization for the moment.)

The Douglas-Gunn algorithm [15] for ADI time discretization of (8.24) is as follows: For $\kappa = 1, \ldots, m$, find $w^{n,\kappa}$ such that

$$\frac{w^{n,\kappa} - w^{n-1}}{\Delta t} + \frac{1}{2} \sum_{i=1}^{\kappa} A_i (w^{n,i} + w^{n-1}) + \sum_{i=\kappa+1}^{m} A_i w^{n-1} = f^{n-1/2}, \tag{8.26}$$

and then to set

$$w^n = w^{n,m}. \tag{8.27}$$

In the above,

$$\sum_{m+1}^{m} A_i w^{n-1} := 0.$$

The Douglas-Gunn algorithm equivalently reads

$$\begin{aligned}\Big(1+\frac{\Delta t}{2}A_1\Big)w^{n,1} &= \Big(1-\frac{\Delta t}{2}A_1-\Delta t\sum_{i=2}^{m}A_i\Big)w^{n-1}+\Delta t f^{n-1/2},\\ \Big(1+\frac{\Delta t}{2}A_\kappa\Big)w^{n,\kappa} &= w^{n,\kappa-1}+\frac{\Delta t}{2}A_\kappa w^{n-1},\ \ \kappa=2,\dots,m,\\ w^n &= w^{n,m}.\end{aligned} \tag{8.28}$$

Splitting error: The intermediate values $w^{n,1},\cdots,w^{n,m-1}$ can be eliminated by recursively operating on the second equation of (8.28) by $(1+\frac{\Delta t}{2}A_\kappa)$ for $\kappa=m-1,\cdots,1$:

$$\frac{w^n-w^{n-1}}{\Delta t}+\frac{1}{2}A(w^n+w^{n-1})+\mathcal{B}_{\Delta t}(w^n-w^{n-1})=f^{n-1/2}, \tag{8.29}$$

where

$$\begin{aligned}\mathcal{B}_{\Delta t} &= \frac{\Delta t}{4}\sum_{1\le i_1<i_2\le m}A_{i_1}A_{i_2}+\frac{\Delta t^2}{8}\sum_{1\le i_1<i_2<i_3\le m}A_{i_1}A_{i_2}A_{i_3}\\ &\quad+\cdots+\frac{\Delta t^{m-1}}{2^m}A_1A_2\cdots A_m.\end{aligned} \tag{8.30}$$

The splitting perturbation is given by $\mathcal{B}_{\Delta t}(w^n-w^{n-1})$, and for sufficiently smooth solutions u,

$$\mathcal{B}_{\Delta t}(u^n-u^{n-1})=\mathcal{O}(\Delta t^2), \tag{8.31}$$

which is of the same order in Δt as the Crank-Nicolson truncation error. But the splitting error can be much larger than the truncation error as shown in the following.

8.3.2. Accuracy of the ADI: Two examples

Let $\Omega \times J = (0,1)^2 \times (0,1)$, $a = \alpha_1 \equiv 1$, and $c = \alpha_2 \equiv 0$ in (8.8). Consider two different solutions:

$$\begin{aligned} u_+ &= \sin(2\pi\nu_t t) + \sin(2\pi\nu_x x) + \sin(2\pi\nu_y y), \\ u_\times &= \sin(2\pi\nu_t t) \cdot \sin(2\pi\nu_x x) \cdot \sin(2\pi\nu_y y). \end{aligned} \tag{8.32}$$

For the moment, take $\nu_t = \nu_x = \nu_y = 1$.

The sources f and g are evaluated so that (8.8) is satisfied. Also, let $n := n_t = n_x = n_y$. To compare computation cost and accuracy, we implemented three algorithms:

- an LU-based algorithm,
- a PCG-ILU0 procedure for the Crank-Nicolson equation derivable from (8.9), and
- the ADI procedure of (8.19).

Here, PCG-ILU0 denotes the conjugate gradient method preconditioned by the zero-level (not allowing fill-in) incomplete LU-factorization. The PCG-ILU0 procedure was initialized at each time level by the extrapolation

$$u^{n,0} = 2u^{n-1} - u^{n-2}, \quad n \geq 2,$$

and the iteration stopped when the residual was reduced by a factor of 10^{-5}.

	$n = 40$		$n = 80$		$n = 160$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
LU-based	0.74	4.10e-3	9.07	1.00e-3	126	2.47e-4
PCG-ILU0	0.46	4.11e-3	5.67	1.00e-3	53.4	2.47e-4
ADI	0.26	4.10e-3	2.16	1.00e-3	17.9	2.47e-4

Table 8.1: The performances of the LU-based, PCG-ILU0, and ADI methods for $u = u_+$. The elapsed time (CPU) is measured in seconds and the L^2-norm of the error is evaluated at $t = 1$.

	$n = 40$		$n = 80$		$n = 160$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
LU-based	0.91	2.46e-4	10.5	5.98e-5	136	1.47e-5
PCG-ILU0	0.83	2.46e-4	12.5	5.97e-5	121	1.42e-5
ADI	0.45	8.44e-3	3.62	2.02e-3	29.0	4.90e-4

Table 8.2: The performances of the LU-based, PCG-ILU0, and ADI methods for $u = u_\times$.

Table 8.1 presents the elapsed times and numerical errors for $u = u_+$ for various grid sizes. As one can see from the table, the three different algorithms show the same errors and their second-order convergence.

Table 8.2 shows the results for $u = u_\times$. The computation cost for the ADI method increases *linearly* as the number of grid points grows, while the PCG-ILU0 calculation shows a slight *superlinearity* in its computation cost. However, the ADI method produces an error approximately 34 times larger than that for the LU-based or PCG-ILU0 methods for the same grid size.

Truncation error vs. splitting error: The truncation error for the Crank-Nicolson difference equation is of the form

$$\mathcal{O}\left(h_x^2 \frac{\partial^4 u}{\partial x^4}\right) + \mathcal{O}\left(h_y^2 \frac{\partial^4 u}{\partial y^4}\right) + \mathcal{O}\left(\Delta t^2 \frac{\partial^3 u}{\partial t^3}\right),$$

while the splitting error of the ADI method is

$$\mathcal{O}\left(\Delta t^2 \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial y^2} \frac{\partial}{\partial t} u\right).$$

This is, roughly speaking, why the ADI method introduces no splitting error for u_+ and a large splitting error for $u_\times$.

Now, since the operators $\mathcal{A}_i$ usually represent second-order differential operators in an x_i direction, it should not be surprising that the higher-order derivatives in $\mathcal{B}_{\Delta t}$ contribute bigger errors than the truncation error. We shall see in §8.3.4 that it is not only possible but also quite feasible to modify the algorithm (8.26) in a rather simple fashion to reduce the splitting error to $\mathcal{O}(\Delta t^3)$.

8.3.3. The general fractional step (FS) procedure

We shall consider the same parabolic problem (8.24) for a FS time discretization. For reasons that will appear below, it is not the usual case to look for an FS procedure based on the Crank-Nicolson equation (8.25); however, it is useful for us to do so.

The appropriate FS algorithm is given by

$$\begin{aligned}
\frac{w^{n,1}-w^{n-1}}{\Delta t}+\frac{1}{2}A_1(w^{n,1}+w^{n-1}) &= f^{n-1/2},\\
\frac{w^{n,\kappa}-w^{n,\kappa-1}}{\Delta t}+\frac{1}{2}A_\kappa(w^{n,\kappa}+w^{n-1}) &= 0,\ \ \kappa=2,\ldots,m-1,\\
\frac{w^{n}-w^{n,m-1}}{\Delta t}+\frac{1}{2}A_m(w^{n}+w^{n-1}) &= 0.
\end{aligned} \tag{8.33}$$

Equivalently,

$$\begin{aligned}
\Big(1+\frac{\Delta t}{2}A_1\Big)w^{n,1} &= \Big(1-\frac{\Delta t}{2}A_1\Big)w^{n-1}+\Delta t f^{n-1/2},\\
\Big(1+\frac{\Delta t}{2}A_\kappa\Big)w^{n,\kappa} &= w^{n,\kappa-1}-\frac{\Delta t}{2}A_\kappa w^{n-1},\quad \kappa=2,\ldots,m-1,\\
\Big(1+\frac{\Delta t}{2}A_m\Big)w^{n} &= w^{n,m-1}-\frac{\Delta t}{2}A_m w^{n-1}.
\end{aligned} \tag{8.34}$$

Splitting error of FS procedure: Again, the intermediate values can be eliminated:

$$\frac{w^n - w^{n-1}}{\Delta t} + \frac{1}{2}A(w^n + w^{n-1}) + \mathcal{B}_{\Delta t}(w^n + w^{n-1}) = f^{n-1/2}, \tag{8.35}$$

with $\mathcal{B}_{\Delta t}$ being the same as for the ADI; see (8.30).

Thus, for the Crank-Nicolson version of the FS method, the splitting perturbation term becomes $\mathcal{B}_{\Delta t}(w^n + w^{n-1})$. We know that

$$\mathcal{B}_{\Delta t}(u^n + u^{n-1}) = \mathcal{O}(\Delta t); \tag{8.36}$$

i.e., the splitting error term is worse than the inherent local error in the Crank-Nicolson equation.

This is the reason that (8.33) is not common; the FS methods have been employed for the backward Euler method rather than the Crank-Nicholson method. However, we shall be able to modify the procedure (8.33) in an equally simple fashion to reduce the splitting error to $\mathcal{O}(\Delta t^3)$ below.

8.3.4. Improved accuracy for LOD procedures

We present a strategy to reduce the perturbation error of ADI and FS procedures and essentially to recover the accuracy of the Crank-Nicolson difference equation for an additional computational cost that is a small fraction of the standard ADI or FS cost.

Correction term for the ADI method: Observation from (8.26), (8.29), and (8.30) is that

> if the right hand side term of (8.26) is $f^{n-1/2}$, then the right hand side of (8.29) is also $f^{n-1/2}$ and the splitting error is given by $\mathcal{B}_{\Delta t}(w^n - w^{n-1})$.

If we could add $\mathcal{B}_{\Delta t}(w^n - w^{n-1})$ to the right hand side of (8.29), then we could cancel the perturbation term completely; but since we do not know w^n, we cannot make this modification in the algorithm.

Our best estimate for $(w^n - w^{n-1})$ is $(w^{n-1} - w^{n-2})$.

Modification of the ADI: Let us modify the ADI algorithm to the following: For $n \geq 2$,

$$\begin{aligned} F_{AD}^n &= f^{n-1/2} + \mathcal{B}_{\Delta t}(z^{n-1} - z^{n-2}), \\ \left(1 + \frac{\Delta t}{2} A_1\right) z^{n,1} &= \left(1 - \frac{\Delta t}{2} A_1 - \Delta t \sum_{i=2}^{m} A_i\right) z^{n-1} + \Delta t \, F_{AD}^n, \\ \left(1 + \frac{\Delta t}{2} A_\kappa\right) z^{n,\kappa} &= z^{n,\kappa-1} + \frac{\Delta t}{2} A_\kappa z^{n-1}, \quad \kappa = 2, \ldots, m, \\ z^n &= z^{n,m}. \end{aligned} \tag{8.37}$$

The evaluation of z^1 will be discussed below by interpreting the modified method as an iterative procedure; for practical purposes, assume that z^1 is obtained by solving the Crank-Nicolson equation for this single time step.

Splitting error: By eliminating the intermediate values (or referring to (8.29)), we see that z^n satisfies

$$\frac{z^n - z^{n-1}}{\Delta t} + \frac{1}{2}A(z^n + z^{n-1}) + \mathcal{B}_{\Delta t}(z^n - 2z^{n-1} + z^{n-2}) = f^{n-1/2}, \quad n \geq 2. \tag{8.38}$$

Now, for a smooth solution u of (8.8),

$$\mathcal{B}_{\Delta t}(u^n - 2u^{n-1} + u^{n-2}) = \mathcal{O}(\Delta t^3), \tag{8.39}$$

and the splitting error is now higher order in Δt than the truncation error of the Crank-Nicolson equation.

We shall both prove the convergence of the solution of (8.37) to that of (8.8) under certain circumstances and demonstrate that the error in the solution of (8.37) is reduced essentially to that of the Crank-Nicolson procedure for the example $u_\times$ considered above, for which the splitting error was many times as large as the Crank-Nicolson error.

Algebraic interpretation: We will interpret (8.38) as the iterative procedure related to the *matrix splitting* [67]

$$1 + \frac{\Delta t}{2}A = \left(1 + \frac{\Delta t}{2}A + \mathcal{B}_{\Delta t}\right) - \mathcal{B}_{\Delta t}.$$

Consider the algorithm: Find $\zeta,\ \geq 1$, by recursively solving

$$\left(1 + \frac{\Delta t}{2}A + \mathcal{B}_{\Delta t}\right)\zeta = \mathcal{B}_{\Delta t}\zeta^{-1} + \left(1 - \frac{\Delta t}{2}A\right)\gamma + f^{n-1/2}. \tag{8.40}$$

The solution w^n of the original ADI method (8.26) is the first iterate ζ^1 of (8.40) for $\gamma = w^{n-1}$ starting with the initial value

$$\zeta^0 = w^{n-1}. \tag{8.41}$$

On the other hand, the solution z^n of (8.37) is the first iterate of (8.40) with $\gamma = z^{n-1}$ and the initial value

$$\zeta^0 = 2z^{n-1} - z^{n-2}. \tag{8.42}$$

Consequently, the algorithm (8.37) is called the *alternating direction implicit method with improved initialization* (ADI-II) [16].

If the general time step code for (8.37) is written to perform the iteration (8.40), then, for $n \geq 2$, (8.42) would be used to initialize the “iteration" and one step of iteration calculated, while for $n = 1$, (8.41) would be used to initialize the iteration and two or more iterations would give z^1 to the desired accuracy.

Reformulation of ADI-II: As for ADI, ADI-II (8.37) can be formulated in a various way. For the 2D problem ($m = 2$), the ADI-II routine in `ADI_HEAT.CF.tar` is implemented based on

$$\begin{aligned}\left(I + \frac{\Delta t}{2}A_1\right)z^{n,1} &= \left(I - \frac{\Delta t}{2}A\right)z^{n-1} + \Delta t f^{n-1/2} \\ &\quad + \frac{\Delta t^2}{4}A_1A_2(2z^{n-1} - z^{n-2}), \\ \left(I + \frac{\Delta t}{2}A_2\right)z^{n} &= z^{n,1}.\end{aligned} \tag{8.43}$$

- It might seem reasonable to use a higher-order extrapolation than (8.42), but experiments have shown that instability can result unless the time step is small enough.
- It has also been observed that (8.42) can over-correct for large time steps, and it is possible that the use of

$$\zeta^0 = z^{n-1} + \eta(z^{n-1} - z^{n-2}), \quad 0 \le \eta \le 1, \tag{8.44}$$

  could lead to better computational results for large time steps.
- However, experiments have shown that, when the time step is reasonably chosen (*e.g.*, $\Delta t \lesssim ah$), ADI-II methods have worked better than ADI methods for various heterogeneous media; see Tables 8.3 and 8.4 in §8.3.6. So, (8.44) does not seem necessary for solving heat equations in practice.

Correction term for the FS method

The FS difference equation (8.35) preserves the right hand side of the FS algorithm (8.34) and exhibits the splitting perturbation $\mathcal{B}_{\Delta t}(w^n+w^{n-1})$. Modify (8.34) as follows. For $n \geq 2$, let

$$\begin{aligned} F^n_{FS} &= f^{n-1/2} + \mathcal{B}_{\Delta t}(3z^{n-1} - z^{n-2}), \\ \left(1 + \frac{\Delta t}{2}A_1\right) z^{n,1} &= \left(1 - \frac{\Delta t}{2}A_1\right) z^{n-1} + \Delta t\, F^n_{FS}, \\ \left(1 + \frac{\Delta t}{2}A_\kappa\right) z^{n,\kappa} &= z^{n,\kappa-1} - \frac{\Delta t}{2}A_\kappa z^{n-1}, \quad \kappa = 2, \cdots, m-1, \\ \left(1 + \frac{\Delta t}{2}A_m\right) z^{n} &= z^{n,m-1} - \frac{\Delta t}{2}A_m z^{n-1}. \end{aligned} \tag{8.45}$$

After the intermediate values are eliminated, we see that z^n satisfies

$$\frac{z^n - z^{n-1}}{\Delta t} + \frac{1}{2}A(z^n + z^{n-1}) + \mathcal{B}_{\Delta t}(z^n - 2z^{n-1} + z^{n-2}) = f^{n-1/2}, \tag{8.46}$$

which is identical to the equation (8.38) satisfied by the solution of the ADI-II algorithm (8.37).

Remarks [16]:

- We have not only shown how to reduce the splitting errors for the ADI and FS methods but also discovered that their improved procedures lead to identical results "(**after several decades of being considered to be different techniques**)."
- Again, it is advisable to obtain z^1 as discussed earlier.
- If the values of $A_i z^{n-1}$ are saved, then there is essentially no difference in the implementation of algorithms (8.37) and (8.45). That being the case, we shall address both algorithms as pertaining to the ADI-II method.

8.3.5. A convergence proof for the ADI-II

Let $\|\cdot\|$ denote the $L^2(\Omega)$ or ${}^2(\Omega)$ norm and $\|\cdot\|_1$ the norm on either $H^1(\Omega)$ or $h^1(\Omega)$, as appropriate. (That is, depending on spatial discretization by finite elements or finite differences.) Assume that the operators $\{A_i\}$ commute:

$$A_iA_j = A_jA_i, \qquad i,j = 1,\dots,m, \tag{8.47}$$

and that

$$(A_iz, z) \geq \alpha\|z\|_1^2, \qquad \alpha > 0. \tag{8.48}$$

By (8.47) and (8.48), it follows that

$$(\mathcal{B}_{\Delta t}z, z) \geq 0.$$

Let $\overline{\partial}_t v^n = (v^n - v^{n-1})/\Delta t$ and $e^n = u^n - z^n$. Then, the error equation associated with ADI-II (8.38) is

$$\overline{\partial}_t e^n + \frac{1}{2}A(e^n + e^{n-1}) + \mathcal{B}_{\Delta t}(e^n - 2e^{n-1} + e^{n-2}) = \delta^n, \tag{8.49}$$

where δ^n is the truncation error on the n-th level, i.e.,

$$\delta^n = \mathcal{O}(\Delta t^2 + h^p), \qquad p \geq 2, \tag{8.50}$$

for any reasonable spatial discretization. Choose $\overline{\partial}_t e^n$ as a test function. Then, for $n \geq 2$,

$$(\overline{\partial}_t e^n, \overline{\partial}_t e^n) + \frac{1}{2}\left(A(e^n + e^{n-1}), \overline{\partial}_t e^n\right) + \Delta t^2\left(\mathcal{B}_{\Delta t}\overline{\partial}_t^2 e^n, \overline{\partial}_t e^n\right) = (\delta^n, \overline{\partial}_t e^n). \tag{8.51}$$

Multiply (8.51) by Δt and sum beginning at $n = 2$ to have

$$\begin{aligned}\sum_{j=2}^{n}\|\overline{\partial}_t e^j\|^2\Delta t + \frac{1}{2}(Ae^n, e^n) + \Delta t^2\sum_{j=2}^{n}(\mathcal{B}_{\Delta t}\overline{\partial}_t^2 e^j, \overline{\partial}_t e^j)\Delta t \\ = \frac{1}{2}(Ae^1, e^1) + \sum_{j=2}^{n}(\delta^j, \overline{\partial}_t e^j)\Delta t.\end{aligned} \tag{8.52}$$

Now, since $b^2 - ab \geq (b^2 - a^2)/2$, we have

$$\begin{aligned}\sum_{j=2}^{n}(\mathcal{B}_{\Delta t}\overline{\partial}_t^2 e^j, \overline{\partial}_t e^j)\Delta t &= \sum_{j=2}^{n}(\mathcal{B}_{\Delta t}[\overline{\partial}_t e^j - \overline{\partial}_t e^{j-1}], \overline{\partial}_t e^j) \\ &\geq \frac{1}{2}(\mathcal{B}_{\Delta t}\overline{\partial}_t e^n, \overline{\partial}_t e^n) - \frac{1}{2}(\mathcal{B}_{\Delta t}\overline{\partial}_t e^1, \overline{\partial}_t e^1).\end{aligned} \tag{8.53}$$

Apply the inequality ($|ab| \le (a^2+b^2)/2$) to the last term in (8.52). Then utilizing (8.53), one can obtain the following inequality:

$$\begin{aligned}&\sum_{j=2}^{n} \|\overline{\partial}_t e^j\|^2 \Delta t + (Ae^n, e^n) + \Delta t^2 (\mathcal{B}_{\Delta t} \overline{\partial}_t e^n, \overline{\partial}_t e^n) \\ &\quad \le \sum_{j=2}^{n} \|\delta^j\|^2 \Delta t + (Ae^1, e^1) + \Delta t^2 (\mathcal{B}_{\Delta t} \overline{\partial}_t e^1, \overline{\partial}_t e^1), \quad n \ge 2.\end{aligned} \tag{8.54}$$

Thus, the estimation of the error generated by the ADI-II method is, in the commutative case, reduced to bounding the errors in z^0 and z^1, thereby emphasizing the remarks above on the evaluation of z^1. Try to compare the above analysis with (8.16) when $\theta = 1/2$.

8.3.6. Accuracy and efficiency of ADI-II

To check the accuracy and efficiency of the ADI-II algorithm, let us choose the domain $\Omega = (0,1)^2$ and the time interval $J = (0,1]$, along with the four diffusion coefficients

$$\begin{aligned} a_1(x,y) &= 1, \\ a_2(x,y) &= 1/(2+\cos(3\pi x)\cdot\cos(2\pi y)), \\ a_3(x,y) &= \begin{cases} 1+0.5\cdot\sin(5\pi x)+y^3, & \text{if } x \le 0.5, \\ 1.5/(1+(x-0.5)^2)+y^3, & \text{else}, \end{cases} \\ a_4(x,y) &= \begin{bmatrix} a_2(x,y) & 0 \\ 0 & a_3(x,y) \end{bmatrix}. \end{aligned} \tag{8.55}$$

- The first time step to obtain z^1 for the ADI-II was made by following the w^1-ADI calculation by SOR iterations to get the Crank-Nicolson value.
- Here, we compare the results of four different algorithms, namely the LU-based, PCG-ILU0, ADI, and ADI-II methods.

	$a = a_1$		$a = a_2$		$a = a_3$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
LU-based	23.6	1.10e-3	27.2	3.52e-3	24.2	5.35e-3
PCG-ILU0	21.6	1.09e-3	24.0	3.52e-3	24.7	5.36e-3
ADI	7.14	1.70e-2	10.9	1.02e-2	7.91	2.67e-2
ADI-II	7.77	1.10e-3	11.3	3.54e-3	8.46	5.35e-3

Table 8.3: The performances of the LU-based, PCG-ILU0, ADI, and ADI-II methods with $c = \alpha_2 \equiv 0$, $\nu_t = 1$, $\nu_x = 4$, $\nu_y = 3$, $n_x = n_y = n_t = 100$ for $u = u_\times$.

	$\Delta t = 2h$		$\Delta t = h$		$\Delta t = h/2$		$\Delta t = h/4$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
LU-based	28.4	2.12e-3	49.6	2.13e-3	92.1	2.13e-3	176	2.13e-3
PCG-ILU0	24.9	2.14e-3	36.5	2.15e-3	57.6	2.14e-3	96.8	2.13e-3
ADI	8.19	2.01e-1	16.3	6.76e-2	32.4	1.75e-2	64.5	4.86e-3
ADI-II	8.80	1.10e-2	16.9	2.17e-3	33.2	2.13e-3	66.1	2.13e-3

Table 8.4: The performances of the LU-based, PCG-ILU0, ADI, and ADI-II methods with $a = a_4$, $c = \alpha_2 \equiv 0$, $\nu_t = 2.0$, $\nu_x = 6.25$, $\nu_y = 7$, $h = h_x = h_y = 1/120$, and $u = u_\times$.

Table 8.3 presents the performances of the four algorithms for the first three diffusion coefficients in (8.55) for $u = u_\times$ with $\nu_t = 1$, $\nu_x = 4$, and $\nu_y = 3$. The error for the ADI method is 16, 3, and 5 times larger than the Crank-Nicolson error for $a = a_1,\ a_2$, and a_3, respectively. The ADI-II method requires only about 5-7% extra cost over the ADI method and its accuracy hardly differs from that of the direct, LU-based solver, when $\Delta t \le h$.

Table 8.4 shows numerical results for various time steps, when $a = a_4$ (an anisotropic diffusivity), $c = \alpha_2 \equiv 0$, $\nu_t = 2$, $\nu_x = 6.25$, and $\nu_y = 7$, and $h = h_x = h_y = 1/120$. The ADI calculations show large splitting errors, even for small time steps. Here again the improved initialization (8.42) greatly improves the accuracy of the alternating direction procedure, for a few percent of extra cost. However, as one can see from the table, the ADI-II algorithm generates a splitting error that is a few times the Crank-Nicolson error for $\Delta t = 2h$. Thus one has to choose Δt sufficiently small, although the splitting error is $\mathcal{O}(\Delta t^3)$.

8.4. Homework

1. Show that all of (8.19), (8.20), and (8.23) are equivalent to each other. Count and compare the required operations for (8.20) and (8.23) in each time level.
2. Show that (8.28) is equivalent to (8.29)-(8.30), for $m = 3$.
3. Check if (8.37) is equivalent to (8.43), when $m = 2$. Count to compare the required operations for them.
4. The given code in Matlab is an implementation for the ADI (8.20) solving the heat equation in 2D. Adjust the code for ADI-II (8.37) with $m = 2$.
   (a) The major step you should fulfill is to adjust `F` in `xy_sweeps.m`.
   (b) Perform error analysis comparing errors from ADI and ADI-II.
   (c) Report your additions to the code.

CHAPTER 9

Domain Decomposition Methods

The development of high-performance parallel computers has promoted the effort to search for new efficient parallel algorithms for scientific computation rather than parallelize existing sequential algorithms. In the last two decades, domain decomposition (DD) methods have been studied extensively for the numerical solution of PDEs.

Contents of Chapter 9

9.1. Introduction to DDMs 302
9.2. Overlapping Schwarz Alternating Methods (SAMs) 305
9.3. Nonoverlapping DDMs 313
9.4. Iterative DDMs Based on Transmission Conditions 320
9.5. Homework 330

9.1. Introduction to DDMs

The earliest DD method for elliptic problems is the alternating method discovered by Hermann A. Schwarz in 1869 [60], so it is called *Schwarz alternating method* (SAM).

Schwarz used the method to establish the existence of harmonic functions on the nonsmooth domains that were constructed as a union of regions where the existence could be established by some other methods; see Figure 9.1.

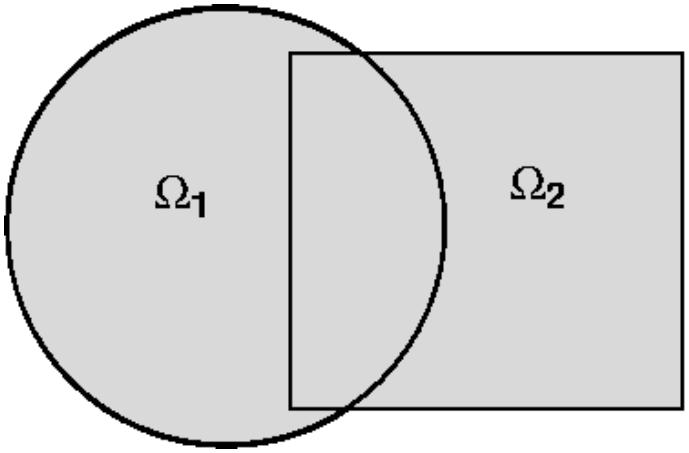


Figure 9.1: The domain used by Schwarz to show the existence of harmonic solutions on irregular domains.

- Indeed, for a given initial value, SAM provided a convergent sequence with a limit that is the harmonic function satisfying the given boundary condition.
- Each iteration of the method consists of two fractional steps.
  - In the first step, the previous approximation on Ω_1 is replaced by the harmonic function for which the Dirichlet data on $\widetilde{\Gamma}_{12}$ $(:= \partial\Omega_1 \cap \Omega_2)$ is given by the previous approximation on Ω_2.
  - The second step, in which new approximation is obtained on Ω_2, is carried out similarly.
- Therefore, an arbitrarily accurate approximation of the harmonic function in the domain $\Omega_1 \cup \Omega_2$ can be computed by using only solvers for circles and rectangles. The method of separation of variables can be used for the solution of these subdomains.

SAM: Historical Backgrounds

- SAM offers a process that can be carried out by a series of fast solvers on relatively smooth subdomains.
- Over last two decades, Schwarz's idea has been extensively applied to various problems defined on general domains.
- It has offered a possibility of efficient numerical algorithms for poorly-conditioned large-scale problems and of parallelism for the very large systems of linear or nonlinear algebraic equations that arise from discretizations of elliptic problems in fluid dynamics, elasticity, wave propagation, and other important areas.
- The main question for the classical SAM and its modern extensions has been to show that the convergence rate of the iteration is satisfactory and that it is independent or grows slowly when the mesh is to be refined and/or when the number of subdomains increases.
- It is not surprising that reducing the amount of overlap without a deterioration of the convergence rate has become an important issue in theoretical analyses and numerical simulations using SAM.

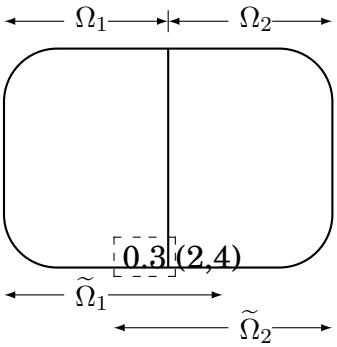


Figure 9.2: Nonoverlapping and overlapping partitions of Ω.

9.2. Overlapping Schwarz Alternating Methods (SAMs)

9.2.1. Variational formulation

Let Ω be a bounded domain in $\mathbb{R}^d$, $d \leq 3$, with Lipschitz boundary $\Gamma = \partial\Omega$.

Consider the following elliptic problem with a homogeneous Dirichlet boundary condition: Find $u \in V = H_0^1(\Omega)$ such that

$$\begin{aligned} Lu := -\nabla \cdot (a(\mathbf{x})\nabla u) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ u &= 0, \qquad\quad \mathbf{x} \in \Gamma, \end{aligned} \tag{9.1}$$

where we assumed that $0 < a_* \leq a(\mathbf{x}) \leq a^* < \infty$.

The problem (9.1) in its variational form reads

$$a(u, v) = (f, v), \quad v \in V, \tag{9.2}$$

where

$$a(u, v) = \int_\Omega a\nabla u \cdot \nabla v d\mathbf{x}, \quad (f, v) = \int_\Omega fv \, d\mathbf{x}.$$

9.2.2. SAM with two subdomains

In the simplest form, SAM decomposes the original domain into two overlapping subdomains $\widetilde{\Omega}_1$ and $\widetilde{\Omega}_2$; see Figure 9.2. Let

$$\widetilde{V}_j = \{v \in V : v = 0 \quad \text{on } \Omega \setminus \overline{\widetilde{\Omega}_j}\}, \quad j = 1, 2.$$

Then, $\widetilde{V}_j$ are subspaces of V and $V = \widetilde{V}_1 + \widetilde{V}_2$. Let an initial guess $u^0 = \{u_1^0, u_2^0\} \in V$ be given. Then, the iterate $u^n \in V$ is determined from u^{n-1} by sequentially solving

$$\begin{array}{lll}
\text{(a)} & Lu_1^{n-1/2} = f, & \text{in } \widetilde{\Omega}_1, \\
\text{(b)} & u_1^{n-1/2} = 0, & \text{on } \widetilde{\Gamma}_1, \\
\text{(c)} & u_1^{n-1/2} = u_2^{n-1}, & \text{on } \widetilde{\Gamma}_{12}, \\
\text{(d)} & Lu_2^{n} = f, & \text{in } \widetilde{\Omega}_2, \\
\text{(e)} & u_2^{n} = 0, & \text{on } \widetilde{\Gamma}_2, \\
\text{(f)} & u_2^{n} = u_1^{n-1/2}, & \text{on } \widetilde{\Gamma}_{21},
\end{array} \tag{9.3}$$

where $\widetilde{\Gamma}_j = \partial\widetilde{\Omega}_j \cap \partial\Omega$ and $\widetilde{\Gamma}_{jk} = \partial\widetilde{\Omega}_j \cap \Omega_k$.

- This *multiplicative* Schwarz method solves at each iteration a series of smaller problems restricted on subdomains.
- These subproblems require an additional boundary condition on the interior (artificial) boundaries $\widetilde{\Gamma}_{jk}$.
- The Schwarz method is easy to implement and can be applied to more general elliptic differential operators and domains.

9.2.3. Convergence analysis

Let us consider the error propagation operator of (9.3); see [47, 70] for details. In (9.3), one may extend $u_1^{n-1/2}$ by u_2^{n-1} on Ω_2 and u_2^n by $u_1^{n-1/2}$ on Ω_1. In the variational form, (9.3) reads

$$\begin{aligned} a(u_1^{n-1/2}, v) &= (f, v), \quad v \in \widetilde{V}_1, \quad u_1^{n-1/2} - u^{n-1} \in \widetilde{V}_1, \\ a(u_2^{n}, v) &= (f, v), \quad v \in \widetilde{V}_2, \quad u_2^{n} - u^{n-1/2} \in \widetilde{V}_2. \end{aligned} \tag{9.4}$$

Since

$$(f, v) = a(u, v), \quad v \in \widetilde{V}_j, \quad j = 1, 2,$$

one can rewrite (9.4) as

$$\begin{aligned} a(u_1^{n-1/2} - u^{n-1}, v) &= a(u - u^{n-1}, v), \quad v \in \widetilde{V}_1, \quad u_1^{n-1/2} - u^{n-1} \in \widetilde{V}_1, \\ a(u_2^{n} - u^{n-1/2}, v) &= a(u - u^{n-1/2}, v), \quad v \in \widetilde{V}_2, \quad u_2^{n} - u^{n-1/2} \in \widetilde{V}_2. \end{aligned} \tag{9.5}$$

It is easy and convenient to describe the method in terms of two projections P_j, $j = 1, 2$, onto $\widetilde{V}_j$, defined by

$$a(P_j v, w) = a(v, w), \quad \forall\, w \in \widetilde{V}_j.$$

Then, (9.5) obviously means

$$\begin{aligned} u^{n-1/2} - u^{n-1} &= P_1(u - u^{n-1}), \\ u^{n} - u^{n-1/2} &= P_2(u - u^{n-1/2}), \end{aligned}$$

or equivalently

$$\begin{aligned} u - u^{n-1/2} &= (I - P_1)\,(u - u^{n-1}), \\ u - u^{n} &= (I - P_2)\,(u - u^{n-1/2}), \end{aligned}$$

where I is the identity operator. Therefore, the error propagates as

$$u - u^n = (I - P_2)\,(I - P_1)\,(u - u^{n-1}). \tag{9.6}$$

Domain Decomposition for FEMs: Now, let V^h be the piecewise linear FE subspace of V corresponding to a regular triangulation $\mathcal{T}_h$. Then the FE method for the variational problem (9.2) can be formulated as follows: Find $u^h \in V^h$ such that

$$a(u^h, v^h) = (f, v^h), \quad v^h \in V^h. \tag{9.7}$$

The FE procedure corresponding to the DDM (9.3) is formulated by finding iterates $\{u^{n-1/2},\, u^n\}$ from V^h. One can consider analogous projections P_j, $j = 1, 2$, onto $\widetilde{V}_j^h$ $(:= \widetilde{V}_j \cap V^h)$ for FE methods. Then, the error for the FE methods propagates as

$$u^h - u^{h,n} = (I - P_2)\,(I - P_1)\,(u^h - u^{h,n-1}). \tag{9.8}$$

So, the FE formulation of (9.3) can be viewed as an iterative method for solving

$$(P_1 + P_2 - P_2P_1)\,u^h = g^h, \tag{9.9}$$

with an appropriate right hand side g^h. Here the upshot/hope is that the condition number of $(P_1 + P_2 - P_2P_1)$ is much smaller than that of the original algebraic system.

Notes

- The multiplicative Schwarz method has an important variant, i.e., the *additive* Schwarz method which decouples the subproblems (9.3.a)-(9.3.c) and (9.3.d)-(9.3.f). In additive Schwarz method, (9.3.f) is replaced by

$$u_2^n = u_1^{n-1}, \quad \text{on } \widetilde{\Gamma}_{21};$$

  the additive algorithm is a simple iterative method for solving

$$(P_1 + P_2)u^h = g_0^h, \tag{9.10}$$

  for some g_0^h; see Exercise 9.1.
- Such Schwarz methods can be generalized immediately to any number of overlapping subdomains $\widetilde{\Omega}_j$ expanded from the original nonoverlapping subdomains Ω_j, $j = 1, 2, \cdots, M$.

9.2.4. Coarse subspace correction

Let H_j measure the size of Ω_j and

$$H = \max_{j=1,\cdots,M} H_j.$$

It is known that a DD preconditioner for which the new iterate is updated by the former solutions on local subregions of diameter on the order of H has a condition number which grows at least as fast as $1/H^2$; see [19] and references therein.

To overcome this difficulty, one can introduce the *coarse subspace correction* technique as a preconditioner. Then, our FE space is represented as the sum of $M+1$ subspaces

$$V^h = V_0^h + \widetilde{V}_1^h + \cdots + \widetilde{V}_M^h, \tag{9.11}$$

where $V_0^h = V^H$, the piecewise linear FE space on the coarse mesh defined by the nonoverlapping partition $\{\Omega_j\}$. (We have implicitly assumed that each subdomain is triangle.)

The corresponding additive algorithm can be viewed as an iterative method for solving

$$Pu^h = (P_0 + P_1 + \cdots + P_M)u^h = G^h, \tag{9.12}$$

for an appropriate G^h, where P_0 is the projection from V^h to V^H.

Known: Let $\lambda_* > 0$ and $\lambda^* > 0$ be the minimum and the maximum eigenvalues for a symmetric positive definite (SPD) matrix A, respectively. The condition number of A, $\kappa(A)$, is defined by

$$\kappa(A) = \lambda^*/\lambda_*.$$

The required iteration number for the CG method to solve SPD systems is $\mathcal{O}\left(\sqrt{\kappa(A)}\right)$ for a given accuracy. (For more general systems, GMRES [59] and QMR [24] can be used.) The following result was established by Dryja and Widlund [19].

Theorem 9.1. *Let* $\delta = \min_{j=1,\cdots,M} \text{dist}(\partial\Omega_j \setminus \partial\Omega, \partial\widetilde{\Omega}_j \setminus \partial\Omega) > 0$. *Assume the problem coefficient* a *is continuous on* $\bar{\Omega}$. *Then, the condition number of the additive Schwarz method for solving* (9.12) *satisfies*

$$\kappa(P) \leq C(1 + H/\delta), \tag{9.13}$$

where C *is independent of* H, h, *and* δ.

If there is no coarse subspace correction, (9.13) must be replaced by (see [45])

$$\kappa(P) \leq C\left(1 + \frac{1}{H_{\min}^2}\frac{H}{\delta}\right),$$

where $H_{\min}$ is the minimum diameter of the subdomains.

Final Notes

- Introducing a global solver at a modest cost is the key to efficiency of iterative algorithms.
- On the other hand, if the overlap is a fraction of H, the condition number in (9.13) is bounded uniformly by a constant.
- In numerical simulations, however, the requirement on the amount of overlap may degrade the algorithm due to a heavy cost of local solvers. Consider the algorithm with a small overlap. The number of CG iterations is higher in such a case, but this can be compensated for by cheaper local problem solvers.
- The condition number for DD methods incorporating a small overlap together with a coarse subspace solver is often bounded by

$$\kappa(P) \leq C(1 + \log(H/h))^r, \quad r = 2, 3, \text{ or } 4, \tag{9.14}$$

  where r depends on the amount of overlap and the regularity of the diffusion coefficient a.
- The convergence analysis of Schwarz method is more complicated when the subdomains overlap less. See [47] and the survey papers [19, 45] for details.

9.3. Nonoverlapping DDMs

9.3.1. Multi-domain formulation

Recall the model problem: Find $u \in V = H_0^1(\Omega)$ such that

$$\begin{aligned} Lu := -\nabla \cdot (a(\mathbf{x})\nabla u) = f(\mathbf{x}), \quad & \mathbf{x} \in \Omega, \\ u = 0, \quad & \mathbf{x} \in \Gamma, \end{aligned} \tag{9.15}$$

where we assumed that $0 < a_* \leq a(\mathbf{x}) \leq a^* < \infty$.

Consider a nonoverlapping partition $\{\Omega_j : j = 1, 2, \cdots, M\}$ of Ω:

$$\begin{aligned} &\overline{\Omega} = \cup_{j=1}^M \overline{\Omega}_j; \quad \Omega_j \cap \Omega_k = \emptyset, \ \ j \neq k; \\ &\Gamma_j = \Gamma \cap \partial\Omega_j; \ \ \Gamma_{jk} = \Gamma_{kj} = \partial\Omega_j \cap \partial\Omega_k. \end{aligned}$$

Let u_j denote the restriction of u to Ω_j.

Then, the problem (9.15) can be formulated as follows: Find $\{u_j\}$ such that

$$\begin{aligned}
&\text{(a)} \quad Lu_j = f, && \mathbf{x} \in \Omega_j, \\
&\text{(b)} \quad u_j = 0, && \mathbf{x} \in \Gamma_j, \\
&\text{(c)} \quad u_j = u_k, && \mathbf{x} \in \Gamma_{jk}, \\
&\text{(d)} \quad \frac{\partial u_j}{\partial \nu_{L,j}} = -\frac{\partial u_k}{\partial \nu_{L,k}}, && \mathbf{x} \in \Gamma_{jk},
\end{aligned} \tag{9.16}$$

where the conormal derivative is defined as

$$\frac{\partial u_j}{\partial \nu_{L,j}} = a \nabla u_j \cdot \mathbf{n}_j,$$

where $\mathbf{n}_j$ indicates the unit outer normal from $\partial \Omega_j$.

- Equations (9.16.c)-(9.16.d) are the *transmission conditions* which impose the continuity of the solution and its conormal fluxes on the subdomain interfaces.
- Nonoverlapping DDMs can be characterized depending on how the transmission conditions are incorporated in the iteration procedure.

We first introduce the *Steklov-Poincaré* operator which is useful for the convergence analysis for the variational formulation of the DDMs.

9.3.2. The Steklov-Poincaré operator

Let λ_{jk} be the unknown value of u on Γ_{jk}. Consider the following Dirichlet problems:

$$\begin{aligned} Lw_j &= f, \quad \mathbf{x} \in \Omega_j, \\ w_j &= 0, \quad \mathbf{x} \in \Gamma_j, \\ w_j &= \lambda_{jk}, \quad \mathbf{x} \in \Gamma_{jk}, \end{aligned} \tag{9.17}$$

for $j = 1, \cdots, M$. Then, we can state that

$$w_j = u_j^0 + u_j^*, \tag{9.18}$$

where $\{u_j^0\}$ and $\{u_j^*\}$ are defined as the solutions of

$$\begin{aligned} Lu_j^0 &= 0, \quad \mathbf{x} \in \Omega_j, \\ u_j^0 &= 0, \quad \mathbf{x} \in \Gamma_j, \\ u_j^0 &= \lambda_{jk}, \quad \mathbf{x} \in \Gamma_{jk}, \end{aligned} \tag{9.19}$$

and

$$\begin{aligned} Lu_j^* &= f, \quad \mathbf{x} \in \Omega_j, \\ u_j^* &= 0, \quad \mathbf{x} \in \Gamma_j, \\ u_j^* &= 0, \quad \mathbf{x} \in \Gamma_{jk}, \end{aligned} \tag{9.20}$$

Note that when $a(\mathbf{x}) = 1$, u_j^0 is the *harmonic extension* of $\{\lambda_{jk}\}$ (for k's such that $\Gamma_{jk} \neq \emptyset$) into Ω_j; for general coefficients, we still call it the harmonic extension and denote by $H_j\lambda_{jk}$. We will also write $G_j f$ instead of u_j^*, $j = 1, \cdots, M$.

It follows from comparing (9.16) with (9.17) that

$$\begin{aligned} &\left\{u_j = w_j,\ \ \forall\, j = 1, \cdots, M\right\} \\ &\iff \left\{\frac{\partial w_j}{\partial \nu_{L,j}} = -\frac{\partial w_k}{\partial \nu_{L,k}},\ \ \forall\, j,k \text{ such that } \Gamma_{jk} \neq \emptyset\right\}. \end{aligned} \tag{9.21}$$

The latter condition equivalently amounts to the requirement that each of $\{\lambda_{jk}\}$ satisfies the *Steklov-Poincaré interface equation*

$$S_{jk}\lambda_{jk} = \chi_{jk}, \tag{9.22}$$

where $S = \{S_{jk}\}$ is the *Steklov-Poincaré* operator defined as

$$S_{jk}\eta = \frac{\partial}{\partial \nu_{L,j}} H_j \eta + \frac{\partial}{\partial \nu_{L,k}} H_k \eta, \tag{9.23}$$

for η defined on $\Gamma_{jk}(\neq \emptyset)$, and

$$\chi_{jk} = -\left(\frac{\partial}{\partial \nu_{L,j}} G_j f + \frac{\partial}{\partial \nu_{L,k}} G_k f\right). \tag{9.24}$$

The operator S is symmetric, positive definite (coercive), and continuous.

Here the goal is to find $\{\lambda_{jk}\}$ such that $\lambda_{jk} = u\big|_{\Gamma_{jk}}$, which must satisfy (9.22). Some DDMs update the iterates $\{\lambda_{jk}^n\}$ by iteratively solving (9.22), of which each step solves the subproblems in (9.19) and (9.20). The process can be understood easily by considering the algebraic system of the *discrete* Steklov-Poincaré operator, which is known as the *Schur complement matrix*.

9.3.3. The Schur complement matrix

Consider the FE method for the variational form (9.7). Let N_j denote the number of interior nodes in Ω_j, $j = 1, 2, \cdots, M$, and N_B be the number of nodal points on $\cup\Gamma_{jk}$. Thus the total number of nodes are $N_1 + \cdots + N_M + N_B$. We order the interior nodes of $\{\Omega_j\}$'first and those on $\cup\Gamma_{jk}$ next. Then, the algebraic system of (9.7) can be written as

$$A\mathbf{u} := \begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{u}_I \\ \mathbf{u}_B \end{bmatrix} = \begin{bmatrix} \mathbf{f}_I \\ \mathbf{f}_B \end{bmatrix}, \tag{9.25}$$

where A_{II} is a block diagonal matrix and $A_{BI} = A_{IB}^T$:

$$\begin{aligned} A_{II} &= \mathbf{diag}(A_{11}, A_{22}, \cdots, A_{MM}), \\ A_{BI} &= (A_{B1}, A_{B2}, \cdots, A_{BM}). \end{aligned}$$

Here the sr-th entry of A_{jj}, the ℓr-th entry of A_{Bj}, and the ℓm-th entry of A_{BB} are given by

$$\begin{aligned} (A_{jj})_{sr} &= a_j(\varphi_r^{(j)}, \varphi_s^{(j)}), && s, r = 1, \cdots, N_j, \\ (A_{Bj})_{\ell r} &= a_j(\varphi_r^{(j)}, \varphi_\ell^{(B)}), && \ell = 1, \cdots, N_B, \quad r = 1, \cdots, N_j, \\ (A_{BB})_{\ell m} &= \sum_j a_j(\varphi_m^{(B)}, \varphi_\ell^{(B)}), && \ell, m = 1, \cdots, N_B, \end{aligned}$$

where $a_j(\cdot, \cdot)$ is the restriction of $a(\cdot, \cdot)$ to Ω_j, and $\varphi_s^{(j)}$ and $\varphi_\ell^{(B)}$ are the basis functions associated with nodes lying in Ω_j and $\cup\Gamma_{jk}$, respectively.

By eliminating all degrees of freedom that are associated with interior nodes of subdomains, (9.25) reduces to the following interface problem:

$$\Sigma \mathbf{u}_B = \mathbf{f}_B - A_{IB}^T A_{II}^{-1} \mathbf{f}_I, \tag{9.26}$$

where Σ is the *Schur complement matrix* defined as

$$\Sigma = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB}.$$

The matrix Σ is exactly the algebraic counterpart of the discrete Steklov-Poincaré operator; it can be proved symmetric positive definite, as for the Steklov-Poincaré operator.

In early substructuring techniques of the 1960's, the interface problem (9.26) was solved by a direct solver (for which a frontal method was often employed mainly due to insufficient computer memory). Most of the recent iterative nonoverlapping DDMs can be explained as preconditioning techniques for solving the interface problem by the CG method.

Each matrix-vector multiplication with Σ involves M subdomain solves, i.e.,

$$A_{II}^{-1} = \text{diag}(A_{11}^{-1}, \cdots, A_{MM}^{-1}),$$

which can be carried out in parallel.

Convergence

- As reported in Le Tallec [45], the condition number of Σ is bounded as

$$\kappa(\Sigma) \leq C \frac{H}{h H_{\min}^2},$$

  where H and $H_{\min}$ are respectively the maximum and minimum diameters of the subdomains.
- Thus a mathematical challenge is to construct a preconditioner for Σ such that the convergence rate of the preconditioned iterative method becomes independent on both h and H.
- However, in practice the incorporation of such an optimal preconditioner may not imply that the resulting algorithm is fastest in computation time. We refer interested readers to Quarteroni and Valli [57].

9.4. Iterative DDMs Based on Transmission Conditions

9.4.1. The Dirichlet-Neumann method

As it is called, some subproblems are solved using Dirichlet data on the interfaces and the others use Neumann data. We may separate the subdomains into two groups by a **red-black coloring**.

Let I_R and I_B be respectively the indices of the red and black subdomains. Then, the method is formulated as follows: For given $\{\lambda_{jk}^0\}$, find $\{u_j^n\}$, $n \geq 1$, by recursively solving

$$
\begin{aligned}
&\text{(a)} \quad \begin{cases} Lu_j^n = f, & \mathbf{x} \in \Omega_j, \\ u_j^n = 0, & \mathbf{x} \in \Gamma_j, \\ u_j^n = \lambda_{jk}^{n-1}, & \mathbf{x} \in \Gamma_{jk}, \end{cases} \qquad j \in I_B, \\
&\text{(b)} \quad \begin{cases} Lu_j^n = f, & \mathbf{x} \in \Omega_j, \\ u_j^n = 0, & \mathbf{x} \in \Gamma_j, \\ \dfrac{\partial u_j^n}{\partial \nu_{L,j}} = -\dfrac{\partial u_k^n}{\partial \nu_{L,k}}, & \mathbf{x} \in \Gamma_{jk}, \end{cases} \qquad j \in I_R, \\
&\text{(c)} \quad \lambda_{jk}^n = \theta_{jk}\, u_{j,R}^n + (1-\theta_{jk})\lambda_{jk}^{n-1},
\end{aligned}
\tag{9.27}
$$

where $\{\theta_{jk}\} > 0$ is an acceleration parameter and $u_{j,R}^n$ denotes the solution from the subdomains colored red.

The acceleration parameter is often set less than one; the method without relaxation (i.e., $\theta_{jk} \equiv 1$) is not necessarily convergent, unless special assumptions are made on the size of the subdomains. We refer readers interested in the Dirichlet-Neumann method to [4, 6, 52] and [57] for details.

9.4.2. The Neumann-Neumann method

This method requires solving the subproblems twice, one with Dirichlet-Dirichlet data and the other with Neumann-Neumann data: For given $\{\lambda_{jk}^0\}$, find $\{u_j^n\}$, $n \geq 1$, satisfying

$$
\begin{aligned}
&\text{(a)} \quad \begin{cases} Lu_j^n = f, & \mathbf{x} \in \Omega_j, \\ u_j^n = 0, & \mathbf{x} \in \Gamma_j, \\ u_j^n = \lambda_{jk}^{n-1}, & \mathbf{x} \in \Gamma_{jk}, \end{cases} \\
&\text{(b)} \quad \begin{cases} Lv_j^n = 0, & \mathbf{x} \in \Omega_j, \\ v_j^n = 0, & \mathbf{x} \in \Gamma_j, \\ \dfrac{\partial v_j^n}{\partial \nu_{L,j}} = \dfrac{\partial u_j^n}{\partial \nu_{L,j}} + \dfrac{\partial u_k^n}{\partial \nu_{L,k}}, & \mathbf{x} \in \Gamma_{jk}, \end{cases} \\
&\text{(c)} \quad \lambda_{jk}^n = \lambda_{jk}^{n-1} - \theta_{jk} \left(\sigma_{jk} v_j^n + (1 - \sigma_{jk}) v_k^n \right) \big|_{\Gamma_{jk}}, \quad j > k,
\end{aligned}
\tag{9.28}
$$

where $\{\theta_{jk}\} > 0$ is again an acceleration parameter and $\{\sigma_{jk}\}$ is an averaging coefficient.

The Neumann-Neumann method was studied in [1, 5, 12, 50]. It is known that the method is efficient when the subdomains are similar [45]. The resulting condition number (without a coarse grid solver) has been shown to be [12]

$$
\kappa(M^{-1}A) \leq \frac{C}{H^2} \left(1 + \log \frac{H}{h} \right)^2,
$$

where M is the Neumann-Neumann preconditioning matrix for A.

9.4.3. The Robin method

The method was first suggested by Lions [48] and has been applied to various physical problems with a great efficiency; see e.g. [13, 17, 36, 38, 41, 42, 53].

For given $\{u_j^0\}$, find $\{u_j^n\}$, $n \geq 1$, satisfying

$$\begin{aligned} &\text{(a)} \quad Lu_j^n = f, && \mathbf{x} \in \Omega_j, \\ &\text{(b)} \quad u_j^n = 0, && \mathbf{x} \in \Gamma_j, \\ &\text{(c)} \quad \frac{\partial u_j^n}{\partial \nu_{L,j}} + \theta_{jk} u_j^n = -\frac{\partial u_k^{n-1}}{\partial \nu_{L,k}} + \theta_{jk} u_k^{n-1}, && \mathbf{x} \in \Gamma_{jk}, \end{aligned} \tag{9.29}$$

where $\{\theta_{jk}\} \geq 0$ is an acceleration parameter with

$$\theta_{jk} + \theta_{kj} > 0.$$

Lions [48] proved the convergence of the method through an energy estimate on the interfaces.

Note that (9.29.c) is defined twice on each of Γ_{jk} from both sides of the interface:

$$\begin{aligned} \frac{\partial u_j^n}{\partial \nu_{L,j}} + \theta_{jk} u_j^n &= -\frac{\partial u_k^{n-1}}{\partial \nu_{L,k}} + \theta_{jk} u_k^{n-1}, \\ \frac{\partial u_k^n}{\partial \nu_{L,k}} + \theta_{kj} u_k^n &= -\frac{\partial u_j^{n-1}}{\partial \nu_{L,j}} + \theta_{kj} u_j^{n-1}. \end{aligned}$$

When the iterates converge, the limit $\{u_j\}$ would satisfy the above equations in the same way (without the superscripts n and $n-1$). By subtracting and adding the equations, one can get the transmission conditions (9.16.c)-(9.16.d).

9.4.4. Remarks on DDMs of transmission conditions

- The DDMs based on transmission conditions ((9.27), (9.28), and (9.29)) require to choose appropriate acceleration parameters to either guarantee or accelerate convergence. However, there is no guide line to be applied to various problems; finding the acceleration parameter is problematic.
- For the Robin method applied, Kim [37, 44] suggested an automatic way of choosing the acceleration parameter to solve the Helmholtz wave problem.

- A very important accuracy issue is related to the *discrete* transmission conditions. Recall that the standard discretization methods such as the FD and FE methods allow the conormal flux to be discontinuous at the element interfaces.
- Since the transmission conditions impose the continuity of both the solution and its conormal flux on the subdomain interfaces, there will be a *flux conservation error*, i.e., the discrete solution u^h would not satisfy (9.16.c)-(9.16.d) unless it is linear across the subdomain interfaces.

Flux conservation error

- In practice, the flux conservation error can severely deteriorate accuracy of the computed solution.
- Thus the conormal flux must be treated with a special care, in particular, when the DDM is to be utilized as the main solver.
- When the DDM is used as a preconditioner, i.e., another algorithm such as a Krylov subspace method is applied as an outer iteration, the flux conservation error may affect the convergence speed of the resulting algorithm; however, the required accuracy of the solution can be achieved by the main solver (the outer iteration).

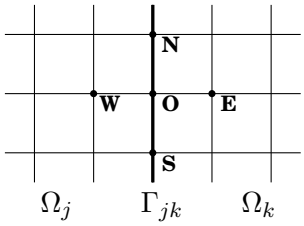


Figure 9.3: The five point stencil at a grid point on the interface Γ_{jk}.

Discretization of the Robin boundary condition: To illustrate a way of dealing with the conormal flux, consider the Robin method applied to the Poisson equation, $L = -\Delta$:

$$\begin{aligned} &\text{(a)} && -\Delta u_j^n = f, && \mathbf{x} \in \Omega_j, \\ &\text{(b)} && u_j^n = 0, && \mathbf{x} \in \Gamma_j, \\ &\text{(c)} && \frac{\partial u_j^n}{\partial \nu_j} + \beta u_j^n = -\frac{\partial u_k^{n-1}}{\partial \nu_k} + \beta u_k^{n-1}, && \mathbf{x} \in \Gamma_{jk}, \end{aligned} \tag{9.30}$$

where $\beta > 0$ is a constant acceleration parameter.

Let the domain be discretized into uniform cells of edge size h and the subdomain interfaces $\{\Gamma_{jk}\}$ coincide with parts of grid lines. Let $\partial_{\mathbf{b},jk} u_j$ and $\partial_{\mathbf{f},jk} u_j$ be the backward and forward differences for $\partial u_j / \partial \nu_j$ on Γ_{jk}, respectively. For example, at the nodal point $\mathbf{o} \in \Gamma_{jk}$ in Figure 9.3, they are defined as

$$\begin{aligned} \partial_{\mathbf{b},jk} u_j(\mathbf{o}) &= (u_j(\mathbf{o}) - u_j(\mathbf{w}))/h, \quad \partial_{\mathbf{f},jk} u_j(\mathbf{o}) = (u_j(\mathbf{e}) - u_j(\mathbf{o}))/h, \\ \partial_{\mathbf{b},kj} u_k(\mathbf{o}) &= (u_k(\mathbf{o}) - u_k(\mathbf{e}))/h, \quad \partial_{\mathbf{f},kj} u_k(\mathbf{o}) = (u_k(\mathbf{w}) - u_k(\mathbf{o}))/h. \end{aligned}$$

(Here we have employed an exterior bordering of the subdomains.)

Let $\Delta_h u_j$ be the central five-point difference approximation of Δu_j. Then the DD iterative algorithm in the FD formulation can be defined as follows: For given $\{u_j^0\}$, find $\{u_j^n\}$, $n \geq 1$, by recursively solving

$$\begin{aligned}
&\text{(a)} && -\Delta_h u_j^n = f, && \mathbf{x} \in \Omega_j, \\
&\text{(b)} && u_j^n = 0, && \mathbf{x} \in \Gamma_j, \\
&\text{(c)} && \partial_{\mathbf{f},jk}\, u_j^n + \beta\, u_j^n = -\partial_{\mathbf{b},kj}\, u_k^{n-1} + \beta\, u_k^{n-1}, && \mathbf{x} \in \Gamma_{jk}.
\end{aligned} \tag{9.31}$$

Note that (9.31.c) imposes the continuity of the discrete solution only, when the algorithm converges. Such a treatment of the Robin condition, a *forward-backward difference matching*, was introduced by Kim [36, 38] to enforce equivalence of the DD method to the original discrete problem of the multilinear FE methods.

Equivalence: In the following, we will check the equivalence of algorithm (9.31) to the original discrete problem. It suffices to consider the algebraic equations of (9.31) at interface grid points. At the point o (in Figure 9.3), the equation (9.31.a) reads

$$4\,u^n_{j,\mathbf{o}} - u^n_{j,\mathbf{E}} - u^n_{j,\mathbf{W}} - u^n_{j,\mathbf{S}} - u^n_{j,\mathbf{N}} = h^2 f_{\mathbf{o}}, \tag{9.32}$$

where $u^n_{j,\mathbf{o}} = u^n_j(\mathbf{o})$, the value of u^n_j at the point o, and the others are similarly defined.

The term $u^n_{j,\mathbf{E}}$ in (9.32) evaluated at a point out of the subdomain Ω_j can be substituted by using (9.31.c). Equation (9.31.c) is written as

$$\frac{u^n_{j,\mathbf{E}} - u^n_{j,\mathbf{o}}}{h} + \beta\, u^n_{j,\mathbf{o}} = \frac{u^{n-1}_{k,\mathbf{E}} - u^{n-1}_{k,\mathbf{o}}}{h} + \beta\, u^{n-1}_{k,\mathbf{o}},$$

or equivalently

$$u^n_{j,\mathbf{E}} - (1-\beta h)\, u^n_{j,\mathbf{o}} = u^{n-1}_{k,\mathbf{E}} - (1-\beta h)\, u^{n-1}_{k,\mathbf{o}}. \tag{9.33}$$

Adding (9.32) and (9.33) reads

$$[4 - (1-\beta h)]\, u^n_{j,\mathbf{o}} - u^n_{j,\mathbf{W}} - u^n_{j,\mathbf{S}} - u^n_{j,\mathbf{N}} = h^2 f_{\mathbf{o}} + u^{n-1}_{k,\mathbf{E}} - (1-\beta h)\, u^{n-1}_{k,\mathbf{o}}. \tag{9.34}$$

In the same manner, one can treat cross points arising in a box-type decomposition of the domain. When the algorithm converges, the limit would clearly satisfy the original algebraic equation

$$4u_{\mathbf{o}} - u_{\mathbf{E}} - u_{\mathbf{W}} - u_{\mathbf{S}} - u_{\mathbf{N}} = h^2 f_{\mathbf{o}},$$

which proves the equivalence of (9.31) to the original discrete problem.

- It should be noticed that the standard FE formulation of (9.30) fails to get the original discrete solution, unless the original solution is linear across the subdomain interfaces. The forward-backward difference matching can be incorporated into the FE formulation to overcome the difficulty. See Exercises 9.2 and 9.3.
- For FD schemes, the normal derivatives in (9.30) can be approximated by the central differences, without a failure for the original FD solution. However, the convergence speed of the iteration may matter.

9.5. Homework

1. Derive (9.10) for the additive Schwarz method for two overlapping subdomains.
2. Consider the bilinear FE method of grid size h on the unit square applied to the DD method (9.30): Given $\{u_j^{h,0}\}$, $u_j^{h,0} \in V_j^h := V^h\big|_{\Omega_j}$, $j = 1, \cdots, M$, find $\{u_j^{h,n}\}$, $n \geq 1$, satisfying

$$\begin{aligned}(\nabla u_j^{h,n}, \nabla v)_{\Omega_j} + \sum_k \langle \beta u_j^{h,n}, v\rangle_{\Gamma_{jk}} &= (f, v)_{\Omega_j} \\ + \sum_k \langle -\frac{\partial u_k^{h,n-1}}{\partial \nu_k}, v\rangle_{\Gamma_{jk}} + \sum_k \langle \beta u_k^{h,n-1}, v\rangle_{\Gamma_{jk}}, &\quad v \in V_j^h.\end{aligned} \tag{9.35}$$

   (a) Show that the algebraic equation of (9.35) at the boundary nodal point **o** as given in Figure 9.3 reads

$$(2 + \beta h)\, u_{j,\mathbf{o}}^n - u_{j,\mathbf{w}}^n - \frac{1}{2} u_{j,\mathbf{s}}^n - \frac{1}{2} u_{j,\mathbf{N}}^n = \frac{h^2}{2} f_{\mathbf{o}} + u_{k,\mathbf{E}}^{n-1} - (1 - \beta h)\, u_{k,\mathbf{o}}^{n-1}, \tag{9.36}$$

   provided that the mass-lumping quadrature rule is used.

   (b) Show that (9.36) is equivalent to (9.34), in their limits, if the discrete solution is linear across the subdomain boundary Γ_{jk}.

3. A modification of (9.35) can be obtained incorporating the forward-backward difference matching (9.31.c) as follows: Given $\{u_j^{h,0}\}$, $u_j^{h,0} \in V_j^h$, $j = 1, \cdots, M$, find $\{u_j^{h,n}\}$, $n \geq 1$, satisfying

$$\begin{aligned}&(\nabla u_j^{h,n}, \nabla v)_{\Omega_j} + \sum_k \langle -\partial_{\mathbf{c},jk} u_j^{h,n}, v\rangle_{\Gamma_{jk}} = (f, v)_{\Omega_j}, \quad v \in V_j^h, \\ &\partial_{\mathbf{f},jk}\, u_j^n + \beta\, u_j^n = -\partial_{\mathbf{b},kj}\, u_k^{n-1} + \beta\, u_k^{n-1}, \quad \mathbf{x} \in \Gamma_{jk},\end{aligned} \tag{9.37}$$

   where $\partial_{\mathbf{c},jk} u_j^{h,n}$ is the central approximation of $\frac{\partial u_j^{h,n}}{\partial \nu_j}$, i.e., $\partial_{\mathbf{c},jk} = (\partial_{\mathbf{b},jk} + \partial_{\mathbf{f},jk})/2$. (We have assumed the outer bordering.) Equations (9.37) can be rewritten as

$$\begin{aligned}(\nabla u_j^{h,n}, \nabla v)_{\Omega_j} + \sum_k \langle \frac{1}{2}(-\partial_{\mathbf{b},jk} u_j^{h,n} + \beta\, u_j^n),\, v\rangle_{\Gamma_{jk}} &\\ = (f, v)_{\Omega_j} + \sum_k \langle \frac{1}{2}(-\partial_{\mathbf{b},kj} u_k^{h,n-1} + \beta\, u_k^{n-1}),\, v\rangle_{\Gamma_{jk}}, &\quad v \in V_j^h.\end{aligned} \tag{9.38}$$

   Prove that the algorithm (9.38) solves the original discrete solution if it converges.

APPENDIX **P**

Projects

Contents of Chapter P

P.1. Coordinate Transformations and FDMs 332
P.2. High-order FEMs for PDEs of One Spacial Variable 334

P.1. Coordinate Transformations and FDMs

Recall: **Example 4.11, p. 104**
Consider the **Poisson equation** defined on a **trapezoidal domain**:

$$\Omega = \{(x_1, x_2) : 0 < x_1 < 1, \;\; 0 < x_2 < (1 + x_1)/2\}. \tag{P.1.1}$$

Define a new coordinate system $\boldsymbol{\xi} \in (0, 1)^2$,

$$\xi_1 = x_1, \quad \xi_2 = \frac{2x_2}{1 + x_1}.$$

Then the Jacobian reads

$$J = \begin{bmatrix} 1 & 0 \\ -2x_2/(1 + x_1)^2 & 2/(1 + x_1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\xi_2/(1 + \xi_1) & 2/(1 + \xi_1) \end{bmatrix} \tag{P.1.2}$$

and

$$B = JAJ^T = JJ^T = \begin{bmatrix} 1 & -\dfrac{\xi_2}{1 + \xi_1} \\ -\dfrac{\xi_2}{1 + \xi_1} & \dfrac{\xi_2^2 + 4}{(1 + \xi_1)^2} \end{bmatrix}. \tag{P.1.3}$$

The matrix $B(\boldsymbol{\xi})$ is clearly symmetric and positive definite on the unit square. The problem

$$-\nabla \cdot B(\boldsymbol{\xi}) \nabla u = f(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in (0, 1)^2, \tag{P.1.4}$$

can be approximated by the standard second-order FD method.

This project allows you to use coordinate transformations to experiment with the **finite difference method** for solving PDEs on **nonrectangular domains**.

What To Do

- Consider

$$\begin{aligned} -\Delta u &= f, \quad \mathbf{x} \in \Omega = (0,1)^2 \\ u &= g, \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{P.1.5}$$

  where Ω is given as in (P.1.1).

- Set the exact solution as

$$u(x,y) = \sin(\pi x)\sin(\pi y), \tag{P.1.6}$$

  so that the right-hand side becomes

$$f(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y).$$

- Implement code using the central finite difference method (FDM) to numerically solve the coordinate-transformed partial differential equation (PDE) in equation (P.1.4).
- Test for second-order error convergence in grid size h by doubling the number of points in both directions (e.g. $n = 20, 40, 80$).
  - To do this, calculate the reverse rule for the transformation.
- You may use Python functions implemented in Section 5.6.

Report: Please structure your report with a summary first, followed by your code, results, and convergence analysis.

P.2. High-order FEMs for PDEs of One Spacial Variable

The provided Python code is implemented for solving

$$\begin{aligned} -u_{xx} &= f, \quad x \in (a,b) \\ u &= g, \qquad x = a,\, b, \end{aligned} \tag{P.2.1}$$

using high-order Galerkin FE methods.

Through the project, you will modify the code for the numerical solution of more general problems of the form

$$\begin{aligned} -(Ku_x)_x + ru &= f, \quad x \in (a,b) \\ Ku_\nu &= g, \qquad\quad x = a,\, b, \end{aligned} \tag{P.2.2}$$

where $K = K(x)$ and r are prescribed continuous positive functions.

Here are your objectives:

- Derive Galerkin FEMs for (P.2.2) of Neumann boundary conditions.
- Modify the code for the problem. You may have to spend a certain amount of time to understand the code. Please save new functions in a new file; do not add any extra functions to `util_FEM_1D.py`.
- Test your code for its convergence, for example, for
  - $(a, b) = (0, \pi)$
  - $K(x) = 1 + x$
  - $r(x) \equiv 1$
  - The exact solution $u(x) = \sin(x)$.

  You have to set f and g correspondingly; for example, $g(0) = 1$ and $g(\pi) = -(1 + \pi)$.

APPENDIX **A**

Basic Concepts in Fluid Dynamics

Physical properties of fluid flow under consideration must be known if one is to either study fluid motion or design numerical methods to simulate it. This appendix is devoted to introducing basic concepts of fluid flows.

Contents of Chapter A

A.1. Conservation Principles . . . 337
A.2. Conservation of Mass . . . 339
A.3. Conservation of Momentum . . . 339
A.4. Non-dimensionalization of the Navier-Stokes Equations . . . 342
A.5. Generic Transport Equations . . . 344
A.6. Homework . . . 345

A.1. Conservation Principles

Conservation laws can be derived by considering a given quantity of matter or *control mass* (CM) and its *extensive* properties such as mass, momentum, and energy. This approach is used to study the dynamics of solid bodies, where the CM is easily identified. However, it is difficult to follow matter in fluid flows. It is more convenient to deal with the flow in a certain spatial region, called the *control volume* (CV).

We first consider the conservation laws for extensive properties: mass and momentum. For mass, which is neither created nor destroyed, the conservation equation reads

$$\frac{\mathrm{d}m}{\mathrm{d}t} = 0, \tag{A.1.1}$$

where t is time and m represents mass. On the other hand, the momentum can

be changed by the action of forces and its conservation equation is Newton's second law of motion

$$\frac{\mathrm{d}(m\mathbf{v})}{\mathrm{d}t} = \sum f, \tag{A.1.2}$$

where $\mathbf{v}$ is the fluid velocity and f is forces acting on the control mass.

We will reformulate these laws with incorporation of the control volume. The fundamental variables will be *intensive*, rather than extensive, properties that are independent of the amount of matter. Examples are density ρ (mass per unit volume) and velocity $\mathbf{v}$ (momentum per unit mass).

For any intensive property ϕ, the corresponding extensive property Φ is by definition given as

$$\Phi = \int_{\Omega_{CM}} \rho\phi \, \mathrm{d}\Omega, \tag{A.1.3}$$

where Ω_{CM} is the volume occupied by the CM. For example, $\phi = 1$ for mass conservation, $\phi = \mathbf{v}$ for momentum conservation, and for a scalar property, ϕ represents the conserved property per unit mass. Using (A.1.3), the left hand side of each of conservation equations, (A.1.1) and (A.1.2), can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CM}} \rho\phi \, \mathrm{d}\Omega = \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CV}} \rho\phi \, \mathrm{d}\Omega + \int_{\partial\Omega_{CV}} \rho\phi(\mathbf{v} - \mathbf{v}_b)\cdot\mathbf{n} \, \mathrm{d}S, \tag{A.1.4}$$

where Ω_{CV} is the CV, $\mathbf{n}$ denotes the unit outward normal to $\partial\Omega_{CV}$, $\mathrm{d}S$ represents the surface element, $\mathbf{v}$ is the fluid velocity, and $\mathbf{v}_b$ denotes the velocity of the CV surface $\partial\Omega_{CV}$. The equation (A.1.4) is called the *control volume equation* or the *Reynolds's transport equation*. For a fixed CV, $\mathbf{v}_b = 0$ and the first derivative on the right hand side of (A.1.4) becomes a local (partial) derivative:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CM}} \rho\phi \, \mathrm{d}\Omega = \frac{\partial}{\partial t}\int_{\Omega_{CV}} \rho\phi \, \mathrm{d}\Omega + \int_{\partial\Omega_{CV}} \rho\phi \, \mathbf{v}\cdot\mathbf{n} \, \mathrm{d}S. \tag{A.1.5}$$

Note that the *material derivative* applied to the control volume is

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\partial}{\partial t} + \mathbf{v}_b \cdot \nabla.$$

For a detailed derivation of this equation, see e.g. [54, 69].

A.2. Conservation of Mass

The integral form of the mass conservation equation follows from the control volume equation (A.1.5), by setting $\phi = 1$:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho \mathrm{d}\Omega + \int_{\partial\Omega} \rho \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = 0, \tag{A.2.1}$$

where we have omitted the subscript CV from Ω. The above equation is also called the *continuity equation*. Recall the Gauss's *divergence theorem*

$$\int_{\Omega} \nabla\cdot\mathbf{A}\,\mathrm{d}\Omega = \int_{\partial\Omega} \mathbf{A}\cdot\mathbf{n}\,\mathrm{d}S, \tag{A.2.2}$$

for any vector field $\mathbf{A}$ defined in the control volume Ω. Applying (A.2.2) to (A.2.1) and allowing the CV to become infinitesimally small, we have the following differential coordinate-free form of the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0, \tag{A.2.3}$$

and its Cartesian form

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v_i)}{\partial x_i} = \frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0, \tag{A.2.4}$$

where x_i $(i = 1, 2, 3)$ or (x, y, z) are the Cartesian coordinates and v_i or (u, v, w) are the Cartesian components of the velocity $\mathbf{v}$. Here we have utilized the Einstein convention that whenever the same index appears twice in any term, summation over the range of that index is applied.

A.3. Conservation of Momentum

Using (A.1.2) and (A.1.5) with $\phi = \mathbf{v}$, one can obtain the integral form of the momentum conservation equation

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\mathbf{v}\,\mathrm{d}\Omega + \int_{\partial\Omega} \rho\mathbf{v}\,\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \sum f. \tag{A.3.1}$$

The right hand side consists of the forces:

– surface forces: pressure, normal and shear stresses, surface tension, etc.;

– body forces: gravity, electromagnetic forces, etc..

The surface forces due to pressure and stresses are the microscopic momentum flux across the surface. For Newtonian fluids, the stress tensor $\mathcal{T}$, which is the molecular transport rate of momentum, reads

$$\mathcal{T} = 2\mu\mathcal{D} + \left[\left(\kappa - \frac{2}{3}\mu\right)\nabla\cdot\mathbf{v} - p\right]I, \tag{A.3.2}$$

where p is the static pressure, μ and κ are respectively the shear coefficient of viscosity and the bulk coefficient of viscosity, I is the unit (identity) tensor, and $\mathcal{D}$ is the rate of strain (deformation) tensor defined by

$$\mathcal{D} = \frac{1}{2}\Big(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\Big). \tag{A.3.3}$$

The following notation is often used in the literature to denote the viscous part of the stress tensor

$$\boldsymbol{\tau} = 2\mu\mathcal{D} + \left[\left(\kappa - \frac{2}{3}\mu\right)\nabla\cdot\mathbf{v}\right]I. \tag{A.3.4}$$

Thus the stress tensor can be written as

$$\mathcal{T} = \boldsymbol{\tau} - pI \tag{A.3.5}$$

and its components read

$$\mathcal{T}_{ij} = \tau_{ij} - p\delta_{ij}, \tag{A.3.6}$$

where

$$\tau_{ij} = 2\mu\mathcal{D}_{ij} + \left(\kappa - \frac{2}{3}\mu\right)\delta_{ij}\nabla\cdot\mathbf{v}, \qquad \mathcal{D}_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right).$$

Assume that gravity g is the only body force. Then, the integral form of the momentum conservation equation becomes

$$\frac{\partial}{\partial t}\int_\Omega \rho\mathbf{v}\,\mathrm{d}\Omega + \int_{\partial\Omega}\rho\mathbf{v}\,\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \int_{\partial\Omega}\mathcal{T}\cdot\mathbf{n}\,\mathrm{d}S + \int_\Omega \rho\mathbf{g}\,\mathrm{d}\Omega. \tag{A.3.7}$$

A coordinate-free vector form of the momentum conservation equation is readily obtained by applying the Gauss's divergence theorem (A.2.2) to the convective and diffusive flux terms of (A.3.7):

$$\frac{\partial(\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = \nabla \cdot \mathcal{T} + \rho \mathbf{g}. \tag{A.3.8}$$

The continuity equation (A.2.3) and the momentum equations (A.3.8) are called the *Navier-Stokes equations*.

The corresponding equation for the ith component of (A.3.8) is

$$\frac{\partial(\rho v_i)}{\partial t} + \nabla \cdot (\rho v_i \mathbf{v}) = \nabla \cdot \mathcal{T}_i + \rho g_i, \tag{A.3.9}$$

where $\mathcal{T}_i$ in the Cartesian coordinates can be expressed as

$$\mathcal{T}_i = \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) I_j + \left[\left(\kappa - \frac{2}{3}\mu \right) \nabla \cdot \mathbf{v} - p \right] I_i, \tag{A.3.10}$$

where I_i is the Cartesian unit vector in the direction of the coordinate x_i.

The integral form of (A.3.9) reads

$$\frac{\partial}{\partial t} \int_\Omega \rho v_i \, \mathrm{d}\Omega + \int_{\partial\Omega} \rho v_i \, \mathbf{v} \cdot \mathbf{n} \, \mathrm{d}S = \int_{\partial\Omega} \mathcal{T}_i \cdot \mathbf{n} \, \mathrm{d}S + \int_\Omega \rho g_i \, \mathrm{d}\Omega. \tag{A.3.11}$$

In index notation, (A.3.9) can be rewritten as

$$\frac{\partial(\rho v_i)}{\partial t} + \frac{\partial(\rho v_j v_i)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i. \tag{A.3.12}$$

In approximating the momentum equations by finite difference schemes, it is often more convenient to deal with the following non-conservative form

$$\rho \left(\frac{\partial v_i}{\partial t} + \mathbf{v} \cdot \nabla v_i \right) = \nabla \cdot \mathcal{T}_i + \rho g_i. \tag{A.3.13}$$

Here we describe the momentum equations for the *incompressible* Newtonian fluid of constant density and viscosity. In this case, since $\nabla \cdot \mathbf{v} = 0$, (A.3.12) becomes

$$\rho \left(\frac{\partial v_i}{\partial t} + v_j \frac{\partial v_i}{\partial x_j} \right) = -\frac{\partial p}{\partial x_i} + \rho g_i + \mu \frac{\partial^2 v_i}{\partial x_j \partial x_j}. \tag{A.3.14}$$

In 2D Cartesian coordinates, (A.3.14) reads

$$\begin{aligned} &\text{(a)} \quad \rho\left(\frac{\partial v_1}{\partial t}+v_1\frac{\partial v_1}{\partial x}+v_2\frac{\partial v_1}{\partial y}\right)=-\frac{\partial p}{\partial x}+\rho g_1+\mu\left(\frac{\partial^2 v_1}{\partial x^2}+\frac{\partial^2 v_1}{\partial y^2}\right),\\ &\text{(b)} \quad \rho\left(\frac{\partial v_2}{\partial t}+v_1\frac{\partial v_2}{\partial x}+v_2\frac{\partial v_2}{\partial y}\right)=-\frac{\partial p}{\partial y}+\rho g_2+\mu\left(\frac{\partial^2 v_2}{\partial x^2}+\frac{\partial^2 v_2}{\partial y^2}\right). \end{aligned} \tag{A.3.15}$$

Thus the complete set of the Navier-Stokes equations for incompressible homogeneous flows becomes (in Gibbs notation)

$$\begin{aligned} &\text{(a)} \quad \nabla\cdot\mathbf{v}=0,\\ &\text{(b)} \quad \frac{\partial\mathbf{v}}{\partial t}+(\mathbf{v}\cdot\nabla)\mathbf{v}=-\nabla p'+\mathbf{g}+\nu\Delta\mathbf{v}. \end{aligned} \tag{A.3.16}$$

where $p'=p/\rho$ and $\nu=\mu/\rho$ is the *kinematic viscosity coefficient*.

In the case of frictionless (inviscid) flow, i.e., $\mu=0$, the equation of motion (A.3.16.b) reduces to the *Euler's equation*,

$$\frac{\partial\mathbf{v}}{\partial t}+(\mathbf{v}\cdot\nabla)\mathbf{v}=-\nabla p'+\mathbf{g}. \tag{A.3.17}$$

A.4. Non-dimensionalization of the Navier-Stokes Equations

Now we will discuss some scaling properties of the Navier-Stokes equations with the aim of introducing a parameter (the Reynolds number) that measures the effect of viscosity.

Let L be a *reference length* L and U a *reference velocity*. These number are chosen in an arbitrary way. For example, if we consider a free-stream flow past a sphere, L can be either the radius or the diameter of the sphere and U can be the magnitude of the fluid velocity at infinity. The choice determines a time scale $T=L/U$. We measure $\mathbf{x}$, $\mathbf{v}$, and t as fractions of these scales, i.e., we introduce the following dimensionless quantities

$$\mathbf{x}'=\frac{\mathbf{x}}{L},\quad \mathbf{v}'=\frac{\mathbf{v}}{U},\quad t'=\frac{t}{T}.$$

Consider the change of variables e.g. for the x-component of the Navier-Stokes equations in 2D Cartesian coordinates (A.3.15.a):

$$\rho\left(\frac{\partial(Uv_1')}{\partial t'}\frac{\partial t'}{\partial t}+Uv_1'\frac{\partial(Uv_1')}{\partial x'}\frac{\partial x'}{\partial x}+Uv_2\frac{\partial(Uv_1')}{\partial y'}\frac{\partial y'}{\partial y}\right)$$
$$=-\frac{\partial p}{\partial x'}\frac{\partial x'}{\partial x}+\rho g_1+\mu\left(\frac{\partial^2(Uv_1')}{\partial(Lx')^2}+\frac{\partial^2(Uv_1')}{\partial(Lx')^2}\right),$$

or

$$\rho\,\frac{U^2}{L}\left(\frac{\partial v_1'}{\partial t'}+v_1'\frac{\partial v_1'}{\partial x'}+v_2'\frac{\partial v_1'}{\partial y'}\right)=-\frac{1}{L}\frac{\partial p}{\partial x'}+\rho g_1+\mu\,\frac{U}{L^2}\left(\frac{\partial^2 v_1'}{\partial x'^2}+\frac{\partial^2 v_1'}{\partial y'^2}\right).$$

Thus we have

$$\frac{\partial v_1'}{\partial t'}+v_1'\frac{\partial v_1'}{\partial x'}+v_2'\frac{\partial v_1'}{\partial y'}=-\frac{1}{\rho U^2}\frac{\partial p}{\partial x'}+\frac{L}{U^2}g_1+\frac{\nu}{LU}\left(\frac{\partial^2 v_1'}{\partial x'^2}+\frac{\partial^2 v_1'}{\partial y'^2}\right).$$

It is straightforward to apply the change of variables to the x-component (and also the other ones) of the Navier-Stokes equations in 3D. It follows from the change of variables that (A.3.16) becomes

$$\begin{aligned}&\text{(a)}\quad \nabla'\cdot\mathbf{v}'=0,\\ &\text{(b)}\quad \frac{\partial\mathbf{v}'}{\partial t'}+\mathbf{v}'\cdot\nabla'\mathbf{v}'=-\nabla' p'+\mathbf{g}'+\frac{1}{R}\Delta'\mathbf{v}',\end{aligned}\tag{A.4.1}$$

where

$$p'=\frac{p}{\rho U^2},\quad \mathbf{g}'=\frac{L\mathbf{g}}{U^2},\quad R=\frac{LU}{\nu}.$$

Here the dimensionless quantity R is the *Reynolds number*. The equations (A.4.1) are the the Navier-Stokes equations in dimensionless variables. (The gravity term $\mathbf{g}'$ is often ignored.)

When R is very small, the flow transport is dominated by the diffusion/dissipation and the convection term (sometimes, called *inertia*) $\mathbf{v}\cdot\nabla\mathbf{v}$ becomes much smaller than the diffusion term $\frac{1}{R}\Delta\mathbf{v}$, i.e.,

$$|\mathbf{v}\cdot\nabla\mathbf{v}|\ll\left|\frac{1}{R}\Delta\mathbf{v}\right|.$$

Ignoring the convection term, we have the *Stokes's equations*

$$\begin{aligned}&\text{(a)}\quad \nabla\cdot\mathbf{v}=0,\\ &\text{(b)}\quad \frac{\partial\mathbf{v}}{\partial t}=-\nabla p+\mathbf{g}+\frac{1}{R}\Delta\mathbf{v}.\end{aligned}\tag{A.4.2}$$

A.5. Generic Transport Equations

The integral form of the equation describing conservation of a scalar quantity ϕ is analogous to the previous equations and reads

$$\frac{\partial}{\partial t}\int_\Omega \rho\phi \,\mathrm{d}\Omega + \int_{\partial\Omega} \rho\phi\, \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \sum f_\phi, \tag{A.5.1}$$

where f_ϕ represents any sources and sinks and transport of ϕ by mechanisms other than convection. Diffusive transport f_ϕ^d is always present and usually expressed by a gradient approximation

$$f_\phi^d = \int_{\partial\Omega} D\nabla\phi\cdot\mathbf{n}\,\mathrm{d}S, \tag{A.5.2}$$

where D is the diffusivity for ϕ. The equation (A.5.2) is called *Fick's law* for mass diffusion or *Fourier's law* for heat diffusion. Since the sources/sinks can be expressed as

$$f_\phi^s = \int_\Omega q_\phi \,\mathrm{d}\Omega,$$

setting $f_\phi = f_\phi^d + f_\phi^s$ and applying the Gauss's divergence theorem, one can obtain the generic transport equation, the coordinate-free form of the equation (A.5.1):

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\cdot(\rho\phi\mathbf{v}) = \nabla\cdot(D\nabla\phi) + q_\phi. \tag{A.5.3}$$

The lecture note will first focus on the numerical methods for (A.5.3). More precisely, we will consider numerical methods for the convection-diffusion equation of the form

$$\begin{aligned}
&\text{(a)} \quad \frac{\partial c}{\partial t} + \nabla\cdot(\mathbf{v}c) - \nabla\cdot(D\nabla c) = f, && (\mathbf{x},t)\in\Omega\times J,\\
&\text{(b)} \quad (D\nabla c)\cdot\nu = 0, && (\mathbf{x},t)\in\Gamma\times J,\\
&\text{(c)} \quad c = c_0, && \mathbf{x}\in\Omega,\ \ t=0,
\end{aligned} \tag{A.5.4}$$

where c is the unknown (e.g. concentration), $\Omega\subset\mathbb{R}^d$, $1\le d\le 3$, is a bounded domain with its boundary $\Gamma=\partial\Omega$ and $J=(0,T]$ the time interval, $T>0$. Here $\mathbf{v}=\mathbf{v}(c)$ is the fluid velocity, ν is the outward normal to Γ, and $f=f(c)$

denotes chemical reactions and source/sink. The diffusion tensor $D = D(\mathbf{v}, c)$ is symmetric and positive definite:

$$D^T = D; \quad D_*|\mathbf{y}|^2 \leq \mathbf{y}^T D(\mathbf{x})\mathbf{y} \leq D^*|\mathbf{y}|^2, \quad \forall\, \mathbf{x} \in \Omega, \ \forall\, \mathbf{y} \in \mathbb{R}^d,$$

for some positive constants D_* and D^*. The velocity either can be obtained by solving another equation such as the pressure equation or is given from experiments.

Special features of the continuity and momentum equations (Navier-Stokes equations) will be considered afterwards as applications of the numerical methods for the generic equation.

A.6. Homework

1. Use $\nabla \cdot (\rho v_i \mathbf{v}) = v_i \nabla \cdot (\rho \mathbf{v}) + \rho \mathbf{v} \cdot \nabla v_i$ to derive (A.3.13) from (A.2.4) and (A.3.9).
2. Derive (A.3.14).

APPENDIX **B**

Orthogonal Polynomials and Quadratures

Contents of Chapter B

B.1. Orthogonal Polynomials 347
B.2. Gauss-Type Quadratures 349

B.1. Orthogonal Polynomials

Let w be a given function defined on $(-1,1)$ and positive there. (The function w is often called a weight function.) Let f and g be defined on the interval $(-1,1)$. Define the *scalar product* of the functions f and g on $(-1,1)$ as

$$(f,g)_w = \int_{-1}^{1} f(x)g(x)w(x)dx. \tag{B.1.1}$$

Then, the *orthogonal polynomials* on $(-1,1)$ with respect to the weight function w are a series of polynomials $\{P_k\}_{k=0,1,2,\cdots}$ satisfying

$$P_k \in \mathbf{P}_k; \quad (P_k, P_m)_w = 0, \;\; k \neq m, \tag{B.1.2}$$

where $\mathbf{P}_k$ denotes the space of polynomials of degree $\leq k$.

Those orthogonal polynomials satisfy a *three-term recurrence relation* of the form

$$P_{k+1}(x) = A_k(x - B_k)P_k(x) - C_k P_{k-1}(x), \quad k = 0, 1, 2, \cdots, \tag{B.1.3}$$

where

$$\begin{aligned} P_{-1} &\equiv 0, \\ A_k &= \frac{\alpha_{k+1}}{\alpha_k}, \\ B_k &= \frac{(xP_k, P_k)_w}{S_k}, \\ C_k &= \begin{cases} \text{arbitrary}, & k = 0, \\ \dfrac{A_k S_k}{A_{k-1} S_{k-1}}, & k > 0. \end{cases} \end{aligned}$$

Here α_k is the leading coefficient of P_k and S_k is defined as

$$S_k = (P_k, P_k)_w.$$

Example B.1. Legendre Polynomials $\{L_k\}$: *the weight function*

$$w(x) \equiv 1.$$

With this choice of the weight function, starting with $L_0(x) = 1$, one can get

$$A_k = \frac{2k+1}{k+1}, \quad B_k = 0, \quad C_k = \frac{k}{k+1},$$

where a normalization is applied for $L_k(1) = 1$. Thus the Legendre polynomials satisfy the following three-term recurrence relation

$$L_{k+1}(x) = \frac{(2k+1)xL_k(x) - kL_{k-1}(x)}{k+1}. \tag{B.1.4}$$

A few first Legendre polynomials are

$$\begin{aligned} L_0(x) &= 1, \\ L_1(x) &= x, \\ L_2(x) &= \frac{3}{2}\left(x^2 - \frac{1}{3}\right), \\ L_3(x) &= \frac{5}{2}\left(x^3 - \frac{3}{5}x\right), \\ L_4(x) &= \frac{35}{8}\left(x^4 - \frac{6}{7}x^2 + \frac{3}{35}\right). \end{aligned} \tag{B.1.5}$$

Relevant properties are

$$\begin{aligned}
&|L_k(x)| \le 1, \quad \forall\, x \in [-1,1],\\
&L_k(\pm 1) = (\pm 1)^k,\\
&|L_k'(x)| \le k(k+1)/2, \quad \forall\, x \in [-1,1],\\
&L_k'(\pm 1) = (\pm 1)^k k(k+1)/2,\\
&(L_k, L_k)_{w=1} = (k+1/2)^{-1}.
\end{aligned} \tag{B.1.6}$$

Example **B.2. Chebyshev Polynomials $\{T_k\}$***: the weight function*

$$w(x) := (1-x^2)^{-1/2}.$$

With this choice of the weight function, one can get the three-term recurrence relation for the Chebyshev polynomials

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x). \tag{B.1.7}$$

A few first Chebyshev polynomials are

$$\begin{aligned}
T_0(x) &= 1,\\
T_1(x) &= x,\\
T_2(x) &= 2x^2 - 1,\\
T_3(x) &= 4x^3 - 3x,\\
T_4(x) &= 8x^4 - 8x^2 + 1.
\end{aligned} \tag{B.1.8}$$

Relevant properties are

$$\begin{aligned}
&|T_k(x)| \le 1, \quad \forall\, x \in [-1,1],\\
&T_k(\pm 1) = (\pm 1)^k,\\
&|T_k'(x)| \le k^2, \quad \forall\, x \in [-1,1],\\
&T_k'(\pm 1) = (\pm 1)^k k^2,\\
&(T_k, T_k)_w = \begin{cases} \pi, & \text{if } k = 0,\\ \pi/2, & \text{if } k \ge 1. \end{cases}
\end{aligned} \tag{B.1.9}$$

B.2. Gauss-Type Quadratures

There are close relations between orthogonal polynomials and Gauss-type integration quadrature formulas on the interval $[-1,1]$. We first review the Gauss-type integration formulas.

Theorem B.3. Gauss Integration. *Let* $\{x_0, x_1, \cdots, x_n\}$ *be the zeros of the* $(n+1)$*-th orthogonal polynomial* P_{n+1}*. Let* $\{w_0, w_1, \cdots, w_n\}$ *be the solution of the linear system*

$$\sum_{j=0}^{n} (x_j)^i w_j \;=\; \int_{-1}^{1} x^i w(x) dx, \quad i = 0, 1, \cdots, n.$$

Then, (1). $w_j > 0$*,* $j = 0, 1, \cdots, n$*, and*

$$\int_{-1}^{1} f(x) w(x) \;=\; \sum_{j=0}^{n} f(x_j) w_j, \quad \forall\, f \in \mathbf{P}_{2n+1}. \tag{B.2.1}$$

(2). *There is no* x_j *and* w_j*,* $j = 0, 1, \cdots, n$*, such that (B.2.1) holds for all* $f \in \mathbf{P}_{2n+2}$*.*

The Gauss integration formula is well known. However, the zeros of P_{n+1} are all in the interior of $[-1, 1]$. Thus, it shows a drawback when a boundary condition is to be imposed. In particular, most finite element methods require the continuity of the solution on element boundaries and introduce nodal points on the boundary. The following Gauss-Lobatto formula is more useful than the Gauss formula in numerical PDEs.

Theorem B.4. Gauss-Lobatto Integration. *Let* $x_0 = -1$*,* $x_n = 1$*, and* x_j*,* $j = 1, 2, \cdots, n-1$*, be the zeros of the first-derivative of the* n*-th orthogonal polynomial,* P_n'*. Let* $\{w_0, w_1, \cdots, w_n\}$ *be the solution of the linear system*

$$\sum_{j=0}^{n} (x_j)^i w_j \;=\; \int_{-1}^{1} x^i w(x) dx, \quad i = 0, 1, \cdots, n.$$

Then,

$$\int_{-1}^{1} f(x) w(x) \;=\; \sum_{j=0}^{n} f(x_j) w_j, \quad \forall\, f \in \mathbf{P}_{2n-1}. \tag{B.2.2}$$

For the Legendre polynomials, the explicit formulas for the quadrature nodes are not known. Thus the nodal points and the corresponding weights must be computed numerically as zeros of appropriate polynomials and the solution of a linear system, respectively. On the other hand, for Chebyshev series, the

points and weights are known explicitly. Here we collect those formulas and explicit expressions:

Legendre-Gauss:

$$\begin{aligned} &x_j \ (= \textbf{zeros of } L_{n+1}), \quad j = 0, 1, \cdots, n, \\ &w_j = \frac{2}{(1-x_j^2)[L'_{n+1}(x_j)]^2}, \quad j = 0, 1, \cdots, n. \end{aligned} \tag{B.2.3}$$

Legendre-Gauss-Lobatto:

$$\begin{aligned} &x_0 = -1, \ \ x_n = 1; \ \ x_j \ (= \textbf{zeros of } L'_n), \quad j = 1, 2, \cdots, n-1, \\ &w_j = \frac{2}{n(n+1)[L_n(x_j)]^2}, \quad j = 0, 1, \cdots, n. \end{aligned} \tag{B.2.4}$$

Chebyshev-Gauss:

$$x_j \ = \ -\cos\left(\frac{(2j+1)\pi}{2n+2}\right), \quad w_j \ = \ \frac{\pi}{n+1}, \quad j = 0, 1, \cdots, n. \tag{B.2.5}$$

Chebyshev-Gauss-Lobatto:

$$x_j \ = \ -\cos\left(\frac{j\pi}{n}\right), \quad w_j \ = \ \begin{cases} \pi/(2n), & j = 0, n, \\ \pi/n, & j = 1, \cdots, n-1. \end{cases} \tag{B.2.6}$$

The following shows a few examples for the Legendre-Gauss-Lobatto points and the corresponding weights on the interval $[-1, 1]$:

	Legendre-Gauss-Lobatto points	weights
$n = 1$	$-1 \quad 1$	$1 \quad 1$
$n = 2$	$-1 \quad 0 \quad 1$	$\frac{1}{3} \quad \frac{4}{3} \quad \frac{1}{3}$
$n = 3$	$-1 \quad -\left(\frac{1}{5}\right)^{1/2} \quad \left(\frac{1}{5}\right)^{1/2} \quad 1$	$\frac{1}{6} \quad \frac{5}{6} \quad \frac{5}{6} \quad \frac{1}{6}$
$n = 4$	$-1 \quad -\left(\frac{3}{7}\right)^{1/2} \quad 0 \quad \left(\frac{3}{7}\right)^{1/2} \quad 1$	$\frac{1}{10} \quad \frac{49}{90} \quad \frac{64}{90} \quad \frac{49}{90} \quad \frac{1}{10}$

(B.2.7)

APPENDIX C

Some Mathematical Formulas

Contents of Chapter C

C.1. Trigonometric Formulas 353
C.2. Vector Identities 353

C.1. Trigonometric Formulas

The following trigonometric formulas are useful

$$\begin{aligned}
&\text{(a)}\quad \sin(x+y) = \sin x\cos y + \cos x \sin y,\\
&\text{(b)}\quad \cos(x+y) = \cos x\cos y - \sin x \sin y,\\
&\text{(c)}\quad \sin x + \sin y = 2\sin\left(\frac{x+y}{2}\right)\cos\left(\frac{x-y}{2}\right),\\
&\text{(d)}\quad \sin x - \sin y = 2\cos\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right),\\
&\text{(e)}\quad \cos x + \cos y = 2\cos\left(\frac{x+y}{2}\right)\cos\left(\frac{x-y}{2}\right),\\
&\text{(f)}\quad \cos x - \cos y = -2\sin\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right).
\end{aligned} \tag{C.1.1}$$

By setting $x = 2\theta$ and $y = 0$ in (C.1.1.e), one also can have

$$2\sin^2\theta = 1 - \cos(2\theta), \quad 2\cos^2\theta = 1 + \cos(2\theta). \tag{C.1.2}$$

C.2. Vector Identities

Let A, B, C, and D be vectors in $\mathbb{R}^3$ and f is scalar. Let

$$\mathbf{A}\cdot\mathbf{B} = A_1B_1 + A_2B_2 + A_3B_3$$

and

$$\begin{aligned}\mathbf{A}\times\mathbf{B} &= (A_2B_3 - A_3B_2,\ A_3B_1 - A_1B_3,\ A_1B_2 - A_2B_1)\\ &= \det\begin{bmatrix}\widehat{j_1} & \widehat{j_2} & \widehat{j_3}\\ A_1 & A_2 & A_3\\ B_1 & B_2 & B_3\end{bmatrix},\end{aligned}$$

where $\widehat{j_i}$ is the unit vector in the x_i-direction. Then

$$\mathbf{A}\cdot\mathbf{B} = |\mathbf{A}|\,|\mathbf{B}|\cos\theta, \quad \mathbf{A}\times\mathbf{B} = |\mathbf{A}|\,|\mathbf{B}|\sin\theta\,\widehat{\mathbf{n}},$$

where θ is the angle between $\mathbf{A}$ and $\mathbf{B}$ and $\widehat{\mathbf{n}}$ is the unit normal vector from the plane containing $\mathbf{A}$ and $\mathbf{B}$ whose orientation is determined by the *right-hand rule*. (When four fingers grab directing from $\mathbf{A}$ to $\mathbf{B}$, then the direction of the thumb determines $\widehat{\mathbf{n}}$.) Let $\nabla\times$ denote the *curl* operator defined as

$$\nabla\times\mathbf{A} = \left(\frac{\partial A_3}{\partial y} - \frac{\partial A_2}{\partial z}, \frac{\partial A_1}{\partial z} - \frac{\partial A_3}{\partial x}, \frac{\partial A_2}{\partial x} - \frac{\partial A_1}{\partial y}\right).$$

Then,

$$\begin{aligned}
&\mathbf{A}\cdot(\mathbf{B}\times\mathbf{C}) = \mathbf{B}\cdot(\mathbf{C}\times\mathbf{A}) = \mathbf{C}\cdot(\mathbf{A}\times\mathbf{B}),\\
&\mathbf{A}\times(\mathbf{B}\times\mathbf{C}) = (\mathbf{A}\cdot\mathbf{C})\mathbf{B} - (\mathbf{A}\cdot\mathbf{B})\mathbf{C},\\
&(\mathbf{A}\times\mathbf{B})\cdot(\mathbf{C}\times\mathbf{D}) = (\mathbf{A}\cdot\mathbf{C})(\mathbf{B}\cdot\mathbf{D}) - (\mathbf{A}\cdot\mathbf{D})(\mathbf{B}\cdot\mathbf{C}),\\
&\nabla(\mathbf{A}\cdot\mathbf{B}) = \mathbf{A}\times(\nabla\times\mathbf{B}) + \mathbf{B}\times(\nabla\times\mathbf{A}) + (\mathbf{A}\cdot\nabla)\mathbf{B} + (\mathbf{B}\cdot\nabla)\mathbf{A},\\
&\nabla\cdot(\mathbf{A}\times\mathbf{B}) = \mathbf{B}\cdot(\nabla\times\mathbf{A}) - \mathbf{A}\cdot(\nabla\times\mathbf{B}),\\
&\nabla\times(f\mathbf{A}) = f(\nabla\times\mathbf{A}) - \mathbf{A}\times(\nabla f),\\
&\nabla\times(\mathbf{A}\times\mathbf{B}) = (\mathbf{B}\cdot\nabla)\mathbf{A} - (\mathbf{A}\cdot\nabla)\mathbf{B} + \mathbf{A}(\nabla\cdot\mathbf{B}) - \mathbf{B}(\nabla\cdot\mathbf{A}),\\
&\nabla\cdot(\nabla\times\mathbf{A}) = 0,\\
&\nabla\times(\nabla f) = 0,\\
&\nabla\times(\nabla\times\mathbf{A}) = \nabla(\nabla\cdot\mathbf{A}) - \nabla^2\mathbf{A}.
\end{aligned} \tag{C.2.1}$$

Associated with vectors are the following integrals.

Gauss's divergence theorem:

$$\int_V \nabla\cdot\mathbf{B}\,d\mathbf{x} = \oint_A \mathbf{B}\cdot\mathbf{n}\,ds$$

Stokes's theorem:

$$\int_A (\nabla\times\mathbf{B})\cdot\mathbf{n}\,ds = \oint_C \mathbf{B}\cdot d\mathbf{l}$$

APPENDIX **D**

Finite Difference Formulas

Contents of Chapter D

Here we summarize second- and fourth-order finite difference formulas. In the following, $h(>0)$ is the spatial variable and $u_i = u(x_0 + ih)$.

Central 2nd-order FD schemes:

$$\begin{aligned} u_x(x_0) &\approx \frac{u_1 - u_{-1}}{2h} \\ u_{xx}(x_0) &\approx \frac{u_1 - 2u_0 + u_{-1}}{h^2} \\ u_{xxx}(x_0) &\approx \frac{u_2 - 2u_1 + 2u_{-1} - u_{-2}}{2h^3} \\ u^{(4)}(x_0) &\approx \frac{u_2 - 4u_1 + 6u_0 - 4u_{-1} + u_{-2}}{h^4} \end{aligned} \tag{D.0.1}$$

Central 4th-order FD schemes:

$$\begin{aligned} u_x(x_0) &\approx \frac{-u_2 + 8u_1 - 8u_{-1} + u_{-2}}{12h} \\ u_{xx}(x_0) &\approx \frac{-u_2 + 16u_1 - 30u_0 + 16u_{-1} - u_{-2}}{12h^2} \\ u_{xxx}(x_0) &\approx \frac{-u_3 + 8u_2 - 13u_1 + 13u_{-1} - 8u_{-2} + u_{-3}}{8h^3} \\ u^{(4)}(x_0) &\approx \frac{-u_3 + 12u_2 - 39u_1 + 56u_0 - 39u_{-1} + 12u_{-2} - u_{-3}}{6h^4} \end{aligned} \tag{D.0.2}$$

One-sided 2nd-order FD schemes:

$$\begin{aligned}
u_x(x_0) &\approx \pm\frac{-3u_0 + 4u_{\pm1} - u_{\pm2}}{2h} \\
u_{xx}(x_0) &\approx \frac{2u_0 - 5u_{\pm1} + 4u_{\pm2} - f_{\pm3}}{h^2} \\
u_{xxx}(x_0) &\approx \pm\frac{-5u_0 + 18u_{\pm1} - 24u_{\pm2} + 14f_{\pm3} - 3u_{\pm4}}{2h^3} \\
u^{(4)}(x_0) &\approx \frac{3u_0 - 14u_{\pm1} + 26u_{\pm2} - 24f_{\pm3} + 11u_{\pm4} - 2u_{\pm5}}{h^4}
\end{aligned} \tag{D.0.3}$$

Bibliography

[1] V. AGHOSKOV, *Poincaré–Steklov's operators and domain decomposition methods in finite dimensional spaces*, in First International Symposium on Domain Decomposition Method for Partial Differential Equations, R. Glowinski, G. Golub, G. Meurant, and J. Periaux, eds., SIAM, Philadelphia, 1988, pp. 73–112.

[2] W. AMES AND D. LEE, *Current development in the numerical treatment of ocean acoustic propagation*, Appl. Numer. Math., 3 (1987), pp. 25–47.

[3] R. BARRETT, M. BERRY, T. CHAN, J. DEMMEL, J. DONATO, J. DONGARRA, V. EIJKHOUT, R. POZO, C. ROMINE, AND H. VAN DER VORST, *Templates for the solution of linear systems: Building blocks for iterative methods*, SIAM, Philadelphia, 1994. The postscript file is free to download from `http://www.netlib.org/templates/` along with source codes.

[4] P. BJORSTAD AND O. WIDLUND, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal., 23 (1986), pp. 1097–1120.

[5] J.-F. BOURGAT, R. GLOWINSKI, P. LE TALLEC, AND M. VIDRASCU, *Variational formulation and algorithm for trace operator in domain decomposition calculations*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., SIAM, Philadelphia, 1989, pp. 3–16.

[6] J. BRAMBLE, J. PASCIAK, AND A. SCHATZ, *An iterative method for elliptic problems on regions partitioned into substructures*, Math. Comput., 46 (1986), pp. 361–369.

[7] S. CANDEL, *A review of numerical methods in acoustic wave propagation*, in Recent Advances in Aeroacoustics, A. Krothapalli and C. A. Smith, eds., Springer-Verlag, New York, 1986, pp. 339–410.

[8] Y. CHA AND S. KIM, *Edge-forming methods for color image zooming*, IEEE Trans. Image Process., 15 (2006), pp. 2315–2323.

[9] R. CLAYTON AND B. ENGQUIST, *Absorbing boundary conditions for acoustic and elastic wave calculations*, Bull. Seismol. Soc. Amer., 67 (1977), pp. 1529–1540.

[10] G. COHEN, P. JOLY, AND N. TORDJMAN, *Construction and analysis of higher order finite elements with mass lumping for the wave equation*, in Second International Conference on Mathematical and Numerical Aspects of Wave Propagation, R. Kleinman, T. Angell, D. Colton, F. Santosa, and I. Stakgold, eds., SIAM, Philadelphia, 1993, pp. 152–160.

[11] G. DAHLQUIST, *A special stability problem for linear multistep methods*, BIT, 3 (1963), pp. 27–43.

[12] Y. DE ROECK AND P. LE TALLEC, *Analysis and test of a local domain decomposition preconditioner*, in Fourth International Symposium on Domain Decomposition Method for Partial Differential Equations, R. Glowinski, G. Meurant, J. Periaux, and O. B. Widlund, eds., SIAM, Philadelphia, 1991, pp. 112–128.

[13] B. DESPRÉS, *Domain decomposition method and the Helmholtz problem*, in Mathematical and Numerical Aspects of Wave Propagation Phenomena, G. Cohen, L. Halpern, and P. Joly, eds., Philadelphia, 1991, SIAM, pp. 44–52.

[14] J. DOUGLAS, JR., *On the numerical integration of* $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}$ *by implicit methods*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 42–65.

[15] J. DOUGLAS, JR. AND J. GUNN, *A general formulation of alternating direction methods Part I. Parabolic and hyperbolic problems*, Numer. Math., 6 (1964), pp. 428–453.

[16] J. DOUGLAS, JR. AND S. KIM, *Improved accuracy for locally one-dimensional methods for parabolic equations*, Mathematical Models and Methods in Applied Sciences, 11 (2001), pp. 1563–1579.

[17] J. DOUGLAS, JR., P. PAES LEME, J. ROBERTS, AND J. WANG, *A parallel iterative procedure applicable to the approximate solution of second order partial differential equations by mixed finite element methods*, Numer. Math., 65 (1993), pp. 95–108.

[18] J. DOUGLAS, JR. AND D. PEACEMAN, *Numerical solution of two-dimensional heat flow problems*, American Institute of Chemical Engineering Journal, 1 (1955), pp. 505–512.

[19] M. DRYJA AND O. WIDLUND, *Some recent results on Schwarz type domain decomposition algorithms*, in Domain Decomposition Methods in Science and Engineering, A. Quarteroni, J. Periaux, Y. Kuznetsov, and O. Widlund, eds., vol. 157 of Contemporary Mathematics, Philadelphia, 1994, SIAM, pp. 53–61.

[20] E. D'YAKONOV, *Difference schemes with split operators for multidimensional unsteady problems (English translation)*, USSR Comp. Math., 3 (1963), pp. 581–607.

[21] B. ENGQUIST AND A. MAJDA, *Absorbing boundary conditions for the numerical simulation of waves*, Math. Comp., 31 (1977), pp. 629–651.

[22] B. ENGQUIST AND A. MAJDA, *Radiation boundary conditions for acoustic and elastic wave calculations*, Comm. Pure Appl. Math., 32 (1979), pp. 314–358.

[23] J. FERZIGER AND M. PERIC, *Computational methods for fluid dynamics, 2nd Edition*, Springer-Verlag, Berlin, Heidelberg, New York, 1999.

[24] R. W. FREUND, *Conjugate gradient–type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 425–448.

[25] S. GERSCHGORIN, *Über die abgrenzung der eigenwerte einer matrix*, Izv. Akad. Nauk SSSR Ser. Mat., 7 (1931), pp. 746–754.

[26] D. GILBERG AND N. TRUDINGER, *Elliptic Partial Differential Equations of Second Order*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1983.

[27] B. GUSTAFSSON, H.-O. KREISS, AND J. OLIGER, *Time Dependent Problems and Difference Methods*, Wiley-Interscience, New York, 1996.

[28] I. HARARI AND R. DJELLOULI, *Analytical study of the effect of wave number on the performance of local absorbing boundary conditions for acoustic scattering*, Appl. Numer. Math., 50 (2004), pp. 15–47.

[29] R. L. HIGDON, *Absorbing boundary conditions for difference approximations to the multi-dimensional wave equation*, Math. Comp., 47 (1986), pp. 437–459.

[30] ——, *Numerical absorbing boundary conditions for the wave equation*, Math. Comp., 49 (1987), pp. 65–90.

[31] F. Q. HU, *Absorbing boundary conditions*, Int. J. Comput. Fluid Dyn., 18 (2004), pp. 513–522.

[32] C. JOHNSON, *Numerical Solutions of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, New York, New Rochelle, Melbourne, Sydney, 1987.

[33] C. KELLY, *Iterative methods for linear and nonlinear equations*, SIAM, Philadelphia, 1995.

[34] H. KIM, Y. CHA, AND S. KIM, *Curvature interpolation method for image zooming*, IEEE Trans. Image Process., 20 (2011), pp. 1895–1903.

[35] S. KIM, *GRADE: Graduate Research and Applications for Differential Equations*. The **modelcode** library is under construction for education and research in Industrial and Computational Mathematics, initiated in Spring 1999; the codes are available through internet access to `http://skim.math.msstate.edu/GRADE`.

[36] ——, *Numerical treatments for the Helmholtz problem by domain decomposition technique*, Contemporary Mathematics, 180 (1994), pp. 245–250.

[37] ——, *Parallel multidomain iterative algorithms for the Helmholtz wave equation*, Appl. Numer. Math., 17 (1995), pp. 411–429.

[38] ——, *Domain decomposition iterative procedures for solving scalar waves in the frequency domain*, Numer. Math., 79 (1998), pp. 231–259.

[39] ——, *An $\mathcal{O}(N)$ level set method for eikonal equations*, SIAM J. Sci. Comput., 22 (2001), pp. 2178–2193.

[40] S. KIM AND R. COOK, *3D traveltime computation using second-order ENO scheme*, Geophysics, 64 (1999), pp. 1867–1876.

[41] S. KIM AND SOOHYUN KIM, *Multigrid simulation for high-frequency solutions of the Helmholtz problem in heterogeneous media*, SIAM J. Sci. Comput., 24 (2002), pp. 684–701.

[42] S. KIM AND M. LEE, *Artificial damping techniques for scalar waves in the frequency domain*, Computers Math. Applic., 31, No. 8 (1996), pp. 1–12.

[43] S. KIM, C. SHIN, AND J. KELLER, *High-frequency asymptotics for the numerical solution of the Helmholtz equation*, Appl. Math. Letters, 18 (2005), pp. 797–804.

[44] S. KIM AND W. SYMES, *Multigrid domain decomposition methods for the Helmholtz problem*, in Mathematical and Numerical Aspects of Wave Propagation, J. A. DeSanto, ed., SIAM, Philadelphia, 1998, pp. 617–619.

[45] P. LE TALLEC, *Domain decomposition methods in computational mechanics*, Comput. Mech. Advances, 1 (1994), pp. 121–220.

[46] H. LIM, S. KIM, AND J. DOUGLAS, JR., *Numerical methods for viscous and nonviscous wave equations*, Appl. Numer. Math., 57 (2007), pp. 194–212.

[47] P. LIONS, *On the Schwarz alternating method I*, in First International Symposium on Domain Decomposition Method for Partial Differential Equations, R. Glowinski, G. Golub, G. Meurant, and J. Periaux, eds., Philadelphia, PA, 1988, SIAM, pp. 1–42.

[48] ——, *On the Schwarz alternating method III: a variant for nonoverlapping subdomains*, in Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., Philadelphia, PA, 1990, SIAM, pp. 202–223.

[49] F. MAGOULÈS, F.-X. ROUX, AND L. SERIES, *Algebraic way to derive absorbing boundary conditions for the Helmholtz equation*, J. Comput. Acoust., 13 (2005), pp. 433–454.

[50] J. MANDEL, *Two-level domain decomposition preconditioning for the p-version finite element method in three dimensions*, Int. J. Numer. Methods Engrg., 29 (1990), pp. 1095–1108.

[51] G. MARCHUK, *Methods of numerical mathematics*, Springer-Verlag, New York, Heidelberg, and Berlin, 1982.

[52] L. MARINI AND A. QUARTERONI, *A relaxation procedure for domain decomposition methods using finite elements*, Numer. Math., 55 (1989), pp. 575–598.

[53] L. MCINNES, R. SUSAN-RESIGA, D. KEYES, AND H. ATASSI, *Additive Schwarz methods with nonreflecting boundary conditions for the parallel computation of Helmholtz problems*, in Domain Decomposition Methods 10, J. Mandel, C. Farhat, and X.-C. Cai, eds., vol. 218 of Contemporary Mathematics, Providence, RI, 1998, American Mathematical

Society, pp. 325–333. Proceedings of the Tenth International Conference on Domain Decomposition Methods, August 10-14, 1997, Boulder, CO.

[54] R. MEYER, *Introduction to mathematical fluid dynamics*, Dover Publications, Inc., New York, 1982.

[55] A. OSTROWSKI, *On the linear iteration procedures for symmetric matrices*, Rend. Mat. e Appl., 14 (1954), pp. 140–163.

[56] D. PEACEMAN AND H. RACHFORD, *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.

[57] A. QUARTERONI AND A. VALLI, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, New York, 1999.

[58] L. RUDIN, S. OSHER, AND E. FATEMI, *Nonlinear total variation based noise removal algorithms*, Physica D, 60 (1992), pp. 259–268.

[59] Y. SAAD AND M. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[60] H. SCHWARZ, *Ueber einige abbildungsaufgaben*, J. Reine Angew. Math., 70 (1869), pp. 105–120.

[61] A. SEI AND W. SYMES, *Dispersion analysis of numerical wave propagation and its computational consequences*, J. Sci. Comput., 10 (1995), pp. 1–27.

[62] P. STEIN AND R. ROSENBERG, *On the solution of linear simultaneous equations by iteration*, J. London Math. Soc., 23 (1948), pp. 111–118.

[63] H. STONE, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Numer. Anal., 5 (1968), pp. 530–558.

[64] J. C. STRIKWERDA, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks/Cole, Pacific Grove, California, 1989.

[65] O. TAUSSKY, *Bounds for characteristic roots of matrices*, Duke Math. J., 15 (1948), pp. 1043–1044.

[66] O. VACUS, *Mathematical analysis of absorbing boundary conditions for the wave equation: the corner problem*, Math. Comp., 74 (2005), pp. 177–200.

[67] R. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[68] ——, *Matrix Iterative Analysis, 2nd Ed.*, Springer-Verlag, Berlin, Heidelberg, 2000.

[69] S. WHITAKER, *Introduction to fluid mechanics*, R.E. Krieger Publishing Company, Malabar, Florida, 1968.

[70] O. WIDLUND, *Optimal iterative refinement methods*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., SIAM, Philadelphia, 1989, pp. 114–125.

[71] N. YANENKO, *Convergence of the method of splitting for the heat conduction equations with variable coefficients (English translation)*, USSR Comp. Math., 3 (1963), pp. 1094–1100.

[72] ——, *The method of fractional steps*, Springer-Verlag, Berlin, Heidelberg, and New York, 1971. (English translation; originally published in Russian, 1967).

Index

L_1-contraction, 256
θ-method, 87, 271
l_1-contracting method, 256
15% Rule, 158

ABM.mw, 48
abstract variational problem, 166
accuracy, 224
accuracy order, 89
Adams-Bashforth method, 47
Adams-Bashforth-Moulton method, 47
Adams-Moulton method, 47
adaptive method, 42
additive Schwarz method, 309
ADI method, 276
ADI-II, 291
advection form, 245
affine mapping, 196
alternating direction implicit method, 276
amplification factor, 80
average slope, 37

backward difference operator, 7
backward Euler method, 57, 87, 265
backward substitution, 122
banded matrix, 125
bandwidth, 125
Beam-Warming scheme, 243
Beyond Equations, 160
Black-Scholes differential equation, 14
boundedness, 83
Burgers's equation, 245

calculus of variations, 17
cardinal functions, 3, 182
Cauchy problem, 227
Cauchy-Schwarz inequality, 165, 167
cell Peclet number, 84
cell-centered FDM, 201
cell-centered finite difference method, 201
central difference operator, 7
CFL condition, 223
CG method, 143
characteristic equation, 20
characteristic function, 198
characteristics, 234
Chebyshev polynomials, 349
Chebyshev-Gauss formula, 351
Chebyshev-Gauss-Lobatto formula, 351
coarse subspace correction, 310
coercive, 167
collocation method, 172
column-wise point ordering, 138
common structures, 159
comparison functions, 18
condition number, 142, 311
conjugate gradient method, 143
conormal flux, 202
conservation, 85, 114
conservation form, 245
conservation laws, 227
conservation of mass, 339
conservation of momentum, 339
conservation principles, 337
conservative method, 245
consistency, 70, 115, 215, 249
consistent, 70
continuity equation, 339
control mass, 337
control volume, 198, 337
control volume equation, 338
control volumes, 201
convection, 13
convection-diffusion equation, 344
convection-diffusion problem, 83

convergence, 72, 116, 217
Coordinate Change, 103
coordinate change, 103
Courant number, 83, 250
Courant-Friedrichs-Lewy condition, 223
Crank-Nicolson method, 60, 88, 265, 275
Crank-Nicolson scheme, 225
curiosity, 158
curl, 354
curve fitting, 2
curve fitting approach, 9
cylindrical coordinate system, 105
cylindrical coordinates, 106

diagonal dominance, 131, 157
difference equation, 20
differential form, 230
differential problem, 168
diffusion equation, 12
direct algebraic solver, 122
directed graph, 129
Dirichlet-Neumann method, 320
discrete five-point Laplacian, 115, 119, 137
discrete maximum principle, 101, 116
discrete minimum principle, 101
dispersion, 240
dispersion analysis, 240
dispersion relation, 240
dispersive equation, 240
divergence, 188
divergence theorem, 188, 339, 354
divided differences, 4
dual problem, 185
duality argument, 185

eigenvalue locus theorem, 130
eigenvalue problem, 156
Einstein convention, 339
element stiffness matrix, 194
elliptic equation, 13
energy method, 77, 93
error analysis, 92
error equation, 73
error estimate for FEM, 183
essential boundary condition, 197
Euler equations, 227
Euler method, 28, 29
Euler's equation, 342
Euler-Lagrange equation, 15, 19
Euler.mw, 33
Euler.py, 58
Euler_T3.mw, 36
explicit scheme, 69
explicit schemes, 214
extensive property, 337

FD schemes, central 2nd-order, 355
FD schemes, central 4th-order, 355
FD schemes, one-sided 2nd-order, 356
FDM, 110
Fick's law, 344
financial freedom, 158
finite difference, 6
finite difference formulas, 355
finite difference method, 68, 110, 213, 332
finite element method, 168
finite volume method, 198, 201
first-order hyperbolic differential equation, 13
fixed-point iteration, 61
fluid mechanics, 13
flux conservation error, 324
flux function, 227
forward difference operator, 7
forward elimination, 122
forward Euler method, 57, 69, 87, 265
forward-backward difference matching, 327
Fourier transform, 96
Fourier's law, 344
Fourth-order Runge-Kutta method, 40
fractional-step method, 276
frequency, 240
fundamental period of the motion, 43
Fundamental Theorem of Calculus, 30

Galerkin method, 172, 176
Gauss elimination, 123
Gauss integration, 350
Gauss-Lobatto integration, 350
Gauss-Lobatto points, 182

Gauss-Seidel method, 133, 134
general parabolic problem, 66
generalized solution, 236
generic transport equation, 344
ghost grid value, 111
Gibbs notation, 342
global error, 39, 40
global point index, 120
Godunov theorem, 259
Godunov's method, 250
gradient, 140
Green's formula, 166, 189
group velocity, 241

$H^r(\Omega)$-norm, 164
$H^s(\mathbb{R}^2)$-norm, 97
harmonic average, 202
harmonic extension, 315
harmonic function, 98
heat equation, 12
Hessian, 140
Heun's method, 39
high-order Galerkin methods, 203, 334
higher-order FEMs, 175
higher-order Taylor methods, 34
Hilbert space, 164
homogeneous heat equation, 67
hyperbolic, 227
hyperbolic equation, 13

ILU, 147
image denoising, 15
image processing, 15
implicit ordinary differential equation, first-order, 61
incomplete LU factorization, 147
incomplete LU-factorization, 147
incompressible fluid flows, 13
initial value problem, 25, 26, 28, 37
integral form, 228
integration by parts, 19, 168, 197
intensive property, 338
interior regularity estimate, 97
interpolation error theorem, 5
interpolation estimate, 184

IODE-1, 61
IODE1.py, 62
irreducible matrix, 128
isothermal equations, 233
isothermal flow, 233
IVP, 26

Jacobi method, 133, 134
Jacobian, 103, 196

kinematic viscosity coefficient, 342
Krylov subspace method, 140

L^2-norm, 164
Lagrange form of interpolating polynomial, 3
Lagrange interpolating polynomial, 3
Lagrange multiplier, 17
Laplacian operator, 66
Lax-Friedrichs scheme, 214, 248
Lax-Milgram Lemma, 166
Lax-Milgram lemma, 171
Lax-Richtmyer Equivalence Theorem, 78, 222
Lax-Wendroff scheme, 239
leapfrog scheme, 214
learning compound interest, 159
least-square approach, 172
Legendre polynomials, 348
Legendre-Gauss formula, 351
Legendre-Gauss-Lobatto formula, 351
line relaxation methods, 137
line SOR method, 156
linear FEM, 174
linear Galerkin FEM, 177, 193
linear Galerkin method, 177
linear iterative method, 126
linear space, 168, 190
linear system, 178
Lipschitz condition, 26, 31
Lipschitz continuity, 249
local truncation error, 39, 40
locally one-dimensional method, 276
LOD method, 276
lower triangular matrix, 122
LU factorization, 122, 123

LU factorization with partial pivoting, 124

M-matrix, 132
m-step method, 46
mass conservation, 339
material derivative, 338
matrix splitting, 291
maximum principle, 66, 83, 91, 98, 101, 116, 157, 268
mean value theorems, 98
mesh points, 28
minimization problem, 169, 190
minimum principle, 98, 101
mixed derivatives, 114
modelcode, 1
modified equation, 239
Modified Euler method, 39
momentum conservation, 339
momentum conservation equation, 339
monotone method, 258
monotonicity preserving method, 255
multi-step methods, 46
multiplicative Schwarz method, 306

natural boundary condition, 197
Navier-Stokes (NS) equations, 13
Navier-Stokes equations, 341
Neumann-Neumann method, 322
Newton form of the interpolating polynomial, 3
Newton polynomial, 3
Newtonian fluid, 340
nodal point, 128, 181
non-dimensionalization, 342
nonlinear stability, 251
nonoverlapping DD method, 313
nonrectangular domain, 332
numerical flux function, 246

one-sided 2nd-order FD schemes, 356
optimal step length, 141
order of accuracy, 224
order of convergence, 44, 45
orthogonal polynomials, 347
Ostrowski's theorem, 136
outer bordering, 112, 330
outward unit normal, 188
overlapping Schwarz method, 305

parabolic boundary, 66
parabolic equation, 12
Parseval's identity, 79, 96
PCG, 146
PCG-ILU0, 283
Peclet number, 84
permutation matrix, 128
Petrov-Galerkin method, 172
phase velocity, 241
pivot, 123
Poincaré inequality, 167, 206
point relaxation method, 137
Poisson equation, 104, 108, 332
polytropic gas, 232
positive definite, 180
preconditioned CG method, 146
Python code, 150, 203, 260

quadrature, 349
quasilinear elliptic equation, 96
quasilinear second-order elliptic equation, 96

Rayleigh-Ritz method, 172
red-black coloring, 320
reducible matrix, 128
reference element, 195
regular splitting, 132, 147
regularity estimate, 97, 165
relaxation methods, 133, 137
relaxation parameter, 135
Reynolds number, 14, 343
Reynolds's transport equation, 338
Rich Dad Poor Dad, 160
right-hand rule, 354
RK2, 38, 39
RK3, 61
RK4, 40
RK4.mw, 41
RK4SYS.mw, 52
RK4SYSTEM.mw, 55
RKF45, 42
Robert Kiyosaki, 160

Robin method, 323
row-wise point ordering, 119, 120
Runge-Kutta method, third-order, 61
Runge-Kutta methods, 37
Runge-Kutta-Fehlberg method, 42

SAM, 302
Schur complement matrix, 316, 318
Schwarz alternating method, 302
search direction, 140
second-order hyperbolic differential equation, 13
Second-order Runge-Kutta method, 38, 39
semi-implicit method, 88, 265
SIP, 147
solution, 100
SOR method, 133, 135, 156
space-time slice, 68, 213
SPD, 311
specific heat, 232
spectral radius, 127
spectrum, 127
spherical coordinates, 106, 107
spline, 172
spring-mass system, 43
stability, 22, 77, 220
stability analysis, 20
stability condition, 81
stability theory, 22
state equations, 232
steepest descent method, 140
Steklov-Poincaré interface equation, 316
Steklov-Poincaré operator, 316
step length, 140
step-by-step methods, 27
stiffness matrix, 194
Stokes's equations, 343
Stokes's theorem, 354
strain tensor, 340
stress tensor, 340
strong maximum principle, 98
strong minimum principle, 98
strong stability, 22
strongly connected, 157
strongly connected directed graph, 129
strongly hyperbolic, 210
strongly implicit procedure, 147
subharmonic function, 98
subsolution, 100
successive over-relaxation method, 135
super-convergence, 203
superharmonic function, 98
supersolution, 100
symmetric positive definite, 140
symmetric positive definite matrix, 311
symmetrization, 112

Taylor method of order m, 34
Taylor series, 28
Taylor series approach, 8
Taylor's formula, 68
Taylor's theorem, 2
Taylor-series methods, 28
thermal diffusivity, 12
three-term recurrence relation, 347
topic examples, 159
total variation, 252
total variation diminishing method, 254
total variation model, 15
total variation stability, 252
transmission conditions, 314
trapezoid method, 60
trapezoidal domain, 104, 332
trial functions, 176
trigonometric formulas, 353
truncation error, 71
TV model, 15
TV-stability, 252
TVD method, 254

unconditional stability, 226
unconditionally stable, 87
unconditionally unstable, 86
upper triangular matrix, 122
upwind scheme, 244, 248

V-elliptic bilinear form, 166
vanishing-viscosity approach, 235
variational calculus, 15, 17
variational formulation, 168
variational problem, 169, 190

vector identities, 353
von Neumann analysis, 77, 79, 222

wave equation, 13
wave number, 240
weak formulation, 168
weak maximum principle, 99
weak minimum principle, 99
weak solution, 236
weight function, 176
weighted residual approach, 172, 176
well-posed equation, 12