University Mohamed Boudiaf of M'Sila

Faculty of Technology
Department of Mechanical Engineering

Advanced Numerical Methods

Course Handouts / Lecture Brochures

Master's Degree in Academics — Energetics

Academic Year 2025–2026

Dr. DJERAD ABDELKADER

Chapter 1. Numerical Methods for First-Order ODEs

Chapter Objectives

- Understand the key concepts and the digital issue.
- Build an algorithm (steps + pseudo-code) and apply it.
- Interpret the error and check the stability/convergence on a simple case.

Table of Contents

1.1. Ordinary Differential Equations of the First Order

1.1.1. Introduction to ODE

We are interested in the resolution of ODE of the first order of the form:

$$x'(t) = f\big(t, x(t)\big), t \geq t^0 \tag{1.1}$$

$$x(t^0) = \eta \tag{1.2}$$

where $\mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is a given function.

Theorem (Existence and Uniqueness).

Let f be a globally Lipschitzian function with respect to x. Thus, there is a constant $L \in \mathbb{R}$ of Lipschitz such that for all $t \in \mathbb{R}, \forall (x_1, x_2) \in \mathbb{R}^{2d}, \forall t \in [t^0, t^0 + T]$:

$$f(t, x_1) - f(t, x_2)\| \leq L\|x_1 - x_2\| \tag{1.3}$$

Then, the problem admits a single solution $x \in C^1([t^0, t^0 + T])$.

1.1.2. ODE Systems

1.1.3. Higher-order equations

An equation of order m can be transformed into a first-order system.

Example. The Van der Pol oscillator:

$$x''(t) + 10\big(1 - x^2(t)\big)x'(t) + x(t) = sin(\pi t) \tag{1.6}$$

By setting $u = x$ and $v = x'$, we obtain:

$$u'(t) = v(t)$$

$$\boldsymbol{v'(t) = -10\left(1 - u^2(t)\right)v(t) + u(t) + sin(\pi t)}$$

1.2. Taylor Series Development

1.2.1. Reminder of the Taylor series

For a function f of class C^{p+1} on $[a, b]$, the Taylor expansion is written:

$$f(x) = f(a) + (x-a)f'(a) + (x-a)^2/2!\, f''(a) + \cdots + (x-a)^p/p!\, f^{(p)}(a) + R_p(x) \tag{1.7}$$

where the remainder $R_p(x)$ can be written:

$$R_p(x) = f^{(p+1)}(\xi)/p!\ (x-a)(x-\xi)^p, \xi \in]a, x[\tag{1.8}$$

1.2.2. Application to ODEs

If $x(t)$ is a solution of $x'(t) = f\big(t, x(t)\big)$ and is sufficiently regular, we have:

$$x(t+h) = x(t) + hx'(t) + h^2/2x''(t) + h^3/6x'''(t) + o(h^4) \tag{1.9}$$

Using $x'(t) = f\big(t, x(t)\big)$, we can calculate the higher-order derivatives:

$$x''(t) = f_t + x'(t) f_x = f_t + f f_x$$

$$\boldsymbol{x'''(t) = ftt + 2fft_x + f^2 f_{xx} + f_x(ft + ff_x)}$$

1.2.3. Landau notation

We write $eh = 1 + O(h)$ if there are two positive constants h_0 and C such that:

$$|z| \leq Chp, \forall 0 < h < h^0 \tag{1.10}$$

Examples:

- $e^h = 1 + O(h)$
- $eh = 1 + h + O(h^2)$
- $eh = 1 + h + h^2/2 + O(h^3)$

1.3. Euler's Method and Error Propagation

1.3.1. Derivation of the explicit Euler method

Using Taylor's development:

$$x(t + h) = x(t) + hx'^{(t)} + R_1(t) \quad (1.11)$$

where $R^1(t) = h^2/2x''(\xi)$ for a certain $\xi \in]t, t + h[$.

By substituting $x'^{(t)} = f\big(t, x(t)\big)$ and neglecting the remainder term, we obtain the explicit Euler method:

$$x_n{}^{+1} = x_n + hf(t_n, x_n), n = 0,1, \dots, N - 1 \quad (1.12)$$

$$x^0 = \eta \quad (1.13)$$

where $t_n = t_0 + n\,h$ and $t_n = t_0 + nh$ *and* $h = (t_x - t_0)/N$.

1.3.2. Errors: Local and Global

Definition (Local truncation error).

$$\varepsilon_n = x(t_n{}^{+1}) - x(t_n) - hf\big(t_n, x(t_n)\big) \quad (1.14)$$

For Euler, $\varepsilon n = h^2/2x''(\xi)$with $\xi \in]t_n, t_{n+1}[$, so $|\varepsilon_n| \leq h^2/2\,\|x''\|\infty = O(h^2)$

Definition (Global error).

$$e_n = x(t_n) - x_n \quad (1.15)$$

1.3.3. Consistency

Definition.

A method is said to be consistent of order p if $\varepsilon_n = O(hp^{+1})$.

Euler's method is consistent of order 1.

1.3.4. Stability

Gronwall's Lemma.

Let $u \in C^1([0,T])$*such that* $|u'(t)| \leq a(t)|u(t)| + b(t)$ *with* $a(t) \geq 0, b(t) \geq 0$, $[0,T]$,*continuous on* $[0,T]$, *then:*

$$|u(t)| \leq |u(0)|\, exp(\int_0 t\, a(s)ds) + \int_0 t\, b(\sigma)\, exp(\int \delta t\, a(s)ds)d\sigma \quad (1.16)$$

Theorem (Explicit Euler stability).

Euler's scheme $x_n{}^{+1} = x_n + hf(t_n, x_n)$*is stable, meaning:* $\exists M \geq 0$ *independent of* n *such that* $\forall (z_n)_n \in \mathbb{N}$ *verifying* $z_{n+1} = z_n + hf(t_n, z_n) + g_n$*, the following inequality is satisfied:*

$$\|x_n - z_n\| \leq M[\|x_0 - z_0\| + \Sigma_{k=0}{}^{n-1} |g_k|], 0 \leq n \leq N \tag{1.17}$$

where $M = e^{LT}$ where L is the Lipschitz constant.

1.3.5. Convergence

Theorem.

The explicit Euler method is convergent of order 1:

$$sup^0 \leq {}_n \leq {}_n\|e_n\| \leq (hTe^{LT}/2)\|x''\|\infty \tag{1.18}$$

Proof.

Consistency + Stability ⇒ Convergence

1.3.6. Implicit Euler method

Using Taylor's backward development:

$$x\, x(t_n) = x(t_n{}^{+1}) - hx'(t_n{}^{+1}) + h^2/2x''(\xi) \tag{1.19}$$

We obtain the implicit Euler method:

$$x_n{}^{+1} = x_n + hf(t_n{}^{+1}, x_n{}^{+1}) \tag{1.20}$$

This method is also consistent of order 1, stable and convergent.

1.3.7. Error propagation

The overall error propagates according to:

$$e_n{}^{+1} \leq (1 + hL)e_n + |\varepsilon_n| \tag{1.21}$$

By applying recursively, we obtain:

$$e_n \leq \Pi_{j=0}{}^{n-1}(1 + hL)e_0 + \Sigma_{k=0}{}^{n-1}|\varepsilon_k|\Pi_{j=k+1}{}^{n-1}(1 + hL) \tag{1.22}$$

Since $1 + x \leq e^x$, we have:

$$e_n \leq e^{(tn-t0)L}e_0 + \Sigma_{k=0}{}^{n-1}|\varepsilon_k|e^{(tn-t_{k+1})L} \tag{1.23}$$

1.4. Runge-Kutta Methods

1.4.1. General Principle

On the interval $[t_n, t_{n+1}]$,we have:

$$x(t_{n+1}) = x(t_n) + \int_{tn}^{tn+1} f(s, x(s))ds \quad (1.24)$$

Change of variable $s = t_n + \sigma h_n$::

$$x(t_{n+1}) = x(t_n) + h_n \int_0^1 g(\sigma)d\sigma, g(\sigma) = f(t_n + \sigma h_n, x(t_n + \sigma h_n)) \quad (1.25)$$

The idea is to approximate the integral by a quadrature formula:

$$\int_0^1 g(\sigma)d\sigma \approx \Sigma_{i=1}^{s} b_i g(c_i \quad (1.26)$$

1.4.2. Explicit Runge-Kutta Methods

Definition.

A Runge-Kutta method with s stages is given by:

$$k_i = f(t_n + c_i h_n, x_n + h_n \Sigma_j^{=1i-1} a_{ij} k_j), i = 1, \dots, s \quad (1.27)$$

$$k_i = f(t_n + c_i h_n, x_n + h_n \Sigma_j^{=1i-1} a_{ij} k_j), i = 1, \dots, s$$

$$x_n^{+1} = x_n + h_n \Sigma_i^{=1s} b_i k_i \quad (1.28)$$

where $ci = \Sigma j^{=1i-1} aij$ and $a_{ij} = 0$ for $j \geq i$ (explicit).

1.4.3. Butcher's Table

We represent an RK method by its Butcher table:

Table 1.1. ***General Butcher tableau of a Runge-Kutta method***

c	**A**
	b^T

where $c = (c^1, \dots, c_s)^T, b = (b^1, \dots, b_s)^T, and A = (a_{ij})$.

1.4.4. Examples of RK Methods

RK of order 2 (Runge method — Trapezoids).

$$k^1 = f(t_n, x_n)$$

$$k^2 = f(t_n + h, x_n + hk^1)$$

$$\boldsymbol{x_n^{+1} = x_n + h/2(k^1 + k^2)}$$

RK of order 4 (Classical RK4).

$$k^1 = f(t_n, x_n)$$

$$k^2 = f(t_n + h/2, x_n + hk^1/2)$$

$$k^3 = f(t_n + h/2, x_n + hk^2/2)k^4 = f(t_n + h, x_n + hk^3)$$

$$\boldsymbol{x_n{}^{+1} = x_n + h/6(k^1 + 2k^2 + 2k^3 + k^4)}$$

1.4.5. Order Conditions for RK Methods

To obtain the order p, we need to check some algebraic relations.

Order 1:

$$\Sigma_i{}^{=1s} b_i = 1 \tag{1.29}$$

Order 2:

$$\Sigma_i{}^{=1s} b_i = 1, \Sigma_i{}^{=1s} b_i c_i = 1/2 \tag{1.30}$$

Order 3:

$$\Sigma_i{}^{=1s} b_i = 1, \Sigma_i{}^{=1s} b_i c_i = 1/2, \Sigma_i{}^{=1s} b_i c_i{}^2 = 1/3 \Sigma_{i,j} b_i a_{ij} c_j = 1/6$$

Order 4: 11 conditions to be checked.

1.4.6. Assessment of errors for RK

Local error for an RK method of order p:

$$\varepsilon_n = x(t_n{}^{+1}) - x_n - h\Sigma_i{}^{=1s} b_i k_i = O(h^{p+1}) \tag{1.31}$$

Stability Function.

Applying an RK method to $x' = \lambda x$, we obtain:

$$x_n{}^{+1} = R(\hat{h}) x_n \tag{1.32}$$

where $\hat{h} = \lambda h$ and $R(\hat{h})$is the stability function. For explicit methods:

$$R(\hat{h}) = 1 + \hat{h} + \hat{h}^2/2! + \cdots + \hat{h}^p/p! \tag{1.33}$$

1.5. Multi-Step Methods

1.5.1. Principle

Multistep methods use the history of values calculated at previous time steps. General form (k-step method):

$$x_n{}^{+k} + \alpha_k{}^{-1} x_n{}^{+k-1} + \cdots + \alpha^0 x_n = h(\beta_k f_n{}^{+k} + \beta_k{}^{-1} f_n{}^{+k-1} + \cdots + \beta^0 f_n) \tag{1.34}$$

with $\alpha_k = 1$ and $f_n = f(t_n, x_n)$. The method is explicit if $\beta k = 0$ and implicit if $\beta k \neq 0$.

1.5.2. Characteristic polynomials

First characteristic polynomial.

$$\rho(r) = \Sigma l^{=0} k \alpha l r l, \alpha k = 1 \tag{1.35}$$

Second characteristic polynomial.

$$\sigma(r) = \Sigma_l^{=0k}\beta_l r^l \tag{1.36}$$

1.5.3. Consistency of Multistep Methods

Theorem.

A multistep method is consistent if and only if:

$$\rho(1) = 0 et \rho'(1) = \sigma(1) \tag{1.37}$$

Order.

A multistep method is of order p if $C^0 = C^1 = \cdots = Cp = 0$ where the coefficients C_m are defined by the expansion of the linear difference operator.

1.5.4. Zero-stability

Definition.

A multistep method is said to be zero-stable if its first characteristic polynomial $\rho(r)$ satisfies the root condition: all its roots are included in $B(0,\bar{1})$, and the roots on the unit circle are simple.

Theorem (Dahlquist, 1956).

A multistep method is convergent if and only if it is consistent and zero-stable.

1.5.5. Classical families of multistep methods

Adams-Bashforth (1883) — Explicit.

$$\rho(r) = r^k - r^{k-1}, ordrep = k. \tag{1.38}$$

$$AB(2): x_n{}^{+2} - x_n{}^{+1} = h/2(3f_n{}^{+1} - f_n)$$

$$\boldsymbol{AB(3): x_n{}^{+3} - x_n{}^{+2} = h/12(23f_n{}^{+2} - 16f_n{}^{+1} + 5f_n)}$$

Adams-Moulton (1926) — Implicit.

$$\rho(r) = r^k - r^{k-1}, ordrep = k + 1. \tag{1.39}$$

$$\boldsymbol{AM(2): x_n{}^{+2} - x_n{}^{+1} = h/12(5f_n{}^{+2} + 8f_n{}^{+1} - f_n)}$$

Backward Differentiation Formulas (BDF, 1952).

$$\sigma(r) = \beta_k r^k, ordrep = k. \tag{1.40}$$

$$\boldsymbol{BDF(2): 3x_n{}^{+2} - 4x_n{}^{+1} + x_n = 2hf_n{}^{+2}}$$

1.5.6. Dahlquist Barriers

First barrier.

The order p of a k-step multistep method satisfies:

- $p \leq k + 2$ if k is even

- $p \leq k + 1$ if k is odd
- $p \leq k$ if $p \leq k$ if $\beta k \leq 0$ (explicit methods)

Second barrier.

- There is no explicit A-stable multistep method.
- An A-stable multistep method cannot have an order $p > 2$.
- The smallest A-stable method of error constant is the trapezoid rule.

1.6. Prediction-Correction Methods

1.6.1. Principle

Prediction-correction methods combine an explicit (predictor) and an implicit (corrector) method. General scheme:

11. Prediction: Calculate an approximate value ${x_{n+1}}^{(0)}$ with an explicit method.
12. Correction: Improve this value by using an implicit method.

1.6.2. Example: Euler Predictor-Corrector Method

Predictor (explicit Euler).

$${u_n}^{+1} = x_n + hf(t_n, x_n) \tag{1.41}$$

Corrector (trapezoidal).

$${x_n}^{+1} = x_n + h/2[f(t_n, x_n) + f({t_n}^{+1}, {u_n}^{+1})] \tag{1.42}$$

1.6.3. PECE Methods

Predict-Evaluate-Correct-Evaluate (PECE) methods are commonly used:

- P: Predict with Adams-Bashforth.
- E: Evaluate f at the predicted point.
- C: Correct with Adams-Moulton.
- E: Evaluate f at the corrected point.
-

1.7. Absolute Stability

1.7.1. Test Equation

Consider the test equation:

$$x'(t) = \lambda x(t), \lambda \in \mathbb{C}, Re(\lambda) < 0 \tag{1.43}$$

The exact solution is $x(t) = ce\lambda^t$ which tends to 0 when $t \to +\infty$.

1.7.2. Definitions

Absolute stability.

A numerical method is said to be absolutely stable for a fixed h-step if, when applied to the test equation with $\hat{h} = \lambda h$, its solution tends to 0 when $n \to \infty$ for any initial data.

Region of absolute stability.

The set $\mathcal{R}$ of values of $\hat{h} \in \mathbb{C}$ for which the method is absolutely stable.

A-stability.

A method is A-stable if $\mathcal{R} \supset \hat{h} \in \mathbb{C}: Re(\hat{h}) < 0$.

1.7.3. Stability of one-step methods

For a one-step method applied to $x' = \lambda x$:

$$x_n{}^{+1} = R(\hat{h})x_n \tag{1.44}$$

where $R(\hat{h})$is the stability function.

Explicit Euler.

$$R(\hat{h}) = 1 + \hat{h}, \mathcal{R}_0 =]-2, 0[\tag{1.45}$$

Implicit Euler.

$$R(\hat{h}) = 1/(1-\hat{h}), \mathcal{R} = \hat{h}: Re(\hat{h}) < 0 (A-stable) \tag{1.46}$$

Trapezoidal.

$$R(\hat{h}) = (1+\hat{h}/2)/(1-\hat{h}/2), \mathcal{R} = \hat{h}: Re(\hat{h}) < 0 (A-stable) \tag{1.47}$$

1.7.4. Stability of Runge-Kutta methods

For explicit RK methods of order s:

$$R(\hat{h}) = 1 + \hat{h} + \hat{h}^2/2! + \cdots + \hat{h}s/s! \tag{1.48}$$

No explicit RK method is A-stable because $lim|R(\hat{h})| = \infty$ when $\hat{h} \to -\infty$.

1.8. Applications to Boundary Layer Problems

1.8.1. Boundary layer equations on a flat plate

The boundary layer equations for an incompressible two-dimensional flow are written:

Continuity.

$$\partial u/\partial x + \partial v/\partial y = 0 \tag{1.49}$$

Momentum.

$$u\partial u/\partial x + v\partial u/\partial y = \nu\partial^2 u/\partial y^2 \quad (1.50)$$

Boundary Conditions.

- $u(x,0) = 0, v(x,0) = 0$ (adhesion to the wall)
- $u(x,\infty) = U\infty$ (external velocity)
- $u(0,y) = U\infty$ (input condition)

1.8.2. Similarity Transformation (Blasius)

By introducing the similarity variable:

$$h = y\sqrt{(U\infty/nx)} \quad (1.51)$$

and the stream function ψ such that $u = \partial\psi/\partial y, v = -\partial\psi/\partial x$ we set:

$$f(\eta) = \psi/\sqrt{(\nu x U\infty)} \quad (1.52)$$

We obtain Blasius' equation:

$$f''' + \frac{1}{2} f f'' = 0 \quad (1.53)$$

with the conditions:

$$f(0) = 0, f'(0) = 0, f'(\infty) = 1 \quad (1.54)$$

1.8.3. Transformation to a first-order system

Setting $f_1 = f, f_2 = f', f_3 = f''$, we obtain:

$$f_1' = f_2$$
$$f_2' = f_3$$
$$f_3' = -\frac{1}{2} f_1 f_3$$

where $f_1(0) = 0, f_2(0) = 0, f^2(\infty) = 1$.

1.8.4. Shooting method

The boundary problem is transformed into the Cauchy problem:

13. Initialization: Choose an initial value s for $f_3(0)$.
14. Integration: Solve the system with an RK4 method from $\eta = 0$ to $\eta = \eta\infty$.
15. Verification: Check if $f^2(\eta\infty) \approx 1$.
16. Iteration: Fit s by Newton-Raphson method until convergence.

$$s_k{}^{+1} = s_k - [f^2(\eta\infty; s_k) - 1]/[\partial f^2/\partial s(\eta\infty; s_k)] \quad (1.55)$$

1.8.5. Forced Convection

For forced convection with heat transfer, we add the energy equation:

$$u\partial T/\partial x + v\partial T/\partial y = \alpha\partial^2 T/\partial y^2 \quad (1.56)$$

With the similarity transformation, we obtain:

$$\theta'' + (1/2)Pr \cdot f\theta' = 0 \quad (1.57)$$

where $\theta = (T - Tv)/(T\infty - Tv) and\ \ Pr = \nu/\alpha$ and $Pr = \nu/\alpha$ (Prandtl number).

The coupled system becomes:

$$f''' + \frac{1}{2}ff'' = 0$$

$$\boldsymbol{\theta'' + (1/2)Pr \cdot f\theta' = 0}$$

1.8.6. Natural Convection

For natural convection, the Grashof number $Gr = g\beta\Delta TL^3/\nu^2$ and the Rayleigh number $Ra = Gr \cdot Pr$ are introduced.

The momentum equation becomes:

$$u\partial u/\partial x + v\partial u/\partial y = \nu\partial^2 u/\partial y^2 + g\beta(T - T\infty) \quad (1.58)$$

1.8.7. Digital Resolution Strategy

Resolution steps:

Spatial discretization.

Create an η mesh with refinement close to the wall:

$$\eta_i = \eta_{max}(i/N)^a, i = 0,1, \dots, N \quad (1.59)$$

where $\alpha > 1$ to refine near $\eta = 0$.

Method of Resolution.

- Use RK4 for integration.
- Adaptive step to control local error.
- Stopping criterion: $f'(\eta\infty) > 0.99$.

Shooting method.

- Search for $f''^{(0)}$ by dichotomy or Newton.
- Tolerance: $f'(\eta\infty) > 0.99$.

Post-processing.

- Coefficient of friction: $C_x = 2f''(0)/\sqrt{Re_x}$.
- Nusselt number: $Nu_x = -\theta'(0)\sqrt{Re_x}$.
- Boundary layer thickness: $\delta^{99} = \eta^{99}/\sqrt{(Re_x/x)}$.

1.8.8. Numerical Example: Flat Plate (Blasius)

Data:

- $U\infty = 1\ m/s$
- $\nu = 1.5 \times 10^{-5} m^2/s$
- $h\infty = 10$

Expected results:

- $f''(0) \approx 0.33206$
- $\boldsymbol{C_x \approx 0.664/\sqrt{Re_x}}$

1.9. Summary and Comparison of Methods

1.9.1. Comparison Table

Table 1.2. ***Comparison of numerical methods for first-order ODEs***

Method	Order	Absolute stability	Cost	Notes
Euler Explicit	1	]−2, 0[	Low	Simple, not very precise
Implicit Euler	1	A-stable	Medium	Needs resolution
Trapezoidal	2	A-stable	Medium	Good compromise
RK4	4	]−2.78, 0[	High	Widely used
Adams-Bashforth	k	Limited	Medium	Requires start-up
BDF	k	A-stable ($k \leq 6$)	Medium	For stiff problems

1.9.2. Practical Recommendations

For non-stiff problems:

- Use RK4 or RK45 (Adaptive).
- High order with moderate pitch.

For stiff problems:

- Use implicit methods (BDF, Radau).
- Essential A-stability.
- Accept implicit resolution cost.

For large systems:

- Multistep methods (lowest cost per step).
- Predictor-corrector.
- Adaptive Step Methods.

1.9.3. Error Control and Adaptive Step

Error estimation:

- Comparing two methods of different orders.
- Richardson extrapolation.
- Embedded methods (e.g., Dormand-Prince RK45).

Adaptation criterion:

$$h_{new} = h_{old} \times (tol/err)^{(1/(p+1))} \times security\_factor \quad (1.60)$$

where p is the order and security_factor ≈ 0.9.

1.10. Conclusion

Numerical methods for ODEs offer a wide range of tools suitable for different types of problems:

One-step methods: Euler, Runge-Kutta.

- Easy to implement.
- Self-starting.
- Easy to adapt the step.

Multistep methods: Adams, BDF.

- Efficiency for long integrations.
- Reuse of information.
- Requires startup.

Essential properties:

- Consistency (local precision).
- Stability (error propagation).
- Convergence (overall accuracy).

Applications:

- Boundary layers.
- Forced and natural convection.
- Stiff problems.
- Large systems.

The choice of method depends on the stiffness of the problem, the precision required, the acceptable cost of calculation, and the nature of the application.

References

J.C. Butcher, Numerical Methods for Ordinary Differential Equations, 2nd edition, John Wiley & Sons, 2008.

E. Hairer, S.P. Nørsett and G. Wanner, Solving Ordinary Differential Equations I, Nonstiff Problems, 3rd edition, Springer, 2008.

D.F. Griffiths and D.J. Higham, Numerical Methods for Ordinary Differential Equations, Initial Value Problems, Springer, 2010.

A. Quarteroni, R. Sacco and F. Saleri, Numerical Methods, Algorithms, Analysis and Applications, Springer, 2007.

Corrected Examples (results + method)

Example 1.1 (Explicit Euler).

Statement. Solve $y' = -2y, y(0) = 1$ over $[0, 1]$ with $h = 0.25$.

Solution (abstract). Scheme:

$y_{n+1} = y_n + h(-2y_n) = (1 - 2h)y_n. With h = 0.25 \Rightarrow 0.5$

With $h = 0.25 \Rightarrow$ factor 0.5. $y_0 = 1, y_1 = 0.5, y_2 = 0.25, y_3 = 0.125, y_4 = 0.0625$. Exact: $y(1) = e^{-2} \approx 0.1353$.

Example 1.2 (Implicit Euler).

Statement. Same ODE with implicit Euler and $h = 0.25$.

Solution (summary). Scheme:

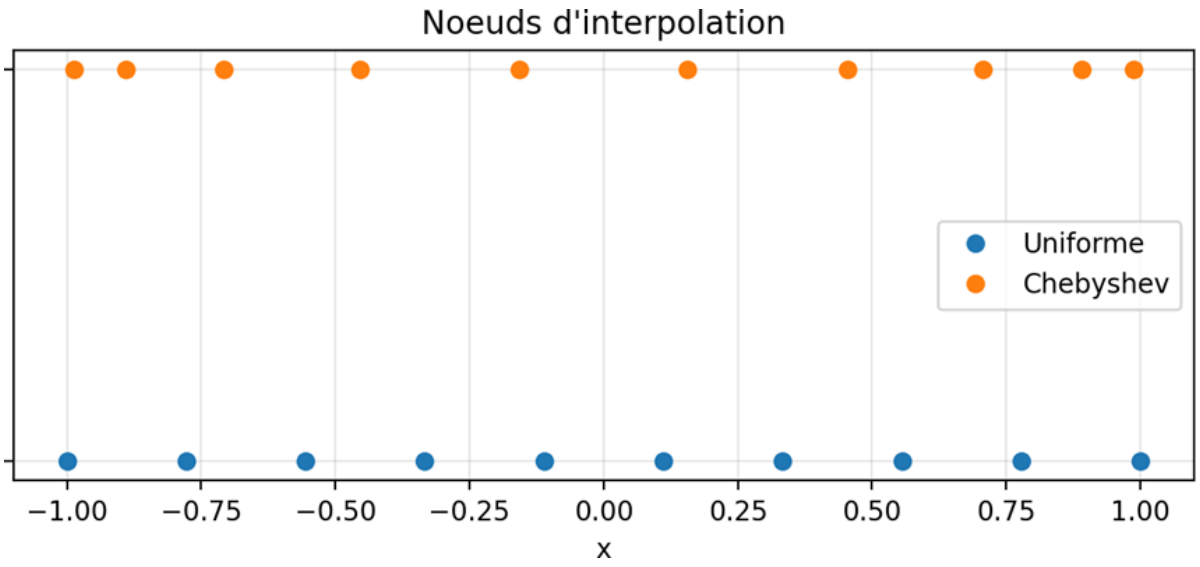
$y_{n+1} = y_n + h(-2y_{n+1}) \Rightarrow y_{n+1} = y_n/(1 + 2h) = y_n/1{,}5.$

$y_0 = 1, y_1 \approx 0.6667, y_2 \approx 0.4444, y_3 \approx 0.2963, y_4 \approx 0.1975.$

Chapter 2 — Finite Difference Methods

Chapter 2. Finite Difference Methods

Solving Partial Differential Equations



Chapter Objectives

- Understand the key concepts and the digital issue.
- Build an algorithm (steps + pseudo-code) and apply it.
- Interpret the error and check the stability/convergence on a simple case.

Table of Contents

2.1. Introduction to Finite Difference Methods

The finite difference method represents a fundamental numerical approach to solving partial differential equations (PDEs). Its principle is based on the transformation of a continuous problem into a discrete problem by approximation of derivatives using finite differences.

2.1.1. Historical Context

Initially developed in the 18th century by Euler, this method has experienced considerable growth with the advent of computers. It remains today one of the most widely used techniques in numerical simulation, particularly for thermal conduction, fluid mechanics and electromagnetism problems.

2.1.2. Areas of application

Finite differences are useful in many areas:

- Heat transfer and heat conduction
- Structural mechanics and stress analysis
- Fluid flows and aerodynamics
- Wave propagation and acoustics
- Diffusion and mass transport

2.1.3. Benefits and limitations

Advantages:

- Design simplicity and ease of implementation.
- Applicability to various geometries with structured meshes.
- Computational efficiency for rectangular domains.
- Well-established mathematical analysis.

Limitations:

- Difficulty with complex geometries.
- Need for regular meshing for optimal accuracy.
- Delicate handling of non-standard boundary conditions.

2.2. Fundamentals of Discretization

2.2.1. Approximation of derivatives

The essence of the method lies in the approximation of differential operators by difference operators. Let us consider a sufficiently regular function $u(x)$.

Forward Difference.

$$u'(x_i) \approx [u(x_i{}^{+1}) - u(x_i)]/h \tag{2.1}$$

Truncation error: $O(h)$.

Backward Difference.

$$u'(x_i) \approx [u(x_i) - u(x_i{}^{-1})]/h \tag{2.2}$$

Truncation error: $O(h)$.

Centered Difference.

$$u'(x_i) \approx [u(x_i) - u(x_i{}^{-1})]/h \tag{2.3}$$

Truncation error: $O(h^2)$.

2.2.2. Second Derivatives

For second derivatives, we mainly use centered differences that offer better precision:

$$u''(x_i) \approx [u(x_i{}^{+1}) - 2u(x_i) + u(x_i{}^{-1})]/h^2 \tag{2.4}$$

With a truncation error in $O(h^2)$.

2.2.3. Extension to the two-dimensional case

For a two-dimensional problem on a rectangular domain, we introduce a Cartesian mesh with h_x and h_y steps:

$$\partial^2 u/\partial x^2 \approx [u_i{}^{+1}{}_{,j} - 2u_{i,j} + u_i{}^{-1}{}_{,j}]/h \tag{2.5}$$

$$\partial^2 u/\partial y^2 \approx [u_{i,j}{}^{+1} - 2u_{i,j} + u_{i,j}{}^{-1}]/h_y^2 \tag{2.6}$$

2.2.4. Concept of stencil

A stencil represents the pattern of points in the mesh used to approximate a differential operator at a given point. For the 2D Laplacian, the 5-point stencil is the most common:

Table 2.1. ***The 5-point stencil for the 2D Laplacian operator***

	$u_{i,j+1}$	
$u_{i-1,j}$	$u_{i,j}$	$u_{i+1,j}$
	$u_{i,j-1}$	

2.3. Model Problem: Two-Dimensional Thermal Conduction

2.3.1. Mathematical Formulation

Consider the problem of stationary thermal conduction in a homogeneous rectangular plate. This problem is governed by the Poisson equation:

$$\partial^2 T/\partial x^2 + \partial^2 T/\partial y^2 = -q(x,y)/k \quad (2.7)$$

where:

- $T(x,y)$: temperature distribution.
- $q(x,y)$: source term (heat generation).
- k: thermal conductivity of the material.

2.3.2. Boundary Domain and Conditions

The domain considered is a rectangle $\Omega = [0, L_x] \times [0, L_y]$.

Dirichlet boundary conditions:

$$T(0,y) = T^1, T(L_x,y) = T^2 \quad (2.8)$$

$$\partial T/\partial n = 0 \quad (2.9)$$

Neumann boundary conditions:

$$\partial T/\partial n = 0 (zero flux, thermal insulation) \quad (2.10)$$

Mixed conditions:

Combination of Dirichlet and Neumann on different boundaries.

2.3.3. Classification: elliptic equation

The Poisson equation belongs to the class of elliptic equations. These equations are characterized by:

- A steady or stationary state.
- Spreading information in all directions.
- A unique solution under conditions at the appropriate limits.
- Data-dependent consistency of the solution.

2.3.4. Physical interpretation

Physically, this problem models the temperature distribution at thermal equilibrium. At each point, the temperature is the result of an equilibrium between:

- The flow of heat entering by conduction.
- The outgoing heat flow.
- Internal heat generation (if present).

The solution therefore represents the stable, time-independent thermal configuration resulting from boundary conditions and internal generation.

2.4. Construction of the Algebraic System

2.4.1. Discretization of the domain

We subdivide the rectangular domain into a uniform grid:

$$x_i = i \cdot h_x, i = 0, 1, \dots, N_x \; avec \; h_x = L_x/N_x \tag{2.11}$$

$$y_j = j \cdot h_y, j = 0,1, \dots, N_y avec h_y = L_y/N_y \tag{2.12}$$

This grid contains $(N_x + 1) \times (N_y + 1)$ points in total. The interior points, where the solution is unknown, are $(N_x - 1) \times (N_y - 1)$.

2.4.2. 5-Point Numerical Scheme

Applying the finite difference approximations to the Laplacian, we obtain for each interior point (i, j):

$$[T_i{}^{+1}{}_{,j} - 2T_{i,j} + T_i{}^{-1}{}_{,j}]/h_x{}^2 + [T_{i,j}{}^{+1} - 2T_{i,j} + T_{i,j}{}^{-1}]/h_y^2 = -q_{i,j}/k \tag{2.13}$$

To simplify, consider the case $h_x = h_y = h$. The equation becomes:

$$T_i{}^{+1}{}_{,j} + T_i{}^{-1}{}_{,j} + T_{i,j}{}^{+1} + T_{i,j}{}^{-1} - 4T_{i,j} = -h^2 q_{i,j}/k \tag{2.14}$$

2.4.3. Matrix Formulation

The system of equations can be written in matrix form $A\,X = B$, where:

- X: vector of unknown temperatures at interior points.
- B: second member vector including source terms and boundary conditions.
- A: stiffness matrix, sparse and structured.

2.4.4. Structure of the Matrix

Matrix A has a particular structure:

- Dimension: $N \times N$ with $N = (N_x - 1) \times (N_y - 1)$.
- Band: block pentadiagonal.
- Symmetry: A is symmetric positive definite.
- Sparsity: about 5 N non-zero elements on N^2 elements.
- Diagonal dominance: property guaranteeing stability.

This particular structure is crucial for the choice of solving methods, especially iterative methods that exploit sparsity.

2.4.5. Numbering of Unknowns

There are two main numbering strategies:

Row-wise numbering:

$$k = i + (j - 1)(N_x - 1) for the point (i, j) \quad (2.15)$$

Advantage: block tridiagonal matrix.

Column-wise numbering:

$$k = (i - 1)(N_y - 1) + j for point (i, j) \quad (2.16)$$

Advantage: alternative depending on the algorithm used.

2.5. Direct Solving Methods

2.5.1. General Principle

Direct methods aim to calculate the exact solution (to the nearest machine precision) of the linear system in a finite number of operations. They are generally based on matrix factorizations.

2.5.2. Gauss Elimination

The Gaussian elimination algorithm transforms the system into a superior triangular system by successive elimination of unknowns.

Complexity:

- Operations: $O(N^3)$for a dense matrix.
- Memory: $O(N^2)$.

For our problem, the exploitation of the sparse structure significantly reduces this complexity.

2.5.3. LU Factorization

The LU factorization decomposes the matrix A into:

$$A = L\,U \tag{2.17}$$

where L is lower triangular and U is upper triangular.

Algorithm:

11. Factorization: Calculate L and U.
12. Descent: Solve $L\,y = b$.
13. Ascent: Solve $U\,x = y$.

Advantages:

- Reuse of factorization for multiple second members.
- Numerical stability with partial pivoting.

2.5.4. Cholesky's method

For a symmetric positive definite matrix like our matrix A, Cholesky factorization is preferable:

$$A = L\,L^T \tag{2.18}$$

Specific benefits:

- Memory saving: only L is stored.
- Compute gain: approximately 2× faster than LU.
- Stability: no pivoting required.
- Complexity: $O(N^3/6)$vs. $O(N^3/3)$ for LU.

2.5.5. Exploitation of the sparse structure

For large sparse matrices resulting from finite differences, specialized methods drastically reduce costs:

- Storage: CSR, CSC formats (Compressed Sparse Row/Column).
- Reordering: minimize fill-in during factorization.
- Effective complexity: $O\left(N^{(3/2)}\right)$in 2D, $O(N^2)$ in 3D.

2.5.6. Limitations of Direct Methods

Despite their advantages, direct methods have limitations:

- Prohibitive memory cost for very large grids ($N > 10^6$).
- Excessive computation time in dimension 3.
- Difficulty of efficient parallelization.

These limitations motivate the use of iterative methods for large problems.

2.6. Iterative Resolution Methods

2.6.1. General Principle

Iterative methods construct a sequence of vectors $\{X^{(k)}\}$converging to the exact solution X^*. They start from an initial estimate $X^{(0)}$ and successively apply a recurrence formula.

Stopping criteria:

$$| X^{(k+1)} - X^{(k)} | < \varepsilon \quad or \quad | A X^{(k)} - B | < \varepsilon \tag{2.19}$$

2.6.2. Jacobi's method

Jacobi's method decomposes $A = D + R$ where D is the diagonal of A:

$$X^{(k+1)} = D^{-1}\left(B - RX^{(k)}\right) \tag{2.20}$$

For our conduction problem, this is written point by point:

$$T_{i,j}^{(k+1)} = \left[T_{i}^{+1,(k)}_{,j} + T_{i}^{-1,(k)}_{,j} + T_{i,j}^{+1(k)} + T_{i,j}^{-1(k)} + h^2 q_{i,j}/k\right]/4 \tag{2.21}$$

Features:

- Easily parallelizable.
- Slow convergence.
- Requires two arrays (old and new).

2.6.3. Gauss-Seidel method

Gauss-Seidel improves Jacobi by immediately using the updated values:

$$T_{i,j}^{(k+1)} = \left[T_{i}^{+1,(k)}_{,j} + T_{i}^{-1,(k+1)}_{,j} + T_{i,j}^{+1(k)} + T_{i,j}^{-1(k+1)} + h^2 q_{i,j}/k\right]/4 \tag{2.22}$$

Advantages:

- Converges about 2 times faster than Jacobi.
- Only one array needed.
- Easy to implement.

Drawbacks:

- Difficult to parallelize effectively.

2.6.4. Successive Over-Relaxation Method (SOR)

SOR introduces a relaxation parameter ω to accelerate convergence:

$$T_{i,j}^{(k+1)} = (1-\omega)T_{i,j}^{(k)} + \omega \cdot \left[T_{i}^{+1},_{j}^{(k)} + T_{i}^{-1},_{j}^{(k+1)} + T_{i,j}^{+1(k)} + T_{i,j}^{-1(k+1)} + h^2 q_{i,j}/k\right]/4 \quad (2.23)$$

Choosing the \omega parameter:

- $\omega = 1$: we recover Gauss-Seidel.
- $\omega < 1$: under-relaxation (stabilization).
- $\omega > 1$: over-relaxation (acceleration).

For our square grid problem, the theoretical optimal parameter is:

$$\omega_{opt} = 2/[1 + sin(\pi h)] \approx 2 - 2\pi h \quad for small h \quad (2.24)$$

With ω optimal, SOR converges about N times faster than Gauss-Seidel.

2.6.5. Gradient Methods

Gradient methods exploit the symmetric positive definite structure of A.

Gradient with optimal step:

Minimizes error at each iteration in the gradient direction.

Conjugate gradient:

Uses conjugate descent directions, ensuring theoretical convergence in N iterations (without rounding errors).

Practical Performance:

- Convergence in $O(\sqrt{N})$ iterations with preconditioning.
- Well suited for very large sparse matrices.
- Low memory cost: no additional matrix storage.

2.6.6. Convergence and Spectral Analysis

The rate of convergence of iterative methods depends on the spectral radius of the iteration matrix $\rho(M)$:

$$| E^{(k)} | \leq C\, \rho(M)^k\, | E^{(0)} | \quad (2.25)$$

For our problem:

- Jacobi: $\rho \approx 1 - \pi^2 h^2$ (slow convergence).
- Gauss-Seidel: $\rho \approx 1 - 2\pi^2 h^2$
- Optimal SOR: $\rho \approx 1 - 2\pi h$ (best convergence).

2.7. Convergence Acceleration Techniques

2.7.1. Preconditioning

Preconditioning consists of transforming the $A\,X = B$ system into an equivalent system with better spectral properties:

$$M^{-1}\,A\,X = M^{-1}\,B \tag{2.26}$$

where M is a preconditioning matrix.

Qualities of a good preconditioner:

- M close to A to reduce conditioning.
- Easy to calculate M^{-1} (low cost).
- Preserved sparse structure.

Common Preconditioners:

- Diagonal (Jacobi): $M = D$.
- Incomplete LU (ILU): approximate factorization.
- Incomplete Cholesky (IC): for symmetric matrices.
- Multigrid: grid hierarchy.

2.7.2. Multigrid Methods

Multigrid methods exploit a fundamental property: classical iterative methods quickly eliminate the high-frequency components of the error but slowly eliminate the low-frequency ones.

Principle:

14. Smoothing: a few iterations on the fine grid.
15. Restriction: transfer of residue to coarse grid.
16. Resolution: dealing with the error on the coarse grid.
17. Extension: interpolation to the fine grid.
18. Correction: updating the solution.

Performance:

Convergence in $O(N)$ operations, optimal for elliptic problems. The number of iterations is independent of the mesh size.

2.7.3. Domain Decomposition

For problems with complex geometry, domain decomposition divides the problem into separately treated subdomains.

Schwarz's method:

- Subdomain overlap.
- Information exchange at interfaces.
- Convergence by iterations between subdomains.

Advantages:

- Natural parallelization.
- Adaptability to complex geometries.
- Ability to use different solvers per subdomain.

2.7.4. Richardson Extrapolation

Extrapolation improves accuracy by combining solutions obtained with different discretization steps.

If $u_h = u + Ch^p + O\left(h^{(p+1)}\right)$, then also calculating $u_{h/2}$, we get:

$$\hat{u} = \left[2^p u_{(h/2)} - u_h\right]/(2^p - 1) = u + O\left(h^{(p+1)}\right) \quad (2.27)$$

Gain an order of accuracy without further refining the mesh.

2.8. Multi-Step Methods and ADI

2.8.1. Introduction to ADI Methods

ADI (Alternating Direction Implicit) methods are a particularly effective strategy for multidimensional problems. They break down a 2D or 3D problem into a sequence of 1D problems that are easier to solve.

2.8.2. Principle of the Classical ADI Method

For the 2D Poisson equation, the ADI method proceeds in two successive half-steps.

First half-step (implicit in x direction, explicit in y):

$$\left[T_{i}^{+1,\,j}{}^{(k+1/2)} - 2T_{i,\,j}{}^{(k+1/2)} + T_{i}^{-1,\,j}{}^{(k+1/2)}\right]/h_x{}^2 + \left[T_{i,\,j}{}^{+1(k)} - 2T_{i,\,j}{}^{(k)} + T_{i,\,j}{}^{-1(k)}\right]/h_y^2 = -q_{i,\,j}/k \quad (2.28)$$

Second half-step (implicit in y-direction, explicit in x):

$$\left[T_{i}^{+1}{}_{,j}^{(k+1/2)} - 2T_{i,j}^{(k+1/2)} + T_{i}^{-1}{}_{,j}^{(k+1/2)}\right]/h_x^2 + \left[T_{i,j}^{+1(k+1)} - 2T_{i,j}^{(k+1)} + T_{i,j}^{-1(k+1)}\right]/h_y^2 = -q_{i,j}/k \quad (2.29)$$

2.8.3. Benefits of the ADI Approach

Effective resolution:

Each half-step requires the resolution of tridiagonal systems (one dimension at a time), which can be solved in $O(N)$ operations by Thomas' algorithm.

Unconditional stability:

For scalability issues, ADI allows the use of large time steps without numerical instability.

Overall cost:

Total complexity of $O(N)$ per iteration for a problem with N unknowns, much lower than the $O\left(N^{(3/2)}\right)$of direct methods.

2.8.4. Thomas Algorithm (TDMA)

Thomas' algorithm efficiently solves the tridiagonal systems appearing at each ADI half-step:

$$a_i\, T_{i-1} + b_i\, T_i + c_i\, T_{i+1} = d_i, \quad i = 1, \dots, n \quad (2.30)$$

Forward sweep:

$$c'_i = c_i/(b_i - a_i c'_i{}^{-1}), d'_i = (d_i - a_i d'_i{}^{-1})/(b_i - a_i c'_i{}^{-1}) \quad (2.31)$$

Backward substitution:

$$T_n = d'_n, \quad T_i = d'_i - c'_i\, T_{i+1} \quad (2.32)$$

2.8.5. Extension to three-dimensional problems

For 3D problems, ADI naturally expands into three half-steps, each dealing with a direction implicitly:

19. Implicit x, y and z explicit direction.
20. Implicit y, x and z explicit direction.
21. Implicit z, x and y explicit direction.

This approach maintains $O(N)$ complexity per iteration, making ADI particularly attractive for 3D simulations.

2.9. Douglas-Rachford Scheme

2.9.1. Motivation and context

The Douglas-Rachford scheme, proposed in 1955, represents a significant improvement over conventional ADI methods. It offers better stability and convergence, especially for parabolic and elliptical problems.

2.9.2. Schema Formulation

For a stationary conduction problem, Douglas-Rachford introduces an acceleration parameter and reformulates the alternating approach:

$$(I + rA_x)T^{(k+½)} = (I - rA_y)T^{(k)} + rb \tag{2.33}$$

$$(I + rA_y)T^{(k+1)} = (I - rA_x)T^{(k+½)} + rb \tag{2.34}$$

where:

- A_x and A_y are the branch operators in the x and y directions.
- r is an acceleration parameter to be optimized.
- I is the identity matrix.

2.9.3. Choosing the Acceleration Parameter

The parameter r plays a crucial role in the rate of convergence. For the Poisson problem on a rectangular domain:

Theoretical optimal value:

$$r_opt = 2/(\lambda_min + \lambda_max) \tag{2.35}$$

where λ_{min} and λ_{max} are the extreme eigenvalues of the discrete Laplacian.

Practical approximation:

For a uniform mesh with h pitch:

$$r_opt \approx h^2/(4\pi^2) \tag{2.36}$$

2.9.4. Convergence Analysis

The Douglas-Rachford convergence factor is expressed as a function of the ratio of the extreme eigenvalues $K = \lambda_{max}/\lambda_{min}$:

$$\rho_{DR} \approx \frac{\sqrt{K} - 1}{\sqrt{K} + 1} \tag{2.37}$$

This dependence on $\sqrt{K}$ (instead of K for classical methods) explains the much faster Douglas-Rachford convergence.

2.9.5. Advantages over classic ADI

Accelerated convergence:

Douglas-Rachford typically converges 2 to 3 times faster than standard ADI.

Enhanced stability:

Less sensitive to parameter choice and complex boundary conditions.

Extended applicability:

Works well for a wide range of problems, including non-symmetric operators.

2.9.6. Variants and extensions

Douglas-Rachford with multiple parameters:

Use of a cyclic sequence of parameters $r^1, r^2, \dots, r_m$ to further accelerate convergence.

Peaceman-Rachford:

Symmetrical variant where both half-steps use the same parameter.

D'Yakonov:

Extension to problems with variable coefficients and non-rectangular geometries.

2.9.7. Practical Implementation

Complete algorithm for a Douglas-Rachford iteration:

1. Calculate the modified second member: $b^* = \left(I - rA_y\right) T^{(k)} + r\, b$.
2. Solve $(I + rA_x)T^{(k+½)} = b *$ line by line (TDMA).
3. Calculate $b^{**} = (I - rA_x)T^{(k+½)} + rb$.
4. Solve $\left(I + rA_y\right)T^{(k+1)} = b^{**}$ column by column (TDMA).
5. Check convergence: $| T^{(k+1)} - T^{(k)} | < \varepsilon$.

2.10. Optimization and Performance Analysis

2.10.1. Performance Criteria

The evaluation of a numerical method is based on several criteria:

Precision:

- Truncation error: $O(h^p)$ where p is the order of the method.
- Global error: influence of the accumulation of local errors.

Computational efficiency:

- Total calculation time.
- Number of operations per iteration.
- Number of iterations needed.

Memory cost:

- Matrix storage.
- Intermediate vectors.

Robustness:

- Numerical stability.
- Parameter sensitivity.
- Behaviour with various boundary conditions.

2.10.2. Comparison of Methods

Table 2.2. ***Comparison of resolution methods for the 2D Poisson problem***

Method	Complexity/iter	Convergence	Memory
Direct Gaussian	$O(N^3)$	1 step	$O(N^2)$
Jacobi	$O(N)$	$O(N)$ $iter$	$O(N)$
Gauss-Seidel	$O(N)$	$O(N)$ $iter$	$O(N)$
Optimal SOR	$O(N)$	$O(\sqrt{N})$ $iter$	$O(N)$
ADI / Douglas-R.	$O(N)$	$O(\sqrt{N})$ $iter$	$O(N)$

For large problems ($N > 10^4$), iterative methods with ADI or Douglas-Rachford offer the best efficiency/cost trade-off.

2.10.3. Matrix Structure Example

For a 1D implicit scheme with $N = 5$ interior points, the tridiagonal matrix takes the form (with $Fo = \alpha\Delta t / \Delta x^2$):

Table 2.3. ***Example tridiagonal matrix*** A = tridiag(-Fo, 1+2Fo, -Fo)

1+2Fo	**-Fo**	0	0	0
-Fo	**1+2Fo**	**-Fo**	0	0
0	**-Fo**	**1+2Fo**	**-Fo**	0
0	0	**-Fo**	**1+2Fo**	**-Fo**
0	0	0	**-Fo**	**1+2Fo**

2.10.4. Mesh Optimization

Adaptive Mesh:

Locally refine the mesh in areas with high gradient to maintain accuracy while limiting the total number of points.

Refinement criteria:

- Gradient of the solution.
- Local error estimation.
- Post-hoc indicators.

Strategies:

- h-adaptation: changing the size of the meshes.
- p-adaptation: modification of local order.
- r-adaptation: redistribution of points.

2.10.5. Parallelization

The use of parallel architectures significantly accelerates calculations:

Domain decomposition:

- Spatial domain partitioning.
- Each processor processes a subdomain.
- Communication at interfaces.

Method parallelism:

- Jacobi: naturally parallel.
- SOR: requires red/black ordering of the mesh.
- ADI: parallelism by rows/columns.

Effectiveness:

The actual acceleration depends on the computation/communication ratio. Methods with high locality (conjugate gradient, multigrid) are more parallel.

2.10.6. Validation Techniques

Manufactured solutions:

Choose an analytical solution u_{exact}, calculate the corresponding source term, and then check the order of convergence.

Successive refinement:

Calculate with $h, h/2, h/4, \ldots$ and check that $| u_h - u_{h/2} | \sim C \cdot h^p$.

Conservation assessments:

Verify that calculated flows comply with physical retention laws.

2.10.7. Practical Recommendations

For small to medium-sized problems ($N < 10^4$):

- Direct methods (Cholesky with sparsity).
- Maximum precision.
- Reasonable computation time.

For large problems ($N < 10^4$):

- Douglas-Rachford or optimal SOR.
- ILU or IC preconditioning.
- Preconditioned conjugate gradient as a last resort.

For very large problems ($N > 10^6$):

- Algebraic multigrid.
- Parallel domain decomposition.
- Massively parallel conjugate gradient.

Conclusion

Finite difference methods are a powerful and versatile tool for the numerical solution of elliptic partial differential equations. Understanding the different approaches — direct, classical iterative, and advanced techniques such as ADI and Douglas-Rachford — allows you to choose the optimal strategy according to the size and characteristics of the problem.

Key points to remember:

- Finite difference discretization transforms a PDE into an algebraic system whose properties determine the choice of the method of resolution.
- Direct methods guarantee the exact solution but become prohibitive for large systems.
- Classical iterative methods (Jacobi, Gauss-Seidel, SOR) are memory-saving but converge slowly without optimization.
- ADI approaches, and particularly the Douglas-Rachford scheme, offer a much improved convergence for multidimensional problems.
- Optimizing convergence parameters and exploiting the particular structure of the arrays are essential to achieve optimal performance.

Mastering these techniques paves the way for the efficient resolution of complex physical problems in thermal, mechanical, and many other fields of engineering and applied physics.

References and Further Reading

R.L. Burden and J.D. Faires, Numerical Analysis, 9th edition, Brooks/Cole, 2010.

W.H. Press et al., Numerical Recipes: The Art of Scientific Computing, 3rd edition, Cambridge University Press, 2007.

R.J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations, SIAM, 2007.

J.W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods, Springer, 1995.

D.M. Young, Iterative Solution of Large Linear Systems, Academic Press, 1971.

G.H. Golub and C.F. Van Loan, Matrix Computations, 4th edition, Johns Hopkins University Press, 2013.

Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd edition, SIAM, 2003.

Corrected Examples (results + method)

Example 2.1 (Poisson 1D, FD).

Statement. Solve $-u''(x) = 1, u(0) = u(1) = 0$ with $N = 4$ ($h = 0.25$).

Solution (summary). $2\,u_i - u_{i-1} - u_{i+1} = h^2 = 0.0625$.

$2\,u_1 - u_2 = 0.0625;\ -u_1 + 2\,u_2 - u_3 = 0.0625;\ -u_2 + 2\,u_3 = 0.0625$.

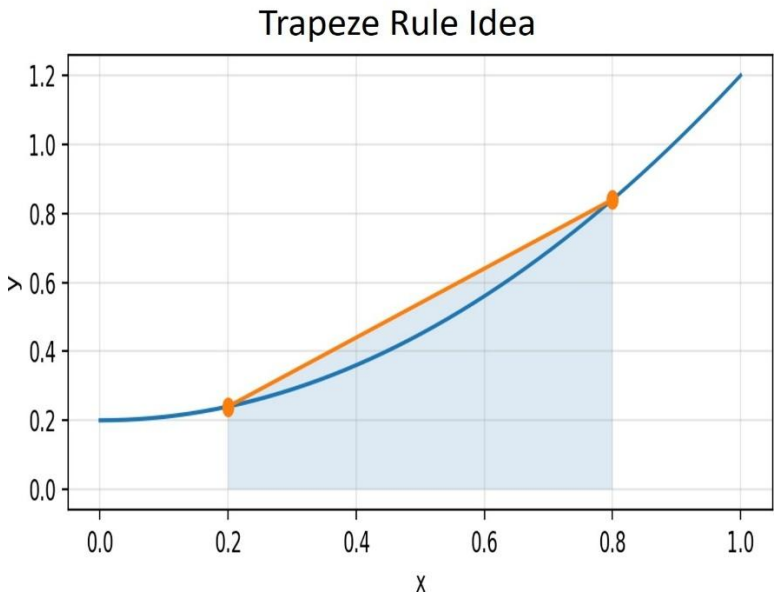
Solution: $u_1 = 0.09375;\ u_2 = 0.125;\ u_3 = 0.09375$.

Example 2.2 (TDMA).

Statement. Solve the tridiagonal system by TDMA.

Solution (summary). Coefficients: $a = -1$, $b = 2$, $c = -1$. Forward elimination then backward substitution. Result: $u_1 = 0.09375;\ u_2 = 0.125;\ u_3 = 0.09375$.

Chapter 3 — Parabolic Equations



Chapter 3. Parabolic Equations

Unsteady Thermal Conduction and Diffusion

Chapter Objectives

- Understand the key concepts and the digital issue.
- Build an algorithm (steps + pseudo-code) and apply it.
- Interpret the error and check the stability/convergence on a simple case.

Table of Contents

Part I — One-dimensional problems

3.1. Introduction to Parabolic Equations

3.1.1. Nature of parabolic equations

Parabolic partial differential equations represent a fundamental class of evolutionary problems in physics and engineering. Unlike elliptic equations that describe states of equilibrium, parabolic equations model time-dependent propagation and diffusion phenomena.

Canonical form:

$$\partial u/\partial t = \alpha \nabla^2 u + f(x,t) \tag{3.1}$$

where:

- $u(x,t)$: dependent variable (temperature, concentration, etc.).
- α: diffusion coefficient.
- $f(x,t)$: source term.
- ∇^2: Laplacian operator.

3.1.2. Key features

Temporal evolution:

The solution evolves from an initial state to a state of equilibrium (if one exists), characterized by a propagation of information over time.

Regularizing effect:

The initial discontinuities are instantly smoothed by the diffusion process, giving a high regularity to the solution for $t > 0$.

Irreversibility:

Time plays a privileged role: the solution evolves in a single temporal direction, reflecting the irreversibility of the diffusion processes.

Area of influence:

A local disturbance propagates at infinite speed (parabolic characteristic), theoretically affecting the entire domain instantaneously.

3.1.3. Associated physical phenomena

Parabolic equations model many phenomena:

Heat Transfer:

Thermal conduction in solids and fluids at rest.

Mass diffusion:

Pollutant transport, molecular scattering, chromatography.

Viscous flows:

Boundary layer development, momentum diffusion.

Financial phenomena:

Black-Scholes equation for valuing options.

Biology:

Diffusion of chemical substances, population models.

3.1.4. Specificities of discretization

Numerical solving of parabolic equations presents specific challenges:

- The need to discretize both space and time.
- Compromise between stability and precision.
- CFL (Courant-Friedrichs-Lewy) conditions for explicit schemes.
- Computational cost of implicit schemes.
- Choice of time step according to spatial step.

3.2. 1D Unsteady Conduction Modeling

3.2.1. One-dimensional heat equation

Consider a homogeneous bar of length L in which the heat propagates only along the longitudinal axis. The governing equation is written:

$$\partial T/\partial t = \alpha \partial^2 T/\partial x^2 + q(x,t)/\left(\rho c_p\right) \tag{3.2}$$

where:

- $T(x,t)$: temperature at point x and time t.
- $\alpha = k/\left(\rho c_p\right)$: thermal diffusivity.
- k: thermal conductivity.
- ρ: density.
- c_p: specific heat.

- $q(x,t)$: heat volume generation.

3.2.2. Formulation of the complete problem

Initial condition:

Temperature distribution at the initial time.

$$T(x,0) = T_0(x) \quad for \ 0 \le x \le L \tag{3.3}$$

Boundary Conditions.

Dirichlet type (imposed temperature):

$$T(0,t) = T_a, \quad T(L,t) = T_b \tag{3.4}$$

Neumann type (imposed flux):

$$\partial T/\partial x|_x{}^{=0} = q_a/k, \partial T/\partial x|_x{}^{=}l = q_b/k \tag{3.5}$$

Robin type (convection):

$$k\partial T/\partial x|_{x=0} = h(T - T\infty) \tag{3.6}$$

3.2.3. Spatial discretization

Division of the domain $[0, L]$ into N equal intervals:

$$x_i = i \cdot \Delta x, \quad i = 0, 1, \dots, N, \quad \Delta x = L/N \tag{3.7}$$

The temperature at point i and time n is denoted $T_i{}^n \approx T(x_i, t_n)$ with $t_n = n \cdot \Delta t$.

3.2.4. Approximation of the second derivative

By centered finite differences, we obtain:

$$\partial^2 T/\partial x^2|i \approx (Ti_{+1} - 2Ti + Ti_{-1})/(\Delta x)^2 \tag{3.8}$$

With a truncation error in $O((\Delta x)^2)$.

3.2.5. Fourier number

The key dimensionless parameter for stability analysis is the Fourier number:

$$Fo = \alpha \Delta t/(\Delta x)^2 \tag{3.9}$$

This number represents the ratio between the diffusion rate and the thermal storage rate. It plays a decisive role in the stability of numerical schemes.

3.3. Explicit Schemes

3.3.1. General Principle

Explicit schemes approximate the time derivative by a forward difference and calculate the solution at time $n + 1$ only from the values known at time n.

3.3.2. FTCS Scheme (Forward Time Central Space)

By approximating $\partial T / \partial t$ by forward difference:

$$\partial T/\partial t|_i^n \approx (T_i^{n+1} - T_i^n)/\Delta t \quad (3.10)$$

We obtain the classical explicit scheme:

$$Ti^{n+1} = Ti^n + Fo(Ti^{+1n} - 2Ti^n + Ti^{-1n}) \quad (3.11)$$

Or in expanded form:

$$T_i^{n+1} = Fo \cdot T_i^{-1n} + (1 - 2Fo) \cdot T_i^n + Fo \cdot T_i^{+1n} \quad (3.12)$$

3.3.3. Calculation algorithm

Walkthrough:

12. Initialize: $T_i^0 = T_0(x_i)$ for $i = 0, \ldots, N$.
13. For each time step $n = 0, 1, 2, \ldots$:

- Apply the boundary conditions: T_0^{n+1} and T_N^{n+1}.
- For $i = 1, \ldots, N-1$: calculate $T_i^{n+1} = \mathrmFo \cdot T_{i-1}^n + (1 - 2\mathrmFo) T_i^n + \mathrmFo \cdot T_{i+1}^n$.

14. Check stopping criterion (final time or convergence).

3.3.4. Stability Analysis

By the von Neumann method, we analyze the growth of a Fourier mode:

$$T_i^n = G^n exp(i\beta x_i) \quad (3.13)$$

The amplification factor G is written:

$$G = 1 - 2Fo + 2Fo \cdot cos(\beta\Delta x) = 1 - 4Fo \cdot sin^2(\beta\Delta x/2) \quad (3.14)$$

Stability Condition:

So that $|G| \leq 1$ for all modes, you must:

$$Fo \leq 1/2 \Leftrightarrow \Delta t \leq (\Delta x)^2/(2\alpha) \quad (3.15)$$

This restrictive condition imposes very small time steps to ensure stability.

3.3.5. Precision Analysis

Local truncation error — by Taylor development:

$$\tau i^n = O(\Delta t) + O((\Delta x)^2) \quad (3.16)$$

The scheme is therefore of order 1 in time and order 2 in space.

Consistency:

When $\Delta t \to 0$ and $\Delta x \to 0$, the truncation error tends to zero. The scheme is therefore consistent.

3.3.6. Pros and Cons

Advantages:

- Extremely easy to implement.
- No linear system resolution.
- Low memory cost.
- Parallelizable naturally.
- Direct calculation, without iteration.

Drawbacks:

- Very restrictive stability condition ($Fo \leq 1/2$).
- Requires very small time steps.
- Potentially high total compute time.
- Ineffective for stiff problems.
- Accumulation of errors over long periods of time.

3.4. Implicit Schemes

3.4.1. Motivation

To overcome the stability limitation of explicit schemes, implicit methods evaluate spatial derivatives at time $n+1$ rather than time n. This approach provides unconditional stability at the cost of linear system resolution at every step.

3.4.2. BTCS Scheme (Backward Time Central Space)

Approximating the spatial derivative at time $n+1$:

$$(T_i^{n+1} - T_i^n)/\Delta t = \alpha(T_i^{+1n+1} - 2T_i^{n+1} + T_i^{-1n+1})/(\Delta x)^2 \quad (3.17)$$

Rearrangement:

$$Fo \cdot T_i^{-1n+1} + (1 + 2Fo) \cdot T_i^{n+1} - Fo \cdot T_i^{+1n+1} = T_i^n \quad (3.18)$$

3.4.3. Matrix Formulation

The complete system is written in matrix form:

$$A \cdot T^{n+1} = T^n \quad (3.19)$$

where A is a tridiagonal matrix:

$$A \cdot T^{n+1} = T^n \quad (3.20)$$

3.4.4. System Resolution — Thomas algorithm (TDMA)

Thanks to the tridiagonal structure, the system is efficiently solved in $O(N)$ operations.

Forward sweep phase:

For $i = 2, \dots, N-1$:

$$m_i = -Fo/(1 + 2Fo - Fo \cdot m_i^{-1}) \quad (3.21)$$

$$d_i = (T_i^n + Fo \cdot d_i^{-1})/(1 + 2Fo - Fo \cdot m_i^{-1}) \quad (3.22)$$

Backward substitution phase:

$$T_{N-1}^{n+1} = d_{N-1} \quad (3.23)$$

For $i = N - 2, \dots, 1$:

$$T_i^{n+1} = d_i - m_i \cdot T_{i+1}^{n+1} \quad (3.24)$$

3.4.5. Unconditional Stability

Von Neumann's analysis gives the amplification factor:

$$G = \frac{1}{1 + 4\,Fo\,.sin^2(\beta\Delta x/2)} \quad (3.25)$$

For all $\mathrmFo > 0$, we have $|G| < 1$, so the scheme is unconditionally stable. No restrictions on the time step are necessary for stability.

3.4.6. Properties of the Implicit Scheme

Precision:

Order $O(\Delta t)$ in time, $O((\Delta x)^2)$ in space.

Numerical dissipation:

The scheme introduces an artificial dissipation that dampens high frequencies, contributing to stability but can affect accuracy.

Choice of time step:

Although stable for any Δt, accuracy nevertheless imposes practical limits. A common criterion is $Fo \approx 1$ for a good precision/efficiency trade-off.

3.4.7. Explicit / Implicit Comparison

Table 3.1. ***Comparison between explicit and implicit 1D schemes***

Criterion	Explicit	Implicit
Stability	$\mathrm{Fo} \leq 1/2$	Unconditional
Cost/step	$O(N)$	$O(N)$
Storage	Minimal	Matrix A
Time accuracy	$O(\Delta t)$	$O(\Delta t)$
Optimal usage	Short durations	Long durations

3.5. Crank-Nicolson Scheme

3.5.1. Principle of the Method

The Crank-Nicolson scheme, proposed in 1947, represents an optimal approach combining the advantages of explicit and implicit methods. It is based on an average of the spatial derivatives at times n and $n+1$, leading to a time-centered scheme.

3.5.2. Derivation of the Scheme

We approximate the equation at the intermediate time $n + 1/2$:

$$(T_i^{n+1} - T_i^n)/\Delta t = \alpha/2 \cdot [(\partial^2 T/\partial x^2)_i^n + (\partial^2 T/\partial x^2)_i^{n+1}] \tag{3.26}$$

This gives:

$$T_i^{n+1} - T_i^n = Fo/2 \cdot [(T_i^{+1n} - 2T_i^n + T_i^{-1n}) + (T_i^{+1n+1} - 2T_i^{n+1} + T_i^{-1n+1})] \tag{3.27}$$

3.5.3. Reorganized Form

Grouping the terms at time $n+1$ on the left:

$$Fo/2 \cdot T_i^{-1n+1} + (1 + Fo) \cdot T_i^{n+1} - Fo/2 \cdot T_i^{+1n+1} = Fo/2 \cdot T_i^{-1n} + (1 - Fo) \cdot T_i^n + Fo/2 \cdot T_i^{+1n} \tag{3.28}$$

3.5.4. Matrix Formulation

The system is written:

$$A \cdot T^{n+1} = B \cdot T^n \tag{3.29}$$

where:

$$A = \text{tridiag}\left(-\frac{Fo}{2}, 1 + Fo, -\frac{Fo}{2}\right)(Fo/2) \quad (3.30)$$

$$B = \text{tridiag}\left(\frac{Fo}{2}, 1 - Fo, \frac{Fo}{2}\right) \quad (3.31)$$

3.5.5. Remarkable properties

Unconditional stability — von Neumann's analysis:

$$G = [1 - 2Fo \cdot sin^2(\beta \Delta x/2)]/[1 + 2Fo \cdot sin^2(\beta \Delta x/2)] \quad (3.32)$$

We verify that $|G| \leq 1$ for all $\backslash mathrmFo > 0$.

Order-2 precision:

The scheme is of order 2 both in time and space: $O((\Delta t)^2) + O((\Delta x)^2)$. This is a major advantage over previous schemes.

Energy conservation:

For the homogeneous problem without sources, Crank-Nicolson preserves exactly the integral $\int T\, dx$, reflecting the physical conservation of energy.

No artificial dissipation:

Unlike the pure implicit scheme, Crank-Nicolson does not introduce excessive numerical dissipation.

3.5.6. Resolution algorithm

At each time step:

15. Calculate the second member: $r^n = B \cdot T^n$.
16. Apply boundary conditions.
17. Solve $A \cdot T^{n+1} = r^n$ by TDMA.
18. Check convergence or stopping criteria.

3.5.7. Practical Considerations

Choice of time step:

Although stable for any Δt, a wise choice is $\backslash mathrmFo \approx 1 - 5$ for good efficiency.

Oscillations:

For very discontinuous initial conditions, the scheme can generate slight temporary oscillations that then dampen.

Computational cost:

Slightly higher than the simple implicit scheme (calculation of $B \cdot T^n$ extra), but compensated by the increased accuracy allowing for longer time steps.

3.6. Comparative Analysis of 1D Schemes

3.6.1. Summary Table

Table 3.2. ***Summary comparison of the three 1D schemes***

Property	Explicit	Implicit	Crank-Nicolson
Stability	$Fo \leq 1/2$	Uncond.	Uncond.
Time order	$O(\Delta t)$	$O(\Delta t)$	$O((\Delta t)^2)$
Space order	$O((\Delta x)^2)$	$O((\Delta x)^2)$	$O((\Delta x)^2)$
System	No	Yes	Yes
Dissipation	Low	Strong	Minimal
Oscillations	No	No	Possible
Usage	Δt *small*	*All* Δt	Optimal

3.6.2. Selection Criteria

Explicit Scheme — Recommended if:

- Short-term simulation.
- Very fine mesh required (Δx small).
- Limited computational resources.
- No access to linear system solver.
- Easily parallelizable problem.

Implicit Scheme — Recommended if:

- Long-term simulation.
- Acceptable coarse mesh size.
- Absolute stability required.
- No variable times.
- Robustness as a priority.

Crank-Nicolson Scheme — Recommended if:

- High precision required.
- Medium-to-long-term simulation.
- Important energy conservation.
- Regular initial conditions.
- Better accuracy/overall cost trade-off.

3.6.3. Comparative numerical example

Test problem: bar of length $L = 1$ m, $\alpha = 10^{-4}$ m^2/s.

Initial condition: $T(x,0) = sin(\backslash pi\ x)$.

Boundary conditions: $T(0,t) = T(1,t) = 0$.

Configuration:

- $\Delta x = 0.01$ m (100 points).
- Final time: $t = 10$ s.

Results:

Table 3.3. ***Numerical results for the three 1D schemes***

Scheme	Δt (s)	Total steps	L^2 error
Explicit	0.0005	20000	1.2×10^{-3}
Implicit	0.01	1000	2.5×10^{-3}
Crank-Nicolson	0.01	1000	4.8×10^{-5}

Crank-Nicolson offers the best accuracy with a reduced step count, demonstrating its overall effectiveness.

Part II — Two-dimensional problems

3.7. Extension to 2D Problems

3.7.1. Two-dimensional diffusion equation

For a rectangular domain $\Omega = [0, L_x] \times [0, L_y]$:

$$\partial T/\partial t = \alpha(\partial^2 T/\partial x^2 + \partial^2 T/\partial y^2) + q(x, y, t) \quad (3.33)$$

with initial conditions and at the appropriate limits on $\partial\Omega$.

3.7.2. 2D Spatial Discretization

Cartesian mesh:

$$x_i = i \cdot \Delta x, \quad i = 0, 1, \ldots, N_x \quad (3.34)$$

$$y_j = j \cdot \Delta y, \quad j = 0, 1, \ldots, N_y \quad (3.35)$$

Notation: $T_{i,j}^{\,n} \approx T(x_i, y_j, t_n)$.

3.7.3. Discrete Laplacian operator

Classic 5-point stencil. For a uniform mesh ($\Delta x = \Delta y = h$):

$$\nabla^2 T|i,j \approx (T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j})/h^2 \quad (3.36)$$

3.7.4. 2D-Specific Challenges

Increase in size:

The number of unknowns goes from N in 1D to $N_x \times N_y$ in 2D, drastically increasing memory and computational requirements.

Stricter stability condition:

For the 2D explicit scheme: $Fo \leq 1/4$ (instead of $1/2$ in 1D), i.e. $\Delta t \leq h^2/(4\alpha)$.

Complexity of implicit systems:

The matrices are no longer tridiagonal but block pentadiagonal, making the solution more expensive.

3.7.5. Resolution Strategies

Three main approaches emerge:

19. Two-level time methods (fully implicit).
20. ADE methods (Alternating Direction Explicit).
21. ADI methods (Alternating Direction Implicit).

Each has different trade-offs between stability, precision and efficiency.

3.8. Two-Level Time Methods

3.8.1. 2D Explicit Scheme

Direct extension of FTCS 1D:

$$T_{i,j}^{n+1} = T_{i,j}^{n} + Fo_x(T_{i}^{+1},_{j}^{n} - 2T_{i,j}^{n} + T_{i}^{-1},_{j}^{n}) + Fo_y(T_{i,j}^{+1n} - 2T_{i,j}^{n} + T_{i,j}^{-1n}) \quad (3.37)$$

where $Fo_x = \alpha\Delta t/(\Delta x)^2$ and ⅄r$Fo_y = \alpha\Delta t/(\Delta y)^2$.

Stability Condition:

$$Fo_x + Fo_y \leq \frac{1}{2} \quad (3.38)$$

For a square mesh: $Fo \leq 1/4$.

3.8.2. Fully 2D Implicit Scheme

All spatial derivatives evaluated at time $n + 1$:

$$T_{i,j}^{n+1} - Fo_x(T_{i}^{+1},_{j}^{n+1} - 2T_{i,j}^{n+1} + T_{i}^{-1},_{j}^{n+1}) - Fo_y(T_{i,j}^{+1n+1} - 2T_{i,j}^{n+1} + T_{i,j}^{-1n+1}) = T_{i,j}^{n} \quad (3.39)$$

Properties:

- Unconditionally stable.
- Order 1 in time, order 2 in space.
- Requires solving a large linear system.

3.8.3. 2D Matrix Structure

Numbering the points line by line, the matrix A is block pentadiagonal:

$$A = \left(I + Fo_x A_x + Fo_y A_y\right) \quad (3.40)$$

where A_x and A_y are the discrete second-differentiation operators.

Complexity:

- Direct methods: $O(N^3)$ — unacceptable for large N.
- Iterative methods: preferable (Gauss-Seidel, SOR, conjugate gradient).
- Each time step requires several iterations.

3.8.4. 2D Crank-Nicolson Scheme

Extension of the time average:

$$T_{i,j}^{n+1} - T_{i,j}^{n} = Fo_x/2 \cdot [(T_{i}^{+1},_{j} - 2T_{i,j} + T_{i}^{-1},_{j})^{n} + (T_{i}^{+1},_{j} - 2T_{i,j} + T_{i}^{-1},_{j})^{n+1}] + Fo_y/2 \cdot [(T_{i,j}^{+1} - 2T_{i,j} + T_{i,j}^{-1})^{n} + (T_{i,j}^{+1} - 2T_{i,j} + T_{i,j}^{-1})^{n+1}] \quad (3.41)$$

Advantages:

- Order 2 in time and space.
- Unconditionally stable.
- Optimal accuracy.

Drawbacks:

Very expensive resolution of the large implicit system.

3.8.5. Practical limitations

Implicit two-level methods, while stable and accurate, become prohibitive for large 2D domains due to:

- Cost of solving the entire system at every step.
- Memory for storing sparse matrices.
- Parallelization difficulty.

These limitations motivate the development of alternating-direction methods.

3.9. Explicit Alternating Direction (ADE) Methods

3.9.1. Directional Separation Concept

The idea of the ADE methods is to break down the 2D problem into two successive 1D subproblems, each dealt with explicitly in a direction.

3.9.2. Basic ADE Scheme

The transition from time n to time $n+1$ is done in two half-steps.

First half step (direction x):

$$T_{i,j}^{n+½} = T_{i,j}^{n} + Fo_x(T_{i}^{+1},_{j}^{n} - 2T_{i,j}^{n} + T_{i}^{-1},_{j}^{n}) \quad (3.42)$$

Second half-step (direction y):

$$T_{i,j}^{n+1} = T_{i,j}^{n+½} + Fo_y(T_{i,j}^{+1n+½} - 2T_{i,j}^{n+½} + T_{i,j}^{-1n+½}) \quad (3.43)$$

3.9.3. Stability Analysis

Von Neumann's analysis shows that the amplification factor is:

$$G = [1 - 4Fo_x \cdot sin^2(\beta_x \Delta x/2)] \cdot [1 - 4Fo_y \cdot sin^2(\beta_y \Delta y/2)] \quad (3.44)$$

Stability Condition:

The ADE method does not improve stability: we find the same condition as the standard 2D explicit scheme. For a square mesh: $Fo \leq 1/4$.

3.9.4. Benefits and limitations

Advantages:

- Explicit calculations, no system to solve.
- Simplicity of implementation.
- Processing line by line and then column by column.
- Caching-friendly locality of operations.

Limitations:

- No stability improvement over the standard explicit.
- Always subject to $Fo \leq 1/4$.
- Splitting error introduced.

3.9.5. ADE Variants

Barakat-Clark ADE:

Improvement by extrapolation to reach $Fo \leq 1/2$ in some cases.

ADE with correction:

Added a patch term to reduce splitting error and improve accuracy.

3.9.6. Conclusion on ADE

Although conceptually interesting, the ADE method does not provide a decisive advantage in practice. The implicit ADI methods, developed in the next section, provide much better performance by removing the stability restriction.

3.10. Peaceman-Rachford ADI Method

3.10.1. Fundamental Principle

The Peaceman-Rachford ADI (Alternating Direction Implicit) method, introduced in 1955, represents a major advance for 2D parabolic problems. It combines:

- Unconditional stability of implicit methods.
- Efficiency of 1D resolutions (tridiagonal systems).
- Order-2 accuracy in time.

3.10.2. Scheme Formulation

Two alternating implicit half-steps.

First half-step (implicit in x, explicit in y):

$$T_{i,j}^{n+½} - Fo_x(T_i^{+1},_j^{n+½} - 2T_{i,j}^{n+½} + T_i^{-1},_j^{n+½}) = T_{i,j}^{n} + Fo_y(T_{i,j}^{+1n} - 2T_{i,j}^{n} + T_{i,j}^{-1n}) \quad (3.45)$$

Second half-step (explicit in x, implicit in y):

$$T_{i,j}^{n+1} - Fo_y(T_{i,j}^{+1n+1} - 2T_{i,j}^{n+1} + T_{i,j}^{-1n+1}) = T_{i,j}^{n+½} + Fo_x(T_i^{+1},_j^{n+½} - 2T_{i,j}^{n+½} + T_i^{-1},_j^{n+½}) \quad (3.46)$$

3.10.3. Algorithmic Interpretation

At each full time step:

First half-step:

For each row j:

- Calculate the explicit second member in y.
- Solving the tridiagonal system in x (TDMA).
- Get $T^{n+1/2}$.

Second half-step:

For each column i:

- Calculate the explicit second member in x.
- Solving the tridiagonal system in y (TDMA).
- Get T^{n+1}.

3.10.4. Stability Analysis

By von Neumann analysis, the amplification factor is:

$$G = \left[\left(1 - 2Fo_y \cdot sin^2\left(\beta_y \Delta y/2\right)\right)/\left(1 + 2Fo_x \cdot sin^2(\beta_x \Delta x/2)\right)\right] \cdot \left[\left(1 - 2Fo_x sin^2(\beta_x \Delta x/2)\right)/\left(1 + 2Fo_y \cdot sin^2\left(\beta_y \Delta y/2\right)\right)\right] \quad (3.47)$$

It is shown that $|G| \leq 1$ for all $Fo_x, Fo_y > 0$. The method is therefore unconditionally stable.

3.10.5. Remarkable properties

Unconditional stability:

No restrictions on Δt for stability.

Order-2 precision:

$$O((\Delta t)^2) + O((\Delta x)^2) + O((\Delta y)^2) \quad (3.48)$$

Computational efficiency:

Complexity $O(N)$ per time step, where $N = N_x \times N_y$.

Symmetry:

The scheme treats x and y symmetrically, ensuring numerical isotropy.

Controlled dissipation:

Proper damping of high frequencies without excess.

3.10.6. Detailed Implementation

Step 1: Line-by-Line Resolution.

For each fixed j, tridiagonal system in i:

$$Fo_x \cdot T_{i}^{-1}{}_{,j}^{n+1/2} + (1 + 2Fo_x) \cdot T_{i,j}^{n+1/2} - Fo_x \cdot T_{i}^{+1}{}_{,j}^{n+1/2} = b_{i,j} \quad (3.49)$$

where $b_{i,j} = T_{i,j}^{n} + Fo_y(T_{i,j}^{+1n} - 2T_{i,j}^{n} + T_{i,j}^{-1n})$.

Step 2: Column-by-Column Resolution.

For each fixed i, tridiagonal system in j:

$$Fo_y \cdot T_{i,j}^{-1n+1} + \left(1 + 2Fo_y\right) \cdot T_{i,j}^{n+1} - Fo_y \cdot T_{i,j}^{+1n+1} = c_{i,j} \quad (3.50)$$

where $c_{i,j} = T_{i,j}^{n+1/2} + Fo_x(T_{i}^{+1}{}_{,j}^{n+1/2} - 2T_{i,j}^{n+1/2} + T_{i}^{-1}{}_{,j}^{n+1/2})$

3.10.7. Treatment of boundary conditions

Dirichlet:

Boundary values are set directly, reducing the size of the systems.

Neumann:

Approximation by one-sided or centered finite differences according to direction.

Example for $\partial T/\partial x|_{x=0} = 0$: $(T_{1,j} - T_{0,j})/\Delta x = 0 \Rightarrow T_{0,j} = T_{1,j}$.

3.10.8. Comparison with Other Methods

Table 3.4. ***Comparison of 2D parabolic schemes***

Method	Stability	Accuracy	Cost/step	Complexity
Explicit 2D	$Fo \leq 1/4$	$O(\Delta t)$	$O(N)$	Very Low
Implicit 2D	Uncond.	$O(\Delta t)$	$O(N^2 - N^3)$	High
2D C-N	Uncond.	$(O(\Delta t)^2)$	$O(N^2 - N^3)$	Very high
ADI P-R	Uncond.	$(O(\Delta t)^2)$	$O(N)$	Moderate

Peaceman-Rachford offers the best compromise: unconditional stability, order 2, and linear complexity.

3.11. Optimization and Practical Applications

3.11.1. Optimal Time-Step Selection

Although Peaceman-Rachford is unconditionally stable, the choice of Δt affects the accuracy and efficiency.

Accuracy Criteria:

To maintain the balance of spatial and temporal errors with an $O(\Delta t^2), O(\Delta x^2)$ scheme:

$$\Delta t \approx C \cdot \Delta x^2 \quad with \ C \approx 0.25 - 1 \tag{3.51}$$

Efficacy Criteria:

Time steps that are too large require more time iterations to reach t_{final}, reducing the stability advantage.

Practical recommendation:

$Fo \approx 0.5 - 2$ for a good compromise.

3.11.2. Spatio-temporal adaptation

Mesh Refinement:

Use a non-uniform, finer mesh in areas with strong gradients (boundary layers, corners, sources).

Adaptive time step:

Adjust Δt according to the evolution of the solution: small initially, then increasing as the solution stabilizes.

Adaptation criterion:

$$\Delta t^{n+1} = \Delta t^{n} \cdot (e_t ol / \|T^{n+1} - T^{n}\|)^{(1/3)} \quad (3.52)$$

3.11.3. Parallelization

First half-step:

The rows are independent → parallelization on the rows (N_y simultaneous systems).

Second half-step:

The columns are independent → parallelization on the columns (N_x concurrent systems).

Scalability:

Excellent up to $min(N_x, N_y)$ processors, then limited by time dependence.

3.11.4. Extensions and Generalizations

Variable coefficients:

For $\alpha = \alpha(x, y)$, adapt the discrete operators taking into account spatial variation.

Non-linear terms:

For equations like $\partial T / \partial t = \nabla \cdot (\alpha(T) \nabla T)$, linearize around T^n (semi-implicit method).

3D problems:

Generalization to three half-steps (x, y, z) or use of LOD (Locally One-Dimensional) methods.

3.11.5. Application: Cooling a Plate

Typical problem: square plate $[0, 1] \times [0, 1]$, initially at $T_0 = 100\,°C$. Edges maintained at $0\,°C$, $\alpha = 10^{-4}\ m^2/s$.

Configuration:

- Mesh: 50×50 stitches.
- $\Delta t = 0.1$ s ($Fo \approx 1$).
- Simulation time: 100 s.

Results:

- Number of steps: 1000.
- Computation time: ~2 s (ADI) vs ~45 s (fully implicit).
- Acceleration: factor 22.
- Accuracy: L^2 error $< 10^{-4}$.

3.11.6. Validation and verification

Analytical Solutions:

Compare with exact solutions when available (variable separation, Fourier transform).

Mesh convergence:

Check that the error decreases as $O(h^2)$ by refining the mesh.

Energy conservation:

For an isolated system, check $\iint T\ dx\,dy = constant$ (to the numerical precision).

General Conclusion

The study of numerical methods for parabolic equations reveals a logical progression of techniques, from elementary schemes to sophisticated approaches optimizing efficiency/accuracy.

One-dimensional problems:

The Crank-Nicolson scheme is the optimal choice in the majority of cases, combining unconditional stability and order 2. Explicit schemes remain relevant for short simulations or simple implementations, while pure implicit schemes offer maximum robustness at the cost of lower temporal accuracy.

Two-dimensional problems:

The Peaceman-Rachford ADI method represents a fundamental advance, eliminating the dilemma between stability and efficiency. By decomposing the 2D problem into 1D sequences, it reduces the complexity from $O(N^2)$ or $O(N^3)$ to $O(N)$ while maintaining order 2 and unconditional stability. This approach naturally generalizes to 3D problems and forms the basis of many modern solvers.

Perspectives:

Recent developments include higher-order schemes (implicit-explicit Runge-Kutta), adaptive methods in space-time, and integration with multigrid techniques. The exploitation of modern parallel architectures (GPUs, clusters) opens up new possibilities for the simulation of complex phenomena on a large scale.

Final recommendations:

- 1D: Crank-Nicolson for general purpose.
- 2D/3D: Peaceman-Rachford as a reference.
- Stiff Problems: Implicit Methods with Preconditioning.
- Large scales: ADI + multigrid + parallelization.

Bibliographical references

J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type", Proceedings of the Cambridge Philosophical Society, vol. 43, pp. 50-67, 1947.

D.W. Peaceman and H.H. Rachford, "The numerical solution of parabolic and elliptic differential equations", Journal of the Society for Industrial and Applied Mathematics, vol. 3, pp. 28-41, 1955.

G.D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, 3rd edition, Oxford University Press, 1985.

J.C. Strikwerda, Finite Difference Schemes and Partial Differential Equations, 2nd edition, SIAM, 2004.

R.J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems, SIAM, 2007.

J.W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods, Springer, 1995.

K.W. Morton and D.F. Mayers, Numerical Solution of Partial Differential Equations, 2nd edition, Cambridge University Press, 2005.

W.F. Ames, Numerical Methods for Partial Differential Equations, 3rd edition, Academic Press, 1992.

Corrected Examples (results + method)

Example 3.1 (Composite Trapezoidal rule).

Statement. Approximate $\int_0^1 (x^2 + 0{,}2)dx$ with $n = 4$.

Solution (summary). $h = 0.25$. $T = 0.25[(f0 + f4)/2 + f1 + f2 + f3] = 0.54375$.

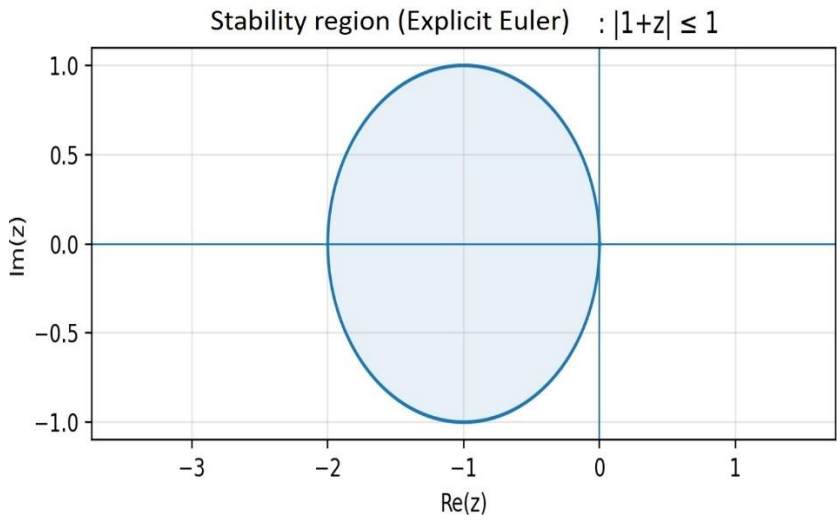
Exact $= 0.53333$. Error ≈ 0.01042.

Example 3.2 (Simpson's rule).

Statement. Same integral with Simpson's rule ($n = 4$).

Solution (summary). $S = h/3[f0 + f4 + 4(f1 + f3) + 2f2] = 0.53333$ (correct here).

Chapter 4 — Hyperbolic Equations



Chapter 4. Hyperbolic Equations

Method of Characteristics, Burgers Equation, and Sound Waves in Fluids

Chapter Objectives

- Understand the key concepts and the digital issue.
- Build an algorithm (steps + pseudo-code) and apply it.
- Interpret the error and check the stability/convergence on a simple case.

Table of Contents

4.1. Introduction to Hyperbolic Equations

4.1.1. Nature of hyperbolic equations

Hyperbolic partial differential equations form a fundamental class of PDEs that model propagation and transport phenomena in nature. They are distinguished by their ability to describe the propagation of information at finite speed, unlike parabolic (instantaneous scattering) or elliptic (spatial equilibrium) equations.

4.1.2. Canonical form and classification

The simplest first-order hyperbolic equation is written:

$$\partial u/\partial t \; + \; c \cdot \partial u/\partial x \; = \; 0 \tag{4.1}$$

where $u(x,t)$ is the transported variable and c is the propagation velocity.

For the second-order equations, the canonical form is the wave equation:

$$\partial^2 u/\partial t^2 - c^2 \cdot \partial^2 u/\partial x^2 = 0 \tag{4.2}$$

4.1.3. Characteristic properties

Directional propagation:

The information propagates along well-defined curves called characteristics, at a speed determined by the coefficients of the equation.

Conservation without dissipation:

In their pure form, hyperbolic equations retain the magnitude of perturbations as they propagate, with no intrinsic dissipation.

Formation of discontinuities:

A remarkable phenomenon: even from regular initial conditions, discontinuities (shocks) can appear in finite time.

4.1.4. Modelled physical phenomena

Hyperbolic equations appear in many fields:

- Acoustics: sound propagation in fluids and solids.
- Gas dynamics: compressible flows, shock waves.
- Elastodynamics: seismic waves, vibrations.
- Electromagnetism: propagation of EM waves.
- Road traffic: vehicle flow modelling.
- Hydraulics: surface waves, free surface flows.

4.2. Mathematical Foundations and Characteristic Curves

4.2.1. Characteristic curve concept

The central concept in the study of hyperbolic equations is that of the characteristic curve. A characteristic is a curve in spacetime (x, t) along which the PDE reduces to an ordinary differential equation.

4.2.2. Linear Advection Equation

Consider the simplest equation:

$$\partial u/\partial t + c \cdot \partial u/\partial x = 0 \quad where\ constant\ c > 0 \tag{4.3}$$

Along a curve $x = x(t)$, the total derivative of u is:

$$du/dt = \partial u/\partial t + (dx/dt) \cdot \partial u/\partial x \tag{4.4}$$

If we choose $dx/dt = c$, then the equation becomes:

$$du/dt = \partial u/\partial t + c \cdot \partial u/\partial x = 0 \tag{4.5}$$

Fundamental conclusion:

Along the lines $x - c\,t = constant$ (characteristics), the solution u remains constant. These lines have a slope $1/c$ in the plane (x, t).

4.2.3. General Solution

Let $u(x, 0) = u_0(x)$ be the initial condition.

The characteristic passing through $(x_0, 0)$ has the equation:

$$x = x_0 + c\,t \tag{4.6}$$

Along this characteristic, u retains its initial value:

$$u(x, t) = u_0(x_0) = u_0(x - c\,t) \tag{4.7}$$

Physical interpretation:

The solution is simply the initial profile that moves rigidly to the right at speed c, without deformation or attenuation.

4.2.4. Areas of Dependence and Influence

Dependency area:

To determine $u(x,t)$, we plot the retrograde characteristic from (x,t) to the $t = 0$ axis. The intersection point $x_0 = x - c\,t$ defines the dependency domain.

Area of influence:

The influence of a point x_0 at $t = 0$ propagates along the characteristic $x = x_0 + c\,t$. At time t, it reaches the point $x = x_0 + c\,t$.

4.2.5. CFL Condition (Courant-Friedrichs-Lewy)

For a numerical scheme to be stable, its numerical dependency domain must contain the analytical dependency domain. This leads to the fundamental condition:

$$CFL = c \cdot \Delta t/\Delta x \leq Cm_{\text{ax}} \tag{4.8}$$

where $C_{\backslash max}$ depends on the scheme (typically $C_{\backslash max} = 1$ for explicit schemes).

4.3. Method of Characteristics: Theory and Applications

4.3.1. Principle of the Method

The method of characteristics transforms a hyperbolic PDE into a system of ordinary differential equations along privileged curves. This approach is particularly powerful for first-order quasilinear equations.

4.3.2. General Wording

For the general quasilinear equation:

$$\partial u/\partial t + a(x,t,u) \cdot \partial u/\partial x = b(x,t,u) \tag{4.9}$$

The system of characteristic equations is:

$$\frac{d\,x}{d\,t} = a(x,t,u)$$

$$\frac{d\,u}{d\,t} = b(x,t,u)$$

with the initial conditions $x(0) = x_0$ and $u(x_0,0) = u_0(x_0)$.

4.3.3. Resolution algorithm

Step 1:

Set initial conditions by x_0.

Step 2:

For each x_0, solve the coupled ODE system:

- Get $x(t;\ x_0)$ and $u(t;\ x_0)$.

Step 3:

The solution $u(x,t)$ is obtained implicitly: find x_0 such that $x = x(t;\ x_0)$, then $u(x,t) = u(t;\ x_0)$.

4.3.4. Detailed example: transport with decay

Problem:

$$\partial C/\partial t + v \cdot \partial C/\partial x = -\lambda C \tag{4.10}$$

where C is the concentration, v is the velocity, λ is the rate of decay.

Solution:

Characteristics: $dx/dt = v \Rightarrow x = x_0 + v\,t$.

Equation along the characteristics: $dC/dt = -\lambda C$.

Solution: $C(t) = C^0 \cdot exp(-\lambda t)$.

So:

$$C(x,t) = C^0(x - vt) \cdot exp(-\lambda t) \tag{4.11}$$

4.3.5. Intersection of Characteristics

A crucial phenomenon occurs for nonlinear equations: characteristics from different points can intersect. When this happens:

- The solution u would take several values at the same point (x,t).
- Violation of the uniqueness of the classic solution.
- Formation of a discontinuity (shock).
- Need to move to weak solutions.

4.3.6. Breaking Time

For some initial conditions, we can calculate the time t_b at which the characteristics meet for the first time:

$$t_b = -1/min_x(du^0/dx) \tag{4.12}$$

Beyond this time, the classical solution ceases to exist and a shock appears.

4.4. Inviscid Burgers Equation

4.4.1. Introduction

The Burgers equation, proposed by J.M. Burgers in 1948, is a prototype model in nonlinear dynamics. Its inviscid form:

$$\partial u/\partial t + u \cdot \partial u/\partial x = 0 \quad (4.13)$$

captures the essence of many non-linear physical phenomena while remaining simple enough to be analyzed in detail.

4.4.2. Conservative form

The equation can be written in conservative form:

$$\partial u/\partial t + \partial(u^2/2)/\partial x = 0 \quad (4.14)$$

This formulation is crucial for the analysis of weak solutions and the construction of conservative numerical schemes.

4.4.3. Characteristic equations

For the Burgers equation, the characteristic system is:

$$\frac{d\,x}{d\,t} = u$$

$$\frac{d\,u}{d\,t} = 0$$

Important consequences:

- u remains constant along each characteristic.
- Each characteristic is a straight line: $x = x_0 + u_0(x_0) \cdot t$.
- The slope $dx/dt = u$ depends on the value of u at the start.
- The characteristics are lines of different slopes.

4.4.4. Implicit Solution

The general solution is implicitly written:

$$u(x,t) = u_0(x_0) \quad where \quad x = x_0 + u_0(x_0) \cdot t \quad (4.15)$$

This relation defines x_0 in terms of x and t. As long as this inversion is unique, the solution is well defined.

4.4.5. Example: Sinusoidal Initial Condition

Let $u_0(x) = sin(x)$.

Analysis:

- Where $u_0'^{(x)} = cos(x) < 0$escent), the characteristics converge.
- Breaking time: $t_b = 1/\max\left|u_0'^{(x)}\right| = 1$.
- For $t < 1$, classical solution exists.
- For $t \geq 1$, formation of a shock.

4.4.6. Rarefaction wave vs. compression wave

Rarefaction wave(u_0 increasing):

- Divergent characteristics.
- Solution remains continuous for all times.
- Profile stretch.

Compression wave (u_0 decreasing):

- Converging characteristics.
- Finite-time shock formation.
- Profile compression.

4.5. Viscous Burgers Equation

4.5.1. Complete Equation

The addition of a viscosity term gives:

$$\partial u/\partial t + u \cdot \partial u/\partial x = \nu \cdot \partial^2 u/\partial x^2 \qquad (4.16)$$

where $\nu > 0$ is the viscosity coefficient.

4.5.2. Convection-diffusion competition

Nonlinear term:

$u \cdot \partial u/\partial x$ tends to create shocks.

Viscous term:

$\nu \cdot \backslash partial^2\ u/\partial x^2$ regulates and smooths discontinuities.

Result:

For ν small, the shocks are replaced by steep but continuous transition layers (shock layers).

4.5.3. Reynolds number

The key dimensionless parameter is:

$$Re = UL/\nu \tag{4.17}$$

- $Re \gg 1$: convection-dominated regime (quasi-hyperbolic).
- $Re \ll 1$: regime dominated by diffusion (quasi-parabolic).
- $Re \sim O(1)$: balanced competition.

4.5.4. Cole-Hopf Transformation

Remarkable result: the Burgers equation can be linearized by the transformation:

$$u = -2\,\nu\frac{\partial\varphi/\partial x}{\varphi} \tag{4.18}$$

where φ satisfies the linear heat equation:

$$\partial\varphi/\partial t = \nu \cdot \partial^2\varphi/\partial x^2 \tag{4.19}$$

Procedure:

11. Solve the linear equation for φ.
12. Calculate u by the transformation.
13. Obtain the exact viscous Burgers solution.

4.5.5. Structure of a shock layer

For $\nu \to 0$, the thickness of the transition layer is:

$$\delta \sim \frac{\nu}{|u_L - u_R|} \tag{4.20}$$

In the limit $\nu \to 0$, the layer becomes infinitely thin and the solution converges towards the inviscid Burgers shock.

4.6. Shock Formation and Weak Solutions

4.6.1. Shocking phenomenon

In nonlinear hyperbolic equations, even smooth initial conditions can lead to finite-time discontinuous solutions.

4.6.2. Rankine-Hugoniot Conditions

Through a discontinuity, the conservation laws must be satisfied in the integral sense. For $\partial u/\partial t + \partial f(u)/\partial x = 0$:

Shock velocity:

$$s = \frac{f(u_R) - f(u_L)}{u_R - u_L} \tag{4.21}$$

where u_L and u_R are the left and right states of the shock.

4.6.3. Entropy condition

The Rankine-Hugoniot conditions are not sufficient to guarantee uniqueness. We impose an entropy condition to select the physical solution:

Lax Condition:

$$f'(u_L) > s > f'(u_R) \tag{4.22}$$

The characteristics must enter into the shock on both sides (principle of irreversibility).

4.6.4. Riemann Problem

Initial condition in step form:

$$u(x,0) = u_L for x < 0 \qquad u(x,0) = u_R for x > 0 \tag{4.23}$$

Solution for Burgers:

- If $u_L > u_R$: shock propagating at $s = (u_L + u_R)/2$.
- If $u_L < u_R$: self-similar rarefaction wave.

4.7. Sound Waves in Fluids

4.7.1. Euler's equations

The equations of 1D compressible fluid dynamics are:

Mass:

$$\partial\rho/\partial t + \partial(\rho u)/\partial x = 0 \tag{4.24}$$

Momentum:

$$\partial(\rho u)/\partial t + \partial(\rho u^2 + p)/\partial x = 0 \tag{4.25}$$

Energy:

$$\partial E/\partial t + \partial[(E + p)u]/\partial x = 0 \tag{4.26}$$

where ρ: density, u: velocity, p: pressure, E: total energy.

4.7.2. Linearization

Around a state at rest $(\rho^0, u^0 = 0, p^0)$, let us assume:

$$\rho = \rho^0 + \rho' \quad with\ |\rho'| \ll r^0$$
$$u = u' \quad with \quad |u'| \ll c$$
$$p = p^0 + p' \quad with \quad |p'| \ll p^0$$

where $c = \sqrt{\partial p / \partial \rho}$ is the speed of sound.

4.7.3. Linear Acoustic Equations

At the first order, we obtain:

$$\partial \rho' / \partial t + \rho^0 \cdot \partial u' / \partial x = 0 \tag{4.27}$$

$$\partial u' / \partial t + (1/\rho^0) \cdot \partial p' / \partial x = 0 \tag{4.28}$$

With $p' = c^2 \rho'$, we eliminate a variable.

4.7.4. Acoustic Wave Equation

Combining the equations, we obtain:

$$\partial^2 p' / \partial t^2 - c^2 \cdot \partial^2 p' / \partial x^2 = 0 \tag{4.29}$$

This is the classical wave equation.

4.7.5. d'Alembert Solution

The general solution is written:

$$p'^{(x,t)} = F(x - c\,t) + G(x + c\,t) \tag{4.30}$$

- $F(x - c\,t)$: wave propagating to the right.
- $G(x + c\,t)$: wave propagating to the left.

4.7.6. Acoustic Characteristics

Two families of characteristics:

$$C^+: \frac{d\,x}{d\,t} = +c; \quad C^-: \frac{d\,x}{d\,t} = -c \tag{4.31}$$

Riemann invariants:

Along C^+:

$$\frac{u'}{c} + \frac{p'}{rho_0\, c^2} = const. \tag{4.32}$$

Along C^-:

$$\frac{u'}{c} - \frac{p'}{\rho^0\, c^2} = const.$$

(4.33)

4.8. Acoustic Propagation and Linearized Euler Equations

4.8.1. Reflection phenomena

Reflection on rigid wall:

Condition $u = 0$ at the wall. Reflected wave with the same amplitude, double the pressure.

Interface between two media:

Reflection and transmission coefficients:

$$R = \frac{Z_2 - Z_1}{Z_2 + Z_1} \tag{4.34}$$

$$T = \frac{2\,Z_2}{Z_2 + Z_1} \tag{4.35}$$

where $Z = \rho c$ is the acoustic impedance.

4.8.2. Nonlinear Waves

For large amplitudes:

- Compressed parts spread faster.
- Progressive profile deformation.
- Formation of an acoustic shock (supersonic boom).

4.9. Numerical Methods for Hyperbolic Equations

4.9.1. Digital Challenges

- Parasitic oscillations near discontinuities.
- Excessive numerical dissipation.
- Numerical dispersion.
- Mandatory CFL condition.
- Need for conservative schemes.

4.9.2. Upwind Scheme

For $\partial u/\partial t + c \cdot \partial u/\partial x = 0$ with $c > 0$:

$$u_i^{n+1} = u_i^n - \frac{c\,\Delta t}{\Delta x}\left(u_i^n - u_{i-1}^n\right) \tag{4.36}$$

Stable if $CFL = c\,\Delta t/\Delta x \leq 1$.

4.9.3. Lax-Friedrichs Scheme

$$u_i^{n+1} = (u_{i+1}^n + u_{i-1}^n)/2 - (c \cdot \Delta t)/(2\Delta x) \cdot (u_{i+1}^n - u_{i-1}^n) \quad (4.37)$$

Very dissipative but stable and simple.

4.9.4. Lax-Wendroff Scheme

Order-2 scheme:

$$u_i^{n+1} = u_i^n - (c \cdot \Delta t)/(2\Delta x) \cdot (u_{i+1}^n - u_{i-1}^n) + (c^2 \cdot \Delta t^2)/(2\Delta x^2) \cdot (u_{i+1}^n - 2u_i^n + u_{i-1}^n) \quad (4.38)$$

Precise but oscillates close to shocks.

4.10. Advanced Numerical Schemes and Applications

4.10.1. TVD Schemes

TVD (Total Variation Diminishing) schemes ensure:

$$TV(u^{n+1}) \leq TV(u^n) \quad (4.39)$$

No creation of new oscillations.

4.10.2. WENO Schemes

Weighted Essentially Non-Oscillatory:

- High order in regular areas.
- Robust shock capture.
- Weighted combination of stencils.

4.10.3. Practical Applications

Sod Shock Tube:

Standard Test for Euler Codes.

Road traffic:

LWR model with plugging.

Aerodynamics:

Supersonic flows, shock waves.

4.10.4. Comparison of Schemes

Table 4.1. ***Comparison of numerical schemes for hyperbolic equations***

Scheme	Order	Dissipation	Oscillations
Upwind	1	Strong	No
Lax-Wendroff	2	Low	Yes
TVD	2	Moderate	No
WENO	3-9	Very low	No

Table 4.2. ***Behaviour of selected schemes by application***

Scheme	Dissipation	Application
Upwind	Strong	Shocks, stability
Centered	None	Smooth waves
Lax-Friedrichs	Very strong	Robustness
Lax-Wendroff	Low	Accuracy

General Conclusion

Hyperbolic equations are a fundamental class of PDEs for the modeling of propagation phenomena. The method of characteristics provides an elegant framework for their analysis, revealing the underlying geometric structure.

The Burgers equation, despite its simplicity, captures the essence of hyperbolic nonlinearity and shock formation. Sound waves illustrate the practical importance of these equations in acoustics and fluid dynamics.

Modern numerical methods (TVD, WENO) make it possible to efficiently simulate these complex phenomena, finding applications in many fields of engineering and physics.

Bibliographical references

R. Courant and D. Hilbert, Methods of Mathematical Physics, Vol. 2, Interscience, 1962.

P.D. Lax, Hyperbolic Systems of Conservation Laws, SIAM, 1973.

R.J. LeVeque, Numerical Methods for Conservation Laws, Birkhäuser, 1992.

E.F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer, 2009.

J.M. Burgers, Advances in Applied Mechanics, vol. 1, pp. 171-199, 1948.

G.B. Whitham, Linear and Nonlinear Waves, Wiley, 1974.

Corrected Examples (results + method)

Example 4.1 (Newton's method).

Statement. Calculate $\sqrt{2}$ by Newton's method, $x_0 = 1$.

Solution (summary). $x_{k+1} = (x_k + 2/x_k)/2$. $x_1 = 1.5$; $x_2 = 1.4166667$; $x_3 = 1.4142157$; $x_4 = 1.4142136$.

Example 4.2 (Fixed-point iteration).

Statement. Solve $x = cos(x)$, $x_0 = 0.5$ (3 iterations).

Solution (summary). $x_1 = 0.87758$; $x_2 = 0.63901$; $x_3 = 0.80269$ (slow convergence).

Chapter 5 — Error Analysis

Chapter 5. Error Analysis

Consistency, Stability, Convergence, Dissipation and Dispersion

Chapter Objectives

- Understand the key concepts and the digital issue.
- Build an algorithm (steps + pseudo-code) and apply it.
- Interpret the error and check the stability/convergence on a simple case.

Table of Contents

5.1. Introduction to Error Analysis

5.1.1. Importance of Error Analysis

When solving a partial differential equation numerically, the solution obtained inevitably differs from the exact solution. This difference stems from several sources of error that are crucial to understand and master to ensure the reliability of numerical results.

5.1.2. Sources of Error

Modeling error:

Physical simplifications introduced during the formulation of the mathematical problem.

Discretization error:

Approximation of differential operators by difference operators. This is the main purpose of this chapter.

Rounding error:

Limitations of precision in floating-point arithmetic. Generally negligible if the scheme is well conditioned.

Iterative error:

For iterative methods, stop before complete convergence.

5.1.3. Basic properties to be verified

A numerical scheme must satisfy three essential properties:

Consistency:

The numerical scheme must correctly approximate the PDE when the discretization steps tend to zero.

Stability:

Errors (of initial data, rounding) must not grow uncontrollably during the calculation.

Convergence:

The numerical solution must tend towards the exact solution when the discretization steps tend to zero.

5.1.4. Fundamental Lax theorem

The central result of numerical analysis is Lax's theorem:

For a well-posed problem and a consistent scheme, stability is equivalent to convergence. In other words:

$$: Consistency + Stability \Leftrightarrow Convergence \quad (5.1)$$

5.1.5. Parasitic phenomena

Beyond these three properties, two phenomena can degrade quality:

Numerical dissipation:

Artificial damping not present in the original equation. It smooths out solutions but can hide important details.

Numerical Dispersion:

Deformation of the wave profile due to frequency-dependent phase velocity. It creates parasitic oscillations.

5.2. Consistency of Numerical Schemes

5.2.1. Definition of consistency

A numerical scheme is said to be consistent with the PDE if the local truncation error tends to zero when the discretization steps tend to zero.

5.2.2. Local truncation error

The local truncation error τ measures how well the exact solution of the PDE satisfies (or does not satisfy) the numerical scheme.

For a PDE $L(u) = 0$ and a discrete scheme $L_h(u) = 0$:

$$\tau = \frac{L_h(u_{exact})}{h^p} \quad (5.2)$$

where p is the order of the scheme, defined such that $\tau \to 0$ when $h \to 0$.

5.2.3. Order of consistency

A scheme is of order p if:

$$\tau = O(h^p) + (O(\Delta t)^q) \quad (5.3)$$

The overall order is $min(p, q)$. A high order means that the error decreases rapidly with the refinement of the mesh.

5.2.4. Calculating the truncation error: Taylor's method

Each term is developed in a Taylor series around the point considered.

Example: Centered difference for $\partial u/\partial x$

Taylor's developments:

$$u(x + h) = u(x) + hu'^{(x)} + \frac{h^2}{2}u''^{(x)} + \frac{h^3}{6}u'''^{(x)} + O(h^4) \quad (5.4)$$

$$u(x - h) = u(x) - hu'^{(x)} + \frac{h^2}{2}u''^{(x)} - \frac{h^3}{6}u'''^{(x)} + O(h^4) \quad (5.5)$$

Subtraction:

$$u(x + h) - u(x - h) = 2hu'^{(x)} + O(h^3) \quad (5.6)$$

So:

$$\frac{u(x+h) - u(x-h)}{2h} = u'^{(x)} + O(h^2) \quad (5.7)$$

The truncation error is $O(h^2)$, the scheme is of order 2.

5.2.5. Examples for parabolic equations

Explicit scheme (FTCS).

Equation: $\partial u/\partial t = \alpha \cdot \partial^2 u/\partial x^2$.

$$\left(u_i^{(n+1)} - u_i^n\right)/\Delta t = \alpha \cdot \left(u_{(i+1)}^n - 2u_i^n + u_{(i-1)}^n\right)/(\Delta x)^2 \quad (5.8)$$

Order: $O(\Delta t) + O((\Delta x)^2)$

Overall order: 1 in time, 2 in space.

Crank-Nicolson scheme.

Average of spatial derivatives at times n and $n + 1$.

Order: $O((\Delta t)^2) + O((\Delta x)^2)$.

Overall order: 2 in time and space.

5.2.6. Necessary Condition of Consistency

For a consistent scheme:

$$lim_{(h\to 0, \Delta t\to 0)}\tau = 0 \quad (5.9)$$

If this condition is not met, the scheme cannot converge to the PDE solution, regardless of its stability.

5.3. Stability of Numerical Schemes

5.3.1. Stability Concept

A numerical scheme is stable if the errors (of initial data, rounding) remain bounded over time. Mathematically, for an appropriate norm:

$$| u^n | \leq C \cdot | u^0 | \quad for\ any\ n \cdot \Delta t \leq T \tag{5.10}$$

where C is a constant independent of Δt and Δx.

5.3.2. Lax-Richtmyer stability

Formal definition:

A scheme is stable if there are constants K and h_0 such that for any $h \leq h_0$ and any $n \cdot \Delta t \leq T$:

$$| u^n | \leq K \cdot | u^0 | \tag{5.11}$$

5.3.3. Stability analysis by the von Neumann method

The most commonly used method for linear PDEs with constant coefficients. The evolution of Fourier modes is studied.

Principle:

11. Break down the error into Fourier modes: $e_i^n = G^n\ exp(i \backslash beta\ x_i)$.
12. Calculate the amplification factor G.
13. Stability requirement: $|G| \leq 1 + O(\Delta t)$ for any β.

5.3.4. Example: Explicit Scheme for Heat Equation

Equation: $\partial u / \partial t = \alpha \cdot \partial^2 u / \partial x^2$.

Scheme:

$$u_i^{(n+1)} = u_i^n + r \cdot \left(u_{(i+1)}^n - 2u_i^n + u_{(i-1)}^n\right) \tag{5.12}$$

where $r = \alpha \cdot \Delta t/(\Delta x)^2$ (Fourier number).

Analysis:

Fourier mode: $e_i^n = G^n \cdot exp(i\beta x_i)$

Substitution in the scheme:

$$G = 1 + r \cdot \left(exp(i\beta\Delta x) - 2 + exp(-i\beta\Delta x)\right) \tag{5.13}$$

$$G = 1 + r \cdot (2cos(\beta\Delta x) - 2) \tag{5.14}$$

Stability Condition:

$|G| \leq 1$ for all β. The minimum of G is reached for $sin^2(\beta \Delta x/2) = 1$:

$$G_{min} = 1 - 4\,r \tag{5.15}$$

Condition: $G_{\backslash min} \geq -1 \Rightarrow r \leq 1/2.$ That is: $\alpha \cdot \Delta t/(\Delta x)^2 \leq 1/2$ (5.16)

5.3.5. Unconditional vs. Conditional Stability

Conditional stability:

The scheme is stable only if a condition on Δt and Δx is satisfied (e.g. explicit schemes with CFL or Fourier condition).

Unconditional stability:

The scheme is stable for any choice of Δt and Δx (e.g. implicit schemes such as Crank-Nicolson).

5.3.6. CFL Condition for Hyperbolic Equations

For the transport equations, the Courant-Friedrichs-Lewy condition is written:

$$CFL = c \cdot \Delta t/\Delta x \leq C_{max} \tag{5.17}$$

Interpretation:

The numerical dependency domain must contain the analytical dependency domain.

5.4. Convergence and Lax's Theorem

5.4.1. Definition of convergence

A numerical scheme is convergent if:

$$lim_{(\Delta x \to 0, \Delta t \to 0)} |u^n - u(x_i,\, t_n)| = 0 \tag{5.18}$$

for all times $t = n \cdot \Delta t \leq T$, where u^n is the numerical solution and $u(x_i, t_n)$ the exact solution.

5.4.2. Lax's equivalence theorem

Fundamental theorem of numerical analysis:

For a well-posed linear problem and a consistent scheme:

$$Stability \Leftrightarrow Convergence \tag{5.19}$$

Practical consequences:

14. To check convergence, simply check:
- Consistency (usually easy).
- Stability (more delicate).
15. An inconsistent scheme can never converge.
16. An unstable scheme diverges even if it is consistent.

5.4.3. Estimation of the overall error

The overall error $e^n = u^n - {u_{exact}}^n$ satisfies:

$$e^n = (accumulated truncation error) + (propagation of e^0) \quad (5.20)$$

If the scheme is of order p:

$$|e^n| \leq C\, h^p \quad for \cdot \Delta t \leq T \quad (5.21)$$

where C depends on T, the derivatives of the exact solution, and the stability.

5.4.4. Digital convergence testing

To verify the convergence of a scheme in practice.

Manufactured solutions method:

17. Choose an exact solution $u_{ex}(x,t)$.
18. Calculate the corresponding source term $f = L(u_{ex})$.
19. Solve numerically with this source term.
20. Compare u_{num} with u_{ex}.

Mesh refinement:

Calculate with h, $h/2$, $h/4$, ...

Check that $|e_h| / |e_{h/2}| \approx 2^p$ where p is the theoretical order of the scheme.

5.4.5. Observed order vs. theoretical order

The observed convergence rate may differ from the theoretical order:

- Boundary conditions less precise than the interior scheme.
- Insufficient regularity of the exact solution.
- Presence of discontinuities (locally reduced order).
- Dominant rounding errors for very small h.

5.5. Truncation Error Analysis

5.5.1. Local vs Global Truncation Error

Local truncation error (LTE):

Error made in a single time step, assuming the exact solution to the previous steps.

Global Truncation Error (GTE):

Accumulation of local errors over a time interval $[0, T]$.

5.5.2. LTE-GTE Relation

For a stable scheme:

$$LTE = O\left(h^{(p+1)}\right), \qquad then \qquad GTE = O(h^p) \tag{5.22}$$

The loss of an order comes from accumulating over $O(1/h)$ time steps to reach a fixed time T.

5.5.3. Main and dominant errors

Full development of the truncation error:

$$\tau = C_p\, h^p + C_{p+1}\, h^{p+1} + \cdots \tag{5.23}$$

- C_p: coefficient of the main error.
- Determines the order of the scheme.
- Guide to scheme enhancement.

5.5.4. Richardson Amendment

Technique for improving the order of a scheme by combining solutions obtained with different steps.

If $u_h = u_{exact} + C\, h^p + O(h^{p+1})$, then:

$$u_{improved} = \frac{2^p\, u_{h/2} - u_h}{2^p - 1} = u_{exact} + O(h^{p+1}) \tag{5.24}$$

Gain an order without changing the scheme.

5.5.5. Detailed Example: Advection Equation

Upwind scheme.

Equation: $\partial u/\partial t + c\ \partial u/\partial x = 0$.

$$\left(u_i^{(n+1)} - u_i^n\right)/\Delta t + c \cdot \left(u_i^n - u_{(i-1)}^n\right)/\Delta x = 0 \quad (5.25)$$

Taylor Development of the Exact Solution:

$$u(x, t+\Delta t) = u(x,t) + \Delta t \cdot \partial u/\partial t + (\Delta t)^2/2 \cdot \partial^2 u/\partial t^2 + \cdots \quad (5.26)$$

$$u(x-\Delta x, t) = u(x,t) - \Delta x \cdot \partial u/\partial x + (\Delta x)^2/2 \cdot \partial^2 u/\partial x^2 - \cdots \quad (5.27)$$

Substitution and simplification:

$$\tau = -\Delta t/2 \cdot \partial^2 u/\partial t^2 + c \cdot \Delta x/2 \cdot \partial^2 u/\partial x^2 + O(\Delta t^2, \Delta x \Delta t, \Delta x^2) \quad (5.28)$$

Using $\partial u/\partial t = -c\ \partial u/\partial x$:

$$\tau = c \cdot \Delta x/2 \cdot (1 - CFL) \cdot \partial^2 u/\partial x^2 + O(\Delta t^2, \Delta x^2) \quad (5.29)$$

The scheme is of order 1.

5.6. Numerical Dissipation

5.6.1. Concept of Numerical Dissipation

Numerical dissipation is an artificial damping introduced by the scheme, even when the original PDE is conservative (without physical dissipation).

It is manifested by a decrease in the amplitude of the waves over time, which is not present in the exact solution.

5.6.2. Origin of dissipation

Dissipation comes from:

- Asymmetric approximations of spatial derivatives.
- Upwind schemes.
- Even-order terms in the truncation error.
- Artificial numerical broadcasting.

5.6.3. Quantification of dissipation

Modified equation analysis:

We find the PDE actually solved by the scheme by expanding the complete truncation error.

Example: Upwind scheme for advection.

Original equation: $\partial u/\partial t + c\ \partial u/\partial x = 0$.

Modified equation:

$$\partial u/\partial t + c \cdot \partial u/\partial x = c \cdot \Delta x/2 \cdot (1 - CFL) \cdot \partial^2 u/\partial x^2 + \cdots \quad (5.30)$$

The term $c \cdot \Delta x/2 \cdot (1 - CFL) \cdot \partial^2 u/\partial x^2$ is a term for artificial scattering.

5.6.4. Numerical diffusion coefficient

The numerical diffusion coefficient n_{num} is defined:

$$n_{num} = \frac{c\,\Delta x}{2}(1 - CFL) \quad (5.31)$$

The greater n_{num}, the stronger the dissipation.

5.6.5. Effects of dissipation

Positive effects:

- Smoothing of stray oscillations.
- Stabilization of the scheme.
- Removing unresolved high-frequency modes.

Negative effects:

- Loss of wave amplitude.
- Smoothing of steep fronts.
- Excessive smoothing of fine details.
- Systematic error in the solution.

5.6.6. Scheme Comparison

Table 5.1. ***Comparative dissipation of selected schemes***

Scheme	Dissipation	Application
Upwind	Strong	Shocks, stability
Centered	None	Smooth waves
Lax-Friedrichs	Very strong	Robustness
Lax-Wendroff	Low	Accuracy

5.7. Numerical Dispersion

5.7.1. Dispersion Concept

Numerical dispersion is the distortion of a wave profile due to the fact that different Fourier components propagate at different speeds in the scheme.

It is manifested by the appearance of parasitic oscillations and the progressive deformation of the waveform.

5.7.2. Numerical Phase Velocity

For a Fourier mode $u_i^n = G^n \cdot exp(i\beta x_i)$:

Exact phase velocity:

$$c_{phase} = \omega/\beta \tag{5.32}$$

Numerical phase velocity:

$$c_{num} = \omega_{num}/\beta \tag{5.33}$$

Dispersion appears when c_{num} depends on β (frequency).

5.7.3. Numerical dispersion relation

Von Neumann analysis for $u_i^n = G^n \cdot exp(i\beta x_i)$:

$$G = exp(-i\omega_{num} \cdot \Delta t) \tag{5.34}$$

The relation $\omega_{num}(\beta)$ is the numerical dispersion relation.

5.7.4. Example: Lax-Wendroff Scheme

For the advection equation $\partial u/\partial t + c\ \partial u/\partial x = 0$:

Amplification factor:

$$G = 1 - i \cdot CFL \cdot sin(\beta\Delta x) - CFL^2 \cdot \left(1 - cos(\beta\Delta x)\right) \tag{5.35}$$

Modulus: $|G| = 1$ (no dissipation).

Phase:

$$\omega_{num} \cdot \Delta t = arctan\left[CFL \cdot sin(\beta\Delta x)/\left(1 - CFL^2 \cdot \left(1 - cos(\beta\Delta x)\right)\right)\right] \tag{5.36}$$

Numerical Phase Velocity:

$$c_{num}/c = arctan[\dots]/(CFL \cdot \beta\Delta x) \tag{5.37}$$

For small $\beta\Delta x$:

$$\beta\Delta x: c_{num} \approx c \cdot \left(1 - (\beta\Delta x)^2/6 \cdot (1 - CFL^2)\right) \quad (5.38)$$

High frequencies propagate less quickly → dispersion.

5.7.5. Effects of Dispersion

- Parasitic oscillations behind a front.
- Progressive profile deformation.
- Localized signal spread.
- Difficulties for multi-scale problems.

5.7.6. Non-dispersive schemes

A scheme is non-dispersive if its dispersion relation is linear:

$$\omega_{num} = c \cdot \beta exactly \quad (5.39)$$

Examples:

- Upwind scheme with $CFL = 1$ (exact solution!).
- Some compact schemes.

5.7.7. Dissipation–dispersion trade-off

Fundamental dilemma:

- Dissipative schemes: dampen oscillations but lose amplitude.
- Dispersive schemes: retain amplitude but create oscillations.
- Optimal schemes: balance between the two.

5.8. Von Neumann Method Analysis

5.8.1. Principle of the Method

The von Neumann method analyzes the evolution of Fourier modes to determine the stability, dissipation, and dispersion of a scheme.

5.8.2. Systematic approach

Step 1:

Suppose a solution of the form $u_i^n = G^n \cdot exp(i\beta x_i)$.

Step 2:

Substitute in the numerical scheme.

Step 3:

Simplify to $G(\beta, CFL) or G(\beta, Fo)$

Step 4:

Analyze:

- Stability: $|G| \leq 1$ for all β.
- Dissipation: $|G| < 1$.
- Dispersion: $arg(G) \neq -c\,\beta\Delta t$.

5.8.3. Detailed Application: Implicit Scheme

Heat equation: $\partial u/\partial t = \alpha \cdot \partial^2 u/\partial x^2$.

Scheme:

$$u_i^{(n+1)} - Fo \cdot \left(u_{(i+1)}^{(n+1)} - 2u_i^{(n+1)} + u_{(i-1)}^{(n+1)}\right) = u_i^n \tag{5.40}$$

where $Fo = \alpha \cdot \Delta t/(\Delta x)^2$.

Analysis:

Substitution:

$$G^{(n+1)} \cdot exp(i\beta x_i) - Fo \cdot G^{(n+1)} \cdot [\dots] = G^n \cdot exp(i\beta x_i) \tag{5.41}$$

After simplification:

$$G \cdot [1 + 4Fo \cdot sin^2(\beta\Delta x/2)] = 1 \tag{5.42}$$

$$G = 1/[1 + 4Fo \cdot sin^2(\beta\Delta x/2)] \tag{5.43}$$

Conclusions:

- $|G| < 1$ for any $Fo > 0$: unconditionally stable.
- Real positive G: no dispersion.
- $G < 1$: intrinsic dissipation (correct damping).

5.8.4. Limitations of the Method

The von Neumann method applies only to:

- Linear equations with constant coefficients.
- Infinite or periodic domains.
- Constant-step schemes.

For other cases, use matrix analysis or energy methods.

5.9. Matrix Analysis of Stability

5.9.1. Matrix Formulation

For a semi-discrete problem:

$$\frac{d\,U}{d\,t} = A \cdot U \tag{5.44}$$

where U is the vector of the values at the grid points and A is the spatial discretization matrix.

5.9.2. Explicit Euler Scheme

$$U^{n+1} = U^n + \Delta t \cdot A \cdot U^n = (I + \Delta t \cdot A)\, U^n \tag{5.45}$$

Stability Condition:

All eigenvalues λ_k of $(I + \Delta t \cdot A)$ must satisfy $|\lambda_k| \leq 1$.

If λ are the eigenvalues of A:

$$|1 + \Delta t\lambda| \leq 1 \tag{5.46}$$

5.9.3. Implicit Schemes

Implicit Euler.

$$U^{n+1} = (I - \Delta t \cdot A)^{-1}\, U^n \tag{5.47}$$

Amplification factor:

$$G = 1/(1 - \Delta t \cdot \lambda) \tag{5.48}$$

Stable if $U^{(n+1)} = (I - \Delta t \cdot A/2)^{(-1)} \cdot (I + \Delta t; A/2) \cdot U^n$ (dissipative problem).

Crank-Nicolson.

$$U^{n+1} = (I - \Delta t\, A/2)^{-1}\, (I + \Delta t\, A/2)\, U^n \tag{5.49}$$

$$G = \frac{1 + \Delta t\, \lambda/2}{1 - \Delta t\, \lambda/2} \tag{5.50}$$

$|G| \leq 1 \quad$ for $\quad Re(\lambda) \leq 0.$

5.9.4. Spectral Radius and Matrix Standard

Stability condition by norm:

$$|\,G\,| \leq 1 + O(\Delta t) \tag{5.51}$$

where G is the amplification matrix.

The spectral radius $\rho(G) = \max|\lambda_k(G)|$ must satisfy $\rho(G) \leq 1$.

5.9.5. Boundary Conditions

Matrix analysis allows to take into account boundary conditions, unlike von Neumann which assumes an infinite domain. The matrix A reflects:

- Interior discretization.
- Boundary conditions (Dirichlet, Neumann, Robin).
- Source terms.

5.10. Optimization of Numerical Schemes

5.10.1. Quality criteria

An optimal scheme must reconcile:

- High order of convergence.
- Robust stability (wide stability range).
- Low numerical dissipation.
- Low numerical dispersion.
- Reasonable computational cost.
- Ease of implementation.

5.10.2. Fundamental Trade-offs

Stability vs Accuracy:

Stable (upwind) schemes often less precise than centered schemes.

High Order vs Complexity:

High-order schemes require more points, more calculations.

Explicit vs Implicit:

Explicit: simple but Δt small. Implicit: stable but expensive.

5.10.3. Optimization Techniques

Hybrid schemes:

IMEX (Implicit-Explicit): implicit stiff part, explicit non-stiff part.

Flux Limiters:

Combination of high-order (smooth areas) and low-order (shocks) schemes.

Adaptive time step:

Adjust Δt as the solution evolves.

Adaptive Mesh Refinement:

Fine mesh locally where required.

5.10.4. Practical Recommendations

For parabolic equations:

- Crank-Nicolson for precision and stability.
- ADI for 2D/3D problems.
- $Fo \approx 0.5–1$ for efficiency.

For hyperbolic equations:

- TVD, WENO schemes to capture shocks.
- Strictly comply with CFL conditions.
- Limiters to prevent oscillations.

For elliptic equations:

- Multigrid methods for rapid convergence.
- Preconditioning for large systems.
- Douglas-Rachford for efficiency.

5.10.5. Validation of Results

Always check:

21. Convergence in mesh (observed order = theoretical order).
22. Conservation of integral quantities.
23. Respect for physical properties (positivity, monotonicity).
24. Mesh independence (converged solution).
25. Comparison with analytical solutions or benchmarks.

5.10.6. Summary Table

Table 5.2. ***Properties to verify for a numerical scheme***

Property	Testing	Indicator	Target	Action
Consistency	Taylor	$\tau \to 0$	$O(h^p)$	Improve scheme
Stability	von Neumann	$\|G\| \leq 1$	$\forall\beta$	Reduce Δt
Convergence	Refinement	$E \sim h^p$	p high	Check
Dissipation	Amplitude	$\|G\|$	≈ 1	Centered scheme
Dispersion	Phase	$arg(G)$	Linear	High order

General Conclusion

Numerical error analysis is the theoretical foundation of any PDE simulation. The concepts of consistency, stability and convergence form an inseparable triptych, formalized by Lax's theorem, which establishes their equivalence for well-posed problems.

Key findings:

- Consistency ensures that the scheme approaches the PDE well. It is easily verified by Taylor expansions and is a necessary but not sufficient condition.
- Stability ensures that errors do not grow uncontrollably. Its analysis, via von Neumann or matrix approach, is often delicate but crucial.
- Convergence, equivalent to stability for a consistent scheme, ensures that the numerical solution tends towards the exact solution. This is the ultimate goal of any numerical method.

Parasitic phenomena:

- Numerical dissipation, useful for stabilization but damaging to accuracy, must be controlled. Centered schemes minimize it while off-center schemes deliberately introduce it for stability.
- Numerical dispersion distorts the wave profiles and creates parasitic oscillations. High-order schemes reduce it but often require limiters near discontinuities.

Practical perspectives:

The choice of a scheme is always the result of a compromise between accuracy, stability, computational cost and ease of implementation. Modern hybrid schemes (IMEX, TVD, WENO) seek to combine the advantages of different approaches.

Rigorous error analysis, far from being a purely theoretical exercise, guides the design of robust and reliable numerical methods, essential for modern scientific and industrial simulations.

Bibliographical references

P.D. Lax and R.D. Richtmyer, "Survey of the stability of linear finite difference equations", Communications on Pure and Applied Mathematics, vol. 9, pp. 267-293, 1956.

G. Strang, "On the construction and comparison of difference schemes", SIAM Journal on Numerical Analysis, vol. 5, pp. 506-517, 1968.

J.C. Strikwerda, Finite Difference Schemes and Partial Differential Equations, 2nd edition, SIAM, 2004.

K.W. Morton and D.F. Mayers, Numerical Solution of Partial Differential Equations, 2nd edition, Cambridge University Press, 2005.

R.J. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations, SIAM, 2007.

A. Iserles, A First Course in the Numerical Analysis of Differential Equations, 2nd edition, Cambridge University Press, 2008.

J.W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods, Springer, 1995.

C. Hirsch, Numerical Computation of Internal and External Flows, Vol. 1 and 2, Butterworth-Heinemann, 2007.

S. Gottlieb, C.W. Shu and E. Tadmor, "Strong Stability-Preserving High-Order Time Discretization Methods", SIAM Review, vol. 43, pp. 89-112, 2001.

Corrected Examples (results + method)

Example 5.1 (Gauss elimination).

Statement. Solve $[[3,-1],[2,4]]\ [x;y] = [1;\ 10]$.

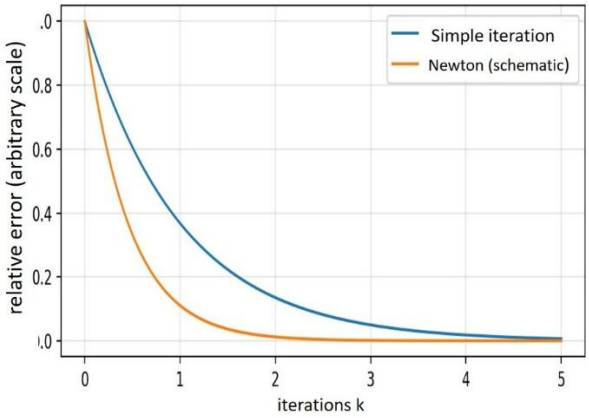
Solution (summary). Elimination $\Rightarrow y\ =\ 2$ then $x\ =\ 1$.

Example 5.2 (Conditioning).

Statement. Interpreting $\kappa(A)$ and error amplification.

Solution (summary). $|\ \delta x\ |/|\ x\ |\ \leq \kappa(A)\ |\ \delta b\ |/|\ b\ |$. Partial pivoting recommended.

Chapter 6 — Finite Volume Method



Chapter 6. Finite Volume Method

Computational Fluid Dynamics Applications and Turbulence Modeling

Chapter Objectives

- Understand the key concepts and the digital issue.
- Build an algorithm (steps + pseudo-code) and apply it.
- Interpret the error and check the stability/convergence on a simple case.

Table of Contents

6.1. Introduction to the Finite Volume Method

6.1.1. Background and Motivation

The finite volume method (FVM) represents a fundamental approach in computational fluid dynamics (CFD). Unlike finite differences that approach differential equations directly, the FVM starts from the integral form of conservation laws, thus naturally ensuring the conservation of fundamental physical quantities.

6.1.2. Fundamental Principle

The FVM subdivides the computational domain into elementary control volumes. For each volume, the conservation of a physical quantity (mass, momentum, energy) is expressed in the form of an integral balance:

$$\int\int\int_V (\partial\varphi/\partial t)dV + \int\int_S (F \cdot n)dS = \int\int\int_V S_\varphi dV \qquad (6.1)$$

where:

- φ: conserved variable (mass, momentum, energy, etc.).
- F: φ flux through surfaces.
- S_φ: volume source term.
- V: control volume.
- S: control volume surface area.

6.1.3. Design advantages

Strict conservation:

FVM automatically ensures that physical quantities are conserved at the discrete level, because the fluxes out of one cell enter the neighboring cell exactly.

Clear Physical Interpretation:

Each discrete equation represents a physical balance on an elementary volume, which makes it easier to understand and verify.

Geometric flexibility:

FVM naturally adapts to unstructured meshes and complex geometries.

6.1.4. Areas of application

The FVM is the gold-standard method for:

- External aerodynamics (flows around obstacles).
- Combustion and chemical reactions.
- Multiphase flows.
- Coupled heat transfers.

- Turbulence and complex flows.
- Compressible gas dynamics.

6.1.5. FVM-Based Software

The most commonly used CFD codes use FVM:

- OpenFOAM (open source).
- ANSYS Fluent.
- STAR-CCM+.
- CFX.
- PHOENICS.

6.2. Mathematical Foundations of FVM

6.2.1. General Transport Equation

General differential form:

$$\partial(\rho\varphi)/\partial t + div(\rho\varphi) = div(\Gamma grad\varphi) + S_\varphi \quad (6.2)$$

where:

- ρ: density.
- u: velocity vector.
- Γ: diffusion coefficient.
- S_φ: source term.

This equation includes:

- Time term: $\partial(\rho\varphi)/\partial t$.
- Convective term: $div(\rho u\varphi)$.
- Diffusive term: $div(\Gamma grad\varphi)$.
- Source term:S_φ.

6.2.2. Integration on a control volume

By integrating on a volume V and applying Gauss's theorem:

$$\int_V(\partial(\rho\varphi)/\partial t)dV + \int_S(\rho\varphi \cdot n)dS = \int_S(\Gamma grad\varphi \cdot n)dS + \int_V S_\varphi dV \quad (6.3)$$

For a finite volume, this expression becomes:

$$d(\rho\varphi V)/dt + \Sigma_f aces(\rho\varphi A)_f = \Sigma_f aces(\Gamma(grad\varphi); A)_f + S_\varphi V \quad (6.4)$$

6.2.3. Temporal discretization

Explicit scheme (forward Euler):

$$(\rho\varphi V)^{(n+1)} = (\rho\varphi V)^n + \Delta t \cdot [F^n + D^n + S^n] \quad (6.5)$$

Implicit scheme (backward Euler):

$$(\rho\varphi V)^{(n+1)} = (\rho\varphi V)^n + \Delta t \cdot \left[F^{(n+1)} + D^{(n+1)} + S^{(n+1)}\right] \quad (6.6)$$

Crank-Nicolson scheme:

$$(\rho\varphi V)^{(n+1)} = (\rho\varphi V)^n + \Delta t/2; \left[(F + D + S)^n + (F + D + S)^{(n+1)}\right] \quad (6.7)$$

6.2.4. Approximation of convective fluxes

The convective flux through a face f is:

$$F_f = (\rho u A)_f \cdot \varphi_f \quad (6.8)$$

Several schemes to approximate φ_f:

Centered Scheme (CDS):

$$\varphi_f = (\varphi_P + \varphi_N)/2 \quad (6.9)$$

Advantage: order 2, symmetrical. Drawback: can oscillate.

Upwind Scheme:

$$\varphi_f = \varphi_P si u_f > 0, \varphi_f = \varphi_N si u_f < 0 \quad (6.10)$$

Advantage: stable, no oscillations. Drawback: order 1, diffusive.

Hybrid scheme:

CDS/upwind combination according to local Péclet number.

6.2.5. Approximation of diffusive fluxes

The diffusive flux through the face f:

$$D_f = \Gamma_f \cdot (grade\varphi)_f \cdot A_f \quad (6.11)$$

For orthogonal meshes:

$$(grade\varphi)_f \approx (\varphi_N - \varphi_P)/d_P N \quad (6.12)$$

where d_{PN} is the distance between the centers P and N.

6.2.6. Assembling the Linear System

For each volume P, the discretized equation takes the form:

$$a_P \cdot \varphi_P = \Sigma_n b a_n b \cdot \varphi_n b + b_P \quad (6.13)$$

where:

- a_P: central coefficient.
- a_{nb}: coefficients of neighbors.
- b_P: source term.

The overall system is sparse and usually solved by iterative methods.

6.3. Finite Volumes vs. Finite Differences Comparison

6.3.1. Fundamental Philosophy

Finite differences (FD):

Algebraic approach: direct approximation of derivatives by differences. Starts from the differential form of the equation.

Finite volumes (FV):

Physical approach: balance of fluxes on control volumes. Starts from the conservative integral form.

6.3.2. Conservation of quantities

In FV, the fluxes leaving a cell enter exactly the neighboring cells. This property ensures overall conservation, which is essential for physically realistic simulations.

6.3.3. Geometric flexibility

Finite differences:

- Typically requires Cartesian structured meshes.
- Difficulty with complex geometries.
- Coordinate transformations required.

Finite volumes:

- Naturally adapts to unstructured meshes.
- Arbitrary polyhedra (tetrahedra, hexahedra, prisms, etc.).
- Ease of local refinement.
- Ideal for complex industrial geometries.

6.3.4. Accuracy and Order

Finite differences:

- Ease of reaching high orders (order 4, 6, 8…).
- Compact schemes possible.
- Excellent accuracy on regular meshes.

Finite volumes:

- Typically order 1 or 2.
- High order more complex to implement.
- Accuracy may degrade on distorted meshes.

6.3.5. Boundary Conditions

Finite differences:

- Treatment sometimes delicate.
- Requires special edge schemes.
- Can affect the overall order.

Finite volumes:

- Natural treatment via fluxes at boundary faces.
- Direct integration into formulation.
- Consistency with the interior of the domain.

6.3.6. Computational Cost

Finite differences:

- Generally less storage.
- Simpler operations.
- Matrices often more structured.

Finite volumes:

- More geometric metadata (connectivity, surfaces, etc.).
- Additional geometric calculations.
- Less structured matrices for unstructured meshes.

6.3.7. Summary Table

Table 6.1. ***Comparison between Finite Differences and Finite Volumes***

Aspect	Finite Differences	Finite Volumes
Conservation	Not guaranteed	Strict
Geometry	Structured mesh	Any mesh size
Max order	Very high (8+)	Moderate (2–3)
Implementation	Simpler	More complex
Applications	Academic, DNS	Industrial, CFD
Robustness	Variable	Excellent
CFD popularity	Limited	Dominant

6.4. Discretization of Conservation Equations

6.4.1. Incompressible Navier-Stokes equations

System of governing equations:

Conservation of mass:

$$div(u) = 0 \tag{6.14}$$

Conservation of momentum:

$$\partial(\rho u)/\partial t + div(\rho u u) = -gradp + div(\mu gradu) + S_M \tag{6.15}$$

where:

- u: velocity vector (U, V, W).
- p: pressure.
- μ: dynamic viscosity.
- S_M: volume forces (gravity, etc.).

6.4.2. Pressure-velocity coupling problem

The Navier-Stokes system poses a major challenge: the continuity equation does not explicitly contain the pressure. It is therefore necessary to develop special algorithms to couple pressure and velocity.

Main difficulties:

- No explicit equation for pressure.
- Necessity of simultaneously satisfying continuity and momentum.
- Potential numerical instabilities.
- High computational cost.

6.4.3. Staggered grids

To avoid pressure oscillations (checkerboard pattern), staggered meshes are used:

- Pressure stored at cell centers.
- Velocity components at cell faces.
- u on faces perpendicular to x.
- v on faces perpendicular to y.
- w on faces perpendicular to z.

Advantage:

Natural coupling between pressure gradients and velocity.

6.4.4. Co-located meshes

Modern alternative to staggered meshes:

- All variables in one place (cell center).
- Easier to implement.
- Requires Rhie-Chow interpolation to avoid oscillations.
- Used in most modern codes.

6.4.5. Discretized Momentum Form

For a velocity component u:

$$a_P \cdot u_P = \Sigma_n b a_n b \cdot u_n b - V_P \cdot (\partial p/\partial x)_P + b_P \qquad (6.16)$$

where:

- Coefficients a include convection and diffusion.
- Pressure term treated separately.
- b_P contains source terms and time.

6.5. Pressure-Velocity Coupling: SIMPLE Family

6.5.1. General Concept

The SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) family of algorithms solves pressure-velocity coupling in a sequential and iterative manner.

Principle:

11. Guess a pressure field p^*.
12. Solve the momentum equations $\rightarrow u^*$.
13. Correct p^* and u^* to satisfy continuity.
14. Iterate to convergence.

6.5.2. Pressure correction equation

Definition of corrections:

$$p = p^* + p' \quad (corrected\ pressure) \tag{6.17}$$

$$u = u^* + u' \quad (corrected\ velocity) \tag{6.18}$$

Inserting into the continuity equation, we obtain the correction equation:

$$div(s) = 0 \rightarrow div(s*) + div(s') = 0 \tag{6.19}$$

The pressure correction equation derives from this condition.

6.5.3. Velocity Correction Equation

Simplification of the momentum equation:

$$u' \approx -V/a_P \cdot gradP' \tag{6.20}$$

This approximation neglects certain terms to avoid a complex implicit loop.

6.5.4. Under-relaxation strategies

To improve stability and convergence:

Velocity under-relaxation:

$$u_new = u_old + a_u; (u* - u_old) \tag{6.21}$$

Pressure under-relaxation:

$$p_new = p_old + \alpha_p \cdot p' \tag{6.22}$$

Typical values: $\alpha_u = 0.7, \quad \alpha_p = 0.3$.

6.6. SIMPLE Algorithm

6.6.1. History

The Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm was developed by Patankar and Spalding in 1972. It remains one of the most widely used algorithms in commercial and academic CFD.

6.6.2. Detailed Flowchart

Step 0: Initialization.

- Guess p^*, u^*, v^*, w^*.
- Define convergence criteria ε.

Step 1: Solve momentum.

Solve for u^* with p^*:

$$a_P \cdot u *_P = \Sigma_n b a_n b \cdot u *_n b - V \cdot (\partial p */\partial x) + b \quad (6.23)$$

- Same for v^* and w^*.

Step 2: Mass flux calculation.

- ${m_f}^* = \rho {u_f}^* A_f$ for each face.
- These fluxes generally do not satisfy continuity.

Step 3: Solve the pressure equation.

- Form the equation:

$$\Sigma_f aces(\rho V/a_P)_f; (\nabla p')_f \cdot A_f = S_f acesm *_f \quad (6.24)$$

- Solve for p'.

Step 4: Pressure correction.

$$p = p * + \alpha_p \cdot p' \quad (6.25)$$

- where $\backslash alpha_p$ is the under-relaxation factor (typ. 0.3).

Step 5: Velocity correction.

$$u = u * -(V/a_P) \cdot (\partial p'/\partial x)$$
$$v = v * -(V/a_P) \cdot (\partial p'/\partial y)$$
$$\boldsymbol{w = w * -(V/a_P) \cdot (\partial p'/\partial z)}$$

Step 6: Solve the scalars.

- Temperature, concentration, turbulence, etc.
- Uses updated velocity field.

Step 7: Convergence testing.

- Check: $|R| < \varepsilon$ where R is the residual.
- If non-converged: back to step 1.
- If converged: end.

6.6.3. Features of SIMPLE

Advantages:

- Robust and stable.
- Well tested and validated.
- Relatively simple implementation.
- Works for a wide range of flows.

Drawbacks:

- Slow convergence.
- Requires significant under-relaxation.
- Sensitive to under-relaxation parameters.
- May require thousands of iterations.

6.6.4. Factors Affecting Convergence

- Mesh quality (orthogonality, aspect ratio).
- Choice of under-relaxation factors.
- Solution field initialization.
- Reynolds number.
- Geometry complexity.
- Presence of recirculations.

6.7. Variants: SIMPLER and SIMPLEC

6.7.1. SIMPLER Algorithm

SIMPLER (SIMPLE Revised) was developed to improve the convergence of SIMPLE. The key idea: solve an explicit equation for pressure instead of correction.

6.7.2. Differences with SIMPLE

In SIMPLE:

- We solve for p' (pressure correction).
- $p = p* + \alpha_p \cdot p'$

In SIMPLER:

- We solve directly for p.
- Equation derived from u^* neglecting neighboring terms.
- p' used only to correct velocity.

6.7.3. SIMPLER Steps

15. Solve momentum with $p^* \rightarrow u^*$.
16. Solve pressure equation $\rightarrow p$ (not p').
17. Correct velocity with p (not p') $\rightarrow u$.
18. Solve momentum again with p new.
19. Solve correction equation $\rightarrow p'$.
20. Correct velocity $\rightarrow u$ final.

6.7.4. Benefits of SIMPLER

- Convergence generally faster than SIMPLE.
- Less sensitive to pressure under-relaxation.
- Best pressure estimation.
- Effective for highly convective problems.

6.7.5. SIMPLEC Algorithm

SIMPLEC (SIMPLE-Consistent) improves SIMPLE by correcting the approximation of the velocity correction equation.

6.7.6. Correction by SIMPLEC

SIMPLE uses:

$$u' = -(V/a_P) \cdot \nabla p' \tag{6.26}$$

SIMPLEC uses:

$$u' = -\big(V/(a_P - \Sigma a_n b)\big) \cdot \nabla p' \tag{6.27}$$

This correction makes the equation more consistent with the full momentum equation.

6.7.7. Benefits of SIMPLEC

- Allows for higher under-relaxation factors ($\backslash alpha_p \approx 0.9–1.0$).
- Faster than SIMPLE convergence.
- Fewer iterations needed.
- Particularly effective for fine meshes.

6.7.8. Comparison of the Three Algorithms

Table 6.2. ***Comparison of SIMPLE, SIMPLER, and SIMPLEC algorithms***

Criterion	SIMPLE	SIMPLER	SIMPLEC
Speed	Slow	Fast	Fast
Typical α_p	0.3	0.5	0.9–1.0
Robustness	Excellent	Good	Good
Cost/iter	Low	High	Low
Usage	Standard	Difficult problems	Fine meshes

6.8. QUICK Discretization Scheme

6.8.1. Motivation

QUICK (Quadratic Upstream Interpolation for Convective Kinematics) was developed by Leonard in 1979 to improve the accuracy of the approximation of convective terms.

Problem:

- Upwind order 1: stable but very diffusive.
- Centered order 2: accurate but can oscillate.
- Need: high order + stability.

6.8.2. Principle of QUICK

QUICK uses quadratic (parabolic) interpolation based on three points: two upstream and one downstream of the point of interest.

For a left-to-right flow ($W \rightarrow P \rightarrow E$):

$$\varphi_f ace = (6\varphi_P + 3\varphi_E - \varphi_W)/8 \qquad (6.28)$$

This formula gives a 3rd-order scheme on uniform meshes.

6.8.3. General Formulation

For a face between P and E with flow $u > 0$:

$$\varphi_e = \varphi_P + (\nabla\varphi)_P \cdot (x_e - x_P) + 1/2 \cdot (\nabla^2 f)_P \cdot (x_e - x_P)^2 \quad (6.29)$$

The gradients are estimated by upwind differences:

$$(\nabla\varphi)_P \approx (\varphi_P - \varphi_W)/\delta x_w p \quad (6.30)$$

$$(\nabla^2\varphi)_P \approx (\varphi_E - 2\varphi_P + \varphi_W)/(\delta x)^2 \quad (6.31)$$

6.8.4. Properties of QUICK

Advantages:

- Order 3 in space (uniform mesh).
- Low numerical diffusion.
- Good gradient capture.
- Suitable for rotational flows.

Drawbacks:

- Can produce oscillations (overshoots/undershoots).
- Requires three upstream points.
- Special treatment near boundaries.
- More expensive than upwind.

6.8.5. Péclet number and stability

The local Péclet number is:

$$Pe = \rho|u|\delta x/\Gamma \quad (6.32)$$

QUICK behavior:

- Small Pe (diffusion dominant): excellent.
- Moderate Pe: very good.
- Very high Pe: risk of oscillations.

6.8.6. Limitation Techniques

To avoid oscillations, use limiters:

TVD-QUICK:

Combine QUICK with the TVD principle to guarantee monotonicity.

Blended QUICK:

$$\varphi_e = \alpha \cdot \varphi_{QUICK} + (1 - \alpha) \cdot \varphi_{upwind} \quad (6.33)$$

where $\alpha \in [0, 1]$ locally adjusted.

6.8.7. Recommended Applications

QUICK is particularly effective for:

- Swirl flows.
- Mixing layers.
- Jets and plumes.
- Flows with strong curvature of streamlines.
- Problems requiring low numerical diffusion.

6.9. Modeling Turbulence: k-ε

6.9.1. Introduction to Turbulence

Turbulence is a phenomenon characterized by three-dimensional chaotic fluctuations in velocity and pressure fields. Direct simulation (DNS) requires prohibitive resources, so turbulence models are used.

6.9.2. Reynolds decomposition

Mean/fluctuation separation:

$$u = \bar{u} + u'(speed) \tag{6.34}$$

$$p = \bar{p} + p'(pressure) \tag{6.35}$$

where the bar indicates the time average.

6.9.3. RANS Equations

The Reynolds-Averaged Navier-Stokes (RANS) equations are written:

$$\partial \bar{u}_i / \partial x_i = 0 \tag{6.36}$$

$$\partial(\rho \bar{u}_i)/\partial t + \partial\left(\rho \bar{u}_i \bar{u}_j\right)/\partial x_j = -\partial \bar{p}/\partial x_i + \partial/\partial x_j\left[\mu\left(\partial \bar{u}_i/\partial x_j\right) - \rho u_i' u_j'\right] \tag{6.37}$$

The term $-\rho u_i' u_j'$ is the Reynolds stress tensor that requires modeling.

6.9.4. Concept of Turbulent Viscosity

Boussinesq's hypothesis:

$$\rho u_i' u_j' = \mu_t\left(\partial \bar{u}_i/\partial x_j + \partial \bar{u}_j/\partial x_i\right) - 2/3 \cdot \rho k \cdot \delta_i j \tag{6.38}$$

where:

- μ_t:turbulent viscosity (to be modelled).
- $k = 1/2 \cdot u_i' u_i'$: turbulent kinetic energy.

6.9.5. Standard k–ε Model

The k–ε model introduces two transport equations.

Equation for k (turbulent kinetic energy):

$$\partial(\rho k)/\partial t + div(\rho u k) = div[(\mu + \mu_t/\sigma_k) grad k] + P_k - \rho\varepsilon \quad (6.39)$$

Equation for ε (turbulent dissipation):

$$\partial(\rho\varepsilon)/\partial t + div(\rho u\varepsilon) = div[(\mu + \mu_t/\sigma_\varepsilon) grad\varepsilon] + C_1\varepsilon \cdot \varepsilon/k \cdot P_k - C_2\varepsilon \cdot \rho\varepsilon^2/k \quad (6.40)$$

where:

- $P_k = \mu_t(\partial \bar{u}_i/\partial x_j)(\partial \bar{u}_i/\partial x_j + \partial \bar{u}_j/\partial x_i)$: production of k.
- $\sigma_k, \sigma_\varepsilon, C_1\varepsilon, C_2\varepsilon$: model constants.

6.9.6. Calculating Turbulent Viscosity

Turbulent viscosity is calculated by:

$$\mu_t = \rho \cdot C_\mu \cdot k^2/\varepsilon \quad (6.41)$$

Standard constants of the k–ε model:

Table 6.3. ***Standard constants of the k–ε turbulence model***

Constant	Value
C_μ	0.09
$C_{1\varepsilon}$	1.44
$C_{2\varepsilon}$	1.92
σ_k	1.0
σ_ε	1.3

6.9.7. Boundary conditions for k and ε

At the inlet:

$$k_in = 3/2 \cdot (U_in \cdot I)^2 \quad (6.42)$$

where I is the turbulent intensity (typ. 1–5%).

$$\varepsilon_i n = C_\mu^{(3/4)} \cdot k^{(3/2)} / L_t \quad (6.43)$$

where L_t is turbulent length scale.

On the walls — Logarithmic wall function:

$$u^+ = 1/\kappa \cdot ln(y^+) + B \quad (6.44)$$

where $\kappa = 0.41$ (von Karman's constant), $B = 5.2$.

6.9.8. Variants of the k–ε model

k–ε RNG (Renormalization Group):

- Theoretically derived constants.
- Best for swirl and recirculations.

k–ε Realizable:

- Satisfies mathematical constraints on Reynolds stresses.
- C_μ variable.
- Best for jets, mixing layers.

6.9.9. Limitations of the k–ε model

- Poor performance near walls.
- Difficulties with separated flows.
- Not suitable for transitional flows.
- Assumes isotropic turbulence.
- Requires wall functions or very fine mesh.

6.10. Choice of Method and Recommendations

6.10.1. Decision tree for method choice

Question 1: Type of geometry?

- Simple, rectangular geometry: finite differences possible.
- Complex, industrial geometry: finite volumes required.

Question 2: Nature of the flow?

- Incompressible, low speed: SIMPLE/SIMPLEC.
- Compressible: pressure-density coupled.
- Turbulent: add k–ε model or higher.

Question 3: Importance of conservation?

- Critical (combustion, multiphase): finite volumes.
- Moderate: finite differences acceptable.

Question 4: Precision required?

- Very high, DNS: high-order finite differences.
- Standard, engineering: finite volumes order 2.
- Areas with gradients: QUICK or MUSCL.

6.10.2. Recommendations by Application Type

External aerodynamics:

- Method: finite volumes.
- Algorithm: SIMPLE or SIMPLEC.
- Discretization: QUICK for convection.
- Turbulence: k–ε Realizable or k–ω SST.
- Mesh: refined near walls, boundary layer prisms.

Heat exchangers:

- Method: finite volumes.
- Algorithm: SIMPLE (robust).
- Discretization: upwind order 2.
- Turbulence: k–ε standard.
- Consideration: fluid-solid thermal coupling.

Pipe flows:

- Method: finite volumes or differences.
- Algorithm: SIMPLEC (rapid convergence).
- Discretization: centered order 2.
- Turbulence: k–ε or k–ω according to Reynolds.

Combustion:

- Method: finite volumes (conservation essential).
- Algorithm: SIMPLE with strong under-relaxation.
- Discretization: upwind or hybrid (stability).
- Turbulence: k–ε RNG or RSM.
- Special: combustion models (EDC, PDF).

6.10.3. Recommended Under-Relaxation Settings

Table 6.4. ***Recommended under-relaxation factors for SIMPLE and SIMPLEC***

Variable	SIMPLE	SIMPLEC
Pressure	0.2–0.3	0.9–1.0
Velocity	0.5–0.7	0.7–0.9
Turbulence (k, ε)	0.5–0.8	0.7–0.9
Energy	0.8–0.95	0.9–1.0
Density	0.5–0.7	0.7–0.9
Scalars	0.8–1.0	0.9–1.0

Note: reduce if instability, increase if convergence is too slow.

6.10.4. Convergence criteria

Residuals:

- Continuity: $< 10^{-3}$ (minimum).
- Velocities: $< 10^{-3}$.
- k, ε: $< 10^{-3}$.
- Energy: $< 10^{-6}$.

For high accuracy: all residuals $< 10^{-5}$.

Monitoring:

- Check the stabilization of the quantities of interest (forces, fluxes).
- Global balances (mass, energy).
- Critical point profiles.

6.10.5. Practical Tips

Initialization:

- Start with a reasonable uniform field.
- Or solution of a simplified case.
- Avoid null values.

Phased Strategy:

- Start with upwind order 1 (stable).
- Upgrade to higher order once converged.
- Introduce turbulence gradually.

Mesh quality:

- Orthogonality $>$ 0.15.
- Aspect ratio $<$ 100 (ideal $<$ 10).
- Expansion $<$ 1.2 between neighboring cells.
- $y^+ \approx$ 30–300 with wall functions.
- $y^+ <$ 1 for full resolution.

6.10.6. Summary Table of Choices

Table 6.5. ***Summary of recommendations by precision level***

Criterion	Simple/Robust	Standard	High precision
Method	Finite volumes	Finite volumes	FD high order
Algorithm	SIMPLE	SIMPLEC	Coupled
Convection	Upwind	QUICK	WENO/Compact
Turbulence	k–ε standard	k–ε Realizable	LES/DNS
Mesh	Coarse	Medium	Very fine
Computation time	Hours	Days	Weeks

General Conclusion

The finite volume method has established itself as the reference approach in computational fluid dynamics thanks to its natural ability to guarantee the conservation of physical quantities and its geometric flexibility. Comparison with finite differences reveals fundamental trade-offs between formal accuracy and physical robustness.

Key points about FVM:

- The conservative formulation ensures accurate physical balances at the discrete level, essential for combustion, multiphase and coupled transfers.
- Adaptability to unstructured meshes makes it possible to deal with complex industrial geometries, impossible with conventional finite differences.
- The pressure-velocity coupling by SIMPLE/SIMPLER/SIMPLEC algorithms offers a robustness/efficiency compromise adapted to the incompressible case.

Discretization schemes:

The choice between upwind (robust but diffusive), centered (precise but oscillating) and QUICK (high order but expensive) must adapt to the problem: stability for difficult cases, precision for smooth flows, QUICK for vortices and strong gradients.

Turbulence modeling:

The k–ε model, despite its known limitations, remains a standard choice for engineering thanks to its robustness/cost balance. The RNG and Realizable variants improve performance for complex flows without prohibitive additional cost.

Practical recommendations:

The success of a CFD simulation is based on: (1) choice of method adapted to the problem, (2) quality mesh with appropriate refinement, (3) optimized under-relaxation parameters, (4) progressive convergence initialization strategy, (5) rigorous validation by balances and monitoring.

FVM, combined with SIMPLE algorithms and RANS turbulence models, forms the basic arsenal of modern industrial CFD, offering the best compromise between accuracy, robustness and computational cost for the vast majority of practical applications.

Bibliographical references

S.V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing, 1980.

S.V. Patankar and D.B. Spalding, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows", International Journal of Heat and Mass Transfer, vol. 15, pp. 1787-1806, 1972.

B.P. Leonard, "A stable and accurate convective modelling procedure based on quadratic upstream interpolation", Computer Methods in Applied Mechanics and Engineering, vol. 19, pp. 59-98, 1979.

H.K. Versteeg and W. Malalasekera, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, 2nd edition, Pearson, 2007.

J.H. Ferziger and M. Peric, Computational Methods for Fluid Dynamics, 3rd edition, Springer, 2002.

B.E. Launder and D.B. Spalding, "The numerical computation of turbulent flows", Computer Methods in Applied Mechanics and Engineering, vol. 3, pp. 269-289, 1974.

T.J. Chung, Computational Fluid Dynamics, 2nd edition, Cambridge University Press, 2010.

F. Moukalled, L. Mangani and M. Darwish, The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab, Springer, 2016.

D.C. Wilcox, Turbulence Modeling for CFD, 3rd edition, DCW Industries, 2006.

S.B. Pope, Turbulent Flows, Cambridge University Press, 2000.

Corrected Examples (results + method)

Example 6.1 (Upwind scheme).

Statement. Write the upstream scheme for $u_t + a\,u_x = 0, a > 0$.

Solution (abstract). $u_i^{n+1} = u_i^n - C(u_i^n - u_{i-1}^n), C = a\Delta t/\Delta x, C = a\,\Delta t/\Delta x$.

Stability: $0 \leq C \leq 1$.

Example 6.2 (CFL).

Statement. $a = 2, \Delta x = 0.01, \backslash mathrmCFL = 0.8 \Rightarrow \Delta t_{\backslash max}$?

Solution (summary). $\Delta t \leq 0.8 \times 0.01/2 = 0.004$ s.