Scientific Computing

DIWEN XU, University of Washington, USA

These notes are primarily based on the notes by J. Nathan Kutz.

1 Initial and Boundary Value Problems of Differential Equations

1.1 Initial Value Problems

$$\frac{d\boldsymbol{y}}{dt} = f(t, \boldsymbol{y}), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0, \quad t \in [0, T].$$

1.1.1 Euler Method.

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\, f(t_n, \boldsymbol{y}_n).$$

1.1.2 General One-Step Method (Runge–Kutta Family).

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\, \boldsymbol{\varphi},$$

where $\boldsymbol{\varphi}$ is chosen to reduce the one-step error.

Second-Order $\boldsymbol{\varphi}$. $Af(t,\ \boldsymbol{y}(t)) + Bf(t + P\Delta t,\ \boldsymbol{y}(t) + Q\Delta t\, f(t,\ \boldsymbol{y}(t)))$.

A Taylor expansion gives the order conditions

$$A + B = 1, \quad BP = \tfrac{1}{2}, \quad BQ = \tfrac{1}{2},$$

leaving one degree of freedom. Two common second-order $\boldsymbol{\varphi}$ are

$$\text{Heun (A=}\frac{1}{2}\text{)}.\ \frac{1}{2}f(t, \boldsymbol{y}(t)) + \frac{1}{2}f(t + \Delta t,\ \boldsymbol{y}(t) + \Delta t\, f(t, \boldsymbol{y}(t))),$$

$$\text{Modified Euler–Cauchy (A=0)}.\ f\left(t + \frac{\Delta t}{2},\ \boldsymbol{y}(t) + \frac{\Delta t}{2} f(t, \boldsymbol{y}(t))\right).$$

1.1.3 Fourth-Order Runge–Kutta Method (RK4).

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{\Delta t}{6}(f_1 + 2f_2 + 2f_3 + f_4),$$

with

$$\begin{aligned} f_1 &= f(t_n, \boldsymbol{y}_n), \\ f_2 &= f\left(t_n + \frac{\Delta t}{2},\ \boldsymbol{y}_n + \frac{\Delta t}{2} f_1\right), \\ f_3 &= f\left(t_n + \frac{\Delta t}{2},\ \boldsymbol{y}_n + \frac{\Delta t}{2} f_2\right), \\ f_4 &= f(t_n + \Delta t,\ \boldsymbol{y}_n + \Delta t\, f_3). \end{aligned}$$

This has local truncation error $O(\Delta t^5)$ and global error $O(\Delta t^4)$.

1.1.4 Adams Method.

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \int_{t_n}^{t_{n+1}} f(t, \boldsymbol{y})\, dt.$$

Approximating the integrand by a polynomial $\boldsymbol{p}(t)$ over the step yields multi-step formulas.

Adams–Bashforth (Explicit).

- 1st order (Euler). $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\, f_n$.
- 2nd order (AB2). $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{\Delta t}{2}(3f_n - f_{n-1})$.

Adams–Moulton (Implicit).

- 1st order (Backward Euler). $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\, f_{n+1}$.
- 2nd order (Trapezoidal). $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{\Delta t}{2}(f_{n+1} + f_n)$.

Predictor–Corrector (AB2/AM2). $\boldsymbol{y}^P_{n+1}$:

$$\begin{aligned} \text{Predict.} \quad \boldsymbol{y}^P_{n+1} &= \boldsymbol{y}_n + \frac{\Delta t}{2}(3f_n - f_{n-1}), \\ \text{Correct.} \quad \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{\Delta t}{2}(f_n + f(t_{n+1}, \boldsymbol{y}^P_{n+1})). \end{aligned}$$

1.2 Error Analysis for Time-Stepping Routines

1.2.1 Accuracy and Local vs. Global Error. Let the *global discretization error* at step k be

$$E_k = \boldsymbol{y}(t_k) - \boldsymbol{y}_k,$$

and the *local discretization error* be

$$\varepsilon_{k+1} = \boldsymbol{y}(t_{k+1}) - (\boldsymbol{y}(t_k) + \Delta t\, \phi),$$

where $\boldsymbol{y}(t_k)$ is the exact solution, $\boldsymbol{y}_k$ is the numerical solution, and $\boldsymbol{y}(t_k) + \Delta t\, \phi$ is a one-step approximation over $[t_k, t_{k+1}]$.

Scheme	Local error ε_k	Global error E_k
Euler (Forward)	$O(\Delta t^2)$	$O(\Delta t)$
2nd-order Runge–Kutta	$O(\Delta t^3)$	$O(\Delta t^2)$
4th-order Runge–Kutta	$O(\Delta t^5)$	$O(\Delta t^4)$
2nd-order Adams–Bashforth	$O(\Delta t^3)$	$O(\Delta t^2)$

1.2.2 Round-Off and Optimal Step-Size. A floating-point representation introduces round-off. For the Euler derivative approximation

$$\frac{d\boldsymbol{y}}{dt} \approx \frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{\Delta t} + \varepsilon(\boldsymbol{y}_n, \Delta t),$$

where $\varepsilon(\boldsymbol{y}_n, \Delta t)$ measures the truncation error. Assume computed values satisfy $\boldsymbol{y}_{n+1} = Y_{n+1} + \boldsymbol{e}_{n+1}$ with round-off $\boldsymbol{e}_{n+1}$. Then,

$$\frac{d\boldsymbol{y}}{dt} = \frac{Y_{n+1} - Y_n}{\Delta t} + E_n(\boldsymbol{y}_n, \Delta t),$$

with a combined error (round-off + truncation)

$$E_n = \frac{\boldsymbol{e}_{n+1} - \boldsymbol{e}_n}{\Delta t} + \varepsilon(\boldsymbol{y}_n, \Delta t) = \frac{\boldsymbol{e}_{n+1} - \boldsymbol{e}_n}{\Delta t} - \frac{\Delta t}{2}\frac{d^2\boldsymbol{y}(c)}{dt^2}.$$

Assuming $|\boldsymbol{e}_{n+1}| \le e_r$, $|-\boldsymbol{e}_n| \le e_r$, and

$$M = \max_{c \in [t_n, t_{n+1}]} \left|\frac{d^2\boldsymbol{y}(c)}{dt^2}\right|,$$

we bound

$$|E_n| \le \frac{2e_r}{\Delta t} + \frac{\Delta t}{2} M.$$

Minimizing w.r.t. Δt,

$$\frac{\partial |E_n|}{\partial(\Delta t)} = -\frac{2e_r}{\Delta t^2} + \frac{M}{2} = 0 \quad \Longrightarrow \quad \Delta t = \left(\frac{4e_r}{M}\right)^{1/2}.$$

Hence, the smallest Δt is not necessarily best. An optimal step balances round-off and truncation.

Author's Contact Information: Diwen Xu, rwbyaloupeep@gmail.com, University of Washington, Seattle, Washington, USA.

1.2.3 *Stability.* Consider

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0.$$

Forward Euler gives

$$y_{n+1} = y_n + \Delta t\, \lambda y_n = (1 + \lambda\Delta t)\, y_n,$$

so after N steps

$$y_N = (1 + \lambda\Delta t)^N y_0.$$

With round-off e in the initial value, the propagated error is

$$E = (1 + \lambda\Delta t)^N e.$$

For $\lambda > 0$, the solution $y_N \longrightarrow \infty$. So although the error also grows, it may not be significant in comparison to the size of the numerical solution. For $\lambda < 0$, stability requires $|1 + \lambda\Delta t| < 1 \Rightarrow \Delta t < -2/\lambda$. A one-step linear recursion $y_{n+1} = Ay_n$ satisfies after N steps

$$y_N = A^N y_0 = S\Lambda^N S^{-1} y_0,$$

so stability is governed by eigenvalues $\{\lambda_i(A)\}$. Instability arises if $|\lambda_i| > 1$ for any i.

AB2. $\rho(\xi) - z\,\sigma(\xi) = 0, \quad \rho(\xi) = \xi^2 - \xi, \quad \sigma(\xi) = \frac{3}{2}\xi - \frac{1}{2}$. The absolute stability region is the set of $z \in \mathbb{C}$ for which *all* roots ξ of $\rho(\xi) - z\sigma(\xi) = 0$ satisfy $|\xi| \le 1$ and any unit-modulus root is simple. Its boundary is given by the parametric curve

$$z(\theta) = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})} = \frac{e^{2i\theta} - e^{i\theta}}{\frac{3}{2}e^{i\theta} - \frac{1}{2}}, \quad \theta \in [0, 2\pi].$$

For a general linear multistep method (LMM)

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j},$$

Consistency.

$$\sum_{j=0}^{k} \alpha_j = 0, \ \ (y(t) = 1) \qquad \sum_{j=0}^{k} j\alpha_j = \sum_{j=0}^{k} \beta_j. \ \ (y(t) = t)$$

This ensures the local truncation error $O(h^{p+1})$.

Zero-stability. Let $\rho(\xi) = \sum_{j=0}^{k} \alpha_j \xi^j$ be the characteristic polynomial. The method is zero-stable if all roots satisfy $|\xi| \le 1$, and any root with $|\xi| = 1$ is simple. This guarantees that perturbations do not grow faster than $O(1)$ as $n \longrightarrow \infty$.

Convergence theorem (Dahlquist). A linear multistep method is convergent of order p iff it is consistent of order p and zero-stable.

1.3 Advanced Time-Stepping Algorithms

1.3.1 *Adaptive Time-Stepping Algorithm.*

(1) Start with a default step Δt_0.
(2) Advance one step of size Δt to get $f_1(t + \Delta t)$.
(3) Halve the step to $\Delta t/2$ and take two substeps to get $f_2(t + \Delta t)$.
(4) Compare $E = \|f_1 - f_2\|$.
(5) If $E >$ tolerance, halve again until tolerance is met.
(6) If $E <$ tolerance, try doubling to $2\Delta t$, until the condition fails.

1.3.2 *Exponential Time-Steppers.* Consider

$$\frac{d\mathrm{y}}{dt} = c\,\mathrm{y} + F(\mathrm{y}, t), \quad |c| \gg 1.$$

$$\frac{d}{dt}\big(ye^{-ct}\big) = F(y,t)e^{-ct}.$$

$$y(t + \Delta t) = y(t)e^{c\Delta t} + e^{c\Delta t} \int_0^{\Delta t} F\big(y(t+\tau), t+\tau\big)e^{-c\tau}\, d\tau.$$

(i) First-Order. Assume $F(y(t+\tau), t+\tau) \approx F(y(t), t) = F_n$. Then,

$$y_{n+1} = y_n e^{c\Delta t} + F_n \frac{e^{c\Delta t} - 1}{c}.$$

(ii) Two-Step.

$$F \;\approx\; F_n + \tau \frac{F_n - F_{n-1}}{\Delta t} + O(\Delta t^2),$$

$$y_{n+1} = y_n e^{c\Delta t} + F_n \frac{(1 + c\Delta t)e^{c\Delta t} - 1 - 2c\Delta t}{c^2\Delta t} + F_{n-1} \frac{1 + c\Delta t - e^{c\Delta t}}{c^2\Delta t}.$$

As $c \longrightarrow 0$, it reduces to the second-order Adams–Bashforth scheme.

(iii) Exponential RK4 (Cox–Matthews). Define coefficient functions

$$A(c\Delta t) = \frac{-4 - c\Delta t + e^{c\Delta t}\big(4 - 3c\Delta t + (c\Delta t)^2\big)}{c^3\Delta t^2},$$

$$B(c\Delta t) = \frac{2\left(2 + c\Delta t + e^{c\Delta t}(-2 + c\Delta t)\right)}{c^3\Delta t^2},$$

$$C(c\Delta t) = \frac{-4 - 3c\Delta t - (c\Delta t)^2 + e^{c\Delta t}(4 - c\Delta t)}{c^3\Delta t^2}.$$

$$y_{n+1} = e^{c\Delta t}y_n + AF(y_n, t_n) + B\Big[F(a_n, t_n + \tfrac{\Delta t}{2}) + F(b_n, t_n + \tfrac{\Delta t}{2})\Big] + CF(c_n, t_n + \Delta t),$$

with the stages

$$a_n = y_n e^{c\Delta t/2} + \frac{e^{c\Delta t/2} - 1}{c} F(y_n, t_n),$$

$$b_n = y_n e^{c\Delta t/2} + \frac{e^{c\Delta t/2} - 1}{c} F(a_n, t_n + \tfrac{\Delta t}{2}),$$

$$c_n = a_n e^{c\Delta t/2} + \frac{e^{c\Delta t/2} - 1}{c} \Big(2F(b_n, t_n + \tfrac{\Delta t}{2}) - F(y_n, t_n)\Big).$$

1.4 Boundary Value Problems

1.4.1 *The Shooting Method.* Consider the second–order boundary value problem

$$\frac{d^2y}{dt^2} = f\left(t,\ y,\ \frac{dy}{dt}\right), \quad t \in [a, b],$$

with general boundary conditions

$$\alpha_1\, y(a) + \beta_1\, y'(a) = \gamma_1,$$
$$\alpha_2\, y(b) + \beta_2\, y'(b) = \gamma_2.$$

(1) Choose a time–stepping scheme for IVP and a value A. Set $y(a) = \alpha$, $y'(a) = A$, and integrate to $t = b$ to obtain $y_A(b)$.
(2) Form the residual $r(A) = y_A(b) - \beta$.
(3) Update A using a one–dimensional root–finder until $|r(A)|$ is below the desired tolerance.
(4) The converged forward trajectory is the numerical BVP solution. Its accuracy reflects both the root–finder tolerance and the underlying time discretization.

In practice, bracketing methods like bisection are robust when two initial guesses produce undershoot/overshoot at $t = b$, whereas secant/Newton methods converge faster when a good initial guess.

Sturm–Liouville Theory. Let $(a, b) \subset \mathbb{R}$. A regular Sturm–Liouville problem seeks nontrivial solutions y and parameters $\lambda \in \mathbb{R}$ to

$$\mathcal{L}y \equiv -\frac{d}{dx}\Big(p(x)\, y'(x)\Big) + q(x)\, y(x) \;=\; \lambda\, w(x)\, y(x), \quad x \in (a, b)$$

subject to separated boundary conditions

$$\alpha_1 y(a) + \alpha_2 p(a) y'(a) = 0, \quad \beta_1 y(b) + \beta_2 p(b) y'(b) = 0,$$

where

$$p \in C^1[a, b], \quad q, w \in C[a, b], \quad p(x) > 0,\; w(x) > 0 \text{ on } [a, b].$$

Here p is the *diffusion* coefficient, q the *potential*, and w the *weight*. *Spectral properties*

(1) All eigenvalues are real and ordered $\lambda_1 < \lambda_2 < \cdots \to +\infty$.
(2) Each eigenvalue is *simple*, i.e. any eigenfunction is unique up to a scalar multiple.
(3) Eigenfunctions corresponding to distinct eigenvalues are orthogonal in $L^2_w(a, b)$, i.e. $\langle y_m, y_n\rangle_w = 0$ $(m \neq n)$.

Eigenvalues and Eigenfunctions on an Infinite Domain. Consider

$$\psi_n'' + \big(n(x) - \beta_n\big)\,\psi_n = 0, \quad x \in \mathbb{R},$$

with decay conditions $\psi_n(x) \to 0$ as $x \to \pm\infty$. Let

$$n(x) = n_0 \begin{cases} 1 - |x|^2, & 0 \le |x| \le 1, \\ 0, & |x| > 1, \end{cases}$$

where $n_0 > 0$ is a constant. When $\beta_n \ge 0$,

$$x = +L.\; \psi_n'(L) + \sqrt{\beta_n}\,\psi_n(L) = 0,$$
$$x = -L.\; \psi_n'(-L) - \sqrt{\beta_n}\,\psi_n(-L) = 0,$$

on a truncated computational interval $[-L, L]$ with $L \gg 1$. So,

$$\frac{d}{dx}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \beta_n - n(x) & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

with boundary conditions

$$x_2(L) = -\sqrt{\beta_n}\, x_1(L), \quad x_2(-L) = \;\sqrt{\beta_n}\, x_1(-L).$$

- an outer `FOR` loop over the number of eigenvalues to find,
- an inner `FOR` loop that iterates on β_n via bracketing/bisection until the boundary condition residual satisfies a tolerance,
- an `IF` block that checks convergence of the boundary residual and decides whether to accept the current β_n,
- an additional loop may be used for amplitude normalization so that eigenfunctions have unit norm.

1.4.2 *The Relaxation Method.*

$$\frac{d^2y}{dt^2} \;=\; p(t)\,\frac{dy}{dt} + q(t)\, y + r(t), \quad t \in [a, b],$$

with Dirichlet conditions $y(a) = \alpha,\; y(b) = \beta$.

$$\frac{y(t + \Delta t) - 2y(t) + y(t - \Delta t)}{\Delta t^2} \;=\; p(t)\,\frac{y(t + \Delta t) - y(t - \Delta t)}{2\Delta t} \;+\; \cdots$$

$$\Big[1 - \tfrac{\Delta t}{2}\, p(t)\Big]\, y_{n+1} - \Big[2 + \Delta t^2 q(t)\Big]\, y_n + \Big[1 + \tfrac{\Delta t}{2}\, p(t)\Big]\, y_{n-1} = \Delta t^2\, r(t).$$

Imposing $y(t_0) = y(a) = \alpha$ and $y(t_N) = y(b) = \beta$, the unknowns are the interior values $\mathbf{x} = [\, y(t_1), \ldots, y(t_{N-1})\,]^\mathsf{T}$. The resulting linear system $A\mathbf{x} = \mathbf{b}$ has tridiagonal coefficient matrix

$$A = \begin{bmatrix} 2 + \Delta t^2 q(t_1) & -1 + \frac{\Delta t}{2} p(t_1) & & \\ -1 - \frac{\Delta t}{2} p(t_2) & 2 + \Delta t^2 q(t_2) & -1 + \frac{\Delta t}{2} p(t_2) & \\ & \ddots & \ddots & \ddots \end{bmatrix},$$

and right-hand side

$$\mathbf{b} = \begin{bmatrix} -\Delta t^2 r(t_1) + \big(1 + \frac{\Delta t}{2} p(t_1)\big)\,\alpha \\ -\Delta t^2 r(t_2) \\ \vdots \\ -\Delta t^2 r(t_{N-2}) \\ -\Delta t^2 r(t_{N-1}) + \big(1 - \frac{\Delta t}{2} p(t_{N-1})\big)\,\beta \end{bmatrix}.$$

For the general nonlinear case,

$$\frac{y(t + \Delta t) - 2y(t) + y(t - \Delta t)}{\Delta t^2} = f\left(t, y(t), \frac{y(t + \Delta t) - y(t - \Delta t)}{2\Delta t}\right).$$

$$2y_1 - y_2 - \alpha + \Delta t^2\, f\Big(t_1,\, y_1,\, \tfrac{y_2 - \alpha}{2\Delta t}\Big) = 0,$$

$$-y_{j-1} + 2y_j - y_{j+1} + \Delta t^2\, f\Big(t_j,\, y_j,\, \tfrac{y_{j+1} - y_{j-1}}{2\Delta t}\Big) = 0, \quad j = 2, \ldots, N - 2,$$

$$-y_{N-2} + 2y_{N-1} - \beta + \Delta t^2\, f\Big(t_{N-1},\, y_{N-1},\, \tfrac{\beta - y_{N-2}}{2\Delta t}\Big) = 0.$$

Such systems can be challenging. Existence/uniqueness is not guaranteed, and robust solution typically requires a relaxation method such as Newton's method or a secant method tailored to the discrete residuals.

1.4.3 *The Collocation Method.*

Formulation. We seek an approximate solution $y_N(x)$ in the form

$$y_N(x) = \sum_{j=1}^{N} c_j \phi_j(x),$$

where $\{\phi_j(x)\}$ are chosen basis functions satisfying the boundary conditions, and $\{c_j\}$ are unknown coefficients.

Collocation Points. We select $N - 2$ distinct points $x_1, \ldots, x_{N-2} \in (a, b)$, called collocation points, and impose that the approximate solution satisfies the differential equation at these points.

$$y_N''(x_i) = f\big(x_i, y_N(x_i), y_N'(x_i)\big), \quad i = 1, 2, \ldots, N - 2.$$

Together with the boundary conditions at a and b, we obtain a system of N equations for the N unknown coefficients c_j.

Choice of Basis and Points. Polynomial basis functions, such as Lagrange or Hermite polynomials. Collocation points chosen as evenly spaced or as special quadrature nodes, such as Gauss–Lobatto or Chebyshev points. These choices affect both accuracy and numerical stability.

Resulting System. Substituting $y_N(x)$ and its derivatives into the differential equation leads to a nonlinear algebraic system for $\{c_j\}$. If the problem is linear, this system is linear in $\{c_j\}$. Otherwise, it may be solved iteratively such as Newton's method.

1.4.4 *Linear Operators and Computing Spectra.*

$$\frac{\partial u}{\partial t} = \mathcal{N}\left(x,\, u,\, \frac{\partial u}{\partial x},\, \frac{\partial^2 u}{\partial x^2},\, \dots\right),$$

where $\mathcal{N}$ may have nonconstant coefficients in x and is in general nonlinear in $u(x,t)$ and its derivatives. We assume solutions exist by Cauchy–Kovalevskaya, with boundary conditions on either a finite or infinite domain. We focus on *equilibrium* (steady) solutions where $\partial u/\partial t = 0$. Thus there exists a time-independent $U(x)$ satisfying

$$\mathcal{N}\left(x,\, U,\, \frac{\partial U}{\partial x},\, \frac{\partial^2 U}{\partial x^2},\, \dots\right) = 0.$$

To form the linear stability problem, linearize about U.

$$u(x,t) \;=\; U(x) + \varepsilon\, v(x,t), \qquad 0 < \varepsilon \ll 1.$$

If $v(x,t) \longrightarrow \infty$ as $t \longrightarrow \infty$ the equilibrium is unstable. If $v(x,t) \longrightarrow 0$ it is asymptotically stable. And if $v(x,t) = O(1)$ it is Lyapunov stable.

$$\frac{\partial v}{\partial t} \;=\; \mathcal{L}[U(x)]\, v \;+\; O(\varepsilon),$$

where $\mathcal{L}[U]$ is the linearized operator about U. Seeking modal solutions $v(x,t) = w(x)e^{\lambda t}$ gives the spectral / eigenvalue problem

$$\mathcal{L}[U(x)]\, w \;=\; \lambda\, w.$$

Stability is determined by the spectrum. The equilibrium is unstable if any $\Re\{\lambda\} > 0$. It is stable if $\Re\{\lambda\} \le 0$, asymptotic stable if < 0.

First derivative, Dirichlet $v(a) = v(b) = 0$. Unknowns are $v(x_j)$ for $j = 1, \dots, N$. The centered $O(\Delta x^2)$ first-derivative matrix is

$$\left.\frac{\partial}{\partial x}\right|_{\text{Dirichlet}} \Rightarrow A_1 \;=\; \frac{1}{2\Delta x}\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix}.$$

First derivative, Neumann $v'(a) = v'(b) = 0$. Using one-sided second-order formulas at the boundaries, the ghost values are

$$v_0 = \tfrac{4}{3}v_1 - \tfrac{1}{3}v_2, \quad v_{N+1} = \tfrac{4}{3}v_N - \tfrac{1}{3}v_{N-1},$$

$$\left.\frac{dv}{dx}\right|_{x_1} = \frac{v_2 - v_0}{2\Delta x} = \frac{\frac{4}{3}(v_2 - v_1)}{2\Delta x}.$$

$$\left.\frac{\partial}{\partial x}\right|_{v'(a)=v'(b)=0} \Rightarrow A_2 \;=\; \frac{1}{2\Delta x}\begin{bmatrix} -\frac{4}{3} & \frac{4}{3} & 0 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -\frac{4}{3} & \frac{4}{3} \end{bmatrix}.$$

First derivative, periodic $v(a) = v(b)$. Unknowns are $v_0, \dots, v_N$ with $v_0 = v_{N+1}$. The centered matrix is

$$\left.\frac{\partial}{\partial x}\right|_{\text{periodic}} \Rightarrow A_3 \;=\; \frac{1}{2\Delta x}\begin{bmatrix} 0 & 1 & 0 & \cdots & -1 \\ -1 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 0 & -1 & 0 \end{bmatrix}.$$

Second derivative, Dirichlet $v(a) = v(b) = 0$.

$$\left.\frac{\partial^2}{\partial x^2}\right|_{\text{Dirichlet}} \Rightarrow B_1 \;=\; \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}.$$

Second derivative, Neumann $v'(a) = v'(b) = 0$.

$$\left.\frac{\partial^2}{\partial x^2}\right|_{v'(a)=v'(b)=0} \Rightarrow B_2 \;=\; \frac{1}{\Delta x^2}\begin{bmatrix} -\frac{2}{3} & \frac{2}{3} & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & \frac{2}{3} & -\frac{2}{3} \end{bmatrix}.$$

Second derivative, periodic $v(a) = v(b)$.

$$\left.\frac{\partial^2}{\partial x^2}\right|_{\text{periodic}} \Rightarrow B_3 \;=\; \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 1 \\ 1 & -2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & 0 & 1 & -2 \end{bmatrix}.$$

1.5 Neural Networks for Time Stepping

Regard a time–stepper as a *flow map* that takes the state at time t directly to the state at time $t + \Delta t$. Neural networks can be trained to learn such a flow map from data. Let $f_\theta(\cdot)$ denote a neural network with trainable parameters θ. Given state trajectories, one can train f_θ so that it approximates the one–step map

$$Y \;=\; f_\theta(X) \quad\Longrightarrow\quad y_{n+1} \;=\; f_\theta(y_n),$$

by minimizing a time–stepping error over pairs of consecutive states (y_n, y_{n+1}). In effect, the network learns the action of the right–hand side of the ODE through its induced discrete flow.

2 Finite Difference Schemes for PDEs

2.1 Fast Poisson Solvers: The Fourier Transform

Consider solving the streamfunction equation

$$\nabla^2 \psi = \omega.$$

A direct discretize-then-solve approach typically costs $O(N^2)$ operations at best. Iterative schemes may outperform this in practice, but without guarantees. An alternative is the fast Fourier transform (FFT), which leverages periodic transforms on a finite interval $x \in [-L, L]$ to obtain near-linear operation counts.
On the infinite line, the Fourier transform and its inverse are defined by

$$\mathcal{F}\{f\}(k) \;=\; \hat{f}(k) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-ikx} f(x)\, dx,$$
$$f(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{ikx}\, \hat{f}(k)\, dk.$$

In computation we restrict to $x \in [-L, L]$ and assume periodic boundary conditions because the kernel $e^{\pm ikx}$ is oscillatory. A continuous eigenfunction expansion in k becomes a discrete sum on the finite domain.

The key property enabling FFT-based PDE solvers is the transform rule for derivatives. Assuming $f(x) \to 0$ as $x \to \pm\infty$,

$$\mathcal{F}\{f^{(n)}(x)\}(k) = (ik)^n \hat{f}(k).$$

This diagonalizes constant-coefficient differential operators in Fourier space and is central to efficiency and simplicity of spectral methods. The Cooley–Tukey FFT (mid-1960s) reduces the cost of discrete Fourier transforms to $O(N \log N)$. In two dimensions,

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} \ = \ \omega.$$

Denote the Fourier transform in x by a hat ($\hat{}$) and in y by a tilde ($\tilde{}$). Transforming in x gives

$$-k_x^2 \hat{\psi} \ + \ \frac{\partial^2 \hat{\psi}}{\partial y^2} \ = \ \hat{\omega},$$

and transforming subsequently in y yields

$$-k_x^2 \tilde{\hat{\psi}} \ - \ k_y^2 \tilde{\hat{\psi}} \ = \ \tilde{\hat{\omega}}.$$

Hence,

$$\tilde{\hat{\psi}}(k_x, k_y) \ = \ -\frac{\tilde{\hat{\omega}}(k_x, k_y)}{k_x^2 + k_y^2}.$$

The solution $\psi(x, y)$ is obtained by inverse transforming in y and x.

2.2 Fast Fourier Transform: FFT, IFFT, FFTSHIFT, IFFTSHIFT

Given a function which has been discretized with 2^n points and represented by a vector x, the FFT is found with the command `fft(x)`. Aside from transforming the function, the algorithm associated with the FFT does three major things. It shifts the data so that $x \in [0, L] \to [-L, 0]$ and $x \in [-L, 0] \to [0, L]$, additionally it multiplies every other mode by -1, and it assumes you are working on a 2π-periodic domain.

By using the command `fftshift`, we can shift the transformed function back to its mathematically correct positions. However, before inverting the transformation, it is crucial that the transform is shifted back. The command `ifftshift` does this. In general, unless you need to plot the spectrum, it is better not to deal with the `fftshift` and `ifftshift` commands.

For transforming in higher dimensions, a couple of choices in `python` are possible. For 2D transformations, it is recommended to use the commands `fft2` and `ifft2`. These will transform a matrix A, which represents data in the x- and y-directions, respectively, along the rows and then columns. For higher dimensions, the `fft` command can be modified to `fft(x,[],N)` where N is the number of dimensions.

2.3 Sparse Matrices: SPDIAGS, SPY

Under discretization, most physical problems yield sparse matrices, i.e. matrices which are largely composed of zeros. The `spdiags` command allows for the construction of sparse matrices in a relatively simple fashion. The sparse matrix is then saved using a minimal amount of memory and all matrix operations are conducted as usual. The `spy` command allows you to look at the nonzero components of the matrix structure.

2.4 Iterative Methods: BICGSTAB, GMRES

The bi-conjugate stabilized gradient method (`bicgstab`) and the generalized minimum residual method (`gmres`). Both are easily implemented in `python`.

```
x, flag, relres, iter = bicgstab(A, b, tol=tol,
                                maxiter=maxit, M1=M1, M2=M2, x0=x0)

x, flag, relres, iter = gmres(A, b, tol=tol, restart=restart,
                                   maxiter=maxit, M=M1, x0=x0)
```

- `tol` = specified tolerance for convergence,
- `maxit` = maximum number of iterations,
- `M1`, `M2` = preconditioning matrices,
- `x0` = initial guess vector,
- `restart` = restart of iterations (`gmres` only).

2.5 Streamfunction Equations: Nonuniqueness

$$\nabla^2 \psi = \omega$$

with the periodic boundary conditions

$$\psi(-L, y, t) = \psi(L, y, t),$$
$$\psi(x, -L, t) = \psi(x, L, t).$$

The solution can only be determined to an arbitrary constant. To overcome this problem numerically, we simply observe that we can arbitrarily add a constant to the solution. Or alternatively, we can pin down the value of the streamfunction at a single location in the computational domain. This constraint fixes the arbitrary constant problem and removes the singularity from the matrix A. Thus to fix the problem, we can simply pick an arbitrary point in our computational domain ψ_{mn} and fix its value.

```
A[0, 0] = 0
```

Then $\det A \neq 0$ and the matrix can be used in any of the linear solution methods. Note that the choice of the matrix component and its value are completely arbitrary.

2.6 Fast Fourier Transforms: Divide by Zero

$$\tilde{\hat{\psi}} = -\frac{\tilde{\hat{\omega}}}{k_x^2 + k_y^2 + \varepsilon},$$

where `eps` is the command for generating a machine precision number which is on the order of $O(10^{-15})$. A second option, which is more highly recommended, is to redefine the k_x and k_y vectors associated with the wavenumbers in the x- and y-directions.

```
kx[0] = 1e-6
ky[0] = 1e-6
```

2.7 Time and Space Stepping Schemes: Method of lines

$$\frac{\partial u}{\partial t} = c\,\frac{\partial u}{\partial x},$$

Forward Euler.

$$u_n^{(m+1)} = u_n^{(m)} + \frac{c\,\Delta t}{2\Delta x}\left(u_{n+1}^{(m)} - u_{n-1}^{(m)}\right).$$

$$u_n^{(m+1)} = u_n^{(m)} + \frac{\lambda}{2}\left(u_{n+1}^{(m)} - u_{n-1}^{(m)}\right).$$

Leap-Frog (2,2).

$$\frac{u_n^{(m+1)} - u_n^{(m-1)}}{2\Delta t} = \frac{c}{2\Delta x}\left(u_{n+1}^{(m)} - u_{n-1}^{(m)}\right).$$

$$u_n^{(m+1)} = u_n^{(m-1)} + \lambda\left(u_{n+1}^{(m)} - u_{n-1}^{(m)}\right).$$

Leap-Frog (2,4).

$$\frac{u_n^{(m+1)} - u_n^{(m-1)}}{2\Delta t} = c\,\frac{-u(x_{n+2}) + 8u(x_{n+1}) - 8u(x_{n-1}) + u(x_{n-2})}{12\,\Delta x}.$$

$$u_n^{(m+1)} = u_n^{(m-1)} + \lambda\left[\frac{4}{3}\left(u_{n+1}^{(m)} - u_{n-1}^{(m)}\right) - \frac{1}{6}\left(u_{n+2}^{(m)} - u_{n-2}^{(m)}\right)\right].$$

Lax–Wendroff Scheme.

$$u(x, t+\Delta t) = u(x,t) + \Delta t\, u_t(x,t) + \frac{\Delta t^2}{2}\, u_{tt}(x,t) + O(\Delta t^3).$$

$$u_t = c\,u_x, \quad u_{tt} = c^2\,u_{xx}.$$

$$u_n^{(m+1)} = u_n^{(m)} + \frac{\lambda}{2}\left(u_{n+1}^{(m)} - u_{n-1}^{(m)}\right) + \frac{\lambda^2}{2}\left(u_{n+1}^{(m)} - 2u_n^{(m)} + u_{n-1}^{(m)}\right).$$

Backward Euler.

$$u_n^{(m+1)} = u_n^{(m)} + \frac{\lambda}{2}\left(u_{n+1}^{(m+1)} - u_{n-1}^{(m+1)}\right).$$

$$-\frac{\lambda}{2}u_{n+1}^{(m+1)} + u_n^{(m+1)} + \frac{\lambda}{2}u_{n-1}^{(m+1)} = u_n^{(m)},$$

MacCormack Predictor–Corrector Scheme.

$$u_n^{(P)} = u_n^{(m)} + \lambda\left(u_{n+1}^{(m)} - u_{n-1}^{(m)}\right).$$

$$u_n^{(m+1)} = \frac{1}{2}\left[u_n^{(m)} + u_n^{(P)} + \lambda\left(u_{n+1}^{(P)} - u_{n-1}^{(P)}\right)\right].$$

Scheme	Stability in terms of λ
Forward Euler	unstable for all λ
Backward Euler	stable for all λ
Leap-frog (2,2)	stable for $\lambda \le 1$
Leap-frog (2,4)	stable for $\lambda \le 0.707$

2.8 von Neumann Analysis

This assumes the solution is of the form

$$u_n^{(m)} = g^m \exp(i\xi nh), \quad \xi \in \left[-\frac{\pi}{h}, \frac{\pi}{h}\right],$$

where $h = \Delta x$ is the spatial discretization parameter. Essentially this assumes the solution can be constructed of Fourier modes. The key then is to determine what happens to g^m as $m \to \infty$.

$$\lim_{m\to\infty} |g|^m \to \infty \qquad \text{unstable scheme,}$$

$$\lim_{m\to\infty} |g|^m \le 1\ (<\infty) \qquad \text{stable scheme.}$$

- It is a general result that a scheme which is forward in time and centered in space is unstable for the one-way wave equation. This assumes a standard forward discretization, not something like Runge–Kutta.
- von Neumann analysis is rarely enough to guarantee stability, i.e., it is necessary but not sufficient. Nonlinearity usually kills the von Neumann analysis immediately. Thus a large variety of nonlinear partial differential equations are beyond the scope of a von Neumann analysis.
- Accuracy versus stability: it is always better to worry about accuracy. An unstable scheme will quickly become apparent by causing the solution to blow up to infinity, whereas an inaccurate scheme will simply give you a wrong result without indicating a problem.

2.9 Hyper-Diffusion

Consider the fourth-order diffusion equation

$$\frac{\partial u}{\partial t} = -c\,\frac{\partial^4 u}{\partial x^4}.$$

Using forward Euler and central differences yields

$$u_n^{(m+1)} = u_n^{(m)} - \lambda\left(u_{n+2}^{(m)} - 4u_{n+1}^{(m)} + 6u_n^{(m)} - 4u_{n-1}^{(m)} + u_{n-2}^{(m)}\right),$$

with CFL number

$$\lambda = \frac{c\Delta t}{\Delta x^4}.$$

To double accuracy, one must set $\Delta t \to \Delta t/16$ and double the number of spatial points, increasing runtime by a factor of 32. This severe restriction is characteristic of numerical stiffness.

Stiffness does not arise from central differencing itself, but from the physics of higher-order derivatives.

$$\frac{\partial \hat{u}}{\partial t} = -c(ik)^4\hat{u} = -ck^4\hat{u},$$

and large wavenumbers k produce very large k^4 terms, severely restricting timestep sizes in any explicit scheme.

Key strategies for handling stiffness include

- using adaptive time-stepping,
- employing implicit schemes,
- choosing solvers specifically designed for stiff problems.

2.10 Operator Splitting Techniques

- **Wave behavior:** $\dfrac{\partial u}{\partial t} = \dfrac{\partial u}{\partial x}$
  - forward Euler: unstable for all λ
  - leap-frog (2,2): stable for $\lambda \le 1$
- **Diffusion behavior:** $\dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2}$
  - forward Euler: stable for $\lambda \le 1/2$
  - leap-frog (2,2): unstable for all λ

An advection–diffusion equation such as

$$\frac{\partial \omega}{\partial t} + [\psi, \omega] = \nu\nabla^2\omega$$

contains both advective and diffusive processes. Over a small time interval Δt, the dynamics can be split into

$$\Delta t: \quad \frac{\partial \omega}{\partial t} + [\psi, \omega] = 0 \qquad \text{(advection only),}$$

$$\Delta t: \quad \frac{\partial \omega}{\partial t} = \nu\nabla^2\omega \qquad \text{(diffusion only).}$$

This allows each physical effect to be advanced independently over the same time step Δt using schemes best suited to that component (e.g., leap-frog for advection, Euler for diffusion). The accuracy of operator splitting depends heavily on the time step Δt. For an equation of the form

$$\frac{\partial u}{\partial t} = Lu + N(u).$$

To improve accuracy, Strang splitting is used.

$$\begin{aligned} \frac{\Delta t}{2}: &\quad \frac{\partial u}{\partial t} + Lu = 0, \\ \Delta t: &\quad \frac{\partial u}{\partial t} + N(u) = 0, \\ \frac{\Delta t}{2}: &\quad \frac{\partial u}{\partial t} + Lu = 0. \end{aligned}$$

This symmetrization reduces error and is generally preferred for accuracy.

3 Spectral Methods for PDEs

$$F(k) = \int_{-\infty}^{\infty} e^{-ikx} f(x)\,\mathrm{d}x,$$
$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ikx} F(k)\,\mathrm{d}k.$$
$$\mathcal{F}\{f^{(n)}(x)\}(k) = (ik)^n F(k), \quad \mathcal{F}\{f\}(k) = F(k).$$
$$F(k) = \sum_{n=1}^{N} f(n)\,\exp\left[-i\,\frac{2\pi(k-1)}{N}\,(n-1)\right], \quad 1 \le k \le N,$$
$$f(n) = \frac{1}{N}\sum_{k=1}^{N} F(k)\,\exp\left[i\,\frac{2\pi(k-1)}{N}\,(n-1)\right], \quad 1 \le n \le N.$$

3.1 Fast Fourier transforms: Cooley–Tukey algorithm

$$(F_N)_{jk} = \omega_N^{jk} = \exp\left(-\frac{i2\pi jk}{N}\right).$$

Coefficients of the matrix are points on the unit circle since $|\omega_N^{jk}| = 1$. They are also the basis functions for the Fourier transform.

$$\omega_{2N}^2 = \omega_N.$$

The FFT is a matrix operation

$$\mathbf{y} = F_N \mathbf{x}.$$

Defining

$$\mathbf{x}^e = \begin{pmatrix} x_0 \\ x_2 \\ x_4 \\ \vdots \\ x_{N-2} \end{pmatrix}, \quad \mathbf{x}^o = \begin{pmatrix} x_1 \\ x_3 \\ x_5 \\ \vdots \\ x_{N-1} \end{pmatrix}.$$
$$\mathbf{y}^e = F_M \mathbf{x}^e, \quad \mathbf{y}^o = F_M \mathbf{x}^o.$$

Thus the computation size goes from $O(N^2)$ to

$$\begin{aligned} y_n &= y_n^e + \omega_N^n\, y_n^o, &\quad n &= 0, 1, 2, \ldots, M-1, \\ y_{n+M} &= y_n^e - \omega_N^n\, y_n^o, &\quad n &= 0, 1, 2, \ldots, M-1. \end{aligned}$$

This is where the shift occurs in the FFT routine which maps the domain $x \in [0, L]$ to $[-L, 0]$ and $x \in [-L, 0]$ to $[0, L]$. The command `fftshift` undoes this shift.

3.2 Chebychev Polynomials and Transform

Generally, one can expand in a variety of basis functions chosen to match the geometry and boundary conditions of the problem.

- Bessel functions: radial, two-dimensional problems,
- Legendre polynomials: three-dimensional Laplace problems,
- Hermite–Gauss polynomials: Schrödinger equations with harmonic potentials,
- Spherical harmonics: radial, three-dimensional problems,
- Chebychev polynomials: bounded one-dimensional domains.

Chebychev polynomials are defined as the solutions $T_n(x)$ of the Sturm–Liouville problem

$$\sqrt{1-x^2}\,\frac{d}{dx}\left(\sqrt{1-x^2}\,\frac{dT_n}{dx}\right) + n^2 T_n = 0, \quad x \in [-1, 1].$$

As a self-adjoint Sturm–Liouville operator,

(1) Real eigenvalues: $\lambda_n = n^2$.
(2) Real eigenfunctions: $T_n(x)$.
(3) Orthogonality:

$$\int_{-1}^{1} (1-x^2)^{-1/2}\,T_n(x)\,T_m(x)\,dx = \frac{\pi}{2}\,c_n\,\delta_{nm},$$

where $c_0 = 2$, $c_n = 1$ for $n > 0$.

(4) The set $\{T_n(x)\}_{n=0}^{\infty}$ forms a complete basis.

Each Chebychev polynomial of degree n can be defined via

$$T_n(\cos\theta) = \cos(n\theta).$$

From this, the first few polynomials follow

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ T_3(x) &= 4x^3 - 3x, \\ T_4(x) &= 8x^4 - 8x^2 + 1. \end{aligned}$$

Introduce

$$x = \cos\theta, \quad \theta \in [0, \pi].$$

Given a function $f(x)$ on $[-1, 1]$, define

$$g(\theta) = f(\cos\theta).$$
$$\frac{dg}{d\theta} = -f'(x)\,\sin\theta,$$

so that

$$\frac{dg}{d\theta} = 0 \quad \text{at } \theta = 0, \pi,$$

corresponding to no-flux (Neumann) boundary conditions. This structure allows the use of a discrete cosine transform in θ. A Chebychev expansion of f is written as

$$f(x) = \sum_{k=0}^{\infty} a_k\,T_k(x),$$

with coefficients determined by orthogonality

$$a_k = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}}\, f(x)\, T_k(x)\, dx.$$

These coefficients can be computed in $O(N \log N)$ time via the DCT. Some useful properties of the Chebychev polynomials include

- Three-term recurrence:

$$T_{n+1}(x) = 2x\, T_n(x) - T_{n-1}(x).$$

- Bounds:

$$|T_n(x)| \le 1, \quad |T_n'(x)| \le n^2.$$

- Endpoint values:

$$T_n(\pm 1) = (\pm 1)^n.$$

- Higher derivatives at the endpoints:

$$\frac{d^p}{dx^p} T_n(\pm 1) = (\pm 1)^{n+p} \prod_{k=0}^{p-1} \frac{n^2 - k^2}{2k+1}.$$

- Parity:

$T_n(x)$ is even if n is even, and odd if n is odd.

For n collocation points, the x-grid is

$$x_m = \cos\left(\frac{(2m-1)\pi}{2n} \right), \quad m = 1, 2, \ldots, n.$$

Thus the points are uniformly spaced in θ but cluster near the endpoints $x = \pm 1$. As the resolution n increases, the spatial resolution increases more strongly near the boundaries than in the interior. Let L be a linear operator and define

$$Lf(x) = \sum_{n=0}^{\infty} b_n\, T_n(x),$$

while

$$f(x) = \sum_{n=0}^{\infty} a_n\, T_n(x).$$

First derivative. For $Lf = f'(x)$, one has

$$c_n\, b_n = 2 \sum_{\substack{p=n+1 \\ p+n \text{ odd}}}^{\infty} p\, a_p,$$

where $c_0 = 2$, $c_n = 1$ for $n > 0$, and $c_n = 0$ for $n < 0$.

Multiplication by x. For $Lf = xf(x)$,

$$b_n = \frac{1}{2}\left(c_{n-1} a_{n-1} + a_{n+1} \right).$$

Multiplication by x^2. For $Lf = x^2 f(x)$,

$$b_n = \frac{1}{4}\left(c_{n-2} a_{n-2} + (c_n + c_{n-1}) a_n + a_{n+2} \right).$$

Here $c_0 = 2$, $c_n = 0$ for $n < 0$, and $c_n = 1$ for $n > 0$.

3.3 Spectral Method Implementation

$$\frac{\partial u}{\partial t} = Lu + N(u), \quad L = a\,\frac{d^2}{dx^2} + b\,\frac{d}{dx} + c.$$

$$\frac{d\widehat{u}}{dt} = \alpha(k)\, \widehat{u} + \widehat{N(u)}.$$

$$a(ik)^2 \widehat{u} + b(ik)\widehat{u} + c\,\widehat{u} = \big(-ak^2 + ibk + c\big)\widehat{u} = \alpha(k)\, \widehat{u}.$$

The nonlinear terms are a bit more difficult to handle,

$$\widehat{f(x)\, u_x} = \mathcal{F}\big(f(x)\, u_x\big), \;\; u_x = \mathcal{F}^{-1}\big(ik\, \widehat{u}\big).$$

$$\widehat{u^3 u_{xx}} = \mathcal{F}\big(u^3 u_{xx}\big), \;\; u_{xx} = \mathcal{F}^{-1}\big(-k^2 \widehat{u}\big).$$

3.4 Pseudo-Spectral Techniques with Filtering

$$\frac{d\hat{u}}{dt} - \alpha(k)\, \hat{u} = \widehat{N(u)}.$$

$$\frac{d}{dt}\Big[\hat{u}\, \exp\big(-\alpha(k)t\big) \Big] = \exp\big(-\alpha(k)t\big)\, \widehat{N(u)}.$$

By defining

$$\hat{v} = \hat{u}\, \exp\big(-\alpha(k)t\big),$$

the system of equations reduces to

$$\frac{d\hat{v}}{dt} = \exp\big(-\alpha(k)t\big)\, \widehat{N(u)},$$
$$\hat{u} = \hat{v}\, \exp\big(\alpha(k)t\big).$$

Thus, the linear, constant-coefficient terms are solved for explicitly, and the numerical stiffness associated with the Lu term is eliminated.

A. Accuracy.

- **Finite Differences:** Accuracy is determined by the Δx and Δy chosen in the discretization. Accuracy is fairly easy to compute and generally much worse than spectral methods.
- **Spectral Method:** Spectral methods rely on a global expansion and are often called spectrally accurate. In particular, spectral methods have infinite-order accuracy.

B. Implementation.

- **Finite Differences:** The greatest difficulty is generating the correct sparse matrices. Further, when solving the resulting system $Ax = b$, it should always be checked whether $\det A = 0$. If $\texttt{cond(A)} > 10^{15}$, then $\det A \approx 0$ and steps must be taken to solve the problem correctly.
- **Spectral Method:** The difficulty with using FFTs is the continual switching between the time or space domain and the spectral domain.

C. Computational Efficiency.

- **Finite Differences:** The computational time for finite differences is determined by the size of the matrices and vectors in solving $Ax = b$. Generally speaking, you can guarantee $O(N^2)$ efficiency by using LU decomposition.
- **Spectral Method:** The FFT algorithm is an $O(N \log N)$ operation. Thus, it is almost always guaranteed to be faster.

D. Boundary Conditions.

- **Finite Differences:** Finite differences are clearly superior when considering boundary conditions. Implementing the generic boundary conditions

$$\alpha\, u(L) + \beta\, \frac{du(L)}{dx} = \gamma$$

is easily done in the finite difference framework. Also, more complicated computational domains may be considered. Generally, any computational domain which can be constructed of rectangles is easily handled by finite difference methods.
- **Spectral Method:** Boundary conditions are the critical limitation on using the FFT method. Specifically, only periodic boundary conditions can be considered. The use of the discrete sine or cosine transform allows for the consideration of pinned or no-flux boundary conditions, but only odd or even solutions are admitted respectively.

3.5 Boundary Conditions and the Chebychev Transform

Consider methods for handling nonperiodic boundary conditions.

Method 1: Periodic extension with FFTs. We can periodically extend a general function $f(x)$ in order to make the function itself periodic. However, the periodic extension will in general generate discontinuities in the periodically extended function. The discontinuities give rise to Gibbs' phenomenon, strong oscillations and errors which are accumulated at the jump locations. This greatly affects the accuracy.

Method 2: Polynomial approximation with equi-spaced points. We can consider polynomial approximation. We discretize the given function $f(x)$ with $N+1$ equally spaced points and fit an Nth degree polynomial through them. This amounts to

$$f(x) \approx a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_N x^N,$$

where the coefficients a_n are determined by an $(N+1) \times (N+1)$ system of equations. However, in this case the Runge phenomenon (polynomial oscillations) generally occurs. A polynomial of degree N typically has $N-1$ combined maxima and minima, and attempting to fit a global high-order polynomial on an equi-spaced grid leads to large oscillations near the boundaries.

Method 3: Polynomial approximation with clustered points. Constructs a polynomial approximation on a *clustered* grid rather than an equally spaced grid. The clustered grid is chosen to be the Chebychev points, which may be viewed as a projection of equally spaced points on the unit circle down to the interval $[-1, 1]$.

$$x_n = \cos\left(\frac{n\pi}{N}\right), \quad n = 0, 1, 2, \dots, N.$$

This clustering of grid points toward the endpoints helps to greatly reduce the effects of the Runge phenomenon. Let p be a unique polynomial of degree $\leq N$ with

$$p(x_n) = V_n, \quad 0 \leq n \leq N,$$

where $V(x)$ is the function we approximate and $V_n = V(x_n)$.

$$w_n = p'(x_n).$$

$$w = D_N v,$$

where D_N represents the action of differentiation, $v = (V_0, \dots, V_N)^T$ and $w = (w_0, \dots, w_N)^T$. By using polynomial interpolation of the Lagrange form, one can construct the matrix elements of $p(x)$ along with the $(N+1) \times (N+1)$ differentiation matrix D_N.

$$\begin{aligned}
(D_N)_{00} &= \frac{2N^2+1}{6}, \\
(D_N)_{NN} &= -\frac{2N^2+1}{6}, \\
(D_N)_{jj} &= -\frac{x_j}{2(1-x_j^2)}, \quad j = 1, 2, \dots, N-1, \\
(D_N)_{ij} &= \frac{c_i(-1)^{i+j}}{c_j(x_i - x_j)}, \quad i, j = 0, 1, \dots, N,\ i \neq j,
\end{aligned}$$

where the parameter $c_j = 2$ for $j = 0$ or $j = N$, and $c_j = 1$ otherwise. In general, D_N^m gives the mth derivative. After transforming via

$$x_n = \cos\left(\frac{n\pi}{N}\right),$$

the discrete Fourier transform (DFT) can be used for real data, while for complex data the regular FFT is used.

3.6 Computing Spectra: The Floquet–Fourier–Hill Method

We consider the eigenvalue problem

$$Lv = \lambda v,$$

where L is a linear differential operator of the form

$$L = \sum_{k=0}^{M} f_k(x)\, \partial_x^k = f_0(x) + f_1(x)\, \partial_x + \cdots + f_M(x)\, \partial_x^M,$$

and the coefficients $f_k(x)$ are assumed to be periodic with period L.

$$f_k(x+L) = f_k(x), \quad k = 0, \dots, M.$$

Thus the method is designed primarily for periodic problems. For problems posed on the real line with solutions s.t. $v(\pm\infty) \to 0$, one can approximate the problem on a large but finite interval.

$$f_k(x) = \sum_{j=-\infty}^{\infty} \hat{f}_{k,j}\, \exp\left(i\frac{2\pi j}{L}x\right), \quad k = 0, \dots, M,$$

$$\hat{f}_{k,j} = \frac{1}{L}\int_{-L/2}^{L/2} f_k(x)\, \exp\left(-i\frac{2\pi j}{L}x\right) dx, \quad k = 0, \dots, M.$$

Floquet theory states that every bounded solution can be written as

$$w(x) = \exp(i\mu x)\, \phi(x),$$

where $\phi(x)$ is periodic with the same period L as the coefficients,

$$\phi(x+L) = \phi(x),$$

and μ can be taken in the interval $\mu \in [0, 2\pi/L)$. The factor $\exp(i\mu x)$ is called the *Floquet multiplier*, while $i\mu$ is the Floquet exponent.

$$\phi(x) = \sum_{j=-\infty}^{\infty} \hat{\phi}_j \exp\left(i\frac{2\pi j}{L}x\right),$$

$$w(x) = \exp(i\mu x) \sum_{j=-\infty}^{\infty} \hat{\phi}_j \exp\left(i\frac{2\pi j}{L}x\right) = \sum_{j=-\infty}^{\infty} \hat{\phi}_j \exp\left(ix\left[\mu + \tfrac{2\pi j}{L}\right]\right),$$

$$\hat{\phi}_j = \frac{1}{L}\int_{-L/2}^{L/2} \phi(x)\, \exp\left(-i\frac{2\pi j}{L}x\right) dx.$$

The nth Fourier coefficient of the transformed eigenvalue problem

$$\sum_{m=-\infty}^{\infty} \sum_{k=0}^{M} \hat{f}_{k,n-m} \left(i\left[\mu + \tfrac{2\pi m}{L}\right]\right)^k \hat{\phi}_m = \lambda\, \hat{\phi}_n, \quad n \in \mathbb{Z}.$$

In other words, the original differential eigenvalue problem has been mapped into an eigenvalue problem in Fourier space,

$$\widehat{L}(\mu)\, \hat{\phi} = \lambda\, \hat{\phi},$$

where

$$\hat{\phi} = (\ldots, \hat{\phi}_{-2}, \hat{\phi}_{-1}, \hat{\phi}_0, \hat{\phi}_1, \hat{\phi}_2, \ldots)^T,$$

and the entries of the bi-infinite matrix $\widehat{L}(\mu)$ are

$$\widehat{L}(\mu)_{nm} = \sum_{k=0}^{M} \hat{f}_{k,n-m} \left(i\left[\mu + \tfrac{2\pi m}{L}\right]\right)^k .$$

We truncate the bi-infinite system by retaining Fourier modes $m = -N, \ldots, N$. This yields a finite-dimensional eigenvalue problem

$$\widehat{L}_N(\mu)\, \hat{\phi}_N = \lambda_N\, \hat{\phi}_N,$$

where $\widehat{L}_N(\mu)$ is a $(2N+1) \times (2N+1)$ matrix and $\hat{\phi}_N \in \mathbb{C}^{2N+1}$. The computed eigenvalues λ_N provide numerical approximations to the eigenvalues of the original operator L, and standard convergence results for spectral methods can be used to analyze the accuracy of this Floquet–Fourier–Hill discretization.

4 Finite Element Schemes for PDEs

4.1 Finite Element Basis

The primary reason to develop this technique is boundary conditions, both complicated domains and general boundary conditions. The key features of the discretization are as follows.

- The width and height of all triangles should be similar.
- All shapes used to span the computational domain should be approximated by polygons.

Three groups of elements are considered basically.

- *Simplex elements*: linear polynomials are used,
- *Complex elements*: higher-order polynomials are used,
- *Multiplex elements*: rectangles are used instead of triangles.

4.2 Discretizing with Finite Elements and Boundaries

Considering the elliptic partial differential equation

$$\frac{\partial}{\partial x}\left(p(x,y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right) + r(x,y)u = f(x,y),$$

where, over part of the boundary,

$$u(x,y) = g(x,y) \quad \text{on } S_1,$$

$$p(x,y)\frac{\partial u}{\partial x}n_x + q(x,y)\frac{\partial u}{\partial y}x_y + g_1(x,y)\,u = g_2(x,y) \quad \text{on } S_2.$$

The method expresses the governing partial differential equation as a functional that is to be minimized.

$$I(\phi) = \iiint_V F\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial z}\right) dV + \iint_S g\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial z}\right) dS,$$

$$\frac{\delta F}{\delta \phi} = \frac{\partial}{\partial x}\left(\frac{\partial F}{\partial \phi_x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial F}{\partial \phi_y}\right) + \frac{\partial}{\partial z}\left(\frac{\partial F}{\partial \phi_z}\right) - \frac{\partial F}{\partial \phi} = 0.$$

$$I(u) = \frac{1}{2}\iint_D \left[p(x,y)\left(\frac{\partial u}{\partial x}\right)^2 + q(x,y)\left(\frac{\partial u}{\partial y}\right)^2 - r(x,y)u^2 \right.$$
$$\left. +2f(x,y)u\right] dx\, dy + \int_{S_2} \left[-g_2(x,y)\,u + \frac{1}{2}g_1(x,y)u^2\right] dS.$$

$$\frac{\delta I_D}{\delta u} = \frac{\partial}{\partial x}\left(p(x,y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right) + r(x,y)u - f(x,y) = 0.$$

$$I(u) = \frac{1}{2}\iint_D \left[p\left(\sum_{i=1}^{m}\gamma_i\frac{\partial \phi_i}{\partial x}\right)^2 + q\left(\sum_{i=1}^{m}\gamma_i\frac{\partial \phi_i}{\partial y}\right)^2 - r\left(\sum_{i=1}^{m}\gamma_i\phi_i\right)^2\right.$$
$$\left. + 2f\left(\sum_{i=1}^{m}\gamma_i\phi_i\right)\right] dx\, dy + \int_{S_2}\left[-g_2\sum_{i=1}^{m}\gamma_i\phi_i + \frac{1}{2}g_1\left(\sum_{i=1}^{m}\gamma_i\phi_i\right)^2\right] dS.$$

$$\frac{\partial I}{\partial \gamma_j} = \iint_D \left[p\sum_{i=1}^{m}\gamma_i\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + q\sum_{i=1}^{m}\gamma_i\frac{\partial \phi_i}{\partial y}\frac{\partial \phi_j}{\partial y} - r\sum_{i=1}^{m}\gamma_i\phi_i\phi_j + f\phi_j\right] dx\, dy$$
$$+ \int_{S_2}\left[-g_2\phi_j + g_1\sum_{i=1}^{m}\gamma_i\phi_i\phi_j\right] dS = 0.$$

$$Ax = b,$$

where

$$x = \begin{pmatrix}\gamma_1\\ \gamma_2\\ \vdots\\ \gamma_n\end{pmatrix}, \quad A = (\alpha_{ij}), \quad b = \begin{pmatrix}\beta_1\\ \beta_2\\ \vdots\\ \beta_n\end{pmatrix},$$

with

$$\beta_i = -\iint_D f\phi_i\, dx\, dy + \int_{S_2} g_2\phi_i\, dS - \sum_{k=n+1}^{m}\alpha_{ik}\gamma_k,$$

$$\alpha_{ij} = \iint_D \left[p\,\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + q\,\frac{\partial \phi_i}{\partial y}\frac{\partial \phi_j}{\partial y} - r\,\phi_i\phi_j\right] dx\, dy + \int_{S_2} g_1\,\phi_i\phi_j\, dS.$$

$$\phi_i = \sum_{j=1}^{3} N_j^{(i)}(x,y)\,\phi_j^{(i)} = \sum_{j=1}^{3}\left(a_j^{(i)} + b_j^{(i)}x + c_j^{(i)}y\right)\phi_j^{(i)}.$$

The full numerical procedure consists of

1. Discretizing the domain into triangular elements.
2. Applying boundary values on triangles that touch boundary.
3. Constructing the shape functions
$$N_j^{(i)} = a_j^{(i)} + b_j^{(i)}x + c_j^{(i)}y.$$
4. Computing the integrals for α_{ij} and β_j.
5. Forming the global matrix A and vector b.
6. Solving $Ax = b$ for the coefficients γ_i.
7. Producing the final solution
$$u(x,y) = \sum_{i=1}^{m}\gamma_i\phi_i(x,y).$$

Acknowledgments

To my parents and teachers, whose guidance and support have shaped who I am today. And to my beloved Sunny Sun, your companionship and encouragement enable me to go further on my journey.