CMSE821 Types of Approximation

Andrew Christlieb

August 25, 2025

The world around us is better understood when we can express it in the form of a model. At its very core a model describes observations. A good model are able to go beyond describing observations and predict events that have not yet been observed. Models may take the form of partial differential equations, the form of data, or a mixture of both.

In this class we will be primarily focused on models coming from partial differential equations and time permitting we will venture into the world of data infused models. For a model to be able to make meaningful predictions, we have to be able to interrogate solutions. When the model is simple, this can be done via analytical approaches. However, writing down closed form analytic solutions to models that are descriptive is often difficult if not impossible. This can be due to geometry nonlinearity or a mix of both. Further, when the model includes aspects that come from data, there does not exist and analytic structure for us to write down a closed form solution.

These notes are meant to augment your CMSE 821 course, numerical methods for partial differential equations. CMSE 821 is an important topic that is only gaining in importance as more and more powerful computers, coupled with innovations in algorithms, enable solutions that were been considered out of reach not even 5 years ago. Coupled with data science methods for making fast surrogates, optimal design of complex systems such as fusion devises is on the horizon.

This course is primarily concerned with providing the student an introduction to key topics in the field of numerical methods. While there are an enormous number of methods that could be talked about we will stick to the fundamental topic of finite difference methods. Fundamentally these methods are based on the idea of replacing continuous derivatives with discrete approximations. This comes from the fact that computers are excellent at adding subtracting multiplying and dividing, hence the key goal of this course is to learn how to convert the partial differential equation into a collection of algebraic equations that represent a ‘” approximation to the original model in some since.

Before getting to the main topic, we will take the rest of this section to provide an overview of modeling and existing analytic tools. It is really important to understand where a model comes from to be able to build a numerical method that reflects key fundamental properties of the model one is trying to create a solution to through numerical methods. For example, if your model is a conservative balance law, then the numerical method had ought to reflect that conservation property. While we do not expect the reader to be an expert in modeling, it is the case that it is a good idea just to have an understanding that this is an important topic that one should get into when studying methods for creating solutions to these equations. In the following sections, we remind the reviewer of classical models form physics and engineering, introduce an overview of method that lead to the derivation of many of the key models in science, introduce nomenclature around partial differential equations, go over the classic linear PDE's and existing methods for analytic solutions, before moving on and diving into the main topic, numerical methods for partial differential equations.

Contents

1 Finite Difference Methods Topics of Importance

1.1 Foundations & Well-posedness

By the end, students will be able to:

F1. Classify PDEs (elliptic/parabolic/hyperbolic) and state well-posedness at a high level (existence/uniqueness/continuous dependence).

F2. Define and *prove* consistency, stability, and convergence; apply Lax equivalence (linear BVP/IVP).

F3. Derive truncation error via Taylor/Peano forms; articulate local vs global error.

F4. Use Big-O, norms, and discrete inner products to bound errors.

1.2 Foundations & Well-posedness

By the end, students will be able to:

F1. Classify PDEs (elliptic/parabolic/hyperbolic) and state well-posedness at a high level (existence/uniqueness/continuous dependence).

F2. Define and *prove* consistency, stability, and convergence; apply Lax equivalence (linear BVP/IVP).

F3. Derive truncation error via Taylor/Peano forms; articulate local vs global error.

F4. Use Big-O, norms, and discrete inner products to bound errors.

1.3 Discrete Analysis Tools

A1. Construct FD stencils on uniform/nonuniform grids (Vandermonde/Fornberg) and compact schemes; analyze order.

A2. Use eigenvalues/eigenvectors of discrete operators A; relate $\rho(A)$ and induced norms.

A3. Perform von Neumann (Fourier/symbol) analysis; quantify dispersion/dissipation.

A4. Apply discrete Green's functions and maximum principles.

A5. Use SBP (summation-by-parts) operators and SAT/penalty BCs for energy stability.

A6. Employ the discrete FFT for Poisson/Helmholtz on rectangles and interpret separability/Kronecker sums.

1.4 Time Discretization & Stability

T1. Select and analyze explicit/implicit/IMEX time integrators; state A-, L-, and SSP-stability.

T2. Derive and use CFL conditions; analyze stiffness and step-size control.

T3. Apply operator splitting (e.g., Strang) and discuss splitting error.

1.5 PDE Models & Structure Preservation

P1. Elliptic: derive/solve discrete Poisson/Helmholtz; treat Dirichlet/Neumann/Robin BCs (ghost cells, one-sided stencils, SAT).

P2. Parabolic: analyze FTCS/Crank–Nicolson/IMEX; prove consistency (order in time/space).

P3. Hyperbolic: design upwind/TVD/ENO/WENO schemes; use flux limiters; conduct von Neumann and monotonicity analysis.

P4. *Structure preservation* (your "stutter preservation"): enforce conservation, positivity, maximum principles, and (where relevant) energy/entropy stability.

P5. Handle curvilinear coordinates and mapped/AMR grids at a conceptual level.

1.6 Linear and Nonlinear Solvers (in the FD context)

S1. Choose/direct banded solvers (LU/Cholesky) for 1D/structured 2D; analyze complexity and storage.

S2. Implement/compare Krylov methods (CG, GMRES, BiCGStab); diagnose convergence by spectra/condition number.

S3. Design preconditioners (Jacobi/SSOR/ILU), geometric and algebraic multigrid (smoothers, restriction/prolongation, cycles), and FFT Poisson solvers; match solvers to PDEs.

S4. Solve nonlinear FD systems via Newton (with Jacobian or Jacobian-free) and Newton–Krylov; assess globalization (line search/trust region).

1.7 Error Estimation, Adaptivity, and Verification/Validation

E1. Perform grid/time refinement and Richardson extrapolation; estimate observed order.

E2. Use Method of Manufactured Solutions (MMS) for code verification.

E3. Explain verification vs validation; design benchmark tests and regression tests.

E4. Outline AMR concepts (refinement criteria, prolongation/restriction, conservation at coarse–fine interfaces).

1.8 Symbolic & Automation Tools

Y1. Use symbolic algebra (e.g., SymPy) to: generate FD weights, series expansions, Jacobians, and code.

Y2. Apply automatic differentiation for Jacobian-free Newton–Krylov diagnostics.

1.9 Software Engineering & Reproducibility

R1. Practice clean, well-documented, object-oriented design for PDE solvers (operators, grids, BC handlers).

R2. Write reusable modules/packages; include unit tests, examples, and API docs (e.g., Sphinx).

R3. Use Git/GitHub (branching, PRs, code review), CI, semantic versioning, and issue tracking.

R4. Maintain reproducible environments (lockfiles/containers), datasets, and experiment logs; license and cite properly.

1.10 Capstone Outcomes

C1. **Derive** a high-order FD scheme on nonuniform grids and **prove** its order and stability.

C2. **Design & implement** a solver stack (discretization + Krylov + preconditioner or multigrid) for a target PDE and **justify** design choices by analysis and experiments.

C3. **Verify** the code with MMS and **demonstrate** predicted orders; **profile** and **optimize** hotspots; **publish** a reproducible repository.

Common gaps this list adds PDE classification and well-posedness; dispersion/dissipation analysis; SBP–SAT boundaries; mapped/AMR grids; multigrid/AMG; floating-point and performance; MMS and V&V; reproducible research practices.

2 Learning Objectives for Engaging with AI in NA & FDM

2.1 Prompting, Provenance, and Traceability

By the end of the course, students will be able to:

- Formulate precise prompts that specify PDE type, BCs, grid, order, and desired outputs (code/LaTeX/figures).
- Constrain AI outputs by *asking for assumptions, symbols, units, and references* explicitly.
- Record prompt–response “provenance” (prompt text, model, date, seed) in a lab notebook or repo.
- Request *two alternative derivations* and reconcile differences as a check on reliability.

2.2 Mathematical Reasoning with AI (FDM Core)

- Derive finite-difference stencils (uniform and nonuniform) via Taylor/Vandermonde, then **verify** AI-derived coefficients by independent symbolic expansion.
- Compute and bound local truncation errors (LTE); **prove** consistency order using AI as a drafting aide.
- Perform von Neumann analysis with AI assistance (symbol manipulation), then **interpret** amplification factors for dispersion/dissipation.
- Establish stability via energy/maximum principle or eigenvalue bounds; **justify** norm choices.
- Apply Lax equivalence: combine consistency and stability to **prove** convergence for a given scheme.

2.3 Symbolic and Computational Tools

- Use AI to generate `SymPy` code for series expansions, stencil weights, Jacobians, and manufactured solutions; **audit** results against manual checks and unit tests.
- Automate **code generation** (e.g., operators, SBP stencils) and **validate** with discrete identities (summation-by-parts, consistency checks).
- Design AI-assisted experiments (grid refinement, Richardson extrapolation) and **plot** observed orders.

2.4 PDE Modeling, Structure, and Boundaries

- Classify PDEs (elliptic/parabolic/hyperbolic) from problem statements the AI helps summarize.
- Choose AI-suggested boundary closures (ghost cells, one-sided, SAT/SBP) and **demonstrate** property preservation (conservation, positivity, maximum principle) on tests.

2.5 Linear/Nonlinear Solvers in the FD Context

- Select solver strategies (banded direct, CG/GMRES, multigrid, FFT Poisson) based on operator spectra and structure **explained** in an AI-aided design memo.
- Use AI to sketch preconditioners (Jacobi/SSOR/ILU, geometric/AMG) and **evaluate** convergence empirically.
- Implement Newton / Jacobian-free Newton–Krylov with AI-generated scaffolding and **verify** quadratic/linear convergence on manufactured problems.

2.6 Verification, Validation, and Numerical Reliability

- Employ the Method of Manufactured Solutions (MMS) that AI helps construct; **confirm** theoretical orders on graded meshes.
- Detect AI errors by stress tests: limiting cases, dimensional analysis, symmetry checks, and conservation balances.

2.7 Software Engineering with AI Assistance

- Maintain **well-documented, object-oriented** solvers (clear APIs, unit tests, type hints) with AI-suggested templates; enforce style via linters/CI.
- Use Git/GitHub workflows (branches, PRs, reviews) and ask AI to **draft** release notes, docs, and test plans.
- Ensure reproducibility (env files/containers, seeds, data versioning); have AI **generate** and **check** build/run scripts and READMEs.

2.8 Critical Thinking, Ethics, and Communication

- Calibrate trust: **triangulate** AI output with textbooks/notes and independent computations; document discrepancies.
- Disclose AI contributions ethically; **paraphrase and prove** core results independently.
- Convert AI drafts to LaTeX with correct notation, labeled equations, and cross-references; **polish** to academic standard.

Quick "Do/Don't" Checklist (for daily use).

- **Do** ask AI for assumptions, symbols, and step-by-step derivations; **verify** with SymPy/tests.
- **Do** request alternative approaches (energy vs Fourier) and compare.
- **Don't** accept unreferenced results; **don't** skip independent checks or MMS.
- **Do** log prompts/responses; **do** cite AI assistance in reports per course policy.

3 Partial Differential Equation and nomenclature

Before we get into the topic of this course, solving nonlinear and linear partial differential equations using computers, we want to start by just going over nomenclature associated with partial differential equation. In this section we introduce classification of classic 1st and 2nd order partial differential equations that are common in the literature, what we mean by first and 2nd order will become self-evident in a few pages. There are partial differential equations that are beyond first and 2nd order, but for the most part they're outside the scope of what we'll discuss here.

After we define nomenclature we will review some common partial differential equations that arise in physics, biology and engineering. This will be followed by a brief discussion on how these models are often formed. This is meant as background because often developing good numerical methods depends on some understanding of the model you are working with and where it comes from.

Suggested use of AI - Sometimes it is very helpful to ask the AI to extend definitions or discuss the reasoning behind a definition or the meaning of a definition. After you read this section I encourage you to have discussion with your favorite AI about what is a partial differential equation. See if this discussion provides any additional insight. Think critically about what the AI tells you.

3.1 Nomenclature

A **partial differential equation (PDE)** is an equation that involves an unknown multivariable function and its partial derivatives with respect to two or more independent variables.

Formally, let $u = u(x_1, x_2, \ldots, x_n)$ be an unknown function of n independent variables. A PDE is a relation of the form

$$F\left(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_i \partial x_j}, \ldots\right) = 0,$$

where F is some given function involving u and its partial derivatives up to some order.

3.2 Definitions of PDEs

- **Order:** The *order* of a PDE is the order of the highest partial derivative that appears in the equation.
  - First-order example:
  $$u_t + c\, u_x = 0.$$
  - Second-order example:
  $$u_{tt} = c^2 \left(u_{xx} + u_{yy}\right).$$
- **Linearity:** A PDE is *linear* if the unknown function u and its derivatives appear *linearly*, i.e., not multiplied together and not inside nonlinear functions. Otherwise, it is *nonlinear*.
  - Linear example:
  $$u_t - k\, u_{xx} = 0.$$
  - Nonlinear example:
  $$u_t + u\, u_x = 0.$$

3.3 Hyperbolic Systems of First-Order PDEs

A **first-order system of PDEs** in one spatial variable can be written as

$$\mathbf{u}_t + A(x,t)\,\mathbf{u}_x = 0,$$

where $\mathbf{u}(x,t) \in \mathbb{R}^m$ is the vector of unknown functions, and $A(x,t)$ is an $m \times m$ coefficient matrix that may depend on space and time.

Definition 3.1 (Definition (Hyperbolicity 1D))**.** The system is called **hyperbolic** if, for every (x,t), the matrix $A(x,t)$ has

- m **real eigenvalues**, and
- a complete set of m linearly independent eigenvectors.
- If all eigenvalues are real but the eigenvectors are not linearly independent, the system is called *weakly hyperbolic.*
- If all eigenvalues are real and distinct, the system is called *strictly hyperbolic.*

This property ensures that the system can be diagonalized and that the solution can be decomposed into *waves* traveling along characteristic curves.

3.3.1 Examples

- **Linear advection equation:**
$$u_t + c\,u_x = 0,$$
with constant speed c, is hyperbolic since the coefficient matrix is $[c]$ with real eigenvalue c.
- **1D Linearized Shallow Water Equations:**
$$\begin{cases} h_t + Hu_x = 0, \\ u_t + gh_x = 0 \end{cases}$$
where h is the water depth, u is velocity, and g is the gravitational constant. The coefficient matrix has eigenvalues $\pm\sqrt{gh}$, which are real, hence the system is hyperbolic.

Definition 3.2 (Hyperbolic System in Multiple Space Dimensions)**.** Consider a first-order system of PDEs in one spatial dimension
$$\mathbf{u}_t + A\,\mathbf{u}_x = 0,$$
where $\mathbf{u}(x,t) \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times m}$ is constant. The system is called **hyperbolic in 1D** if A has m real eigenvalues and a complete set of eigenvectors.

In two spatial dimensions, the system takes the form

$$\mathbf{u}_t + A\,\mathbf{u}_x + B\,\mathbf{u}_y = 0,$$

with constant matrices $A, B \in \mathbb{R}^{m\times m}$. It is called **hyperbolic in 2D** if, for every real pair $(\xi,\eta) \in \mathbb{R}^2$, the combined matrix
$$\xi A + \eta B$$

is diagonalizable with real eigenvalues. That is, the system reduces to a hyperbolic 1D system in any spatial direction.

In three spatial dimensions, the system takes the form

$$\mathbf{u}_t + A\,\mathbf{u}_x + B\,\mathbf{u}_y + C\,\mathbf{u}_z = 0,$$

with constant matrices $A, B, C \in \mathbb{R}^{m\times m}$. It is **hyperbolic in 3D** if, for every real triple $(\xi, \eta, \zeta) \in \mathbb{R}^3$, the matrix

$$\xi A + \eta B + \zeta C$$

is diagonalizable with real eigenvalues. Again, this means the system is hyperbolic in every spatial direction.

3.4 Classification of second order PDEs + Time

A general second-order PDE in two spatial variables and time can be written as

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_{tt} + \text{lower order terms} = 0,$$

where the classification depends on the discriminant of the spatial part

$$\Delta = B^2 - AC.$$

3.4.1 Elliptic PDEs

- **Condition:** $\Delta < 0$.
- **Interpretation:** Governs steady-state, equilibrium-type problems with no explicit time dependence.
- **Canonical Example (Laplace equation):**

$$u_{xx} + u_{yy} = 0.$$

- **Physical context:** Steady-state heat conduction, incompressible fluid flow, electrostatics.

3.4.2 Parabolic PDEs

- **Condition:** $\Delta = 0$.
- **Interpretation:** Models diffusion or dissipative processes, with one direction playing the role of time.
- **Canonical Example (Heat/Diffusion equation):**

$$u_t = u_{xx} + u_{yy}.$$

- **Physical context:** Heat conduction, diffusion of particles.

3.4.3 Hyperbolic PDEs

- **Condition:** $\Delta > 0$.
- **Interpretation:** Models wave propagation, with finite speed of disturbances.
- **Canonical Example (Wave equation):**

$$u_{tt} = u_{xx} + u_{yy}.$$

- **Physical context:** Vibrations of a drum membrane, acoustic and electromagnetic waves.

3.4.4 Summary Table

Type	**Condition**	**Canonical PDE (2D + time)**	**Physical Meaning**
Elliptic	$B^2 - AC < 0$	$u_{xx} + u_{yy} = 0$	Steady-state equilibrium
Parabolic	$B^2 - AC = 0$	$u_t = u_{xx} + u_{yy}$	Diffusion / Heat flow
Hyperbolic	$B^2 - AC > 0$	$u_{tt} = u_{xx} + u_{yy}$	Wave propagation

4 Example Models in STEM where Scientific Computing has changed our approach to science and engineering.

Why we model. Across physics, engineering, and biology, the tradition of *mathematical modeling* is to write down idealized, quantitative descriptions of real systems so that we can predict, control, and understand them. A model replaces the full complexity of nature with variables (fields, parameters, random processes) and governing relationships (algebraic laws, differential equations, conservation principles). This substitution is not merely descriptive; it is an explicit claim about which mechanisms dominate and which can be neglected on the scales of interest. Done well, modeling lets us pose "what if" questions, design experiments and devices, test hypotheses against data, and transfer insight from one system to another. The sections below illustrate this arc with three workhorse models and how they connect to technology and science.

In the following section after this overview, we provide a brief background of one of the fundamental tools for deriving models in STEM known as the calculus of variations. While there are many methods for modeling, this has been a workhorse approach to developing models since Euler first invented it. Again, keep in mind this is meant to be background material for the reader.

Suggested use of AI - once you read this section, pick a topic you are interested and ask the AI about PDE/ODE models used in this area. Ask the I to cite books and papers using google scholar. Its really good to get an idea of how AI can accurate literature searches when combined with RAG (how good AI's do it now). You need to verify the articles it suggests but it new AI is better than old AI was at this.

Suggested use of AI - have the AI suggest which papers to read on a given topic and justify why these papers. If you are new to an area, tell the AI you are new. See if you agree with its reasoning for why you should read these papers.

4.1 Electromagnetism: Maxwell's Equations and Antenna Modeling

What the equations say. In macroscopic media, Maxwell's equations couple electric and magnetic fields:

$$\nabla\cdot \boldsymbol{D} = \rho_f, \qquad \text{(Gauss for electricity)} \tag{4.1}$$

$$\nabla\cdot \boldsymbol{B} = 0, \qquad \text{(Gauss for magnetism)} \tag{4.2}$$

$$\boldsymbol{\nabla}\times \boldsymbol{E} = -\,\partial_t \boldsymbol{B}, \qquad \text{(Faraday's law)} \tag{4.3}$$

$$\boldsymbol{\nabla}\times \boldsymbol{H} = \boldsymbol{J}_f + \partial_t \boldsymbol{D}. \qquad \text{(Ampère–Maxwell)} \tag{4.4}$$

Definitions and closures.

- $\boldsymbol{E}$ [V/m]: electric field; $\boldsymbol{B}$ [T]: magnetic flux density.
- $\boldsymbol{D}$ [C/m^2]: electric displacement; $\boldsymbol{H}$ [A/m]: magnetic field intensity.
- ρ_f [C/m^3]: free charge density; $\boldsymbol{J}_f$ [A/m^2]: free current density.
- Constitutive relations (linear, isotropic, homogeneous media): $\boldsymbol{D} = \varepsilon\boldsymbol{E}$, $\boldsymbol{B} = \mu\boldsymbol{H}$, and (ohmic) $\boldsymbol{J}_f = \sigma\boldsymbol{E}$. Here ε [F/m] is permittivity, μ [H/m] is permeability, and σ [S/m] is conductivity.
- Boundary data: tangential $\boldsymbol{E}$ and normal $\boldsymbol{B}$ are constrained by material interfaces; impressed sources (feeds) prescribe $\boldsymbol{J}_f$ or voltages.

Equations (4.1)–(4.4) imply wave equations (e.g., for $\boldsymbol{E}$) and energy balance via the Poynting vector $\boldsymbol{S} = \boldsymbol{E}\times\boldsymbol{H}$ and energy density $u = \frac{1}{2}(\boldsymbol{E}\cdot\boldsymbol{D} + \boldsymbol{B}\cdot\boldsymbol{H})$.

How this models antennas (cellular and Wi-Fi). An antenna is a material structure that converts guided currents into radiated electromagnetic waves and vice versa. Modeling proceeds by:

1. *Frequency–domain reduction.* Assume time-harmonic fields, $\boldsymbol{E}(\boldsymbol{x}, t) = \mathrm{Re}\{\tilde{\boldsymbol{E}}(\boldsymbol{x}) e^{i\omega t}\}$, turning (4.3)–(4.4) into curl–curl equations for $\tilde{\boldsymbol{E}}$ and $\tilde{\boldsymbol{H}}$ at carrier frequencies in the 1 GHz–7 GHz range (cellular, Wi-Fi).
2. *Geometry & materials.* Specify substrate permittivities, conductors' conductivities, and handset chassis. Impedance boundary conditions capture finite conductivity at RF.
3. *Solve and postprocess.* Compute input impedance, radiation patterns, gain/efficiency, and SAR (specific absorption rate). Figures of merit feed design loops (matching networks, MIMO arrays, beamforming).

Common numerical solvers are method of moments (surface integral equations for conductors), finite element method (inhomogeneous dielectrics), and FDTD (time domain), each enforcing (4.1)–(4.4) plus constitutive laws.

4.2 Fluid Mechanics: Navier–Stokes Equations for Flight and Blood Flow

Governing model. Let $\rho(\boldsymbol{x}, t)$ be density, $\boldsymbol{u}(\boldsymbol{x}, t)$ velocity, $p(\boldsymbol{x}, t)$ pressure, and $\boldsymbol{f}$ body force per mass. The compressible Navier–Stokes equations (mass, momentum, energy) read

$$\partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{4.5}$$
$$\rho\big(\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}\big) = -\boldsymbol{\nabla} p + \nabla \cdot \boldsymbol{\tau} + \rho \boldsymbol{f}, \tag{4.6}$$
$$\partial_t(\rho E) + \nabla \cdot \big((\rho E + p)\boldsymbol{u}\big) = \nabla \cdot (\boldsymbol{\tau} \boldsymbol{u}) - \nabla \cdot \boldsymbol{q} + \rho \boldsymbol{f} \cdot \boldsymbol{u}. \tag{4.7}$$

Closures and definitions.

- $E = e + \frac{1}{2}|\boldsymbol{u}|^2$ (specific total energy), e internal energy.
- Viscous stress (Newtonian fluid): $\boldsymbol{\tau} = \mu\big(\boldsymbol{\nabla} \boldsymbol{u} + (\boldsymbol{\nabla} \boldsymbol{u})^\top\big) + \lambda\,(\nabla \cdot \boldsymbol{u})\,\boldsymbol{I}$, with dynamic viscosity μ and bulk viscosity λ (Stokes' hypothesis: $\lambda = -\frac{2}{3}\mu$).
- Heat flux (Fourier): $\boldsymbol{q} = -k \boldsymbol{\nabla} T$, with thermal conductivity k and temperature T.
- Equation of state: for gases, $p = \rho R T$ and $e = c_v T$ (ideal gas).

In many liquids and moderate-speed gases one uses the incompressible limit:

$$\nabla \cdot \boldsymbol{u} = 0, \qquad \rho\big(\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}\big) = -\boldsymbol{\nabla} p + \mu \Delta \boldsymbol{u} + \rho \boldsymbol{f}. \tag{4.8}$$

How this models flight. Aircraft aerodynamics is governed by (4.5)–(4.7) with compressibility (Mach number M) and turbulence effects:

- *Regimes.* Subsonic/transonic/supersonic flows differ by shock formation and compressibility; boundary layers and separation drive lift/drag.
- *Models of turbulence.* RANS (Reynolds-averaged) with eddy-viscosity closures for design loops; LES for separated flows; DNS only for canonical, low-Re cases.
- *Quantities of interest.* Lift, drag, moments, buffet onset, aeroacoustics. Geometry and conditions enter via boundary conditions (no-slip walls, far-field inflow) and equations of state.

How this models blood flow. Hemodynamics in large vessels is well-approximated by (4.8), with important biology-specific features:

- *Rheology.* Blood is mildly non-Newtonian (shear-thinning); Carreau–Yasuda laws adjust $\mu(\dot{\gamma})$. In major arteries a Newtonian assumption often suffices.
- *Pulsatility and scales.* The Womersley number $\alpha = R\sqrt{\omega\rho/\mu}$ gauges unsteady inertial dominance; wave propagation couples to vessel compliance.
- *Fluid–structure interaction.* Wall elasticity matters (FSI) for aneurysm risk, stent design, and valve performance; boundary conditions encode measured inflow waveforms and Windkessel outflow models.

4.3 Kinetic Theory: Boltzmann Equation and Gyrokinetic Reductions for Fusion

Governing model. The Boltzmann equation evolves the one-particle distribution $f(\boldsymbol{x}, \boldsymbol{v}, t)$ in phase space:

$$\partial_t f + \boldsymbol{v} \cdot \boldsymbol{\nabla}_{\boldsymbol{x}} f + \frac{\boldsymbol{F}}{m} \cdot \boldsymbol{\nabla}_{\boldsymbol{v}} f = Q(f, f). \tag{4.9}$$

Definitions.

- f [$\mathrm{s}^3/\mathrm{m}^6$]: probability density in $(\boldsymbol{x}, \boldsymbol{v})$; m particle mass.
- $\boldsymbol{F}$ external force; for plasmas, $\boldsymbol{F} = q\big(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}\big)$ (Lorentz force with charge q).
- $Q(f, f)$ collision operator (bilinear, conserving mass/momentum/energy). Simplified surrogates include BGK and Fokker–Planck forms.
- Moments of f produce continuum fields (density, momentum, temperature) and, via Chapman–Enskog, recover Euler/Navier–Stokes as asymptotic limits.

Gyrokinetic reduction for magnetized fusion plasmas. In toroidal devices (tokamaks, stellarators), strong magnetic fields make charged particles gyrate rapidly about field lines with Larmor frequency Ω_c and small radius ρ_L. When turbulence and transport evolve on timescales $\gg \Omega_c^{-1}$ and spatial scales $\gg \rho_L$, one averages over the fast gyro-angle to obtain *gyrokinetics*:

- *Variables and dimension reduction.* The distribution is expressed in guiding-center coordinates $(\boldsymbol{R}, v_\parallel, \mu, t)$, where μ is the magnetic moment. The phase space drops from 6D to 5D.
- *Equations.* The gyrokinetic Vlasov equation evolves the fluctuating part δf coupled to field equations (quasi-neutrality and, for electromagnetic models, parallel vector potential). Collisions use reduced operators.
- *Use.* This model captures drift-wave microturbulence and predicts heat/particle fluxes that set confinement quality—central to performance projections for devices like ITER. Numerically, codes (e.g., δf particle-in-cell and Eulerian solvers) discretize the 5D equations on complex magnetic geometries.

4.4 On Models, Solvability, and Numerical Methods

“All models are wrong, but some are useful.” — George E. Box

“Everything should be made as simple as possible, but not simpler.” — attributed to Albert Einstein

“With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.” — John von Neumann

These aphorisms remind us that even externally accurate models can be analytically intractable. Maxwell’s equations in real handset geometries, Navier–Stokes at flight Reynolds numbers, and gyrokinetic turbulence in fusion devices rarely admit closed-form solutions. In practice, insight flows from *numerical methods*: finite elements, finite volumes, spectral and discontinuous Galerkin schemes for PDEs; method of moments and FDTD in electromagnetics; RANS/LES/DNS and FSI solvers in fluids; DSMC for rarefied gases; particle-in-cell and Eulerian solvers for kinetic and gyrokinetic plasmas. Discretization, linear/nonlinear solvers, adaptivity, and high-performance computing turn principled models into predictive tools—making the models useful, even when exact solutions are out of reach.

5 Example of modeling form physical precipices

It's critical to understand that modeling pause it balance laws in one form or another and then extracts from those balance laws a macroscopic description that is meaningful in the context of the problem one is looking at. A fundamental approach introduced by Euler is known as the calculus of variations. The calculus of variations lets one write down partial differential equations for describing how the world evolves around us based on a few key principles. There are entire books dedicated to this topic and here we only want to provide the reader with the notion of how this is done. Below we outline the basic principles and the calculus of variations followed by going through the example of deriving Euler's equations of fluid dynamics. This is meant as an informative example.

It should also be understood that Data Science offers another approach for building models. A key question wether the model form physical principles or data is the validity and accuracy of the model. In many cases, for modern problems in STEM, blending models form data with models that come form physical precipices are at the cutting edge of doing critical work. Why this is beyond what we do here, it is important to know that one need a core knowledge of both data science and scientific computing to make real progress on some of the most challenging problems in science and engineering.

We now give a brief overview of the calculates of variations as one approach to modeling.

Suggested use of AI - once you read this section, for an extend background knowledge, talk with your favorite AI about this topic and get it to provide you with references that you check out and take a look at. Its really good to get an idea of the tools available to you in the world of modeling.

5.1 From a control volume balance to a pointwise conservation law

Modeling setup. Let $\Omega \subset \mathbb{R}^3$ be the spatial region occupied by a continuum, and let $V \subset \Omega$ be an arbitrary fixed (time-independent) control volume with piecewise smooth boundary ∂V and outward unit normal $\boldsymbol{n}$. Consider any extensive quantity A (e.g. mass, linear momentum, energy) with:

- a *density* $a(\boldsymbol{x}, t)$ per unit volume;
- an associated *flux* $\boldsymbol{F}(\boldsymbol{x}, t)$ across surfaces (per unit area per unit time);
- an optional *source* $s(\boldsymbol{x}, t)$ (production per unit volume).

The *integral balance* over V states that the time rate of change of A inside V equals the net inflow through ∂V plus the integrated sources:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V a\,\mathrm{d}V + \int_{\partial V} \boldsymbol{F}\cdot\boldsymbol{n}\,\mathrm{d}S = \int_V s\,\mathrm{d}V. \tag{5.1}$$

Divergence theorem and localization. Apply the divergence theorem to the surface flux term:

$$\int_{\partial V} \boldsymbol{F}\cdot\boldsymbol{n}\,\mathrm{d}S = \int_V \nabla\cdot\boldsymbol{F}\,\mathrm{d}V,$$

and (since V is fixed in time) commute $\frac{\mathrm{d}}{\mathrm{d}t}$ with the volume integral in the first term of (5.1). We obtain

$$\int_V \Big(a_t + \nabla\cdot\boldsymbol{F} - s\Big)\,\mathrm{d}V = 0 \quad \text{for every fixed control volume } V \subset \Omega. \tag{5.2}$$

Fundamental lemma of the calculus of variations (FLCoV).

Lemma 5.1 (Fundamental lemma, volume form)**.** *If $\int_V g(\boldsymbol{x},t)\,\mathrm{d}V = 0$ for all (sufficiently regular) subdomains $V \subset \Omega$, then $g(\boldsymbol{x},t) = 0$ almost everywhere in Ω.*

This is the same localization principle used when deriving Euler–Lagrange equations: an integral identity holding for *arbitrary variations* (here, arbitrary choices of V) forces the integrand to vanish pointwise.

Local conservation law. Applying the lemma to (5.2) yields the *pointwise* or *local* conservation law

$$\boxed{a_t + \nabla\cdot\boldsymbol{F} \;=\; s \quad \text{in } \Omega.} \tag{5.3}$$

Examples.

- **Mass:** $a = \rho$, $\boldsymbol{F} = \rho\boldsymbol{u}$, $s = 0$:
$$\rho_t + \nabla\cdot(\rho\boldsymbol{u}) = 0. \tag{5.4}$$
- **Linear momentum (inviscid):** $a = \rho\boldsymbol{u}$, $\boldsymbol{F} = \rho\boldsymbol{u}\otimes\boldsymbol{u} + p\,\mathbb{I}$, $s = \rho\boldsymbol{f}$:
$$\partial_t(\rho\boldsymbol{u}) + \nabla\cdot\big(\rho\boldsymbol{u}\otimes\boldsymbol{u} + p\,\mathbb{I}\big) = \rho\boldsymbol{f}. \tag{5.5}$$
- **Total energy (inviscid, adiabatic):** with specific total energy $E = e + \frac{1}{2}|\boldsymbol{u}|^2$,
$$\partial_t(\rho E) + \nabla\cdot\big((\rho E + p)\boldsymbol{u}\big) = \rho\,\boldsymbol{f}\cdot\boldsymbol{u}. \tag{5.6}$$

Remark (moving control volumes). If the control surface itself moves and deforms, the Reynolds Transport Theorem provides the corresponding generalization; localizations analogous to (5.3) still follow.

5.2 Calculus of variations viewpoint: how a conservation law arises

There are two complementary ways variational ideas produce conservation laws:

(i) Localization of an integral identity. As just used, an identity that holds for all test sets (or, equivalently, for all smooth compactly supported test functions) implies pointwise vanishing of the integrand. This is precisely the logical step at the heart of the Euler–Lagrange derivation.

(ii) Noether's theorem (symmetry $\Rightarrow$ conservation). Let $S = \int L(\text{fields}, \partial\text{fields}, x, t)\,\mathrm{d}x\,\mathrm{d}t$ be an action functional. If S is invariant under a continuous group of transformations, then there exists a corresponding *conserved current* J^μ whose divergence vanishes, $\partial_\mu J^\mu = 0$. For instance,

- Time translations $\Rightarrow$ energy conservation.
- Spatial translations $\Rightarrow$ linear momentum conservation.
- Rotations $\Rightarrow$ angular momentum conservation.
- Particle-relabeling symmetry in ideal fluids $\Rightarrow$ Kelvin's circulation theorem.

Thus, in addition to the control-volume derivation, conservation laws also appear as a structural consequence of symmetries of L.

5.3 Deriving the compressible Euler equations via a variational principle

We derive the inviscid, barotropic Euler equations. Two routes are common. The material (Lagrangian) route *builds in* mass conservation; the Eulerian route enforces it by a constraint. Both give the same PDEs.

5.3.1 Material (Lagrangian) description

Kinematics. Let $\eta : B \times [t_0, t_1] \to \Omega$ be the flow map taking each label $X \in B$ in the reference configuration to its current position $\boldsymbol{x} = \eta(X, t)$. Denote the deformation gradient $F = \partial\eta/\partial X$ and its Jacobian $J = \det F$. The current density is

$$\rho(\boldsymbol{x}, t) \;=\; \frac{\rho_0(X)}{J(X,t)} \quad \text{with } \boldsymbol{x} = \eta(X,t), \tag{5.7}$$

expressing mass conservation in material form. The velocity field is $\boldsymbol{u}(\boldsymbol{x}, t) = \partial_t \eta(X, t)$ evaluated at $X = \eta^{-1}(\boldsymbol{x}, t)$.

Energetics and Lagrangian density. Assume a barotropic fluid with specific internal energy $e = e(\rho)$ and (optional) body potential $\Phi(\boldsymbol{x})$ (so body force $\boldsymbol{f} = -\nabla\Phi$). The material action is

$$S[\eta] \;=\; \int_{t_0}^{t_1}\int_B \Big[\tfrac{1}{2}\rho_0(X)\,|\partial_t\eta(X,t)|^2 \;-\; \rho_0(X)\,e\Big(\frac{\rho_0(X)}{J(X,t)}\Big) \;-\; \rho_0(X)\,\Phi(\eta(X,t))\Big]\,\mathrm{d}X\,\mathrm{d}t. \tag{5.8}$$

The pressure–density relation induced by $e(\rho)$ is

$$p(\rho) \;=\; \rho^2\, e'(\rho). \tag{5.9}$$

First variation. Consider variations $\eta \mapsto \eta + \varepsilon\,\boldsymbol{w}$ with $\boldsymbol{w}(X, t_0) = \boldsymbol{w}(X, t_1) = \boldsymbol{0}$ and either fixed or compatible boundary data on ∂B. The key identities are

$$\delta J = J\,\mathrm{tr}\big(F^{-1}\delta F\big), \qquad \delta\Big(\frac{1}{J}\Big) = -\,\frac{1}{J}\,\mathrm{tr}\big(F^{-1}\delta F\big), \tag{5.10}$$

$$\delta\, e\Big(\frac{\rho_0}{J}\Big) = e'(\rho)\,\delta\rho = e'(\rho)\,\big(-\rho\,\mathrm{tr}\big(F^{-1}\delta F\big)\big)\,, \quad \rho = \frac{\rho_0}{J}. \tag{5.11}$$

After substituting (5.10)–(5.11) into δS, integrating by parts in t and X, and pulling the result forward to Eulerian variables (this is a standard but lengthy calculation), one obtains the Euler–Lagrange equation in Eulerian form:

$$\rho\,(\partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}) + \nabla p + \rho\,\nabla\Phi \;=\; \boldsymbol{0}, \tag{5.12}$$

together with the (already built-in) mass conservation (5.4).

Boundary and initial data. Natural boundary conditions arise from the boundary terms in δS: for example, impermeable walls correspond to $\boldsymbol{u}\cdot\boldsymbol{n} = 0$; traction-free boundaries correspond to $p = 0$ on the boundary. Initial conditions are prescribed as $\eta(X, t_0)$ and $\partial_t\eta(X, t_0)$ (equivalently, $\rho(\cdot, t_0)$ and $\boldsymbol{u}(\cdot, t_0)$).

5.3.2 Eulerian (constrained) description

Fields and constraint. Work directly with Eulerian fields $(\rho, \boldsymbol{u})$ and enforce mass conservation (5.4) via a Lagrange multiplier $\lambda(\boldsymbol{x}, t)$. Consider the action

$$S[\rho, \boldsymbol{u}, \lambda] = \int_{t_0}^{t_1} \int_\Omega \Big[\tfrac{1}{2}\rho|\boldsymbol{u}|^2 \;-\; \rho\, e(\rho) \;-\; \rho\,\Phi(\boldsymbol{x}) \;+\; \lambda\big(\rho_t + \nabla\cdot(\rho\boldsymbol{u})\big)\Big]\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t. \tag{5.13}$$

Variations. Assuming suitable decay or boundary conditions so that boundary terms vanish, compute:

$$\begin{aligned}
\delta_{\boldsymbol{u}} S: &\quad \rho\,\boldsymbol{u} - \nabla\lambda\,\rho \;=\; \boldsymbol{0} \;\;\Rightarrow\;\; \boldsymbol{u} = \nabla\lambda \quad \text{(Clebsch form; more generally one can allow vorticity with additional potenti}\\
\delta_\rho S: &\quad \tfrac{1}{2}|\boldsymbol{u}|^2 - e(\rho) - \rho e'(\rho) - \Phi - \lambda_t - \boldsymbol{u}\cdot\nabla\lambda \;=\; 0,\\
\delta_\lambda S: &\quad \rho_t + \nabla\cdot(\rho\boldsymbol{u}) \;=\; 0.
\end{aligned}$$

Combine these (eliminating λ) to recover the momentum equation in either advective form (5.12) or conservative form (5.5). This route makes explicit how the conservation law (5.4) enters variationally as a constraint.

5.4 Final forms: Euler equations (compressible, inviscid, barotropic)

With body force $\boldsymbol{f} = -\nabla\Phi$ and equation of state $p(\rho) = \rho^2 e'(\rho)$:

$$\begin{aligned}
\text{Mass (continuity):} &\quad \rho_t + \nabla\cdot(\rho\boldsymbol{u}) = 0, && (5.14)\\
\text{Momentum (advective form):} &\quad \rho\,(\partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}) + \nabla p = \rho\,\boldsymbol{f}, && (5.15)\\
\text{Momentum (conservative form):} &\quad \partial_t(\rho\boldsymbol{u}) + \nabla\cdot\big(\rho\boldsymbol{u}\otimes\boldsymbol{u} + p\,\mathbb{I}\big) = \rho\,\boldsymbol{f}, && (5.16)\\
\text{(Optional) Total energy:} &\quad \partial_t(\rho E) + \nabla\cdot\big((\rho E + p)\boldsymbol{u}\big) = \rho\,\boldsymbol{f}\cdot\boldsymbol{u}, \quad E = e(\rho) + \tfrac{1}{2}|\boldsymbol{u}|^2. && (5.17)
\end{aligned}$$

[Incompressible limit] Impose $\nabla\cdot\boldsymbol{u} = 0$ (or $J \equiv 1$ in the material picture) via a Lagrange multiplier; that multiplier appears as the (mechanical) pressure enforcing incompressibility, yielding

$$\nabla\cdot\boldsymbol{u} = 0, \qquad \rho\,(\partial_t\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}) + \nabla p = \rho\,\boldsymbol{f}.$$

[Noether perspective] Space/time translation invariance of the action underlying (5.13) yields, via Noether's theorem, the conservation statements (5.5) and (5.6). Particle-relabeling symmetry yields Kelvin's circulation theorem; all are consistent with the control-volume balances.

5.5 Checklist for modeling from "precipices" to PDEs

1. Choose state fields (e.g. ρ, $\boldsymbol{u}$) and specify which extensive quantities you will balance.
2. Write the control-volume balance and localize it via divergence theorem + FLCoV.
3. Specify constitutive relations (e.g. $p = p(\rho)$ or $e = e(\rho)$).

4. If using a variational route, build the action (kinetic − internal − potential), add constraints as needed.
5. Take first variations, integrate by parts, read off PDEs and natural boundary terms; supply initial data.

6 Classical linear PDE's on simple domains and solutions

In this section we review classic linear PDE's that we can derive analytic solutions for on simple domains. These problems represent a class of problems that we start developing numerical methods for with the intent of addressing more complex nonlinear problems. The techniques used to analyze these problems and develop analytic solutions also have a discrete analog in the numerical analysis community. While we will not be spending much time on the idea of separation of variables, the core result that it allows you to derive an eigenfunction along with the eigenvalues for a given linear PDE will have an exact duology and the finite difference context. This duology leads to many of the analysis techniques we will use in understanding our finite difference approximations.

6.1 Linear PDE's of STEM

In cases when the partial differential equation is relatively simple, i.e. linear, on simple domains there are a wide range of approaches for constructing analytic solutions. Classically the most well known is separation of variables. Separation of variables posits and ansatz $u(t, x, y) = T(t)X(x)Y(y)$ and then proceeds to show that this assumed form leads to a non trivial solution. another common approach is called the method of Green's functions. It leverages ideas from calculus 3 and the idea of a delta function to construct an inverse operator that lets one write a general solution for the linear PDE given any right hand side. Both of these approaches lead to analytic tools that end up being helpful in numerical analysis. There are entire courses dedicated to these topics and so the material below is meant to provide a synopsis of the ideas for classical equations that we consider. The reader is not required to read the following sections if they wish to just skip ahead to topics on finite difference methods, it is only meant as a background for tools that will be leveraged later when considering aspects of numerical methods around stability and consistency. We now go over the three class linear PDE's.

NOTE: there is a companion document that holds up the solution to each of the following three examples in a Jupiter notebook. The companion document is called 8_25_2025_Serpation_of_Vareables_soluton_Poisson_Heat_Wave_2D.ipynb

6.2 2D Poisson Equation with Homogeneous Dirichlet Data

Problem. Given a rectangle $\Omega = (0, L_x) \times (0, L_y)$, find u such that

$$-\Delta u \equiv -\big(u_{xx} + u_{yy}\big) = f(x, y) \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega.$$

6.2.1 Separation of Variables: Why the "quotients" must be constants

We first analyze the associated eigenvalue problem

$$-\Delta\phi = \lambda\,\phi \quad \text{in } \Omega, \qquad \phi = 0 \quad \text{on } \partial\Omega.$$

Seek product solutions $\phi(x, y) = X(x)Y(y)$ with $X \not\equiv 0$, $Y \not\equiv 0$. Substituting and dividing by $X(x)Y(y)$ gives

$$-\frac{X''(x)}{X(x)} - \frac{Y''(y)}{Y(y)} = \lambda.$$

Rearrange in two equivalent ways to isolate the x- and y-dependence:

$$-\frac{X''(x)}{X(x)} = \lambda + \frac{Y''(y)}{Y(y)},$$
$$-\frac{Y''(y)}{Y(y)} = \lambda + \frac{X''(x)}{X(x)}.$$

The left-hand side of the first line depends only on x, while the right-hand side depends only on y. The only way a function of x can equal a function of y for *all* $x \in (0, L_x)$ and $y \in (0, L_y)$ is that both are equal to a *constant.* (A standard argument: differentiate the first identity with respect to x to get a function of x equal to 0 for all (x, y), hence that function is 0; similarly differentiating with respect to y forces the other side to be constant.)

Therefore there exist constants $\mu, \nu \in \mathbb{R}$ such that

$$-\frac{X''}{X} = \mu, \qquad -\frac{Y''}{Y} = \nu, \qquad \mu + \nu = \lambda.$$

Equivalently,

$$X'' + \mu X = 0, \qquad Y'' + \nu Y = 0.$$

6.2.2 Admissible constants and nontrivial solutions

The Dirichlet boundary conditions inherited from $\phi = 0$ on $\partial\Omega$ are

$$X(0) = X(L_x) = 0, \qquad Y(0) = Y(L_y) = 0.$$

We now classify the possibilities for μ (the same reasoning applies to ν).

Case $\mu < 0$. Write $\mu = -\alpha^2$ with $\alpha > 0$. Then $X'' - \alpha^2 X = 0$ so $X(x) = Ae^{\alpha x} + Be^{-\alpha x}$. The Dirichlet conditions $X(0) = X(L_x) = 0$ force $A + B = 0$ and $Ae^{\alpha L_x} + Be^{-\alpha L_x} = 0$, which imply $A = B = 0$. Hence only the trivial solution exists. Thus $\mu < 0$ is *not* admissible.

Case $\mu = 0$. Then $X'' = 0$, so $X(x) = A + Bx$. The Dirichlet conditions $X(0) = X(L_x) = 0$ give $A = 0$ and $BL_x = 0 \Rightarrow B = 0$. Again only the trivial solution. Thus $\mu = 0$ is *not* admissible.

Case $\mu > 0$. Write $\mu = \alpha^2$ with $\alpha > 0$. Then $X'' + \alpha^2 X = 0$, so $X(x) = A\cos(\alpha x) + B\sin(\alpha x)$. The Dirichlet conditions give $A = 0$ and $\sin(\alpha L_x) = 0$, hence $\alpha L_x = m\pi$ for $m = 1, 2, \ldots$. Therefore

$$\mu_m = \Big(\frac{m\pi}{L_x}\Big)^2, \qquad X_m(x) = \sin\Big(\frac{m\pi x}{L_x}\Big), \qquad m = 1, 2, \ldots$$

By the same argument for Y,

$$\nu_n = \Big(\frac{n\pi}{L_y}\Big)^2, \qquad Y_n(y) = \sin\Big(\frac{n\pi y}{L_y}\Big), \qquad n = 1, 2, \ldots$$

Conclusion. The only nontrivial separated solutions under homogeneous Dirichlet BCs correspond to *positive* separation constants. All admissible (product) eigenfunctions and eigenvalues are

$$\phi_{mn}(x, y) = X_m(x)Y_n(y) = \sin\Big(\frac{m\pi x}{L_x}\Big)\sin\Big(\frac{n\pi y}{L_y}\Big), \qquad \lambda_{mn} = \mu_m + \nu_n = \Big(\frac{m\pi}{L_x}\Big)^2 + \Big(\frac{n\pi}{L_y}\Big)^2.$$

6.2.3 Orthogonality, completeness, and building the general solution

The family $\{\phi_{mn}\}_{m,n\geq 1}$ is orthogonal in $L^2(\Omega)$:

$$\begin{aligned}\int_0^{L_x}\int_0^{L_y} \phi_{mn}(x,y)\,\phi_{pq}(x,y)\,dy\,dx &= \Big(\int_0^{L_x} \sin\Big(\tfrac{m\pi x}{L_x}\Big)\sin\Big(\tfrac{p\pi x}{L_x}\Big)\,dx\Big)\Big(\int_0^{L_y} \sin\Big(\tfrac{n\pi y}{L_y}\Big)\sin\Big(\tfrac{q\pi y}{L_y}\Big)\,dy\Big)\\ &= \frac{L_xL_y}{4}\,\delta_{mp}\delta_{nq}.\end{aligned}$$

Moreover, these eigenfunctions form a complete orthogonal basis of $L^2(\Omega)$ adapted to the Dirichlet boundary (this is a standard theorem for Sturm–Liouville problems on rectangles).

Eigenfunction expansion of the Poisson solution. Expand f and u in the basis:

$$\begin{aligned} f(x,y) &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} F_{mn}\,\phi_{mn}(x,y),\\ u(x,y) &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} U_{mn}\,\phi_{mn}(x,y),\end{aligned}$$

with Fourier–sine coefficients

$$F_{mn} = \frac{4}{L_xL_y}\int_0^{L_x}\int_0^{L_y} f(x,y)\,\sin\Big(\frac{m\pi x}{L_x}\Big)\sin\Big(\frac{n\pi y}{L_y}\Big)\,dy\,dx.$$

Using $-\Delta\phi_{mn} = \lambda_{mn}\phi_{mn}$ and orthogonality,

$$-\Delta u = \sum_{m,n} U_{mn}\,(-\Delta\phi_{mn}) = \sum_{m,n} U_{mn}\,\lambda_{mn}\,\phi_{mn} = \sum_{m,n} F_{mn}\,\phi_{mn},$$

so, mode-by-mode,

$$\lambda_{mn}\,U_{mn} = F_{mn} \qquad\Longrightarrow\qquad U_{mn} = \frac{F_{mn}}{\lambda_{mn}}.$$

Hence the solution is the *sum of all separated solutions* (i.e., of all eigenmodes) weighted by the data:

$$\boxed{u(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\frac{F_{mn}}{\lambda_{mn}}\,\sin\Big(\frac{m\pi x}{L_x}\Big)\sin\Big(\frac{n\pi y}{L_y}\Big),\quad \lambda_{mn} = \Big(\frac{m\pi}{L_x}\Big)^2 + \Big(\frac{n\pi}{L_y}\Big)^2.}$$

Why this series solves the boundary value problem.

- *Boundary conditions:* Each basis function ϕ_{mn} vanishes on $\partial\Omega$, so the series satisfies $u|_{\partial\Omega} = 0$ termwise.
- *PDE:* Applying $-\Delta$ termwise yields $-\Delta u = \sum F_{mn}\phi_{mn} = f$ (convergence in L^2; with smoother f the convergence is pointwise and uniform on compact subsets).
- *Uniqueness:* The Dirichlet Poisson problem has a unique solution; thus the series equals u.

6.2.4 Worked Example: Piecewise-Constant Forcing on the Unit Square

Let $\Omega = (0,1)\times(0,1)$ and take

$$f(x,y) = \begin{cases} 1, & x \in \left[\frac{1}{4},\frac{1}{2}\right], \quad y \in \left[\frac{1}{2},\frac{3}{4}\right], \\ 0, & \text{otherwise.} \end{cases}$$

The Fourier–sine coefficients factor:

$$F_{mn} = 4\int_{1/4}^{1/2} \sin(m\pi x)\,dx \int_{1/2}^{3/4} \sin(n\pi y)\,dy = \frac{4}{mn\,\pi^2}\left[\cos\Big(\frac{m\pi}{4}\Big) - \cos\Big(\frac{m\pi}{2}\Big)\right]\left[\cos\Big(\frac{n\pi}{2}\Big) - \cos\Big(\frac{3n\pi}{4}\Big)\right].$$

Since on $(0,1)^2$ the eigenvalues are $\lambda_{mn} = (m^2+n^2)\pi^2$, we obtain

$$U_{mn} = \frac{F_{mn}}{\lambda_{mn}} = \frac{4}{mn\,(m^2+n^2)\,\pi^4}\left[\cos\Big(\frac{m\pi}{4}\Big) - \cos\Big(\frac{m\pi}{2}\Big)\right]\left[\cos\Big(\frac{n\pi}{2}\Big) - \cos\Big(\frac{3n\pi}{4}\Big)\right],$$

and the solution is the sine series

$$u(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{4}{mn\,(m^2+n^2)\,\pi^4}\left[\cos\Big(\frac{m\pi}{4}\Big) - \cos\Big(\frac{m\pi}{2}\Big)\right]\left[\cos\Big(\frac{n\pi}{2}\Big) - \cos\Big(\frac{3n\pi}{4}\Big)\right] \sin(m\pi x)\sin(n\pi y).$$

Remarks. (1) The necessity that $-\frac{X''}{X}$ and $-\frac{Y''}{Y}$ be constants follows from independence of x and y; only $\mu > 0$, $\nu > 0$ yield nontrivial Dirichlet modes.
(2) The "sum of all separated solutions" means an L^2-convergent expansion in the complete eigenbasis $\{\phi_{mn}\}$, with coefficients determined by the data.
(3) For nonhomogeneous Dirichlet boundary data, write $u = w + v$ where w matches the boundary values and solve $-\Delta v = f + \Delta w$ with $v = 0$ on $\partial\Omega$; then expand v as above.

6.3 2D Heat Equation with Homogeneous Neumann BCs: Separation of Variables (Detailed)

Problem.

$$\begin{aligned} u_t &= \kappa\,(u_{xx} + u_{yy}) \quad \text{in } (0,1)\times(0,1)\times(0,\infty), \\ u_x(0,y,t) &= u_x(1,y,t) = 0, \qquad u_y(x,0,t) = u_y(x,1,t) = 0, \\ u(x,y,0) &= f(x,y). \end{aligned}$$

6.3.1 Separation Ansatz and Reduction

Assume a product form

$$u(x,y,t) = X(x)\,Y(y)\,T(t).$$

Substitute into the PDE and divide by XYT:

$$\frac{T'(t)}{\kappa\,T(t)} = \frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = -\lambda.$$

Hence there exists a separation constant $\lambda \geq 0$ and

$$\frac{X''}{X} = -\mu, \qquad \frac{Y''}{Y} = -\nu, \qquad \mu + \nu = \lambda.$$

6.3.2 Spatial Eigenproblems with Neumann BCs

From the boundary conditions $\partial_x u = 0$ at $x = 0, 1$ and $\partial_y u = 0$ at $y = 0, 1$, we get

$$X'(0) = X'(1) = 0, \qquad Y'(0) = Y'(1) = 0.$$

Thus X and Y solve the 1D Sturm–Liouville problems

$$X'' + \mu X = 0, \quad X'(0) = X'(1) = 0; \qquad Y'' + \nu Y = 0, \quad Y'(0) = Y'(1) = 0.$$

Solve for X. The general solution is $X(x) = A\cos\big(\sqrt{\mu}\,x\big) + B\sin\big(\sqrt{\mu}\,x\big)$. The Neumann condition at $x = 0$ gives $X'(0) = B\sqrt{\mu} = 0$, hence $B = 0$. The condition at $x = 1$ requires $X'(1) = -A\sqrt{\mu}\sin\big(\sqrt{\mu}\big) = 0$. For nontrivial A, we need $\sin\big(\sqrt{\mu}\big) = 0 \Rightarrow \sqrt{\mu} = m\pi$, $m = 0, 1, 2, \dots$. Therefore

$$X_m(x) = \cos(m\pi x), \qquad \mu_m = (m\pi)^2, \qquad m = 0, 1, 2, \dots$$

(The $m = 0$ mode $X_0 \equiv 1$ is allowed under Neumann BCs.)

Solve for Y. Analogously,

$$Y_n(y) = \cos(n\pi y), \qquad \nu_n = (n\pi)^2, \qquad n = 0, 1, 2, \dots$$

2D eigenpairs.

$$\phi_{mn}(x, y) = \cos(m\pi x)\cos(n\pi y), \qquad \lambda_{mn} = \mu_m + \nu_n = (m^2 + n^2)\pi^2.$$

6.3.3 Temporal Factors

With $u = XYT$ and $\lambda = \lambda_{mn}$,

$$T'_{mn}(t) + \kappa\,\lambda_{mn}\,T_{mn}(t) = 0 \quad \Rightarrow \quad T_{mn}(t) = \begin{cases} e^{-\kappa\,\lambda_{mn}\,t}, & (m, n) \neq (0, 0), \\ 1, & (m, n) = (0, 0) \text{ (mean mode)}. \end{cases}$$

The $(0, 0)$ mode corresponds to $\lambda_{00} = 0$ and carries the conserved spatial mean.

6.3.4 Orthogonality and Coefficients

The cosine basis is orthogonal on $[0, 1]$:

$$\int_0^1 \cos(m\pi x)\cos(p\pi x)\,dx = \begin{cases} 1, & m = p = 0, \\ \frac{1}{2}, & m = p \geq 1, \\ 0, & m \neq p. \end{cases}$$

Hence an L^2 expansion (cosine–cosine series) takes the form

$$\begin{aligned} u(x, y, t) &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} a_{mn}\,e^{-\kappa\,\lambda_{mn} t}\,\phi_{mn}(x, y), \\ a_{mn} &= \alpha_m\,\alpha_n \int_0^1\!\!\int_0^1 f(x, y)\,\cos(m\pi x)\cos(n\pi y)\,dy\,dx, \\ \alpha_0 &= 1, \qquad \alpha_k = 2\ (k \geq 1). \end{aligned}$$

6.3.5 Worked Example: Box Initial Data

Let

$$f(x,y) = \begin{cases} 1, & x \in \left[\frac{1}{4}, \frac{3}{4}\right], \quad y \in \left[\frac{1}{4}, \frac{3}{4}\right], \\ 0, & \text{otherwise.} \end{cases}$$

Separate the coefficient integral:

$$a_{mn} = \alpha_m \alpha_n \, I_m \, J_n, \quad I_m := \int_{1/4}^{3/4} \cos(m\pi x)\, dx, \quad J_n := \int_{1/4}^{3/4} \cos(n\pi y)\, dy.$$

Compute the 1D pieces:

$$I_m = \begin{cases} \frac{1}{2}, & m = 0, \\ \dfrac{\sin\left(\frac{3m\pi}{4}\right) - \sin\left(\frac{m\pi}{4}\right)}{m\pi}, & m \geq 1, \end{cases} \qquad J_n = \begin{cases} \frac{1}{2}, & n = 0, \\ \dfrac{\sin\left(\frac{3n\pi}{4}\right) - \sin\left(\frac{n\pi}{4}\right)}{n\pi}, & n \geq 1. \end{cases}$$

The solution is therefore

$$u(x,y,t) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \alpha_m \alpha_n \, I_m \, J_n \, e^{-\kappa\,\pi^2 (m^2+n^2)\,t} \cos(m\pi x)\cos(n\pi y),$$
$$\text{with the mean mode } a_{00} = I_0 J_0 = \tfrac{1}{4} \text{ (time-independent).}$$

Notes. (1) Neumann BCs admit the constant eigenfunction; the spatial average $\int u$ is conserved. (2) Each nonconstant mode decays like $e^{-\kappa\pi^2(m^2+n^2)t}$, so high frequencies smooth out fastest. (3) The series converges to f in L^2 as $t \downarrow 0$ and is smooth for any $t > 0$.

6.4 2D Wave Equation on $(0,1) \times (0,1)$ by Separation of Variables

Problem. Find $u(x,y,t)$ such that

$$\begin{gathered} u_{tt} = c^2 \left(u_{xx} + u_{yy}\right) \quad \text{in } (0,1) \times (0,1) \times (0,\infty), \\ u = 0 \quad \text{on } \partial\big((0,1) \times (0,1)\big) \text{ for all } t \geq 0, \\ u(x,y,0) = f(x,y), \qquad u_t(x,y,0) = g(x,y). \end{gathered}$$

6.4.1 Separation of variables

Seek product solutions $u(x,y,t) = X(x)Y(y)T(t)$. Substituting and dividing by XYT gives

$$\frac{T''(t)}{c^2\, T(t)} = \frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = -\lambda,$$

for some constant λ. This yields the spatial Dirichlet eigenproblems

$$X'' + \mu X = 0, \quad X(0) = X(1) = 0; \qquad Y'' + \nu Y = 0, \quad Y(0) = Y(1) = 0; \qquad \mu + \nu = \lambda,$$

with nontrivial solutions

$$X_m(x) = \sin(m\pi x), \quad \mu_m = (m\pi)^2; \qquad Y_n(y) = \sin(n\pi y), \quad \nu_n = (n\pi)^2, \quad m,n = 1,2,\ldots$$

so the spatial eigenfunctions and eigenvalues are

$$\phi_{mn}(x,y) = \sin(m\pi x)\sin(n\pi y), \qquad \lambda_{mn} = (m^2+n^2)\pi^2.$$

The temporal factor satisfies

$$T''_{mn}(t) + \omega_{mn}^2\, T_{mn}(t) = 0, \qquad \omega_{mn} = c\,\sqrt{\lambda_{mn}} = c\pi\sqrt{m^2+n^2},$$

so $T_{mn}(t) = A_{mn}\cos(\omega_{mn}t) + B_{mn}\sin(\omega_{mn}t)$.

6.4.2 General series solution

Using the orthogonal basis $\{\phi_{mn}\}$, write

$$u(x,y,t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \big(a_{mn}\cos(\omega_{mn}t) + b_{mn}\sin(\omega_{mn}t)\big)\,\phi_{mn}(x,y).$$

Coefficients are fixed by the initial data using the $L^2(0,1)^2$ inner product. Since

$$\int_0^1\int_0^1 \phi_{mn}\,\phi_{pq}\,dx\,dy = \frac{1}{4}\,\delta_{mp}\delta_{nq},$$

the (unnormalized) sine coefficients are

$$a_{mn} = 4\int_0^1\int_0^1 f(x,y)\,\sin(m\pi x)\sin(n\pi y)\,dy\,dx, \qquad b_{mn} = \frac{4}{\omega_{mn}}\int_0^1\int_0^1 g(x,y)\,\sin(m\pi x)\sin(n\pi y)\,dy\,dx.$$

6.4.3 Worked example: box initial displacement, zero initial velocity

Let

$$f(x,y) = \begin{cases} 1, & x \in \left[\tfrac14,\tfrac34\right],\ y \in \left[\tfrac14,\tfrac34\right], \\ 0, & \text{otherwise}, \end{cases} \qquad g(x,y) \equiv 0.$$

Then

$$a_{mn} = 4\left(\int_{1/4}^{3/4}\sin(m\pi x)\,dx\right)\left(\int_{1/4}^{3/4}\sin(n\pi y)\,dy\right) = \frac{4}{mn\pi^2}\Big[\cos\!\big(\tfrac{m\pi}{4}\big) - \cos\!\big(\tfrac{3m\pi}{4}\big)\Big]\Big[\cos\!\big(\tfrac{n\pi}{4}\big) - \cos\!\big(\tfrac{3n\pi}{4}\big)\Big],$$

and $b_{mn} = 0$ (since $g \equiv 0$). Therefore

$$\boxed{u(x,y,t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{4}{mn\pi^2}\Big[\cos\!\big(\tfrac{m\pi}{4}\big) - \cos\!\big(\tfrac{3m\pi}{4}\big)\Big]\Big[\cos\!\big(\tfrac{n\pi}{4}\big) - \cos\!\big(\tfrac{3n\pi}{4}\big)\Big]\cos\!\big(c\pi\sqrt{m^2+n^2}\,t\big)\,\sin(m\pi x)\sin(n\pi y)}.$$

The series converges pointwise for $t > 0$ and in L^2 at $t = 0$, and enforces $u = 0$ on the boundary termwise.

6.5 Green's Functions for Poisson's Equation: 1D and 2D

6.5.1 What is a Green's Function? (Method Overview)

For a linear operator L on a domain Ω with prescribed boundary conditions (BCs), a *Green's function* $G(x,\xi)$ is a kernel that solves

$$\begin{aligned} L_x\, G(x,\xi) &= \delta(x-\xi) \quad \text{for } x\in\Omega,\\ \text{(BCs in } x) \quad & G(\cdot,\xi) \text{ satisfies the same boundary conditions as the solution.}\end{aligned}$$

Then, for Poisson's equation $Lu = f$ (with appropriate BCs), the solution is represented by

$$u(x) = \int_\Omega G(x,\xi)\, f(\xi)\, d\xi \quad \text{(plus possible boundary terms, which vanish for homogeneous BCs).}$$

6.5.2 1D Poisson on $(0,L)$ with Homogeneous Dirichlet BCs

Problem.

$$\begin{aligned} -u''(x) &= f(x), \quad x\in(0,L),\\ u(0) &= u(L) = 0.\end{aligned}$$

Green's function definition (1D). Seek $G(x,\xi)$ such that

$$\begin{aligned} -\partial_{xx}G(x,\xi) &= \delta(x-\xi), \quad x\in(0,L),\\ G(0,\xi) &= G(L,\xi) = 0.\end{aligned}$$

Construction (piecewise). For fixed $\xi\in(0,L)$, away from $x=\xi$ we have $G_{xx}=0$, so G is linear in x on $(0,\xi)$ and (ξ,L):

$$G(x,\xi) = \begin{cases} A_-(\xi)\,x + B_-(\xi), & 0\le x\le \xi,\\ A_+(\xi)\,x + B_+(\xi), & \xi\le x\le L.\end{cases}$$

Impose:

- Dirichlet BCs: $G(0,\xi)=0 \Rightarrow B_-(\xi)=0$; $\quad G(L,\xi)=0 \Rightarrow A_+(\xi)\,L + B_+(\xi) = 0$.
- Continuity at $x=\xi$: $G(\xi^-,\xi) = G(\xi^+,\xi)$.
- *Jump condition for the derivative.* Integrate $-G_{xx} = \delta(\cdot-\xi)$ across $(\xi-\varepsilon,\xi+\varepsilon)$:

  $$-\int_{\xi-\varepsilon}^{\xi+\varepsilon} G_{xx}\, dx = -\left[G_x\right]_{\xi^-}^{\xi^+} = \int_{\xi-\varepsilon}^{\xi+\varepsilon} \delta(x-\xi)\, dx = 1,$$

  hence

  $$G_x(\xi^+,\xi) - G_x(\xi^-,\xi) = -1.$$

Solving these four conditions gives the explicit Green's function:

$$\boxed{G(x,\xi) = \begin{cases} \dfrac{x\,(L-\xi)}{L}, & 0\le x\le\xi,\\ \dfrac{\xi\,(L-x)}{L}, & \xi\le x\le L,\end{cases} \qquad \text{equivalently} \quad G(x,\xi) = \frac{\min\{x,\xi\}\,\big(L-\max\{x,\xi\}\big)}{L}.}$$

One checks G is continuous at $x=\xi$ and $G_x(\xi^+) - G_x(\xi^-) = -1$, as required.

6.5.3 How does the Greens function work:

Multiply and integrate. Multiply the PDE by $G(\cdot,\xi)$ and integrate over $(0,L)$:

$$\int_0^L \big(-u''(x)\big)\,G(x,\xi)\,dx = \int_0^L f(x)\,G(x,\xi)\,dx.$$

Two integrations by parts. First IBP:

$$\int_0^L (-u'')G\,dx = \Big[-u'(x)\,G(x,\xi)\Big]_0^L + \int_0^L u'(x)\,G_x(x,\xi)\,dx.$$

Second IBP on the remaining integral:

$$\int_0^L u' G_x\,dx = \Big[u(x)\,G_x(x,\xi)\Big]_0^L - \int_0^L u(x)\,G_{xx}(x,\xi)\,dx.$$

Combine:

$$\int_0^L (-u'')G\,dx = \Big[u\,G_x - u'\,G\Big]_0^L - \int_0^L u(x)\,G_{xx}(x,\xi)\,dx.$$

Insert $-G_{xx} = \delta(\cdot-\xi)$.

$$\int_0^L f\,G\,dx = \Big[u\,G_x - u'\,G\Big]_0^L + \int_0^L u(x)\,\delta(x-\xi)\,dx = \Big[u\,G_x - u'\,G\Big]_0^L + u(\xi).$$

General 1D representation (with boundary term).

$$\boxed{u(\xi) = \int_0^L G(x,\xi)\,f(x)\,dx \;-\; \Big[u(x)\,G_x(x,\xi) - u'(x)\,G(x,\xi)\Big]_{x=0}^{x=L}.}$$

Dirichlet specialization. If $u(0) = u(L) = 0$ and $G(0,\xi) = G(L,\xi) = 0$, then the boundary bracket vanishes, yielding

$$\boxed{u(\xi) = \int_0^L G(x,\xi)\,f(x)\,dx.}$$

Remark. For Neumann BCs one uses a Neumann Green's function ($G_x = 0$ at the boundary), which changes the boundary term and requires the usual mean-zero compatibility condition on f.

Representation formula (1D). For $-u'' = f$ with $u(0) = u(L) = 0$,

$$\boxed{u(x) = \int_0^L G(x,\xi)\,f(\xi)\,d\xi.}$$

6.5.4 2D Poisson: $-\Delta u = f$ with Homogeneous Dirichlet BCs

Problem.

$$\begin{aligned} -\Delta u(x) &= f(x), \quad x \in \Omega \subset \mathbb{R}^2,\\ u|_{\partial\Omega} &= 0. \end{aligned}$$

Green's function definition (2D). Find $G(x, \xi)$ such that

$$\begin{aligned} -\Delta_x G(x,\xi) &= \delta(x-\xi), \quad x \in \Omega, \\ G(\cdot,\xi)|_{\partial\Omega} &= 0. \end{aligned}$$

Representation via Green's second identity. Green's second identity states, for smooth u, v,

$$\int_\Omega \big(u\,\Delta v - v\,\Delta u\big)\,dx = \int_{\partial\Omega} \big(u\,\partial_n v - v\,\partial_n u\big)\,ds.$$

Set $v(\cdot) = G(\cdot,\xi)$, use $-\Delta u = f$ and $-\Delta v = \delta(\cdot - \xi)$:

$$\begin{aligned} \int_\Omega \big(u(-\delta) - G(-f)\big)\,dx &= \int_{\partial\Omega} \big(u\,\partial_n G - G\,\partial_n u\big)\,ds, \\ -u(\xi) + \int_\Omega G(x,\xi)\,f(x)\,dx &= \int_{\partial\Omega} \big(u\,\partial_n G - G\,\partial_n u\big)\,ds. \end{aligned}$$

With homogeneous Dirichlet BCs $u|_{\partial\Omega} = 0$ and $G|_{\partial\Omega} = 0$, the boundary integral vanishes, giving

$$\boxed{u(\xi) = \int_\Omega G(x,\xi)\,f(x)\,dx.}$$

Free-space fundamental solution (2D). In $\mathbb{R}^2$, the fundamental solution of $-\Delta$ is

$$\Phi(x-\xi) = -\frac{1}{2\pi}\log|x-\xi| \quad \text{since} \quad -\Delta\Phi = \delta.$$

On a bounded Ω with Dirichlet BCs, the Green's function can be written as

$$G(x,\xi) = \Phi(x-\xi) - H(x,\xi),$$

where, for each fixed ξ, the *harmonic corrector* $H(\cdot,\xi)$ solves

$$\Delta_x H(\cdot,\xi) = 0 \ \text{ in } \Omega, \qquad H(\cdot,\xi) = \Phi(\cdot - \xi) \ \text{ on } \partial\Omega,$$

so that $G(\cdot,\xi)$ vanishes on $\partial\Omega$.

Explicit example (unit disk). If $\Omega = \{x \in \mathbb{R}^2 : |x| < 1\}$, then with $\xi^* = \xi/|\xi|^2$,

$$\boxed{G(x,\xi) = \frac{1}{2\pi}\log\frac{|x-\xi^*|}{|x-\xi|}, \qquad |x|<1,\ |\xi|<1,}$$

and indeed $G(\cdot,\xi) = 0$ on $|x| = 1$.

Eigenfunction series (rectangle). On $\Omega = (0, L_x) \times (0, L_y)$ with Dirichlet BCs, let

$$\phi_{mn}(x,y) = \sin\Big(\frac{m\pi x}{L_x}\Big)\sin\Big(\frac{n\pi y}{L_y}\Big), \quad \lambda_{mn} = \Big(\tfrac{m\pi}{L_x}\Big)^2 + \Big(\tfrac{n\pi}{L_y}\Big)^2.$$

Then

$$\boxed{G(x,\xi) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{\phi_{mn}(x)\,\phi_{mn}(\xi)}{\lambda_{mn}},}$$

and for $-\Delta u = f$,

$$u(x) = \sum_{m,n} \frac{F_{mn}}{\lambda_{mn}}\,\phi_{mn}(x), \qquad F_{mn} = \int_\Omega f(\xi)\,\phi_{mn}(\xi)\,d\xi,$$

which is the Green's representation expressed in the eigenbasis.

6.5.5 Where does Green's identity come from (2D):

Setup and assumptions. Let $\Omega \subset \mathbb{R}^2$ be a bounded region with piecewise C^1 boundary $\partial\Omega$ and outward unit normal $\mathbf{n} = (n_x, n_y)$. Let $u, v \in C^2(\Omega) \cap C^1(\overline{\Omega})$. Write $\nabla u = (u_x, u_y)$ and $\Delta u = u_{xx} + u_{yy}$. Define the normal derivative $\partial_{\mathbf{n}} v := \nabla v \cdot \mathbf{n}$.

A. Vector calculus identities (product rules).

Start from the product rules for divergence:

$$\begin{aligned}
\nabla \cdot \big(u\,\nabla v\big) &= \partial_x\big(u\,v_x\big) + \partial_y\big(u\,v_y\big) \\
&= u_x v_x + u\,v_{xx} + u_y v_y + u\,v_{yy} \\
&= \nabla u \cdot \nabla v \;+\; u\,\Delta v, \\
\nabla \cdot \big(v\,\nabla u\big) &= \nabla v \cdot \nabla u \;+\; v\,\Delta u.
\end{aligned}$$

B. Divergence Theorem.

For any $\mathbf{F} \in C^1(\overline{\Omega}; \mathbb{R}^2)$,

$$\int_\Omega \nabla \cdot \mathbf{F}\,dA = \int_{\partial\Omega} \mathbf{F} \cdot \mathbf{n}\,ds.$$

Apply this with $\mathbf{F}_1 = u\,\nabla v$ and $\mathbf{F}_2 = v\,\nabla u$.

C. Green's first identity ("2D integration by parts").

Using $\mathbf{F}_1 = u\,\nabla v$ and the product rule above,

$$\begin{aligned}
\int_\Omega \big(\nabla u \cdot \nabla v + u\,\Delta v\big)\,dA &= \int_\Omega \nabla \cdot (u\,\nabla v)\,dA \\
&= \int_{\partial\Omega} (u\,\nabla v) \cdot \mathbf{n}\,ds \\
&= \int_{\partial\Omega} u\,\partial_{\mathbf{n}} v\,ds.
\end{aligned}$$

Rearrange to the standard form:

$$\boxed{\int_\Omega \nabla u \cdot \nabla v\,dA = \int_{\partial\Omega} u\,\partial_{\mathbf{n}} v\,ds \;-\; \int_\Omega u\,\Delta v\,dA.}$$

D. Green's second identity.

Do the same with $\mathbf{F}_2 = v\,\nabla u$:

$$\int_\Omega \big(\nabla v \cdot \nabla u + v\,\Delta u\big)\,dA = \int_{\partial\Omega} v\,\partial_{\mathbf{n}} u\,ds.$$

Subtract the two identities (first minus this one):

$$\int_\Omega \big(u\,\Delta v - v\,\Delta u\big)\,dA = \int_{\partial\Omega} \big(u\,\partial_{\mathbf{n}} v - v\,\partial_{\mathbf{n}} u\big)\,ds.$$

Hence

$$\boxed{\int_\Omega \big(u\,\Delta v - v\,\Delta u\big)\,dA = \int_{\partial\Omega} \big(u\,\partial_{\mathbf{n}} v - v\,\partial_{\mathbf{n}} u\big)\,ds.}$$

E. Common specializations.

- **Dirichlet boundary data** $u|_{\partial\Omega} = 0$: Green's first identity reduces to $\int_\Omega \nabla u \cdot \nabla v\,dA = -\int_\Omega u\,\Delta v\,dA$.
- **Neumann boundary data** $\partial_{\mathbf{n}} u|_{\partial\Omega} = 0$: Green's second identity becomes $\int_\Omega (u\,\Delta v - v\,\Delta u)\,dA = \int_{\partial\Omega} u\,\partial_{\mathbf{n}} v\,ds$.

F. Relation to the 1D integration by parts.

In 1D, on (a, b) with outward normals $n(a) = -1$, $n(b) = +1$,

$$\int_a^b u'(x)\,v'(x)\,dx = \big[u\,v'\big]_a^b - \int_a^b u(x)\,v''(x)\,dx,$$

which is exactly Green's first identity with $\partial_{\mathbf{n}} v = v'$ on $\{a, b\}$.

G. (Optional) Green's representation for Poisson.

If $v(\cdot) = G(\cdot, \xi)$ solves $-\Delta_x G = \delta(\cdot - \xi)$, then Green's second identity with Dirichlet data yields

$$u(\xi) = \int_\Omega G(x, \xi)\,(-\Delta u(x))\,dA \; - \int_{\partial\Omega} \big(u\,\partial_{\mathbf{n}} G - G\,\partial_{\mathbf{n}} u\big)\,ds,$$

and for homogeneous Dirichlet boundary conditions the boundary integral vanishes.

Remarks.

- For homogeneous Neumann BCs one uses a Neumann Green's function ($\partial_n G = 0$ on $\partial\Omega$), and the boundary term simplifies differently; solvability requires $\int_\Omega f = 0$ and the solution is defined up to an additive constant.
- The symmetry $G(x, \xi) = G(\xi, x)$ follows from the self-adjointness of $-\Delta$ with the imposed BCs.

7 Approximations to differential operators

We now arrive at the main topic of this course which is how we approximate differ operators with algebraic approximations. At the heart of it this is the critical aspect that we cover in this course which is what makes the solution to many complex systems tractable.

Let's go back to calculus 1 and think about things from the perspective of the definition of a derivative. The definition of a derivative starts by taking a difference of two points on a function operated by distance h then dividing that function by h and taking the limit as h goes to 0. The simplest way to define a difference operator is to drop the limiting process and work with a discrete difference. Here are three examples.

Definition 7.1 (First Difference Approximation)**.** A **difference approximation** replaces derivatives of a smooth function $u(x)$ by discrete operators acting on values of u at grid points. Let the grid spacing be $h > 0$, and consider grid points $x_j = jh$. We define the following **difference operators**:

- **Forward difference operator:**

$$D_+u(x_j) = \frac{u(x_{j+1}) - u(x_j)}{h},$$

  which approximates $u'(x_j)$ using the point to the right.

- **Backward difference operator:**

$$D_-u(x_j) = \frac{u(x_j) - u(x_{j-1})}{h},$$

  which approximates $u'(x_j)$ using the point to the left.

- **Centered difference operator:**

$$D_0u(x_j) = \frac{u(x_{j+1}) - u(x_{j-1})}{2h},$$

  which approximates $u'(x_j)$ using points on both sides.

We can next define a second difference by applying the difference equation to itself. this second difference is the approximation to a second derivative.

Definition 7.2 (Second Differences)**.** Second difference operators approximate second derivatives of a smooth function $u(x)$ by applying first difference operators twice.

- **Forward second difference:**

$$D_+^2u(x_j) = D_+(D_+u(x_j)) = \frac{u(x_{j+2}) - 2u(x_{j+1}) + u(x_j)}{h^2}.$$

- **Backward second difference:**

$$D_-^2u(x_j) = D_-(D_-u(x_j)) = \frac{u(x_j) - 2u(x_{j-1}) + u(x_{j-2})}{h^2}.$$

- **Centered second difference:**

$$D_+D_-u(x_j) = \frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2}.$$

  Equivalently, one can express this as applying the centered operator twice:

$$D_0^2 u(x_j) = D_0(D_0 u(x_j)) = \frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2}.$$

At this point, two fundamental questions arise when we do this

- how good is the approximation we wrote down,
- is this approximation lead to a stable method

much of what we do in this class will be around answering these two question. What will become clear as this class goes on is that it's really easy to write down a method that's unstable, i.e., it blows up. What this really means it's a method you can't compute with. Hence we will need to find ways to analyze these method, even if it is only in simplified settings so that we gain key intuition into the behavior of the numerical method we're working with. On nonlinear problems, we won't be able to fully analyze the equations the way we want, however doing analysis on prototypical examples that are similar to the nonlinear problem we care about will provide great insight into being able to make stable and robust numerical methods for a range of non-linear PDEs.

8 Key tools for approximations

Discrete approximations are going to be something that will need to be able to write down. there are many tools for constructing discrete approximations. We now go over several of these tools before using them to analyze our proposed methods.

The two primary tools are Taylor polynomials and Lagrange polynomials over a collection of points. A Taylor polynomial is an expansion of a function about a point a in terms of derivatives of the function. A Lagrange polynomial is a approximation over a collection of points is a polynomial that passes through. For example a Lagrange polynomial would be an approximation to a function $f(x)$ when it passes through the points $\{(x_0, f(x_0)), (x_1, f(x_1)), \cdots, (x_n, f(x_n))\}$. The spacing of the points of a Lagrange polynomial can change the properties of polynomials accuracy over the interval. For example, a Chebyshev polynomial is a Lagrange polynomial with a special spacing of points that increases the uniformity of the accuracy of the approximation over an interval (see *An Introduction to Numerical Analysis* (Atkinson, 2nd ed., Wiley)). A general Hermit polynomial blends a a Lagrange polynomials and a Taylor polynomials. The idea is to use information about the function and its derivatives over a collection of points.

In general, Taylor polynomials is good for approximating derivatives and Lagrange polynomials are good for approximating integrals. We now define these different approximation methods of after we go over these base methods.

NOTE: there is a companion document that holds up the solution to each of the following three examples in a Jupiter notebook. The companion document is called 8_25_2025_Ploynmial_Approximation.ipynb

NOTE: At the end of this text, appendix B, is an introduction to symbolic manipulation in python. There is a jupyter notebook that goes with this tutorial on Symbolic manipulation in Python. I highly recommend practicing constructing different types

of approximations using this package. For example construct Taylor expansions of a sine function cosine function, exponential function, and then turn them into lambda functions for rapid evaluation and plot the functions you constructed. Then build Lagrange polynomials and Hermit polynomials for the same functions. You need to get good at this your self so that when you ask an AI to help with this, you can find the bugs. 8_25_2025_symbolic_fd_exercises_v2.ipynb

Suggested use of AI - Once you've practiced symbolic manipulation feed your code into the AI to ask for suggestions on other ways to do it. Ask the AI to explain its choices and suggestions. Evaluate whether these are effective ways to do it or if you're better with the way you came up with.

8.1 Taylor polynomials

Taylor's Theorem with Remainder. If f is $(n+1)$-times differentiable on an interval containing a and x, then

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n(x),$$

where the remainder term is

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1},$$

for some ξ between a and x.
Sketch of Proof. Assume that $f \in C^{n+1}$ and apply integration by parts n times to $f(x) - f(a) = \int_a^x \frac{d}{dy} f(y)dy$ and the integral mean value theorem to the last term.

Polynomial Uniqueness Theorem (Numerical Analysis). Suppose $x_0, x_1, \ldots, x_n$ are $n+1$ distinct points in an interval $[a, b]$. Then there exists a *unique polynomial* $P_n(x)$ of degree at most n such that

$$P_n(x_i) = f(x_i), \qquad i = 0, 1, \ldots, n.$$

Sketch of Proof. Existence follows from an explicit construction, such as the Lagrange interpolation formula. For uniqueness, suppose two polynomials $P_n(x)$ and $Q_n(x)$ of degree at most n both interpolate the data. Then their difference $R(x) = P_n(x) - Q_n(x)$ is a polynomial of degree at most n that has $n+1$ distinct roots (at $x_0, x_1, \ldots, x_n$). The only polynomial of degree at most n with $n+1$ distinct roots is the zero polynomial, so $R(x) \equiv 0$. Hence $P_n(x) \equiv Q_n(x)$, proving uniqueness.

As we need to use Rolles thermo form calculus 1 in establishing the validity of the next tool, we state it hear for the reader.

Theorem 8.1 (Rolle's Theorem). *Let $f : [a, b] \to \mathbb{R}$ be continuous on $[a, b]$, differentiable on (a, b), and suppose $f(a) = f(b)$. Then there exists $c \in (a, b)$ such that $f'(c) = 0$.*

8.2 Lagrange polynomials

Polynomial Remainder Theorem (Numerical Analysis). Let f be a function with $n+1$ continuous derivatives on $[a, b]$. Let $P_n(x)$ be the unique interpolating polynomial of degree at most n that satisfies

$$P_n(x_i) = f(x_i), \qquad i = 0, 1, \ldots, n,$$

for distinct interpolation nodes $x_0, x_1, \ldots, x_n$ in $[a, b]$.

Then, for any $x \in [a,b]$, the interpolation error (remainder) is given by

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,(x-x_0)(x-x_1)\cdots(x-x_n),$$

for some ξ between the smallest and largest of the points $\{x_0, x_1, \ldots, x_n, x\}$.

Proof via Rolle's theorem. If x coincides with some node x_j, then $P_n(x) = f(x)$ by interpolation and the formula holds with the product zero; hence assume $x \neq x_i$ for all i.

Define the nodal polynomial

$$\omega_{n+1}(t) := \prod_{i=0}^{n}(t - x_i),$$

and consider the auxiliary function

$$G(t) \;:=\; f(t) - P_n(t) \;-\; \frac{f(x) - P_n(x)}{\omega_{n+1}(x)}\,\omega_{n+1}(t).$$

By construction,

$$G(x_i) = f(x_i) - P_n(x_i) - \frac{f(x) - P_n(x)}{\omega_{n+1}(x)}\,\underbrace{\omega_{n+1}(x_i)}_{=0} = 0 \quad (i = 0, \ldots, n),$$

and also

$$G(x) = f(x) - P_n(x) - \frac{f(x) - P_n(x)}{\omega_{n+1}(x)}\,\omega_{n+1}(x) = 0.$$

Thus G has $n{+}2$ distinct zeros in $[a,b]$. Since $f \in C^{n+1}[a,b]$ and P_n, ω_{n+1} are polynomials, we have $G \in C^{n+1}[a,b]$. Applying Rolle's theorem repeatedly, there exists ξ in the open interval spanned by $\{x_0, \ldots, x_n, x\}$ such that

$$G^{(n+1)}(\xi) = 0.$$

But P_n has degree $\leq n$, so $P_n^{(n+1)} \equiv 0$, and

$$G^{(n+1)}(t) = f^{(n+1)}(t) - \frac{f(x) - P_n(x)}{\omega_{n+1}(x)}\,\omega_{n+1}^{(n+1)}(t).$$

Since ω_{n+1} is a monic polynomial of degree $n{+}1$, we have $\omega_{n+1}^{(n+1)}(t) = (n{+}1)!$ for all t. Evaluating at $t = \xi$ and using $G^{(n+1)}(\xi) = 0$ gives

$$0 \;=\; f^{(n+1)}(\xi) - \frac{f(x) - P_n(x)}{\omega_{n+1}(x)}\,(n{+}1)!,$$

hence

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n{+}1)!}\,\omega_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n{+}1)!}\,\prod_{i=0}^{n}(x - x_i),$$

which is the desired remainder formula. □

Lagrange Interpolation: Given $n+1$ distinct interpolation nodes

$$x_0, x_1, \ldots, x_n$$

with corresponding function values

$$f(x_0), f(x_1), \ldots, f(x_n),$$

the *Lagrange interpolating polynomial* $P_n(x)$ of degree at most n is defined as

$$P_n(x) = \sum_{i=0}^{n} f(x_i)\, L_i(x),$$

where each *Lagrange basis polynomial* $L_i(x)$ is given by

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

These basis polynomials satisfy the property

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases}$$

so that $P_n(x_i) = f(x_i)$ for all $i = 0, 1, \ldots, n$.

8.3 Chebyshev polynomials

Chebyshev Polynomials of the First Kind: The Chebyshev polynomials of the first kind, denoted $T_n(x)$, form a sequence of orthogonal polynomials on $[-1, 1]$. They may be defined in several equivalent ways:

- **Trigonometric definition:**

$$T_n(x) = \cos\big(n \arccos(x)\big), \qquad -1 \leq x \leq 1.$$

- **Recurrence relation:**

$$T_0(x) = 1, \quad T_1(x) = x,$$
$$T_{n+1}(x) = 2x\, T_n(x) - T_{n-1}(x), \qquad n \geq 1.$$

- **Explicit examples:**

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_2(x) = 2x^2 - 1, \quad T_3(x) = 4x^3 - 3x.$$

Key properties:

- *Orthogonality:* $T_m(x)$ and $T_n(x)$ are orthogonal on $[-1, 1]$ with respect to the weight $(1-x^2)^{-1/2}$.
- *Extremal property:* $|T_n(x)| \leq 1$ for $x \in [-1, 1]$.
- *Roots:* The n roots of $T_n(x)$ are

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \qquad k = 1, 2, \ldots, n,$$

which define the Chebyshev nodes used in polynomial interpolation.

8.4 Polynomials via Divided Differences

Sometimes you need to construct a polynomial adaptively as more information is added. that is given a collection of points x_0 to x_n, we need to extend the approximation to include an additional point, x_{n+1}. Divided differences are a way to construct polynomial that introduce this flexibility. At the heart of it, this tool is key in Hyperbolic methods to construct essentially non oscillatory and weighted essentially non oscillatory method.

Divided Differences. Given data points

$$(x_0, f(x_0)),\ (x_1, f(x_1)),\ \ldots,\ (x_n, f(x_n)),$$

the *divided differences* of f are defined recursively as follows:

- **Zeroth-order divided difference:**

$$f[x_i] = f(x_i).$$

- **First-order divided difference:**

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}.$$

- **Higher-order divided differences (for $k \geq 2$):**

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}.$$

These divided differences are used in the Newton form of the interpolating polynomial:

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, \ldots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}).$$

Newton Interpolating Polynomial (constructed with divided differences). Given distinct nodes

$$x_0, x_1, \ldots, x_n$$

with corresponding values $f(x_0), f(x_1), \ldots, f(x_n)$, the Newton interpolating polynomial $P_n(x)$ of degree at most n is defined as

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}],$$

where each coefficient $f[x_i, \ldots, x_j]$ is a *divided difference.*

8.5 Hermite polynomials

Hermite Interpolation. Hermite interpolation generalizes Lagrange interpolation by requiring the polynomial to match both function values and derivatives at given nodes.

Suppose we are given $n + 1$ distinct nodes

$$x_0, x_1, \ldots, x_n,$$

and for each node x_i the values

$$f(x_i),\ f'(x_i),\ f''(x_i),\ \ldots,\ f^{(m_i)}(x_i).$$

The Hermite interpolating polynomial $H(x)$ is the unique polynomial of degree at most

$$N-1, \qquad N=\sum_{i=0}^{n}(m_i+1),$$

such that

$$H^{(k)}(x_i)=f^{(k)}(x_i), \qquad k=0,1,\ldots,m_i,\ \ i=0,1,\ldots,n.$$

Construction: A standard way to construct $H(x)$ is to use a generalized divided difference table. Each node x_i is repeated (m_i+1) times in the table. For example, if at x_0 we require $f(x_0)$ and $f'(x_0)$, then we place x_0, x_0 twice. The divided differences are then defined by

$$f[x_i,x_i]=f'(x_i), \quad f[x_i,x_i,x_i]=\tfrac{1}{2!}f''(x_i), \quad \ldots$$

whenever the denominator vanishes.

Newton form: Once the table is built, the Hermite interpolating polynomial takes the Newton form

$$H(x)=f[x_0]+f[x_0,x_0](x-x_0)+f[x_0,x_0,x_1](x-x_0)^2+\cdots+f[x_0,x_0,\ldots,x_n](x-x_0)(x-x_1)\cdots(x-x_{N-2}).$$

Special cases:

- If $m_i=0$ for all i, Hermite interpolation reduces to *Lagrange interpolation.*
- If all conditions are given at one node x_0, Hermite interpolation reduces to the *Taylor polynomial* at x_0.

9 Definition (Big-O near a point and at infinity)

Before we go on to talk about how we derive finite difference methods, we first need to get some notation out of the way. This section defines something called big O notation. it's an important concept that we go over here and we'll be used to shorthand what will be referred to as remainders. So lets get to it.

Let $f, g : \mathbb{R} \to \mathbb{R}$ be functions and let $a \in \mathbb{R} \cup \{\infty\}$. We write

$$f(x) = O\big(g(x)\big) \quad (x \to a)$$

if there exist constants $C > 0$ and a neighborhood ("eventual domain") of a such that

$$|f(x)| \le C\,|g(x)| \quad \text{whenever } x \text{ is sufficiently close to } a \text{ (for } a \in \mathbb{R}) \text{ or sufficiently large (for } a = \infty).$$

More explicitly:

- If $a \in \mathbb{R}$, then $\exists\, C, \delta > 0$ with $|f(x)| \le C|g(x)|$ for all x such that $0 < |x - a| < \delta$.
- If $a = \infty$, then $\exists\, C > 0, X_0$ with $|f(x)| \le C|g(x)|$ for all $x \ge X_0$.

For a sequence $\{a_n\}$ and $\{b_n\}$, we write $a_n = O(b_n)$ as $n \to \infty$ if $\exists\, C > 0, N \in \mathbb{N}$ such that $|a_n| \le C\,|b_n|$ for all $n \ge N$.

Remarks.

- Big-O provides an *upper bound*; it is not unique: if $f = O(x^3)$ near 0, then also $f = O(x^2)$ near 0 (a weaker bound).
- A sharper statement often uses $o(\cdot)$ (little-o) or $\Theta(\cdot)$ (two-sided bound).

Examples

1. Polynomial growth at infinity

Let $f(x) = 3x^2 + 5x + 7$ and $g(x) = x^2$. Then $f(x) = O(x^2)$ as $x \to \infty$. Indeed, for $x \ge 1$,

$$|f(x)| \le 3x^2 + 5x + 7 \le 3x^2 + 5x^2 + 7x^2 = 15\,x^2.$$

Thus we can take $C = 15$ and $X_0 = 1$.

2. Sine near the origin

We have $\sin x = O(x)$ as $x \to 0$ because the elementary inequality $|\sin x| \le |x|$ holds for all real x. Hence we may take $C = 1$ and any $\delta > 0$.

3. Logarithm Taylor remainder near 0

Consider

$$R_2(x) \;=\; \ln(1 + x) - \left(x - \tfrac{x^2}{2}\right).$$

Then $R_2(x) = O(x^3)$ as $x \to 0$. Proof (with an explicit constant on a concrete domain): for $|x| \le \frac{1}{2}$, the third derivative $f^{(3)}(x) = \frac{2}{(1+x)^3}$ satisfies $|f^{(3)}(x)| \le \frac{2}{(1-\frac{1}{2})^3} = 16$. By Taylor's theorem with remainder,

$$|R_2(x)| \le \frac{\max_{|t|\le|x|} |f^{(3)}(t)|}{3!} |x|^3 \le \frac{16}{6} |x|^3 = \frac{8}{3} |x|^3,$$

so $R_2(x) = O(x^3)$ with $C = \frac{8}{3}$ on $|x| \le \frac{1}{2}$. (Consequently, $R_2(x) = O(x^2)$ also holds, but $O(x^3)$ is sharper.)

4. Exponential near 0

We have $e^x - 1 - x = O(x^2)$ and, more precisely, $O(x^3)$ as $x \to 0$. From the Taylor expansion $e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots$,

$$e^x - 1 - x = \frac{x^2}{2} + O(x^3).$$

Thus on $|x| \le 1$, for example, $|e^x - 1 - x| \le \frac{1}{2}x^2 + \frac{e}{6}|x|^3 \le \left(\frac{1}{2} + \frac{e}{6}\right)x^2$, so we may take $C = \frac{1}{2} + \frac{e}{6}$ showing $O(x^2)$. The sharper $O(x^3)$ follows directly from the remainder term.

5. A sequence bound

Let $a_n = \dfrac{(-1)^n}{n} + \dfrac{1}{n^2}$. Then $a_n = O\left(\frac{1}{n}\right)$ as $n \to \infty$ because

$$|a_n| \le \frac{1}{n} + \frac{1}{n^2} \le \frac{2}{n} \quad \text{for all } n \ge 1.$$

Hence we can take $C = 2$ and $N = 1$.

6. Logarithm grows slower than any power

For any fixed $\alpha > 0$, we have $\ln x = O(x^\alpha)$ as $x \to \infty$. One way to see this is

$$\lim_{x\to\infty} \frac{\ln x}{x^\alpha} = 0 \quad \Rightarrow \quad \exists\, X_0,\, C > 0 : \ \frac{\ln x}{x^\alpha} \le C \text{ for all } x \ge X_0.$$

(For a concrete choice, monotonicity gives $\ln x \le x^\alpha$ for all sufficiently large x; thus we may take $C = 1$ once x exceeds a suitable X_0 depending on α.)

Takeaway. Big-O supplies an eventual upper bound with a constant and a domain (where x is sufficiently close to a point or sufficiently large). When reporting asymptotic accuracy (e.g., in numerical analysis), prefer the sharpest power you can justify, and provide the point of approach (e.g., $x \to 0$ vs. $x \to \infty$).

10 Deriving Finite-Difference Approximation

In this section we get to the nuts and bolts of actually deriving finite difference methods. Just because we can derive an approximation does not mean it's a good approximation. Here we define order of the method, consistency, explain the method of moments, and give examples. it's worth noting that nowadays of the algebra that we do is better done with computers than doing it ourselves. As such there are two companion documents to this section. one is on symbolic manipulation in Python. this walks through the basics of doing symbolic manipulation in the context of a finite difference method. the other package reviews the method of moments in the context of symbolic manipulation within Python. It's really important that you're able to do this by yourself for small examples, but in situations where you want a 10th order finite difference method, I highly recommend using symbolic manipulation. In the context of this class you'll be expected to be able to do this by hand, but symbolic manipulation offers a way to check yourself.

NOTE: are two companion document that holds up the solution to each of the following three examples in a Jupiter notebook. The companion documents are called 8_25_2025_symbolic_fd_exercises_v2.ipynb and 8_25_2025_fd_method_of_moments.ipynb

10.1 Order of a Finite-Difference Approximation

Let f be sufficiently smooth at a point x. Fix a grid spacing $h > 0$ and distinct offsets $s_1, \dots, s_m \in \mathbb{R}$ measured in steps of h. A finite-difference (FD) formula that approximates the d-th derivative of f at x has the form

$$\boxed{f^{(d)}(x) \;\approx\; \frac{1}{h^d}\sum_{j=1}^{m} c_j\, f(x+s_j h),} \tag{10.1}$$

for some coefficients c_j (independent of h). We say the formula *has order* p (or is p-th order accurate) if there exists a constant C_p (depending on the stencil but not on h) such that, as $h \to 0$,

$$f^{(d)}(x) - \frac{1}{h^d}\sum_{j=1}^{m} c_j\, f(x+s_j h) \;=\; C_p\, h^p\, f^{(d+p)}(x) \;+\; \mathcal{O}(h^{p+1}). \tag{10.2}$$

The number p is the *order*, and C_p is the *leading truncation-error constant.*

10.2 The Method of Moments (Moment Matching)

The method of moments derives the c_j by forcing (10.1) to be *exact* on polynomials up to a certain degree. Expand each sample by Taylor's theorem:

$$f(x+s_j h) = \sum_{k\ge 0} \frac{f^{(k)}(x)}{k!}\,(s_j h)^k. \tag{10.3}$$

Insert (10.3) into (10.1) and collect by derivatives $f^{(k)}(x)$:

$$\frac{1}{h^d}\sum_{j=1}^{m} c_j f(x+s_j h) = \sum_{k\ge 0}\left(\frac{1}{k!}\sum_{j=1}^{m} c_j s_j^k\right) h^{k-d}\, f^{(k)}(x).$$

To make the right-hand side reproduce $f^{(d)}(x)$ while cancelling all lower-degree terms, we enforce the *moment conditions*

$$\boxed{\sum_{j=1}^{m} c_j\, s_j^k \;=\; \begin{cases} 0, & k = 0, 1, \ldots, m-1,\ k \neq d, \\ d!, & k = d, \end{cases}} \tag{10.4}$$

i.e., exactness on all monomials x^k up to degree $m-1$ (except the derivative-defining row $k = d$). This yields an $m \times m$ linear system for $c = (c_1, \ldots, c_m)^\top$:

$$A\,c = b, \qquad A_{kj} = s_j^k, \quad k = 0, \ldots, m-1, \quad j = 1, \ldots, m, \qquad b_k = \begin{cases} d!, & k = d, \\ 0, & k \neq d. \end{cases} \tag{10.5}$$

The matrix A is Vandermonde-like in the offsets.

Order and leading error from higher moments. By construction, (10.4) annihilates all contributions through $k = m-1$. Let $k^\star \geq m$ be the smallest power for which the *higher moment* $\sum_j c_j s_j^{k^\star} \neq 0$. Then from the Taylor bookkeeping above,

$$\text{TE} \;\equiv\; f^{(d)}(x) - \frac{1}{h^d}\sum_{j=1}^{m} c_j f(x + s_j h) \;=\; \frac{\sum_j c_j s_j^{k^\star}}{k^\star!}\, h^{\,k^\star - d}\, f^{(k^\star)}(x) \;+\; \mathcal{O}(h^{\,k^\star - d + 1}). \tag{10.6}$$

Hence the *order* is

$$\boxed{p = k^\star - d}$$

and the *leading TE constant* is

$$\boxed{C_p = \frac{\sum_j c_j s_j^{k^\star}}{k^\star!}.}$$

Symmetry heuristics (useful in practice). If the offsets are symmetric, $s_j \in \{-r, \ldots, 0, \ldots, r\}$, then

- for an *odd* derivative (d odd), the solution c is antisymmetric and $c_{\text{center}} = 0$;
- for an *even* derivative (d even), the solution c is symmetric.

This makes many odd/even higher moments vanish automatically, often boosting the order.

10.3 Worked Examples (Algebra Included)

Throughout, the factor $1/h^d$ is understood as in (10.1).

Example 1: 3-point second derivative (central, 2nd order)

Offsets $s = \{-1, 0, 1\}$, target $d = 2$ ($m = 3$). The system (10.5) is

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{-1} \\ c_0 \\ c_{+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2! \end{bmatrix}.$$

Solving gives $c = \begin{bmatrix}1, & -2, & 1\end{bmatrix}^\top$. Thus

$$f''(x) \;\approx\; \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}.$$

The first nonzero higher moment is at $k^\star = 4$: $\sum c_j s_j^4 = 2$, hence

$$\mathrm{TE} \;=\; \frac{2}{4!}\, h^{4-2} f^{(4)}(x) + \cdots \;=\; \frac{1}{12}\, h^2 f^{(4)}(x) + \cdots,$$

so $p = 2$ and $C_p = \frac{1}{12}$.

Example 2: 5-point first derivative (central, 4th order)

Offsets $s = \{-2, -1, 0, 1, 2\}$, target $d = 1$ ($m = 5$). The moment equations for $k = 0, \ldots, 4$ are:

$$\begin{aligned} &k=0: \;\sum c_j = 0, \qquad && k=1: \;\sum c_j s_j = 1, \\ &k=2: \;\sum c_j s_j^2 = 0, \qquad && k=3: \;\sum c_j s_j^3 = 0, \qquad && k=4: \;\sum c_j s_j^4 = 0. \end{aligned}$$

Solving gives

$$c = \begin{bmatrix} \tfrac{1}{12}, & -\tfrac{2}{3}, & 0, & \tfrac{2}{3}, & -\tfrac{1}{12} \end{bmatrix}^\top,$$

hence

$$f'(x) \;\approx\; \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12\,h}.$$

The smallest $k^\star \geq 5$ with nonzero higher moment is $k^\star = 5$, $\sum c_j s_j^5 = -4$, so

$$\mathrm{TE} \;=\; \frac{-4}{5!}\, h^4 f^{(5)}(x) + \cdots \;=\; -\frac{1}{30}\, h^4 f^{(5)}(x) + \cdots,$$

i.e., $p = 4$ with $C_p = -\frac{1}{30}$.

Example 3: 4-point one-sided first derivative (forward, 3rd order)

Offsets $s = \{0, 1, 2, 3\}$, target $d = 1$ ($m = 4$). The system for $k = 0, 1, 2, 3$ is

$$\begin{aligned} k=0&: \; c_0 + c_1 + c_2 + c_3 = 0, \\ k=1&: \; 0\cdot c_0 + 1\cdot c_1 + 2c_2 + 3c_3 = 1, \\ k=2&: \; 0\cdot c_0 + 1\cdot c_1 + 4c_2 + 9c_3 = 0, \\ k=3&: \; 0\cdot c_0 + 1\cdot c_1 + 8c_2 + 27c_3 = 0. \end{aligned}$$

Solving yields

$$c = \begin{bmatrix} -\tfrac{11}{6}, & 3, & -\tfrac{3}{2}, & \tfrac{1}{3} \end{bmatrix}^\top,$$

so

$$f'(x) \;\approx\; \frac{-\frac{11}{6} f(x) + 3f(x+h) - \frac{3}{2} f(x+2h) + \frac{1}{3} f(x+3h)}{h}.$$

The first nonzero higher moment occurs at $k^\star = 3$ with $\sum c_j s_j^3 = 1$, so

$$\mathrm{TE} \;=\; \frac{1}{3!}\, h^2 f^{(3)}(x) + \cdots \;=\; \frac{1}{6}\, h^2 f^{(3)}(x) + \cdots.$$

(Equivalently, one may report the constant via the usual forward-difference normalization; conventions differ by sign depending on layout.)

Example 4: 5-point second derivative (central, 4th order)

Offsets $s = \{-2, -1, 0, 1, 2\}$, target $d = 2$ $(m = 5)$. Solving (10.5) gives

$$c = \left[-\tfrac{1}{12},\ \tfrac{4}{3},\ -\tfrac{5}{2},\ \tfrac{4}{3},\ -\tfrac{1}{12}\right]^\top,$$

hence

$$f''(x) \ \approx\ \frac{-f(x-2h) + 16f(x-h) - 30f(x) + 16f(x+h) - f(x+2h)}{12\,h^2}.$$

The smallest higher moment is at $k^\star = 6$ with $\sum c_j s_j^6 = -8$, so

$$\mathrm{TE} \ =\ \frac{-8}{6!}\,h^4 f^{(6)}(x) + \cdots \ =\ -\frac{1}{90}\,h^4 f^{(6)}(x) + \cdots\,,$$

i.e., $p = 4$ and $C_p = -\frac{1}{90}$.

Example 5: Nonuniform first derivative (three points, 2nd order)

Offsets $s = \{0, \frac{1}{2}, \frac{3}{2}\}$, target $d = 1$ $(m = 3)$. Solve (10.5):

$$c = \left[-\tfrac{8}{3},\ 3,\ -\tfrac{1}{3}\right]^\top,$$

so

$$f'(x) \ \approx\ \frac{-\frac{8}{3}f(x) + 3f\big(x+\frac{h}{2}\big) - \frac{1}{3}f\big(x+\frac{3h}{2}\big)}{h}.$$

Here $k^\star = 3$ with $\sum c_j s_j^3 = -\frac{3}{4}$, hence

$$\mathrm{TE} \ =\ \frac{-\frac{3}{4}}{3!}\,h^2 f^{(3)}(x) + \cdots \ =\ -\frac{1}{8}\,h^2 f^{(3)}(x) + \cdots\,,$$

so $p = 2$ with $C_p = -\frac{1}{8}$ even on this nonuniform stencil.

10.4 Practical Recipe

Given desired derivative order d and offsets $s_1, \ldots, s_m$:

1. Build $A \in \mathbb{R}^{m\times m}$ with $A_{kj} = s_j^k$ for $k = 0, \ldots, m-1$.
2. Set $b \in \mathbb{R}^m$ with $b_d = d!$ and $b_k = 0$ for $k \neq d$.
3. Solve $Ac = b$ for $c = (c_1, \ldots, c_m)^\top$.
4. Determine order p and C_p by scanning higher moments $\sum_j c_j s_j^k$ for $k \geq m$ until the first nonzero at $k^\star$, and use (10.6).

Remarks.

- The method works identically for one-sided and nonuniform stencils.
- On symmetric stencils, parity arguments often predict zeros in c and in higher moments, explaining why high order is achievable with relatively few points.
- In implementation, it is common to compute c symbolically (to rational form) once, then hard-code those coefficients in a numerical kernel.

10.5 Consistency of Finite-Difference Schemes

Definition: Local Truncation Error (LTE)

Let L be a differential operator (e.g. $L = \frac{d}{dx}$ or $L = \frac{d^2}{dx^2}$) and let L_h be a finite difference (FD) operator intended to approximate L at mesh point x_i. The *local truncation error* at x_i when the exact (smooth) f is inserted into L_h is

$$\tau_h[f](x_i) \;:=\; \big(L_h f\big)_i \;-\; \big(Lf\big)(x_i).$$

Definition: Consistency and Order

A family $\{L_h\}$ is *consistent* with L (at x_i) if, for every sufficiently smooth f,

$$\tau_h[f](x_i) \xrightarrow[h\to 0]{} 0.$$

It is *p-th order consistent* if there exists a constant C (independent of h but depending on bounds of higher derivatives of f) such that

$$|\tau_h[f](x_i)| \;\le\; C\,h^p \qquad \text{for all sufficiently small } h.$$

Taylor tools

We will repeatedly use the Taylor expansions at x_i:

$$f(x_i \pm h) = f_i \pm hf_i' + \tfrac{h^2}{2}f_i'' \pm \tfrac{h^3}{6}f_i^{(3)} + \tfrac{h^4}{24}f_i^{(4)} \pm \tfrac{h^5}{120}f_i^{(5)} + \mathcal{O}(h^6),$$
$$f(x_i \pm 2h) = f_i \pm 2hf_i' + \tfrac{(2h)^2}{2}f_i'' \pm \tfrac{(2h)^3}{6}f_i^{(3)} + \tfrac{(2h)^4}{24}f_i^{(4)} \pm \tfrac{(2h)^5}{120}f_i^{(5)} + \mathcal{O}(h^6).$$

10.5.1 First derivative $f'(x)$: forward, backward, and central

Forward difference (one-sided, order $\mathcal{O}(h)$).

$$(D^+f)_i := \frac{f_{i+1} - f_i}{h}.$$

Using Taylor at x_i for f_{i+1},

$$(D^+f)_i = \frac{f_i + hf_i' + \frac{h^2}{2}f_i'' + \frac{h^3}{6}f_i^{(3)} + \cdots - f_i}{h} = f_i' + \frac{h}{2}f_i'' + \frac{h^2}{6}f_i^{(3)} + \mathcal{O}(h^3).$$

Thus

$$\boxed{\tau_h[f](x_i) = (D^+f)_i - f_i' = \frac{h}{2}f_i'' + \mathcal{O}(h^2)} \quad \Rightarrow \quad \text{consistent of order } p = 1.$$

Backward difference (one-sided, order $\mathcal{O}(h)$).

$$(D^-f)_i := \frac{f_i - f_{i-1}}{h}.$$

Using Taylor for f_{i-1},

$$(D^-f)_i = \frac{f_i - \big(f_i - hf_i' + \frac{h^2}{2}f_i'' - \frac{h^3}{6}f_i^{(3)} + \cdots\big)}{h} = f_i' - \frac{h}{2}f_i'' + \frac{h^2}{6}f_i^{(3)} + \mathcal{O}(h^3),$$

so

$$\boxed{\tau_h[f](x_i) = (D^-f)_i - f_i' = -\frac{h}{2}f_i'' + \mathcal{O}(h^2)} \quad \Rightarrow \quad p = 1.$$

Central difference (order $\mathcal{O}(h^2)$).

$$(D^0 f)_i := \frac{f_{i+1} - f_{i-1}}{2h}.$$

Subtract Taylor series for f_{i+1} and f_{i-1}:

$$(D^0 f)_i = \frac{\big(f_i + hf_i' + \frac{h^2}{2} f_i'' + \frac{h^3}{6} f_i^{(3)} + \cdots\big) - \big(f_i - hf_i' + \frac{h^2}{2} f_i'' - \frac{h^3}{6} f_i^{(3)} + \cdots\big)}{2h} = f_i' + \frac{h^2}{6} f_i^{(3)} + \mathcal{O}(h^4),$$

hence

$$\boxed{\tau_h[f](x_i) = (D^0 f)_i - f_i' = \frac{h^2}{6} f_i^{(3)} + \mathcal{O}(h^4)} \quad \Rightarrow \quad p = 2.$$

10.5.2 Second derivative $f''(x)$: forward, backward, and central

Central second difference (order $\mathcal{O}(h^2)$).

$$(\Delta^0 f)_i := D^+ D^-(f_i) = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}.$$

Add Taylor series for $f_{i\pm1}$:

$$(\Delta^0 f)_i = \frac{\big(f_i + hf_i' + \frac{h^2}{2} f_i'' + \frac{h^3}{6} f_i^{(3)} + \frac{h^4}{24} f_i^{(4)} + \cdots\big) - 2f_i + \big(f_i - hf_i' + \frac{h^2}{2} f_i'' - \frac{h^3}{6} f_i^{(3)} + \frac{h^4}{24} f_i^{(4)} + \cdots\big)}{h^2}.$$

Terms in f_i and f_i' cancel, leaving

$$(\Delta^0 f)_i = f_i'' + \frac{h^2}{12} f_i^{(4)} + \mathcal{O}(h^4),$$

so

$$\boxed{\tau_h[f](x_i) = (\Delta^0 f)_i - f_i'' = \frac{h^2}{12} f_i^{(4)} + \mathcal{O}(h^4)} \quad \Rightarrow \quad p = 2.$$

Forward second difference (one-sided, order $\mathcal{O}(h)$). A standard forward stencil uses x_i, x_{i+1}, x_{i+2}:

$$(\Delta^+ f)_i := D^+ D^+(f_i) = \frac{f_i - 2f_{i+1} + f_{i+2}}{h^2}.$$

Insert Taylor expansions for f_{i+1}, f_{i+2}; after cancellation one finds

$$(\Delta^+ f)_i = f_i'' + h\, f_i^{(3)} + \frac{7}{12} h^2 f_i^{(4)} + \mathcal{O}(h^3),$$

hence

$$\boxed{\tau_h[f](x_i) = (\Delta^+ f)_i - f_i'' = h\, f_i^{(3)} + \mathcal{O}(h^2)} \quad \Rightarrow \quad p = 1.$$

Backward second difference (one-sided, order $\mathcal{O}(h)$). Using x_i, x_{i-1}, x_{i-2}:

$$(\Delta^- f)_i := D^- D^-(f_i) = \frac{f_{i-2} - 2f_{i-1} + f_i}{h^2}.$$

Taylor expansion about x_i yields

$$(\Delta^- f)_i = f_i'' - h\, f_i^{(3)} + \frac{7}{12} h^2 f_i^{(4)} + \mathcal{O}(h^3),$$

so

$$\boxed{\tau_h[f](x_i) = (\Delta^- f)_i - f_i'' = -\,h\, f_i^{(3)} + \mathcal{O}(h^2)} \quad \Rightarrow \quad p = 1.$$

10.5.3 Summary of consistency orders (uniform grid)

Operator	Stencil	LTE (leading term)
$f'(x)$	forward $(f_{i+1} - f_i)/h$	$\frac{h}{2} f''(x_i) + \mathcal{O}(h^2)$
$f'(x)$	backward $(f_i - f_{i-1})/h$	$-\frac{h}{2} f''(x_i) + \mathcal{O}(h^2)$
$f'(x)$	central $(f_{i+1} - f_{i-1})/(2h)$	$\frac{h^2}{6} f^{(3)}(x_i) + \mathcal{O}(h^4)$
$f''(x)$	forward $(f_i - 2f_{i+1} + f_{i+2})/h^2$	$h\, f^{(3)}(x_i) + \mathcal{O}(h^2)$
$f''(x)$	backward $(f_{i-2} - 2f_{i-1} + f_i)/h^2$	$-h\, f^{(3)}(x_i) + \mathcal{O}(h^2)$
$f''(x)$	central $(f_{i+1} - 2f_i + f_{i-1})/h^2$	$\frac{h^2}{12} f^{(4)}(x_i) + \mathcal{O}(h^4)$

Conclusion. Each stencil is *consistent* because its LTE $\to 0$ as $h \to 0$. One-sided first/second derivatives are first-order accurate; central stencils improve the order to second order via cancellation of odd/even Taylor terms.

11 Tools for analysis of matrices that arise in finite difference methods.

We will need two key ideas to analyze discretizations of Boundary Value Problems, the idea of Consistency and Stability. Stability is related to truncation error, and Consistency is related to know if an inverse of the finite difference approximation exists independent of h, as $h \to 0$. For basic finite difference methods, to address this second question, we will want to make use of matrix norms.

To properly analyze finite difference methods for boundary value problems or implicit equations we will need to analyze matrices. In particular we'll need to be able to say something about the measure of the inverse of a matrix. In mathematics, a measure is an unsigned distance. In the context of linear algebra the measure is in terms of the size of a vector in some distance metric that is defined. There are infinitely many valid distance metrics, often we will focus on the classic 3 known as the L_∞ norm, the L_1 norm and the L_2 norm. The L_2 norm is simply the discrete form of Euclidean distance, the L_1 is sums of absolute values, and the L_∞ is max of the absolute values. We use these measures of distance to look at bounding the inverse of the matrix in a way that is independent of the discretization parameter h. before we go through the process of defining discretizations of boundary value problems and writing out the matrices and then trying to understand stability we're going to review the basics of norms. Readers who know this material can skip ahead to the next section. **A good book on linear algebra can be an essential aid when studying stability of finite difference methods.**

Vector and Matrix Norms: Definitions, Classical Examples, Norm Equivalence, Induced Matrix Norms, and Symmetric-Matrix Relations

11.1 Vector norms

Let V be a real or complex vector space.

Definition 11.1 (Vector norm)**.** A map $\|\cdot\|: V \to [0,\infty)$ is a *norm* if for all $x, y \in V$ and scalars α:

(N1) **Positive-definiteness:** $\|x\| = 0 \iff x = 0$.

(N2) **Absolute homogeneity:** $\|\alpha x\| = |\alpha|\|x\|$.

(N3) **Triangle inequality:** $\|x+y\| \le \|x\| + \|y\|$.

On $\mathbb{R}^n$ (or $\mathbb{C}^n$), three classical norms are:

$$\|x\|_1 := \sum_{i=1}^n |x_i|, \qquad \|x\|_2 := \Big(\sum_{i=1}^n |x_i|^2\Big)^{1/2}, \qquad \|x\|_\infty := \max_{1\le i\le n} |x_i|.$$

(N1) and (N2) are immediate; (N3) follows from Minkowski (for $p = 1, 2$) and from $\max_i |a_i + b_i| \le \max_i |a_i| + \max_i |b_i|$ for $p = \infty$.

Basic inequalities (explicit constants). For all $x \in \mathbb{R}^n$,

$$\|x\|_\infty \le \|x\|_2 \le \|x\|_1 \le \sqrt{n}\,\|x\|_2 \le n\,\|x\|_\infty. \tag{11.1}$$

Proofs: $\|x\|_2^2 = \sum |x_i|^2 \ge \max_i |x_i|^2 = \|x\|_\infty^2$. Also $\|x\|_1 = \sum |x_i| \ge \sqrt{\sum |x_i|^2} = \|x\|_2$ by Cauchy–Schwarz with $1^\top x$. Next, $\|x\|_1 \le \sqrt{n}\,\|x\|_2$ by Cauchy–Schwarz; finally $\|x\|_2 \le \sqrt{n}\,\|x\|_\infty$.

11.2 Norm equivalence on finite-dimensional spaces

Theorem 11.2 (Norm equivalence)**.** *On any finite-dimensional vector space V (e.g. $\mathbb{R}^n$), all norms are equivalent: if $\|\cdot\|_a, \|\cdot\|_b$ are norms on V, then*

$$\exists\, m, M > 0 \ \text{ such that } \ m\,\|x\|_a \le \|x\|_b \le M\,\|x\|_a \quad \forall x \in V.$$

Proof via compactness. Consider the unit sphere $S_a = \{x : \|x\|_a = 1\}$, which is compact in finite dimensions. The function $x \mapsto \|x\|_b$ is continuous, hence attains its minimum m and maximum M on S_a, with $0 < m \le M < \infty$ since $\|x\|_b > 0$ on S_a. For any $x \neq 0$, write $x = \|x\|_a \cdot (x/\|x\|_a)$ with $x/\|x\|_a \in S_a$ to obtain $m\|x\|_a \le \|x\|_b \le M\|x\|_a$. □

In $\mathbb{R}^n$, the explicit inequalities (11.1) give simple constants between $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$.

11.3 Matrix norms induced by vector norms

Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. The *induced* (or *subordinate*) matrix norm on $\mathbb{R}^{n\times n}$ is

$$\|A\| := \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|.$$

Then:

- **Consistency:** $\|Ax\| \le \|A\|\,\|x\|$, $\|AB\| \le \|A\|\,\|B\|$, and $\|I\| = 1$.
- **Spectral radius bound:** For any induced norm, $\rho(A) \le \|A\|$, where $\rho(A) = \max_i |\lambda_i(A)|$.

Three classical induced matrix norms (with reductions)

For $A = (a_{ij}) \in \mathbb{R}^{n\times n}$:

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}| \qquad \text{(max column sum)},$$

$$\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}| \qquad \text{(max row sum)},$$

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^\top A)} \qquad \text{(spectral norm)}.$$

Proof of the 1-norm reduction. For any x, $\|Ax\|_1 = \sum_i \left|\sum_j a_{ij}x_j\right| \le \sum_{i,j} |a_{ij}|\,|x_j| = \sum_j \Big(\sum_i |a_{ij}|\Big)|x_j| \le \Big(\max_j \sum_i |a_{ij}|\Big)\|x\|_1$. Hence $\|A\|_1 \le \max_j \sum_i |a_{ij}|$. Equality is attained by choosing $x = e_{j^\star}$ sgn with all mass on a maximizing column $j^\star$ and signs matching $a_{ij^\star}$ so the inequalities become equalities. □

Proof of the ∞-norm reduction. For any x, $\|Ax\|_\infty = \max_i \left|\sum_j a_{ij}x_j\right| \le \max_i \sum_j |a_{ij}|\,|x_j| \le \Big(\max_i \sum_j |a_{ij}|\Big)\|x\|_\infty$. Take x with entries $x_j = \operatorname{sgn}(a_{i^\star j})$ for a maximizing row $i^\star$ to get equality. □

Proof of the 2-norm formula. By definition, $\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2 = \sup_{\|x\|_2=1} \sqrt{x^\top A^\top Ax} = \sqrt{\sup_{\|x\|_2=1} x^\top A^\top Ax}$. Since $A^\top A$ is symmetric positive semidefinite, the Rayleigh–Ritz theorem yields $\sup_{\|x\|_2=1} x^\top A^\top Ax = \lambda_{\max}(A^\top A)$. □

11.4 Spectral (Orthogonal) Diagonalization

A special case that happens in finite difference methods is that the matrix A that will arise is a symmetric matrix. this can greatly simplify the analysis. To understand that we briefly go over symmetric matrices and spectral (orthogonal) diagonalization.

Let $A \in \mathbb{R}^{n\times n}$ be *symmetric* ($A = A^\top$). The spectral theorem states that

$$A \;=\; Q\,\Lambda\,Q^\top,$$

where

- $Q \in \mathbb{R}^{n\times n}$ is **orthogonal**: $Q^\top Q = QQ^\top = I$. Its columns form an orthonormal basis of eigenvectors of A.
- $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ is **diagonal** with the (real) eigenvalues of A on the diagonal.

If A is symmetric positive definite (SPD), then $\lambda_i > 0$ for all i.

Complex analogue. For Hermitian $A \in \mathbb{C}^{n\times n}$, one has $A = U\,\Lambda\,U^*$ with U unitary ($U^*U = I$) and Λ real diagonal.

Contrasts with other factorizations.

- **SVD**: $A = U\,\Sigma\,V^\top$ (or V^*); holds for any A. $\Sigma \geq 0$ contains *singular values*, not eigenvalues in general.
- **Cholesky** (SPD only): $A = R^\top R$ (or $LL^\top$); not an eigendecomposition.
- **General diagonalization**: $A = V\,\Lambda\,V^{-1}$ (if A is diagonalizable); V need not be orthogonal.

Tiny example.

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \qquad Q = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix},$$

and indeed $A = Q\,\Lambda\,Q^\top$.

11.5 Symmetric matrices: eigenvalues, inverses, and spectral radius

In this section "symmetric" means $A = A^\top$ (real case). By the spectral theorem, $A = Q\Lambda Q^\top$ with Q orthogonal and $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ real.

Theorem 11.3 (Key relations for symmetric A)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric.*

(S1) ***Spectral norm equals largest eigenvalue magnitude:*** $\|A\|_2 = \max_i |\lambda_i|$.

(S2) ***Inverse eigenvalues:*** *If A is invertible, then $A^{-1} = Q\Lambda^{-1}Q^\top$ and the eigenvalues of A^{-1} are $1/\lambda_i$. Consequently $\|A^{-1}\|_2 = 1/\min_i |\lambda_i|$.*

(S3) ***Condition number (2-norm):*** $\kappa_2(A) := \|A\|_2\|A^{-1}\|_2 = \dfrac{\max_i |\lambda_i|}{\min_i |\lambda_i|}$ *(well-defined if A is invertible).*

(S4) ***Powers:*** $\|A^k\|_2 = (\|A\|_2)^k = (\max_i |\lambda_i|)^k$ *for $k \in \mathbb{N}$.*

(S5) ***Spectral radius bound:*** *For any induced norm, $\rho(A) \le \|A\|$; for symmetric A, $\rho(A) = \|A\|_2$.*

Proof. (S1) Using the spectral theorem and orthogonal invariance of $\|\cdot\|_2$, $\|A\|_2 = \|Q\Lambda Q^\top\|_2 = \|\Lambda\|_2 = \max_i |\lambda_i|$. For Λ, this is immediate from the definition. (S2) follows from $A^{-1} = Q\Lambda^{-1}Q^\top$. (S3) is immediate from (S1)–(S2). (S4) From $A^k = Q\Lambda^k Q^\top$ and (S1). (S5) For $\rho(A) \le \|A\|$: pick an eigenpair $Av = \lambda v$ with $v \ne 0$ and any induced norm; then $\|A\| \ge \dfrac{\|Av\|}{\|v\|} = \dfrac{\|\lambda v\|}{\|v\|} = |\lambda|$, hence $\rho(A) \le \|A\|$. For symmetric A, combine with (S1). □

Rayleigh quotient bounds (SPD case). If A is symmetric positive definite (SPD), then for all $x \ne 0$,

$$\lambda_{\min} \le \frac{x^\top A x}{x^\top x} \le \lambda_{\max},$$

and with $y = Ax$ also

$$\frac{1}{\lambda_{\max}} \le \frac{x^\top A^{-1} x}{x^\top x} \le \frac{1}{\lambda_{\min}}.$$

These imply $\|A\|_2 = \lambda_{\max}$ and $\|A^{-1}\|_2 = 1/\lambda_{\min}$, and for the error/residual relations often used in numerical linear algebra.

11.6 Worked examples (symmetric matrices)

We compute $\|A\|_1, \|A\|_\infty, \|A\|_2$, eigenvalues, $\rho(A)$, and (when invertible) $\|A^{-1}\|_2$ and $\kappa_2(A)$.

Example 1 (Diagonal)

$A = \operatorname{diag}(3, -1, 2)$. Then

$$\|A\|_1 = \max\{3,1,2\} = 3, \quad \|A\|_\infty = \max\{3,1,2\} = 3, \quad \|A\|_2 = \max\{3,1,2\} = 3.$$

Eigenvalues are $(3, -1, 2)$, so $\rho(A) = 3$. A invertible with $A^{-1} = \operatorname{diag}(1/3, -1, 1/2)$; hence $\|A^{-1}\|_2 = \max\{1/3, 1, 1/2\} = 1$, and $\kappa_2(A) = 3$.

Example 2 (Permutation/flip)

$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Column sums and row sums are 1, so $\|A\|_1 = \|A\|_\infty = 1$. $A^\top A = I$, hence $\|A\|_2 = \sqrt{\lambda_{\max}(I)} = 1$. Eigenvalues are $\{1, -1\}$, so $\rho(A) = 1$; $A^{-1} = A$ and $\|A^{-1}\|_2 = 1$, $\kappa_2(A) = 1$.

Example 3 (Tridiagonal SPD)

$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$. Row/column sums in absolute value are 3, so $\|A\|_1 = \|A\|_\infty = 3$. Eigenvalues solve $\det(A - \lambda I) = 0$, giving $\lambda \in \{1, 3\}$. Thus $\|A\|_2 = \lambda_{\max} = 3$, $\rho(A) = 3$. Inverse $A^{-1} = \frac{1}{3}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$, so $\|A^{-1}\|_2 = 1/\lambda_{\min} = 1$ and $\kappa_2(A) = 3$.

Example 4 (Mixed signs, invertible symmetric)

$A = \begin{bmatrix} 4 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 2 \end{bmatrix}$ (SPD). Absolute column sums: $(5, 5, 3)$ so $\|A\|_1 = 5$; row sums: $(5, 5, 3)$ so $\|A\|_\infty = 5$.
Eigenvalues are positive (SPD); by Gershgorin, they lie in $\cup_i B(a_{ii}, R_i) = B(4, 1) \cup B(3, 2) \cup B(2, 1)$, hence in $[1, 5]$. Therefore $1 \le \lambda_{\min} \le \lambda_{\max} \le 5$, giving $1 \le \|A\|_2 \le 5$; more precisely, a direct computation gives $\lambda_{\max} \approx 4.246$, hence $\|A\|_2 \approx 4.246$, $\|A^{-1}\|_2 = 1/\lambda_{\min} \approx 1/1.554 \approx 0.643$, and $\kappa_2(A) \approx 2.73$. Note $\rho(A) = \lambda_{\max} \approx 4.246 \le \|A\|_1 = \|A\|_\infty = 5$ as predicted.

Example 5 (Diagonal with wide spread)

$A = \mathrm{diag}(10^{-3}, 1, 10^2)$. Then

$$\|A\|_1 = \|A\|_\infty = \|A\|_2 = 10^2, \quad \|A^{-1}\|_2 = 10^3, \quad \kappa_2(A) = 10^5.$$

Eigenvalues are $(10^{-3}, 1, 10^2)$ so $\rho(A) = 10^2 = \|A\|_2$. This illustrates how κ_2 measures sensitivity: large spread in eigenvalues for symmetric matrices leads to large condition number.

11.7 Additional standard relations

For any induced norm $\|\cdot\|$ and any A, B:

$$\begin{aligned} &\|AB\| \le \|A\|\,\|B\| \quad \text{(submultiplicativity)}, \\ &\|A^k\| \le \|A\|^k \quad (k \in \mathbb{N}), \\ &\rho(A) = \lim_{k\to\infty} \|A^k\|^{1/k} \quad \text{(Gelfand's formula; equality for any consistent norm)}, \\ &\rho(A) \le \|A\| \quad \text{and} \quad \rho(A^k) = (\rho(A))^k. \end{aligned}$$

For symmetric A, the singular values equal $|\lambda_i|$, so all the above sharpen: $\|A\|_2 = \rho(A)$ and $\|A^{-1}\|_2 = 1/\min|\lambda_i|$ when invertible.

Proof of $\rho(A) \le \|A\|$ for induced norms. If $Av = \lambda v$ and $\|v\| = 1$, then $\|A\| \ge \|Av\| = \|\lambda v\| = |\lambda|$. Taking maximum over eigenvalues yields $\rho(A) \le \|A\|$.

Proof sketch of Gelfand's formula. One direction is immediate from submultiplicativity: $\|A^k\|^{1/k} \le \|A\|$. The converse uses Jordan/Schur decomposition to show that for any $\varepsilon > 0$ and large k one has $\|A^k\| \ge (\rho(A) - \varepsilon)^k$, whence $\liminf_{k\to\infty} \|A^k\|^{1/k} \ge \rho(A)$.

11.8 Summary

- Vector norms satisfy positivity, homogeneity, triangle inequality; $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$ are canonical and pairwise equivalent in finite dimensions with explicit constants.
- Induced matrix norms are operator norms consistent with the underlying vector norm; classical reductions give max column sum ($\|\cdot\|_1$), spectral norm ($\|\cdot\|_2$), and max row sum ($\|\cdot\|_\infty$).
- For symmetric A, $\|A\|_2 = \rho(A) = \max|\lambda_i|$, $\|A^{-1}\|_2 = 1/\min|\lambda_i|$, and $\kappa_2 = \lambda_{\max}/\lambda_{\min}$ (when invertible).

- Spectral radius always provides a lower bound for any induced matrix norm; for symmetric A, it equals the spectral norm.

12 Boundary Value Problems

In this section we go over the introduction of setting up consistent boundary value problems and begin to understand the complicating issues with defining stability and or robustness of a numerical method. the real goal is to introduce the developer to the notion of stability and prototype equations. Proving stability for nonlinear problems can be very difficult, but making use of prototype equations that are linearizations of the original problem can often lead to good numerical methods. as we are focused on finite difference methods the tools we will look at will be relatively constrained. Methods such as finite element methods spectrum methods collocation methods, etc offer other ways to understand these problems and often have additional tools for understanding stability and consistency beyond what we use in finite difference methods.

12.1 Consistency of Finite-Difference Schemes

Setting (stationary PDE with boundary conditions). Let $\Omega \subset \mathbb{R}^d$ be a domain with boundary $\partial\Omega$ and consider

$$\mathcal{L}u = g \quad \text{in } \Omega, \qquad \mathcal{B}u = b \quad \text{on } \partial\Omega,$$

where $\mathcal{L}$ is a (possibly nonlinear) differential operator and $\mathcal{B}$ a boundary operator. Let $\{\mathcal{T}_h\}$ be a family of grids with mesh parameter $h \to 0$. Denote by S_h the *sampling/restriction* of a smooth function to grid values, e.g. $(S_h u)_i = u(x_i)$, and by $\mathcal{L}_h$ and $\mathcal{B}_h$ the discrete FD operators on $\mathcal{T}_h$. Let $\|\cdot\|_h$ be a discrete norm on grid functions that is uniformly equivalent to a standard norm over compact subsets where u is smooth.

Definition 12.1 (Consistency (stationary))**.** The FD scheme

$$\mathcal{L}_h u_h = g_h \quad \text{in } \mathcal{T}_h, \qquad \mathcal{B}_h u_h = b_h \quad \text{on } \partial\mathcal{T}_h$$

is *consistent* with the continuous problem if, for every $u \in C^k(\overline{\Omega})$ solving $\mathcal{L}u = g$, $\mathcal{B}u = b$, the *residuals* obtained by inserting the exact u satisfy

$$\|\, \mathcal{L}_h(S_h u) - S_h(\mathcal{L}u) \,\|_h \xrightarrow[h\to 0]{} 0, \qquad \|\, \mathcal{B}_h(S_h u) - S_h(\mathcal{B}u) \,\|_{h,\partial} \xrightarrow[h\to 0]{} 0.$$

If there exists $p > 0$ and a constant C (independent of h but possibly depending on u on compact sets) such that

$$\|\, \mathcal{L}_h(S_h u) - S_h(\mathcal{L}u) \,\|_h \le C\, h^p, \qquad \|\, \mathcal{B}_h(S_h u) - S_h(\mathcal{B}u) \,\|_{h,\partial} \le C\, h^p,$$

we say the scheme is *p-th order consistent (in space).*

Equivalently (local truncation error). At a grid point $x_i \in \mathcal{T}_h$, the *local truncation error (LTE)* for a smooth u is

$$\tau_h[u](x_i) \ :=\ \mathcal{L}_h(S_h u)(x_i) - S_h(\mathcal{L}u)(x_i).$$

Pointwise consistency means $\max_{x_i\in\mathcal{T}_h} |\tau_h[u](x_i)| \to 0$ as $h \to 0$; p-th order means $|\tau_h[u](x_i)| \le C\, h^p$ (uniformly on compact subsets where u is smooth). An analogous definition holds on boundary nodes for $\mathcal{B}_h$.

Setting (time-dependent PDE). Consider

$$u_t = \mathcal{L}u + f \quad \text{in } \Omega \times (0,T], \qquad \mathcal{B}u = b \text{ on } \partial\Omega \times (0,T], \qquad u(\cdot,0) = u_0.$$

Let $\Delta t > 0$ be the time step and $t^n = n\Delta t$. Let $\Phi_{\Delta t,h}$ denote a one-step FD method that advances grid data from t^n to t^{n+1}.

Definition 12.2 (Consistency (time-dependent))**.** Insert the exact solution into one step of the scheme and define the *(one-step) residual*

$$R_h^{n+1} \;:=\; S_h u(\cdot, t^{n+1}) \;-\; \Phi_{\Delta t,h}\big(S_h u(\cdot, t^n)\big).$$

The method is *consistent* if $\|R_h^{n+1}\|_h \to 0$ as $(\Delta t, h) \to (0,0)$ for all sufficiently smooth u. It is (p_t, p_x)*-th order consistent* if there exists C such that

$$\|R_h^{n+1}\|_h \;\le\; C\big(\Delta t^{\,p_t+1} + h^{p_x}\big) \quad \text{for } 0 \le t^n \le T.$$

Sometimes one uses the *normalized LTE* $\tau_h^{n+1} := R_h^{n+1}/\Delta t$; then p_t-th order in time corresponds to $\|\tau_h^{n+1}\|_h = \mathcal{O}(\Delta t^{\,p_t})$.

Remarks.

- Consistency is a property of the discrete operator vis-à-vis the *exact* PDE: it asks that the discrete equations reduce to the continuous equations in the limit of mesh refinement.
- For linear, well-posed problems, the Lax Equivalence Theorem states:

$$\text{Consistency} + \text{Stability} \iff \text{Convergence}.$$

- In practice, p is obtained by Taylor expanding u about a grid node and matching the discrete stencil to the continuous derivatives; the first nonvanishing higher-order term determines the order.

Examples (brief)

Poisson, 3-point second derivative (1D). With uniform spacing h, the central difference $(-u_{xx})(x_i) \approx -(u_{i-1} - 2u_i + u_{i+1})/h^2$ has local truncation error $\tau_h[u](x_i) = \frac{1}{12}h^2 u^{(4)}(x_i) + \mathcal{O}(h^4)$, hence it is 2-nd order consistent in space.

Heat, FTCS. For $u_t = \alpha u_{xx}$, the FTCS scheme $(u_i^{n+1} - u_i^n)/\Delta t = \alpha(u_{i-1}^n - 2u_i^n + u_{i+1}^n)/h^2$ gives residual $R_h^{n+1} = \frac{\Delta t}{2}u_{tt} - \frac{\alpha h^2}{12}u_{xxxx} + \mathcal{O}(\Delta t^2 + h^4)$, so it is first order in time, second in space (consistent).

1D model problem and stability recap

Continuous problem. Given $L > 0$, solve

$$\begin{cases} -u''(x) = f(x), & x \in (0,L), \\ u(0) = \alpha, \quad u(L) = \beta. \end{cases} \tag{12.1}$$

Uniform mesh and discrete operator. Let $N \in \mathbb{N}$ be the number of interior nodes, $h = L/(N+1)$, and $x_i = i\,h$ for $i = 0, \dots, N+1$. For interior unknowns $U_i \approx u(x_i)$ $(i = 1, \dots, N)$, the second-order centered FD for $-u''$ is

$$\big[L_h U\big]_i := -\,\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2}, \qquad i = 1, \dots, N.$$

Collecting $U = (U_1, \dots, U_N)^\top$, we obtain the linear system

$$A\,U = b, \qquad A = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}, \tag{12.2}$$

with right-hand side

$$b_1 = f(x_1) + \frac{\alpha}{h^2}, \quad b_i = f(x_i)\ (i = 2, \dots, N-1), \quad b_N = f(x_N) + \frac{\beta}{h^2}. \tag{12.3}$$

A is SPD, strictly diagonally dominant, and an M-matrix.

Mesh function and local truncation error (LTE). Let $u_h = (u(x_1), \dots, u(x_N))^\top$ be the sampling of the exact solution on the grid. The LTE at x_i is

$$\tau_i := \big[L_h u_h\big]_i - f(x_i) = -\,\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} - f(x_i) = -\frac{h^2}{12}u^{(4)}(\xi_i) = \mathcal{O}(h^2),$$

for some $\xi_i \in (x_{i-1}, x_{i+1})$ (Taylor expansion). Thus the scheme is second-order *consistent.*

Error equation and stability. Let $e := U - u_h$. Subtracting the equations gives

$$A\,e = r, \qquad r := b - Au_h \ \ \text{(the LTE vector)}.$$

If the inverse is uniformly bounded (in a chosen norm) independently of h,

$$\|A^{-1}\| \le C_{\text{stab}},$$

then $\|e\| \le \|A^{-1}\|\,\|r\| \le C_{\text{stab}}\,\mathcal{O}(h^2)$: consistency + stability $\Rightarrow$ second-order *convergence.* A discrete maximum principle provides such a uniform bound in ℓ^∞, with constants independent of h.

2D, 3D, 4D, etc extension: set-up, truncation error, and stability

Often in finite difference, it is convenient to use tensor notation when writing these representations. Because this is something that is helpful in both data science and modern scientific computing, we take the time to go over tensor notation here. We keep it to 2D for simplicity, noting the complication is notation not the idea itself. Once we go over this, we will then introduce a 2D discreet representation of Poisson's equation.

Tensor product (of vector spaces)

Let V, W be vector spaces over a field $\mathbb{F}$.

Definition 12.3 (Universal property)**.** A *tensor product* of V and W is a pair $(V \otimes W,\, \otimes)$ consisting of a vector space $V \otimes W$ and a bilinear map

$$\otimes : \ V \times W \longrightarrow V \otimes W, \qquad (v, w) \mapsto v \otimes w,$$

such that for every vector space U and every bilinear map $b : V \times W \to U$ there exists a unique linear map $L : V \otimes W \to U$ with

$$b(v, w) \ = \ L(v \otimes w) \qquad \text{for all } (v, w) \in V \times W.$$

This property characterizes $V \otimes W$ uniquely up to unique isomorphism.

Consequences.

- The elements $v \otimes w$ are called *simple (or pure) tensors.* In general, an element of $V \otimes W$ is a finite sum $\sum_k v_k \otimes w_k$.
- Bilinear maps $V \times W \to U$ are in natural bijection with linear maps $V \otimes W \to U$:

$$\mathrm{Bilin}(V \times W,\, U) \ \cong \ \mathrm{Lin}(V \otimes W,\, U).$$

Concrete construction (quotient of a free space). Let F be the free vector space over $\mathbb{F}$ with basis symbols $[v, w]$ for each $(v, w) \in V \times W$. Let $R \subset F$ be the subspace generated by all relations enforcing bilinearity:

$$\begin{gathered}[v_1 + v_2, w] - [v_1, w] - [v_2, w], \quad [av, w] - a[v, w], \\ [v, w_1 + w_2] - [v, w_1] - [v, w_2], \quad [v, aw] - a[v, w],\end{gathered}$$

for $v, v_1, v_2 \in V,\ w, w_1, w_2 \in W,\ a \in \mathbb{F}$. Define

$$V \otimes W \ := \ F/R, \qquad v \otimes w := [v, w] + R.$$

One checks this satisfies the universal property above.

Bases and dimension. If $\{e_i\}_{i=1}^m$ is a basis of V and $\{f_j\}_{j=1}^n$ a basis of W, then

$$\{\, e_i \otimes f_j : 1 \le i \le m,\ 1 \le j \le n \,\}$$

is a basis of $V \otimes W$. Hence

$$\dim(V \otimes W) \ = \ (\dim V)\,(\dim W).$$

Tensor product of linear maps. If $A : V \to V'$ and $B : W \to W'$ are linear, there is a unique linear map

$$A \otimes B : \ V \otimes W \longrightarrow V' \otimes W', \qquad (A \otimes B)(v \otimes w) := Av \otimes Bw,$$

extended linearly. The construction is functorial and satisfies

$$(A_1 A_2) \otimes (B_1 B_2) = (A_1 \otimes B_1)(A_2 \otimes B_2), \quad \mathrm{Id}_V \otimes \mathrm{Id}_W = \mathrm{Id}_{V \otimes W}.$$

Relation to the Kronecker product (matrices). Fix bases of V, W, V', W'. The matrix of $A \otimes B$ in the tensor-product basis $\{e_i \otimes f_j\}$ is the *Kronecker product* of the matrices of A and B, denoted $A \otimes_{\text{Kron}} B$:

$$A \otimes_{\text{Kron}} B \;=\; \begin{bmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mm}B \end{bmatrix}.$$

For finite-difference matrices T_x, T_y, the 2D Laplacian discretization on a tensor grid is the *Kronecker sum*:

$$A \;=\; I_y \otimes_{\text{Kron}} T_x \;+\; T_y \otimes_{\text{Kron}} I_x,$$

which is the matrix of the operator $I \otimes T_x + T_y \otimes I$ on $V \otimes W$.

Examples.

1. $V = \mathbb{R}^m$, $W = \mathbb{R}^n$. With standard bases e_i, f_j,

$$\mathbb{R}^m \otimes \mathbb{R}^n \;\cong\; \mathbb{R}^{mn}, \qquad e_i \otimes f_j \longleftrightarrow \text{unit vector at index } (i,j).$$

2. If $v = \sum_i v_i e_i$ and $w = \sum_j w_j f_j$, then

$$v \otimes w \;=\; \sum_{i,j} v_i w_j \,(e_i \otimes f_j),$$

showing bilinearity and how simple tensors expand in the tensor-product basis.

3. For matrices $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$ and a matrix $X \in \mathbb{R}^{m\times n}$, the vectorization identity links PDE/separable operators:

$$\operatorname{vec}(AXB^\top) \;=\; (B \otimes_{\text{Kron}} A)\operatorname{vec}(X).$$

Remarks.

- The tensor product is associative and (canonically) commutative up to isomorphism: $(U \otimes V) \otimes W \cong U \otimes (V \otimes W)$, $V \otimes W \cong W \otimes V$.
- In applications, "tensor product" (space/operation) and "Kronecker product" (matrix representation) are often used together; the former is basis-free, the latter is basis-dependent.

12.2 Continuous problem (2D Poisson with Dirichlet data)

Let $\Omega = (0, L_x) \times (0, L_y)$ and consider

$$\begin{cases} -\Delta u(x,y) = f(x,y), & (x,y) \in \Omega, \\ u(x,y) = g(x,y), & (x,y) \in \partial\Omega, \end{cases} \tag{12.4}$$

with sufficiently smooth f and boundary data g.

12.3 Uniform tensor-product mesh and the 5-point stencil

Choose integers $N_x, N_y \geq 1$, spacings $h_x = L_x/(N_x + 1)$, $h_y = L_y/(N_y + 1)$, and grid points

$$x_i = i\, h_x, \quad i = 0, \dots, N_x + 1; \qquad y_j = j\, h_y, \quad j = 0, \dots, N_y + 1.$$

Interior unknowns are $U_{i,j} \approx u(x_i, y_j)$ for $i = 1, \dots, N_x$, $j = 1, \dots, N_y$. The standard second-order 5-point FD approximation to $-\Delta u$ is

$$\big[L_h U\big]_{i,j} := -\frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h_x^2} - \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h_y^2}, \qquad 1 \leq i \leq N_x, \;\; 1 \leq j \leq N_y. \tag{12.5}$$

Interior equations enforce $\big[L_h U\big]_{i,j} = f(x_i, y_j)$, while boundary values enter through g at ghost locations, contributing to the right-hand side b (see below).

12.4 Matrix form and boundary incorporation

Vectorize U in lexicographic order, e.g. $k = i + (j-1)N_x$ ("i fastest"):

$$\mathbf{U} = \big(U_{1,1}, U_{2,1}, \dots, U_{N_x,1},\; U_{1,2}, \dots, U_{N_x,2},\; \dots,\; U_{N_x,N_y}\big)^\top \in \mathbb{R}^{N_x N_y}.$$

Let

$$T_x = \frac{1}{h_x^2}\,\text{tridiag}(-1,\, 2,\, -1) \in \mathbb{R}^{N_x \times N_x}, \qquad T_y = \frac{1}{h_y^2}\,\text{tridiag}(-1,\, 2,\, -1) \in \mathbb{R}^{N_y \times N_y},$$

and I_x, I_y be identity matrices of sizes N_x, N_y. Then the global stiffness matrix is the Kronecker sum

$$\boxed{A = I_y \otimes T_x \;+\; T_y \otimes I_x \;\in \mathbb{R}^{(N_x N_y)\times(N_x N_y)}.} \tag{12.6}$$

A is sparse, symmetric positive definite, strictly diagonally dominant, and an M-matrix.

Right-hand side and Dirichlet data. Define the interior sampling $f_{i,j} = f(x_i, y_j)$. Dirichlet boundary values contribute to b at interior nodes adjacent to the boundary. In stencil form,

$$\begin{aligned} b_{i,j} = f_{i,j} &+ \frac{1}{h_x^2}\big[g(0, y_j)\,\mathbf{1}_{\{i=1\}} + g(L_x, y_j)\,\mathbf{1}_{\{i=N_x\}}\big] \\ &+ \frac{1}{h_y^2}\big[g(x_i, 0)\,\mathbf{1}_{\{j=1\}} + g(x_i, L_y)\,\mathbf{1}_{\{j=N_y\}}\big], \qquad 1 \leq i \leq N_x,\; 1 \leq j \leq N_y, \end{aligned}$$

and then vectorize b in the same lexicographic order as $\mathbf{U}$.

12.5 Local truncation error in 2D

Let u_h denote the grid sampling of the exact solution. The LTE is

$$\tau_{i,j} := \big[L_h u_h\big]_{i,j} - f(x_i, y_j).$$

A 2D Taylor expansion about (x_i, y_j) gives (for $u \in C^4$)

$$\frac{u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)}{h_x^2} = u_{xx}(x_i, y_j) + \frac{h_x^2}{12}\, u_{xxxx}(\xi_{i,j}), \tag{12.7}$$

$$\frac{u(x_i,y_{j-1})-2u(x_i,y_j)+u(x_i,y_{j+1})}{h_y^2}=u_{yy}(x_i,y_j)+\frac{h_y^2}{12}u_{yyyy}(\zeta_{i,j}), \tag{12.8}$$

for some nearby points $\xi_{i,j},\zeta_{i,j}$. Since $f=-(u_{xx}+u_{yy})$, the 5-point stencil yields

$$\tau_{i,j}=-\frac{h_x^2}{12}u_{xxxx}(\xi_{i,j})-\frac{h_y^2}{12}u_{yyyy}(\zeta_{i,j})=\mathcal{O}(h_x^2+h_y^2),$$

uniform on compact subsets where the fourth derivatives are bounded. Hence the method is second-order *consistent* in space on anisotropic meshes (order in each direction).

12.6 Stability in 2D and robustness as $h\to 0$

Let $\mathbf{U}$ solve $A\mathbf{U}=\mathbf{b}$ and $\mathbf{u}_h$ be the sampling of the exact solution. The error $\mathbf{e}=\mathbf{U}-\mathbf{u}_h$ satisfies

$$A\,\mathbf{e}=\mathbf{r}, \qquad \mathbf{r}=\mathbf{b}-A\mathbf{u}_h \ \text{ (vector of LTEs).}$$

If there exists a constant C_{stab} independent of h_x,h_y such that

$$\|A^{-1}\|\ \le\ C_{\text{stab}}, \tag{12.9}$$

then $\|\mathbf{e}\|\le C_{\text{stab}}\,\|\mathbf{r}\|=\mathcal{O}(h_x^2+h_y^2)$.

Discrete maximum principle (DMP) in 2D. For the 5-point M-matrix A, a DMP holds: if a grid function W satisfies homogeneous Dirichlet data on $\partial\Omega_h$ and $\big[L_hW\big]_{i,j}\ge 0$ on all interior nodes, then

$$\max_{(i,j)\in\overline{\Omega}_h} W_{i,j}\ \le\ 0,$$

and similarly for minima. This implies an ℓ^∞ stability bound of the form

$$\|\mathbf{U}\|_\infty\ \le\ C\Big(\|\mathbf{f}\|_\infty+\|g\|_{\infty,\partial\Omega}\Big), \qquad C \text{ independent of } h_x,h_y, \tag{12.10}$$

and hence (12.9) in the ℓ^∞ operator norm. (Sharper domain-dependent constants exist but the key point is h-independence.)

Definition (stability for the 2D FD BVP). The 5-point FD discretization (12.5)–(12.6) with Dirichlet data is *stable* if there is a constant $C>0$, independent of h_x,h_y, such that the discrete solution obeys an a priori bound like (12.10). Equivalently, A^{-1} is uniformly bounded (in a chosen operator norm) independently of mesh refinement.

Robustness as the mesh is refined. Combining second-order consistency ($\|\mathbf{r}\|=\mathcal{O}(h_x^2+h_y^2)$) with stability (12.9) yields

$$\|\mathbf{U}-\mathbf{u}_h\|=\mathcal{O}(h_x^2+h_y^2),$$

i.e. the method is robust and convergent of second order as $(h_x,h_y)\to(0,0)$.

12.7 Takeaways

- The 1D and 2D second-order FD schemes for Poisson with Dirichlet data lead to SPD, strictly diagonally dominant M-matrices (A and its Kronecker-sum analogue).
- Local truncation error is $\mathcal{O}(h^2)$ in 1D, and $\mathcal{O}(h_x^2 + h_y^2)$ in 2D.
- Stability is expressed as an a priori bound (or uniform boundedness of A^{-1}), often proved via a discrete maximum principle; constants are independent of mesh sizes.
- Consistency + Stability $\Rightarrow$ Convergence: the discrete solution converges to the exact solution sampled on the grid with the expected second-order rate.

12.8 Stability of the 2D Five–Point Discrete Laplacian with Homogeneous Dirichlet Data

In this section we go through the steps of proving that the five point stencil for the 2D discrete Laplacian is indeed stable for Dirichlet homogeneous boundary conditions. This proof uses the equivalent idea of an eigen functions form the 2D Laplacian example section 3 on separation of variables in this set of notes. You can think of it as guessing the form of the eigen vector given the solution to the continuous problem, or if you know the discreet Fourier transform, that is were it really comes from. The eigen vectors that make since, change with the boundary condition being investigated.

As you will see chapter 2 of in your text book, this can be done using a Green's function approach. Which is why I go through ans set up a section in the beginning of these notes that talks about Green's functions. I strongly recommend going those examples.

Goal and notion of stability

We consider the Poisson equation on a rectangle with homogeneous Dirichlet boundary conditions and its standard second–order finite–difference (FD) discretization using the 5–point stencil. We prove, with explicit eigenvalue computations, that the resulting stiffness matrix A is *uniformly stable* in the ℓ^2 (spectral–norm) sense: its smallest eigenvalue is bounded below by a positive constant *independent of the mesh size.* Equivalently, $\|A^{-1}\|_2$ is bounded uniformly as the grid is refined. (The spectral radius $\rho(A)$—the largest eigenvalue—is also computed; it grows like h^{-2}, which is expected and does not affect stability of A^{-1}.)

12.8.1 Continuous problem and uniform grid

Let $\Omega = (0, L_x) \times (0, L_y)$ and seek $u : \overline{\Omega} \to \mathbb{R}$ solving

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{12.11}$$

Choose integers $N_x, N_y \geq 1$, mesh sizes

$$h_x := \frac{L_x}{N_x + 1}, \qquad h_y := \frac{L_y}{N_y + 1},$$

and nodes $x_i = i\, h_x$ $(i = 0, \ldots, N_x + 1)$, $y_j = j\, h_y$ $(j = 0, \ldots, N_y + 1)$. Unknowns live at interior nodes (x_i, y_j) with $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$:

$$U_{i,j} \approx u(x_i, y_j).$$

12.8.2 Five–point stencil and matrix form

The standard second–order FD approximation to $-\Delta u$ at (x_i, y_j) is

$$\big[L_h U\big]_{i,j} = \frac{4U_{i,j} - U_{i-1,j} - U_{i+1,j} - U_{i,j-1} - U_{i,j+1}}{h^2} \quad \text{(square mesh, } h_x = h_y =: h\text{)}. \tag{12.12}$$

(For unequal spacings, replace $4/h^2$ by $2/h_x^2 + 2/h_y^2$ and the neighbor weights accordingly.) Homogeneous Dirichlet data sets ghost values to zero, so there are no boundary contributions in b.

Vectorize U in lexicographic order $k = i + (j-1)N_x$ to obtain

$$A\,\mathbf{U} = \mathbf{b}, \qquad A = I_{N_y} \otimes T_x \;+\; T_y \otimes I_{N_x}, \tag{12.13}$$

where I_n is the $n \times n$ identity and

$$T_x = \frac{1}{h_x^2}\,\mathrm{tridiag}(-1,\,2,\,-1) \in \mathbb{R}^{N_x \times N_x}, \qquad T_y = \frac{1}{h_y^2}\,\mathrm{tridiag}(-1,\,2,\,-1) \in \mathbb{R}^{N_y \times N_y}.$$

Thus A is a *Kronecker sum* of two 1D Dirichlet discrete Laplacians. It is symmetric positive definite (SPD).

12.8.3 Eigenpairs of the 1D Dirichlet Laplacian

We first diagonalize T_x (the y–direction is identical). Consider the discrete sine vectors

$$v^{(p)} = \big(\sin\big(\tfrac{p\pi i}{N_x+1}\big)\big)_{i=1}^{N_x} \in \mathbb{R}^{N_x}, \qquad p = 1, 2, \ldots, N_x.$$

Lemma 12.4 (1D eigenpairs)**.** *For each p, $T_x v^{(p)} = \lambda_x(p)\, v^{(p)}$ with*

$$\lambda_x(p) = \frac{4}{h_x^2}\,\sin^2\Big(\frac{p\pi}{2(N_x+1)}\Big). \tag{12.14}$$

Proof. Compute the action of T_x componentwise at $i = 1, \ldots, N_x$:

$$(T_x v^{(p)})_i = \frac{1}{h_x^2}\Big(-v_{i-1}^{(p)} + 2v_i^{(p)} - v_{i+1}^{(p)}\Big),$$

with the Dirichlet conventions $v_0^{(p)} = v_{N_x+1}^{(p)} = 0$. Using $\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$,

$$\begin{aligned} -v_{i-1}^{(p)} + 2v_i^{(p)} - v_{i+1}^{(p)} &= -\sin\Big(\tfrac{p\pi(i-1)}{N_x+1}\Big) + 2\sin\Big(\tfrac{p\pi i}{N_x+1}\Big) - \sin\Big(\tfrac{p\pi(i+1)}{N_x+1}\Big) \\ &= 2\sin\Big(\tfrac{p\pi i}{N_x+1}\Big)\big(1 - \cos\tfrac{p\pi}{N_x+1}\big) = 4\sin^2\Big(\tfrac{p\pi}{2(N_x+1)}\Big)\,\sin\Big(\tfrac{p\pi i}{N_x+1}\Big), \end{aligned}$$

where we used $1 - \cos\theta = 2\sin^2(\theta/2)$. Dividing by h_x^2 yields (12.14). □

12.8.4 Eigenpairs of the 2D matrix via Kronecker sums

Let $w^{(q)} \in \mathbb{R}^{N_y}$ be the y–analogue of $v^{(p)}$ and define the rank–one mode

$$\Phi^{(p,q)} := v^{(p)} \otimes w^{(q)} \in \mathbb{R}^{N_x N_y}, \qquad 1 \le p \le N_x,\; 1 \le q \le N_y.$$

Lemma 12.5 (2D eigenpairs)**.** *$A\,\Phi^{(p,q)} = \lambda_{p,q}\,\Phi^{(p,q)}$ with*

$$\lambda_{p,q} = \lambda_x(p) + \lambda_y(q) = \frac{4}{h_x^2}\sin^2\Big(\frac{p\pi}{2(N_x+1)}\Big) + \frac{4}{h_y^2}\sin^2\Big(\frac{q\pi}{2(N_y+1)}\Big). \tag{12.15}$$

Proof. Using $(B\otimes I)(u\otimes v) = (Bu)\otimes v$ and $(I\otimes C)(u\otimes v) = u\otimes(Cv)$,

$$A\,\Phi^{(p,q)} = (I\otimes T_x + T_y\otimes I)(v^{(p)}\otimes w^{(q)}) = (T_x v^{(p)})\otimes w^{(q)} \;+\; v^{(p)}\otimes(T_y w^{(q)}),$$

and Lemma 12.4 gives $T_x v^{(p)} = \lambda_x(p)v^{(p)}$, $T_y w^{(q)} = \lambda_y(q)w^{(q)}$, hence (12.15). □

Thus the *entire* spectrum of A is known explicitly:

$$\sigma(A) = \big\{\,\lambda_{p,q} : 1\le p\le N_x,\; 1\le q\le N_y\,\big\},\quad \text{with } \lambda_{p,q}>0 \text{ (so } A \text{ is SPD).}$$

12.8.5 Extremal eigenvalues and uniform bounds

Let

$$\lambda_{\min} := \min_{p,q}\lambda_{p,q} = \lambda_{1,1}, \qquad \lambda_{\max} := \max_{p,q}\lambda_{p,q} = \lambda_{N_x,N_y}.$$

We now bound these in terms of L_x, L_y only (independent of h_x, h_y).

Lower bound for $\lambda_{\min}$ (uniform coercivity)

For $t\in[0,\pi/2]$ we have the elementary inequality

$$\sin t \;\ge\; \frac{2}{\pi}\,t. \tag{12.16}$$

Apply (12.16) to $t_x := \frac{\pi}{2(N_x+1)} \in (0,\frac{\pi}{2})$ and $t_y := \frac{\pi}{2(N_y+1)}$:

$$\sin^2 t_x \;\ge\; \frac{4}{\pi^2}\,t_x^2 \;=\; \frac{4}{\pi^2}\cdot\frac{\pi^2}{4(N_x+1)^2} \;=\; \frac{1}{(N_x+1)^2}.$$

Therefore, using $h_x = \frac{L_x}{N_x+1}$,

$$\frac{4}{h_x^2}\sin^2 t_x \;\ge\; \frac{4}{h_x^2}\cdot\frac{1}{(N_x+1)^2} = \frac{4}{L_x^2}.$$

The same bound holds in y. Hence

$$\boxed{\lambda_{\min} = \lambda_{1,1} \;=\; \frac{4}{h_x^2}\sin^2\Big(\frac{\pi}{2(N_x+1)}\Big) \;+\; \frac{4}{h_y^2}\sin^2\Big(\frac{\pi}{2(N_y+1)}\Big) \;\ge\; \frac{4}{L_x^2}+\frac{4}{L_y^2}.} \tag{12.17}$$

In particular, $\lambda_{\min}$ is bounded *away from* 0 by a constant depending only on the domain, *independent of the mesh sizes h_x, h_y.*

Asymptotic sharpness. Using $\sin z = z + \mathcal{O}(z^3)$ as $z\to 0$ with $z = \frac{\pi}{2(N_x+1)}$, a standard small–angle expansion gives

$$\lambda_{\min} = \frac{\pi^2}{L_x^2} + \frac{\pi^2}{L_y^2} + \mathcal{O}(h_x^2 + h_y^2),$$

which matches the first Dirichlet eigenvalue of the continuous $-\Delta$.

Upper bound for $\lambda_{\max}$ (spectral radius)

Since $\sin^2 t \le 1$,

$$\boxed{\lambda_{\max} = \lambda_{N_x,N_y} \;=\; \frac{4}{h_x^2}\sin^2\Big(\frac{N_x\pi}{2(N_x+1)}\Big) \;+\; \frac{4}{h_y^2}\sin^2\Big(\frac{N_y\pi}{2(N_y+1)}\Big) \;\le\; \frac{4}{h_x^2}+\frac{4}{h_y^2}.} \tag{12.18}$$

(Indeed $\sin\big(\frac{N\pi}{2(N+1)}\big) = \cos\big(\frac{\pi}{2(N+1)}\big) \to 1$ as $N\to\infty$, so $\lambda_{\max} \sim 4/h_x^2 + 4/h_y^2$.)

12.8.6 Stability in ℓ^2 (spectral norm)

Because A is SPD, its spectral norm and the norm of its inverse are

$$\|A\|_2 = \lambda_{\max}, \qquad \|A^{-1}\|_2 = \frac{1}{\lambda_{\min}}.$$

Combining (12.17) and (12.18) yields the *uniform stability bound*

$$\boxed{\|A^{-1}\|_2 \;\le\; \frac{1}{\frac{4}{L_x^2}+\frac{4}{L_y^2}} \quad \text{independent of } h_x, h_y,} \tag{12.19}$$

whereas $\|A\|_2 \le \frac{4}{h_x^2}+\frac{4}{h_y^2}$ grows as the mesh is refined (which is expected for stiffness matrices).

Interpretation as stability of the FD scheme. Let $\mathbf{U}$ solve $A\mathbf{U}=\mathbf{b}$ and let $\mathbf{u}_h$ be the sampling of the exact solution. The error satisfies $A(\mathbf{U}-\mathbf{u}_h)=\mathbf{r}$, where $\mathbf{r}$ is the vector of local truncation errors (of order $h_x^2+h_y^2$). Then

$$\|\mathbf{U}-\mathbf{u}_h\|_2 \;\le\; \|A^{-1}\|_2\,\|\mathbf{r}\|_2 \;\le\; \frac{1}{\frac{4}{L_x^2}+\frac{4}{L_y^2}}\,\|\mathbf{r}\|_2,$$

so consistency together with the uniform bound (12.19) implies *robust* (second–order) convergence as $h_x, h_y \to 0$.

Summary

- The 2D five–point Dirichlet Laplacian matrix is the Kronecker sum $A = I\otimes T_x + T_y\otimes I$.
- Its eigenpairs are explicit: $\lambda_{p,q} = \lambda_x(p)+\lambda_y(q)$ with $\lambda_x(p) = \frac{4}{h_x^2}\sin^2\big(\frac{p\pi}{2(N_x+1)}\big)$, and similarly in y.
- The smallest eigenvalue satisfies $\lambda_{\min} \ge \frac{4}{L_x^2}+\frac{4}{L_y^2}$, independent of h; the largest satisfies $\lambda_{\max} \le \frac{4}{h_x^2}+\frac{4}{h_y^2}$.
- Hence $\|A^{-1}\|_2 = 1/\lambda_{\min}$ is uniformly bounded as the grid is refined: the scheme is (spectrally) stable.

12.9 Reality - non-linear problems

Well this is great that we were able to analyze the stability of the five point Laplacian, in reality most problems that we care about are nonlinear. Unfortunately techniques for analyzing discretizations of nonlinear problems are often a topic that needs to be approached on a case by case basis. In finite difference methods this is particularly true. However, the key tool for dealing with solving nonlinear PDE's is Newtons method. Newtons method is a type of linearization that takes the original problem and defines a sequence of iterations that "ideally" marches an initial guess to the correct solution to the system of algebraic equations. In this case we might analyze the stability of the linearized system that is used in the iteration. In which case we will be able to say something about the stability of each step of the iterative method.

There are also other types of iterative methods, known as fixed point methods, that can also be used to define a sequence of iterations. Here we end this section with examples of non-linear boundary value problems before getting into methods for matrix inversion for linear and non-linear systems of equations.

Examples of Nonlinear Elliptic Boundary–Value Problems: 1D and 2D

Below are standard examples of *nonlinear elliptic* problems with Dirichlet boundary conditions. Ellipticity means, roughly, that the principal (second–order) part has a positive coefficient (1D), or that the linearization has a positive–definite coefficient matrix (multi-D). Uniform ellipticity means the positivity has a mesh/point–independent lower bound.

1D (semilinear): Bratu–Gelfand problem

Find $u : (0,1) \to \mathbb{R}$ such that

$$\begin{cases} -u''(x) = \lambda\, e^{u(x)} & \text{in } (0,1), \\ u(0) = 0, \quad u(1) = 0, \end{cases} \qquad \lambda > 0.$$

This is *semilinear* (the highest derivative enters linearly with coefficient $+1$), hence elliptic in 1D. **Weak form:** Find $u \in H_0^1(0,1)$ with

$$\int_0^1 u'(x)\, v'(x)\, dx \;=\; \lambda \int_0^1 e^{u(x)} v(x)\, dx \quad \forall v \in H_0^1(0,1).$$

1D (quasilinear): nonlinear diffusion

Let $a : \mathbb{R} \to (0,\infty)$ be C^1 with $a(s) \geq a_0 > 0$. Seek $u : (0,1) \to \mathbb{R}$ solving

$$\begin{cases} -\,(a(u)u_x)_x = f(x) & \text{in } (0,1), \\ u(0) = \alpha, \quad u(1) = \beta. \end{cases}$$

This is *quasilinear* and *uniformly elliptic* since the principal part contributes $a(u)\, u_{xx}$ with $a(u) \geq a_0$.

2D (quasilinear): p-Laplacian (regularized)

On a bounded Lipschitz domain $\Omega \subset \mathbb{R}^2$, for $p > 1$, $\varepsilon > 0$, find $u : \Omega \to \mathbb{R}$:

$$\begin{cases} -\nabla\cdot\Big((\varepsilon + |\nabla u|^2)^{\frac{p-2}{2}}\, \nabla u\Big) = f(x) & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases}$$

For $\varepsilon > 0$ this operator is *uniformly elliptic*. As $\varepsilon \to 0$ it approaches the p-Laplacian $-\nabla\cdot(|\nabla u|^{p-2}\nabla u)$, which is elliptic but (for $p \neq 2$) generally *degenerate/singular* where $\nabla u = 0$. **Weak form:** Find $u \in H_0^1(\Omega)$ with

$$\int_\Omega (\varepsilon + |\nabla u|^2)^{\frac{p-2}{2}} \nabla u \cdot \nabla v \, dx \; = \; \int_\Omega f\, v \, dx \quad \forall v \in H_0^1(\Omega).$$

2D (quasilinear): minimal surface equation (with load)

Find $u : \Omega \to \mathbb{R}$ such that

$$\begin{cases} -\nabla\cdot\Big(\dfrac{\nabla u}{\sqrt{1+|\nabla u|^2}}\Big) = f(x) & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega. \end{cases}$$

The operator is quasilinear elliptic: the linearization has coefficient matrix

$$A(\nabla u) \; = \; \frac{1}{(1+|\nabla u|^2)^{3/2}}\Big((1+|\nabla u|^2)I - \nabla u\,\nabla u^\top\Big),$$

which is symmetric positive definite for all ∇u (hence elliptic).

2D (fully nonlinear): Monge–Ampère (elliptic in the convex class)

Given $f : \Omega \to (0, \infty)$, seek a *convex* u satisfying

$$\begin{cases} \det D^2 u(x) = f(x) & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega. \end{cases}$$

This is *fully nonlinear.* It is (degenerately) elliptic *in the convex class* because the derivative of $F(M) = \det M$ with respect to $M = D^2u$ is $\operatorname{cof}(D^2u)$, which is positive semidefinite when $D^2u \succeq 0$.

Notes on ellipticity.

- **1D:** A second–order ODE $-(a(u)u_x)_x = \cdots$ is uniformly elliptic if $a(u) \geq a_0 > 0$.
- **2D quasilinear:** Write the operator as $-\nabla\cdot(A(\nabla u)\nabla u)$; ellipticity requires $A(\xi) \succeq \alpha I$ for all ξ (uniformly for some $\alpha > 0$).
- **Fully nonlinear:** For $F(D^2u, x) = 0$, ellipticity means F is nondecreasing in the matrix order: $M \succeq N \Rightarrow F(M, x) \geq F(N, x)$; uniform ellipticity adds quantitative bounds.

13 Lax Equivalence Theorem for Boundary–Value Problems: Consistency + Stability $\iff$ Convergence

What this note proves. For *linear* boundary–value problems (BVPs), a family of finite–difference (or more generally linear) discretizations is **convergent** if and only if it is **stable**, provided it is **consistent**. This is the BVP version of the classical Lax (Lax–Richtmyer) equivalence theorem.

NOTE: In this section the theorems are stated correctly however we've not introduced the mathematical machinery to fully support what's in this section. That is to say we have not discussed what we mean by a Sobolev norm or a Hilbert space. In future versions of these notes we will make this a more complete. In terms of this class it's enough to understand that consistency plus stability guarantees convergence of the approximation to our boundary value problem. That is to say, section 13.3 is expected level of manipulation and understand that is required for this class.

13.1 Setting and notation

Let X and Y be Banach (or Hilbert) spaces, and let $L : X \to Y$ be a bounded invertible linear operator (modeling a well-posed boundary–value problem, e.g. $Lu = -\Delta u$ with Dirichlet data). Assume data $f \in Y$ is given and the exact solution $u \in X$ satisfies

$$Lu = f, \qquad \|u\|_X \le C_{\text{wp}} \|f\|_Y,$$

for a constant C_{wp} independent of f ("well-posedness").

For each mesh parameter $h > 0$ (grid size), let X_h and Y_h be finite-dimensional spaces of grid functions, equipped with norms $\|\cdot\|_{X,h}$ and $\|\cdot\|_{Y,h}$. Let

- $R_h : X \to X_h$ be a *restriction/sampling* operator (e.g. pointwise sampling on the grid);
- $Q_h : Y \to Y_h$ a *data discretization* (e.g. sampling or quadrature);
- $L_h : X_h \to Y_h$ the *discrete operator* (e.g. finite–difference matrix with boundary rows included).

The discrete solution $U_h \in X_h$ is defined by

$$L_h U_h = Q_h f \quad \text{in } Y_h. \tag{13.1}$$

Consistency (order p). We say the scheme is *consistent of order* $p > 0$ if, for all $u \in \mathcal{D} \subset X$ sufficiently smooth,

$$\tau_h[u] \;:=\; L_h(R_h u) - Q_h(Lu) \;\in Y_h, \qquad \|\tau_h[u]\|_{Y,h} \;\le\; C_{\text{cons}}\, h^p\, \|u\|_{\mathcal{X}}, \tag{13.2}$$

where $\|\cdot\|_{\mathcal{X}}$ is a smoothness norm (e.g. a Sobolev norm) and C_{cons} is independent of h.

Stability (uniform bounded inverse). We say the scheme is *stable* in $Y_h \to X_h$ if

$$L_h \text{ is invertible for all small } h, \text{ and} \qquad \|L_h^{-1}\|_{\mathcal{L}(Y_h, X_h)} \;\le\; C_{\text{stab}} \quad \text{with } C_{\text{stab}} \text{ independent of } h. \tag{13.3}$$

Convergence (order p). The scheme is *convergent of order p* if for every data $f \in Y$ with exact solution $u \in \mathcal{D}$, the discrete solution U_h to (13.1) satisfies

$$\|U_h - R_h u\|_{X,h} \;\le\; C_{\text{conv}}\, h^p\, \|u\|_{\mathcal{X}} \qquad \text{with } C_{\text{conv}} \text{ independent of } h. \tag{13.4}$$

13.2 Lax equivalence theorem for BVPs (statement and proof)

Theorem 13.1 (Lax equivalence for BVPs)**.** *Let $L : X \to Y$ be boundedly invertible and $\{(X_h, Y_h, L_h, R_h, Q_h)\}_{h>0}$ a family of linear discretizations. Assume* consistency (13.2)*. Then the scheme is* convergent (13.4) *if and only if it is* stable (13.3)*. Moreover, under stability,*

$$\|U_h - R_h u\|_{X,h} \;\le\; C_{\text{stab}}\, \|\tau_h[u]\|_{Y,h},$$

so the convergence order *equals the consistency order.*

Proof (sufficiency: consistency + stability ⇒ convergence). Let u solve $Lu = f$ and U_h solve $L_hU_h = Q_hf$. Define the *discrete error* $e_h := U_h - R_hu \in X_h$. Subtract the discrete equation from the discrete operator applied to the restricted exact solution:

$$L_he_h = L_hU_h - L_h(R_hu) = Q_hf - \big[Q_h(Lu) + \tau_h[u]\big] = -\,\tau_h[u].$$

By stability,

$$\|e_h\|_{X,h} \;=\; \|L_h^{-1}(-\tau_h[u])\|_{X,h} \;\le\; \|L_h^{-1}\|\, \|\tau_h[u]\|_{Y,h} \;\le\; C_{\text{stab}}\, C_{\text{cons}}\, h^p\, \|u\|_{\mathcal{X}},$$

which is (13.4) with $C_{\text{conv}} = C_{\text{stab}}C_{\text{cons}}$. □

Proof (necessity: convergence ⇒ stability on Q_hY). Assume convergence holds uniformly for all $f \in Y$ with solutions in $\mathcal{D}$, and assume Q_h is bounded and $R_hL^{-1} : Y \to X_h$ is bounded uniformly in h. Define the discrete solution operator

$$T_h := L_h^{-1}Q_h : Y \longrightarrow X_h.$$

From the discrete problem $L_hU_h = Q_hf$, we have $U_h = T_hf$. Convergence states $\|T_hf - R_hL^{-1}f\|_{X,h} \to 0$ as $h \to 0$ for all such f. By the uniform boundedness principle (Banach–Steinhaus), the family $\{T_h\}$ is uniformly bounded on Y:

$$\sup_h \|T_h\|_{\mathcal{L}(Y,X_h)} < \infty.$$

Hence

$$\|L_h^{-1}\|_{\mathcal{L}(Q_hY,\,X_h)} = \sup_{\substack{g\in Y\\ g\neq 0}} \frac{\|L_h^{-1}Q_hg\|_{X,h}}{\|Q_hg\|_{Y,h}} = \sup_{\substack{g\in Y\\ g\neq 0}} \frac{\|T_hg\|_{X,h}}{\|Q_hg\|_{Y,h}} \;\le\; C \sup_{\substack{g\in Y\\ g\neq 0}} \frac{\|g\|_Y}{\|Q_hg\|_{Y,h}} \;\le\; C'\,,$$

provided Q_h is norm–equivalent on Y (typical for sampling/quadrature on quasiuniform meshes). Thus stability holds at least on the range Q_hY; if Q_h is onto Y_h (as in standard FD with Dirichlet rows), this yields full stability (13.3). □

Takeaway. For *linear* BVPs, once the local residual τ_h is small (consistency), *uniform* control of the inverse (i.e. a mesh–independent a priori bound) is exactly what is needed for convergence; no further subtlety is hidden.

13.3 How this looks in a concrete FD BVP (worked model)

Consider the 2D Poisson Dirichlet problem on $\Omega = (0, L_x) \times (0, L_y)$:

$$-\Delta u = f \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega,$$

with $u \in C^4(\overline{\Omega})$. Let $h_x = L_x/(N_x+1)$, $h_y = L_y/(N_y+1)$ and use the standard 5–point stencil with homogeneous Dirichlet ghost values. In matrix form

$$A\,\mathbf{U} = \mathbf{b}, \qquad A = I \otimes T_x + T_y \otimes I,$$

where $T_x = \frac{1}{h_x^2}\,\mathrm{tridiag}(-1, 2, -1)$ (and analogously T_y). Let $R_h u$ be the grid sampling of u and $Q_h f$ the grid sampling of f.

Consistency (order two).

By the 1D Taylor expansions in x and y,

$$\big[L_h(R_h u)\big]_{i,j} - f(x_i, y_j) = -\tfrac{h_x^2}{12}\, u_{xxxx}(\xi_{i,j}) - \tfrac{h_y^2}{12}\, u_{yyyy}(\zeta_{i,j}) = \mathcal{O}(h_x^2 + h_y^2),$$

uniformly on interior nodes. Thus $\|\tau_h[u]\|_{Y,h} \le C(h_x^2 + h_y^2)\,\|u\|_{C^4}$.

Stability (uniform bound for A^{-1}).

A is symmetric positive definite and its eigenvalues are

$$\lambda_{p,q} = \tfrac{4}{h_x^2}\sin^2\!\Big(\tfrac{p\pi}{2(N_x+1)}\Big) + \tfrac{4}{h_y^2}\sin^2\!\Big(\tfrac{q\pi}{2(N_y+1)}\Big),$$

$1 \le p \le N_x$, $1 \le q \le N_y$. Hence

$$\lambda_{\min}(A) \;\ge\; \tfrac{4}{L_x^2} + \tfrac{4}{L_y^2}, \qquad \|A^{-1}\|_2 = \tfrac{1}{\lambda_{\min}(A)} \;\le\; C \quad \text{independent of } h_x, h_y.$$

(Equivalently one may use a discrete maximum principle for an ℓ^∞ bound.)

Convergence.

Let $\mathbf{U}$ solve $A\mathbf{U} = \mathbf{b}$ and set $\mathbf{e} = \mathbf{U} - R_h u$. The error equation $A\,\mathbf{e} = -\,\boldsymbol{\tau}$ with $\|\boldsymbol{\tau}\| = \mathcal{O}(h_x^2 + h_y^2)$ gives

$$\|\mathbf{e}\| \;\le\; \|A^{-1}\|\,\|\boldsymbol{\tau}\| \;\le\; C\,\big(h_x^2 + h_y^2\big),$$

which is second–order convergence by Theorem 13.1.

13.4 Remarks and variants

- The proof works verbatim for other linear BVPs (variable coefficients, Robin/Neumann data), provided the discrete operator family is *uniformly stable* in a relevant norm (often proved via an energy estimate, a discrete Poincaré inequality, or a discrete maximum principle).
- The "necessity" (convergence $\Rightarrow$ stability) needs linearity and a mild surjectivity/compatibility assumption on data discretization Q_h; otherwise one gets stability *on the range* $Q_h Y$.
- Changing norms is allowed as long as consistency is measured in Y_h and stability in $Y_h \to X_h$ for the same pair.

14 Solving $Ax = b$

In this section we review many methods for solving $Ax = B$. The two key tools are fixed point iteration (+multi-grid) and Krylov methods. Further, non-linear systems of algebraic equations, we will extend these tools by introducing additional tools such a Newtons method, Secant method and Anderson acceleration.

15 Time stepping methods

16 Parabolic problems

17 Hyperbolic problems

18 Systems of Mixed PDEs

19 Physics Informed Neural Networks

20 Structure Preserving Neural Networks

21 Neural ODE's

22 Blended Computing to Address Scale Separation

A Coding standard (summary)

All submitted code must follow these rules:

1. **Object–oriented design.** Encapsulate state and behavior in classes (e.g., solver objects, operators, meshes).
2. **Modularity.** No body of code may exceed one page. If it does, *split* into subroutines (helper methods).
3. **Header comment block.** *Every* subroutine begins with a well–formed comment block containing:
   - name of the function; purpose; author; date written; date last modified;
   - inputs (names and types); outputs (names and types);
   - dependencies (other subroutines this one calls).
4. **Inline commentary.** Inside the code, comment key operations and algorithmic intentions (not obvious syntax).
5. **Types and I/O.** Use Python type hints for `def` signatures; document units and shapes.
6. **Testing/examples.** Provide a minimal usage example (doctest or short script) demonstrating expected output.

A.1 Header comment block template (paste into each subroutine)

```
"""
Name:           <function_or_method_name>
Purpose:        <what this subroutine computes and why>
Author:         <Your Name>
Date written:   YYYY-MM-DD
Last modified:  YYYY-MM-DD
Inputs:
  - <arg1>: <type> ... <meaning/units/shape>
  - <arg2>: <type> ... <meaning/units/shape>
Outputs:
  - <ret>:  <type> ... <meaning/units/shape>
Dependencies:
  - <SubroutineA>, <SubroutineB>, ... (list all helpers called here)
"""
```

A.2 Example: Object–oriented 1D Poisson solver (Dirichlet), split into subroutines

Description. This small class solves $-u''(x) = f(x)$ on (a, b) with $u(a) = u(b) = 0$ using a second–order finite difference scheme. The solver is split into short, documented subroutines: matrix assembly, right–hand side assembly, and a lightweight Conjugate Gradient routine.

Listing 1: `poisson1d.py`: object-oriented, modular, and fully commented

```
from __future__ import annotations
from typing import Callable, Optional, Tuple
import numpy as np

class Poisson1DSolver:
    """
    Name:           Poisson1DSolver
    Purpose:        Encapsulate a 1D Dirichlet Poisson solver: -u'' = f on (a,b), u(a)
    Author:         Your Name
    Date written:   2025-08-22
    Last modified:  2025-08-22
    Inputs (constructor):
      - n: int                    ... total grid points including boundaries (n >= 3)
      - f: Callable[[np.ndarray], np.ndarray] ... RHS sampled at coordinates x
      - a: float                  ... left endpoint of domain
      - b: float                  ... right endpoint of domain
    Outputs:
      - creates a solver object with grid, spacing, and callable methods
    Dependencies:
      - _assemble_matrix, _assemble_rhs, _conjugate_gradient
    """
    def __init__(self, n: int, f: Callable[[np.ndarray], np.ndarray],
                 a: float = 0.0, b: float = 1.0) -> None:
```

```
        assert n >= 3, "Need at least 3 nodes (including boundaries)."
        self.n = n
        self.a, self.b = float(a), float(b)
        self.x = np.linspace(self.a, self.b, n)        # grid including endpoints
        self.h = float(self.x[1] - self.x[0])          # uniform spacing
        self.f = f

    def _assemble_matrix(self) -> np.ndarray:
        """
        Name:           _assemble_matrix
        Purpose:        Build the interior (n-2) x (n-2) tridiagonal matrix for -d2/dx
        Author:         Your Name
        Date written:   2025-08-22
        Last modified:  2025-08-22
        Inputs:
          - none (uses self.h and self.n)
        Outputs:
          - A: np.ndarray of shape (n-2, n-2) ... SPD tridiagonal
        Dependencies:
          - none
        """
        m = self.n - 2                              # number of interior unknowns
        # Tridiagonal with 2/h^2 on diag and -1/h^2 on off-diagonals
        main = (2.0 / self.h**2) * np.ones(m)
        off  = (-1.0 / self.h**2) * np.ones(m-1)
        A = np.diag(main) + np.diag(off, k=1) + np.diag(off, k=-1)
        return A

    def _assemble_rhs(self) -> np.ndarray:
        """
        Name:           _assemble_rhs
        Purpose:        Assemble the interior RHS, including boundary contributions.
        Author:         Your Name
        Date written:   2025-08-22
        Last modified:  2025-08-22
        Inputs:
          - none (uses self.f, self.x, self.h)
        Outputs:
          - b: np.ndarray of shape (n-2,) ... interior load vector
        Dependencies:
          - none
        """
        xi = self.x[1:-1]                           # interior coordinates
        b = self.f(xi)                              # sample f at interior nodes
        # Homogeneous Dirichlet: endpoints are zero, so no additive terms here.
        # If u(a)=ua, u(b)=ub nonzero, add ua/h^2 to b[0] and ub/h^2 to b[-1].
        return b
```

```
def _conjugate_gradient(self, A: np.ndarray, b: np.ndarray,
                        x0: Optional[np.ndarray] = None,
                        tol: float = 1e-10, maxiter: Optional[int] = None
                        ) -> Tuple[np.ndarray, int]:
    """
    Name:           _conjugate_gradient
    Purpose:        Solve SPD system A x = b using the (unpreconditioned) CG metho
    Author:         Your Name
    Date written:   2025-08-22
    Last modified:  2025-08-22
    Inputs:
      - A: np.ndarray (m,m) ... symmetric positive definite matrix
      - b: np.ndarray (m,)   ... right-hand side
      - x0: Optional initial guess (defaults to zeros)
      - tol: float           ... stopping tolerance on relative residual
      - maxiter: Optional[int] ... iteration cap (defaults to m)
    Outputs:
      - x: np.ndarray (m,)   ... approximate solution
      - k: int               ... iterations performed
    Dependencies:
      - none
    """
    m = b.size
    if maxiter is None:
        maxiter = 5 * m                    # light safety cap
    x = np.zeros_like(b) if x0 is None else x0.copy()
    r = b - A @ x                          # residual r_0
    p = r.copy()
    rs_old = np.dot(r, r)
    bnorm = max(np.linalg.norm(b), 1.0)
    for k in range(1, maxiter + 1):
        Ap = A @ p
        alpha = rs_old / np.dot(p, Ap)
        x += alpha * p                     # update iterate
        r -= alpha * Ap                    # update residual
        if np.linalg.norm(r) / bnorm < tol:
            return x, k
        rs_new = np.dot(r, r)
        beta = rs_new / rs_old
        p = r + beta * p                   # new search direction
        rs_old = rs_new
    return x, k

def solve(self, tol: float = 1e-10, maxiter: Optional[int] = None) -> np.ndarray:
    """
    Name:           solve
    Purpose:        Solve -u'' = f with u(a)=u(b)=0; return nodal values including
    Author:         Your Name
```

```
        Date written:    2025-08-22
        Last modified:   2025-08-22
        Inputs:
          - tol: float              ... CG tolerance
          - maxiter: Optional[int]  ... CG max iterations
        Outputs:
          - u: np.ndarray (n,)      ... solution on full grid [a,b], u[0]=u[-1]=0
        Dependencies:
          - _assemble_matrix, _assemble_rhs, _conjugate_gradient
        """
        A = self._assemble_matrix()
        b = self._assemble_rhs()
        ui, _ = self._conjugate_gradient(A, b, tol=tol, maxiter=maxiter)
        u = np.zeros(self.n)
        u[1:-1] = ui                            # apply Dirichlet boundary values (zero
        return u
```

A.3 Minimal usage example (documented and testable)

```
if __name__ == "__main__":
    # Test problem: -u'' = f with exact solution u(x)=sin(pi x), u(0)=u(1)=0
    import numpy as np

    def rhs(x: np.ndarray) -> np.ndarray:
        return (np.pi**2) * np.sin(np.pi * x)

    solver = Poisson1DSolver(n=64, f=rhs, a=0.0, b=1.0)
    u_num = solver.solve(tol=1e-12)
    u_ex  = np.sin(np.pi * solver.x)
    err   = np.linalg.norm(u_num - u_ex, ord=np.inf)

    print(f"grid h = {solver.h:.5f},  max error = {err:.3e}")
```

Notes.

- Each method is short (well below a page) and begins with a full header comment block.
- Key steps (*why* each operation is done) are commented inline.
- If you add features (nonzero Dirichlet data, Neumann BC, preconditioning), create new subroutines with their own headers, rather than elongating any one method.

B Learning Python for Symbolic Finite Differences: From Taylor Series to Working Solvers (with Detailed API Explanations)

This note shows how to use Python's symbolic library `SymPy` to *derive* finite-difference (FD) stencils and truncation-error constants directly from Taylor-series constraints, then plug those stencils into `NumPy/SciPy` to build working PDE solvers. It includes fully explained Python code and a glossary of the functions used. The material is self-contained: you can copy the listings, run them, and verify convergence.

B.1 Setup

Install the core packages once:

```
pip install sympy numpy scipy matplotlib
```

We will use:

- **SymPy** for exact algebra (symbols, matrices, series, solving).
- **NumPy** for arrays and vectorized operations.
- **SciPy.sparse** for sparse matrices and linear solvers.
- **Matplotlib** (optional) for plotting.

B.2 Mathematical background: moment matching for FD stencils

Let f be smooth. Choose distinct integer offsets s_j (in grid steps h), $j = 1, \dots, m$. We seek coefficients c_j such that the d-th derivative at x satisfies

$$f^{(d)}(x) \approx \frac{1}{h^d} \sum_{j=1}^{m} c_j \, f(x + s_j h). \tag{B.1}$$

Using Taylor series,

$$f(x + s_j h) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x)}{k!} \, (s_j h)^k.$$

Insert into (B.1) and equate coefficients of $f^{(k)}(x)$:

$$\sum_{j=1}^{m} c_j s_j^k = \begin{cases} 0 & \text{for } k = 0, 1, \dots, m-1, \ k \neq d, \\ d! & \text{for } k = d. \end{cases} \tag{B.2}$$

This $m \times m$ linear system determines the c_j. When (B.2) holds for $k = 0, \dots, m-1$, the first *uncontrolled* moment occurs at some $k^\star \geq m$. The truncation error then takes the form

$$\text{TE} \ = \ \frac{h^{k^\star - d}}{k^\star!} \Big(\sum_{j=1}^{m} c_j s_j^{k^\star} \Big) \, f^{(k^\star)}(x) \ + \ \mathcal{O}\big(h^{k^\star - d + 1}\big).$$

Define the *order* $p = k^\star - d$ and leading constant $C_p = \big(\sum_j c_j s_j^{k^\star} \big) / k^\star!$.

Remarks. This framework works for central, one-sided, and even nonuniform stencils (if the s_j are not integers but real offsets measured in units of h).

B.3 Symbolic derivation in SymPy

B.3.1 Computing weights

Listing 2: Finite-difference weights by moment matching.

```
import sympy as sp

def fd_weights(offsets, d):
    """
    offsets: list of offsets s_j (e.g., [-2,-1,0,1,2]) measured in grid steps
    d: derivative order (d=1 for first derivative, d=2 for second, etc.)
    returns: sympy Matrix [c_j] such that
             f^(d)(x)      (1/h**d) *   _j  c_j * f(x + s_j*h)
    """
    m = len(offsets)
    # Build the m m linear system A c = b for moments k=0..m-1
    A = sp.Matrix([[sp.Integer(s)**k for s in offsets] for k in range(m)])
    b = sp.Matrix([0]*m)
    b[d] = sp.factorial(d)       # Enforce     c_j s_j^d = d!
    c = A.LUsolve(b)             # Exact solve (rational arithmetic)
    return sp.simplify(c)
```

Explanation of functions used:

- `sp.Matrix(...)` constructs a symbolic matrix (exact arithmetic).
- `sp.Integer(s)` ensures exact integers (not floats).
- `sp.factorial(d)` returns $d!$ exactly.
- `A.LUsolve(b)` solves $Ac = b$ via LU factorization symbolically, returning exact rationals when inputs are exact.
- `sp.simplify(c)` cleans up expressions (e.g., reduces fractions).

B.3.2 Order and leading truncation-error constant

Listing 3: Detecting the first nonzero uncontrolled moment.

```
def leading_error_constant(offsets, d):
    """
    Returns (p, C_p) where p is the order and TE      C_p * h**p * f^(d+p)(x).
    """
    c = fd_weights(offsets, d)
    m = len(offsets)
    k = m  # first candidate beyond enforced moments 0..m-1
    while True:
        mom = sum(c[j] * sp.Integer(offsets[j])**k for j in range(m))
        if mom != 0:
            p = k - d
            Cp = sp.simplify(mom / sp.factorial(k))
            return int(p), Cp
        k += 1
```

What's happening: We test successive moments $\sum_j c_j s_j^k$ for $k = m, m+1, \ldots$ until the first nonzero one. That index $k^\star$ gives $p = k^\star - d$ and $C_p = \left(\sum_j c_j s_j^{k^\star}\right)/k^\star!$.

B.3.3 Examples: classic stencils

```
# 4th-order central for f'(x) on five points
c1 = fd_weights([-2,-1,0,1,2], d=1)
p1, C1 = leading_error_constant([-2,-1,0,1,2], d=1)
print("f'(x) weights:", list(c1))        # [-1/12, 2/3, 0, -2/3, 1/12]
print("Order =", p1, " TE constant =", C1)

# 3-point second derivative
c2 = fd_weights([-1,0,1], d=2)
p2, C2 = leading_error_constant([-1,0,1], d=2)
print("f''(x) weights:", list(c2))       # [1, -2, 1]
print("Order =", p2, " TE constant =", C2)
```

Expected results. For $[-2,-1,0,1,2]$ and $d = 1$, the weights are $\{-\frac{1}{12}, \frac{2}{3}, 0, -\frac{2}{3}, \frac{1}{12}\}$, so

$$f'(x) \approx \frac{-f(x-2h) + 8f(x-1h) - 8f(x+h) + f(x+2h)}{12h},$$

with fourth-order accuracy. For $[-1,0,1]$ and $d = 2$, we recover $[1,-2,1]$ with second-order accuracy.

B.4 Taylor expansions in SymPy

There are two common use modes.

B.4.1 Expanding known functions in the step size h

```
x, h = sp.symbols('x h')
series_of_sin = sp.series(sp.sin(x + h), h, 0, 6)
print(series_of_sin)
```

Functions explained:

- `sp.symbols('x h')` creates symbolic variables x, h.
- `sp.series(expr, var, point, order)` expands `expr` in `var` about `point` up to (but not including) the given `order`.

This is useful for sanity checks with concrete f (e.g., sin or e^x).

B.4.2 Unknown f: rely on moment matching

For an arbitrary smooth f, the *moment conditions* already capture the series bookkeeping. The routine `leading_error_constant` returns the order p and C_p without naming f.

B.5 From stencils to a working solver: 1D Poisson

We solve $-u''(x) = f(x)$ on $[0,1]$ with $u(0) = u(1) = 0$. Choose the manufactured solution $u(x) = \sin(\pi x)$ so that $f(x) = \pi^2 \sin(\pi x)$.

B.5.1 Discretization

Let $x_i = ih$, $i = 0, \dots, n+1$ with $h = \frac{1}{n+1}$. For interior nodes $i = 1, \dots, n$,

$$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(x_i).$$

This yields a tridiagonal linear system $A\,\mathbf{u} = \mathbf{b}$ for the interior unknowns.

B.5.2 Implementation (NumPy/SciPy)

Listing 4: Poisson solver with Dirichlet conditions.

```
import numpy as np
from scipy.sparse import diags
from scipy.sparse.linalg import spsolve

def poisson_1d_dirichlet(n):
    """
    n: number of interior points (unknowns)
    Grid: x_i = i*h, i=0..n+1, h=1/(n+1)
    Returns: (x, u, u_true, err_inf)
    """
    h = 1.0/(n+1)
    x = np.linspace(0.0, 1.0, n+2)      # includes boundaries
    u_true = np.sin(np.pi*x)
    f = (np.pi**2) * np.sin(np.pi*x)

    # Assemble A = (1/h^2) * tridiag(-1, 2, -1) on interior nodes
    main = 2.0*np.ones(n)
    off  = -1.0*np.ones(n-1)
    A = diags([off, main, off], offsets=[-1, 0, 1], shape=(n, n)) / (h**2)

    # Right-hand side for interior
    b = f[1:-1].copy()  # Dirichlet zero => no boundary contribution to subtract

    # Solve A u_interior = b
    u_interior = spsolve(A, b)

    # Insert boundaries
    u = np.zeros_like(x)
    u[1:-1] = u_interior

    err_inf = np.linalg.norm(u - u_true, ord=np.inf)
    return x, u, u_true, err_inf

if __name__ == "__main__":
    for n in [31, 63, 127, 255]:
        x, u, u_true, err = poisson_1d_dirichlet(n)
        print(f"n={n:4d}  max|error|={err:.3e}")
```

Functions explained (NumPy/SciPy).

- `np.linspace(a,b,N)`: N evenly spaced points from a to b (inclusive).
- `np.ones(n)`: array of ones of length n.
- `diags(data, offsets, shape)`: builds a sparse matrix with given diagonal bands. Here we create tridiagonal $\mathrm{diag}(-1, 2, -1)$.
- `spsolve(A, b)`: sparse direct solve (A must be square).
- `np.linalg.norm(v, ord=np.inf)`: infinity norm (max absolute value).
- `np.zeros_like(x)`: zeros array with same shape/dtype as `x` (to insert BCs).

What to expect. With second-order central differences, halving h (doubling $n + 1$) reduces the error by about $4\times$.

B.6 How each Python function is used (quick reference)

SymPy (symbolic)

Function	Where	Purpose
`sp.Matrix`	§3	Create exact linear systems for moments.
`sp.Integer`	§3	Force integers (no floating-point drift).
`sp.factorial`	§3	Right-hand side $d!$ in moment equations.
`Matrix.LUsolve`	§3	Solve $Ac = b$ symbolically (exact rationals).
`sp.simplify`	§3	Clean final expressions/ratios.
`sp.symbols`	§4	Declare symbolic variables (e.g., x, h).
`sp.series`	§4	Taylor series for sanity checks on known f.

NumPy / SciPy (numeric)

Function	Where	Purpose
`np.linspace`	§5	Build grid (including boundaries).
`np.ones`	§5	Fill diagonal bands of A.
`diags`	§5	Assemble sparse tridiagonal matrix A.
`spsolve`	§5	Solve $Au = b$ (direct sparse solver).
`np.zeros_like`	§5	Insert Dirichlet BC values.
`np.linalg.norm`	§5	Compute error in ℓ_∞ norm.

B.7 Extending the recipe

Nonuniform and one-sided stencils. Replace integer offsets by real offsets (s_j) measured in units of h. For example, near a boundary you might use $s = \{0, 1, 2, 3\}$ to approximate $f'(x)$ one-sided; `fd_weights` handles this unchanged.

Boundary conditions. For Dirichlet, values at boundaries are known and either removed (as above) or moved to the RHS if nonzero. For Neumann/Robin, derive ghost-point relations with another call to `fd_weights` using an appropriate one-sided stencil.

Higher order & multi-D. Use larger stencils (e.g., 5- or 7-point for fourth-/sixth-order). In 2D, the standard 5-point Laplacian comes from tensor products of 1D second-derivative stencils; 9-point variants improve isotropy.

Time stepping. For $u_t = \alpha u_{xx}$, couple `fd_weights` (space) with explicit/implicit time integrators. Stability of explicit Euler demands $\Delta t \lesssim h^2/(2\alpha)$; Crank–Nicolson is unconditionally stable for the linear heat equation.

B.8 Worked exercises

1. Use `fd_weights([0,1,2,3], d=1)` to derive a third-order one-sided stencil for $f'(x)$.
2. Modify the Poisson solver to *nonzero* Dirichlet boundaries and verify you correctly subtract boundary contributions on the RHS.
3. Build a 2D Poisson solver on an $N \times N$ grid using `scipy.sparse.kron` with 1D Laplacians; measure $\mathcal{O}(h^2)$ convergence.
4. For the heat equation, implement explicit Euler and empirically find the largest stable Δt for a given h.

B.9 Selected stencils (from the code)

- $f'(x)$ on $\{-2, -1, 0, 1, 2\}$:
$$\frac{-f(x-2h) + 8f(x-1h) - 8f(x+h) + f(x+2h)}{12h} \quad \text{(4th order)}.$$
- $f''(x)$ on $\{-1, 0, 1\}$:
$$\frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \quad \text{(2nd order)}.$$

B.9.1 Complete code (copy-paste)

```
import sympy as sp
import numpy as np
from scipy.sparse import diags
from scipy.sparse.linalg import spsolve

def fd_weights(offsets, d):
    m = len(offsets)
    A = sp.Matrix([[sp.Integer(s)**k for s in offsets] for k in range(m)])
    b = sp.Matrix([0]*m)
    b[d] = sp.factorial(d)
    c = A.LUsolve(b)
    return sp.simplify(c)

def leading_error_constant(offsets, d):
    c = fd_weights(offsets, d)
    m = len(offsets)
    k = m
    while True:
        mom = sum(c[j] * sp.Integer(offsets[j])**k for j in range(m))
        if mom != 0:
            p = k - d
            Cp = sp.simplify(mom / sp.factorial(k))
            return int(p), Cp
        k += 1

def poisson_1d_dirichlet(n):
    h = 1.0/(n+1)
    x = np.linspace(0.0, 1.0, n+2)
    u_true = np.sin(np.pi*x)
    f = (np.pi**2) * np.sin(np.pi*x)
    main = 2.0*np.ones(n)
    off  = -1.0*np.ones(n-1)
    A = diags([off, main, off], offsets=[-1, 0, 1], shape=(n, n)) / (h**2)
    b = f[1:-1].copy()
    u_interior = spsolve(A, b)
    u = np.zeros_like(x)
    u[1:-1] = u_interior
    err_inf = np.linalg.norm(u - u_true, ord=np.inf)
    return x, u, u_true, err_inf

if __name__ == "__main__":
    # Demonstrate stencil generation
    c1 = fd_weights([-2,-1,0,1,2], d=1)
    p1, C1 = leading_error_constant([-2,-1,0,1,2], d=1)
    print("f'(x) weights:", list(c1), "order:", p1, "TE const:", C1)
```

```
c2 = fd_weights([-1,0,1], d=2)
p2, C2 = leading_error_constant([-1,0,1], d=2)
print("f''(x) weights:", list(c2), "order:", p2, "TE const:", C2)

# Convergence check for Poisson
for n in [31, 63, 127, 255]:
    _, _, _, err = poisson_1d_dirichlet(n)
    print(f"n={n:4d}  max|error|={err:.3e}")
```