NOTES AND STUDIES IN SPECTRAL METHODS

Computational Études

A Spectral Approach

Dr. Denys Dutykh

Mathematics Department
College of Computing and Mathematical Sciences
Khalifa University of Science and Technology
Abu Dhabi, UAE

2026

Contents

Preface

The purpose of this book is twofold: to explain to students why spectral methods work and why they are so remarkably efficient, and to teach them how to implement these methods using modern computational tools.

This text is conceived as a collection of **Computational Études**. In musical education, an étude is a composition designed to perfect a specific technique (be it rapid scales, complex arpeggios, or delicate phrasing) while remaining a pleasing piece of music in its own right. Similarly, each chapter here presents a focused mathematical concept paired with its computational realization: a study that is both instructive and complete.

Who Is This Book For?

We have written primarily for graduate students in applied mathematics, physics, and engineering who seek a hands-on understanding of spectral methods. The reader should be comfortable with calculus, linear algebra, and basic programming. Prior exposure to numerical methods is helpful but not essential; we build the necessary foundations as we proceed.

How Is the Book Organized?

Each chapter is designed to be largely self-contained. We begin with fundamental concepts (interpolation and differentiation) before advancing to time-stepping schemes and applications. The mathematical exposition is deliberately concise, favoring clarity over exhaustive rigor. Proofs are included when they illuminate; otherwise, we direct the reader to authoritative references.

Reproducible Science

This book is also an experiment in **reproducible science**. Every figure, every table, every numerical result you see in these pages was generated by code available in the accompanying repository. We provide implementations in both Python and MATLAB, allowing readers to choose their preferred environment. The Python code emphasizes accessibility and integration with the open-source scientific ecosystem; the MATLAB code leverages its historical significance in numerical computing and, where appropriate, the Advanpix Multiprecision Computing Toolbox for extended precision arithmetic.

How to Use This Book

We invite you to treat this book not as a static reference, but as a workshop. Clone the repository, run the scripts, modify the parameters, break the code, and fix it. That is the only way to truly master the art of spectral methods.

The complete source code and the Typst manuscript are available at:
https://github.com/dutykh/computational-etudes/

Dr. Denys Dutykh

Abu Dhabi, UAE
January 2026

CHAPTER 1

Introduction

Differential equations serve as the fundamental language of the physical sciences, describing phenomena ranging from the propagation of sound waves to the flow of heat and the dynamics of fluids. Finding exact analytical solutions to these equations is a luxury rarely afforded in practical applications. Consequently, the scientist and the engineer must turn to numerical approximation.

Broadly speaking, numerical methods for differential equations fall into two categories: local methods and global methods. The former, including Finite Difference and Finite Element Methods, approximate the unknown solution using functions that are non-zero only on small sub-domains (elements or grid stencils). These methods are robust and flexible, handling complex geometries with grace. However, their accuracy is typically algebraic; refining the grid by a factor of two might improve the error by a factor of four or eight, but rarely more. From a computational perspective, local methods are *myopic*: to compute a derivative at a grid point, they look only at immediate neighbors.

Spectral methods represent the global approach. They approximate the solution as a linear combination of continuous, global basis functions, typically trigonometric polynomials (Fourier series) for periodic problems or Chebyshev polynomials for non-periodic ones. In stark contrast to local schemes, spectral methods are *holistic*: the derivative at any single point depends on the function values at *every other point* in the domain. Mathematically, this is equivalent to fitting a single high-degree polynomial through all data points. This global coupling is what allows information to propagate instantly across the grid, granting us the remarkable convergence that we call "spectral accuracy." The theory and practice of spectral methods are comprehensively developed in the classical texts by [1], [2], and [3].

1.1 The Spectral Promise

The fundamental argument for spectral methods is one of efficiency. If the solution to a problem is smooth, the coefficients of its expansion in a proper global basis decay exponentially fast. This phenomenon is known as spectral accuracy.

In practical terms, this means that spectral methods can achieve a level of precision with a few dozen degrees of freedom that a finite difference scheme might require thousands of grid points to match. While a fourth-order finite difference method implies that the error $\varepsilon \sim O(N^{-4})$, a spectral method boasts $\varepsilon \sim O(c^{-N})$ for some constant $c > 1$. When the solution is analytic, the convergence is explosive; the error drops into the "spectral valley" until it hits the floor of machine precision.

This global dependence has a computational consequence: spectral differentiation matrices are *dense*, not sparse. Where a finite difference scheme produces banded matrices that are cheap to store and invert, spectral methods fill in every entry. However, the extraordinary accuracy means we need so few points (often just dozens where finite differences would require thousands) that we can afford this density. The cost per degree of freedom is higher, but the total cost for a given accuracy is dramatically lower.

However, this power is not without its price. Spectral methods are unforgiving regarding grid placement. We cannot simply choose points where we please; for non-periodic problems, the mathematics dictates that points must cluster at boundaries (the celebrated Chebyshev points) to prevent the interpolation from diverging. Attempting high-degree polynomial interpolation on an equispaced grid leads to the notorious Runge phenomenon, where oscillations grow

without bound near the boundaries. This sensitivity to geometry is what restricts spectral methods primarily to simple domains, but within those domains, they reign supreme.

This book aims to demystify this "spectral magic." We will see that it is not magic at all, but a direct consequence of the smoothness of the underlying functions and the careful choice of basis and grid.

1.2 The Philosophy of "Études"

The title of this volume, Computational Études, reflects a specific pedagogical philosophy. In musical education, an étude is a composition designed to practice a particular technical skill (be it rapid scales or complex arpeggios) while remaining a pleasing piece of music in its own right.

In this text, our "technical skills" are not rapid scales or arpeggios, but rather handling stiffness in time-stepping, managing aliasing in nonlinear products, enforcing boundary conditions through tau methods or lifting functions, and filtering spurious oscillations. Just as a Chopin Étude transforms a technical exercise into art, a well-written spectral code transforms a mathematical formula into a robust simulation. The études collected here are designed to cultivate this virtuosity.

We approach spectral methods not through dry, abstract theorems, but through concrete, self-contained studies. Each chapter focuses on a specific mathematical concept (interpolation, differentiation, aliasing, or time-stepping) and explores it through a compact, runnable implementation.

We deliberately restrict our focus primarily to one-dimensional problems. This choice is strategic. The mathematical essence of spectral methods (the treatment of boundaries, the distribution of collocation points, and the structure of differentiation matrices) is fully present in one dimension. Extending these ideas to two or three dimensions usually involves tensor products, which add significant programming overhead without necessarily adding new conceptual depth. By staying in 1D, we keep our code short, readable, and focused on the physics and mathematics.

1.3 Collocation: Computing in Physical Space

While the theory of spectral methods relies on orthogonal expansions (Fourier series for periodic problems, Chebyshev series otherwise), the actual computation often proceeds differently. Rather than manipulating expansion coefficients directly (the *modal* or *Galerkin* approach), we typically work with function values at carefully chosen grid points (the *nodal* or *collocation* approach, also called *pseudospectral*).

The collocation philosophy dominates this book for a practical reason: it handles nonlinear terms with ease. When the governing equation contains products like $u \cdot u_x$, the modal approach requires computing convolutions of coefficient sequences, a tedious operation. Collocation simply evaluates the product pointwise on the grid. This directness makes pseudospectral methods the tool of choice for most computational applications, and it is the approach we shall master through these études.

The reader should be aware that both viewpoints (modal and nodal) illuminate the same underlying mathematics. The Fast Fourier Transform provides the bridge, allowing us to move efficiently between coefficient space and physical space as needed.

1.4 A Modern Workflow

Finally, this book is an experiment in reproducible science. The days of presenting numerical results as static, unverifiable images are passing. The results you see in these pages were generated by the code available in the accompanying repository. We utilize a dual-language approach:

- **Python**: For accessibility and integration with the vast open-source scientific ecosystem.
- **Matlab**: For its historical significance in this field and its concise matrix syntax, often utilizing the Advanpix Multiprecision Computing Toolbox to explore phenomena that lie beyond standard double precision.

We invite you to treat this book not as a static reference, but as a workshop. Run the scripts, change the parameters, break the code, and fix it. That is the only way to truly learn the art of spectral methods.

CHAPTER 2

Classical Second Order PDEs and Separation of Variables

In this opening chapter we derive exact solutions for three classical linear partial differential equations: the *heat equation* (parabolic), the *wave equation* (hyperbolic), and the *Laplace equation* (elliptic). These solutions are found by the *method of separation of variables*, which expresses the solution as an infinite series of eigenfunctions.
Why begin a book on *numerical* methods with *analytical* solutions? Because separation of variables is the theoretical ancestor of spectral methods. When we later truncate these infinite series at some finite N and compute with only the first N modes, we are doing exactly what a spectral solver does, but with pen and paper first. This chapter thus serves as the conceptual bridge between classical analysis and modern computation. The methods presented here are classical and thoroughly developed in standard references such as [4].

We treat three model problems:

- heat equation with periodic boundary conditions in one spatial dimension,
- wave equation on a bounded interval,
- Laplace equation in a simple domain.

We begin with a complete analytic solution of the heat equation. The other two examples will follow the same pattern.

2.1 Heat Equation with Periodic Boundary Conditions

We consider the one dimensional heat equation on the interval $[0, 2\pi]$ with periodic boundary conditions. The unknown $u(x, t)$ represents, for example, the temperature at point x and time t.

The problem is

$$\frac{\partial u}{\partial t}(x,t) = \frac{\partial^2 u}{\partial x^2}(x,t), \quad x \in [0, 2\pi],\ t > 0,$$

with periodic boundary conditions

$$u(x + 2\pi, t) = u(x, t)$$

for all real x and all $t > 0$, and initial condition

$$u(x, 0) = f(x), \quad x \in [0, 2\pi].$$

We assume that f is smooth and 2π periodic:

$$f(x + 2\pi) = f(x).$$

Our goal is to obtain an explicit representation of $u(x, t)$ as an infinite series and to see how separation of variables leads naturally to a Fourier series in space. The technique we employ here follows the classical approach presented in [4].

2.1.1 Step 1: Separation Ansatz

We look for nontrivial solutions of the form

$$u(x, t) = X(x) \cdot T(t),$$

where X depends only on x and T depends only on t.

Substituting into the PDE gives

$$X(x) \cdot T'(t) = X''(x) \cdot T(t).$$

We assume X and T are not identically zero, so we can divide both sides by $X(x) \cdot T(t)$:

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)}.$$

The left side depends only on t, the right side only on x. Therefore both sides must be equal to the same constant, which we denote by $-\lambda$:

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} = -\lambda.$$

We obtain two ordinary differential equations:

$$T'(t) + \lambda T(t) = 0,$$
$$X''(x) + \lambda X(x) = 0.$$

The periodic boundary conditions for u imply periodic conditions for X:

$$X(0) = X(2\pi), \quad X'(0) = X'(2\pi).$$

We have arrived at a spatial eigenvalue problem for X. The systematic treatment of such eigenvalue problems is a cornerstone of the theory of partial differential equations; see [4] for a comprehensive exposition.

2.1.2 Step 2: Spatial Eigenvalue Problem with Periodic Boundary Conditions

We now solve

$$X''(x) + \lambda X(x) = 0,$$

with

$$X(0) = X(2\pi), \quad X'(0) = X'(2\pi).$$

We consider three cases: $\lambda < 0$, $\lambda = 0$, and $\lambda > 0$.

Case 1: $\lambda < 0$

Write $\lambda = -\mu^2$ with $\mu > 0$. The equation becomes

$$X''(x) - \mu^2 X(x) = 0.$$

The general solution is

$$X(x) = Ae^{\mu x} + Be^{-\mu x}.$$

Imposing periodicity $X(0) = X(2\pi)$ gives

$$A + B = Ae^{2\mu\pi} + Be^{-2\mu\pi}.$$

Imposing $X'(0) = X'(2\pi)$ gives

$$\mu(A - B) = \mu(Ae^{2\mu\pi} - Be^{-2\mu\pi}).$$

Since $\mu > 0$, we can divide by μ and rewrite both conditions as a homogeneous linear system:

$$A(1 - e^{2\mu\pi}) + B(1 - e^{-2\mu\pi}) = 0,$$
$$A(1 - e^{2\mu\pi}) - B(1 - e^{-2\mu\pi}) = 0.$$

Adding these two equations gives

$$2A(1 - e^{2\mu\pi}) = 0.$$

Since $\mu > 0$, we have $e^{2\mu\pi} > 1$, so $1 - e^{2\mu\pi} \neq 0$. Therefore $A = 0$.

Subtracting the second equation from the first gives

$$2B(1 - e^{-2\mu\pi}) = 0.$$

Since $\mu > 0$, we have $e^{-2\mu\pi} < 1$, so $1 - e^{-2\mu\pi} \neq 0$. Therefore $B = 0$.

Hence the only solution is $A = B = 0$, the trivial solution. There are no nontrivial periodic eigenfunctions for $\lambda < 0$, and we discard this case.

Case 2: $\lambda = 0$

The equation reduces to

$$X''(x) = 0.$$

Its general solution is

$$X(x) = A + Bx.$$

Periodicity $X(0) = X(2\pi)$ gives

$$A = A + 2\pi B$$

so $B = 0$. Then $X(x) = A$ is constant.

The derivative is $X'(x) = 0$, so $X'(0) = X'(2\pi)$ is automatically satisfied.

Thus $\lambda = 0$ gives one eigenfunction

$$X_0(x) = 1$$

(up to a multiplicative constant).

Case 3: $\lambda > 0$

Write $\lambda = k^2$ with $k > 0$. The equation becomes

$$X''(x) + k^2 X(x) = 0.$$

The general solution is

$$X(x) = A\cos(kx) + B\sin(kx).$$

We now impose periodicity. First,

$$X(0) = A, \quad X(2\pi) = A\cos(2\pi k) + B\sin(2\pi k).$$

The condition $X(0) = X(2\pi)$ gives

$$A = A\cos(2\pi k) + B\sin(2\pi k).$$

Next,

$$X'(x) = -Ak\sin(kx) + Bk\cos(kx),$$

so

$$X'(0) = Bk, \quad X'(2\pi) = -Ak\sin(2\pi k) + Bk\cos(2\pi k).$$

The condition $X'(0) = X'(2\pi)$ gives

$$Bk = -Ak\sin(2\pi k) + Bk\cos(2\pi k).$$

We can divide by k the last identity (for $k \neq 0$) and write the system as

$$A(1 - \cos(2\pi k)) - B\sin(2\pi k) = 0,$$

$$A\sin(2\pi k) + B(1 - \cos(2\pi k)) = 0.$$

For a nontrivial pair (A, B) the determinant of the system must vanish:

$$(1 - \cos(2\pi k))^2 + (\sin(2\pi k))^2 = 0.$$

The left side is a sum of squares, so it is zero if and only if

$$1 - \cos(2\pi k) = 0, \quad \sin(2\pi k) = 0.$$

Hence

$$\cos(2\pi k) = 1, \quad \sin(2\pi k) = 0.$$

This happens exactly when k is an integer:

$$k = n, \quad n \in \mathbb{Z}.$$

The case $n = 0$ corresponds to $\lambda = 0$, which we have already treated. For $n \geqslant 1$ we obtain eigenvalues

$$\lambda_n = n^2, \quad n = 1, 2, 3, ...$$

For each $n \geqslant 1$ the corresponding eigenfunctions can be chosen as

$$X_n^{(c)}(x) = \cos(nx), \quad X_n^{(s)}(x) = \sin(nx).$$

These functions are 2π periodic, and their derivatives are also 2π periodic, so the boundary conditions are satisfied.

We have therefore found a complete set of spatial eigenfunctions for the heat equation with periodic boundary conditions:

- a constant mode $X_0(x) = 1$ (eigenvalue $\lambda_0 = 0$),
- cosine modes $X_n^{(c)}(x) = \cos(nx)$,
- sine modes $X_n^{(s)}(x) = \sin(nx)$,

with eigenvalues $\lambda_n = n^2$ for $n \geqslant 1$.

2.1.3 Step 3: Time Dependent Factors

For each eigenvalue λ the corresponding time factor satisfies

$$T'(t) + \lambda T(t) = 0.$$

The solution is

$$T(t) = Ce^{-\lambda t}.$$

For the constant mode $\lambda_0 = 0$ we obtain

$$T_0(t) = C_0$$

(constant in time).

For $n \geqslant 1$ we have

$$T_n(t) = C_n e^{-n^2 t}.$$

Combining space and time, we form products $u(x,t) = X(x) \cdot T(t)$ to obtain separated solutions. For each mode, the arbitrary constant from the time factor can be absorbed into a single new constant. Specifically, we write

$$u_0(x,t) = A_0,$$
$$u_n^{(c)}(x,t) = A_n \cos(nx) \cdot e^{-n^2 t},$$
$$u_n^{(s)}(x,t) = B_n \sin(nx) \cdot e^{-n^2 t},$$

where we have renamed the constants: $A_0 = C_0$, and for $n \geqslant 1$, the constant A_n absorbs the coefficient from the cosine mode while B_n absorbs the coefficient from the sine mode. Note that each separated solution carries its own independent arbitrary constant.

Since the heat equation is linear, any linear combination of these separated solutions is again a solution. Therefore the general solution that satisfies the periodic boundary conditions can be written as an infinite series

$$u(x,t) = A_0 + \sum_{n=1}^{\infty} (A_n \cos(nx) + B_n \sin(nx)) e^{-n^2 t}.$$

The coefficients A_0, A_n, B_n remain to be determined from the initial condition.

2.1.4 Step 4: Imposing the Initial Condition and Fourier Series

We impose the initial condition

$$u(x,0) = f(x).$$

Setting $t = 0$ in the general solution we obtain

$$u(x,0) = A_0 + \sum_{n=1}^{\infty} (A_n \cos(nx) + B_n \sin(nx)) = f(x).$$

This is exactly the Fourier series expansion of the 2π periodic function f. Under mild regularity assumptions, f has a Fourier series

$$f(x) = a_0 + \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx))$$

with Fourier coefficients

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x)\, \mathrm{d}x,$$

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(nx)\, \mathrm{d}x, \quad n \geqslant 1,$$

$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(nx)\,\mathrm{d}x, \quad n \geqslant 1.$$

By uniqueness of the Fourier expansion, we must have

$$A_0 = a_0, \quad A_n = a_n, \quad B_n = b_n.$$

Thus the coefficients in the heat equation solution are exactly the Fourier coefficients of the initial data.

2.1.5 Step 5: Final Explicit Solution and Interpretation

Substituting these coefficients into the general solution we obtain the explicit formula

$$u(x,t) = a_0 + \sum_{n=1}^{\infty}(a_n\cos(nx) + b_n\sin(nx))e^{-n^2t}, \quad t > 0.$$

This infinite sum solves the heat equation with periodic boundary conditions and initial data f. Each Fourier mode decays exponentially in time at a rate proportional to its eigenvalue n^2. High frequency modes (large n) decay faster, which expresses the smoothing effect of the heat equation.

The constant term a_0 does not decay. It represents the average value of f on $[0, 2\pi]$, which is preserved by the heat flow.

From a spectral viewpoint, the functions

$$1, \cos(x), \sin(x), \cos(2x), \sin(2x), \ldots$$

form an eigenbasis of the spatial operator

$$Lu = \frac{\partial^2 u}{\partial x^2}$$

with periodic boundary conditions. The evolution of each eigenmode is independent and given simply by multiplication by e^{-n^2t} in time.

Later, in the numerical part of this book, we will approximate $u(x,t)$ by truncating the sum to finitely many modes:

$$u_{N(x,t)} = a_0 + \sum_{n=1}^{N}(a_n\cos(nx) + b_n\sin(nx))e^{-n^2t}.$$

This truncation is the essence of a Fourier spectral method. The analytic solution derived here is the infinite dimensional limit of that numerical procedure.

2.2 Numerical Illustration

To visualize the smoothing effect of heat diffusion, we compute the truncated Fourier series solution for a triangle wave initial condition:

$$f(x) = \pi - |x - \pi|, \quad x \in [0, 2\pi].$$

This function is continuous but has a corner (non-differentiable point) at $x = \pi$. Its Fourier series contains only cosine terms with coefficients decaying as $1/n^2$:

$$f(x) = \frac{\pi}{2} + \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\cos((2k-1)x)}{(2k-1)^2}.$$

The key portion of the implementation evaluates the truncated Fourier series at any point in space and time. In Python:

```python
def heat_solution(x, t, a0, a_n, b_n):
```

```
    u = np.full_like(x, a0, dtype=float)
    n_modes = len(a_n) - 1
    for n in range(1, n_modes + 1):
        decay = np.exp(-n**2 * t)
        u += (a_n[n] * np.cos(n * x) + b_n[n] * np.sin(n * x)) * decay
    return u
```

The equivalent MATLAB implementation:

```
u = a0 * ones(size(x));
for n = 1:N_MODES
    decay = exp(-n^2 * t);
    u = u + (a_n(n+1) * cos(n*x) + b_n(n+1) * sin(n*x)) * decay;
end
```

Figure 1 shows the evolution of $u_N(x,t)$ with $N = 50$ modes at several time values. At $t = 0$ the triangle wave is faithfully reproduced. As time increases, the higher frequency modes decay exponentially faster than the lower ones (the n-th mode decays as $e^{-n^2 t}$), and the solution rapidly smooths toward the constant equilibrium $u = \pi/2$.

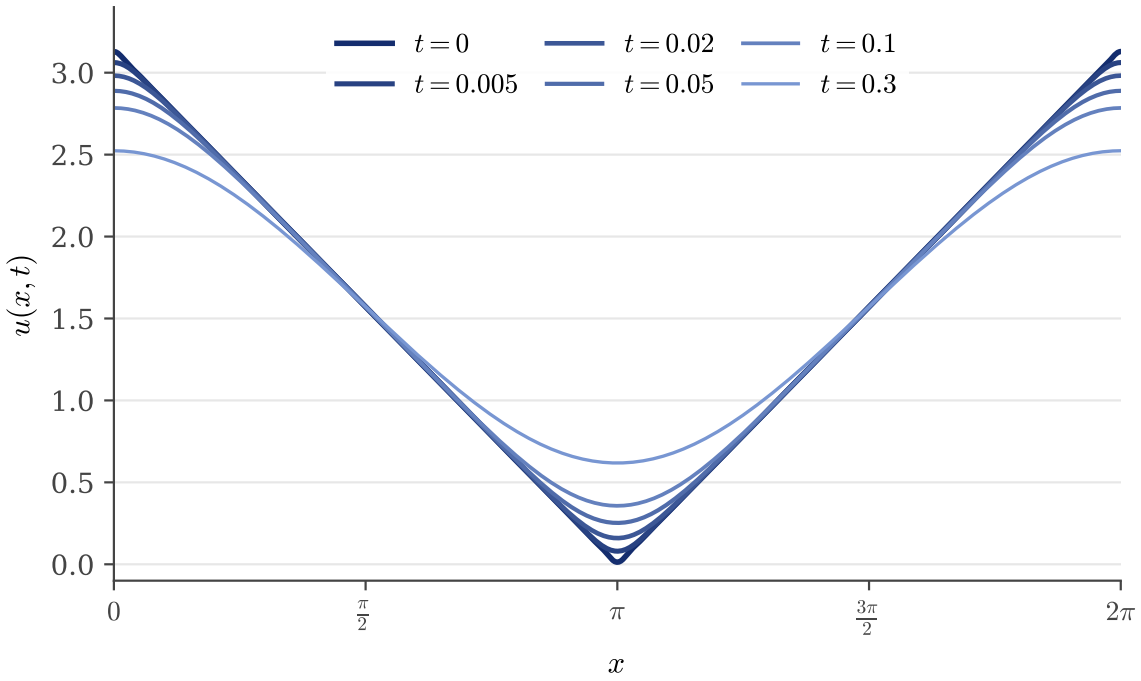

Figure 1: *Evolution of the heat equation solution with a triangle wave initial condition. The higher frequency modes decay rapidly, smoothing the initial corner at $x = \pi$.*

The code that generated this figure is available in both Python and MATLAB:

- `codes/python/ch02_classical_pdes/heat_equation_evolution.py`
- `codes/matlab/ch02_classical_pdes/heat_equation_evolution.m`

A complementary view of the solution is provided by the waterfall plot in Figure 2, which displays the entire space-time evolution as a three-dimensional surface. The smoothing effect of the heat equation is clearly visible: the initial sharp triangle wave rapidly flattens as time progresses, with the solution approaching the constant equilibrium state $u = \pi/2$.

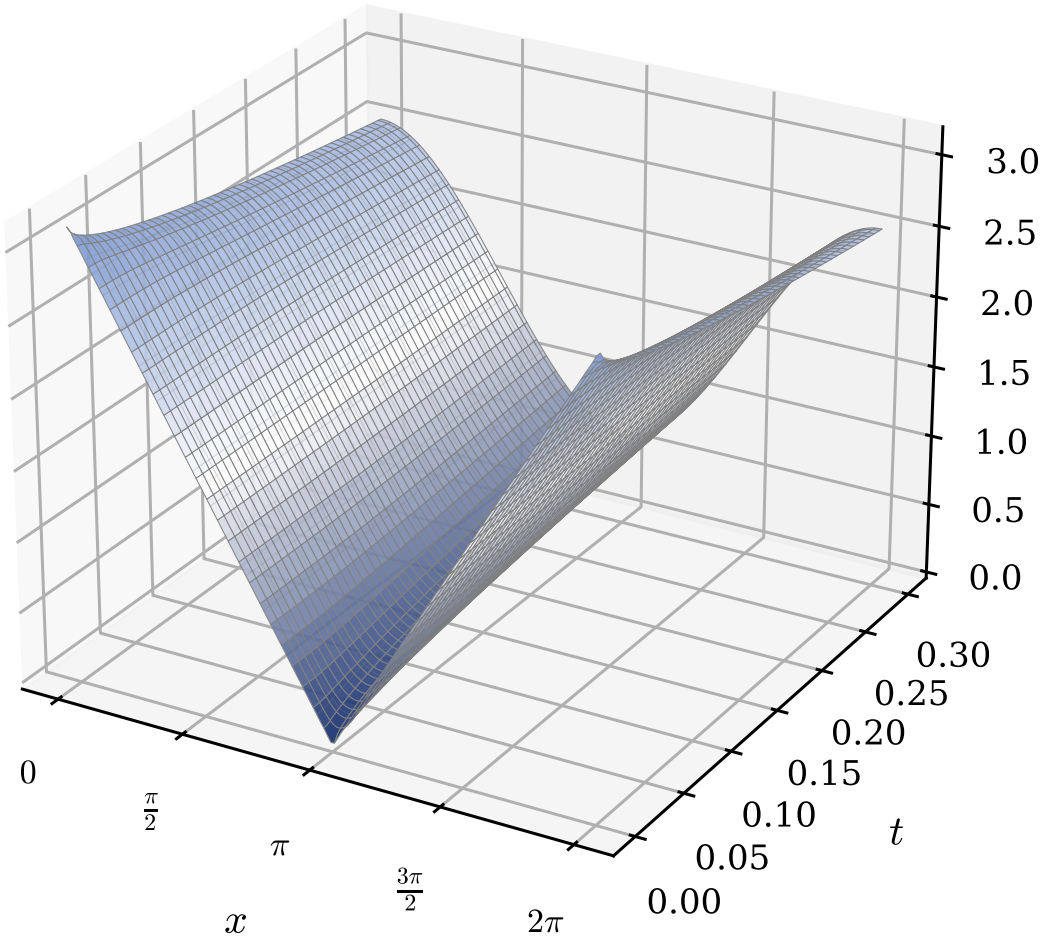

Figure 2: *Waterfall plot showing the complete space-time evolution of the heat equation solution. The initial triangle wave smooths rapidly as higher frequency modes decay exponentially.*

2.3 Wave Equation with Dirichlet Boundary Conditions

We now consider the one dimensional wave equation on a finite interval with homogeneous Dirichlet boundary conditions. This is the classical model for a vibrating string of length L with both ends fixed.

Let $u(x,t)$ denote the vertical displacement of the string at position $x \in [0,L]$ and time $t > 0$. The equation of motion is

$$\frac{\partial^2 u}{\partial t^2}(x,t) = c^2 \frac{\partial^2 u}{\partial x^2}(x,t), \quad 0 < x < L,\ t > 0,$$

where $c > 0$ is the wave speed.

The boundary conditions express that the endpoints of the string are clamped:

$$u(0,t) = 0, \quad u(L,t) = 0, \quad t > 0.$$

We prescribe the initial displacement and initial velocity:

$$u(x,0) = f(x), \quad \frac{\partial u}{\partial t}(x,0) = g(x), \quad 0 < x < L,$$

with suitable functions f and g that vanish at $x = 0$ and $x = L$.

As in the heat equation example, we use separation of variables and obtain a representation of the solution as an infinite series in spatial eigenfunctions. This time the temporal factors are oscillatory instead of decaying. The mathematical theory of the vibrating string is a classical topic covered extensively in [4].

2.3.1 Step 1: Separation Ansatz

We look for nontrivial solutions of the form

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

Substituting into the wave equation gives

$$X(x) \cdot T''(t) = c^2 X''(x) \cdot T(t).$$

Assuming X and T are not identically zero, we divide both sides by $c^2 X(x) \cdot T(t)$:

$$\frac{T''(t)}{c^2 T(t)} = \frac{X''(x)}{X(x)}.$$

The left side depends only on t, the right side only on x. Therefore both sides must be equal to a constant, which we denote by $-\lambda$:

$$\frac{T''(t)}{c^2 T(t)} = \frac{X''(x)}{X(x)} = -\lambda.$$

We obtain the pair of ordinary differential equations

$$T''(t) + c^2 \lambda T(t) = 0,$$
$$X''(x) + \lambda X(x) = 0,$$

with boundary conditions

$$X(0) = 0, \quad X(L) = 0.$$

As in the heat equation case, we have a spatial eigenvalue problem for X.

2.3.2 Step 2: Spatial Eigenvalue Problem with Dirichlet Boundary Conditions

We must solve

$$X''(x) + \lambda X(x) = 0, \quad 0 < x < L,$$

with

$$X(0) = 0, \quad X(L) = 0.$$

We again consider three cases: $\lambda < 0$, $\lambda = 0$, and $\lambda > 0$.

Case 1: $\lambda < 0$

Write $\lambda = -\mu^2$ with $\mu > 0$. The equation becomes

$$X''(x) - \mu^2 X(x) = 0.$$

The general solution is

$$X(x) = Ae^{\mu x} + Be^{-\mu x}.$$

The boundary condition at $x = 0$ gives

$$X(0) = A + B = 0 \quad \Rightarrow \quad B = -A.$$

Then

$$X(L) = Ae^{\mu L} - Ae^{-\mu L} = A(e^{\mu L} - e^{-\mu L}).$$

The condition $X(L) = 0$ implies

$$A(e^{\mu L} - e^{-\mu L}) = 0.$$

Since $e^{\mu L} \neq e^{-\mu L}$ for $\mu > 0$, we must have $A = 0$. Then $B = 0$ and the solution is trivial. Therefore there are no nontrivial eigenfunctions for $\lambda < 0$.

Case 2: $\lambda = 0$

The equation reduces to

$$X''(x) = 0,$$

whose general solution is

$$X(x) = A + Bx.$$

The boundary conditions give

$$X(0) = A = 0,$$
$$X(L) = A + BL = BL = 0.$$

Hence $B = 0$ and X is again trivial. There is no nontrivial eigenfunction for $\lambda = 0$.

Case 3: $\lambda > 0$

Write $\lambda = k^2$ with $k > 0$. The equation becomes

$$X''(x) + k^2 X(x) = 0.$$

The general solution is

$$X(x) = A\cos(kx) + B\sin(kx).$$

The boundary condition at $x = 0$ gives

$$X(0) = A = 0.$$

So $X(x) = B\sin(kx)$. The boundary condition at $x = L$ gives

$$X(L) = B\sin(kL) = 0.$$

For a nontrivial solution we need $B \neq 0$, so we must have

$$\sin(kL) = 0.$$

Therefore

$$kL = n\pi, \quad n = 1, 2, 3, ...$$

The corresponding values of k are

$$k_n = \frac{n\pi}{L}, \quad n = 1, 2, 3, ...$$

We conclude that the eigenvalues and eigenfunctions are

$$\lambda_n = k_n^2 = \left(\frac{n\pi}{L}\right)^2,$$
$$X_n(x) = \sin\left(\frac{n\pi x}{L}\right), \quad n = 1, 2, 3, ...$$

Each X_n vanishes at $x = 0$ and $x = L$, as required by the Dirichlet boundary conditions. The family $\{X_n\}_{n\geqslant 1}$ is orthogonal in $L^2(0, L)$:

$$\int_0^L \sin\left(\frac{n\pi x}{L}\right)\sin\left(\frac{m\pi x}{L}\right) \mathrm{d}x = \begin{cases} 0 & \text{if } n \neq m \\ L/2 & \text{if } n = m. \end{cases}$$

These eigenfunctions will form the spatial basis in our series solution.

2.3.3 Step 3: Time Dependent Factors

For each eigenvalue λ_n the time factor T_n satisfies

$$T_n''(t) + c^2\lambda_n T_n(t) = 0.$$

Using $\lambda_n = (n\pi/L)^2$ we can write

$$T_n''(t) + \omega_n^2 T_n(t) = 0,$$

where

$$\omega_n = c\frac{n\pi}{L}, \quad n = 1, 2, 3, ...$$

The general solution of this second order linear ODE is

$$T_n(t) = A_n \cos(\omega_n t) + B_n \sin(\omega_n t),$$

where A_n and B_n are constants.

Combining the space and time factors, we obtain separated solutions

$$u_n(x,t) = (A_n \cos(\omega_n t) + B_n \sin(\omega_n t)) \sin\Big(\frac{n\pi x}{L}\Big), \quad n = 1, 2, 3, ...$$

Because the wave equation is linear, any linear combination of these separated solutions is again a solution. Therefore the general solution satisfying the Dirichlet boundary conditions can be written as an infinite series

$$u(x,t) = \sum_{n=1}^{\infty} (a_n \cos(\omega_n t) + b_n \sin(\omega_n t)) \sin\Big(\frac{n\pi x}{L}\Big),$$

for suitable coefficients a_n and b_n.

These coefficients will be determined from the initial conditions.

2.3.4 Step 4: Imposing the Initial Conditions and Sine Series

We now use the initial displacement and velocity.

At $t = 0$ we have

$$u(x,0) = \sum_{n=1}^{\infty} (a_n \cos(0) + b_n \sin(0)) \sin\Big(\frac{n\pi x}{L}\Big) = \sum_{n=1}^{\infty} a_n \sin\Big(\frac{n\pi x}{L}\Big).$$

The initial condition $u(x,0) = f(x)$ becomes

$$f(x) = \sum_{n=1}^{\infty} a_n \sin\Big(\frac{n\pi x}{L}\Big).$$

This is the Fourier sine series of f on the interval $(0, L)$.

Similarly, we differentiate u with respect to t:

$$\frac{\partial u}{\partial t}(x,t) = \sum_{n=1}^{\infty} (-a_n \omega_n \sin(\omega_n t) + b_n \omega_n \cos(\omega_n t)) \sin\Big(\frac{n\pi x}{L}\Big).$$

Evaluating at $t = 0$ gives

$$\frac{\partial u}{\partial t}(x,0) = \sum_{n=1}^{\infty} b_n \omega_n \sin\Big(\frac{n\pi x}{L}\Big).$$

The initial condition $\frac{\partial u}{\partial t}(x,0) = g(x)$ becomes

$$g(x) = \sum_{n=1}^{\infty} b_n \omega_n \sin\Big(\frac{n\pi x}{L}\Big).$$

So the sequence $\{a_n\}$ consists of the Fourier sine coefficients of f, and the sequence $\{b_n \omega_n\}$ consists of the Fourier sine coefficients of g.

Using the orthogonality relations, we obtain explicit formulas for the coefficients. For $n \geqslant 1$,

$$a_n = \frac{2}{L} \int_0^L f(x) \sin\Big(\frac{n\pi x}{L}\Big)\, \mathrm{d}x,$$

and

$$b_n \omega_n = \frac{2}{L} \int_0^L g(x) \sin\Big(\frac{n\pi x}{L}\Big)\, \mathrm{d}x.$$

Therefore

$$b_n = \frac{2}{L\omega_n} \int_0^L g(x) \sin\Big(\frac{n\pi x}{L}\Big)\, \mathrm{d}x = \frac{2}{L} \frac{1}{cn\pi/L} \int_0^L g(x) \sin\Big(\frac{n\pi x}{L}\Big)\, \mathrm{d}x.$$

In summary,

$$a_n = \frac{2}{L}\int_0^L f(x)\sin\Big(\frac{n\pi x}{L}\Big)\,\mathrm{d}x,$$
$$b_n = \frac{2}{L\omega_n}\int_0^L g(x)\sin\Big(\frac{n\pi x}{L}\Big)\,\mathrm{d}x, \quad \omega_n = c\frac{n\pi}{L}.$$

2.3.5 Step 5: Final Explicit Solution and Interpretation

Substituting these coefficients into the series, we obtain the explicit solution of the wave equation with Dirichlet boundary conditions:

$$u(x,t) = \sum_{n=1}^{\infty}[a_n\cos(\omega_n t) + b_n\sin(\omega_n t)]\sin\Big(\frac{n\pi x}{L}\Big),$$

where $\omega_n = cn\pi/L$ and the Fourier sine coefficients are

$$a_n = \frac{2}{L}\int_0^L f(y)\sin\Big(\frac{n\pi y}{L}\Big)\,\mathrm{d}y, \quad b_n = \frac{2}{n\pi c}\int_0^L g(y)\sin\Big(\frac{n\pi y}{L}\Big)\,\mathrm{d}y.$$

Each term in the sum is a normal mode of vibration: a standing wave with spatial shape $\sin(n\pi x/L)$ and temporal oscillation at frequency ω_n. The coefficients of $\cos(\omega_n t)$ and $\sin(\omega_n t)$ are determined by the initial displacement f and initial velocity g through their Fourier sine coefficients.

Comparing with the heat equation:

- For the heat equation, each mode decayed like $e^{-n^2 t}$ and the solution became smoother in time.
- For the wave equation, each mode oscillates periodically in time with constant amplitude, reflecting conservation of energy in the undamped string.

From a spectral viewpoint, the functions

$$\sin\Big(\frac{\pi x}{L}\Big), \sin\Big(\frac{2\pi x}{L}\Big), \sin\Big(\frac{3\pi x}{L}\Big), \ldots$$

form an eigenbasis of the spatial operator

$$Lu = \frac{\partial^2 u}{\partial x^2}$$

with Dirichlet boundary conditions. In this basis, the evolution is diagonal: each mode evolves independently according to a simple harmonic oscillator in time.

As in the heat equation example, a spectral method will approximate $u(x,t)$ by truncating the infinite sum. For some integer $N \geqslant 1$ we consider the finite approximation

$$u_N(x,t) = \sum_{n=1}^{N}(a_n\cos(\omega_n t) + b_n\sin(\omega_n t))\sin\Big(\frac{n\pi x}{L}\Big).$$

The analytic series above is the infinite dimensional limit of this spectral representation.

2.4 Numerical Illustration

To visualize the oscillatory behavior of the vibrating string, we compute the truncated Fourier sine series solution for a plucked string initial condition. The string is plucked at its center, forming a triangular initial displacement:

$$f(x) = \begin{cases} \frac{2h}{L}x & \text{for } 0 \leqslant x \leqslant L/2 \\ 2h(1 - x/L) & \text{for } L/2 \leqslant x \leqslant L \end{cases}$$

with zero initial velocity $g(x) = 0$. Here h denotes the height of the pluck at the center.

The Fourier sine coefficients of this triangular shape are

$$a_n = \frac{8h}{n^2\pi^2} \sin\left(\frac{n\pi}{2}\right),$$

which gives nonzero values only for odd n, with alternating signs.

The key portion of the implementation computes the solution at any point in space and time. In Python:

```python
def wave_solution(x, t, a_n, b_n, L, c):
    u = np.zeros_like(x, dtype=float)
    n_modes = len(a_n) - 1
    for n in range(1, n_modes + 1):
        omega_n = c * n * np.pi / L
        spatial = np.sin(n * np.pi * x / L)
        temporal = a_n[n] * np.cos(omega_n * t) + b_n[n] * np.sin(omega_n * t)
        u += temporal * spatial
    return u
```

The equivalent MATLAB implementation:

```matlab
u = zeros(size(x));
for n = 1:N_MODES
    omega_n = C * n * pi / L;
    spatial = sin(n * pi * x / L);
    temporal = a_n(n+1) * cos(omega_n * t) + b_n(n+1) * sin(omega_n * t);
    u = u + temporal * spatial;
end
```

Figure 3 shows the evolution of $u_N(x, t)$ with $N = 50$ modes at several time values within half a period $T = 2L/c$. The string oscillates back and forth, with the triangular shape inverting at $t = T/2$. Unlike the heat equation, the wave equation preserves energy and the solution does not decay; it continues oscillating indefinitely.

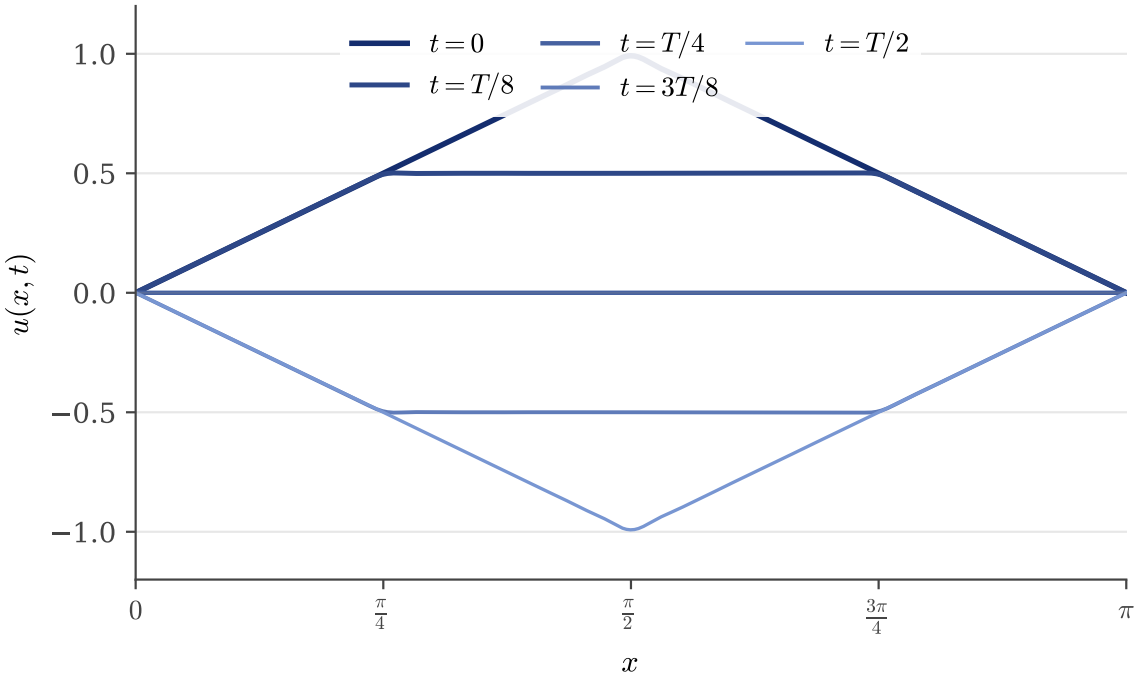

Figure 3: *Evolution of the wave equation solution with a plucked string initial condition. The string oscillates with period $T = 2\pi$, inverting at $t = T/2$.*

The code that generated this figure is available in both Python and MATLAB:

- `codes/python/ch02_classical_pdes/wave_equation_evolution.py`
- `codes/matlab/ch02_classical_pdes/wave_equation_evolution.m`

The waterfall plot in Figure 4 provides a complete view of the oscillatory dynamics over one full period. Unlike the heat equation, the wave equation conserves energy: the solution oscillates indefinitely without decay, and the periodic nature of the motion is clearly visible in the three-dimensional representation.

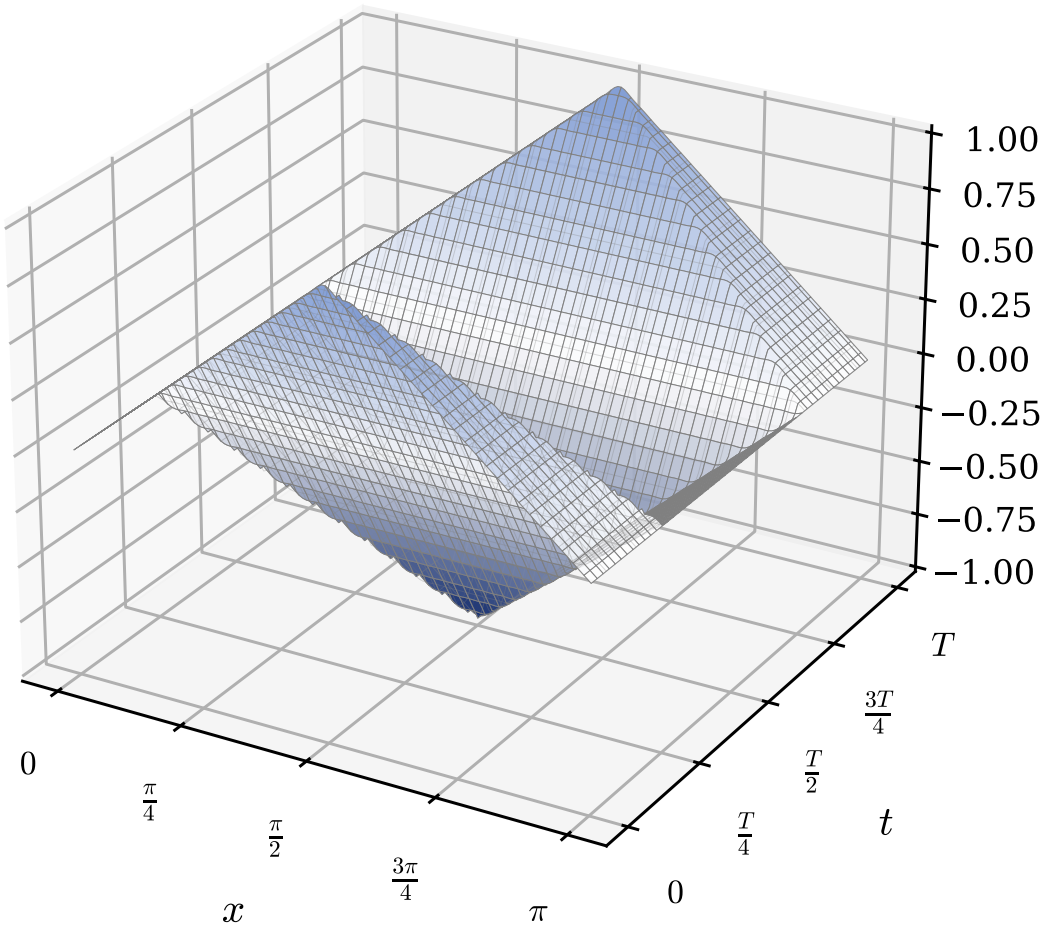

Figure 4: *Waterfall plot showing the complete space-time evolution of the wave equation solution over one period T. The plucked string oscillates between its initial shape and its mirror image.*

2.5 Laplace Equation in a Periodic Strip

For the elliptic case we consider the Laplace equation in a simple two dimensional domain that is periodic in one direction and bounded in the other. This setting connects naturally with the periodic heat equation example and again leads to a Fourier series representation in the periodic direction.

Let

$$D = \{(x, y) \in \mathbb{R}^2 : 0 < x < 2\pi,\ 0 < y < 1\}.$$

We seek a harmonic function $u(x, y)$ solving

$$u_{xx}(x, y) + u_{yy}(x, y) = 0, \quad (x, y) \in D,$$

with periodic boundary conditions in x

$$u(x + 2\pi, y) = u(x, y), \quad \text{for all real } \ x,\ 0 < y < 1,$$

and Dirichlet conditions in y

$$u(x,0) = f(x), \quad u(x,1) = 0, \quad 0 \leqslant x \leqslant 2\pi.$$

We assume that f is 2π periodic and smooth:

$$f(x + 2\pi) = f(x).$$

As in the parabolic and hyperbolic examples, we apply separation of variables and obtain a representation of u as an infinite Fourier series in x with y dependent coefficients. The theory of harmonic functions and Laplace's equation is presented in detail in [4].

2.5.1 Step 1: Separation Ansatz

We look for nontrivial separated solutions of the form

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

Substituting into the Laplace equation gives

$$X''(x) \cdot Y(y) + X(x) \cdot Y''(y) = 0.$$

Assuming X and Y are not identically zero, we divide by $X(x) \cdot Y(y)$:

$$\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = 0.$$

The first term depends only on x, the second only on y. Therefore both must be equal to constants whose sum is zero. We introduce a separation constant λ and write

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

This leads to the two ordinary differential equations

$$X''(x) + \lambda X(x) = 0,$$
$$Y''(y) - \lambda Y(y) = 0.$$

The periodic boundary conditions for u in the x direction imply the periodic conditions

$$X(x + 2\pi) = X(x), \quad \text{for all real} \quad x.$$

The Dirichlet conditions in y will be enforced later on the full series. For the separated functions Y we will impose appropriate conditions at $y = 1$, while the condition at $y = 0$ will be handled via the Fourier coefficients of f.

We have once more an eigenvalue problem in the periodic direction.

2.5.2 Step 2: Eigenfunctions in the Periodic Direction

The equation for X with periodic boundary conditions is exactly the same as in the heat equation example:

$$X''(x) + \lambda X(x) = 0, \quad X(x + 2\pi) = X(x).$$

We recall the result: there is a constant mode with eigenvalue $\lambda_0 = 0$,

$$X_0(x) = 1,$$

and for each integer $n \geqslant 1$ there are cosine and sine modes with eigenvalues

$$\lambda_n = n^2,$$
$$X_n^{(c)}(x) = \cos(nx), \quad X_n^{(s)}(x) = \sin(nx).$$

The family

$$1, \cos(x), \sin(x), \cos(2x), \sin(2x), \ldots$$

forms an orthogonal basis in $L^2(0, 2\pi)$ for 2π periodic functions.

For each such eigenvalue we now solve the corresponding equation for Y.

2.5.3 Step 3: Equations for the y Dependent Factors

For each eigenvalue λ we have

$$Y''(y) - \lambda Y(y) = 0.$$

We treat separately the constant mode $\lambda_0 = 0$ and the nonzero modes $\lambda_n = n^2$ for $n \geqslant 1$.

Constant mode $\lambda_0 = 0$

For $\lambda_0 = 0$ the equation reduces to

$$Y_0''(y) = 0.$$

The general solution is

$$Y_0(y) = A_0 + B_0 y.$$

We want separated solutions that satisfy the homogeneous boundary condition at $y = 1$:

$$u(x, 1) = 0 \quad \text{for all} \quad x.$$

For the constant mode this means

$$X_0(x) \cdot Y_0(1) = Y_0(1) = 0,$$

hence

$$Y_0(1) = A_0 + B_0 = 0.$$

We choose a convenient normalization so that $Y_0(0) = 1$. Then $A_0 = 1$ and the relation $A_0 + B_0 = 0$ gives $B_0 = -1$. Thus

$$Y_0(y) = 1 - y.$$

This separated mode

$$u_0(x, y) = X_0(x) \cdot Y_0(y) = 1 - y$$

is harmonic, periodic in x, and vanishes at $y = 1$, with value 1 at $y = 0$.

Higher modes $\lambda_n = n^2$ for $n \geqslant 1$

For $\lambda_n = n^2$ the equation is

$$Y_n''(y) - n^2 Y_n(y) = 0.$$

The general solution can be written in hyperbolic form

$$Y_n(y) = \alpha_n \cosh(ny) + \beta_n \sinh(ny).$$

We require that each separated mode vanish at $y = 1$:

$$Y_n(1) = 0.$$

To match later the Fourier coefficients of f at $y = 0$, it is convenient to normalize so that

$$Y_n(0) = 1.$$

Imposing $Y_n(0) = 1$ gives

$$\alpha_n = 1.$$

Then

$$Y_n(1) = \cosh(n) + \beta_n \sinh(n) = 0$$

so

$$\beta_n = -\frac{\cosh(n)}{\sinh(n)} = -\coth(n).$$

Thus

$$Y_n(y) = \cosh(ny) - \coth(n) \cdot \sinh(ny).$$

An alternative and simpler expression uses the hyperbolic sine function of the distance to the boundary $y = 1$. One checks that

$$Y_n(y) = \frac{\sinh(n(1-y))}{\sinh(n)}$$

satisfies

$$Y_n''(y) - n^2 Y_n(y) = 0,$$
$$Y_n(1) = 0,$$

$$Y_n(0) = 1.$$

Indeed, $Y_n(1) = \sinh(0)/\sinh(n) = 0$, $Y_n(0) = \sinh(n)/\sinh(n) = 1$, and

$$Y_n''(y) = n^2 \frac{\sinh(n(1-y))}{\sinh(n)} = n^2 Y_n(y).$$

We will use the form

$$Y_n(y) = \frac{\sinh(n(1-y))}{\sinh(n)}, \quad n \geqslant 1.$$

2.5.4 Step 4: Building the Series Solution

Each separated solution corresponding to the eigenfunctions in x and the functions Y_n in y has the form

$$u_0(x,y) = C_0 \cdot Y_0(y) = C_0(1-y),$$
$$u_n^{(c)}(x,y) = C_n \cdot \cos(nx) \cdot Y_n(y),$$
$$u_n^{(s)}(x,y) = D_n \cdot \sin(nx) \cdot Y_n(y), \quad n \geqslant 1,$$

for some constants C_0, C_n, D_n.

Since the Laplace equation is linear and the boundary condition at $y = 1$ is homogeneous, any linear combination of these separated solutions is again a solution that vanishes at $y = 1$. Therefore a general solution satisfying the periodic condition in x and the Dirichlet condition $u(x,1) = 0$ can be written as

$$u(x,y) = C_0(1-y) + \sum_{n=1}^{\infty} [C_n \cos(nx) + D_n \sin(nx)] \frac{\sinh(n(1-y))}{\sinh(n)}.$$

It remains to impose the boundary condition at $y = 0$,

$$u(x,0) = f(x).$$

At $y = 0$ we obtain

$$u(x,0) = C_0 + \sum_{n=1}^{\infty} [C_n \cos(nx) + D_n \sin(nx)],$$

because $Y_0(0) = 1$ and $Y_n(0) = 1$ for $n \geqslant 1$.

Hence the boundary condition $u(x,0) = f(x)$ becomes

$$f(x) = C_0 + \sum_{n=1}^{\infty} [C_n \cos(nx) + D_n \sin(nx)].$$

This is exactly the Fourier series expansion of f. For a 2π periodic f we have

$$f(x) = a_0 + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)],$$

with coefficients

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x)\,\mathrm{d}x,$$

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(nx)\,\mathrm{d}x, \quad n \geqslant 1,$$

$$b_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(nx)\,\mathrm{d}x, \quad n \geqslant 1.$$

By uniqueness of the Fourier expansion, we must have

$$C_0 = a_0, \quad C_n = a_n, \quad D_n = b_n.$$

2.5.5 Step 5: Final Explicit Solution and Interpretation

Substituting these coefficients into the series we obtain the explicit representation

$$u(x,y) = a_0(1-y) + \sum_{n=1}^{\infty}[a_n\cos(nx) + b_n\sin(nx)]\frac{\sinh(n(1-y))}{\sinh(n)}, \quad 0 < y < 1.$$

Here a_0, a_n, and b_n are the Fourier coefficients of the boundary data f as defined above.

This series converges (under mild assumptions on f) to the unique harmonic function that is periodic in x, equal to f on $y = 0$, and zero on $y = 1$.

From a spectral viewpoint:

- The functions 1, $\cos(nx)$, $\sin(nx)$ are eigenfunctions of the one dimensional Laplacian $X \mapsto X''$ with periodic boundary conditions in x, with eigenvalues $\lambda_0 = 0$ and $\lambda_n = n^2$.
- For each spatial frequency n in the periodic direction, the dependence in the transverse direction y is determined by the simple ordinary differential equation $Y'' - \lambda_n Y = 0$ with boundary condition $Y(1) = 0$ and normalization $Y(0) = 1$. This gives the hyperbolic profiles

$$Y_0(y) = 1 - y,$$
$$Y_n(y) = \frac{\sinh(n(1-y))}{\sinh(n)}, \quad n \geqslant 1.$$

- The boundary data f at $y = 0$ is expanded in the eigenbasis $\{1, \cos(nx), \sin(nx)\}$ and each Fourier mode is propagated into the interior of the strip with its own y dependent factor $Y_n(y)$.

Analytically, the solution is an infinite sum of separated solutions. In a spectral method we will truncate this sum to finitely many modes in x,

$$u_N(x,y) = a_0(1-y) + \sum_{n=1}^{N}[a_n\cos(nx) + b_n\sin(nx)]\frac{\sinh(n(1-y))}{\sinh(n)},$$

and approximate the harmonic function inside the strip by this finite Fourier representation.

2.6 Numerical Illustration

To visualize the structure of harmonic functions in the strip, we compute the truncated Fourier series solution for a boundary condition that contains two modes:

$$f(x) = \sin(x) + \frac{1}{2}\sin(3x).$$

For this particular boundary data, the Fourier coefficients are simply $b_1 = 1$ and $b_3 = 1/2$, with all other coefficients zero. The solution can be written explicitly as

$$u(x,y) = \sin(x)\frac{\sinh(1-y)}{\sinh(1)} + \frac{1}{2}\sin(3x)\frac{\sinh(3(1-y))}{\sinh(3)}.$$

The key portion of the implementation evaluates the truncated series on a two-dimensional grid. In Python:

```python
def laplace_solution(x, y, a0, a_n, b_n):
    u = a0 * (1 - y)
    n_modes = len(a_n) - 1
    for n in range(1, n_modes + 1):
```

```
        if abs(a_n[n]) < 1e-15 and abs(b_n[n]) < 1e-15:
            continue
        y_factor = np.sinh(n * (1 - y)) / np.sinh(n)
        u += (a_n[n] * np.cos(n * x) + b_n[n] * np.sin(n * x)) * y_factor
    return u
```

The equivalent MATLAB implementation:

```
U = a0 * (1 - Y);
for n = 1:N_MODES
    if abs(a_n(n+1)) < 1e-15 && abs(b_n(n+1)) < 1e-15
        continue;
    end
    y_factor = sinh(n * (1 - Y)) / sinh(n);
    U = U + (a_n(n+1) * cos(n*X) + b_n(n+1) * sin(n*X)) .* y_factor;
end
```

Figure 5 shows the solution $u(x, y)$ in the strip $[0, 2\pi] \times [0, 1]$. At the bottom boundary $y = 0$, the solution matches the prescribed boundary data $f(x)$. As y increases toward the top boundary, the solution decays to zero. Crucially, the higher frequency mode ($n = 3$) decays much faster than the lower frequency mode ($n = 1$), as the hyperbolic factor $\sinh(n(1-y))/\sinh(n)$ decreases more rapidly for larger n. This illustrates the smoothing effect of harmonic extension into the interior.

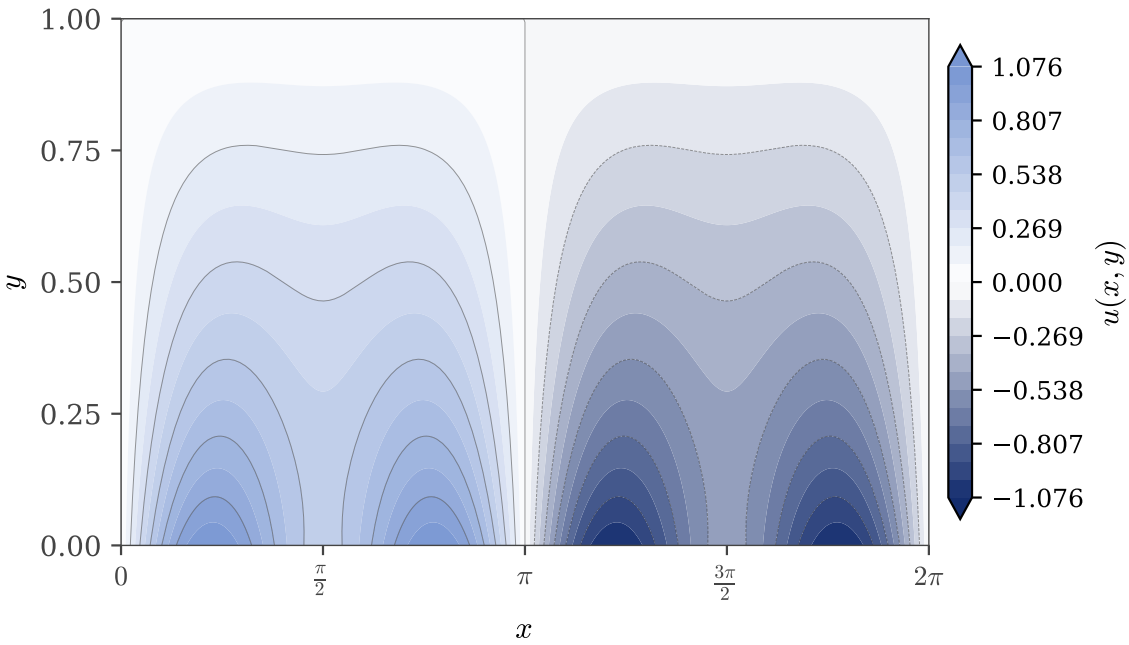

Figure 5: *Solution of the Laplace equation in the periodic strip with boundary data* $f(x) = \sin(x) + \frac{1}{2}\sin(3x)$ *at* $y = 0$ *and* $u = 0$ *at* $y = 1$. *Higher frequency modes decay faster toward the interior.*

The code that generated this figure is available in both Python and MATLAB:

- `codes/python/ch02_classical_pdes/laplace_equation_2d.py`
- `codes/matlab/ch02_classical_pdes/laplace_equation_2d.m`

2.7 Conclusions

The three examples presented in this chapter (the heat equation, the wave equation, and the Laplace equation) share a common mathematical structure that will guide us throughout the rest of this book. In each case, separation of variables reduces a partial differential equation to a family of ordinary differential equations: an eigenvalue problem in the spatial variable and a simpler equation governing the behavior in the remaining variable (time or the transverse coordinate). The eigenfunctions form an orthogonal basis, and the solution is expressed as an infinite series

$$u(x,t) = \sum_k \hat{u}_k(t)\varphi_k(x)$$

whose coefficients are determined by the initial or boundary data through Fourier projections. This is the DNA of spectral methods.

Let us distill the key ideas:

1. **Separation**. We decompose the solution into modes that evolve independently (or nearly so). Each mode satisfies a simpler equation than the original PDE.
2. **Basis**. The spatial part of each mode is an eigenfunction of a differential operator: Fourier exponentials for periodic problems, trigonometric functions for Dirichlet conditions, Chebyshev or Legendre polynomials for more general settings.
3. **Truncation**. In practice we cannot sum infinitely many terms due to the apparent finitude of our Universe. We retain only the first N modes, and the accuracy of this approximation depends critically on how fast the coefficients $\hat{u}_k$ decay.

When the solution is smooth, the coefficients decay *exponentially*, and a modest N suffices for high accuracy. This is the source of spectral methods' legendary efficiency. For a comprehensive treatment of these classical methods and their mathematical foundations, the reader is referred to [4].

The analytical solutions derived in this chapter are beautiful, but they are also fragile. They apply only to linear equations on simple domains with special boundary conditions. The moment we encounter a nonlinearity, a complicated geometry, or variable coefficients, we must turn to computation. The chapters that follow will develop the computational machinery needed to turn these analytical insights into practical algorithms:

- **FFT and its cousins**: how to move efficiently between physical space and coefficient space.
- **Differentiation matrices**: how to compute derivatives spectrally.
- **Quadrature rules**: how to compute inner products and projections.
- **Time stepping**: how to advance the ODE system for the coefficients.

Armed with these tools, we will be able to solve problems far beyond the reach of pen-and-paper analysis.

CHAPTER 3

Mise en Bouche

In French cuisine, a *mise en bouche* is a small appetizer offered by the chef to stimulate the palate before the main courses arrive. In this chapter we offer a similar intellectual appetizer: a compact, self-contained taste of spectral methods that illuminates the essential mechanics before we develop the full machinery of Fourier and Chebyshev approximation. Rather than jumping immediately to high-degree polynomials with $N = 100$, we perform hand calculations with just $N = 2$ or $N = 3$ unknowns. This low-dimensional setting makes every step transparent. We can verify each formula by direct computation and gain intuition that will guide us through the more sophisticated developments to come.

The techniques presented here follow the classical exposition in [3], adapted to our pedagogical goals. The Method of Weighted Residuals provides the unifying framework that connects the collocation (pseudospectral) approach we favor in this book with the Galerkin methods that dominate finite element analysis.

3.1 The Method of Weighted Residuals

3.1.1 Series Expansions and the Residual Function

The central idea of spectral methods is to approximate the unknown function $u(x)$ by a finite sum of basis functions:

$$u(x) \approx u_N(x) = \sum_{n=0}^{N} a_n \varphi_n(x),$$

where $\{\varphi_n(x)\}$ are known basis functions and $\{a_n\}$ are unknown coefficients to be determined.

When we substitute this approximation into a differential equation

$$\mathcal{L}u = f(x),$$

where $\mathcal{L}$ is a linear differential operator, the result is generally not zero. The *residual function* measures this discrepancy:

$$R(x; a_0, a_1, ..., a_N) = \mathcal{L}u_N - f.$$

For the exact solution, $R(x) \equiv 0$. The challenge is to choose the coefficients $\{a_n\}$ so that the residual is as small as possible. Different spectral methods correspond to different strategies for minimizing this residual.

3.1.2 Two Minimization Strategies

The two most important strategies are:

1. **Collocation (Pseudospectral) Method**: Force the residual to be exactly zero at a set of carefully chosen points $\{x_j\}$, called collocation points:
$$R(x_j; a_0, ..., a_N) = 0, \quad j = 1, 2, ..., N+1.$$
This gives $N + 1$ equations for $N + 1$ unknowns.
2. **Galerkin Method**: Require the residual to be orthogonal to each basis function in the sense of a weighted inner product:
$$\int_{-1}^{1} R(x)\varphi_k(x)w(x)\,\mathrm{d}x = 0, \quad k = 0, 1, ..., N,$$
where $w(x)$ is a weight function (often $w(x) = 1$ for polynomial bases).

Both methods convert the differential equation into a system of algebraic equations for the unknown coefficients. The collocation approach is simpler to implement and handles nonlinear terms easily, while the Galerkin approach often provides better global accuracy in weighted norms.

3.2 A First Collocation Example

We illustrate the collocation method with a complete worked example that can be verified by hand calculation.

3.2.1 Problem Statement

Consider the boundary value problem on $[-1, 1]$:

$$u''(x) - (4x^2 + 2)u(x) = 0, \quad -1 \leqslant x \leqslant 1,$$

with boundary conditions

$$u(-1) = 1, \quad u(1) = 1.$$

3.2.2 The Exact Solution

The exact solution is

$$u_{\text{exact}}(x) = e^{x^2-1}.$$

Let us verify this claim. The first derivative is

$$u'_{\text{exact}}(x) = 2xe^{x^2-1}.$$

The second derivative is

$$u''_{\text{exact}}(x) = (2 + 4x^2)e^{x^2-1} = (4x^2 + 2)u_{\text{exact}}(x).$$

Substituting into the ODE:

$$u''_{\text{exact}} - (4x^2 + 2)u_{\text{exact}} = (4x^2 + 2)u_{\text{exact}} - (4x^2 + 2)u_{\text{exact}} = 0.\checkmark$$

The boundary conditions are satisfied:

$$u_{\text{exact}}(\pm 1) = e^{1-1} = e^0 = 1.\checkmark$$

3.2.3 Trial Function

To satisfy the boundary conditions automatically, we write the approximation in a form that equals 1 at $x = \pm 1$ regardless of the coefficient values. A convenient choice is:

$$u_2(x) = 1 + (1 - x^2)(a_0 + a_1x + a_2x^2).$$

The factor $(1 - x^2)$ vanishes at the endpoints, so

$$u_2(\pm 1) = 1 + 0 \cdot (\cdots) = 1$$

for any values of a_0, a_1, a_2. We have three undetermined coefficients.

Expanding the trial function:

$$\begin{aligned} u_2(x) &= 1 + a_0 + a_1x + a_2x^2 - a_0x^2 - a_1x^3 - a_2x^4 \\ &= (1 + a_0) + a_1x + (a_2 - a_0)x^2 - a_1x^3 - a_2x^4. \end{aligned}$$

3.2.4 Computing the Residual

The residual is

$$R(x; a_0, a_1, a_2) = u_2''(x) - (4x^2 + 2)u_2(x).$$

Computing the second derivative of u_2:

$$u_2'(x) = a_1 + 2(a_2 - a_0)x - 3a_1x^2 - 4a_2x^3,$$
$$u_2''(x) = 2(a_2 - a_0) - 6a_1x - 12a_2x^2.$$

Substituting into the residual and simplifying (a calculation best verified with computer algebra), the residual is a polynomial of degree six in x with coefficients that depend linearly on a_0, a_1, a_2.

3.2.5 Collocation Conditions

We have three unknowns, so we choose three collocation points in the interior of the interval. A simple choice is

$$x_1 = -\frac{1}{2}, \quad x_2 = 0, \quad x_3 = \frac{1}{2}.$$

Setting the residual to zero at these points gives three linear equations:

At $x = -1/2$:

$$R\left(-\frac{1}{2}\right) = -\frac{17}{4}a_0 + \frac{33}{8}a_1 - \frac{25}{16}a_2 - 3 = 0.$$

At $x = 0$:

$$R(0) = -4a_0 + 2a_2 - 2 = 0.$$

At $x = 1/2$:

$$R\left(\frac{1}{2}\right) = -\frac{17}{4}a_0 - \frac{33}{8}a_1 - \frac{25}{16}a_2 - 3 = 0.$$

3.2.6 Solving the System

From the second equation:

$$-4a_0 + 2a_2 = 2 \quad \Rightarrow \quad a_2 = 1 + 2a_0.$$

Adding the first and third equations (the a_1 terms cancel):

$$-\frac{17}{2}a_0 - \frac{25}{8}a_2 = 6.$$

Substituting $a_2 = 1 + 2a_0$:

$$-\frac{17}{2}a_0 - \frac{25}{8}(1 + 2a_0) = 6$$
$$-\frac{17}{2}a_0 - \frac{25}{8} - \frac{25}{4}a_0 = 6$$
$$-\left(\frac{34}{4} + \frac{25}{4}\right)a_0 = 6 + \frac{25}{8}$$
$$-\frac{59}{4}a_0 = \frac{73}{8}$$
$$a_0 = -\frac{73}{118}.$$

Then

$$a_2 = 1 + 2 \cdot \left(-\frac{73}{118}\right) = 1 - \frac{146}{118} = -\frac{28}{118} = -\frac{14}{59}.$$

Subtracting the first equation from the third:

$$-\frac{33}{4}a_1 = 0 \quad \Rightarrow \quad a_1 = 0.$$

The vanishing of a_1 reflects the symmetry of the problem: both the differential equation and the boundary conditions are symmetric about $x = 0$, so the solution must be an even function. An odd coefficient like a_1 would break this symmetry.

3.2.7 The Approximate Solution

Substituting the coefficients back:

$$u_2(x) = 1 + (1 - x^2)\left(-\frac{73}{118} - \frac{14}{59}x^2\right).$$

After simplification, this becomes the even polynomial:

$$u_2(x) = \frac{14}{59}x^4 + \frac{45}{118}x^2 + \frac{45}{118}.$$

The boundary conditions are satisfied:

$$u_2(\pm 1) = \frac{14}{59} + \frac{45}{118} + \frac{45}{118} = \frac{28}{118} + \frac{90}{118} = \frac{118}{118} = 1.\checkmark$$

The implementation of this approximation is straightforward. In Python:

```python
def u_approx(x):
    """Evaluate the collocation approximation."""
    a0, a1, a2 = -73/118, 0, -14/59
    return 1 + (1 - x**2) * (a0 + a1*x + a2*x**2)
```

The equivalent MATLAB implementation:

```matlab
% Collocation coefficients
a0 = -73/118;  a1 = 0;  a2 = -14/59;

% Approximate solution (anonymous function)
u_approx = @(x) 1 + (1 - x.^2) .* (a0 + a1*x + a2*x.^2);
```

3.2.8 Error Analysis

The following table compares the exact and approximate solutions at several points:

x	$u_{\text{exact}}(x)$	$u_{\text{approx}}(x)$	Error
-1	1.00000	1.00000	0.00000
-0.5	0.47237	0.49153	-0.01916
0	0.36788	0.38136	-0.01348
0.5	0.47237	0.49153	-0.01916
1	1.00000	1.00000	0.00000

Table 1: *Comparison of exact and three-coefficient collocation approximation.*

The maximum error is approximately 2×10^{-2}, which is remarkably good for such a low-order approximation. Figure 6 shows the solutions graphically.

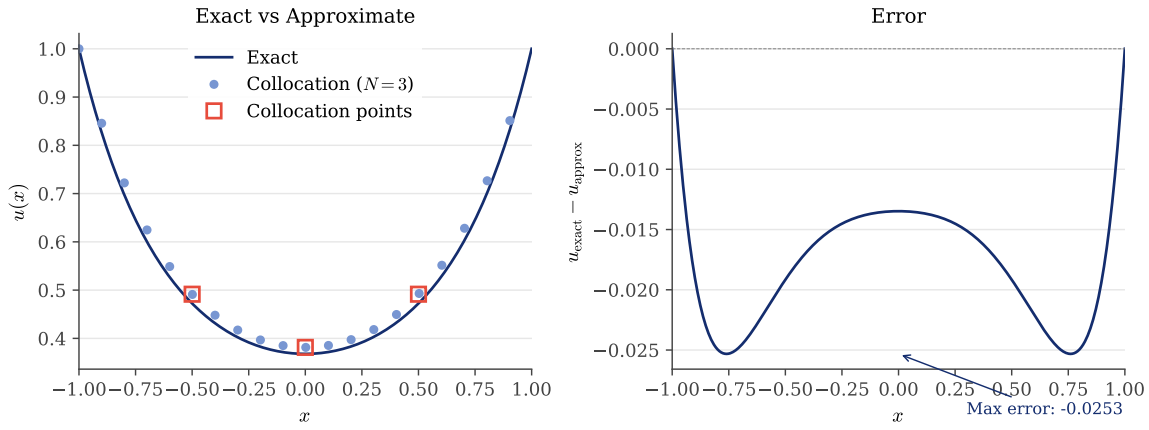

Figure 6: *Left: exact solution $u(x) = e^{x^2-1}$ compared with the three-coefficient collocation approximation. The collocation points $x = -1/2, 0, 1/2$ are marked with squares. Right: the error $u_{\text{exact}} - u_{\text{approx}}$.*

The code that generated this figure is available in both Python and MATLAB:

- `codes/python/ch03_mise_en_bouche/collocation_example1.py`
- `codes/matlab/ch03_mise_en_bouche/collocation_example1.m`

3.3 Collocation versus Galerkin

To compare the two main approaches to the Method of Weighted Residuals, we consider a second example where both methods can be applied with explicit hand calculations.

3.3.1 Problem Statement

Consider the reaction-diffusion equation on $[-1, 1]$:

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} - 4u = -1, \quad -1 \leqslant x \leqslant 1,$$

with homogeneous Dirichlet boundary conditions:

$$u(-1) = 0, \quad u(1) = 0.$$

3.3.2 The Exact Solution

The homogeneous equation $u'' - 4u = 0$ has the general solution $u_h = A\cosh(2x) + B\sinh(2x)$. A particular solution for the constant forcing -1 is $u_p = 1/4$. The general solution is therefore

$$u(x) = A\cosh(2x) + B\sinh(2x) + \frac{1}{4}.$$

The boundary condition $u(-1) = 0$ gives $A\cosh(2) - B\sinh(2) + 1/4 = 0$. The boundary condition $u(1) = 0$ gives $A\cosh(2) + B\sinh(2) + 1/4 = 0$.

Adding these equations: $2A\cosh(2) + 1/2 = 0$, so $A = -1/(4\cosh(2))$. Subtracting: $2B\sinh(2) = 0$, so $B = 0$.

The exact solution is:

$$u_{\text{exact}}(x) = \frac{1}{4}\left(1 - \frac{\cosh(2x)}{\cosh(2)}\right).$$

The maximum value occurs at $x = 0$:

$$u_{\text{exact}}(0) = \frac{1}{4}\left(1 - \frac{1}{\cosh(2)}\right) \approx 0.1834.$$

3.3.3 Trial Function and Basis

Since the boundary conditions are homogeneous, we choose basis functions that automatically vanish at $x = \pm 1$. For a symmetric problem like this one, we use even functions:

$$\varphi_0(x) = 1 - x^2, \quad \varphi_1(x) = (1 - x^2)x^2 = x^2 - x^4.$$

Both functions vanish at $x = \pm 1$ and are even in x. Our trial function is:

$$u_1(x) = a_0\varphi_0(x) + a_1\varphi_1(x) = a_0(1 - x^2) + a_1(x^2 - x^4).$$

3.3.4 The Residual

The operator is $\mathcal{L}u = u'' - 4u$. We compute:

$$\varphi_0'' = -2, \quad \varphi_1'' = 2 - 12x^2.$$

Applying the operator to each basis function:

$$\mathcal{L}\varphi_0 = -2 - 4(1 - x^2) = -6 + 4x^2,$$
$$\mathcal{L}\varphi_1 = (2 - 12x^2) - 4(x^2 - x^4) = 2 - 16x^2 + 4x^4.$$

The residual is:

$$R(x) = a_0\mathcal{L}\varphi_0 + a_1\mathcal{L}\varphi_1 - (-1)$$
$$= a_0(-6 + 4x^2) + a_1(2 - 16x^2 + 4x^4) + 1.$$

3.3.5 Collocation Method

With two unknowns, we need two collocation points. Due to symmetry, we can use points in $[0, 1)$. We choose $x_1 = 0$ and $x_2 = 0.5$.

At $x = 0$:

$$\mathcal{L}\varphi_0(0) = -6, \quad \mathcal{L}\varphi_1(0) = 2.$$
$$R(0) = -6a_0 + 2a_1 + 1 = 0 \quad \Rightarrow \quad 6a_0 - 2a_1 = 1.$$

At $x = 0.5$:

$$\mathcal{L}\varphi_0(0.5) = -6 + 4 \cdot 0.25 = -5,$$
$$\mathcal{L}\varphi_1(0.5) = 2 - 16 \cdot 0.25 + 4 \cdot 0.0625 = 2 - 4 + 0.25 = -1.75.$$
$$R(0.5) = -5a_0 - 1.75a_1 + 1 = 0 \quad \Rightarrow \quad 5a_0 + 1.75a_1 = 1.$$

Solving the system:

$$\begin{cases} 6a_0 - 2a_1 = 1 \\ 5a_0 + 1.75a_1 = 1 \end{cases}$$

Multiply the first equation by 5 and the second by 6:

$$30a_0 - 10a_1 = 5, \quad 30a_0 + 10.5a_1 = 6.$$

Subtracting: $20.5a_1 = 1$, so $a_1 \approx 0.04878$.

Substituting back: $6a_0 = 1 + 2 \cdot 0.04878 = 1.09756$, so $a_0 \approx 0.1829$.

The collocation solution is:

$$u_{\text{coll}}(x) \approx 0.1829(1 - x^2) + 0.0488(x^2 - x^4).$$

The following Python code assembles and solves the collocation system:

```python
def solve_collocation():
```

```
    """Solve the collocation system at x = 0 and x = 0.5."""
    # Operator L[φ] = φ'' - 4φ applied to basis functions
    L_phi0 = lambda x: -2 - 4*(1 - x**2)
    L_phi1 = lambda x: (2 - 12*x**2) - 4*(x**2 - x**4)

    # Build system matrix at collocation points
    A = np.array([[L_phi0(0.0), L_phi1(0.0)],
                  [L_phi0(0.5), L_phi1(0.5)]])
    b = np.array([1.0, 1.0])  # RHS from f = -1
    return np.linalg.solve(-A, b)
```

The equivalent MATLAB implementation:

```
% Operator L = d²/dx² - 4 applied to basis functions
L_phi0 = @(x) -2 - 4*(1 - x.^2);
L_phi1 = @(x) (2 - 12*x.^2) - 4*(x.^2 - x.^4);

% Build and solve collocation system
A_coll = [L_phi0(0.0), L_phi1(0.0);
          L_phi0(0.5), L_phi1(0.5)];
coeffs = A_coll \ [-1; -1];
```

3.3.6 Galerkin Method

The Galerkin conditions require the residual to be orthogonal to each basis function:

$$\int_{-1}^{1} R(x)\varphi_k(x)\,\mathrm{d}x = 0, \quad k = 0, 1.$$

This gives a symmetric matrix system $\boldsymbol{Aa} = \boldsymbol{b}$ where:

$$A_{ij} = \int_{-1}^{1} \mathcal{L}\varphi_j(x) \cdot \varphi_i(x)\,\mathrm{d}x,$$

$$b_i = \int_{-1}^{1} (-1) \cdot \varphi_i(x)\,\mathrm{d}x = -\int_{-1}^{1} \varphi_i(x)\,\mathrm{d}x.$$

Computing the integrals (using standard formulas for powers of x):

For $\varphi_0 = 1 - x^2$:

$$\int_{-1}^{1} \varphi_0(x)\,\mathrm{d}x = \int_{-1}^{1} (1 - x^2)\,\mathrm{d}x = \left[x - \frac{x^3}{3}\right]_{-1}^{1} = 2 - \frac{2}{3} = \frac{4}{3}.$$

For $\varphi_1 = x^2 - x^4$:

$$\int_{-1}^{1} \varphi_1(x)\,\mathrm{d}x = \left[\frac{x^3}{3} - \frac{x^5}{5}\right]_{-1}^{1} = \frac{2}{3} - \frac{2}{5} = \frac{4}{15}.$$

The right-hand side is:

$$b_0 = -\frac{4}{3}, \quad b_1 = -\frac{4}{15}.$$

The matrix entries require more computation. Using computer algebra or careful integration:

$$A_{00} = -\frac{104}{15}, \quad A_{01} = A_{10} = -\frac{8}{7}, \quad A_{11} = -\frac{328}{315}.$$

Solving the system yields:

$$a_0 \approx 0.1832, \quad a_1 \approx 0.0550.$$

The Galerkin method requires numerical integration to assemble the system. In Python:

```python
def solve_galerkin():
    """Solve using Galerkin: ⟨R, φₖ⟩ = 0 for k = 0, 1."""
    from scipy import integrate

    # Matrix entries: A_{ij} = ∫ L[φⱼ] φᵢ dx
    A00, _ = integrate.quad(lambda x: L_phi0(x) * phi0(x), -1, 1)
    A01, _ = integrate.quad(lambda x: L_phi1(x) * phi0(x), -1, 1)
    A10, _ = integrate.quad(lambda x: L_phi0(x) * phi1(x), -1, 1)
    A11, _ = integrate.quad(lambda x: L_phi1(x) * phi1(x), -1, 1)

    A = np.array([[A00, A01], [A10, A11]])
    b0, _ = integrate.quad(lambda x: -phi0(x), -1, 1)
    b1, _ = integrate.quad(lambda x: -phi1(x), -1, 1)
    return np.linalg.solve(A, [b0, b1])
```

The equivalent MATLAB implementation uses the built-in `integral` function:

```matlab
% Matrix entries: A_{ij} = ∫ L[φⱼ] φᵢ dx
A00 = integral(@(x) L_phi0(x) .* phi0(x), -1, 1);
A01 = integral(@(x) L_phi1(x) .* phi0(x), -1, 1);
A10 = integral(@(x) L_phi0(x) .* phi1(x), -1, 1);
A11 = integral(@(x) L_phi1(x) .* phi1(x), -1, 1);

A_gal = [A00, A01; A10, A11];

% RHS: b_i = ∫ f φᵢ dx where f = -1
b0 = integral(@(x) -phi0(x), -1, 1);
b1 = integral(@(x) -phi1(x), -1, 1);
coeffs = A_gal \ [b0; b1];
```

3.3.7 Comparison

The following table compares the two methods at the central point $x = 0$:

Method	$u(0)$	**Absolute Error**
Exact	0.1835	0
Collocation ($N = 2$)	0.1829	0.0006
Galerkin ($N = 2$)	0.1832	0.0003

Table 2: *Comparison of spectral approximations at the central maximum.*

For this problem, the Galerkin method is more accurate both at the central point and in a global sense. This is consistent with the error plot in Figure 7, which shows the Galerkin error (green) remaining closer to zero across the entire interval. The Galerkin method minimizes the

error in a root-mean-square sense, which typically leads to better overall accuracy for smooth problems.

Figure 7 shows both approximate solutions compared to the exact solution, along with the error profiles.

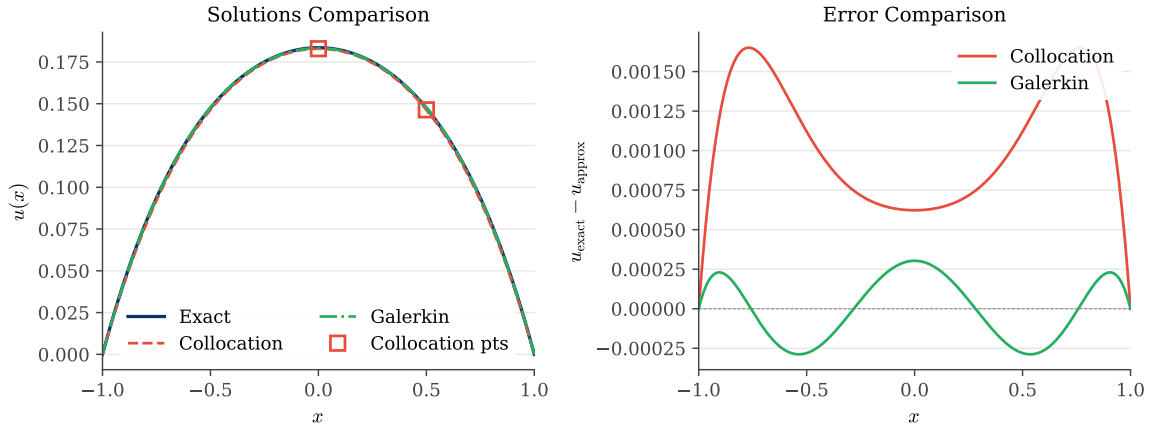

Figure 7: *Left: exact solution compared with collocation and Galerkin approximations. The collocation points $x = 0$ and $x = 0.5$ are marked. Right: error profiles for both methods.*

The code that generated this figure is available in both Python and MATLAB:

- `codes/python/ch03_mise_en_bouche/collocation_vs_galerkin.py`
- `codes/matlab/ch03_mise_en_bouche/collocation_vs_galerkin.m`

3.4 Conclusions and Questions

These simple examples illuminate several important features of spectral methods:

1. **Automatic satisfaction of boundary conditions**. By choosing trial functions that vanish at the boundaries (or equal the prescribed boundary values), we eliminate the boundary conditions from the algebraic system.
2. **Symmetry exploitation**. When the problem has symmetry, the solution inherits that symmetry. In our examples, the vanishing of odd coefficients ($a_1 = 0$) reflects the even symmetry of the exact solution.
3. **Simplicity of implementation**. Even with hand calculations, we can obtain remarkably accurate approximations with just a few coefficients.
4. **Trade-offs between methods**. Collocation is simpler to implement (no integrals to compute), while Galerkin typically provides better accuracy by minimizing a global error measure.

These examples raise important questions that we will address in subsequent chapters:

- **What is the optimal choice of basis functions?** Using simple powers of x works for small N, but becomes numerically unstable for large N. Chebyshev and Fourier bases are far superior.

- **What are the optimal collocation points?** Our ad hoc choices $x = -1/2, 0, 1/2$ worked well, but there exist optimal point distributions (Gauss and Gauss–Lobatto points) derived from orthogonal polynomial theory.
- **How fast does the error decrease as N increases?** For smooth solutions, spectral methods achieve exponential convergence (the error decreases like c^{-N} for some $c > 1$), which is dramatically faster than the algebraic convergence $O(N^{-p})$ of finite difference and finite element methods.

The following chapters will develop the theory and algorithms needed to answer these questions and to apply spectral methods to a wide range of problems.

3.5 A Broader Perspective

For demanding students who wish to understand how spectral methods fit into the wider landscape of numerical analysis, this section provides a high-level comparison with other approaches, particularly finite element methods. The discussion follows [3, Section 1.3].

3.5.1 Local versus Global Basis Functions

The fundamental distinction between spectral methods and finite element methods lies in the *support* of the basis functions. In finite element methods, the computational domain is divided into many small sub-intervals (or triangles, tetrahedra in higher dimensions), and the basis functions $\varphi_n(x)$ are *local*: they are polynomials of fixed, low degree (typically linear or quadratic) that are non-zero only over one or two adjacent elements.

In contrast, spectral methods use *global* basis functions. Each $\varphi_n(x)$ is a polynomial (or trigonometric polynomial) of potentially high degree that is non-zero (except at isolated points) over the entire computational domain. This global character is both the source of spectral methods' power and the reason for some of their limitations.

Figure 8 illustrates this distinction schematically. The finite element basis function (left) has compact support and contributes to the solution only locally. The spectral basis function (right) influences the solution everywhere.

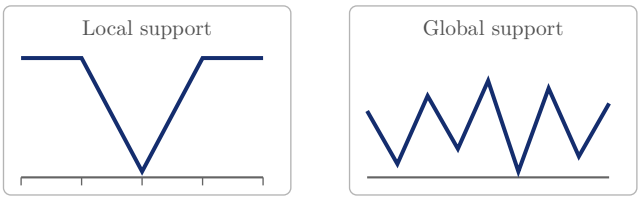

Figure 8: *Schematic comparison of local (finite element) and global (spectral) basis functions. The finite element "hat function" is non-zero only over two adjacent elements, while the spectral basis function oscillates across the entire domain.*

3.5.2 Refinement Strategies

When a numerical approximation is insufficiently accurate, there are three fundamentally different strategies to improve it, illustrated schematically in Figure 9:

1. ***h*-refinement**: Subdivide each element into smaller pieces, reducing the mesh spacing h uniformly throughout the domain. This increases the number of elements while keeping the polynomial degree fixed.
2. ***r*-refinement** (adaptive): Redistribute the mesh points, clustering them in regions where the solution has steep gradients or other features requiring high resolution. The total number of degrees of freedom remains roughly constant.
3. ***p*-refinement**: Keep the mesh fixed while increasing p, the polynomial degree within each element. For a single-element domain, this is precisely what spectral methods do.

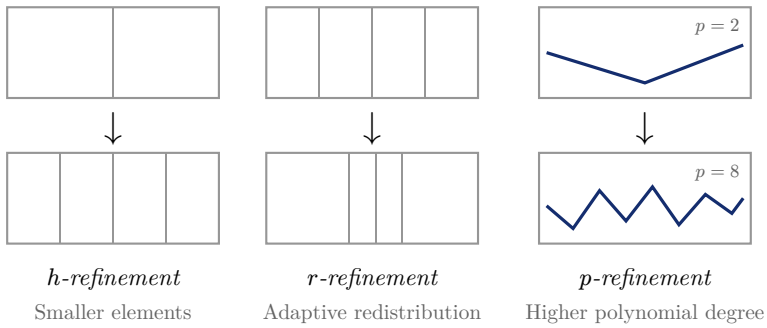

Figure 9: *Three strategies for improving accuracy in numerical methods. Spectral methods employ p-refinement: increasing the polynomial degree while using a single element (or few elements) spanning the entire domain.*

Spectral methods can be viewed as the extreme form of p-refinement: a single element spans the entire domain, and accuracy is improved solely by increasing the polynomial degree. This strategy is devastatingly effective when the solution is smooth, but struggles when the solution has discontinuities or sharp gradients.

3.5.3 Trade-offs: Sparse versus Full Matrices

The choice between local and global basis functions entails fundamental trade-offs:

Finite element advantages:

- *Sparse matrices*: Since each basis function is non-zero over only a few elements, the stiffness matrix has mostly zero entries. Sparse matrix solvers can exploit this structure, dramatically reducing computational cost for large systems.
- *Geometric flexibility*: The small elements (triangles, tetrahedra) can be fitted to irregularly shaped domains like automobile bodies or aircraft wings.

Finite element disadvantages:

- *Low accuracy per degree of freedom*: Each basis function is a polynomial of low degree, so many elements are needed for high accuracy.

Spectral method advantages:

- *High accuracy for smooth problems*: The high-degree global polynomials capture smooth solutions with far fewer degrees of freedom.

- *Efficiency with iterative solvers*: When fast iterative methods are used, spectral methods can be much more efficient than low-order methods for many problem classes.

Spectral method disadvantages:

- *Full matrices*: The global basis functions create dense matrices where most entries are non-zero.
- *Geometric limitations*: Spectral methods are most natural on simple domains (intervals, rectangles, disks) and require more sophisticated techniques for irregular geometries.

For problems with smooth solutions on regular domains (many important problems in fluid dynamics, quantum mechanics, and wave propagation fall into this category), the accuracy advantage of spectral methods often outweighs the matrix structure disadvantage.

3.5.4 Spectral Element Methods

A natural question arises: can we combine the geometric flexibility of finite elements with the high accuracy of spectral methods? The answer is yes, through *spectral element methods*.

In spectral element methods, the domain is subdivided into elements (as in finite elements), but within each element, the polynomial degree p is chosen to be moderately high, typically $p = 6$ to 8. This hybrid approach inherits several advantages:

- The element subdivision provides geometric flexibility and matrix sparsity.
- The high polynomial degree within each element provides spectral-like accuracy.
- The theoretical framework is essentially the same as for global spectral methods.

Spectral element codes are typically written so that p is a user-adjustable parameter, allowing practitioners to balance accuracy and cost for their specific application. We will not develop spectral element methods in detail in this book, but the reader should be aware that much of the theory developed for global spectral methods transfers directly to the spectral element context.

3.5.5 The Convergence of Methods at High Order

Perhaps the most profound insight from the comparison between finite element and spectral methods is this: *for sufficiently high polynomial degree, the two approaches become essentially equivalent.*

Low-order finite elements (linear, quadratic) can be derived, justified, and implemented without knowledge of Fourier or Chebyshev convergence theory. However, as the polynomial degree increases, ad hoc approaches become increasingly ill-conditioned and numerically unstable. The only practical way to implement well-behaved high-order finite elements (say, sixth order or higher) is to use the technology of spectral methods: Chebyshev or Legendre basis functions, Gaussian quadrature, and the convergence theory we will develop in subsequent chapters.

Thus, the question "Are finite elements or spectral methods better?" becomes somewhat artificial for high-order approximations. The real question is: *Does the problem at hand require high-order accuracy, or is second or fourth order sufficient?*

When the solution is smooth and high accuracy is needed, the spectral/high-order approach is clearly superior. When the solution has discontinuities, shocks, or boundary layers, or when the geometry is highly irregular, low-order methods with adaptive mesh refinement may be more practical. The wise practitioner chooses the tool appropriate to the problem.

Bibliography

[1] B. Fornberg, *A practical guide to pseudospectral methods*. Cambridge University Press, 1996, p. 242.
[2] L. N. Trefethen, *Spectral methods in MatLab*. Society for Industrial, Applied Mathematics, Philadelphia, PA, USA, 2000, p. 184. [Online]. Available: http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/spectral.html
[3] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2nd ed. Dover Publications, New York, 2000, p. 688.
[4] A. N. Tikhonov and A. A. Samarskii, *Equations of Mathematical Physics*. Dover Publications, Inc., 1963, p. 785.