

Intro to PDEs: representations and basic numerical methods

Andrej Risteski

August 31, 2026

Contents

1	What is a partial differential equation?	2
1.1	Poisson equation	3
1.2	Heat equation	4
1.3	Advection	6
1.4	A simple running example	6
2	Strong, weak, and energy forms	7
2.1	Strong form	7
2.2	Weak form	7
2.3	Variational (energy) form	8
3	The basic numerical solvers	10
3.1	Finite differences (on a grid)	10
3.2	Galerkin finite elements (on an unstructured mesh)	15
3.3	Spectral representations: finitize in a global basis	17
3.3.1	Fourier analysis for periodic functions	18
3.3.2	Fourier analysis of $\mathbb{R}$	20
3.3.3	Some remarks	20
4	Time discretization: from continuous dynamics to an iteration	21
4.1	Explicit Euler and the local one-step error	21
4.2	Error propagation after multiple steps	22
4.3	Absolute stability and implicit Euler	24
4.4	Stiffness and wastefulness of explicit methods	26
5	How space and time discretization interact	27
5.1	Diffusion: spatial refinement creates fast decaying modes	27
5.2	Difficulties in general PDEs	28

1 What is a partial differential equation?

A scalar field is a function

$$u : \Omega \times [0, T] \rightarrow \mathbb{R},$$

where $x \in \Omega \subset \mathbb{R}^d$ denotes spatial position and t denotes time. A *PDE* (partial differential equation) relates the field to its spatial and temporal derivatives. In some generality, a PDE of order at most m can be written as:

$$F(x, t, u, \nabla u, \partial_t u, \nabla^2 u, \partial_t \nabla u, \dots, D^m u) = 0,$$

where F is a prescribed function and $D^m u$ denotes the collection of derivatives of u up to order m . Depending on the problem, some of these arguments may of course be absent. For example, *steady-state equations* will not have a time ingredient as they model the steady state of some process.

To make the solution unique—and also because it often is physically natural, we also sometimes need to specify what happens on the boundary $\partial\Omega$ by imposing *boundary conditions*. Common examples are

$$u = g \quad \text{on } \partial\Omega \quad (\text{Dirichlet boundary condition}),$$

which specifies the value of the field at the boundary, and

$$\frac{\partial u}{\partial n} := \nabla u \cdot n = g \quad \text{on } \partial\Omega \quad (\text{Neumann boundary condition}),$$

which specifies its normal derivative (or, physically, a flux) through the boundary. Here n denotes the outward-pointing unit normal to $\partial\Omega$.

For a time-dependent PDE, we typically also prescribe an *initial condition*, for example

$$u(x, 0) = u_0(x), \quad x \in \Omega.$$

Notational reminder: To remind ourselves of some basic definitions, for a scalar field $u = u(x)$, the

gradient is the vector of first spatial derivatives, $\nabla u = \begin{pmatrix} \partial_{x_1} u \\ \vdots \\ \partial_{x_d} u \end{pmatrix}$, while for a vector field $v = (v_1, \dots, v_d)$,

the divergence is

$$\nabla \cdot v = \sum_{i=1}^d \partial_{x_i} v_i.$$

The Hessian of u is the matrix of second spatial derivatives,

$$\nabla^2 u = \left(\frac{\partial^2 u}{\partial x_i \partial x_j} \right)_{i,j=1}^d.$$

The Laplacian is the trace of the Hessian, or equivalently the divergence of the gradient:

$$\Delta u := \text{tr}(\nabla^2 u) = \nabla \cdot \nabla u = \sum_{i=1}^d \frac{\partial^2 u}{\partial x_i^2}.$$

Some common examples of PDEs to start us off we'll consider are:

$$\begin{aligned} \text{Poisson equation:} & \quad -\Delta u = f, \\ \text{Variable-coefficient Poisson equation:} & \quad -\nabla \cdot (\kappa(x) \nabla u) = f, \end{aligned}$$

$$\begin{aligned}\text{Heat equation:} & \quad u_t = \nu \Delta u, \\ \text{Advection equation:} & \quad u_t + b \cdot \nabla u = 0.\end{aligned}$$

These equations encode some basic physical mechanisms, and we do a “first physical principles” derivation below.

1.1 Poisson equation

Poisson models a source or load f which creates a static equilibrium field. For example, a distributed force that bends a string, or a steady-state temperature profile from applying heat. We can in fact derive this equation “from first physical principles”. For simplicity, let’s focus on the 1-D case.

Consider a taut string whose equilibrium configuration is the graph

$$y = u(x), \quad x \in (0, L).$$

We take the vertical coordinate to be positive upward. Suppose an external transverse force of density $f(x)$ is applied per unit *horizontal* length. We view $f(x)$ as a signed force density, with $f > 0$ meaning upward force. At a point x , let $\theta(x)$ denote the angle made by the tangent to the string with the horizontal. Since the string is the graph $y = u(x)$, the tangent at point x is $(1, u'(x))$. Hence,

$$\tan \theta(x) = u'(x).$$

The tension force acts along the tangent to the string. Write its horizontal and vertical components as $(H(x), V(x))$. Because the tension vector has the same direction as the tangent, we have $\frac{V(x)}{H(x)} = \tan \theta(x) = u'(x)$. Hence

$$V(x) = H(x)u'(x).$$

If the string is in equilibrium, both the horizontal and vertical forces have to balance. Consider a portion of the string lying above the horizontal interval $[x, x + \Delta x]$.

The tension at the left endpoint contributes horizontal force $-H(x)$, while the tension at the right endpoint contributes $+H(x + \Delta x)$. Because the external load is purely vertical, there is no external horizontal force on this piece. Horizontal force balance therefore gives

$$-H(x) + H(x + \Delta x) = 0.$$

Since this holds for every interval, the horizontal component of the tension is constant:

$$H(x) \equiv H.$$

Now let’s consider the vertical balance. At the left endpoint, the tension exerted on this piece points outward to the left, so its vertical component is $-Hu'(x)$. At the right endpoint, the tension points outward to the right, so its vertical component is $Hu'(x + \Delta x)$. Thus the net vertical force due to tension is

$$H(u'(x + \Delta x) - u'(x)).$$

The external distributed force on this interval is

$$\int_x^{x+\Delta x} f(s) ds \approx f(x)\Delta x.$$

Static equilibrium requires the total vertical force to vanish, implying:

$$H(u'(x + \Delta x) - u'(x)) + f(x)\Delta x \approx 0.$$

Dividing by Δx and sending $\Delta x \rightarrow 0$ gives $Hu''(x) + f(x) = 0$, or equivalently

$$-Hu''(x) = f(x).$$

Thus, after rescaling the load by the constant horizontal tension H , this is of course the one-dimensional Poisson equation $-u''(x) = \frac{f(x)}{H}$. For a string fixed at its endpoints, one supplements this with the boundary condition $u(0) = u(L) = 0$.

For a **variable-coefficient Poisson equation**, the positive coefficient $\kappa(x)$ describes how stiff the material locally is (or how easily it conducts or diffuses in the steady-state heat setting). The solution develops steeper gradients in regions where transport is harder.

1.2 Heat equation

The heat equation models how heat diffuses—moving the field from high values toward nearby low values, so peaks spread out and flatten over time.

Let's also do a derivation from “first physical principles”. Again, let's do the 1-D case for simplicity. Consider a thin rod, and let $u(x, t)$ denote its temperature at position x and time t .

Let $q(x, t)$ denote the heat flux through the rod at position x : the amount of heat crossing a unit cross-sectional area per unit time, with positive flux corresponding to flow in the positive x -direction. Fourier's law states that

$$q(x, t) = -\kappa u_x(x, t),$$

where $\kappa > 0$ is the thermal conductivity. The minus sign encodes the fact that heat flows from hot regions toward cold regions.

Rather than balancing forces, now we'll balance heat. Consider a small piece of the rod, $[x, x + \Delta x]$. Heat enters through its left endpoint at rate $q(x, t)$, and leaves through its right endpoint at rate $q(x + \Delta x, t)$. Hence the net rate at which heat enters the interval is: $q(x, t) - q(x + \Delta x, t)$. For small Δx ,

$$q(x + \Delta x, t) - q(x, t) \approx q_x(x, t)\Delta x,$$

and therefore

$$\text{net heat influx} \approx -q_x(x, t)\Delta x = \frac{\partial}{\partial x}(\kappa u_x(x, t))\Delta x.$$

We must now relate this heat influx to the resulting change in temperature. Let ρ denote the mass density of the material and let c denote its specific heat capacity. Then

$$\rho c u_t(x, t)\Delta x$$

is the rate of increase of thermal energy in the small interval. Conservation of energy therefore gives

$$\rho c u_t(x, t)\Delta x = \frac{\partial}{\partial x}(\kappa u_x(x, t))\Delta x.$$

Cancelling Δx , and setting $\nu := \frac{\kappa}{\rho c}$ as the thermal diffusivity, we obtain the heat equation

$$u_t = \nu u_{xx}.$$

If heat is generated internally at a rate $f(x)$ per unit volume, the equation becomes

$$\rho c u_t = \nabla \cdot (\kappa \nabla u) + f.$$

At steady state, $u_t = 0$, and hence

$$-\nabla \cdot (\kappa \nabla u) = f.$$

Thus the variable-coefficient Poisson equation can be interpreted as the steady-state heat equation in a heterogeneous material.

There is another way to derive the heat equation as a “limit” of a discrete *random walk*. Consider a symmetric random walk on the one-dimensional lattice

$$h\mathbb{Z} = \{\dots, -2h, -h, 0, h, 2h, \dots\},$$

where the walker moves a distance h to the left or right with equal probability at each step. If you like, you can think of the walker as representing a randomly chosen molecule undergoing microscopic random motion: it does left or right with equal probability.

Suppose that one step of the walk corresponds to a time increment Δt , and let $x_i = ih$, $t_n = n\Delta t$. Let p_i^n denote the probability that the walker is at x_i after n steps. Since the walker can arrive at x_i only from one of its two neighbors,

$$p_i^{n+1} = \frac{1}{2}p_{i-1}^n + \frac{1}{2}p_{i+1}^n.$$

Subtracting p_i^n from both sides gives

$$p_i^{n+1} - p_i^n = \frac{1}{2} (p_{i-1}^n - 2p_i^n + p_{i+1}^n).$$

The quantity in parentheses is the discrete analogue of a second derivative. Indeed, if we define the “discrete Laplacian” by¹

$$(\Delta_h u)(x_i) := \frac{u(x_i + h) - 2u(x_i) + u(x_i - h)}{h^2},$$

then the random-walk evolution can be written as

$$\frac{p_i^{n+1} - p_i^n}{\Delta t} = \frac{h^2}{2\Delta t} \Delta_h p_i^n.$$

Strictly speaking, p_i^n is a probability mass rather than a probability density. In the continuum limit, if $p(t, x)$ denotes the corresponding density, then $p_i^n \approx h, p(t_n, x_i)$. Since this common factor of h cancels from the evolution equation, we suppress this distinction in what follows.

To obtain a nontrivial continuum limit, we let $h, \Delta t \rightarrow 0$ while keeping

$$\frac{h^2}{2\Delta t} \longrightarrow \nu$$

for some fixed $\nu > 0$. This yields:

$$\frac{p_i^{n+1} - p_i^n}{\Delta t} \longrightarrow \partial_t p(t, x), \quad \Delta_h p_i^n \longrightarrow \partial_{xx} p(t, x).$$

Thus the limiting evolution is

$$\partial_t p = \nu \partial_{xx} p$$

which is precisely the one-dimensional heat equation.

The scaling $\Delta t \asymp h^2$ is called the *diffusive scaling*. Its origin can already be seen from a variance calculation. A single step has mean zero and variance h^2 , so after n independent steps, $\text{Var}(X_n) = nh^2$.

¹See Lemma 6 for a more formal derivation.

Since $n = t/\Delta t$, we have $\text{Var}(X_t) = t \frac{h^2}{\Delta t}$. Therefore a *finite, but nonzero amount of spreading over a fixed macroscopic time interval* requires $\Delta t \asymp h^2$.

If you know what Brownian motion is: the choice $\frac{h^2}{2\Delta t} \rightarrow \nu$, we have $\text{Var}(X_t) \rightarrow 2\nu t$, which is exactly the variance of Brownian motion with diffusion coefficient ν . Thus, under diffusive scaling, the random walk converges to Brownian motion, while the evolution of its probability distribution converges to the heat equation.

1.3 Advection

The advection equation models the transport of a field by a prescribed velocity. For a constant velocity $b \in \mathbb{R}^d$, it takes the form $u_t + b \cdot \nabla u = 0$. On the whole space $\mathbb{R}^d$, or on a periodic domain, the solution with initial condition $u(x, 0) = u_0(x)$ is

$$u(x, t) = u_0(x - bt).$$

Thus the initial profile is translated at velocity b without changing shape.

On a bounded domain, boundary data are prescribed only where the flow “enters” the domain. The *inflow boundary* is

$$\Gamma_- := \{x \in \partial\Omega : b \cdot n(x) < 0\}.$$

One typically specifies $u = g$ on Γ_- , while the solution on the outflow boundary is determined by transport from the interior. This is very easy to see in a 1-d example: consider

$$u_t + bu_x = 0, \quad x \in [0, 1], \quad b > 0.$$

Consider the curve $x(t) = x_0 + bt$: by chain rule, we have

$$\frac{d}{dt}u(x(t), t) = u_t + bu_x = 0.$$

Thus the solution is transported to the right without changing value: $u(x, t) = F(x - bt)$. Since $b > 0$, information enters the interval through $x = 0$ and exits through $x = 1$. Indeed, the outward normals are

$$n(0) = -1, \quad n(1) = +1,$$

so

$$bn(0) = -b < 0, \quad bn(1) = b > 0.$$

Hence $x = 0$ is the inflow boundary and $x = 1$ is the outflow boundary. We therefore prescribe $u(0, t) = g(t)$, while $u(1, t)$ is determined by transporting values from the interior. You can see how the appropriate boundary conditions will depend on the type and order of the PDE.

1.4 A simple running example

A running example in a lot of the subsequent sections, for simplicity, will just be the 1-D Poisson equation with zero boundary condition.

Example 1 (A one-dimensional Poisson equation). On $\Omega = (0, 1)$, consider

$$-u''(x) = f(x), \quad u(0) = u(1) = 0. \tag{1}$$

We want to solve these equations, which means we need to represent them on a computer somehow. Several related decisions have to be made:

- u is a function—so what “finitized” representation will we use to encode it?
- In what sense will the PDE be “enforced”?
- How will the resulting system of equations be solved?
- In what sense is the solution an approximation of the intended one?

2 Strong, weak, and energy forms

We first need to understand what it means for u to solve the equation. There are several useful ways to formalize this idea.

2.1 Strong form

The strong form asks that (1) is satisfied in a pointwise fashion. Looking ahead, some numerical solvers (collocation methods, finite differences) naturally fit this formulation: they select points x_i and fit some loss encouraging

$$-u_h''(x_i) \approx f(x_i).$$

Unsurprisingly, some machine learning-based methods—in particular, standard physics-informed neural networks (PINNs)—train on a loss of this form [5].

2.2 Weak form

The weak form asks to satisfy (1) “on average”, i.e. when integrated against suitable “test functions”. Let v be a smooth test function with $v(0) = v(1) = 0$. Multiply (1) by v and integrate:

$$-\int_0^1 u'' v \, dx = \int_0^1 f v \, dx.$$

Why would we do this? Integration by parts rewrites the above equation as:

$$\int_0^1 u' v' \, dx = \int_0^1 f v \, dx. \quad (2)$$

since

$$-\int_0^1 u'' v \, dx = -[u' v]_0^1 + \int_0^1 u' v' \, dx.$$

From a computational point of view, the second derivative of u has disappeared, so the formulation still makes sense for functions whose first derivative is square-integrable even when a classical second derivative does not exist everywhere. For example, take

$$f(x) = \begin{cases} 1, & 0 < x < \frac{1}{2}, \\ 0, & \frac{1}{2} < x < 1, \end{cases}$$

where the value at $x = 1/2$ is irrelevant. The zero-boundary-value solution is

$$u(x) = \begin{cases} -\frac{1}{2}x^2 + \frac{3}{8}x, & 0 \leq x \leq \frac{1}{2}, \\ \frac{1}{8}(1-x), & \frac{1}{2} \leq x \leq 1. \end{cases}$$

This function and its first derivative are continuous, but its second derivative jumps at $x = 1/2$. Thus it is not a classical C^2 solution on the whole interval, while it does satisfy the weak identity (2). This is a simple example in which the weak formulation naturally accommodates limited regularity.

Numerical methods such as the Galerkin finite element method use this formulation: they choose finite-dimensional test spaces and enforce the identity for every selected test function in their span. Machine-learning variants such as variational PINNs fit the same pattern [2, 3].

As one might expect, if a strong solution exists, it is also a weak solution:

Proposition 2 (A strong solution is a weak solution). *Suppose $u \in C^2([0, 1])$ satisfies (1). Then (2) holds for every $v \in C^1([0, 1])$ with $v(0) = v(1) = 0$.*

Proof. Multiply $-u'' = f$ by v and integrate. Integration by parts yields

$$-\int_0^1 u''v \, dx = -[u'v]_0^1 + \int_0^1 u'v' \, dx.$$

The boundary term vanishes because $v(0) = v(1) = 0$. The left-hand side is $\int_0^1 f v \, dx$, proving the identity. $\square$

2.3 Variational (energy) form

The weak equation becomes especially transparent when written using the bilinear form

$$\mathbf{a}(w, v) := \int_0^1 w'(x)v'(x) \, dx, \quad \ell(v) := \int_0^1 f(x)v(x) \, dx. \quad (3)$$

With this notation, the weak problem is simply

$$\text{find } u \in V \text{ such that } \mathbf{a}(u, v) = \ell(v) \quad \text{for every } v \in V,$$

where V is a linear space of admissible functions that vanish at both endpoints. We can define a corresponding energy functional (sometimes called Ritz energy) as follows:

$$\mathcal{E}(v) = \frac{1}{2}\mathbf{a}(v, v) - \ell(v) = \frac{1}{2} \int_0^1 |v'(x)|^2 \, dx - \int_0^1 f(x)v(x) \, dx \quad (4)$$

for functions with zero endpoint values.

We can now view $w \in V$ as directions to perturb in, and write down the first variation

$$\left. \frac{d}{d\varepsilon} \mathcal{E}(v + \varepsilon w) \right|_{\varepsilon=0} = \mathbf{a}(v, w) - \ell(w).$$

Consequently, the weak formulation asks for stationarity of the energy in every admissible direction. This suggests that solving the weak problem should be equivalent to minimizing the energy functional $\mathcal{E}$ over V . This is sometimes called a *variational* formulation. The following theorem makes this equivalence precise:

Theorem 3 (Equivalence of the weak and variational formulations). *Let V be a linear space of continuously differentiable functions satisfying $v(0) = v(1) = 0$. A function $u \in V$ minimizes $\mathcal{E}$ over V if and only if*

$$\int_0^1 u'v' \, dx = \int_0^1 f v \, dx \quad \text{for every } v \in V. \quad (5)$$

Moreover, such a minimizer is unique whenever it exists.

Proof. Assume first that u minimizes $\mathcal{E}$. For any $v \in V$, define $g(\varepsilon) = \mathcal{E}(u + \varepsilon v)$. Since g has a minimum at 0,

$$0 = g'(0) = \int_0^1 u'v' \, dx - \int_0^1 f v \, dx,$$

which is (5).

Conversely, suppose (5) holds. For any $w \in V$, write $w = u + v$. Expanding (4) gives

$$\begin{aligned} \mathcal{E}(u + v) - \mathcal{E}(u) &= \int_0^1 u'v' \, dx - \int_0^1 f v \, dx + \frac{1}{2} \int_0^1 |v'|^2 \, dx \\ &= \frac{1}{2} \int_0^1 |v'|^2 \, dx \geq 0. \end{aligned}$$

Thus u is a minimizer. If both u_1 and u_2 minimize, apply the identity with $v = u_2 - u_1$ to obtain $\int_0^1 |(u_2 - u_1)'|^2 = 0$. Hence $u_2 - u_1$ is constant; its zero endpoint values force that constant to be zero. $\square$

The preceding theorem is not a “coincidence”: it can be understood perhaps more conceptually through strong convexity. In fact, the energy $\mathcal{E}$ is strongly convex:

Proposition 4 (Strong convexity of the Ritz energy). *Define the energy norm on V by*

$$\|v\|_V^2 := a(v, v) = \int_0^1 |v'(x)|^2 \, dx.$$

Then for every $v, w \in V$,

$$\mathcal{E}(w) = \mathcal{E}(v) + D\mathcal{E}(v)[w - v] + \frac{1}{2}\|w - v\|_V^2.$$

In particular, $\mathcal{E}$ is 1-strongly convex with respect to $\|\cdot\|_V$.

Proof. Let $h = w - v$. Since

$$\mathcal{E}(z) = \frac{1}{2}a(z, z) - \ell(z),$$

bilinearity and symmetry of a give

$$\begin{aligned} \mathcal{E}(v + h) &= \frac{1}{2}a(v + h, v + h) - \ell(v + h) \\ &= \mathcal{E}(v) + a(v, h) - \ell(h) + \frac{1}{2}a(h, h). \end{aligned}$$

But

$$D\mathcal{E}(v)[h] = a(v, h) - \ell(h),$$

and $a(h, h) = \|h\|_V^2$, which proves the identity. $\square$

Thus, “under the hood,” Theorem 3 is simply the infinite-dimensional analogue of a familiar fact from optimization: for a strongly convex objective, a stationary point is necessarily the unique global minimizer. The weak equation

$$D\mathcal{E}(u)[v] = 0 \quad \text{for every } v \in V$$

is precisely the statement that u is stationary in every admissible direction.

From a machine learning point-of-view this formulation is also useful: it gives us a loss we could try to optimize if we’re trying to find a solution to the PDE. Restricting the minimization to a finite-dimensional trial space gives the classical Ritz method; parametrizing the trial function by a neural network gives the Deep Ritz method [1]. Even more useful: “approximately optimizing” this objective guarantees that we’re “approximately close” to the solution of the PDE:

Corollary 5 (Energy gap equals squared error in the energy norm). *Let u satisfy the weak equation. Then for every $w \in V$,*

$$\mathcal{E}(w) - \mathcal{E}(u) = \frac{1}{2} \|w - u\|_V^2 = \frac{1}{2} \int_0^1 |w'(x) - u'(x)|^2 dx.$$

Proof. Apply the strong-convexity identity with $v = u$. Since u satisfies the weak equation,

$$D\mathcal{E}(u)[w - u] = 0,$$

and the claim follows. $\square$

Some remarks. The strong convexity statement above concerns the energy as a *functional of the function* w . If we parametrize $w = w_\theta$ by a neural network, the map $\theta \mapsto \mathcal{E}(w_\theta)$ is generally nonconvex. Thus strong convexity in function space does not imply that optimizing over neural-network parameters is easy. It does imply that if one succeeds in finding a network with small *global objective gap*, then its represented function is close to the PDE solution in the energy norm.

Also, note not every PDE admits a Ritz energy of this form. The construction relies on the symmetry of the underlying bilinear form. Nonsymmetric operators, such as advection, generally possess weak formulations but not a static scalar energy whose first variation is the original PDE.

3 The basic numerical solvers

We next provide an overview of the simplest, most basic numerical solvers—for the time being thinking of PDEs that don’t have a time component.

3.1 Finite differences (on a grid)

We show first a simple way we could approximate the solution to (1) to an arbitrary accuracy — at the expense of computational cost. The idea is quite simple: we will approximate all “continuous” quantities by a natural “discrete” approximation.

The first thing we need to deal with is the term u'' —which we will approximate using a *finite difference* on a sufficiently fine grid. Let $x_i = ih$ for $i = 0, \dots, N+1$, with $h = 1/(N+1)$. Taylor expansion gives the centered approximation

$$u''(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1}))}{h^2}. \quad (6)$$

We show that this approximation gets better and better with $h \rightarrow 0$ in the following sense:

Lemma 6 (Truncation error of the centered second difference). *If $u \in C^4$ near x_i , then*

$$\frac{u(x_i + h) - 2u(x_i) + u(x_i - h))}{h^2} = u''(x_i) + \frac{h^2}{12} u^{(4)}(\xi_i)$$

for some $\xi_i \in (x_i - h, x_i + h)$. In particular, the local truncation error is $O(h^2)$.

Proof. Taylor’s theorem with fourth-order remainder gives

$$u(x_i + h) = u_i + hu'_i + \frac{h^2}{2}u''_i + \frac{h^3}{6}u'''_i + \frac{h^4}{24}u^{(4)}(\xi_+),$$

$$u(x_i - h) = u_i - hu'_i + \frac{h^2}{2}u''_i - \frac{h^3}{6}u'''_i + \frac{h^4}{24}u^{(4)}(\xi_-).$$

Adding the above two, subtracting $2u_i$, and dividing by h^2 yields:

$$\frac{u(x_i + h) - 2u_i + u(x_i - h)}{h^2} = u''_i + \frac{h^2}{24}(u^{(4)}(\xi_+) + u^{(4)}(\xi_-)).$$

By continuity, the average of the two fourth derivatives equals $u^{(4)}(\xi_i)$ for some point between them. $\square$

After substituting this approximation for u'' , we approximate (1) by a linear system of equations. We represent u by the vector of its grid values and enforce the differential equation at each interior grid point. The discrete problem is

$$A_h U = F, \quad A_h = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & \\ & & & -1 & 2 \end{bmatrix}. \quad (7)$$

From the point of view of a linear solver, A_h is sparse, which makes specialized algorithms available. But a small equation residual is not yet a small solution error. Let $u_h = (u(x_1), \dots, u(x_N))^T$ be the exact PDE solution sampled on the grid and define the truncation residual

$$\tau_h := A_h u_h - F.$$

Lemma 6 gives us $\|\tau_h\|_\infty = O(h^2)$. Since the computed solution satisfies $A_h U = F$, the error in approximating *the solution* obeys

$$U - u_h = -A_h^{-1} \tau_h,$$

In particular, the matrix A_h^{-1} could amplify the error in our approximation. (e.g. we might worry the division by h^2 worsens the smallest eigenvalue of the matrix). We will show this doesn't happen for the Poisson problem we've been considering:

Lemma 7 (Second-order convergence of the finite-difference Poisson solver). *Consider*

$$-u''(x) = f(x), \quad x \in (0, 1), \quad u(0) = u(1) = 0,$$

and assume $u \in C^4([0, 1])$.

Let

$$x_i = ih, \quad i = 0, \dots, N+1, \quad h = \frac{1}{N+1},$$

and let $U = (U_1, \dots, U_N)^T$ solve

$$A_h U = F,$$

where

$$A_h = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \quad F_i = f(x_i).$$

Then

$$\max_{1 \leq i \leq N} |U_i - u(x_i)| \leq \frac{h^2}{96} \|u^{(4)}\|_{L^\infty(0,1)}.$$

In particular,

$$\max_i |U_i - u(x_i)| = O(h^2).$$

Proof. Let

$$u_h = \begin{bmatrix} u(x_1) \\ \vdots \\ u(x_N) \end{bmatrix}$$

be the exact PDE solution sampled at the grid points. By Lemma 6, we have

$$\frac{-u(x_{i-1}) + 2u(x_i) - u(x_{i+1}))}{h^2} = -u''(x_i) + \tau_i,$$

where $|\tau_i| \leq \frac{h^2}{12} \|u^{(4)}\|_{L^\infty(0,1)}$. Since $-u''(x_i) = f(x_i)$, this says $A_h u_h = F + \tau$, where $\tau = (\tau_1, \dots, \tau_N)^\top$.

The numerical solution satisfies

$$A_h U = F.$$

Subtracting the two equations gives

$$A_h(U - u_h) = -\tau.$$

Therefore, if $e := U - u_h$, then $e = -A_h^{-1}\tau$. Thus all that remains is to control the “size” of A_h^{-1} in an appropriate norm. If you know what these words mean: the lemma is trying to control $\|e\|_\infty$, and $\|A_h^{-1}\tau\|_\infty \leq \|A_h^{-1}\|_{\infty \rightarrow \infty} \|\tau\|_\infty$; so we should control $\|A_h^{-1}\|_{\infty \rightarrow \infty}$, that is the maximum row-sum of A_h^{-1} . Write

$$A_h = \frac{1}{h^2} T_N, \quad T_N = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

We claim the inverse of T_N has a nice form:

$$(T_N^{-1})_{ij} = \frac{\min(i, j)(N + 1 - \max(i, j))}{N + 1}. \quad (8)$$

While you can just check this, a derivation is instructive, and makes a connection to Green’s functions. To see this, fix j , and let $g^{(j)}$ denote the j -th column of T_N^{-1} . By definition,

$$T_N g^{(j)} = e_j,$$

where e_j is the j -th coordinate vector. Writing this equation in components gives

$$-g_{i-1}^{(j)} + 2g_i^{(j)} - g_{i+1}^{(j)} = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases}$$

with the convenient notational convention that

$$g_0^{(j)} = g_{N+1}^{(j)} = 0.$$

Away from $i = j$, we therefore have

$$-g_{i-1}^{(j)} + 2g_i^{(j)} - g_{i+1}^{(j)} = 0.$$

Equivalently,

$$g_{i+1}^{(j)} - g_i^{(j)} = g_i^{(j)} - g_{i-1}^{(j)},$$

Thus $g_i^{(j)}$ is linear in i on either side of j . Since $g_0^{(j)} = 0$, this linear function has no constant coefficient so we may write

$$g_i^{(j)} = Ai, \quad i \leq j.$$

Similarly, since $g_{N+1}^{(j)} = 0$,

$$g_i^{(j)} = B(N+1-i), \quad i \geq j.$$

The two expressions must agree at $i = j$, so

$$Aj = B(N+1-j).$$

It remains to use the equation at the source point $i = j$:

$$-g_{j-1}^{(j)} + 2g_j^{(j)} - g_{j+1}^{(j)} = 1.$$

Substituting the two linear forms,

$$-A(j-1) + 2Aj - \frac{Aj}{N+1-j}(N-j) = 1.$$

Solving this equation, we get

$$A = \frac{N+1-j}{N+1}, \quad B = \frac{j}{N+1}.$$

We conclude that

$$g_i^{(j)} = \begin{cases} \frac{i(N+1-j)}{N+1}, & i \leq j, \\ \frac{j(N+1-i)}{N+1}, & i \geq j. \end{cases}$$

Equivalently,

$$(T_N^{-1})_{ij} = \frac{\min(i, j)(N+1-\max(i, j))}{N+1}.$$

as we wanted.

Remark on Green's functions. If you know what it is—the operator above is a natural discrete analogue of the continuous Green's function for the Poisson problem we're considering. In particular, the continuous Green's function for

$$-u'' = \delta_y, \quad u(0) = u(1) = 0,$$

is

$$G(x, y) = \begin{cases} x(1-y), & x \leq y, \\ y(1-x), & x \geq y. \end{cases}$$

has the same piecewise-linear shape. In fact, because $A_h = h^{-2}T_N$, the formulas agree with the precise scaling

$$(A_h^{-1})_{ij} = h G(x_i, x_j). \tag{9}$$

Therefore the discrete solution can be written as

$$U_i = \sum_{j=1}^N (A_h^{-1})_{ij} F_j = h \sum_{j=1}^N G(x_i, x_j) f(x_j),$$

which is the Riemann-sum analogue of the continuous representation $u(x) = \int_0^1 G(x, y) f(y) \, dy$.² Returning to our bound on the “size” of A_h^{-1} , fix a row i . From

$$(T_N^{-1})_{ij} = \frac{\min(i, j)(N+1 - \max(i, j))}{N+1},$$

we split the row sum according to whether $j \leq i$ or $j > i$:

$$\begin{aligned} \sum_{j=1}^N (T_N^{-1})_{ij} &= \sum_{j=1}^i \frac{j(N+1-i)}{N+1} + \sum_{j=i+1}^N \frac{i(N+1-j)}{N+1} \\ &= \frac{N+1-i}{N+1} \sum_{j=1}^i j + \frac{i}{N+1} \sum_{j=i+1}^N (N+1-j) \\ &= \frac{N+1-i}{N+1} \frac{i(i+1)}{2} + \frac{i}{N+1} \frac{(N-i)(N-i+1)}{2} \\ &= \frac{i(N+1-i)}{2(N+1)} [(i+1) + (N-i)] \\ &= \frac{i(N+1-i)}{2}. \end{aligned}$$

Therefore

$$\sum_{j=1}^N (A_h^{-1})_{ij} = h^2 \sum_{j=1}^N (T_N^{-1})_{ij} = h^2 \frac{i(N+1-i)}{2}.$$

Since the choice of discretization points satisfies

$$x_i = ih, \quad (N+1-i)h = 1 - x_i,$$

we obtain

$$\sum_{j=1}^N (A_h^{-1})_{ij} = \frac{x_i(1-x_i)}{2}.$$

Since $x_i(1-x_i) \leq \frac{1}{4}$, we obtain $\sum_{j=1}^N (A_h^{-1})_{ij} \leq \frac{1}{8}$. Therefore

$$|e_i| = \left| \sum_{j=1}^N (A_h^{-1})_{ij} \tau_j \right| \leq \|\tau\|_\infty \sum_{j=1}^N (A_h^{-1})_{ij} \leq \frac{1}{8} \|\tau\|_\infty.$$

Taking the maximum over i ,

$$\|e\|_\infty \leq \frac{1}{8} \|\tau\|_\infty.$$

Finally,

$$\|\tau\|_\infty \leq \frac{h^2}{12} \|u^{(4)}\|_{L^\infty(0,1)},$$

so

$$\|U - u_h\|_\infty \leq \frac{h^2}{96} \|u^{(4)}\|_{L^\infty(0,1)}.$$

Thus

$$\max_i |U_i - u(x_i)| = O(h^2).$$

□

²If this remark doesn't make sense right now, don't worry, it's not essential!

How to interpret the proof. The argument is an instance of a more general principle:

- The first property that $A_h u_h - F = O(h^2)$ is called *consistency*, and it controls how well the exact solution satisfies the discretized equations.
- The property $\|A_h^{-1}\|_{\infty \rightarrow \infty} \leq \frac{1}{8}$ is called *stability* and prevents the discrete solve from amplifying that residual as the grid is refined.

3.2 Galerkin finite elements (on an unstructured mesh)

On a general domain, a mesh may partition Ω into intervals, triangles, quadrilaterals, tetrahedra, or other irregular cells. To make the Galerkin construction concrete, let's consider the variable-coefficient Poisson problem

$$-\nabla \cdot (\kappa(x) \nabla u) = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega, \quad 0 < a_0 \leq \kappa(x) \leq a_1. \quad (10)$$

Let V denote the admissible functions: they vanish on the boundary and have square-integrable first derivatives. The standard shorthand for this space is $H_0^1(\Omega)$, but the important point here is just that all first-derivative integrals below are well defined. Define

$$\mathbf{a}(w, v) := \int_{\Omega} \kappa(x) \nabla w \cdot \nabla v \, dx, \quad \ell(v) := \int_{\Omega} f v \, dx. \quad (11)$$

The weak problem is

$$\text{find } u \in V \quad \text{such that} \quad \mathbf{a}(u, v) = \ell(v) \quad \text{for every } v \in V.$$

A finite-element space V_h is spanned by mesh-based functions $\phi_1, \dots, \phi_N$, usually supported on only a few neighboring cells. We represent

$$u_h(x) = \sum_{j=1}^N U_j \phi_j(x).$$

The method is called *conforming* when every finite-element function is also admissible for the original weak problem, that is, $V_h \subset V$. The Galerkin approximation is

$$\text{find } u_h \in V_h \quad \text{such that} \quad \mathbf{a}(u_h, v_h) = \ell(v_h) \quad \text{for every } v_h \in V_h.$$

Testing with each basis function $v_h = \phi_i$ turns this statement into the *linear system* $KU = b$, with

$$K_{ij} = \mathbf{a}(\phi_j, \phi_i) = \int_{\Omega} \kappa(x) \nabla \phi_j \cdot \nabla \phi_i \, dx, \quad b_i = \ell(\phi_i) = \int_{\Omega} f \phi_i \, dx.$$

Thus the weak differential equation has again become a finite linear-algebra problem !

Consistency is particularly simple for a conforming method. Every $v_h \in V_h$ is a valid test function for the exact weak solution, so u already satisfies

$$\mathbf{a}(u, v_h) = \ell(v_h) \quad \text{for every } v_h \in V_h.$$

The approximation error instead comes from requiring $u_h \in V_h$, while in general $u \notin V_h$. Céa's lemma provides a way to quantify this error:

Theorem 8 (Céa's lemma). *Let $u \in V$ solve the weak problem and $u_h \in V_h$ solve the conforming Galerkin problem for V_h . Suppose*

$$|\mathbf{a}(w, v)| \leq M \|w\|_V \|v\|_V, \quad \mathbf{a}(v, v) \geq \alpha \|v\|_V^2$$

for all $w, v \in V$, where $M, \alpha > 0$. Then

$$\|u - u_h\|_V \leq \frac{M}{\alpha} \inf_{w_h \in V_h} \|u - w_h\|_V.$$

In other words, up to a “conditioning” factor M/α , the Galerkin solution is as accurate as the best approximation to u available in the finite-element space.

Before proving the theorem, we need one short identity: the error is orthogonal to the finite-element space in the bilinear form $\mathbf{a}$.

Lemma 9 (Galerkin orthogonality). *Let $u \in V$ solve the weak problem and let $u_h \in V_h$ solve the Galerkin problem. Then*

$$\mathbf{a}(u - u_h, v_h) = 0 \quad \text{for every } v_h \in V_h.$$

Proof. For every $v_h \in V_h \subset V$, the exact solution satisfies $\mathbf{a}(u, v_h) = \ell(v_h)$; likewise, the finite-element solution satisfies $\mathbf{a}(u_h, v_h) = \ell(v_h)$. Subtracting the two equalities gives $\mathbf{a}(u - u_h, v_h) = 0$. $\square$

The relevant stability property is *coercivity* (alternatively, positive-definiteness of the bilinear form)—one of the conditions in Céa’s lemma. For our equation, we can show this condition is satisfied. Since $\kappa(x) \geq a_0 > 0$,

$$\mathbf{a}(v, v) = \int_{\Omega} \kappa(x) |\nabla v|^2 dx \geq a_0 \|\nabla v\|_{L^2(\Omega)}^2.$$

Same as before, we use the natural energy norm³

$$\|v\|_V := \|\nabla v\|_{L^2(\Omega)}.$$

Thus

$$\mathbf{a}(v, v) \geq a_0 \|v\|_V^2.$$

Moreover,

$$|\mathbf{a}(w, v)| \leq a_1 \|w\|_V \|v\|_V$$

by the Cauchy–Schwarz inequality.

Proof of Céa’s lemma. Let $w_h \in V_h$ be arbitrary. By coercivity,

$$\alpha \|u - u_h\|_V^2 \leq \mathbf{a}(u - u_h, u - u_h).$$

Write

$$u - u_h = (u - w_h) + (w_h - u_h).$$

Therefore

$$\mathbf{a}(u - u_h, u - u_h) = \mathbf{a}(u - u_h, u - w_h) + \mathbf{a}(u - u_h, w_h - u_h).$$

Since

$$w_h - u_h \in V_h,$$

Galerkin orthogonality implies

$$\mathbf{a}(u - u_h, w_h - u_h) = 0.$$

Hence

$$\alpha \|u - u_h\|_V^2 \leq \mathbf{a}(u - u_h, u - w_h).$$

³If you know what these words mean: because functions in V vanish on the boundary, the Poincaré inequality ensures that this is a genuine norm rather than merely a seminorm.

Using boundedness of the bilinear form,

$$\alpha \|u - u_h\|_V^2 \leq M \|u - u_h\|_V \|u - w_h\|_V.$$

If $u = u_h$, there is nothing to prove. Otherwise, dividing by $\|u - u_h\|_V$ gives

$$\|u - u_h\|_V \leq \frac{M}{\alpha} \|u - w_h\|_V.$$

Since $w_h \in V_h$ was arbitrary, taking the infimum over V_h yields

$$\|u - u_h\|_V \leq \frac{M}{\alpha} \inf_{w_h \in V_h} \|u - w_h\|_V.$$

□

For the variable-coefficient Poisson problem (10), we may take

$$M = a_1, \quad \alpha = a_0,$$

so

$$\|u - u_h\|_V \leq \frac{a_1}{a_0} \inf_{w_h \in V_h} \|u - w_h\|_V.$$

What the theorem implies. Céa's lemma reduces the numerical-analysis problem to an approximation problem. If the spaces V_h become rich enough that

$$\inf_{w_h \in V_h} \|u - w_h\|_V \longrightarrow 0 \quad \text{as } h \rightarrow 0,$$

then immediately

$$\|u - u_h\|_V \longrightarrow 0.$$

For example, continuous piecewise-linear finite elements on a sufficiently regular family of meshes have this approximation property.

3.3 Spectral representations: finitize in a global basis

So far, our basic strategy has been to “finitize” the PDE: replace an infinite-dimensional function u by finitely many numbers, then solve for those numbers.

For finite differences, the finite representation was local:

$$u \longmapsto (u(x_1), \dots, u(x_N)).$$

For finite elements, we instead wrote

$$u_h(x) = \sum_{j=1}^N U_j \phi_j(x),$$

where the basis functions ϕ_j are usually locally supported on a mesh.

But there is no reason that a finite representation must be local. In particular, if the solution is globally smooth, it can be much more efficient to use a small number of *global* basis functions:

$$u_K(x) = \sum_{k=1}^K U_k \phi_k(x),$$

where each ϕ_k is nonzero over most or all of the domain. The basic hope is that we can choose a basis in which two things happen:

1. The solution can be represented accurately using relatively few coefficients;
2. The differential operator itself becomes “simple” in some usable way in those coordinates.

Fourier bases provide the canonical example. Consider the periodic function

$$u(x) = \sin(3x) + 0.1 \sin(8x), \quad x \in [0, 2\pi].$$

A grid-based representation would keep track of the values

$$u(x_1), \dots, u(x_N)$$

at many points in order to resolve both oscillations.

In the Fourier basis, however, the entire function is represented by only two nonzero sine coefficients:

$$u(x) = 1 \cdot \sin(3x) + 0.1 \cdot \sin(8x).$$

Thus a globally smooth field may have a very economical representation in a global basis.

3.3.1 Fourier analysis for periodic functions

There’s a general phenomenon here that will be useful for us and is worth recapping. A function $u : \mathbb{R} \rightarrow \mathbb{R}$ is called 2π -periodic if

$$u(x + 2\pi) = u(x) \quad \text{for every } x.$$

The natural oscillatory building blocks for such functions are e^{ikx} , $k \in \mathbb{Z}$. From Euler’s identity we have:

$$e^{ikx} = \cos(kx) + i \sin(kx),$$

so these are simply sine and cosine waves written compactly. The integer k is called the *frequency* or *Fourier mode*: small $|k|$ corresponds to slowly varying wave, while large $|k|$ corresponds to rapidly oscillating wave. Any sufficiently regular periodic function can be expanded as

$$u(x) = \sum_{k \in \mathbb{Z}} \hat{u}_k e^{ikx},$$

where

$$\hat{u}_k = \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-ikx} dx$$

is the k -th *Fourier coefficient*. Thus the Fourier coefficients are simply the coordinates of u in the *basis* $\{e^{ikx}\}_{k \in \mathbb{Z}}$. A spectral approximation keeps only finitely many modes:

$$u_K(x) = \sum_{|k| \leq K} \hat{u}_k e^{ikx}.$$

Thus, if the coefficients decay rapidly with $|k|$ (i.e. there are no “very fine scale, rapid oscillations”), then a small number of global modes can approximate u very accurately.

Why is this a useful basis? It turns out the Fourier basis plays very nicely with the **Laplacian** operator. Namely, we have:

$$\frac{d}{dx} e^{ikx} = ik e^{ikx},$$

and

$$-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}.$$

Recalling that an eigenfunction of an operator $\mathcal{L}$ is a nonzero function ϕ for which $\mathcal{L}\phi = \lambda\phi$: applying the operator changes only its scale, not its shape. The identity above says that e^{ikx} is an eigenfunction of the *negative* periodic Laplacian $-d^2/dx^2$, with eigenvalue k^2 .⁴ This is the function-space analogue of a matrix eigenvector. Consequently, if

$$u(x) = \sum_k \hat{u}_k e^{ikx},$$

then

$$-u''(x) = \sum_k k^2 \hat{u}_k e^{ikx}.$$

Equivalently,

$$\widehat{-u''}_k = k^2 \hat{u}_k.$$

This is the infinite-dimensional analogue of diagonalizing a matrix: after changing coordinates to the Fourier basis, applying the differential operator reduces to multiplying each coordinate by a scalar.

To see why this property is useful, consider the Poisson equation with periodic boundary conditions:

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

Write

$$u(x) = \sum_{k \in \mathbb{Z}} \hat{u}_k e^{ikx}, \quad f(x) = \sum_{k \in \mathbb{Z}} \hat{f}_k e^{ikx}.$$

Substituting into the PDE gives

$$\sum_k k^2 \hat{u}_k e^{ikx} = \sum_k \hat{f}_k e^{ikx}.$$

Matching Fourier coefficients,

$$k^2 \hat{u}_k = \hat{f}_k,$$

and therefore, for $k \neq 0$ ⁵,

$$\hat{u}_k = \frac{\hat{f}_k}{k^2}.$$

Thus the PDE decouples into one scalar equation for each Fourier mode.

If we truncate to the nonzero modes $0 < |k| \leq K$, the finite-dimensional system is diagonal:

$$\begin{bmatrix} \ddots & & \\ & k^2 & \\ & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \hat{u}_k \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \hat{f}_k \\ \vdots \end{bmatrix}.$$

Compare this with the finite-difference discretization $A_h U = F$, where A_h is sparse but not diagonal.

So the attraction of a spectral representation is not only that it stores the field differently—but the right global coordinates can make the PDE itself much simpler.

⁴For simplicity, we're doing the 1-D case—but the picture is completely analogous in the higher dimensional case.

⁵A periodic solution exists only if the forcing has zero mean, $\hat{f}_0 = 0$, because the integral of $-u''$ over a period is zero. The solution is determined only up to an additive constant, so we select the unique zero-mean solution by imposing $\hat{u}_0 = 0$.

3.3.2 Fourier analysis of $\mathbb{R}$

On the whole real line, frequencies are no longer restricted to integers. Instead, every real frequency $\xi \in \mathbb{R}$ is allowed. Precisely, the discrete Fourier coefficients are replaced by a continuous function of frequency, the *Fourier transform*:

$$\widehat{u}(\xi) = \int_{\mathbb{R}} u(x) e^{-i\xi x} dx.$$

Under suitable assumptions, u can be recovered from $\widehat{u}$ through the inverse Fourier transform,

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

Thus the Fourier series

$$u(x) = \sum_{k \in \mathbb{Z}} \widehat{u}_k e^{ikx}$$

and the Fourier transform

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

express the same basic idea: decompose a function into a sum/integral of oscillations of different frequencies. The difference is that periodicity produces a *discrete* set of frequencies $k \in \mathbb{Z}$, whereas on $\mathbb{R}$ frequency is a *continuous* variable $\xi \in \mathbb{R}$.

This Fourier basis is also useful as it diagonalizes the Laplacian over $\mathbb{R}$. Consider, for instance, the heat equation

$$u_t = \nu u_{xx}.$$

Taking Fourier coefficients gives

$$\frac{d\widehat{u}_\xi}{dt} = -\nu \xi^2 \widehat{u}_\xi.$$

Hence every mode evolves independently:

$$\widehat{u}_\xi(t) = e^{-\nu \xi^2 t} \widehat{u}_\xi(0).$$

Thus, the PDE has been decomposed into a collection of decoupled scalar ODEs. This also makes the physics transparent: high-frequency modes, corresponding to large $|k|$, decay much faster than low-frequency modes. This is the spectral signature of diffusion.

3.3.3 Some remarks

Something is still finite: A spectral computation still retains only finitely many modes, so its efficiency depends on the omitted Fourier tail being small. Smooth functions have rapidly decaying Fourier coefficients; for analytic functions the decay is often exponentially fast. Consequently, a modest cutoff K can give very high accuracy. For nonsmooth solutions the coefficients decay more slowly, so many more modes may be needed. This finite spectral cutoff will reappear when we discuss Fourier neural operators [4] and related methods that operate in Fourier space.

The price of globality. The same global structure that makes spectral methods powerful can also make them awkward. A finite-element basis function is typically local: changing one coefficient modifies the approximation only near one part of the mesh. A Fourier mode is nonzero everywhere. Changing one Fourier coefficient modifies the approximation over the entire domain. As a result, spectral representations are especially attractive for smooth solutions; periodic or geometrically simple domains; problems whose differential operators have a simple spectral representation.

They are less natural when the geometry is highly irregular; strong local refinement is needed; the solution contains discontinuities or shocks.

Local versus global finitization. The three representations we have seen can now be compared as follows:

method	finite representation	main advantage
finite differences	values on a regular grid	simple local mesh
finite elements	coefficients of local basis functions	complex mesh geometry, local refinement
spectral methods	coefficients of global basis functions	rapid approximation, simple operators

4 Time discretization: from continuous dynamics to an iteration

A time-dependent differential equation contains a second continuous quantity that must be finitized: time. This issue already appears for an ordinary differential equation, before any spatial discretization is introduced. Consider the initial-value problem

$$\dot{y}(t) = f(y(t)), \quad y(0) = y_0, \quad y(t) \in \mathbb{R}^d. \quad (12)$$

A time-stepping method replaces the continuous trajectory $y(t)$ by a finite sequence

$$Y^0, Y^1, \dots, Y^N, \quad Y^n \approx y(t_n), \quad t_n = n\Delta t.$$

There are three immediate, but distinct questions:

1. How accurately does one numerical step approximate one exact step?
2. How are errors propagated by repeated numerical steps?
3. Does the numerical method reproduce the qualitative growth or decay of the underlying dynamics?

4.1 Explicit Euler and the local one-step error

Approximating the derivative at t_n by a forward difference gives

$$\dot{y}(t_n) \approx \frac{y(t_{n+1}) - y(t_n)}{\Delta t}.$$

This motivates the *explicit Euler method*

$$Y^{n+1} = Y^n + \Delta t f(Y^n). \quad (13)$$

The update is called *explicit* because Y^{n+1} is computed directly from the already known value Y^n . It is useful to describe a time-stepping method as a one-step map

$$Y^{n+1} = \Phi_{\Delta t}(Y^n).$$

For explicit Euler,

$$\Phi_{\Delta t}(v) = v + \Delta t f(v).$$

Let $S_{\Delta t}(v)$ denote the exact solution after time Δt when the initial state is v .

Definition 10 (Local one-step defect). The local one-step defect of a numerical map $\Phi_{\Delta t}$ is

$$d_{\Delta t}(v) := S_{\Delta t}(v) - \Phi_{\Delta t}(v).$$

The local defect measures the error made by *one step started from the exact state*. It does not yet measure the error after many numerical steps.

Lemma 11 (Local defect of explicit Euler). *Suppose the exact solution of (12) is twice continuously differentiable on $[0, T]$, and let*

$$M := \max_{0 \leq t \leq T} \|\ddot{y}(t)\|.$$

Then the explicit Euler defect at step n ,

$$d^{n+1} := y(t_{n+1}) - y(t_n) - \Delta t f(y(t_n)),$$

satisfies

$$\|d^{n+1}\| \leq \frac{M}{2}(\Delta t)^2.$$

Thus explicit Euler has local order one.

Proof. Taylor's theorem with integral remainder gives

$$y(t_{n+1}) = y(t_n) + \Delta t \dot{y}(t_n) + \int_{t_n}^{t_{n+1}} (t_{n+1} - s) \ddot{y}(s) ds.$$

Since $\dot{y}(t_n) = f(y(t_n))$,

$$d^{n+1} = \int_{t_n}^{t_{n+1}} (t_{n+1} - s) \ddot{y}(s) ds.$$

Therefore

$$\|d^{n+1}\| \leq M \int_{t_n}^{t_{n+1}} (t_{n+1} - s) ds = \frac{M}{2}(\Delta t)^2.$$

□

A single explicit Euler step therefore makes an $O((\Delta t)^2)$ error. However, on a fixed interval $[0, T]$, the method takes approximately $N = \frac{T}{\Delta t}$ steps. Even if the local errors merely added, one would expect

$$\frac{T}{\Delta t} O((\Delta t)^2) = O(\Delta t)$$

global error. The real issue is slightly subtler because an error introduced early in the trajectory may be amplified by later updates.

4.2 Error propagation after multiple steps

Assume that f is L -Lipschitz:

$$\|f(v) - f(w)\| \leq L\|v - w\|. \quad (14)$$

Then the explicit Euler map satisfies

$$\begin{aligned} \|\Phi_{\Delta t}(v) - \Phi_{\Delta t}(w)\| &= \|v - w + \Delta t(f(v) - f(w))\| \\ &\leq (1 + L\Delta t)\|v - w\|. \end{aligned}$$

Thus one step can *amplify* a perturbation by at most $1 + L\Delta t$. Over n steps, we'd expect

$$(1 + L\Delta t)^n \leq e^{L\Delta t n}.$$

On every fixed time interval, this amplification remains bounded independently of the number of steps. (But depends on the length of the interval.) Quantitatively, we have:

Theorem 12 (Fixed-interval error of explicit Euler). *Suppose:*

1. f is L -Lipschitz on a region containing the exact and numerical trajectories;
2. the exact solution satisfies

$$\max_{0 \leq t \leq T} \|\ddot{y}(t)\| \leq M;$$

3. $Y^0 = y_0$.

Then, for every $t_n = n\Delta t \leq T$,

$$\|Y^n - y(t_n)\| \leq \frac{MT}{2} e^{LT} \Delta t.$$

In particular, explicit Euler converges with first-order global accuracy:

$$\max_{t_n \leq T} \|Y^n - y(t_n)\| = O(\Delta t).$$

Proof. Define the global error

$$e^n := Y^n - y(t_n).$$

The numerical update is

$$Y^{n+1} = Y^n + \Delta t f(Y^n),$$

while the exact solution satisfies

$$y(t_{n+1}) = y(t_n) + \Delta t f(y(t_n)) + d^{n+1}.$$

Subtracting gives

$$e^{n+1} = e^n + \Delta t (f(Y^n) - f(y(t_n))) - d^{n+1}.$$

Using Lipschitz continuity and the local-defect bound,

$$\|e^{n+1}\| \leq (1 + L\Delta t)\|e^n\| + \frac{M}{2}(\Delta t)^2.$$

Since $e^0 = 0$, iteration yields

$$\|e^n\| \leq \frac{M}{2}(\Delta t)^2 \sum_{j=0}^{n-1} (1 + L\Delta t)^j.$$

For $j \leq n$,

$$(1 + L\Delta t)^j \leq e^{Lj\Delta t} \leq e^{LT}.$$

Consequently,

$$\|e^n\| \leq \frac{M}{2}(\Delta t)^2 n e^{LT}.$$

Since $n\Delta t = t_n \leq T$,

$$\|e^n\| \leq \frac{MT}{2} e^{LT} \Delta t.$$

□

The proof can in some sense be viewed as a time-stepping analogue of the consistency–stability–convergence message:

$$\underbrace{\text{one-step defect } O((\Delta t)^2)}_{\text{analogue of consistency}} + \underbrace{\text{controlled perturbation amplification}}_{\text{analogue of stability}} \implies \underbrace{\text{global error } O(\Delta t)}_{\text{convergence}}.$$

Note that when L or T is large, this worst-case error bound can grow quickly. Thus, if the undiscretized system itself is expansive, we should expect accurate long rollouts to be more difficult. We will return to this point when discussing forecasting and chaotic behavior.

Remark 13 (Two meanings of stability). The error-propagation estimate above is a finite-time perturbation-stability statement. Numerical ODE analysis also uses the more specialized notion of *absolute stability*, introduced below, to ask whether a method reproduces rapidly decaying modes without numerical blow-up. The two ideas are related but are not identical.

Lookahead to machine learned solvers The same issue appears in learned time evolution. An autoregressive simulator is itself a one-step map,

$$\hat{U}^{n+1} = G_\theta(\hat{U}^n).$$

Let $S_{\Delta t}$ denote the true evolution over one timestep, and suppose that on the relevant states $\|G_\theta(v) - S_{\Delta t}(v)\| \leq \varepsilon$. If G_θ is L_θ -Lipschitz, then

$$\begin{aligned} \|\hat{U}^{n+1} - U^{n+1}\| &= \|G_\theta(\hat{U}^n) - S_{\Delta t}(U^n)\| \\ &\leq \|G_\theta(\hat{U}^n) - G_\theta(U^n)\| + \|G_\theta(U^n) - S_{\Delta t}(U^n)\| \\ &\leq L_\theta \|\hat{U}^n - U^n\| + \varepsilon. \end{aligned}$$

Thus, starting from the exact initial state,

$$\|\hat{U}^n - U^n\| \leq \varepsilon \sum_{j=0}^{n-1} L_\theta^j.$$

Just as for a numerical integrator, small one-step error is therefore not enough: it also matters how errors are propagated under repeated application of the learned update.

If $L_\theta > 1$, perturbations can be amplified from step to step; if $L_\theta \approx 1$, even small errors can accumulate over a long rollout; and strong contraction can suppress errors but may also distort dynamics that are not themselves strongly dissipative.

4.3 Absolute stability and implicit Euler

To motivate the idea of implicit Euler, let us consider a very simple scalar equation

$$\dot{y} = -\alpha y, \quad \alpha > 0.$$

The solution to the equation is $y(t) = e^{-\alpha t} y(0)$, meaning the exact solution shrinks by the factor $e^{-\alpha \Delta t}$ over one timestep. Suppose a numerical method, on this very simple ODE, produces iterates satisfying:

$$Y^{n+1} = R(-\alpha \Delta t) Y^n.$$

For example, for explicit Euler,

$$Y^{n+1} = Y^n - \alpha \Delta t Y^n = (1 - \alpha \Delta t) Y^n,$$

so we have $R(z) = 1 + z$. It's then clear that the iterates Y^n do not explode in magnitude whenever $|R(-\alpha \Delta t)| \leq 1$.

To paint the full picture for this simple scalar equation, and cover both oscillating and decaying solutions, consider the equation

$$\dot{y} = \lambda y, \quad \lambda \in \mathbb{C}. \tag{15}$$

Its exact solution is $y(t) = e^{\lambda t} y_0$. To understand its behavior, write

$$\lambda = a + ib, \quad a = \operatorname{Re}(\lambda), \quad b = \operatorname{Im}(\lambda).$$

Then

$$e^{\lambda t} = e^{(a+ib)t} = e^{at}e^{ibt} = e^{at}(\cos(bt) + i \sin(bt)).$$

Thus the real part of λ controls the size of the solution, since

$$|y(t)| = e^{at}|y_0| = e^{\operatorname{Re}(\lambda)t}|y_0|,$$

while the imaginary part controls oscillation.

There are three basic cases:

- If $\operatorname{Re}(\lambda) < 0$, then $e^{\operatorname{Re}(\lambda)t} \rightarrow 0$, so the solution *decays exponentially*. If $\operatorname{Im}(\lambda) \neq 0$, it oscillates while decaying.
- If $\operatorname{Re}(\lambda) = 0$, then $|y(t)| = |y_0|$, so the solution neither grows nor decays. If $\operatorname{Im}(\lambda) \neq 0$, it undergoes a pure oscillation with angular frequency $|\operatorname{Im}(\lambda)|$.
- If $\operatorname{Re}(\lambda) > 0$, then $e^{\operatorname{Re}(\lambda)t} \rightarrow \infty$, so the solution *grows exponentially*. If $\operatorname{Im}(\lambda) \neq 0$, it oscillates while its amplitude grows.

In particular, when λ is real there is no oscillation: negative λ gives exponential decay, while positive λ gives exponential growth.

Let's write a one-step method applied to (15) as

$$Y^{n+1} = R(z)Y^n, \quad z := \lambda\Delta t.$$

The function $R(z)$ is the *stability function*: it is the factor by which one numerical step multiplies the mode.

Definition 14 (Absolute-stability region). The absolute-stability region of a time-stepping method is

$$\mathcal{S} := \{z \in \mathbb{C} : |R(z)| \leq 1\}.$$

Suppose $\operatorname{Re}(\lambda) < 0$ —to track the ground truth, we want $\lambda\Delta t \in \mathcal{S}$, so that a physically decaying system is not amplified numerically.

For explicit Euler, since

$$Y^{n+1} = Y^n + \Delta t \lambda Y^n = (1 + \lambda\Delta t)Y^n,$$

we have

$$R_{\text{EE}}(z) = 1 + z.$$

Implicit Euler is a different discretization method, which instead evaluates the vector field at the new state:

$$Y^{n+1} = Y^n + \Delta t f(Y^{n+1}). \tag{16}$$

Note that this update doesn't specify Y^{n+1} explicitly — one has to solve this equation to find what Y^{n+1} is. Hence, the name of the method. Nevertheless, for the scalar test equation, this equation is easy to invert, and just simplifies to

$$Y^{n+1} = Y^n + \lambda\Delta t Y^{n+1} \Leftrightarrow Y^{n+1} = \frac{1}{1 - \lambda\Delta t} Y^n$$

and therefore

$$R_{\text{IE}}(z) = \frac{1}{1 - z}.$$

It is then easy to see what is the range of z for which implicit and explicit Euler doesn't explode (still just for this simple scalar equation):

Proposition 15 (Absolute stability of explicit and implicit Euler). *For the scalar test equation (15):*

1. *Explicit Euler is absolutely stable precisely when $|1 + z| \leq 1$. For real $\lambda < 0$, this is equivalent to*

$$0 \leq \Delta t \leq \frac{2}{|\lambda|}.$$

2. *Implicit Euler is absolutely stable throughout the closed left complex half-plane:*

$$\operatorname{Re}(z) \leq 0 \implies \left| \frac{1}{1-z} \right| \leq 1.$$

Proof. For explicit Euler, the amplification factor is $1 + z$, so the definition of absolute stability gives

$$|1 + z| \leq 1.$$

When $z = \lambda \Delta t$ is real and nonpositive, this becomes $-2 \leq z \leq 0$. Since $\lambda < 0$, $0 \leq \Delta t \leq \frac{2}{|\lambda|}$.

For implicit Euler, $R_{\text{IE}}(z) = \frac{1}{1-z}$. If $\operatorname{Re}(z) \leq 0$, then

$$|1 - z|^2 = (1 - \operatorname{Re} z)^2 + (\operatorname{Im} z)^2 \geq 1.$$

Hence

$$|R_{\text{IE}}(z)| = \frac{1}{|1 - z|} \leq 1.$$

□

This yields the basic tradeoff:

	explicit Euler	implicit Euler
update	$Y^{n+1} = Y^n + \Delta t f(Y^n)$	$Y^{n+1} = Y^n + \Delta t f(Y^{n+1})$
work per step	evaluate f	solve an equation
stability region	bounded	contains the entire left half-plane

Note: stability is *not* accuracy. Consider

$$\dot{y} = -1000y, \quad y(0) = 1.$$

With $\Delta t = 0.005$, implicit Euler is stable and multiplies the solution by $\frac{1}{1+1000\Delta t} = \frac{1}{6}$ per step. After one step it predicts $Y^1 = 1/6$, whereas the exact value is $y(0.005) = e^{-5} \approx 6.74 \times 10^{-3}$. The trajectory is bounded and decaying but badly mis-estimates the rapid initial transient. Absolute stability only rules out artificial growth. A stable method may damp a mode too much, too little, or with the wrong phase.

4.4 Stiffness and wastefulness of explicit methods

Consider

$$\dot{y} = \begin{bmatrix} -1 & 0 \\ 0 & -1000 \end{bmatrix} y. \tag{17}$$

For $y(0) = (1, 1)^\top$, the exact solution is

$$y(t) = \begin{bmatrix} e^{-t} \\ e^{-1000t} \end{bmatrix}.$$

The two components therefore decay on dramatically different time scales. To see the consequence for a numerical method, take $\Delta t = 0.05$, a reasonable step for the slow rate 1 but far larger than the explicit fast-mode cutoff $2/1000 = 0.002$. After a short initial transient, the second component is essentially zero, and the scientifically interesting evolution may be governed by the slow first component. Nevertheless, explicit Euler must satisfy

$$\Delta t < \frac{2}{1000} = 0.002$$

because of the fast mode. Even at the stability limit, simulating until $T = 10$ requires 5000 steps, and in practice more are needed, although the fast component of the exact physical solution became negligible near the beginning of the trajectory. This illustrates the operational meaning of a phenomenon called *stiffness*:

A problem is stiff when stability forces an explicit method to use a timestep much smaller than the timestep needed to resolve the scientifically relevant slow evolution.

Implicit methods can often step over rapidly decaying modes, although each implicit step requires a linear or nonlinear solve.

5 How space and time discretization interact

So far we have discussed spatial discretization and time discretization largely separately. For a time-dependent PDE

$$\partial_t u = \mathcal{F}(u),$$

a spatial discretization produces a finite-dimensional ODE

$$\dot{U}(t) = F_h(U(t)).$$

We could then apply an ODE time-stepping method to this system. This “discretize space first, then integrate in time” approach is called the *method of lines*.

A natural question is whether the spatial resolution h and timestep Δt can be chosen independently. The answer is often no: refining the spatial grid can create increasingly fast modes in the semi-discrete ODE, which may force the timestep to shrink as well. The heat equation gives the simplest example.

5.1 Diffusion: spatial refinement creates fast decaying modes

We will illustrate the interaction between spatial and temporal discretization. A finer spatial grid does not only reduce the spatial approximation error—it also changes the ODE that the time integrator must solve. In particular, in this case, the largest-magnitude eigenvalues grow like h^{-2} . For explicit Euler, stability therefore will force $\Delta t = O(h^2)$. Thus h and Δt cannot be chosen independently.

Consider the periodic heat equation $u_t = \nu u_{xx}$. On a periodic grid $x_j = jh$, use the centered second-difference operator

$$(L_h U)_j = (U_{j+1} - 2U_j + U_{j-1})/h^2.$$

The spatially discretized ODE is therefore $\dot{U} = \nu L_h U$.

Assume for simplicity that the periodic grid has an even number of points. Consider a discrete Fourier mode, namely the vector with coordinates $U_j = e^{ij\theta}$, where $\theta = 2\pi k/N$ for $0 \leq k < N, k \in \mathbb{N}$. Applying L_h gives

$$(L_h U)_j = \frac{e^{i(j+1)\theta} - 2e^{ij\theta} + e^{i(j-1)\theta}}{h^2} = \frac{e^{ij\theta}e^{i\theta} - 2e^{ij\theta} + e^{ij\theta}e^{-i\theta}}{h^2}$$

$$\begin{aligned}
&= \frac{e^{i\theta} - 2 + e^{-i\theta}}{h^2} e^{ij\theta} = \frac{2\cos\theta - 2}{h^2} e^{ij\theta} = \frac{2(\cos\theta - 1)}{h^2} e^{ij\theta} \\
&= -\frac{4}{h^2} \sin^2\left(\frac{\theta}{2}\right) e^{ij\theta} = -\frac{4}{h^2} \sin^2\left(\frac{\theta}{2}\right) U_j.
\end{aligned} \tag{18}$$

Thus—as we saw before—the Fourier modes are eigenvectors for L_h , and the eigenvalues of L_h range down to approximately $-4/h^2$. As $h \rightarrow 0$, spatial refinement introduces increasingly fast decaying modes.

Proposition 16 (Explicit heat-equation stability restriction). *Apply explicit Euler to this discretized system, giving $U^{n+1} = U^n + \nu\Delta t L_h U^n$. Let $r := \nu\Delta t/h^2$. The update is nonexpansive in the ℓ_2 norm for all Fourier modes if $0 \leq r \leq 1/2$. If $r > 1/2$, the highest-frequency Fourier mode is amplified.*

Proof. By (18), the explicit update multiplies a Fourier mode by $g(\theta) = 1 - 4r \sin^2(\theta/2)$. Because L_h is diagonalized by the discrete Fourier basis, the update is nonexpansive in ℓ_2 precisely when $|g(\theta)| \leq 1$ for every discrete frequency.

Since $0 \leq \sin^2(\theta/2) \leq 1$, the condition $0 \leq r \leq 1/2$ implies $-1 \leq g(\theta) \leq 1$ for every mode. If $r > 1/2$, the grid contains the mode $\theta = \pi$, for which $g(\pi) = 1 - 4r < -1$. Hence that mode is amplified. $\square$

Thus, explicit Euler requires that $\Delta t \leq h^2/(2\nu)$. Implicit Euler instead gives $(I - \nu\Delta t L_h)U^{n+1} = U^n$.

Proposition 17 (Unconditional contractivity of implicit heat stepping). *For every $\Delta t > 0$, this implicit update is nonexpansive in the discrete ℓ_2 norm.*

Proof. For a Fourier mode, the implicit update has multiplier $g_{\text{imp}}(\theta) = 1/(1 + 4r \sin^2(\theta/2))$. For every $r \geq 0$, we have $0 < g_{\text{imp}}(\theta) \leq 1$, so $\|U^{n+1}\|_2 \leq \|U^n\|_2$. $\square$

How generic is this phenomenon? Spatial refinement in general can expose faster spatial modes. An explicit time integrator must remain stable on all of them, even if the actual quantity of interest is governed mainly by slower modes. However, the precise scaling of Δt as a function of h is a function both of the equation and the space discretization algorithm.

Lookahead to machine learned solvers This observation also matters for learned time-stepping schemes. A learned update G_θ trained using data at one spatial resolution h and one timestep Δt has learned a map associated with that particular discretization. Changing h changes the finite-dimensional dynamical system being evolved—as the heat-equation example makes explicit—while changing Δt changes the evolution map itself. Thus a model trained at one pair $(h, \Delta t)$ should not automatically be expected to remain accurate or stable at another.

5.2 Difficulties in general PDEs

After spatial discretization, a general, nonlinear PDE gives a finite-dimensional ODE $\dot{U} = F_h(U)$. The main new difficulty for generic PDEs is that the right-hand side is now nonlinear. In the heat equation, the spatial discretization produced a fixed matrix L_h , so the same collection of modes and decay rates governed the dynamics at every time. For a nonlinear problem, the effective rates of change may depend on the current state $U(t)$ and may vary during the simulation. Moreover, a PDE may contain several terms with very different time scales; one term may force an explicit method to take very small timesteps even though the remaining terms could be evolved accurately with much larger ones.

This motivates more involved methods, e.g. *implicit-explicit* (IMEX) methods. Suppose we split

$$\dot{U} = F_{\text{exp}}(U) + F_{\text{imp}}(U),$$

where F_{imp} contains the part that would otherwise impose a severe stability restriction, while F_{exp} is inexpensive to evaluate. A simple IMEX Euler step is

$$U^{n+1} = U^n + \Delta t F_{\text{exp}}(U^n) + \Delta t F_{\text{imp}}(U^{n+1}).$$

Thus only part of the dynamics is treated implicitly. For example, in a diffusion–reaction equation one might treat the diffusion term implicitly, since we have already seen that its spatial discretization produces very fast decaying modes as h becomes small, while treating a simpler reaction term explicitly.

A simple IMEX example. Consider the nonlinear ODE

$$\dot{y} = y - y^2.$$

To illustrate the IMEX idea, treat the linear term y implicitly and the nonlinear term $-y^2$ explicitly. The update is

$$\frac{y^{n+1} - y^n}{\Delta t} = y^{n+1} - (y^n)^2.$$

The unknown y^{n+1} appears implicitly, but only linearly. Rearranging gives

$$y^{n+1} - \Delta t y^{n+1} = y^n - \Delta t (y^n)^2,$$

and therefore

$$y^{n+1} = \frac{y^n - \Delta t (y^n)^2}{1 - \Delta t} = \frac{y^n(1 - \Delta t y^n)}{1 - \Delta t}.$$

The advantage comes with a computational cost: an implicit term places the unknown U^{n+1} on both sides of the update, so each timestep requires solving an algebraic equation. If the implicitly treated part is linear, this is a linear system; if it is nonlinear, one may need an iterative nonlinear solver. IMEX methods are useful because a suitable splitting can obtain much of the stability benefit of implicit stepping while keeping the solve substantially simpler than making the entire nonlinear problem implicit.

References

- [1] W. E and B. Yu. The Deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2018. <https://doi.org/10.1007/s40304-018-0127-z>.
- [2] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations. arXiv:1912.00873, 2019. <https://arxiv.org/abs/1912.00873>.
- [3] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. *hp*-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021. <https://doi.org/10.1016/j.cma.2020.113547>.
- [4] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. <https://openreview.net/forum?id=c8P9NQVtmn0>.
- [5] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. <https://doi.org/10.1016/j.jcp.2018.10.045>.