MA5233 Computational Mathematics

Lecture 7: Finite Differences

Simon Etter

Semester I, AY 2020/2021

Finite Differences

Problem statement

Given $\Omega \subset \mathbb{R}^d$ and $f : \Omega \to \mathbb{R}$, determine $u : \Omega \to \mathbb{R}$ such that

$$-\Delta u(x) = f(x) \qquad \text{for all } x \in \Omega.$$

Terminology and notation

- $\Delta = \frac{\partial^2}{\partial x_1^2} + \ldots + \frac{\partial^2}{\partial x_d^2}$ is called the *Laplace operator* or *Laplacian*.
- The above problem is known as the *Poisson equation*.
- The Poisson equation is a particular example of a *partial differential equation (PDE)*, i.e. an equation in terms of an unknown function $u(x)$ and its partial derivatives.

Finite Differences

Example

For $\Omega = (0, 1)$ and $f(x) = x - x^2$, Poisson's equation becomes

$$-u''(x) = x - x^2 \qquad \text{for all } x \in (0, 1).$$

Taking antiderivatives, we obtain

$$\begin{aligned} -u'(x) &= \tfrac{1}{2}\, x^2 - \tfrac{1}{3}\, x^3 + c_1, \\ -u(x) &= \tfrac{1}{6}\, x^3 - \tfrac{1}{12}\, x^4 + c_1\, x + c_2, \end{aligned}$$

where $c_1, c_2 \in \mathbb{R}$ are some unspecified parameters.

Discussion

The above example shows that additional constraints are required to guarantee that Poisson's equation has a unique solution $u(x)$.

These additional constraints typically take the form of boundary conditions, see next slide.

Finite Differences

Terminology: Boundary conditions

- Dirichlet boundary conditions: $u(x) = g(x)$ for all $x \in \partial\Omega$.
- Neumann boundary conditions: $\frac{\partial u}{\partial n}(x) = g(x)$ for all $x \in \partial\Omega$.

If $g(x) = 0$, then these boundary conditions are called *homogeneous*.

$\partial\Omega$ denotes the boundary of $\Omega \subset \mathbb{R}$.
$\frac{\partial u}{\partial n}$ denotes the directional derivative of $u(x)$ in the direction normal to the boundary.

Example (continued)

For $\Omega = (0, 1)$, the homogeneous Dirichlet boundary conditions are

$$u(0) = u(1) = 0.$$

These conditions provide two equations for determining the parameters c_1, c_2 in the formula

$$u(x) = \tfrac{1}{6}\,x^3 - \tfrac{1}{12}\,x^4 + c_1\,x + c_2$$

derived on the previous slide.

Finite Differences

Example (continued)

For $\Omega = (0, 1)$, the homogeneous Neumann boundary conditions are

$$u'(0) = u'(1) = 0.$$

These conditions provide two equations for determining the c_1 in

$$u(x) = \tfrac{1}{6}\, x^3 - \tfrac{1}{12}\, x^4 + c_1\, x + c_2,$$

and no equation for determining c_2 since

$$u'(x) = \tfrac{1}{2}\, x^2 - \tfrac{1}{3}\, x^3 + c_1$$

is independent of c_2. Poisson's equation with Neumann boundary conditions can therefore have none or many solutions $u(x)$.

I will focus on homogeneous Dirichlet boundary conditions for most of this module.

Finite Differences

Discussion

Poisson's equation $-\Delta u = f$ is the only partial differential equation I will consider in this module.

Reasons for focussing on only a single PDE include:

- Poisson's equation allows me to demonstrate many of the numerical techniques for solving PDEs at a simple, concrete and reasonably intuitive example.
- Many practically relevant PDEs can be summarised as "Poisson + some extra complications".

Poisson's equation will be important both in this and the following lectures. In the following slides, I will therefore provide some intuition for the physics described by this equation by demonstrating how it arises in a simple model of a real-world system.

Finite Differences

Derivation of Poisson's equation (not examinable)

Consider the following model.

- $\Omega \subset \mathbb{R}^2$ describes an empty sandpit containing a colony of ants.
- $u : \Omega \to \mathbb{R}$ describes the *concentration* of ants.
- $f : \Omega \to \mathbb{R}$ describes the net flow of ants entering or leaving their nest, i.e. if $f(x) = 1$, then every second, we have one more ant leaving the nest at point $x \in \Omega$ than entering.

  I will refer to $f(x)$ as a *source term*, since from the point of view of an outside observer it describes the rate at which ants appear or disappear at any given location $x \in \Omega$.
- $J : \Omega \to \mathbb{R}^2$ describes the *net flow* of ants, i.e. if $J(x) = (1, 0)$, then every second, one more ant crosses the point $x \in \Omega$ from left to right than from right to left, and the number of ants crossing from top to bottom equals the number of ants crossing from bottom to top.

`https://www.shutterstock.com/video/clip-9873545-many-ants-on-sand`

Finite Differences

Derivation of Poisson's equation (not examinable, continued)

Let us assume that the number of ants in a domain $\Omega' \subset \Omega$ can only change if ants either enter or leave the nest within Ω, or if ants move into or out of Ω.

In mathematical terms, this means that we must have

$$\underbrace{\frac{\partial}{\partial t} \int_{\Omega'} u \, dx}_{\#\text{ ants in } \Omega'} = \underbrace{- \int_{\partial\Omega'} n \cdot J \, dx}_{\#\text{ ants crossing } \partial\Omega'} + \underbrace{\int_{\Omega'} f \, dx}_{\substack{\#\text{ ants entering or} \\ \text{leaving the nest within } \Omega'}} .$$

$n = n(x)$ denotes the exterior normal vector at $x \in \partial\Omega'$.

This relationship between concentration $u(x)$, flow $J(x)$ and source term $f(x)$ is called *conservation of mass*.

Finite Differences

Derivation of Poisson's equation (not examinable, continued)

Conservation of mass equation copied from above:

$$\frac{\partial}{\partial t}\int_{\Omega'} u\,dx = -\int_{\partial\Omega'} n\cdot J\,dx\ + \int_{\Omega'} f\,dx.$$

Applying the divergence law

$$\int_{\partial\Omega'} n\cdot J\,dx = \int_{\Omega'} \nabla\cdot J\,dx$$

and moving all terms to the left, we then obtain

$$\int_{\Omega'}\left(\tfrac{\partial u}{\partial t} + \nabla\cdot J - f\right)dx = 0.$$

The next slide demonstrates how to translate this statement into a PDE.

Finite Differences

Theorem (not examinable)

Assume we have for all $\Omega' \subset \Omega$ that

$$\int_{\Omega'} \left(\frac{\partial u}{\partial t} + \nabla \cdot J - f \right) dx = 0. \tag{1}$$

Then we have for all $x \in \Omega \setminus \Omega_0$ that

$$\frac{\partial u}{\partial t}(x) + \nabla \cdot J(x) - f(x) = 0 \tag{2}$$

where Ω_0 denotes some unspecified set of measure 0.

Proof. Choose $\Omega' \subset \Omega$ to be the set of all x such that

$$\frac{\partial u}{\partial t}(x) + \nabla \cdot J(x) - f(x) > 0.$$

If this set is of positive measure, then

$$\int_{\Omega'} \left(\frac{\partial u}{\partial t} + \nabla \cdot J - f \right) dx > 0$$

which contradicts (2). Repeating this argument with $\Omega' \subset \Omega$ the set of all x such that the left-hand side of (2) is negative completes the proof.

Finite Differences

Terminology: Almost all and almost everywhere

A PDE like

$$\frac{\partial u}{\partial t}(x) + \nabla \cdot J(x) - f(x) = 0$$

is said to be satisfied for *almost all* $x \in \Omega$ or *almost everywhere on* Ω if it is satisfied for all $x \in \Omega \setminus \Omega_0$ where $\Omega_0 \subset \Omega$ denotes some unspecified set of measure 0.

Remark

The theorem on the previous slide indicates that it would be natural to require that a PDE like

$$\frac{\partial u}{\partial t}(x) + \nabla \cdot J(x) - f(x) = 0$$

holds only for almost all $x \in \Omega$, and there are also several mathematical arguments leading to the same conclusion.

This idea can be turned into a rigorous mathematical theory (Lebesgue and Sobolev spaces) but leads to many technical complications which are well beyond the scope of this module.

Finite Differences

Remark (continued)

Instead, I will assume that PDEs are satisfied for all $x \in \Omega$ and not almost all. This greatly simplifies the theory but comes at the price that there will be some phenomena which cannot be explained with the mathematical tools used in this module.

Derivation of Poisson's equation (not examinable, continued)

Let us now continue with the derivation of Poisson's equation in the context of our "ants in sandpit" model.

We have seen that in this module, the ant concentration $u(x) \in \mathbb{R}$, the ant entry and exits rate $f(x) \in \mathbb{R}$ and the and flow $J(x) \in \mathbb{R}^2$ are related by

$$\frac{\partial u}{\partial t} + \nabla \cdot J - f = 0.$$

To turn this equation into Poisson's equation, we need to

- assume we are in a steady state, i.e. $\frac{\partial u}{\partial t} = 0$, and
- provide an expression for $J(x)$.

The second item is tackled on the next slide.

Finite Differences

Derivation of Poisson's equation (not examinable, continued)

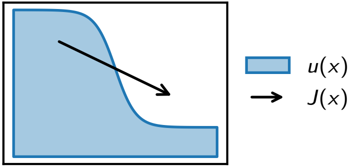
Let us assume that the ants try to spread out as much as possible, i.e. if $u(x)$ is large at some point x_1 but low at a nearby point x_2, then the ants will move from x_1 to x_2.

One way to formalise this assumption is to assume that the ant flow $J(x)$ is given by

$$J(x) = -D\,\nabla u(x) \qquad \text{for some } D > 0.$$

This assumption is known as *Fick's law*.

Fick's law occurs frequently in physics because it arises in particular if the ants move around randomly. In this context, the proportionality constant D is called the *diffusion coefficient*.



Finite Differences

Derivation of Poisson's equation (not examinable, continued)

Inserting the steady state assumption $\frac{\partial u}{\partial t} = 0$ and Fick's law $J(x) = -D\,\nabla u(x)$ into our model equation

$$\frac{\partial u}{\partial t} + \nabla \cdot J - f = 0$$

yields

$$-\nabla \cdot \nabla u = -\Delta u = f,$$

which is Poisson's equation.

Finite Differences

Interpretation of boundary conditions

In the context of our "ants in sandpit" model, the boundary conditions introduced on slide 4 can be interpreted as follows.

- Homogeneous Dirichlet boundary conditions, $u(\partial\Omega) = 0$.

  No ants at the boundary. One way to achieve this would be to paint the sandpit walls with glue such that any ant touching the wall gets stuck and no longer counts as a freely moving ant.

  Note that in the context of Fick's law, $u(x)$ must describe the concentration of freely moving ants since $u(x)$ is related to the ant flow $J(x)$ through $J(x) = -\nabla u(x)$.

- Homogeneous Neumann boundary conditions, $\frac{\partial u}{\partial n}(\partial\Omega) = 0$.

  No net ant flow $J(x) = -\nabla u$ across the boundary. This condition is satisfied if every ant hitting the boundary turns around and continues its random walk.

Finite Differences

Physical meaning of Poisson's equation

The above "ants in sandpit" model is obviously somewhat silly, but I believe it is the clearest way to illustrate how Poisson's equation arises from Fick's law and conservation of mass.

Conservation of mass and Fick's law are ubiquitous in physics and hence so is Poisson's equation. Here are some examples of real-world phenomena described by PDEs which are equal / very similar to Poisson's equation.

- Conductive heat transfer: $\frac{\partial T}{\partial t} = \Delta T + f$ where T denotes temperature and f denotes heat sources and sinks.
- Electrostatics / gravity: $-\Delta \phi = \rho$ where ϕ denotes the electric / gravitational potential and ρ denotes the charge / mass density.
- Fluid dynamics (Navier-Stokes equations): $\frac{\partial u}{\partial t} = \nu \, \Delta u - u \cdot \nabla u$ where u denotes the flow velocity and ν denotes the viscosity.
- Quantum mechanics (Schrödinger equation): $i \frac{\partial \psi}{\partial t} = -\Delta \psi + V \, \psi$ where ψ denotes the wave function and V the potential energy.

Finite Differences

Discussion

Our main goal in this lecture is to develop numerical methods for evaluating the map

$$\big(f : \Omega \to \mathbb{R}\big) \quad \mapsto \quad \big(u : \Omega \to \mathbb{R} \text{ such that } -\Delta u = f\big).$$

To do so, we must address the following question:

How do we represent an arbitrary function $f : \Omega \to \mathbb{R}$ on a computer?

The following slides will list several options and discuss their respective strengths and weaknesses.

Finite Differences

Representing functions

Option 1: Function handles

Any reasonable programming language allows you to treat functions like any other piece of data, so you could write a function

```
solve_poisson(f) -> u
```

which takes in a function `f(x) -> Float64` and returns another function `u(x) -> Float64`.

Good: Allows for exact representation up to floating-point rounding.

Bad: Unclear how we would determine the correct output $u(x)$.

In particular, we should at the very least be able to verify that a proposed solution $u(x)$ indeed satisfies $-\Delta u(x) = f(x)$, but checking this property for every floating-point input $x \in \Omega \cap \texttt{Float64}$ is clearly not feasible and if we test less then $u(x)$ is underspecified.

This leaves us to conclude that while having complete freedom to choose $u(x)$ may seem tempting at first, it is actually not a very good choice in the long run.

Finite Differences

Representing functions (continued)

Option 2: Polynomials

If $f(x)$ is a polynomial $f \in \mathcal{P}_n$, then we can easily compute and represent the exact solution $u \in \mathcal{P}_{n+2}$. Polynomial approximation allows us to turn any function into a polynomial up to a controllable approximation error, so we can approximately solve $-\Delta u(x) = f(x)$ by combining these two ideas.

This idea has been explored in the mathematical literature, see e.g.

`https://github.com/JuliaApproximation/ApproxFun.jl`,
`https://github.com/JuliaApproximation/SingularIntegralEquations.jl`,
`http://www.chebfun.org/examples/pde/Erosion.html`.

but it is not used very often in the applied sciences and industry.

Finite Differences

Representing functions (continued)

Option 2: Polynomials (continued)

Possible reasons why global polynomial approximation is rarely used in real-world applications include:

- Polynomials are tricky. For example, we have seen in Lecture 4 that polynomial interpolation may both converge or divergence depending on how we choose the interpolation points.
- Polynomials couple things which should not be coupled.

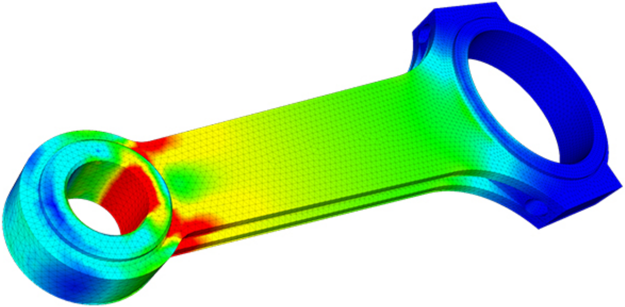
  Many real-world $f(x)$ and $u(x)$ are "simple" (e.g. analytic) in most of Ω but "complicated" (e.g. highly oscillating, not differentiable) in very narrow regions. Polynomials do not allow to separate such regions.

Finite Differences

Representing functions (continued)

Option 3: Piecewise polynomials

Using piecewise polynomials for solving partial differential equations is known as the finite element method (FEM) and the current state of the art for solving complicated real-world PDEs.



https://www.simscale.com/blog/2016/10/what-is-finite-element-method/

Finite Differences

Representing functions (continued)

Option 3: Piecewise polynomials (continued)

The challenge in using piecewise polynomial for solving partial differential equations is to make sense of the derivatives at interval boundaries where the approximate solution $u \in P_m\mathcal{P}_n$ may fail to be differentiable.

Addressing this challenge requires the notion of derivatives defined almost everywhere discussed on slide 11; hence I will not discuss the finite element method in this lecture.

Instead, let me just mention that the finite element method turns the partial differential equation $-\Delta u = f$ into a large and sparse linear system $Ax = b$ and therefore leads to similar computational problems as the finite difference method which I will discuss in this module.

Finite Differences

Representing functions (continued)

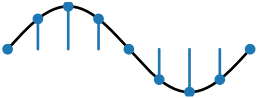
Option 4: Point values

The problem with the function handle approach from slide 18 may be further illustrated as follows.

Consider $\Omega = (1, 2)$. This interval contains $2^{53} - 1$ `Float64` numbers; hence we need $2^{53} - 1$ equations to determine all values of the function $\texttt{u} : (1, 2) \cap \texttt{Float64} \to \texttt{Float64}$, which is clearly not feasible.

Once formulated this way, a possible solution to the problem presents itself: instead of sampling the solution $u(x)$ at the resolution of floating-point numbers, we can artificially restrict the number of samples to some $n \ll 2^{53}$ so we end up with only a manageable number of unknown point values.

$$u[i] = u\left(\tfrac{i}{n+1}\right) \qquad \longleftrightarrow$$



This approach is known as *finite difference method* and the main topic of this lecture.

Finite Differences

Functions and vectors of point values

The simple formula $u[i] = u\big(\frac{i}{n+1}\big)$ requires some comments.

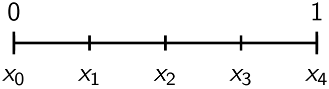
- I use the same symbol u to denote both a vector of point values $u[i]$ and a function $u(x)$. Context will clarify which representation is meant.
- The rationale for associating entry $u[i]$ with the value of $u(x)$ at position $x = \frac{i}{n+1}$ is as follows.
  - For concreteness, I assume $\Omega = [0, 1]$. Extension to arbitrary intervals is straightforward.
  - For simplicity, I want the locations x_i of the point values $u[i]$ to be equispaced over $[0, 1]$. Generalisation to an arbitrary collection of points x_i is not difficult but significantly complicates the notation.
  - I want to enumerate the unknowns $u[i]$ using the index range $i \in \{1, \dots, n\}$ for consistency with Julia.
  - (continued on next slide).

Finite Differences

Functions and vectors of point values

- Rationale for associating entry $u[i]$ with the value $u(\frac{i}{n+1})$:
  (Continued from above.)
  - For concreteness and simplicity, I assume homogeneous Dirichlet boundary conditions; thus the values of $u(x)$ at locations $x = 0$ and $x = 1$ are known and should fall just outside the range of unknown indices $i \in \{1, \ldots, n\}$.

  Combining all of the above yields the formula $u[i] = u\left(\frac{i}{n+1}\right)$.



Note that $u[i] = u\left(\frac{i}{n+1}\right)$ is specific to Dirichlet boundary conditions. If instead we assumed Neumann boundary conditions, then $u(0)$ and $u(1)$ would also be unknown and the formula for relating entries to point values would be $u[i] = u\left(\frac{i-1}{n-1}\right)$.

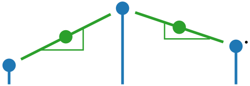
Finite Differences

Finite differences

Representing the solution $u : \Omega \to \mathbb{R}$ as a vector of point values $u[i]$ rather than a function handle `u(x) -> Float64` solves the problem of having too much freedom for choosing $u(x)$, but it still leaves some freedom and hence we still at the very least need a criterion for checking whether a trial solution satisfies $-\Delta u = f$.

In the point values approach, a trial solution u is a vector and not a function; hence taking derivatives of tentative solutions does not make sense. Fortunately, it is not hard to come up with an alternative, namely we can set

$$u'\left(\tfrac{i+1/2}{n+1}\right) \approx \frac{u\left(\frac{i+1}{n+1}\right) - u\left(\frac{i}{n+1}\right)}{1/(n+1)} \qquad \longleftrightarrow$$



We then obtain for the second derivative

$$\begin{aligned} u''\left(\tfrac{i}{n+1}\right) &\approx \frac{u'\left(\frac{i+1/2}{n+1}\right) - u'\left(\frac{i-1/2}{n+1}\right)}{1/(n+1)} \\ &= (n+1)^2 \left(u\left(\tfrac{i+1}{n+1}\right) - 2\,u\left(\tfrac{i}{n+1}\right) + u\left(\tfrac{i-1}{n+1}\right) \right). \end{aligned}$$

Finite Differences

Finite differences (continued)

Replacing point values with vector entries, the above formula becomes

$$u''[i] \approx (n+1)^2 \big(u[i-1] - 2\,u[i] + u[i+1]\big).$$

This formula corresponds to a matrix-vector product $u'' \approx \Delta_n u$ where the matrix $\Delta_n \in \mathbb{R}^{n \times n}$ is given by

$$\Delta_n = (n+1)^2 \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{pmatrix}.$$

Empty entries denote 0.

Terminology: Discrete Laplacian

I will call the matrix Δ_n defined above the *discrete Laplacian* or *Laplace matrix* (as opposed to continuous Laplacian / Laplace operator).

Finite Differences

Remark

Discrete Laplacian copied from above:

$$\Delta_n = (n+1)^2 \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{pmatrix}.$$

Note how the left / blue 1 is missing in the first row of Δ_n. This is because the first row corresponds to the equation

$$u''[1] \approx (n+1)^2 \big(u[0] - 2\,u[1] + u[2] \big)$$

and the homogeneous Dirichlet boundary conditions specify that

$$u[0] = u(0) = 0.$$

The right / green 1 on the last row is missing for the same reason.

Finite Differences

Finite differences (continued)

Our starting point for the derivation of the discrete Laplacian Δ_n was that we wanted a criterion for determining whether a trial solution $u \in \mathbb{R}^n$ satisfies Poisson's equation $-\Delta u = f$.

Based on the above, we conclude that $-\Delta_n u = f$ could be a reasonable choice for such a criterion.

Note how in this equation, f must refer to the vector of point values $f[i] = f\left(\frac{i}{n+1}\right)$ rather than the function $f : [0,1] \to \mathbb{R}$ since otherwise the comparison with the left-hand side does not make sense.

Terminology: Finite difference discretisation

Replacing $-\Delta u = f$ with $-\Delta_n u = f$ is known as the *finite difference discretisation* of Poisson's equation.

Implementation

Finite difference discretisation turns the partial differential equation $-\Delta u = f$ into a linear system $-\Delta_n u = f$ which can be solved using Julia's backslash function \ (or Matlab's \, or Python's `numpy.linalg.solve()`).

Implementation of this scheme is straightforward, see `solve_poisson_1d()` and `example_1d()`.

Finite Differences

Discussion

We now have a concrete algorithm for evaluating the map

$$f \in \mathbb{R}^n \quad \mapsto \quad u \in \mathbb{R}^n.$$

Like most algorithms in computational mathematics, this algorithm is based on heuristics rather than rigorous mathematical analysis, and `example()` shows that it is not correct in the sense that

$$\big(-\Delta_n^{-1} f\big)[i] \quad \neq \quad u\big(\tfrac{i}{n+1}\big).$$

This raises the question why $f \mapsto -\Delta_n^{-1} f$ is a good algorithm for solving the Poisson equation, and the answer is of course that we can show

$$-\Delta_n^{-1} f \;\to\; u \qquad \text{for } n \to \infty.$$

I will next explain how we can prove a result like the above, and as usual the starting point for doing so is to establish a metric for measuring errors.

Finite Differences

Def: Weighted 2-norm

Given $u \in \mathbb{R}^n$, we define

$$\|u\|_{2,n} = \frac{1}{\sqrt{n+1}} \sqrt{\sum_{i=1}^{n} u[i]^2} = \frac{\|u\|_2}{\sqrt{n+1}}.$$

Discussion

The purpose of the $1/\sqrt{n+1}$ prefactor is to ensure that if $u : [0,1] \to \mathbb{R}$ is a bounded function, then $\lim_{n\to\infty} \|u\|_{2,n} < \infty$.

In fact, it is easy to see that $\|u\|_{2,n}$ is the trapezoidal-rule approximation of

$$\|u\|_{L^2([0,1])} = \sqrt{\int_0^1 u(x)^2\,dx}$$

and hence $\lim_{n\to\infty} \|u\|_{2,n} = \|u\|_{L^2([0,1])} < \infty$.

For later reference, I would further like to point out that for all $A \in \mathbb{R}^{n\times n}$, we have $\|A\|_{2,n} = \|A\|_2$ since

$$\|A\|_{2,n} = \sup_{u\in\mathbb{R}^n} \frac{\|Au\|_{2,n}}{\|u\|_{2,n}} = \sup_{u\in\mathbb{R}^n} \frac{\frac{1}{\sqrt{n+1}}\,\|Au\|_2}{\frac{1}{\sqrt{n+1}}\,\|u\|_2} = \sup_{u\in\mathbb{R}^n} \frac{\|Au\|_2}{\|u\|_2} = \|A\|_2.$$

Finite Differences

Our convergence analysis for the finite difference discretisation will be based on the following result.

Stability and consistency lemma

Let $u : [0, 1] \to \mathbb{R}$ be the solution to $-\Delta u = f$ with homogeneous Dirichlet boundary conditions, and let $u_n \in \mathbb{R}^n$ be the solution to $-\Delta_n u_n = f$. Then,

$$\|u - u_n\|_{2,n} \leq \|\Delta_n^{-1}\|_{2,n} \, \|\Delta_n u + f\|_{2,n}.$$

Note how the meaning of u and f changes depending on the context.

Also, note how I use different symbols to refer to the solutions of $-\Delta u = f$ and $-\Delta_n u_n = f$ since u and u_n are generally not the same even if we interpret both of them as vectors of point values. In fact, $u \neq u_n$ is the main reason why we are interested in this result.

Proof.

$$\begin{aligned} \|u - u_n\|_{2,n} &= \|\Delta_n^{-1} \left(\Delta_n u - \Delta_n u_n\right)\|_{2,n} \\ &= \|\Delta_n^{-1} \left(\Delta_n u + f\right)\|_{2,n} \\ &\leq \|\Delta_n^{-1}\|_{2,n} \, \|\Delta_n u + f\|_{2,n} \end{aligned}$$

where on the last line I used that $\|Ax\| \leq \|A\| \, \|x\|$, which is an immediate consequence of the definition of induced matrix norms.

Finite Differences

Discussion

The result on the previous slide decomposes the error $\|u - u_n\|_{2,n}$ into the two factors $\|\Delta_n^{-1}\|_{2,n}$ and $\|\Delta_n u + f\|_{2,n}$.

These factors have reasonably intuitive interpretations.

Stability factor $\|\Delta_n^{-1}\|_{2,n}$

Assume u_n and v_n solve the discretised Poisson equations

$$-\Delta_n u_n = f \qquad \text{and} \qquad -\Delta_n v_n = g$$

for two different right-hand sides f and g, respectively.
The difference between u_n and v_n is then upper bounded by

$$\|u_n - v_n\|_{2,n} \leq \|\Delta_n^{-1}\|_{2,n} \; \|f - g\|_{2,n};$$

thus $\|\Delta_n^{(-1)}\|_{2,n}$ provides an upper bound on how much the solution u_n can change for a given change in the right-hand side f.

For this reason, $\|\Delta_n^{-1}\|_{2,n}$ is called the *stability factor* of Δ_n.

This notion of stability is different from the notion of stability introduced in Lecture 3 and the two should not be confused!

Finite Differences

Discussion (continued)

Consistency error $\|\Delta_n u + f\|_{2,n}$

$\|\Delta_n u + f\|_{2,n}$ measures how well the solution u to the continuous problem satisfies the equation $-\Delta_n u_n = f$ which specifies the discrete solution u_n. It is therefore called the *consistency error* of the discretised equation.

We will next estimate the stability and consistency factors and thereby prove convergence of the finite difference discretisation.

Stability of Δ_n

Δ_n is a symmetric matrix; hence we have

$$\|\Delta_n^{-1}\|_{2,n} = \|\Delta_n^{-1}\|_2 = \lambda_{\text{min}}^{-1}$$

where λ_{min} denotes the eigenvalue of smallest absolute value of Δ_n.

The second of the above equalities is a well known result from linear algebra which you should know from your undergraduate studies.

We can hence estimate the stability factor $\|\Delta_n^{-1}\|_{2,n}$ by determining the eigenvalues of Δ_n.

To do so, it is useful to first gather some inspiration by determining the eigenvectors in the continuous case.

Finite Differences

Stability of Δ_n (continued)

Straightforward calculations reveal that

$$\Big(\quad \lambda_k = -\pi^2 \, k^2, \quad u_k(x) = \sin(\pi k \, x) \quad \Big)_{k=1}^{\infty}$$

are eigenpairs of $\Delta\big|_{[0,1]}$ combined with homogeneous Dirichlet boundary conditions, i.e. these pairs (λ_k, u_k) satisfy

$$\Delta u_k(x) = u_k''(x) = \lambda_k \, u_k(x) \qquad \text{and} \qquad u_k(0) = u_k(1) = 0.$$

We may therefore hope that

$$u_k[i] = \sin\!\big(\pi k \tfrac{i}{n+1}\big)$$

would be eigenvectors of Δ_n, and we can check that this is indeed the case using explicit calculations, see next slide.

Note that I am using u_n to denote the solution to $-\Delta_n u_n = f$ and u_k to denote the eigenvectors of Δ_n. Since both n and k are just integers, this notation is strictly speaking ambiguous, so I will try my best to always point out whether u_n denotes the solution to the discrete problem or the eigenvectors.

Finite Differences

Stability of Δ_n (continued)

$$\begin{aligned}
\big(\Delta_n u_k\big)[j] &= (n+1)^2\left(u_l[j-1] - 2u_k[j] + u_k[j+1]\right)\\
&= (n+1)^2\left(\sin\!\left(\pi k\tfrac{j-1}{n+1}\right) - 2\sin\!\left(\pi k\tfrac{j}{n+1}\right) + \sin\!\left(\pi k\tfrac{j+1}{n+1}\right)\right)\\
&= \frac{(n+1)^2}{2i}\Bigg(e^{\pi ik\frac{j-1}{n+1}} - e^{-\pi ik\frac{j-1}{n+1}} - 2e^{\pi ik\frac{j}{n+1}} + 2e^{-\pi ik\frac{j}{n+1}} + \dots\\
&\qquad\qquad e^{\pi ik\frac{j+1}{n+1}} - e^{-\pi ik\frac{j+1}{n+1}}\Bigg)\\
&= \frac{(n+1)^2}{2i}\Bigg(e^{\pi ik\frac{j}{n+1}}\left(e^{-\pi ik\frac{1}{n+1}} - 2 + e^{\pi ik\frac{1}{n+1}}\right) - \dots\\
&\qquad\qquad e^{-\pi ik\frac{j}{n+1}}\left(e^{\pi ik\frac{1}{n+1}} - 2 + e^{-\pi ik\frac{1}{n+1}}\right)\Bigg)\\
&= \frac{(n+1)^2}{2i}\left(e^{-\pi ik\frac{1}{n+1}} - 2 + e^{\pi ik\frac{1}{n+1}}\right)\left(e^{\pi ik\frac{j}{n+1}} - e^{-\pi ik\frac{j}{n+1}}\right)\\
&= (n+1)^2\left(2\cos\!\left(\pi\tfrac{k}{n+1}\right) - 2\right)\sin\!\left(\pi k\tfrac{j}{n+1}\right)\\
&= (n+1)^2\left(2\cos\!\left(\pi\tfrac{k}{n+1}\right) - 2\right)u_k[j]
\end{aligned}$$

We conclude that $u_k[i] = \sin\!\left(\pi k\frac{i}{n+1}\right)$ are indeed eigenvectors of Δ_n, and the associated eigenvalues are as presented on the next slide.

Finite Differences

Thm: Eigenpairs of Δ_n

The n eigenpairs of Δ_n are given by

$$\left(\quad \lambda_k = (n+1)^2 \left(2\cos\left(\pi \tfrac{k}{n+1}\right) - 2\right), \qquad u_k[i] = \sin\left(\pi k \tfrac{i}{n+1}\right) \quad \right)_{k=1}^{n}$$

Proof. We have already seen on the previous slide that $\Delta_n u_k = \lambda_k u_k$; hence it remains to show that $(u_k)_{k=1}^n$ are linearly independent.

According to basic results from linear algebra, we can do so by checking that $(\lambda_k)_{k=1}^n$ are distinct and $(u_k \neq 0)_{k=1}^n$. The first of these properties follows from the observation that $\left(\tfrac{k}{n+1}\right)_{k=1}^n$ are n distinct points in $(0,1)$, and the second property is obvious.

Finite Differences

Stability of Δ_n (continued)

Recall that our motivation for determining the eigenpairs of Δ_n was to estimate $\|\Delta_n^{-1}\|_{2,n}$.

We therefore observe that

$$\left(\quad \lambda_k = (n+1)^2 \left(2\cos\left(\pi\tfrac{k}{n+1}\right) - 2\right) \quad \right)_{k=1}^{n}$$

is of smallest absolute value for $k = 1$, and using the Taylor series expansion

$$\cos(x) = 1 - \tfrac{x^2}{2!} + O(x^4)$$

we obtain

$$\lambda_1 = (n+1)^2 \left(2 - \tfrac{\pi^2}{(n+1)^2} + O(n^{-4}) - 2\right) = -\pi^2 + O(n^{-2}).$$

We hence conclude:

Lemma: Stability of Δ_n

$$\|\Delta_n^{-1}\|_{2,n} = \pi^{-2} + O(n^{-2}) = O(1) \qquad \text{for } n \to \infty.$$

Finite Differences

Consistency of $-\Delta_n u = f$

Let us next derive an estimate for the consistency error $\|\Delta_n u + f\|_{2,n}$. According to Taylor's theorem, we have

$$\begin{aligned} u\Big(\tfrac{i\pm 1}{n+1}\Big) = u\Big(\tfrac{i}{n+1}\Big) \ &\pm\ u'\Big(\tfrac{i}{n+1}\Big)\,\tfrac{1}{n+1} \ +\ \tfrac{1}{2!}\,u''\Big(\tfrac{i}{n+1}\Big)\,\tfrac{1}{(n+1)^2} \ +\ \ldots \\ &\pm\ \tfrac{1}{3!}\,u'''\Big(\tfrac{i}{n+1}\Big)\,\tfrac{1}{(n+1)^3} \ +\ O\big(n^{-4}\big) \end{aligned}$$

and hence

$$\begin{aligned} (\Delta_n u)[i] &= (n+1)^2\left(u\big(\tfrac{i-1}{n+1}\big) - 2\,u\big(\tfrac{i}{n+1}\big) + u\big(\tfrac{i+1}{n+1}\big)\right) \\ &= (n+1)^2\Bigg((1-2+1)\,u\Big(\tfrac{i}{n+1}\Big) \ +\ \ldots \\ &\qquad\qquad (-1+1)\,u'\Big(\tfrac{i}{n+1}\Big)\tfrac{1}{n+1} \ +\ \ldots \\ &\qquad\qquad\quad (1+1)\,\tfrac{1}{2!}\,u''\big(\tfrac{i}{n+1}\big)\,\tfrac{1}{(n+1)^2} \ +\ \ldots \\ &\qquad\qquad\qquad (-1+1)\,\tfrac{1}{3!}\,u'''\big(\tfrac{i}{n+1}\big)\,\tfrac{1}{(n+1)^3} \ +\ O\big(n^{-4}\big)\Bigg) \\ &= u''\big(\tfrac{i}{n+1}\big) + O\big(n^{-2}\big). \end{aligned}$$

Finite Differences

Consistency of $-\Delta_n u = f$ (continued)

Copied from above:

$$(\Delta_n u)[i] = u''(\tfrac{i}{n+1}) + O\big(n^{-2}\big).$$

Since $u(x)$ solves the continuous Poisson's equation $-\Delta u = f$, we have

$$u''(x) = -f(x)$$

and thus

$$\|\Delta_n u + f\|_{2,n} = \|\Delta_n u - u''\|_{2,n} = \sqrt{\tfrac{n}{n+1}\, O\big(n^{-2}\big)^2} = O(n^2).$$

We therefore conclude:

Lemma: Consistency of $-\Delta_n u = f$

Assume u is a four times differentiable solution to $-\Delta u = f$. Then,

$$\|\Delta_n u + f\|_{2,n} = O(n^{-2}) \qquad \text{for } n \to \infty.$$

Finite Differences

Corollary: Convergence of finite difference discretisation

Let $u : [0,1] \to \mathbb{R}$ be the solution to $-\Delta u = f$ with homogeneous Dirichlet boundary conditions, and let $u_n \in \mathbb{R}^n$ be the solution to $-\Delta_n u_n = f$. Then,

$$\|u - u_n\|_{2,n} = O(n^{-2}) \qquad \text{for } n \to \infty.$$

Proof. Combining the stability and consistency lemma from slide 32 with the stability estimate from slide 38 and the consistency estimate from slide 40, we obtain

$$\|u - u_n\|_{n,2} \leq \|\Delta_n^{-1}\|_{2,n} \, \|\Delta_n u + f\|_{2,n} = O(1) \, O\big(n^{-2}\big) = O\big(n^{-2}\big).$$

Numerical demonstration

See `convergence()`.

Finite Differences

Remark: Other stability proofs

Recall: we estimated the stability factor $\|\Delta_n^{-1}\|_{2,n}$ by

1. guessing that $u_k[i] = \sin\!\left(\frac{i}{n+1}\right)$ should be the eigenvectors, and then
2. verifying our guess through explicit calculations.

This approach is known as *Fourier analysis* because it is closely related to the idea of Fourier series which you may have encountered in other modules.

Fourier analysis is not the only technique for proving stability. Other approaches include

- the energy / integration by parts method, and
- the maximum principle.

Unlike Fourier analysis, these techniques estimate $\|\Delta_n^{-1}\|$ directly rather than indirectly via the eigenvalues. This makes these techniques somewhat more versatile, i.e. they may allow us to prove stability under circumstances where Fourier analysis no longer applies, but it also means that these techniques provide us with less information.

The choice of which stability proof to pursue therefore depends both on which techniques are applicable and also which techniques allow us to deduce all the desired information.

Finite Differences

Remark: Other stability proofs (continued)

The main difference between the energy method and the maximum principle is that the energy method estimates $\|\Delta_n^{-1}\|_{2,n}$ (like the Fourier analysis) while the maximum principle estimates $\|\Delta_n^{-1}\|_\infty$.

I presented the Fourier analysis technique in this lecture because knowing the eigenvectors and -values will be important in the lectures on Krylov and multigrid methods.

I will not discuss the energy method and maximum principle, but I believe it is nevertheless worth mentioning them for two reasons.

- If you ever need alternatives to Fourier analysis, you now know what keywords to search for.
- If you encounter the energy method or maximum principle elsewhere, you now know how they relate to the theory presented here.

Finite Differences

Discussion

We now know how to numerically solve Poisson's equation in one dimension. Unfortunately, we live in a three-dimensional world; hence most PDEs of practical interest are posed on domains $\Omega \subset \mathbb{R}^3$.

Generalising finite differences from $d = 1$ to $d = 3$ dimensions introduces nontrivial difficulties, but it turns out that all of these difficulties arise already for $d = 2$ dimensions, and working in two dimensions is more convenient because it is easier to visualise and less computationally demanding.

I will therefore next demonstrate how to generalise finite differences to two-dimensional problems, and after that I will briefly outline how we can go from two to three dimensions.

Finite Differences

Two-dimensional finite differences

Finite differences in 2d replaces functions $u : [0,1]^2 \to \mathbb{R}$ with matrices of point values

$$\Big(u[i_1, i_2] = u\big(\tfrac{i_1}{n+1}, \tfrac{i_2}{n+1}\big)\Big)_{i_1,i_2=1}^{n} \;\in\; \mathbb{R}^{n\times n},$$

and since

$$\frac{\partial^2 u}{\partial x_1^2}\big(\tfrac{i_1}{n+1}, \tfrac{i_2}{n+1}\big) \approx (n+1)^2\,\big(u[i_1-1, i_2] - 2u[i_1, i_2] + u[i_1+1, i_2]\big),$$
$$\frac{\partial^2 u}{\partial x_2^2}\big(\tfrac{i_1}{n+1}, \tfrac{i_2}{n+1}\big) \approx (n+1)^2\,\big(u[i_1, i_2-1] - 2u[i_1, i_2] + u[i_1, i_2+1]\big),$$

it replaces the continuous Laplace operator

$$\Delta : \big([0,1]^2 \to \mathbb{R}\big) \to \big([0,1]^2 \to \mathbb{R}\big), \qquad \Delta u = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}$$

with the discrete Laplacian $\Delta_n^{(2)} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ given by

$$(\Delta_n^{(2)} u)[i_1, i_2] = (n+1)^2\,\big(u[i_1-1,i_2] + u[i_1,i_2-1] - 4u[i_1,i_2] + u[i_1+1,i_2] + u[i_1,i_2+1]\big).$$

From now on, I will distinguish the one-dimensional discrete Laplacian $\Delta_n^{(1)}$ and its two-dimensional counterpart $\Delta_n^{(2)}$ using a superscript (d) to avoid confusion.

Finite Differences

Two-dimensional finite differences (continued)

Two-dimensional finite differences thus amounts to the following problem.

$$\text{Determine } u_n \in \mathbb{R}^{n\times n} \text{ such that } -\Delta_n^{(2)}\, u_n = f.$$

$\Delta_n^{(2)}$ is a linear function; hence it must be possible to translate this problem into a linear system of the form $Ax = b$ which can be solved using Julia's backslash function.

To do so, we first transform the matrix of unknowns $u_n \in \mathbb{R}^{n\times n}$ into a vector of unknowns $\in \mathbb{R}^{n^2}$ using the vectorisation operation introduced on the next slide.

Finite Differences

Def: Vectorisation of a matrix

The *vectorisation* $\mathrm{vec}(u) \in \mathbb{R}^{mn}$ of a matrix $u \in \mathbb{R}^{m \times n}$ is given by

$$\mathrm{vec}(u)[i_1 + m\,(i_2 - 1)] = u[i_1, i_2].$$

Example

$\mathrm{vec}(u)$ enumerates the entries of a 4×4 matrix u as follows:

$1 = 1+4\times0$	$5 = 1+4\times1$	$9 = 1+4\times2$	$13 = 1+4\times3$
$2 = 2+4\times0$	$6 = 2+4\times1$	$10 = 2+4\times2$	$14 = 2+4\times3$
$3 = 3+4\times0$	$7 = 3+4\times1$	$11 = 3+4\times2$	$15 = 3+4\times3$
$4 = 4+4\times0$	$8 = 4+4\times1$	$12 = 4+4\times2$	$16 = 4+4\times3$

Finite Differences

Two-dimensional finite differences (continued)

Next, we translate the linear function

$$\Delta_n^{(2)} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$$

into a matrix

$$\mathrm{mat}\big(\Delta_n^{(2)}\big) \in \mathbb{R}^{n^2\times n^2}$$

such that

$$\mathrm{vec}\big(\Delta_n^{(2)}\, u_n\big) = \mathrm{mat}\big(\Delta_n^{(2)}\big)\, \mathrm{vec}(u_n).$$

This will be achieved using the Kronecker product $A \otimes B$ introduced on the next slide.

Finite Differences

Def: Kronecker product

The *Kronecker product* $A \otimes B \in \mathbb{R}^{(m_a m_b)\times(n_a n_b)}$ of two matrices

$$A \in \mathbb{R}^{m_A \times n_A}, \qquad B \in \mathbb{R}^{m_B \times n_B}$$

is given by

$$(A \otimes B)\big[{\scriptstyle i_B + m_B\,(i_A - 1),\; j_B + n_B\,(j_A - 1)}\big] = A[i_A, j_A]\, B[i_B, j_B],$$

or put differently,

$$A \otimes B = \begin{pmatrix} A[1,1]\,B & \cdots & A[1,n_A]\,B \\ \vdots & \ddots & \vdots \\ A[m_A,1]\,B & \cdots & A[m_A,n_A]\,B \end{pmatrix}.$$

Finite Differences

Lemma

Let $A \in \mathbb{R}^{m_A \times n_A}$, $B \in \mathbb{R}^{m_B \times n_B}$ and $C \in \mathbb{R}^{n_B \times n_A}$. Then,

$$\mathrm{vec}(BCA^T) = (A \otimes B)\,\mathrm{vec}(C).$$

Proof.

$$\begin{aligned}
&\mathrm{vec}(BCA^T)[_{i_B + m_B\,(i_A - 1)}] \\
&\qquad = \sum_{j_B=1}^{n_B} \sum_{j_A=1}^{n_A} B[i_B, j_B]\, C[j_B, j_A]\, A[i_A, j_A] \\
&\qquad = \sum_{j_B=1}^{n_B} \sum_{j_A=1}^{n_A} A[i_A, j_A]\, B[i_B, j_B]\, C[j_B, j_A] \\
&\qquad = \sum_{j_B=1}^{n_B} \sum_{j_A=1}^{n_A} (A \otimes B)[_{i_B + m_B\,(i_A - 1),\; j_B + n_B\,(j_A - 1)}]\, \mathrm{vec}(C)[_{j_B + m_B\,(j_A - 1)}] \\
&\qquad = \Big((A \otimes B)\,\mathrm{vec}(C)\Big)[_{i_B + m_B\,(i_A - 1)}].
\end{aligned}$$

Finite Differences

Corollary

We have

$$\operatorname{mat}\big(\Delta_n^{(2)}\big) = I_n \otimes \Delta_n^{(1)} + \Delta_n^{(1)} \otimes I_n,$$

i.e. the two-dimensional discrete Laplacian $\Delta_n^{(2)}$ from slide 45 satisfies

$$\operatorname{vec}\big(\Delta_n^{(2)}\, u_n\big) = \big(I_n \otimes \Delta_n^{(1)} + \Delta_n^{(1)} \otimes I_n\big) \operatorname{vec}(u_n).$$

$I_n \in \mathbb{R}^{n\times n}$ denotes the $n \times n$ identity matrix.

Proof. It follows from the formula on slide 45 that $\Delta_n^{(2)}$ can be expressed as

$$\Delta_n^{(2)}\, u_n = \Delta_n^{(1)}\, u_n + u_n\, \Delta_n^{(1)}.$$

The claim then follows immediately from the lemma on the previous slide.

Implementation of two-dimensional finite differences

See `solve_poisson_2d()` and `example_2d()`.

Ignore the `sparse()` in `sparse(laplacian_1d(n))` and assume `sparse(I,n,n)` allocates the $n \times n$ identity matrix for now.

Finite Differences

Example

The matrix

$$\operatorname{mat}\left(\Delta_4^{(2)}\right) = I_4 \otimes \Delta_4^{(1)} + \Delta_4^{(1)} \otimes I_4$$

is given by

$$(n+1)^2 \begin{pmatrix}
-4 & 1 & & & 1 & & & & & & & & & & & \\
1 & -4 & 1 & & & 1 & & & & & & & & & & \\
 & 1 & -4 & 1 & & & 1 & & & & & & & & & \\
 & & 1 & -4 & & & & 1 & & & & & & & & \\
1 & & & & -4 & 1 & & & 1 & & & & & & & \\
 & 1 & & & 1 & -4 & 1 & & & 1 & & & & & & \\
 & & 1 & & & 1 & -4 & 1 & & & 1 & & & & & \\
 & & & 1 & & & 1 & -4 & & & & 1 & & & & \\
 & & & & 1 & & & & -4 & 1 & & & 1 & & & \\
 & & & & & 1 & & & 1 & -4 & 1 & & & 1 & & \\
 & & & & & & 1 & & & 1 & -4 & 1 & & & 1 & \\
 & & & & & & & 1 & & & 1 & -4 & & & & 1 \\
 & & & & & & & & 1 & & & & -4 & 1 & & \\
 & & & & & & & & & 1 & & & 1 & -4 & 1 & \\
 & & & & & & & & & & 1 & & & 1 & -4 & 1 \\
 & & & & & & & & & & & 1 & & & 1 & -4
\end{pmatrix}.$$

Note that the colour scheme here is unrelated to the colour scheme from slide 27.

Observation

Only $O(n^2)$ out of the n^4 entries of $\Delta_n^{(2)}$ are nonzero!

Finite Differences

Terminology: Dense and sparse matrices

- A matrix with only few nonzero entries is called *sparse*.
- A matrix where most or all entries are nonzero is called *dense*.

Note that dense and sparse are relative rather than absolute terms.

There is no fixed rule to decide when a matrix is sparse enough to be called sparse, but a matrix like $\Delta_n^{(2)}$ with at most five nonzero entries per column independently of n is clearly sparse.

Finite Differences

Implementation of two-dimensional finite differences (continued)

Exploiting sparsity is crucial to ensure that the code runs on an average laptop: $\Delta_{300}^{(2)}$ has 300^4 entries; hence storing all entries would require

$$300^4\,\texttt{Float64} \times 8\,\frac{\text{bytes}}{\texttt{Float64}} \quad \approx \quad 65\,\text{gigabytes}$$

of memory, which is much larger than e.g. the 15 gigabytes of memory offered by my computer.

In contrast, if we were to store just the nonzero entries $A[i,j]$ and their indices i,j, then we would require at most

$$3 \times 5 \times 300^2\,[\texttt{Float64}\text{ or }\texttt{Int64}] \times 8\,\frac{\text{bytes}}{[\texttt{Float64}\text{ or }\texttt{Int64}]} \quad \approx \quad 11\,\text{megabytes},$$

where 3 is the number of variables $i, j, A[i,j]$ per nonzero entry, and 5 is the number of nonzero entries per column in $\Delta_n^{(2)}$.

This memory footprint is small enough to comfortably fit on any computing platform that I am aware of.

Finite Differences

The Compressed Sparse Columns (CSC) format

The purpose of the calls to `sparse()` in `laplacian_2d()` is to avoid the aforementioned memory problems by converting all matrices to a representation which exploits sparsity.

Julia (and also Matlab and Python) actually represents sparse matrices in a format which is slightly more efficient than the $(i, j, A[i, j])$ format suggested above.

This format is called *Compressed Sparse Columns (CSC)*.

I will not discuss this format here, but see

`https://en.wikipedia.org/wiki/Sparse_matrix#Compressed_sparse_row_(CSR,_CRS_or_Yale_format)`

if you would like to know more.

Part of the reason why Julia, Matlab and Python all agree on the representation of sparse matrices is because they are all based on the same underlying software library, namely SuiteSparse which is written mostly in C. See `https://people.engr.tamu.edu/davis/suitesparse.html`.

Finite Differences

Notation for two-dimensional finite differences

On slide 45, I introduced the two-dimensional discrete Laplacian as a function $\Delta_n^{(2)} : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$.

This conventions was mainly intended to help us get started with finite differences in two dimensions.

From now on, $\Delta_n^{(2)}$ will always denote the Laplacian matrix

$$\Delta_n^{(2)} = I_n \otimes \Delta_n^{(1)} + \Delta_n^{(1)} \otimes I_n \quad \in \quad \mathbb{R}^{n^2 \times n^2}.$$

The rationale for doing so is that we will be studying algorithms for solving the linear system $-\Delta_n^{(2)} u_n = f$ in the upcoming lectures, and for this purpose it is much more convenient to work with a matrix $\Delta_n^{(2)} : \mathbb{R}^{n^2 \times n^2}$ rather than a function $\Delta_n^{(2)} : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$.

On the other hand, u_n may denote either the matrix of point values $u_n \in \mathbb{R}^{n \times n}$ or its vectorisation $u_n \in \mathbb{R}^{n^2}$ depending on the context.

Finite Differences

Convergence of two-dimensional finite differences

It remains to show that $-\Delta_n^{(2)} u_n = f$ implies $\lim_{n \to \infty} u_n = u$.

As in 1d, the first step towards this goal is to introduce a metric for measuring errors.

Def: Weighted 2-norm for two-dimensional finite differences

Given $u \in \mathbb{R}^{n \times n}$, we define

$$\|u\|_{2,n} = \frac{1}{n+1} \sqrt{\sum_{i_1=1}^{n} \sum_{i_2=1}^{n} u[i_1, i_2]^2} = \frac{\|\mathrm{vec}(u)\|_2}{n+1}.$$

As before, the purpose of the $1/(n+1)$ prefactor is to ensure that if $u : [0,1]^2 \to \mathbb{R}$ is bounded, then $\|u\|_{2,n}$ is bounded for $n \to \infty$.

Moreover, $\|u\|_{2,n}$ can be interpreted as a trapezoidal-rule approximation to

$$\|u\|_{L^2([0,1]^2)} = \sqrt{\int_0^1 \int_0^1 u(x_1, x_2)\, dx_1\, dx_2},$$

and we have $\|A\|_{2,n} = \|A\|_2$.

Finite Differences

Convergence of two-dimensional finite differences (continued)

Next, I will employ the stability and consistency theorem

$$\|u - u_n\|_{2,n} \leq \left\|(\Delta_n^{(2)})^{-1}\right\|_{2,n} \left\|\Delta_n^{(2)}\, u + f\right\|_{2,n}$$

from slide 32 to show that $\lim_{n\to\infty} u_n = u$.

To do so, we must show that $\Delta_n^{(2)}$ is stable, i.e. $\|(\Delta_n^{(2)})^{-1}\|_{2,n} = O(1)$, and the consistency error $\|\Delta_n^{(2)}\, u + f\|_{2,n}$ vanishes.

As before, I will estimate the stability factor $\|(\Delta_n^{(2)})^{-1}\|_{2,n}$ by determining an explicit formula for the eigenvalues of $\Delta_n^{(2)}$ and then estimating the smallest eigenvalue.

This will be achieved using the result on the following slide.

Finite Differences

Lemma

Let $A \in \mathbb{R}^{m_A \times n_A}$, $B \in \mathbb{R}^{m_B \times n_B}$ and $a \in \mathbb{R}^{n_a}$, $b \in \mathbb{R}^{n_b}$. Then,

$$(A \otimes B)\,(a \otimes b) = (Aa) \otimes (Bb)$$

Proof.

$$\begin{aligned}
&\big((A \otimes B)\,(a \otimes b)\big)[i_B + m_B\,(i_A - 1)] \\
&\qquad = \sum_{j_A=1}^{n_a} \sum_{j_B=1}^{n_b} (A \otimes B)[i_B + m_B\,(i_A-1), j_B + n_B\,(j_A - 1)]\,(a \otimes b)[j_B + n_B\,(j_A - 1)] \\
&\qquad = \sum_{j_A=1}^{n_a} \sum_{j_B=1}^{n_b} A[i_A, j_A]\,B[i_B, j_B]\,a[j_A]\,b[j_B] \\
&\qquad = \left(\sum_{j_A=1}^{n_a} A[i_A, j_A]\,a[j_A]\right)\left(\sum_{j_B=1}^{n_b} B[i_B, j_B]\,b[j_B]\right) \\
&\qquad = \big(Aa\big)[i_A]\,\big(Bb\big)[i_B] \\
&\qquad = \big((Aa) \otimes (Bb)\big)[i_B + m_B\,(i_A - 1)].
\end{aligned}$$

Finite Differences

Thm: Eigenpairs of discrete two-dimensional Laplacian

Let $(\lambda_k, u_k)_{k=1}^n$ be the eigenpairs of $\Delta_n^{(1)}$.

Then, the eigenpairs of $\Delta_n^{(2)} = I_n \otimes \Delta_n^{(1)} + \Delta_n^{(1)} \otimes I_n$ are given by

$$\Big(\quad \lambda_{k_1,k_2} = \lambda_{k_1} + \lambda_{k_2}, \qquad u_{k_1,k_2} = u_{k_1} \otimes u_{k_2} \quad \Big)_{k_1,k_2=1}^n.$$

Proof.

$$\begin{aligned}
\Delta_n^{(2)} u_{k_1,k_2} &= \Big(I_n \otimes \Delta_n^{(1)} + \Delta_n^{(1)} \otimes I_n \Big) \big(u_{k_1} \otimes u_{k_2} \big) \\
&= u_{k_1} \otimes \big(\Delta_n^{(1)} u_{k_2} \big) + \big(\Delta_n^{(1)} u_{k_1} \big) \otimes u_{k_2} \\
&= \lambda_{k_2} \, u_{k_1} \otimes u_{k_2} + \lambda_{k_1} \, u_{k_1} \otimes u_{k_2} \\
&= \big(\lambda_{k_1} + \lambda_{k_2} \big) \, u_{k_1} \otimes u_{k_2} \\
&= \big(\lambda_{k_1} + \lambda_{k_2} \big) \, u_{k_1,k_2}.
\end{aligned}$$

Corollary: Stability of $\Delta_n^{(2)}$

$$\big\| (\Delta_n^{(2)})^{-1} \big\|_{2,n} = \tfrac{1}{2} \, \pi^{-2} + O(n^{-2}) = O(1) \qquad \text{for } n \to \infty.$$

Proof. We know from slide 38 that $\min_k |\lambda_k| = \lambda_1 = \pi^2 + O(n^{-2})$; hence $\min_{k_1,k_2} |\lambda_{k_1 k_2}| = 2\lambda_1 = 2\pi^2 + O(n^{-2})$.

Finite Differences

Thm: Consistency of $-\Delta_n^{(2)}\, u = f$

Assume $u : [0,1]^2 \to \mathbb{R}$ is a four times differentiable solution to $-\Delta u = f$. Then,

$$\|\Delta_n^{(2)}\, u + f\|_{2,n} = O(n^{-2}) \qquad \text{for } n \to \infty.$$

Proof. We know from the one-dimensional consistency estimate from slide 40 that

$$\tfrac{\partial^2 u}{\partial x_1^2} = (I_n \otimes \Delta_n^{(1)})\, u + O(n^{-2}) \quad \text{and} \quad \tfrac{\partial^2 u}{\partial x_2^2} = (\Delta_n^{(1)} \otimes I_n)\, u + O(n^{-2});$$

hence we conclude

$$\begin{aligned}
\big\|\Delta_n^{(2)}\, u + f\|_{2,n} &= \big\|\Delta_n^{(2)}\, u - \Delta u\|_{2,n} \\
&= \big\|(I_n \otimes \Delta_n^{(1)})\, u - \tfrac{\partial^2 u}{\partial x_1^2} + (\Delta_n^{(1)} \otimes I_n)\, u - \tfrac{\partial^2 u}{\partial x_2^2}\big\|_{2,n} \\
&= \big\|O(n^{-2})\big\|_{2,n} \\
&= O(n^{-2}).
\end{aligned}$$

Finite Differences

Corollary: Convergence of two-dimensional finite differences

Let $u : [0,1]^2 \to \mathbb{R}$ be the solution to $-\Delta u = f$ with homogeneous Dirichlet boundary conditions, and let $u_n \in \mathbb{R}^{n^2}$ be the solution to $-\Delta_n^{(2)} u_n = f$. Then,

$$\|u - u_n\|_{2,n} = O(n^{-2}) \qquad \text{for } n \to \infty.$$

Numerical demonstration

See `convergence_2d()`.

Finite differences in three dimensions

Everything that has been said about finite differences in two dimensions generalises straightforwardly to finite differences in three dimensions.

See `solve_poisson_3d()` and `convergence_3d()` for demonstration.

Finite Differences

Summary

- Heuristic derivation of $\Delta_n^{(1)}$.
- Stability and consistency lemma $\|u - u_n\| \leq \|\Delta_n^{-1}\| \, \|\Delta_n u + f\|$.
- Fourier analysis for showing stability.
- Taylor series for showing consistency.
- Vectorisation and Kronecker product for finite differences in $d > 1$ dimensions.