Lecture Content Summaries for MECH3750

Travis Mitchell

Updated: 10 October, 2019

Week 02

With thanks to Alex Muirhead:

Tutors: Alex Muirhead, Bryce Hill, Kyle Mclaren, Luke Bartholomew, William Snell

Quizzes: Weeks 3, 6, 9

Content:

- Taylor Expansions;
- Newton's Method;
- Least Squares.

Taylor Series

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots$$

a is the fixed point

$f(x)$: function we want to approx

$(x-a)$: distance from the fixed point

$\dots$: infinite series

$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

But can't always have all infinte terms.

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \underbrace{O((x-a)^2)}_{\text{error term}}$$

OR

aka Higher Order Terms in mult-variable

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a)$$

Accurate to 2^{nd} order

Note: $f_x = \frac{\partial f}{\partial x}$ etc.

What about multivariable functions?

$$f(x, y, \dots) \quad \text{OR} \quad f(x_0, x_1, \dots, x_n) = f(\vec{x})$$

two fixed points

$$f(x,y) = f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$$

$$\dots + \frac{1}{2}\left[f_{xx}(a,b)(x-a)^2 + 2f_{xy}(a,b)(x-a)(y-b) + f_{yy}(a,b)(y-b)^2 \right]$$

$$\dots + \text{H.O.T} \longleftarrow O\left(\sqrt{(x-a)^2 + (y-b)^2}\right)$$

In general:

$$f(\vec{x}) = f(\vec{x_0}) + \left[\nabla f(\vec{x_0})\right]^T (\vec{x} - \vec{x_0}) + \frac{1}{2}(\vec{x} - \vec{x_0})^T H (\vec{x} - \vec{x_0}) + \text{H.O.T}$$

where Hessian is $H = \nabla(\nabla f) = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$

Newtons Method

Want to find roots of function:

$$f(x) = \emptyset$$

$$f(x_0 + h) \approx f(x_0) + f'(x_0)h$$

(x_0: initial guess; h: step)

Let our next guess $x_1 = x_0 + h$ be a "root".

$$\therefore \quad f'(x_0)h = -f(x_0)$$

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Repeat until $f(x_n) \approx 0$

For multi-variable functions, and vector functions (aka multiple stacked equations).

$$\vec{f}(\vec{x}) = 0$$

$$\vec{f}'(\vec{x}_0) = \nabla \vec{f}(\vec{x}_0) = J \quad \{\text{Jacobian}\}$$

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots \\ \frac{\partial f_2}{\partial x_1} & \ddots & \vdots \\ \vdots & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}, \quad \vec{f}(\vec{x}) = \begin{bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_m(\vec{x}) \end{bmatrix}$$

$$\vec{f}(\vec{x_0} + \vec{h}) \approx \vec{f}(x_0) + J\vec{h} = \emptyset$$

$$\therefore \quad J\vec{h} = -\vec{f}(\vec{x_0}) \quad \leftarrow \text{easier to solve numerically}$$

$$\text{OR} \quad \vec{h} = -J^{-1}\vec{f}(\vec{x_0}) \quad \leftarrow \text{easier to solve by hand IFF } J \text{ is } 2\times 2 \text{ matrix.}$$

Least Squares

Want to approximate some vector $\vec{f}$ with vectors $\vec{p}^{(1)}, \vec{p}^{(2)}, \ldots, \vec{p}^{(n)}$

$$\vec{f} \approx \alpha_1 \vec{p}^{(1)} + \alpha_2 \vec{p}^{(2)} + \ldots + \alpha_n \vec{p}^{(n)}$$

Define **positive** error:

$$E = \sum_i \left(\alpha_1 p_i^{(1)} + \alpha_2 p_i^{(2)} + \ldots + \alpha_n p_i^{(n)} - f_i \right)^2$$

square to ensure error is positive

summing over components of the vectors

Minimise error by setting derivatives to zero.

$$\frac{\partial E}{\partial \alpha_j} = 0 \quad \forall j \in 1, 2, \ldots, n$$

$$\frac{1}{2} \frac{\partial E}{\partial \alpha_j} = \sum_i p_i^{(j)} \left(\alpha_1 p_i^{(1)} + \alpha_2 p_i^{(2)} + \ldots + \alpha_n p_i^{(n)} - f_i \right)$$

$$0 = \sum_i p_i^{(j)} \underbrace{\left(\alpha_1 p_i^{(1)} + \alpha_2 p_i^{(2)} + \ldots + \alpha_n p_i^{(n)} \right)}_{\text{unknowns}} - \underbrace{\sum_i p_i^{(j)} f_i}_{\text{knowns}}$$

In matrix notation:

$$\underbrace{\begin{bmatrix} p^{(1)} \cdot f \\ p^{(2)} \cdot f \\ \vdots \\ p^{(n)} \cdot f \end{bmatrix}}_{P^T f} = \underbrace{\begin{bmatrix} p^{(1)} \cdot p^{(1)} & p^{(1)} \cdot p^{(2)} & \cdots & p^{(1)} \cdot p^{(n)} \\ p^{(2)} \cdot p^{(1)} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ p^{(n)} \cdot p^{(1)} & \cdots & \cdots & p^{(n)} \cdot p^{(n)} \end{bmatrix}}_{P^T P} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}}_{\alpha}$$

$$\vec{\alpha} = (P^T P)^{-1} P^T \vec{f}$$

Week 03

With thanks to Alex Muirhead:

Tutors: Alex Muirhead, Bryce Hill, Kyle Mclaren, Luke Bartholomew, William Snell

Quiz: Weeks 3 - 1pm

Content:

- Least Squares continued;
- Inner product;
- Orthogonal polynomials;
- Fourier Series.

Least Squares (cont.)

Can represent discrete points on some function $f(x)$ by a vector.

$$f(\vec{x}) = \left(f(x_0), f(x_1), \dots, f(x_n) \right)^T = \vec{f}$$

Approximate vector with basis "vectors" made from functions.

$$\vec{p}^{(1)} = \left(p^{(1)}(x_0), p^{(1)}(x_1), \dots, p^{(1)}(x_n) \right)^T$$

etc.

Can be any set of **linearly independent** functions $\{p^{(1)}, p^{(2)}, \dots p^{(m)}\}$

$$\therefore \quad f(\vec{x}) = \vec{f} \approx \sum_m \alpha_m \vec{p}^{(m)}$$

$$\begin{bmatrix} \vec{p}^{(1)} \cdot \vec{p}^{(1)} & \vec{p}^{(1)} \cdot \vec{p}^{(2)} & \cdots & \vec{p}^{(1)} \cdot \vec{p}^{(m)} \\ \vec{p}^{(2)} \cdot \vec{p}^{(1)} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \vec{p}^{(m)} \cdot \vec{p}^{(1)} & \cdots & \cdots & \vec{p}^{(m)} \cdot \vec{p}^{(m)} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \vdots \\ \alpha_m \end{bmatrix} = \begin{bmatrix} \vec{f} \cdot \vec{p}^{(1)} \\ \vdots \\ \vdots \\ \vec{f} \cdot \vec{p}^{(m)} \end{bmatrix}$$

Same form as before.

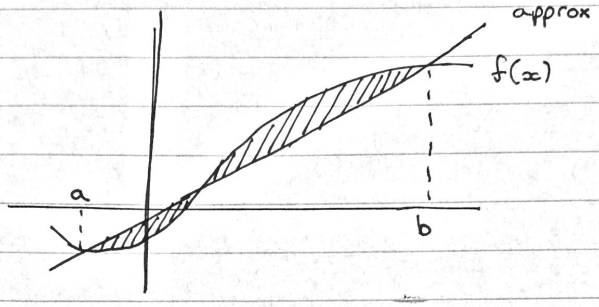
Written more concisely as:

$$P^T P \vec{\alpha} = P^T \vec{f}$$

$$\downarrow$$

$$\vec{\alpha} = (P^T P)^{-1} P^T \vec{f}$$

What if we want to approximate a continuous function over an interval?



Define **positive** error based on the area between $f(x)$ and approximation.

$$E = \int_a^b \left(\sum_i^n \alpha_i p^{(i)}(x) - f(x) \right)^2 dx$$

Follow previous derivation, set $\partial_{\alpha_j} E = 0$

$$0 = \sum_i^n \int_a^b \alpha_i p^{(i)}(x) p^{(j)}(x) dx - \int_a^b p^{(j)}(x) f(x) \, dx$$

In matrix form:

$$\begin{bmatrix} \int_a^b p^{(1)} p^{(1)} dx & \cdots & \int_a^b p^{(1)} p^{(n)} dx \\ \vdots & \ddots & \vdots \\ \int_a^b p^{(n)} p^{(1)} dx & \cdots & \int_a^b p^{(n)} p^{(n)} dx \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \int_a^b p^{(1)} f \, dx \\ \vdots \\ \int_a^b p^{(n)} f \, dx \end{bmatrix}$$

Discrete vector form & continuous function forms of least squares very similar...

Generalise by introducing **inner product.**

$\langle p, q \rangle$

vectors ↙ ↘ functions

$\vec{p} \cdot \vec{q}$ $\qquad$ $\int p(x) q(x)\, dx$

Some properties:

Norm (or length/size) $\Rightarrow \|p\| = \sqrt{\langle p, p \rangle}$

Distance $\Rightarrow d(p,q) = \|p - q\|$

Orthogonality $\Rightarrow \langle p, q \rangle \equiv 0$

Distance between two functions becomes:

$$d(p,q) = \sqrt{\int (p(x) - q(x))^2 \, dx}$$

$\therefore$ Least squares minimises distance squared!

There exist sets of linearly independent orthogonal functions. These simplify matrix form, as non-diagonal elements become $\emptyset$.

Legendre Polynomials $\qquad \int_0^1 p^{(i)} p^{(j)} \, dx \propto \delta_{ij} \equiv \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$

$P_0(x) = 1$

$P_1(x) = x$

$P_2(x) = \frac{1}{2}(3x^2 - 1)$

$P_3(x) = \frac{1}{2}(5x^3 - 3x)$

$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n$

Don't need to remember this!

Trigonometric Functions

Specifically sine and cosine, as they can be represented as:

$$e^{ix} = \cos x + i \sin x$$

Note that orthogonality holds:

$$\int_{-\pi}^{\pi} \sin(mx)\sin(nx)\,dx = \begin{cases} \pi & \text{if } m = n \\ 0 & \text{otherwise} \end{cases}$$

$$\int_{-\pi}^{\pi} \cos(mx)\cos(nx)\,dx = \begin{cases} \pi & \text{if } m = n \ * \\ 0 & \text{otherwise} \end{cases}$$

$$\int_{-\pi}^{\pi} \sin(mx)\cos(nx)\,dx = 0$$

* if $m, n > 0$

Fourier Series

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(nx) + b_n \sin(nx)$$

↑ exactly equal for an infinite sum

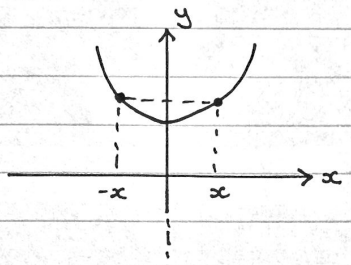
where $$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\cos(nx)\,dx$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\sin(nx)\,dx$$

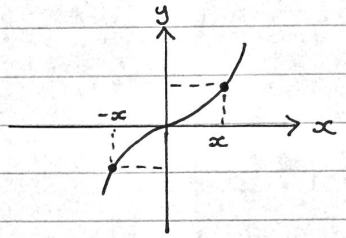
Shortcuts

Introduced **even** and **odd** functions.

Even



Odd



$f(-x) = f(x)$

{ Reflection Symmetry about y-axis }

$f(-x) = -f(x)$

{ Rotation Symmetry about origin }

Even × Even $\Rightarrow f(-x)g(-x) = f(x)g(x) \Rightarrow$ Even

Odd × Odd $\Rightarrow f(-x)g(-x) = (-1)^2 f(x)g(x) \Rightarrow$ Even

Even × Odd $\Rightarrow f(-x)g(-x) = -f(x)g(x) \Rightarrow$ Odd

$$\int_{-a}^{a} \text{Even}\, dx = \int_{-a}^{0} f(x)\,dx + \int_{0}^{a} f(x)\,dx = 2\int_{0}^{a} f(x)\,dx$$

$$\int_{-a}^{a} \text{Odd}\, dx = \int_{-a}^{0} g(x) + \int_{0}^{a} g(x)\,dx = \int_{0}^{a} -g(x) + g(x)\,dx = \emptyset$$

∴ As cosine is even & sine is odd

if $f(x)$ is even:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(nx)$$

if $f(x)$ is odd:

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

Week 04

Tutors: Luke Bartholomew, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Quiz 1 results are available for collection;
- Next quiz week 6;
- Assignment 1 due week 7.

Content:

- Fourier series continued;
- Discrete Fourier Series.

1 Fourier Series continued

Example 1.1.
Find the Fourier Series of $f(x) = x^2$ on $[-\pi, \pi]$.

This example shows a key point to note (particularly for quizzes/exams), and that is to check if the function is *odd* or *even*! The first step here is to determine our Fourier coefficients:

$$\begin{aligned} b_j &= \frac{1}{\pi}\int_{-\pi}^{\pi} \underbrace{x^2}_{even}\underbrace{\sin(jx)}_{odd}dx \\ &= 0 \\ a_j &= \frac{1}{\pi}\int_{-\pi}^{\pi} \underbrace{x^2}_{even}\underbrace{\cos(jx)}_{even}dx \\ &= \frac{2}{\pi}\int_{0}^{\pi} x^2\cos(jx)dx \end{aligned}$$

The coefficients, a_j, can then be determined through integration by parts to be,

$$a_j = \left(\frac{4(-1)^j}{j^2}\right)$$

We now evaluate, a_0, and then we can determine the Fourier series,

$$\begin{aligned} a_0 &= \frac{2}{\pi}\int_0^{\pi} x^2 dx \\ &= 2\pi^2/3 \end{aligned}$$

And therefore, as $f(x)$ is even,

$$\begin{aligned} f(x) &= a_0/2 + \sum_{n=1}^{\infty} a_n cos(nx) \\ &= \pi^2/3 - \frac{4\cos(x)}{1^2} + \frac{4\cos(2x)}{2^2} - \frac{4\cos(3x)}{3^2} + \cdots \end{aligned}$$

Remarks from Fourier Theorem, namely is we can restate the formulation as,

$$\begin{aligned} f(x) &= \sum_{j=-n}^{n} c_j e^{ijx} \\ c_j &= \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-ijx}dx, \quad j = -n,\ldots,-1,0,1,\ldots,n \end{aligned}$$

This makes use of Euler's formula (or we can also show with Taylor series) that, $e^{ijx} = \cos jx + i\sin(jx)$.

1.1 Extension to arbitrary domain

Here we first state the result for the interval $[-L, L]$,

$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{j=1}^{\infty} a_j \cos\left(\frac{j\pi x}{L}\right) + b_j \sin\left(\frac{j\pi x}{L}\right), \\ a_j &= \frac{1}{L}\int_{-L}^{L} \cos\left(\frac{j\pi x}{L}\right) f(x)dx, \quad j = 0,1,2,\ldots \\ b_j &= \frac{1}{L}\int_{-L}^{L} \sin\left(\frac{j\pi x}{L}\right) f(x)dx \end{aligned}$$

To come to this result, we simply make a transformation in which we search for the Fourier series of $F(z)$ on the domain $[-\pi, \pi]$, but set $f(x) = F(z)$ with $z = \pi x/L$.

2 Discrete Fourier Transform (D.F.T.)

Namely, we have looked at how to fit a Fourier series to a function, $f(x)$, but what if we don't know the function? E.g. we have experimental data, so we want to fit a Fourier series to a *discrete* set of data.

2.1 *Aside:* Complex inner product and complex vectors

Here we take two complex numbers,

$$\mathbf{u} = (u_1, u_2) = (a + bi, c + di)$$
$$\mathbf{v} = (v_1, v_2) = (e + fi, g + hi)$$

Therefore, the inner product is defined using the conjugate as,

$$\langle \mathbf{u}, \mathbf{v} \rangle = \bar{\mathbf{u}} \cdot \mathbf{v} = \sum_{i=1}^{n} \bar{u}_i v_i$$

Note that $\langle \mathbf{u}, \mathbf{v} \rangle \neq \langle \mathbf{v}, \mathbf{u} \rangle$

2.2 So how does the method work?

So we have a vector of complex or real data that we will state as,

$$\mathbf{f} = (f_0, f_1, \ldots, f_{N-1}).$$

To do this, we decide to approximate the discrete points using a basis, $\mathbf{p}^{(k)}$,

$$\mathbf{y} = a_0\mathbf{p}^{(0)} + a_1\mathbf{p}^{(1)} + \cdots + a_{N-1}\mathbf{p}^{(N-1)}.$$

From here, least squares is used such that we minimise $||\mathbf{y}-\mathbf{f}||^2$. This gives a normal set of equations (as seen in the least squares methods), and as we have an orthogonal basis set, all off diagonal terms of our design matrix are 0. As shown in the lectures, the diagonal terms are given by, $1/N$.

For the D.F.T. the basis set is written as,

$$\mathbf{p}_n^{(k)} = \frac{e^{ikx_n}}{N}, \quad \text{where } x_n = \frac{2\pi n}{N}, \quad n = 0, 1, \ldots, N-1, \quad k = 0, 1, \ldots, N-1.$$

If we work through the normal equations, we end up finding:

$$a_k = \sum_{n=0}^{N-1} e^{-ikx_n} f_n$$

This then gives us the coefficients for the approximation, $\mathbf{y}$, above. One particular point here is that we have used N data points and N vectors to fit our data, so we can achieve an exact representation of our vector $\mathbf{f}$,

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} a_k e^{ikx_n} = \sum_{n=0}^{N-1} a_n \mathbf{p}^{(n)}$$

The inverse DFT, which allows us to recover values of, $\mathbf{f}$, from our coefficients gives,

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} a_k e^{ikx_n} = \sum_{n=0}^{N-1} a_n \mathbf{p}^{(n)}$$

However, we note here that we do not always use all N points, in which case the equalities above become approximations.

Week 05

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Next quiz week 6;
- Assignment 1 due week 7.

Content:

- Introduction to PDEs;
- Introduction to separation of variables.

1 Introduction to partial differential equations (PDEs)

There are three core PDEs that we will look at:

- Wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

- Heat or diffusion equation:

$$\frac{\partial u}{\partial t} = k \frac{\partial^2 u}{\partial x^2}$$

- Laplace equation:

$$0 = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

These equations here are presented in lower dimensions, but not that when we have 2D or 3D analysis the operator, $\nabla^2 = \nabla(\nabla \cdot u)$ is used:

$$\nabla^2 = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

1.1 Initial Conditions

In order to solve these equations, we typically need to know what they originally look like in order to resolve them forward through time. For example, to solve the wave equation we typically need,

$$u(x,0) = \text{Initial position of e.g. a string at } t = 0,$$
$$\frac{\partial u}{\partial t}(x,0) = \text{velocity of e.g. a string at } t = 0.$$

1.2 Boundary Conditions

For solving these equations, it is also important to know what the boundaries are - i.e. what is limiting the motion of our system. Again, an example of this for the wave equation could be,

$$u(0,t) = u_0 = \text{the start of the string is fixed for all } t,$$
$$u(L,t) = u_L = \text{the end of the string may also be fixed for all , } t.$$

2 Introduction to the heat (or diffusion) equation

Used to model how heat would diffuse for example through a plate or how a concentration of particles may diffuse in space. In the lectures, we used a population of walkers on a 2D grid in order to derive the equation where we found a discrete model,

$$\frac{u(x,y,t+\Delta t) - u(x,y,t)}{\Delta t} = \frac{p(\Delta x)^2}{\Delta t} \times \left(\frac{u(x+\Delta x, y, t) - 2u(x,y,t) + u(x-\Delta x, y, t)}{\Delta x^2} + \frac{u(x, y+\Delta y, t) - 2u(x,y,t) + u(x, y-\Delta y, t)}{\Delta x^2} \right)$$

$$\frac{\partial u}{\partial t} = k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

2.1 Initial Conditions

For this system, as was seen by our lattice example, only an initial concentration is required,

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

2.2 Boundary Conditions

Here, we can define the boundaries to be sinks like was done in the lecture, where $u = 0$ on all boundaries. This type of 'fixed-value' boundary is typically termed a **Dirichlet boundary condition**, and we will touch on these throughout the course. We note here, that the boundaries don't need to be fixed at 0. For example, if we consider the diffusion equation in terms of the diffusion of heat... We could have a boundary fixed at a particular temperature.

The other boundary condition that we commonly see is a **Neumann boundary condition**, which we would write for example as

$$\frac{\partial u}{\partial x}(L, y) = g(y),$$

on the right hand side boundary of our 2D domain. This effectively specifies a flow, or flux, of a property. Again considering heat, a flow of heat in or out of the system could be seen. A particular type of interest is an insulating boundary, in which no heat can flow, $g(y) = 0$.

3 Laplace Equation

If we consider the heat on a plate as $t \to \infty$, such that the heat has reached a steady-state. We can then determine that,

$$\frac{\partial u}{\partial t} = 0.$$

And then the diffusion equation can be seen to satisfy the Laplace equation as $u(x, y, t) \to u(x, y)$. This equation is also commonly seen in incompressible fluid mechanics,

$$0 = \nabla u.$$

4 Fourier's method for the wave equation

We will move to a short-hand notation in which the wave equation can be written as,

$$u_{tt} = c^2 u_{xx}$$

with

$$\begin{aligned} \text{Initial cond's} \quad & u(x, 0) = f(x) \\ & u_t(x, 0) = g(x) \\ \text{Boundary cond's} \quad & u(0, t) = u(L, t) = 0. \end{aligned}$$

To solve this, we will seek a solution of the form,

$$u(x, t) = F(x)G(t),$$

namely, we are assuming the function is *separable*.

4.1 Introduction to separation of variables

Let us try to apply this to the boundary conditions of our previous system,

$$u(0,t) = 0 = F(0)G(t),$$
$$u(L,t) = 0 = F(L)G(t)$$

So either $G(t) = 0$ (bad) or $F(0) = 0$, and similarly we have $F(L) = 0$. Let us now look if we can apply this to the actual wave equation, so the left-hand and right-hand side of the equation will require,

$$u_{tt} = F(x)G''(t),$$
$$u_{xx} = F''(x)G(t).$$

Substituting this in and separating our variables we find,

$$\frac{G''(t)}{c^2 G(t)} = \frac{F''(x)}{F(x)}.$$

As one of these is a function of time and the other of space, we conclude that these ratios must be a constant. Which if this is true, we have two ODE's to solve,

$$F''(x) - kF(x) = 0$$
$$G''(t) - c^2 kG(t) = 0.$$

There are solutions to these, and for which we consider the constant k to be 0, μ^2 (positive) and

From this, we find $k = 0$ to be useless. From k positive we find,

$$\begin{aligned} F(x) &= Ae^{\mu x} + Be^{-\mu x} \\ &= a\cosh(\mu x) + b\sinh(\mu x). \\ \therefore F(0) = 0 &\implies a = 0 \\ F(L) = 0 &\implies b\sinh(\mu L) = 0 \end{aligned}$$

This again is not useful, so we conclude that $k < 0$, so set $k = -p^2$ which gives,

$$\begin{aligned} F(x) &= Ae^{ipx} + Be^{-ipx} \\ &= a\cos(px) + b\sin(px) \end{aligned}$$

We can then apply the boundary conditions ($F(0) = F(L) = 0$), which for us leave,

$$p = n\pi/L, \quad n = 1,2,3,\ldots$$

Now apply this value of k into the ODE for $G(t)$,

$$\begin{aligned} &G'' + p^2 c^2 G = 0 \\ &\implies G(t) = A\cos(n\pi ct/L) + B\sin(n\pi ct/L) \end{aligned}$$

with this, we can now conclude that the solutions for u are,

$$\begin{aligned} u(x,t) = F(x)G(t) &= b\sin(n\pi x/L)[A\cos(n\pi ct/L) + B\sin(n\pi ct/L)] \\ &= [A_n\cos(n\pi ct/L) + B_n\sin(n\pi ct/L)]\sin(n\pi x/L) \end{aligned}$$

Week 06

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Next quiz TODAY!;
- Assignment 1 due week 7.

Content:

- Separation of variables.

1 Separation of Variables Conti.

Last week we managed to conclude:

$$\begin{aligned} u(x,t) = F(x)G(t) &= b\sin(n\pi x/L)[A\cos(n\pi ct/L) + B\sin(n\pi ct/L)] \\ &= [A_n\cos(n\pi ct/L) + B_n\sin(n\pi ct/L)]\sin(n\pi x/L) \end{aligned}$$

Aside: A note on linearity
As each value of n gives us a new solution to the equation, as they are linear equations, the superposition of these is also a solution. This means we can write,

$$u(x,t) = \sum_1^\infty \left(A_n\cos\left(\frac{n\pi ct}{L}\right) + B_n\sin\left(\frac{n\pi ct}{L}\right)\right)\sin\left(\frac{n\pi x}{L}\right).$$

Here, we now apply initial conditions to get our coefficients. This aligns quite closely with our past work on Fourier series. We obtain:

$$A_n = \frac{2}{L}\int_0^L \sin\left(\frac{n\pi x}{L}\right) f(x)dx$$

where, $f(x) = u(x,0)$. The other initial condition, $g(x) = u_t(x,0)$ gives,

$$B_n = \frac{2}{n\pi c}\int_0^L \sin\left(\frac{n\pi x}{L}\right) g(x)dx$$

2 Fourier's method for 1-D heat equation

Looking to solve:

$$\partial_t u = k\partial_{xx} u$$

and we are considering thermally insulated ends (boundary conditions),

$$\begin{aligned} \partial_x u(0,t) &= 0, \\ \partial_x u(L,t) &= 0. \end{aligned}$$

We also have the initial condition,

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

As per the previous case, we are assuming we can find a solution in a separable form:

$$u(x,t) = F(x)G(t).$$

Now apply the boundary conditions again and eliminate useless solutions to obtain,

$$F'(0) = F'(L) = 0.$$

Now formulate the partials and sub them into the heat equation. Separating our variables onto the left and right hand side again gives us,

$$\frac{G'}{c^2G} = \frac{F''}{F}.$$

This is again a function of t equal to a function of x, so it is only possible if they equal a constant. Therefore, we now have two partial differential equations that we know how to solve,

$$\begin{aligned} G' &= kc^2G \\ F'' &= kF, \quad F'(0) = F'(L) = 0. \end{aligned}$$

If we go through the process, we again find that the constant needs to be negative so for convenience set, $k = -p^2$. Thus we need to solve,

$$\begin{aligned} F'' &= -p^2 F \\ \therefore F &= a\cos(px) + b\sin(px). \end{aligned}$$

Applying our boundary conditions for F gives,

$$\begin{aligned} b = 0, \quad p &= \frac{n\pi}{L}, \\ \implies F(x) &= a\cos\left(\frac{n\pi x}{L}\right). \end{aligned}$$

Therefore, our temporal function becomes

$$\begin{aligned} G' &= kc^2 G \\ \implies G(t) &= D\exp\left[-\left(\frac{n\pi c}{L}\right)^2 t\right] \end{aligned}$$

Combining these equations we get,

$$\begin{aligned} u(x,t) &= A_0 \\ u_n(x,t) &= A_n \exp\left[-\left(\frac{n\pi c}{L}\right)^2 t\right]\cos\left(\frac{n\pi x}{L}\right). \\ \therefore u(x,t) &= A_0 + \sum_n u_n. \end{aligned}$$

This currently satisfies the PDE and the BCs, so we then use the initial condition to define the coefficients. From this,

$$A_n = \frac{2}{L}\int_0^L f(x)\cos\frac{n\pi x}{L}dx$$

2.1 Steady-state, 2D heat equation

We can apply the same ideas to,

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

Doing the normal procedure, $u(x,y) = F(x)G(y)$ we achieve the two ODEs,

$$\begin{aligned} F'' &= kF \\ G'' &= -kG. \end{aligned}$$

Again, taking k as negative, we find,

$$\begin{aligned} F &= A\cos(px) + B\sin(px), \quad F(0) = F(a) = 0 \\ \implies F(x) &= B_n \sin\left(\frac{n\pi}{a}x\right) \\ G &= C\cosh\left(\frac{n\pi y}{a}\right) + D\sinh\left(\frac{n\pi y}{a}\right), \quad G(0) = 0 \\ G(y) &= D\sinh\left(\frac{n\pi y}{a}\right). \end{aligned}$$

So combining these,

$$u(x,y) = \sum_n B_n \sin\left(\frac{n\pi}{a}x\right)\sinh\left(\frac{n\pi}{a}y\right)$$

After this process there is conveniently one BC left to fix B_n at $u(x,b) = f(x)$. Using a Fourier sine series we find,

$$B_n = \frac{2}{a\sinh(2\pi b/a)}\int_0^a f(x)\sin\frac{n\pi x}{a}dx$$

Week 07

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Collect your quiz results;
- Hopefully you have submitted your assignment;
- Next quiz is in Week 9;
- Next assignment isn't until Week 12.

Content:

- Started Part II of the course! Well done making it through Part I;
- Classification of PDEs;
- Numerical solutions to PDEs.

1 Classification of PDEs

Chris gave a pretty good description of these, so one of the take home messages is how to determine which kind of PDE you are dealing with so that we can figure out appropriate solution techniques. To do this, consider the PDE in a general form (we will look at second-order PDEs here),

$$A\partial_{xx}u + 2B\partial_{xy}u + C\partial_{yy}u = D\partial_x u + E\partial_y u.$$

Thus, if we consider this to be a quadratic equation, we can calculate the determinant by,

$$(2B)^2 - 4AC = 0.$$

With this, we then have the cases:

$$B^2 - AC \begin{cases} = 0, & \text{Parabolic} \\ < 0, & \text{Elliptic} \\ > 0, & \text{Hyperbolic} \end{cases}$$

2 Numerical solutions for PDEs

2.1 The diffusion equation

$$\partial_t u = D\partial_{xx}u$$

We can solve this explicitly using a forward difference in time,

$$\partial_t u = \frac{u_j^{m+1} - u_j^m}{\Delta t} + O(\Delta t),$$

or implicitly using a backward difference in time,

$$\partial_t u = \frac{u_j^m - u_j^{m-1}}{\Delta t} + O(\Delta t).$$

For the spatial derivative, we often use a second order central difference,

$$\partial_{xx}u = \frac{u_{j+1}^m - 2u_j^m + u_{j-1}^m}{(\Delta x)^2}.$$

Therefore, if we substitute these both into the diffusion equation above, the explicit update would become,

$$\boxed{u_j^{m+1} = u_j^m + \frac{D\Delta t}{(\Delta x)^2}\left(\frac{u_{j+1}^m - 2u_j^m + u_{j-1}^m}{(\Delta x)^2}\right)}$$

This equation will then govern the dynamics of our initial population. In a numerical problem, we also tend to describe boundary conditions that will need to be taken into account. In the lectures we went through writing this in matrix form, this can be a very convenient way to code up the solutions and useful to analyse stability.

Week 08

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Next quiz is in Week 9;
- Next assignment due in Week 12.

Content:

- Backwards difference schemes and stability;
- Boundary conditions and implementation;
- Crank-Nicolson scheme.

Last week we generalised the explicit finite difference scheme. Additionally, the stability of these schemes were analysed using a *von Neumann* analysis. From here boundary conditions were studied, in particular, how to build these into the general matrix scheme that was derived.

1 Backwards difference scheme for parabolic PDEs

So far we have looked at using a *forwards in time, centred in space* approach sometimes referred to as FTCS scheme. However, as this leads to an **explicit** scheme, it has stability criterion - i.e. the CFL number in which the σ coefficient (following lecture notation) must be less than or equal to 1/2.

If we consider the diffusion equation,

$$\partial_t u = D\partial_{xx} u$$

and discretise this using a backwards difference we get,

$$\frac{u_j^m - u_j^{m-1}}{\Delta t} = \frac{D(u_{j+1}^m - 2u_j^m + u_{j-1}^m)}{(\Delta x)^2}$$
$$u_j^{m-1} = u_j^m - \frac{D\Delta t}{(\Delta x)^2}\left(u_{j+1}^m - 2u_j^m + u_{j-1}^m\right)$$

Now, taking the normal approach we set $\sigma = \frac{D\Delta t}{(\Delta x)^2}$ and write,

$$u_j^{m-1} = (1+2\sigma)u_j^m - \sigma u_{j+1}^m - \sigma u_{j-1}^m$$

In order to solve this, we need to write our system of equations in matrix form so that we have,

$$A\mathbf{u}^m = \mathbf{u}^{m-1}$$
$$\implies \mathbf{u}^m = A^{-1}\mathbf{u}^{m-1}$$

Therefore, we are able to find our timestep m from the previous, $m-1$, and update our system. In doing this, be careful about the top and bottom row of your matrix A, as this must incorporate boundary conditions!

1.1 Boundary conditions

For Neumann (or flux) boundary conditions, we can treat these as first or second order,

$$\frac{\partial u}{\partial x} \approxeq \begin{cases} \frac{u_{j+1}^{m+1} - u_j^{m+1}}{\Delta x} = c & \implies u_{j+1}^{m+1} - u_j^{m+1} = c\Delta x \quad \text{, first order} \\ \frac{-3u_j^{m+1} + 4u_{j+1}^{m+1} - u_{j+2}^{m+1}}{2\Delta x} = c & \implies -3u_j^{m+1} + 4u_{j+1}^{m+1} - u_{j+2}^{m+1} = 2c\Delta x \quad \text{, second order} \end{cases}$$

1.2 Characteristics of implicit scheme

- The backward difference scheme is unconditionally stable – permits any value of σ;
- The local truncation error is $O(\Delta x^2) + O(\Delta t)$, the same as for the forward difference scheme
- Unconditional stability permits much larger time steps, but they attract larger truncation error
- If the time step is chosen as roughly equivalent to the grid spacing, then $O(\Delta t) \ll O(\Delta x^2)$, and so the error of the backward difference scheme becomes $\sim O(\Delta t)$
- It is possible to use a mixed approach, which maintains unconditional stability, and reduces the leading error term of the local truncations

2 Crank-Nicolson (CN) scheme for diffusion

Okay, so we achieved unconditional stability - but what if we want to reduce error? This is where the CN scheme comes in. The idea here is that we use a mixed central difference, namely we take the average central difference between the current and previous timestep,

$$\partial_{xx}u \approx \frac{1}{2}\left(\frac{u_{j+1}^{m}-2u_{j}^{m}+u_{j-1}^{m}}{\Delta x^2}+\frac{u_{j+1}^{m-1}-2u_{j}^{m-1}+u_{j-1}^{m-1}}{\Delta x^2}\right)$$

In addition to this, we use our backwards difference in time for the temporal derivative. Substituting these into the diffusion equation and arranging terms of matching timestep on LHS and RHS (*try this yourself!*) we get,

$$-\frac{\sigma}{2}u_{j-1}^{m}+(1+\sigma)u_{j}^{m}-\frac{\sigma}{2}u_{j+1}^{m}=\frac{\sigma}{2}u_{j-1}^{m-1}+(1-\sigma)u_{j}^{m-1}+\frac{\sigma}{2}u_{j+1}^{m-1}$$

This, we can then write into matrix form and look to solve. In the end, this gives us a scheme that has error $O(\Delta x^2)+O(\Delta t^2)$.

2.1 Boundary conditions

These can be applied in the same way as our previous schemes, namely, discretise the if a Neumann boundary condition and form this equation in the top/bottom rows of the matrices. For the CN scheme, the matrix associated with the previous timestep may have all zeros for the top and bottom row. This is a result of the boundary conditions not being dependent (necessarily) on the previous time.

Week 09

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Quiz is in Week 9;
- Next assignment due in Week 12.

Content:

- Hyperbolic PDEs - what are they and where are they used;
- Hyperbolic PDEs - how can we solve them?

1 Hyperbolic PDEs

1.1 First order hyperbolic PDEs

Here we will start by studying the **convection equation** which has many practical uses e.g. flows of air and water, transport phenomena and traffic behaviour. THe general form of the first order wave equation can be written as,

$$a\partial_t u + v\partial_x u = w.$$

In this equation we have, u which is a multivariate function (e.g. a concentration), $v = v(x,t)$ is a velocity field and w which is a source term. The coefficient a is typically set to one.

In terms of the convection equation, it is often non-homogenous (i.e. $w \neq 0$) but can be homogenous,

$$\partial_t u + v\partial_x u = 0.$$

Furthermore, the equation can be linear (if v and w are independent of u) or non-linear. A few examples of non-linearity include,

$$\partial t u + u\partial_x u = x \qquad \partial t u - x\partial_x u = 1/u.$$

1.2 Second order hyperbolic PDEs

To study hyperbolic PDEs of second order, we will focus on the wave equation,

$$\partial_{tt} u = c^2 \partial_{xx} u,$$

where c is the wave speed.

Examples of where this is used:

Gas Dynamics
If we start with our compressible Navier-Stokes equations, we can linearise these and introduce an equation of state to obtain,

$$\partial_{tt}\rho = a^2\partial_{xx}\rho, \quad a^2 = \left[\frac{dp}{d\rho}\right]_{s=s_0}$$

Oscillations (e.g. guitar string)
In lectures we created a free body diagram of a *discrete* representation of a continuous string. Applying Newton's second law of motion to these we are able to obtain,

$$\partial_{tt} y = a^2\partial_{xx} y, \quad a^2 = \left[\frac{F}{\rho A}\right]$$

This is all well and good, but how do we actually solve these problems?

1.3 Finite difference solution for the wave equation

So starting with the wave equation, we can apply second order central differences to both the temporal and spatial derivatives,

$$\partial_{tt} u = c^2 \partial_{xx} u$$
$$\frac{u_i^{m+1} - 2u_i^m + u_i^{m+1}}{\Delta t^2} = c^2 \frac{u_i^{m+1} - 2u_i^m + u_i^{m+1}}{\Delta x^2}$$

From here, we can arrange an explicit update for time, $m+1$. This involves multiplying by Δt^2 and taking terms to the right hand side giving,

$$
\begin{aligned}
u_i^{m+1} &= 2u_i^m - u_i^{m-1} + c^2 \frac{\Delta t^2 (u_{i+1}^m - 2u_i^m + u_{i-1}^m)}{\Delta t^2} \\
u_i^{m+1} &= 2u_i^m - u_i^{m-1} + \frac{c^2 \Delta t^2}{\Delta t^2} \left[u_{i+1}^m - 2u_i^m + u_{i-1}^m \right] \\
u_i^{m+1} &= 2u_i^m - u_i^{m-1} + \sigma^2 \left[u_{i+1}^m - 2u_i^m + u_{i-1}^m \right] \\
u_i^{m+1} &= \sigma^2 u_{i+1}^m + (2 - 2\sigma^2) u_i^m + \sigma^2 u_{i-1}^m - u_i^{m-1}
\end{aligned}
$$

It can be seen, that our general update depends not only on the previous time step, but the time step before this one as well. As such, we typically need both an initial position and velocity to solve this equation. The velocity, we approximate with a central difference and use to eliminate the term u_i^{m-1}. After this is done, a matrix equation can be formed as per usual.

1.4 Non-reflecting boundary conditions

When applying normal boundary conditions to the wave equation, it is possible that wave reflections back into the domain occur. Depending on the problem, this may not be physical and as such, schemes are required to let the waves propagate out of the domain. To understand the cause of (and ultimately how to eliminate) this issue, we consider the *characteristics* of the wave equation. Therefore, we can analyse the wave equation as,

$$
0 = \left(\partial_{tt} - c^2 \partial_{xx} \right) u = (\partial_t - c\partial_x)(\partial_t + c\partial_x) u
$$

Here we see that we actually have two *one*-way convention equations that propagate with velocity, c and $-c$. When analysing PDEs analytically, this so called *Method-of-Characteristics* can be applied to determine a solution. Numerically though, understanding these allows us to remove them at the boundary resulting in a non-reflecting BC.

Therefore, we must solve for our boundary node using the outgoing convective equation and ignore the incoming wave. Thus on the left hand boundary condition,

$$
\begin{aligned}
\partial_t u - c\partial_x u &= 0 \\
\frac{u_0^{m+1} - u_0^m}{\Delta t} - c\frac{u_1^m - u_0^m}{\Delta x} &= 0 \\
u_0^{m+1} &= u_o^m + \sigma(u_1^m - u_0^m)
\end{aligned}
$$

The same can be done for the right hand side.

Week 10

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Quiz is the exam;
- Next assignment due in Week 12 (looking like a possible extension).

Content:

- Convection equations;
- Advanced methods for convection terms.

1 Convection Equation

We started this week by looking at the convection equation,

$$\partial_t u + v\partial_x u = w.$$

To numerically discretise the spatial derivative we looked at three potential schemes,

- **Upwind**

$$\partial_x u = \frac{u_j - u_{j-1}}{\delta x} + O(\delta x)$$

  Can be solved with the typical schemes we've seen, i.e. explicit, implicit, Crank-Nicolson (mixed). In this equation is where the term numerical diffusion was highlighted. Here we substituted in a Taylor series for our partial and could define a diffusion term related to grid size as,

$$\partial_t u + v\partial_x u = D_{num}\partial_{xx} u + H.o.T$$

- **Downwind –> ALWAYS UNSTABLE**

$$\partial_x u = \frac{u_{j+1} - u_j}{\delta x} + O(\delta x)$$

- **Central**

$$\partial_x u = \frac{u_{j+1} - u_{j-1}}{2\delta x} + O(\delta x^2)$$

  This method is stable if written implicitly, and marginally stable in Crank-Nicolson form. The central difference scheme is typically avoided for these advection type problems due to reflections and sharp shapes.

Keep in mind for solving convection terms that the numerics must match the physics! This is important to consider for boundary conditions as well as the general formulation of your scheme.

1.1 Advanced solution techniques

- **Lax-Wendroff Scheme**
  This scheme allows us to solve the convection equation with $O(\delta x^2)$ and $O(\delta t^2)$, which is not universally the case with the previous methods. To do this, a second order Taylor series is used as a basis. This means we take the value of u at time $t + \delta t$ as,

$$\begin{aligned} u_j^{m+1} &= u_j^m + \delta t\partial_t u_{j,m} + \frac{\delta t^2}{2}\partial_{tt} u_{j,m} + H.o.T \\ &\approx u_j^m - v\delta t\partial_x u + v^2\frac{\delta t^2}{2}\partial_{xx} u \\ &\approx u_j^m - v\delta t\frac{u_{j+1}^m - u_{j-1}^m}{2\delta x} + v^2\frac{\delta t^2}{2}\frac{u_{j+1}^m - 2u_j^m + u_{j-1}^m}{\delta x^2} \end{aligned}$$

  Therefore, taking $\sigma = v\delta t/\delta x$ and rearranging we find,

$$u_j^{m+1} = \frac{\sigma}{2}(1+\sigma)u_{j-1}^m + (1-\sigma^2)u_j^m - \frac{\sigma}{2}(1-\sigma)u_{j+1}^m + O(\delta x^2) + O(\delta t^2)$$

- **MacCormack Method** This method takes the form of an explicit predictor-corrector FD scheme that is also second order accurate in space and time. This is done in steps,

  1. Update prediction, u_j^p, with a forward in time, backward in space difference,

$$u_j^p = u_j^m - \sigma(u_j^m - u_{j-1}^m)$$

2. The corrector, u_j^c, is then determined with a backward in time, forward in space difference,

$$u_j^c = u_j^m - \sigma(u_{j+1}^p - u_j^p)$$

3. Now we update, which for linear equations is equivalent to the Lax-Wendroff scheme,

$$u_j^{m+1} = \frac{u_j^p + u_j^c}{2}$$

The MacCormack method is quite general and is often applied to hyperbolic systems, linear and non-linear with waves propagating in both directions. However, it is not monotonic and as such is not good for handling jumps/shocks in a system.

2 Convection and Diffusion

Just introduced, but may be rather important for your assignment!

$$\partial_t u + \mathbf{v} \cdot \nabla u = D\Delta u$$

Week 11

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

- Elliptic PDEs;
- Sparse matrices and flattening data.

Content:

- Convection equations;
- Advanced methods for convection terms.

1 Elliptic PDEs

This week we saw the start up of a new lecturer and a focus on elliptical PDEs,

$$\begin{aligned}\partial_{xx}f + \partial_{yy}f = 0,\\ \nabla^2 f = 0,\\ \Delta f = 0.\end{aligned}$$

Here we see that the PDE has **no** temporal derivative and thus governs *steady-state* processes where the solution is influenced by the B.C. not the I.C.. As such problems are often defined by their type of boundary,

1. First boundary value problem (BVP) has Dirichlet boundaries, i.e. f is fixed on the bounding surface;
2. Second BVP has Neumann boundaries where $\partial_n f$ is defined on the bounding surface;
3. Third BVP has Robin or mixed boundaries where f may be prescribed on a portion of the boundary and $\partial_n f$ is prescribed on the remainder.

1.1 Polar coordinates

Often it can be useful to transform from a Cartesian type discretisation and use polar coordiantes to resolve a system. This however has to be taken into account in our governing equation,

$$\begin{aligned}x = rcos\theta \qquad y = rsin\theta\\ \implies \partial_{rr} + \partial_r f/r + \partial_{\theta\theta} f/r^2 = 0\end{aligned}$$

The fundamental solutions in polar coordinates:

- source/sink: $f(r) = \frac{Qln(r)}{2\pi}$
- vortex: $f(\theta) = \frac{\Gamma\theta}{2\pi}$

1.2 Poisson equation

The Laplace equation is a homogeneous PDE, if we have a source term it is called Poisson's equation,

$$\partial_{xx}f + \partial_{yy}f = S(x,y).$$

1.3 Numerical solution for the Laplace equation

If we consider this for a steady state heat problem, we can discretise the Laplace equation using centred finite differences,

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0.$$

This gives us a two dimensional stencil which isn't typically convenient to work with so we often need to construct a single-ordinate representation of this. In doing so, we can assemble nodal equations in the form $Ax = b$, where A is our coefficient matrix, x is our unknown temperatures and b is full of known values.

To move from the (i, j) mesh, one technique in Python is to use lambda functions,

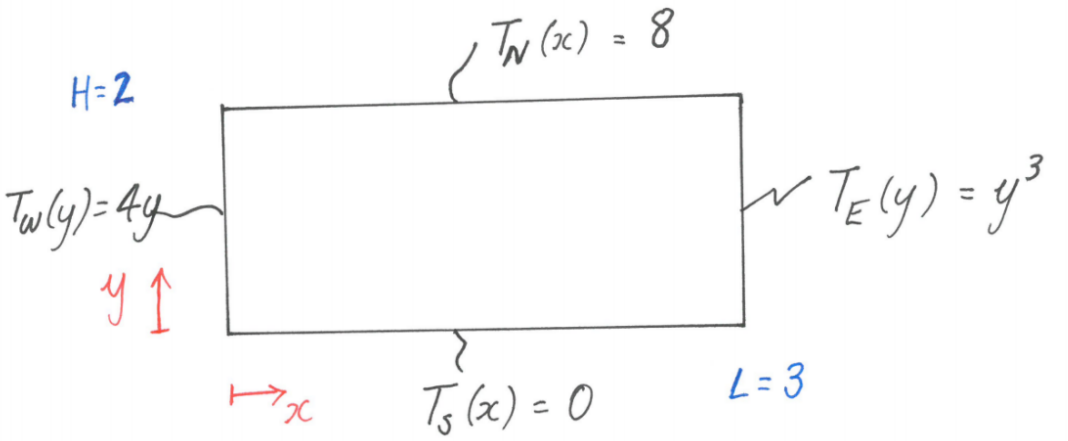
```
1 X, Y = ##domain length and height
2 dx, dy= ##chosen discretisation
3 nx, ny = X/dx + 1, Y/dy + 1
4 m = lambda i,j: j * nx + i
```

Using this, we can then formulate our matrix,

```
A = scipy.sparse.lil_matrix((nx*ny,nx*ny))
for j in range(ny):
    for i in range(nx):
        p = m(i,j)
        A[p, m(i, (j-1) % ny)] = # i,j-1 stencil
        A[p, m((i-1) % nx, j)] = # i-1,j stencil
        A[p, p] = # main diagonal
        A[p, m(i, (j+1) % ny)] = # i,j+1 stencil
        A[p, m((i+1) % nx, j)] = # i+1,j stencil
A.tocsr() #compressed sparse row matrix
```

If the problem size is small, using direct methods of solving can suffice, however, when we move to larger scale problems e.g. potentially your assignment.... Iterative methods are often favoured, Gauss-Seidel, Conjugate gradient, steepest descent etc.

We can practice this on the question the lecture:



Example: Find the steady-state distribution of temperature in the plate of length 3 m and height 2 m. Boundary conditions are shown in the figure. Use the finite-difference solution method with 11 nodes in the i-direction and 6 nodes in the j-direction.

Week 12

Tutors: Luke Bartholomew, Nathan di Vaira, Bryce Hill, Kyle Mclaren, Travis Mitchell, Alex Muirhead, William Snell

Assessment:

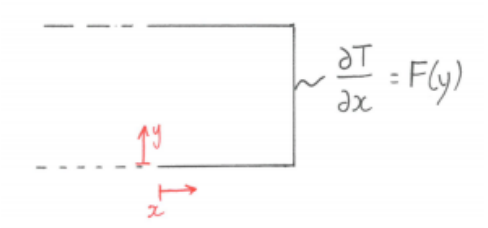
- Assignment is due next week!

Content:

- Elliptic PDEs - implementing boundary conditions;
- Verification;
- Grid generation.

1 Treatment of Neumann boundary conditions

Simple approach
The simple approach here is to take a one-sided finite difference approximation: Here, the one-sided differ-



$$\frac{T_{imax,j} - T_{imax-1,j}}{\Delta x} = F(y_j)$$

ence has been used such that information relies only on the interior of the domain.

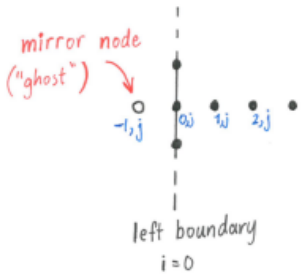
Second-order approach
Can you write a second order accurate method for this boundary? What is the form of this and the related error term? *Hint:* can use method of undetermined coefficients, or derive through a Taylor expansion.

1.1 Symmetry boundary conditions

This is a special type of Neumann boundary condition in which the derivative with respect to the normal of the domain boundary is 0,

$$\frac{\partial T}{\partial \mathbf{n}} = 0.$$

Example for Laplace equation:



At node $(0, j)$:

$$\frac{T_{-1,j} - 2T_{0,j} + T_{1,j}}{\Delta x^2} + \frac{T_{0,j-1} - 2T_{0,j} + T_{0,j+1}}{\Delta y^2} = 0$$

Find expression for ghost value T_{-1} using symmetry boundary condition:

$$\frac{\partial T}{\partial n} = 0 \quad \rightarrow \quad \frac{T_{1,j} - T_{-1,j}}{2\Delta x} \approx 0 \quad \rightarrow \quad T_{-1,j} = T_{i,j}$$

Finally giving:

$$\frac{2T_{1,j} - 2T_{0,j}}{\Delta x^2} + \frac{T_{0,j-1} - 2T_{0,j} + T_{0,j+1}}{\Delta y^2} = 0$$

2 Discretisation error

In solving the Laplace (or Poisson) equations, we have been using second order central differences to determine both the second derivative with respect to x and y. For some discretisation, $(\delta x, \delta y)$ the corresponding error terms are,

$$\varepsilon_{xx} = \frac{\delta x^2}{12}\frac{\partial^4 f}{\partial x^4}(\xi_i, y_j), \quad \text{with } x_{i-1} \le \xi_i \le x_{i+1}$$
$$\varepsilon_{yy} = \frac{\delta y^2}{12}\frac{\partial^4 f}{\partial y^4}(x_i, \eta_j), \quad \text{with } y_{j-1} \le \eta_i \le y_{j+1}$$

Here it can be seen that the global error is $O(\delta x^2 + \delta y^2)$, namely a second order scheme.

Boundary influence
What happens when we use a first order one-sided difference for our boundary in this second order scheme? It turns out that 'in-general' one can use a boundary discretisation that is one order less than the interior scheme without having a significant impact on the global order of accuracy.

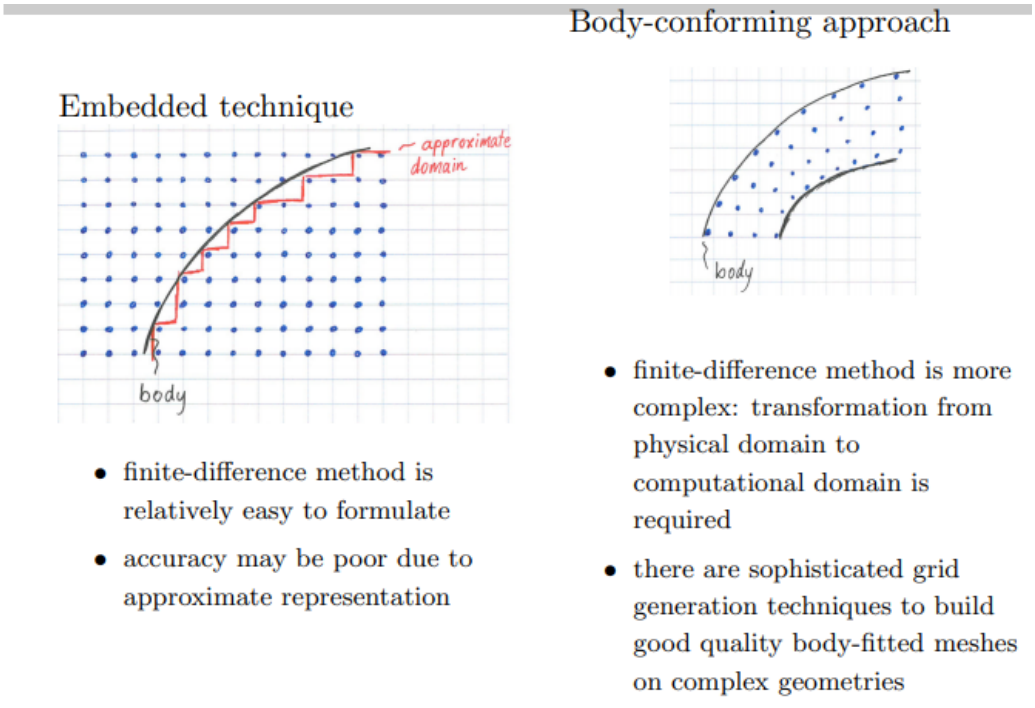
3 Applying finite difference to real world problems

3.1 Verification

Before we can trust our code, we first have to verify that it is solving what we wanted it to solve. The best way to do this is to compare our error and how this converges in comparison to an analytical solution. Namely, for the Laplace equation we have been looking at, we know our scheme is second order, so with mesh refinement we should approach an analytical solution accordingly.

3.2 Grid generation

If only life was lived on a square grid.... To cater for this we can either use an embedded or body-conforming approach, It turns out that elliptic PDEs can be used for grid generation themselves. So by solving a pre-



determined PDE, we can determine the location for our grid points.

Structured grid generation

Here we will look at two classes of methods:

- algebraic grid generation (e.g. transfinite interpolations)
- PDE-based grid generation (e.g. elliptic grid generation)

The use of a PDE solution comes with a number of benefits including the determination of grid locations based solely on the boundary specification and the smoothness of the resulting mesh.