1 Welcome to Computational methods for partial differential equations

Here we

1. Introduce the topic of computational methods for PDEs
2. Discuss the lecture notes and programming / coding
3. Discuss assessment

1.1 Computational methods for PDEs

The purpose of this module is to introduce you to the analysis and implementation of computational methods for partial differential equations. We shall study **finite difference** methods, **spectral collocation** methods, the **ultraspherical spectral** method and, time permitting, the **finite element** method.

A broader perspective: A quote from *An Applied Mathematician's Apology* by L.N. Trefethen, 2022:

... the language of the laws of nature, partial differential equations or PDEs. When Maxwell discovered that light and radio waves are one, it was because of a PDE. When Einstein predicted gravitational waves, it was because of a

PDE. Chemistry is built on Schrodinger's equation, fluid mechanics on the Navier-Stokes equations, and civil engineering on the equations of elasticity.

A few important PDEs (there are many variations!)

- Laplace equation ($f = 0$) / Poisson equation ($f \neq 0$)

$$\nabla^2 u = f \qquad \text{or} \qquad \Delta u = f$$

An elliptic equation

- Heat equation

$$u_t - \nabla^2 u = f$$

A parabolic equation

- Transport / Advection and Wave equation

$$u_t + (v \cdot \nabla) u = 0$$
$$u_{tt} - \nabla^2 u = f$$

Wave equation: a hyperbolic equation

- Navier's equations : small-strain elasticity

$$-\mu\nabla^2\mathbf{u} + (\lambda+\mu)\nabla(\nabla\cdot\mathbf{u}) = \mathbf{f}$$

- Navier-Stokes equations : incompressible viscous fluids

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} = \nu\nabla^2\mathbf{u} - \nabla p, \quad \nabla\cdot\mathbf{u} = 0$$

Numerical solutions to the (quasilinear) Navier-Stokes equations have been computed for decades (for example, these equations are used for weather prediction), however the mathematical analysis of these PDEs is still not settled (see the Wiki page on Navier-Stokes existence and smoothness): the Clay Mathematics Institute in May 2000 made this problem one of its seven Millennium Prize problems in mathematics. It offered a US$1,000,000 prize to the first person providing a solution for a specific statement of the problem:

Prove or give a counter-example of the following statement:

In three space dimensions and time, given an initial velocity field, there exists a vector velocity and a scalar pressure field, which are both smooth and globally defined, that solve the Navier-Stokes equations.

- Reaction-Diffusion systems; example, Brusselator: simple model for an autocatalytic reaction (arises also in enzymatic reactions, plasma and laser

physics)

$$u_t = B + u^2 v - (A+1)u + \alpha \nabla^2 u$$
$$v_t = Au - u^2 v + \alpha \nabla^2 v$$

- Black-Scholes equation (mathematical finance)

$$\frac{\partial u}{\partial t} + \frac{1}{2} x^2 \sigma^2 \frac{\partial^2 u}{\partial x^2} + rx \frac{\partial u}{\partial x} - ru = 0$$

Through a change of variables, this equation can be reduced to the heat equation. According to Nassim Taleb, this equation should be called the Fokker-Plank and / or Kolmogorov Forward Equation and he also explains why we never use the Black-Scholes equation. The use of mathematical models have resulted in mixed fortunes in finance: for example, there's the cautionary tale When Genius Failed: The Rise and Fall of Long-Term Capital Management (in which Myron Scholes was involved) and then there's the extremely successful Renaissance Technologies, a hedge fund founded by the mathematician Jim Simons, see the book *The Man who Solved the Market: How Jim Simons Launched the Quant Revolution*.

- Maxwell's equations
- Einstein field equations
- Dirac equation
- ...

Only in very few cases are we able to solve a PDE exactly, so we use computational methods to approximate solutions to PDEs. Here are some videos of simulations produced by computing solutions to differential equations:

- Cahn-Hilliard equation: (phase separation of a binary fluid)

$$c_t = \Delta\left(c^3 - c - \gamma\Delta c\right)$$

- Molecular Dynamics: (or more general Hamiltonian dynamics)

$$m_n \ddot{r_n} = -\nabla E(r_1, \ldots, r_N)$$

used here to model water turing into ice.

- Nonlinear elasticity
- thermal convection in earth's mantle

- spread of covid-19
- simulating the spread of an epidemic
- salt crystal melting in water
- NASA climate simulations
- Predicting matter distribution in the universe
- flow past an airfoil - high resolution
- Aerodynamics of Different Geometries - 2d Navier-Stokes Simulation
- Another Navier-Stokes simulation

1.1.1 Reality $\rightarrow$ Model$\rightarrow$ Simulation $\rightarrow$ Prediction

- Mathematical models of "real-world" processes (physics, chemistry, life sciences, engineering, ...) more often than not take the form of a system of ordinary or partial differential equations, integral equations, or closely related models. (sometimes stochastic, on networks, etc, but this is beyond the scope of this module)
- In this module we take the mathematical model as given and study how it is solved numerically.

- But we should not forget what the origin of the mathematical model is so that we are appreciative of the requirements.
  - the model is an approximation of reality, the numerical scheme / computational method is an approximation of the model.
  - how accurate is the model, i.e. how accurate should the numerical solution be?
  - how accurate are the model parameters?
  - are real-time simulations required?
  - long-time behaviour?
  - inverse and control problems

1.2 Some Admin

1.2.1 Jupyter Notebooks

This course will be taught via Jupyter notebooks. These notebooks provide a nice environment mixing web technologies, mathematical text (latex) and code. I will use Julia for this course (The "Ju" in Jupyter) but these notebooks can in principle also used with Python ("Py"), R, and other languages: docs"The Jupyter system supports over 100 programming languages (called

"kernels" in the Jupyter ecosystem) including Python, Java, R, Julia, Matlab, Octave, Scheme, Processing, Scala, and many more. "

- For today's lecture just follow along; I will say a little more about Jupyter, Julia and our eco-system in the next lecture.
- You can try to access the Jupyter hub or install Julia/Jupyter/IJulia by yourself until then, following the `WS01 notebook` which can be accessed through the course github website.

1.3 Lecture notes and coding

The importance of computational methods

How to run Julia code

1.3.1 Sources for the lecture notes

1.4 Admin

Blackboard and Github

assessment