Math 405/607E: Numerical Methods for DEs: Introduction

Colin Macdonald, cbm@math.ubc.ca

course websites: www.math.ubc.ca/~cbm/math405/2018, and canvas.ubc.ca

office: LSK 303c

office hours: TBA

Motivation & Introduction

- Look at Wright et al's video of Earth's mantle convection: www.youtube.com/watch?v=-kDb0HlDsIM

This is **modelled** using partial differential equations (PDEs).

Modelling: describing a physical (or otherwise) problem using mathematics. Often approximating the situation by making assumptions about the relative importance of various aspects of the problem.

Unknowns in Grady's model would include, amongst other things:

$T(x, y, z, t)$ temperature, $\vec{u}(x, y, z, t)$ velocity.

Can't store a continuum function $T(x, y, z, t)$ on a computer, so it is **discretized**, that is, stored at a finite set of points (or otherwise represented using a finite number of degrees of freedom). The PDEs in the model are approximating using these discrete values.

Time-stepping: causality: often discrete soln values at next (discrete) time determined approximately in terms of soln values at previous discrete time. This gives a large nonlinear system (millions of unknowns).

Discrete solver: find a (the?!) discrete soln of this nonlinear system, typically via iteration: errors will be made here as we truncate a sequence after some finite number of iterations.

Results visualized (approximately) or overall statistics collected or, ... From this, design decisions can be made (how to change the shape of the airplane, where to drill the next oil well, which stock to buy, etc). Or results compared against a simpler qualitative model (e.g., continental drift).

Often in a tight loop, results guide changes to the problem, then run simulation again. E.g., inverse problems.

Finally, usually all parts of this happen on a computer with a very fine but none-the-less discrete number system (floating point arithmetic) where *every* calculation introduces a small error.

The Many Sources of Error

- Modelling errors
- Experimental errors (e.g., data collection)
- Discrete errors:
  - Discretization (space and time)
  - Iteration
  - Visualization
  - Rounding Error (floating point)

We cannot just assume all the above will work! Or give accurate enough results to be usable in design decisions, scientific inquiry, etc.

Modelling error might be Someone Else's Problem but we want to be able to force the discrete errors (**"numerical" errors**) in the simulation small enough so that we (or the user) can *see* the modelling error. E.g., to compare it to *physical experiment* or some other validation with reality.

Controlling numerical errors

How? Choose/design/invent numerical methods:

- discrete solver should get arbitrarily close to the (exact) solution of the discretized problem (e.g., as number of iterations increase).
- discretized problem should become arbitrarily close to the PDE as the number of discrete points increases.
  - actually, want *solution* of discretion to be become arbitrarily close to the *PDE solution*.

These properties are called **convergence** (of the iterative solver and of the discretization respectively).

Objectives of this course

Introduce basic ideas of discretization, approximation, iterative solution of systems, convergence of methods to approximate differential equations (ODEs and PDEs).

Numerical Analysis: plenty of nice mathematics here—we'll see some—but one can also go very far down the rabbit hole of the "analysis" in "numerical analysis".

Some ingredients which will hopefully be useful if your career intersects with computer simulation (and so many do!)

Start of training into developing your own methods for new models.

- Often best approach here is reuse of what exists (sometimes software, sometimes algorithms) and invent only the novel new bits.
- As computers get bigger, we can solve the same problems faster but no wants to solve the same problems. Science/engineering/practice always driving bigger problems (or wrapping the old problems up in tight loops).
- Scientific computing problems are usually too big and too complex to repeatedly re-invent the wheel. And the importance of good algorithms increases as problem size increases. "Big Oh is very patient".

This course, compressed into one short demo

Let's do derivation, the algorithms, and a from-scratch software implementation for (approximately) solving the heat equation PDE...

Next topics: root-finding

Bisection and Newton's method.

Convergence rates: - linearly - superlinear - quadratic

Newton for systems.

Intro to optimization.