Course outline for

Open topics in applied mathematics:

Fast Methods in Scientific Computation

Review of background material

- Basic techniques from numerical analysis such as quadrature rules for evaluating integrals, and the FFT for evaluating Fourier transforms.
- How do you measure the performance of a numerical algorithm? How does its execution speed depend on problem size, on requested accuracy, on the number of processors available, etc.? How do you estimate these metrics in practice?
- Methods for improving the execution speed of programs in Matlab. Good programming habits; backups; how to structure a large coding project, etc.

The Laplace and Poisson equations — basic solution strategies

The Laplace and Poisson equations are the prototype equations for the large class of so called *elliptic equations*. Many (but not all) of the basic techniques and results for these equations generalize straight-forwardly to the broader class. In this class, we will for the most part talk about Laplace and Poisson, and then briefly review how other equations (Maxwell, elasticity, Stokes, etc.) are different. This section of the class will review some basic results for Laplace/Poisson:

- Derivation and motivation.
- Classical analytic methods for solving the Laplace and Poisson equations (fundamental solutions, Green's functions, separation of variables, Fourier methods, etc). How can you use such methods for computationally constructing solutions?
- Very brief review of finite difference, finite element, and FFT based methods. Introduction to data structures for sparse matrices and iterative solvers.

The Fast Multipole Method and other hierarchical codes

The Fast Multipole Method (FMM) is a technique for rapidly evaluating a sum such as

$$u_i = \sum_{j=1}^{N} K(\boldsymbol{x}_i,\, x_j)\, q_j \tag{1}$$

where

$$\begin{aligned} \{\boldsymbol{x}_i\}_{i=1}^N &\quad \text{is a set of particle locations, } \boldsymbol{x}_i \in \mathbb{R}^d, \\ \{q_i\}_{i=1}^N &\quad \text{is a set of charges, } q_i \in \mathbb{R} \text{ or } q_i \in \mathbb{C}, \\ \{u_i\}_{i=1}^N &\quad \text{is a set of potentials, } u_i \in \mathbb{R} \text{ or } u_i \in \mathbb{C}. \end{aligned}$$

and where $K(x, y)$ is some given *kernel function*. For instance, if

$$K(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{4\,\pi\,|\boldsymbol{x} - \boldsymbol{y}|},$$

then the sum (1) corresponds to the evaluation of the electric potentials u_i caused by N electric charges q_i placed at points $\boldsymbol{x}_i$.

Sums such as (1) arise directly in a broad range of applications (astrophysics, biochemical models, models of semi-conductors, etc.), and also arise indirectly in many fast methods for solving PDEs.

Naïve evaluation of the sum (1) would require $O(N^2)$ operations (since there are $N(N-1)/2$ distinct pairwise interactions). The FMM is a tool for computing an approximation to the sum in $O(N)$ operations. Any given accuracy can be obtained, with the caveat that the higher the accuracy you want, the larger the constant of proportionality in the $O(N)$ expression.

The class will describe the FMM in some depth; in part because it is a very powerful tool in its own right, and in part because it nicely illustrates the broader concept of algorithms based on hierarchical domain partitioning.

Boundary Integral Equations

That partial differential equations are useful for modeling a very broad range of problems in science and engineering is very well known. What is less well appreciated is that it is in many situations possible and advantageous to rewrite a differential equation as an integral equation. As an example, consider the Laplace equation with Dirichlet boundary data on a domain Ω with boundary Γ:

$$\begin{cases} -\Delta\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ \quad u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases} \tag{2}$$

Equation (2) is in a strong sense equivalent to the *integral equation*

$$-\frac{1}{2}\sigma(x) + \int_\Gamma \frac{\boldsymbol{n}(\boldsymbol{y}) \cdot (\boldsymbol{x} - \boldsymbol{y})}{2\,\pi\,|\boldsymbol{x} - \boldsymbol{y}|^2}\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma, \tag{3}$$

where for a point $\boldsymbol{y} \in \Gamma$, the vector $\boldsymbol{n}(\boldsymbol{y})$ is the unit normal vector. to Γ. The integral equation formulation (3) has many advantages over the differential equation formulation (2):

- The dimensionality of (3) is lower. If Ω is a domain in $\mathbb{R}^2$, then (2) is an equation on a two-dimensional domain, and (3) is an equation on a one-dimensional domain.
- The equation (3) is "nicer" from a mathematical point of view; it involves a bounded operator, it is a "second kind Fredholm equation", etc.

A drawback of working with integral equations is that they typically lead to dense linear systems upon discretization.

The class will cover the following topics:

- Describe when integral equation formulations are possible, and how to derive them.
- Describe efficient solution strategies for solving integral equations.
- Discuss advantages and disadvantages of using integral equations.

Iterative solvers

Very often when solving PDEs numerically one needs to find a vector $\boldsymbol{x}$ that solves a linear system of the form

$$\mathsf{A}\,\boldsymbol{x} = \boldsymbol{b}, \tag{4}$$

where A is a given $N \times N$ matrix, and $\boldsymbol{b}$ is a given data function. When N is of moderate size (say a few thousands at most), one can use elementary techniques such Gaussian elimination. However, these require $O(N^3)$ floating point operations, and quickly become too expensive. The class will describe some *iterative* solvers that constructs a sequence of successively better approximate solutions to (4) by applying A to a sequence of vectors. When A can be applied rapidly (for instance because it is sparse, or is amenable to a fast methods such as the FMM), iterative methods can be highly effective, often solving the equation in $O(N)$ operations.

The depth of the discussion will be based on student preferences. (If many students already know this material, we will do only a very basic review.)

Extensions to other equations

In this part of the class, we will conduct a brief review of some of the standard PDEs encountered in applications, for instance:

- Helmholtz' equation.
- Maxwell's equation.
- The equations of linear elasticity.
- The Navier-Stokes', and the Stokes' equations.

We will describe how each equation arises, discuss its characteristic mathematical properties, and outline how it can be solved efficiently.

Computing in the real world

At the end of the class, we will discuss what simplifying assumptions we made, and what additional challenges are encountered when the techniques described are to be used in practice. For instance, most of the class will focus on 2D problems, whereas of course almost all engineering applications involve 3D problems. How much harder is 3D than 2D? What are the major differences? Other practical questions involve:

- When should you use a better method, and when should you simply buy a faster computer? (Human time is very expensive. . .)
- Practical problems often involve many different kinds of physics at once. How do you handle that?
- Techniques for representing geometries, etc.