Lecture 1: Sci. Comp. for DPhil Stduents

Nick Trefethen, Tuesday 16.10.18

Today

- Mechanics of this course
- I.1 View of the field
- I.2 How fast can we solve $Ax = b$?
- I.3 Sparse matrices

Handouts

I generate lots of handouts. Buy a notebook!

- Flier
- Fact sheet
- Outline of lectures
- Quiz 1: abbreviations
- Assignment 1, due next Tuesday at 11:00
- p. 1 of Gilbert-Moler-Schreiber and Tim Davis web page
- Sparse matrix sec. from Higham & Higham Matlab book, all on one page
- m1_howfast.m and m2_sparsity.m
- Quiz 1 solutions

Welcome

Numerical computation is a universal subject these days, and perhaps the biggest academic subfield of mathematics. (The UK has about 1000 academic mathematicians, and about 100 of them work in numerical analysis.) I want to expose you to this highly developed field, to acquaint you with its high expectations, to get you in the habit of interacting intimately with numbers.

This course aims to be *interesting* and *useful*.

I want you to see a wider picture than you are used to from your own work — which, in the long run, will strengthen your work.

MATLAB is a beautiful tool, for it enables us to explore all kinds of computational ideas and *do* things rather than just talk about them. Its use is nearly universal among experts in this field.

Please do the quiz now; answers will be handed out at the end.

Items to note from flier

- Which departments do you all come from?
- Linear algebra and optimization this term; ODE and PDE next term
- Course grades at the end, reported to you and your supervisor
- First assignment counts just 5%

N.B.

- This week only: second lecture Friday 12:00
- No assignments accepted after 11:00 on due dates
- We'll record your names and addresses on Friday
- Frequent the course Web site! `https://www.maths.ox.ac.uk/courses/course/[TBA]`

I. Sparse matrices and iterative methods

I.1 View of the field

Science and engineering are formulated mathematically; the solutions are usually numerical on the computer.

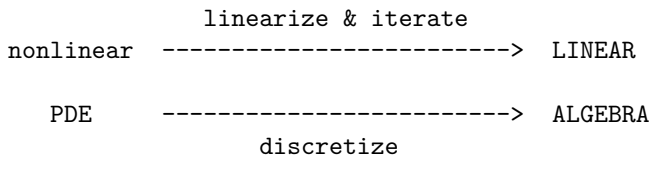
```
                    NUMERICAL ANALYSIS
                            =
                  SCIENTIFIC COMPUTING


       LINEAR ALGEBRA          ODE & PDE      OPTIMIZATION


                                                    OTHER
                                                approximation
systems     least-      eigs &                    data-fitting
of eqs      squares     SVD                        integration
                                                       FFT
                                                   random nos.
                                                       etc.

  dense/classical
      v.
  sparse/large-scale/iterative/parallel
      v.
  randomized/very large-scale/data science
```

Over and over again we see a pattern like this:



Because of this, computers have brought linear algebra, and numerical linear algebra, to the forefront of the mathematical sciences.

You may think your particular computational needs lie in other directions. Stick around; you may be surprised.

I.2 How fast can we solve $Ax = b$?

$Ax = b$ system of N linear equations in N unknowns

$A = N$xN matrix $b = N$x1 right-hand side $x = N$x1 vector of unknowns

How big a dimension N can we handle?

Here's a quote from Wilkinson in 1951:

> By 1949 the major components of the Pilot ACE were complete and undergoing trials.... During 1951 a program for solving simultaneous linear algebraic equations was used for the first time. 26th June, 1951 was a landmark in the history of the machine, for on that day it first rivalled alternative computing methods by yielding by 3pm the solution to a set of 17 equations submitted the same morning.

"Big" values of N at various years:

```
1950: N = 20
1965: N = 200
1980: N = 2000
1995: N = 20000
2010: N = 200000
```

Increase in N reflected here: factor of 10^4

Increase in speed of computers over the same period: factor of 10^{12}, from flops to teraflops

Google Top500 to see the very fastest machines—now in the hundreds of petaflops range (100 times 10^{15}) and heading for exaflops (10^{18}).
Even the 500th fastest machine is around a petaflop.

In the formula $10^{12} = (10^4)^3$ we see played out in history the fact that standard algorithms take

```
O(N^3) flops  (floating point operations)
```

There are "fast" algorithms that need as few as $O(N^{2.373})$ flops, but these are of theoretical use only, so far, no good for practical values of N. If you can find an $O(N^2)$ algorithm, you will be world-famous.

Now let's see what we can do in MATLAB. `[m1_howfast.m]`

I.3 Sparse matrices

Why do we want N to be so large? Because discretizations of integral and differential equations force this upon us. (Also for data science problems, but we won't focus on that in this course.) Here e.g. is a 3×3 grid of unknowns:

```
7 - 8 - 9
|   |   |
4 - 5 - 6
|   |   |
1 - 2 - 3
```

If the variables are coupled as indicated we get a sparse 9x9 matrix:

```
      | x x   x           |
      | x x x   x         |
      |   x x     x       |
      | x     x x   x     |
A =   |   x   x x x   x   |
      |     x   x x     x |
      |       x     x x   |
      |         x   x x x |
      |           x   x x |
```

A $100 \times 100 \times 100$ grid would lead to a sparse matrix of dimension 10^6. A matrix is "sparse" if most of its entries are zero... or more precisely, if enough of its entries are zero for this to be taken advantage of !

There are two ways to beat the $O(N^3)$ bottleneck in practice:

1. parallel computing, and/or
2. special algorithms that take advantage of sparsity or other special structure of A: (a) direct (b) iterative

In recent years flop count is less and less important at the high end (i.e. for many processors) — communication is a bigger bottleneck.

Handouts

- Gilbert-Moler-Schreiber / UMFPACK and Tim Davis
- sparse matrix chapter from Higham + Higham

Let's explore sparse matrices in MATLAB. `[m2_sparsity.m]`