October 12, 2016

Finish up Least Squares
Iterative & Conditioning

In this class:

- *What you need to know about iterative methods for your homework. (We'll*
- *How we can use QR to solve a least squares problem.*
- *How any computation can be wrong on a computer: ill-conditioned problems and unstable algorithms.*

Next class

More Conditioning & Iterative methods!
G&C – Chapter 7.4, Chapter 12.2

Next next class

Eigenvalues
G&C – Chapter 12.1

How to solve Ax=b another way.

When A is super-large (1 million-by-1 million) then using GE and LU often don't work.

But these large matrices are often very special (see handout).

- They are sparse, which means they have many zeros and computing y = Ax is fast and easy
- They are structured, which means computing y = Ax is fast

Think about the matrices we saw for resizing images on the homework. (32x32 -> 16x16) there were tons of zeros!

The key ideas (1)

If we have a big matrix, but we also have a *program* to compute $y = Ax$ (matvec) then we can still solve $Ax = b$!

- We can check a potential solution y via the quantity $r = b - A y$ (matvec and subtract) and $\| r \|$
- r is called the residual and
- $\| r \| / \| b \|$ is the relative residual

If relative residual is small (10^{-8}), then we have a *good enough* solution,

=> so we can tell when to stop

The key ideas (2)

If we have a big matrix, but we also have a *program* to compute y = Ax (matvec) then we can still solve Ax = b!

- There are a variety of ways to turn $x = A^{-1} b$ into a sequence of matvecs.
- The easiest is

$$\boldsymbol{A}^{-1}\mathbf{b} = \mathbf{b} + (\boldsymbol{I} - \boldsymbol{A})\mathbf{b} + (\boldsymbol{I} - \boldsymbol{A})^2\mathbf{b} + (\boldsymbol{I} - \boldsymbol{A})^3\mathbf{b} + \ldots$$

  which doesn't always work, but will for our cases.
- Evaluating k terms here only involves matvecs with A!

The overall idea

If we have a big matrix, but we also have a *program* to compute y = Ax (matvec) then we can still solve Ax = b!

- Use

$$\boldsymbol{A}^{-1}\mathbf{b} = \mathbf{b} + (\boldsymbol{I} - \boldsymbol{A})\mathbf{b} + (\boldsymbol{I} - \boldsymbol{A})^2\mathbf{b} + (\boldsymbol{I} - \boldsymbol{A})^3\mathbf{b} + \ldots$$

- Evaluate k terms, check relative residual.
- If not small enough, evaluate the next term and repeat.

This is an *iteration* and hence *iterative methods*

The overall idea simplified

We'll work through this in class, but this idea is actually super simple once you work out the a few other facts.

```
function richardson(A,b;tol=10⁻⁸,niter=10000,omega=1.)
x = b
normb = norm(b)
for k=1:niter
  r = b – A*x
  if norm(r)/norm(b) <= tol, break, end
  x = x + omega*r
end
return x
```