Math 5600 6/24/14

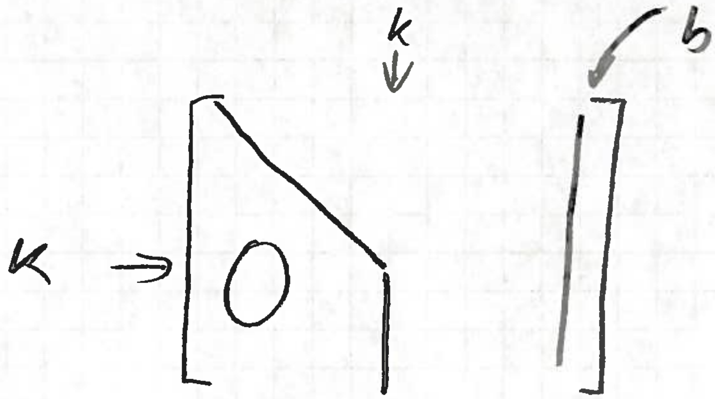
- How to solve $Ax = b$?

  A square, $n \times n$, given, the "coefficient matrix"

  $b \in \mathbb{R}^n$ given, the "right hand side"
  (even if written as $b = Ax$)

  $x = ?$ the solution. We want to compute it quickly and accurately

- College Algebra: augment the matrix, use elementary row operations (multiply rows with constants, interchange rows, add multiples of rows to other rows) to reduce the system to upper triangular form, then use backward substitution.



$a_{i,n+1} = b_i$

For $k =$

For $i =$

$m_{ik} =$

For $j =$

- what can go wrong?
- must not divide by zero.
- should not divide by something close to zero
  - partial pivoting: interchange rows
    - unscaled find largest pivot in k-th column
    - scaled: find pivot in k-th column that is the largest relative to other entries in that row
  - total pivoting: interchang rows and columns.

Backward substitution

$$x_n = \frac{a_{n,n+1}}{a_{nn}}$$

$$x_i = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}$$

$i = n-1, n-2, \ldots, 1$

- what if we have several right hand sides?
- Could augment the matrix with more than one column.
- But what if we get the new right hand side only after we solve the linear system!
- Example: a version of Newton's method where we keep the Jacobian constant, to save time.
-

Gaussian Elimination

For $k =$

For $i =$

For $j =$

+ pivoting!

- Here is another idea. What if we factored

$$A = LU$$

L: lower triangular
with 1s along the diagonal
("unit lower triangular")

U: upper triangular

$Ax = b \rightarrow$ $Ly = b$ Forward Subst.

$Ux = y$ Backward Subst.

Gaussian Elimination computes the LU factorization.
Argument with elementary matrices is tedious.
Following simple argument due to Gil Strang.
Consider 4×4, ignore pivoting

$$\begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ m_{21} & \times & \times & \times \\ m_{31} & \times & \times & \times \\ m_{41} & \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ m_{21} & \otimes & \otimes & \otimes \\ m_{31} & m_{32} & \times & \times \\ m_{41} & m_{42} & \times & \times \end{bmatrix} \to \begin{bmatrix} \otimes & \otimes & \otimes & \otimes \\ m_{21} & \otimes & \otimes & \otimes \\ m_{31} & m_{32} & \otimes & \otimes \\ m_{41} & m_{42} & m_{43} & \otimes \end{bmatrix}$$

$r_i = \text{row } i$

$$r_3(U) = r_3(A) - m_{31}\, r_1(U) - m_{32}\, r_2(U)$$

$$r_3(A) = r_3(U) + m_{31}\, r_1(U) + m_{32}\, r_2(U)$$

This is just what we get when multiplying $\begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{bmatrix}$ with U

Pivoting

- A pivot matrix is one that is all zero except that it has one entry 1 in each row and column.

- $$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix}$$

- multiplying with a pivot matrix from the left interchanges rows, multiplying from the right interchanges columns.

- so we factor $A = PLU$

- The idea to express algorithms as matrix factorizations is central in Numerical Linear Algebra.

- what about computing the inverse?

$$Ax = b \iff x = A^{-1} b$$

- 1 flop = 1 mult/div + 1 add/subtract

$A = LU$ $\quad \frac{n^3}{3} + O(n^2)$

$Ly = b$ $\quad \frac{n^2}{2} - \frac{n}{2}$

$Ux = y$ $\quad \frac{n^2}{2} + \frac{n}{2}$

$\left.\begin{matrix} \frac{n^2}{2} - \frac{n}{2} \\ \frac{n^2}{2} + \frac{n}{2} \end{matrix}\right\} n^2$

$A^{-1}b$ $\quad n^2$

A^{-1} $\quad n^3 + O(n^2)$

- Forward and Backward Substitution take exactly as much effort as multiplying with A^{-1}

- To compute A^{-1} we first compute LU, getting A^{-1} can only make errors worse

- A^{-1} is more expensive than LU

- most important and significant: the inverse of a sparse matrix is usually not sparse.