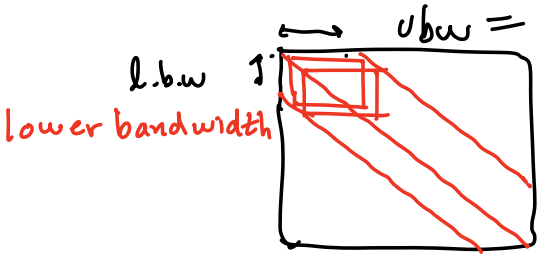
Welcome to Ma221! (Feb 17)

Sparse Matrices

Band matrices



Case: without pivoting: each step of GE costs lbw · ubw +, *s

$\Rightarrow$ Cost $= 2n \cdot ubw \cdot lbw + n \cdot lbw$

$= O(n)$ for small bandwidth

Case: with pivoting:

ubw(U) = ubw(A) + lbw(A)

"lbw(L)" = lbw(A) but not all adjacent nonzeroes in each column of L

Band matrices arise from discretizing D.E.s

each unknown only depends on nearest neighbors $\Rightarrow$ banded

Ex: Sturm-Liouville problem

$$-y''(x) + q(x)\cdot y(x) = r(x) \qquad x \in [0,1]$$

$$y(0)=\alpha,\ y(1)=\beta \qquad q(x) \geq \bar{q} > 0$$

discretize at $x(i) = ih$ $\quad h = \frac{1}{N+1}$

unknowns $y(i) = y(x(i))$ for $i = 1, \ldots, N$

approximate: $y''(i) \simeq \dfrac{y(i+1) - 2y(i) + y(i-1)}{h^2}$

$q(i) = q(x(i)) \quad r(i) = r(x(i))$

$$-\frac{y(i+1) - 2y(i) + y(i-1)}{h^2} + q(i)\cdot y(i) = r(i)$$

$i = 1$ to N

or $Ay = v$ for

$$v = r + \left[\frac{\alpha}{h^2}, 0, \ldots, 0, \frac{\beta}{h^2}\right]^T$$

$A = \text{diag}\left(\frac{2}{h^2} + q(i)\right)$

$+\text{diag}\left(\frac{-1}{h^2}, 1\right)$ above diagonal

$+\text{diag}\left(\frac{-1}{h^2}, -1\right)$ below diagonal

Prove A spd to use Cholesky

Gershgorin's Thm: All evals of A lie in n circles in complex plane,

circle i has center A_{ii}
and radius $\sum_{\substack{j=1 \\ j\neq i}}^{n} |A(i,j)|$

proof: $Ax=\lambda x$, $|x(i)|$ largest entry

$$(A(i,i)-\lambda)x(i) = \sum_{\substack{j=1 \\ j\neq i}}^{n} A(i,j)\,x(j)$$

$$|A(i,i)-\lambda| \le \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|A(i,j)\frac{x(j)}{x(i)}\right| \le \sum_{\substack{j=1 \\ j\neq i}}^{n} |A(i,j)|$$

Apply Gershgorin to Sturm-Liouville

circles are centered at $\frac{2}{h^2}+q(i)$

with radii $\frac{2}{h^2}$, $q(i) \ge \bar{q} > 0$

$\Rightarrow$ all eigenvalue positive (and real)

$\Rightarrow$ spd.

Ex: Poisson's Eqn in 2D

(A) $$\frac{\partial^2 v(x,y)}{\partial x^2} + \frac{\partial^2 v(x,y)}{\partial y^2} + q(x,y)v(x,y) = r(x,y)$$

in square $x, y \in [0,1]$

discretize on 2D mesh

$$[x(i), y(j)] = [ih, jh], \quad h = \frac{1}{N+1}$$

Simple case: $q = r = 0$

(A) approximated by

$$v(i-1,j) - 2v(i,j) + v(i+1,j)$$
$$+ v(i,j-1) - 2v(i,j) + v(i,j+1)$$
$$= \sum 4 \text{ neighbors} \quad -4 \cdot v(i,j)$$

General Sparse Matrices

Importance of ordering equations

what is best permutation?

PLU $P_r L U P_c$ $P_r L L^T P_r^T$

Graph Theory:

Def: A weighted, undirected graph G is a collection of 3 set V, E, W

V = vertices (aka nodes)

E = edges connecting pairs of vertices (v, w), undirected means same as (w, v)

(v, v) allowed

W = weight (number) for each edge

Relationship to Sparse matrices!

V = one row/col per vertex

E = locations of nonzeros

(u, v) means $A_{u,v}$ nonzero

undirected → symmetric

(u, u) → diagonal

W = values of nonzeros

To build a matrix from a graph, need to order vertices

lots of interesting choices (see Fig 2.9 - 2.11 in text)

---

data structures for sparse matrices

Goal: store and operate only on nonzeros

Simplest: COO: Coordinate format

List of all nonzeros and locations

$$A = \begin{bmatrix} 2 & 0 & 7 & 0 & 5 \\ 0 & 1 & 4 & 0 & 3 \\ 0 & 0 & 8 & 0 & 0 \end{bmatrix}$$

COO(A) = ((2,1,1), (7,1,3), (5,1,5), (1,2,2) ... (8,3,3))

Better: CSR Compressed sparse row:

val: array of nonzeros in each row, from row 1 to n, left to right

col_index: array of columns in which each nonzero in val lives

row_begin = pointer to start of each row

entries of row i live in

rowbegin(i) to rowbegin(i+1) − 1

val = (2, 7, 5, 1, 4, 3, 8), colindex = (1, 3, 5, 2, 3, 5, 3)

row_begin = (1, 4, 7, 8) (also need m, n)

saves $\frac{1}{3}$ memory vs COO

Analogous CSC = compressed sparse column

more details in Chap 6

How do we pick best order of rows & columns?

Goals: minimize time + memory, and be backward stable

Ordering only affects stability of LU, not Cholesky

"Easy" case: start with Cholesky

Thm: picking best order to minimize work (out of n! possibilities) is NP-hard, exponential cost

⇒ Need heuristics

RCM = Reverse Cuthill-McKee
= Breadth-First Search, reverse order

Def: A path in graph from v_1 to v_k is a set of edges $(v_1, v_2), (v_2, v_3) \ldots (v_{k-1}, v_k)$

Def: Distance between any 2 vertices in Graph = length of shortest path connecting them

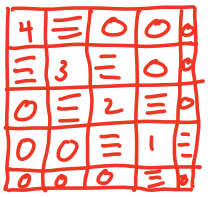
RCM: ① pick any starting vertex, call it root

② compute distance from root to all other vertices
distance(root) = 0
distance = 1 for all nearest neighbors of root
distance to all nearest neighbors of vertices at distance k (that you haven't visted already) = k+1
cost = O(# nonzeros)

③ order vertices in reverse order visited

Fact: vertices at distance k from from root can only be connected to vertices at distance $k-1$ or $k+1$

$\Rightarrow$ matrix block tridiagonal



label = distance from root

Matlab: symrcm

MD: minimum degree

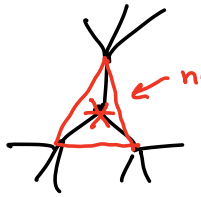
Def: degree of vertex = # edges it touches

Fact: If I pick vertex v as pivot, v has degree d, $\Rightarrow$ d nonzeroes in its row + column

$\Rightarrow d^2$ multiplies + adds to take one step of Cholesky/LU

$\Rightarrow$ fill in at most d^2

Greedy Algorithm: at each step, pick vertex of minimum degree, (update degrees after each step)



Matlab: amd, symamd, colamd

Next: Nested Dissection