Welcome to Ma 221! Lecture 15, Sep 27

Last time: Sturm-Liouville $\to$ tridiagonal

Poisson's Eqn (Elliptic PDEs)

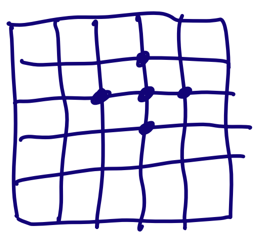
$$(*)\quad \frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} + q(x,y)\cdot u(x,y) = r(x,y)$$

on square $[x,y] \in [0,1]^2$

discretize as before: 2D mesh

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

approx $(*)$ as before $u(i-1,j) - 2u(i,j) + u(i+1,j)$
$+ u(i,j-1) - 2u(i,j) + u(i,j+1)$



16 unknowns 4x4 mesh

General Sparse Matrices:

Importance of choosing order of rows + cols to do LU, or Cholesky

possibilities: $A = PLU$, $P_r L U P_c$, $PLL^T P^T$

Best case choice of P (or P_r and P_c)

cost can drop from $O(n^3) \to O(n)$

memory " " " $O(n^2) \to O(n)$

Choosing Best P or $P_r + P_c$: NP-complete
exponential cost

$\Rightarrow$ use heuristics

Graph Theory:

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

V = vertices (aka nodes)

E = edges connecting pairs of vertices (u,v), undirected means $(u,v) = (v,u)$

(u,u) allowed

W = weight (number) for each edge

V = one row/col per vertex

E = locations of nonzeros

$(u,v) \in E \Rightarrow A(u,v)$ nonzero

undirected $\Rightarrow$ symmetric

$(u,u) \Leftrightarrow$ diagonal

W = values of nonzeros

To choose perm P given matrix, build graph, choose order for vertices permute matrix accordingly

(example: 2D mesh of bridge)

Data Structures for sparse Matrices

Goal: only store and compute on nonzeros
(ignore cancellation)

Simplest: COO: Coordinate Format

List of all nonzeroes + locations

$$A = \begin{bmatrix} 2 & 0 & 7 & 0 & 5 \\ 0 & 1 & 4 & 0 & 3 \\ 0 & 0 & 8 & 0 & 0 \end{bmatrix} \qquad COO(A) = ((2,1,1),(7,1,3),(5,1,5),(1,2,2)\ldots)$$

any order of entries legal

Better: CSR Compressed Sparse Row: 3 arrays

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

val: [2, 7, 5, 1, 4, 3, 8]

col_index: array of columns in which each nonzero lies

col_index = [1, 3, 5, 2, 3, 5, 3]

row_begin = pointers to start of each row

= [1, 4, 7, 8] one past end

saves up to $\sim\frac{1}{3}$ memory vs COO

Analogous: CSC = compressed sparse column

How to pick best (cheapest) order of rows, cols for LU?

Goals: minimize time + memory

Ordering impacts stability of GE,
not Cholesky: start with "easy" case:
Cholesky (don't worry about stability)

Thm: Still NP-complete, still need heuristics

RCM = Reverse Cuthill-McKee
= Breadth-First-Search, backwards

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

Def: Distance from v_1 to v_k is length (# edges) in shortest path.

RCM: ① pick any starting vertex, call it root

② compute distances from root to all other $v \in V$, $dist(v)$, $dist(root) = 0$

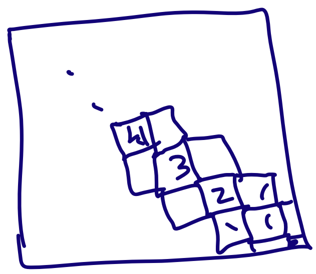
distance from root to all its immediate neighbors is 1, etc

③ order vertices in reverse of order visted

cost = O(# nonzeros)

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

$\Rightarrow$ matrix block tridiagonal



Matlab: symrcm

Next time: MD = minimum Degree

ND = nested dissection