Welcome back to Ma221! Lecture 34, Nov 13

Recall Poisson Equation

1D: $T_N = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} = Z \Lambda Z^T$

2D: $T_N V + V T_N = F$, solve for $N \times N$ V

turn V into vector $\mathrm{vec}(V)$

$(I_N \otimes T_N + T_N \otimes I_N)\, \mathrm{vec}(V) = \mathrm{vec}(F)$

$\to (Z \otimes Z)(I_N \otimes \Lambda + \Lambda \otimes I_N)(Z \otimes Z)^T$

diagonal, all pairs $\lambda_i + \lambda_j$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$

cond # $= \dfrac{2 \cdot \lambda_N}{2 \cdot \lambda_1}$ same as for T_N

3D: $(T_N \otimes I_N \otimes I_N + I_N \otimes T_N \otimes I_N + I_N \otimes I_N \otimes T_N)$

$= (Z \otimes Z \otimes Z)(\Lambda \otimes I_N \otimes I_N + I_N \otimes \Lambda \otimes I_N + I_N \otimes I_N \otimes \Lambda)$
$\cdot (Z \otimes Z \otimes Z)^T$

diagonal, all triples $\lambda_i + \lambda_j + \lambda_k$

same condition number

Summary of Performance of all Algorithms on Poisson in 2D and 3D

Count #flops (all in $O(\cdot)$ sense)

Memory needed (#words)

Parallel steps on "perfect" parallel
computers with as many processors
as needed

Processors needed
(see CS267 for practical parallel algs)

Entries sorted in 2 orders

from slowest to fastest for Poisson
(roughly) from most general, to
most specific for Poisson

Some Terminology:

"Explicit Inverse" = we compute and
store A^{-1} ahead of time, don't count
cost of A^{-1}, just multiplying by it

SOR: Successive Overrelaxation

SSOR/Chebyshev ≡ Symmetric SOR
with Chebyshev acceleration

FFT = Fast Fourier Transform

BCR = Block Cyclic Reduction

Lower Bound = assume 1 flop per component
of answer

SpMV = Sparse-Matrix-Vector-Multiplication
cost(SpMV) = # nonzeros

For 2D Poisson on $n \times n$ mesh
$N = n^2 = \#$ unknowns

3D Poisson on $n \times n \times n$ mesh
$N = n^3 = \#$ unknowns

If table entry for 2D and 3D different, then 3D in parentheses

All entries in $O(\cdot)$ sense

Method	direct or iterative	#flops	Mem	#Parallel steps	#Procs
Dense Cholesky any spd. matrix	D	N^3	N^2	N	N^2
Explicit Inverse any matrix	D	N^2	N^2	$\log N$	N^2
Band Cholesky	D	N^2 $(N^{7/3})$	$N^{3/2}$ $(N^{5/3})$	N N	N $(N^{4/3})$

works on any band s.p.d. matrix, cost $= O(N \cdot bw^2)$

2D: $bw = n = N^{1/2}$
3D: $bw = n^2 = N^{2/3}$

Method	direct or iterative	#flops	Mem	#Parallel steps	#Procs
Jacobi	I	N^2 $(N^{5/3})$	N N	N $(N^{2/3})$	N N

$$
\begin{aligned}
\#\text{flops} &= O(\#\text{flops\_per\_iteration} \cdot \#\text{iterations}) \\
&= O(\#\text{flops}(SpMV) \cdot \#\text{iterations}) \\
&= O(\text{nnz}(A) \cdot \#\text{iterations}) \\
&= O(\text{nnz}(A) \cdot \text{cond}(A)) \text{ for Poisson}
\end{aligned}
$$

works for any diagonally dominant matrix

Gauss-Seidel	I	N^2 $(N^{5/3})$	N N	N $(N^{2/3})$	N N

cost analysis like Jacobi, better constants

Works for any diagonally dominant or s.p.d. matrix

Sparse Cholesky	D	$N^{3/2}$ (N^2)	$N \cdot \log N$ $(N^{4/3})$	$N^{1/2}$ $(N^{2/3})$	N $(N^{4/3})$

assumes we use best reordering of rows and columns (nested dissection)

$\Rightarrow$ bottleneck is dense Cholesky on trailing dense submatrix of size $n \times n$ in 2D, $n^2 \times n^2$ in 3D

Conjugate Gradients (CG)	I	$N^{3/2}$ $(N^{4/3})$	N N	$N^{1/2} \cdot \log N$ $(N^{1/3} \log N)$	N N

$$\#\text{flops} = O(\#\text{flops\_per\_iteration} \cdot \#\text{iterations})$$
$$= O\left(\#\text{flops}(\text{SpMV}) \cdot \sqrt{\text{cond}(A)}\right)$$

works on any s.p.d. matrix

SOR	I	$N^{3/2}$ $(N^{4/3})$	N N	$N^{1/2}$ $(N^{1/3})$	N N

works on any s.p.d. matrix

\# steps depends on cond(A)

analogous cost analysis to CG

SSOR/Chebyshev	I	$N^{5/4}$	N	$N^{1/4}$	N
		$(N^{7/6})$	N	$(N^{1/6})$	N
		$\#flops = O(\text{cost}(SpMV) \cdot (\text{cond}(A))^{1/4})$ need to know λ_{max} and λ_{min}, works for Poisson, any s.p.d. matrix			
FFT	D	$N \cdot \log N$	N	$\log N$	N
		works for Poisson			
BCR	D	$N \cdot \log N$	N	?	?
		slightly more general than FFT			
Multigrid	I	N	N	$\log^2 N$	N
		many variants beyond Poisson, used for FEM, Elliptic PDEs etc			
Lower Bound		N	N	$\log N$	