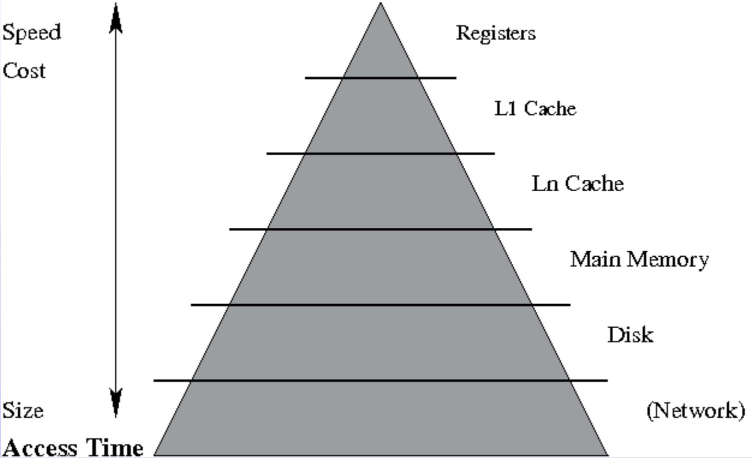
Numerical Methods for Solving Large Linear Systems

Paolo Bientinesi

AICES, RWTH Aachen
pauldj@aices.rwth-aachen.de

3rd LHC Detector Alignment Workshop
June 15-16, 2009
CERN, Switzerland



Cache misses. Data movement. Storage by row/columns. Data locality.

Modularity and Data Movement

Linear Algebra operations decomposed into simpler operations.

Linear Algebra operations decomposed into simpler operations.

BLAS-1:	$y := y + \alpha x$	$x, y \in \mathbb{R}^n$
	$dot := \alpha + x^T y$	
BLAS-2:	$y := y + Ax$	$L \in R^{n\times n},\ x, y \in R^n$
	$y := A^{-1} x$	$L \in R^{n\times n} \wedge$ triangular
BLAS-3:	$C := C + AB$	$A, B, C \in R^{n\times n}$
	$C := L^{-1} B$	$L \in R^{n\times n} \wedge$ triangular

Linear Algebra operations decomposed into simpler operations.

BLAS-1:	$y := y + \alpha x$	$x, y \in \mathbb{R}^n$
	$dot := \alpha + x^T y$	
BLAS-2:	$y := y + Ax$	$L \in R^{n \times n},\ x, y \in R^n$
	$y := A^{-1} x$	$L \in R^{n \times n} \wedge$ triangular
BLAS-3:	$C := C + AB$	$A, B, C \in R^{n \times n}$
	$C := L^{-1} B$	$L \in R^{n \times n} \wedge$ triangular

BLAS	#FLOPS	Mem. refs.	Ratio	Proc. use
Level 1	$2n$	$3n$	$\mathbf{2/3}$	low
Level 2	$2n^2$	n^2	$\mathbf{2}$	medium-low
Level 3	$2n^3$	$4n^2$	$\boldsymbol{n/2}$	very high

1. High Performance Computing
2. Sparse Matrices
3. Linear Systems
4. Error Analysis
5. Eigensolvers

What is a Sparse Matrix?

- Sparse matrix: concept of convenience.
- No formal definition in terms of number of non-zeros, patterns, properties.
- Practical definition in terms of cost:
  operation count, storage requirement, ...

What is a Sparse Matrix?

- Sparse matrix: concept of convenience.
- No formal definition in terms of number of non-zeros, patterns, properties.
- Practical definition in terms of cost:
  operation count, storage requirement, . . .

A matrix is sparse when is has enough zeros that pays off to exploit them (Wilkinson)

Objectives:

- Storage space.
- Accessing, inserting matrix elements.
- Matrix operations and fill in.

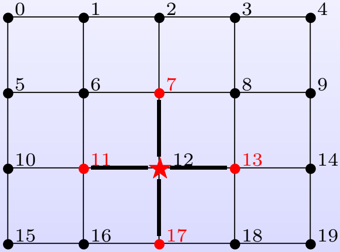
Sparse Matrices — Structured Matrices — Dense Matrices

$$
\begin{bmatrix}
\star & & & & & \star & \\
& \star & \star & & \star & & \\
& \star & \star & & & & \star \\
& & & \star & & & \star \\
& \star & & & \star & & \\
\star & & & & & \star & \\
& & \star & \star & & & \star
\end{bmatrix}
\qquad
\begin{bmatrix}
\star & \star & & & & & \\
\star & \star & \star & & & & \\
& \star & \star & \star & & & \\
& & \star & \star & \star & & \\
& & & \star & \star & \star & \\
& & & & \star & \star & \star \\
& & & & & \star & \star
\end{bmatrix}
$$

- Structured matrices:
  bidiagonal, tridiagonal, banded, blocked, . . .

  Small number of non-zeros (NNZ), but **known** structure!

Sparsity:
interactions between particles, components, neighbors, degrees of freedom.



The finer the discretization, the higher the sparsity.

- 50% of NNZ $\rightarrow$ NOT a sparse matrix.
- 10% of NNZ: if $n = 100.000$, then 10.000 interations per row. Still not very sparse.
- Large sparse matrices: NNZ $\ll 1\%$.

Linear Systems

Direct methods: LU factorization, Cholesky, ...

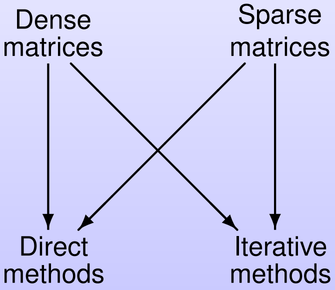
Iterative methods: Gauss-Seidel, Conjugate Gradient, GMRES, ...

Linear Systems

Direct methods: LU factorization, Cholesky, ...

Iterative methods: Gauss-Seidel, Conjugate Gradient, GMRES, ...

Dense vs. Sparse (vs. Structured)



Direct Methods vs. Iterative Methods

Direct	Iterative
Accuracy: fixed (cond num)	Variable accuracy
Matrix-matrix operations	Matrix-vector operations
Cost: $O(n^3)$, predictable	Cost not known: $k\ O(n^2)$ Convergence (spectrum) Preconditioning Stopping criteria
Factorization re-use: multiple right-hand sides	Single/few right-hand sides
Fill-in, reordering	Exploit sparsity

Direct Methods vs. Iterative Methods

Direct	Iterative
• Accuracy: fixed (cond num)	• Variable accuracy
• Matrix-matrix operations	• Matrix-vector operations
• Cost: $O(n^3)$, predictable	• Cost not known: $k\ O(n^2)$ Convergence (spectrum) Preconditioning Stopping criteria
• Factorization re-use: multiple right-hand sides	• Single/few right-hand sides
• Fill-in, reordering	• Exploit sparsity
BLAS, LAPACK, FLAME & PETSc HSL, MUMPS, UMFPACK, . . .	PETSc, Trilinos, . . .

Google: "Linear Algebra Software"

Survey of freely available libraries

Parallelism — Multi-cores

30k, 1 core

Memory:	7GB
LU fact:	1640 secs (28m)
Ax = b:	1.9 secs (1 rhs)
AX = B:	172 secs (2k rhs)

RWTH AACHEN UNIVERSITY

30k, 1 core

Memory:	7GB
LU fact:	1640 secs (28m)
Ax = b:	1.9 secs (1 rhs)
AX = B:	172 secs (2k rhs)

40k, 8-core

Memory:	12GB
LU fact:	513secs (9m)
Ax = b:	2.1secs
AX = B:	93.5secs

30k, 1 core

Memory:	7GB
LU fact:	1640 secs (28m)
Ax = b:	1.9 secs (1 rhs)
AX = B:	172 secs (2k rhs)

40k, 8-core

Memory:	12GB
LU fact:	513secs (9m)
Ax = b:	2.1secs
AX = B:	93.5secs

100k — extrapolating:

Memory	80GB
# of 8-cores	8 with 10-12GB each
LU fact	20 minutes

Perturbation Results

$$Ax = b$$

- Acquisition and representation errors:

$$A \rightarrow \hat{A} = A + \delta A \qquad b \rightarrow \hat{b} = b + \delta b$$

$(A + \delta A)\hat{x} = b + \delta b$

- $\hat{x} = x + \delta x$
- $\dfrac{\|\delta x\|}{\|x\|} = \mu(A) \dfrac{\|\delta A\|/\|A\| + \|\delta b\|/\|b\|}{1 - \mu(A)\|\delta A\|/\|A\|}$
- $\mu(A) = \|A\|\|A^{-1}\|$ is the **condition number** of A

Sensitivity to perturbations. Independent of the solution method.
Well vs. ill conditioned problems.

$$f : X \to Y \qquad \hat{f} \text{ is an implementation of } f$$

- Question: $|f - \hat{f}|$?

Exact arithmetic	Floating point arithmetic
$x \longrightarrow f(x)$	$x \longrightarrow \hat{f}(x)$
	$(\hat{x} \longrightarrow \hat{f}(\hat{x}))$

Backward/Forward Stability

$$f : X \to Y \qquad \hat{f} \text{ is an implementation of } f$$

- Question: $|f - \hat{f}|$?

Exact arithmetic	Floating point arithmetic
$x \longrightarrow f(x)$	$x \longrightarrow \hat{f}(x)$ $(\hat{x} \longrightarrow \hat{f}(\hat{x}))$

- **Forward stability**: $\forall x \; \|f(x) - \hat{f}(x)\|$ is small.
- Let $\bar{x}$ be such that $\hat{f}(x) = f(\bar{x})$. Exact sol. to a different probl.
  Backward stability: $\forall x \; \exists \bar{x} \; . \; \|x - \bar{x}\|$ is small.

Factorizations are backward stable.
Iterative methods $\rightarrow$ convergence & convergence rate.

Symmetric Eigenproblem

$$AV = V\Lambda$$

Symmetric Eigenproblem

$$AV = V\Lambda$$

- Three stages:
  1) Reduction to tridiagonal form
  2) **Tridiagonal eigensolver** ($TZ = Z\Lambda$)
  3) Backtransformation

Symmetric Eigenproblem

$$AV = V\Lambda$$

- Three stages:
  1) Reduction to tridiagonal form
  2) **Tridiagonal eigensolver** ($TZ = Z\Lambda$)
  3) Backtransformation
- Reduction: $O(n^3)$, perfectly stable, destroys sparsity.
- Backtransformation: matrix-matrix multiplication, $O(n^3)$, perfectly stable.
- Tridiagonal eigensolvers: MR^3, QR, D&C, ...

  Cost: $O(n^2)$ — $O(n^3)$

  Accuracy: $\|Z^T Z - I\| \leq c\, n\epsilon \quad \wedge \quad \|TZ - Z\Lambda\| \leq c\, n\epsilon \|T\|$

Symmetric Eigenproblem

$$AV = V\Lambda$$

- Three stages:
  1) Reduction to tridiagonal form
  2) **Tridiagonal eigensolver** ($TZ = Z\Lambda$)
  3) Backtransformation
- Reduction: $O(n^3)$, perfectly stable, destroys sparsity.
- Backtransformation: matrix-matrix multiplication, $O(n^3)$, perfectly stable.
- Tridiagonal eigensolvers: MR^3, QR, D&C, ...

  Cost: $O(n^2)$ — $O(n^3)$

  Accuracy: $\|Z^T Z - I\| \leq c\, n\epsilon \quad \wedge \quad \|TZ - Z\Lambda\| \leq c\, n\epsilon \|T\|$
- Eigenvalues AND(?) eigenvectors? How many? Accuracy?
- LAPACK, PMR3, ScaLAPACK. Sparse solver: ARPACK.

Future?

- Exploiting structures, properties.
- Knowledge from applications.
- Massive parallelism:
  hybrid multi-core + distributed architectures.

- Exploiting structures, properties.
- Knowledge from applications.
- Massive parallelism:
  hybrid multi-core + distributed architectures.

Thank you!

For more information: pauldj@aices.rwth-aachen.de