Sparse Matrices and Numerical Linear Algebra Software

Last Lecture (not Last Supper)

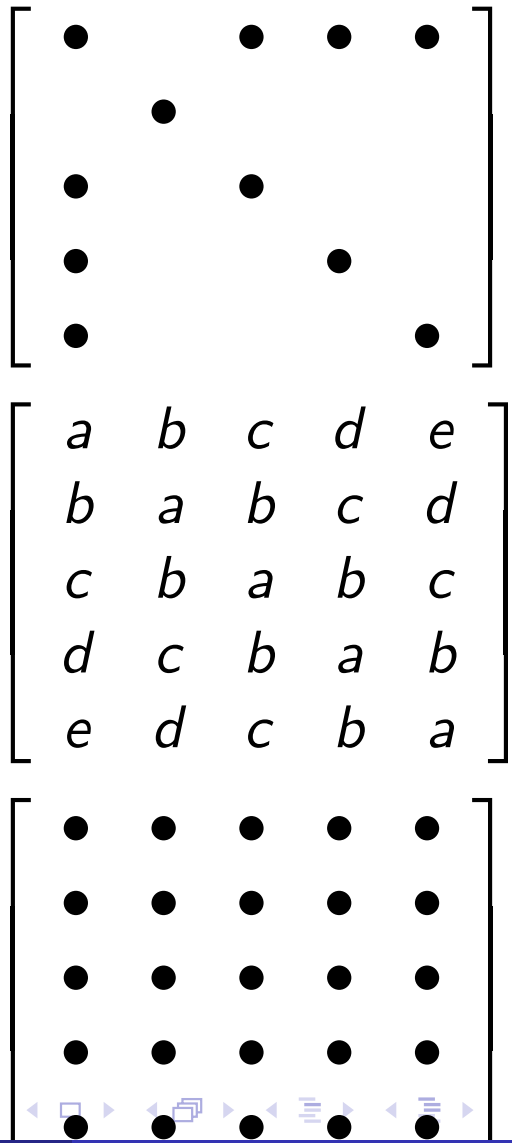
Radu Trîmbiţaş

"Babeş-Bolyai" University

May 20, 2008

Sparse vs. Dense Matrices

- A *sparse matrix* is a matrix with enough zeros that it is worth taking advantage of them [Wilkinson]
- A *structured matrix* has enough structure that it is worthwhile to use it (e.g. Toeplitz)
- A *dense matrix* is neither sparse nor structured



MATLAB Sparse Matrices: Design Principles

- Most operations should give the same results for sparse and full matrices
- Sparse matrices are never created automatically, but once created they propagate
- Performance is important – but usability, simplicity, completeness, and robustness are more important
- Storage for a sparse matrix should be $O(nonzeros)$
- Time for a sparse operation should be close to O(flops)

Data Structures for Matrices

Full:

- Storage: Array of real (or complex) numbers
- Memory: `nrows*ncols`

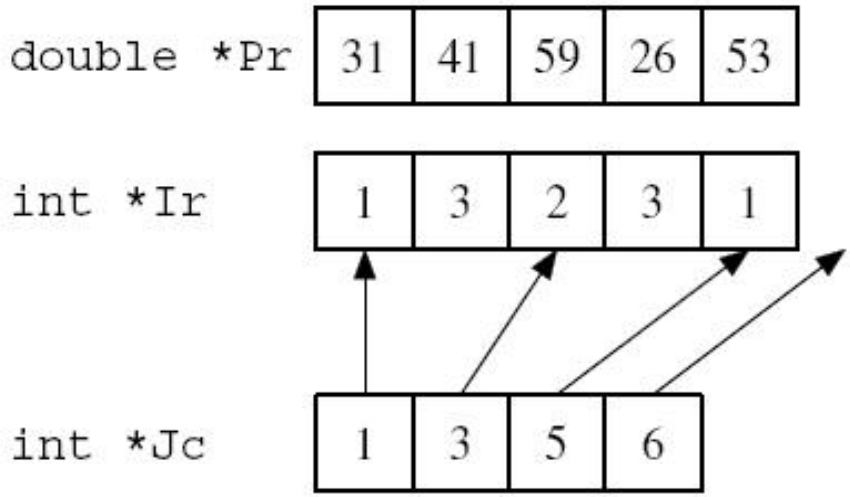
31	0	53
0	59	0
41	26	0

`double *A`

Sparse:

- Compressed column storage
- Memory: About

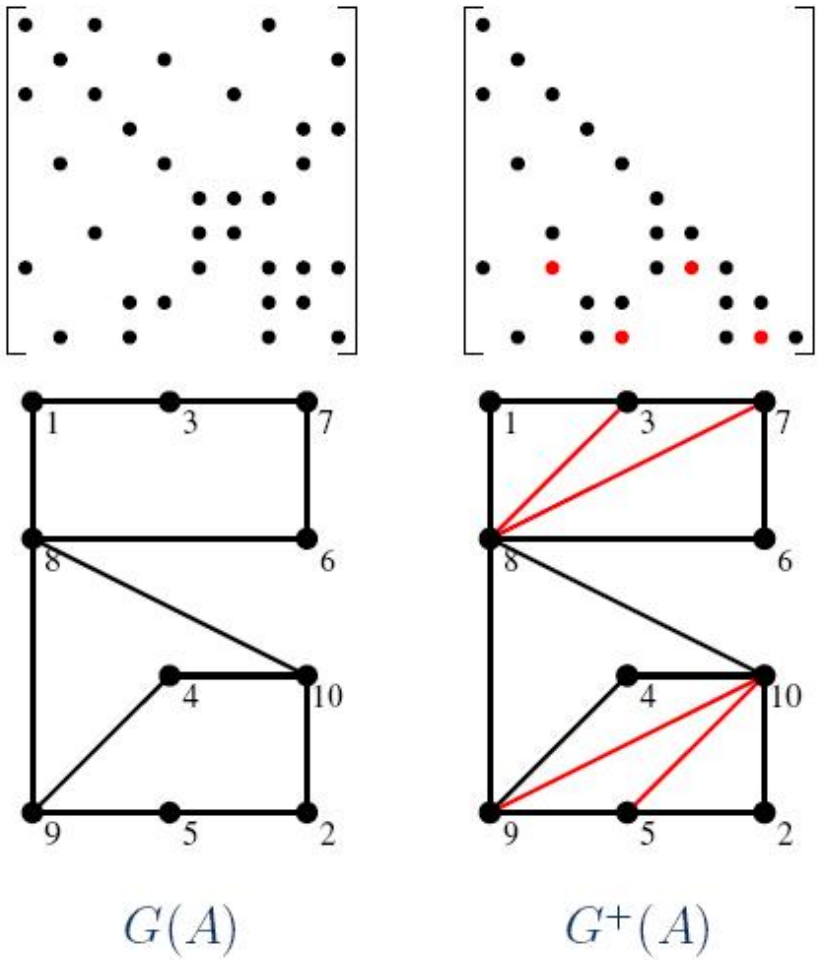
$$1.5 * \texttt{nnz} + .5 * \texttt{ncols}$$



Compressed Column Format -Observations

- Element look-up: $O(log$ #elements in column) time
- Insertion of new nonzero very expensive
- Sparse vector = Column vector (not Row vector)

Graphs and Sparsity: Cholesky Factorization



- **Fill**: New nonzeros in factor
- Symmetric Gaussian Elimination:

  for j=1 **to** N

  Add edges between *j*'s higher numbered neighbors

Permutations of the 2-D Model Problem

- 2-D Model Problem: Poisson's Equation on $n \times n$ finite difference grid
- Total number of unknowns $n^2 = N$
- Theoretical results for the fill-in:
  - With natural permutation: $O(N^{3/2})$ fill
  - With any permutation: $\Omega(N \log N)$ fill
  - With a nested dissection permutation: $O(N \log N)$ fill

Nested Dissection Ordering

- A *separator* in a graph G is a set S of vertices whose removal leaves at least two connected components
- A *nested dissection ordering* for an N-vertex graph G numbers its vertices from 1 to N as follows:
  - Find a separator S, whose removal leaves connected components T_1, T_2, $\ldots$, T_k
  - Number the vertices of S from $N - |S| + 1$ to N
  - Recursively, number the vertices of each component: T_1 from 1 to $|T_1|$, T_2 from $|T_1| + 1$ to $|T_1| + |T_2|$, etc.
  - If a component is small enough, number it arbitrarily
- It all boils down to finding good separators!

Heuristic Fill-Reducing Matrix Permutations

- Banded orderings (Reverse Cuthill-McKee, Sloan, etc):
  - Try to keep all nonzeros close to the diagonal
  - Theory, practice: Often wins for "long, thin" problems
- Minimum degree:
  - Eliminate row/col with fewest nonzeros, add fill, repeat
  - Hard to implement efficiently – current champion is "Approximate Minimum Degree" [Amestoy, Davis, Duff]
  - Theory: Can be suboptimal even on 2-D model problem
  - Practice: Often wins for medium-sized problems

Heuristic Fill-Reducing Matrix Permutations

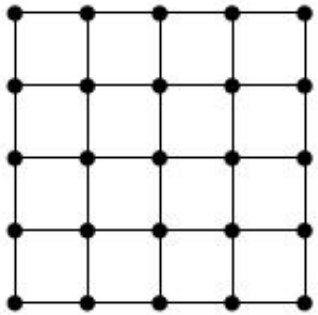
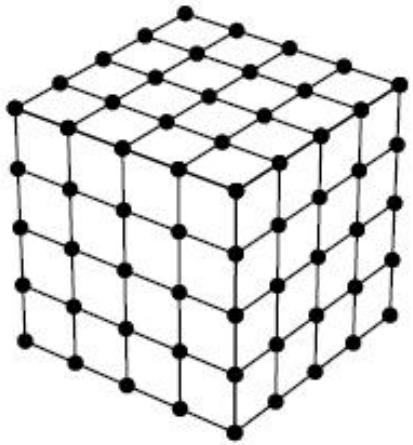
- Nested dissection:
  - Find a separator, number it last, proceed recursively
  - Theory: Approximately optimal separators $\Longrightarrow$ approximately optimal fill and flop count
  - Practice: Often wins for very large problems
- The best modern general-purpose orderings are ND/MD hybrids

Fill-Reducing Permutations in MATLAB

- Reverse Cuthill-McKee:
  - `p=symrcm(A);`
  - Symmetric permutation: `A(p,p)` often has smaller bandwidth than `A`
- Symmetric approximate minimum degree:
  - `p=symamd(A);`
  - Symmetric permutation: `chol(A(p,p))` sparser than `chol(A)`
- Nonsymmetric approximate minimum degree:
  - `p=colamd(A);`
  - Column permutation: `lu(A(:,p))` sparser than `lu(A)`
- Symmetric nested dissection:
  - Not built into MATLAB, several versions in the MESHPART toolbox

Complexity of Direct Methods

- Time and space to solve any problem on any well-shaped finite element mesh with N nodes:

	 1-D	 2-D	 3-D
Space(fill)	$O(N)$	$O(N \log N)$	$O(N^{4/3})$
Time(flops)	$O(N)$	$O(N^{3/2})$	$O(N^2)$

BLAS

- *Basic Linear Algebra Subroutines* (BLAS)
  - Standardized interface for simple vector and matrix operations
  - Manufacturers provide optimized implementations for their machines
- History:
  - BLAS1 (1970s) – Vector operations: $\alpha = x^T y$, $y = \alpha x + y$
  - BLAS2 (mid 1980s) – Matrix-vector operations: $y = Ax + y$
  - BLAS3 (late 1980s) – Matrix-matrix operations: $C = AB + C$
- Efficient cache-aware implementations give almost peak performance for BLAS3 operations
- High level algorithms (Gaussian elimination, etc) use BLAS but no other machine dependent code
  - Performance and portability

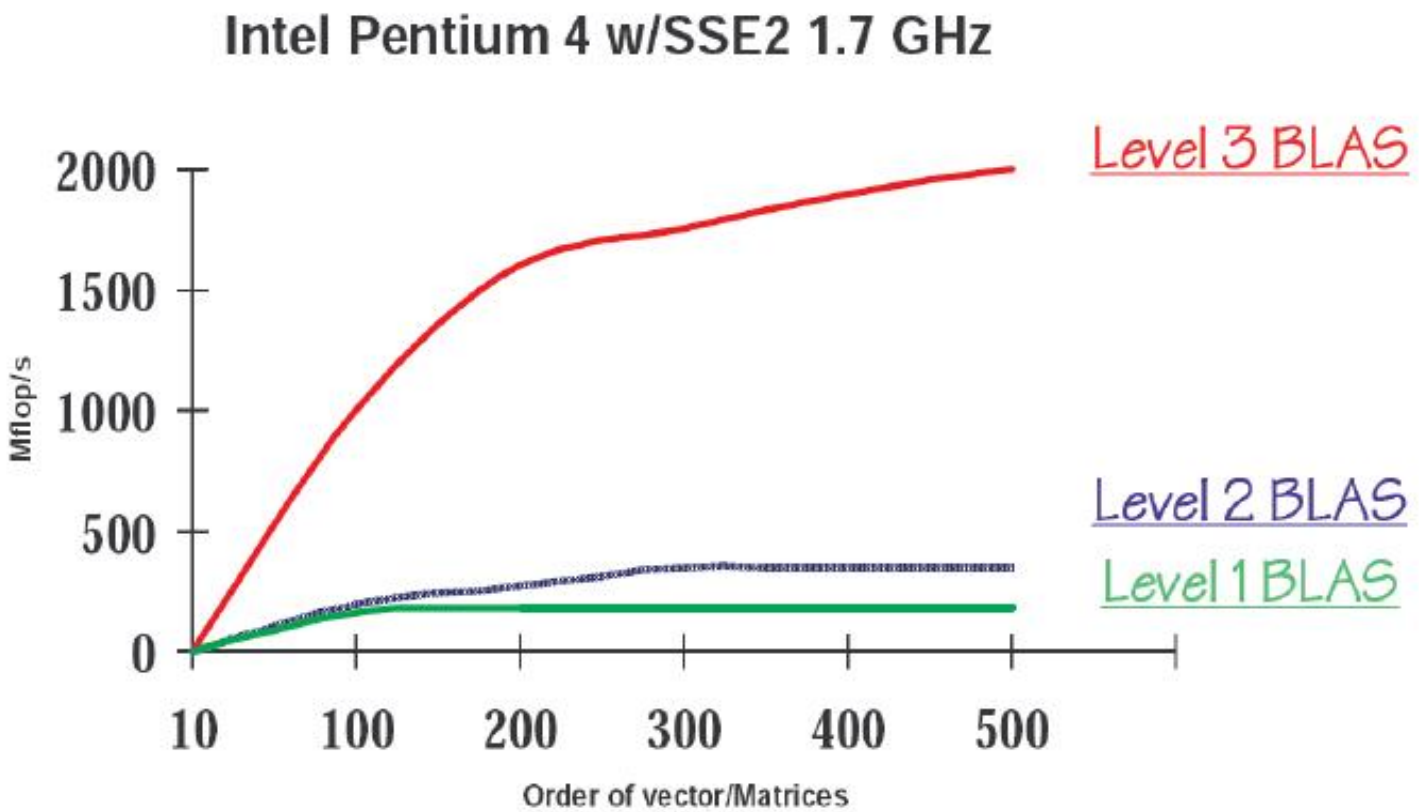
Memory Hierarchy and High Level BLAS

- Modern computers use a *memory hierarchy*
- From fast/expensive to cheap/slow: Registers, L1 cache, L2 cache, local memory, remote memory, secondary memory
- Fast algorithms perform many operations on each memory block to minimize memory access (*cache reuse*)
- Only BLAS3 has potential for very high performance

BLAS	Memory Refs	Flops	Flops/Memory Ref
Level 1 ($y = \alpha x + y$)	$3n$	$2n$	$2/3$
Level 2 ($y = Ax + y$)	n^2	$2n^2$	2
Level 3 ($C = AB + C$)	$4n^2$	$2n^3$	$n/2$

BLAS Performance

- For high performance write algorithms in terms of BLAS3 operations



BLAS Implementations

- Vendor provided:
  - Intel Math Kernel Library (MKL), AMD Core Math Library (ACML)
  - Sun Performance Library
  - SGI Scientific Computing Software Library
- Automatically Tuned Linear Algebra Software (ATLAS)
  - Analyzes hardware to produce BLAS libraries for any platform
  - Used in MATLAB, precompiled libraries freely available
  - Sometimes outperforms vendor libraries
- GOTO BLAS (mainly for Intel processors)
  - Manually optimized assembly code, currently the fastest implementation

Calling BLAS from C

- BLAS standard based on Fortran 77:
  - All memory must be preallocated
  - All variables are passed by reference
- Example: Double precision matrix-matrix multiply ($C = AB + C$):

```
dgemm_(&transa,&transb,&m,&n,&k,&alpha,A,&lda,
       B,&ldb,&beta,C,&ldc);
```

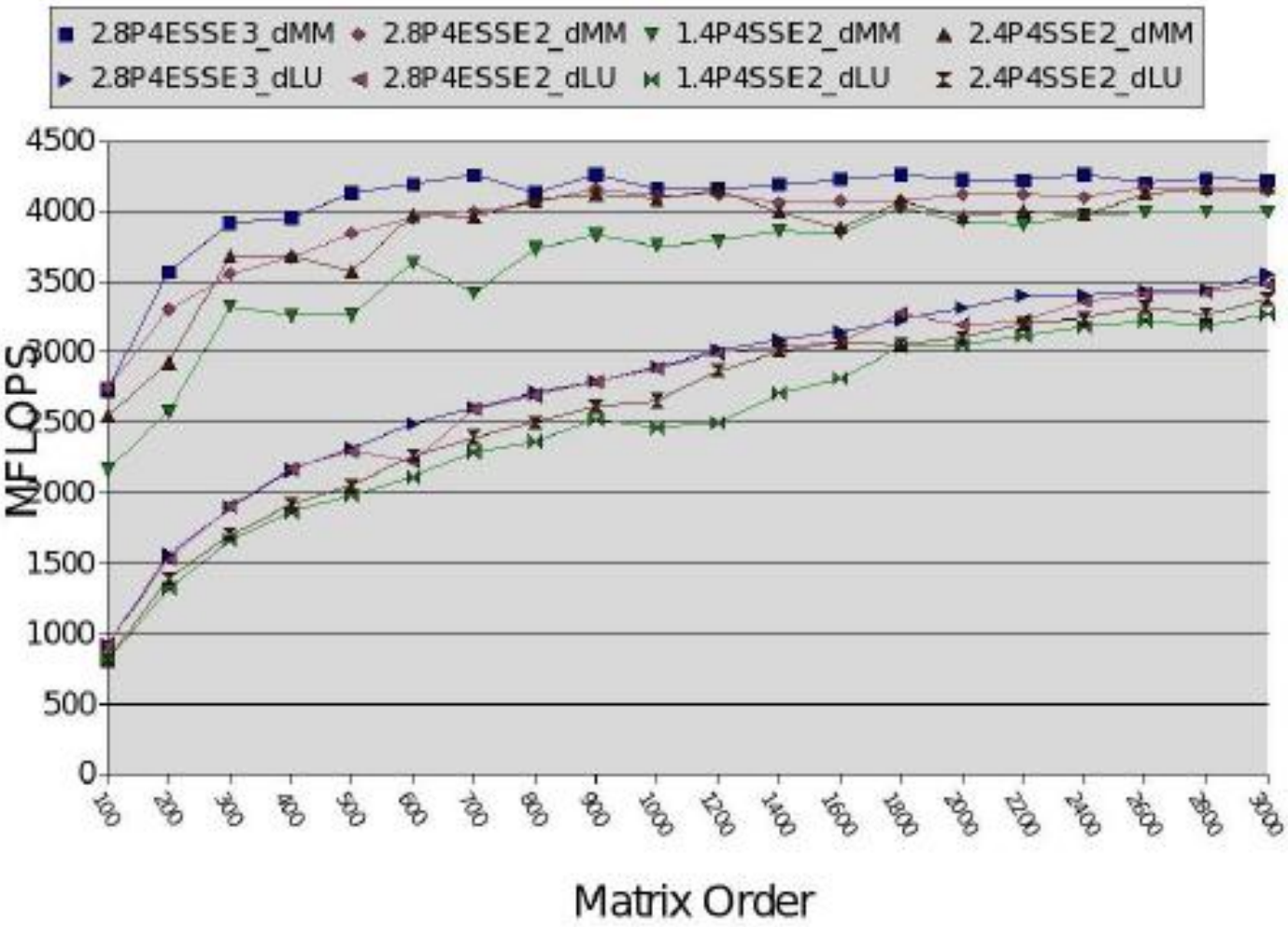
  - `transa`, etc: Matrix transpose ('T') or not ('N')
  - `lda`, etc: Leading dimensions of matrices
  - Some platforms/compilers do not require the trailing underscore
  - In C++, declare functions with `extern "C"`
- See also C BLAS interface in ATLAS

LAPACK

- Standard library for dense/banded linear algebra
  - Linear systems: $Ax = b$
  - Least squares problems: $\min_x \|Ax - b\|_2$
  - Eigenvalue problems: $Ax = \lambda x$, $Ax = \lambda Bx$
  - Singular value decomposition (SVD): $A = U\Sigma V^T$
- Algorithms use BLAS3 as much as possible
- Used by MATLAB (since version 6)
- *LAPACK Search Engine* useful for finding routines

LAPACK Performance

- Matrix-matrix multiply and LU factorization as function of matrix size
- About 80% of peak performance for LU factorization of large matrices



Sparse Solver Packages

- UMFpack (Unsymmetric MultiFrontal method)
  - Used in MATLAB (since version 7.1), no parallel version
- PARADISO
  - Serial and shared memory, used in Intel MKL
- SuperLU
  - Versions for serial and parallel computers (shared/distributed)
  - "Static pivoting" for distributed machines (increase small pivots, iterative refinement for accuracy)
- MUMPS (MUltifrontal Massively Parallel sparse direct Solver)
  - Versions for serial and parallel computers (distributed)