Computational Linear Algebra
CES 703

Imre Pólik, PhD

McMaster University
School of Computational Engineering and Science

2007/2008 Winter Term

Overview

- Linear systems
  - solution of $Ax = b$
  - least squares problems
  - eigenvalues (real symmetric)
- How to exploit structure, sparsity?
- Why does a method work/does not work in practice?
- What are the challenges in implementing a certain algorithm?
  - accuracy
  - speed
  - memory consumption
  - sparsity, structure
  - parallelizability
- When to use particular method?
- Available implementations

Organizational matters

Instructor: Imre Pólik, PhD (CES)
imre.polik@gmail.com
http://imre.polik.googlepages.com
ITB 126, #24030

Office hours: Monday, 12:30-1:30 (right after the class)

Coursework: 2 assignments for 20% each
takehome project for 60% (or final?)
schedule TBD

Course outline I

Week 1: Basics, structured/sparse matrices, storage schemes, elementary operations. Computer architectures, BLAS, LAPACK. Linear systems, solvability. Conditioning, error analysis.

Week 2: Direct methods for linear systems: Gaussian elimination, Iterative refinement. LU and Cholesky factorizations. Pivoting, fill-in, column/row orderings.

Week 3: Iterative methods for linear systems: Jacobi, Gauss-Seidel, Krylov method. Convergence. Preconditioning, conjugate gradient algorithm.

Course outline II

Week 4: Linear least squares problems: normal equations. QR factorization, Householder transformation, Givens rotation. Solution using singular value decomposition.

Week 5: Real symmetric eigenvalue problems: sensitivity of eigenvalues, solution by factorization, approximation, largest, smallest eigenvalues.

Week 6: Power method, inverse iteration, Rayleigh quotient, Jacobi method, Lanczos method.

An introduction to linear systems

Imre Pólik, PhD

McMaster University
School of Computational Engineering and Science

January 7, 2008

Outline

1. Basic concepts
   - Sparse, dense and structured matrices
   - Storage schemes
2. Computing resources
   - Hardware
   - Software
   - Algorithms
3. Linear systems
   - Solvability
   - Condition number
   - Solution of triangular systems

Basic concepts

- What is a sparse matrix?
  - enough zeros that it is worth taking advantage of them (Wilkinson)
  - no fixed percentage limit exists
  - depends on the application
- What is a structured matrix?
  - enough structure that ...
  - examples: lower triangular, symmetric, low rank, few parameters (Hilbert), etc.
  - can be structured and sparse!
- What is a dense matrix?
  - neither sparse nor structured
  - every element has to be manipulated
  - rare in practice

A bit more about structure

- Where is this matrix coming from?
  - dense matrices are rare
- Storage can destroy the structure due to truncation
  - floating point Hilbert matrix?

$$\begin{pmatrix} 1 & 0.500 & 0.333 \\ 0.500 & 0.333 & 0.250 \\ 0.333 & 0.250 & 0.200 \end{pmatrix}$$

  - similar issue with rank 1 matrices
- Adapt the algorithm to the structure (hardcoding)

Storage schemes

- Dense matrices
  - row (C) or column (Fortran) oriented
  - other schemes for efficiency
- Sparse matrices
  - data and indices (overhead!)
  - some operations are cumbersome
  - multiple storage
- Structured matrices
  - symmetric: store half of the matrix
  - narrow band: store numbers diagonally
  - rank 1: store the factors

Hardware architectures

- Word length
  - 32, 64, 128bit
  - larger address space
  - more operations per cycle
- Parallelism
  - shared or distributed memory
  - communication cost
- Cache architecture
  - access to RAM is slow
  - L1, L2, L3
  - the faster the smaller
  - avoid cache misses, reuse data already in cache
  - contiguity

Software resources

- Programming languages and compilers
  - interpreted: Matlab, Octave, Scilab, Python, etc.
  - compiled: C/C++, Fortran, Java
- Prebuilt libraries
  - don't reinvent the wheel!
  - BLAS, LAPACK, ScaLAPACK, MA27, MA47, Sparspak

BLAS and LAPACK

- Building blocks for linear algebra routines
  - BLAS 1: vector-vector operations
  - BLAS 2: matrix-vector operations
  - BLAS 3: matrix-matrix operations
  - LAPACK: factorizations, solution of systems
- Available for most OSs and architectures
  - MKL (Intel)
  - ACML (AMD)
  - VecLib (Apple)
  - SPL (SUN)
  - GotoBLAS (TACC, various platforms)
  - ATLAS (GPL, compiled from source)
- Efficient parallel implementations
- Mostly dense but special routines for:
  - symmetric
  - narrow band
  - low rank

Algorithms in general

- Direct (finite)
  - exact result (in exact arithmetic)
  - higher storage cost
  - typical for dense matrices
  - examples: LU, Cholesky factorization, Gaussian elimination
- Iterative (asymptotic)
  - no inversions
  - approximate result
  - low storage
  - very efficient for sparse matrices
  - examples: power iteration for largest eigenvalue

$$x_{k+1} = \frac{Ax_k}{\|Ax_k\|} \tag{1}$$

Solution of linear systems

Solve $Ax = b$, where $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$

$m < n$: underdetermined, infinitely many solutions (if any)

$m > n$: overdetermined, no solution

- closest solution, least squares (see Week 4)

$m = n$: unique solution if A is full rank, underdetermined otherwise

The interesting case for now: $m = n$, A is full rank, square, positive definite, $\det A \neq 0$

Condition number

Solve $Ax = b$, where $x, b \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$, A is full rank $(\det A \neq 0)$

- A^{-1} exists, $x = A^{-1}b$
- Relative error in x: $(x + \Delta x = A^{-1}(b + \Delta b))$

$$\frac{\|\Delta x\| / \|x\|}{\|\Delta b\| / \|b\|} = \frac{\left\|A^{-1}\Delta b\right\| / \left\|A^{-1}b\right\|}{\|\Delta b\| / \|b\|} \leq \underbrace{\left\|A^{-1}\right\| \|A\|}_{\kappa(A)} \quad (2)$$

- $\kappa(A)$ is the ratio of the largest and smallest singular value of A
- if $\kappa(A)$ is large then A is close to singular and the system is called ill-conditioned
- numerical solution will suffer loss of accuracy regardless of the algorithm, even in exact arithmetics!

Triangular systems (lower)

- Row oriented approach

$$\begin{aligned} i &= 1, \dots, n \\ x_i &= (b_i - \sum_{k=1}^{i-1} A_{ik} x_k)/A_{ii} \end{aligned}$$

  - can't exploit sparsity in x

- Column oriented approach

$$\begin{aligned} i &= 1, \dots, n \\ x_i &= b_i/A_{ii} \\ b_{i+1:n} &\leftarrow b_{i+1:n} - x_i A_{i+1:n,i} \end{aligned}$$

  - exploits sparsity in x
  - operation count: Ax

- Building block for general systems

Alan Edelman.

MIT 18.337: Applied parallel computing.

Lecture notes, 2004.

Chapter 4 and 5.

Alan George and Joseph W. Liu.

Computer Solutions of Large Sparse Positive Definite Systems.

Prentice Hall, 1981.