Iterative methods for linear systems

Imre Pólik, PhD

McMaster University
School of Computational Engineering and Science

January 21, 2008

Outline

Overview of iterative methods

- Input: A, b, (x_0)
- Output: x_{app}
- Iteration: update the solution
- Stationary:
  - $x^{(k+1)} = f\left(x^{(k)}\right)$, f does not change
- Nonstationary:
  - $x^{(k+1)} = f_k\left(x^{(k)}\right)$, f_k updated at every iteration
- Approximate solution at every iteration
- Stop if close enough (no progress)
- Error: $e^{(k)} = \left\|x^{(k)} - x\right\|$
- Residual: $r^{(k)} = \left\|Ax^{(k)} - b\right\|$

Why do we need iterative methods?

- Very sparse matrix
  - quick to compute Au
- A is not represented explicitly
  - $\mathcal{A}(X) = AX + XB$
- Memory
  - not enough space to factorize
- Quick and dirty solution needed
  - preconditioning

Issues

- Convergence (does it work?)
- Convergence rate (how fast is it?)
- Work per iteration
- Extra storage
- Stopping criteria
- Preconditioning: solve $MAx = Mb$
- Parallelizability

1. Introduction
2. Stationary methods
   - The Jacobi method
   - The Gauss-Seidel method
   - Matrix splitting in general
   - Successive overrelaxation
3. Nonstationary methods
4. Implementation

The Jacobi method

- Solve for x_i given the other components

$$x_i^{(k)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)}}{a_{ii}}$$

- Matrix form ($A = D - L - U$):

$$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b$$

- Trivial efficient parallelization
- Needs an extra copy of x
- Converges if A is strongly diagonally dominant
- Not very efficient in practice

The Gauss-Seidel method

- Solve for x_i given the other components, BUT
- Use the updated value if possible

$$x_i^{(k)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)}}{a_{ii}}$$

- Matrix form ($A = D - L - U$):

$$x^{(k)} = (D - L)^{-1}(U x^{(k-1)} + b)$$

- Efficient parallelization is tricky (except for sparse)
- x can be updated in place
- Everything depends on the order of the update
- Converges if A is strongly diagonally dominant or SPD
- Better than Jacobi

Matrix splitting in general

- Decomposition: $A = M - N$
- Iteration: $x^{(k)} = M^{-1}Nx^{(k-1)} + M^{-1}b$
- M should be easy to invert (diagonal, triangular)
- Error: $e^{(k)} = \left(M^{-1}N\right)^k e^{(0)}$
- Converges if $\left\|M^{-1}N\right\| < 1$
- Can exploit special structure

Successive overrelaxation

- Combine iterates of the GS method

$$x_i^{(k)} = \omega x_{\mathrm{GS}i}^{(k)} + (1-\omega) x_i^{(k-1)}$$

- Matrix form ($A = D - L - U$):

$$x^{(k)} = (D - \omega L)^{-1}(\omega U + (1-\omega)D)x^{(k-1)} + \omega(D - \omega L)^{-1}b$$

- Choose ω to accelerate convergence
- $\omega = 1$ gives GS
- Optimal ω is hard to find
- Converges for any SPD A, $\omega \in (0, 2)$

1. Introduction
2. Stationary methods
3. Nonstationary methods
   - The Conjugate gradient algorithm
   - Variants of the CG method
4. Implementation

The Conjugate gradient algorithm I

- u, v are conjugate: $u^T A v = 0$
- $p^{(1)}, \ldots, p^{(n)}$ mutually conjugate vectors
- $x = \alpha_1 p^{(1)} + \ldots \alpha_n p^{(n)}$
- $\alpha_k = \frac{{p^{(k)}}^T b}{{p^{(k)}}^T A p^{(k)}}$
- Theoretical algorithm
  - find the conjugate directions
  - compute the coefficients
- Practice: construct the directions on-the-fly

The Conjugate gradient algorithm II

$$
\begin{aligned}
r^{(0)} &= b - Ax^{(0)} \\
p^{(0)} &= r^{(0)} \\
k &= 0 \\
\alpha^{(k)} &= \frac{{r^{(k)}}^T r^{(k)}}{{p^{(k)}}^T A p^{(k)}} \\
x^{(k+1)} &= x^{(k)} + \alpha^{(k)} p^{(k)} \\
r^{(k+1)} &= r^{(k)} - \alpha^{(k)} A p^{(k)} \\
\beta^{(k)} &= \frac{{r^{(k+1)}}^T r^{(k+1)}}{{r^{(k)}}^T r^{(k)}} \\
p^{(k+1)} &= r^{(k+1)} + \beta^{(k)} p^{(k)} \\
k &= k + 1
\end{aligned}
$$

The Conjugate gradient algorithm III

- Apply the general CG to $\min \frac{1}{2} x^T A x - b^T x$
- Convergence ($\|u\|_A = u^T A u$, A is SPD):

$$\left\| x^{(k)} - x \right\|_A \leq 2 \left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}} \right)^k \left\| x^{(0)} - x \right\|_A$$

- $x^{(k)}$ is the minimizer over an increasing subspace

$$\operatorname{span}\left(\left\{ b, Ab, A^2 b, \ldots, A^{k-1} b \right\} \right) \text{ Krylov}$$

- Easy to parallelize
- Very effective in practice

Variants of the CG method I

- CGNE, CGNR
  - CG for $A^T A = A^T b$
  - simple treatment for nonsymmetric systems
  - poor numerical performance
  - slow convergence
- GMRES
  - orthogonalized Krylov system
  - handles nonsymmetric systems
  - computes only the residual without the iterate!
  - keeps all iterates, needs regular restarts
- Biconjugate
  - two conjugate sequences
  - handles nonsymmetric systems
  - quite unstable, irregular convergence
  - requires A^T

Variants of the CG method II

- QMR
  - improves BiCG
  - more stable
  - requires A^T
- CGS, BiCGSTAB, etc.
- The Chebyshev iteration
  - requires bounds on the eigenvalues
  - no inner products

Work per iteration

Method	Inner product	$Ax + y$	Matrix-vector
JACOBI			1
SOR (GS)		1	1
CG	2	3	1
GMRES	$i + 1$	$i + 1$	1
BiCG	2	5	$1 + 1^T$
QMR	2	12	$1 + 1^T$
CHEBYSHEV		2	1

Storage requirement

JACOBI	$+3n$
SOR (GS)	$+2n$
CG	$+6n$
GMRES	$+(i+5)n$
BiCG	$+10n$
QMR	$+16n$
CHEBYSHEV	$+5n$

Stopping criteria

- Small error/residual
- No progress
- Iteration/time limit reached
- Examples:

$$\left\| r^{(k)} \right\| \leq \varepsilon \left(\|A\| \cdot \left\| x^{(k)} \right\| + \|b\| \right)$$

$$\left\| r^{(k)} \right\| \leq \varepsilon \|b\|$$

$$\left\| r^{(k)} \right\| \leq \varepsilon \left(\left\| x^{(k)} \right\| / \|A^{-1}\| \right)$$

What to try?

1. Chebyshev
2. CG
3. QMR
4. CGS, BiCGSTAB
5. GMRES

Available software

- ITPACK
  - `http://rene.ma.utexas.edu/CNA/ITPACK/`
  - open source, written in Fortran
  - modified CG
  - sophisticated heuristics to boost convergence
- Matlab

R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst.

Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.

SIAM, Philadelphia, PA, 1994.

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.