Outline

Sparse Matrices —
Non Stationnary Iterative Methods
Matrix Computations — CPSC 5006 E
Julien Dompierre
Department of Mathematics and Computer Science
Laurentian University
Sudbury, December 8, 2010

Stationary Iterative Methods

Iterative methods that can be expressed in the simple form
\[ x^{(k)} = Bx^{(k-1)} + c \]

where neither \( B \) nor \( c \) depend upon the iteration count \( k \) are called stationary iterative methods. The four main stationary methods are Jacobi, Gauss-Seidel, successive over-relaxation (SOR) and symmetric successive over-relaxation (SSOR).

Stationary methods are older, simpler to understand and implement, but usually not as effective.

Jacobi Method

The Jacobi method is based on solving for every variable locally with respect to the other variables; one iteration of the method corresponds to solving for every variable once. The resulting method is easy to understand and implement, but convergence is slow.
Gauss-Seidel

The **Gauss-Seidel method** is like the Jacobi method, except that it uses updated values as soon as they are available. In general, if the Jacobi method converges, the Gauss-Seidel method will converge faster than the Jacobi method, though still relatively slowly.

SOR

**Successive Over Relaxation** (SOR) can be derived from the Gauss-Seidel method by introducing an extrapolation parameter $\omega$. For the optimal choice of $\omega$, SOR may converge faster than Gauss-Seidel by an order of magnitude.

SSOR

**Symmetric Successive Over Relaxation** (SSOR) has no advantage over SOR as a standalone iterative method; however, it is useful as a preconditioner for non stationary methods.

Non Stationary Iterative Methods

The **non stationary methods** differ from stationary methods in that the computations involve information that changes at each iteration. Typically, constants are computed by taking inner products of residuals or other vectors arising from the iterative method. They are mostly based on the idea of sequences of orthogonal vectors.

The non stationary methods are a relatively recent development; their analysis is usually harder to understand, but they can be highly effective.
The conjugate gradient method (CG) derives its name from the fact that it generates a sequence of conjugate (or $A$-orthogonal) vectors. These vectors are the residuals of the iterates. They are also the gradients of a quadratic functional, the minimization of which is equivalent to solving the linear system. CG is an extremely effective method when the coefficient matrix is symmetric positive definite, since storage for only a limited number of vectors is required.

**MINRES and SYMMLQ**

Minimum Residual (MINRES) and Symmetric LQ (SYMMLQ). These methods are computational alternatives for CG for coefficient matrices that are symmetric but possibly indefinite. SYMMLQ will generate the same solution iterates as CG if the coefficient matrix is symmetric positive definite.

**CGNE and CGNR**

Conjugate Gradient on the Normal Equations (CGNE) and CGNR. These methods are based on the application of the CG method to one of two forms of the normal equations for $Ax = b$. CGNE solves the system $(AA^T)y = b$ for $y$ and then computes the solution $x = A^Ty$. CGNR solves $(A^TA)x = \tilde{b}$ for the solution vector $x$ where $\tilde{b} = A^Tb$. When the coefficient matrix $A$ is non symmetric and non singular, the normal equations matrices $AA^T$ and $A^TA$ will be symmetric and positive definite, and hence CG can be applied. The convergence may be slow, since the spectrum of the normal equations matrices will be less favorable than the spectrum of $A$.

**GMRES**

The Generalized Minimal Residual method (GMRES) computes a sequence of orthogonal vectors (like MINRES), and combines these through a least-squares solve and update. However, unlike MINRES (and CG) it requires storing the whole sequence, so that a large amount of storage is needed. For this reason, restarted versions of this method are used. In restarted versions, computation and storage costs are limited by specifying a fixed number of vectors to be generated. This method is useful for general non symmetric matrices.
BiCG

The Biconjugate Gradient method generates two CG-like sequences of vectors, one based on a system with the original coefficient matrix $A$, and one on $A^T$. Instead of orthogonalizing each sequence, they are made mutually orthogonal, or "bi-orthogonal". This method, like CG, uses limited storage. It is useful when the matrix is non symmetric and non singular; however, convergence may be irregular, and there is a possibility that the method will break down. BiCG requires a multiplication with the coefficient matrix and with its transpose at each iteration.

QMR

The Quasi-Minimal Residual method applies a least-squares solve and update to the BiCG residuals, thereby smoothing out the irregular convergence behavior of BiCG. Also, QMR largely avoids the breakdown that can occur in BiCG. On the other hand, it does not effect a true minimization of either the error or the residual, and while it converges smoothly, it does not essentially improve on the BiCG. Conjugate Gradient Squared (CGS).

CGS

The Conjugate Gradient Squared method is a variant of BiCG that applies the updating operations for the $A$-sequence and the $A^T$-sequences both to the same vectors. Ideally, this would double the convergence rate, but in practice convergence may be much more irregular than for BiCG. A practical advantage is that the method does not need the multiplications with the transpose of the coefficient matrix.

Bi-CGSTAB

The Biconjugate Gradient Stabilized method is a variant of BiCG, like CGS, but using different updates for the $A^T$-sequence in order to obtain smoother convergence than CGS.
Cost of the Methods (Barrett et al, p. 31)

<table>
<thead>
<tr>
<th>Method</th>
<th>Inner Product</th>
<th>SAXPY</th>
<th>Matrix-Vector Product</th>
<th>Precond Solve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td></td>
<td>1^a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GS</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOR</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CG</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GMRES</td>
<td>i + 1</td>
<td>i + 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BiCG</td>
<td>2</td>
<td>5</td>
<td>1/1</td>
<td>1/1</td>
</tr>
<tr>
<td>QMR</td>
<td>2</td>
<td>8 + 4</td>
<td>1/1</td>
<td>1/1</td>
</tr>
<tr>
<td>CGS</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Bi-CGSTAB</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

* The method performs no real matrix-vector product or preconditioner solve, but the number of operations is equivalent to a matrix-vector multiply. 1/1 means 1 multiplication with the matrix, 1 with the transpose.

Storage Requirements (Barrett et al, p. 32)

<table>
<thead>
<tr>
<th>Method</th>
<th>Storage Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td>matrix + 3n</td>
</tr>
<tr>
<td>GS</td>
<td>matrix + 2n</td>
</tr>
<tr>
<td>SOR</td>
<td>matrix + 6n</td>
</tr>
<tr>
<td>CG</td>
<td>matrix + (i + 5)n</td>
</tr>
<tr>
<td>GMRES</td>
<td>matrix + 10n</td>
</tr>
<tr>
<td>BiCG</td>
<td>matrix + 11n</td>
</tr>
<tr>
<td>QMR</td>
<td>matrix + 10n</td>
</tr>
<tr>
<td>CGS</td>
<td>matrix + 16n</td>
</tr>
<tr>
<td>Bi-CGSTAB</td>
<td>matrix + 5n</td>
</tr>
</tbody>
</table>

Barrett et al. Templates for the Solution of Linear Systems, p.32

Choice of an Iterative Method

Not every method will work on every problem type, so knowledge of matrix properties is the main criterion for selecting an iterative method.

Selecting the "best" method for a given class of problems is largely a matter of trial and error. It also depends on how much storage one has available (GMRES), on the availability of $A^T$ (BiCG and QMR), and on how expensive the matrix vector products (and Solve steps with $M$) are in comparison to SAXPYs and inner products. If these matrix vector products are relatively expensive, and if sufficient storage is available then it may be attractive to use GMRES and delay restarting as much as possible.

Conjugate Gradient Algorithm (p. 527)

Algorithm 1 Conjugate Gradients. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix, $b \in \mathbb{R}^n$, the right-hand side, $x^{(0)}$, an initial guess, $\varepsilon$ a stopping criterion, and MaxIter the maximum number of iterations.

1: $k = 0$, $r^{(0)} = b - Ax^{(0)}$
2: while $k < \text{MaxIter}$ and $\|r^{(k)}\|/\|r^{(0)}\| > \varepsilon$
3: $k = k + 1$
4: if $k = 1$ then
5: $p^{(1)} = r^{(0)}$
6: else
7: $\beta^{(k)} = r^{T(k-1)}r^{(k-1)}/r^{T(k-2)}r^{(k-2)}$
8: $p^{(k)} = r^{(k-1)} + \beta^{(k)}p^{(k-1)}$
9: end
10: $\alpha^{(k)} = r^{T(k-1)}r^{(k-1)}/p^{T(k)}Ap^{(k)}$
11: $x^{(k)} = x^{(k-1)} + \alpha^{(k)}p^{(k)}$
12: $r^{(k)} = r^{(k-1)} - \alpha^{(k)}Ap^{(k)}$
13: end
In the unsymmetric $Ax = b$ problem, it is tempting to solve the equivalent symmetric positive definite system

$$A^T Ax = A^T b$$

using existing conjugate gradient technology. Indeed, if we make the substitution $A \leftarrow A^T A$ in the previous CG algorithm and note that a normal equation residual $A^b - A^T Ax(k)$ is $A^T$ times the "true" residual $b - Ax(k)$, then we obtain the **Conjugate Gradient Normal Equation Residual method**.

In general, this normal equation approach is handicapped by the squaring of the condition number. However, there are occasions where they are effective.

---

**Algorithm 2** CGNR

Let $A \in \mathbb{R}^{n \times n}$ be a non singular matrix, $b \in \mathbb{R}^n$, the right-hand side, $x^{(0)}$, an initial guess, $\varepsilon$ a stopping criterion, and $\text{MaxIter}$ the maximum number of iterations.

1: $k = 0$, $r(0) = b - Ax^{(0)}$
2: while $k < \text{MaxIter}$ and $\|r_k\|/\|r_0\| > \varepsilon$
3:  $k = k + 1$
4:  if $k = 1$
5:  $p(1) = A^T r(0)$
6:  else
7:     $\beta(k) = (A^T r(k-1))^T (A^T r(k-1))/(A^T r(k-2))^T (A^T r(k-2))$
8:     $p(k) = A^T r(k-1) + \beta(k)p(k-1)$
9:  end
10:  $\alpha(k) = (A^T r(k-1))^T (A^T r(k-1))/(Ap(k))^T (Ap(k))$
11:  $x(k) = x(k-1) + \alpha(k)p(k)$
12:  $r(k) = r(k-1) - \alpha(k)Ap(k)$
13: end

---

**General framework**

Saad (p. 130)

---

**Prototype of Projection Method**

Saad (p. 131)
Krylov Subspaces

Saad (p. 152)

Arnoldi Method

Gram-Schmidt $A$-orthogonalisation. Saad p. 154

Modified Gram-Schmidt $A$-orthogonalisation. Saad p. 156

Householder $A$-orthogonalisation. Saad p. 157
General Principle of GMRES

Demmel 306

GMRES (p. 549)

Algorithm 3 GMRES. Let $A \in \mathbb{R}^{n \times n}$ be a non singular matrix, $b \in \mathbb{R}^n$, the right-hand side, $x^{(0)}$, an initial guess, then the following algorithm computes $x \in \mathbb{R}^n$ so $Ax = b$.

1: $k = 0$, \quad $r^{(0)} = b - Ax^{(0)}$, \quad $h_{10} = \|r^{(0)}\|_2$
2: \textbf{while} ($h_{k+1,k} > 0$)
3: \qquad $q^{(k+1)} = r^{(k)}/h_{k+1,k}$
4: \qquad $k = k + 1$
5: \qquad $r^{(k)} = Aq^{(k)}$
6: \qquad \textbf{for} $i = 1 : k$
7: \qquad \quad $h_{ik} = q^{(i)\top}w$
8: \qquad \quad $r^{(k)} = r^{(k)} - h_{ik}q^{(i)}$
9: \qquad \textbf{end}
10: \qquad $h_{k+1,k} = \|r^{(k)}\|_2$
11: \qquad $x^{(k)} = x^{(0)} + Q^{(k)}y^{(k)}$ where $\|\rho_0e_1 - \tilde{H}^{(k)}y^{(k)}\|_2 = \min$
12: \textbf{end}

Restarted GMRES

The main problem with “unlimited GMRES” is that the $k$th iteration involves $O(kn)$ flops and also $O(kn)$ memory floats. A practical GMRES implementation requires a restart strategy to avoid excessive amounts of computation and memory traffic. For example, if at most $m$ steps are tolerable, then $x^{(m)}$ can be used as the initial vector for the next GMRES sequence.