

# CMSC 661 Notes

David Greenfieldboyce

March 31, 2005

## 1 Space and Computational Complexity for the Arnoldi Algorithm

Applying Krylov Subspace methods for solving the matrix equation:

$$Ax^* = b$$

requires a set of basis vectors for the Krylov subspace:

$$\mathcal{K}_k(\hat{G}, c) = \text{span}\{c, \hat{G}c, \hat{G}^2c, \dots, \hat{G}^{k-1}c\}$$

The obvious choice of basis vectors,  $c, \hat{G}c, \hat{G}^2c, \dots, \hat{G}^{k-1}c$ , are too close to linearly-dependent for these methods to be computationally accurate. The Arnoldi algorithm, based on the process of Gram-Schmidt orthogonalization, will allow us to compute a set of *B-orthonormal* basis vectors for  $\mathcal{K}_k$ .

The Arnoldi algorithm computes a basis  $[P, H]$  such that:

$$\hat{G}P_k = P_{k+1}H_k$$

Where the space requirements of the various elements are:

$$\begin{aligned}\hat{G} &: n \times n \\ P_k &: n \times k \\ P_{k+1} &: n \times (k+1) \\ H_k &: k+1 \times k\end{aligned}$$

The algorithm requires storage for  $P, H, \hat{G}$ , and  $B$  where  $B$  is the matrix for which the resulting basis vectors will be  $B$ -orthonormal.

Typical values for  $k$  and  $n$  will be 10 and  $10^6$ , respectively.

Each iteration of the algorithm requires the computation of:

- $\hat{G}p^j$
- one matrix multiply by  $B$
- $n - 1$  adds
- $n$  multiplies
- one square root
- $n$  divides

In addition, there will be  $j + 1$  iterations of the inner loop, each of which will require:

- $B$  multiplies,  $n - 1$  adds, and  $n$  multiplies to compute  $h_{ij}$
- $n$  multiplies and  $n$  subtractions to compute  $p^{j+1}$

We compute an estimate for  $x$  using the basis functions  $P_k$  as:

$$x^k = P_k y^{(k)}$$

The GMRES and CG algorithms allow us to solve for the coefficients  $y^{(k)}$  using  $P$  and  $H$  computed via the Arnoldi Algorithm.

## 2 Conjugate Gradients Algorithm

The CG algorithm is based on the Arnoldi algorithm, but is simplified by the requirement that  $A$  and  $M$  be symmetric, positive definite matrices. Rather than storing all of  $P_k$ , the CG algorithm only needs to store the most-recently computed vector  $p$ .

Recall that  $A$  is partitioned as  $M - N$ . The matrix  $M$  is called the preconditioner. The choice of  $M$  will determine both the cost-per-iteration and the rate of convergence for GMRES. The extremes would be:

- $M = I$ , where the cost-per-iteration is low but the number of iterations is high
- $M = A$ , where the cost-per-iteration is high but only one iteration is required

Each iteration of the CG algorithm requires the computation of:

- $5n$  multiplies
- one matrix multiply by  $A$
- the solution of  $Mz = r$

Note: to avoid repeating the calculation  $Ap$ , the resulting computation will also need to be stored.

### 3 The practical form of GMRES

In each iteration of GMRES, the algorithm expands the Krylov subspace that is used for the approximation. It is impractical for both storage and computation to run the algorithm more than 100 iterations at most.

In practice, the algorithm is run for a limited number of iterations and then restarted using the residual from the previous run as the starting value for  $c$ .

Note: the note at the bottom of page 1 from part 3 of the notes on *Solution of Sparse Linear Systems* is somewhat mis-leading. Since the first  $k$  columns of matrix  $H_{k+1}$  are equivalent to  $H_k$ , each successive solution to the equation  $H_k^T H_k y^{(k)} = H_k^T P_{k+1}^T c$  will only require time  $O(k^2)$ , rather than  $O(k^3)$  that would normally be required to solve it.

### 4 Convergence for GMRES

For a diagonalizable matrix, we have an expression for convergence of the norm of the residual which is bounded by a value proportional to  $\epsilon_k$ , where  $\epsilon_k$  is defined as:

$$\epsilon_k = \min_{p \in \mathcal{P}_k} \max_{j=1..n} |p(\lambda_j)|$$

where  $\mathcal{P}_k$  is the set of all polynomials of degree at least  $k$  where  $p(0) = 1$ .

As a result, matrices which have eigenvalues near the origin will have convergence ratios close to 1 and will not converge quickly. If the eigenvalues are far from the origin, there will be a polynomial that will reach small values at the eigenvalues, especially as we increase  $k$  and have more degrees of freedom for the polynomial.

If all of the eigenvalues lie within an ellipse that does not contain the origin, we can compute a bound on  $\epsilon_k$ .