

CS 689-001
Parallel Numerical Computation

Lecture 6*
Krylov Subspace Methods (II)

Professor Jun Zhang
Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046
March 1, 1999

*Reference: *Iterative Methods for Sparse Linear Systems*, by Yousef Saad, 1996.

Lanczos Biorthogonalization

Lanczos biorthogonalization algorithm builds a pair of biorthogonal bases for the two bases

$$\mathcal{K}_m(A, v_1) = \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\}$$

$$\mathcal{K}_m(A^T, w_1) = \text{span}\{w_1, A^T w_1, \dots, (A^T)^{m-1}w_1\}$$

Advantages and disadvantages:

- uses short recurrence relations, no need to store all basis vectors
- need operations with A^T , inconvenient if A^T is not available
- more easily break down $\delta_{j+1} = 0$

Lanczos Biorthogonalization Procedure

1. Choose two vectors v_1, w_1
so that $(v_1, w_1) = 1$
2. Set $\beta_1 = \delta_1 = 0, v_0 = w_0 = 0$
3. For $j = 1, 2, \dots, m$, do:
4. $\alpha_j = (Av_j, w_j)$
5. $\bar{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
6. $\bar{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
7. $\delta_{j+1} = |(\bar{v}_{j+1}, \bar{w}_{j+1})|^{1/2}$
If $\delta_{j+1} = 0$ stop
8. $\beta_{j+1} = (\bar{v}_{j+1}, \bar{w}_{j+1}) / \delta_{j+1}$
9. $w_{j+1} = \bar{w}_{j+1} / \beta_{j+1}$
10. $v_{j+1} = \bar{v}_{j+1} / \delta_{j+1}$
11. End do

Lanczos Biorthogonalization (II)

The parameters δ_{j+1} and β_{j+1} can be chosen in any manner to ensure that $(v_{j+1}, w_{j+1}) = 1$. It is only necessary that

$$\delta_{j+1} \beta_{j+1} = (\bar{v}_{j+1}, \bar{w}_{j+1})$$

The vectors v_j belong to $\mathcal{K}_m(A, v_1)$ and w_j to $\mathcal{K}_m(A^T, w_1)$

The Lanczos procedure generates a matrix

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & & & \\ \delta_2 & \alpha_2 & \beta_3 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m & & \\ & & & \delta_m & \alpha_m & & \end{bmatrix}$$

In algorithm of the previous slide, δ_j 's are positive and $\beta_j = \pm \delta_j$

Properties and Relations

If the Lanczos algorithm does not break down before step m , then the vectors $v_i, i = 1, \dots, m$ and $w_j, j = 1, \dots, m$, form a biorthogonal system, i.e.,

$$(v_i, w_j) = \delta_{ij} \quad 1 \leq i, j \leq m.$$

Furthermore, $\{v_i\}_{i=1, \dots, m}$ is a basis of $\mathcal{K}_m(A, v_1)$ and $\{w_j\}_{j=1, \dots, m}$ is a basis of $\mathcal{K}_m(A^T, w_1)$, and the following relations hold

$$\begin{aligned} AV_m &= V_m T_m + \delta_{m+1} v_{m+1} e_m^T \\ A^T W_m &= W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T \\ W_m^T AV_m &= T_m \end{aligned}$$

Biconjugate Gradient (BCG)

1. Compute $r^{(0)} = b - Au^{(0)}$, choose $r_*^{(0)}$ such that $(r^{(0)}, r_*^{(0)}) \neq 0$
2. Set $p_0 = r^{(0)}, p_0^* = r_*^{(0)}$
3. For $j = 1, 2, \dots$, until convergence, do:
 4. $\alpha_j = (r^{(j)}, r_*^{(j)}) / (Ap_j, p_j^*)$
 5. $u^{(j+1)} = u^{(j)} + \alpha_j p_j$
 6. $r^{(j+1)} = r^{(j)} - \alpha_j Ap_j$
 7. $r_*^{(j+1)} = r_*^{(j)} - \alpha_j A^T p_j^*$
 8. $\beta_j = (r^{(j+1)}, r_*^{(j+1)}) / (r^{(j)}, r_*^{(j)})$
 9. $p_{j+1} = r_{j+1} + \beta_j p_j$
 10. $p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$
11. End do

Note that two matrix-vector products are needed at each iteration

Practical Implementations

Deal with the breakdowns

$$\delta_{j+1} = (\bar{v}_{j+1}, \bar{w}_{j+1}) = 0$$

lucky breakdown if either (both) $\bar{v}_{j+1} = 0$ or (and) $\bar{w}_{j+1} = 0$

serious breakdown if $(\bar{v}_{j+1}, \bar{w}_{j+1}) = 0$, but both $\bar{v}_{j+1} \neq 0$ and $\bar{w}_{j+1} \neq 0$

look-ahead Lanczos algorithms: v_{j+2} and w_{j+2} can be defined even v_{j+1}, w_{j+1} are not defined. The strategy is to skip the $(j+1)$ th step and directly go to the $(j+2)$ th step.

The drawback of *look-ahead* implementations is the additional costs.

Issues of BCG

BCG implicitly compute the solutions with A and A^T . Half of the operations war wasted. Not very efficient in terms of computational cost

Numerical Experiments with BCG

Matrix	Iters	Kflops	Residual	Error
F2DA	163	2974	0.17E-03	0.86E-04
F3D	123	10768	0.34E-04	0.17E-03
ORS	301	6622	0.50E-01	0.37E-02

The first line represents 81 steps of BCG. One extra matrix-vector to compute initial residual

Other Lanczos Methods

There are other methods based on the Lanczos algorithm. Most notable is the *Quasi-Minimal Residual* (QMR) method.

However, the most important variants of these methods are their transpose-free version. In such implementations, the matrix A^T is not required. They are

- Conjugate Gradient Squared (CGS)
- Biconjugate Gradient Stabilized (BiCGSTAB)
- Transpose-Free QMR (TFQMR)

CGS may exhibit irregular convergence behavior residual norm may jump and decrease

Parallelizing GMRES

The major time-consuming operations of GMRES are:

- matrix vector multiplications
- vector updates
- inner products
- orthogonalization of the vector Av_i against all previous v 's (modified Gram-Schmidt)

The first two parts are relatively easy to parallelize. The inner products may cause some problems on parallel computers with many processors. Vendor supports for global reduction operations are emerging

GMRES and BiCGSTAB

GMRES really minimizes residual norms at the expenses of high storage cost. BiCGSTAB and TFQMR do not minimize anything, but they use almost fixed memory space with respect to iteration counts.

The performances of these Krylov subspace methods are somehow similar. BiCGSTAB and TFQMR may be advantageous if many iterations are required. GMRES is advantages if a restart is not required.

All are comparable if they work with a preconditioner and the quality of the preconditioner seems to make a difference

Parallelizing Gram-Schmidt

Modified Gram-Schmidt consists of a sequence of subprocesses of the form:

- compute $\alpha = (y, v)$
- compute $\tilde{y} = y - \alpha v$

The outer loop is sequential. The inner loop with the inner product and SAXPY operations may be parallelized on a small number of processors.

An alternative is to use *standard* Gram-Schmidt procedure with matrix operations

$$\tilde{y} = y - VV^T$$

to increase computation granularity

Methods for Normal Equations

If a matrix A is nonsymmetric, we can solve an equivalent system with an SPD $A^T A$

$$A^T A u = A^T b,$$

which is the *normal equation* associated with the least-squares problem (over-determined)

$$\text{minimize } \|b - Au\|_2 \quad \text{NR}$$

We can also form an normal equation

$$AA^T x = b$$

for under-determined system for the least-squares problem

$$\text{minimize } \|u_* - A^T x\|_2 \quad \text{NE}$$

with $A^T x = u$

Gauss-Seidel on Normal Equations

Considering to solve $Au = b$ in the form of

$$A^T Au = A^T b \quad \text{NR}$$

Let the new iterate at the i th step be

$$u_{\text{new}} = u + \delta_i e_i$$

we force the i th component of residual vanish

$$(A^T b - A^T A(u + \delta_i e_i), e_i) = 0$$

with $r = b - Au$, we obtain

$$\delta_i = \frac{(r, Ae_i)}{\|Ae_i\|_2^2}$$

The new iterate and residual are

$$\begin{aligned} u_{\text{new}} &= u + \delta_i e_i \\ r_{\text{new}} &= r - \delta_i Ae_i \end{aligned}$$

Gauss-Seidel on Normal Equations

Considering to solve $Au = b$ in the form of

$$AA^T x = b \quad \text{NE}$$

Let the new iterate at the i th step be

$$x_{\text{new}} = x + \delta_i e_i$$

we force the i th component of residual vanish

$$(b - AA^T(x + \delta_i e_i), e_i) = 0$$

with $r = b - AA^T x$, we obtain

$$\delta_i = \frac{(r, e_i)}{\|Ae_i\|_2^2} = \frac{b_i - (A^T x, A^T e_i)}{\|A^T e_i\|_2^2}$$

The new iterate with the u variable is

$$u_{\text{new}} = u + \delta_i A^T e_i$$

CG for Normal Residual

The conjugate gradient method can be applied to the normal equations in normal residual form with A being replaced by $A^T A$

1. $r^{(0)} = b - Au^{(0)}$, $z^{(0)} = A^T r^{(0)}$, $p_0 = z^{(0)}$
2. For $j = 1, 2, \dots$, until convergence, do:
 3. $w_j = Ap_j$
 4. $\alpha_j = \|z^{(j)}\|_2^2 / \|w_j\|_2^2$
 5. $u^{(j+1)} = u^{(j)} + \alpha_j p_j$
 6. $r^{(j+1)} = r^{(j)} - \alpha_j w_j$
 7. $z^{(j+1)} = A^T r^{(j+1)}$
 8. $\beta_j = \|z^{(j+1)}\|_2^2 / \|z^{(j)}\|_2^2$
 9. $p_{j+1} = z^{(j+1)} + \beta_j p_j$
10. End do

CG for Normal Residual (II)

The Krylov subspace for CGNR is

$$\begin{aligned} & \mathbf{u}^{(0)} + \mathcal{K}_m(A^T A, A^T \mathbf{r}^{(0)}) \\ &= \mathbf{u}^{(0)} + \text{span}\{A^T \mathbf{r}^{(0)}, A^T A A^T \mathbf{r}^{(0)}, \dots, \\ & \quad (A^T A)^{m-1} A^T \mathbf{r}^{(0)}\}. \end{aligned}$$

The approximate solution $\mathbf{u}^{(m)}$ minimizes

$$\begin{aligned} f(\mathbf{u}) &= (A^T A(\mathbf{u}^{(*)} - \mathbf{u}), (\mathbf{u}^{(*)} - \mathbf{u})) \\ &= (A(\mathbf{u}^{(*)} - \mathbf{u}), A(\mathbf{u}^{(*)} - \mathbf{u})) \\ &= \|\mathbf{b} - A\mathbf{u}\|_2^2 \end{aligned}$$

Hence, both CGNR and GMRES minimize 2-norm residual, but in different Krylov subspaces

CG for Normal Error

Conjugate gradient method for normal error equation can be written analogously

1. $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{u}^{(0)}$, $\mathbf{p}_0 = A^T \mathbf{r}^{(0)}$
2. For $j = 1, 2, \dots$, until convergence, do:
 3. $\alpha_j = (\mathbf{r}^{(j)}, \mathbf{r}^{(j)}) / (\mathbf{p}_j, \mathbf{p}_j)$
 4. $\mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} + \alpha_j \mathbf{p}_j$
 5. $\mathbf{r}^{(j+1)} = \mathbf{r}^{(j)} - \alpha_j A \mathbf{p}_j$
 6. $\beta_j = (\mathbf{r}^{(j+1)}, \mathbf{r}^{(j+1)}) / (\mathbf{r}^{(j)}, \mathbf{r}^{(j)})$
 7. $\mathbf{p}_{j+1} = A^T \mathbf{r}^{(j+1)} + \beta_j \mathbf{p}_j$
8. End do

CGNE and CGNR find approximations from the same subspace, CGNR minimizes residual, CGNE minimizes error

CG for Normal Residual (III)

One drawback for normal equation approaches is that the condition number of $A^T A$ doubles that of A and CG convergence may be slow

$$\text{Cond}_2(A^T A) = \|A^T A\|_2 \|(A^T A)^{-1}\|_2$$

$$\text{Cond}_2(A^T A) = \|A\|_2^2 \|A^{-1}\|_2^2 = \text{Cond}_2^2(A)$$

Numerical Experiments with CGNR

Matrix	Iters	Kflops	Residual	Error
F2DA	300	4847	0.23E+02	0.62E+00
F3D	300	23704	0.42E+00	0.15E+00
ORS	300	5981	0.30E+02	0.60E-02

No of the problems were solved by CGNR. Preconditioners may improve convergence, but are difficult to construct

After Notes

Note that Jacobi iteration can be applied to the normal equations as the Gauss-Seidel. Both methods can also be accelerated by a relaxation parameter ω to form an SOR method. Jacobi, Gauss-Seidel, and SOR are used to precondition the conjugate gradient method. They have the advantages of not forming the matrix $A^T A$ or AA^T explicitly. Since the product matrix may be much more dense than A

Normal equation approaches are receiving some attention recently, but existing methods are still not very competitive with GMRES