Lecture 23

Solve $Ax = d$: Generalized Minimum Residual Method

(Lecture notes taken by Zhibin Deng and Xiaoyin Ji)

- Comparison of Block Tri-diagonal (BT), Conjugate Gradient (CG) and Generalized Minimum Residual (GMRES)

<table>
<thead>
<tr>
<th>nx=ny=nz</th>
<th>L_2 Error</th>
<th>Compute Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BT</td>
<td>CG</td>
</tr>
<tr>
<td>6</td>
<td>4.9582</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.2461</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0.3120</td>
<td>1.6104</td>
</tr>
<tr>
<td>41</td>
<td>0.0782</td>
<td>93.2287</td>
</tr>
</tbody>
</table>

The result is obtained in MATLAB 2010, Core i-7-920.

Remark:
1. Block Tri-diagonal Algorithm requires the matrix form is block tri-diagonal. However, there are less linear solvers in it, hence the compute time is less. This method is efficient if the matrix is sparse.
2. Conjugate Gradient method requires the matrix form is symmetric positive definite. It only needs to store the last direction.
3. GMRES has no special requirement in matrix form, however, it has to store all directions in the process.

- Krylov Space.

**Definition:** Subspace $\mathcal{K}_{m+1}$ spanned by vectors $\{r^0, Ar^0, A^2r^0, A^3r^0, \ldots, A^mr^0\}$ is called Krylov space, where $r^0 = d - Ax^0$ is an initial vector.

Note that $Ax = d \iff r(x) = d - Ax = 0 \iff \min_y r^T(y) r(y) = 0$

The main idea of GMRES algorithm is to approximate the solution of system $Ax = d$ by a linear combination of Krylov vectors $A^i r^0$. Define $x^{m+1} = x^0 + \alpha_0 r^0 + \alpha_1 Ar^0 + \cdots + \alpha_m A^m r^0$. We need to find $\alpha = (\alpha_0, \ldots, \alpha_m)$ such that $r^T(x^{m+1}) r(x^{m+1})$ is minimized.

Let $y = x^0 + \mathcal{K}_{m+1}$, and $\mathcal{K}_{m+1} = \{ z | z = \sum_{i=0}^{m} \alpha_i A^i r^0 \}$, then

$$r^T(x^{m+1}) r(x^{m+1}) = \min_{y \in x^0 + \mathcal{K}_{m+1}} r^T(y) r(y)$$
Note: \[ r(x^{m+1}) = d - Ax^{m+1} \]
\[ = d - A(x^0 + \alpha_0 r^0 + \alpha_1 A r^0 + \cdots + \alpha_m A^m r^0) \]
\[ = [I - (\alpha_0 A + \alpha_1 A^2 + \cdots + \alpha_m A^{m+1})] r^0. \]

Let \( q_{m+1}(A) = I - (\alpha_0 A + \alpha_1 A^2 + \cdots + \alpha_m A^{m+1}) \), which is polynomial of matrix \( A \).

When matrix \( A \) is invertible and \( m = n - 1 \), consider matrix \( A \)'s characteristic polynomial. Then, \( f(A) = 0 \), and we have a solution \( r(x) = 0 \). However, generally we have \( m \ll n \) when algorithm terminates.

- **Hessenburg matrix.**

--efficient method for finding \( \alpha \)

\[
\begin{array}{ll}
{n \times m} & {\mathbf{K} = [A r^0, \ldots, A^{m-1} r^0]} \\
{n \times n} & {\mathbf{K}_m = \alpha} \\
{n \times 1} & {\mathbf{r}^0, \alpha \in \mathbb{R}^n, n \gg m} \\
\end{array}
\]

\[ x^m = x^0 + \mathbb{K}_m \]

This is a LS (least square) problem. We use QR to solve the normal equation. In order to deal with that, we replace \( \mathbb{K}_m \) by \( \mathbb{V}_m \), whose columns \( v_j \)s are orthogonal. For example,

**Col 1:** \( r^0 = b v_1 \)

\[ v_1^T v_1 = 1 \]
\[ b = \sqrt{r^0 r^0} \]
\[ v_1 = r^0 / b \]

**Col 2:** \( A \mathbb{K}_1 \subset \mathbb{K}_2 \)

\[ Av_1 = v_1 h_{11} + v_2 h_{21} \]
\[ v_1^T Av_1 = v_1^T v_1 h_{11} + v_1^T v_2 h_{21} = h_{11} \]
\[ z = Av_1 - v_1 h_{11} = v_2 h_{21} \text{, and note } v_2^T v_2 = 1 \]
\[ z^T z = h_{21}^2 \]
\[ h_{21} = \sqrt{z^T z} \]
\[ v_2 = z/h_{21} \]
\[ A[v_1] = [v_1 \ v_2 \begin{bmatrix} h_{11} \\ h_{21} \end{bmatrix} \]

**Col. 3**: \( AK \subseteq K_3 \)

\[ Av_2 = v_1h_{12} + v_2h_{22} + v_3h_{32} \]
\[ A[v_1 \ v_2] = [v_1 \ v_2 \ v_3] \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ h_{31} & h_{32} \end{bmatrix}_{3 \times 2} \]

... 
\[ A[v_1 \ \ldots \ v_m] = [v_1 \ \ldots \ v_m \ v_{m+1}] \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}_{m+1 \times m \text{ Hessenberg}} \]

**Reduction Theorem**: The following are equivalent

1. LS problem \( AK_m \alpha = r^0 \) has solution \( x^{m+1} = K_m \alpha + x^0 \)
2. LS problem \( A\beta = \beta_{m+1} e_1 b \) has solution \( x^{m+1} = \beta_{m} + x^0 \)
3. LS problem \( H\beta = e_1 b \) has solution \( x^{m+1} = \beta_{m} + x^0 \)

Note that we need matrix \( \beta_m \) to compute new iteration solution \( x^{m+1} \), that’s why we have to store all the directions \( v_i \).

- **Givens transforms and QR factors.**

According to Reduction Theorem, we convert the LS problem \( AK_m \alpha = r^0 \) into a new LS problem \( H\beta = e_1 b \). Note that \( H \) is a Hessenberg matrix, we can use Givens rotations to solve the new LS problem easily if \( m \ll n \).
- **GMRES algorithm.**

Choose initial guess $x^0$, let $r^0 = d - Ax^0$, $b = \sqrt{r^0}r^0$, $V(:,1) = r^0/b$,

for $k = 1$ to $\maxm$

$V(:,k + 1) = AV(:,k)$

Compute the columns of $V(:,k + 1)$ and $H(:,k + 1)$

Compute the QR factors of $H$

Test for convergence

$x^{k+1} = x^0 + V\beta$

gmresfull.m Code:

clear;
% Solves a block tridiagonal non SPD from the finite difference method of
% $- u_{xx} - u_{yy} + a_1 u_x + a_2 u_y + c u = f(x,y)$
% with zero boundary conditions.
% Uses the generalized minimum residual method.
% Define the NxN coefficient matrix AA where N = n^2.

n = 10; N = n*n;
errtol = 0.001;
kmax = 200;
A = zeros(n); a1 = 10; a2 = 1; c = 1; h = 1./(n+1);
for i = 1:n
    A(i,i) = 4 + c*h*h + a1*h + a2*h;
    if (i>1)
        A(i,i-1) = -1 - a1*h ;
    end
    if (i<n)
        A(i,i+1) = -1;
    end
end
I  = eye(n);
II = I + I*a2*h;
AA = zeros(N);
for i = 1:n
    newi = (i-1)*n + 1;
    lasti = i*n;
    AA(newi:lasti,newi:lasti) = A;
    if (i>1)
        AA(newi:lasti,newi-n:lasti-n) = -II;
    end
    if (i<n)
        AA(newi:lasti,newi+n:lasti+n) = -I;
    end
end
\[ u = \text{zeros}(N, 1); \]
\[ r = \text{zeros}(N, 1); \]
\[ \text{for } j = 1:n \]
\[ \quad \text{for } i = 1:n \]
\[ \quad \quad I = (j-1)n + i; \]
\[ \quad \quad r(I) = h*h*2000*(1+\sin(\pi*(i-1)*h)*\sin(\pi*(j-1)*h)); \]
\[ \quad \text{end} \]
\[ \text{end} \]
\[ \rho = (\text{r'*r})^{.5}; \]
\[ \text{errtol} = \text{errtol}\cdot\rho; \]
\[ g = \rho*\text{eye}(kmax+1, 1); \]
\[ v(:, 1) = r/\rho; \]
\[ k = 0; \]
\[ \% \text{Begin gmres loop.} \]
\[ \text{while } (\rho > \text{errtol}) \text{ & } (k < kmax) \]
\[ \quad k = k+1; \]
\[ \% \text{Matrix vector product.} \]
\[ v(:, k+1) = AA*\text{v}(; : , k); \]
\[ \% \text{Begin modified GS. May need to reorthogonalize.} \]
\[ \quad \text{for } j = 1:k \]
\[ \quad \quad h(j, k) = v(:, j)'*v(:, k+1); \]
\[ \quad \quad v(:, k+1) = v(:, k+1) - h(j, k)*v(:, j); \]
\[ \quad \text{end} \]
\[ \quad h(k+1, k) = (v(:, k+1)'*v(:, k+1))^{.5}; \]
\[ \quad \text{if } (h(k+1, k) \neq 0) \]
\[ \quad \quad v(:, k+1) = v(:, k+1)/h(k+1, k); \]
\[ \quad \text{end} \]
\[ \% \text{Apply old Givens rotations to } h(1:k, k). \]
\[ \% \text{if } k>1 \]
\[ \quad \text{for } i=1:k-1 \]
\[ \quad \quad \text{hik} = c(i)*h(i, k) - s(i)*h(i+1, k); \]
\[ \quad \quad \text{hipk} = s(i)*h(i, k) + c(i)*h(i+1, k); \]
\[ \quad \quad h(i, k) = \text{hik}; \]
\[ \quad \quad h(i+1, k) = \text{hipk}; \]
\[ \quad \text{end} \]
\[ \text{end} \]
\[ \text{normh} = \text{norm}(h(k:k+1, k)); \]
\[ \% \text{May need better Givens implementation.} \]
\[ \% \text{Define and apply new Givens rotations to } h(k:k+1, k). \]
\[ \% \text{if } \text{normh} \neq 0 \]
\[ \quad \text{c(k) = } h(k, k)/\text{normh}; \]
\[ \quad \text{s(k) = } -h(k+1, k)/\text{normh}; \]
\[ \quad \text{h(k, k) = } c(k)*h(k, k) - s(k)*h(k+1, k); \]
\[ \quad \text{h(k+1, k) = 0;} \]
\[ \quad \text{gk = } c(k)*g(k) - s(k)*g(k+1); \]
\[ \quad \text{gkp = } s(k)*g(k) + c(k)*g(k+1); \]
\[ \quad \text{g(k) = } gk; \]
\[ \quad \text{g(k+1) = } gkp; \]
\[ \text{end} \]
\[ \text{rho} = \text{abs}(g(k+1)); \]
\[ \text{mag(k) = rho;} \]
\[ \text{end} \]
\[ \% \text{End of gmres loop.} \]
% $h(1:k,1:k)$ is upper triangular matrix in QR.
y = h(1:k,1:k) \backslash g(1:k);
% Form linear combination.
for i = 1:k
    u(:) = u(:) + v(:,i)*y(i);
end
k
figure(1)
semilogy(mag)
for j = 1:n
    for i = 1:n
        I = (j-1)*n + i;
        u2d(i,j) = u(I);
    end
end
figure(2)
uu2d = zeros(n+2);
uu2d(2:n+1,2:n+1) = u2d;
mesh(uu2d)