MATH 571: Computational Assignment #2

Due on Tuesday, November 26, 2013

TTH 12:40pm

Wenqiang Feng

Contents

Let $Ndim$ to be the Dimension of the matrix and $Niter$ to be the iterative numbers. In the whole report, b was generated by Ax, where x is a corresponding vector and x's entries are random numbers between 0 and 10. The initial iterative values of x are given by $\vec{0}$.

Problem 1

1. Listing 1 shows the implement of Jacobi Method.
2. Listing 2 shows the implement of SOR Method.
3. The numerical results:
   (a) From the records of the iterative number, I got the following results:
   For case (2), the Jacobi Method is not convergence, because it has a big Condition Number. For case (1) and case (3), if $Ndim$ is small, roughly speaking, $Ndim \leq 10 - 20$, then the $Ndim$ and $Niter$ have the roughly relationship $Niter = log(Ndim + C)$, when $Ndim$ is large, the $Niter$ is not depent on the $Ndim$ (see Figure (1)).

   (b) When $\omega = 1$, the SOR Method degenerates to the Gauss-seidel Method. For Gauss-seidel Method, I get the similar results as Jacobi Method (see Figure (2)). But, the Gauss-Seidel Method is more stable than Jacobi Method and case (3) is more stable than case (1) (see Figure (1) and Figure (2)).

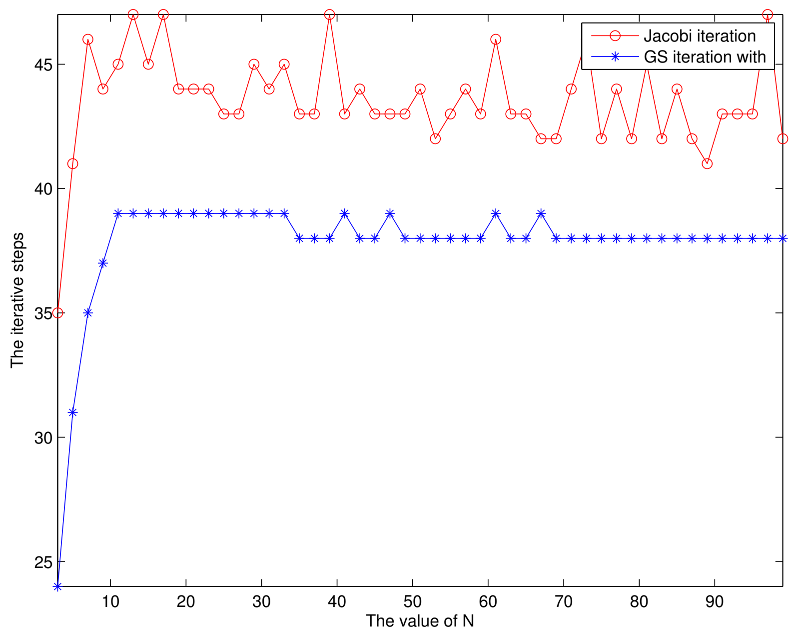

Figure 1: The relationship between $Ndim$ and $Niter$ for case(1)

   (c) The optimal w
      i. For case (1), the optimal w is around 1, but this optimal w is not optimal for all (see Figure (3) and Figure (4));

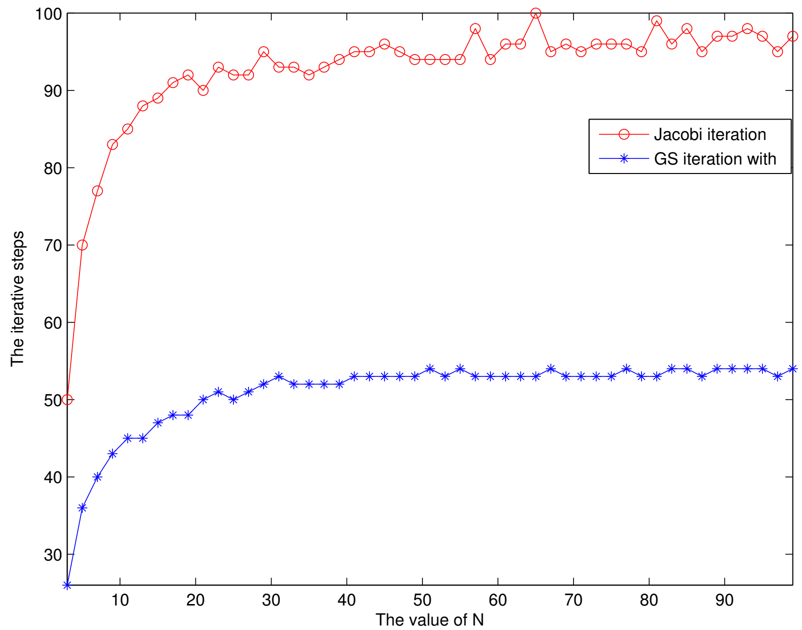

Figure 2: The relationship between $Ndim$ and $Niter$ for case (3)

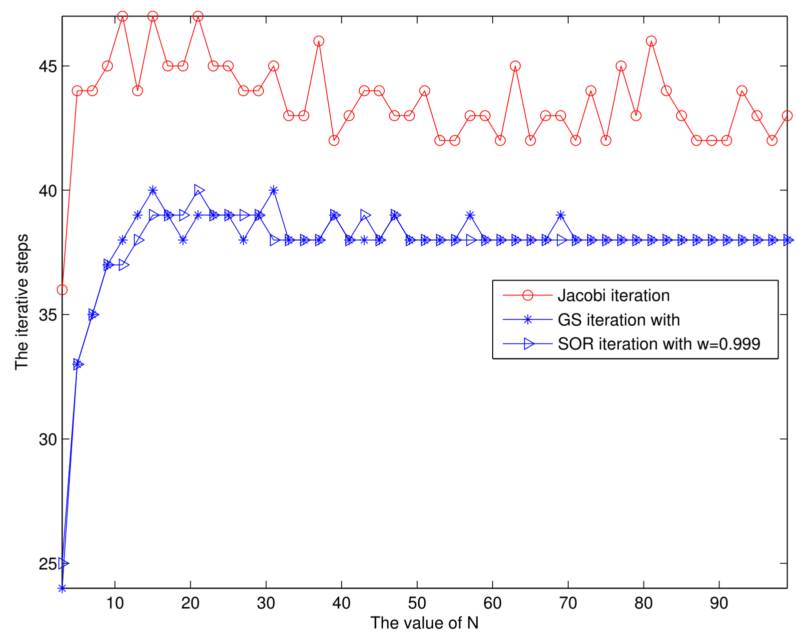

Figure 3: The relationship between $Ndim$ and $Niter$ for case(1)

ii. For case (2), In general, the SOR Method is not convergence, but SOR is convergence for some small $Ndim$;

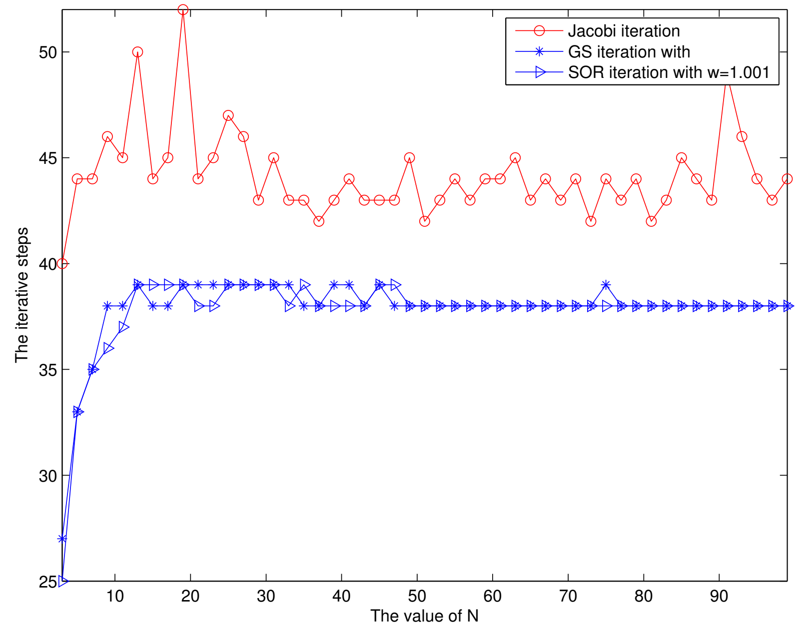

Figure 4: The relationship between $Ndim$ and $Niter$ for case(1)

iii. For case(3), the optimal w is around 1.14; This numerical result is same as the theoretical result. Let D $= diag(diag(A))$; $E = A - D$; $T = D\backslash E$,

$$w_{opt} = \frac{2}{\sqrt{1-\rho(T)^2}} \approx 1.14.$$

Where, the $\rho(T)$ is the spectral radius of T (see Figure (5)).

(d) In general, for the convergence case, $Niter_{Jacobi} > Niter_{Gauss-Sediel} > Niter_{SOR_{opt}}$. I conclude that SOR_{opt} is more efficient than $Gauss - Sediel$ and $Gauss - Sediel$ is more efficient than $Jacobi$ for convergence case (see Figure (5)).

Listing 1: Jacobi Method

```
function [x iter]=jacobi(A,b,x,tol,max_iter)
% jacobi:  Solve the linear system with Jacobi iterative algorithm
%
% USAGE
%        jacobi(A,b,x0,tol)
%
% INPUT
%      A: N by N LHS coefficients matrix
%      b: N by 1 RHS vector
%      x: Initial guess
%      tol: The stop tolerance
%      max_iter: maxmum iterative steps
%
% OUTPUT
%     x: The solutions
```

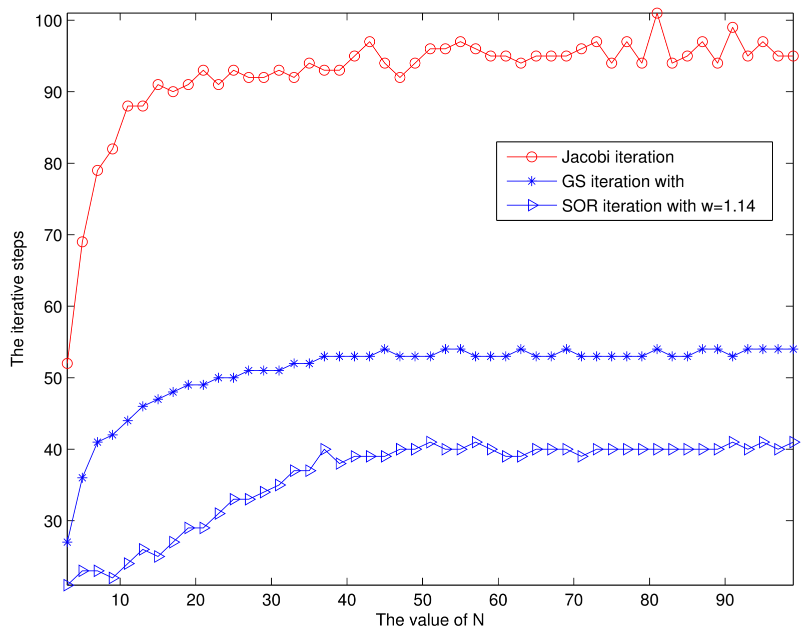

Figure 5: The relationship between $Ndim$ and $Niter$ for case(3)

```
%     iter: iterative steps
%
% AUTHOR
%    Wenqiang Feng
%    Department of Mathematics
%    University of Tennessee at Knoxville
%    E-mail: wfeng@math.utk.edu
%    Date:   11/13/2013
n=size(A,1);

% Set default parameters
if (nargin<3), x=zeros(n,1);tol=1e-16;max_iter=500;end;
%Initial some parameters
error=norm(b - A*x);
iter=0 ;
%split the matrix for Jacobi interative method
D = diag(diag(A));
E=D-A;

while (error>tol&&iter<max_iter)
    x1=x;
    x= D\(E*x+b);
    error=norm(x-x1);
    iter=iter+1;
end
```

Listing 2: SOR Method

Problem 1 continued on next page...

```
function [x iter]=sor(A,b,w,x,tol,max_iter)
% jacobi:  Solve the linear system with SOR iterative algorithm
%
% USAGE
%        jacobi(A,b,epsilon,x0,tol,max_iter)
%
% INPUT
%      A: N by N LHS coefficients matrix
%      b: N by 1 RHS vector
%      w: Relaxation parameter
%      x: Initial guess
%      tol: The stop tolerance
%      max_iter: maxmum iterative steps
%
% OUTPUT
%     x: The solutions
%     iter: iterative steps
%
% AUTHOR
%    Wenqiang Feng
%    Department of Mathematics
%    University of Tennessee at Knoxville
%    E-mail: wfeng@math.utk.edu
%    Date:   11/13/2013

n=size(A,1);
% Set default parameters
if (nargin<4), x=zeros(n,1);tol=1e-16;max_iter=500;end;
%Initial some parameters
error=norm(b - A*x)/norm( b );
iter=0 ;
%split the matrix for Jacobi interative method
     D=diag(diag( A ));
     b = w * b;
     M =  w * tril( A, -1 ) + D;
     N = -w * triu( A,  1 ) + ( 1.0 - w ) * D;

while (error>tol&&iter<max_iter)
    x1=x;
    x= M\(N*x+b);
    error=norm(x-x1)/norm( x );
    iter=iter+1;
end
```

Problem 2

1. Listing 3 shows the implement of ADI Method.

2. Yes, The Σ and Λ are the SPD matrices. Moreover, they are commute, since $\Sigma\Lambda = \Lambda\Sigma$.

3. The optimal τ for the ADI method:
   The optimal τ for the ADI method is same as the $SSOR$ and SOR method. Let D $= diag(diag(A))$; $E = A - D$; $T = D\backslash E$,

$$\tau_{opt} = \frac{2}{\sqrt{1-\rho(T)^2}}.$$

   Where, the $\rho(T)$ is the spectral radius of T.

4. The expression of x^{k+1}:

   By adding and subtracting scheme (1) and scheme (2), we get that

$$(I+\tau A_1)(I+\tau A_2)x^{k+1} - (I-\tau A_1)(I-\tau A_2)x^k = 2\tau f. \tag{1}$$

5. The expression of the error's control:

$$(I+\tau A_1)(I+\tau A_2)e^{k+1} = (I-\tau A_1)(I-\tau A_2)e^k. \tag{2}$$

6. Now, I will show $[x,y] = (A_1A_2x, y)$ is an inner product, i.e, I will show the $||x||_B^2 = [x,x]$ satisfies parallelogram law:
   It's easy to show that the B-norm $||x||_B^2 = [x,x]$ satisfies the parallelogram law,

$$\begin{aligned}
||x+y||_B^2 + ||x-y||_B^2 &= (A_1A_2(x+y), x+y) + (A_1A_2(x-y), x-y)\\
&= (A_1A_2x, x) + (A_1A_2x, y) + (A_1A_2y, x) + (A_1A_2y, y)\\
&\quad +(A_1A_2x, x) - (A_1A_2x, y) - (A_1A_2y, x) + (A_1A_2y, y)\\
&= 2(||x||_B^2 + ||y||_B^2).
\end{aligned}$$

   So, The norm space can induce a inner product, so $[x,y] = (A_1A_2x, y)$ is a inner product.

7. Take inner product (2) with $e^{k+1} + e^k$, we get,

$$\left((I+\tau A_1)(I+\tau A_2)e^{k+1}, e^{k+1}+e^k\right) = \left((I-\tau A_1)(I-\tau A_2)e^k, e^{k+1}+e^k\right). \tag{3}$$

   By using the distribution law, we get

$$\begin{aligned}
&\left(e^{k+1}, e^{k+1}\right) + \tau\left(Ae^{k+1}, e^{k+1}\right) + \tau^2\left(A_1A_2e^{k+1}, e^{k+1}\right) &(4)\\
&+\left(e^{k+1}, e^k\right) + \tau\left(Ae^{k+1}, e^k\right) + \tau^2\left(A_1A_2e^{k+1}, e^k\right) &(5)\\
&= \left(e^k, e^{k+1}\right) - \tau\left(Ae^k, e^{k+1}\right) + \tau^2\left(A_1A_2e^k, e^{k+1}\right) &(6)\\
&+\left(e^k, e^k\right) - \tau\left(Ae^k, e^k\right) + \tau^2\left(A_1A_2e^k, e^k\right). &(7)
\end{aligned}$$

   Since, $A_1A_2 = A_2A_1$, so $\left(A_1A_2e^{k+1}, e^k\right) = \left(A_1A_2e^k, e^{k+1}\right)$. Therefore, (4) reduces to

$$\begin{aligned}
&\left(e^{k+1}, e^{k+1}\right) + \tau\left(A(e^{k+1}+e^k), e^{k+1}+e^k\right) + \tau^2\left(A_1A_2e^{k+1}, e^{k+1}\right) &(8)\\
=\ &\left(e^k, e^k\right) + \tau^2\left(A_1A_2e^k, e^k\right). &(9)
\end{aligned}$$

   Therefore,

$$||e^{k+1}||_2^2 + \tau||e^{k+1}+e^k||_A^2 + \tau^2||e^{k+1}||_B^2 = ||e^k||_2^2 + \tau^2||e^k||_B^2. \tag{10}$$

Summing over k from 0 to K, we get

$$||e^{K+1}||_2^2 + \tau \sum_{k=0}^{K} ||e^{k+1} + e^k||_A^2 + \tau^2 ||e^{K+1}||_B^2 = ||e^0||_2^2 + \tau^2 ||e^0||_B^2. \tag{11}$$

Therefore, from (11), we get $||e^{k+1} + e^k||_A^2 \to 0\ \forall \tau > 0$. So $\frac{1}{2}(x^{k+1} + x^k) \to x$ with respect to $||\cdot||_A$.

Listing 3: ADI Method

```
function [x iter]=adi(A,b,A1,A2,tau,x,tol,max_iter)
% jacobi:  Solve the linear system with ADI algorithm
%
% USAGE
%         adi(A,b,A1,A2,tau,x,tol,max_iter)
%
% INPUT
%       A: N by N LHS coefficients matrix
%       b: N by 1 RHS vector
%       A1: The decomposition of A: A=A1+A2 and A1*A2=A2*A1
%       A2: The decomposition of A: A=A1+A2 and A1*A2=A2*A1
%       x: Initial guess
%       tol: The stop tolerance
%       max_iter: maxmum iterative steps
%
% OUTPUT
%      x: The solutions
%      iter: iterative steps
%
% AUTHOR
%    Wenqiang Feng
%    Department of Mathematics
%    University of Tennessee at Knoxville
%    E-mail: wfeng@math.utk.edu
%    Date:   11/13/2013
n=size(A,1);

% Set default parameters
if (nargin<6), x=zeros(n,1);tol=1e-16;max_iter=300;end;

%Initial some parameters
error=norm(b - A*x);
iter=0 ;
I=eye(n);

while (error>tol&&iter<max_iter)
    x1=x;
    x=(tau*I+A1)\((tau*I-A2)*x+b); % the first half step
    x=(tau*I+A2)\((tau*I-A1)*x+b); % the second half step
    error=norm(x-x1);
    iter=iter+1;
end
```