MATH 5620 HW 2 SOLUTIONS

Problem 1 Let λ, $u \neq 0$ be an eigenpair of A, then:

$$A^i u = \lambda^i u$$

$$\Rightarrow \quad p(A)u = \left(\sum_{i=0}^{n} a_i A^i\right) u$$

$$= \left(\sum_{i=0}^{n} a_i \lambda^i\right) u$$

$$= p(\lambda) u$$

Thus $p(\lambda)$, $u \neq 0$ is an eigenpair of $p(A)$

Problem 2 To show convergence of Richardson's method we need that for some induced norm:

$$\| I - A \| < 1.$$

Here:

$$\| I - A \|_\infty = \left\| \begin{bmatrix} 0 & a_{12} & \cdots & & a_{1n} \\ a_{21} & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & & \ddots & & \\ \vdots & & & \ddots & a_{n-1n} \\ a_{n1} & & & a_{nn-1} & 0 \end{bmatrix} \right\|_\infty$$

$$= \max_{i=1 \ldots n} \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| < 1.$$

$\Rightarrow$ Richardson iteration converges.

Problem 3 (a) Let $\|x\|' = \|Sx\|$, S invertible, we verify $\|\circ\|'$ satisfies axioms of a norm:

i) $\|x\|' = \|Sx\| \geq 0$, since $\|y\| \geq 0$.

ii) $\|x\|' = 0 \Rightarrow \|Sx\| = 0 \Rightarrow Sx = 0$

$\Rightarrow x = 0$, since S is non-sing.

iii) $\|\lambda x\|' = \|\lambda Sx\| = |\lambda| \, \|Sx\| = |\lambda| \, \|x\|'$

iv) $\|x+y\|' = \|S(x+y)\| \leq \|Sx\| + \|Sy\|$
$= \|x\|' + \|y\|'$

(b) Let $\|A\|' = \|SAS^{-1}\|$.

$$= \sup_{x \neq 0} \frac{\|SAS^{-1}x\|}{\|x\|}$$

letting $y = S^{-1}x$

$$= \sup_{y \neq 0} \frac{\|SAy\|}{\|Sy\|}$$

$$= \sup_{y \neq 0} \frac{\|Ay\|'}{\|y\|'}.$$

An alternative proof is to verify axioms i) – iv) which follow in a similar way as (a) and:

$$\|AB\|' = \|SABS^{-1}\| = \|SAS^{-1}SBS^{-1}\| \leq \underbrace{\|SAS^{-1}\|}_{\|A\|'} \underbrace{\|SBS^{-1}\|}_{\|B\|'}$$

QED.

```
%%%% problem 4 %%%%

% Sytems come from 7.3.6

% system matrices
As{1}=[ 4  1   -1
       -1  3    1
        2  2    5];

As{2}=[-2  1  1/2
        1 -2 -1/2
        0  1    2];

% system right hand sides
bs{1}=[5;-4;1];
bs{2}=[4;-4;0];

% system names
names{1} = 'part a'; names{2} = 'part b';

% to compare with results in book
%As{3}=[3 -1 1; 3 6 2; 3 3 7]; bs{3}=[1;0;4]; names{3}='7.3.1a';

% do computations
for i=1:length(As),
 fprintf([names{i},' ','Jacobi\n']);
 [xjac,jjac] = jacobi(As{i},bs{i},zeros(size(bs{i})),100,1e-3);
 xjacs{i} = xjac; jjacs{i} = jjac;
 fprintf([names{i},' ','Gauss-Seidel\n']);
 [xgs,jgs] = gs(As{i},bs{i},zeros(size(bs{i})),100,1e-3);
 xgss{i} = xgs; jgss{i} = jgs;
 fprintf([names{i},' ','SOR\n']);
 [xsor,jsor] = sor(As{i},bs{i},zeros(size(bs{i})),1.2,100,1e-3);
 xsors{i} = xsor; jsors{i} = jsor;
end;

% print tables
fprintf('%10s %10s %10s %10s\n',' ','Jacobi','GS','SOR');
for i=1:length(As),
 fprintf('%10s %10d %10d %10d\n',names{i},jjacs{i},jgss{i},jsors{i});
end;
fprintf('\n\n');
format long
% print solutions
fprintf('solutions with different methods are:\n');
for i=1:length(As),
 fprintf([names{i} '\n']);
 fprintf('  Jacobi: '); fprintf('%13.9g ',xjacs{i}); fprintf('\n');
 fprintf('  GS:     '); fprintf('%13.9g ',xgss{i}); fprintf('\n');
 fprintf('  SOR:    '); fprintf('%13.9g ',xsors{i}); fprintf('\n');
end;


%  >> prob4
%  part a Jacobi
%  x=[   0.00000000    0.00000000    0.00000000 ]
%  x=[   1.25000000   -1.33333333    0.20000000 ]
%  x=[   1.63333333   -0.98333333    0.23333333 ]
%  part a Gauss-Seidel
%  x=[   0.00000000    0.00000000    0.00000000 ]
%  x=[   1.25000000   -0.91666667    0.06666667 ]
%  x=[   1.49583333   -0.85694444   -0.05555556 ]
%  part a SOR
%  x=[   0.00000000    0.00000000    0.00000000 ]
%  x=[   1.50000000   -1.00000000    0.00000000 ]
%  x=[   1.50000000   -0.80000000   -0.09600000 ]
%  part b Jacobi
%  x=[   0.00000000    0.00000000    0.00000000 ]
%  x=[  -2.00000000    2.00000000    0.00000000 ]
```



```
%  x=[  -1.00000000    1.00000000   -1.00000000 ]
%  part b Gauss-Seidel
%  x=[   0.00000000    0.00000000    0.00000000 ]
%  x=[  -2.00000000    1.00000000   -0.50000000 ]
%  x=[  -1.62500000    1.31250000   -0.65625000 ]
%  part b SOR
%  x=[   0.00000000    0.00000000    0.00000000 ]
%  x=[  -2.40000000    0.96000000   -0.57600000 ]
%  x=[  -1.51680000    1.47072000   -0.76723200 ]
%                  Jacobi         GS         SOR
%      part a           9          6           8
%      part b          20          7           6
%
%
%  solutions with different methods are:
%  part a
%     Jacobi:    1.44821591  -0.835659259 -0.0450897222
%     GS:        1.44781635  -0.835817304 -0.0447996185
%     SOR:       1.44750381  -0.835929762 -0.0445516532
%  part b
%     Jacobi:   -1.45408793    1.45408793   -0.72760766
%     GS:       -1.45505345    1.45412213  -0.727061063
%     SOR:      -1.45458285    1.45449886  -0.727330271
%
```

```
%%%%% Problem 5 %%%%%%

% build matrix (see Trefethen and Bau, Section 40)
D = ones(1000,5); D(:,3) = sqrt(1:1000)+0.5;
A = spdiags(D,[-100,-1,0,1,100],1000,1000);
DA = spdiags(0.5*sqrt(1:1000)',0,1000,1000);

% solve system with CG routine
tol = 1e-10; maxiter=100; x0 = zeros(1000,1);  b = ones(1000,1);
[x,res_prec] = mycg(A,x0,b,DA,tol,maxiter);
[x,res_uprec] = mycg(A,x0,b,speye(1000),tol,maxiter);

% plot both residuals in semilogy plot
semilogy(1:length(res_prec),res_prec,1:length(res_uprec),res_uprec)
legend('preconditioned','unpreconditioned');
xlabel('iterations'); ylabel('||b-Ax||');
print('-depsc2','residuals.eps');
% Note:
%   * the unpreconditioned case takes much longer to converge.
%
%   * We know that the residual in CG is monotonically decreasing, however the
%     preconditioned residual does increase a little bit! There is no
%     contradiction. The _preconditioned_ residual does decrease monotonically.
```



```
% function [x,j] = jacobi(A,b,x0,maxiter,tol)
%
%   Jacobi method for solving linear systems iteratively
%
% inputs
%       A       system matrix
%       b       system right hand side
%       x0      initial guess
%       maxiter max number of iterations
%       tol     tolerance between  two consecutive iterates (in l_inf norm)
%
% outputs
%       x       current solution
%       j       iteration number
function [x,j] = jacobi(A,b,x0,maxiter,tol)
n = size(A,1); x = zeros(size(x0));
for j=1:maxiter

 % print only first few iterates
 if (j<4) fprintf('x=['); fprintf('%13.8f ',x0); fprintf(']\n'); end;

 % compute Jacobi update
 for i=1:n,
  x(i) = (-A(i,1:i-1)*x0(1:i-1) -A(i,i+1:n)*x0(i+1:n)+ b(i))/A(i,i);
 end;

 % check for convergence
 if norm(x-x0,'inf')/norm(x0,'inf') <tol return; end;

 x0=x;
end;
fprintf('Maximum number of iterations reached and tolerance not met\n');
```

```
% function [x,j] = gs(A,b,x0,maxiter,tol)
%
%     Gauss-Seidel method for solving linear systems iteratively
%
% inputs
%       A       system matrix
%       b       system right hand side
%       x0      initial guess
%       maxiter max number of iterations
%       tol     tolerance between  two consecutive iterates (in l_inf norm)
%
% outputs
%       x       current solution
%       j       iteration number
function [x,j] = gs(A,b,x0,maxiter,tol)
n = size(A,1); x = zeros(size(x0));
for j=1:maxiter
 % print only first few iterates
 if (j<4) fprintf('x=['); fprintf('%13.8f ',x0); fprintf(']\n'); end;

 % Gauss-Seidel update
 for i=1:n,
  x(i) =  (-A(i,1:i-1)*x(1:i-1) -A(i,i+1:n)*x0(i+1:n)+ b(i))/A(i,i);
 end;

 % check for convergence
 if norm(x-x0,'inf')/norm(x0,'inf')<tol return; end;

 x0=x;
end;
fprintf('Maximum number of iterations reached and tolerance not met\n');
```

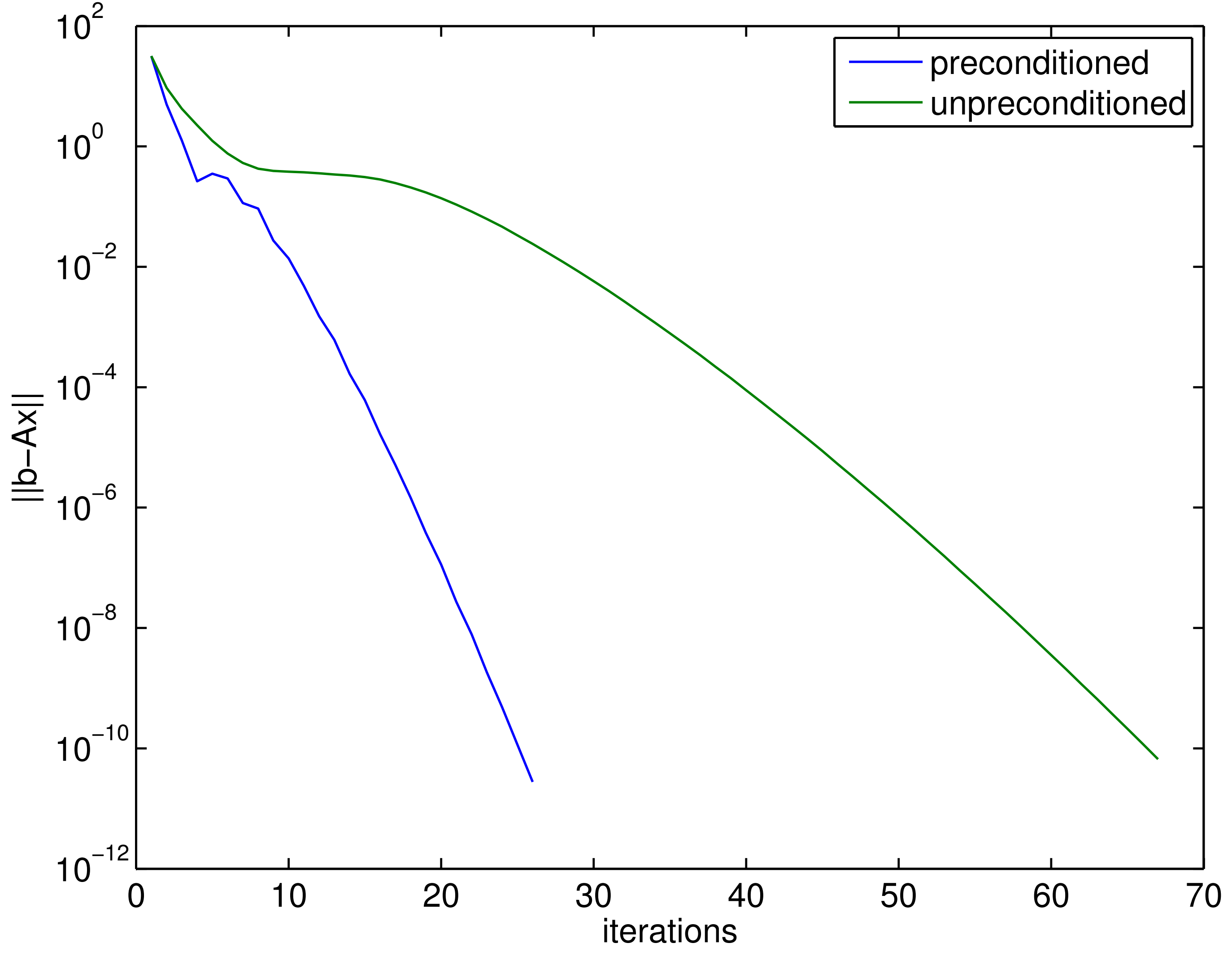


```
% function [x,j] = sor(A,b,x0,om,maxiter,tol)
%
%   SOR method for solving linear systems iteratively
% inputs
%       A       system matrix
%       b       system right hand side
%       x0      initial guess
%       om      omega parameter for SOR
%       maxiter max number of iterations
%       tol     tolerance between  two consecutive iterates (in l_inf norm)
%
% outputs
%       x       current solution
%       j       iteration number
function [x,j] = sor(A,b,x0,om,maxiter,tol)
n = size(A,1); x = zeros(size(x0));
for j=1:maxiter
 % print only first few iterates
 if (j<4) fprintf('x=['); fprintf('%13.8f ',x0); fprintf(']\n'); end;

 % SOR update
 for i=1:n,
  x(i) = (1-om)*x0(i) + om*(-A(i,1:i-1)*x(1:i-1) -A(i,i+1:n)*x0(i+1:n)+ b(i))/A(
i,i);
 end;

 % check for convergence
 if norm(x-x0,'inf')/norm(x0,'inf')<tol return; end;
 x0=x;
end;
fprintf('Maximum number of iterations reached and tolerance not met\n');
```

```
function [x,res] = mycg(A,x,b,M,tol,maxiter)
r = b - A*x; res = norm(r);
y = M\r;
p = y;
for n=1:maxiter,
 al = (r'*y)/(p'*A*p);
 x  = x + al*p;
 rold=r; yold=y;
 r  = r - al*A*p;
 res = [res,norm(r)]; % store residuals for later use
 if (norm(r)<tol)
  fprintf('convergence to tol=%g in %d iter\n',tol,n);
  break;
 end;
 y  = M\r;
 bt = (y'*r)/(yold'*rold);
 p  = y + bt*p;
end;
```



preconditioned
unpreconditioned
||b−Ax||
iterations
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
0
10
20
30
40
50
60
70