6.4 Preconditioned Conjugate Gradient

Error for the CG is a function of the condition number of A, $k_2(A) = \frac{\max|l|}{\min|l|}$.

The fastest convergence of the CG method occurs when $k_2(A) \approx 1$. Preconditioning can

Be viewed as finding an equivalent $\hat{A}\hat{X} = \hat{d}$ such that

$$K_2(\hat{A}) - 1 < K_2(A) - 1$$

There are three equivalent descriptions of the CG scheme:

1. $J(x^{m+1}) = \min_C J(x^m + c_o r_o + ... + c_m r_m)$

   where r_i are residual directions,

2. $J(x^{m+1}) = \min_C J(x^m + c_o p_o + ... + c_m p_m)$

   where p_i are conjugate directions, and

3. $J(x^{m+1}) = \min_C J(x^o + c_o r_o + c_1 A r_o ...... + c_m A^m r_o)$

   where $A^i r_0$ are Krylov directions.

Proposition 8. If A is SPD, then 1,2 and 3 are equivalent.

Proof.

$1 \leftrightarrow 2$, see formal proof on Stoer and Bulrich.

$2 \leftrightarrow 3$, see Kelley.

Connection among 1,2,3:

Let p_i be the conjugate directions as defined in the conjugate gradient algorithm.

$$p_o \equiv r_o$$

$$x^1 = x^o + a_o p_o$$

$$\begin{aligned}
r_1 &= r_o - a_o A p_o \\
p_1 &= r_1 + b_o p_o \\
x^2 &= x^1 + a_1 p_1 \\
&= x^1 + a_1 (r_1 + b_o p_o) \\
&= x^1 + a_1 (r_1 + b_o r_o) \ \ , residual \ \ directions \\
&= x^o + a_o r_o + a_1 (r_o - a_o A p_o + b_o r_o) \\
&= x^o + c_o r_o + c_1 A r_o \ \ , Krylov \ \ directions.
\end{aligned}$$

Proposition 9. If A is SPD, then

$$\left\| x^{m+1} - x \right\|_A \le 2 \left(\frac{\sqrt{k_2} - 1}{\sqrt{k_2} + 1} \right)^{m+1} \left\| x^o - x \right\|_A$$

where Ax = d, $k_2 = k_2(A)$ and $\left\| x \right\|_A^2 = x^T A x.$

“Outline of proof"

Use the Algebraic Lemma

$$J(x^{m+1}) - J(x) = 1/2 (x^{m+1} - x)^T A (x^{m+1} - x)$$

$$= 1/2 \left\| x^{m+1} - x \right\|_A^2 .$$

$$\begin{aligned}
x - x^{m+1} &= x - (x^o + c_o r_o + \text{.........} c_m A^m r_o) \\
&= x - x^o - (c_o r_o + \text{.........} c_m A^m r_o) \\
&= x - x^o - (c_o I + c_1 A + \text{.........} c_m A^m) r_o
\end{aligned}$$

$$\begin{aligned}
r_o &= d - A x^o \\
&= A x - A x^o \\
&= A (x - x^o)
\end{aligned}$$

$$x - x^{m+1} = x - x^o - (c_o I + c_1 A + \text{.........} c_m A^m) r_o A (x^o - x)$$

$$= (I - (c_o I + c_1 A + ...c_m A^m)A)(x^o - x)$$

So, by the Algebraic Lemma

$$2(J(x^{m+1}) - J(x)) = \|x^{m+1} - x\|_A^2 \le \|q_m(A)(x - x^o)\|_A^2 \quad \text{where}$$

$$q_m(z) = 1 - (c_o z + ...+ c_m z^{m+1}).$$

To obtain an error estimate choose a "good" polynomial $q_m(z)$.

Form of Preconditioner.

$$A = M - N$$
M is SPD
$$M^{-1} = S^T S$$
$$Ax = d$$
$$M^{-1}Ax = M^{-1}d$$
$$S^T SAx = S^T Sd$$
$$S^T SAS^T (S^{-T}x) = S^T Sd$$
$$(SAS^T)(S^{-T}x) = Sd$$
Let $\hat{A} = SAS^T$
$\hat{x} = S^{-T}x$
$\hat{d} = Sd,$

Apply CG to $\hat{A}\hat{x} = \hat{d}$ and use the definition $M^{-1} = S^T S$ to get the PCG .

Examples.

1. M = diagonal part of A

   or

   = block diagonal part of A
2. M = incomplete Cholesky factorization
3. M = incomplete domain decomposition

4. M for symmetric SOR splitting as follows:

$$Let\ w = 1.$$
$$A = D - L - L^T$$
$$(D-L)x^{m+1/2} = d + L^T x^m \qquad \text{Forward SOR}$$
$$(D-L^T)x^{m+1} = d + Lx^{m+1/2} \qquad \text{Backward SOR}$$
$$= d + L(D-L)^{-1}(d + L^T x^m)$$
$$x^{m+1} = (D-L^T)^{-1}[d + L(D-L)^{-1}(d + L^T x^m)]$$
$$= (D-L^T)^{-1}d + (D-L^T)^{-1}L(D-L)^{-1}d + (D-L^T)^{-1}L(D-L)^{-1}L^T x^m$$
$$= M^{-1}d + M^{-1}Nx^m$$
$$M^{-1} = (D-L^T)^{-1} + (D-L^T)^{-1}L(D-L)^{-1}$$
$$= (D-L^T)^{-1}[(D-L) + L](D-L)^{-1}$$
$$= (D-L^T)^{-1}D(D-L)^{-1}$$

$$Solve \quad M\hat{r} = r$$
$$(D-L)D^{-1}(D-L^T)\hat{r} = r.$$

$$For\ w \neq 1$$
$$M_w^{-1} = \left(\frac{1}{w}(D-wL^T)\right)^{-1}\left(\frac{2-w}{w}\right)D\left(\frac{1}{w}(D-wL)\right)^{-1}.$$

Matlab Preconditioned Conjugate Gradient with SSOR (cgssor.m)

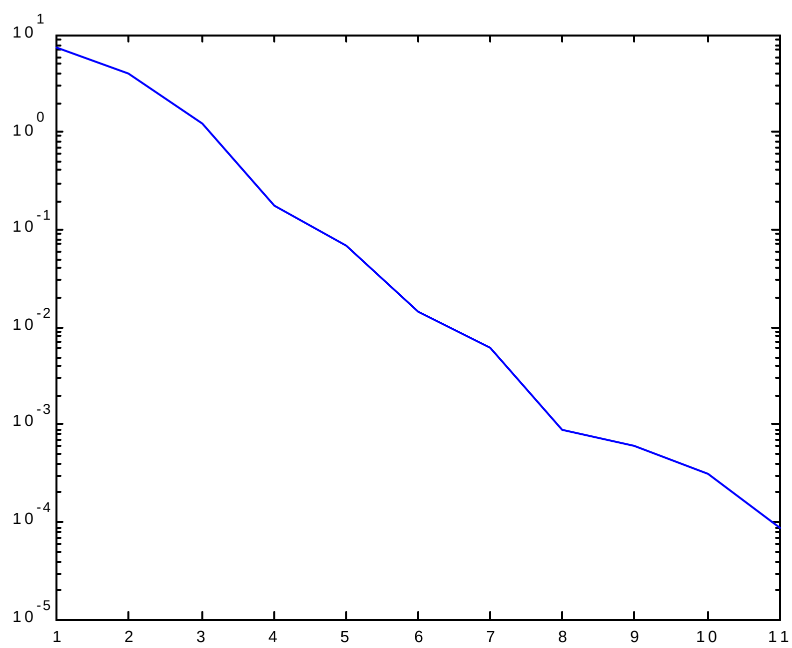
```
clear;
%
%  Solves  -uxx -uyy = 200+200sin(pi x)sin(pi y) with zero BCs
%  Uses PCG with SSOR preconditioner
%  Uses 2D arrays for the column vectors
%  Does not explicity store the matrix
%
w = 1.5;
n = 20;
h = 1./n;
u(1:n+1,1:n+1)= 0.0;
r(1:n+1,1:n+1)= 0.0;
```

```
rhat(1:n+1,1:n+1) = 0.0;
%  Define right side of PDE
for j= 2:n
   for i = 2:n
      r(i,j)= h*h*(200+200*sin(pi*(i-1)*h)*sin(pi*(j-1)*h));
   end
end




p(1:n+1,1:n+1)= 0.0;
q(1:n+1,1:n+1)= 0.0;
err = 1.0;
m = 0;
rho = 0.0;
%  Begin PCG iterations
while ((err>.0001)*(m<200))
   m = m+1;
   oldrho = rho;
%  Execute SSOR preconditioner
   for j= 2:n
      for i = 2:n
         rhat(i,j)=w*(r(i,j)+rhat(i-1,j)+rhat(i,j-1))/4.;
      end
   end
   rhat(2:n,2:n) = ((2.-w)/w)*(4.)*rhat(2:n,2:n);
   for j= n:-1:2
      for i = n:-1:2
         rhat(i,j)=w*(rhat(i,j)+rhat(i+1,j)+rhat(i,j+1))/4.;
      end
   end
%  Find conjugate direction
   rho = sum(sum(r(2:n,2:n).*rhat(2:n,2:n)));
   if (m==1)
      p = rhat;
   else
      p = rhat + (rho/oldrho)*p;
   end
%
%  Use the following line for steepest descent method
%   p=r;
%
%  Executes the matrix product q = Ap without storage of A
   for j= 2:n
      for i = 2:n
         q(i,j)=4.*p(i,j)-p(i-1,j)-p(i,j-1)-p(i+1,j)-p(i,j+1);
      end
   end
%  Executes the steepest descent segment
   alpha = rho/sum(sum(p.*q));
   u = u + alpha*p;
   r = r - alpha*q;
%  Test for convergence via the infinity norm of the residual
```

```
   err = max(max(abs(r(2:n,2:n))));
   reserr(m) = err;
end
m
semilogy(reserr)
```



Log(norm(r)) versus m for PCG with SSOR