Numerical Linear Algebra: Solutions for Sheet 7

1. The MINRES algorithm essentially consists of two steps. The first step is one iteration of the Lanczos algorithm to compute a new vector for the next Krylov subspace. The costs here are determined by one matrix vector product as the most expensive part and a couple of inner products. The computational cost for the matrix vector product are $O(n)$ since the matrix is sparse, i.e. A has only $nnz \ll n^2$ entries the multiplication by A is in $O(n)$. Therefore, the one step of the Lanczos process is $O(n)$. The second step of the algorithm is updating the QR factorization and solving the least squares system. Since the matrix $T_{k+1,k}$ is tridiagonal the triangular factor R of the QR decomposition has only three nonzero diagonals. Therefore, solving with R can be done in $O(n)$ steps and the Q factor can be updated by just one Givens rotation which is in $O(1)$. This gives the overall cost.

2. Obviously, $(v_0, v_1) = 0$ is true and γ_1 can be chosen to make $(v_1, v_1) = 1$. We suppose that

$$(v_i, v_k) = 0 \text{ and } (v_k, v_k) = 1.$$

Now for v_{k+1} and (v_i, v_{k+1}), we get

$$(v_i, v_{k+1}) = \frac{1}{\gamma_{k+1}} (v_i, Av_k - \delta_k v_k - \gamma_k v_{k-1}).$$

We only have to consider the cases $i = k-1, k, k+1$. Starting with $i = k-1$, we have

$$\frac{1}{\gamma_{k+1}} ((\gamma_k v_k + w, v_k) - (v_{k-1}, \delta_k v_k) - \gamma_k (v_{k-1}, v_{k-1})) = \frac{1}{\gamma_{k+1}} (\gamma_k - \gamma_k) = 0.$$

For $i = k$

$$\frac{1}{\gamma_{k+1}} ((Av_k, v_k) - \delta_k (v_k, v_k) - \gamma_k (v_k, v_{k-1})) = \frac{1}{\gamma_{k+1}} (\delta_k - \delta_k) = 0$$

and the case $(v_{k+1}, v_{k+1}) = 1$ can easily be satisfied since we are free to chose γ_{k+1}. It is left to show that we have a basis for the Krylov subspace $\mathcal{K}_{k+1}(A, r_0)$. The initial step is trivial and we assume we have a basis $(v_1, \ldots, v_k)$ for $\mathcal{K}_k(A, r_0)$. We now get

$$A^{l+1} r_0 = A(A^l r_0) = A \sum_{i=1}^{k} \alpha_i v_i = \sum_{i=1}^{k} \alpha_i A v_i = \sum_{i=1}^{k} \alpha_i (\gamma_{i+1} v_{i+1} + \delta_i v_i + \gamma_i v_{i-1}) \in \operatorname{span}\{v_1, \ldots, v_{k+1}\}$$

and everything is shown.

3. We get

$$r_{k+1} = b - Ax_{k+1} = b - A(x_k + \alpha_k pk) = r_k - \alpha_k A p_k$$

Now for

$$r_{k+1}^T p_k = (r_k - \alpha_k A p_k)^T p_k) = r_k^T p_k - \alpha_k p_k A p_k = 0$$

using the definition of α_k. Furthermore,

$$p_{k+1}^T A p_k = (r_{k+1} + \beta_k p_k)^T A p_k = r_{k+1}^T A p_k + \beta_k p_k^T A p_k = 0$$

by definition of β_k and using that A is symmetric. For $r_{k+1}^T r_k$ we get

$$\begin{array}{rcc} r_{k+1}^T r_k & = & (r_k - \alpha_k A p_k)^T r_k \\ & = & (r_k - \alpha_k A p_k)^T (p_k - \beta_{k-1} p_{k-1}) \\ & = 0 & \end{array}$$

by using previous results and the symmetry of A. The initial step of the induction is

$$\begin{array}{rcl} r_1^T p_0 & = & 0 \\ r_1^T r_0 & = & 0 \\ p_1^T A p_0 & = & 0 \end{array}.$$

We now assume this is true for k and get

$$\begin{aligned} r_k^T p_j &= 0 \\ r_k^T r_j &= 0 \\ p_k^T A p_j &= 0 \end{aligned}$$

where $j = 1, 2, \ldots, k-1$. We consider this for $k+1$. We get $r_{k+1}^T p_k = 0$ from the above and for $j = 1, \ldots, k-1$

$$\begin{aligned} r_{k+1}^T p_j &= (r_k - \alpha_k A p_k)^T p_j \\ &= r_k^T p_j - \alpha_k p_k^T A p_j \\ &= -\alpha_k p_k^T A p_j \\ &= 0 \end{aligned}$$

which means $r_{k+1}^T p_j = 0$ for $j = 0, \ldots, k$. Now,

$$\begin{aligned} r_{k+1}^T r_j &= (r_k - \alpha_k A p_k)^T r_j \\ &= r_k^T r_j - \alpha_k p_k^T A r_j \\ &= -\alpha_k p_k^T A r_j \\ &= -\alpha_k p_k^T A (p_j - \beta_{j-1} p_{j-1}) \\ &= -\alpha_k p_k^T A p_j + \alpha_k \beta_{j-1} p_k^T A p_{j-1}) \\ &= 0 \end{aligned}$$

for $j = 0, \ldots, k-1$ and with the previous results we get $r_{k+1}^T r_j = 0$ for $j = 0, \ldots, k$. We now consider $p_{k+1}^T A p_j$ $j = 0, \ldots, k-1$ the case $j = k$ was already shown

$$\begin{aligned} p_{k+1}^T A p_j &= (r_{k+1} + \beta_{k+1} p_k)^T A p_j \\ &= r_{k+1}^T A p_j + \beta_{k+1} p_k^T A p_j \\ &= r_{k+1}^T A p_j \\ &= r_{k+1}^T \frac{1}{\alpha_j}(r_j - r_{j+1}) \\ &= \frac{1}{\alpha_j} r_{k+1}^T r_j - \frac{1}{\alpha_j} r_{k+1}^T r_{j+1} \\ &= 0 \end{aligned}$$

for $j = 0, \ldots, k-1$ by using the previous result and that $\alpha_j \neq 0$. Otherwise the algorithm would have stopped in a previous step. This now gives $p_{k+1}^T A p_j = 0$ for all $j = 0, \ldots, k$ which completes the proof.

4. We can express

$$\|x - (x_k + \alpha p_k)\|_A^2$$

as

$$f(\alpha) = (x - x_k)^T A (x - x_k) - 2\alpha p_k^T A (x - x_k) + \alpha^2 p_k^T A p_k$$

and we take the derivative with respect to α and get

$$-2 p_k^T A (x - x_k) + 2\alpha p_k^T A p_k = 0.$$

The result is

$$\alpha = \frac{p_k^T r_k}{p_k^T A p_k}.$$

Next,

$$(r_k, r_k) = (p_k - \beta_k p_{k-1}, r_k) = (p_k, r_k) - \beta_{k-1}(p_{k-1}, r_k) = (p_k, r_k)$$

by using Question 3.

$$\beta_{k+1} = -\frac{(p_k, A r_{k+1})}{(p_k, A p_k)} = \frac{1}{\alpha_k} \frac{(r_{k+1} - r_k, r_{k+1})}{(p_k, A p_k)} = \frac{(p_k, A p_k)}{(p_k, r_k)} \frac{(r_{k+1}, r_{k+1})}{(p_k, A p_k)} = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)}$$

and all the desired equations are shown.

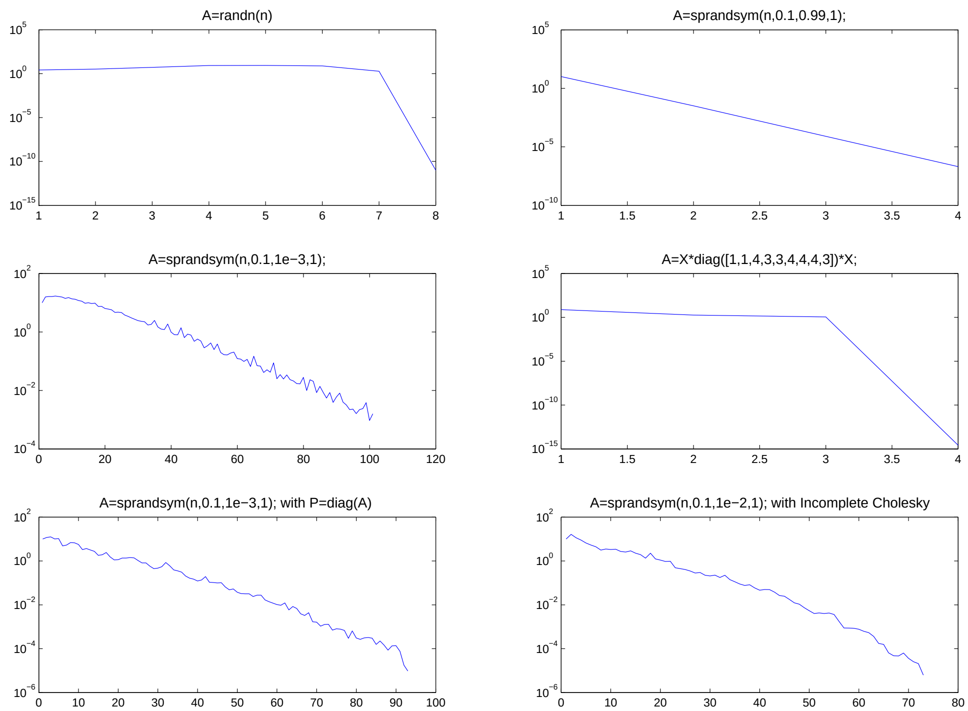
5. The algorithm detailed in question 4 essentially gives us the efficient implementation we require. We simply note that only one matrix vector product is required at each iteration - $A\mathbf{p}_k$. If this is stored the first time it is calculated in each iteration, then it can be re-used later on in the iteration, so we use only one matrix vector product in each iteration.

```
x = x0;
r = b - A*x;
r2 = r'*r;
p = r;
do while r2 < tol
  Ap = A*p;
  a = r2 / p'*Ap;
  r = r - a*Ap;
  x = x + a*p;
  r2b = r'*r;
  b = r2b / r2;
  r2 = r2b;
  p = r + b*p;
loop
```

6. Results can be obtained using the MATLAB code below and are shown in Figure below.

```
subplot(3,2,1)
n=7
A=randn(n);
b=ones(n,1);
A=A'*A;
[x,flag,relres,iter,resvec] = pcg(A,b,1e-6,n);
semilogy(resvec,'b')

subplot(3,2,2)
A=sprandsym(100,0.1,0.99,1);
b=ones(100,1);
[x,flag,relres,iter,resvec] = pcg(A,b,1e-6,n);
semilogy(resvec,'b')

subplot(3,2,3)
n=100
A=sprandsym(n,0.1,1e-3,1);
b=ones(n,1);
[x,flag,relres,iter,resvec] = pcg(A,b,1e-6,n);
semilogy(resvec,'b')

subplot(3,2,4)
X=randn(9,9);
X=orth(X);
A=X*diag([1,1,4,3,3,4,4,4,3])*X';
b=sum(A,2);
[x,flag,relres,iter,resvec] = pcg(A,b,1e-6,9);
semilogy(resvec,'b')


subplot(3,2,5)
A=sprandsym(n,0.1,1e-3,1);
b=ones(n,1);
[x,flag,relres,iter,resvec] = pcg(A,b,1e-6,n,diag(diag(A)));
semilogy(resvec,'b')

subplot(3,2,6)
A=sprandsym(n,0.1,1e-2,1);
```

```
R=cholinc(A,'0');
[x,flag,relres,iter,resvec] = pcg(A,b,1e-6,n,R,R');
semilogy(resvec,'b')
```



7. We have a splitting of a symmetric matrix A such that the splitting matrix M is also symmetric. Then if $S = I - M^{-1}A$ we have

$$\begin{aligned}
\langle S\mathbf{x}, \mathbf{y}\rangle_A &= (S\mathbf{x})^T A\mathbf{y} \\
&= \mathbf{x}^T S^T A\mathbf{y} \\
&= \mathbf{x}^T (I - M^{-1}A)^T A\mathbf{y} \\
&= \mathbf{x}^T (A - AM^{-1}A)\mathbf{y} \qquad \text{(As } M, A \text{ symmetric)} \\
&= \mathbf{x}^T A(I - M^{-1}A)\mathbf{y} \\
&= \mathbf{x}^T A(S\mathbf{y}) \\
\text{so we have} \quad \langle S\mathbf{x}, \mathbf{y}\rangle_A &= \langle \mathbf{x}, S\mathbf{y}\rangle_A
\end{aligned}$$

and so S is symmetric with respect to the A inner product, and so may be used as a preconditioner with CG.