Math 584 Homework 7
Due Soon
By Marvyn Bailly

Problem 1 *Let*

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad A_\epsilon = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \epsilon & 0 & 0 & 0 \end{bmatrix}, \quad \epsilon > 0.$$

What are the eigenvalues of A and of A_ϵ? Would you say that the problem of computing the eigenvalues of a matrix A of this form is well-conditioned or ill-conditioned? Explain your answer.

Solution. Let

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad A_\epsilon = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \epsilon & 0 & 0 & 0 \end{bmatrix}, \quad \epsilon > 0.$$

To find the eigenvalues of A observe that the characteristic polynomial is given by

$$\lambda^4 = 0,$$

So all the eigenvalues of A are 0. The characteristic polynomial of A_ϵ is given by

$$\lambda^4 = \epsilon$$

so the eigenvalues of A are the 4th roots of ϵ,

$$\epsilon^{1/4}, -\epsilon^{1/4}, -i\epsilon^{1/4}, i\epsilon^{1/4}.$$

Computing the eigenvalues of A_ϵ is is ill conditioned because a change in magnitude in ϵ will largely effect the eigenvalues of A. For example, if ϵ is perturbed by a very small amount, let's say 10^{-12}, then the eigenvalue corresponding to $\epsilon^{1/4}$ will change by a factor of 10^{-4}.

□

Problem 2 *p. 218, Exercise 28.1.*

Solution.

Let A be an orthogonal matrix. Then when applying the unshifted QR algorithm to A, since A is already orthogonal R is just the identity matrix. Thus we have that $A = AR^{-1} = AI = A$. Thus we do not experience the behavior described in Theorem 28.4 of $A^{(k)}$ as $k \to \infty$. This makes sense since A is orthogonal, the eigenvalues of A must have magnitude 1 and therefore we do not meet the requirement of Theorem 28.4 which says that each eigenvalue must be strictly greater than the next. $\square$

Problem 3

1. *Describe an $O(k^2)$ algorithm based on QR factorization by Givens rotations for solving the $k+1$ by k upper Hessenberg least squares problem*

$$H_{k+1,k}y \approx \|r_0\|\xi_1,$$

that arises in the GMRES algorithm.

2. *Show how the operation count can be reduced to $O(k)$ if the problems at steps 1 through $k-1$ have already been solved. [A general description is given at the bottom of p. 3 and top of p. 4 of the notes on Krylov Space Methods, but fill in the details.]*

Solution.

1. We begin by decomposing $H_{k+1,k}$ using QR factorization to get

$$Q_{k+1,k_1}H_{k_1,k} = R_{k+1,k} = \begin{pmatrix} R_{k,k} \\ 0 \end{pmatrix}.$$

Now notice that updating $H_{k+1,k}$ to $H_{k+2,k+1}$ can be done by

$$H_{k+2,k+1} = \begin{pmatrix} H_{k+1,k} & h_{k+1} \\ 0 & k_{k+3,k+2} \end{pmatrix}.$$

Thus when multiplying $Q_{k+1,k+1}$ by $H_{k+2,k+1}$ we get

$$Q_{k+2,k+2}H_{k+2,k+1} = \begin{pmatrix} Q_{k+1,k+1} & 0 \\ 0 & 1 \end{pmatrix} H_{k+2,k+1} = \begin{pmatrix} R_{k,k} & r_{k+1} \\ 0 & p \\ 0 & q \end{pmatrix},$$

which is almost upper triangle. We can utilize Given rotations to get $q = 0$ with the matrix

$$G_{k+2,k+2} = \begin{pmatrix} I_{k,k} & 0 & 0 \\ 0 & c_k & s_k \\ 0 & -s_k & c_k \end{pmatrix}$$

where $c_k = p/(p^2+q^2)$ and $s_k = q/(p^2+q^2)$. Then let's multiply $Q_{k+2,k+2}$ by the Givens matrix to get

$$Q_{k+2,k+2} = G_{k+2,k_2}\begin{pmatrix} Q_{k+1,k+1} & 0 \\ 0 & 1 \end{pmatrix}.$$

Thus we have that

$$Q_{k+2,k+2}H_{k+2,k+1} = \begin{pmatrix} R_{k,k} & r_{k+1} \\ 0 & \sqrt{p^2+q^2} \\ 0 & 0 \end{pmatrix},$$

which is indeed upper triangle. This means that we preform $O(k^2)$ operation because we have to preform k matrix multiplications. The upper triangle matrix can now be used to solve the least squares problem and minimize y.

2. If the problem has been solved at steps 1 through $k-1$, then at the kth step, we only need to apply one more given rotation to zero the last entry of the last column of $H_{k+1,k}$ since we have saved all the previous given rotations. This final given rotation will only effect the last column and thus there will only be $0(k)$ cost at this step.

□

Problem 4 *Show that if A is Hermitian and positive definite, the 'A-norm' of a vector, defined as*

$$\|v\|_A := (v^* A v)^{1/2}$$

is, indeed, a norm.

Solution.
Let A be a Hermitian positive definite matrix. Then by definition of A we have that

$$v^* A v > 0 \quad \forall v \neq 0 \quad \text{and} \quad A = LL^*,$$

where LL^* is the Cholesky decomposition. We define the A-norm as

$$\|v\|_A := (v^* A v)^{1/2},$$

and wish to verify that it is a norm. Firstly observe that

$$\|A\|_A = (v^* A v)^{1/2} > 0,$$

since $v^* A v > 0$ where equality is achieved only if $v = 0$. Next let α be some constant, then

$$\begin{aligned}
\|\alpha v\|_A &= ((\alpha v)^* A (\alpha v))^{1/2} \\
&= (v^* \alpha^* A \alpha v)^{1/2} \\
&= (\alpha^* \alpha)^{1/2} (v^* A v)^{1/2} \\
&= |\alpha| \|v\|_A.
\end{aligned}$$

Lastly note that we can use the Cholesky decomposition to rewrite the A-norm as

$$\|v\|_A = (v^* A v)^{1/2} = (v^* L L^* v)^{1/2} = ((L^* v)^* (L^* v))^{1/2} = \|L^* v\|_2$$

Then we can show the triangle inequality by observing

$$\begin{aligned}
\|u + v\|_A &= ((u + v)^* A (u + v))^{1/2} \\
&= ((u^* + v^*) L L^* (u + v))^{1/2} \\
&= ((L^*(u + v))^* (L^*(u + v)))^{1/2} \\
&= \|L^*(u + v)\| \\
&= \|L^* u + L^* v\|_2 \\
&\leq \|L^* u\|_2 + \|L^* v\|_2 \\
&= \|u\|_A + \|v\|_A.
\end{aligned}$$

Thus we have verified that the A-norm is a norm. $\square$

Problem 5

1. *The CG algorithm is applied to a Hermitian positive definite linear system $Ax = b$. The A-norm of the error in the initial guess is 1, and the A-norm of the error at step 10 is 2×2^{-10}. Based on this information alone, what bound can you give on $\kappa(A)$? Explain how you get your answer.*

2. *Suppose a positive definite matrix has a small, well-separated eigenvalue: $\lambda_1 << \lambda_2 \leq \ldots \leq \lambda_m$ (that is, $\lambda_1/\lambda_2 << 1$). Derive an error bound for CG using the maximum value of a polynomial that is the product of a linear factor $(\lambda_1 - z)/\lambda_1$ and a $(k-1)$st degree Chebyshev polynomial on the interval $[\lambda_2, \lambda_m]$. Comparing your result with inequality (10) in the notes on Krylov Space Methods, would you say that it is more advantageous to have a small, well-separated eigenvalue or a large, well-separated eigenvalue as in (10)?*

Solution.

1. Consider the CG algorithm applied to a Hermitian positive definite linear system $Ax = b$. We have that the error in the initial guess is $\|e_0\|_A = 1$ and that the error at step 10 is $\|e_{10}^{\text{CG}}\|_A = 2 \times 2^{-10}$. By Theorem 1 in the Krylov notes we have that

$$\frac{\|e_k^{\text{CG}}\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k,$$

where $\kappa = \lambda_{\max}/\lambda_{\min}$. Note that

$$\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}} = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} = \frac{\lambda_{\max}}{\lambda_{\min}} = \kappa.$$

Plugging in our values and solving for κ we get

$$2 \times 2^{-10} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{10} \implies \frac{1}{2} \leq \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \implies -\sqrt{\kappa} \leq -3 \implies \kappa \geq 9.$$

Thus we have bounded $\kappa(A) \geq 9$.

2. Suppose a positive definite matrix has a small, well-separated eigenvalue: $\lambda_1 << \lambda_2 \leq \ldots \leq \lambda_m$ (that is, $\lambda_1/\lambda_2 << 1$). To derive an error bound consider the product of $(\lambda_1 - z)/\lambda_1$ and a $(k-1)$st degree Chebyshev polynomial on the interval $[\lambda_2, \lambda_m]$. That is,

$$p_k(z) = \left(\frac{\lambda_1 - z}{\lambda_1}\right)\left(T_{k-1}\left(\frac{2z - \lambda_m - \lambda_2}{\lambda_m - \lambda_2}\right) \Big/ T_{k-1}\left(\frac{-\lambda_m - \lambda_2}{\lambda_m - \lambda_2}\right)\right).$$

Then the first term is bounded by $\left|\frac{\lambda_1 - \lambda_m}{\lambda_1}\right|$ and the second term is bounded by $2\left(\frac{\sqrt{\kappa_m}-1}{\sqrt{\kappa_m}+1}\right)^{k-1}$ where $\kappa_m = \lambda_m/\lambda_2$ by the Theorem 1 in the Krylov notes. Then the error bound is given by

$$\frac{\|e_k^{\text{CG}}\|_A}{\|e_0\|_A} \leq 2\left|\frac{\lambda_1 - \lambda_m}{\lambda_1}\right|\left(\frac{\sqrt{\kappa_m}-1}{\sqrt{\kappa_m}+1}\right)^{k-1}.$$

Comparing this bound with the inequality (10) in the Krylov notes, we can see that it is more advantageous to have a large, well-separated eigenvalue as in (10).

□

Problem 6 *You can set up a sparse matrix defining a finite difference approximation for Laplace's equation on an L-shaped domain in Matlab by typing* `A = delsq(numgrid('L',500))`. *Matlab stores just the nonzero entries of the matrix along with their row numbers and column numbers. You can see where A has nonzeros by typing* `spy(A)`*, and you can see the size of A by typing* `[m,m] = size(A)`*. Set a random vector b of length m and use unpreconditioned CG to solve $Ax = b$ for x. You may use* `pcg` *in Matlab. Type:*

```
tic, [x,flag,relres,iter,resvec] = pcg(A,b,tol,m), toc
```

where `tol` *is a desired tolerance, say,* $1.0e-8$*. This will time the routine and print out the time spent. The array* `resvec` *returned by* `pcg` *contains the 2-norm of the residual at each step of the CG algorithm. Plot the entries of* `resvec` *vs. iteration number using a* `semilogy` *plot so that small values of the residual norm can be distinguished. Now try solving $Ax = b$ using CG with an incomplete Cholesky decomposition as a preconditioner. You can get the incomplete Cholesky decomposition of A by typing* `L = ichol(A)`*. You can then send this to* `pcg` *by typing*

```
tic, [x,flag,relres,iter,resvecic] = pcg(A,b,tol,m,L,L'); toc
```

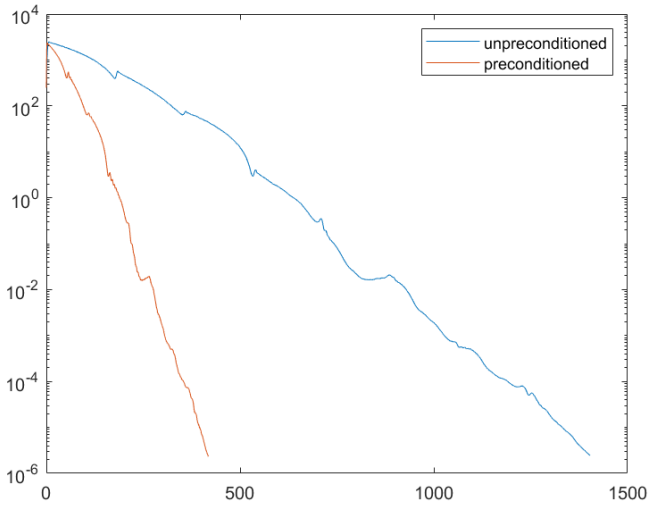
Plot the residual norms at each step of the preconditioned algorithm on the same plot as those of the unpreconditioned algorithm. Turn in your plot of residual norms, with the different convergence curves labeled. Try a few different runs and report on which algorithm is usually faster.

Solution.
I wrote the following MatLab code for this problem

```
A = delsq(numgrid('L',500));
[m,m] = size(A);
b = rand(m,1);
tol = 1.0 * 10^(-8);
tic,
[x,flag,relres,iter,resvec] = pcg(A,b,tol,m);
toc

semilogy(0:iter,resvec)
hold on

L = ichol(A);
tic,
[x,flag,relres,iter,resvecic] = pcg(A,b,tol,m,L,L');
toc
```

```
semilogy(0:iter,resvecic)
hold off

legend("unpreconditioned","preconditioned");
```

which gives the following plot



and the following output

```
>> hw7q6
    Elapsed time is 8.819515 seconds.
    Elapsed time is 6.576605 seconds.
>> hw7q6
    Elapsed time is 8.947682 seconds.
    Elapsed time is 6.400742 seconds.
```

Running the algorithm a few times and observing the time results of each algorithm shows that the preconditioned algorithm runs faster and takes less iterations than the unpreconditioned. □