AMSC 660 Homework 5
Due Next Week
By Marvyn Bailly

Problem 1

(a) Consider the set $\mathcal{L}$ of all $n \times n$ lower-triangular matrices with positive diagonal entries.

- Prove that the product of any two matrices in $\mathcal{L}$ is also in $\mathcal{L}$.
- Prove that the inverse of any matrix in $\mathcal{L}$ is also is $\mathcal{L}$.

(b) Prove that the Cholesky decomposition for any $n \times n$ symmetric positive definite matrix is unique. *Hit. Proceed from converse. Assume that there are two Cholesky decomposition $A = LL^T$ and $A = MM^T$. Show that then $M^{-1}LL^TM^{-T} = I$. Conclude that $M^{-1}L$ must be orthogonal. Then use item (a) of this problem to complete the argument.*

Solution

Proof. (a) Consider the set $\mathcal{L}$ of all $n \times n$ lower-triangular matrices with positive diagonal entries. Let $A, B \in \mathcal{L}$, then if $C = AB$, the elements of C are given by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Thus for c_{ij} with $i = j$

$$\begin{aligned} c_{ii} &= a_{i1}b_{1i} + a_{i2}b_{2i} + \cdots + a_{ii-1}b_{i-1i} + a_{ii}b_{ii} + a_{ii+1}b_{i+1i} + \cdots + a_{in-1}b_{n-1j} + a_{in}b_{nj} \\ &= a_{i1}0 + a_{i2}0 + \cdots + a_{ii-1}0 + a_{ii}b_{ii} + 0b_{i+1i} + \cdots + 0b_{n-1j} + 0b_{nj} \\ &= a_{ii}b_{ii}. \end{aligned}$$

So $c_{ii} = a_{ii}b_{ii} > 0$. If $i < j$, c_{ij} is given by

$$\begin{aligned} c_{ij} &= a_{i1}b_{1j} + \cdots + a_{ii}b_{ij} + \cdots + a_{ij}b_{jj} + \cdots + a_{in}b_{nj} \\ &= a_{i1}0 + \cdots + a_{ii}0 + \cdots + 0b_{jj} + \cdots + 0b_{nj} \\ &= 0. \end{aligned}$$

So $c_{ij} = 0$ for $i < j$. Therefore C is lower triangular with positive diagonal elements and thus $C \in \mathcal{L}$. Next, let $A \in \mathcal{L}$. Let $B = A^{-1}$ where b_{ij} are the elements of B. Then $AB = C = I$. Notice that, $c_{ii} = a_{ii}b_{ii} = 1$ and since $a_{ii} > 0$, then $b_{ii} > 0$. To see that the inverse is also lower triangular, denote the columns of B as b_i for $1 \leq i \leq n$. Then

$$AB = A[b_1|\cdots|b_n] = [Ab_1|\cdots|Ab_n] = I,$$

and thus

$$Ab_i = e_i,$$

where e_i has 1 in the ith position and zeros elsewhere for $1 \leq i \leq n$. Then since e_i has zeros above the ith row and A is lower triangular, b_i has only zeros above the ith row. Thus B is lower triangular. Since A^{-1} is lower triangular and has positive elements along the main diagonal we have shown that $A \in \mathcal{L} \implies A^{-1} \in \mathcal{L}$.

(b) Consider A to be an $n \times n$ symmetric positive definite matrix which has two Cholesky decomposition $A = LL^T$ and $A = MM^T$. Observe that

$$\begin{aligned}
& LL^T = MM^T \\
\iff & M^{-1}LL^T = M^T \\
\iff & M^{-1}LL^TM^{-T} = I \\
\iff & (M^{-1}L)(M^{-1}L)^{-T} = I. \qquad (1)
\end{aligned}$$

From (a), we know that the product and inverse of lower triangular matrices with positive diagonal entries are lower triangular with positive diagonal entries, and thus $M^{-1}L$ is lower triangular with positive diagonal entries. Rearranging the terms of Eq. (1) gives

$$(M^{-1}L) = (M^{-1}L)^T,$$

and since both sides of the equation are lower triangular, we conclude that $M^{-1}L$ must be diagonal. Let's say that the diagonal elements of $M^{-1}L$ are given by $d_1, \dot{;} d_n$ and so the diagonal elements of $(M^{-1}L)^{-T}$ are given by $d_1^{-1}, \dots, d_n^{-1}$. By Eq. (1), $d_i \cdot d_i^{-1} = 1$ which shows that $d_i = 1, \forall i = 1, \dots n$. Therefore

$$M^{-1}L = I \implies L = M,$$

which shows that the Cholesky decomposition of a SPD matrix is unique.

$\square$

Problem 2

The Cholesky algorithm is the cheapest way to check if a symmetric matrix is positive definite.

(a) Program the Cholesky algorithm. If any L_{jj} turns out to be either complex or zero, make it terminate with a message: "The matrix is not positive definite."

(b) Generate a symmetric 100×100 as follows: generate $\tilde{A}$ with entries being random numbers uniformly disturbed in $(0, 1)$ and defined $A := \tilde{A} + \tilde{A}^T$. Use the Cholesky algorithm to check if A is a symmetric positive definite. Compute the eigenvalue of A using a standard command (`eig`), find minimal eigenvalue, and check if the conclusion of your Cholesky-based test for positive definiteness is correct. If A is positive definite, compute its Cholesky factorization using a stand command and print the norm of the difference of the Cholesky factors computed by your routine and by standard one.

(c) Repeat item (b) with A defined by $A = \tilde{A}^T \tilde{A}$. The point of this task is to check that your Cholesky routine works correctly.

Solution

Proof. (a) I coded the check as

```
    if(L(j,j) == 0 || ~isreal(L(j,j)))
        fprintf("The matrix is not positive definite")
        result = 0;
        return
    end
```

(b) I made the following code snippets to perform part (b). Here is the Cholesky method with the check from part (a).

```
    function result = cholesky(A)
        n = size(A);
        L = zeros(n,n)a;

        % Check if A is SPD
        for j = 1 : n
            L(j,j) = (A(j,j) - sum(L(j,1:j-1).^2))^(1/2);
            if(L(j,j) == 0 || ~isreal(L(j,j)))
                fprintf("The matrix is not positive definite")
                result = 0;
                return
            end
            for i = j + 1 : n
```

```
            L(i,j) = (A(i,j) - sum(L(i,1:j-1).*L(j,1:j-1)))/L(j,j);
        end
    end
    result = L;
end
```

In the following code snippets, we preform the instruction from part (b):

```
function question2()
    At = rand(10);  %generate random guy

    A = At + At';

    L = cholesky(A);

    if(length(L) > 1)
        fprintf("The matrix is postive definte\n")
        mineval = min(eig(A));  %get smallest eigevalue
        if(mineval ~= 0 && isreal(mineval))
            fprintf("The minimal eigenvalue is %d which is positive and real\n",mineval)
        else
            fprintf("The algorithm failed")
        end
        err = norm(L - chol(A,'lower'));
        fprintf("The norm error of the algorithm and matlab cholesky is %d\n",err)
    end
end
```

which outputted `The matrix is not positive definite` over multiple attempts.

(c) To preform part (c), I modified how we defined A as

```
function question2()
    At = rand(10);  %generate random guy

    A = At'*At;

    L = cholesky(A);

    if(length(L) > 1)
        fprintf("The matrix is postive definte\n")
        mineval = min(eig(A));  %get smallest eigevalue
```

```
            if(mineval ~= 0 && isreal(mineval))
                fprintf("The minimal eigenvalue is %d which is positive and real\n",mineval)
            else
                fprintf("The algorithm failed")
            end
            err = norm(L - chol(A,'lower'));
            fprintf("The norm error of the algorithm and matlab cholesky is %d\n",err)
        end
    end
```

Now running the question 2 gives the following output:

```
The matrix is postive definte
The minimal eigenvalue is 4.880531e-03 which is positive and real
The norm error of the algorithm and matlab cholesky is 9.828340e-15
```

Code can be found at `https://github.com/MarvynBailly/AMSC660/tree/main/homework5`.

□

Problem 3

An $n \times n$ matrix is called *tridiagonal* if it is of the form

$$A = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & & \\ 0 & a_3 & b_3 & c_3 & \\ & & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & a_n & b_n \end{pmatrix}$$

There is a fast algorithm for solving linear systems of $Ay = f$ with invertible and strickly diagonal dominant (i.e. $|b_i| > |a_i| + |c_i| \forall i$) tridiagonal matrices A. Sometimes it is referred to as the *Thomas algorithm:*

```
function TridiagSolver(a,b,c,f)
    n = length(f);
    v = zeros(n,1);
    y = v;
    w = b(1);
    y(1) = f(1)/w;
    for i=2:n
        v(i-1) = c(i-1)/w;
        w = b(i) - a(i)*v(i-1);
        y(i) = ( f(i) - a(i)*y(i-1) )/w;
    end
    for j=n-1:-1:1
        y(j) = y(j) - v(j)*y(j+1);
    end
end
```

Calculate the number of flops for the Thomas algorithm.

Solution

Proof. We wish to count the number of flops in the Thomas algorithm. Observe that

```
function TridiagSolver(a,b,c,f)
    n = length(f);
    v = zeros(n,1);
    y = v;
    w = b(1);
    y(1) = f(1)/w;                      %one flop: "/"
    for i=2:n
        v(i-1) = c(i-1)/w;              % one flop: "/"
        w = b(i) - a(i)*v(i-1);         % two flops: "-" and "*"
```

```
        y(i) = ( f(i) - a(i)*y(i-1) )/w;% three flops: "-", "*", and "/"
    end
    for j=n-1:-1:1
        y(j) = y(j) - v(j)*y(j+1);      % two flops: "-" and "*"
    end
end
```

Collecting the flops yields

$$W(n) = 1 + \sum_{i=2}^{n} 6 + \sum_{i=1}^{n-1} 2 = 1 + 6(n-1) + 2(n-1) = 8n - 7$$

$\square$

Problem 4

Calculate the number of flops for the modified Gram-Schmidt algorithm for computing the QR factorization of an $n \times n$ matrix A. Here is a vectorized Matlab code implementing the modified Gram-Schmidt.

```
A = rand(n);
Q = zeros(n); R = zeros(n);
for i = 1 : n
    Q(:,i) = A(:,i);
    for j = 1 : i-1
        R(j,i) = Q(:,j)'*Q(:,i);
        Q(:,i) = Q(:,i) - R(j,i)*Q(:,j);
    end
    R(i,i) = norm(Q(:,i));
    Q(:,i) = Q(:,i)/R(i,i);
end
```

*Hint: The command Q(:,j)'*Q(:,i) means* $\sum_{k=1}^{n} Q_{kj} Qki$ *and the command Q(:,i) = Q(:,i) - R(j,i)*Q(k,j) means the for-loop*

```
for k = 1 : n
    Q(k,i) = Q(k,i) - R(j,i)*Q(k,j);
end
```

Solution

Proof. We wish to calculate the number of flops in the following modified Gram-Schmidt algorithm

```
A = rand(n);
Q = zeros(n);
R = zeros(n);
for i = 1 : n
    Q(:,i) = A(:,i);
    for j = 1 : i-1
        R(j,i) = Q(:,j)'*Q(:,i);    % 2n-1 flops: n "*" and n-1 "+"
        Q(:,i) = Q(:,i) - R(j,i)*Q(:,j);    % 2n flops: n "*" and n "-"
    end
    R(i,i) = norm(Q(:,i));      % 2n flops: n "*", n-1 "+", and "sqrt"
    Q(:,i) = Q(:,i)/R(i,i);                 % n flop: "/"
end
```

Collecting the flops gives

$$\begin{aligned}
W(n) &= \sum_{i=1}^{n} \left(\sum_{j=1}^{i-1} (2n - 1 + 2n) + 2n + n \right) \\
&= \sum_{i=1}^{n} (2n-1)(i-1) + 2n(i-1) + 3n \\
&\approx \int_0^n (2n-1)(x-1) + 2n(x-1) + 3n \mathrm{d}x \\
&= 2n^3 + \mathcal{O}(n^2).
\end{aligned}$$

Thus the amount of flops the modified Gram-Schmidt algorithm takes is approximately $2n^3$.

$\square$

Problem 5

(a) Prove the cyclic property of the trace:

$$\operatorname{trace}(ABC) = \operatorname{trace}(BCA) = \operatorname{trace}(CAB)$$

for all A, B, C such that their product is defined and is a square matrix.

(b) Prove that

$$\|A\|_F^2 = \sum_{i=1}^{d} \sigma_i^2.$$

Hint: use the full SVD of A and the cyclic property of trace.

(c) Prove that

$$\|A + B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 + 2\langle A, B\rangle_F,$$

where $\langle A, B\rangle_F$ is the Frobenius inner product. The Frobenius inner product is defined as

$$\langle A, B\rangle_F := \sum_{i,j} a_{ij} b_{ij} = \operatorname{trace}(A^T B) = \operatorname{trace}(B^T A)$$

Solution

Proof. (a) Consider the matrices A be $n \times a$, B be $a \times b$, and C be $b \times n$. Let $D = AB$ which is a $n \times b$ matrix. Then observe that

$$\begin{aligned}
\operatorname{trace}(DC) &= \sum_i (DC)_{ii} \\
&= \sum_i \sum_j d_{ij} c_{ji} \\
&= \sum_i \sum_j c_{ij} d_{ji} \\
&= \sum_i (CD)_{ii} \\
&= \operatorname{trace}(CD),
\end{aligned}$$

thus we have that $\operatorname{trace}(ABC) = \operatorname{trace}(DC) = \operatorname{trace}(CD) = \operatorname{trace}(CAB)$. Now if we let $E = BC$ and apply what we found above, we get $\operatorname{trace}(BCA) = \operatorname{trace}(EA) = \operatorname{trace}(AE) = \operatorname{trace}(ABC)$. Thus we have that

$$\operatorname{trace}(CAB) = \operatorname{trace}(ABC) = \operatorname{trace}(BCA).$$

(b) Let $A = U\Sigma V^T$ be the SVD decomposition of A. Recall that

$$\|A\|_F^2 = \operatorname{trace}(AA^T).$$

Thus we have that

$$\begin{aligned}
\|A\|_F^2 &= \operatorname{trace}\big(AA^T\big) \\
&= \operatorname{trace}\big(U\Sigma V^T V\Sigma U^T\big) \\
&= \operatorname{trace}\big(U\Sigma^2 U^T\big) \\
&= \operatorname{trace}\big(\Sigma^2 U^T U\big) \\
&= \operatorname{trace}\big(\Sigma^2\big) \\
&= \sum_{i=1}^{d} \sigma_i^2
\end{aligned}$$

(c) Let $A = U\Sigma V^T$ and $B = \tilde{U}\tilde{\Sigma}\tilde{V}^T$ be the SVD decomposition of A and B respectively. Observe that

$$\begin{aligned}
\|A+B\|_F^2 &= \operatorname{trace}\big((A+B)(A+B)^T\big) \\
&= \operatorname{trace}\Big(\Big(U\Sigma V^T + \tilde{U}\tilde{\Sigma}\tilde{V}^T\Big)\Big(U\Sigma V^T + \tilde{U}\tilde{\Sigma}\tilde{V}^T\Big)^T\Big) \\
&= \operatorname{trace}\Big(\Big(U\Sigma V^T + \tilde{U}\tilde{\Sigma}\tilde{V}^T\Big)\Big(\tilde{V}\tilde{\Sigma}\tilde{U}^T + V\Sigma U^T\Big)\Big) \\
&= \operatorname{trace}\Big(U\Sigma V^T\tilde{V}\tilde{\Sigma}\tilde{U}^T + \tilde{U}\tilde{\Sigma}\tilde{V}^T\tilde{V}\tilde{\Sigma}\tilde{U}^T + U\Sigma V^T V\Sigma U^T + \tilde{U}\tilde{\Sigma}\tilde{V}^T V\Sigma U^T\Big) \\
&= \operatorname{trace}\Big(AB^T + \tilde{U}\tilde{\Sigma}^2\tilde{U}^T + U\Sigma^2 U^T + BA^T\Big).
\end{aligned}$$

As the trace is a linear mapping and $\operatorname{trace}(AB^T) = \operatorname{trace}(B^T A) = \operatorname{trace}(BA^T)$ we get

$$\begin{aligned}
\|A+B\|_F^2 &= 2\operatorname{trace}\big(AB^T\big) + \operatorname{trace}\Big(\tilde{U}\tilde{\Sigma}^2\tilde{U}^T\Big) + \operatorname{trace}\big(U\Sigma^2 U^T\big) \\
&= 2\operatorname{trace}\big(AB^T\big) + \operatorname{trace}\Big(\tilde{\Sigma}^2\Big) + \operatorname{trace}\big(\Sigma^2\big) \\
&= \|A\|_F^2 + \|B\|_F^2 + 2\langle A, B\rangle_F.
\end{aligned}$$

Therefore $\|A+B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 + 2\langle A, B\rangle_F$

$\square$