Scientific Computing HW 5

Ryan Chen

October 1, 2024

1. Throughout we use the Matlab syntax for matrices and submatrices.

   (a) **for** $i = 1, \ldots, n$ **do**
   $M[i, i:] \leftarrow M[i, i:]/M[i, i]$
   for $j \neq i$ **do**
   $M[j, i:] \leftarrow M[j, i:] - M[j, i] * M[i, i:]$
   end for
   end for

   The flop count in the reassignment of $M[j, i:]$ is $2(2n - i + 1)$ due to it being a row vector with $2n - i + 1$ entries. The flop count of the whole algorithm is

$$
\begin{aligned}
W(n) &= \sum_{i=1}^{n} 2(2n - i + 1)(n - 1) \\
&\sim (2n - 1) \int_0^n (2n - x + 1) dx \\
&= 2(n - 1) \left(2nx - \frac{1}{2}x^2 + x \right) \Big|_0^n \\
&= 2(n - 1) \left(2n^2 - \frac{1}{2}n^2 + n \right) \\
&= 3n^3 + O(n^2)
\end{aligned}
$$

   (b) We can view the problem in four parts: The LU decomposition, finding L^{-1}, finding U^{-1}, and finding $U^{-1}L^{-1}$.

   - The LU decomposition flop count is $W_1(n) = \frac{2}{3}n^3 + O(n^2)$.
   - The idea of finding L^{-1}: From $LL^{-1} = I$, we can find the kth column of L^{-1} by solving $Lx = e_k$. Since L is lower triangular, we do this by forward substitution.

     $L^{-1} \leftarrow 0_{n \times n}$
     for $k = 1, \ldots, n$ **do**
     $x \leftarrow 0_{n \times 1}$
     $x[k] \leftarrow 1$
     for $j = k + 1, \ldots, n$ **do**
     $S \leftarrow -L[j, k]$
     for $i = k + 1, \ldots, j - 1$ **do**
     $S \leftarrow S - L[j, i] * x[i]$
     end for
     $x[j] \leftarrow S$
     end for
     $L^{-1}[:, k] \leftarrow x$
     end for

The flop count in the reassignment of S is 2, so the flop count in the i loop is $2(j-1-k)$. The flop count of the algorithm is

$$
\begin{aligned}
W_2(n) &= \sum_{k=1}^{n}\sum_{j=k+1}^{n} 2(j-1-k) \\
&\sim 2\int_0^n \int_x^n (y-1-x)dydx \\
&= 2\int_0^n \left(\frac{1}{2}y^2 - y - xy\right)\Big|_x^n dx \\
&= 2\int_0^n \left(\frac{1}{2}n^2 - n - nx + \frac{1}{2}x^2 + x\right) dx \\
&= 2\left(\frac{1}{2}n^2x - nx - \frac{1}{2}nx^2 + \frac{1}{6}x^3 + \frac{1}{2}x^2\right)\Big|_0^n \\
&= 2\left(\frac{1}{2}n^3 - n^2 - \frac{1}{2}n^3 + \frac{1}{6}n^3 + \frac{1}{2}n^2\right) \\
&= \frac{1}{3}n^3 + O(n^2)
\end{aligned}
$$

- Similarly, we can find the kth column of U^{-1} by solving $Ux = e_k$. Since U is upper triangular, we do this by backward substitution.

  $U^{-1} \leftarrow 0_{n\times n}$
  for $k = 1, \ldots, n$ **do**
      $x \leftarrow 0_{n\times 1}$
      for $j = k-1, \ldots, 1$ (decrement j after each iteration) **do**
          $S \leftarrow 0$
          for $i = j+1, \ldots, k$ **do**
              $S \leftarrow S - U[j,i] * x[i]$
          end for
          $x[j] \leftarrow S/U[j,j]$
      end for
      $U^{-1}[:,k] \leftarrow x$
  end for

  The flop count in the reassignment of S is 2, so the flop count in the i loop is $2(k-j)$. Using WolframAlpha, the flop count of the algorithm is

$$
\begin{aligned}
W_3(n) &= \sum_{k=1}^{n}\sum_{j=1}^{k-1}(2(k-j)+1) \\
&\sim \int_0^n \int_0^x (2x - 2y + 1)dydx \\
&= \int_0^n (2xy - y^2 + y)\Big|_0^x dx \\
&= \int_0^n (x^2 + x)dx \\
&= \frac{1}{3}n^3 + O(n^2)
\end{aligned}
$$

- Note that $U^{-1}L^{-1}$ is a product of an upper triangular matrix and a lower triangular matrix.

The positions of the zeros allows us to drop some terms when computing entries,

$$(A^{-1})_{ij} = \sum_{k=1}^{n} (U^{-1})_{ik}(L^{-1})_{kj} = \sum_{k=\max(i,j)}^{n} (U^{-1})_{ik}(L^{-1})_{kj}$$

$A^{-1} \leftarrow 0_{n\times n}$
for $i = 1, \dots, n$ **do**
  for $j = 1, \dots, n$ **do**
    $S \leftarrow 0$
    for $k = \max(i,j), \dots, n$ **do**
      $S \leftarrow S + U^{-1}[i,k] * L^{-1}[k,j]$
    end for
    $A^{-1}[i,j] \leftarrow S$
  end for
end for

The flop count in the reassignment of S is 2, so the flop count in the k loop is $2(n-\max(i,j)+1)$. Using WolframAlpha, the flop count of the algorithm is

$$\begin{aligned}
W_4(n) &= \sum_{i=1}^{n}\sum_{j=1}^{n}(2(n-\max(i,j))+1) \\
&\sim \int_0^n \int_0^n (2n - 2\max(x,y)+1)dydx \\
&= 2n^3 - 2\int_0^n \underbrace{\int_0^n \max(x,y)dy}_{=:I(x)} dx + n^2
\end{aligned}$$

To rewrite $I(x)$, consider

$$\max(x,y) = \begin{cases} x, & y \le x \\ y, & y > x \end{cases}$$

Then compute

$$\begin{aligned}
I(x) &= \int_0^x xdy = \int_x^n ydy \\
&= x^2 + \frac{1}{2}(n^2 - x^2) \\
&= \frac{1}{2}(n^2 + x^2)
\end{aligned}$$

Thus

$$\begin{aligned}
W_4(n) &\sim 2n^3 - \int_0^n (n^2 + x^2)dx + n^2 \\
&= 2n^3 - n^3 - \frac{1}{3}n^3 + n^2 \\
&= \frac{2}{3}n^3 + O(n^2)
\end{aligned}$$

In total, the flop count of finding A^{-1} via its LU decomposition is

$$W(n) = W_1(n) + W_2(n) + W_3(n) + W_4(n) = 2n^3 + O(n^2)$$

2. (a) i. Fix $A, B \in \mathcal{L}$ and let $C := AB$. For $i < j$,

$$\begin{aligned} c_{ij} &= \sum_{k=1}^{n} a_{ik} b_{kj} \\ &= \sum_{k=1}^{i} a_{ik} b_{kj} + \sum_{k=i+1}^{n} a_{ik} b_{kj} \\ &= 0 + 0 = 0 && b_{kj} = 0 \text{ for } k < j \text{ and } a_{ik} = 0 \text{ for } k > i \end{aligned}$$

Thus C is lower triangular. For all i,

$$\begin{aligned} c_{ii} &= \sum_{k=1}^{n} a_{ik} b_{ki} \\ &= \sum_{k=1}^{i-1} a_{ik} b_{ki} + a_{ii} b_{ii} + \sum_{k=i+1}^{n} a_{ik} b_{ki} \\ &= 0 + a_{ii} b_{ii} + 0 && b_{ki} = 0 \text{ for } k < i \text{ and } a_{ik} = 0 \text{ for } k > i \\ &> 0 && b_{ii}, a_{ii} > 0 \end{aligned}$$

Thus C has positive diagonal entries. We conclude $C \in \mathcal{L}$.

ii. Fix $A \in L$. Since A is lower triangular, $\det A$ is the product of its diagonal entries, which are positive, so $\det A \neq 0$. Thus A is nonsingular, so let $B := A^{-1}$. To show $B \in \mathcal{L}$, we fix j and aim to show $b_{ij} = 0$ for all $i < j$, hence B is lower triangular, and $b_{jj} > 0$, hence B has positive diagonal entries. From $AB = I$ and A being lower triangular, hence $a_{ik} = 0$ for $k > i$, we have

$$\sum_{k=1}^{i} a_{ik} b_{kj} = \delta_{ij}$$

Write out the corresponding equations for all $i < j$:

$$\begin{aligned} i = 1 &: a_{11} b_{1j} = 0 \\ i = 2 &: a_{21} b_{1j} + a_{22} b_{2j} = 0 \\ &\vdots \\ i = j &: a_{j1} b_{1j} + a_{j2} b_{2j} + \cdots + a_{jj} b_{jj} = 1 \end{aligned}$$

We forward solve, using the fact A has positive diagonal entries.

- The $i = 1$ equation gives $b_{1j} = 0$.
- Substituting $b_{1j} = 0$ into the $i = 2$ equation gives $a_{22} b_{2j} = 0$, hence $b_{2j} = 0$.
- Substituting $b_{1j} = b_{2j} = 0$ into the $i = 3$ equation gives $a_{33} b_{3j} = 0$, hence $b_{3j} = 0$.
- Proceed in a similar fashion to get $b_{ij} = 0$ for all $i < j$.
- Substituting $b_{ij} = 0$ for all $i < j$ into the $i = j$ equation gives $a_{jj} b_{jj} = 1$, hence $b_{jj} = \frac{1}{a_{jj}} > 0$.

(b) Assume that A has two Cholesky decompositions,

$$A = LL^T = MM^T, \quad L, M \in \mathcal{L}$$

Then

$$(M^{-1}L)(M^{-1}L)^T = M^{-1}LL^T M^{-T} = M^{-1}MM^T M^{-T} = I$$

hence $U := M^{-1}L$ is orthogonal. Since $\mathcal{L}$ is a group wrt matrix multiplication and $M, L \in \mathcal{L}$, we have $U \in \mathcal{L}$. The only orthogonal lower triangular matrix with positive diagonal entries is the identity, giving $M^{-1}L = I$. We conclude $L = M$.

3. Code: https://github.com/RokettoJanpu/scientific-computing-1-redux/blob/main/hw5.ipynb

```python
import numpy as np
import numpy.linalg as la
```

```python
def cholesky(A):
    n,m=A.shape
    if n!=m:
        print('The matrix is not square.')
        return
    L = np.zeros((n,n))
    for j in range(n):
        aux = A[j,j] - np.sum(L[j,:j]**2)
        if aux<=0:
            print('The matrix is not positive definite.')
            return
        L[j,j] = np.sqrt(aux)
        for i in range(j+1,n):
            L[i,j] = (A[i,j] - L[i,:j]@L[j,:j])/L[j,j]
    return L
```

```python
A=np.random.rand(100,100)
B=A+A.T
C=A.T@A

for X in [B,C]:
    print('A+A^T:' if (X==B).all() else 'A^TA:')
    evals = la.eigvals(X)
    print('Minimum eigenvalue:',np.min(evals))
    L=cholesky(X)
    if L is not None:
        print('Matrix is SPD.')
        L_numpy = la.cholesky(X)
        print('Error between algorithms:',la.norm(L-L_numpy))
```

```
A+A^T:
Minimum eigenvalue: -7.788187643262669
The matrix is not positive definite.
A^TA:
Minimum eigenvalue: 9.176001668332046e-05
Matrix is SPD.
Error between algorithms: 2.2256399230782706e-12
```