AMATH 584 Assignment 6

Tyler Chen

Exercise 1

Let A be an m by m nonsingular matrix and let b be a given nonzero m-vector. Suppose x satisfies $Ax = b$ and $\hat{x}$ satisfies $A\hat{x} = \hat{b}$, where $\hat{b}$ is slightly different from b. Show that

$$\frac{\|\hat{x} - x\|_2}{\|x\|_2} \leq \kappa(A) \frac{\|\hat{b} - b\|_2}{\|b\|_2},$$

where $\kappa(A) = \|A\|_2 \|A^{-1}\|_2$ is the 2-norm condition number of A. Show that there are nonzero vectors b and $\hat{b} - b$ for which equality holds. [Note: You might want to review material in Lecture 12 of the text.]

Solution

We have $Ax = b$ and $A\hat{x} = \hat{b}$ so that $A(\hat{x} - x) = A\hat{x} - Ax = \hat{b} - b$. Since A is nonsingular, we have $\hat{x} - x = A^{-1}(\hat{b} - b)$.

Then, $\|b\|_2 = \|Ax\|_2 \leq \|A\|_2 \|x\|_2$ and $\|\hat{x} - x\|_2 = \|A^{-1}(\hat{b} - b)\|_2 \leq \|A^{-1}\|_2 \|\hat{b} - b\|_2$. Rearranging the first equation we have $1 / \|x\|_2 \leq \|A\|_2 / \|b\|_2$. Therefore,

$$\frac{\|\hat{x} - x\|_2}{\|x\|_2} \leq \|A\|_2 \left\|A^{-1}\right\|_2 \frac{\|\hat{b} - b\|_2}{\|b\|_2} = \kappa(A) \frac{\|\hat{b} - b\|_2}{\|b\|_2} \qquad \square$$

Recall the first right singular vector v_1 of a matrix A satisfies $\|A\| = \|Av\| / \|v\|$ so that $\|A\| \|v\| = \|Av\|$.

Then take x as the first right singular vector of A and $b = Ax$ so that $\|b\|_2 = \|A\|_2 \|x\|_2$. Likewise, take $(\hat{b} - b)$ as the first right singular vector of A^{-1} so that $\|\hat{x} - x\|_2 = \left\|A^{-1}\right\|_2 \|\hat{b} - b\|_2$. Then,

$$\frac{\|\hat{x} - x\|_2}{\|x\|_2} = \|A\|_2 \left\|A^{-1}\right\|_2 \frac{\|\hat{b} - b\|_2}{\|b\|_2} = \kappa(A) \frac{\|\hat{b} - b\|_2}{\|b\|_2} \qquad \square$$

Exercise 2

The idea of this exercise is to carry out an experiment analogous to the one described in Lec. 16 of the text, but for the SVD instead of QR factorization.

(a) Write a program that constructs a 50×50 matrix `A = U*S*V'`, where `U` and `V` are random orthogonal matrices and `S` is a diagonal matrix whose diagonal entries are uniformly distributed numbers in $[0,1]$, sorted into nonincreasing order. You can use the following lines in MATLAB:

```
[U,X] = qr(randn(50));
[V,X] = qr(randn(50));
S = diag(sort(rand(50,1),'descend'));
A = U*S*V';
```

Compute the SVD of A: `[U2,S2,V2] = svd(A);` Recall that the SVD of a real square matrix is not quite uniquely determined. Make sure that the signs of the columns of `U2` and `V2` match those of `U` and `V` as follows:

```
for j=1:50,
  if U2(:,j)'*U(:,j) < 0,    % The signs are different for the jth column.
    U2(:,j) = -U2(:,j); V2(:,j) = -V2(:,j);      % Change them to match.
  end;
end;
```

Now compute `norm(U2-U)`, `norm(V2-V)`, `norm(S2-S)/norm(S)`, and `norm(A - U2*S2*V2')/norm(A)`. Run your program with five different random matrices and comment on whether the various differences seem to be connected with the condition number of A, `cond(A)`.

(b) For each of the matrices in part(a), replace the diagonal entries in `S` by their sixth powers (thus making the condition number of `A` much larger) and repeat the experiment. Do you see significant differences between these results and those of the experiment for QR factorization in the text?

Solution

We implement this in python as,

```
def exercise_2():
    out=[]
    n=50
    [U,X]=np.linalg.qr(np.random.randn(n,n))
    [V,X]=np.linalg.qr(np.random.randn(n,n))
    ss=np.sort(np.random.rand(n))
    SS = [np.diag(np.flip(ss,0)),
          np.diag(np.flip(ss**6,0))]

    for i in range(2):
        S= SS[i]
        A=U@S@V

        [U2,S2,V2]=np.linalg.svd(A,full_matrices=True)
```

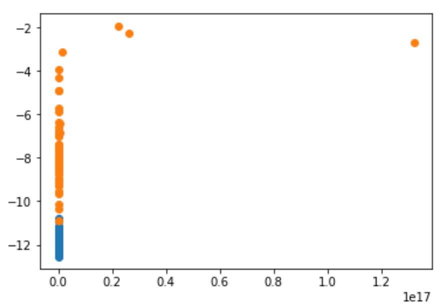
```
        S2=np.diag(S2)

        for j in range(n):
            if np.dot(U2[:,j],U[:,j])<0:
                U2[:,j] = -U2[:,j]
                V2[j,:] = -V2[j,:]
        res=[np.linalg.norm(U2-U),
             np.linalg.norm(V2-V),
             np.linalg.norm(S2-S)/np.linalg.norm(S),
             np.linalg.norm(A-U2@S2@V2),
             np.linalg.cond(A)]
        out.append(res)
    return out
```

(a) We test with 50 matrices as follows:

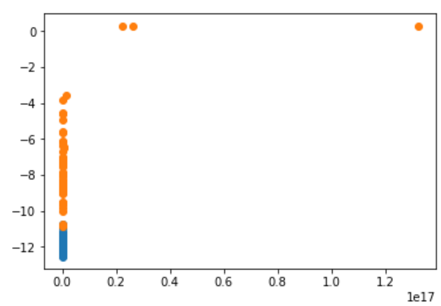
```
for i in range(4):
    plt.scatter(x[:,0,4], np.log10(x[:,0,i]))
    plt.scatter(x[:,1,4], np.log10(x[:,1,i]))
    plt.show()
```

It doesn't seem like there is a correlation between the condition number and the output.

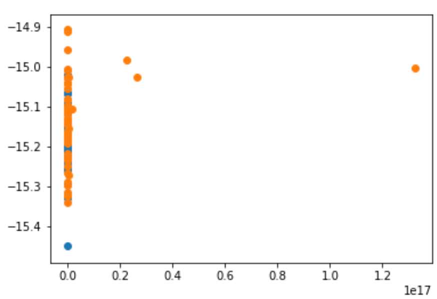
(b) The plots show no clear relationship between the condition number and the norms, other than that U_2, V_2, S_2 are slightly less accurate as the condition number increases. Strangely, it seems that $U_2 S_2 V_2$ is marginally more accurate as the condition number increases.



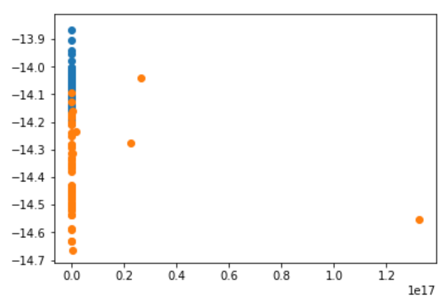
(a) $\|U_2 - U\|$ vs $\kappa(A)$



(b) $\|V_2 - V\|$ vs $\kappa(A)$



(c) $\|S_2 - S\| \,/\, \|S\|$ vs $\kappa(A)$



(d) $\left\|A - USV^T\right\|$ vs $\kappa(A)$

Figure 1: orange: higher condition number runs

However, in the QR factorization from the text was unstable and resulted in bad results as the condition number increase.

In particular, the Q and R factors were very off, although the product was not. Again we have the errors in U_2 and V_2 about 10^{-8} although the product $U_2S_2V_2$ has error of about 10^{-14}.

Note that there are some outputs of the norm function which give exactly 2.0. This seems to be a numerical error in the implementation of the 2-norm in numpy for matrices with norms very close to zero.

Exercise 3

Write the following matrix in the form LU, where L is a unit lower triangular matrix and U is an upper triangular matrix:

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix}$$

Without completely redoing the calculation, write the same matrix in the form LL^T, where L is lower triangular. Explain how you can derive the LL^T-factorization from the LU factorization, and vice-versa.

Solution

We apply row operations to zero the first entry in the second and third rows.

$$\begin{bmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ 1/4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix} = \begin{bmatrix} 4 & -1 & -1 \\ 0 & 15/4 & -5/4 \\ 0 & -5/4 & 15/4 \end{bmatrix}$$

We apply row operators to zero the second entry in the third row.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/3 & 1 \end{bmatrix} \begin{bmatrix} 4 & -1 & -1 \\ 0 & 15/4 & -5/4 \\ 0 & -5/4 & 15/4 \end{bmatrix} = \begin{bmatrix} 4 & -1 & -1 \\ 0 & 15/4 & -5/4 \\ 0 & 0 & 10/3 \end{bmatrix}$$

Thus,

$$L = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ 1/4 & 0 & 1 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ 1/4 & 1/3 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -1/4 & 1 & 0 \\ -1/4 & -1/3 & 1 \end{bmatrix}$$

And,

$$U = \begin{bmatrix} 4 & -1 & -1 \\ 0 & 15/4 & -5/4 \\ 0 & 0 & 10/3 \end{bmatrix}$$

To go from a LU decomposition to a LL^T transition first take the diagonal entries of U.

$$D = \begin{bmatrix} 4 & & \\ & 15/4 & \\ & & 10/3 \end{bmatrix}$$

We then rewrite $U = DU'$ with unit lower triangular matrix $U' = L^T$ and a diagonal matrix D^{-1}.

$$U' = D^{-1}U = \begin{bmatrix} 1/4 & & \\ & 4/15 & \\ & & 3/10 \end{bmatrix} \begin{bmatrix} 4 & -1 & -1 \\ 0 & 15/4 & -5/4 \\ 0 & 0 & 10/3 \end{bmatrix} = \begin{bmatrix} 1 & -1/4 & -1/4 \\ 0 & 1 & -1/3 \\ 0 & 0 & 1 \end{bmatrix} = L^T$$

Then, $A = LDL^T$. Now rewrite D as the product of two diagonal matrices,

$$\sqrt{D} = \begin{bmatrix} \sqrt{4} & & \\ & \sqrt{15/4} & \\ & & \sqrt{10/3} \end{bmatrix}$$

Then let,

$$L' = L\sqrt{D} = \begin{bmatrix} 1 & 0 & 0 \\ -1/4 & 1 & 0 \\ -1/4 & -1/3 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{4} & & \\ & \sqrt{15/4} & \\ & & \sqrt{10/3} \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ -1/2 & \sqrt{15/4} & 0 \\ -1/2 & -\sqrt{5/12} & \sqrt{10/3} \end{bmatrix}$$

Thus,

$$(L')(L')^T = (L\sqrt{D})(\sqrt{D}^T L^T) = LDL^T = LU = A$$

To return from a LL^T decomposition, first factor L into a lower unit triangular matrix by the method shown above, and then left multiply L^T by the appropriate factor. That is, let D be a diagonal matrix with the diagonal entires of L. Then LD^{-1} is unit lower triangular, so $(LD^{-1})(DL^T)$ is a LU decomposition.

Exercise 20.1

Let $A \in \mathbb{C}^{m\times m}$ be nonsingular. Show that A has an LU factorization if and only if for each k with $1 \le k \le m$, the upper-left $k \times k$ block $A_{1:k,1:k}$ is nonsingular. Prove this LU factorization is unique. Hint: The row operations of Gaussian eliminiation leave the determindants $\det(A_{1:k,1:k})$ unchanged.

Solution

Since A is nonsingular, $\det(A) \neq 0$.

Suppose each block $A_{1:k,1:k}$ is nonsingular for $1 \le k \le m$.

Since $A_{1:1,1:1}$ is nonsingular, the first pivot position is nonzero, so we can zero the entries below the pivot. In the second step, $A^{(1)}_{1:2,1:2}$ is nonsingular, so $A_{2,2}$ is nonzero so we can again pivot. Repeating this argument means we can perform the Gaussian elimination algorithm without partial pivoting and it will not fail. So A has a LU decomposition.

Now, suppose A has an LU decomposition. Then $0 \neq \det(A) = \det(L)\det(U)$. The determinant of a triangular matrix is the product of the diagonal entries so U has no zeros on the diagonals. Thus, $\det(L_{1:k,1:k}) \neq 0$ and $\det(U_{1:k,1:l}) \neq 0$ for all k with $1 \le k \le m$.

Observe $A_{1:k,1:k} = L_{1:k,1:k}U_{1:k,1:k}$. So, $\det(A_{1:k,1:k}) = \det(L_{1:k,1:k}U_{1:k,1:k}) = \det(L_{1:k,1:k})\det(U_{1:k,1:k}) \neq 0$. This proves $A_{1:k,1:k}$ is nonsingular for $1 \le k \le m$.

Gaussian elimination produces an L which is unit lower triangular. Thus,

$$(LU)_{1j} = \sum_{k=1}^{m} L_{1k}U_{kj} = L_{11}U_{1j} = U_{1j}$$

So U_{1j} is uniquely determined for $j = 1, ..., m$.

Now observe that,

$$(LU)_{i1} = \sum_{k=1}^{m} L_{ik}U_{k1} = L_{i1}U_{11}$$

Since U_{11} has been uniquely determined as above, then L_{i1} is also uniquely determined.

Repeating this argument on the submatrix $A_{k:m,k:m}$ for $k = 1, ..., m$ proves the LU decomposition obtained is unique. $\square$

Exercise 20.2

Suppose $A \in \mathbb{C}^{m\times m}$ satisfies the condition of Exercise 20.1 and is banded with bandwidth $2p+1$, i.e., $a_{ij} = 0$ for $|i-j| > p$. What can you say about the sparsity patterns of the factors L and U of A.

Count the number of operations required to compute L and U, assuming that you do not operate on entries that are 0 and are known to remain 0 after elimination.

Solution

The condition above from Exercise 20.1 means the Gaussian elimination without pivoting algorithm will not fail.

At each step, the L_k constructed will be banded with bandwidth $2p+1$, as entries outside this band of A are zero and so we do not need a row operation to eliminate them. The product of banded matrices is banded, as is the inverse, so L is also banded with bandwidth $2p+1$ (and obviously still lower triangular).

In the k-th step the k-th row will have zeros after the $k+p$-th position. We subtract some multiple of this row from everything below. So the entries to the right of the $k+p$-th position in rows above the k-th row will not be affected. This means $L_k..L_2L_1A$ remains banded with bandwidth $2p+1$ at every step. So U is also banded with bandwidth $2p+1$ (and obviously still upper triangular).

Then both L and U are banded with bandwidth $2p+1$. □

We compute the number of operations for an arbitrary banded matrix A. The general equation for Gaussian elimination without partial pivoting is:

```
U = A,  L = I
for k = 1 to m − 1:
    for j = k + 1 to m:
        l_jk = u_jk / u_kk
        u_j,k:m = u_j,k:m − l_jk u_k,k:m
```

Assuming A is banded with bandwidth $2p+1$ we know $l_{jk} = 0$ for $j > k+p$ as $u_{jk} = 0$ if $j > k+p$. We can then modify the bounds of the inner loop to be from $j = k+1$ to $j = k+p$. Similarly, we know that $u_{k,i} = 0$ for $i > k+p$ so we can modify the reassignment of u_j as $u_{j,k:k+p} = u_{j,k:k+p} - l_{jk}u_{k,k:k+p}$. However, in all of these cases, if $m \geq \max k+p$ we must use m instead.

We therefore have,

```
U = A,  L = I
for k = 1 to m − 1:
    n=min(k + p, m)
    for j = k + 1 to n:
        a=min(j + p, m)
        l_jk = u_jk / u_kk
        u_j,k:a = u_j,k:a − l_jk u_k,k:a
```

We assume we know n and a at each stage. Calculating l_{jk} requires a single division. Calculating $u_{j,k:a} = u_{j,k:a} - l_{jk}u_{k,k:a}$ requires $a-k+1$ multiplications and $a-k+1$ subtractions. If $p+j < m$ then $a = p+j$ so $n-k+1 = p+j-k+1$. Otherwise $n-k+1 = m-k+1$.

However, this is tedious to calculate so for ease we assume $n = k+p$ and $a = j+p$ for all

k, j. In the inner loop we do one division, $a - k + 1 = j + p - k + 1$ multiplications, and $a - k + 1 = j + p - k + 1$ subtractions. Therefore the operation count is,

$$\begin{aligned}
&\sum_{k=1}^{m-1} \sum_{j=k+1}^{\min(k+p,m)} 1 + 2(\min(j+p, m) - k + 1) \\
&\sim \sum_{k=1}^{m-1} \sum_{j=k+1}^{k+p} 1 + 2(j + p - k + 1) \\
&\sim \sum_{k=1}^{m-1} (k + p - (k+1) + 1) \frac{[1 + 2(p+2) + [1 + 2(2p+1)]}{2} \\
&= \sum_{k=1}^{m-1} p[1 + (3p + 3)] \\
&= (m-1)(3p^2 + 4p) \\
&\sim 3mp^2
\end{aligned}$$

Exercise 21.2

Suppose $A \in \mathbb{C}^{m \times m}$ is banded with bandwidth $2p+1$, and a factorization $PA = LU$ is computed by Gaussian elimination with partial pivoting. What can you say about the sparsity of L and U?

Solution

Use $A^{(k)}$ to denote the matrix we are working on in the k-th step. So $A^{(1)} = A$ and $A^{(m+1)} = U$.

Suppose we are in step k working in the k-th column. Then $A^{(k)}_{k+p,k}$ is the lowest entry in the k-th column which could be nonzero. Suppose this is our pivot row. Then the entry $A^{(k)}_{k+p,k+2p}$ is the furthest right entry in the $k+p$-th row which may be nonzero. This entry would be pivoted to the position $k, k+2p$.

So U will be banded with bandwidth $2(2p)+1$, and by definition upper triangular.

Each L_k will again be banded, however the L'_k permute the subdiagonal entires of L_k and so it may no longer be banded. In particular, $L'_1 = P_{m-1}...P_2L_1P_2^{-1}...P_{m-1}^{-1}$. Each pair of right and left multiplications by P_k^{-1} and P_k will permute the entries of L_1. So P_2 and P_2^{-1} can move an entry one space past the $p+1$-th row, and P_3 and P_3^{-1} can move this entry down an additional row. So in the worst case this entry can be carried all the way down to the m-th row. In this case L'_1 has no banded structure, and so the product $L = (L'_{m-1}...L'_2L'_1)^{-1}$ also has no banded structure.

Although L'_k is no longer banded, it is still sparse. In particular, each L'_k can have at most p below the diagonal entries since the subdiagonal entries of L_k are just permuted around. Therefore the product L will also have at most p entries below the main diagonal.