

# CSE 6643 Homework 1

Name: Muyang Guo  
GT username: mguo34

2020 Jan 26th

## 1 P1.1.3

Given an  $O(n^2)$  algorithm for computing  $C = (xy^T)^k$  where  $x$  and  $y$  are  $n$ -vectors. (Present the highest order term with its coefficient, need to present the fastest algorithm to get full credit. )

**Solution :**

$$\begin{aligned} C &= (xy^T)^k \\ &= (xy^T)(xy^T)\dots(xy^T) \quad \text{for } k \text{ terms} \\ &= x(y^T x)^{k-1} y^T \end{aligned}$$

After the transformation, we can observe that when  $k \geq 2$ ,  $y^T x$  is a dot product. For each  $y^T x$  term, involves  $n$  iterations, each iteration with one addition and one multiplication, thus,  $2n$  flops, and the result is a constant. For this step, *Big - O* is  $O(n)$

Denote  $y^T x$  as constant  $a$ , then we have:

$$C = x(a)^{k-1} y^T = a^{k-1} xy^T$$

Now the term  $xy^T$  is a matrix-vector multiplication, it required  $2n^2$  flops, For this step, *Big - O* is  $O(n^2)$   
Thus, for the complete computation with these two major steps, the flops-wise complexity is:

$$O(2n^2) + O(2n)$$

Thus, for  $k \geq 2$ , or  $k = 1$ , the Big-O is  $O(n^2)$  .

## 2 P1.1.5

Suppose we have real  $n$ -by- $n$  matrices  $C, D, E$  and  $F$ , show how to compute real  $n$ -by- $n$  matrices  $A$  and  $B$  with just three real  $n$ -by- $n$  matrix multiplications so that

$$A + iB = (C + iD)(E + iF)$$

hint: Compute  $W = (C + D)(E - F)$

**Solution :**

We have:

$$(C + iD)(E + iF) = CE - DF + i(CF + DE)$$

Thus,

$$\begin{aligned} A &= CE - DF \\ B &= CF + DE \end{aligned}$$

Suppose:

$$\begin{aligned} W &= (C + D)(E - F) \\ W &= CE + DE - CF - DF \end{aligned}$$

Then:

$$\begin{aligned} A + B &= W + 2CF \quad \dots \quad (1) \\ A - B &= W - 2DE \quad \dots \quad (2) \end{aligned}$$

We have:

$$\begin{aligned} A &= W + CF - DE \\ B &= CF + DE \end{aligned}$$

Thus, we can compute A and B with three real n-n matrices: W, CF, DE, where  $W = (C+D)(E-F)$ .

### 3 P1.2.2

Specify an algorithm that computes the first column of the matrix  $M = (A - \lambda_1 I) \dots (A - \lambda_r I)$  where  $A \in \mathbf{R}^{n \times n}$  is upper hessenberg and  $\lambda_1 \dots \lambda_r$  are given scalars, how many flops are required assuming that  $r \ll n$ ? (For the flop count, present the highest order term with its coefficient. You need to present the fastest algorithm to get full credit.)

**Solution:**

To compute the first column of  $M$ , we need to have a basis vector:  $e$ , that  $e = [1, 0, \dots, 0]^T \in \mathbf{R}^n$  to compute the product of  $Me$ .

$$\begin{aligned} FirstColumn &= (A - \lambda_1 I) \dots (A - \lambda_r I) e \\ &= (A - \lambda_1 I) \dots (A - \lambda_{r-1} I) (a_1 - \lambda_r e) \end{aligned}$$

Define the last term:  $y_1 = a_1 - \lambda_r e$  where it requires only one flops to compute  $y_1$ , which is just the first entry of the first column  $a_{11}$  minus the  $\lambda_r$ , as  $a_{11} - \lambda_r$ . moving forward to the last second term:

$$FirstColumn = (A - \lambda_1 I) \dots (A - \lambda_{r-2} I) (Ay_1 - \lambda_{r-1} y_1)$$

Then define  $y_2 = Ay_1 - \lambda_{r-1} y_1$ , since A is U.H matrix, the first column of A always have 2 entries only, the flops to count is:

$$2 \times 5 + 1 = 11 \quad \text{flops}$$

By observation, for next y, the result will be adding one more entry to the column. In this case, we can apply recursion to for the following matrices, until we get  $y_r$  which is the final result.

For  $y_{i+1} = Ay_i - \lambda_{r-i} y_i$ , flops count in each step following:  $2i^2 + i + 1$ , where  $i = 2, 3 \dots r-1$  Thus, total:

$$\begin{aligned} 1 + \sum_{i=2}^{r-1} (2i^2 + i + 1) &= 1 + \frac{2r^3 - 3r^2 + r - 6}{3} + \frac{r^2 - r - 2}{2} + r - 2 \\ &= \frac{2r^3}{3} - \frac{r^2}{2} + \frac{5r}{6} - 4 \end{aligned}$$

So the time complexity is  $O(r^3)$

#### 4 P1.3.9

Suppose  $A^{(k)} \in \mathbf{R}^{n_k \times n_k}$  for  $k = 1 : r$ , and that  $x \in \mathbf{R}^n$  where  $n = n_1 \dots n_r$ . Give an efficient algorithm for computing  $y = (A^{(r)} \otimes \dots \otimes A^{(2)} \otimes A^{(1)})x$

**Solution :**

We have:

$$Y = CXB^T$$

which is  $\text{vec}(Y) = (B \otimes C)\text{vec}(X)$ .

So  $y = (A^{(r)} \otimes \dots \otimes A^{(2)} \otimes A^{(1)})x = A^{(r-1)} \otimes \dots \otimes A^{(2)} \otimes A^{(1)} \text{reshape}(x, \frac{n}{nr}, nr)((A^{(r)})^T)$  Thus algorithm can be designed with recursion:

---

**Algorithm 1** P1.3.9 recursion

---

```

1: procedure GETY( $A^r, x$ )
2:    $n \leftarrow \text{sizeof } x, n_r \leftarrow \text{shape}(A^{(r)})[0]$ 
3:    $x = \text{reshape}(x, \frac{n}{n_r}, n_r)$ 
4:    $B = x * (A^{(r)})^T$ 
5:   for  $i = 1, 2, \dots, n_r$  do
6:      $Y[:, i] = \text{getY}(A^{(r-1)}, x)$ 
7:   return  $Y$ 
```

---

#### 5 P2.1.4

Assume that both  $A$  and  $A + uv^T$  are non-singular where  $A \in \mathbf{R}^{n \times n}$ , and  $u, v \in \mathbf{R}^n$ . Show that if  $x$  solves  $(A + uv^T)x = b$ , then it also solves a perturbed right-hand-side problem of the form  $Ax = b + \alpha u$ . Give an expression for  $\alpha$  in terms of  $A, u$  and  $v$ .

**Solution :**

$$(A + uv^T)x = b$$

$$Ax + uv^T x = b$$

$$Ax + u(v^T x) = b$$

$$Ax = b - u(v^T x)$$

$$\text{Since } Ax = b + \alpha u$$

$$\text{We can have: } \alpha = -(v^T x)$$

And recently  $x = (A + uv^T)^{-1}b$ , we can apply Sherman-Morrison formula:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}$$

Thus:

$$\alpha = -v^T (A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u})b$$

#### 6 P2.1.6

Suppose  $A \in \mathbf{R}^{n \times n}$  is symmetric and nonsingular and define  $\tilde{A} = A + \alpha(uu^T + vv^T) + \beta(uv^T + vu^T)$ , where  $u, v \in \mathbf{R}$  and  $\alpha, \beta \in \mathbf{R}$ . Assuming that  $\tilde{A}$  is non-singular, use the Sherman-Morrison-Woodbury formula to

develop a formula for  $\tilde{A}^{-1}$ .

**Solution :**

Denote two matrices with  $u, v$  as column.

$$\begin{aligned}\tilde{A} &= A + \alpha(uu^T + vv^T) + \beta(uv^T + vu^T) \\ &= A + [u \ v][\alpha u + \beta v \ \alpha v + \beta u]^T\end{aligned}$$

Let  $U = [u \ v]$ ,  $V = [\alpha u + \beta v \ \alpha v + \beta u]$ , we have:

$$\tilde{A} = A + UV^T$$

Applying Sherman-Morrison-Woodbury formula, assume  $\tilde{A}$  is non-singular:

$$\tilde{A}^{-1} = A^{-1} - A^{-1}U(I_2 + V^T A^{-1}U)^{-1}V^T A^{-1}$$

,where  $U, V$  are defined as  $U = [u \ v]$ ,  $V = [\alpha u + \beta v \ \alpha v + \beta u]$

## 7 P2.1.8

Suppose  $Q \in \mathbf{R}^{n \times n}$  is orthogonal and  $z \in \mathbf{R}^n$ . Give an efficient algorithm for setting up an  $m$ -by- $m$  matrix  $A = (a_{ij})$  defined by  $a_{ij} = v^T(Q^i)^T(Q^j)v$

**Solution :**

$Q$  is orthogonal matrix, the product of  $Q$  is also orthogonal matrix.

$$\begin{aligned}(Q^i)^T(Q^j) &= (Q^{i-j})^T && \text{for } i > j \\ &= I && \text{for } i = j \\ &= (Q^{j-i}) && \text{for } i < j\end{aligned}$$

Thus for the  $a_{ij}$  we have:

$$\begin{aligned}a_{ij} &= v^T(Q^{i-j})^T v && \text{for } i > j \\ &= v^T v && \text{for } i = j \\ &= v^T(Q^{j-i}) v && \text{for } i < j\end{aligned}$$

So the algorithm can be set up by the above conditions to get  $a_{ij}$ .

## 8 P2.2.9

Prove or disprove:  $v \in \mathbf{R}^n \implies \|v\|_1 \|v\|_\infty \leq \frac{1+\sqrt{n}}{2} \|v\|_2^2$

**Solution :**

let  $x_i$  denote  $|v_i|$ , the inequality is now:

$$\sum_{i=1}^n x_i x_{max} \leq \frac{1+\sqrt{n}}{2} \sum_{i=1}^n x_i^2$$

Denote  $x_{max} = x_n$ , suppose we have  $x_i$  in an ascending order. We have:

$$\begin{aligned}x_1 x_n + x_2 x_n + \dots + x_n^2 &\leq \frac{1+\sqrt{n}}{2} (x_1^2 + x_2^2 + \dots + x_n^2) \\ x_1 x_n + x_2 x_n + \dots + x_{n-1} x_n &\leq \frac{1+\sqrt{n}}{2} (x_1^2 + x_2^2 + \dots + x_{n-1}^2) + \frac{\sqrt{n}-1}{2} x_n^2\end{aligned}$$

the right side is :

$$\begin{aligned} \frac{1+\sqrt{n}}{2}(x_1^2 + x_2^2 + \dots + x_{n-1}^2) + \frac{\sqrt{n}-1}{2}x_n^2 &= \frac{1+\sqrt{n}}{2}(x_1^2 + x_2^2 + \dots + x_{n-1}^2) + \frac{n-1}{2(\sqrt{n}+1)}x_n^2 \\ &= \sum_{i=1}^{n-1} \left( \frac{1+\sqrt{n}}{2}x_i^2 + \frac{x_n^2}{2(\sqrt{n}+1)} \right) \end{aligned}$$

Since  $a^2 + b^2 \geq 2ab$ , we could have  $\frac{1+\sqrt{n}}{2}x_i^2 + \frac{x_n^2}{2(\sqrt{n}+1)} \geq 2\sqrt{\frac{1+\sqrt{n}}{2}x_i^2} \sqrt{\frac{x_n^2}{2(\sqrt{n}+1)}} = x_i x_n$ . Thus, the right side is larger than  $\sum_{i=1}^{n-1} x_i x_n$  which is just the left side of the inequality. So the inequality is correct.

## 9 P3.1.2

Suppose  $L = I_n - N$  is unit lower triangular where  $N \in \mathbf{R}^{n \times n}$ . Show that  $L^{-1} = I_n + N + N^2 + \dots + N^{n-1}$ . What is the value of  $\|L^{-1}\|_F$  if  $N_{ij} = 1$  for all  $i > j$ ?

**Solution :**

Since  $L = I_n - N$ : Compute:

$$\begin{aligned} (I_n - N)(I_n + N + N^2 + \dots + N^{n-1}) &= I_n + N + N^2 + \dots + N^{n-1} - N - N^2 - \dots - N^{n-1} - N^n \\ &= I_n - N^n \\ &= I_n \quad (\text{N is unit lower triangular matrix, } N^n = 0) \end{aligned}$$

Thus,  $L^{-1} = I_n + N + N^2 + \dots + N^{n-1}$  satisfied. Then consider that  $N_{ij} = 1$  for all  $i > j$ , then:

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ -1 & -1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ -1 & -1 & -1 & \dots & 1 \end{bmatrix}$$

For  $L^{-1}$  following the above equation:

$$L^{-1} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 2^0 & 1 & 0 & \dots & 0 \\ 2^1 & 2^0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 2^{n-2} & 2^{n-3} & 2^{n-4} & \dots & 1 \end{bmatrix}$$

Then the F-norm is:

$$\|L^{-1}\|_F = \sqrt{n + \sum_{k=1}^{n-1} (n-k)4^{k-1}} = \frac{1}{3}\sqrt{4n^2 + 6n - 1}$$

## 10 P3.1.7

Suppose  $L, K \in \mathbf{R}^{n \times n}$  are all lower triangular and  $B \in \mathbf{R}^{n \times n}$ . Give an algorithm for computing  $X \in \mathbf{R}^{n \times n}$  so that  $LXK = B$ .

**Solution :**

Assume L and K are nonsingular, Let  $A = XK$ ,  $A \in \mathbf{R}^{n \times n}$ , since  $LXK = B$ , we can have  $LA = B$ .

$$\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \dots & \dots & \dots & 0 \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}$$

Where  $a_k, b_k \in \mathbf{R}^n$ ,  $1 \leq k \leq n$ , then the pattern is :

$$\begin{aligned} b_1 &= 1_{11}a_1 \\ b_2 &= 1_{21}a_1 + l_{22}a_2 \\ &\dots \\ b_n &= 1_{n1}a_1 + 1_{n2}a_2 + \dots + l_{nn}a_n \end{aligned}$$

We can apply the following algorithm to find the A from LA = B:

---

**Algorithm 2** P3.1.7(1) Compute A from LA= B

---

```

1: compute  $a_1 = b_1/l_{11}$ 
2: for  $i = 2, 3 \dots n$  do
3:   Define  $s = 0, s \in \mathbf{R}^n$  ,
4:   for  $j = 1, 2 \dots, i - 1$  do
5:      $s := s + l_{i,j}a_j$ 
6:   end for
7:    $a_i = (b_i - s)/l_{ii}$ 
8: end for
9: return  $A = [a_1, a_2, \dots, a_n] \in \mathbf{R}^{n \times n}$ 

```

---

Then we just need to compute X from A = XK, similarly but in an inverse like pattern, we have:

$$\begin{aligned} a_n &= k_{nn}x_n \\ a_n - 1 &= k_{n-1,n-1}x_{n-1} + k_{n,n-1}x_n \\ &\dots \\ a_1 &= k_{1,1}x_1 + k_{21}x_2 + \dots + k_{n1}x_n \end{aligned}$$

We have the following algorithm to compute the X from A = XK:

---

**Algorithm 3** P3.1.7(2) Compute X from XK= A

---

```

1: compute  $x_n = a_n/k_{nn}$ 
2: for  $i = n - 1, \dots, 1$  do
3:   Define  $s = 0, s \in \mathbf{R}^n$  ,
4:   for  $j = i + 1, \dots, n$  do
5:      $s := s + k_{j,i}x_j$ 
6:   end for
7:    $x_i = (a_i - s)/k_{ii}$ 
8: end for
9: return  $X = [x_1, x_2, \dots, x_n] \in \mathbf{R}^{n \times n}$ 

```

---

## 11 P3.2.5

Describe a variant of Gaussian elimination that introduces zeros into the columns of A in the order, n: - 1 :2 and which produces the factorization A = UL where U is unit upper triangular and L is lower triangular .

**Solution :**

We could follow the LU factorization, to get A = UL: Let

$$A_{(1)} = A = \begin{bmatrix} x & x & \dots & x & a_{1n} \\ x & x & \dots & x & a_{2n} \\ x & x & \dots & x & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ x & x & \dots & x & a_{nn} \end{bmatrix}$$

$$M_1 = \begin{bmatrix} 1 & & & -a_{1n}/-a_{nn} \\ & 1 & & -a_{2n}/-a_{nn} \\ & & 1 & -a_{3n}/-a_{nn} \\ & & \dots & \dots \\ & & & 1 \end{bmatrix}$$

And we could have:

$$A_{(2)} = M_1 A_{(1)} = \begin{bmatrix} x & x & \dots & a_{1(n-1)}^{(2)} & 0 \\ x & x & \dots & a_{2(n-1)}^{(2)} & 0 \\ x & x & \dots & a_{3(n-1)}^{(2)} & 0 \\ \dots & \dots & \dots & \dots & \dots \\ x & x & \dots & a_{n(n-1)}^{(2)} & a_{nn} \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & & & -a_{1(n-1)}^{(2)}/a_{n(n-1)}^{(2)} \\ & 1 & & -a_{2(n-1)}^{(2)}/a_{n(n-1)}^{(2)} \\ & & 1 & -a_{3(n-1)}^{(2)}/a_{n(n-1)}^{(2)} \\ & & \dots & \dots \\ & & & 1 \end{bmatrix}$$

Continuing for  $A_3, A_4 \dots$  we could see the pattern, eventually for  $A_{n-1} = M_{n-1}M_{n-2}\dots M_2M_1A_1$  will be come a lower triangular matrix.

Then we have:

$$A_1 = (M_{n-1}M_{n-2}\dots M_2M_1)^{-1}A_{n-1}$$

Thus we have:  $U = (M_{n-1}M_{n-2}\dots M_2M_1)^{-1}$ , and  $L = A_{n-1}$  which could satisfy the conditions.

## 12 P3.2.6

Matrices  $\in \mathbf{R}^{n \times n}$  of the form  $N(y, k) = I - ye_k^T$  where  $y \in \mathbf{R}^n$  are called Gauss-Jordan transformations. (a) Give a formula for  $N(y, k)^{-1}$  assuming it exists. (b) Given  $x \in \mathbf{R}^n$ , under what conditions can  $y$  be found so  $N(y, k)x = e_k$ ? (c) Give an algorithm using Gauss-Jordan transformations that overwrites  $A$  with  $A^{-1}$ . What conditions on  $A$  ensure the success of your algorithm?

**Solution :**

**a:** We can use Sherman-Morrison-Woodbury formula:

$$\begin{aligned} N(y, k) &= I + (-ye_k^T) \\ N(y, k)^{-1} &= I^{-1} - \frac{I^{-1}(-y)e_k^T I^{-1}}{I + e_k^T I^{-1}(-y)} \\ &= I + \frac{ye_k^T}{1 - ye_k^T} \end{aligned}$$

**b:** Let

$$\begin{aligned}
(I - ye_k^T)x &= e_k \\
x - e_k &= ye_k^T x \\
y &= \frac{x - e_k}{e_k^T x}
\end{aligned}$$

Thus, when  $e_k^T$  is not 0,  $y$  can be found.

**c:**

When  $i \neq j$ ,  $N(y, k)e_j = e_j$ , we could have the following algorithm:

---

**Algorithm 4** P3.2.6 G-J

---

```

1: for  $k = 1, 2, \dots, n$  do
2:    $y = (A(:, k) - e_k)/A_{kk}$ 
3:    $A(:, k) = e_k$ 
4:    $A = A - yA(k, :)$ 
5: end for

```

---

Similarly to LU factorization, considering each iteration,  $A^{k-1} \leftarrow A$ , thus,  $A^{k-1} = N_{k-1}N_{k-2}\dots N_1A$ , due to  $\det(N_i)$  is not 0, then,  $\det(A_{11})$  and  $\det(A[1 : k, 1 : k])$  are not 0. we could have  $A_{kk}^{k-1} \neq 0$  Thus to successfully ensure the algorithm,  $\det(A(1 : k, 1 : k)) \neq 0$ , where  $k = 1, 2, \dots, n$  needs to be satisfied.

### 13 P3.4.1

Let  $A = LU$  be the LU factorization of  $n$ -by- $n$   $A$  with  $|l_{ij}| \leq 1$ . Let  $a_i^T$  and  $u_i^T$  denote the  $i$ th rows of  $A$  and  $U$ , respectively. Verify the equation  $u_i^T = a_i^T - \sum_{j=1}^{i-1} l_{ij}u_j^T$  and use it to show that  $\|U\|_\infty \leq 2^{n-1}\|A\|_\infty$  (Hint: Take norms and use induction)

**Solution :**

$$a_{ij} = \sum_{k=1}^i l_{ik}u_{kj} + \sum_{k=i+1}^n l_{ik}u_{kj}^T$$

For  $k > i$ ,  $l_{ik} = 0$ ,  $a_{ij} = \sum_{k=1}^i l_{ik}u_{kj}$ , then the row of  $A$  is :  $a_i^T = \sum_{j=1}^i l_{ij}u_j^T = l_{ii}u_i^T + \sum_{j=1}^{i-1} l_{ij}u_j^T$  Since  $l_{ii} = 1$ . thus proving  $u_i^T = a_i^T - \sum_{j=1}^{i-1} l_{ij}u_j^T$ .

Then using induction, check the first row of  $A$  and  $U$ . Denote  $A_i$  as  $A$ 's 1st to  $i$ th rows. Similarly for  $U_i$ : Thus when  $k = 1$ ,

$$u_1 = a_1 \|U_1\|_\infty \leq 2^{1-1} \|A_1\|_\infty$$

For  $k = i-1$ , we then have  $\|U_{i-1}\|_\infty \leq 2^{i-2} \|A_{i-1}\|_\infty$

For  $k = i$ , suppose  $\|U_i\|_\infty = \|U_{i-1}\|_\infty$ , due to  $\|U_{i-1}\|_\infty \leq 2^{i-2} \|A_{i-1}\|_\infty$ . Then we have  $\|U_1\|_\infty \leq 2^{i-1} \|A_i\|_\infty$  Then consider when  $\|U_i\|_\infty \neq \|U_{i-1}\|_\infty$ , the infinity norm is the  $i$ th row of  $U$  which is largest,

$$\begin{aligned}
u_i^T &= a_i^T - \sum_{j=1}^{i-1} l_{ij}u_j^T \\
\|U_i\|_\infty &= \|a_i^T - \sum_{j=1}^{i-1} l_{ij}u_j^T\|_\infty \\
\|U_i\|_\infty &\leq \|a_i^T\|_\infty + \|u_1^T\|_\infty + \|u_2^T\|_\infty + \dots + \|u_{i-1}^T\|_\infty \\
&= \|A_i^T\|_\infty + \|U_1^T\|_\infty + \|U_2^T\|_\infty + \dots + \|U_{i-1}^T\|_\infty \\
&\leq \|A_i^T\|_\infty + \|A_1^T\|_\infty + 2\|A_2^T\|_\infty + \dots + 2^{i-2}\|A_{i-1}^T\|_\infty \\
&= 2^{i-1}\|A_i^T\|_\infty
\end{aligned}$$



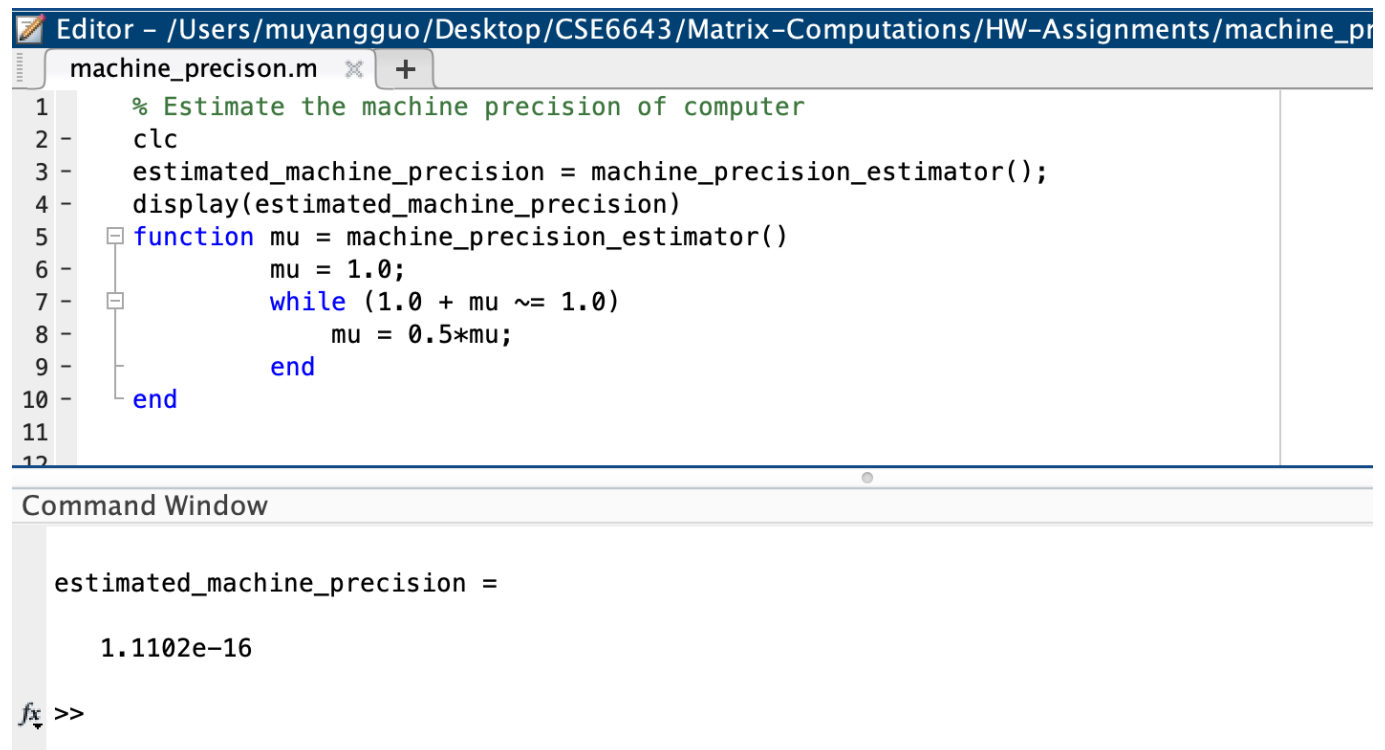
Thus,  $\|U_i\|_\infty \leq 2^{i-1}\|A_i^T\|_\infty$  when  $i = 1, 2, \dots, n$ , and  $\|U\|_\infty \leq 2^{i-1}\|A\|_\infty$  when  $i = n$ . We have validated the inequality.

## 14 Matlab for machine precision programming question

Present an algorithm to estimate the machine precision of a floating point system assuming you do not know the base  $b$  nor the number of mantissa digits  $t$ . Justify your answer. Then write a computer program based on your algorithm where the output is the estimated machine precision for double precision computation on your system. Present the code and the output. I encourage you to do all programming in Matlab. If you are not familiar with Matlab, then you may use other language for HW1. But it will make your work easier to eventually use Matlab for studying numerical algorithms.

### Solution :

We could use a one-order or magnitude of the true value, which is the machine precision falls within  $[\mu, 2\mu)$ , to perform a linear search to approximate the machine precision. As we do not know the base and mantissa. The function in Matlab is shown below:



```

Editor - /Users/muyanguo/Desktop/CSE6643/Matrix-Computations/HW-Assignments/machine_pr
machine_precision.m
1 % Estimate the machine precision of computer
2 clc
3 estimated_machine_precision = machine_precision_estimator();
4 display(estimated_machine_precision)
5 function mu = machine_precision_estimator()
6     mu = 1.0;
7     while (1.0 + mu ~= 1.0)
8         mu = 0.5*mu;
9     end
10 end
11
12

Command Window

estimated_machine_precision =

    1.1102e-16

fx >>

```

And my estimated machine precision by this algorithm is 1.1102e-16.