# Computational Linear Algebra CW1

JIAQI LI

$1^{th}$ December 2021

## 1 Question2

### 1.1 question a

To calculate $C = (xy^T)^k$, I can use the algorithm $C = (xy^T)^k = x(y^Tx)^{k-1}y^T$.
For $y^Tx$, I need to do n products and $n-1$ additions here because $x, y \in \mathcal{R}^n$.
For $(y^Tx)^{k-1}$, because the result of $y^Tx$ is a value, I only need to do $k-1$ products here.
For $x(y^Tx)^{k-1}$, this is the multiplication with a value $(y^Tx)^{k-1}$ and a vector $x$, so I need to do $n$ products here.
For $x(y^Tx)^{k-1}y^T$, because $x(y^Tx)^{k-1}, y \in \mathcal{R}^n$, this is a multiplication with 2 vectors, I need to do $n^2$ products here.
To find out the leading order $\mathcal{O}$ term, I only need to find the largest element contributing to this algorithm, I can find that this algorithm is in $\mathcal{O}(n^2)$.

### 1.2 question b

For $(A^TB)A$: First we calculate $A^TB$, where $A^T \in \mathcal{R}^{n\times m}$ and $B \in \mathcal{R}^{m\times m}$, so to calculate $A^TB$, each entry of $A^TB$ need to do $2m-1$ multiplications, and the size of $A^TB$ is $n \times m$, so the total operation count is $(2m-1)mn$. Secondly, I will calculate $(A^TB)A$, where $A^TB \in \mathcal{R}^{n\times m}$ and $A \in \mathcal{R}^{m\times n}$, similar with the operation in $A^TB$, the total operation count is $(2m-1)n^2$. So for the whole $(A^TB)A$, the operation count is $(2m-1)(mn+n^2)$.
For $A^T(BA)$: First we calculate $BA$, where $B \in \mathcal{R}^{m\times m}$ and $A \in \mathcal{R}^{m\times n}$, so to calculate $BA$, each entry of $BA$ need to do $2m-1$ multiplications, and the size of $BA$ is $m \times n$, so the total operation count is $(2m-1)mn$. Secondly, I will calculate $A^T(BA)$, where $A^T \in \mathcal{R}^{n\times m}$ and $BA \in \mathcal{R}^{m\times n}$, similar with the operation in $BA$, the total operation count is $(2m-1)n^2$. So for the whole $(A^TB)A$, the operation count is $(2m-1)(mn+n^2)$.
So for each value of m and n, these two algorithms have the same operation count.

### 1.3 question c

**Method:**
To find out the real and imaginary part of $A = (P+iQ)(R+iS)$, which are $PR-QS$, and $QR+PS$ I will do in the follow steps,

1. Calculate the matrix multiplication:

$$T = (P + Q)(R - S) = PR - QS + QR - PS$$

2. Calculate the matrix multiplications QR and PS
3. I can get that the real part of $A = T - QR + PS$, and the imaginary part of $A = QR + PS$

**Operation Count:**

To compute $T = (P + Q)(R - S) = PR - QS + QR - PS$, because $P, Q, R, S \in \mathcal{R}^{m\times m}$, I need to do $m^2$ additions in both of $P + Q$ and $R - S$, then I need to do $2m^2$ additions in computing T. In addition, I still need to do the multiplication part in T, because $P + Q, R - S \in \mathcal{R}^{m\times m}$, and each position in the result of T need to have $2m - 1$ multiplications, and the size of T is $m \times m$, then I can find that the total operation count in T is $2m^2 + (2m - 1)m^2 = 2m^3 + m^2$.
To compute QR ad PS, do similar with $(P + Q)(R - S)$, the operation count in QR and PS are both $(2m - 1)m^2 = 2m^3 - m^2$.
Then for computing the real part of A, $Re(A) = T - QR + PS$, that means I need to do $m^2$ additions and $m^2$ subtractions here. Combine with the operation of T, QR and PS(because I need to use T, QR and PS here), the total operation count in real part is $2(2m^3 - m^2) + 2m^3 + m^2 + 2m^2 = 6m^3 + m^2$.
Finally, for computing the imaginary part of A, $Im(A) = QR + PS$, that means I need to do $m^2$ additions in this process. Combine with the operation of QR and PS(because I need to use QR and PS here), the total operation count in imaginary part is $m^2 + 2(2m^3 - m^2) = 4m^3 - m^2$.

# 2 Question3

## 2.1 question a

Let $A \in \mathcal{R}^{2\times 2}$, and set a error matrix $E \in \mathcal{R}^{2\times 2}$ where $\widetilde{A} = A + E$. And I assume A has 2 eigenvalues $\lambda_1$ and $\lambda_2$, $\widetilde{A}$ has 2 eigenvalues $\widetilde{\lambda_1}$ and $\widetilde{\lambda_2}$.
Stability: We can write the algorithm as $\widetilde{f}$ which is a floating point implementation and the problem to find out the eigenvalue $f$, then this algorithm is stable if for each $A \in \mathcal{R}^{2\times 2}$, $\frac{\|\widetilde{f}(A) - f(\widetilde{A})\|}{\|f(\widetilde{A})\|} = \mathcal{O}(\epsilon)$, $\exists \frac{\|\widetilde{A} - A\|}{\|A\|} = \mathcal{O}(\epsilon)$. And by the assumption that A has 2 eigenvalues $\lambda_1$, $\lambda_2$ and $\widetilde{A}$ has 2 eigenvalues $\widetilde{\lambda_1}$, $\widetilde{\lambda_2}$, this also means that for each $A \in \mathcal{R}^{2\times 2}$, $\frac{\|\lambda_1 - \widetilde{\lambda_1}\|}{\|\widetilde{\lambda_1}\|} = \mathcal{O}(\epsilon)$ and $\frac{\|\lambda_2 - \widetilde{\lambda_2}\|}{\|\widetilde{\lambda_2}\|} = \mathcal{O}(\epsilon)$, $\exists \frac{\|\widetilde{A} - A\|}{\|A\|} = \mathcal{O}(\epsilon)$

## 2.2 question b

Let $A \in \mathcal{R}^{2\times 2}$, and set a error matrix $E \in \mathcal{R}^{2\times 2}$ where $\widetilde{A} = A + E$. And I assume A has 2 eigenvalues $\lambda_1$ and $\lambda_2$, $\widetilde{A}$ has 2 eigenvalues $\widetilde{\lambda_1}$ and $\widetilde{\lambda_2}$.
Backward Stability: We can write the algorithm as $\widetilde{f}$ which is a floating point implementation and the problem to find out the eigenvalue as $f$, then this algorithm is backward stable if for each $A \in \mathcal{R}^{2\times 2}$, $\exists \widetilde{A}$, such that $\widetilde{f}(A) = f(\widetilde{A})$ with $\frac{\|\widetilde{A} - A\|}{\|A\|} = \mathcal{O}(\epsilon)$. And by the assumption that A has 2 eigenvalues $\lambda_1$, $\lambda_2$ and $\widetilde{A}$ has 2 eigenvalues $\widetilde{\lambda_1}$, $\widetilde{\lambda_2}$, this also means that for each $A \in \mathcal{R}^{2\times 2}$, $\exists \widetilde{A}$, such that $\lambda_1 = \widetilde{\lambda_1}$ and $\lambda_2 = \widetilde{\lambda_2}$ with $\frac{\|\widetilde{A} - A\|}{\|A\|} = \mathcal{O}(\epsilon)$.

## 2.3 question c

The code are shown in cw2.CW2.py by function $soleiv1(A)$ abd $soleiv2(A)$.
Use the algorithm by python scripts I can find that the results to find out the eigenvalues of $A_1$ and $A_2$. At the same time, I also calculate the exact solution of A1 and A2 by hand, then I will compare with them and talk about their differences.
$A_1$: I can calculate the result of $A_1$ by python script and denote them as $\widetilde{\lambda_{11}}$ and $\widetilde{\lambda_{12}}$ where $\widetilde{\lambda_{11}} = 1.0$ and $\widetilde{\lambda_{12}} = 1.0$. Then I also calculate by hand and denote eigenvalues as $\lambda_{11}$ and $\lambda_{12}$, I find that $\lambda_{11} = 1$ and $\lambda_{12} = 1$. So $\|\widetilde{\lambda_{11}} - \lambda_{11}\| = 0$ and $\|\widetilde{\lambda_{12}} - \lambda_{12}\| = 0$ so the error is 0 for $A_1$. But $\widetilde{\lambda_{11}}, \lambda_{11}$ and $\widetilde{\lambda_{12}}, \lambda_{12}$ both of these pair are in different type, $\widetilde{\lambda_{11}}$ and $\widetilde{\lambda_{12}}$ are floats, $\lambda_{11}$ and $\lambda_{12}$ which are calculated by hand are integers because the number calculated by $sqrt()$ function in python scripts are in float form.
$A_2$: I can calculate the result of $A_2$ by python script and denote them as $\widetilde{\lambda_{21}}$ and $\widetilde{\lambda_{22}}$ where $\widetilde{\lambda_{21}} = 1.0000000149011663$ and $\widetilde{\lambda_{22}} = 0.9999999850988439$. Then I also calculate by hand and denote eigenvalues as $\lambda_{21}$ and $\lambda_{22}$, I find that $\lambda_{21} = 1.00000000000001$ and $\lambda_{22} = 1$. So $\|\widetilde{\lambda_{21}} - \lambda_{21}\| = 1.4901156308866348 \times 10^{-8}$ and $\|\widetilde{\lambda_{12}} - \lambda_{12}\| = 1.4901156086821743 \times 10^{-8}$ so there are errors in computing the eigenvalue of $A_2$.

## 2.4 question d

By machine epsilon I want to calculate the expected error, then I will write $f(x+\delta) - f(x)$ first.
$f(x+\delta) = \widetilde{\lambda_1}, \widetilde{\lambda_2}$.
$\widetilde{\lambda_1} = \frac{(2+1\times10^{-14})+\sqrt{(2+1\times10^{-14})^2-4\times(1+1\times10^{-14})}}{2} = \frac{(2+1\times10^{-14}+\sqrt{1\times10^{-28}}}{2} = 1 + \frac{1\times10^{-14}+\sqrt{1\times10^{-28}}}{2}$,
$\widetilde{\lambda_2} = \frac{(2+1\times10^{-14})-\sqrt{(2+1\times10^{-14})^2-4\times(1+1\times10^{-14})}}{2} = \frac{(2+1\times10^{-14}-\sqrt{1\times10^{-28}}}{2} = 1 + \frac{1\times10^{-14}-\sqrt{1\times10^{-28}}}{2}$.
$f(x) = \lambda_1, \lambda_2$.
$\lambda_1 = \frac{2+\sqrt{4-4}}{2} = \frac{2}{2} = 1$, $\lambda_2 = \frac{2-\sqrt{4-4}}{2} = \frac{2}{2} = 1$.
So $f(x+\delta) - f(x) = \widetilde{\lambda_1} - \lambda_1, \widetilde{\lambda_2} - \lambda_2$.
$\widetilde{\lambda_1} - \lambda_1 = 1 + \frac{1\times10^{-14}+\sqrt{1\times10^{-28}}}{2} - 1 = \frac{1\times10^{-14}+\sqrt{1\times10^{-28}}}{2}$
$\widetilde{\lambda_2} - \lambda_2 = 1 + \frac{1\times10^{-14}-\sqrt{1\times10^{-28}}}{2} - 1 = \frac{1\times10^{-14}-\sqrt{1\times10^{-28}}}{2}$
. By the machine epsilon is: $\epsilon = 2^{-52} \approx 2.2 \times 10^{-16}$, so the machine precision here is $2.2 \times 10^{-16}$, and by the fact that $10^{-28}$ is much smaller than $10^{-16}$, I can only see $1 \times 10^{-28}$ as $2.2 \times 10^{-16}$, so the expected error can be seen as $f(x+\delta) - f(x) = \widetilde{\lambda_1} - \lambda_1, \widetilde{\lambda_2} - \lambda_2$.
$\widetilde{\lambda_1} - \lambda_1 = \frac{1\times10^{-14}+\sqrt{2.2\times10^{-16}}}{2} = \frac{1\times10^{-14}+1\times10^{-8}\sqrt{2.2}}{2} \approx 0.74 \times 10^{-8} + 1 \times 10^{-16} \approx 0.74 \times 10^{-8}$,
$\widetilde{\lambda_2} - \lambda_2 = \frac{1\times10^{-14}-\sqrt{2.2\times10^{-16}}}{2} = \frac{1\times10^{-14}-1\times10^{-8}\sqrt{2.2}}{2} \approx -0.74 \times 10^{-8} + 1 \times 10^{-16} \approx -0.74 \times 10^{-8}$.
By this I can find that the expected error in this question for $A_2$ can be calculated approximately $0.74 \times 10^{-8}$ and $-0.74 \times 10^{-8}$, which is similar with the results that I find by python scripts, and the small error is due to the system errors in Python itself.

# 3 Question4

## 3.1 question a

I was able to randomly output A, L and U through the code. Through observation, I found that for A in $4n+1 \times 4n+1$ size, L was formed by n $5 \times 5$ lower triangular matrices with one number overlapping each other, which was a banded matrix with lower bandwidths = 4. Specifically, for k that could take in range$(0, n-1)$, when $4k+1 \leq i, j \leq 4k+5$, there is a $5 \times 5$ lower triangular matrix stored in that place, and at $i, j = 4(k+1)+1$ these matrices overlap each other. In addition U was formed by n $5 \times 5$ upper triangular matrices with one number overlapping each other, which was a banded matrix with upper bandwidths = 4. Specifically, for k that could take in range$(0, n-1)$, when $4k+1 \leq i, j \leq 4k+5$, there is a $5 \times 5$ upper triangular matrix stored in that place, and at $i, j = 4(k+1)+1$ these matrices overlap each other.
Then I will compare the bandwidth of A, L and U. The expected sparsity structure of banded matrices with the upper and lower bandwidths of A are both 4, because there are n $5 \times 5$ matrices in the diagonal of A. And by the observation of L and U, the lower bandwidths of L and the upper bandwidth of U are also 4, which is the same as the expected upper and lower bandwidths of A, besides, the upper bandwidth of L and the lower bandwidth of U are both 0.
For example when n = 3:

$$A = \begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} & a_{59} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{65} & a_{66} & a_{67} & a_{68} & a_{69} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} & a_{78} & a_{79} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{85} & a_{86} & a_{87} & a_{88} & a_{89} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{95} & a_{96} & a_{97} & a_{98} & a_{99} & a_{9,10} & a_{9,11} & a_{9,12} & a_{9,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{10,9} & a_{10,10} & a_{10,11} & a_{10,12} & a_{10,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11,9} & a_{11,10} & a_{11,11} & a_{11,12} & a_{11,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{12,9} & a_{12,10} & a_{12,11} & a_{12,12} & a_{12,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{13,9} & a_{13,10} & a_{13,11} & a_{13,12} & a_{13,13}
\end{pmatrix}$$

$$L = \begin{pmatrix}
l_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l_{21} & l_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l_{31} & l_{32} & l_{33} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l_{41} & l_{42} & l_{43} & l_{44} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l_{51} & l_{52} & l_{53} & l_54 & l_{55} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & l_{65} & l_{66} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & l_{75} & l_{76} & l_{77} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & l_{85} & l_{86} & l_{87} & l_{88} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & l_{95} & l_{96} & l_{97} & l_{98} & l_{99} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_{10,9} & l_{10,10} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_{11,9} & l_{11,10} & l_{11,11} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_{12,9} & l_{12,10} & l_{12,11} & l_{12,12} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & l_{13,9} & l_{13,10} & l_{13,11} & l_{13,12} & l_{13,13}
\end{pmatrix}$$

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_{33} & u_{34} & u_{35} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_{44} & u_{45} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_{55} & u_{56} & u_{57} & u_{58} & u_{59} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & u_{66} & u_{67} & u_{68} & u_{69} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_{77} & u_{78} & u_{79} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{88} & u_{89} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{99} & u_{9,10} & u_{9,11} & u_{9,12} & u_{9,13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{10,10} & u_{10,11} & u_{10,12} & u_{10,13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{11,11} & u_{11,12} & u_{11,13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{12,12} & u_{12,13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{13,13} \end{pmatrix}$$

### 3.2 question b

The original algorithm for the banded matrix which is mentioned in notes(The Gaussian elimination algorithm without pivoting):

1.U = A

2.L = U

3.for k in range(m - 1):

  for $j = k + 1$ TO $min(k + p, m)$:

    $L[j, k] = U[j, k]/U[k, k]$

    $n = min(k + q, m)$

    $U[j, k : m] = U[j, k : m] - L[j, k : n]U[k, k : n]$

    END FOR

  END FOR

Note: p and q are both 5 here.

For a modification of the banded matrix algorithm I use the method similar with the one-step inplace_LU which is mentioned in exercise6:

1. $m = length(A)$ which means A is an $m \times m$ matrix

2. for k in range(m - 1):

  $a = min(k + 5 - kmod4, m)$ which can be wrote as $a = k + 5 - kmod4$, because when k choose the maximum value $m - 2$, $k + 5 - kmod4$ is still smaller than m.

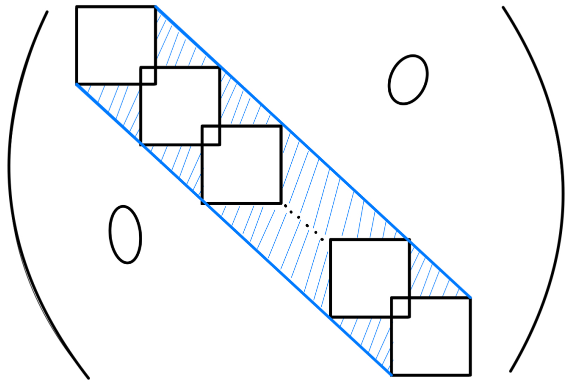
  $A[k + 1 : a, k] = A[k + 1 : a, k]/A[k, k]$

  $A[k+1 : a, k+1 : a] = A[k+1 : a, k+1 : a]-$ outer product of $A[k+1 : a, k] and A[k, k+1 : a]$

  END FOR

Then I will explain the improvement of the new algorithm compared with the original form:

We can see A as the below form:



The shaded part are also zero, but in the original algorithm these zero parts will do some addition or multiplication which is complex and meaningless, then I use the new algorithm to avoid these types of calculation. We can find that for calculating both L and U, they only need to focus on $k+5-kmod4$ rows or columns, for example, when I computing U, In the first block in $A[0:5,0:5]$, I generate in row 0, 1, 2, 3, 4 step by step, and I need to computing 4 elements in row 0 (element $l_{12}$, $l_{13}$, $l_{14}$, $l_{15}$), 3 elements in row 1($l_{23}$, $l_{24}$, $l_{25}$), 2 elements in row 2($l_{34}$, $l_{35}$), etc.

**Operation count:** The operation count is $\mathcal{O}(n)$.

Explain: Before I starting I will first mentioned that A is an $m \times m$ matrix, and $m = 4*n+1$, by the definition of A in question stem.

Ignoring calculate operation count in index number:

In $A[k+1:a,k] = A[k+1:a,k]/A[k,k]$: There are $num-k$ divisions in each k. By $num = k+5-kmod4$ I can make each 4 k in one group, then each 4 k will do $4+3+2+1 = 10$ divisions. By k is from 0 to $m-2$, there are $m-1$ k here, also by $m = 4*n+1$, I can find that there are $10n$ divisions in this formula.

In $A[k+1:a,k+1:a] = A[k+1:a,k+1:a] - np.outer(A[k+1:a,k], A[k,k+1:a])$: For the outer product part, because the size of vectors $A[k+1:a,k]$ and $A[k,k+1:a]$, there are $(num-k)^2$ operations. For the subtraction part, also because of their shape, there are $(num-k)^2$ operations. Similarly with the last part in this question, I make each 4 k in one group. Then in each group there are $4^2+3^2+2^2+1^2 = 30$ operations in outer product and $4^2+3^2+2^2+1^2 = 30$ operations in subtraction. In addition by k is from 0 to $m-2$, there are $m-1$ k, and I also know $m = 4*n+1$, so I can find the total operation count in this part is $60n$.

So for the total algorithm, the operation count is $60n+10n = 70n$, then I can say that the operation count is $\mathcal{O}(n)$.

### 3.3 question c

The code are shown in cw2.CW2.py by function CWLU_inplace(A).

The automated tests are created in folder test_cw2, in test_coursework2.py, and use 2 functions testA_CWLU_inplace(n, epsilon) and testLU_CWLU_inplace(n, epsilon). In addition I also use function creatA(n, epsilon) which are stored in cw2.CW2.py to create the specific A in this question.

### 3.4 question d

**Algorithm:**
Step1:
Do row elimination of A and b where A is in specific form(for example when n = 3):

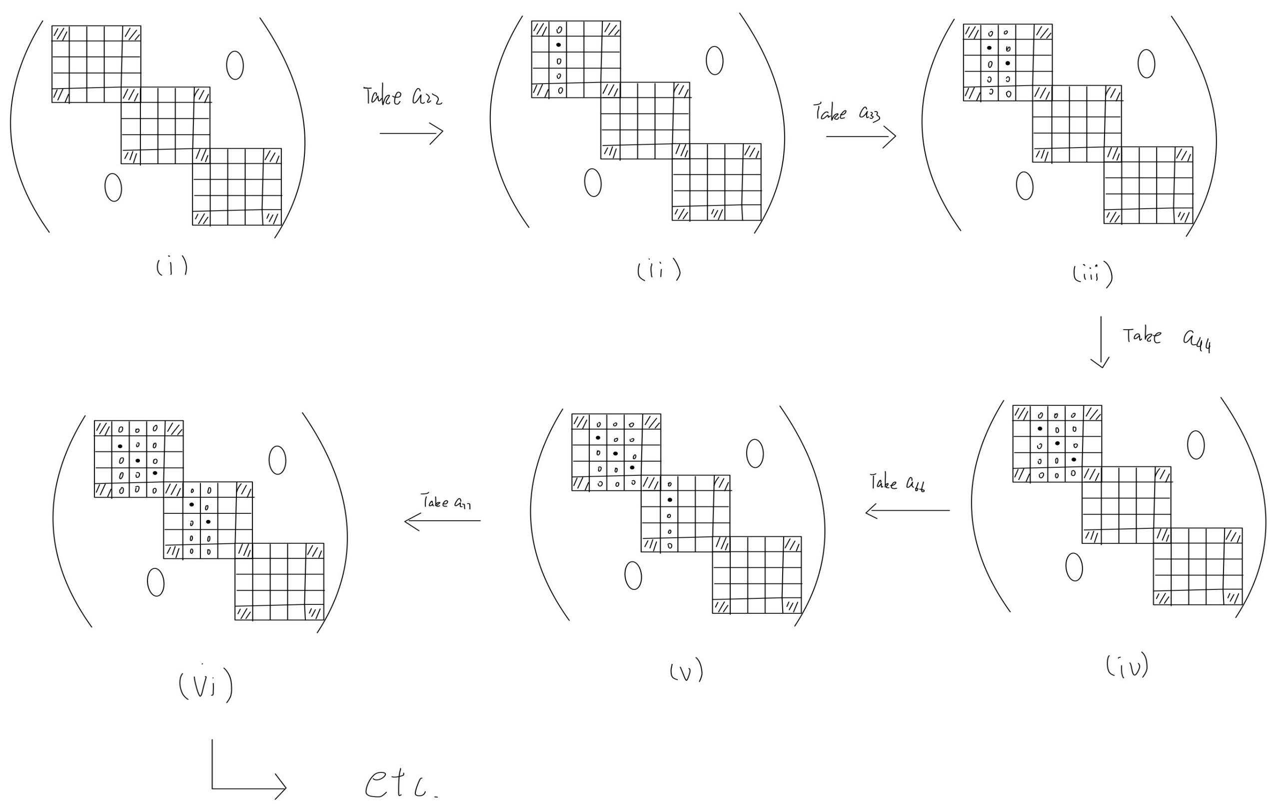
$$A = \begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} & a_{59} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{65} & a_{66} & a_{67} & a_{68} & a_{69} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} & a_{78} & a_{79} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{85} & a_{86} & a_{87} & a_{88} & a_{89} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_{95} & a_{96} & a_{97} & a_{98} & a_{99} & a_{9,10} & a_{9,11} & a_{9,12} & a_{9,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{10,9} & a_{10,10} & a_{10,11} & a_{10,12} & a_{10,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11,9} & a_{11,10} & a_{11,11} & a_{11,12} & a_{11,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{12,9} & a_{12,10} & a_{12,11} & a_{12,12} & a_{12,13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{13,9} & a_{13,10} & a_{13,11} & a_{13,12} & a_{13,13}
\end{pmatrix}$$

And I need to do row elimination end in the form which is shown below:

$$\widetilde{A} = \begin{pmatrix}
\widetilde{a_{11}} & 0 & 0 & 0 & \widetilde{a_{15}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\widetilde{a_{21}} & \widetilde{a_{22}} & 0 & 0 & \widetilde{a_{25}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\widetilde{a_{31}} & 0 & \widetilde{a_{33}} & 0 & \widetilde{a_{35}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\widetilde{a_{41}} & 0 & 0 & \widetilde{a_{44}} & \widetilde{a_{45}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\widetilde{a_{51}} & 0 & 0 & 0 & \widetilde{a_{55}} & 0 & 0 & 0 & \widetilde{a_{59}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \widetilde{a_{65}} & \widetilde{a_{66}} & 0 & 0 & \widetilde{a_{69}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \widetilde{a_{75}} & 0 & \widetilde{a_{77}} & 0 & \widetilde{a_{79}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \widetilde{a_{85}} & 0 & 0 & \widetilde{a_{88}} & \widetilde{a_{89}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \widetilde{a_{95}} & 0 & 0 & 0 & \widetilde{a_{99}} & 0 & 0 & 0 & \widetilde{a_{9,13}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \widetilde{a_{10,9}} & \widetilde{a_{10,10}} & 0 & 0 & \widetilde{a_{10,13}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \widetilde{a_{11,9}} & 0 & \widetilde{a_{11,11}} & 0 & \widetilde{a_{11,13}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \widetilde{a_{12,9}} & 0 & 0 & \widetilde{a_{12,12}} & \widetilde{a_{12,13}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \widetilde{a_{13,9}} & 0 & 0 & 0 & 0
\end{pmatrix}$$

To show how I can do this:
Focus on each $5 \times 5$ block matrix in A,

Take $n=3$ for example




By this graph, from (i) to (ii) I choose $a_{22}$ entry of A and do row operations to make $c_{i2} = 0\ \forall i$.
from (ii) to (iii) I choose $a_{33}$ entry of A and do row operations to make $c_{i3} = 0\ \forall i$.
from (iii) to (iv) I choose $a_{44}$ entry of A and do row operations to make $c_{i4} = 0\ \forall i$.
from (iv) to (v) I choose $a_{66}$ entry of A and do row operations to make $c_{i6} = 0\ \forall i$.
from (v) to (vi) I choose $a_{77}$ entry of A and do row operations to make $c_{i7} = 0\ \forall i$ and so on similar operations for each $5 \times 5$ block in A.

Step2:

Use the tridiagonal matrix and vector $\widetilde{A}, \widetilde{b}$ which are in lower dimensions to calculate $\widetilde{x} = \begin{bmatrix} x_1 \\ x_5 \\ x_9 \\ x_{13} \\ \vdots \end{bmatrix}$.

To show more clearly, use the example when n =3, in this case I need to recreate:

$$\widetilde{A} = \begin{pmatrix} \widetilde{a_{11}} & \widetilde{a_{15}} & 0 & 0 \\ \widetilde{a_{51}} & \widetilde{a_{55}} & \widetilde{a_{59}} & 0 \\ 0 & \widetilde{a_{95}} & \widetilde{a_{99}} & \widetilde{a_{9,13}} \\ 0 & 0 & \widetilde{a_{13,9}} & \widetilde{a_{13,13}} \end{pmatrix}$$

$$\text{and } \widetilde{b} = \begin{bmatrix} \widetilde{b_1} \\ \widetilde{b_5} \\ \widetilde{b_9} \\ \widetilde{b_{13}} \end{bmatrix}$$

to solve some entries of x which can be denoted as $\widetilde{x}$

$$\widetilde{x} = \begin{bmatrix} x_1 \\ x_5 \\ x_9 \\ x_1 3 \end{bmatrix}$$

So I only need to solve the equation $\widetilde{A}\widetilde{x} = \widetilde{b}$:

$$\begin{pmatrix} \widetilde{a_{11}} & \widetilde{a_{15}} & 0 & 0 \\ \widetilde{a_{51}} & \widetilde{a_{55}} & \widetilde{a_{59}} & 0 \\ 0 & \widetilde{a_{95}} & \widetilde{a_{99}} & \widetilde{a_{9,13}} \\ 0 & 0 & \widetilde{a_{13,9}} & \widetilde{a_{13,13}} \end{pmatrix} \begin{bmatrix} x_1 \\ x_5 \\ x_9 \\ x_1 3 \end{bmatrix} = \begin{bmatrix} \widetilde{b_1} \\ \widetilde{b_5} \\ \widetilde{b_9} \\ \widetilde{b_{13}} \end{bmatrix}$$

Note: $a_{i,j}$ here represents the entries of $\widetilde{A}$, and $\widetilde{A}$ is the matrix A after full row elimination. $b_i$ here represents the entries of $\widetilde{b}$, and $\widetilde{b}$ is the vector b after full row elimination because b need to do the row elimination synchronously with A.

Step3:

Calculate the other entries of x by thee middle $3 \times 3$ part of each $5 \times 5$ block in A.

For each block, we need to recreate a matrix $A_{middle}$ and a vector $b_{middle}$ to solve the $x_{middle}$. In the condition of n = 3, take the first block for example:

$$A_{middle_1} = \begin{pmatrix} \widetilde{a_{22}} & 0 & 0 \\ 0 & \widetilde{a_{33}} & 0 \\ 0 & 0 & \widetilde{a_{44}} \end{pmatrix}$$

$$b_{middle_1} = \begin{bmatrix} \widetilde{b_2} - d_2 \\ \widetilde{b_3} - d_4 \\ \widetilde{b_4} - d_4 \end{bmatrix}$$

Then we can caluculate $x_{middle_1} = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}$ by the formula $A_{middle_1} x_{middle_1} = b_{middle_1}$:

$$\begin{pmatrix} \widetilde{a_{22}} & 0 & 0 \\ 0 & \widetilde{a_{33}} & 0 \\ 0 & 0 & \widetilde{a_{44}} \end{pmatrix} \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \widetilde{b_2} - d_2 \\ \widetilde{b_3} - d_4 \\ \widetilde{b_4} - d_4 \end{bmatrix}$$

Note:
1.$a_{i,j}$ here represents the entries of $\widetilde{A}$, and $\widetilde{A}$ is the matrix A after full row elimination. $b_i$ here represents the entries of $\widetilde{b}$, and $\widetilde{b}$ is the vector b after full row elimination because b need to do the row elimination synchronously with A.
2.$d_i = \widetilde{a_{i,i-k}} x_{i-k} + \widetilde{a_{i,i-k+4}} x_{i-k+4}$ and here $k = imod4 - 1$. To explain why we need $d_i$ here, take an example, in the first block of $\widetilde{A}$, except the first and fifth row the remaining part of $\widetilde{A}$ can be shown as

$$\begin{bmatrix} \widetilde{a_{21}} & \widetilde{a_{22}} & \widetilde{a_{23}} & \widetilde{a_{24}} & \widetilde{a_{25}} \\ \widetilde{a_{31}} & \widetilde{a_{32}} & \widetilde{a_{33}} & \widetilde{a_{34}} & \widetilde{a_{35}} \\ \widetilde{a_{41}} & \widetilde{a_{42}} & \widetilde{a_{43}} & \widetilde{a_{44}} & \widetilde{a_{45}} \end{bmatrix}$$

So I can get that

$$\begin{aligned} \widetilde{a_{22}} x_2 + \widetilde{a_{23}} x_3 + \widetilde{a_{24}} x_4 &= b_2 - (\widetilde{a_{21}} x_1 + \widetilde{a_{25}} x_5) \\ \widetilde{a_{32}} x_2 + \widetilde{a_{33}} x_3 + \widetilde{a_{34}} x_4 &= b_3 - (\widetilde{a_{31}} x_1 + \widetilde{a_{35}} x_5) \\ \widetilde{a_{42}} x_2 + \widetilde{a_{43}} x_3 + \widetilde{a_{44}} x_4 &= b_4 - (\widetilde{a_{41}} x_1 + \widetilde{a_{45}} x_5) \end{aligned} \tag{1}$$

I want to delete the extra term like $\widetilde{a_{21}} x_1 + \widetilde{a_{25}} x_5$ so after generalized, I create a new symbol d to represent that in $d_i = \widetilde{a_{i,i-k}} x_{i-k} + \widetilde{a_{i,i-k+4}} x_{i-k+4}$ form.
So in the end of this algorithm, I only need to combine all x that I calculated together to return the complete A by python scripts.
This method can be generalized to all arbitrary n, because this algorithm in each block are same, when I generalized this I only need to repeat my steps that are shown above.

**Operation count:**
The total operation count in this algorithm is $\mathcal{O}(n)$.
Explain: Before I starting I will first mentioned that A is an $m \times m$ matrix, and $m = 4 * n + 1$, by the definition of A in question stem.
In the first step, I do the row elimination, in combineAb[i, index] = combineAb[i, index]/(combineAb[i, j]/combineAb[j, j]) - combineAb[j, index], I will do $(index - i) + 1$ divisions and $index - i$ subtractions for each k. We know that i can be all values from $4k$ to $4k + 4$ except when $i = j$ which means I do not do row elimination in the diagonal of matrix, and we also know about index , so $index - i \leq 9$ must happen in this question. So the operation count for each k can be write as $\mathcal{O}(1)$. And we know that k are in range(n), so we need to do n loops here, so totally, the operation count in step1 is $\mathcal{O}(n)$
In the second step, I will use the matrix and vector $\widetilde{A}, \widetilde{b}$ which are in lower dimensions to calculate

$\widetilde{x} = \begin{bmatrix} x_1 \\ x_5 \\ x_9 \\ x_{13} \\ \vdots \end{bmatrix}$. In this step the operation count for each i in each for loop are all $\mathcal{O}(1)$, because all calculation here are the number calculation. And there 3 loops individually need to do $n+1, n, n+1$, so the operation count in each loops can all be written as $\mathcal{O}(n)$, so the total operation count in step2 can be written as $\mathcal{O}(n)$.
In the third step, I need to do 6 multiplications and 6 subtractions in each i then we can see the operation count for each i is $\mathcal{O}(1)$ here. And we also know that $i \in [0, n-1]$, so the total operation count in step3 is $\mathcal{O}(n)$.
So the total operation count in this algorithm is $\mathcal{O}(n)$.

### 3.5 question e

The code are shown in cw2.CW2.py by function solve_bandtri(A, b), and the automated test are created in folder test_cw2, in test_coursework2.py, use the function testsolve_bandtri(n, epsilon), and I also use function creatA(n, epsilon) which are stored in cw2.CW2.py to create the specific A in this question.

## 4 Question5

### 4.1 question a

(in the next page)

To solve this problem, I will rewrite $S_{1,1}$, $S_{1,2}$, $S_{1,3}$..., $S_{1,n-1}$, $S_{2,1}$, $S_{2,2}$, $S_{2,3}$..., $S_{2,n-1}$... first.

$$\begin{aligned} S_{1,1} &= (\frac{b^1_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,1} + (-\frac{b^1_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{0,1} + (\frac{b^2_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,2} + (-\frac{b^2_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,0} + (\frac{4\mu}{\triangle x^2} + c)u_{1,1} \\ &= (\frac{b^1_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,1} + (-\frac{b^1_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (\frac{b^2_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,2} + (-\frac{b^2_{1,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (\frac{4\mu}{\triangle x^2} + c)u_{1,1} \end{aligned} \tag{2}$$

$$\begin{aligned} S_{1,2} &= (\frac{b^1_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,2} + (-\frac{b^1_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{0,2} + (\frac{b^2_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,3} + (-\frac{b^2_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,1} + (\frac{4\mu}{\triangle x^2} + c)u_{1,2} \\ &= (\frac{b^1_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,2} + (-\frac{b^1_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (\frac{b^2_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,3} + (-\frac{b^2_{1,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,1} + (\frac{4\mu}{\triangle x^2} + c)u_{1,2} \end{aligned} \tag{3}$$

$$\begin{aligned} S_{1,3} &= (\frac{b^1_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,3} + (-\frac{b^1_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{0,3} + (\frac{b^2_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,4} + (-\frac{b^2_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,2} + (\frac{4\mu}{\triangle x^2} + c)u_{1,3} \\ &= (\frac{b^1_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,3} + (-\frac{b^1_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (\frac{b^2_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,4} + (-\frac{b^2_{1,3}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,2} + (\frac{4\mu}{\triangle x^2} + c)u_{1,3} \end{aligned} \tag{4}$$

$$\vdots$$

$$\begin{aligned} S_{1,n-1} &= (\frac{b^1_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,n-1} + (-\frac{b^1_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{0,n-1} + (\frac{b^2_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,n} + (-\frac{b^2_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,n-2} + (\frac{4\mu}{\triangle x^2} + c)u_{1,n-1} \\ &= (\frac{b^1_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,n-1} + (-\frac{b^1_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (\frac{b^2_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (-\frac{b^2_{1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,n-2} + (\frac{4\mu}{\triangle x^2} + c)u_{1,n-1} \end{aligned} \tag{5}$$

$$\begin{aligned} S_{2,1} &= (\frac{b^1_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{3,1} + (-\frac{b^1_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,1} + (\frac{b^2_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,2} + (-\frac{b^2_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,0} + (\frac{4\mu}{\triangle x^2} + c)u_{2,1} \\ &= (\frac{b^1_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{3,1} + (-\frac{b^1_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,1} + (\frac{b^2_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,2} + (-\frac{b^2_{2,1}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (\frac{4\mu}{\triangle x^2} + c)u_{2,1} \end{aligned} \tag{6}$$

$$S_{2,2} = (\frac{b^1_{2,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{3,2} + (-\frac{b^1_{2,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,2} + (\frac{b^2_{2,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,3} + (-\frac{b^2_{2,2}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,1} + (\frac{4\mu}{\triangle x^2} + c)u_{2,2} \tag{7}$$

$$\vdots$$

$$\begin{aligned} S_{2,n-1} &= (\frac{b^1_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{3,n-1} + (-\frac{b^1_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,n-1} + (\frac{b^2_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,n} + (-\frac{b^2_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,n-2} + (\frac{4\mu}{\triangle x^2} + c)u_{2,n-1} \\ &= (\frac{b^1_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{3,n-1} + (-\frac{b^1_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{1,n-1} + (\frac{b^2_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (-\frac{b^2_{2,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{2,n-2} + (\frac{4\mu}{\triangle x^2} + c)u_{2,n-1} \end{aligned} \tag{8}$$

$$\vdots$$

$$S_{n-1,n-1} = (\frac{b^1_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{n,n-1} + (-\frac{b^1_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{n-2,n-1} + (\frac{b^2_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{n-1,n} + (-\frac{b^2_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{n-1,n-2}$$
$$+ (\frac{4\mu}{\triangle x^2} + c)u_{n-1,n-1}$$
$$= (\frac{b^1_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{n,n-1} + (-\frac{b^1_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{n-2,n-1} + (\frac{b^2_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})0 + (-\frac{b^2_{n-1,n-1}}{2\triangle x} - \frac{\mu}{\triangle x^2})u_{n-1,n-2}$$
$$+ (\frac{4\mu}{\triangle x^2} + c)u_{n-1,n-1} \tag{9}$$

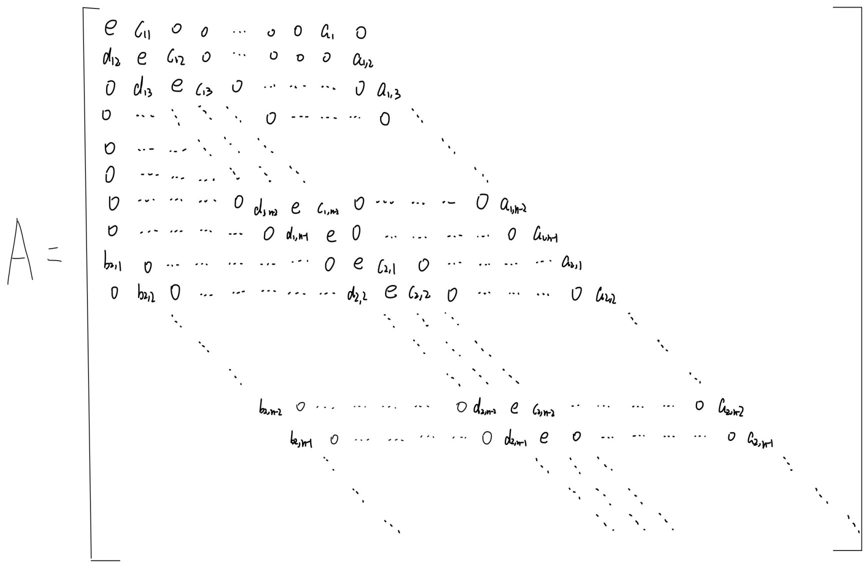
And we know that $Av = S$, where A is a parametric matrix, we can try to write down the form of A, then we can see some patterns. And the first $2n-1 \times 2n-1$ block of A can be shown below:

$$A[1:2n{-}2, 1:2n{-}2] = \begin{pmatrix}
e & c_{1,1} & 0 & \cdots & \cdots & \cdots & 0 & a_{1,1} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\
d_{1,2} & e & c_{1,2} & 0 & \cdots & \cdots & 0 & 0 & a_{1,2} & 0 & \cdots & \cdots & \cdots & 0 \\
0 & d_{1,3} & e & c_{1,3} & 0 & \cdots & 0 & 0 & 0 & a_{1,3} & 0 & \cdots & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & d_{1,n-3} & e & c_{1,n-3} & 0 & 0 & \cdots & \cdots & 0 & a_{1,n-3} & 0 & 0 \\
0 & \cdots & \cdots & 0 & d_{1,n-2} & e & c_{1,n-2} & 0 & \cdots & \cdots & \cdots & 0 & a_{1,n-2} & 0 \\
0 & \cdots & \cdots & \cdots & 0 & d_{1,n-1} & e & 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_{1,n-1} \\
b_{2,1} & 0 & \cdots & \cdots & \cdots & \cdots & 0 & e & c_{2,1} & 0 & \cdots & \cdots & \cdots & 0 \\
0 & b_{2,2} & 0 & \cdots & \cdots & \cdots & 0 & d_{2,2} & e & c_{2,2} & 0 & \cdots & \cdots & 0 \\
0 & 0 & b_{2,3} & 0 & \cdots & \cdots & 0 & 0 & d_{2,3} & e & c_{2,3} & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & b_{2,n-3} & 0 & 0 & 0 & \cdots & 0 & d_{2,n-3} & e & c_{2,n-3} & 0 \\
0 & \cdots & \cdots & \cdots & 0 & b_{2,n-2} & 0 & 0 & \cdots & \cdots & 0 & d_{2,n-2} & e & c_{2,n-2} \\
0 & \cdots & \cdots & \cdots & \cdots & 0 & b_{2,n-1} & 0 & \cdots & \cdots & \cdots & 0 & d_{2,n-1} & e
\end{pmatrix}$$

where for $0 < i, j < n$,$a_{i,j} = \frac{b^1_{i,j}}{2\triangle x} - \frac{\mu}{\triangle x^2}$, $b_{i,j} = -\frac{b^1_{i,j}}{2\triangle x} - \frac{\mu}{\triangle x^2}$, $c_{i,j} = \frac{b^2_{i,j}}{2\triangle x} - \frac{\mu}{\triangle x^2}$, $d_{i,j} = -\frac{b^2_{i,j}}{2\triangle x} - \frac{\mu}{\triangle x^2}$, $e = \frac{\mu}{\triangle x^2} + c$ and $v_{i,j} = u_{i,j}$,

$$v = \begin{bmatrix} \mathrm{v}_{1,1} \\ \mathrm{v}_{1,2} \\ \mathrm{v}_{1,3} \\ \vdots \\ \mathrm{v}_{1,n-1} \\ \mathrm{v}_{2,1} \\ \mathrm{v}_{2,2} \\ \vdots \\ \mathrm{v}_{2,n-1} \\ \vdots \\ \mathrm{v}_{n-1,n-1} \end{bmatrix}$$

To show more clearly, the matrix are drawn below(the empty space are all 0):



Bandwidth: the upper and lower bandwidth of A are both $n-1$.
Expected Operation Count: By notes I can find that the operation count of doing LU decomposition with a banded matrix M is $\mathcal{O}(mpq)$, where p is the upper bandwidth of M, q is the lower bandwidth of M, and m is shape of M, so in this question, the upper and lower bandwidth of A are both $n-1$, and $A \in \mathcal{C}^{(n-1)^2 \times (n-1)^2}$, then $m = (n-1)^2$. By this I can find that the expected Operation Count to compute L and U are $\mathcal{O}((n-1)^2 \times (n-1)(n-1))$, so in this case, the expected operation count is $\mathcal{O}(n^4)$.

## 4.2 question b

The code are shown in cw2.CW2.py by function $CWLU\_inplace5(A)$.

**Operation count:**
The total operation count in this algorithm is $\mathcal{O}(n^4)$.
In this algorithm there are $(n-1)^2$ loops by "for k in range((n - 1)**2 - 1)", besides in each loop there are $num - k = k + n - 1 - k = n - 1$ divisions in "A[k + 1:num + 1, k] = A[k + 1:num + 1, k] / A[k ,k]" function, $(num - k)^2 = (n-1)^2$ multiplications and $(num - k)^2 = (n-1)^2$ subtractions in function "A[k + 1:num + 1, k + 1:num + 1] = A[k + 1:num + 1, k + 1:num + 1] - np.outer(A[k + 1:num + 1, k], A[k, k + 1:num + 1])". So in each loop there are $2(n-1)^2 + (n-1)$ operations, so I can denote the operation count in each loop by $\mathcal{O}$ as $\mathcal{O}(n^2)$. And I have already showed above that there are totally $(n-1)^2$ loops, so the total operation count in this algorithm denoted by $\mathcal{O}$ is $\mathcal{O}(n^4)$.

## 4.3 question c

Rewrite equation(6):

$$(\frac{b^1_{i,j}}{2\triangle x}-\frac{\mu}{\triangle x^2})u^{k+\frac{1}{2}}_{i+1,j}+(-\frac{b^1_{i,j}}{2\triangle x}-\frac{\mu}{\triangle x^2})u^{k+\frac{1}{2}}_{i-1,j}+(\frac{4\mu}{\triangle x^2}+c)u^{k+\frac{1}{2}}_{i,j} = S_{i,j}+(-\frac{b^2_{i,j}}{2\triangle x}+\frac{\mu}{\triangle x^2})u^k_{i,j+1}+(\frac{b^2_{i,j}}{2\triangle x}+\frac{\mu}{\triangle x^2})u^k_{i,j-1}$$

Then I can set $a_{i,j} = \frac{b^1_{i,j}}{2\triangle x} - \frac{\mu}{\triangle x^2}$, $f_{i,j} = -\frac{b^1_{i,j}}{2\triangle x} - \frac{\mu}{\triangle x^2}$, $e = \frac{4\mu}{\triangle x^2} + c$, $c_{i,j} = -\frac{b^2_{i,j}}{2\triangle x} + \frac{\mu}{\triangle x^2}$, and $d_{i,j} = \frac{b^2_{i,j}}{2\triangle x} + \frac{\mu}{\triangle x^2}$.

Rewrite equation(6) by using equation(5): $v_{(n-1)(i-1)+j} = u_{i,j}$, $0 < i,j < n$ for $\mathbf{v} \in \mathcal{R}^{(n-1)^2}$:

$$a_{i,j}v^{k+\frac{1}{2}}_{(n-1)i+j} + f_{i,j}v^{k+\frac{1}{2}}_{(n-1)(i-2)+j} + ev^{k+\frac{1}{2}}_{(n-1)(i-1)+j} = S_{i,j} + c_{i,j}v^k_{(n-1)(i-1)+j+1} + d_{i,j}v^k_{(n-1)(i-1)+j-1}$$

Do in the similar way in part(a) of Q5, I can create a matrix-vector equation for v in $Av_1 = S+Bv_2$ form. Then take n = 4 for example:

$$\begin{bmatrix} e & 0 & 0 & a_{1,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & e & 0 & 0 & a_{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & e & 0 & 0 & a_{1,3} & 0 & 0 & 0 \\ f_{2,1} & 0 & 0 & e & 0 & 0 & a_{2,1} & 0 & 0 \\ 0 & f_{2,2} & 0 & 0 & e & 0 & 0 & a_{2,2} & 0 \\ 0 & 0 & f_{2,3} & 0 & 0 & e & 0 & 0 & a_{2,3} \\ 0 & 0 & 0 & f_{3,1} & 0 & 0 & e & 0 & 0 \\ 0 & 0 & 0 & 0 & f_{3,2} & 0 & 0 & e & 0 \\ 0 & 0 & 0 & 0 & 0 & f_{3,3} & 0 & 0 & e \end{bmatrix} \begin{bmatrix} v_1^{k+\frac{1}{2}} \\ v_2^{k+\frac{1}{2}} \\ v_3^{k+\frac{1}{2}} \\ v_4^{k+\frac{1}{2}} \\ v_5^{k+\frac{1}{2}} \\ v_6^{k+\frac{1}{2}} \\ v_7^{k+\frac{1}{2}} \\ v_8^{k+\frac{1}{2}} \\ v_9^{k+\frac{1}{2}} \end{bmatrix}$$
$$= \begin{bmatrix} S_{1,1} \\ S_{1,2} \\ S_{1,3} \\ S_{2,1} \\ S_{2,2} \\ S_{2,3} \\ S_{3,1} \\ S_{3,2} \\ S_{3.3} \end{bmatrix} + \begin{bmatrix} 0 & c_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ d_{1,2} & 0 & c_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{2,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_{2,2} & 0 & c_{2,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_{2,3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{3,1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d_{3,2} & 0 & c_{3,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{3,3} & 0 \end{bmatrix} \begin{bmatrix} v_1^k \\ v_2^k \\ v_3^k \\ v_4^k \\ v_5^k \\ v_6^k \\ v_7^k \\ v_8^k \\ v_9^k \end{bmatrix} \tag{10}$$

Then rewrite equation(7):

$$(\frac{b^2_{i,j}}{2\triangle x}-\frac{\mu}{\triangle x^2})u^{k+1}_{i,j+1}+(-\frac{b^2_{i,j}}{2\triangle x}-\frac{\mu}{\triangle x^2})u^{k+1}_{i,j-1}+(\frac{4\mu}{\triangle x^2}+c)u^{k+1}_{i,j} = S_{i,j}+(-\frac{b^1_{i,j}}{2\triangle x}+\frac{\mu}{\triangle x^2})u^{k+\frac{1}{2}}_{i+1,j}+(\frac{b^1_{i,j}}{2\triangle x}+\frac{\mu}{\triangle x^2})u^{k+\frac{1}{2}}_{i-1,j}$$

Then I can set the same $a_{i,j}, f_{i,j}, e, c_{i,j}, d_{i,j}$ that are defined above in when I rewrite equation(6). Then review equation(7) by using equation(5): $v_{(n-1)(i-1)+j} = u_{i,j}$, $0 < i,j < n$ for $\mathbf{v} \in \mathcal{R}^{(n-1)^2}$:

$$-c_{i,j}v^{k+1}_{(n-1)(i-1)+j+1} - d_{i,j}v^{k+1}_{(n-1)(i-1)+j-1} + ev^{k+1}_{(n-1)(i-1)+j} = S_{i,j} - a_{i,j}v^{k+\frac{1}{2}}_{(n-1)i+j} - f_{i,j}v^{k+\frac{1}{2}}_{(n-1)(i-2)+j}$$

Do in the similar way with the above part of solving equation(6), I can create a matrix-vector equation for v in $Cv_3 = S + Dv_1$. Take n = 4 for example:

$$
\begin{bmatrix}
e & -c_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-d_{1,2} & e & -c_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -d_{1,3} & e & -c_{1,3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -d_{2,1} & e & -c_{2,1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -d_{2,2} & e & -c_{2,2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -d_{2,3} & e & -c_{2,3} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -d_{3,1} & e & -c_{3,1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -d_{3,2} & e & -c_{3,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_{3,3} & e
\end{bmatrix}
\begin{bmatrix}
v_1^{k+1} \\ v_2^{k+1} \\ v_3^{k+1} \\ v_4^{k+\frac{1}{2}} \\ v_5^{k+\frac{1}{2}} \\ v_6^{k+\frac{1}{2}} \\ v_7^{k+\frac{1}{2}} \\ v_8^{k+\frac{1}{2}} \\ v_9^{k+\frac{1}{2}}
\end{bmatrix}
$$

$$
= \begin{bmatrix}
S_{1,1} \\ S_{1,2} \\ S_{1,3} \\ S_{2,1} \\ S_{2,2} \\ S_{2,3} \\ S_{3,1} \\ S_{3,2} \\ S_{3.3}
\end{bmatrix}
+
\begin{bmatrix}
0 & 0 & 0 & -a_{1,1} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -a_{1,2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -a_{1,3} & 0 & 0 & 0 \\
-f_{2,1} & 0 & 0 & 0 & 0 & 0 & -a_{2,1} & 0 & 0 \\
0 & -f_{2,2} & 0 & 0 & 0 & 0 & 0 & -a_{2,2} & 0 \\
0 & 0 & -f_{2,3} & 0 & 0 & 0 & 0 & 0 & -a_{2,3} \\
0 & 0 & 0 & -f_{3,1} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -f_{3,2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -f_{3,3} & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
v_1^{k+\frac{1}{2}} \\ v_2^{k+\frac{1}{2}} \\ v_3^{k+\frac{1}{2}} \\ v_4^{k+\frac{1}{2}} \\ v_5^{k+\frac{1}{2}} \\ v_6^{k+\frac{1}{2}} \\ v_7^{k+\frac{1}{2}} \\ v_8^{k+\frac{1}{2}} \\ v_9^{k+\frac{1}{2}}
\end{bmatrix}
\tag{11}
$$

**Solution:**
For given $u_{i,j}^k$, I will first transform it to $v_{(n-1)(i-1)+j}^k$ and $v_{(n-1)(i-1)+j}^{k+\frac{1}{3}}$ forms, this can help me solve this question by the above matrix-vector equation to find out the v. For example, if I want to compute$Av_1 = S + Bv_2$, then I will compute $S + Bv_2$ first, and I can solve $v_1$ by $Ax = b$.
Then in the similar method, $v^{k+1}$ can also be found after computing $v^{k+\frac{1}{2}}$.
**Operation Count:**
The operation count in this algorithm is $\mathcal{O}(n^2)$, because there are $\mathcal{O}(1)$ operation count for each loop, and there are $(n-1)^2$ loops, because the size of A and S are are $((n-1)^2, (n-1)^2)$, so the total operation count in this algorithm is $\mathcal{O}(n^2)$.

## 4.4 question d

## 4.5 question e