TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

BÁO CÁO MÔN GIẢI TÍCH SỐ

Đề tài: *Phương pháp phân tích LU và Cholesky*

Giảng viên hướng dẫn: TS. Hà Thị Ngọc Yến

Sinh viên thực hiện: Nguyễn Đức Quân

MSSV: 20200505

Lớp: CTTN Toán – Tin K65

Hà Nội, 1/2022

Lời nói đầu

Đối với sinh viên ngành *Toán ứng dụng và Tin học*, việc trang bị kiến thức về môn *Giai tích số* là thực sự cần thiết. Từ đó, sinh viên có khả năng tiếp thu các môn khoa học khác phục vụ cho đào tạo ngành, đồng thời hiểu được mối quan hệ giữa toán học và thực tiễn kỹ thuật khi giải quyết công việc.

Sau quá trình học tập, nghiên cứu kết hợp với những kiến thức nền tãng sẵn có, dưới sự hướng dẫn của giảng viên bộ môn – cô Hà Thị Ngọc Yến, em đã có bài báo cáo tìm hiểu về hai phương pháp phân tích ma trận, đó là phân tích LU và phân tích Cholesky. Đồng thời, áp dụng phân tích Cholesky trong việc tìm nghiệm đúng của hệ phương trình đại số tuyến tính $A.X = B$.

Trong báo cáo này, em sẽ trình bày 3 vấn đề chính, đó là:

- Phân tích LU.
- Phân tích Cholesky.
- Ứng dụng phân tích Cholesky để giải phương trình $A.X = B$.

Trong mỗi phần, em sẽ trình bày lý thuyết của phương pháp, thuật toán và code để giải quyết cũng như đưa ra hệ thống ví dụ để minh họa về các phương pháp trên.

Mục lục

I. Phân tích LU

1. Cơ sở lý thuyết

- Phân tích LU là cách biểu diễn một ma trận vuông A thành tích của hai ma trận, một ma trận tam giác dưới và một ma trận tam giác trên:

$$A = L.U$$

trong đó, L là một ma trận tam giác dưới

U là một ma trận tam giác trên

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{bmatrix} . \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix} = L.U$$

- Từ phương trình $A = L.U$ như trên ta được hệ phương trình bao gồm n^2 phương trình và $n^2 + n$ ẩn ($l_{ij}(i \geq j), u_{ij}(i \leq j)$).

$\Rightarrow$ Có nhiều cách phân tích LU khác nhau mà ta không kiểm soát được hết.

2. <u>Thuật toán Doolittle</u>

- Chọn $l_{11} = l_{22} = l_{33} = \cdots = l_{nn} = 1$.
- Từ đó, ta có phân tích:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix} . \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix} = L.U$$

- Ta được hệ phương trình bao gồm n^2 phương trình và n^2 ẩn.
  - Bước 1: Nhân lần lượt hàng 1 của ma trận L với từng cột của ma trận U.

    $h_{1L}.c_{kU} = a_{1k}, k = \overline{1,n}$ (n phương trình), ta được:

    $$u_{11} = a_{11}$$
    $$u_{12} = a_{12}$$
    $$u_{13} = a_{13}$$
    $$\vdots$$
    $$u_{1n} = a_{1n}$$
    $$\Rightarrow u_{1k} = a_{1k}, k = \overline{1,n}$$

  - Bước 2: Nhân lần lượt hàng 2 của ma trận L với từng cột của ma trận U.

    $h_{2L}.c_{kU} = a_{2k}, k = \overline{1,n}$ (n phương trình), ta được:

    $$k = 1 \Rightarrow l_{21}.u_{11} = a_{21} \Rightarrow l_{21} = \frac{a_{21}}{u_{11}} (ĐK : u_{11} \neq 0)$$
    $$k = 2 \Rightarrow l_{21}.u_{12} + u_{22} = a_{22} \Rightarrow u_{22} = a_{22} - l_{21}.u_{12}$$
    $$k = 3 \Rightarrow l_{21}.u_{13} + u_{23} = a_{23} \Rightarrow u_{23} = a_{23} - l_{21}.u_{13}$$
    $$\vdots$$
    $$k = n \Rightarrow l_{21}.u_{1n} + u_{2n} = a_{2n} \Rightarrow u_{2n} = a_{2n} - l_{21}.u_{1n}$$
    $$\Rightarrow u_{2k} = a_{2k} - l_{21}.u_{1k}, (k = \overline{2,n})$$

- Bước 3: Nhân lần lượt hàng 3 của ma trận *L* với từng cột của ma trận *U*.

  $h_{3L}.c_{kU} = a_{3k}, k = \overline{1,n}$ (n phương trình), ta được:

  $k = 1 \Rightarrow l_{31}.u_{11} = a_{31} \Rightarrow l_{31} = \dfrac{a_{31}}{u_{11}} (ĐK: u_{11} \neq 0)$

  $k = 2 \Rightarrow l_{31}.u_{12} + l_{32}.u_{22} = a_{32} \Rightarrow l_{32} = \dfrac{a_{32} - l_{31}.u_{12}}{u_{22}} (ĐK: u_{22} \neq 0)$

  $k = 3 \Rightarrow l_{31}.u_{13} + l_{32}.u_{23} + u_{33} = a_{33} \Rightarrow u_{33} = a_{33} - (l_{31}.u_{13} + l_{32}.u_{23})$

  $\vdots$

  $k = n \Rightarrow l_{31}.u_{1n} + l_{32}.u_{2n} + u_{3n} = a_{3n} \Rightarrow u_{3n} = a_{3n} - (l_{31}.u_{1n} + l_{32}.u_{2n})$

  $\Rightarrow u_{3k} = a_{3k} - (l_{31}.u_{1k} + l_{32}.u_{2k})$

Cứ tiếp tục làm như vậy cho đến khi nhân lần lượt hàng n của ma trận *L* với từng cột của ma trận *U*.

Ta được công thức tổng quát của thuật toán Doolittle như sau:

$$u_{1j} = a_{1j}, (j = \overline{1,n})$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}.u_{kj}, (i = \overline{2,j}; j = \overline{1,n})$$

$$l_{i1} = \frac{a_{i1}}{u_{11}}, (i = \overline{2,n})$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}.u_{kj}}{u_{jj}}, (j = \overline{2,i-1}; i = \overline{1,n})$$

- Từ đó, ta nhận thấy để có phân tích LU thì các phần tử trên đường chéo chính của ma trận *U* phải khác 0 $(u_{jj} \neq 0, i = \overline{1,n})$, tức là: $\det U \neq 0$.

  Mà $\det A = \det L . \det U$

  $\Rightarrow \det A \neq 0$
- Do đó, điều kiện để có phân tích LU là định thức của ma trận *A* phải khác 0.

3. <u>Ví dụ minh họa</u>

Phân tích LU các ma trận sau:

a) $A = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$

+) Kiểm tra định thức của ma trận *A*: $\det A = 24 \neq 0$.

Do đó, *A* có phân tích LU.

$u_{11} = a_{11} = 2$

$u_{12} = a_{12} = -1$

$u_{13} = a_{13} = -2$

$l_{21}.u_{11} = a_{21} \Rightarrow l_{21} = \dfrac{a_{21}}{u_{11}} = -2$

$l_{21}.u_{12} + u_{22} = a_{22} \Rightarrow u_{22} = a_{23} - l_{21}.u_{12} = 4$

$l_{21}.u_{13} + u_{23} = a_{23} \Rightarrow u_{23} = a_{23} - l_{21}.u_{13} = -1$

$l_{31}.u_{11} = a_{31} \Rightarrow l_{31} = \dfrac{a_{31}}{u_{11}} = -2$

$l_{31}.u_{12} + l_{32}.u_{22} = a_{32} \Rightarrow l_{32} = \dfrac{a_{32} - l_{31}.u_{12}}{u_{22}} = -1$

$l_{31}.u_{13} + l_{32}.u_{23} + u_{33} = a_{32} \Rightarrow u_{33} = a_{33} - (l_{31}.u_{31} + l_{32}.u_{32}) = 3$

Vậy ta có phân tích LU của ma trận *A* là:

$$\begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -2 & -1 & 1 \end{bmatrix} . \begin{bmatrix} 2 & -1 & -2 \\ 0 & 4 & -1 \\ 0 & 0 & 3 \end{bmatrix}$$

b) $B = \begin{bmatrix} 2 & -3 & -5 \\ -4 & 6 & 10 \\ 1 & 7 & 5 \end{bmatrix}$

B không có phân tích LU vì $\det B = 0$

II. Phân tích Cholesky

1. Cơ sở lý thuyết

- Phân tích Cholesky là một trường hợp đạc biệt của phân tích LU, ma trận vuông *A* cũng được biểu thành tích của một ma trận tam giác dưới và một ma trận tam giác trên, trong đó ma trận tam giác trên là ma trận chuyển vị của ma trận tam giác dưới.

$$A = L.L^T$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{bmatrix} . \begin{bmatrix} l_{11} & l_{21} & l_{31} & \dots & l_{n1} \\ 0 & l_{22} & l_{32} & \dots & l_{n2} \\ 0 & 0 & l_{33} & \dots & l_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & l_{nn} \end{bmatrix} = L.L^T$$

- Ta có: $A = L.L^T \Rightarrow A^T = (L.L^T)^T = L.L^T = A$

  Điều kiện cần: A là một ma trận vuông đối xứng. (1)

- Mặt khác, $x.A.x^T = x.L.L^T.x^T = \|x.L\|^2 \geq 0$

  Nên *A* là ma trận xác định không âm.

  Đồng thời, *A* là ma trận không suy biến (theo điều kiện của phân tích LU).

  Do đó, *A* là ma trận xác định dương. (2)

- Từ (1) và (2), suy ra điều kiện để có phân tích Cholesky là:

  Ma trận vuông *A* đối xứng và xác định dương.

2. <u>Mô tả thuật toán</u>

- Bước 1: Nhân lần lượt hàng 1 của ma trận L với từng cột của ma trận L^T.

$$l_{11}.l_{11} = a_{11} \Rightarrow l_{11} = \sqrt{a_{11}}$$

$$l_{11}.l_{21} = a_{12} \Rightarrow l_{21} = \frac{a_{12}}{l_{11}}$$

$$l_{11}.l_{31} = a_{13} \Rightarrow l_{31} = \frac{a_{13}}{l_{11}}$$

$$\vdots$$

$$l_{11}.l_{n1} = a_{1n} \Rightarrow l_{n1} = \frac{a_{1n}}{l_{11}}$$

$$\Rightarrow l_{k1} = \frac{a_{1k}}{l_{11}}, (k = \overline{2,n})$$

- Bước 2: Nhân lần lượt hàng 2 của ma trận L với từng cột của ma trận L^T.

$$l_{21}.l_{21} + l_{22}.l_{22} = a_{22} \Rightarrow l_{22} = \sqrt{a_{22} - l_{21}{}^2}$$

$$l_{21}.l_{31} + l_{22}.l_{32} = a_{23} \Rightarrow l_{32} = \frac{a_{23} - l_{21}.l_{31}}{l_{22}}$$

$$l_{21}.l_{41} + l_{22}.l_{42} = a_{24} \Rightarrow l_{42} = \frac{a_{24} - l_{21}.l_{41}}{l_{22}}$$

$$\vdots$$

$$l_{21}.l_{n1} + l_{22}.\mathrm{l}_{n2} = a_{2n} \Rightarrow l_{n2} = \frac{a_{2n} - l_{21}.l_{n1}}{l_{22}}$$

$$\Rightarrow l_{k2} = \frac{a_{2k} - l_{21}.l_{k1}}{l_{22}}, (k = \overline{2,n})$$

- Bước 3: Nhân lần lượt hàng 3 của ma trận L với từng cột của ma trận L^T.

$$l_{31}.l_{31} + l_{32}.l_{32} + l_{33}.l_{33} = a_{33} \Rightarrow l_{33} = \sqrt{a_{33} - (l_{31}{}^2 + l_{32}{}^2)}$$

$$l_{31}.l_{41} + l_{32}.l_{42} + l_{33}.l_{43} = a_{34} \Rightarrow l_{43} = \frac{a_{34} - (l_{31}.l_{41} + l_{32}.l_{42})}{l_{33}}$$

$$\vdots$$

$$l_{31}.l_{n1} + l_{32}.l_{n2} + l_{33}.l_{n3} = a_{3n} \Rightarrow l_{n3} = \frac{a_{3n} - (l_{31}.l_{n1} + l_{32}.l_{n2})}{l_{nn}}$$

Cứ tiếp tục làm như vậy cho đến khi nhân lần lượt hàng n của ma trận L với từng cột của ma trận L^T.

Tổng quát:

$$l_{11} = \sqrt{a_{11}}$$

$$l_{i1} = \frac{a_{1i}}{l_{11}}, (i = \overline{2,n})$$

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}{}^2}, (j = \overline{2,n})$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}.l_{jk}}{l_{jj}}, (i > j)$$

$$l_{ij} = 0, (i < j)$$

3. <u>Ví dụ minh họa</u>

Phân tích Cholesky ma trận sau:

$$A = \begin{bmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{bmatrix}$$

Code giải:

```
import math

# Hàm phân tích Cholesky
def Cholesky_Decomposition(matrix, n):

    lower = [[0 for x in range(n + 1)]
                for y in range(n + 1)];

    # Phân tách thành ma trận tam giác dưới
    for i in range(n):
        for j in range(i + 1):
            sum1 = 0;

            # Tổng đường chéo chính
            if (j == i):
                for k in range(j):
                    sum1 += pow(lower[j][k], 2);
                lower[j][j] = float(math.sqrt(matrix[j][j] - sum1));
            else:

                # Tính L(i, j) qua L(j, j)
                for k in range(j):
                    sum1 += (lower[i][k] * lower[j][k]);
                if(lower[j][j] > 0):
                    lower[i][j] = float((matrix[i][j] - sum1) / lower[j][j]);

    # Hiển thị ma trận tam giác dưới và ma trận chuyển vị của nó
    print("Lower Triangular\t\tTranspose");
    for i in range(n):

        # Ma trận tam giác dưới
        for j in range(n):
            print(lower[i][j], end = "\t");
        print("", end = "\t");

        # Ma trận chuyển vị
        for j in range(n):
            print(lower[j][i], end = "\t");
        print("");

# Gọi hàm
n = 3;
matrix = [[4, 12, -16],
          [12, 37, -43],
          [-16, -43, 98]]
Cholesky_Decomposition(matrix, n);
```

Kết quả:

```
Lower Triangular                Transpose
2.0     0       0               2.0     6.0     -8.0
6.0     1.0     0               0       1.0     5.0
-8.0    5.0     3.0             0       0       3.0
```

III. Ứng dụng phân tích Cholesky để giải phương trình *A.X = B*

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

1. Ý tưởng

- Trong thực tế, ma trận *A* trong hệ trên là ma trận vuông bất kỳ không đối xứng. Nhưng để áp dụng phân tích Cholesky ta cần ma trận vuông *A* phải đối xứng. Từ đó, ta hướng tới cách làm tổng quát sau:

  Từ phương trình

  $$A.X = B \qquad (1)$$

  Nhân A^T vào hai vế của phương trình (1)

  $$A^{T.}A.X = A^T.B$$
  $$\Leftrightarrow A_1.X = B_1 \quad (2)$$

  Phân tách Cholesky: $A_1 = S^T.S$

  Từ đó, (2)$\Leftrightarrow$ $S^{T.}S.X = B_1$ (3)

  Đặt $SX = Y$ (4). Ta có $S^T.Y = B_1$ (5)

  Giải (5) được *Y* sau đó giải (4) được *X* là nghiệm cần tìm.

2. Thuật toán tổng quát

- Từ ý tưởng trên, ta xây dựng thuật toán tổng quát như sau:

Input *A*, *B*.

Ouput *X*.

+) Bước 1: Kiểm tra xem *A* có phải ma trận đối xứng hay không?

* Nếu *A* đối xứng thì gán $A_1 = A$, $B_1 = B$ rồi chuyển đến bước 4.
* Nếu *A* không đối xứng thì thực hiện bước 2.

+) Bước 2: Tính $A_1 = A^T.A$.

+) Bước 3: Tính $B_1 = A^T.B$.

+) Bước 4: Tìm *S* qua phân tích Cholesky ma trận A_1:

* Nếu trả về None thì in ra “Không giải được bằng phương pháp Cholesky”, rồi kết thúc chương trình.
* Nếu trả về ma trận *S* thì thực hiện bước 5.

+) Bước 5: Giải phương trình $S^T.Y = B$ tìm *Y*.

+) Bước 6: Giải phương trình $S.X = Y$ tìm *X*.

+) Bước 7: Trả về nghiệm *X* và kết thúc chương trình.

3. <u>Thuật toán các gói</u>

a) Gói kiểm tra ma trận A có đối xứng hay không?

Input A_{nxn}
Output *True* or *False*

```
for i = 1 to n
    for j = 1 to n
        if A[i][j] ≠ A[j][i]
            return False
return True
```

```python
def isSymmetric(mat):
    n = len(mat)
    for i in range(n):
        for j in range(n):
            if mat[i][j] != mat[j][i]:
                return False
    return True
```

b) Gói nhân A^T với A (nếu A không đối xứng)

Input A
Ouput A_1

```
for i = 1 to n:
    for j = 1 to n:
        sum = 0
        for k = 1 to n:
            sum = sum + (A[k][i] * A[k][j])
    A1[i][j] = sum
return A1
```

```python
def transposition_mult(mat):
    n = len(mat)
    ans = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            ans[i][j] = sum([mat[k][i] * mat[k][j] for k in range(n)])
    return ans
```

c) Gói nhân A^T với B (nếu A không đối xứng)

Input A_{nxn}, B_{nx1}
Ouput B_1

for i = 1 to n:
    sum = 0
    for j = 1 to n:
        sum = sum + (A[j][i] * B[j][1])
    B_1[i][1] = sum
return B_1

```
def mult_tmatrix(M, N):
    n = len(M)
    ans = [[0] for _ in range(n)]
    for i in range(n):
        ans[i][0] = sum([M[j][i] * N[j][0] for j in range(n)])
    return ans
```

d) Gói tìm ma trận S theo phân tích Cholesky (*)

Input: Ma trận đối xứng A_1

Output: Ma trận S hoặc giá trị 'None' nếu không phân tích được

Bước 1: Tạo ma trận $S_{nxn} = 0$

Bước 2: Gán $i = 1$

Bước 3: Kiểm tra i ≤ n nếu đúng chuyển sang bước 4, ngược lại chuyển sang bước 13.

Bước 4: Gán $j = 1$

Bước 5: Kiểm tra $j \leq i + 1$ nếu đúng chuyển sang bước 6, ngược lại chuyển sang bước 10.

Bước 6: Gán $Sum = \sum_{k=1}^{j} S_{ki} S_{kj}$

Bước 7: Nếu $Sum = A_{1_{ii}}$ thì chuyển đến bước 10, ngược lại chuyển đến bước 8.

Bước 8: Kiểm tra i = j nếu đúng thì gán $S_{ji} = \sqrt{A_{1_{ii}} - Sum}$, ngược lại thì gán

$$S_{ji} = \frac{A_{1_{ji}} - Sum}{S_{ji}}$$

Bước 9: $j = j + 1$ và trở về bước 5.

Bước 10: Kiểm tra $S_{ii} = 0$, nếu đúng thì chuyển đến bước 12, ngược lại chuyển sang bước 11.

Bước 11: $i = i + 1$ và trở về bước 3

Bước 12: Trả về "None" và kết thúc gói

Bước 13: Trả về S và kết thúc gói

```
def Cholesky(matA1):
    n = len(matA1)
    S = [[0] * n for i in range(n)]
    for i in range(n):
        for j in range(i + 1):
            Sum = sum(S[k][i] * S[k][j] for k in range(j))
            if Sum == matA1[i][i]:
                break
            if i == j:
                S[j][i] = math.sqrt(matA1[i][i] - Sum)
            else:
                S[j][i] = (matA1[j][i] - Sum) / S[j][j]
        if S[i][i] == 0:
            break
    return S
```

e) Gói giải phương trình ma trận tam giác dưới $S^T.Y = B_1$

Input S, B_1
Ouput Y_{nx1}

```
if S[n][n] = 0:
        return None
Y_nx1 = 0
for i = 1 to n:
    sum = 0
    for k = 1 to i:
        sum = sum + (S[k][i] * Y[k])
    Y[i] = (B_1[i][0] - sum) / S[i][i]
return Y
```

```python
def solve1(matS, colB):
    n = len(matS)
    Y = [0] * n
    for i in range(n):
        Y[i] = (colB[i][0] - sum(matS[k][i] * Y[k] for k in range(i))) / matS[i][i]
    return Y
```

f) Gói giải phương trình ma trận tam giác trên $S.X = Y$

Input S, Y
Ouput X

```
X_nx1 = 0
for i = 1 to n + 1:
    sum = 0
    for k = 1 to i:
        sum = sum + (S[n + 1 -  i][n + 1 – k] * X[n + 1 – k])
    X[n + 1 - i] =  (Y[n + 1 – i] – sum) / S[n + 1 – i][n + 1 – i]
return X
```

```python
def solve2(matS, colY):
    n = len(matS)
    X = [0] * n
    for i in range(1, n + 1):
        X[-i] = (colY[-i] - sum(matS[-i][-k] * X[-k] for k in range(1, i))) / matS[-i][-i]
    return X
```

4. Lập trình tính toán

```python
import math
import numpy as np

# Đọc input từ file, input là ma trận bổ sung có dạng (A|B)
def loadFromFile(pathtofile):
    mat = []
    col = []
    with open(pathtofile) as f:
        lines = f.readlines()
    for i,line in enumerate(lines):
        mat.append([float(a) for a in line.split()])
        col.append([mat[i].pop()])
    return mat,col

# In ma trận
def printMat(matrix, detail = None, precision = 15):
    """ In ma trận với độ chính xác `precision` số sau dấu phẩy """
    if matrix:
        if detail:
            print('\n' + detail)
        for row in matrix:
            print('[' +', '.join([("% .*f" % (8,x)) for x in row])+ ']' if type(row)
== list else "% .*f" % (8,row))

# Kiểm tra ma trận đối xứng
def isSymmetric(mat):
    n = len(mat)
    for i in range(n):
        for j in range(n):
            if mat[i][j] != mat[j][i]:
                return False
    return True

def transposition_mult(mat):
    """ Nhân ma trận vuông chuyển vị A_T với ma trận ban đầu A """
    n = len(mat)
    ans = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            ans[i][j] = sum([mat[k][i] * mat[k][j] for k in range(n)])
    return ans

def mult_tmatrix(M, N):
```

```
    """ Nhân ma trận kích thước n * n với ma trận kích thước n * 1 """
    n = len(M)
    ans = [[0] for _ in range(n)]
    for i in range(n):
        ans[i][0] = sum([M[j][i] * N[j][0] for j in range(n)])
    return ans

# Gói phân tích Cholesky
def Cholesky(matA1):
    n = len(matA1)
    S = [[0] * n for i in range(n)]
    for i in range(n):
        for j in range(i + 1):
            Sum = sum(S[k][i] * S[k][j] for k in range(j))
            if Sum == matA1[i][i]:
                break
            if i == j:
                S[j][i] = math.sqrt(matA1[i][i] - Sum)
            else:
                S[j][i] = (matA1[j][i] - Sum) / S[j][j]
        if S[i][i] == 0:
            break
    return S

def solve1(matS, colB):
    """ Giải phương trình ma trận tam giác dưới """
    n = len(matS)
    Y = [0] * n
    for i in range(n):
        Y[i] = (colB[i][0] - sum(matS[k][i] * Y[k] for k in range(i))) / matS[i][i]
    return Y

def solve2(matS, colY):
    """ Giải phương trình ma trận tam giác trên """
    n = len(matS)
    X = [0] * n
    for i in range(1, n + 1):
        X[-i] = (colY[-i] - sum(matS[-i][-k] * X[-k] for k in range(1, i))) / matS[-
i][-i]
    return X

def main():
    # matA, colB = inputMatrix()
    matA, colB = loadFromFile('Cholesky.txt')  # Nhập ma trận từ file 'Cholesky.txt'
    matAA = matA.copy()
```

```python
    if not isSymmetric(matA):                   # Nếu ma trận A không đối xứng thì t
        colB = mult_tmatrix(matA,colB)          # Tính A_T * B
        matA = transposition_mult(matA)         # Tính A_T * A

    printMat(matA, "A1 =", 6)             # In ra ma trận A1 và B1 để dễ theo dõi
    printMat(colB, "B1 =", 6)

    s = Cholesky(matA)                    # Phân tích Cholesky
    if s is not None:                     # Nếu phân tích được thì s sẽ không là None
        printMat(s, "Ma trận S tìm được: ")
        y   = solve1(s, colB)                   # Giải S_T * Y = B tìm Y
        x   = solve2(s, y)                      # Giải S * X = Y tìm X
        printMat(x, "Nghiệm: ")

        """ Kiểm tra lại kết quả """
        x = [[i] for i in x]
        print("Kiểm tra nhân AX")
        print(np.asmatrix(matAA) * np.asmatrix(x))

    else: # Ngược lại nếu không phân tích được thì thông báo và kết thúc chương trình
        print("Không giải được bằng phương pháp Cholesky")

main()
```

- *Hướng dẫn sử dụng code*

- Nhập ma trận bổ sung có dạng $A|B$ vào file **Cholesky.txt** trong cùng thư mục với file chương trình **Cholesky.py** rồi lưu lại.

  Ví dụ: Với hai ma trận cho trước: $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, B = \begin{pmatrix} 8 \\ 5 \\ 2 \end{pmatrix}$

  thì input nhập vào sẽ là:

      1 2 3 8
      4 5 6 5
      7 8 9 2

- Chạy file chương trình Python là **Cholesky.py** để tìm nghiệm đúng của hệ phương trình (nếu có).

5. Hệ thống ví dụ

Ví dụ 1: Ma trận đầu vào là $A = \begin{pmatrix} 7 & 4 & 3 & 1 \\ 4 & 8 & 2 & 5 \\ 3 & 2 & 7 & 4 \\ 1 & 5 & 4 & 11 \end{pmatrix}$ và là $B = \begin{pmatrix} 7 \\ 10 \\ 9 \\ 9 \end{pmatrix}$

```
A1 =
[ 7.00000000,  4.00000000,  3.00000000,  1.00000000]
[ 4.00000000,  8.00000000,  2.00000000,  5.00000000]
[ 3.00000000,  2.00000000,  7.00000000,  4.00000000]
[ 1.00000000,  5.00000000,  4.00000000,  11.00000000]

B1 =
[ 7.00000000]
[ 10.00000000]
[ 9.00000000]
[ 9.00000000]

Ma trận S tìm được:
[ 2.64575131,  1.51185789,  1.13389342,  0.37796447]
[ 0.00000000,  2.39045722,  0.11952286,  1.85260434]
[ 0.00000000,  0.00000000,  2.38746728,  1.40316059]
[ 0.00000000,  0.00000000,  0.00000000,  2.33583825]

Nghiệm:
 0.00000000
 1.00000000
 1.00000000
 0.00000000
Kiểm tra nhân AX
[[ 7.]
 [10.]
 [ 9.]
 [ 9.]]
```

Ví dụ 2: Ma trận đầu vào là $A = \begin{pmatrix} 2 & -3 & 5 \\ -4 & 6 & 10 \\ 1 & 7 & 5 \end{pmatrix}$ và là $B = \begin{pmatrix} 5 \\ 8 \\ 9 \end{pmatrix}$

```
A1 =
[ 21.00000000, -23.00000000, -25.00000000]
[-23.00000000,  94.00000000,  80.00000000]
[-25.00000000,  80.00000000,  150.00000000]

B1 =
[-13.00000000]
[ 96.00000000]
[ 150.00000000]

Ma trận S tìm được:
[ 4.58257569, -5.01901148, -5.45544726]
[ 0.00000000,  8.29515062,  6.34335047]
[ 0.00000000,  0.00000000,  8.94427191]

Nghiệm:
 1.00000000
 0.50000000
 0.90000000
Kiểm tra nhân AX
[[5.]
 [8.]
 [9.]]
```

Ví dụ 3: Ma trận đầu vào là $A = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & -1 \\ 2 & -1 & 0 \end{pmatrix}$ và là $B = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix}$

```
A1 =
[ 0.00000000,  1.00000000,  2.00000000]
[ 1.00000000,  0.00000000, -1.00000000]
[ 2.00000000, -1.00000000,  0.00000000]

B1 =
[ 3.00000000]
[ 0.00000000]
[ 1.00000000]
Không giải được bằng phương pháp Cholesky
```

- *Bình luận*

- Thông qua việc chạy chương trình bằng các ví dụ khác nhau, em thấy được chương trình của mình có một vài đặc điểm như sau:
  +) Chương trình chạy được với hệ phương trình có nghiệm duy nhất và ma trận có phân tích Cholesky. *(Ví dụ 1 và 2)*
  +) Chương trình sẽ báo lỗi nếu hệ phương trình có vô số nghiệm hoặc vô nghiệm.
  +) Đối với hệ phương trình có nghiệm nhưng ma trận không có phân tích Cholesky thì sẽ đưa ra thông báo là: **"Không giải được bằng phương pháp Cholesky".** *(Ví dụ 3)*

Tài liệu tham khảo

1. Lê Trọng Vinh (2007), *Giáo trình Giải tích số*, Nhà xuất bản Khoa học và Kỹ thuât, Hà Nội.
2. Qingkai Kong, Timmy Siauw, Alexandre M.Bayen, *Python Programming and Numerical Method*, Academic Press..
3. https://en.wikipedia.org/wiki/LU_decomposition.
4. https://www.quantstart.com/articles/Cholesky-Decomposition-in-Python-and-NumPy/.

Lời cảm ơn

Bài báo cáo của em đến đây là kết thúc, mong nhận được sự góp ý của cô để em có thể hoàn thiện hơn bản báo cáo trên.

Em xin chân thành cảm ơn sự hướng dẫn của cô đã giúp em hoàn thiện đề tài này nói riêng và môn học *Giải tích số* nói chung.