MỤC LỤC

A Lời nói đầu

Sau khi học một số Toán đại cương, với những kiến thức nền tảng và dưới sự hướng dẫn của giáo viên bộ môn, cô Hà Thị Ngọc Yến, cùng sự giúp đỡ của các thành viên lớp CTTN Toán Tin K64, chúng em tiến hành tìm hiểu về phương pháp Solepski để giải bài toán hệ phương trình tuyến tính dưới dạng ma trận AX=B. Nội dung chính của bản báo cáo bao gồm:

- Lý thuyết cơ bản
- Thuật toán
- Ví dụ
- Ứng dụng

Với khả năng có hạn về kiến thức và thời gian nên bản báo cáo không thể tránh khỏi những thiếu sót về mặt nội dung cũng như hình thức. Rất mong giáo viên bộ môn cùng bạn đọc thông cảm và góp ý để bản báo cáo được hoàn thiện hơn.

Lời cảm ơn

Chúng em xin chân thành cảm ơn cô Hà Thị Ngọc Yến đã giúp đỡ và hướng dẫn chúng em tận tình trong suốt thời gian viết bài báo cáo, tạo những tiền đề, những kiến thức để tiếp cận vấn đề, phân tích giải quyết vấn đề. Nhờ đó mà chúng em hoàn thành bài báo cáo của mình được tốt hơn.

Chúng em cũng xin cảm ơn bạn bè, anh chị đã tận tình chỉ bảo, giúp đỡ trong quá trình hoàn thành bài báo cáo, tạo cho em hiểu thêm về những kiến thức liên quan.

Mặc dù đã có nhiều cố gắng trong suốt quá trình thực hiện, song có thể còn có những mặt hạn chế, thiếu sót. Chúng em rất mong nhận được ý kiến đóng góp và sự chỉ dẫn của các thầy cô giáo và bạn bè.

Chúng em xin gửi tới mọi người lời chúc thành công trên con đường sự nghiệp của mình.

B Nội dung bài

I Giới thiệu phương pháp

Trong đại số tuyến tính, khai triển LU (LU decomposition, LU factorization) là phương pháp phân tích ma trận thành tích của một ma trận tam giác dưới và một ma trận tam giác trên. Phép phân tích này thường được dùng trong giải tích số để giải hệ phương trình tuyến tính hoặc tính định thức của ma trận. Phương pháp Solepski (hay Cholesky) là phương pháp nâng cấp của khai triển LU.

II Lý thuyết

1 Bài toán

Nhiều bài toán thực tế trong nhiều lĩnh vực khác nhau như sinh học, thương mại, hóa học, khoa học máy tính, điện, kỹ thuật và cả các khoa học xã hội cần áp dụng việc giải hệ phương trình tuyến tính rất phổ biến. Tuy nhiên ở phương pháp này, chúng ta chỉ xét đến việc hệ phương trình tuyến tính có nghiệm duy nhất hay không? Hay hệ phương trình tuyến tính có n ẩn và n phương trình.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

Trong đó a_{ij}, b_i, x_i là các ẩn (i,j = 1, 2, 3, …, n).

Hay còn có thể được viết gọn dưới dạng phương trình AX=B với:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}}_{B}$$

Giải hệ phương trình A = (7) và B = (21) hay phương trình 7x=21

- Cách 1: Giải trực tiếp theo phép chia x = 21/7 = 3
- Cách 2: Nghịch đảo 7^{-1} rồi nhân với 21 sẽ dẫn đến

$$x = 7^{-1} \times 21 = 0.142857 \times 21 = 2.999997$$

Rõ ràng cách 1 tốt hơn cách 2, thêm nữa cách 2 còn có khối lượng tính toán lớn hơn khi đi xác định nghịch đảo 7^{-1}.

Ngay cả trong lời giải tổng quát, khi ta xét hệ gồm nhiều phương trình thì việc giải nó thường là tạo ra lời giải trực tiếp mà không qua tính giá trị nghịch đảo A^{-1}. Chẳng hạn ở đây, chúng ta sẽ đi đến cách giải bằng phân tích LU và phân tích Cholesky.

2 Sự phân hủy LU (hay phân tích LU):

Gọi A là một ma trận vuông (với $\det A \neq 0$). Phân tích LU của A là cách viết A thành tích của một ma trận tam giác trên và một ma trận tam giác dưới.

A=LU

Ví dụ với ma trận A cỡ 3x3

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}}_{U}$$

Hay với ma trận A vuông cấp n:

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

Từ A=LU ta cần giải ra các nghiệm L_{ij} và U_{ij}. Ta có $n^2 + n$ ẩn với n^2 phương trình. Như vậy, để tìm được ra ma trận L và U ta cần biết trước n ẩn.

Thông thường trong trường hợp này người ta chọn $L_{ii} = 1$ ∀i, ta được hệ n^2 phương trình với n^2 ẩn.

Chúng ta có thể tìm ma trận L và U bằng phương pháp khử Gauss như ví dụ sau đây:

Đây là cách làm trong bộ môn đại số tuyến tính. Sử dụng phương pháp Gauss, khi chuyển ma trận A ban đầu về ma trận tam giác trên U, sẽ xuất hiện các hệ số tương ứng với các phần tử của ma trận L (L được gọi là ma trận chứa các nhân tử).

3 Phương pháp Solepski (hay Cholesky)

3.1 Sơ lược về phương pháp

Giống như mục 2, phương pháp Cholesky là sự khai triển LU có dạng:

$$A = S^t.S$$

Trong đó A là ma trận đối xứng và S là ma trận tam giác trên và ký hiệu S^t là ma trận chuyển vị của S.

Do ma trận đối xứng là **trường hợp lý tưởng** (Vì thực tế, ma trận đầu vào sẽ là một ma trận bất đối xứng) nên thay vì sử dụng phân tích Cholesky cho A, ta sẽ thực hiện cho $A^t.A$ đối xứng.

Ta sẽ bắt đầu giải phương trình

$$AX = B \tag{1}$$

Nhân A^t vào hai vế của phương trình

$$A^t.A.X = A^t.B$$

$$\Leftrightarrow S^t.S.X = A^t.B$$

Đặt $A^t.B = B_1$ và $S.X = Y$ (2). Khi đó

$$S^t.Y = B_1 \tag{3}$$

Giải phương trình (3) ta sẽ tìm được ma trận Y.

Sau đó, giải phương trình (2) ta sẽ tìm được ma trận X là nghiệm của phương trình (1).

3.2 Mô tả cụ thể cách làm

(1) Tìm khai triển Cholesky S của ma trận A:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}}_{A} = \begin{pmatrix} s_{11} & 0 & 0 & \dots & 0 \\ s_{21} & s_{22} & 0 & \dots & 0 \\ s_{31} & s_{32} & s_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ s_{n1} & s_{n2} & s_{n3} & \dots & s_{nn} \end{pmatrix} \begin{pmatrix} s_{11} & s_{12} & s_{13} & \dots & s_{1n} \\ 0 & s_{22} & s_{23} & \dots & s_{2n} \\ 0 & 0 & s_{33} & \dots & s_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & s_{nn} \end{pmatrix}$$

$$\Rightarrow a_{ij} = \sum_{k=1}^{n} s_{ki} s_{kj} \ (j \geq i)$$

Khi đó $S = [s_{ij}]_{n \times n}$, xác định bởi $\begin{cases} s_{ii} = \sqrt{a_{ii} - \sum\limits_{k=1}^{i-1} s_{ki}^2} & \left(\text{Với } i = 1 \text{ thì } s_{11} = 0\right) \\ s_{ij} = \dfrac{a_{ij} - \sum\limits_{k=1}^{i-1} s_{ki}.s_{kj}}{s_{ii}} & \left(\text{Với } i < j\right) \\ s_{ij} = 0 & \left(\text{Với } i > j\right) \end{cases}$

(2) Giải hệ (3) tìm Y:

$$\begin{pmatrix} s_{11} & 0 & 0 & \dots & 0 \\ s_{21} & s_{22} & 0 & \dots & 0 \\ s_{31} & s_{32} & s_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ s_{n1} & s_{n2} & s_{n3} & \dots & s_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_n \end{pmatrix} \Rightarrow b_i = \sum_{k=1}^{i} s_{ki} y_k \text{ hay } y_i = \frac{b_i - \sum\limits_{k=1}^{i-1} s_{ki} y_k}{s_{ii}}$$

(3) Giải hệ (2) tìm X:

$$\begin{pmatrix} s_{11} & s_{12} & s_{13} & \dots & s_{1n} \\ 0 & s_{22} & s_{23} & \dots & s_{2n} \\ 0 & 0 & s_{33} & \dots & s_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & s_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \\ y_n \end{pmatrix} \Rightarrow y_i = \sum_{k=i}^{n} s_{ik} x_k \text{ hay } x_i = \frac{y_i - \sum\limits_{k=i+1}^{n} s_{ik} x_k}{s_{ii}}$$

Như vậy, sau khi giải xong 3 bước trên, ta sẽ thu được ma trận X chính là nghiệm của hệ phương trình (1).

III THUẬT TOÁN

1 Sơ đồ khối mô tả thuật toán

Do phải thực hiện nhiều phép toán phức tạp nên ta chia sơ đồ thuật toán thành 6 phần:

(1) Cholesky1: Tính $A_1 = A^t.A$

(2) Cholesky2: Tính $B_1 = A^t.B$

(3) Cholesky : Tìm ma trận tam giác trên S theo khai triển Cholesky của A_1

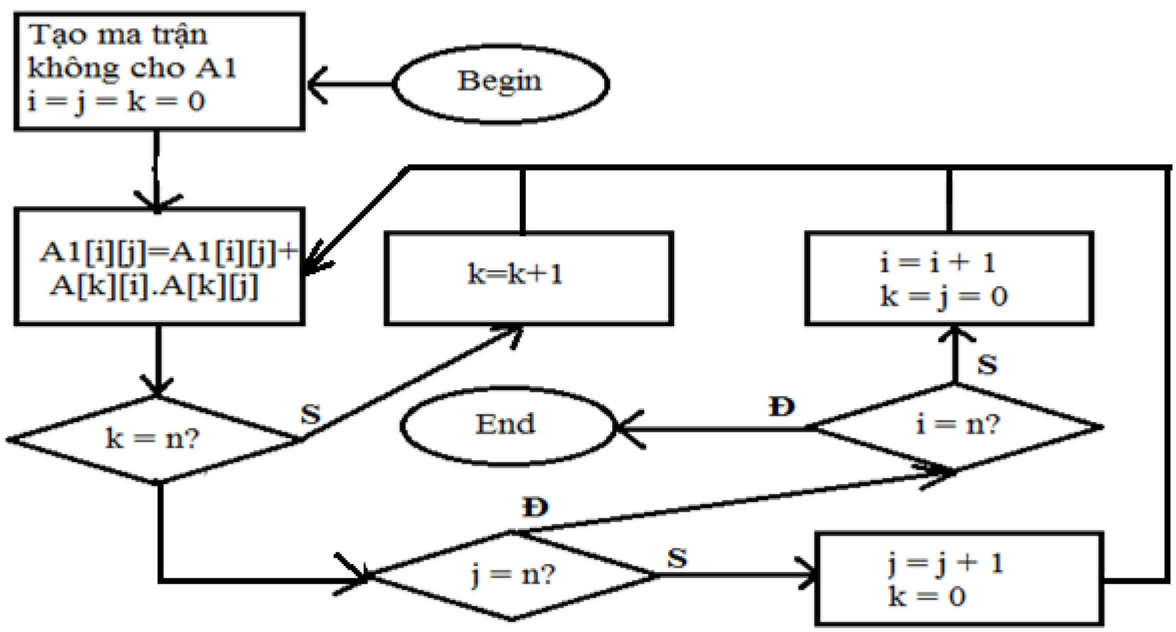
(4) Cholesky3: Giải phương trình S.Y=B để tìm Y

(5) Cholesky4: Giải phương trình $S^t.X=Y$ để tìm X

(6) Thuật toán cuối cùng.

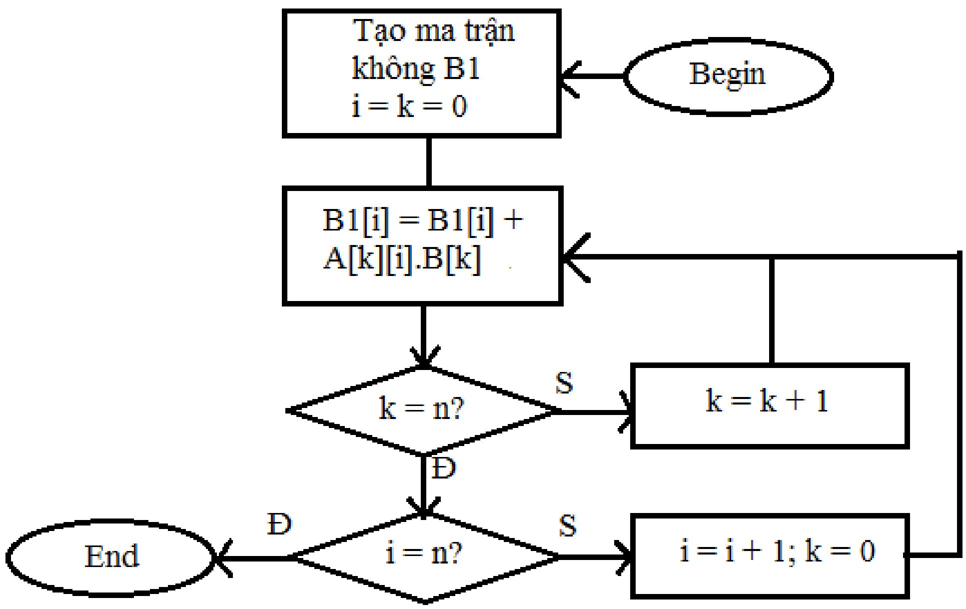
Ta sẽ đi vào từng phần nhỏ:

(1) Cholesky1:



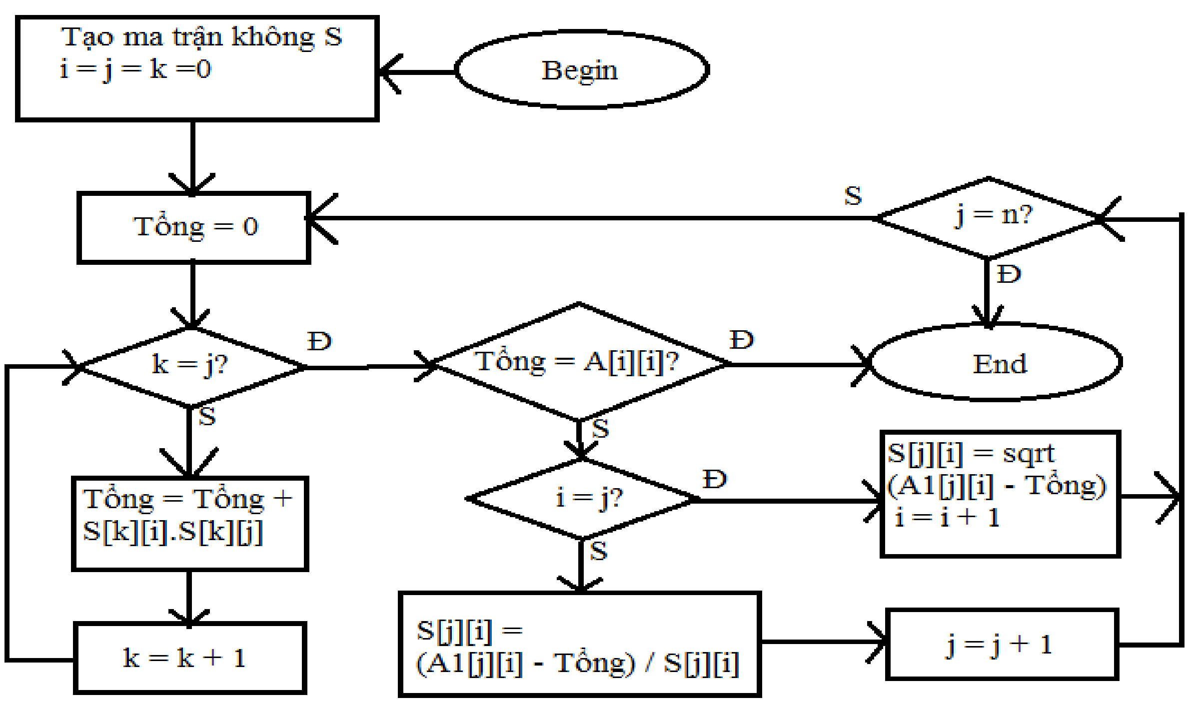
Ở bước này, ta sử dụng tối đa n^3 phép nhân và n^3 phép cộng.

(2) Cholesky2:



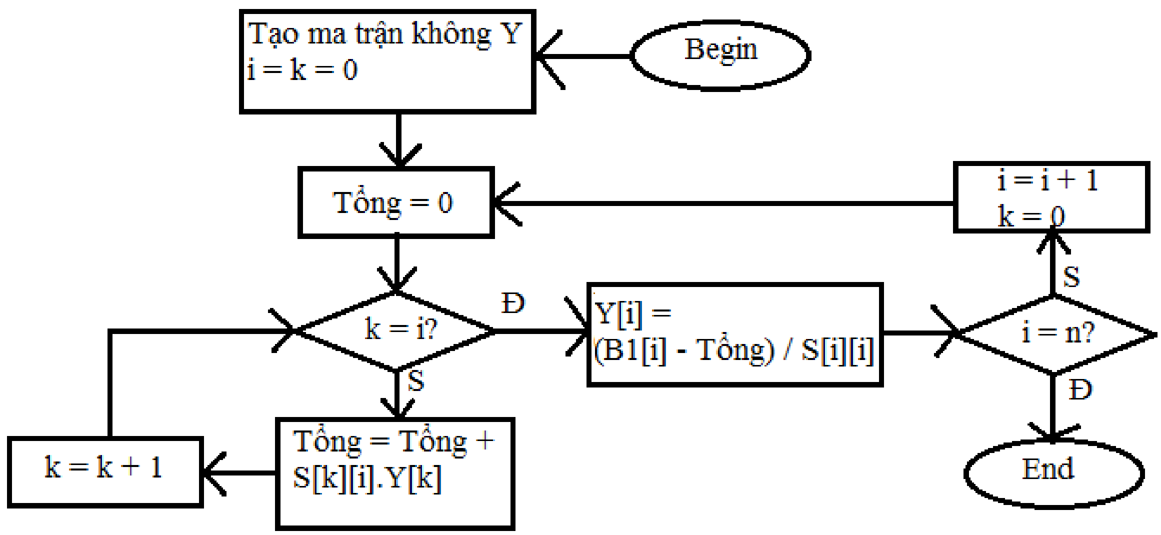
Ở bước này, ta sử dụng tối đa n^2 phép nhân và n^2 phép cộng.

(3) Cholesky:



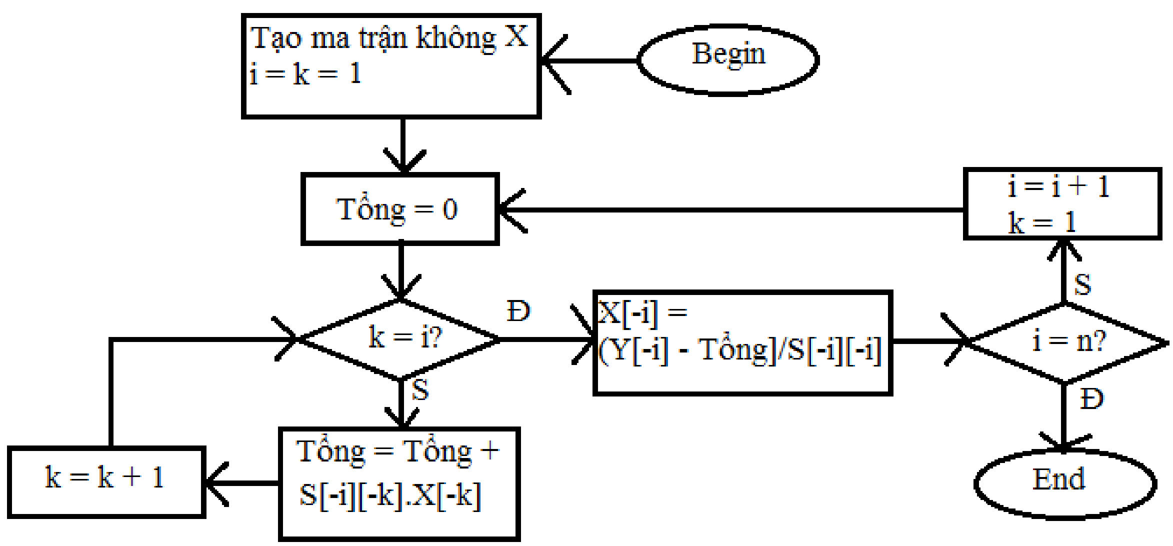
Ở bước này, ta sử dụng tối đa $(n^3+6n^2+11n)/6$ phép nhân; $(n^3+6n^2+17n)/6$ phép cộng.

(4) Cholesky3:



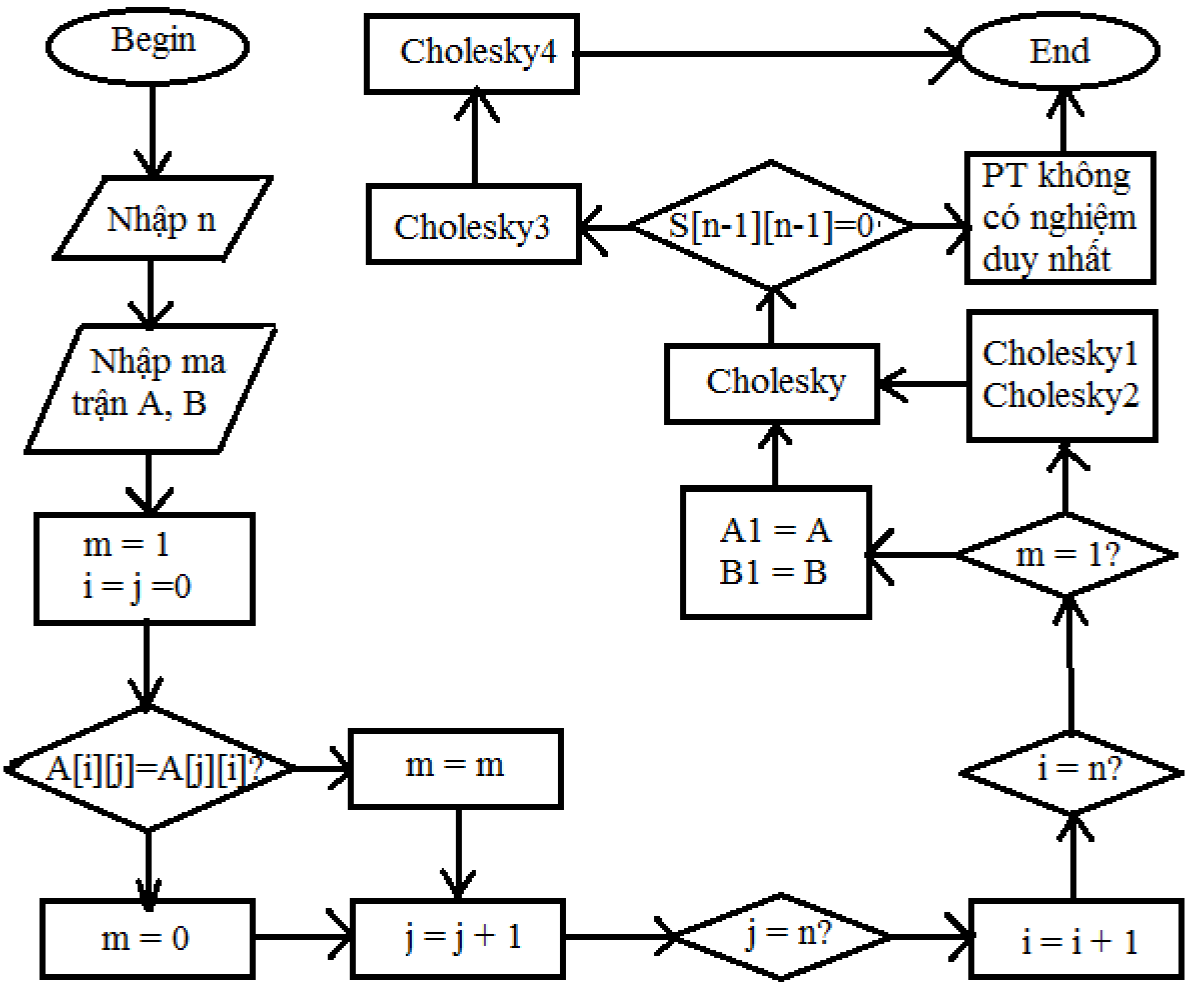
Ở bước này, ta sử dụng tối đa $(n^2+n)/2$ phép nhân và $(n^2+n)/2$ phép cộng.

(5) Cholesky4:



Ở bước này, ta sử dụng tối đa $(n^2+n)/2$ phép nhân và $(n^2+n)/2$ phép cộng.

(6) Thuật toán cuối cùng:



Ở bước này, chúng ta không sử dụng thêm các phép tính, mà chỉ sử dụng các hàm được tạo ra sẵn. Vì vậy, cả thuật toán sử dụng tất cả $(7n^3+18n^2+17n)/6$ phép nhân và $(7n^3+18n^2+23n)/6$ phép cộng.

Ta có bảng sau đây:

n	Số phép nhân	Số phép cộng
3	67	70
5	235	240
10	1495	1505
100	1196950	1197050
1000	1169669500	1169670500

2 Chương trình

2.1 Lập trình trên Python

Chương trình được viết sử dụng Python, xử lí được trường hợp ma trận A không phải ma trận đối xứng, hơn nữa còn xử lí được cả trên trường số phức.

```
from cmath import sqrt
import pprint
import scipy
import scipy.linalg
"'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'''''''''''''"
def cholesky1():
  A1=[[0]*n for i in range(n)]
  for i in range(n):
    for j in range(n):
      A1[i][j]=sum(A[k][i]*A[k][j] for k in range(n))
  return A1
"''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''"
def cholesky2():
  B1=[0]*n
  for i in range(n):
    B1[i]=sum(A[k][i]*B[k] for k in range(n))
  return B1
"''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''"
def cholesky():
  S=[[0]*n for i in range(n)]
  for i in range(n):
    for j in range(i+1):
      tổng = sum(S[k][i] * S[k][j] for k in range(j))
      if abs(tổng- A1[i][i])<0.0000000001:
        break
      if i==j:
        S[j][i] = sqrt(A1[i][i] - tổng)
      else:
        S[j][i] = (A1[j][i] - tổng)/S[j][j]
    if S[i][i]==0:
      break
  return S
```

```
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
def cholesky3():
  Y=[0]*n
  for i in range(n):
    Y[i]=(B1[i]-sum(S[k][i]*Y[k] for k in range(i)))/S[i][i]
  return Y
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
def cholesky4():
  X=[0]*n
  for i in range(1, n+1):
    X[-i]=(Y[-i]-sum(S[-i][-k]*X[-k] for k in range(1, i)))/S[-i][-
i]
  return X
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
n=int(input('Nhập bậc của ma trận A: '))
A=[[0]*n for i in range(n)]
for i in range(n):
  for j in range(n):
    A[i][j]=complex(input('Nhập phần tử A[{}][{}]= '.format(i+1,j+1
)))
print('A= '+str(scipy.array(A)))
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
print('')
print('Nhập ma trận B: ')
B=[0]*n
for i in range(n):
  B[i]=complex(input('Nhập phần tử B[{}]= '.format(i+1)))
print('B= '+str(scipy.array(B)))
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
print('')
print('Giải hệ phương trình tuyến tính bằng phương pháp Cholesky: A
X=B')
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
print('')
print('Tìm ma trận A1 đối xứng:')
m=1
for i in range(n):
  for j in range(n):
    if A[i][j]==A[j][i]:
      m*=1
```

```
    else:
      m*=0
if m==1:
  print('A là ma trận vuông đối xứng có thể khai triển theo Cholesk
y')
  A1=A
  B1=B
else:
  print('A không đối xứng')
  print('Để giải được theo Cholesky ta cần nhân cả 2 vế với A^t:')
  A1=cholesky1()
  B1=cholesky2()
print('')
print('Khi đó phương trình trở thành A1*X=B1, với: ')
print('A1= '+str(scipy.array(A1)))
print('và B1= '+str(scipy.array(B1)))
S=cholesky()
if S[n-1][n-1]==0:
  print('Ma trận A1 có định thức bằng 0 nên không thể khai triển Ch
olesky')
  print('Suy ra phương trình AX=B không có nghiệm duy nhất')
else:
  print('')
  print('Phân tích A1 theo Cholesky: A1=S^t*S')
  print('S= '+str(scipy.array(S)))
  print('')
  print('Khi đó phương trình trở thành: S^t*S*X=B1')
  print('Đặt Y=S*X có:  S^t*Y=B1')
  print('Giải phương trình này ta được: ')
  Y=cholesky3()
  print('Y= '+str(scipy.array(Y)))
  print('')
  print('Giải phương trình SX=Y ta được nghiệm của phương trình:')
  X=cholesky4()
  print('X= '+str(scipy.array(X)))
```

2.2 Ví dụ minh họa

VD1: Giải hệ phương trình $\begin{cases} 5x_1 + 9x_2 + 6x_3 = 5 \\ 7x_1 + 8x_2 + 3x_3 = 8 \\ 4x_1 + 2x_2 + x_3 = 9 \end{cases}$

Gọi $X = (x_1 \ \ x_2 \ \ x_3)^t$ là nghiệm của hệ phương trình. Khi đó ta đưa về giải vector X thỏa mãn

$$\begin{pmatrix} 5 & 9 & 6 \\ 7 & 8 & 3 \\ 4 & 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \\ 9 \end{pmatrix}$$

Tiến hành nhập đầu vào cho chương trình và lấy kết quả đầu ra

```
Nhập bậc của ma trận A: 3
Nhập phần tử A[1][1]= 5
Nhập phần tử A[1][2]= 9
Nhập phần tử A[1][3]= 6
Nhập phần tử A[2][1]= 7
Nhập phần tử A[2][2]= 8
Nhập phần tử A[2][3]= 3
Nhập phần tử A[3][1]= 4
Nhập phần tử A[3][2]= 2
Nhập phần tử A[3][3]= 1
A= [[5.+0.j 9.+0.j 6.+0.j]
 [7.+0.j 8.+0.j 3.+0.j]
 [4.+0.j 2.+0.j 1.+0.j]]
Nhập ma trận B:
Nhập phần tử B[1]= 5
Nhập phần tử B[2]= 8
Nhập phần tử B[3]= 9
B= [5.+0.j 8.+0.j 9.+0.j]
Giải hệ phương trình tuyến tính bằng phương pháp Cholesky: AX=B
Tìm ma trận A1 đối xứng:
A không đối xứng
Để giải được theo Cholesky ta cần nhân cả 2 vế với A^t:
Khi đó phương trình trở thành A1*X=B1, với:
A1= [[ 90.+0.j 109.+0.j  55.+0.j]
 [109.+0.j 149.+0.j  80.+0.j]
 [ 55.+0.j  80.+0.j  46.+0.j]]
và B1= [117.+0.j 127.+0.j  63.+0.j]
Phân tích A1 theo Cholesky: A1=S^t*S
S= [[ 9.48683298+0.j 11.48960883+0.j  5.79750904+0.j]
 [ 0.          +0.j  4.12175799+0.j  3.24834426+0.j]
 [ 0.          +0.j  0.          +0.j  1.3554145 +0.j]]
Khi đó phương trình trở thành: S^t*S*X=B1
Đặt Y=S*X có:  S^t*Y=B1
Giải phương trình này ta được:
Y= [12.33288287+0.j -3.56643938+0.j  2.2760734 +0.j]
Giải phương trình SX=Y ta được nghiệm của phương trình:
X= [ 2.9245283 +0.j -2.18867925+0.j  1.67924528+0.j]
```

IV Ứng dụng trong Toán học

1 Tính định thức

Cholesky có thể được dùng để tính định thức của ma trận rất hiệu quả.

$$\det A = \det L . \det U = 1 . \det U = \prod_{i=1}^{n} u_{ii}$$

Hoặc

$$\det A = \det S^t . \det S = \prod_{i=1}^{n} s_{ii}^2$$

2 Tìm ma trận nghịch đảo

Khi giải hệ phương trình, thường b được xem là vecto có chiều dài bằng với số dòng của ma trận A_{nxn}. Nếu thay vì vecto b, ta có ma trận B, với B là ma trận có kích thước giống A_{nxn}: B_{nxn}

Ta vẫn có phương trình: AX=B

Có thể sử dụng cùng phương pháp ở trên để giải cho mỗi cột của ma trận X. Với giả sử rằng B là ma trận đơn vị với kích thước n thì X khi đó sẽ là nghịch đảo của A.

Ta có công thức sau đây:

$$A^{-1} = A^t . (A . A^t)^{-1}$$

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. Bùi Xuân Diệu (2009), *Bài giảng Đại số tuyến tính*, Trường Đại học Bách khoa Hà Nội.

3. https://www.Wikipedia.org/

4. https://www.quantstart.com/articles/Cholesky-Decomposition-in-Python-and-NumPy/