TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA: CÔNG NGHỆ THÔNG TIN
BỘ MÔN: CÔNG NGHỆ PHẦN MỀM

ĐỀ THI GIỮA KỲ 1 năm học 2023-2024

Tên học phần: Toán ứng dụng CNTT
Mã học phần: **……** Số tín chỉ: **03**
Phương pháp đánh giá (*): Tự luận có giám sát Thời gian làm bài: **120** phút

☐ Sinh viên không được sử dụng tài liệu khi làm bài.

Họ tên: Trương Quang Lộc **Lớp**:21TCLC_DT3 **MSSV: 102210214**

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV_HọTên.pdf và nộp bài thông qua MSTeam

Câu 1 (*2 điểm*): Cho hệ phương trình

$$\begin{cases} x \equiv a_1 (mod\ m_1) \\ x \equiv a_2 (mod\ m_2) \\ \quad \cdots \\ x \equiv a_k (mod\ m_k) \end{cases}$$

a) Trình bày thuật toán giải hệ trên

 # Trả lời: Trình bày thuật toán bằng ngôn ngữ tự nhiên.
Sử dụng định lý thặng dư Trung Hoa để giải hệ phương trình đồng dư bậc nhất

Bước 1: Tính M = m1.m2....mk
Bước 2: Tính Mk = M/mk
Bước 3: Tính yk ☰ NghichDaoModule(Mk) (mod mk)
Bước 4: Hệ có nghiệm duy nhất: x ☰ (a1M1y1 + a2M2y2 + ... + akMkyk) (mod M)

 # Trả lời: Trình bày thuật toán bằng mã giả.

```
Function NghichDaoModule(A, M):
    For x:=1 -> M:
        If ((A mod M)*(X mod M)) mod M == 1:
            Return x

Function ThangDuTrungHoa(int k): #k là số phương trình của hệ
    Khởi tạo:
                các array: ai[], mi[], Mi[], yi[]
                M = 1, x = 0
    For i:=0 -> k-1:
        Input(ai)
```

```
        Input(mi)
        M *= mi[i]
    For i:=0 -> k-1:
        Mi[i] = M/mi[i]
        yi[i] = NghichDaoModule(Mi[i], mi[i])
        x += ai[i]*Mi[i]*yi[i]

    Print("Nghiem {x}")
```

b) Hãy viết chương trình giải hệ phương trình trên

 # Trả lời: Dán code vào bên dưới và có giải thích chi tiết

```python
def main():
    print("Tim nghiem he phuong trinh thang du Trung Hoa")
    k = int(input("Nhap so phuong trinh cua he: "))
    thang_du_trung_hoa(k)

def thang_du_trung_hoa(k):
    ai = [] # List ai chứa các hệ số ai của hệ
    mi = [] # List mi chứa các hệ số mi của hệ
    Mi = [] # List Mi chứa M1, M2,...Mk
    yi = [] # List yi chứa yk là các nghịch đảo module mk của Mk
    M = 1
    x = 0
    # Nhập các phần tử của hệ
    for i in range(0, k):
        ai.append(int(input(f"a{i+1} = ")))
        mi.append(int(input(f"m{i+1} = ")))
        M *= mi[i]
    # List Mi chứa M1, M2,...Mk
    for i in range(k):
        Mi.append(int(M/mi[i]))

    # List yi chứa yk là các nghịch đảo module mk của Mk
    for i in range(0, k):
        yi.append(NghichDaoModule(Mi[i], mi[i]))

    # Tính nghiệm x
    for i in range(0, k):
        x += ai[i]*Mi[i]*yi[i]

    print(f"Nghiem: {x} mod {M} (chua rut gon)")

# Hàm tính nghịch đảo module M của A
def NghichDaoModule(A, M):
    for X in range(1, M):
        if (((A % M) * (X % M)) % M == 1):
            return X
```

```
    return -1

if __name__ == "__main__":
    main()
```

 # Trả lời: Dán kết quả của code với trường hợp

$$\begin{cases} x \equiv 3(mod\ 5) \\ x \equiv 5(mod\ 7) \\ x \equiv 8(mod\ 11) \\ x \equiv 7(mod\ 13) \end{cases}$$

```
Nhap so phuong trinh cua he: 4
a1 = 3
m1 = 5
a2 = 5
m2 = 7
a3 = 8
m3 = 11
a4 = 7
m4 = 13
Nghiem: 30973 mod 5005 (chua rut gon)
```

Câu 2 (3 *điểm*): Cho ma trận A

a) Trình bày thuật toán để phân rã ma trận A bằng SVD

 # Trả lời: Mô tả thuật toán bằng ngôn ngữ tự nhiên

Thuật toán phân rã ma trận A = UDV^T

Với

A: ma trận đầu vào kích thước nxm

U: ma trân vecto riêng trái của A kích thước nxn

V^T: ma trận vector riêng phải của A kích thước mxm

D: ma trận chéo hóa

Bước 1: Tìm V^T

- Tính AT là ma trận chuyển vị của A
- Tính AT.A

Bước 1: Tìm V^T:

- Tìm trị riêng và vecto riêng của A^T.A thu được V là các vecto riêng
- Lặp for duyệt qua từng hàng và cột A, lưu giá trị phần tử ở cột thứ i của A vào hàng thứ i của A^T

Bước 2: Tìm D

- Gọi lamda(i) lần lượt là căn bậc 2 trị riêng của A^T.A D là ma trận đường chéo tạo bởi lamda(i)

Bước 3: Tìm U

- Lặp i từ 1-> length(lamda)
- Tìm vecto Ui theo công thức: Ui=A.Vi/lamda(i)

Trả lời: Mô tả thuật toán bằng mã giả.

```
Ma trận A nxm
For i=0->m-1:
    Row[]
    For j=0->n
        row.add(A[j][i])
        AT.add(row)
//Thuật toán tìm D
    A2= A^T.A
    E,V= numpy.linalg.eig(A2)
    for i từ 0-> (len(E)-1):
        D[i][i]=sqrt(E[i])

//Thuật toán tìm U
    for i từ 0->n-1:
        col []=A.V[i]/E[i]
        U.add(col)
```

b) Viết chương trình để thực hiện phân rã SVD

Trả lời: Dán code vào bên dưới và có giải thích chi tiết.

```python
import numpy
import math

def phan_ra_svd(A):
    A = numpy.array(A, dtype=float)

    AT = []  # AT la ma tran chuyen vi cua A
    for i in range(len(A)):
        row = []
        for j in range(len(A)):
            row.append(A[j][i])
        AT.append(row)

    AT = numpy.array(AT, dtype=float)
    A2 = numpy.matmul(AT, A)  # A2=AT.A
    A2 = numpy.array(A2, dtype=float)

    # Giá trị riêng E, vector riêng V
    E, V = numpy.linalg.eig(A2)
    E = numpy.array(E, dtype=float)
    E = numpy.flip(E, axis=0)
    V = numpy.array(V, dtype=float)

    for i in range(len(V)):
        V[i] = V[i].round(3)

    E1 = []
    for i in range(len(E)):
        if numpy.isclose(E[i], 0):
            continue
        else:
```

```
            E1.append(math.sqrt(E[i]))



    E1 = numpy.array(E1, dtype=float)
    E1 = numpy.flip(E1, axis=0)
    # Ma trận chéo hóa
    D = []
    for i in range(len(E1)):
        row = []
        for j in range(len(E1)):
            if i == j:
                row.append(E1[i])
            else:
                row.append(0)
        D.append(row)

    # Ma trận vector riêng trái
    U = []
    for i in range(len(E1)):
        U.append(numpy.dot(A, V[i]) / E1[i])

    U = numpy.array(U, dtype=float)

    print("\nA=UDV^T")
    print("Ma tran U:")
    for row in U:
        for col in row:
            print(round(col, 3), end=" ")
        print()


    print("\n\nMa tran D:")
    for i in range(len(D)):
        for j in range(len(D)):
            print(round(D[j][i], 3), end="\t")
        print()


    for i in range(len(A)):
        for j in range(i, len(A)):
            t = V[i][j]
            V[i][j] = V[j][i]
            V[j][i] = t

    print("\n\nMa tran V^T:")
    for i in range(len(V)):
        for j in range(len(V)):
            print(round(V[j][i], 3), end="\t")
        print()

def main():
```

```
    A = [
        [1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5]
    ]
    print("Ma tran A:")
    for i in range(len(A)):
        for j in range(len(A)):
            print(A[j][i], end="\t")
        print()
    phan_ra_svd(A)


if __name__ == "__main__":
    main()
```

Trả lời: Thực thi và dán kết quả minh hoạ cho ma trận 15 x 15.

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & \dots & 0 & 0 \\ 2 & 5 & 2 & 0 & \dots & 0 & 0 \\ 0 & 2 & 5 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 2 & 5 & 2 & 0 \\ 0 & 0 & \dots & 0 & 2 & 5 & 2 \\ 0 & 0 & \dots & 0 & 0 & 2 & 5 \end{pmatrix}$$

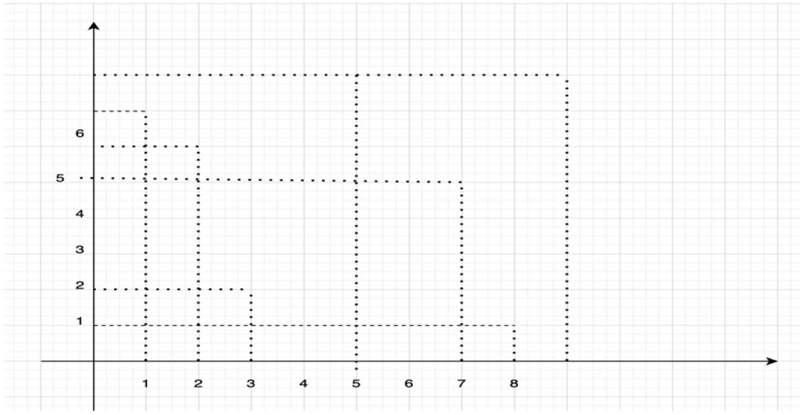
```
Ma tran A:
1       2       0       0       0       0       0       0       0       0       0       0       0       0       0
2       5       2       0       0       0       0       0       0       0       0       0       0       0       0
0       2       5       2       0       0       0       0       0       0       0       0       0       0       0
0       0       2       5       2       0       0       0       0       0       0       0       0       0       0
0       0       0       2       5       2       0       0       0       0       0       0       0       0       0
0       0       0       0       2       5       2       0       0       0       0       0       0       0       0
0       0       0       0       0       2       5       2       0       0       0       0       0       0       0
0       0       0       0       0       0       2       5       2       0       0       0       0       0       0
0       0       0       0       0       0       0       2       5       2       0       0       0       0       0
0       0       0       0       0       0       0       0       2       5       2       0       0       0       0
0       0       0       0       0       0       0       0       0       2       5       2       0       0       0
0       0       0       0       0       0       0       0       0       0       2       5       2       0       0
0       0       0       0       0       0       0       0       0       0       0       2       5       2       0
0       0       0       0       0       0       0       0       0       0       0       0       2       5       2
0       0       0       0       0       0       0       0       0       0       0       0       0       2       5
```

```
Ma tran U:
-0.008 -0.006 0.051 0.095 0.056 -0.016 0.093 0.088 -0.103 -0.182 -0.336 -0.563 -0.318 -0.096 0.014
-0.032 -0.025 0.182 0.327 0.194 -0.048 0.187 0.199 -0.121 -0.2 -0.019 0.204 0.054 -0.03 -0.004
-0.051 -0.052 0.228 0.353 0.208 -0.018 -0.113 -0.066 0.23 0.381 0.245 0.019 0.159 0.157 -0.008
-0.062 -0.087 0.154 0.153 0.052 0.07 -0.231 -0.253 -0.058 -0.157 -0.215 -0.107 -0.289 -0.25 0.028
-0.06 -0.124 -0.022 -0.151 -0.201 0.104 0.131 0.065 -0.249 -0.244 -0.036 0.065 0.304 0.279 -0.064
-0.042 -0.147 -0.239 -0.368 -0.347 -0.045 0.326 0.341 0.285 0.396 0.309 0.06 -0.194 -0.225 0.121
-0.005 -0.133 -0.407 -0.354 -0.183 -0.254 -0.117 -0.106 0.123 -0.128 -0.378 -0.191 -0.018 0.085 -0.206
0.056 -0.049 -0.439 -0.111 0.256 -0.149 -0.486 -0.468 -0.495 -0.291 0.144 0.239 0.271 0.131 0.327
0.142 0.138 -0.28 0.214 0.593 0.371 0.021 0.225 0.165 0.413 0.262 -0.159 -0.48 -0.398 -0.49
0.255 0.463 0.072 0.416 0.358 0.654 0.724 0.646 0.63 -0.094 -0.557 -0.028 0.542 0.682 0.696
0.392 0.936 0.558 0.357 -0.476 -0.134 0.268 -0.515 -0.636 -0.344 0.477 0.234 -0.363 -0.937 -0.941
0.535 1.514 1.066 0.062 -1.135 -1.486 -1.001 -0.841 -0.637 0.427 0.03 -0.328 -0.087 1.117 1.194
0.64 2.027 1.416 -0.285 -0.516 -1.132 -0.967 1.145 1.331 -0.052 -0.718 0.223 0.7 -1.153 -1.365
0.611 2.083 1.374 -0.437 1.216 1.791 0.946 0.767 0.423 -0.37 1.09 0.007 -1.098 0.961 1.267
```

```
Ma tran D:
8.916   0       0       0       0       0       0       0       0       0       0       0       0
0       8.669   0       0       0       0       0       0       0       0       0       0       0
0       0       8.267   0       0       0       0       0       0       0       0       0       0
0       0       0       7.729   0       0       0       0       0       0       0       0       0
0       0       0       0       7.075   0       0       0       0       0       0       0       0
0       0       0       0       0       6.333   0       0       0       0       0       0       0
0       0       0       0       0       0       5.533   0       0       0       0       0       0
0       0       0       0       0       0       0       4.709   0       0       0       0       0
0       0       0       0       0       0       0       0       3.894   0       0       0       0
0       0       0       0       0       0       0       0       0       3.122   0       0       0
0       0       0       0       0       0       0       0       0       0       2.427   0       0
0       0       0       0       0       0       0       0       0       0       0       1.839   0
0       0       0       0       0       0       0       0       0       0       0       0       1.387
0       0       0       0       0       0       0       0       0       0       0       0       0       1.1
```

```
Ma tran V^T:
0.025   -0.049  0.072   0.095   0.116   -0.136  0.153   0.167   -0.177  -0.182  -0.18   -0.866  -0.167  -0.136  0.08
0.097   -0.187  0.263   0.32    0.353   -0.362  0.346   0.309   -0.256  -0.193  -0.129  0.433   -0.07   -0.026  0.004
0.166   -0.295  0.358   0.341   0.25    -0.105  -0.06   -0.212  0.319   0.364   0.346   -0.217  0.277   0.183   -0.087
0.228   -0.354  0.321   0.146   -0.094  0.291   -0.362  -0.278  0.08    -0.148  -0.316  0.108   -0.368  -0.305  0.166
0.28    -0.354  0.167   -0.142  -0.347  0.3     -0.036  0.252   -0.363  -0.225  0.061   -0.054  0.305   0.367   -0.237
0.321   -0.295  -0.049  -0.34   -0.267  -0.092  0.353   0.242   0.121   0.359   0.237   0.027   -0.113  -0.358  0.296
0.348   -0.188  -0.246  -0.322  0.071   -0.361  0.13    -0.288  0.296   -0.113  -0.367  -0.014  -0.126  0.28    -0.34
0.36    -0.049  -0.354  -0.099  0.34    -0.149  -0.318  -0.2    -0.285  -0.254  0.234   0.007   0.312   -0.148  0.367
0.358   0.097   -0.332  0.187   0.282   0.262   -0.215  0.317   -0.139  0.351   0.065   -0.003  -0.367  -0.013  -0.376
0.34    0.228   -0.188  0.354   -0.047  0.323   0.261   0.154   0.362   -0.076  -0.318  0.002   0.269   0.172   0.366
0.309   0.321   0.024   0.296   -0.331  -0.046  0.284   -0.339  -0.061  -0.28   0.344   -0.001  -0.057  -0.297  -0.338
0.264   0.36    0.228   0.05    -0.296  -0.354  -0.185  -0.104  -0.328  0.338   -0.124  0.0     -0.178  0.365   0.293
0.208   0.34    0.348   -0.228  0.024   -0.19   -0.334  0.354   0.243   -0.038  -0.184  -0.0    0.339   -0.362  -0.234
0.144   0.264   0.341   -0.361  0.321   0.227   0.096   0.053   0.193   -0.303  0.361   0.0     -0.357  0.289   0.163
0.073   0.144   0.208   -0.264  0.309   0.341   0.359   -0.362  -0.35   0.322   -0.281  -0.0    0.226   -0.16   -0.083
```

Câu 3 (2 *điểm*): Cho không gian Oxy và 7 điểm tương ứng như hình vẽ dưới:



a) *Trình bày thuật toán xác định bao lồi*

 # Trả lời: Trình bày thuật toán trên bằng ngôn ngữ tự nhiên
Bước 1: Sắp xếp các điểm tăng dần theo hoành độ x (hoành độ bằng nhau thì xét tung độ y)
Bước 2: Xác định các điểm thuộc bao trên Lupper và bao dưới Llower
Bước 3:

- Xác định bao trên:
  - Bước con 1: Bổ sung 2 điểm đầu tiên vào bao trên
  - Bước con 2: Bổ sung điểm tiếp theo
  - Bước con 3: Kiểm tra nếu 3 điểm cuối cùng trong danh sách không tạo thành hướng kim đồng hồ (rẽ phải) thì xóa điểm ở giữa.
  - Quay lại bước con 2

- Xác định bao dưới:
  - Bước con 1: Duyệt danh sách nhưng bị đảo ngược
  - Bước con 1: Bổ sung 2 điểm đầu tiên vào bao dưới
  - Bước con 2: Bổ sung điểm tiếp theo
  - Bước con 3: Kiểm tra nếu 3 điểm cuối cùng trong danh sách không tạo thành hướng kim đồng hồ (rẽ trái) thì xóa điểm ở giữa.
  - Quay lại bước con 2

Bước 4: Xóa điểm đầu và cuối của bao dưới Llower
Bước 5: Kết hợp Lupper và Llower ta được các điểm bao lồi cần tìm

 # Trả lời: Trình bày thuật toán trên bằng mã giả

```
Function orientation(Point p, Point q, Point r) // Hàm kiểm tra hướng tạo thành của 3 điểm
    int val = (q[1] - p[1]) * (r[0] - q[0]) - (q[0] - p[0]) * (r[1] - q[1])
    if (val == 0): return 0 // 3 điểm thẳng hàng
    if (val > 0): return 1 // 3 điểm tạo thành kim đồng hồ
    if (val < 0): return -1 // 3 điểm tạo thành hướng ngược kim đồng hồ

Function convex_hull(List<Point> points) // Hàm tìm các điểm bao lồi
    if (length(points) <3):  // trả về lại các điểm points nếu số điểm bé hơn 3
        return points
    else:
        points.sort() // sắp xếp các điểm tăng dần theo hoành độ x
        upper_hull = [], lower_hull = [] // bao trên và bao dưới
        For point in points:
            while len(upper_hull) >= 2 và orientation(upper_hull[-2], upper_hull[-1], point) != -1: // 3
điểm cuối cùng không rẽ trái
                    upper_hull.pop()
            upper_hull.append(point)
```

For point in reverse(points):
while len(lower_hull) >= 2 và orientation(lower_hull[-2], lower _hull[-1], point) != -1: // 3 điểm cuối cùng không rẽ trái
lower _hull.pop()
lower_hull.append(point)

Xóa điểm đầu và điểm cuối lower_hull
Kết hợp upper_hull và lower_hull là danh sách các điểm bao lồi cần tìm

b) Viết chương trình xác định bao lồi kèm giải thích chi tiết

Trả lời: viết câu trả lời vào bên dưới:

```
def orientation(p, q, r): # hướng tạo thành của 3 điểm
    # -1 rẽ trái
    # 0 thẳng hàng
    # 1 rẽ phải
    val = (q[1] - p[1]) * (r[0] - q[0]) - (q[0] - p[0]) * (r[1] - q[1])
    if val == 0:
        return 0
    return 1 if val > 0 else -1

def convex_hull(points):
    n = len(points)

    if n < 3:
        return points

    # Sắp xếp các điểm tăng dần theo tạo độ x (x bằng nhau thì xét y)
    points.sort(key=lambda p: (p[0], p[1]))

    # khởi tạo 2 list bao trên và bao dưới
    upper_hull = []
    lower_hull = []

    # bao trên
    for point in points:
        # bổ sung 2 điểm đầu
        # bổ sung điểm thứ 3, nếu 3 điểm cuối của list ko tạo thành rẽ phải thì xóa điểm giữa
        while len(upper_hull) >= 2 and orientation(upper_hull[-2], upper_hull[-1], point) != -1:
            upper_hull.pop()
        upper_hull.append(point)

    # bao dưới
    for point in reversed(points):
        # bổ sung 2 điểm cuối
        # bổ sung điểm thứ 3, nếu 3 điểm cuối của list ko tạo thành rẽ phải thì xóa điểm giữa
        while len(lower_hull) >= 2 and orientation(lower_hull[-2], lower_hull[-1], point) != -1:
```

```
            lower_hull.pop()
        lower_hull.append(point)

    # kết hợp bao trên và bao dưới
    convex_hull = upper_hull[:-1] + lower_hull[:-1]

    return convex_hull

def main():
    points = [(1, 7), (2, 6), (3, 2), (5, 8), (7, 5), (8, 1), (9, 8)]
    convex_hull_points = convex_hull(points)
    print("Toa do cac diem bao loi:", convex_hull_points)

if __name__ == "__main__":
    main()
```

Trả lời: Dán kết quả thực thi minh họa với 7 điểm cho ở trên

```
points = [(1, 7), (2, 6), (3, 2), (5, 8), (7, 5), (8, 1), (9, 8)]
convex_hull_points = convex_hull(points)
```

```
Toa do cac diem bao loi: [(1, 7), (3, 2), (8, 1), (9, 8), (5, 8)]
```

Câu 4 (*3 điểm*): Cho ma trận A

a) Trình bày thuật toán để phân rã ma trận A theo Cheslosky

Trả lời: Trình bày thuật toán bằng ngôn ngữ tự nhiên
Bước 1: Kiểm tra điều kiện A là ma trận vuông, ma trận đối xứng, và ma trận xác định dương
Bước 2: Khởi tạo ma trận tam giác dưới L có kích thước bằng ma trận A
Bước 3: Các phần tử của ma trận L được tính theo công thức sau:

$$\left\{\begin{array}{ll} L_{i,j} = \sqrt{a_{i,j} - \sum_{k=1}^{j-1} L_{i,k}^2}\,, & i = j \\ L_{i,j} = \frac{1}{L_{j,j}}\left(a_{i,j} - \sum_{k=1}^{j-1} L_{i,k}L_{j,k}\right), & i \neq j \end{array}\right\} i = j, \ldots, n, \qquad j = 1, \ldots, n$$

Bước 4: Khởi tạo ma trận tam giác trên L* là ma trận chuyển vị của L

Trả lời: Trình bày thuật toán bằng mã giả

Function is_hermitian(matrix A): // hàm kiểm tra ma trận vuông và đối xứng, xác định dương
    return np.allclose(matrix, matrix.conj().T)

```
Function phan_ra_cholesky(matrix A): // hàm phân rã ma trận Cholesky
    Khởi tạo ma trận L kích thước bằng ma trận A
    n = length(A) // A là ma trận vuông nxn
    For i:=0 -> n: // Tính các phần tử ma trận L
        For j:=0 -> i+1:
            if (i == j): // các phần tử tại đường chéo
                sum_of_square = 0
                for k:=0 -> j-1:
                    sum_of_square += L[i][k] ^2
                L[i][j] = sqrt((A[i][i]-sum_of_square))
            else: // các phần tử không thuộc đường chéo
                sum_of_product = 0
                for k:=0 -> j-1:
                    sum_of_products += L[i][k] * L[j][k]
                L[i][j] = (A[i][j] - sum_of_products) / L[j][j]
    return L // ma trận tam giác dưới
```

b) Viết chương trình thực hiện phân rã Cholesky và có giải thích chi tiết

Trả lời: Dán code vào bên dưới:

```python
import numpy as np

def main():
    A = np.array([
            [1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            [2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5]], dtype=float)

    if (not is_hermitian(A)):
        print("Khong phai la ma tran vuong, doi xung")
    else:
        L = phan_ra_cholesky(A)
        print("Ma tran tam giac duoi L:")
        for i in range(len(L)):
            for j in range(len(L)):
                print(L[i][j], end="\t")
            print()
```

```
        print("Ma tran tam giac tren L*:")
        for i in range(len(L)):
            for j in range(len(L)):
                print(L[j][i], end="\t")
            print()

# Ham kiem tra ma tran vuong va doi xung (ma tran Hermitian)
def is_hermitian(matrix):
    return np.allclose(matrix, matrix.conj().T)

def phan_ra_cholesky(A):
    n = len(A)
    L = np.zeros((n, n), dtype=float) # Khoi tao ma tran tam giac duoi L

    for i in range(n):
        for j in range(i + 1):
            if i == j: # tại đường chéo
                sum_of_squares = 0
                for k in range(j):
                    sum_of_squares += L[i][k] ** 2
                L[i][j] = (A[i][i] - sum_of_squares) ** 0.5
            else:
                sum_of_products = 0
                for k in range(j):
                    sum_of_products += L[i][k] * L[j][k]
                L[i][j] = (A[i][j] - sum_of_products) / L[j][j]

    return L

if __name__ == "__main__":
    main()
```

Trả lời: Thực thi chương trình và dán kết quả cho ma trận A kích cở 15*15.

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & \dots & 0 & 0 \\ 2 & 5 & 2 & 0 & \dots & 0 & 0 \\ 0 & 2 & 5 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 2 & 5 & 2 & 0 \\ 0 & 0 & \dots & 0 & 2 & 5 & 2 \\ 0 & 0 & \dots & 0 & 0 & 2 & 5 \end{pmatrix}$$

- Nhập ma trận A:

```python
A = np.array([
        [1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 2],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5]], dtype=float)
```

- Kết quả:

```
Ma tran tam giac duoi L:
1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  2.0  1.0
```

```
Ma tran tam giac tren L*:
1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  2.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

Đà Nẵng, ngày 08 tháng 10 năm 2023

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

TRƯỞNG BỘ MÔN
(đã duyệt)