**Lab Session #6**

**Cholesky Decomposition in Python**

**Aim:**

To explore various functions available in SciPy and Math modules in python and implement Cholesky Decomposition using SciPy, Math and PPrint Library modules.

**Problem Definition**:

Develop Python Programs using Math module for following:

1. Declare a 2X2 Matrix and find Cholesky Decomposition for same.
2. Declare a 4X4 Matrix and find Cholesky Decomposition for same.
3. Take a 4X4 Matrix as input from user and find Cholesky Decomposition for same.

Develop Python Programs using SciPy and PPrint module for following:

1. Take a 4X4 Matrix as input from user and find Cholesky Decomposition for same, use pprint for display.

**Theory:**

In linear algebra, a matrix decomposition or matrix factorization is a factorization of a matrix into a product of matrices. There are many different matrix decompositions. One of them is Cholesky Decomposition.

The Cholesky decomposition or Cholesky factorization is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose. The Cholesky decomposition is roughly twice as efficient as the LU decomposition for solving systems of linear equations.

The Cholesky decomposition of a Hermitian positive-definite matrix A is a decomposition of the form A = [L] [L] T, where L is a lower triangular matrix with real and positive diagonal entries, and LT denotes the conjugate transpose of L. Every Hermitian positive-definite matrix (and thus also every real-valued symmetric positive-definite matrix) has a unique Cholesky decomposition.



Every symmetric, positive definite matrix A can be decomposed into a product of a unique lower triangular matrix L and its transpose: **A = L LT**

The following formulas are obtained by solving above lower triangular matrix and its transpose. These are the basis of Cholesky Decomposition Algorithm:



Input:



Output:



**Applications:**

The Cholesky decomposition is mainly used for the numerical solution of linear equations Ax = b. If A is symmetric and positive definite, then we can solve A x = b by first computing the Cholesky decomposition A = L L ∗, then solving L y = b for y by forward substitution, and finally solving L ∗ x = y for x by back substitution.

For linear systems that can be put into symmetric form, the Cholesky decomposition (or its LDL variant) is the method of choice, for superior efficiency and numerical stability. Compared to the LU decomposition, it is roughly twice as efficient.

* **Linear least squares**
* **Non-linear optimization**
* **Monte Carlo simulation**
* **Kalman filters**
* **Matrix inversion**

**SciPy**

SciPy provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and many other classes of problems. The algorithms and data structures provided by SciPy are broadly applicable across domains. SciPy extends NumPy providing additional tools for array computing and provides specialized data structures, such as sparse matrices and k-dimensional trees. SciPy wraps highly-optimized implementations written in low-level languages like FORTRAN, C, and C++. We can enjoy the flexibility of Python with the speed of compiled code. SciPy’s high level syntax makes it accessible and productive for programmers from any background or experience level.

**PPRINT**

The **pprint** module provides a capability to “pretty-print” arbitrary Python data structures in a well-formatted and more readable way

**Codes:**

**1]**

**Program:**

import math

import pprint

A1 = [[4, 2], [2, 5]]

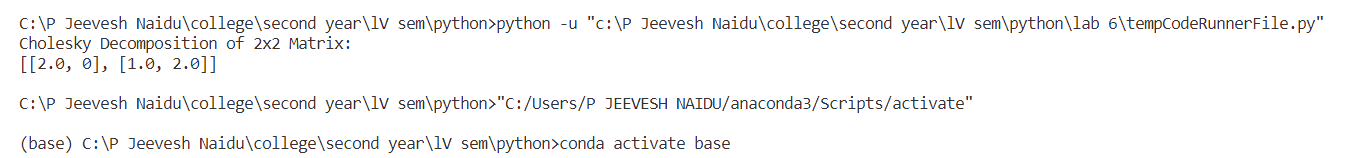
L1 = [[math.sqrt(A1[0][0]), 0], [A1[1][0] / math.sqrt(A1[0][0]), math.sqrt(A1[1][1] - (A1[1][0]

\*\* 2) / A1[0][0])]]

print("Cholesky Decomposition of 2x2 Matrix:")

pprint.pprint(L1)

**Output:**



**2]**

**Program:**

import math

# Declare a 4x4 matrix

A = [[18, 22, 54, 42],

[22, 70, 86, 62],

[54, 86, 174, 134],

[42, 62, 134, 106]]

# Ini􀆟alize an empty 4x4 matrix L

L = [[0] \* 4 for \_ in range(4)]

# Perform Cholesky Decomposi􀆟on

for i in range(len(A)):

for j in range(i+1):

if i == j:

sum\_sq = sum(L[i][k] \*\* 2 for k in range(j))

L[i][j] = math.sqrt(A[i][i] - sum\_sq)

else:

sum\_prod = sum(L[i][k] \* L[j][k] for k in range(j))

L[i][j] = (A[i][j] - sum\_prod) / L[j][j]

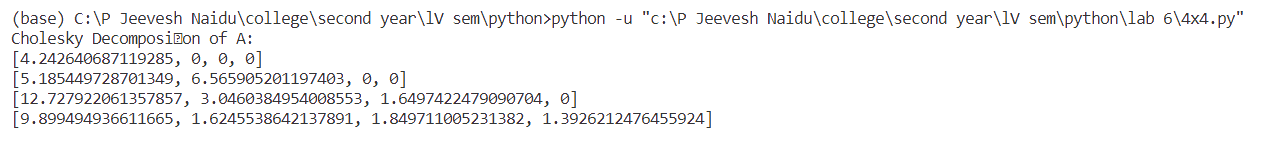
# Print the Cholesky Decomposi􀆟on

print("Cholesky Decomposi􀆟on of A:")

for row in L:

print(row)

**Output:**



**3]**

**Program:**

import math

# Take a 4x4 matrix as input from the user

A = []

print("Enter the elements of the 4x4 matrix:")

for i in range(4):

row = []

for j in range(4):

element = float(input("Enter the value for A[{}][{}]: ".format(i+1, j+1)))

row.append(element)

A.append(row)

# Ini􀆟alize an empty 4x4 matrix L

L = [[0] \* 4 for \_ in range(4)]

# Perform Cholesky Decomposi􀆟on

for i in range(len(A)):

for j in range(i+1):

if i == j:

sum\_sq = sum(L[i][k] \*\* 2 for k in range(j))

L[i][j] = math.sqrt(A[i][i] - sum\_sq)

else:

sum\_prod = sum(L[i][k] \* L[j][k] for k in range(j))

if L[j][j] != 0:

L[i][j] = (A[i][j] - sum\_prod) / L[j][j]

else:

L[i][j] = 0

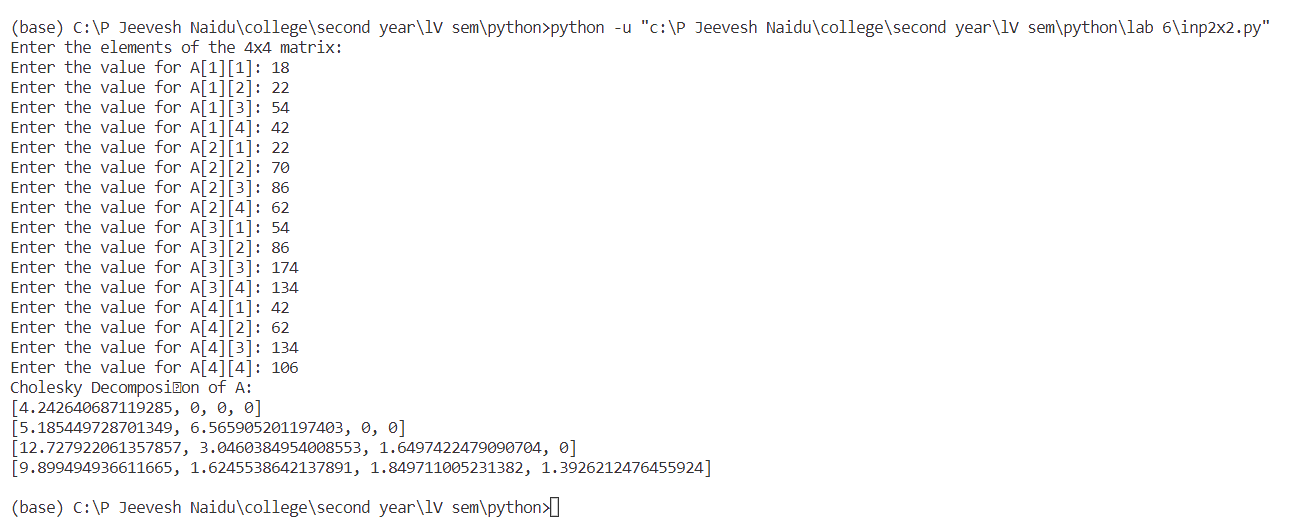
# Print the Cholesky Decomposi􀆟on

print("Cholesky Decomposi􀆟on of A:")

for row in L:

print(row)

**Output:**



**4]**

**Program:**

import numpy as np

import scipy.linalg as la

import pprint

# Take a 4x4 matrix as input from the user

A = np.zeros((4, 4))

for i in range(4):

for j in range(4):

A[i, j] = float(input("Enter the value for A[{}][{}]: ".format(i+1, j+1)))

try:

# Find Cholesky Decomposi􀆟on

L = la.cholesky(A, lower=True)

# Display Cholesky Decomposi􀆟on using pprint

print("Cholesky Decomposi􀆟on of A:")

pp = pprint.PreyPrinter(indent=4)

pp.pprint(L)

except la.LinAlgError:

print("Cholesky decomposi􀆟on cannot be performed. The matrix is not posi􀆟ve definite.")

**Output:**



**CONCLUSION:**

The various functions available in SciPy and Math modules in python were used to implement Cholesky Decomposition using SciPy, Math and PPrint Library modules.