In [1]:
```
import numpy as np
from scipy.linalg import lu, cholesky
from scipy.linalg import solve
import math
```

In [32]:
```
def GENERATE(n):
    A = np.zeros((n, n))
    b = np.random.rand(n, 1)
    for i in range(1, n + 1):              #function to generate matrix A and a random m
        for j in range(1, n + 1):
            A[i - 1, j - 1] = 1 / (i + j - 1)
    return A, b
def row_interchange(B,g,h):
    B[g],B[h]=B[h].copy(),B[g].copy()      #helper function to interchange rows
    return B

def col_interchange(p,g,h):
    p[:,[g,h]]=p[:,[h,g]]                  #helper function to interchange columns
    return p

def forward_sub(L, b):
    n = len(L)
    y = np.zeros((n,1))
    for i in range(n):
        y[i][0] = (b[i][0] - np.dot(L[i, :i], y[:i])) / L[i, i]
    return y

def backward_sub(U, y):
    n = len(U)
    x = np.zeros((n,1))
    for i in range(n - 1, -1, -1):
        x[i][0] = (y[i][0] - np.dot(U[i, i + 1:], x[i + 1:])) / U[i, i]
    return x

def cholesky_solver_scipy(A, b):
    L = cholesky(A, lower=True)
    x = solve(L.T, solve(L, b))
    return x
```

In [11]:
```
def GEPP(A,b,epsilon=10**-10):
    n=len(b)
    for i in range(n-1):
        piv=abs(A[i][i])
        piv_row=i
        for k in range(i+1,n):
            if abs(A[k][i])>piv:
                piv=A[k][i]
                piv_row=k                          #Gaussian Elimination With Parti
        if abs(piv)<epsilon:
            return("Pivot is too small")
        elif piv_row!=i:
            A=row_interchange(A,i,piv_row)
            b=row_interchange(b,i,piv_row)
        for j in range(i+1,n):
            q=A[j][i]/A[i][i]
            b[j][0]=b[j][0]-q*b[i][0]
            for l in range(i,n):
                A[j][l]=A[j][l]-q*A[i][l]
    y=backward_sub(A,b)
    return y
```

In [12]:
```
def GERP(A,b,epsilon=10**-10):
```

```
    n=len(b)
    pos=[]
    for i in range(n-1):
        piv=abs(A[i][i])
        pivpos=(i,i)
        for k in range(i+1,n):
            if abs(A[k][i])>piv:
                piv=A[k][i]
                pivpos=(k,i)
            if abs(A[i][k])>piv:
                piv=A[i][k]
                pivpos=(i,k)
        if abs(piv)<epsilon:
            return("Pivot is too small")
        elif pivpos!=(i,i):                                    #Gaussian Elimination wit
            if pivpos[0]==i:
                A=col_interchange(A,i,pivpos[1])
                pos.append((i,pivpos[1]))
            elif pivpos[1]==i:
                A=row_interchange(A,i,pivpos[0])
                b=row_interchange(b,i,pivpos[0])

        for j in range(i+1,n):
            q=A[j][i]/A[i][i]
            b[j][0]=b[j][0]-q*b[i][0]
            for l in range(i,n):
                A[j][l]=A[j][l]-q*A[i][l]
    y=backward_sub(A,b)
    return y
```

In [31]:

```
def GECP(A,b,epsilon=10**-10):
    n=len(b)
    pos=[]


    for i in range(n-1):
        piv=abs(A[i][i])
        pivpos=(i,i)
        for k in range(i,n):
            for m in range(i,n):
                if abs(A[k][m])>piv:
                    piv=A[k][m]
                    pivpos=(k,m)

        if abs(piv)<epsilon:
            return("Pivot is too small")
        elif pivpos!=(i,i):                                    #Gaussian Eliminat
            if pivpos[0]==i:
                A=col_interchange(A,i,pivpos[1])
                pos.append((i,pivpos[1]))
            elif pivpos[1]==i:
                A=row_interchange(A,i,pivpos[0])
                b=row_interchange(b,i,pivpos[0])

        for j in range(i+1,n):
            q=A[j][i]/A[i][i]
            b[j][0]=b[j][0]-q*b[i][0]
            for l in range(i,n):
                A[j][l]=A[j][l]-q*A[i][l]


    y=backward_sub(A,b)
    for x in pos:
        y=row_interchange(y,x[0],x[1])
    return y
```

In [18]:
```
def Cholesky(A, b, epsilon=10**-10):
    n = len(A)
    L = np.zeros((n, n))


    for i in range(n):
        for j in range(i + 1):
            if j == i:
                sum_sq = sum(L[i][k] ** 2 for k in range(j))
                L[i][j] = math.sqrt(max(A[i][i] - sum_sq, 0))
            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]


    y = forward_sub(L, b)
    x = backward_sub(L.T, y)

    return x
```

In [29]:
```
A,b=GENERATE(10)
```

In [ ]:
```

```

Results

In [1]:
```
import pandas as pd
from scipy.linalg import lu_solve, lu_factor
```

In [2]:
```
%run helper_function_assignment2.ipynb
```

Epsilon=10^-10

In [3]:
```
geppnorm=[]
gerpnorm=[]
gecpnorm=[]
chopnorm=[]
scipylu=[]
scipycholesky=[]

for n in range(1,15):
    A,b=GENERATE(n)
    try:
        geppnorm.append(np.linalg.norm((np.dot(A,GEPP(A,b))-b),ord=2))
    except:
        geppnorm.append("Can not Find")
    A,b=GENERATE(n)
    try:
        gerpnorm.append(np.linalg.norm((np.dot(A,GERP(A,b))-b),ord=2))
    except:
        gerpnorm.append("Can not Find")
    A,b=GENERATE(n)
    try:
        gecpnorm.append(np.linalg.norm((np.dot(A,GECP(A,b))-b),ord=2))
    except:
        gecpnorm.append("Can not Find")
    A,b=GENERATE(n)
    try:
        chopnorm.append(np.linalg.norm((np.dot(A,Cholesky(A,b))-b),ord=2))
    except:
        chopnorm.append("Can not Find")
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_13220\238268824.py:20: RuntimeWarning: divid
e by zero encountered in divide
  y[i][0] = (b[i][0] - np.dot(L[i, :i], y[:i])) / L[i, i]
C:\Users\Admin\AppData\Local\Temp\ipykernel_13220\3386481476.py:13: RuntimeWarning: divi
de by zero encountered in scalar divide
  L[i][j] = (A[i][j] - sum_prod) / L[j][j]
```

In [4]:
```
for i in range(1,15):
    A, b = GENERATE(i)
    LU, piv = lu_factor(A)
    x = lu_solve((LU, piv), b)

    try:
        scipylu.append(np.linalg.norm((np.dot(A, x) - b), ord=2))
    except:
        scipylu.append("Can not Find")


    B, c = GENERATE(i)
```

```
    try:
        x = cholesky_solver_scipy(B, c)
        scipycholesky.append(np.linalg.norm((np.dot(B, x) - c), ord=2))
    except:
        scipycholesky.append("Can not Find")
```

In [5]:
```
pd.set_option('display.float_format', lambda x: '%.20f' % x)
```

In [6]:
```
comparison_table = pd.DataFrame(columns=["n","GEPP", "GERP", "GECP","CHOP","SCIPYLU","SC
comparison_table["GEPP"]=geppnorm
comparison_table["GERP"]=gerpnorm
comparison_table["GECP"]=gecpnorm
comparison_table["CHOP"]=chopnorm
comparison_table["SCIPYLU"]=scipylu
comparison_table["SCIPYCHOLESKY"]=scipycholesky
comparison_table["n"]=[x for x in range(1,15)]

print(comparison_table)
```

```
     n                   GEPP                   GERP                   GECP  \
0    1 0.00000000000000000000 0.00000000000000000000 0.00000000000000000000
1    2 0.00000000000000000000 0.00000000000000000000 0.00000000000000000000
2    3 0.00000000000000059009 0.00000000000000266454 0.00000000000000087595
3    4 0.00000000000002720648 0.00000000000001091553 0.00000000000005196377
4    5 0.00000000000012501536 0.00000000000076691621 0.00000000000020671632
5    6 0.00000000000409541875 0.00000000000652635781 0.00000000006926948669
6    7 0.00000000017679219742 0.00000000032963960969 0.00000000002316059965
7    8 0.00000003197346633063 0.00000007158435312902 0.00000010143816377428
8    9 0.00000025955197799163 0.00000042257969876555 0.00000061073277402565
9   10 0.00007669579352038701 0.00000070831445932736 0.00001593181440658700
10  11           Can not Find           Can not Find 0.00363336394134876686
11  12           Can not Find           Can not Find           Can not Find
12  13           Can not Find           Can not Find           Can not Find
13  14           Can not Find           Can not Find           Can not Find

                    CHOP                  SCIPYLU          SCIPYCHOLESKY
0  0.00000000000000000000 0.00000000000000000000 0.00000000000000000000
1  0.00000000000000000000 0.00000000000000011102 0.00000000000000011102
2  0.00000000000000178673 0.00000000000000144329 0.00000000000000308074
3  0.00000000000005023891 0.00000000000010703005 0.00000000000000075299
4  0.00000000000098567273 0.00000000000138928628 0.00000000000187988041
5  0.00000000008024990413 0.00000000005875921478 0.00000000001453411220
6  0.00000000154910271914 0.00000000137500173317 0.00000000002407628497
7  0.00000009694323125368 0.00000014019553226499 0.00000002440652385564
8  0.00000296170766502015 0.00000051708918921470 0.00000072305960608547
9  0.00011295955860059955 0.00004384866314664532 0.00012955643463650497
10 0.00185036062171633348 0.00179966590317280628 0.00130170481091962669
11 0.10107898434333280335 0.00049722772139242685 0.09364016528330465494
12           Can not Find 0.44111435272384907913 2.13371098429536143470
13           Can not Find 0.65817275952144693729           Can not Find
```

In [7]:
```
comparison_table
```

Out[7]:

	n	GEPP	GERP	GECP	CHOP	
0	1	0.00000000000000000000	0.00000000000000000000	0.00000000000000000000	0.00000000000000000000	0.0000
1	2	0.00000000000000000000	0.00000000000000000000	0.00000000000000000000	0.00000000000000000000	0.0000
2	3	0.00000000000000059009	0.00000000000000266454	0.00000000000000087595	0.00000000000000178673	0.0000
3	4	0.00000000000002720648	0.00000000000001091553	0.00000000000005196377	0.00000000000005023891	0.0000
4	5	0.00000000000012501536	0.00000000000076691621	0.00000000000020671632	0.00000000000098567273	0.0000
5	6	0.00000000000409541875	0.00000000000652635781	0.00000000006926948669	0.00000000008024990413	0.0000

6	7	0.00000000017679219742	0.00000000032963960969	0.00000000002316059965	0.00000000154910271914	0.0000
7	8	0.00000003197346633063	0.00000007158435312902	0.00000010143816377428	0.00000009694323125368	0.0000
8	9	0.00000025955197799163	0.00000042257969876555	0.00000061073277402565	0.00000296170766502015	0.0000
9	10	0.00007669579352038701	0.00000708314459327 36	0.00000159318144065870	0.00011295955860059955	0.0000
10	11	Can not Find	Can not Find	0.00363336394134876686	0.00185036062171633348	0.0017
11	12	Can not Find	Can not Find	Can not Find	0.10107898434333280335	0.0004
12	13	Can not Find	Can not Find	Can not Find	Can not Find	0.4411
13	14	Can not Find	Can not Find	Can not Find	Can not Find	0.6581

In [ ]:

Epsilon=10^-8

In [8]:
```
geppnorm=[]
gerpnorm=[]
gecpnorm=[]
chopnorm=[]
scipylu=[]
scipycholesky=[]
c=10**-8
for n in range(1,15):
    A,b=GENERATE(n)
    try:
        geppnorm.append(np.linalg.norm((np.dot(A,GEPP(A,b,c))-b),ord=2))
    except:
        geppnorm.append("Can not Find")
    A,b=GENERATE(n)
    try:
        gerpnorm.append(np.linalg.norm((np.dot(A,GERP(A,b,c))-b),ord=2))
    except:
        gerpnorm.append("Can not Find")
    A,b=GENERATE(n)
    try:
        gecpnorm.append(np.linalg.norm((np.dot(A,GECP(A,b,c))-b),ord=2))
    except:
        gecpnorm.append("Can not Find")
    A,b=GENERATE(n)
    try:
        chopnorm.append(np.linalg.norm((np.dot(A,Cholesky(A,b,c))-b),ord=2))
    except:
        chopnorm.append("Can not Find")
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_13220\238268824.py:20: RuntimeWarning: divid
e by zero encountered in divide
  y[i][0] = (b[i][0] - np.dot(L[i, :i], y[:i])) / L[i, i]
C:\Users\Admin\AppData\Local\Temp\ipykernel_13220\3386481476.py:13: RuntimeWarning: divi
de by zero encountered in scalar divide
  L[i][j] = (A[i][j] - sum_prod) / L[j][j]
```

In [9]:
```
for i in range(1,15):
    A, b = GENERATE(i)
    LU, piv = lu_factor(A)
    x = lu_solve((LU, piv), b)

    try:
        scipylu.append(np.linalg.norm((np.dot(A, x) - b), ord=2))
```

```
    except:
        scipylu.append("Can not Find")


    B, c = GENERATE(i)
    try:
        x = cholesky_solver_scipy(B, c)
        scipycholesky.append(np.linalg.norm((np.dot(B, x) - c), ord=2))
    except:
        scipycholesky.append("Can not Find")
scipylu
```

Out[9]:
```
[0.0,
 1.2412670766236366e-16,
 3.0887354325867102e-15,
 1.7825089645747714e-13,
 2.8840606935409507e-13,
 1.0215331403602658e-10,
 6.934945699724709e-09,
 2.5835465173631645e-08,
 1.7502246813701903e-06,
 3.717327165034969e-05,
 0.00038722991508699734,
 0.011370963757870587,
 0.9777890393736918,
 2.0322146238470973]
```

In [10]:
```
pd.set_option('display.float_format', lambda x: '%.20f' % x)
```

In [11]:
```
comparison_table1 = pd.DataFrame(columns=["n","GEPP", "GERP", "GECP","CHOP","SCIPYLU","S
comparison_table1["GEPP"]=geppnorm
comparison_table1["GERP"]=gerpnorm
comparison_table1["GECP"]=gecpnorm
comparison_table1["CHOP"]=chopnorm
comparison_table1["SCIPYLU"]=scipylu
comparison_table1["SCIPYCHOLESKY"]=scipycholesky
comparison_table1["n"]=[x for x in range(1,15)]

print(comparison_table1)
```

```
     n                    GEPP                    GERP                    GECP  \
0    1 0.00000000000000000000 0.00000000000000000000 0.00000000000000000000
1    2 0.00000000000000000000 0.00000000000000022204 0.00000000000000000000
2    3 0.00000000000000070217 0.00000000000000015762 0.00000000000000028610
3    4 0.00000000000005553873 0.00000000000000206812 0.00000000000000806805
4    5 0.00000000000169246878 0.00000000000044810670 0.00000000000002785404
5    6 0.00000000000494155015 0.00000000002288281843 0.00000000000303778606
6    7 0.00000000054893339712 0.00000000075704517585 0.00000000026850513574
7    8 0.00000000553672507413 0.00000000607820117602 0.00000001143647463945
8    9          Can not Find          Can not Find 0.00000022089348061130
9   10          Can not Find          Can not Find          Can not Find
10  11          Can not Find          Can not Find          Can not Find
11  12          Can not Find          Can not Find          Can not Find
12  13          Can not Find          Can not Find          Can not Find
13  14          Can not Find          Can not Find          Can not Find

                    CHOP                 SCIPYLU          SCIPYCHOLESKY
0  0.00000000000000000000 0.00000000000000000000 0.00000000000000000000
1  0.00000000000000024825 0.00000000000000012413 0.00000000000000000000
2  0.00000000000000044409 0.00000000000000308874 0.00000000000000120089
3  0.00000000000008489461 0.00000000000017825090 0.00000000000007625729
4  0.00000000000163507791 0.00000000000028840607 0.00000000000029393766
5  0.00000000002532924328 0.00000000010215331404 0.00000000001379144273
6  0.00000000132147907664 0.00000000693494569972 0.00000004223076653 73
7  0.00000012320273700724 0.00000002583546517363 0.00000007319688982398
8  0.00000022110440700529 0.00000175022468137019 0.00000184654279182172
```

```
9  0.00005659856793894411 0.00003717327165034969 0.00002171486441136075
10 0.00135221880139015744 0.00038722991508699734 0.00169098957462365915
11 0.10436930539931590922 0.01137096375787058702 0.00016130712384469277
12           Can not Find 0.97778903937369177068 0.37648535644954467250
13           Can not Find 2.03221462384709727900           Can not Find
```

In [ ]: