In [1]:

```python
import numpy as np
from scipy.linalg import lu
from scipy.linalg import solve
import random
import time
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```python
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
```

In [3]:

```python
def LUFACT(A):

    n=len(A)
    U=np.copy(A).astype(float)                       #Initializing U and L as A and I res
    p=[(0,0) for i in range(n+1)]
    l=[np.eye(n) for r in range(n+1)]


    for k in range(n-1):
        maxp=abs(U[k][k])
        maxrow=k
        for z in range(k+1,n):
            if abs(U[z][k])>maxp:
                maxp=abs(U[z][k])                              #partial pivoting step
                maxrow=z
        if maxp==0:
            return("can not find non zero pivots")            #To avoid Singular matrices
        elif maxrow!=k:
            p[k]=(k,maxrow)
            U=row_interchange(U,k,maxrow)

        for j in range(k+1,n):
            l[k][j][k]=(U[j][k]/U[k][k])
            for i in range(k,n):
                U[j][i]=U[j][i]-l[k][j][k]*U[k][i]


    #Multiplying permutation matrices with the lower triangualar matrices we got from ea

    for d in range(n-1):
        for c in range (d+1,n+1):
            l[d]=col_interchange(row_interchange(l[d],p[c][0],p[c][1]),p[c][0],p[c][1])

    #calculatitn P
    prod1=np.eye(n)
    prod=np.eye(n)
    for x in range(n):
        prod=np.dot(prod,l[x])
        prod1=row_interchange(prod1,p[x][1],p[x][0])

    L=prod
    P=prod1
    return P,L,U
```

```python
def solution(A,b):
    if type(LUFACT(A))=="str":                          #stops the algorithm if the input
        return ("input is out of scope of this algorithm")
    P,L,U=LUFACT(A)
    b=np.dot(P,b)
    b1=forward_sub(L,b)
    sol=backward_sub(U,b1)
    return sol

def forward_sub(a,b):
    n=len(a)
    y=np.zeros((n,1))
    for i in range(n):
        k=0
        y[i][0]=b[i][0]
        for j in range(i):
            k+=a[i][j]*y[j][0]
        y[i][0]=(y[i][0]-k)
    return y


def backward_sub(a,b):
    n=len(a)
    x=np.zeros((n,1))
    for i in range(n-1,-1,-1):
        k=0
        x[i][0]=b[i][0]
        for j in range(i+1,n):
            k+=a[i][j]*x[j][0]
        x[i][0]=(x[i][0]-k)/a[i][i]
    return x
```

In [4]:
```python
def matrix_generator(a,c):              #argument "a" is for the number of matrices and "b"
    matrices=[]
    for i in range(a):

        k=random.randint(1,c)           #generating random matrix sizes

        A=100*np.random.rand(k,k)      #genearting coefficient matrix,constant matrices wi
        b=100*np.random.rand(k,1)
        matrices.append([A,b])
    return matrices
```

In [5]:
```python
def measure_timeSOL(L):
    time1=[]
    scipy_time=[]
    for x in L:

        start1=time.time()
        p=solution(x[0],x[1])
        end1=time.time()
        t1=end1-start1
        time1.append(t1)

        start2=time.time()
        p=solve(x[0],x[1])
        end2=time.time()
        t2=end2-start2
        scipy_time.append(t2)
    return time1,scipy_time
```

In [6]:
```
def measure_timeLU(L):
    time1=[]
    scipy_time=[]
    for x in L:

        start1=time.time()
        p=LUFACT(x[0])
        end1=time.time()
        t1=end1-start1
        time1.append(t1)

        start2=time.time()
        p=lu(x[0])
        end2=time.time()
        t2=end2-start2
        scipy_time.append(t2)
    return time1,scipy_time
```

RESULTS

First let us run the helper function file.

In [1]:
```
%run helper_functions.ipynb
```

Using the matrix generator function we've created,we can generate desired number of random matrices.

In [2]:
```
M=matrix_generator(15,100)
```

Now that we've the list of randomly created matrices let us calculate the LU factorization of the matrices,the difference between PA and LU,Solution of AX=b,and the difference betwween AX and b.

In [3]:
```
normLU=[]
normSOL=[]
for x in M:
    P1,L1,U1=LUFACT(x[0])
    q=np.dot(P1,x[0])
    r=np.dot(L1,U1)
    n=np.linalg.norm(q-r)
    normLU.append(n)



    y=solution(x[0],x[1])
    m=np.dot(x[0],y)
    z=np.linalg.norm(m-x[1])
    normSOL.append(z)

    #print("Size of the Matrix:",len(x))
    #print("P:\n",P1,"\n")
    #print("L:\n",L1,"\n")
    #print("U:\n",U1,"\n")
    #print("PA-LU is :\n",q-r,"\n")

    #print("Solution to AX=b is:\n",y)

print("Norms of PA-LU is:\n",normLU)
print("Norms of AX-b is:\n",normSOL)
```

```
Norms of PA-LU is:
 [1.2917348974815226e-12, 2.2061114442076724e-12, 1.0587608382197967e-12, 1.104176593309
202e-12, 2.408824824607806e-12, 3.4891090523611284e-13, 5.697490303155319e-13, 2.3705298
79295217e-13, 1.8147882078901266e-12, 1.7290261366949012e-13, 1.8888818762248754e-12, 7.
78360544769648e-13, 0.0, 5.624352792277627e-13, 9.646294471829118e-13]
Norms of AX-b is:
 [3.341450722137136e-11, 1.8644260642387194e-11, 1.1515979044762974e-12, 1.1977186432104
14e-12, 6.6205559648287545e-12, 2.4815687562588183e-13, 6.609629022663084e-13, 5.0179644
05823497e-13, 1.8280402488730182e-12, 1.602669906413122e-13, 4.069787837625156e-12, 1.52
6146152194083e-12, 1.4210854715202004e-14, 1.7563788883643145e-12, 2.1781036994377764e-1
2]
```

Now,let us calculate the same for the inbuilt function in Scipy

In [4]:
```
sci_normLU=[]
sci_normSOL=[]
size=[]
for x in M:
```

```
    size.append(len(x[0]))
    P1,L1,U1=lu(x[0])
    q=np.dot(P1,x[0])
    r=np.dot(L1,U1)
    n=np.linalg.norm(q-r)
    sci_normLU.append(n)



    y=solve(x[0],x[1])
    m=np.dot(x[0],y)
    z=np.linalg.norm(m-x[1])
    sci_normSOL.append(z)

    #print("Size of the Matrix:",len(x[0]))
    #print("P:\n",P1,"\n")
    #print("L:\n",L1,"\n")
    #print("U:\n",U1,"\n")
    #print("PA-LU is :\n",q-r,"\n")

    #print("Solution to AX=b is:\n",y)

print("Norms of PA-LU is:\n",sci_normLU)
print("Norms of AX-b is:\n",sci_normSOL)
```

```
Norms of PA-LU is:
 [2557.2852831088107, 3568.5418145349745, 2308.2833095777924, 2329.021155330863, 3832.23
35613025525, 1207.0661638502777, 1626.487246066234, 929.0850187356434, 3056.95952975964
1, 717.0071241935564, 3239.942395628104, 1864.0318151622005, 1.7763568394002505e-15, 164
4.012668814204, 2182.4761502057054]
Norms of AX-b is:
 [2.1055697730753597e-11, 2.1673102355754512e-11, 1.0024107717115615e-12, 7.856010923420
094e-13, 4.995415996888695e-12, 3.5020848058227755e-13, 7.774240471697501e-13, 3.2009191
19439219e-13, 1.5051905723307699e-12, 1.704568041361993e-13, 3.666499945758256e-12, 1.47
6237234047517e-12, 5.859285502108464e-14, 1.627469899365532e-12, 2.091348161237309e-12]
```

Time taken to solve Ax=b

In [5]:

```
my_time,scipy_time=measure_timeSOL(M)
print("Time taken by my code:\n",my_time)
print("Time taken by Scipy:\n",scipy_time)
```

```
Time taken by my code:
 [0.27573251724243164, 0.609276533126831, 0.25189757347106934, 0.20448851585388184, 0.78
10640335083008, 0.031244516372680664, 0.07810378074645996, 0.015621900558471680, 0.453064
2032623291, 0.015580654144287111, 0.5154988765716553, 0.14060282707214355, 0.0, 0.0780944
82421875, 0.15621328353881836]
Time taken by Scipy:
 [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, 0.0]
```

Time taken to do LU Factorization

In [6]:

```
my_time1,scipy_time1=measure_timeLU(M)
print("Time taken by my code:\n",my_time1)
print("Time taken by Scipy:\n",scipy_time1)
```

```
Time taken by my code:
 [0.14969968795776367, 0.32289552688598633, 0.10930538177490234, 0.09372544288635254, 0.
37672924995422363, 0.015627145767211914, 0.031237363815307617, 0.015623092651367188, 0.2
4931573867797852, 0.0, 0.24989652633666992, 0.0624845027923584, 0.0, 0.0312423706054687
5, 0.1249699592590332]
Time taken by Scipy:
 [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, 0.0]
```

Table

In [11]:
```
Gaussian=pd.DataFrame()
Gaussian["n"]=size
Gaussian["Norm of PA-LU using my code"]=normLU
Gaussian["Norm of Ax-b using my code"]=normSOL
Gaussian["Norm of PA-LU using Scipy"]=sci_normLU
Gaussian["Norm of Ax-b using Scipy"]=sci_normSOL
Gaussian["Time taken to do LU factorozation using my code"]=my_time1
Gaussian["Time taken to do LU factorozation using Scipy"]=scipy_time1
Gaussian["Time taken solve Ax=b using my code"]=my_time
Gaussian["Time taken solve Ax=b using Scipy"]=scipy_time
Gaussian.sort_values(by="n")
```

Out[11]:

	n	Norm of PA-LU using my code	Norm of Ax-b using my code	Norm of PA-LU using Scipy	Norm of Ax-b using Scipy	Time taken to do LU factorozation using my code	Time taken to do LU factorozation using Scipy	Time taken solve Ax=b using my code	Time taken solve Ax=b using Scipy
12	2	0.000000e+00	1.421085e-14	1.776357e-15	5.859286e-14	0.000000	0.0	0.000000	0.0
9	18	1.729026e-13	1.602670e-13	7.170071e+02	1.704568e-13	0.000000	0.0	0.015581	0.0
7	24	2.370530e-13	5.017964e-13	9.290850e+02	3.200919e-13	0.015623	0.0	0.015622	0.0
5	30	3.489109e-13	2.481569e-13	1.207066e+03	3.502085e-13	0.015627	0.0	0.031245	0.0
13	40	5.624353e-13	1.756379e-12	1.644013e+03	1.627470e-12	0.031242	0.0	0.078094	0.0
6	41	5.697490e-13	6.609629e-13	1.626487e+03	7.774240e-13	0.031237	0.0	0.078104	0.0
11	48	7.783605e-13	1.526146e-12	1.864032e+03	1.476237e-12	0.062485	0.0	0.140603	0.0
14	54	9.646294e-13	2.178104e-12	2.182476e+03	2.091348e-12	0.124970	0.0	0.156213	0.0
3	58	1.104177e-12	1.197719e-12	2.329021e+03	7.856011e-13	0.093725	0.0	0.204489	0.0
2	59	1.058761e-12	1.151598e-12	2.308283e+03	1.002411e-12	0.109305	0.0	0.251898	0.0
0	64	1.291735e-12	3.341451e-11	2.557285e+03	2.105570e-11	0.149700	0.0	0.275733	0.0
8	76	1.814788e-12	1.828040e-12	3.056960e+03	1.505191e-12	0.249316	0.0	0.453064	0.0
10	80	1.888882e-12	4.069788e-12	3.239942e+03	3.666500e-12	0.249897	0.0	0.515499	0.0
1	87	2.206111e-12	1.864426e-11	3.568542e+03	2.167310e-11	0.322896	0.0	0.609277	0.0
4	94	2.408825e-12	6.620556e-12	3.832234e+03	4.995416e-12	0.376729	0.0	0.781064	0.0

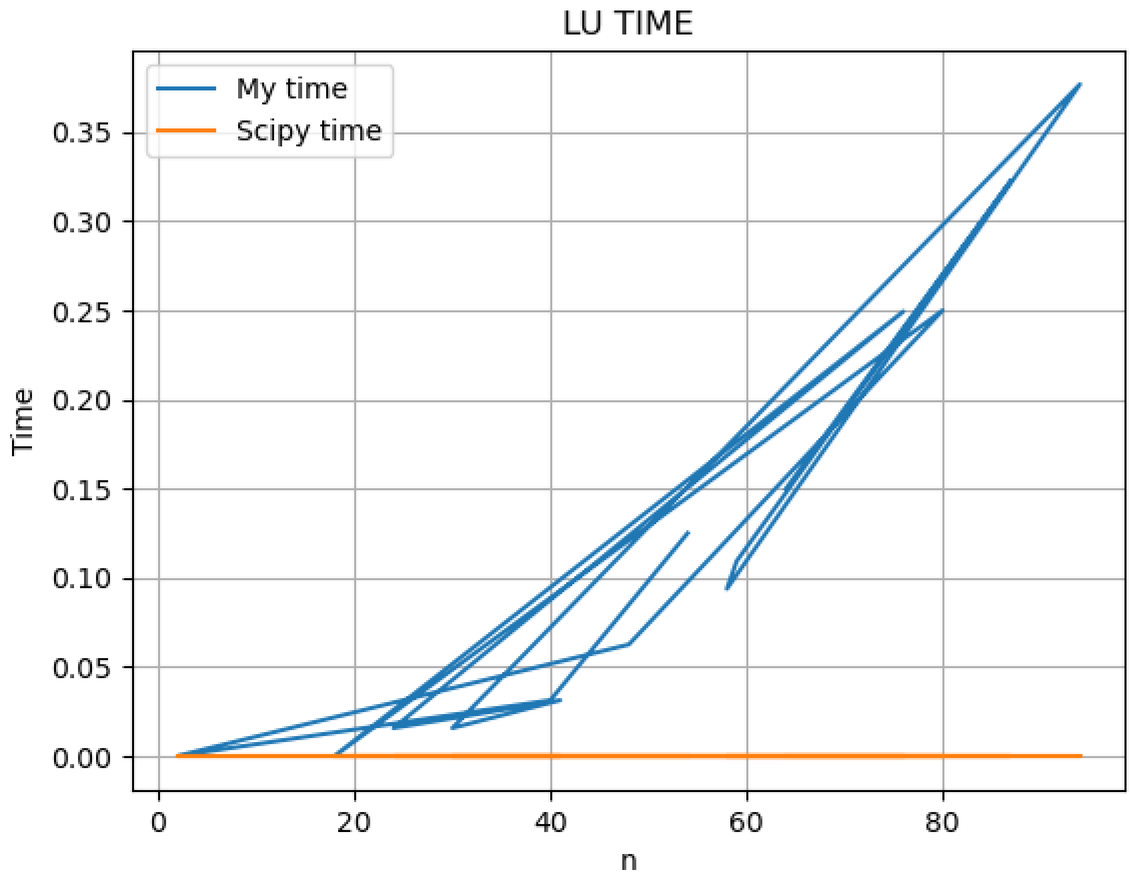
In [8]:
```
Gaussian
```

Out[8]:

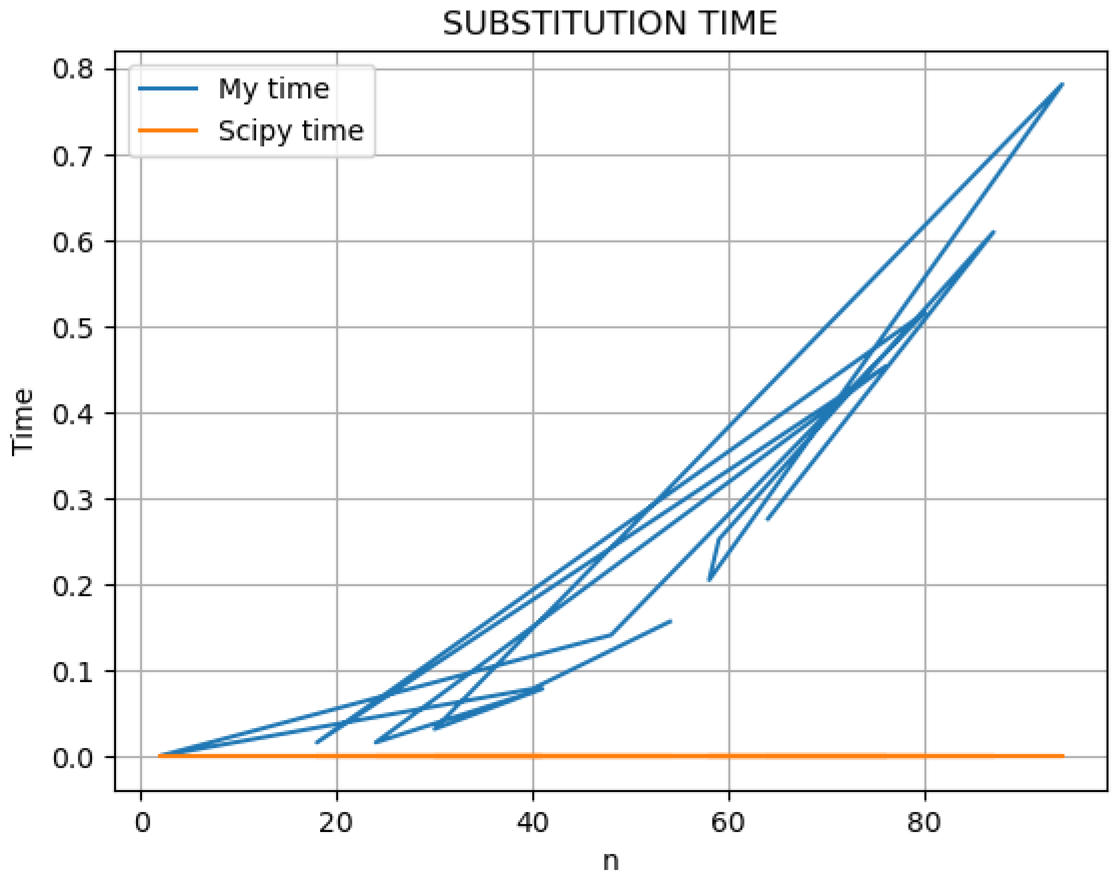
	n	Norm of PA-LU using my code	Norm of Ax-b using my code	Norm of PA-LU using Scipy	Norm of Ax-b using Scipy	Time taken to do LU factorozation using my code	Time taken to do LU factorozation using Scipy	Time taken solve Ax=b using my code	Time taken solve Ax=b using Scipy
0	64	1.291735e-12	3.341451e-11	2.557285e+03	2.105570e-11	0.149700	0.0	0.275733	0.0
1	87	2.206111e-12	1.864426e-11	3.568542e+03	2.167310e-11	0.322896	0.0	0.609277	0.0
2	59	1.058761e-12	1.151598e-12	2.308283e+03	1.002411e-12	0.109305	0.0	0.251898	0.0
3	58	1.104177e-12	1.197719e-12	2.329021e+03	7.856011e-13	0.093725	0.0	0.204489	0.0
4	94	2.408825e-12	6.620556e-12	3.832234e+03	4.995416e-12	0.376729	0.0	0.781064	0.0
5	30	3.489109e-13	2.481569e-13	1.207066e+03	3.502085e-13	0.015627	0.0	0.031245	0.0
6	41	5.697490e-13	6.609629e-13	1.626487e+03	7.774240e-13	0.031237	0.0	0.078104	0.0
7	24	2.370530e-13	5.017964e-13	9.290850e+02	3.200919e-13	0.015623	0.0	0.015622	0.0
8	76	1.814788e-12	1.828040e-12	3.056960e+03	1.505191e-12	0.249316	0.0	0.453064	0.0
9	18	1.729026e-13	1.602670e-13	7.170071e+02	1.704568e-13	0.000000	0.0	0.015581	0.0
10	80	1.888882e-12	4.069788e-12	3.239942e+03	3.666500e-12	0.249897	0.0	0.515499	0.0
11	48	7.783605e-13	1.526146e-12	1.864032e+03	1.476237e-12	0.062485	0.0	0.140603	0.0
12	2	0.000000e+00	1.421085e-14	1.776357e-15	5.859286e-14	0.000000	0.0	0.000000	0.0
13	40	5.624353e-13	1.756379e-12	1.644013e+03	1.627470e-12	0.031242	0.0	0.078094	0.0
14	54	9.646294e-13	2.178104e-12	2.182476e+03	2.091348e-12	0.124970	0.0	0.156213	0.0

In [12]:

```
# Plot the change in values of the two variables
plt.plot(Gaussian["n"], Gaussian["Time taken to do LU factorozation using my code"], lab
plt.plot(Gaussian["n"],Gaussian["Time taken to do LU factorozation using Scipy"] , label

# Add labels and title
plt.xlabel('n')
plt.ylabel('Time')
plt.title('LU TIME')
plt.legend()   # Add legend

# Show the plot
plt.grid(True)   # Add grid
plt.show()
```



In [13]:
```
plt.plot(Gaussian["n"], Gaussian["Time taken solve Ax=b using my code"], label='My time'
plt.plot(Gaussian["n"],Gaussian["Time taken solve Ax=b using Scipy"] , label='Scipy time

plt.xlabel('n')
plt.ylabel('Time')
plt.title('SUBSTITUTION TIME')
plt.legend()  # Add legend

# Show the plot
plt.grid(True)  # Add grid
plt.show()
```


SUBSTITUTION TIME
My time
Scipy time
Time
n
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80

In [ ]: