KLE Technological University | Creating Value, Leveraging Knowledge

DEPARTMENT OF MATHEMATICS

Mathematics Lab Assessment No.3

Name: Anup Wadekar
Division: S
SRN: 01FE24BAR008
Date: 13/11/2025

Question

1) Find the displacements u_1, u_2, u_3, u_4 using **Gauss Elimination Method**.

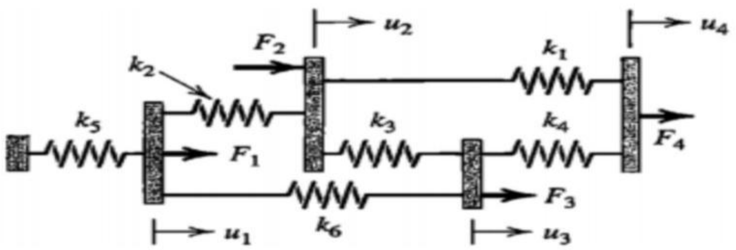


Fig 1: Springs and blocks arrangement

Springs and blocks arrangement is shown in Fig 1. The displacement of springs allowed in horizontal direction only and the blocks are considered as rigid and connected linear springs with stiffness. Write the system of equilibrium of equations for the applied forces **$F_1=2F_2=20$ kN, $F_3=F_4=30$ kN.** Evaluate the displacement fields for the applied loads

$$k_1 = 100\ N/m \quad k_2 = 200\ N/m \quad k_3 = 300\ N/m$$
$$k_4 = 500\ N/m \quad k_5 = 400\ N/m \quad k_6 = 150\ N/m$$

1. Solve the given system using **LU Decomposition Method**:

$$\begin{aligned} x_1 - 3x_2 + x_3 &= -5 \\ 2x_1 - 2x_2 + 4x_3 &= 0 \\ 3x_1 + 2x_2 + 5x_3 &= 7 \end{aligned}$$

Solution

i) Identify the parameters and mathematical concept :

System Type:
Linear spring–block system with one-dimensional horizontal motion.
Parameters:

- Rigid blocks connected by linear springs
- Unknown displacements: u_1, u_2, u_3, u_4
- Known spring stiffness values and applied forces

Mathematical Concept:
The system satisfies static equilibrium and follows Hooke's law.
The governing stiffness equation is:

$$[K]\{u\} = \{F\}$$

Governing Equations

$$Ax = b$$

Using LU decomposition:

$$A = LU$$
$$LUx = b$$

ii) Solve analytically

Given data

$k_1 = 100\ N/m \quad k_2 = 200\ N/m$

$k_3 = 300\ N/m \quad k_4 = 500\ N/m$

$k_5 = 400\ N/m \quad k_6 = 150\ N/m$

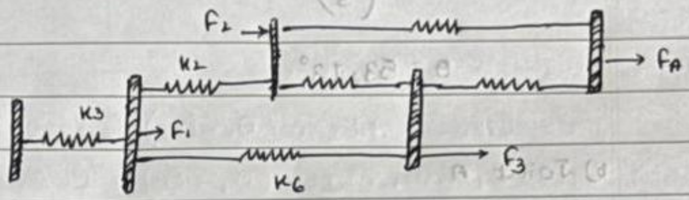
Applied force

$F_1 = 20 \quad F_2 = 10$

$F_3 = 30 \quad F_4 = 80$

Displacement Vector

$$\{U\} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$



node wise Equilibrium eqn.

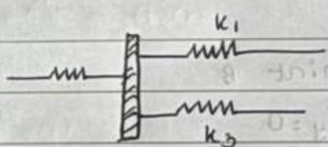
Node 1, $F_1 = (k_2 + k_5 + k_6)U_1 - k_2 U_2 - k_6 U_3$

node 2, $F_2 = -k_2 U_1 + (k_1 + k_2 + k_3) U_2 - k_3 U_3 - k_1 U_4$

node 3, $F_3 = -k_6 U_1 - k_3 U_2 + (k_3 + k_4 + k_6) U_3 - k_4 U_4$.

node 4, $F_4 = -k_1 U_2 - k_4 U_3 + (k_1 + k_4) U_4$

Global stiffness matrix



$U_1 = 0.225$.

$U_2 = 0.402$.

$U_3 = 0.455$.

$U_4 = 0.497$

Final displacement Vector

$$\{U\} = \begin{bmatrix} 0.225 \\ 0.402 \\ 0.455 \\ 0.497 \end{bmatrix}$$

iii) GeoGebra Screenshot / Program Execution

1]

```
import numpy as np
#B=np.array([[2,1,1,10],
#            [3,2,3,18],
#             [1,4,9,16]],
#             dtype=float)

A=np.array([[ 750, -200, -150,    0,20],
            [-200,  600, -300, -100,10],
            [-150, -300,  950, -500,30],
            [   0, -100, -500,  600,30]],dtype=float)

n=len(A)

for i in range(n):
    pivot=A[i][i]
    A[i]=A[i]/pivot
    for j in range (i+1,n):
        A[j]=A[j]-A[j][i]*A[i]

x=np.zeros(n)

for i in range (n-1,-1,-1):
    x[i]=A[i][n]-np.sum(A[i][i+1:n]*x[i+1:n])

print("SOLUTIONS:")
print("x=",x[0])
print("y=",x[1])
print("z=",x[2])
print("a=",x[3])
```

2]

```python
import numpy as np

def lu_decompose(A):
    A = A.astype(float)
    n = A.shape[0]
    L = np.eye(n)
    U = A.copy()

    for i in range(n):
        for j in range(i+1, n):
            factor = U[j, i] / U[i, i]
            L[j, i] = factor
            U[j] = U[j] -factor * U[i]
    return L, U

def solve_LU(A, b):
    L, U = lu_decompose(A)
    n = A.shape[0]

    y = np.linalg.solve(L,b)
    x = np.linalg.solve(U,y)

    return x

A = np.array([[1, -3, 1],
              [0,4, 2],
              [0, 0, -6]])

b = np.array([-5, 10, -18])

x = solve_LU(A, b)
print("Solution x:", x)
```

i) Results and analysis from the graph

OUTPUT

1)

```
a= 0.22500000000000002
b= 0.402157360406609174
c= 0.4554568527918786
d= 0.4965736040609141
```

Gauss Elimination Output

$$\boxed{u_1 = 0.2250,\ u_2 = 0.4021,\ u_3 = 0.4554,\ u_4 = 0.4965}$$

2)

```
L:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
U:
 [[ 1. -3.  1.]
 [ 0.  4.  2.]
 [ 0.  0. -6.]]
Solution x:
 [-5.  1.  3.]
```

LU Decomposition Output

$$\boxed{\boldsymbol{x_1 = -5,\ x_2 = 1,\ x_3 = 3}}$$

iii) GeoGebra Screenshot / Program Execution

1]

```
import numpy as np
#B=np.array([[2,1,1,10],
#            [3,2,3,18],
#             [1,4,9,16]],
#             dtype=float)

A=np.array([[ 750, -200, -150,    0,20],
            [-200,  600, -300, -100,10],
            [-150, -300,  950, -500,30],
            [   0, -100, -500,  600,30]],dtype=float)

n=len(A)

for i in range(n):
    pivot=A[i][i]
    A[i]=A[i]/pivot
    for j in range (i+1,n):
        A[j]=A[j]-A[j][i]*A[i]

x=np.zeros(n)

for i in range (n-1,-1,-1):
    x[i]=A[i][n]-np.sum(A[i][i+1:n]*x[i+1:n])

print("SOLUTIONS:")
print("x=",x[0])
print("y=",x[1])
print("z=",x[2])
print("a=",x[3])
```

	2]	

```
import numpy as np

def lu_decompose(A):
    A = A.astype(float)
    n = A.shape[0]
    L = np.eye(n)
    U = A.copy()

    for i in range(n):
        for j in range(i+1, n):
            factor = U[j, i] / U[i, i]
            L[j, i] = factor
            U[j] = U[j] -factor * U[i]
    return L, U

def solve_LU(A, b):
    L, U = lu_decompose(A)
    n = A.shape[0]

    y = np.linalg.solve(L,b)
    x = np.linalg.solve(U,y)

    return x

A = np.array([[1, -3, 1],
              [0,4, 2],
              [0, 0, -6]])

b = np.array([-5, 10, -18])

x = solve_LU(A, b)
print("Solution x:", x)
```