Engineering Analysis Homework 2

September 12, 2023

Gabe Morris

[1]:
```
# Notebook Preamble
from eng_analysis import gauss_solve, choleski_decomposition, choleski_solve,␣
 ↪lu_decomposition, lu_solve, conj_grad
import numpy as np
import sympy as sp
```

Important: I am creating a python package for this class. You might not use it in the future as I believe one of your primary goals is for students to figure out how to run the code, but with this package, everything will be organized appropriately, and you'll be able to install via `pip install eng_analysis`. There are many issues with the existing code in the book from a practical standpoint and from neglect of code etiquette.

The project and the code will be changed significantly as we continue with the class. The content may be found here.

Contents

1 Problem 1

Solve the equations $Ax = b$ using Gauss elimination for

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 2 & -1 \\ -1 & 2 & -1 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

1.1 Solution

Remember, the code for the package is found here.

[2]:
```
A = [
    [2, -1, 0, 0],
    [0, 0, -1, 1],
    [0, -1, 2, -1],
    [-1, 2, -1, 0]
]
b = [1, 0, 0, 0]

x = gauss_solve(A, b)
x
```

[2]:
```
array([1., 1., 1., 1.])
```

One important contribution from the `eng_analysis` package is that the datatype for the input does not matter. Also, a huge mistake with the existing source code was that it was not using copies of the numpy arrays, which means that it was changing them in place. This is undesirable as the matrices A and b could be used later after getting the solution.

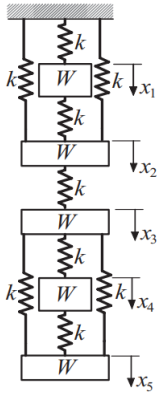
The solution can be verified using `numpy`.

[3]:
```
np.linalg.solve(A, b)
```

[3]:
```
array([1., 1., 1., 1.])
```

2 Problem 2

Find the displacement for each of the weights in the following static systems of weights and springs with



2.1 Solution

The system of equations are shown from the below code cell.

[4]:
```
x = x1, x2, x3, x4, x5 = sp.symbols('x1:6')
W, k = sp.symbols('W k')

eq1 = sp.Eq(W - k*x1 + k*(x2 - x1), 0)
eq2 = sp.Eq(W + -2*k*x2 + k*(x1 - x2) + k*(x3 - x2), 0)
eq3 = sp.Eq(W + k*(x2 - x3) + k*(x4 - x3) + 2*k*(x5 - x3), 0)
eq4 = sp.Eq(W + k*(x3 - x4) + k*(x5 - x4), 0)
eq5 = sp.Eq(W + k*(x4 - x5) + 2*k*(x3 - x5), 0)
display(*[eval(f'eq{i}') for i in range(1, 6)])
```

$$W - kx_1 + k\left(-x_1 + x_2\right) = 0$$

$$W - 2kx_2 + k\left(x_1 - x_2\right) + k\left(-x_2 + x_3\right) = 0$$

$$W + k\left(x_2 - x_3\right) + k\left(-x_3 + x_4\right) + 2k\left(-x_3 + x_5\right) = 0$$

$$W + k\left(x_3 - x_4\right) + k\left(-x_4 + x_5\right) = 0$$

$$W + 2k\left(x_3 - x_5\right) + k\left(x_4 - x_5\right) = 0$$

`sympy` can put the system into the matrix form as seen below.

[5]:
```
A, b = sp.linear_eq_to_matrix([eq1, eq2, eq3, eq4, eq5], x)
display(A, b)
```

$$\begin{bmatrix} -2k & k & 0 & 0 & 0 \\ k & -4k & k & 0 & 0 \\ 0 & k & -4k & k & 2k \\ 0 & 0 & k & -2k & k \\ 0 & 0 & 2k & k & -3k \end{bmatrix}$$

$$\begin{bmatrix} -W \\ -W \\ -W \\ -W \\ -W \end{bmatrix}$$

From the construction of the matrices above, it is apparent that A is symmetric. The Choleski method is a good choice to solve this system. Since the solution vector (b) is all the same value, we can take its value to be 1. It is understood that the output of the choleski solution would then be in terms of W/k.

[6]:
```
A = [
    [2, -1, 0, 0, 0],
    [-1, 4, -1, 0, 0],
    [0, -1, 4, -1, -2],
    [0, 0, -1, 2, -1],
    [0, 0, -2, -1, 3]
]
b = [1, 1, 1, 1, 1]
L = choleski_decomposition(A)
x = choleski_solve(L, b)
x
```

[6]:
```
array([1.4, 1.8, 4.8, 5.6, 5.4])
```

[7]:
```
# Verification using numpy
np.linalg.solve(A, b)
```

[7]:
```
array([1.4, 1.8, 4.8, 5.6, 5.4])
```

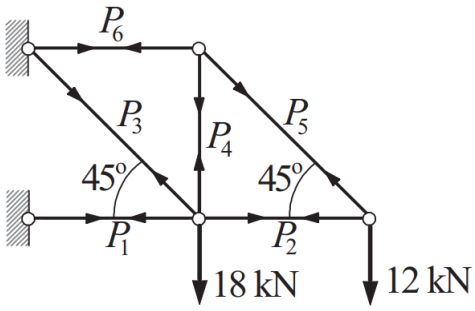
If W was 25 lbs and k was 50 lbs/in then the displacements would be

[8]:
```
x*25/50  # inches
```

[8]:
```
array([0.7, 0.9, 2.4, 2.8, 2.7])
```

3 Problem 3

Consider the static truss below. Find the member force P_n in each support.



3.1 Solution

The system of equations can be found by using the method of joints on the three points that are not the fixed/grounded points (the reactions).

[9]:
```
p = p1, p2, p3, p4, p5, p6 = sp.symbols('P1:7')

eq1 = sp.Eq(p2 - p1 - p3*sp.cos(sp.pi/4), 0)
eq2 = sp.Eq(p4 - 18 + p3*sp.sin(sp.pi/4), 0)
eq3 = sp.Eq(-p2 - p5*sp.cos(sp.pi/4), 0)
eq4 = sp.Eq(p5*sp.sin(sp.pi/4), 12)
eq5 = sp.Eq(p5*sp.cos(sp.pi/4) - p6, 0)
eq6 = sp.Eq(-p4 - p5*sp.sin(sp.pi/4), 0)
display(eq1, eq2, eq3, eq4, eq5, eq6)
```

$$-P_1 + P_2 - \frac{\sqrt{2}P_3}{2} = 0$$

$$\frac{\sqrt{2}P_3}{2} + P_4 - 18 = 0$$

$$-P_2 - \frac{\sqrt{2}P_5}{2} = 0$$

$$\frac{\sqrt{2}P_5}{2} = 12$$

$$\frac{\sqrt{2}P_5}{2} - P_6 = 0$$

$$-P_4 - \frac{\sqrt{2}P_5}{2} = 0$$

[10]:
```
A, b = sp.linear_eq_to_matrix([eq1, eq2, eq3, eq4, eq5, eq6], p)
display(A, b)
```

$$\begin{bmatrix} -1 & 1 & -\frac{\sqrt{2}}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & -1 \\ 0 & 0 & 0 & -1 & -\frac{\sqrt{2}}{2} & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 18 \\ 0 \\ 12 \\ 0 \\ 0 \end{bmatrix}$$

The format of this matrix is not symmetric, so LU decomposition is a valid option.

[11]:
```
LU, seq = lu_decomposition(np.array(A))
x = lu_solve(LU, np.array(b).reshape((6,)), seq)
x
```

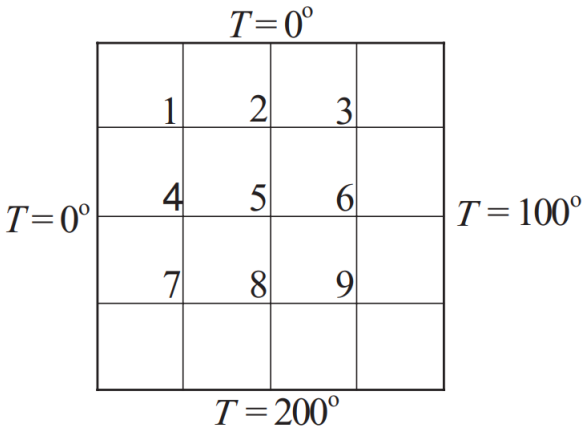
[11]:
```
array([-42.        , -12.        ,  42.42640687, -12.        ,
        16.97056275,  12.        ])
```

[12]:
```
# Verification
np.linalg.solve(np.array(A, dtype=np.float64), np.array(b, dtype=np.float64).
 ↪reshape((6,)))
```

[12]:
```
array([-42.        , -12.        ,  42.42640687, -12.        ,
        16.97056275,  12.        ])
```

4 Problem 4

Consider a square, thermally conducting plate with its edges kept at constant temperature as shown below.



Assume the heat equation governing the temperature, T, at each point in the plate in the steady state is given by:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

By dividing the plate into a 4 x 4 grid of squares and rewriting the above as a numerical derivative, find the temperature of each of the squares using (a) a direct method of your choice and (b) an iterative method of your choice. (c) What is the temperature at the center of the square? How does your prediction change as you go from a 4 x 4 to a 6 x 6 to an 8 x 8 grid?

4.1 Solution

The solution can be obtained by getting the second order derivative approximation for x and y, then superimposing both results. The final result is

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 100 \\ 0 \\ 0 \\ 100 \\ 200 \\ 200 \\ 300 \end{bmatrix}$$

4.1.1 Part A

Since this system is symmetric and appears to be positive definite, the Choleski method is a valid choice.

[13]:
```
A = [
    [4, -1, 0, -1, 0, 0, 0, 0, 0],
    [-1, 4, -1, 0, -1, 0, 0, 0, 0],
    [0, -1, 4, 0, 0, -1, 0, 0, 0],
    [-1, 0, 0, 4, -1, 0, -1, 0, 0],
    [0, -1, 0, -1, 4, -1, 0, -1, 0],
    [0, 0, -1, 0, -1, 4, 0, 0, -1],
    [0, 0, 0, -1, 0, 0, 4, -1, 0],
    [0, 0, 0, 0, -1, 0, -1, 4, -1],
    [0, 0, 0, 0, 0, -1, 0, -1, 4]
]
b = [0, 0, 100, 0, 0, 100, 200, 200, 300]
L = choleski_decomposition(A)
x = choleski_solve(L, b)
x
```

[13]:
```
array([ 21.42857143,  38.39285714,  57.14285714,  47.32142857,
        75.        ,  90.17857143,  92.85714286, 124.10714286,
       128.57142857])
```

4.1.2 Part B

The matrix is sparse, so the conjugate gradient method is a good choice.

[14]:
```
Av = lambda x_: np.matmul(A, x_)
x, iter_num = conj_grad(Av, np.zeros(len(A)), b)
x, iter_num
```

[14]:
```
(array([ 21.42857143,  38.39285714,  57.14285714,  47.32142857,
         75.        ,  90.17857143,  92.85714286, 124.10714286,
        128.57142857]),
 4)
```

4.1.3 Part C

The temperature at the center of the square is $T_5 = 75^\circ$. As for making the mesh finer, the solution becomes more accurate and the difference in temperature from one grid point to the next will decrease. Programmatically, the pattern of the matrix would need to extend to fit the greater number of grid points. The automation of this process is what makes up finite element analysis software.