EGR 103L – Fall 2019

Laboratory 6 - Linear Algebra

Marcus Deans (md374)
Lab Section 4, Tuesdays 11:45-2:35
27 October 2019

I understand and have adhered to all the tenets of the Duke Community Standard in completing every part of this assignment. I understand that a violation of any part of the Standard on any part of this assignment can result in failure of this assignment, failure of this course, and/or suspension from Duke University.

Contents

List of Figures

1 Based on Chapra Problem 8.3

$$\begin{bmatrix} 0 & -7 & 5 \\ 0 & 4 & -7 \\ -4 & 3 & -7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 40 \\ -30 \\ 50 \end{bmatrix}$$

The calculated solutions for x_i are:

$$x = \begin{bmatrix} -18.8793 \\ -4.4828 \\ 1.7241 \end{bmatrix}$$

The transpose and inverse of A are:

$$A' = \begin{bmatrix} 0 & 0 & -4 \\ -7 & 4 & 3 \\ 5 & -7 & -7 \end{bmatrix} \qquad \text{inv}(A) = \begin{bmatrix} 0.0603 & 0.2931 & -0.2500 \\ -0.2414 & -0.1724 & -0.0000 \\ -0.1379 & -0.2414 & -0.0000 \end{bmatrix}$$

The condition numbers of A are:
1-norm: 13.4310
2-norm: 7.0040
Frobenius norm: 8.2214
inf-norm: 8.4483

The condition numbers each relate to the accuracy of the relevant calculations. The larger the condition number, the less precise the calculation. Moreover, by analysing the base 10 logarithm of each of the respective condition numbers, the relevant number of significant figures that should be included can be identified. This is based on the matrix norm that is calculated by the relevant Python function. Specifically, the relative error of the norm of the computed solution can be as large as the relative error of the norm of the coeffficients of the respective matrix, multiplied by the condition number. Essentially, these condition numbers allow us to determine the precision that the matrix holds and the format of the values we are outputting.

2 Chapra Problem 8.10

The matrix equation can be written as:

$$\begin{bmatrix} \cos(30^o) & 0 & -\cos(60^o) & 0 & 0 & 0 \\ \sin(30^o) & 0 & \sin(60^o) & 0 & 0 & 0 \\ -\cos(30^o) & -1 & 0 & -1 & 0 & 0 \\ -\sin(30^o) & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & \cos(60^o) & 0 & 0 & 0 \\ 0 & 0 & -\sin(60^o) & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ H_2 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} F_{1,h} \\ F_{1,v} \\ F_{2,h} \\ F_{2,v} \\ F_{3,h} \\ F_{3,v} \end{bmatrix}$$

The solutions written into the text file are:

F1: −1.000000e+03
F2: +8.660254e+02
F3: −1.732051e+03
H2: +0.000000e+00
V2: +5.000000e+02
V3: +1.500000e+03

3 Chapra Problem 8.16

The scripts and plot are in the appendices.

4 Parameter Sweep 1

There was a linear relationship between the voltage drop between nodes 6 and 1 and the current from nodes 5 to 2. As the voltage drop increased, the current decreased correspondingly. This similarly corresponds to the electrical equations representing the underlying physics of the circuit, such as Ohm's Law, $V = IR$ where V is change in electrical potential in volts, I is current, and R is resistance in Ohms. These equations can presumably be interpreted via their absolute values as voltage is often considered to be negative. Therefore, as the voltage increases, the current must also increase correspondingly if the resistance is staying the same.

5 Parameter Sweep 2

The relationship between resistance between nodes 5 and 2 and the current from nodes 5 to 2 was found to be logarithmic. As resistance increased, current decreased in absolute value, although in a logarithmic as opposed to linear fashion. This similarly follows the relevant electrical formulas which dictate these values.

The graph of the relationship between the resistance between nodes 5 and 2 and the condition number of the coefficient matrix was an interesting graph. The base 10 logarithm of the condition number was specifically represented which led to an interesting graph. There was an initial dip in the value of the condition number followed by a continuous increase of a concave down fashion. This graph seems to be the most unique and assuredly merits further investigation.

A Codes and Output

A.1 Chapra 8.3

```
# -*- coding: utf-8 -*-
"""
[Chapra 8.3]
[Marcus Deans]
[20 October 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""

import numpy as np

A = np.array([[0,-7,5],[0,4,-7],[-4,3,-7]])
B = np.array([[40],[-30],[50]])
x = np.linalg.inv(A).dot(B)
inve = np.linalg.inv(A)
trans = np.matrix.transpose(A)
n1 = np.linalg.cond(A, 1)
n2 = np.linalg.cond(A, 2)
frob = np.linalg.cond(A, 'fro')
inom = np.linalg.cond(A, np.inf)
```

A.2 Chapra 8.10

```
# -*- coding: utf-8 -*-
"""
[Chapra 8.10]
[Marcus Deans]
[20 October 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""

import numpy as np
import math as m

A = np.array([[(m.cos(m.pi/6)),0,(-1*m.cos(m.pi/3)),0,0,0],
                [(m.sin(m.pi/6)),0,(m.sin(m.pi/3)),0,0,0],
                [(-1*m.cos(m.pi/6)),-1,0,-1,0,0],
                [(-1*m.sin(m.pi/6)),0,0,0,-1,0],
                [0,1,(m.cos(m.pi/3)),0,0,0],
                [0,0,(-1*m.sin(m.pi/3)),0,0,-1]])
B = np.array([[0],[-2000],[0],[0],[0],[0]])
x = np.linalg.inv(A).dot(B)
soln_vec = x[:,0]
names = ['F1', 'F2', 'F3', 'H2', 'V2', 'V3']
f = open("truss_data.txt","w+")
for y in range(len(names)):
    f.write("{}: {:+3e}\n".format(names[y],soln_vec[y]))
f.close()
```

A.3 Chapra 8.16

```python
# -*- coding: utf-8 -*-
"""
[Chapra 8.16]
[Marcus Deans]
[20 October 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""

import numpy as np
import math as m

def rotate_2d(ang, x, y):
    ang = m.radians(ang)
    coords = np.array([x,y])
    R = np.array([[(m.cos(ang)), (-1*m.sin(ang))],
                  [(m.sin(ang)),(m.cos(ang))]])
    new = np.dot(R, coords)
    return new[0], new[1]

if __name__ == "__main__":
    print(rotate_2d(360,np.array([1,2,3,4,5]),np.array([1,2,3,4,5])))
```

A.4 Creative

```python
# -*- coding: utf-8 -*-
"""
[Chapra 8.16]
[Marcus Deans]
[20 October 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""

import numpy as np
import matplotlib.pyplot as plt
from chapra_08_16 import rotate_2d

x = np.array([1, 3, 5, 7, 10, 13, 16, 19, 21, 23, 25, 27, 29])
y = np.array([0, 1, 0, 1, 0, 2, 1, 3, 2, 3, 2, 3, 2])
plt.figure(1).clf()
fig, ax = plt.subplots(num=1)

for k in np.arange(0, 360, 15):
    x2, y2 = rotate_2d(k, x, y)
    ax.plot(x2, y2,
            color=[(1+np.cos(2*np.pi*k/360))/2,
                   (1+np.cos(2*np.pi*(k+120)/360))/2,
                   (1+np.cos(2*np.pi*(k+240)/360))/2])
ax.axis('equal')
fig.tight_layout()

```

```
fig.savefig("Rotated.eps")
```

A.5 Sweep 1

```
# -*- coding: utf-8 -*-
"""
[Sweep 1]
[Marcus Deans]
[20 October 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
in creating this code.
Signed: [md374]
"""
import numpy as np
import matplotlib.pyplot as plt

soln_vec = []
n_vec = []
v=0
for t in range(0, 101):
    A = np.array([[1,1,1,0,0,0],
                  [0,-1,0,1,-1,0],
                  [0,0,-1,0,0,1],
                  [0,0,0,0,1,-1],
                  [0,10,-10,0,-15,-5],
                  [5,-10,0,-20,0,0]])
    B = np.array([[0],[0],[0],[0],[0],[v]])
    y = np.linalg.inv(A).dot(B)
    soln_vec.append(float(y[1]))
    v += (400/100)

fig, ax = plt.subplots(num = 1, clear = True)
xl = []
v=0
for x in range(0,101):
    xl.append(v)
    v += (400/100)
ax.plot(xl,soln_vec,'purple', markevery=5)
ax.grid(True)
ax.set_xlabel("Voltage Drop from Node 6 to 1 in Volts, $V$")
ax.set_ylabel("Current from Node 5 to 2 in Amperes, $A$")

fig.tight_layout()
fig.savefig("CurrentVoltage.eps")
if __name__ == "__main__":
    print(soln_vec)
    print(soln_vec[50])
```

A.6 Sweep 2

```
# -*- coding: utf-8 -*-
"""
[Sweep 2]
[Marcus Deans]
[20 October 2019]

I understand and have adhered to all the tenets of the Duke Community Standard
```

```
in creating this code.
Signed: [md374]
"""
import numpy as np
import matplotlib.pyplot as plt
import math as m
soln_vec = []
n_vec = []
xl = []
v=0
for t in range(0, 201):
    A = np.array([[1,1,1,0,0,0],
                  [0,-1,0,1,-1,0],
                  [0,0,-1,0,0,1],
                  [0,0,0,0,1,-1],
                  [0,v,-10,0,-15,-5],
                  [5,-v,0,-20,0,0]])
    B = np.array([[0],[0],[0],[0],[0],[200]])
    x = np.linalg.inv(A).dot(B)
    soln_vec.append(float(x[1]))
    n_vec.append(np.linalg.cond(A, 2))
    xl.append(v)
    v += (100/200)

fig, ax = plt.subplots(num = 1, clear = True)
ax.plot(xl,soln_vec,'orange', markevery=5)
ax.grid(True)
ax.set_xlabel("Resistance between Nodes 5 and 2, $\Omega$")
ax.set_ylabel("Current from Node 5 to 2 in Amperes, $A$")
fig.tight_layout()
fig.savefig("CurrentResist.eps")

fig, ax = plt.subplots(num = 2,clear=True)
n_new = []
for x in range(0,len(n_vec)):
    n_new.append(m.log10(n_vec[x]))
ax.plot(xl,n_new,'k-')
ax.set_xlabel("Resistance between Nodes 5 and 2, $\Omega$")
ax.set_ylabel("Base 10 Logarithm of Condition Number of Coefficient Matrix")
fig.tight_layout()
fig.savefig("ConditionResist.eps")
if __name__ == "__main__":
    print(xl[20])
    print(soln_vec[20])
    #print(n_vec)
```

B Figures

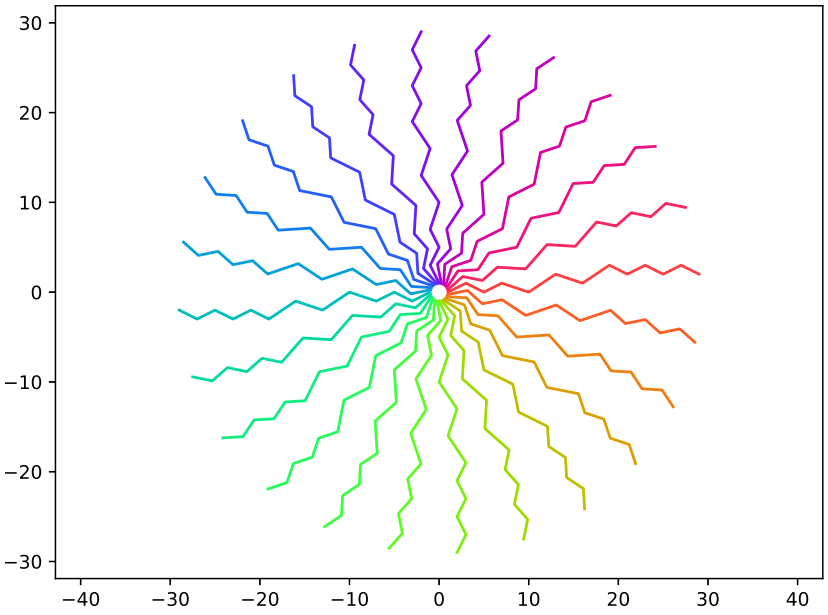

Figure 1: Creative Rotated Shape

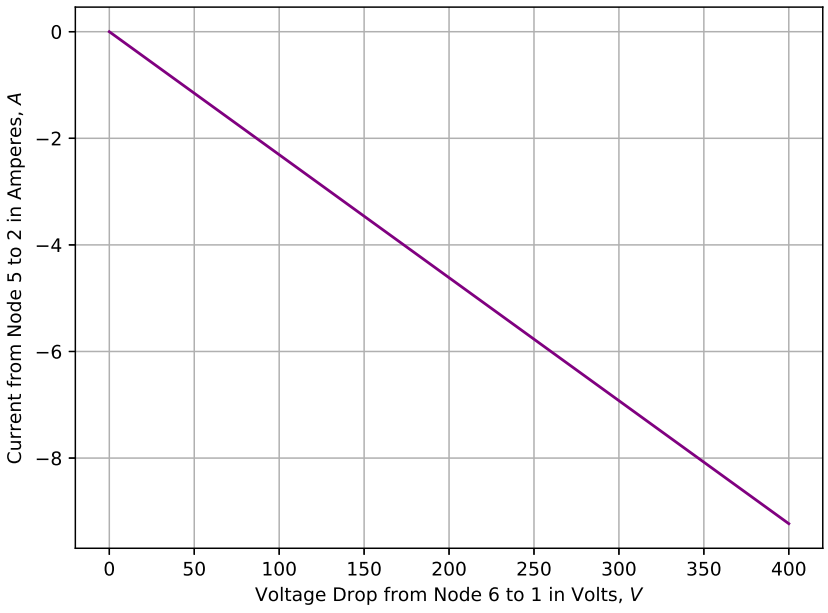

Figure 2: Current vs. Voltage

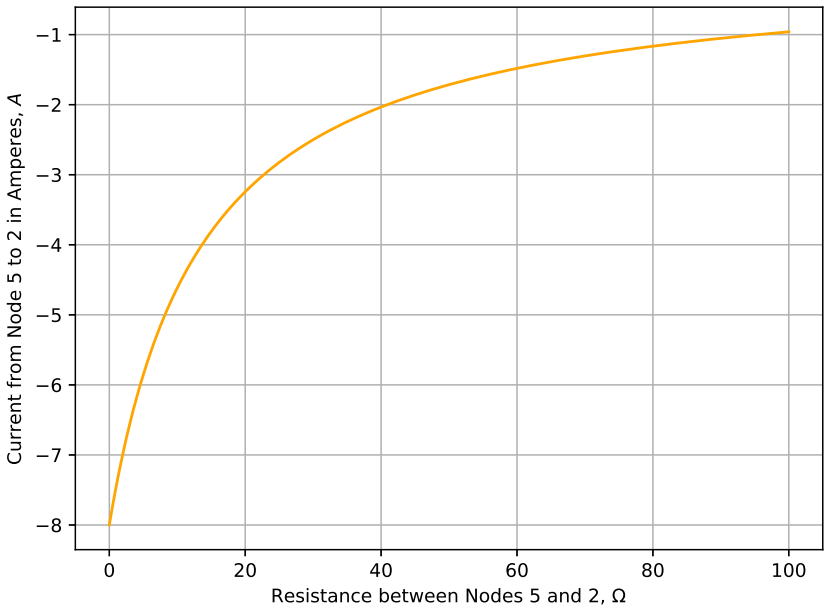

Figure 3: Current vs. Resistance

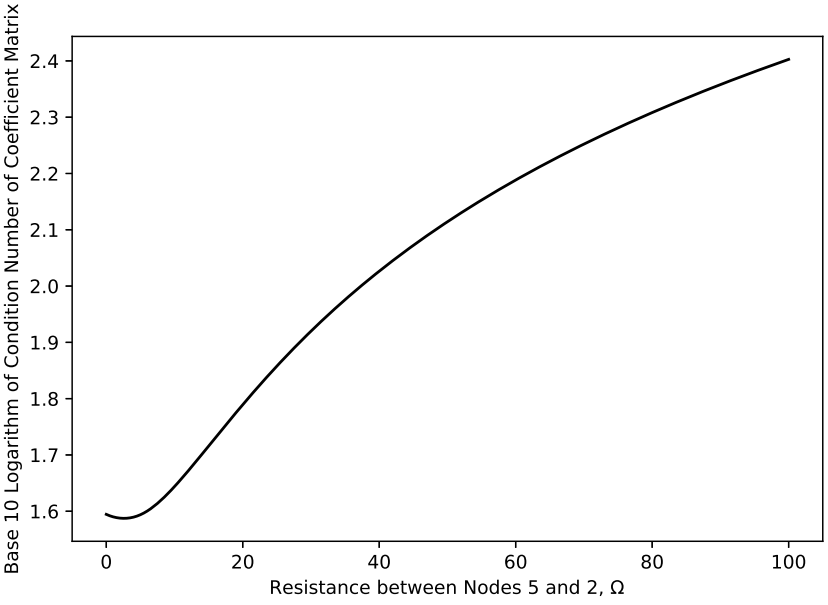

Figure 4: Condition Number vs. Resistance