Ryan Herrmann

Lab 6b

BMED 430

**Introduction**

The purpose of this lab was to repeat the same lab as 6a, but to use the Gauss-Sidel model through line by line solution. This part also used a heat source at a point on the rod to solve for a new temperature profile.

**Numerical Methods**

The numerical methods were to use the gauss-sidel method using the governing equation of and from there, using the finite difference form for the matrix becomes . This was used in an iterative approach.

**Pseudo- Code**

* Import required packages
* Define constants and input data
  + Sigfigs
  + Length of rod
  + Points on rod
  + Max error
  + Convergence criteria
  + Heat source
  + Position of source
  + Source length
* Create matrices
* Use gauss-sidel model in iterative approach.
* Add heat source
* Iterate again
* Graph outputs
* Save graph
* Save table

**Output**

The graph for the gauss-sidel solution without the heat source is given in Figure 1

A graph of different temperature and position

Description automatically generated

**Figure 1: Temperature vs Position for Both numerical methods**. The top shows linear decomp and the bottom shows the gauss sidel method.

Figure 2 shows the gauss-sidel method with the heat source added

A graph showing the temperature and position

Description automatically generated

**Figure 2: The solution for temperature versus position for an added heat source**

The temperature at each point is given in Table 1

**Table 1: Gauss-Sidel resulting temperature at each position**

|  |  |
| --- | --- |
| Position (m) | GS Temperature (C) |
| 0 | 0 |
| 0.0256 | 4.553 |
| 0.0513 | 9.106 |
| 0.0769 | 13.66 |
| 0.1026 | 18.213 |
| 0.1282 | 22.766 |
| 0.1538 | 27.319 |
| 0.1795 | 31.544 |
| 0.2051 | 35.44 |
| 0.2308 | 39.007 |
| 0.2564 | 42.245 |
| 0.2821 | 45.155 |
| 0.3077 | 47.736 |
| 0.3333 | 49.988 |
| 0.359 | 51.911 |
| 0.3846 | 53.835 |
| 0.4103 | 55.758 |
| 0.4359 | 57.682 |
| 0.4615 | 59.605 |
| 0.4872 | 61.528 |
| 0.5128 | 63.452 |
| 0.5385 | 65.376 |
| 0.5641 | 67.299 |
| 0.5897 | 69.223 |
| 0.6154 | 71.146 |
| 0.641 | 73.07 |
| 0.6667 | 74.993 |
| 0.6923 | 76.917 |
| 0.7179 | 78.84 |
| 0.7436 | 80.764 |
| 0.7692 | 82.688 |
| 0.7949 | 84.611 |
| 0.8205 | 86.535 |
| 0.8462 | 88.458 |
| 0.8718 | 90.382 |
| 0.8974 | 92.306 |
| 0.9231 | 94.229 |
| 0.9487 | 96.153 |
| 0.9744 | 98.076 |
| 1 | 100 |

The final results are shown in table 2 which includes the convergence criteria to stop and the absolute error.

**Table 2: Iterations with convergence criteria and absolute error**

|  |  |
| --- | --- |
|  | Results |
| Iterations | 1675 |
| Absolute Error (degC) | 1.00E-05 |
| Convergence Criteria | 1.00E-05 |

**Discussion**

This showed that python can use an iterative approach for gauss sidel and use that to create heat data for the length of a rod. It can approach a lower error and the iterative approach was able to get to the accurate value. The iterations at 1E-5 made the loop go through 1675 times. The interesting view of the graph was how the result went from straight linear to steep and then not.

**Appendix**

import numpy as np

import matplotlib.pyplot as plt

import pandas

#setup m matrix finite diff metrix

#setup c matrix solution matrix

L = 1 #m

sigfigs = 4

epi = 1e-5 #convergence criteria

m\_err = 10.0 #max error

LeftHandSide = 0.0

RightHandSide = 100.0

k\_therm = 0.2 #W/mK

source = 100 # W/m^3

sourcew = L/5

sourcec = L/4

u\_lim = sourcec+sourcew/2

l\_lim = sourcec-sourcew/2

n = 38

n1 = n+1

n2 = n1 + 1

dx = L/(n2-1)

sources = -dx\*dx\*source/k\_therm

mMat = np.zeros((n2,n2))

cMat = np.zeros(n2)

u1 = np.zeros(n2)

u1n = np.zeros(n2)

L\_xp = [0]

L\_xpf = ['%.\*f' % (sigfigs,LeftHandSide)]

u1f = []

#append format so that I dont have to keep writing the same thing over and over

def L\_xpfAppend(n):

    L\_xpf.append('%.\*f' % (sigfigs,n))

for i in range(1,n1):

    mMat[i,i] = -2

    mMat[i, i-1] = 1

    mMat[i, i+1] = 1

    cMat[i] = 0

    L\_xp.append(i\*dx)

    L\_xpfAppend(i\*dx)

mMat[n1,n1] = 1

mMat[0,0] = 1

cMat[-1] = 100.00

u1[0] = 0.0

u1[-1] = 100

#test print matrix

#print(mMat)

#print(cMat)

L\_xp.append(L)

L\_xpfAppend(L)

for i in range(0,n1):

    dist = i\*dx

    if (dist > l\_lim and dist < u\_lim):

        cMat[i] = sources

#linsolve\_solve = np.linalg.solve(mMat,cMat)

icount = 0

L\_merr = []

L\_err = []

L\_count = []

#gauss sidell method use u1 and u1n

while m\_err > epi:

    icount += 1

    for j in range(1,n1):

        u1n[j] = (1/mMat[j,j])\*(cMat[j] - mMat[j,j-1]\*u1[j-1] - mMat[j,j+1]\*u1[j+1])

        err = np.abs(u1n[j]-u1[j]) #absolute error

        L\_err.append(err)

        #print(u1[j])

        u1[j] = u1n[j]

    m\_err = max(L\_err)

    L\_err = []

    L\_merr.append('%.\*g' % (sigfigs,m\_err))

    L\_count.append(icount)

#test print

#print(linsolve\_solve)

#print(u1)

#print(icount)

#L\_aberr = [] #absolute error

for i in range(0,n2):

    u1f.append('%.\*f' % (sigfigs-1,u1[i]))

fig = plt.figure(figsize = (10,8))

plt.plot(L\_xp, u1, '--', color = "tab:blue")

plt.plot(L\_xp, u1, 'o', color = "tab:red")

plt.title('GS Temperature vs Position', fontsize = 12)

plt.ylabel('Temperature $^\\circ$C', fontsize = 12)

plt.xlabel('Position (m)', fontsize = 12)

plt.grid(True)

plt.show()

fig.savefig('Heat\_Transfer/Gaussidell6b\_p2.jpeg',dpi = 300,bbox\_inches = 'tight')

#Data Frame

results = {'Position (m)': L\_xpf, 'GS Temperature (C)':u1f,}

if 'LU Decomp Temperature (C)' in results and all(isinstance(val, (int, float)) for val in results['LU Decomp Temperature (C)']):

    results['LU Decomp Temperature (C)'] = [f"{val:.3f}" for val in results['LU Decomp Temperature (C)']]

df1 = pandas.DataFrame(results)

df1.set\_index("Position (m)",inplace=True)

print(df1)

#Iterations DataFrame

stats\_dict = {'Iterations':L\_count, 'Max Error':L\_merr}

df\_stats = pandas.DataFrame(stats\_dict)

df\_stats.set\_index('Iterations',inplace=True)

print(df\_stats)

L\_header = ["Iterations", "Absolute Error (degC)", "Convergence Criteria"]

L\_stats = [L\_count[-1], L\_merr[-1], epi]

stats\_dict2 = {'Results': L\_stats}

df\_stats2 = pandas.DataFrame(stats\_dict2)

df\_stats2.index = L\_header

print(df\_stats2)

L\_dfs = [df1,df\_stats,df\_stats2]

with open('Heat\_Transfer/TempTable6bp2','w',newline='') as f:

    for df in L\_dfs:

        df.to\_csv(f)

        f.write("\n")