Asjal Ahmad
Fan Lee
Frank De Luna
Dr. Balreira
Differential Equations and Linear Algebra (MATH 3316-2)
5/10/22

"pledged"

Final Project

a) Using any 'n x n' grid, one can solve this problem using the following method. The variables we used to calculate this are 'n' for the number of rows and columns for the square matrix, 'top' temperature, 'right' temperature, 'left' temperature and 'bottom' temperature. These are the sole inputs for the user to calculate the center temperature of the plate. We found the linear system by creating an adjacency matrix denoted by $\boldsymbol{C}$ to emulate the grid. The (i, j) entry is a 1 if the points corresponding to x_i and x_j and are connected in the grid, and 0 otherwise. The size of $\boldsymbol{C}$ is given by $n^2 \, x \, n^2$. The matrix $\boldsymbol{X}$ denotes the equilibrium temperatures at each grid point. The size of $\boldsymbol{X}$ is given by $n^2 \, x \, 1$ The matrix $\boldsymbol{B}$ denotes the sum of the surrounding edge temperatures for each grid point. So, for example, if the grid point is in the 'middle' or is not directly connected to any of the edges where temperature is constant, then the corresponding entry for it in the matrix $\boldsymbol{B}$ will be 0. The size of $\boldsymbol{B}$ is given by $n^2 \, x \, 1$. The solution is given by:

$$4X = CX + B$$

Rewrite the equation using $A = 4I - C$, so that the matrix equation is given by

$$AX = B$$

This can be solved by $X = A^{-1}B$.

b) The temperature at the center of a 1 x 1 plate is exactly 50.0 degrees. This can be solved easily by hand, as shown in Figure 1 below. The center temperature for the 5 x 5 matrix is 50.00000000000001, which was found using the following code. Last, for the 23 x 23 matrix, the center temperature was solved to be 50.00000000000005 using the same process.

c) Ostensibly, the more rows and columns are added to the matrix as grid points, the more accurate the result yields, since we can sample more points for a fixed dimension of the grid plate. The key factor to consider when deciding an acceptable value of n for a grid is the application. This will determine the physical dimensions of the plate being used, and the precision needed for the grid points. For example, in a mission critical industrial application such as transportation of oil and other dangerous substances, a 10 x 10 matrix on a grid size of 10 meters by 10 meters is not acceptable, but having a 10 x 10 matrix on a grid size of 10 centimeters by 10 centimeters is acceptable to ensure maximum reliability when making logistical decisions. Conversely, for applications like the foundation for commercial buildings, a 10 x 10 matrix on a size of 1 meter by 1 meter is acceptable, since in this use case, it is not crucial to know the temperatures of the 'grid points' with such fine precision. Conclusively, the 'acceptable' value of n depends on the use case of the grid plate.

d) Attached below is the code on how to solve any given matrix size.

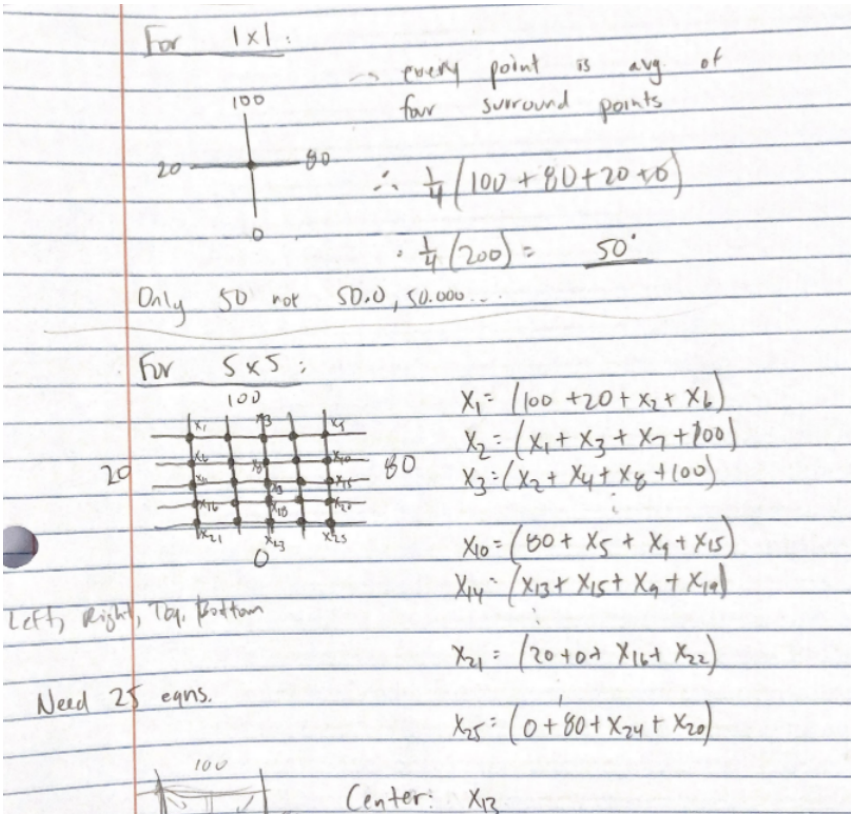

Figure 1: Hand calculation for 1x1 and 5 x 5 matrix

```
Enter top: 100
Enter right: 80
Enter bottom: 0
Enter left: 20
[(1, 50.0, 0.0), (3, 50.0, 0.0), (5, 50.00000000000001, -7.105427357601002e-15), (7, 50.000000000000014,
-1.4210854715202004e-14), (9, 49.999999999999986, 1.4210854715202004e-14), (11, 50.0, 0.0), (13,
49.99999999999996, 4.263256414560601e-14), (15, 50.000000000000014, -1.4210854715202004e-14), (17,
50.00000000000002, -2.1316282072803006e-14), (19, 50.000000000000014, -1.4210854715202004e-14), (21,
50.000000000000036, -3.552713678800501e-14), (23, 50.00000000000005, -4.973799150320701e-14), (25,
50.000000000000014, -1.4210854715202004e-14), (27, 49.99999999999996, 4.263256414560601e-14), (29,
49.99999999999998, 2.1316282072803006e-14), (31, 50.00000000000007, -7.105427357601002e-14), (33,
49.99999999999992, 7.815970093361102e-14), (35, 50.00000000000009, -9.237055564881302e-14), (37,
50.000000000000064, -6.394884621840902e-14), (39, 50.0000000000002, -1.9895196601282805e-13), (41,
49.999999999999986, 1.4210854715202004e-14), (43, 50.00000000000007, -7.105427357601002e-14), (45,
50.0000000000001, -9.947598300641403e-14), (47, 49.99999999999997, 2.842170943040401e-14), (49, 49.9999999999998,
1.9895196601282805e-13), (51, 50.0, 0.0), (53, 50.000000000000014, -1.4210854715202004e-14), (55,
49.99999999999982, 1.7763568394002505e-13), (57, 49.99999999999993, 7.105427357601002e-14), (59,
50.00000000000013, -1.2789769243681803e-13), (61, 49.99999999999997, 2.842170943040401e-14), (63,
49.99999999999995, 4.973799150320701e-14), (65, 49.99999999999969, 3.126388037344441e-13), (67, 50.00000000000002,
-2.1316282072803006e-14), (69, 49.999999999999986, 1.4210854715202004e-14), (71, 49.99999999999992,
7.815970093361102e-14), (73, 49.99999999999964, 3.623767952376511e-13), (75, 50.000000000000114,
-1.1368683772161603e-13), (77, 49.999999999999794, 2.0605739337042905e-13), (79, 50.000000000000284,
-2.8421709430404007e-13), (81, 50.000000000000008, -7.815970093361102e-14), (83, 50.000000000000135,
-1.3500311979441904e-13), (85, 49.99999999999963, 3.694822225952521e-13), (87, 50.00000000000012,
-1.2079226507921703e-13), (89, 50.000000000000019, -1.9184653865522705e-13), (91, 50.000000000000036,
-3.552713678800501e-14), (93, 49.99999999999989, 1.0658141036401503e-13), (95, 50.00000000000027,
-2.7000623958883807e-13), (97, 49.999999999999574, 4.263256414560601e-13), (99, 49.99999999999985,
1.4921397450962104e-13)]
```

Figure 2: Output of center temperature values for 'n x n' matrix from n = 1 to n = 99 in increments of 2 in form of (n x n, average temperature, distance from 50.0)

Python Code

A Google Collab document (DE.ipynb) has been shared with you. The script can be run on the browser, so you don't need to download any code. However, it only works for grids of maximum size of 137 x 137 grid points because Google Collab has RAM limitations for free subscription users. For larger matrices, you will need to run the code on a local machine or pay for the subscription plan for higher speed and RAM. Link to google colab and the code: https://colab.research.google.com/drive/1TKBnhL8CDkAgLl1BVcRurLKkIXBS92Hg?usp=sharing

```
import pandas
import numpy


#an adjacency matrix to emulate a 2d grid of given size
def make_adjMat(size):
	n = size * size
	M = numpy.zeros(shape=(n, n))
	for r in range(size):
	for c in range(size):
		i = r * size + c
		if c > 0:
		M[i - 1, i] = M[i, i - 1] = 1
		if r > 0:
		M[i - size, i] = M[i, i - size] = 1
	return M


def make_edgeMat(top, right, bottom, left, n):
	b = numpy.zeros(shape=(n * n))
	for i in range(n * n):
	if i < n:
		b[i] += top
	if i % n == (n - 1):
		b[i] += right
```

```
        if i >= n * (n - 1):
                b[i] += bottom
        if i % n == 0:
                b[i] += left
        return b
if __name__ == "__main__":
        # input entries
        n = int(input("Enter n for an n by n grid: "))  # n by n matrix
        top = int(input("Enter top: "))  # T_top
        right = int(input("Enter right: "))  # T_right
        bottom = int(input("Enter bottom: "))  # T_bottom
        left = int(input("Enter left: "))  # T_left

        ###
        # Solving 4x = Cx+B, where2
        # x is the vector of equilibrium temperatures at grid points
        # C is the adjacency matrix of grid points
        # B is the vector of equilibrium temperatures at edges
        #
        # write 4x = Cx+B in the form Ax = B where A=4I-C
        # solve x= A^-1 B
        ###
        adj_mat = make_adjMat(n)  # make adjacency matrix
        c = numpy.dot(numpy.identity(n * n), 4)

        a = numpy.subtract(c, adj_mat)

        b = make_edgeMat(top, right, bottom, left, n)
        x = numpy.linalg.solve(a,b)
        x_table = numpy.split(x,n)

        middle_element = x_table[int(len(x_table)/2)][int(len(x_table)/2)]
        print("Temperature at each grid point presented as a table . . .")
```

```
        print(pandas.DataFrame(x_table, columns=list(range(1, n + 1)),
index=list(range(1, n + 1))))
        print("*"*80)
        print(f"Temperature of the center grid point: {middle_element}")
```

References

Trial #1:

Using a sweep from 1-97 (n) in increments of 6:
Standard deviation = 1.7465141465243E-13
Sd —> (17 elements)

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}.$$

$$\sigma^2 = \frac{\Sigma(x_i - \mu)^2}{N}$$
$$= \frac{(50.0 - 50)^2 + \ldots + (49.999999999999595 - 50)^2}{17}$$
$$= \frac{5.185529828816E\text{-}25}{17}$$
$$= 3.0503116640094E\text{-}26$$
$$\sigma = \sqrt{3.0503116640094E\text{-}26}$$
$$= 1.7465141465243E\text{-}13$$

Since our standard deviation is very low, this means that the center temperature does not vary much, very small (E-13)

True value: 50.0 - all approach 50.0

Closest to true value are for **n=19, n=25**
Where they deviate -7.105427357601002e-15 from 50.0, which has the smallest difference compared to rest of matrices

So most optimal would be for either of the two above, but if you take into consideration the cost for drilling and the risk of damaging the material, it is more optimal to have 361 holes (n=19) rather than 625.

However, choosing n=7 is not a bad choice either because it is only -2.1316282072803006e-14 away from our chosen true value (50.0), which is the second most accurate to our true value.

It depends how accurate one one's measurement is, because having an standard error of (below) is quite accurate itself. Where standard error is a measurement of how close any sample point is to the true mean (insignificant)

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{N}} = 4.2359190016206E\text{-}14$$

Frequency Table

Value	Frequency
49.999999999999595	1 (5.8823529411765%)
49.99999999999962	1 (5.8823529411765%)
49.99999999999966	1 (5.8823529411765%)
49.99999999999981	1 (5.8823529411765%)
49.99999999999983	1 (5.8823529411765%)
49.99999999999996	1 (5.8823529411765%)
49.99999999999997	1 (5.8823529411765%)
50.0	1 (5.8823529411765%)
50.00000000000001	2 (11.764705882353%)
50.00000000000002	1 (5.8823529411765%)
50.00000000000003	2 (11.764705882353%)
50.000000000000064	2 (11.764705882353%)
50.00000000000008	1 (5.8823529411765%)
50.00000000000026	1 (5.8823529411765%)

Table 1 showing mode of the sweep (1-97 in steps of 6)

Trial #2:

Using a sweep from n=1 to n=99 using 2 increments (48 samples)
Standard deviation found to be:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}.$$

$$\sigma^2 = \frac{\Sigma(x_i - \mu)^2}{N}$$

$$= \frac{(50.0 - 50)^2 + \ldots + (49.99999999999985 - 50)^2}{48}$$

$$= \frac{1.0333194334181E\text{-}24}{48}$$

$$= 2.1527488196211E\text{-}26$$

$$\sigma = \sqrt{2.1527488196211E\text{-}26}$$

$$= 1.4672248701617E\text{-}13$$

We would like to explore our results when we run more matrices to see if the answers vary from Trial 1, and if, by how much. We assume that the 48 matrices ranging from 1x1 to 99x99 would yield more accurate results.

Where μ is the calculated mean (50.0), which we will use as our true value. It is a valid 'true' value, since we analyzed 48 different cases, and there were little differences from 50. None of them differed by even a hundredth of a percent, so we can conclude that the answers approach 50.0.

-In addition, the mean of the 48 sample points is 50, so from this we can use this as our true value.

According to Table 2, when n=5, the temperature at the center of the plate is closest to our true value, which differs by -7.105427357601002e-15.

However, when n=1, n=3, n=11 and n= 51, temperature is exactly 50.0 in the middle, true to our absolute value.

Value	Frequency
49.999999999999574	1 (2.0833333333333%)
49.99999999999963	1 (2.0833333333333%)
49.99999999999964	1 (2.0833333333333%)
49.99999999999969	1 (2.0833333333333%)
49.999999999999794	1 (2.0833333333333%)
49.9999999999998	1 (2.0833333333333%)
49.99999999999982	1 (2.0833333333333%)
49.99999999999985	1 (2.0833333333333%)
49.99999999999989	1 (2.0833333333333%)
49.99999999999992	2 (4.1666666666667%)
49.99999999999993	1 (2.0833333333333%)
49.99999999999995	1 (2.0833333333333%)
49.99999999999996	2 (4.1666666666667%)
49.99999999999997	1 (2.0833333333333%)
49.99999999999998	1 (2.0833333333333%)
49.999999999999986	3 (6.25%)
50.0	4 (8.3333333333333%)
50.00000000000001	1 (2.0833333333333%)
50.000000000000014	5 (10.416666666667%)
50.00000000000002	2 (4.1666666666667%)
50.000000000000036	2 (4.1666666666667%)
50.00000000000005	1 (2.0833333333333%)
50.000000000000064	1 (2.0833333333333%)
50.00000000000007	2 (4.1666666666667%)
50.00000000000008	1 (2.0833333333333%)
50.00000000000009	1 (2.0833333333333%)
50.0000000000001	1 (2.0833333333333%)
50.000000000000114	1 (2.0833333333333%)
50.00000000000012	1 (2.0833333333333%)
50.000000000000135	1 (2.0833333333333%)
50.00000000000019	1 (2.0833333333333%)
50.0000000000002	1 (2.0833333333333%)
50.00000000000027	1 (2.0833333333333%)
50.000000000000284	1 (2.0833333333333%)

Table 2 showing all values generated and frequency of occurrence

Value for sweep (1-99; skipping 2 → n=1,3,5…):

50.0, 50.0, 50.00000000000001, 50.000000000000014, 49.999999999999986, 50.0, 49.99999999999996, 50.000000000000014, 50.00000000000002, 50.000000000000014, 50.000000000000036, 50.00000000000005, 50.000000000000014, 49.99999999999996, 49.99999999999998, 50.00000000000007, 49.99999999999992, 50.00000000000009, 50.000000000000064, 50.0000000000002, 49.999999999999986, 50.00000000000007, 50.0000000000001, 49.99999999999997, 49.9999999999998, 50.0, 50.000000000000014, 49.99999999999982, 49.99999999999993, 49.99999999999995, 49.99999999999969, 50.00000000000002, 49.999999999999986, 49.99999999999992, 49.99999999999964, 50.000000000000114, 49.999999999999794, 50.000000000000284, 50.00000000000008, 50.000000000000135, 49.99999999999963, 50.00000000000012, 50.00000000000019, 50.000000000000036, 49.99999999999989, 50.00000000000027, 49.999999999999574, 49.99999999999985