

# CIS 4930 - 001: Summer 2018

## Homework - 3

Total Points: 100  
Due: Tuesday 07/03/2018

### 1 Objective

The objective for this assignment is to make sure

- You can use `numpy` to write small Python applications to solve simple scientific problems.
- You can use `matplotlib` to plot basic graphs.
- You are comfortable with Python - syntax and coding conventions, to write Pythonic code.

### 2 2 Dimensional Heat Distribution

For this problem, you are required to model how heat spreads through a material on 2 dimensions. You need to model the material using a 2 dimensional grid using `numpy` array and calculate how the heat flows through it. We are going to mimic a Jacobi iteration that determines the value of the current cell is determined by the 4 cardinal neighbors of the cell.

1. Heat is generated at the left wall of the material, so the left wall is at a constant temperature. every other cell starts off at 0.

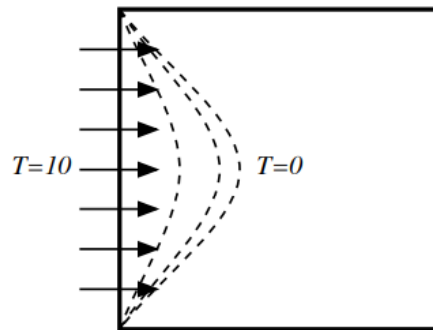


Figure 1: Temperature at the left wall

2. At every cell, the current temperature is calculated using the temperature at the cardinal neighbors at the previous state, using the following equation:

$$temp[i, j] = \frac{1}{4}(oldTemp[i - 1, j] + oldTemp[i + 1, j] + oldTemp[i, j - 1] + oldTemp[i, j + 1])$$

3. The calculation ends at convergence, where two consecutive iterations result in the same values.

### 3 Specifications

For this homework, you will write a program that involves the third party Python libraries `numpy` and `matplotlib`. These cannot be tested on linprog. You need to have a personal setup up and running.

- Get the starting temperature from the user. This will be an integer. (5 points)
- Create a 2-dimensional `numpy` array. the grid will have as many rows and columns as the starting temperature. You will also need *halo cells*. You will need an extra row on the top and the bottom and an extra column at the left and the right. (5 points)
- Set the leftmost column values to the starting temp. Set all the other values in the grid to 0. (5 points)
- You would need 2 of these, one for the previous state and one for the current state. (5 points)
- Use the given equation to calculate the temperature at each cell. (20 points)

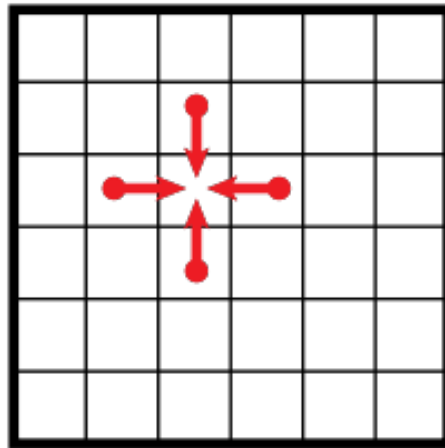


Figure 2: Calculating temperature at a cell

- While calculating the temperatures at each iteration, do not change the values of the halo cells. These cells will stay at starting temperature for the left wall and 0 otherwise.
- At the end of the iteration, check if the current state is EXACTLY the same as the new state. If so, we're done. Otherwise repeat the process. (10 points)
- Sometimes, we will not get convergence for a very long time. In the interest of time, we will stop at 3000 iterations if we don't see convergence by then. (10 points)
- Once the grid has been finalized, plot each point onto a graph. The color of the point is determined by the temperature of the cell at the end of the last iteration. (40 points)
- We have used 8 colors for our graphs where `darkred` is the hottest spot and `darkblue` is the coolest spots. divide the temperatures equally on the scale between 0 and the starting temperature.

### 4 Sample Output

This section shows the sample output for the program for 2 different runs, one when the temperature is 15 and once when it is 150. The output looks more and more like a heatmap when the initial temperature

values are higher. For reference, the colors used for the heatmap are - darkred, red, orange, yellow, lawngreen, aqua, blue, darkblue

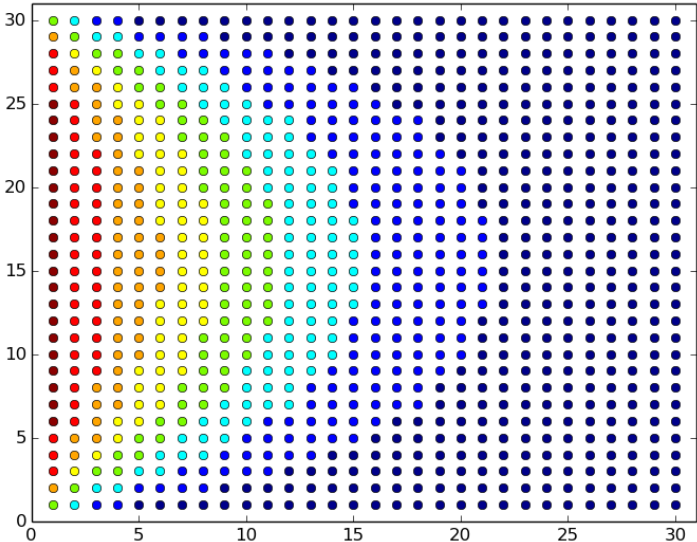


Figure 3: Starting Temperature: 30

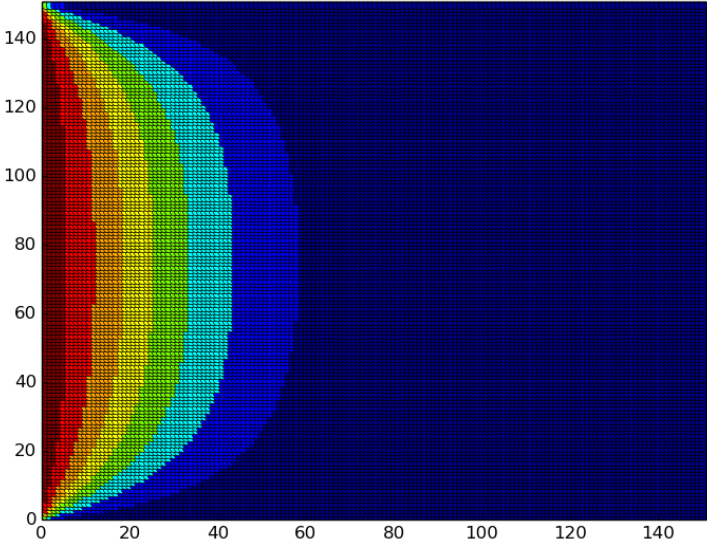


Figure 4: Starting Temperature: 150

## 5 Submission

- Try and use Python 3 as much as possible. For this homework, version should ideally not affect anything
- Please name your files `heatDiff.py`
- Please add your name and FSUID as a comment at the top of your programs.
- Please solve the problem yourself instead of Googling “Jacobi Method for heat distribution”. We are only mimicking the process and not using the actual Jacobi method (which requires a diagonally dominant matrix to work).
- This might take about a minute to run. This is OK.
- You might not see convergence at the same point or get exactly the same answer as the example, due to floating point errors.
- Please turn in the program through Canvas.