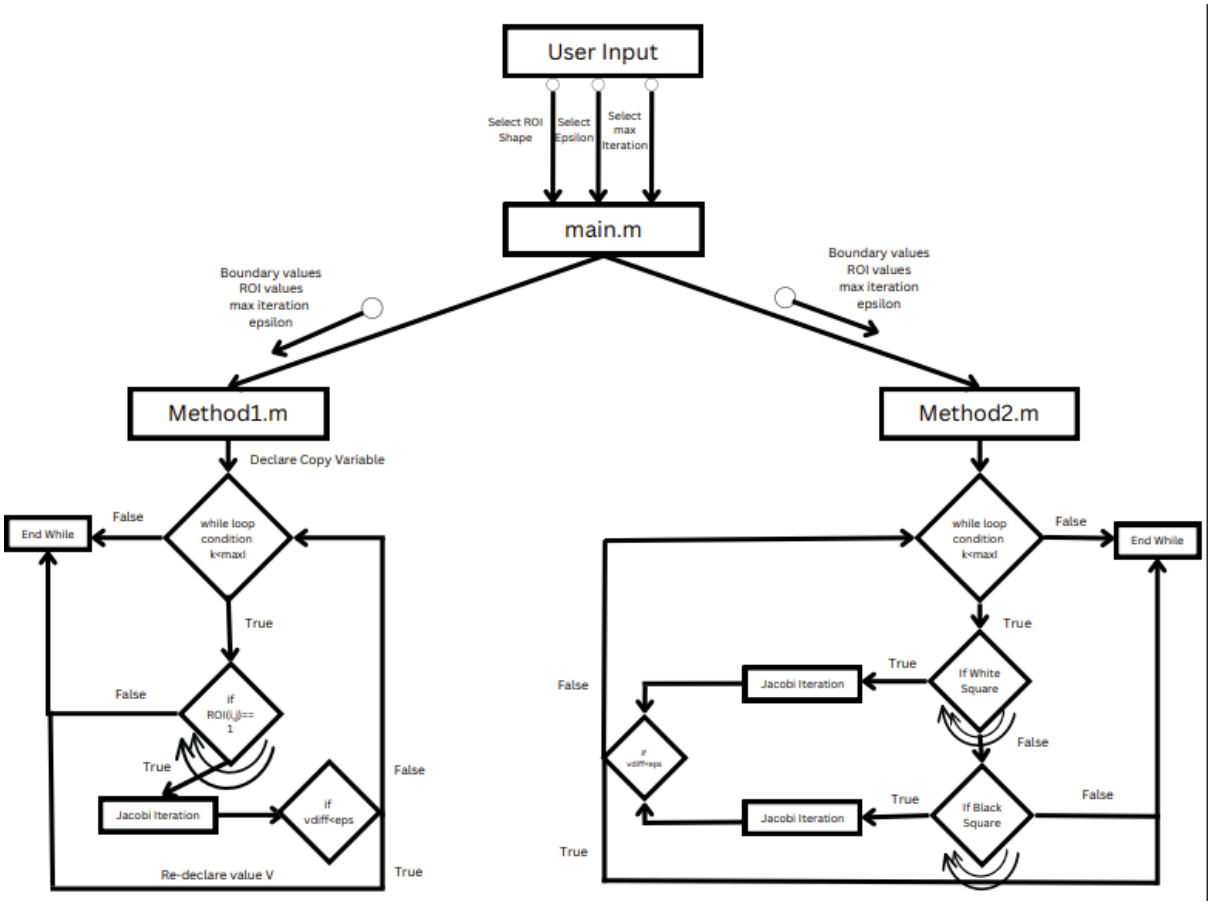
Computational Methods Solving PDE's Coursework 3 Written Report

Mohammed AbuSadeh

Program Structure Chart:



Description of different files that run the program:

- main.m (Driver File):

The main file loads all three ROI (Region of Interest) and Boundary Values files. There are 3 different shapes used for each pair of ROI and boundary value files; Square, Circle and a Diamond shaped regions of interest. The boundary values files are loaded into variables V1, V2 and V3. The ROI files are loaded into variables ROI1, ROI2 and ROI3.

The user is then asked to input values 1, 2 or 3, in order to choose which shapes they would like to solve the boundary values for. 1 for Square shaped region, 2 for circle shaped region and 3 for diamond shaped region. The user is then asked to determine the limitations of solving for the values inside of the picked region. The user would input the maximum number of iterations allowed for the program to perform and the degree of tolerance (epsilon). Then there is a switch statement that is incorporated in the driver file to solve for the shape chosen by the user.

The driver file solves the indicated region by performing two methods on the region of interest and the boundary values already given by the files. The first method performs the Jacobi Iteration by creating a copy of the current index of the point. The second method performs the Jacobi Iteration

without using a copy of the current value, but instead created a chessboard-like background in order to be able to iterate over each other value in order to update the unchosen for other-coloured values around it.

Finally the main file plots two images to show the solution for the predicted values within the region of interest based on the boundary values. In the figure shown, the time taken to solve the values and the number of iterations are shown in the title part above each method image.

- Method1.m:

In the first method, the size (dimensions) of the ROI are generated into variables N and M. A while loop runs the main part of the method based on the condition of having the number of current iterations to be less than the indicated max number of iterations, otherwise while loop will break. A copy variable is declared to save the current solution of the iteration. Then, two for loops are incorporated, the first looping through the rows of the file and the second looping through the columns. Then inside the for loops the Iterative Jacobi method is used to update the current element based on the four values surrounding it. This new value of the point is then saved in the same location in the copy variable in order to reference it again in the next iteration.

Finally another conditional if statement is used to check if the difference between the new copied value and the original value is less than the degree of tolerance indicated by the user. Finally the file returns a new file containing all the new solved values as an output to then be illustrated in the main file.

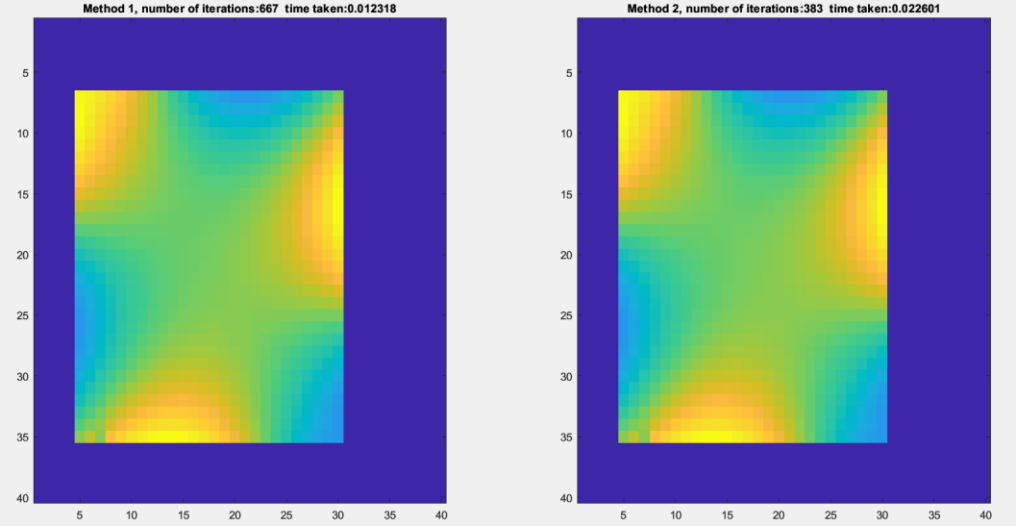
- Method2.m:

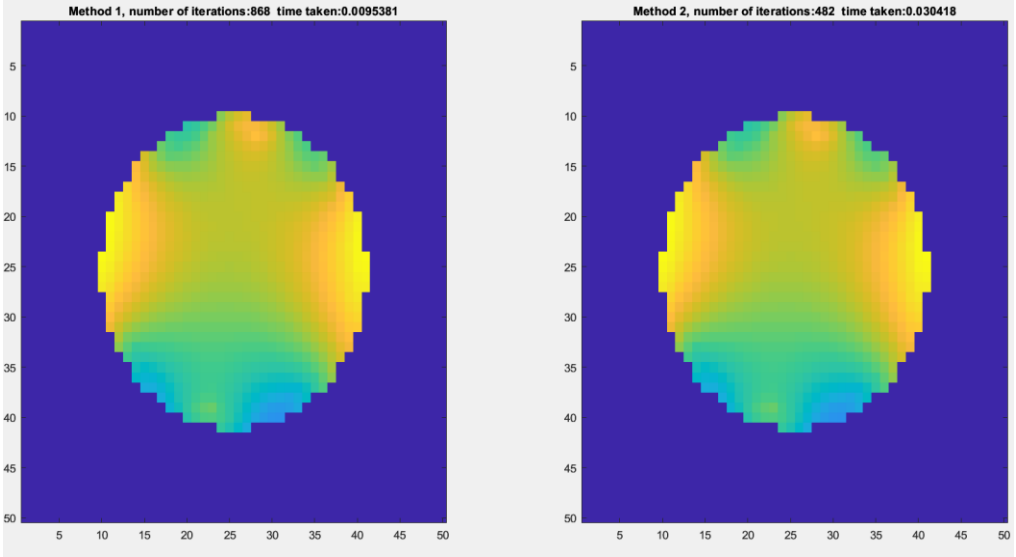
In the second method, the size (dimensions) of the ROI are generated into variables N and M. A while loop runs the main part of the method based on the condition of having the number of current iterations to be less than the indicated max number of iterations, otherwise while loop will break. In this method the copy variable was only declared to calculate the difference between the new copied value and the original value only and not to be included in any way in the iterative method.

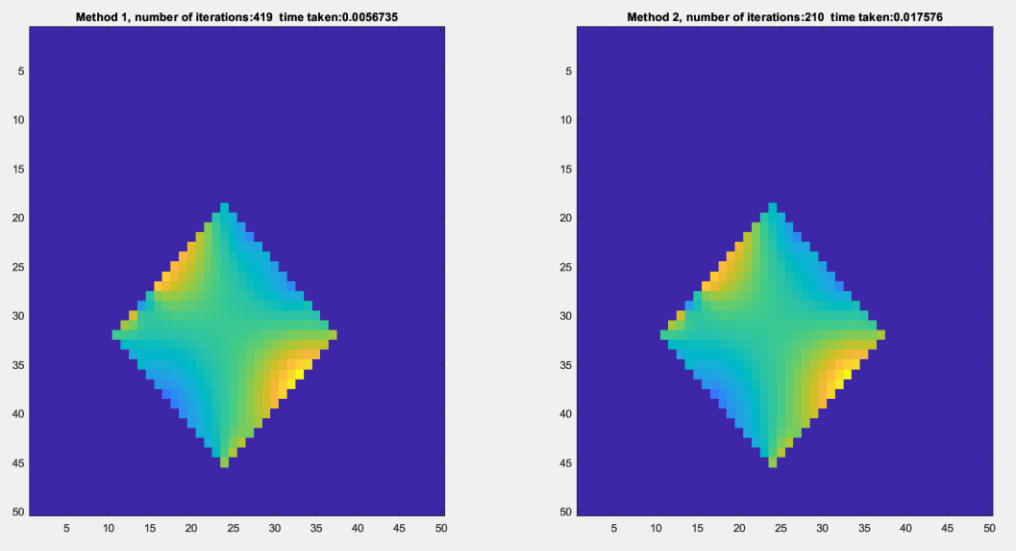
Two sets of for loops that run through the rows and columns of the file are made. A chessboard-like grid is made in this case instead of storing the current copy of the changed value. Therefore, the first two for loops only change the values that are set on the “white” squares of the squares and then are updated based on the four black squares around it. Similarly, the second two for loops only change the “black” squares based on the four white squares around it.

Finally another conditional if statement is used to check if the difference between the new copied value and the original value is less than the degree of tolerance indicated by the user. Finally the file returns a new file containing all the new solved values as an output to then be illustrated in the main file.

Comparing Both Methods:







As seen in running Method A and Method B for all three shapes, the second method produces the solved images with almost half the number of iterations. However, the time taken to generate the solution is always longer for the second method. Therefore, if the user is willing to use less computational power while understanding the cost of the time taken to solve the regions, the second method would be a better incorporation of solving for the boundary values for each shape.