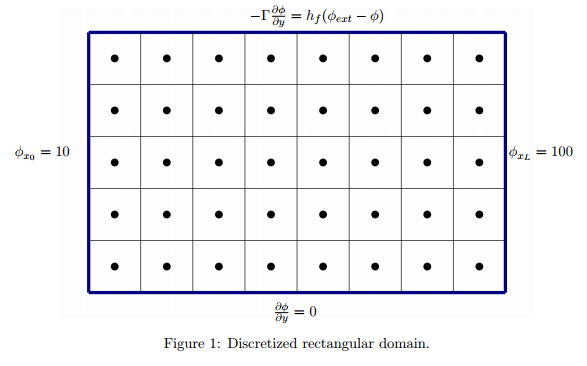
|  |
| --- |
| U of T |
| MIE1210 |
| Project #2 Report |

|  |
| --- |
| Kyu Mok Kim  998745381 |

The goal of this project is to develop a finite volume solver to solve the equation

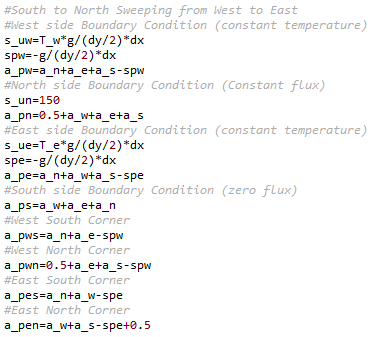
, where φ = φ (x,y) in Ω, which is a rectangular domain of size Lx x Ly. Γ is the constant diffusivity coefficient. The following figure below shows how the domain is broken into discrete control volumes including the boundary conditions.



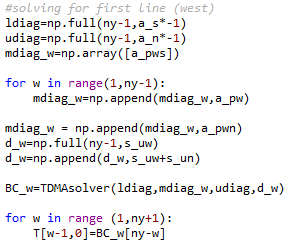
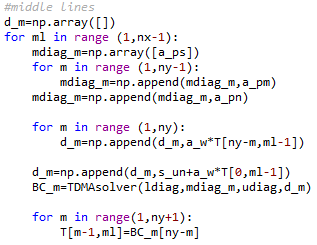
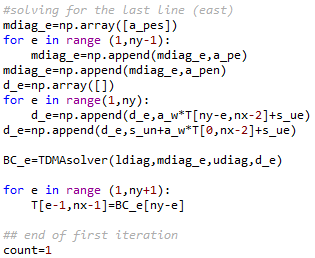
For my code, I have divided the above boundary conditions in to four sides: North, West, East, and South. For the South side, it is insulated with zero flux, therefore the source term would be zero. For the West and East side, they are constant temperature boundary condition (Dirichlet) and could calculate the source term using the diffusivity constant (T\* Γ/(dx/2)\*dy). Notice that the grid spacing for the boundary conditions are half of that of the middle nodes. For the North boundary condition, it is a convective heat flux boundary (Robin or mix of Dirichlet and Neumann). The constants for the flux equation are plugged in and solved: dφ/dy=150-0.5φ. For each boundary node on the North side will then have source term of 150 and +0.5 on the coefficient a\_p. More details on how I discretized the boundary conditions are shown in the next section.

The code to solve this problem is named as “DiffusionFVM12.py”. It includes the TDMAsolver from previous project. The problem is approached by setting the grid dimensions, resolution, and the constants. Then, the grid spacing and the coefficients are calculated.

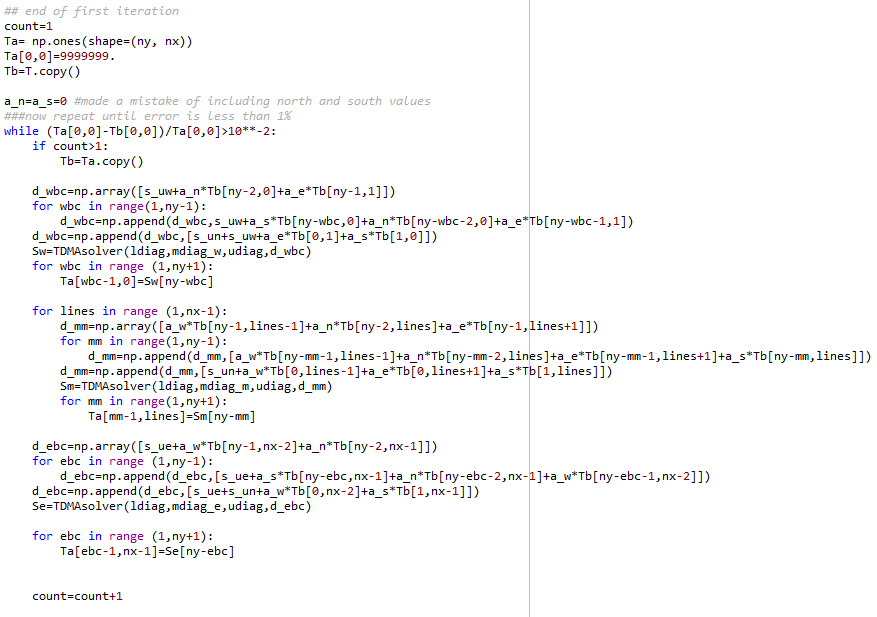
For the boundary conditions, I have divided the coefficients for the corners and the sides as shown:



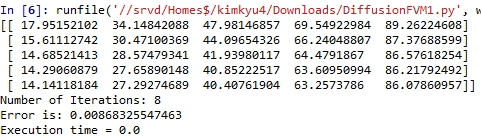
Then, I have decided to use TDMA iterative method, from south to north, sweeping from west to east.

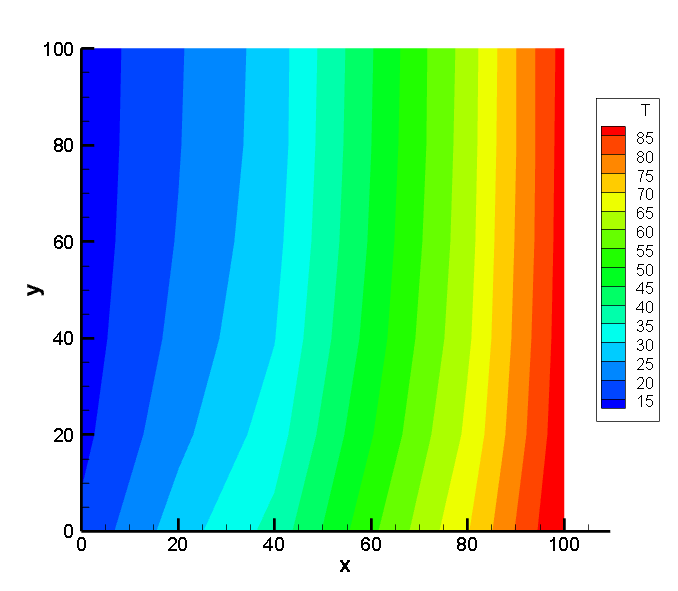
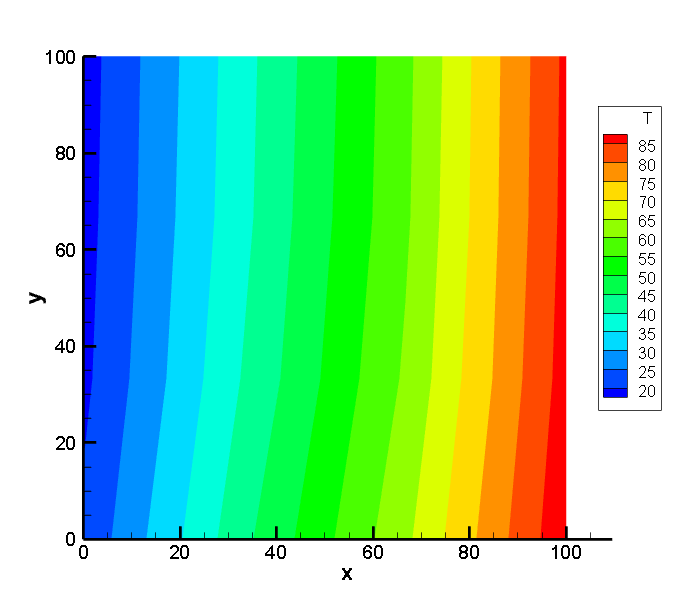
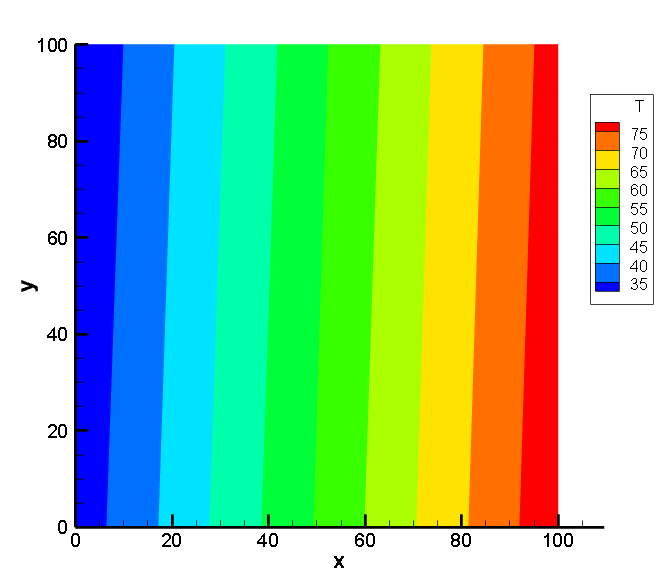
After the first iteration, the process becomes repetitive and was able to loop the iteration until a set error that you can vary as shown on the next page.

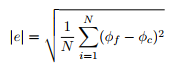


Then, testing the code, 5x5 matrix returns the following:



Also, contour plots for the solution is shown below (from coarse to fine mesh)

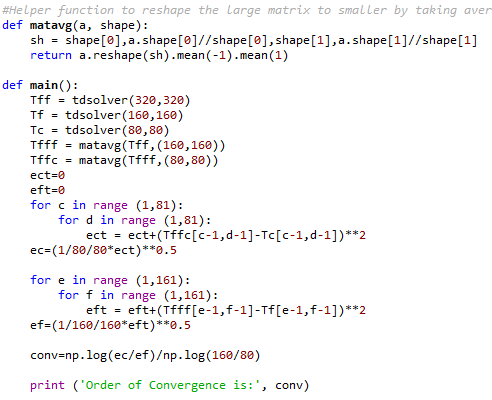


To estimate the order of convergence, , resolutions of 80x80 as coarse mesh, 160x160 as fine mesh, and 320x320 as exact solution are assumed. The error norm used for this estimation is the normalized L2 norm, computed as: .

To make the calculation easier, I have converted main function from “DiffusionFVM12.py” to “tdsolver” and added a helper function to resize the array.

Now letting 320x320 as φf, error norm for 80x80 (ec) and 160x160 (ef) are calculated.

Then, the order convergence is calculated as shown below (included as “DiffusionFVM12.py”:



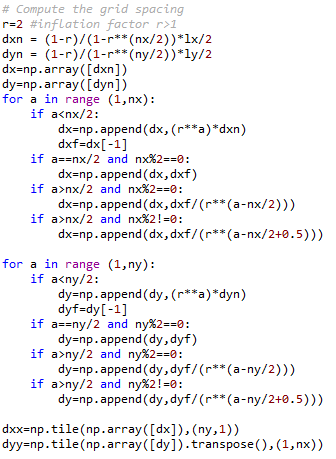
With the output of



In order to allow the code to work for varying node space, inflation factor r>1 is to be defined (2 in my code “DiffusionFVM3.py”). Then, the initial spacing and each subsequent spacing is computed using the following equations:

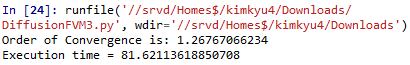


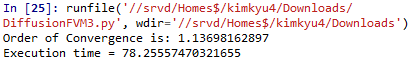
This is put into python as following:



I have decided to put dx and dy in the form of an array to make calculations easier. Then, the coefficients and the constants can also be put into array form and the rest of the code can just use the values from the array as shown in the attached code “Diffusion FVM3.py”.

Then, the convergence of the problem is calculated as shown below:

for r=2

for r=4

The finite volume method is used here and is a common approach used in CFD codes. This is due to its many advantages. First, there are no requirements on the grid to be structured so it can be used for complex geometries. Such unstructured grids are more computationally efficient than the structured grids due to the rigid connectivity restriction.