Patrick Chabelski
MIE 1210 HS
Assignment 3 – 2D Advection-Diffusion Solver
November 23, 2016

DESCRIPTION OF CODE

Diffusion Module Remake

The diffusion solver used for Assignment 2 has been overhauled after receiving feedback. The previous code was unnecessarily complex and yielded inaccurate results, prompting the changes. As a result the entire program was rewritten and decreased from over 600 lines (consisting of main function and library of sub-functions) to less than 200 lines. The methodology behind the code has been slightly altered, as it now sweeps west-to-east, with each line starting from the southern point and solving northward. In addition simpler and cleaner grid-point spacing has been implemented, which allows for both uniform and non-uniform meshes.

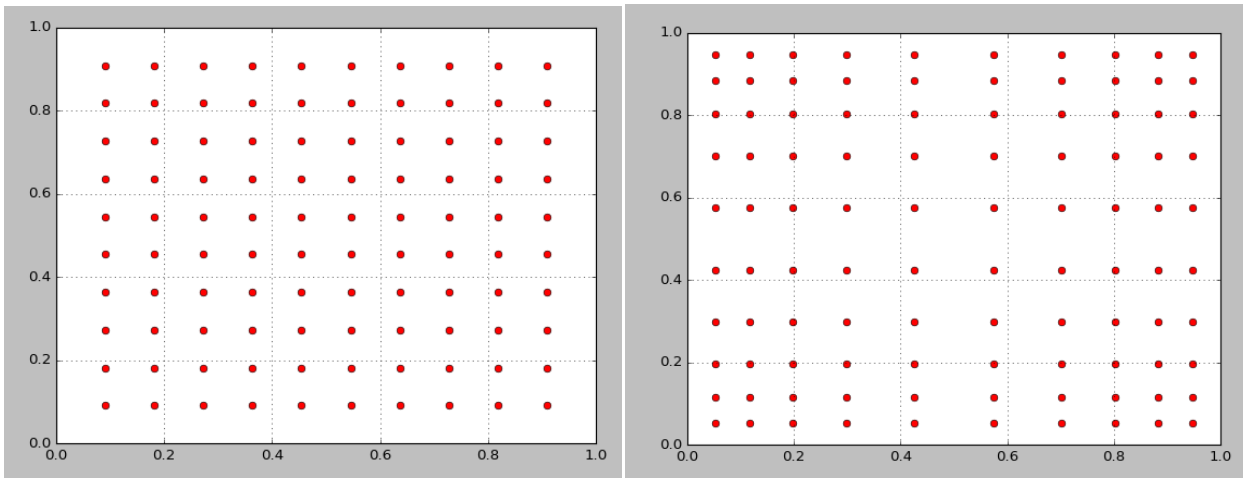

Figure 1: (LEFT) Uniform Mesh of 10x10 Nodes. (RIGHT) Nonuniform Mesh of 10x10 Nodes, with r = 1.5

$$\begin{bmatrix} 4 & 8 & 12 & 16 \\ 3 & 7 & 11 & 15 \\ 2 & 6 & 10 & 14 \\ 1 & 5 & 9 & 13 \end{bmatrix}$$

Figure 2: Numbering System for a 4x4 Grid. Each line proceeds south to north, with the overall line sweep proceeding west to east.

The boundary condition handling has been updated extensively. Instead of having a massive block of code for each wall and corner, the conditions are now embedded within the nested loop that generates the coefficients. These are handled by using if statements that check whether or not the loop increment is at the wall or corner; if the increment counter is 0 or (max number of nodes), the code will implement the relevant boundary conditions for that point's coefficients.

Finally, the convergence criteria was handled in a simpler manner. Now after each full sweep, the average temperature of the system is calculated. This is then compared to the previous

iteration's average temperature. If the percentage difference between both is less than 1%, the new temperature matrix will be considered the final output. Otherwise, the process continues to iterate.

Rerunning Assignment 2 with the given boundary conditions and setup, the following are the results (compared to what was output previously).

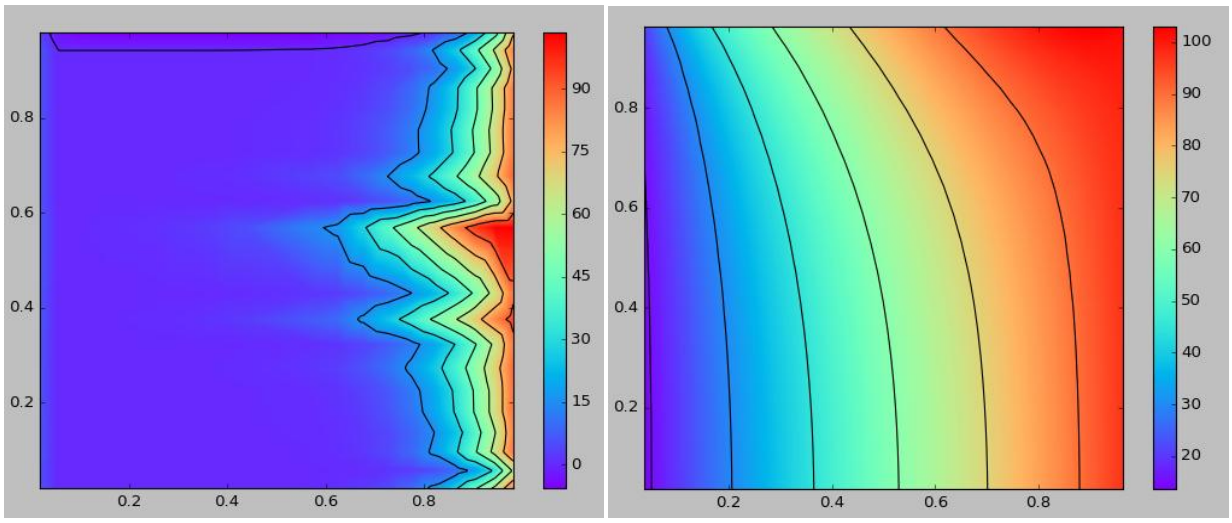

Figure 3: Comparison of Old Assignment 2 Code (LEFT) and New Diffusion Solver Code (RIGHT) for 20x20 Nodes, r = 1.05

The results make more sense and actually depict the diffusion process accurately. This version of the diffusion solver is then used to build the Assignment 3 code.

Convection Solver Code Overview

Assignment 3 focuses on the comparison between Central Differencing and Upwind Schemes for Advection-Diffusion problems. In this iteration of my code, the user is prompted to select which scheme they wish to use at the start. Both schemes are implemented directly onto the diffusion solver; since the diffusion solver is essentially a nested loop which determines each node's coefficient values based on boundaries and input, the schemes are directly implemented onto these. In this iteration, there is a separate function for each scheme, which can easily be replaced by the hybrid scheme (which is dependant on the Peclet number, calculated within the code).

For both schemes, the diffusion components (for all nodes) are calculated and stored as D values, while the velocity-dependant portion are calculated and stored as F. Positive flow is chosen to be west-east for U, and south-north for V. The difference in the derivation and final form of both schemes can be found in the Appendix.

When running the code, the user can specify values for the diffusion constant K, U and V velocities, and grid inflation factor. This allows for both pure diffusion and pure convection problems to be solved (by selecting 0 for the appropriate variables).

RESULTS AND DISCUSSION

Central Difference Advection Solver

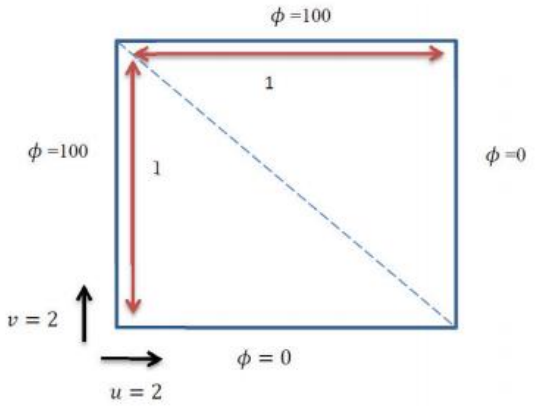

Figure 4: Assignment 3 Boundaries and Velocities

Using the setup as outlined above, the central difference method advection solver was analysed with no diffusion. For this analysis, a uniform grid (r = 1) was used with varying mesh resolutions (N = 10, 40, 100). However, all of these tests resulted in a solver error. This is because central differencing cannot give a result using no diffusion, as the resulting a_p values (to be passed into TDMA) will become zero, therefore leading to instability. This idea matches what is given in Figure 5.15 of the Versteeg Textbook, which describe the values of phi along the X-X line in our setup. Here we can see that there is no plot for Central Differencing; these issues are resolved using more detailed solvers such as Upwind or QUICK schemes. Ideally, for a case with no diffusion and constant u and v velocity as given, the result of pure convection would be two halves of 100oC and 0oC, split along the y = x line.

Re-coupling the diffusion solver to the central differencing scheme (by using a diffusion constant of K = 5) and analysing the results for various resolutions, the following plots are generated:

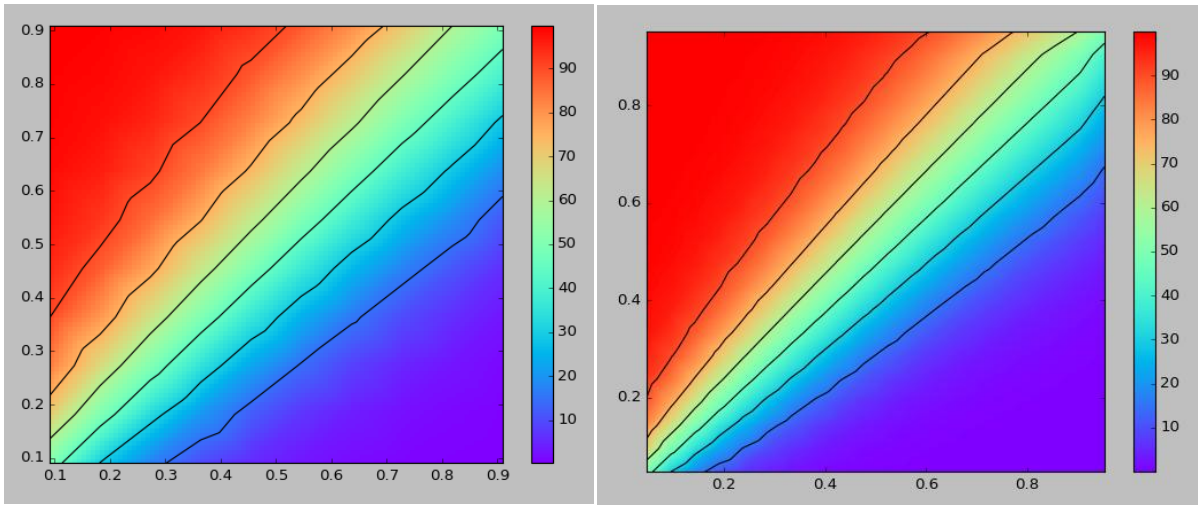

Figure 5: (LEFT) Temperature Plot with 10x10 uniform grid, converged after 24 iterations. (RIGHT) 20x20 uniform grid converged after 6 iterations.

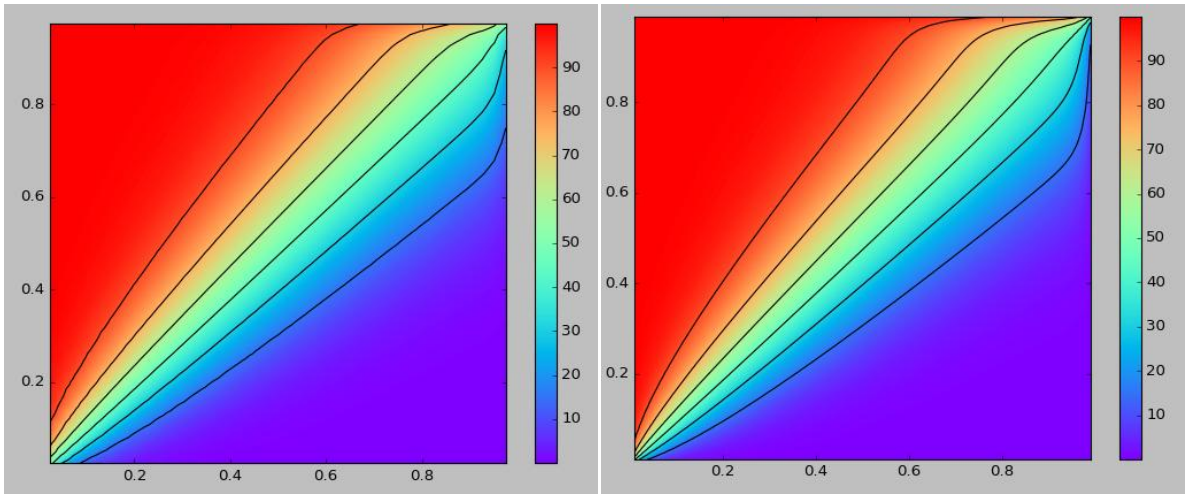

Figure 6: (LEFT) Temperature Plot with 40x40 uniform grid, converged after 35 iterations. (RIGHT) 100x100 uniform grid, converged after 216 iterations.

It can be seen from the above figures that including a small amount of diffusivity with central differencing will allow for a solution to be found. These solutions are similar to the false diffusion phenomena which is associated with more advanced solvers; this will be proven when analyzing the Upwind Scheme. By increasing the resolution, it can be seen that the temperature values converge at the north-eastern corner. This is due to the u and v velocities skewing the temperature gradient toward this corner; higher resolutions show that the boundary conditions are adhered to in this area as a result.

The above comparison is done using a non-uniform mesh to determine any computational differences. For this example, a grid of 100x100 nodes is used with an inflation factor of r = 1.05:

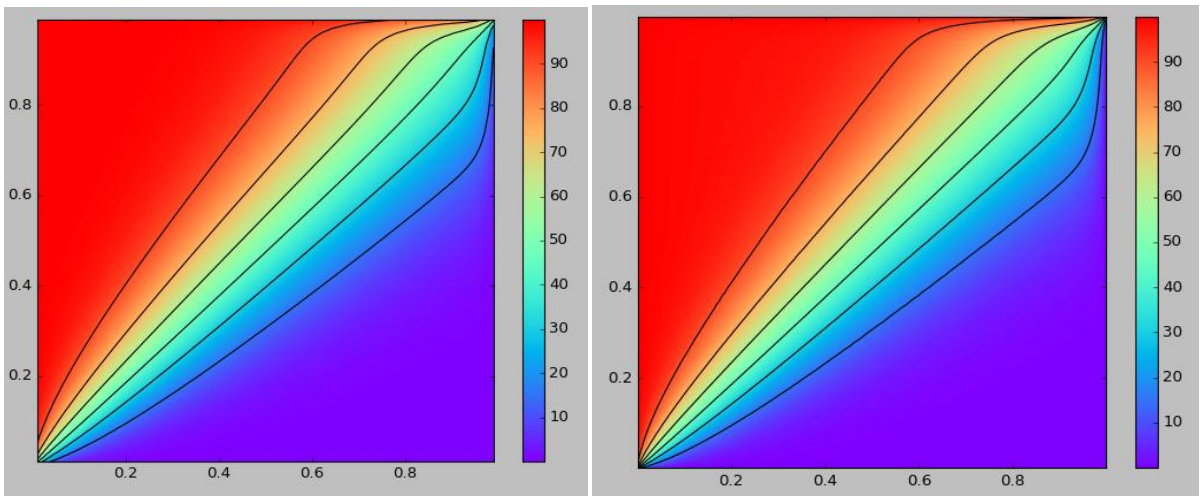

Figure 7: (LEFT) 100x100 uniform grid with r = 1.0, 216 iterations and 46 seconds to convergence. (RIGHT) 100x100 non-uniform grid with r = 1.05, 278 iterations and 27 seconds to convergence.

It can be seen that the solution is more detailed when using a finer mesh near the boundaries. This allows us to view the temperature convergence more accurately as a result. The non-uniform mesh was also much faster in terms of computing time to reach convergence, as it was nearly twice as fast as the uniform mesh. It required more iterations to solve, but solved them much quicker. This is due to the fact that the central nodes (which do not contain detailed information) are larger in the non-uniform mesh, and are therefore calculated faster than in the uniform mesh.

The central differencing scheme was then analyzed using the boundary and initial conditions from Assignment 2. Here the goal was to verify the integrity of the solver by attempting to recover the results from Assignment 2, but using convection in addition to less diffusion. Application of the boundaries and iteration of the velocities was done to a uniform grid with 40x40 nodes:

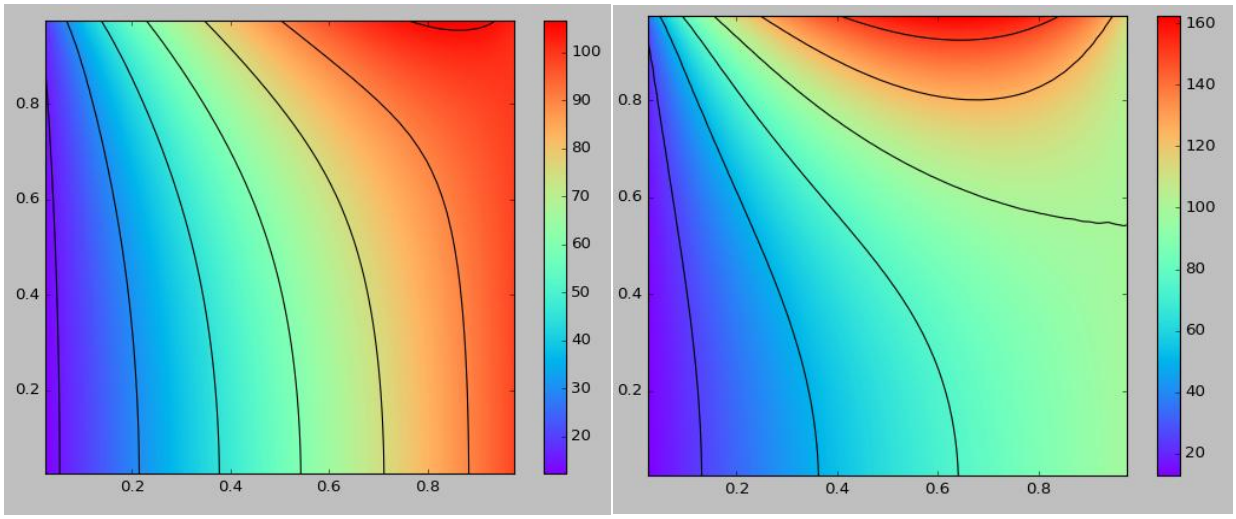

Figure 8: (LEFT) Assignment 2 result for 40x40 Nodes, r = 1, K = 20. (RIGHT) Coupled solver results for 40x40 Nodes, r = 1, K = 5, u = v = 0.

It can be seen that with less diffusion, the flux term at the north wall becomes more dominant

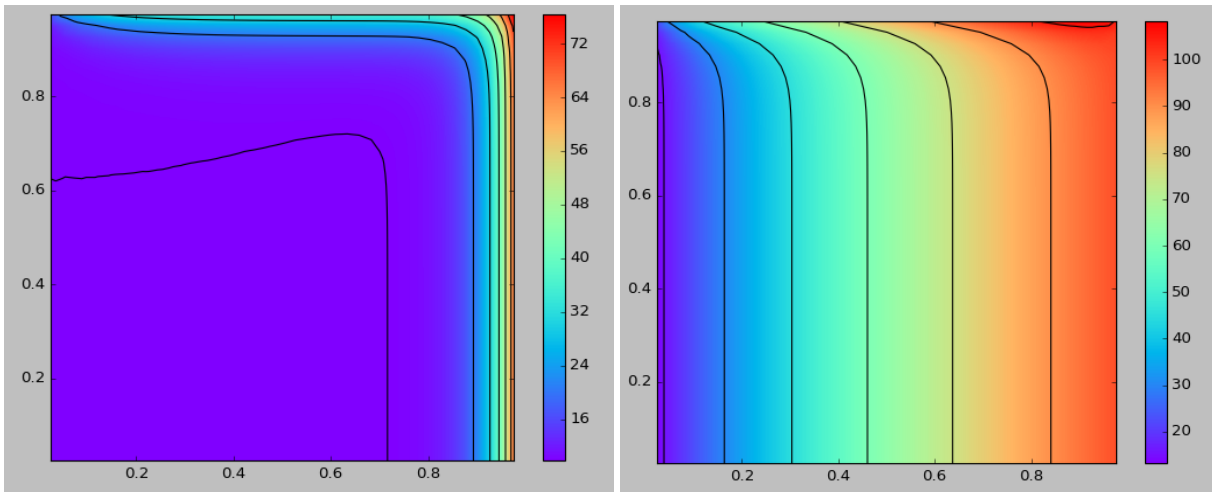

Figure 9: (LEFT) Coupled Solver Results, 40x40 Nodes, r = 1, K =5, u = v = 1. (RIGHT) Coupled Solver Results with u = -0.05, v = 1.1

It can be seen from the figures above that the velocities need to be selected carefully in order to reach a similar temperature distribution to that of Assignment 2. Using u = v = 1 generates flow from the south-western corner, which is strong enough to impact the majority of the domain. However, tweaking the values such that u is near zero and v is near 1 will generally produce a distribution in-line with what was seen in Assignment 2. The only major difference is the sharpness of the northern flux-related temperature curve; the convection model has a much sharper turn that the pure diffusion model. This could be attributed to the fact that there is less diffusion, so any temperature gradients and curves will be much sharper as a result (no large amounts of diffusion resulting in wider spread of temperature).

Upwind Advection Solver

The analysis that was performed on the Central Difference Scheme was then performed on the Upwind Scheme. To start, the scheme decoupled from the diffusion solver (by setting K = 0) and determining the results for pure convection. As previously stated, the ideal result is a direct split between 100oC and 10oC temperatures across the middle diagonal.

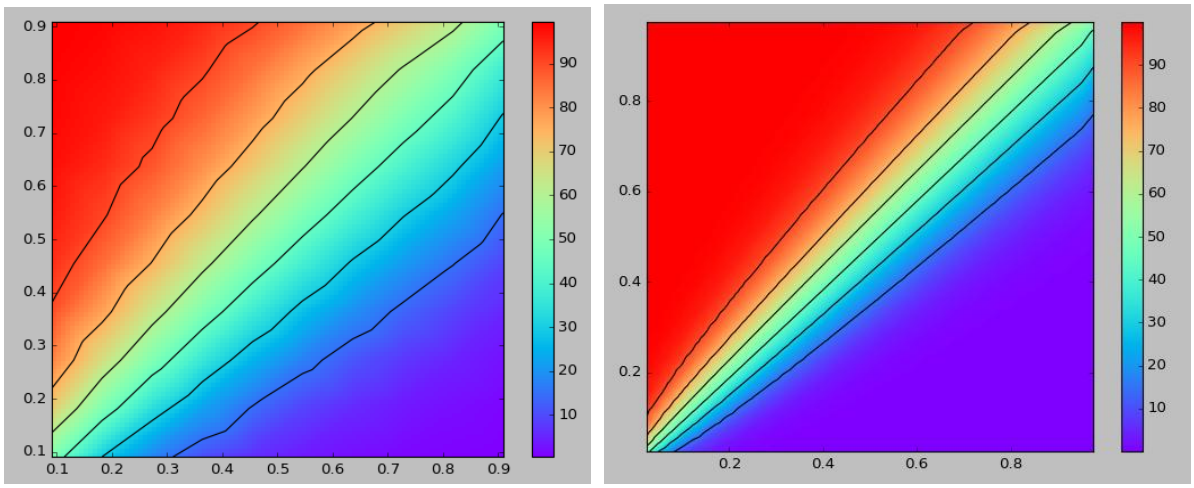

Figure 10 (LEFT), 10x10 Nodes, converged after 3 iterations. (RIGHT) 40x40, converged after 3 iterations

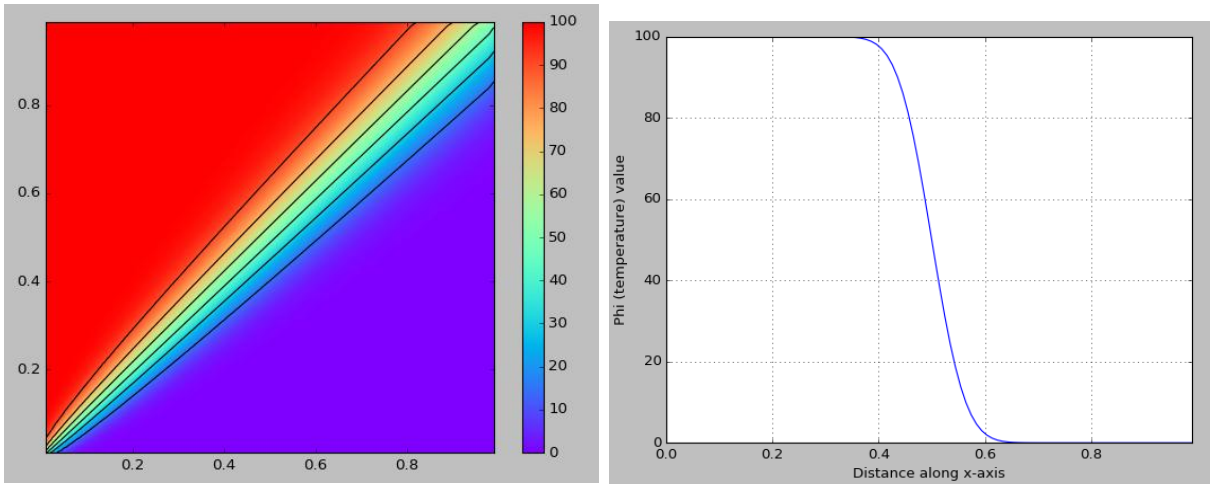

Figure 11: (LEFT) 100x100 Nodes, converged after 3 iterations. (RIGHT Temperature values across diagonal (northwest corner to southeast corner).

It can be seen that even with diffusion turned off, the solver automatically adds some in; this is the false diffusion phenomenon that was alluded to earlier. This is less severe for finer meshes, as can be seen by the comparison between the 10, 40, and 100 node grids. As a result, these match the Upwind comparisons in Figure 5.15 of the Versteeg Textbook, which shows that as the number of grid points increases, the slope of the temperature curve becomes steeper. Finally,

the number of iterations for each case remained constant at 3; therefore the convection solver alone does not require much computational power, showing the benefit of the Upwind scheme versus the Central Differencing.

When the Upwind scheme is coupled with diffusion and analyzed using K = 5, the results obtained are very similar to those with no diffusion.

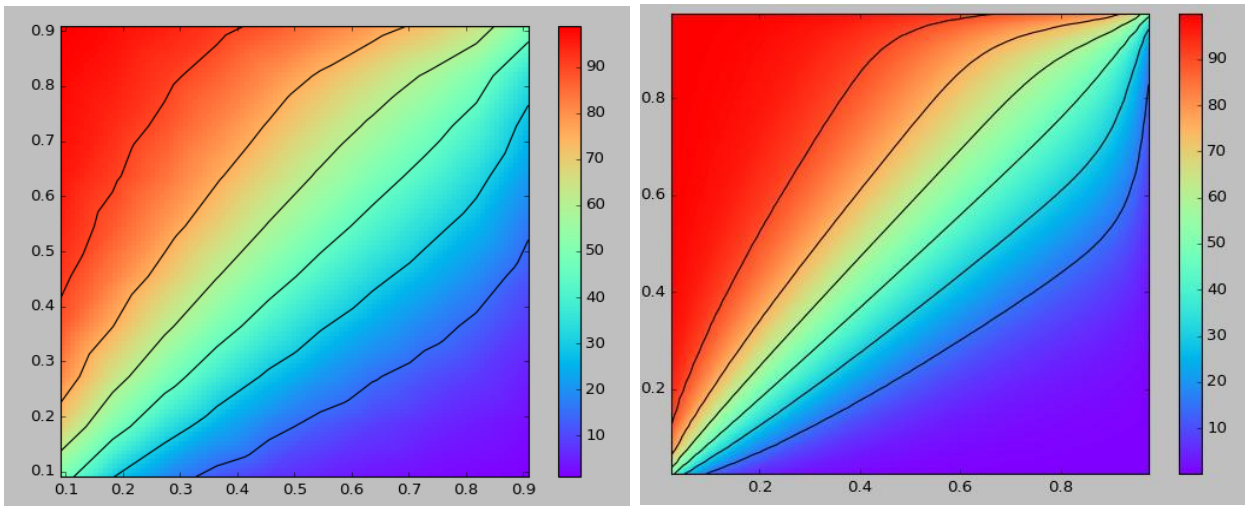

Figure 12: (LEFT) 10x10 Nodes, K = 5, 16 iterations. (RIGHT) 40x40 Nodes, K = 5, 103 iterations.

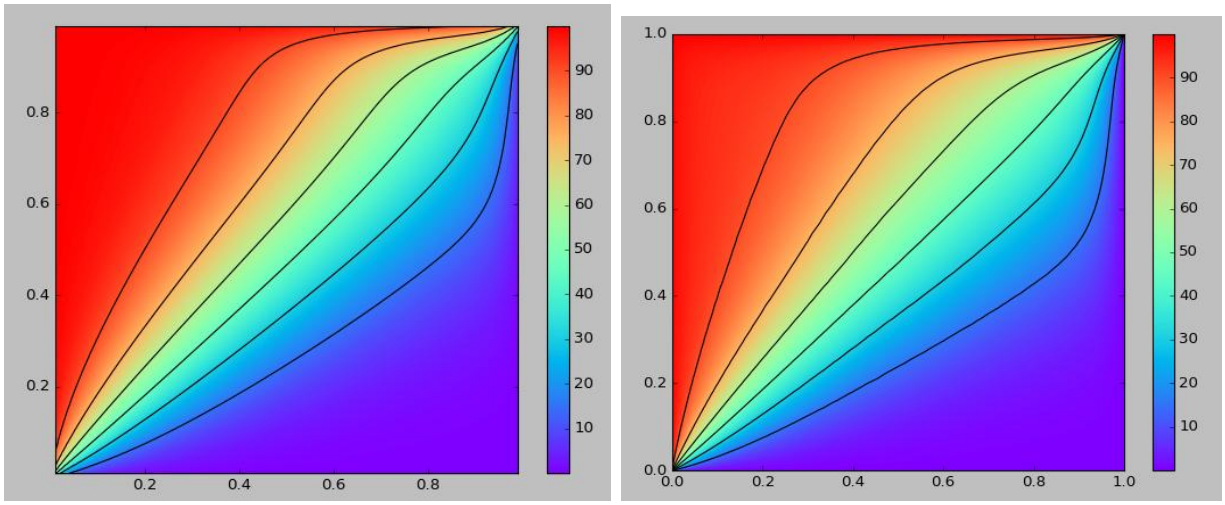

Figure 13: (LEFT) 100x100 Nodes, 458 iterations, 40 seconds, uniform mesh (r = 1). (RIGHT) 100x100 Nodes, 415 iterations, 35 seconds, nonuniform mesh (r = 1.1)

However, it can be noted that the presence of real diffusion causes the temperature contours to curve near the corners. This is due to the effect of the boundaries, and is more detailed for finer meshes. This behavior is expected, and was seen previously in the Central Differencing Scheme. The number of iterations for solving the system has increased dramatically with the addition of

diffusion, as the system has become more complex. Using a non-uniform grid with r = 1.1, it can be seen once again that the diffusive effects near the corners are far more detailed. In addition, the computing time has also been decreased from 40 to 35 seconds. This is due to the larger spacing of the interior nodes, which provide less information and therefore can be less accurate; more accurate results and heavier computation time for boundary nodes.

The Upwind Scheme was then analyzed using the boundary and initial conditions from Assignment 2. This was done to validate the code and determine if the results from Assignment 2 could be recovered using a diffusion factor of K =5 instead of 20.

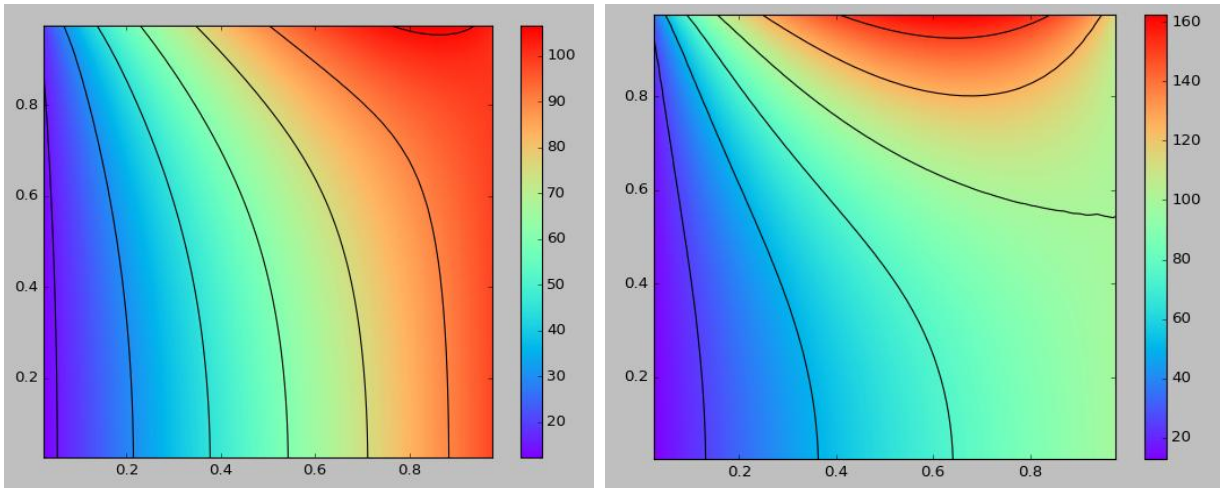

Figure 14: (LEFT) Assignment 2 result for 40x40 Nodes, r = 1, K = 20. (RIGHT) Coupled solver results for 40x40 Nodes, r = 1, K = 5, u = v = 0.

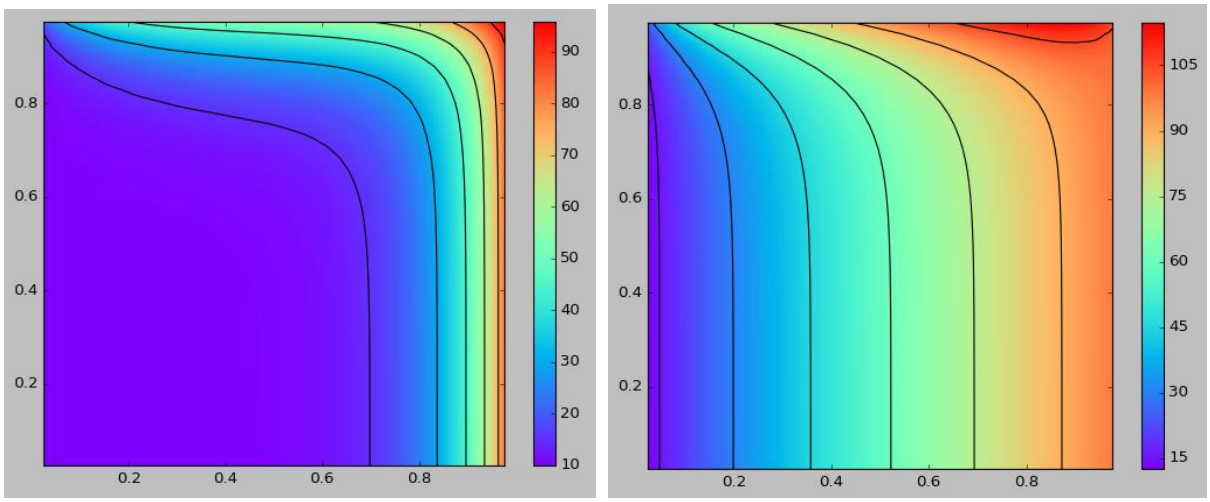

Figure 15: (LEFT) Coupled Solver Results, 40x40 Nodes, r = 1, K =5, u = v = 1. (RIGHT) Coupled Solver Results with u = -0.05, v = 1.1

It can be seen that once again selecting u velocities near zero, and v velocities near 1 (therefore a south-north dominated flow) will result in similar contours to that of Assignment 2. Compared to

the Central Difference scheme, the temperature gradient curves are much larger and less sharp. This is due to the increased amount of diffusion in the Upwind Scheme, caused by the false diffusion phenomena.