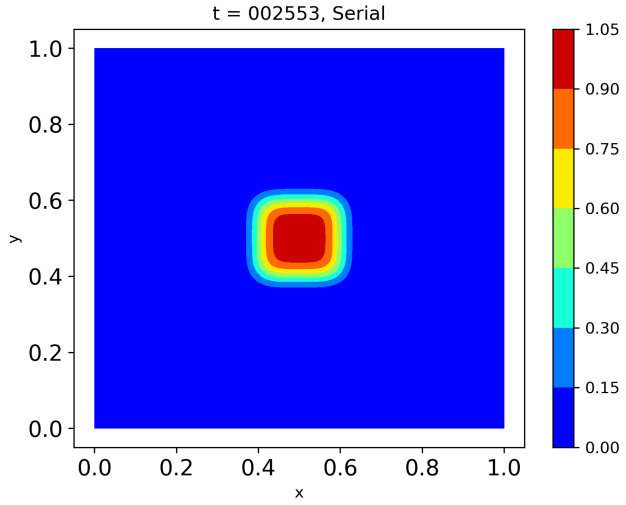
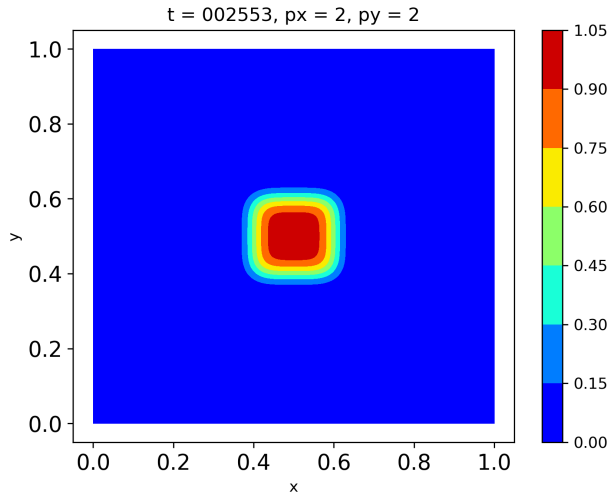
CO21BTECH11002
Aayush Kumar
ME5470 HW5 Report

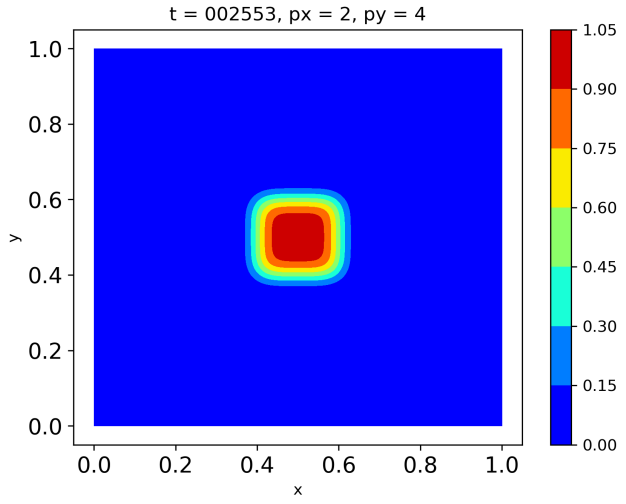
Q1)

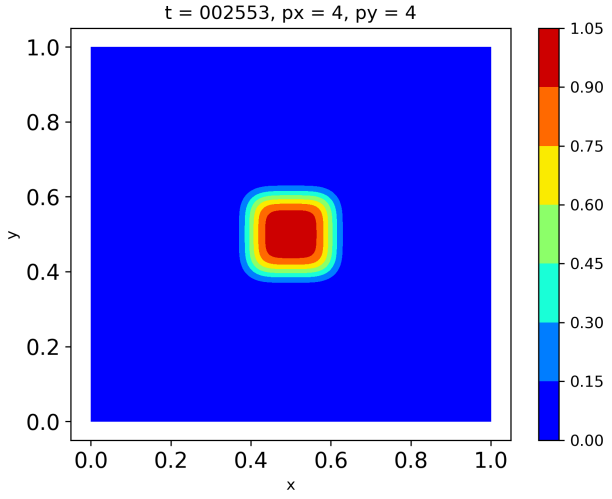
Taking dt = 0.1 / kdiff * (min_dx_dy * min_dx_dy) to ensure that it satisfies the stability condition
For given dx, dy, kdiff we have dt = 0.00000015664135844
Number of time steps = (0.001/dt) + 1 = 6385

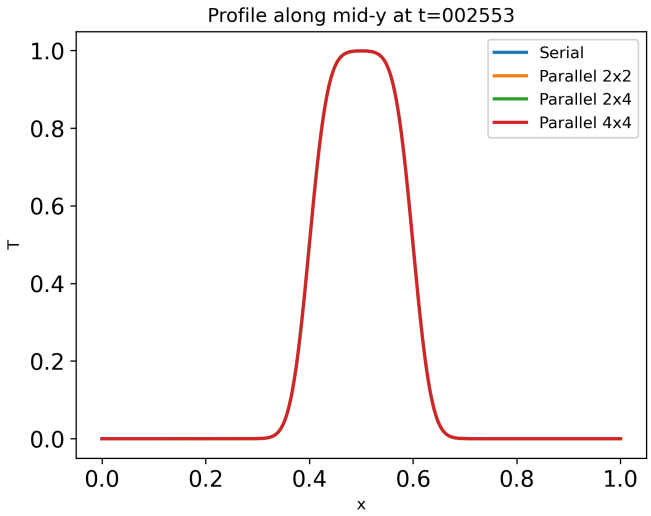
a)
Time step: 2553



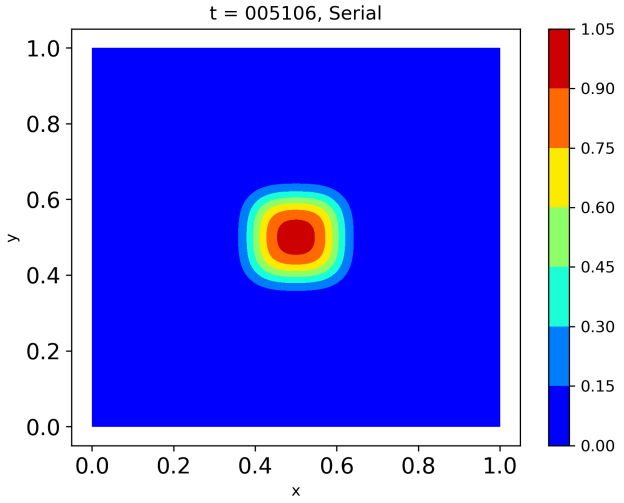


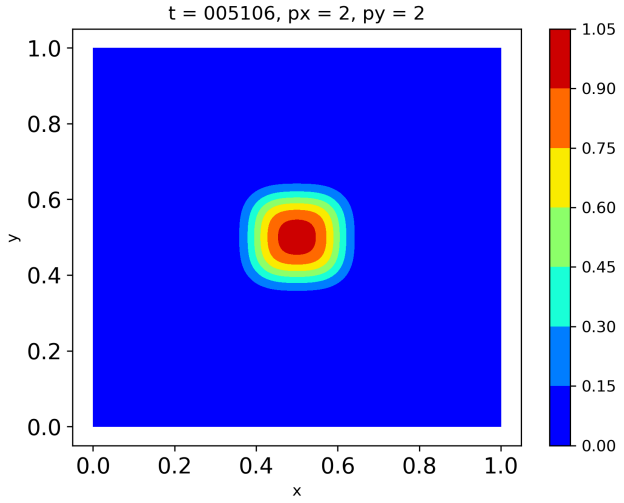


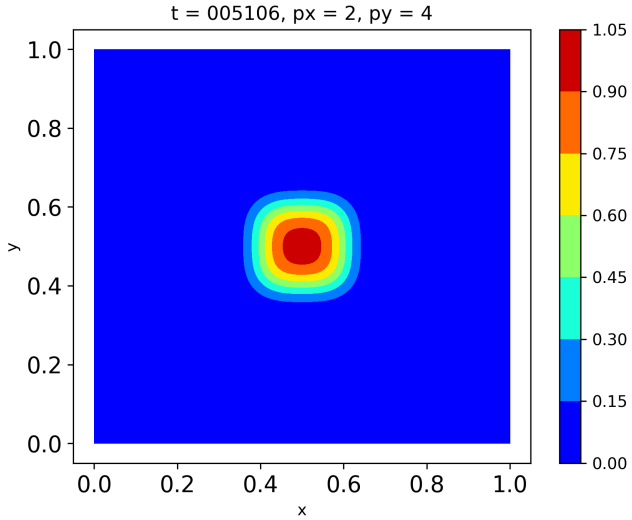


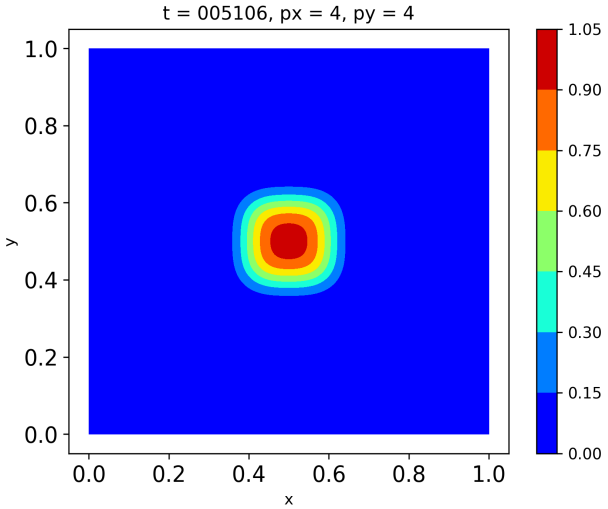


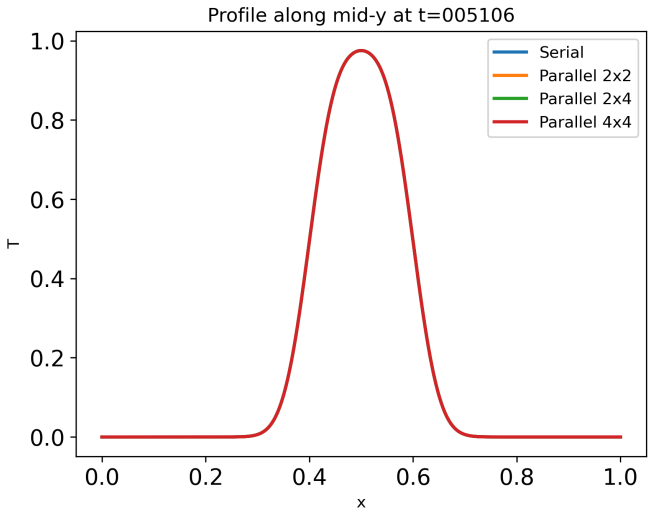
Time step: 5106



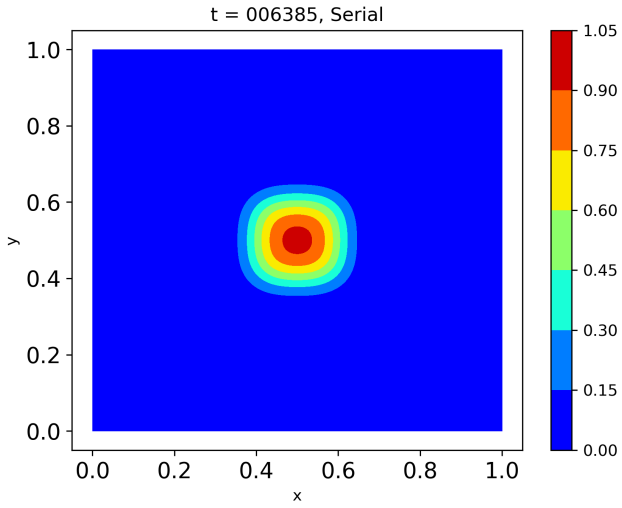


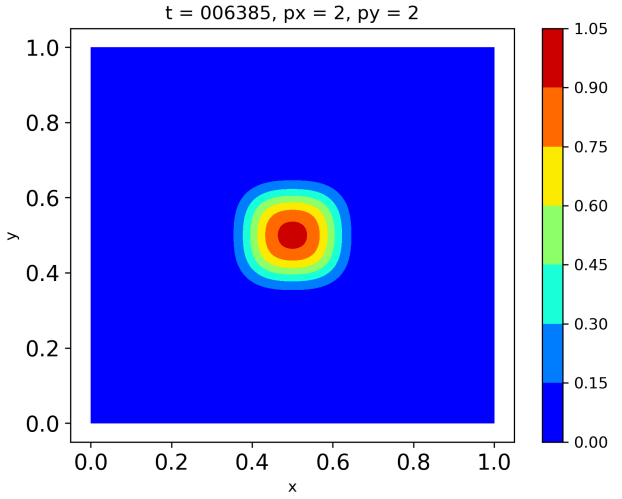


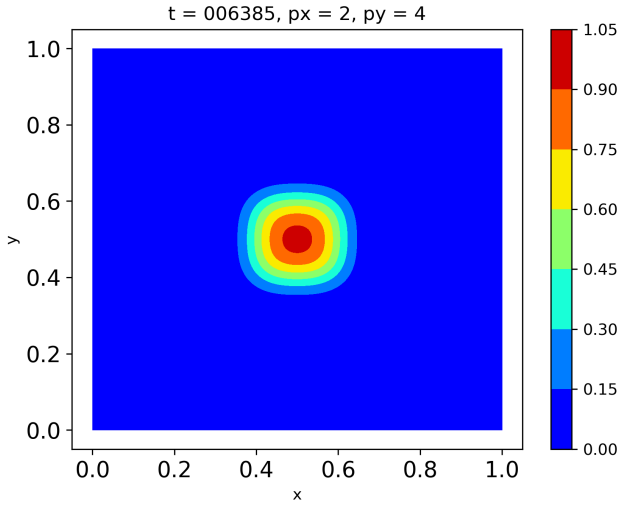


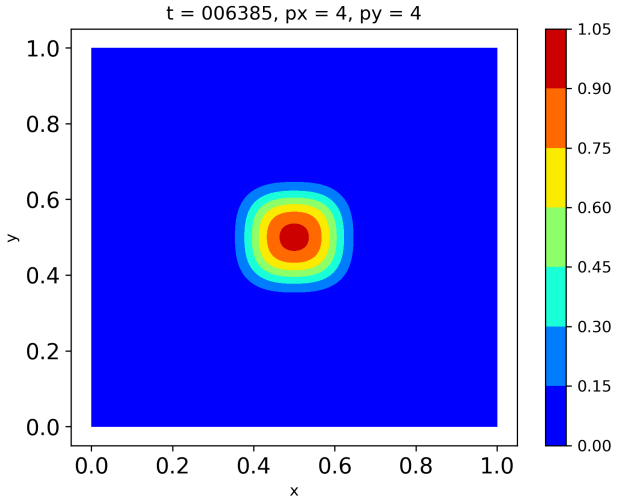


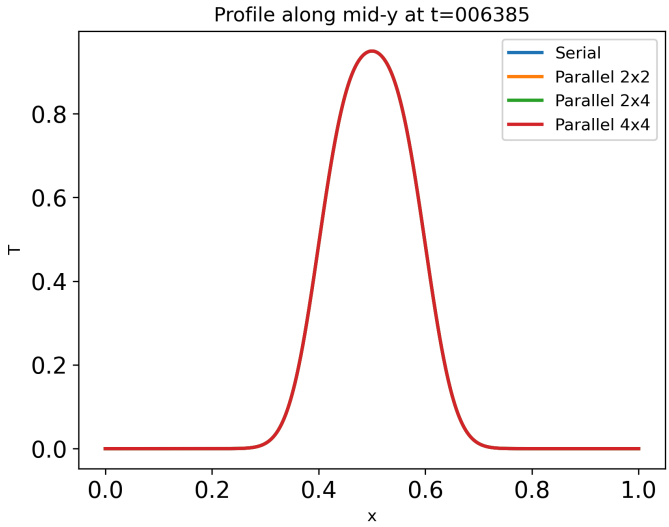
Time Step: 6385











b)

At the end of time steps, here the values at some random coordinates

Coordinates	Serial	Parallel (2 x 2)	Parallel (2 x 4)	Parallel(4 x 4)
0.500626, 0.511890	0.944706	0.944706	0.944706	0.944706
0.604506, 0.329161	0.026061	0.026061	0.026061	0.026061
0.635795, 0.703379	0.002208	0.002208	0.002208	0.002208
0.414268, 0.418023	0.410341	0.410341	0.410341	0.410341

From the table as well as the plots, we can see that there is no difference in results of serial and parallel codes for up to 6 decimal places.

c)

The time taken per time step for parallel runs are averaged over all the processes.

Execution Type	Time per time step (in seconds)
Serial	0.015403
Parallel (2 x 2)	0.007456
Parallel (2 x 4)	0.008368
Parallel (4 x 4)	0.014075

The results show that parallel execution generally reduces the time per time step compared to the serial implementation. The configuration with 2x2 processes achieves the fastest execution time, suggesting that a moderate number of processes optimally balances workload and synchronization overhead. However, performance degrades slightly with 2x4 and 4x4 configurations, likely due to increased communication and synchronization costs.