4. Step 4: Solve the system in Firedrake with the provided or other Firedrake codes. Plot the numerical results for uh(x; y) with Paraview as a contour plot (with clear labelling/indicating of the values used in that plot). Plot the difference of numerical and exact solutions) as a contour plot in Firedrake for a few suitable resolutions ({). Mention the function spaces used and the order of accuracy. Explore different order (p-refinement) and mesh resolutions (h-refinements). Explain and show which hp combinations are (roughly) equivalent and why? Provide clear figure captions with information on resolution, etc., such as h:p

Plotting functions were added to the code in order to plot as a contour and as a 3D scatter (poissoneq_1).

Results below are shown for a 16x16 Resolution (256 elements), using first order function space. For this grid:

Mesh resolution: Δx = 0.0625

L2 error: Method1 = 0.0015930107731038869, Method2 = 0.0015930107731052336

L2 norm between the two results: 1.3635909300800727e-15

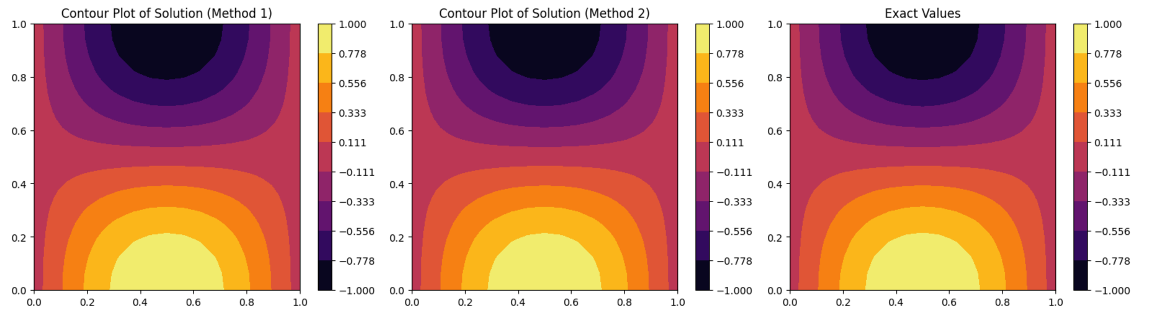
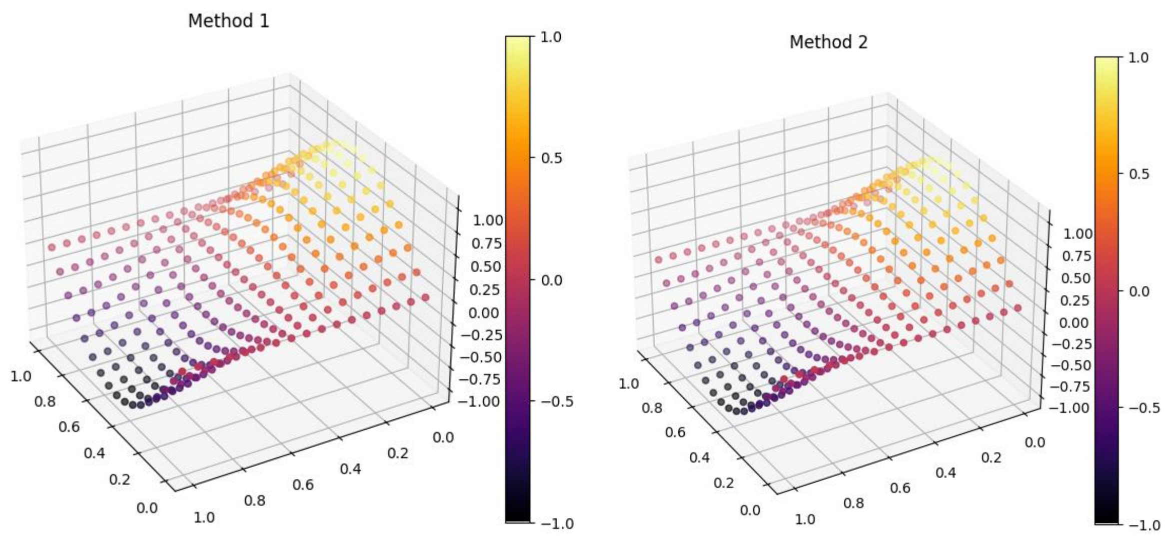


Figure 1: Contour Plots for a 16x16 resolution (256 elements), first order simulation



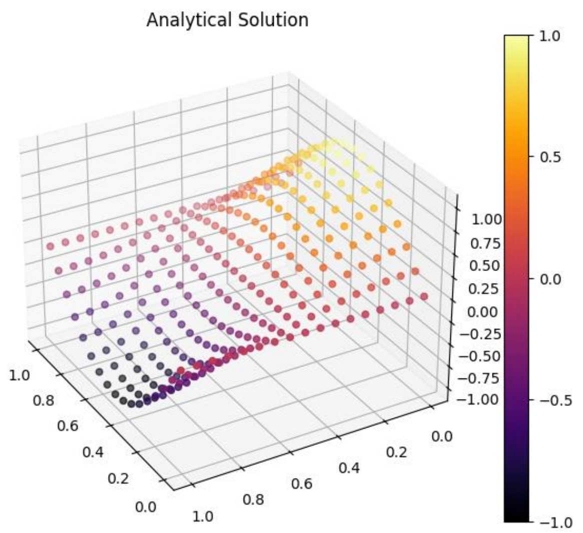


Figure 2: 3D Plots for a 16x16 resolution, first order simulation

For both visualisations the solutions look virtually indistinguishable. The difference between numerical and analytical results was also plotted. It can be seen that the largest differences occur at the highest and lowest values in the solution.

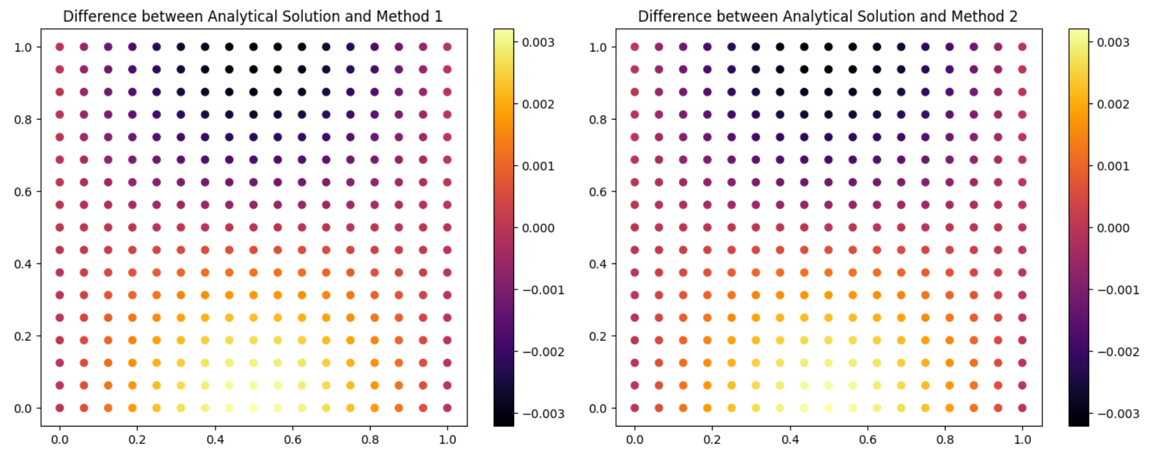


Figure 3: Difference between analytical and numerical results for a 16x16 resolution, first order simulation

The resolution was changed to 100x100 (10,000 elements), with results shown below. For this grid:

Mesh resolution: Δx = 0.01

L2 error: Method1 = 4.111455863008147e-05, Method2 = 4.111455869912797e-05

L2 norm between the two results: 6.967537548863908e-14

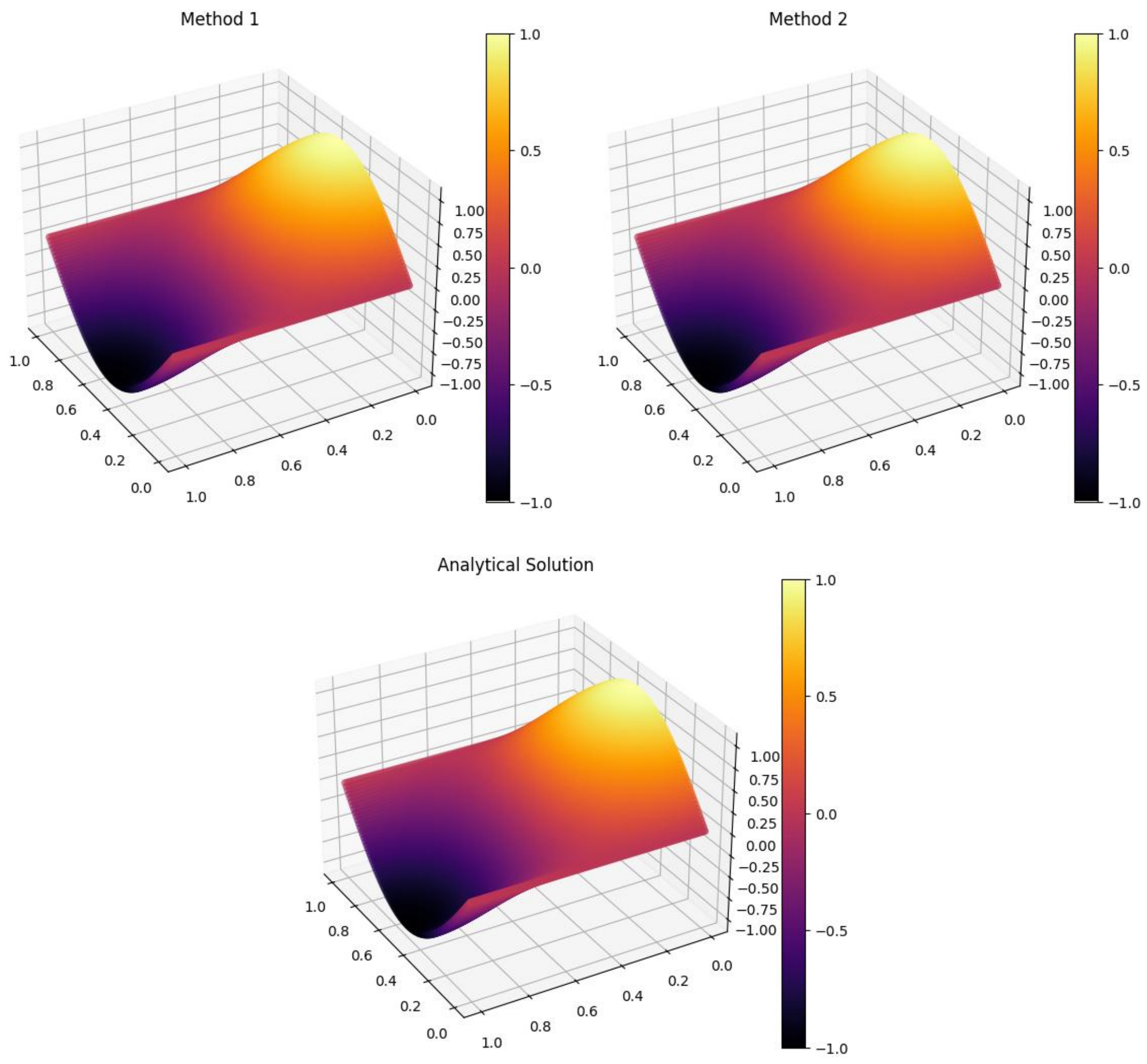


Figure 4: 3D Plots for a 100x100 resolution (10,000 elements), first order simulation

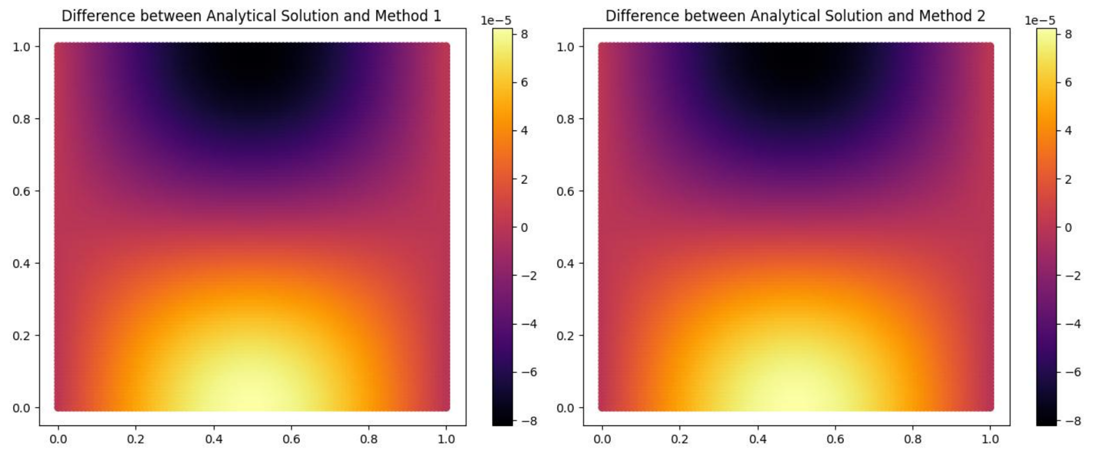


Figure 5: Difference between analytical and numerical results for a 100x100 resolution, first order simulation

Increasing the resolution reduces the error. Changing the order also reduces error, for the 16x16 resolution using a second order calculation reduced the error for both methods the following L2 values found:

L2 error: Method1 = 1.0427252041804732e-06, Method2 = 1.0427252512841122e-06

L2 norm between the two results: 6.279138467476401e-11

The code was adapted to test difference values of h and p (poissoneq_testph) with for values of p=[1,2,3,4] and nx=ny= [16,20,50,100,200,500] i.e. h=[0.0625, 0.05 , 0.02 , 0.01 , 0.005 , 0.002] . It can be seen that the error generally decreases for higher order methods and more refined grids, due to decrease in discretisation error. However for 3rd and 4th order methods, error begins to increase as the grid resolution decreases. This is likely due to computational accuracy/rounding errors. It was found that for a first order method to have the approximately the same error as the 2nd order method with resolution 16x16 listed above , it would require a resolution of 620x620 with the followijg values found:

Mesh resolution: Δx = 0.0016

L2 error: Method1 = 1.0527535870634885e-06, Method2 = 1.0527557899380368e-06

L2 norm between the two results: 2.63937994921989e-12

.

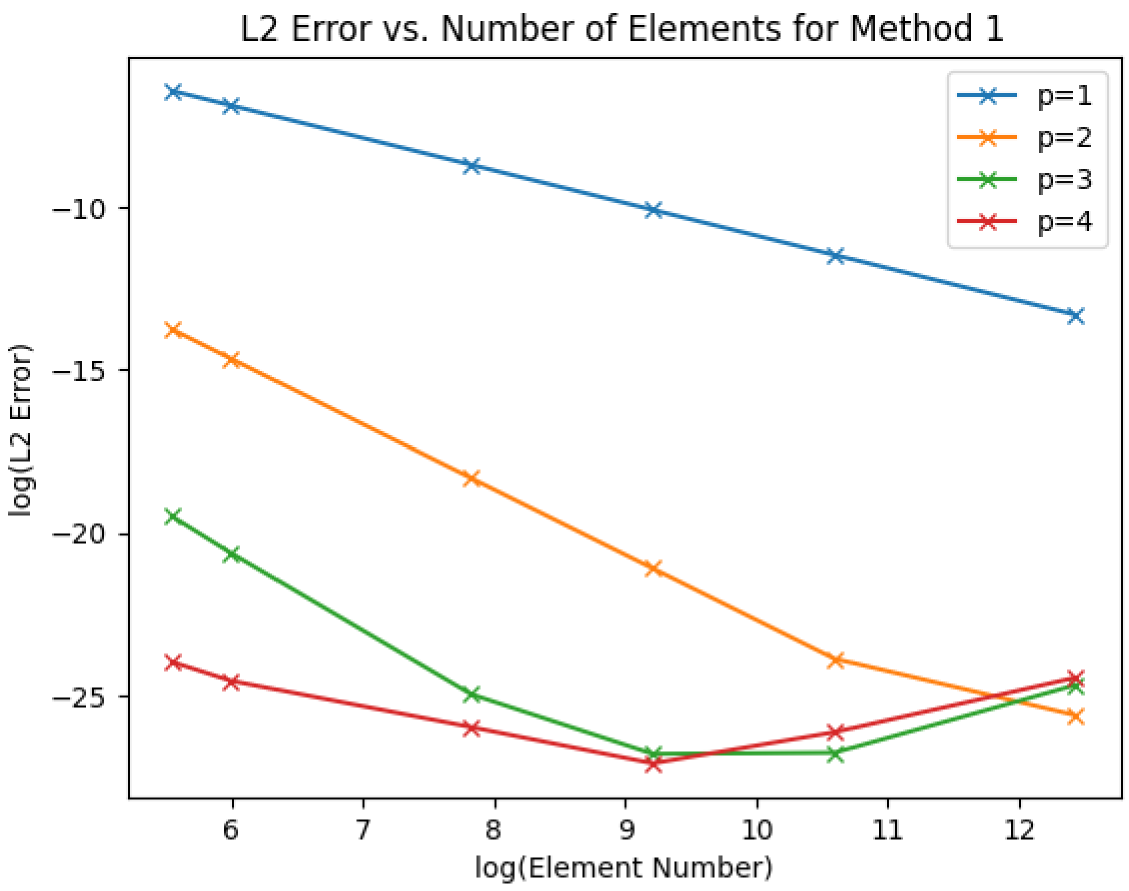


Figure 6: Log(L2 error) vs Log(Element Number) for Method 1

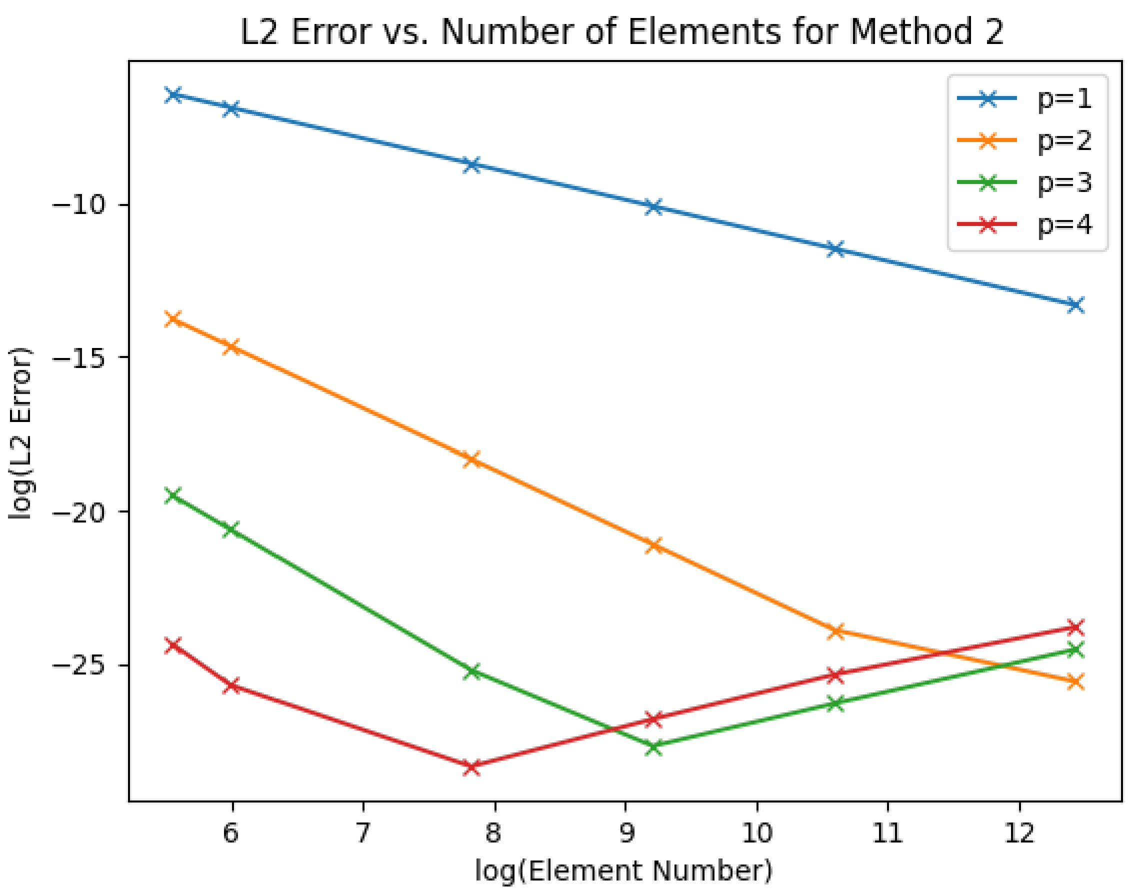


Figure 7: Log(L2 error) vs Log(Element Number) for Method 2

So what is the order? What happens for p=3, 4? -0.25

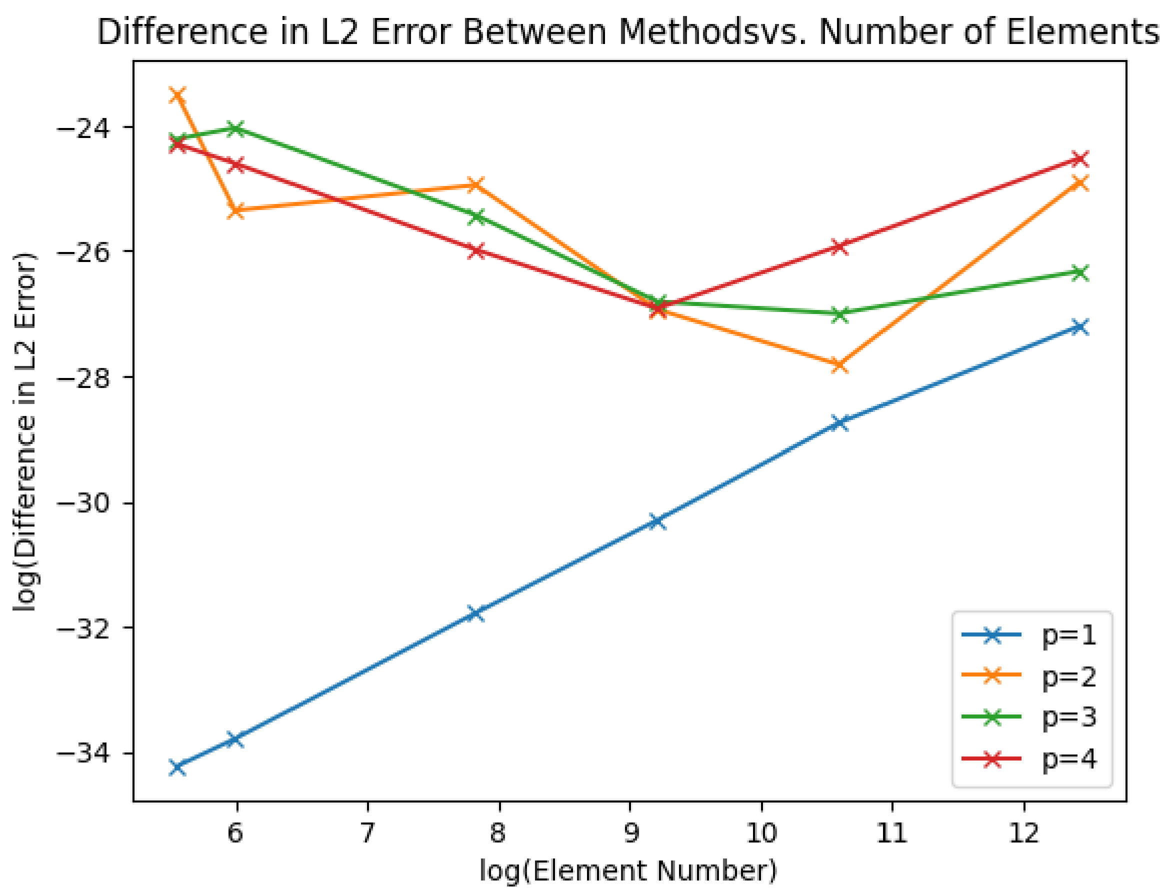


Figure 8: Difference in errors between method 1 and 2 vs number of elements

5. Explain how the above first four steps are implemented in Firedrake, also by adding clear comments to your code.

The different steps have been indicated in comments in the code (poissoneq_1)

6. Change and implement the boundary conditions for a different function f(x; y) and exact solution u(x; y). Test it with clear instructions how to reproduce your results. Test it for use from a terminal.

A example was used with Dirichlet boundary conditions of 0 on all boundaries, and a exact solution u=16x(1-x)y(1-y) resulting in a function f=32(y(1-y)+x(1-x))[1]. This code is saved as poissoneq_edited. The below results are shown for a first order solver with resolution nx=ny=200 (deltax=0.05).

Bit too simple perhaps.

[1] Tutorial 1: Poisson problem with Dirichlet conditions and code validation — MA8502 - Numerical Solutions of Partial Differential Equations

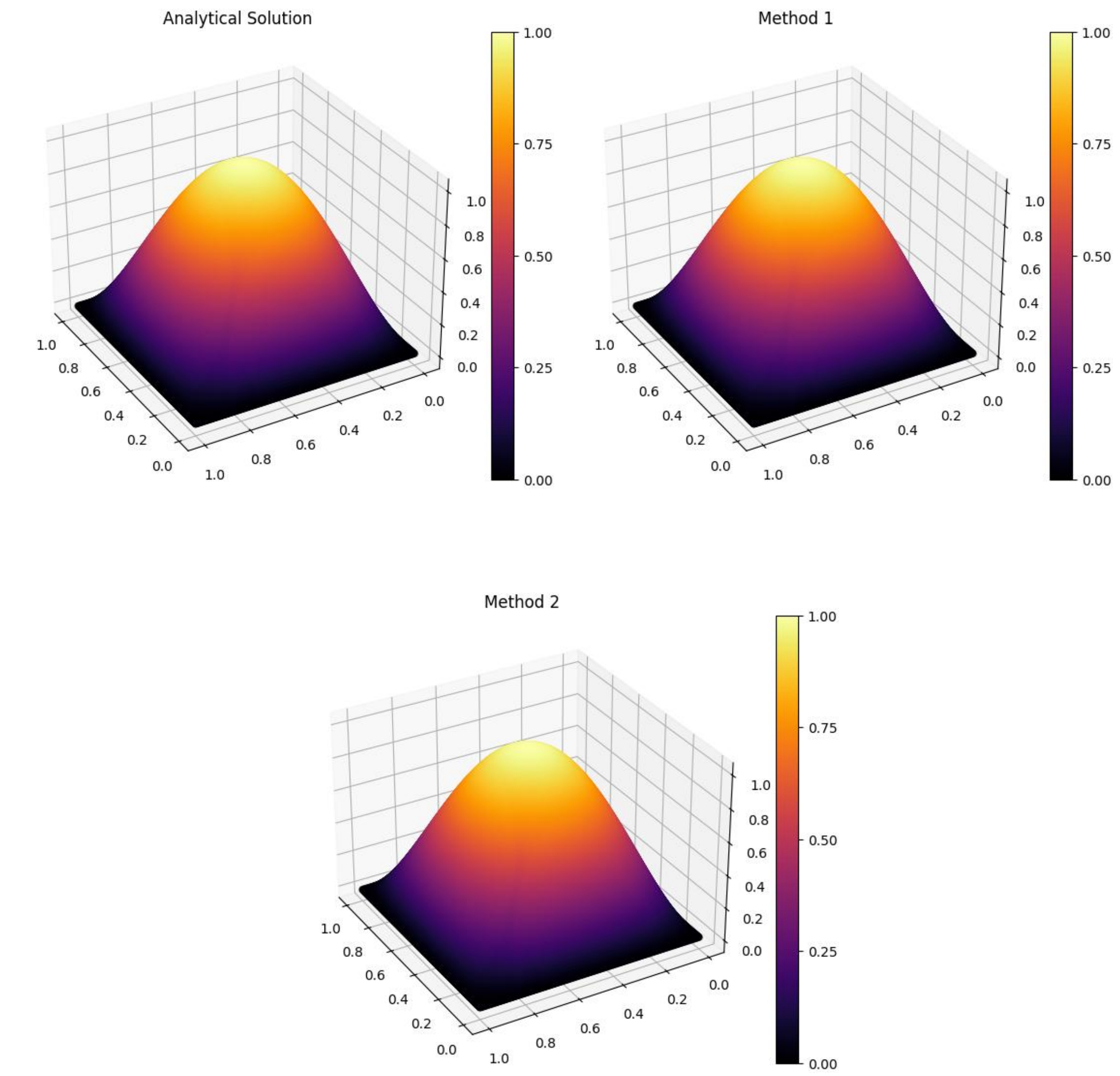


Figure 4: 3D Plots for a 200x200 resolution (40,000 elements), first order simulation

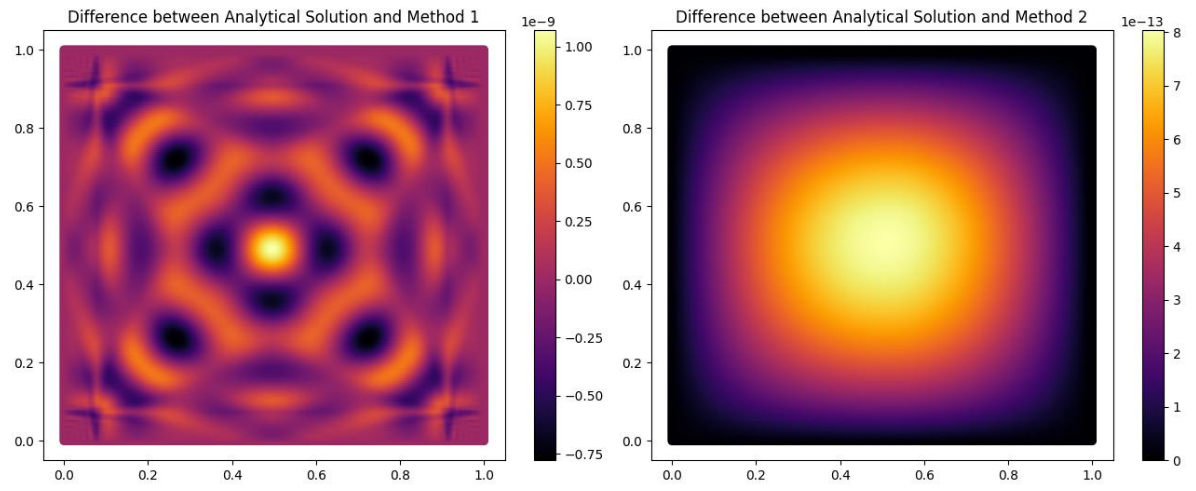


Figure 5: Difference between analytical and numerical results for a 200x200 resolution, first order simulation. It can be seen that error follows a different pattern for both methods

What about its convergence? -0.25

14/20