Chapter 1

Numerical Results

In this chapter we do several numerical experiments with the algorithms covered in this thesis. And briefly discuss the code used to do the experiments.

1.1 Computer Code

Most of the code used in this thesis can be found on https://github.com/trulsmoholt/masterthesis, and is written in python and numpy. It is primarily intended for educational purposes and for comparing convergence rates of different spatial approximation techniques. An example of a very simple use case can be seen in 1.1

```python
 1
 2 from discretization.mesh import Mesh
 3 from discretization.FVML import compute_matrix,compute_vector
 4 import numpy as np
 5 import math
 6
 7 #Function to perturb mesh from unit square. Takes a 2d numpy
       vector and returns a 2d numpy vector. This particular choice
       makes a paralellogram mesh.
 8 perturbation=lambda p: np.array([p[0],0.5*p[0]+p[1]])
 9
10 #Number of grid points in x and y direction
11 nx=ny=10
12
13 mesh = Mesh(nx,ny,perturbation,ghostboundary=True)
14 source = lambda x,y:math.sin(y)*math.cos(x)
15 boundary_condition = lambda x,y:0
16 tensor = np.eye(2)
17 permeability = np.ones((mesh.num_unknowns,mesh.num_unknowns))
18
19 A = np.zeros((mesh.num_unknowns,mesh.num_unknowns))#stiffness
       matrix
20 f = np.zeros(mesh.num_unknowns)#load vector
21
22 compute_matrix(mesh,A,tensor,permeability)
23 compute_vector(mesh,f,source,boundary_condition)
24
25 u = np.linalg.solve(A,f)
26 mesh.plot_vector(u)
27
```

Listing 1.1: Solving simple Poisson equation.

The code is centred around the mesh class, which contains information about how the domain is discretized into quadrilaterals and its properties. This class also contains public functions to make plots of different kinds and compute errors. The spatial numerical methods implemented are: TPFA, MPFA-L, MPFA-O and linear Lagrange finite elements. They all have the same call signature, as in 1.1 line 22 and 23. The control volume methods also has the option of taking a matrix to store the flux stencils. One can use sparse matrices instead of dense numpy matrices in 1.1, as long as the indexing signature is the same as in numpy, for example scipy has a compatible sparse matrix library.

The code also has implementations of mass matrix and the gravitation term. Also included in the github are an example of how to solve Richards' equation using L-scheme linearization and backward Euler, see 1.2 for some of the code.

```python
u_l = u.copy() #linearization/L-scheme iterate
u_t = u.copy() #timestep iterate
F = u.copy() #source vector
A = np.zeros((mesh.num_unknowns,mesh.num_unknowns)) #stiffness matrix
B = mass_matrix(mesh)

#time iteration
for t in time_partition[1:]:
  #empty source vector
  F.fill(0)
  #compute source vector
  compute_vector(mesh,F,lambda x,y: f(x,y,t),lambda x,y:u_exact(x,y,t))
  #L-scheme iteration
  while True:
    #compute the heterogeneous hydraulic conductivity, kappa
    conductivity = kappa(np.reshape(u_l, (mesh.cell_centers.shape[0],mesh
    .cell_centers.shape[1]),order='F'))
    A.fill(0)#empty the stiffness matrix
    compute_matrix(mesh, A, K,conductivity)#compute stiffnes matrix
    lhs = L*B+tau*A
    rhs = L*B@u_l + B@theta(u_t) - B@theta(u_l) + tau*F
    u = np.linalg.solve(lhs,rhs)
    #check if L-scheme linearization has acceptable error
    if np.linalg.norm(u-u_l)<=TOL+TOL*np.linalg.norm(u_l):
      #quit linearization and do another time step
      break
    else:
      #update linearization iterate
      u_l = u
  #update time step iterate
  u_t = u
  #update linearization iterate
  u_l = u
```

Listing 1.2: Linearization and time stepping of Richards' equation.

1.2 Elliptic Equation

The convergence tests in this section are similar to some of the tests done in chapter three of [?]. We consider the elliptic model problem (??), find $u = u(x)$ such that

$$\begin{cases} \nabla \cdot \boldsymbol{q} = f, & \text{in } \Omega \\ \boldsymbol{q} = -\boldsymbol{K}\nabla u, & \text{in } \Omega \\ u = u_{\partial\Omega}, & \text{on } \partial\Omega \end{cases} \tag{1.1}$$

We set the solution

$$u = cosh(\pi x)cos(\pi y), \tag{1.2}$$

set $\boldsymbol{K}$ to be the identity matrix and compute the Dirichlet boundary condition, $u_{\partial\Omega}$, from the equation above. The domain Ω will vary through the rest of this section.

As in [?] page 1340 we define the normalized discrete L_2 norms:

$$\|u - u_h\|_{0,h} = \left(\frac{1}{V}\sum_i V_i(u_{h,i} - u_i)^2\right)^{\frac{1}{2}} \tag{1.3}$$

$$\|q - q_h\|_{0,h} = \left(\frac{1}{Q}\sum_a Q_a(q_{h,a} - q_a)^2\right)^{\frac{1}{2}}, \tag{1.4}$$

where $q_a := -\hat{\boldsymbol{n}}\cdot\boldsymbol{q}$ is the normal flow density over edge a, with $\hat{\boldsymbol{n}}$ being unit normal to the edge and $\boldsymbol{q}$ evaluated at the midpoint of the edge. $q_{h,a}$ is the discrete flux over a defined similarly with $\boldsymbol{q}_h$ being the discrete normal flow density, for a finite volume method this would be the flux across some edge divided by the edge length. For the finite element method, we use the MPFA-L flux stencil to recover the flux in the experiments where it is present. Let $u_{h,i}$ denote the discrete potential at cell i, and u_i is the potential evaluated at cell i. For the finite element method, this would be the function value at the grid points/nodes. Q_a is the volume associated with edge a, i.e., the sum of the two volumes sharing edge a. $V = \sum_i V_i$ and $Q = \sum_a Q_a$.

In the figures (5.12-5.15) in this section we see convergence rates in the $\|\cdot\|_{0,h}$ norm for different spatial numerical methods. The y-axis is the log_2 of the error and the x-axis is $log_2 n$, where n is the number of points in x and y direction, and thus proportional with the inverse of the mesh diameter. The slope of the graph in the plots are the convergence rate.

In figures 1.3 and 1.3 we see that all the methods converge with the same quadratic rate. This fits well with the fact that all the methods covered in this thesis are equivalent to the TPFA method for uniform grids.

In figures 1.5, 1.6 and 1.7 we observe that the TPFA method does not converge for parallelogram mesh. This makes sense as the grid is not K-orthogonal. The others methods still have quadratic convergence for potential and flow density.

In figures 1.8, 1.9 and 1.10 the MPFA-L, MPFA-O and FEM still converges quadratically for the potential on a rough grid, where every grid point is perturbed randomly by a factor proportional to the gird diameter. For the normal flow density however, the convergence rate drops to about one.

In figure 1.11 we introduce grids with aspect ratio, that is grids with more points in the y direction. In figure 1.12 we observe that MPFA-L, MPFA-O and

FEM has a convergence rates for the potential of about 1.5 for the grid with aspect ratio 0.1. In figure 1.14 we see that the MPFA-O method fail to converge for the grid with aspect ratio 0.01. Thus the MPFA-L method wins this round of numerical experiments.

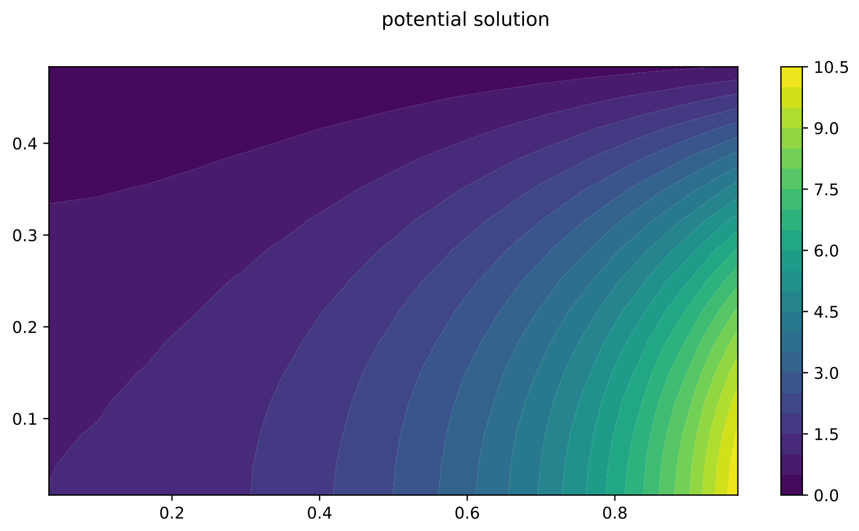

Figure 1.1: The solution (1.2) on half the unit square

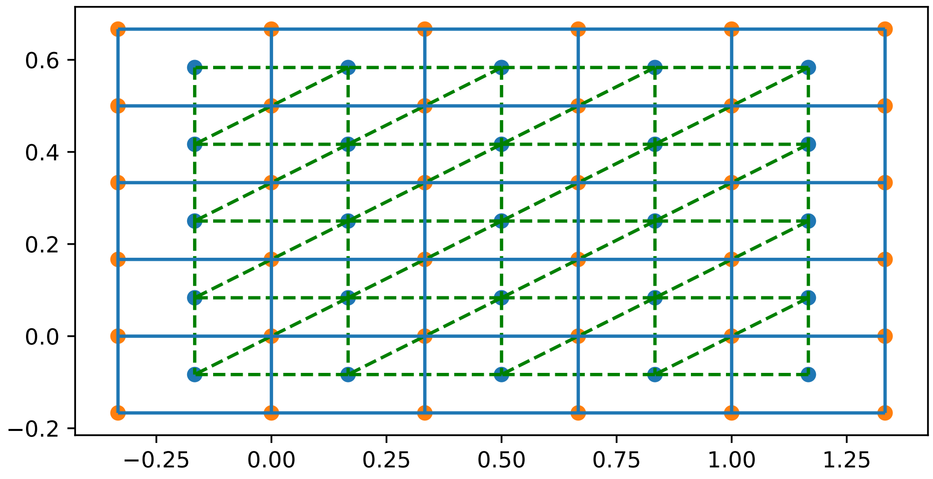

Figure 1.2: Unifrom rectangular mesh on half the unit square. The triangles are used for the finite element solution and are spanned between the nodes of the cell centers of the finite volume methods. The ghost cell boundary is included, so this mesh has nine degrees of freedom, i.e., the interior cells.

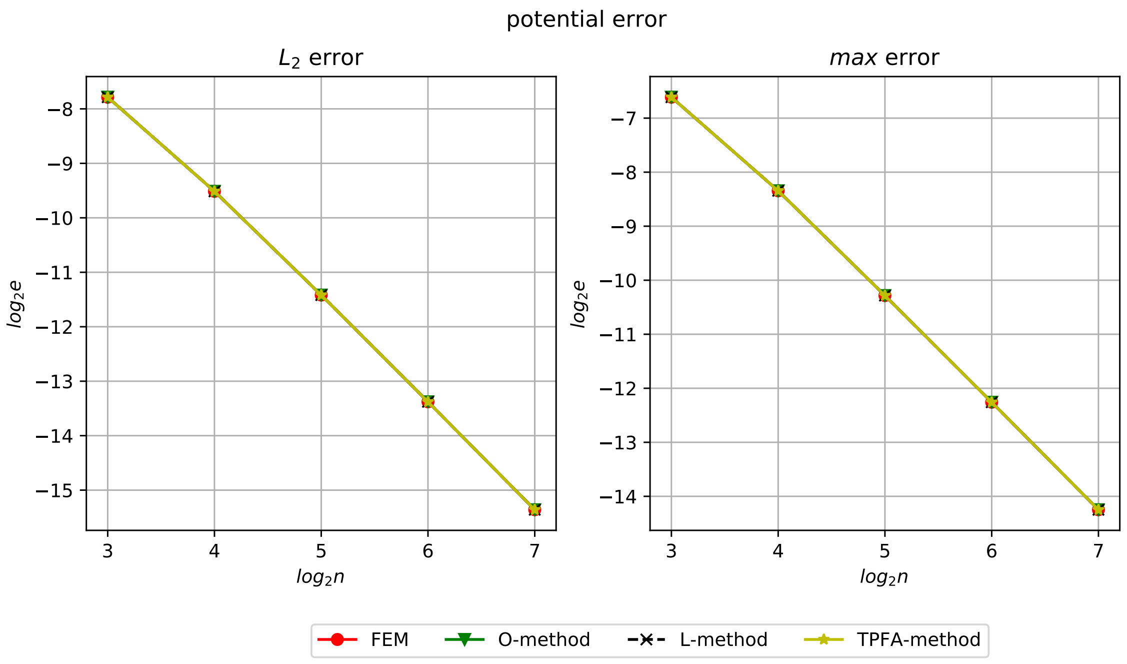

Figure 1.3: Potential error on refinements of the uniform rectangular mesh 1.2. The convergence is the same for all the schemes.

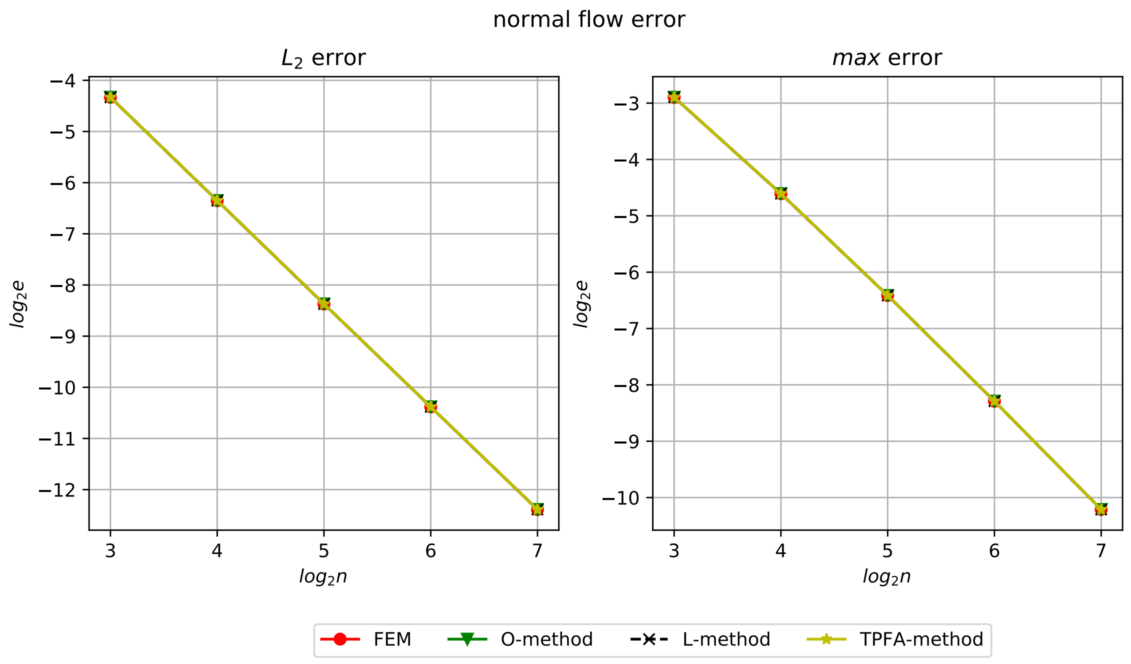

Figure 1.4: Normal flow density error on refinements of the uniform rectangular mesh 1.2. The convergence is the same for all the schemes.

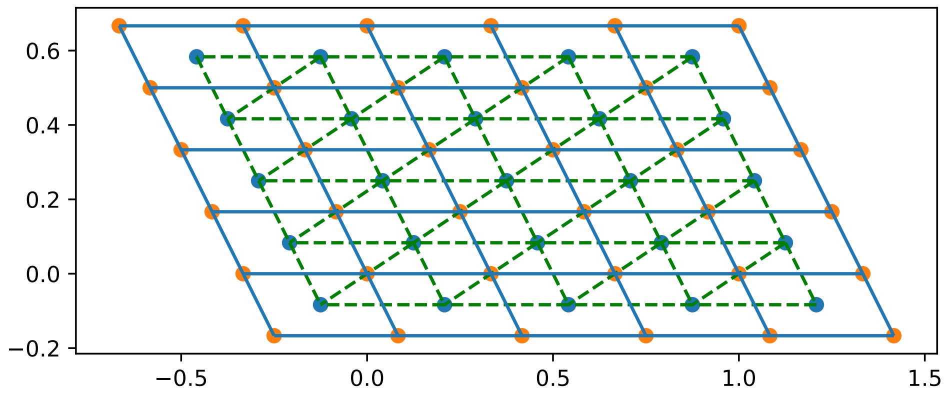

Figure 1.5: Trapezoidal mesh, now every point is transformed by $(x, y) \mapsto (x - 0.5y, y)$

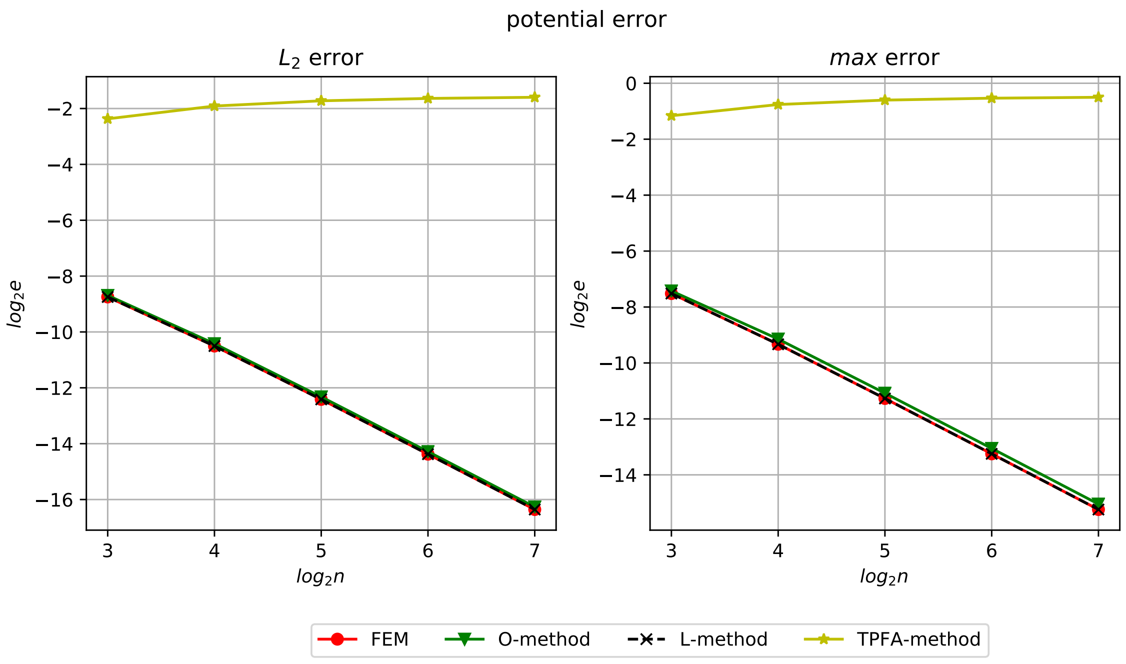

Figure 1.6: Pressure error on refinements of the mesh 1.5

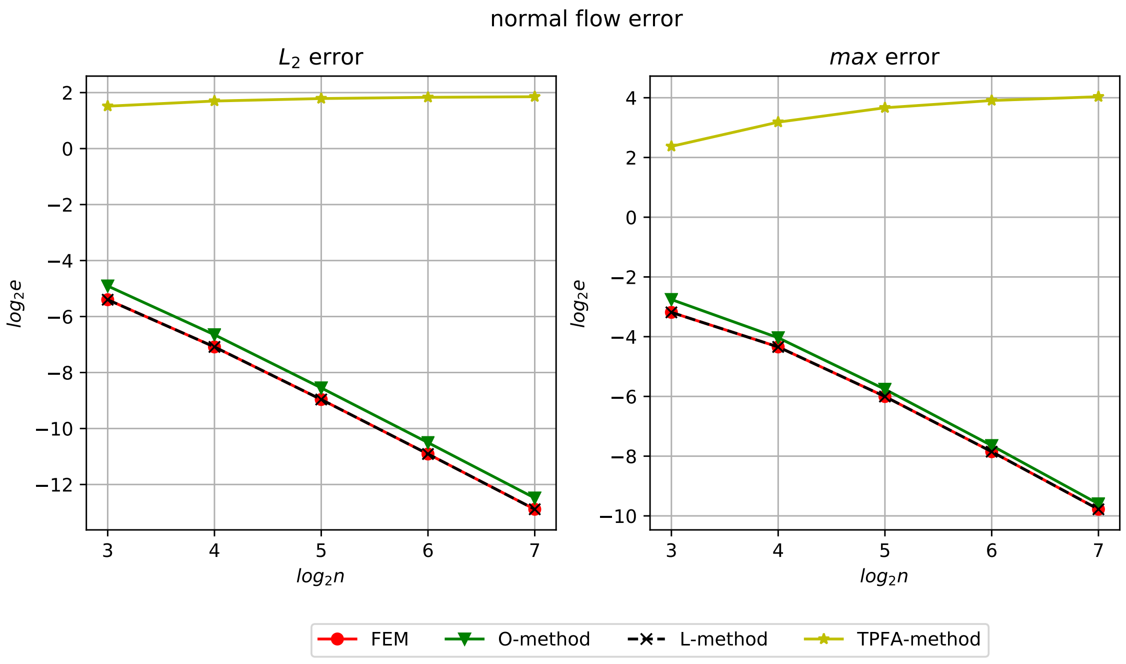

Figure 1.7: Normal flow density error on refinements of the mesh 1.5

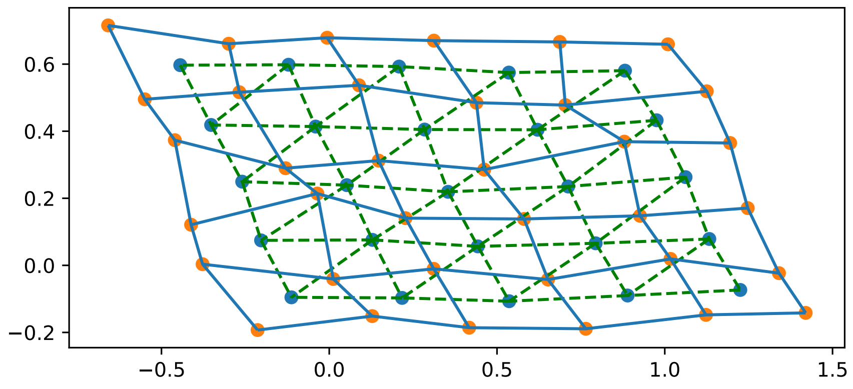

Figure 1.8: Perturbed mesh, every point in the mesh is perturbed by a random number which is $O(\frac{h}{5})$, in both x and y direction.

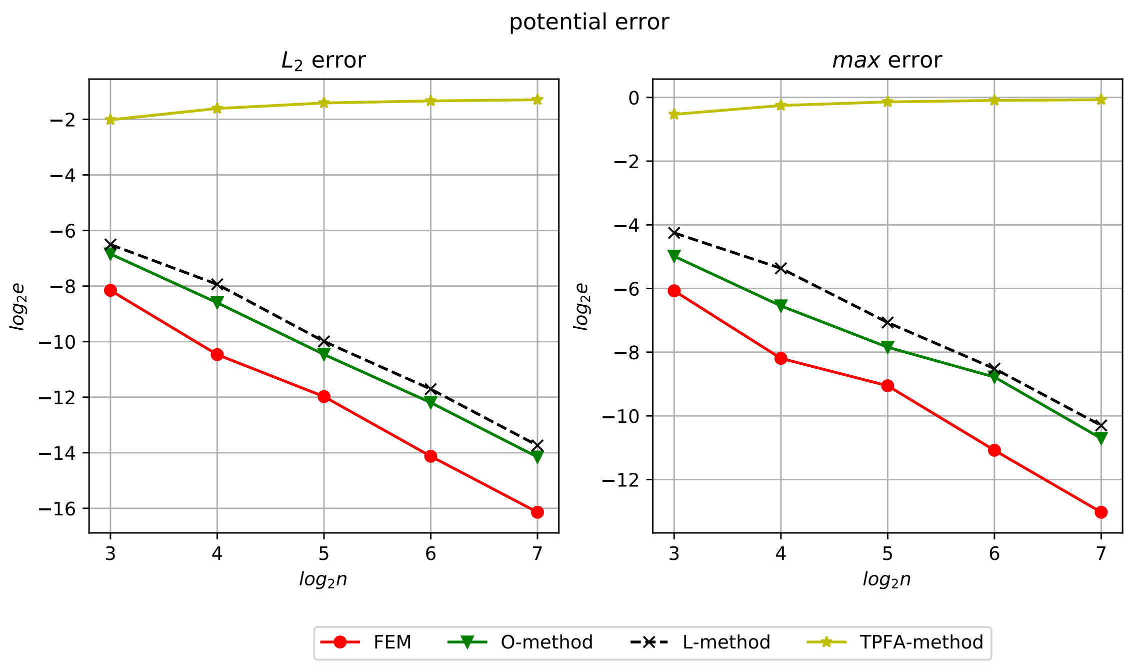

Figure 1.9: The pressure error of perturbed mesh.

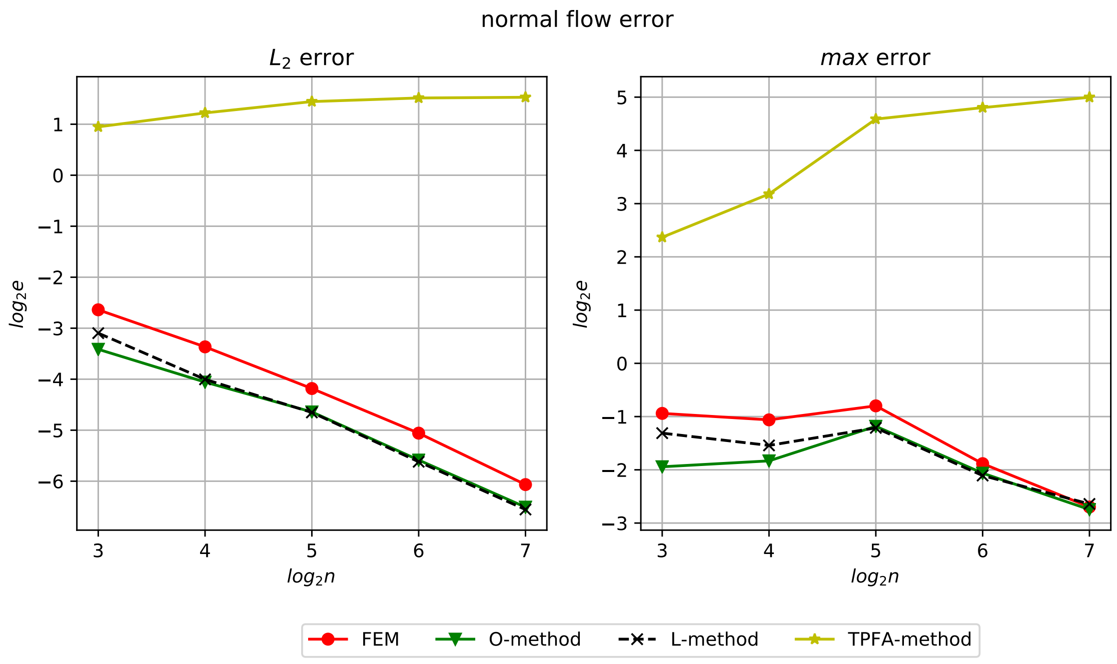

Figure 1.10: The normal flow density error of perturbed mesh

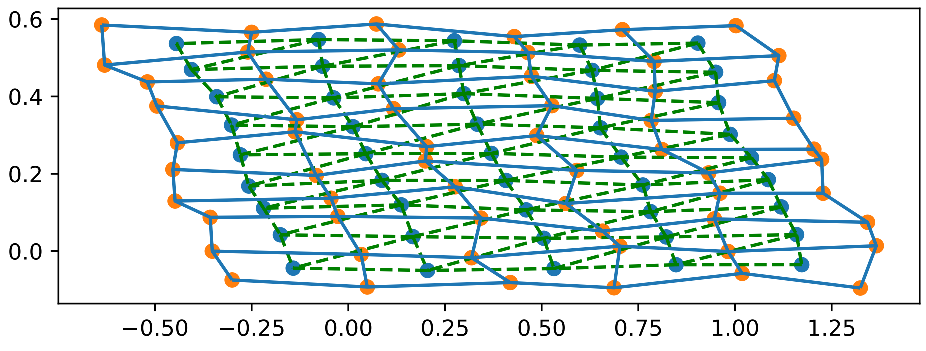

Figure 1.11: Perturbed mesh with aspect ratio 0.5, there are half as many points in the x-direction as in the y-direction.

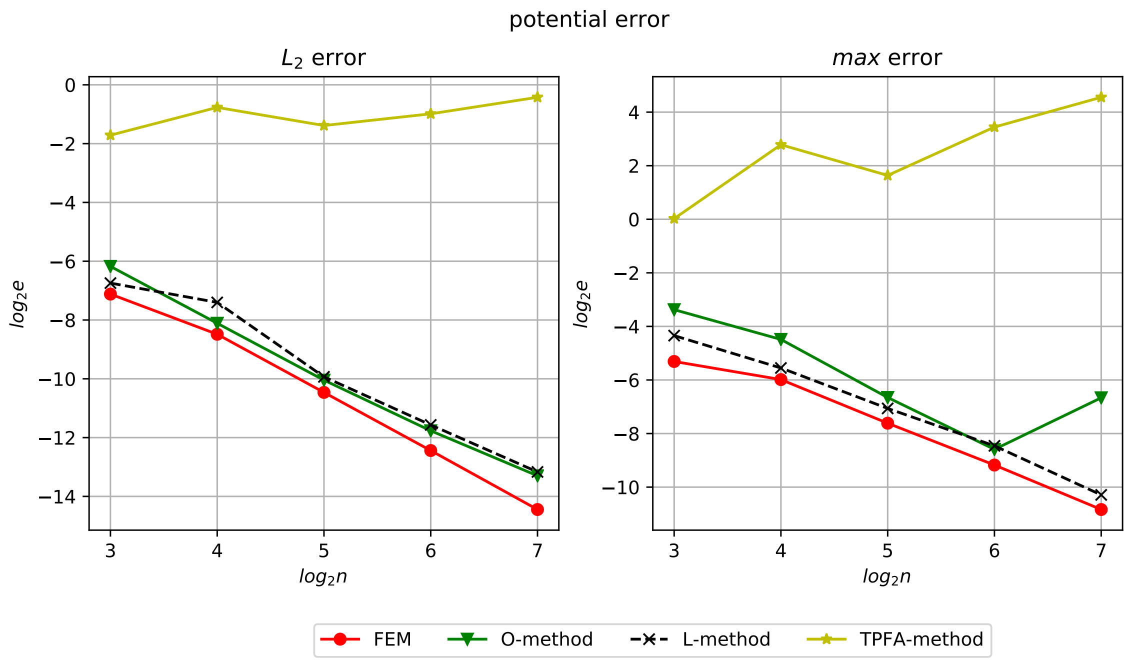

Figure 1.12: The pressure error of perturbed mesh with aspect ratio 0.1.

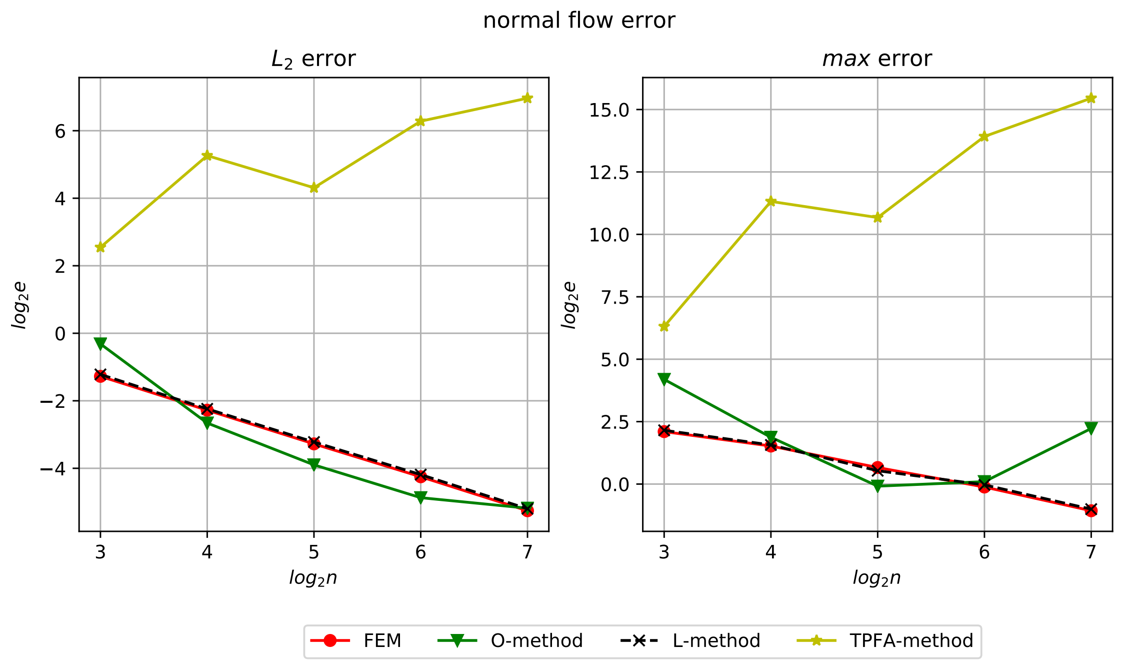

Figure 1.13: The normal flow density error of perturbed mesh with aspect ratio 0.1.

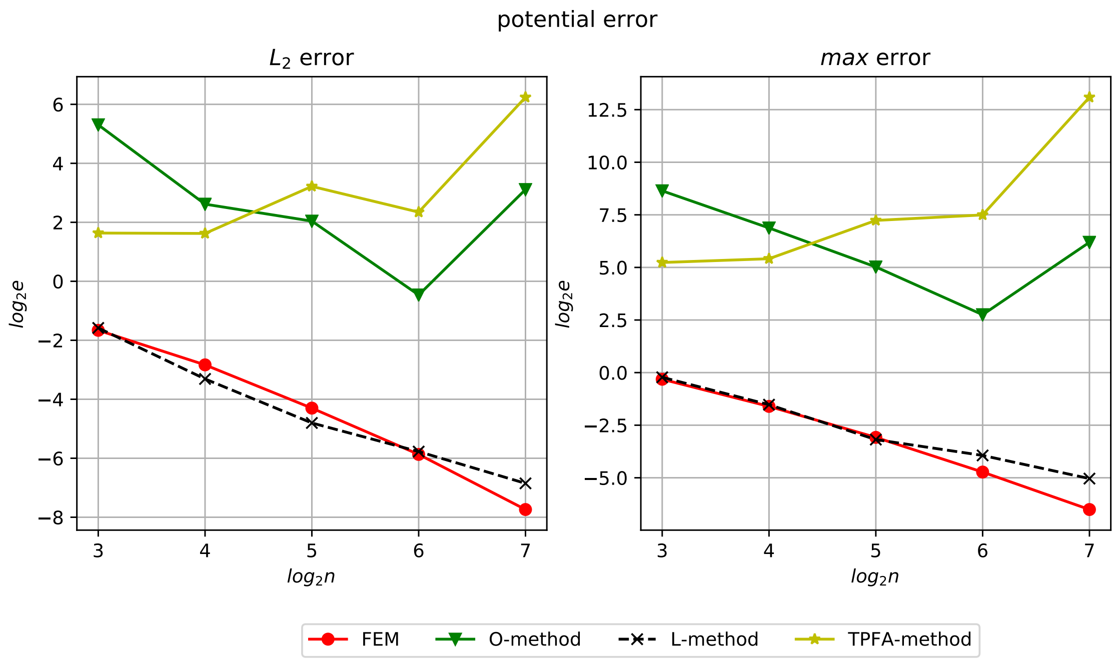

Figure 1.14: The pressure error of perturbed mesh with aspect ratio 0.01.

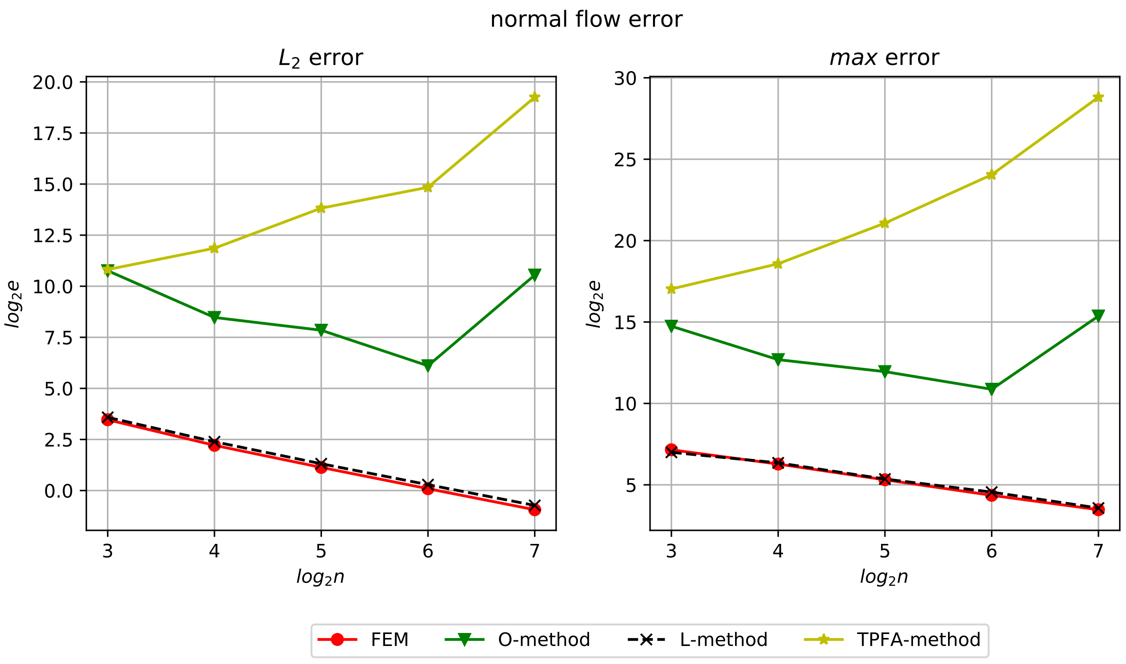

Figure 1.15: The normal flow density error of perturbed mesh with aspect ratio 0.01.

1.3 Richards' Equation

This section is not yet done

1.3.1 Constant Hydraulic Conductivity

In this section we consider numerical experiments for (**??**), with Dirichlet boundary conditions: Find $u = u(x,t)$ such that

$$\begin{cases} \partial_t b(u) - \nabla \cdot \nabla u = f, & \text{in } \Omega \times (0,T] \\ u = u|_{\Gamma_D}, & \text{on } \partial\Gamma_D \times (0,T] \\ u = u_0, & \text{on } \Omega \times \{t=0\} \end{cases} \tag{1.5}$$

We define

$$b(u) := \frac{1}{1-u}, \tag{1.6}$$

and compute the source term, f, such that the solution becomes

$$u = -tx(1-x)y(1-y) - 1, \tag{1.7}$$

which is the same equation and solution as they use in [**?**]. We let Ω be the unit square perturbed by $(x,y) \mapsto (x - 0.5y, y)$. The L-scheme linearization has the parameters $L := 1.5$ and error tolerance $TOL = 5e^{-9}$. The grid we use can be seen in figure (1.16). In table 1.4 we observe a quadratic convergence both when the time step length is set equal to the square of the mesh diameter, $\tau = h^2$. We observe the same convergence rate in table 1.2, when $\tau = h$. One would expect this, as the the time discretization has linear convergence rate. An explanation could be that the solution (1.6) is linear in time, and even with the non-linearity $b(\cdot)$, is approximated exactly by the backward Euler time discretization.

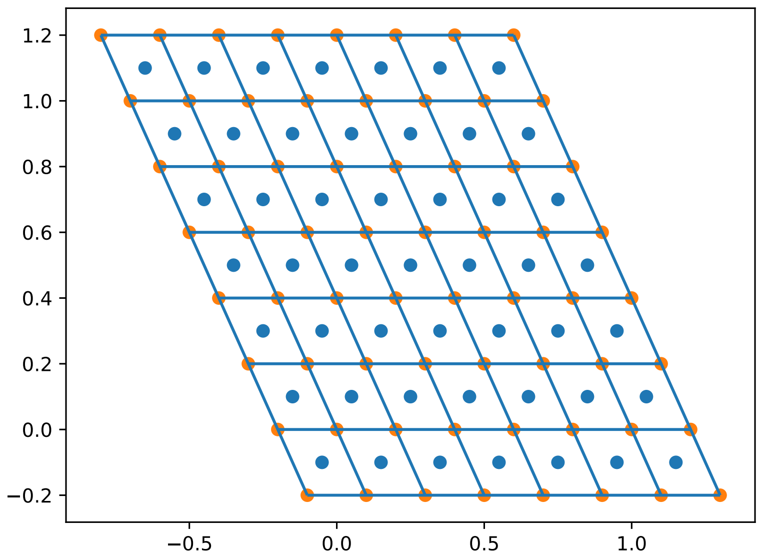

Figure 1.16: Parallelogram grid, with ghost Dirichlet boundary cells.

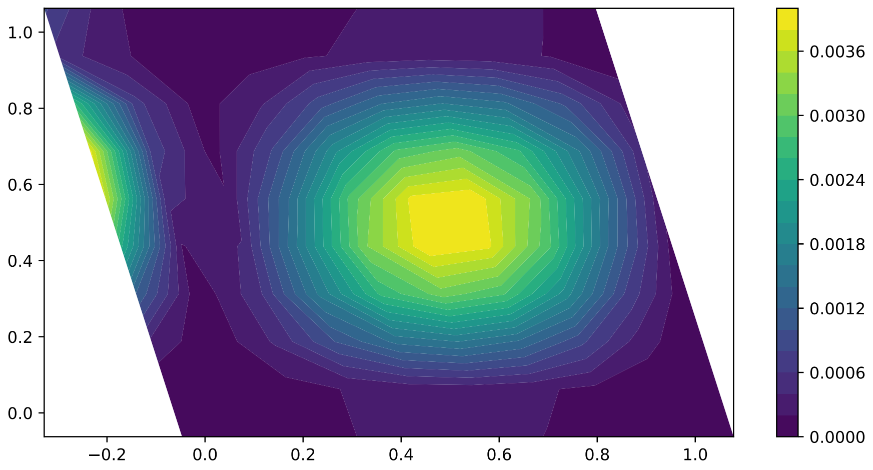

Figure 1.17: The solution of (1.5) at $T = 1$, with the ghost Dirichlet boundary.

number	mesh diameter, h	time step length, τ	discrete $L_2(\Omega)$ error	improvement
1	0.45069	0.20312	0.001695	-
2	0.22535	0.05078	0.000375	4.51623
3	0.11267	0.01270	0.000087	4.31530
4	0.05634	0.00317	0.000021	4.20040

Table 1.1: Convergence table for (1.5),(1.6) and (1.7). The time step length, τ, is set proportional to the square of the mesh diameter, that is $\tau = h^2$.

number	mesh diameter, h	time step length, τ	discrete $L_2(\Omega)$ error	improvement
1	0.45069	0.45069	0.001694	-
2	0.22535	0.22535	0.000374	4.52868
3	0.11267	0.11267	0.000086	4.33993
4	0.05634	0.05634	0.000020	4.24067
5	0.02817	0.02817	0.000005	4.19727

Table 1.2: Convergence table for (1.5),(1.6) and (1.7). The time step length, τ, is set proportional to the square of the mesh diameter, that is $\tau = h$.

If we however construct a solution which is not linear in time

$$u = -t^2 x(1-x)y(1-y) - 1. \tag{1.8}$$

And do the same experiment as described above, with the time step length $\tau = h$, we do not observe quadratic convergence, see table 1.3.

number	mesh diameter, h	time step length, τ	discrete $L_2(\Omega)$ error	improvement
1	0.45069	0.45069	0.001922	-
2	0.22535	0.22535	0.000471	4.08322
3	0.11267	0.11267	0.000125	3.76197
4	0.05634	0.05634	0.000036	3.43254
5	0.02817	0.02817	0.000012	2.97651

Table 1.3: Convergence table for (1.5),(1.6) and (1.8). The time step length, τ, is set proportional to the square of the mesh diameter, that is $\tau = h$.

1.3.2 Non-Linear Hydraulic Conductivity

Here, we consider Richards' equation (**??**) in pressure variable, find $p = p(x,t)$ such that

$$\begin{cases} \partial_t \theta(p) - \nabla \cdot \kappa(\theta(p))\nabla u = f, & \text{in } \Omega \times (0,T] \\ p = p|_{\Gamma_D}, & \text{on } \partial\Gamma_D \times (0,T] \\ p = u_0, & \text{on } \Omega \times \{t=0\} \end{cases} \tag{1.9}$$

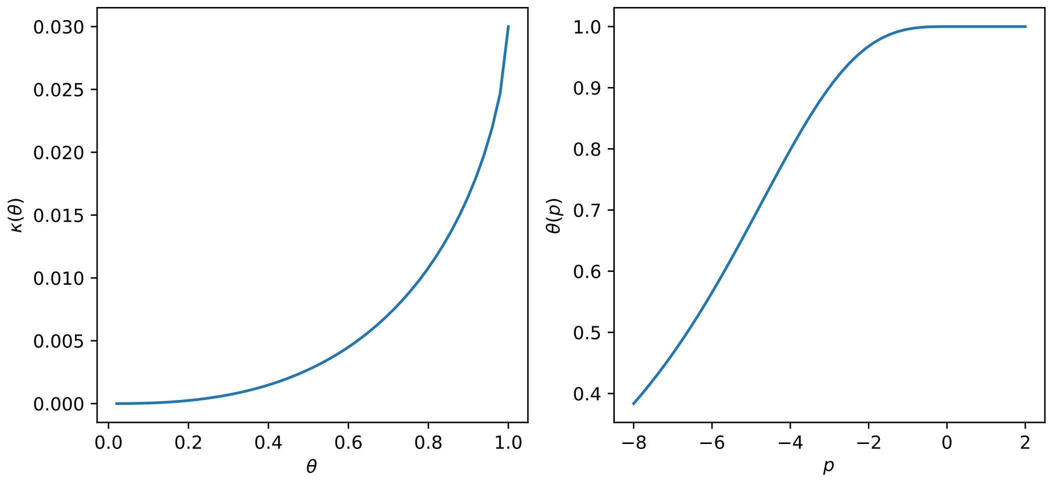

Figure 1.18: The Van Genuchten-Mualem non-linearities, (1.10) and (1.11).

With Ω being the perturbed unit square as before (see figure 1.16), and $T = 1$. We consider the Van Genuchten-Mualem parametrizations

$$\theta(p) := \begin{cases} (1 + (-\alpha_{vG} p)^{n_{vG}})^{-\frac{n_{vG}-1}{n_{vG}}}, & p \leq 0 \\ 1, & p > 0 \end{cases} \tag{1.10}$$

and

$$\kappa(\theta) := \frac{\kappa_{abs}}{\mu} \sqrt{\theta} \left(1 - \left(1 - \theta^{\frac{n_{vG}}{n_{vG}-1}}\right)^{\frac{n_{vG}-1}{n_{vG}}}\right)^2, \tag{1.11}$$

and compute the source term, f, such that the solution becomes (1.7).

number	mesh diameter, h	time step length, τ	discrete $L_2(\Omega)$ error	improvement
1	0.45069	0.20312	0.001863	-
2	0.22535	0.05078	0.000455	4.09403
3	0.11267	0.01270	0.000110	4.13154
4	0.05634	0.05634	0.000027	4.06126

Table 1.4: Convergence table for (1.5),(1.6) and (1.7). The time step length, τ, is set proportional to the square of the mesh diameter, that is $\tau = h^2$.

number	mesh diameter, h	time step length, τ	discrete $L_2(\Omega)$ error	improvement
1	0.45069	0.45069	0.001866	-
2	0.22535	0.22535	0.000459	4.06011
3	0.11267	0.11267	0.000113	4.05814
4	0.05634	0.05634	0.000028	4.00838

Table 1.5: Convergence table for (1.5),(1.6) and (1.7). The time step length, τ, is set proportional to the square of the mesh diameter, that is $\tau = h$.