Notes on Laplacian on domains with fractal boundary

Qile Yan

October 23, 2023

1 Problem setting

1.1 fractal boundary with a few steps of Koch snowflake

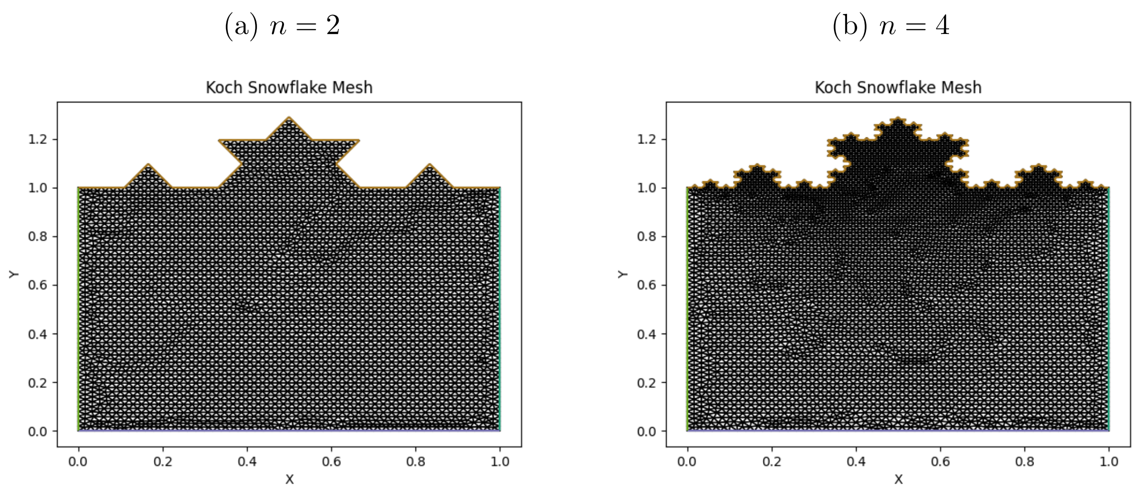


Figure 1: Unit square with the top edge replaced by a Koch snowflake with n iterations.

2D: Let n be the number of iterations in the snowflake (See Fig. 1, meshsize=0.02 for all vertices in .geo). The total number of small sides is 4^n and the small length scale $l = \left(\frac{1}{3}\right)^n$. Thus, the perimeter of the snowflake $L_p = \left(\frac{4}{3}\right)^n$.

Python script for creating the 2D mesh: snow_square.py.

3D: Let n be the number of iterations in the snowflake (See Fig. 2, meshsize=0.1 for all vertices in .geo). The total number of small squares is 13^n and the small area is $l = (\frac{1}{9})^n$. Thus, the perimeter of the snowflake $L_p = \left(\frac{13}{9}\right)^n$.

Python script for creating the 3D mesh: snow_cube.py.

(a) $n = 2$ (b) $n = 3$

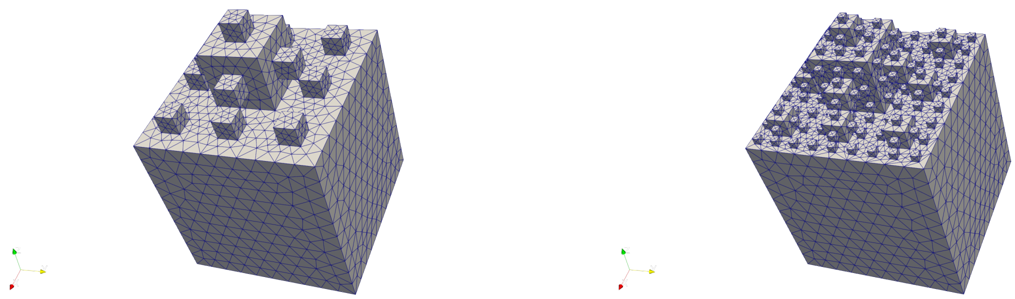


Figure 2: Unit cube with the top surface replaced by a Koch snowflake with n iterations.

1.2 PDEs

Solve

$$-\text{div}(D\text{grad}u) = 0 \quad \text{on } \Omega \tag{1.1}$$

Ω:

(a) the unit square in which the top edge has been replaced by a prefractal.

(b) the unit cube in which the top face is replaced by a prefractal.

Boundary conditions:

(a) on bottom edge, Dirichlet: $u = 1$.

(b) on sides, homogeneous Neumann: $\frac{\partial u}{\partial n} = 0$.

(c) on prefractal top edge, Robin boundary conditions: $\Lambda \frac{\partial u}{\partial n} + u = 0$.

The total flux through the top edge:

$$\Phi := \int_{top} -D \frac{\partial u}{\partial n} d\sigma = \frac{D}{\Lambda} \int_{top} u d\sigma.$$

We need to see how Φ depends on Λ for $0 \leq \Lambda \leq 2L_p$.

2 Test firedrake solver on 2D and 3D

Consider the Laplace equation $-\text{div}(D\text{grad}u) = f$ with non homogeneous condition:

(a) on bottom edge/surface, Dirichlet: $u = g$.

(b) on sides, homogeneous Neumann: $\frac{\partial u}{\partial n} = k$.

(c) on prefractal top edge/surface, Robin boundary conditions: $\Lambda \frac{\partial u}{\partial n} + u = l$.

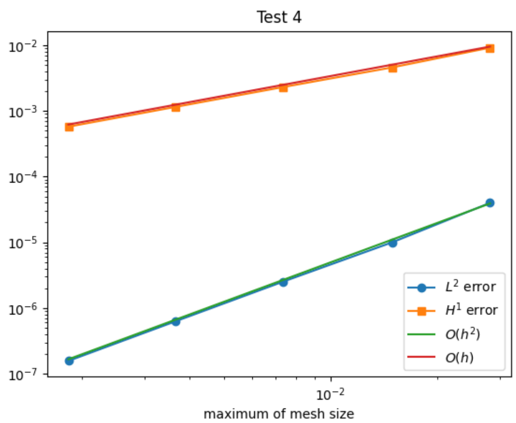
The weak formulation: Find $u \in H^1$ with $u = g$ on bottom such that

$$\int_\Omega D\mathrm{grad}(u)\cdot\mathrm{grad}(v)dx + \int_{top}\frac{D}{\Lambda}uvds = \int_\Omega fvdx + \int_{top}\frac{1}{\Lambda}lvds + \int_{sides}kvds, \quad \forall v \in H_0^1$$

In Fig. 3, we plot the error in L^2 and H^1 norm with respect to mesh size h. The manufactured solution is $u(x,y) = 2 + x^2 + y$ on the unit square with snowflake ($n = 4$ iterations). The Lagrange linear element is used. The uniform refinement is done by built-in function MeshHierarchy.

The test python script on 2D is: test-robin-solver.py.

(a) Error vs mesh size



(b) Error vs degree of freedom

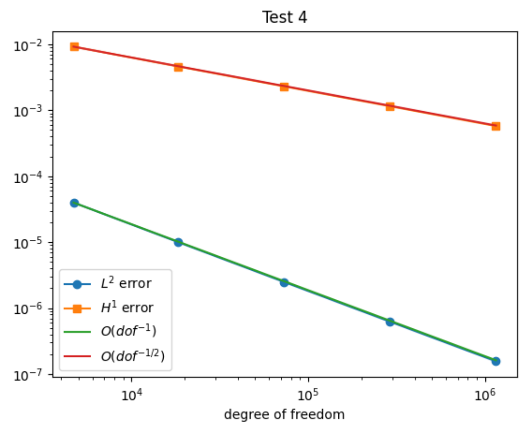


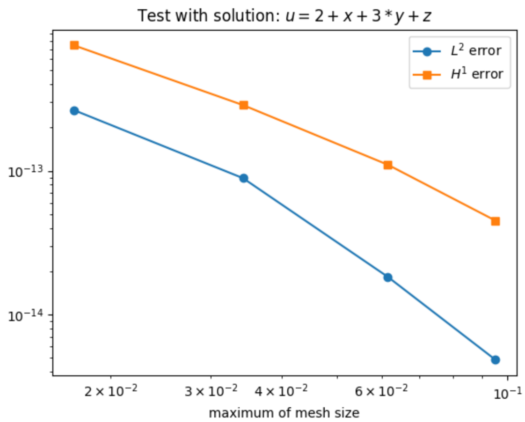
Figure 3: Unit square: Solution $u = x^2 + y + 2$.

In Fig. 4 and Fig. 5, we plot the error in L^2 and H^1 norm with respect to mesh size h and degree of freedom where domain is unit cube with snowflake ($n = 3$ iterations). In Fig. 4, the manufactured solution is $u(x,y,z) = 2 + x + 3y + z$. In Fig. 5, the manufactured solution is $u(x,y,z) = 2 + x^2 + 3xy + yz$. The Lagrange linear element is used. The uniform refinement is done by built-in function MeshHierarchy.

The test python script on 3D is: test-robin-solver-cube.py. The script is run by the following parallelsim command

mpiexec -n 16 python test-robin-solver-cube.py

(a) Error vs mesh size



(b) Error vs degree of freedom

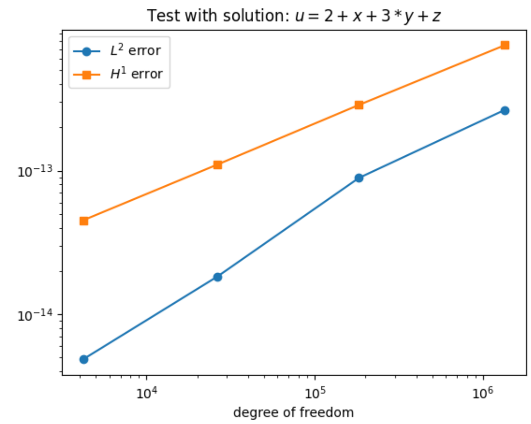
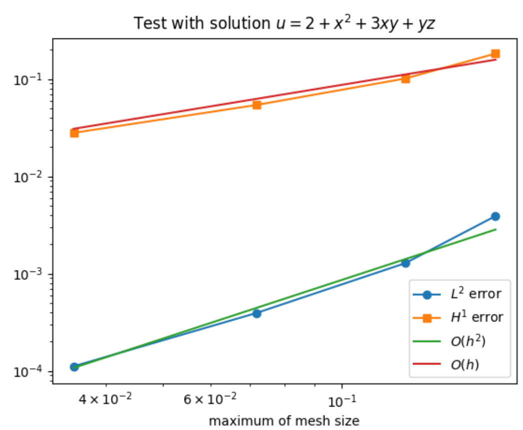


Figure 4: Unit cube: Solution $u = 2 + x + 3y + z$. Linear finite element.

(a) Error vs mesh size



(b) Error vs degree of freedom

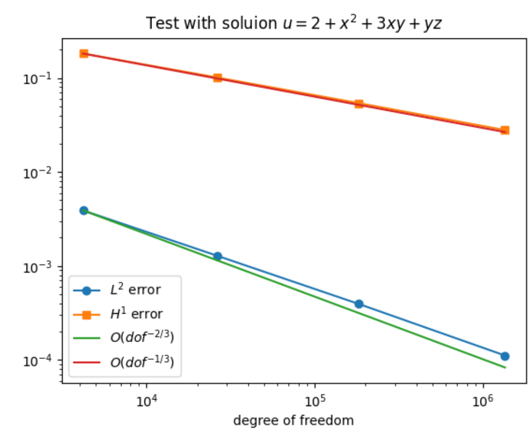


Figure 5: Unit cube. Solution $u = 2 + x^2 + 3xy + yz$. Linear finite element.

3 Ω is unit square and D is constant

Now, we solve the PDE (1.1) on 2D unit square using firedrake with the finest mesh in Fig. **??**. In Fig. 6, we plot the flux $\Phi = \int_{top} -D\frac{\partial u}{\partial n} d\sigma$ with different choice of Λ.

(a) Solution when $D = 1, \Lambda = 1$ (b) Total flux vs Λ

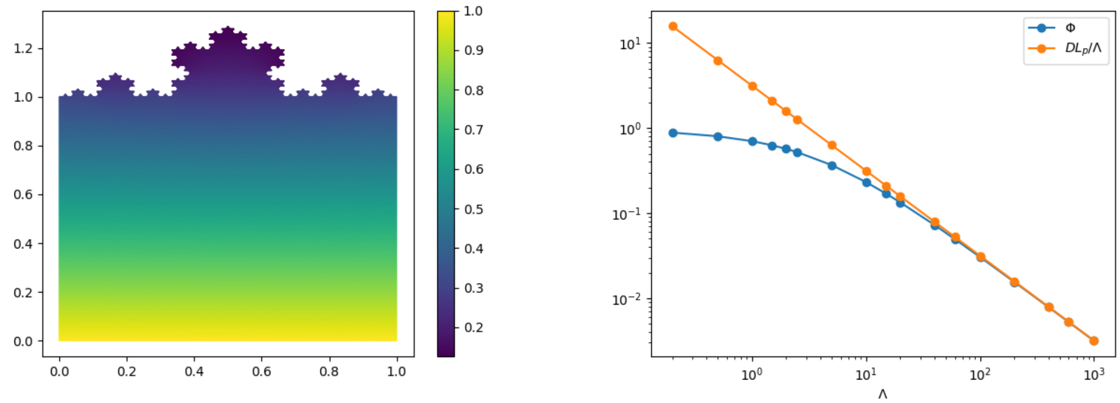


Figure 6: Snowflake with $n = 4$ iterations. (a). Solutions. (b) Total flux vs Λ when $D = 1$. Python script: main_flux.py.

4 Ω is unit cube and D is constant

Now, we solve the PDE (1.1) on 3D unit cube using firedrake with the finest mesh in Fig. ??. In Fig. 7, we plot the flux $\Phi = \int_{top} -D\frac{\partial u}{\partial n} d\sigma$ with different choice of Λ.

(a) Total flux vs Λ

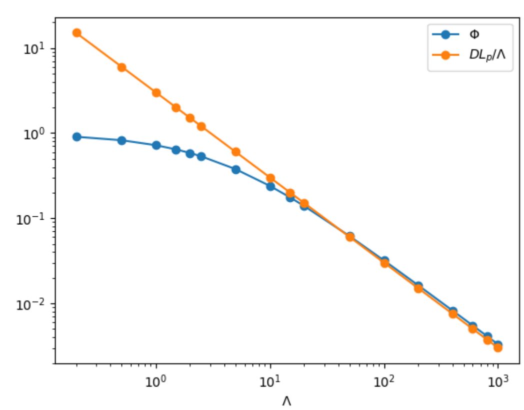


Figure 7: Snowflake with $n = 3$ iterations on Cube. Total flux vs Λ when $D = 1$. Python script: main_flux.py.

5 Fractal boundary with Dirichlet Boundary conditon

In this section, we solve the PDE

$$-\Delta u = f \quad \text{on } \Omega \tag{5.1}$$

with $u = 0$ on $\partial\Omega$. Here, Ω is a unit square where each edges are replaced by the Koch snowflake (See Fig. 8).

(a) $n = 2$ (b) $n = 4$

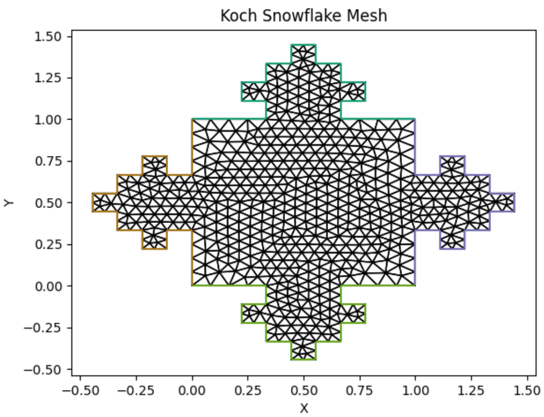


Figure 8: Unit square with each edges replaced by a Koch snowflake with n iterations. Meshsize is 0.1 for each vertices in .geo file.

In Fig. 9 and Fig. 10, we plot the error in L^2 and H^1 norm with respect to mesh size h and degree of freedom where domain is unit cube with snowflake ($n = 3$ iterations). In Fig. 9, the manufactured solution is $u(x, y) = 2 + x + 3y$. In Fig. 10, the manufactured solution is $u(x, y, z) = 2 + x^2 + y$. The Lagrange linear element is used. The uniform refinement is done by built-in function MeshHierarchy.

(a) Error vs mesh size (b) Error vs degree of freedom

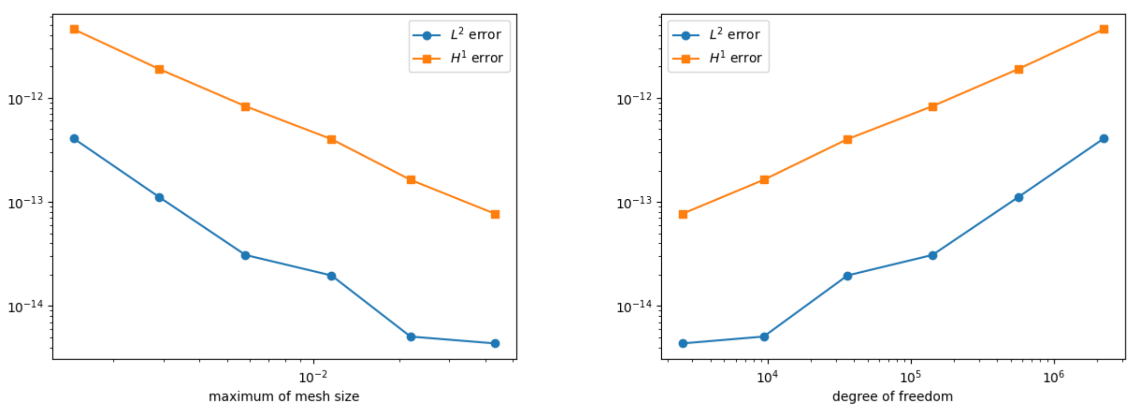


Figure 9: Unit square: Solution $u = x + 3y + 2$. linear finite element space

(a) Error vs mesh size (b) Error vs degree of freedom

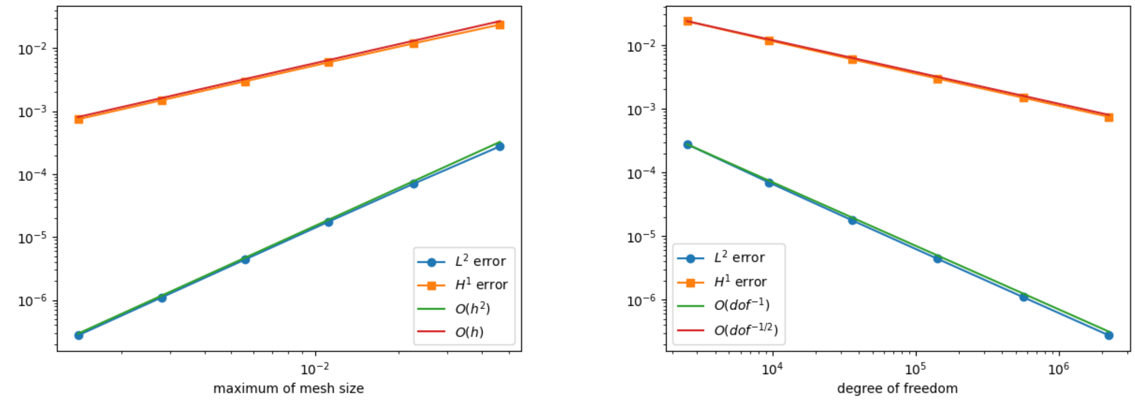


Figure 10: Unit square: Solution $u = x^2 + y + 2$. linear finite element space

In Fig. 11(b), we plot the solution of the PDE (5.1) with f defined as in Fig. 11(a).

(a) f (b) Solution u

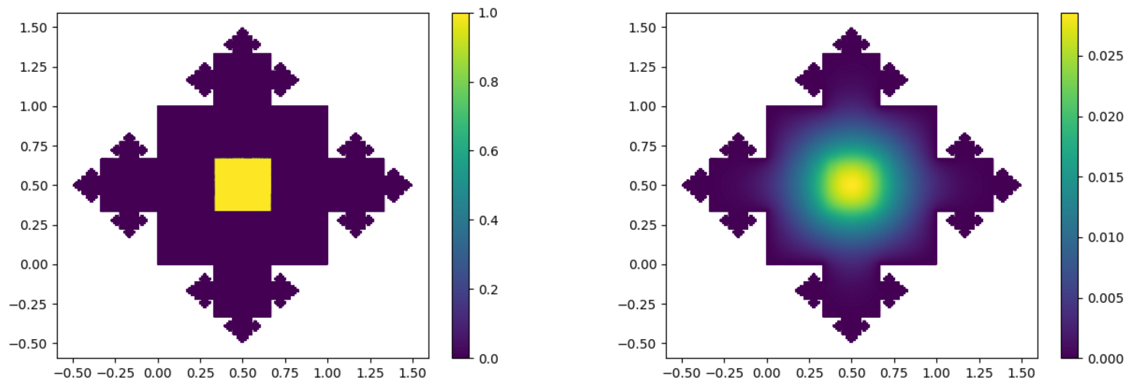


Figure 11: On unit square with snowflake. (a). force term f. (b) solution of the PDE (5.1).

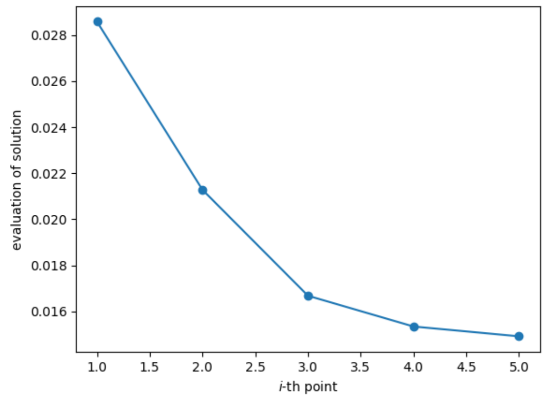


Figure 12: On unit square with snowflake. Evaluation of solution at a sequence points $\{\mathbf{x}_i\}_{i=0}^4$ where $\mathbf{x}_i$ is the center of a i-th snowflake square.