Notes on Laplacian on domains with fractal boundary

Qile Yan

November 12, 2023

1 Problem setting

1.1 fractal boundary with a few steps of Koch snowflake

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

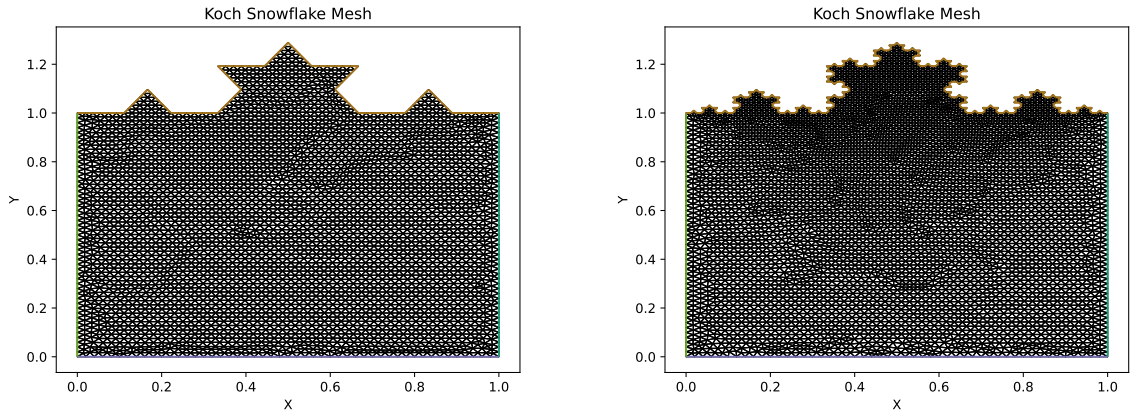


Figure 1: Unit square with the top edge replaced by a Koch snowflake with n iterations.

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) $n = 3$

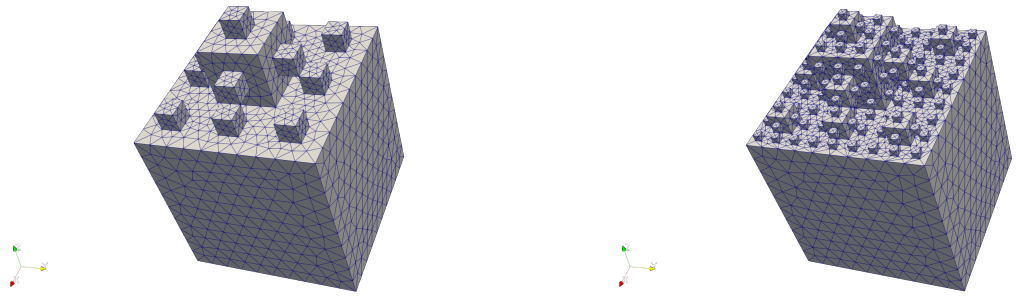


Figure 2: Unit cube with the top surface replaced by a Koch snowflake with n iterations.

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(a) the unit square in which the top edge has been replaced by a prefractal.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

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

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

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

2.1 2D case

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

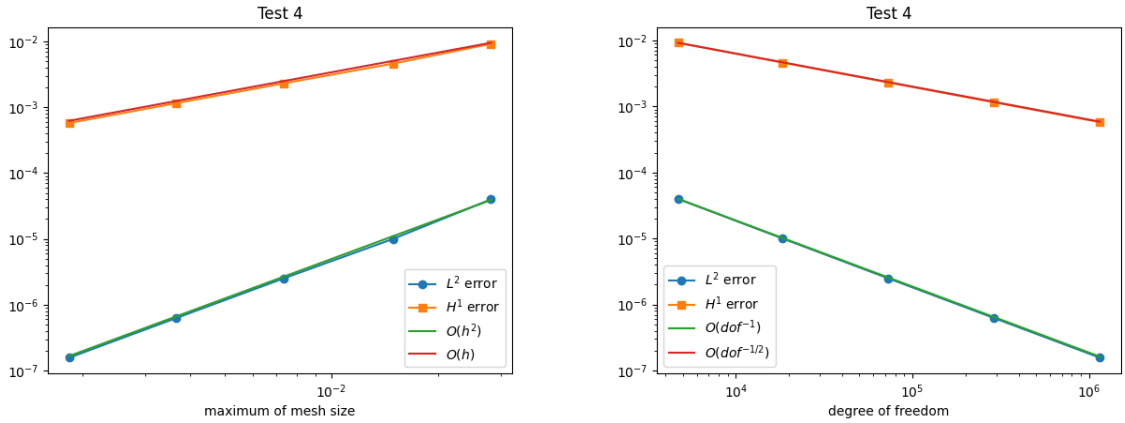


Figure 3: Unit square: Solution $u = x^2 + y + 2$.

2.2 3D case

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. Here, we use the standard Lagrange linear element is used. To check the convergence rate, the uniform refinement is done by built-in function MeshHierarchy on the mesh in Fig. 2 (b). We consider two manufactured solutions (a). $u(x, y, z) = 2 + x + 3y + z$. (b). $u(x, y, z) = 2 + x^2 + 3xy + yz$. The first one is a linear function in the element space while the second one is not.

In Fig. 4, the discrete linear system is solved by computing LU factorisation. In the code, we set " solver_parameters='ksp_type': 'preonly', 'pc_type': 'lu') "

In Fig. 5, the linear system is solved using Krylov subspace methods (conjugate gradient method since the operator is SPD). Besides, the preconditioner is chosen to be BoomerAMG. In the code, we set "solver_parameters='ksp_type': 'cg', 'pc_type': 'hypre','pc_hypre_type': 'boomeramg' ".

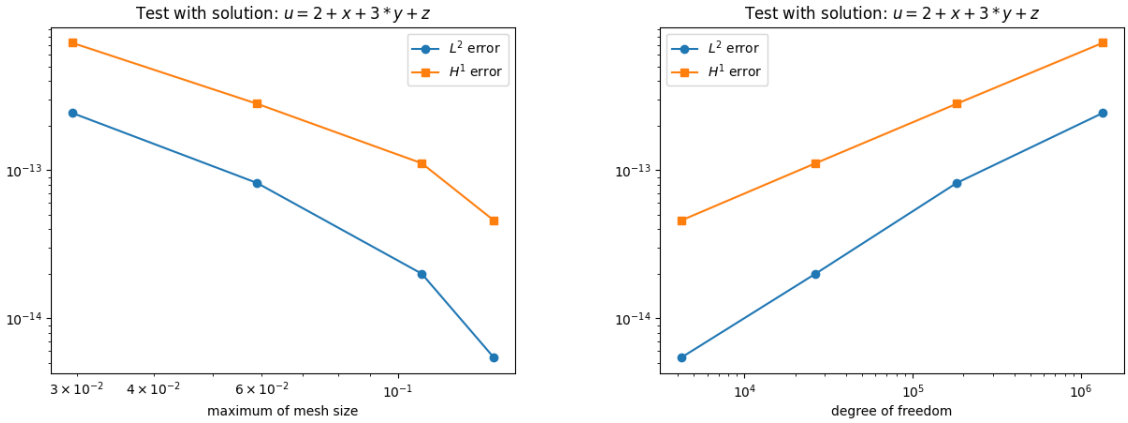
From the first row of Fig. 4 and Fig. 5, we can see that for linear solution u the only round off error appears in the case when LU factorisation is used. But the error from using Krylov subspace is around $5 * 10^{-7}$ to $5 * 10^{-6}$. On the other hand, the Krylov subspace method solve the linear system faster and also could deal with a finer mesh when degree of freedom is 10^7.

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



(c) Error vs mesh size

(d) Error vs degree of freedom

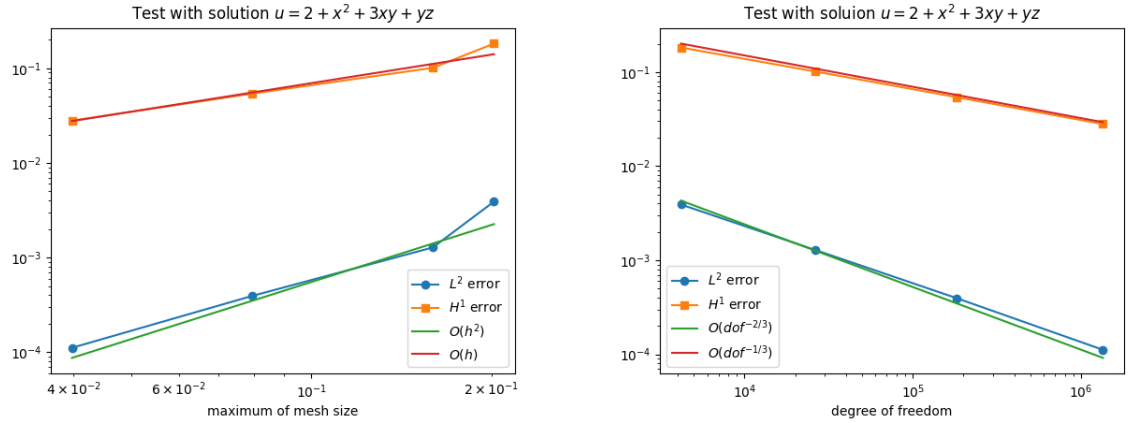
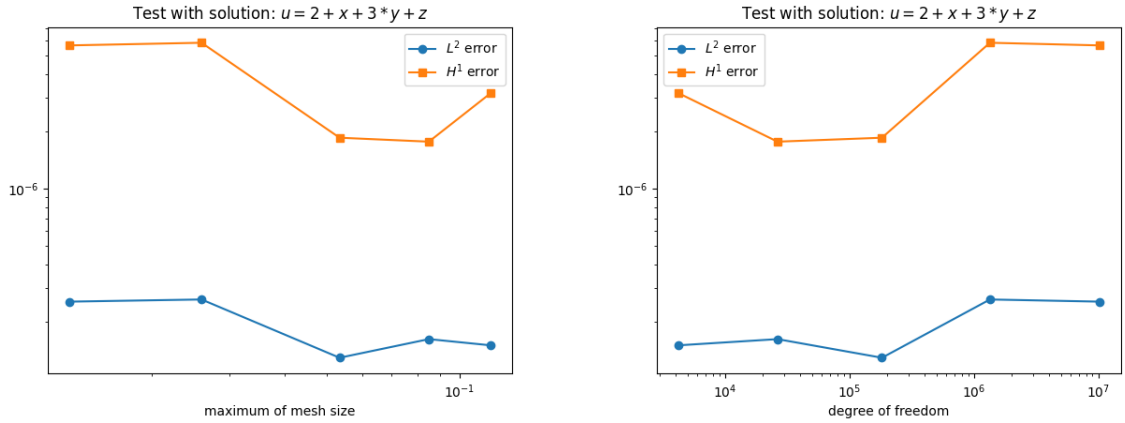


Figure 4: Unit cube: The linear system is solved by LU factorisation.

(a) Error vs mesh size

(b) Error vs degree of freedom



(c) Error vs mesh size

(d) Error vs degree of freedom

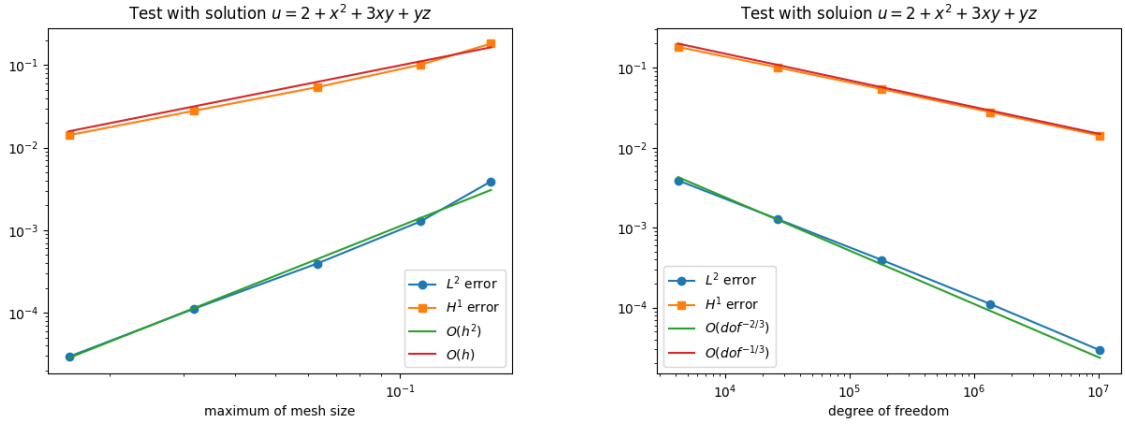


Figure 5: Unit cube. The linear system is solved by Krylov subspace methods

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

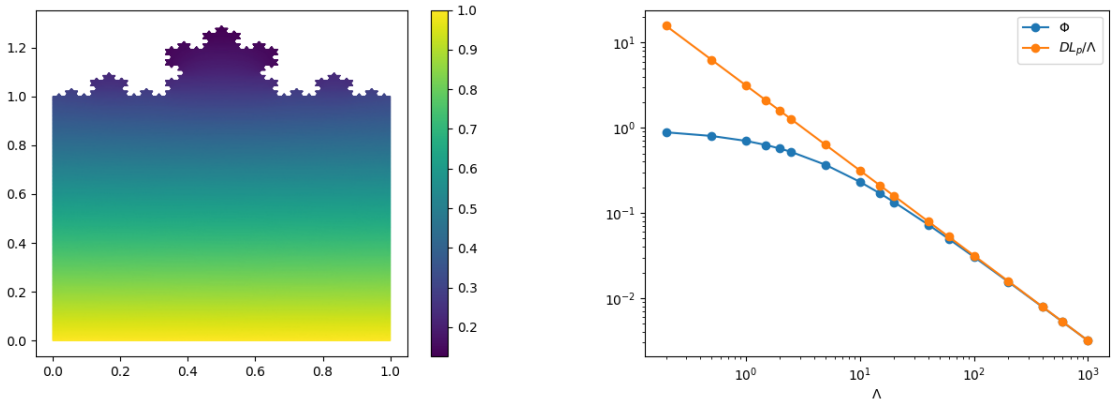


Figure 6: Snowflake with $n = 4$ iterations. (a). Solutions. (b) Total flux vs Λ when $D = 1$. Python script: main_flux.py.

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

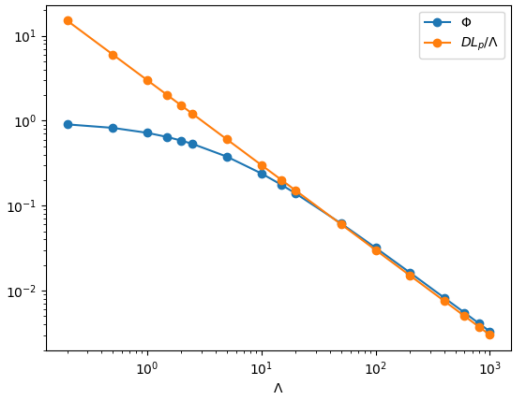


Figure 7: Snowflake with $n = 3$ iterations on Cube. Total flux vs Λ when $D = 1$. Python script: main_flux.py.

5 Fractal harmonic measure

5.1 Result on unit disk

In this section, we consider the domain Ω is a unit disk (see Fig. 8).

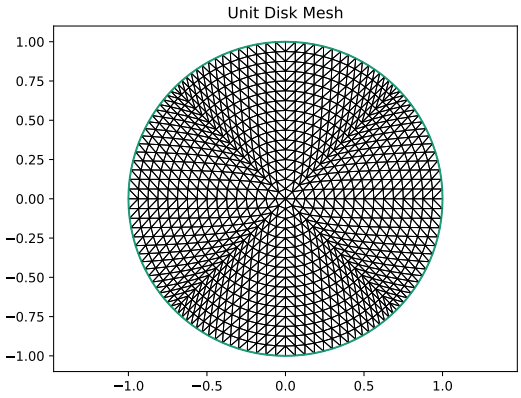


Figure 8: Unit disk.

We first test the firedrake solver on the PDE:

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

with $u = g$ on $\partial\Omega$.

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. 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

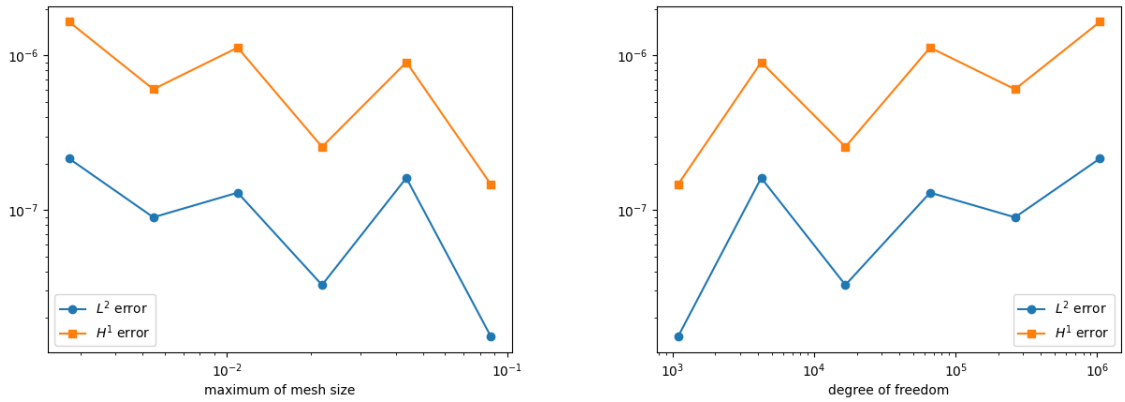


Figure 9: Unit disk: manufactured solution $u = x+3y+2$. Note that only round off error appears here since the solution is in the linear finite element space. Remark: Here, we use the Krylov subspace method with BoomerAMG in preconditioner in the solver. If we change the solver to be LU factorisation, the error here would be around $10^{-14} \sim 10^{-15}$.

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

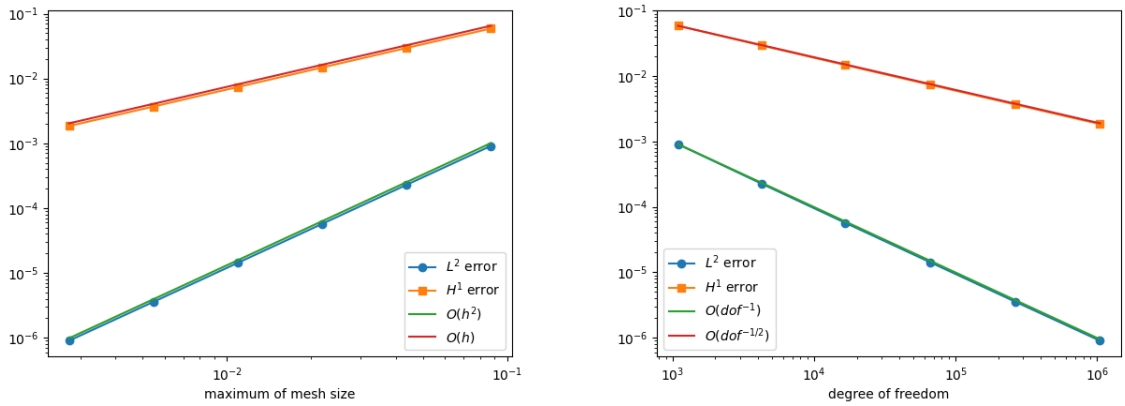


Figure 10: Unit disk: manufactured solution $u = x^2 + y + 2$.

Next, we solve the PDE

$$-\Delta u = f \quad \text{on } \Omega \tag{5.2}$$

with $u = 0$ on the unit disk and evaluate the solution at points from center to boundary. In particular, we evaluate the solutions at

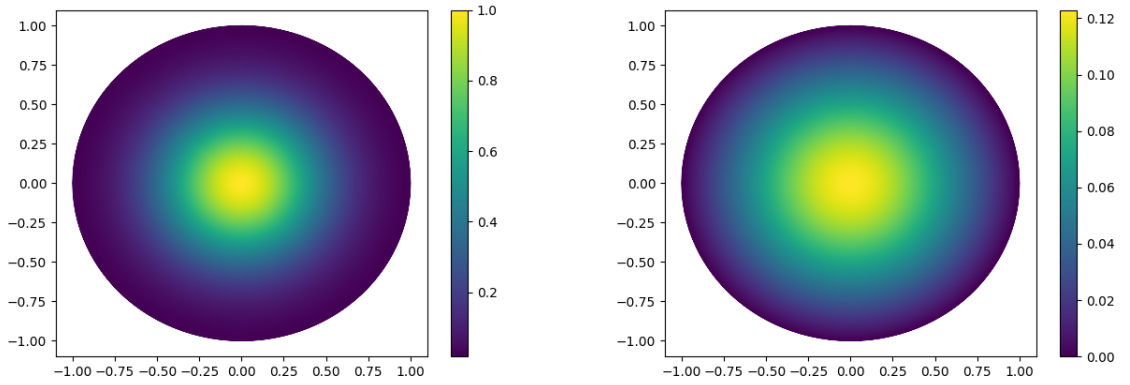
$$\mathbf{x}_i = (0, 1 - 1/2^i), \quad 0 \le i \le n_{max}. \tag{5.3}$$

In Fig. 11, we show the results of the PDE (5.2) with a different f. That is, we define f as

$$f(x, y) = e^{-4((x-0.5)^2+(y-0.5)^2)}.$$

(a) f

(b) Solution u



(c) Evaluation at $\{\mathbf{x}_i\}_{i=0}^4$.

(d) Evaluation at $\{\mathbf{x}_i\}_{i=0}^4$. loglog plot

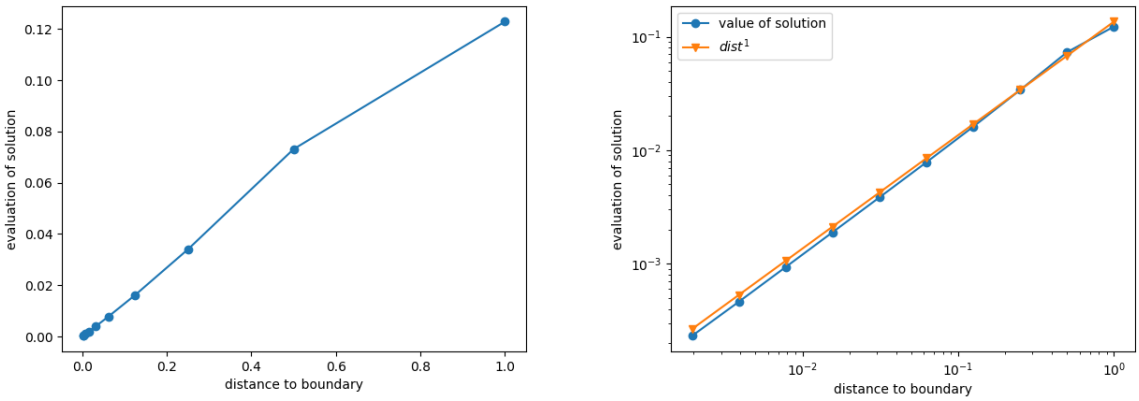


Figure 11: On unit disk. (a). force term f. (b) solution u. (c). Evaluation of solution at a sequence points $\{\mathbf{x}_i\}_{i=0}^{10}$ where $\mathbf{x}_i$ is defined as in (5.3). (d). loglog plot

5.2 On snowflake boundary domain

In this note we solve the PDE

$$-\Delta u = f \quad \text{on } \Omega \tag{5.4}$$

with $u = 0$ on $\partial\Omega$. Here, Ω is a unit square where each edges are replaced by the Koch snowflake (See Fig. 12). In this note, we use the FEM python package firedrake solving the PDE.

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

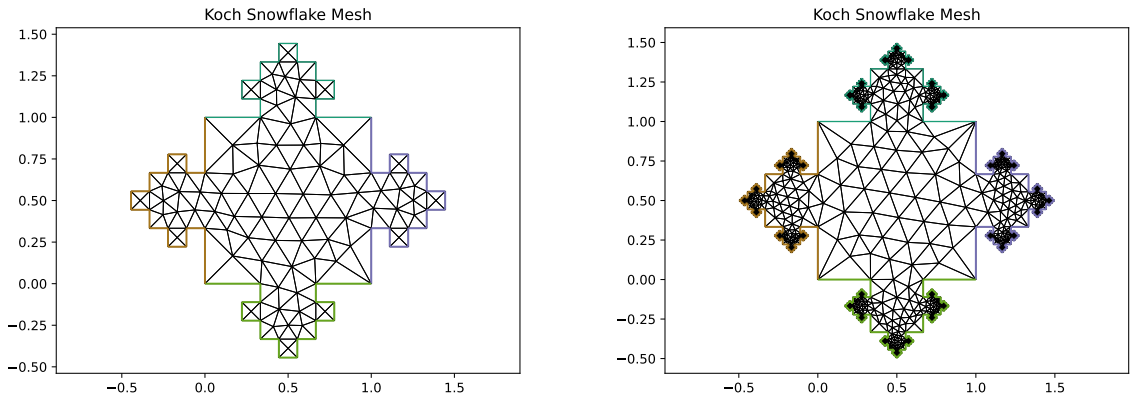


Figure 12: Unit square with each edges replaced by a Koch snowflake with n iterations. Meshsize is 0.5 for each vertices in .geo file.

In Fig. 13 and Fig. 14, 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 square with snowflake ($n = 4$ iterations). In Fig. 13, the manufactured solution is $u(x, y) = 2 + x + 3y$. In Fig. 14, 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

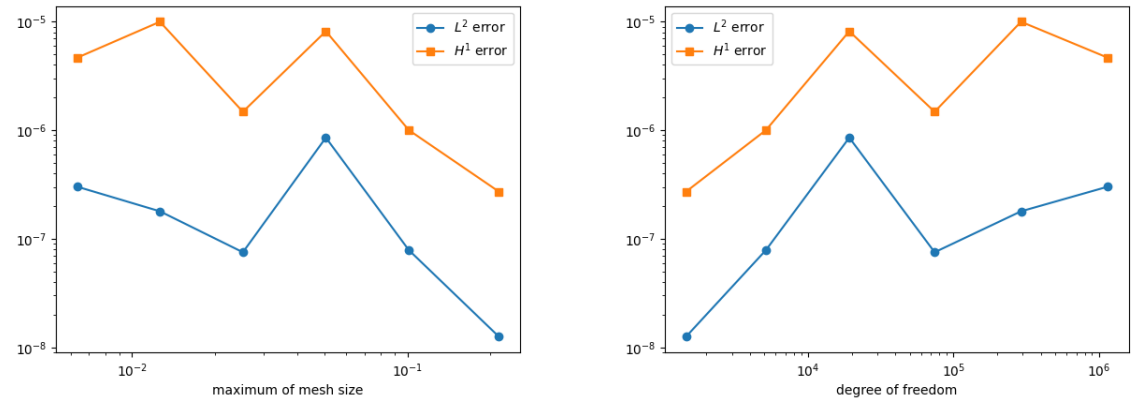


Figure 13: Unit square: manufactured solution $u = x + 3y + 2$. Note that only round off error appears here since the solution is in the linear finite element space.

(a) Error vs mesh size

(b) Error vs degree of freedom

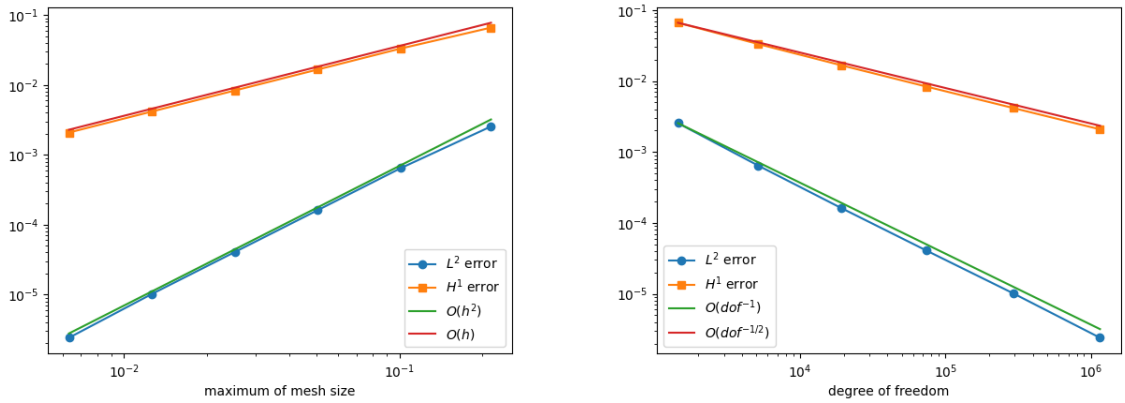


Figure 14: Unit square: manufactured solution $u = x^2 + y + 2$.

In Fig. 15(b), we plot the solution of the PDE (5.4) where Ω is the square with 4 snowflake iterations as in Fig. 12(b) and f is defined as

$$f(x, y) = e^{-4((x-0.5)^2+(y-0.5)^2)}.$$

(a) f

(b) Solution u

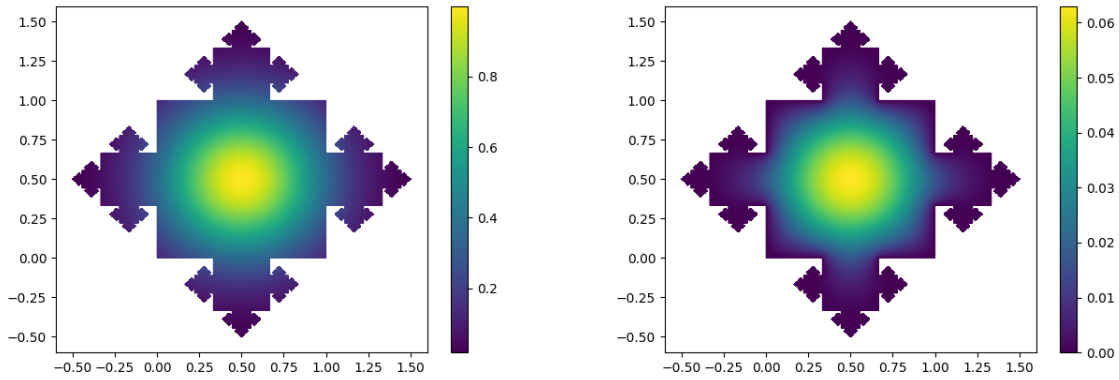


Figure 15: On unit square with 8 snowflake iterations. (a). force term f. (b) solution of the PDE (5.4).

In Fig. 17, we evaluate the solution along a random path with each points $\mathbf{x}_i, 0 \leq i \leq n$ is a center of the square from i-th snowflake iteration. In particular, we create the sequence of the points along the path as followings:

1). Set $\mathbf{x}_0 = (0.5, 0.5)$

2). Pick a random integer from 0 to 3. Each integer corresponds to one of four normal directions $\vec{v}_1$. (upward, downward, left, right).

3). Set $\mathbf{x}_1 = \mathbf{x}_0 + h_0\vec{v}_1$ with $h_0 = \frac{2}{3}$.

4). For $i = 2, ..., n$, Pick a random integer from 0 to 2. Each integer corresponds to one of three normal directions $\vec{v}_i$ (forward, left, right). Set

$$\mathbf{x}_i = \mathbf{x}_{i-1} + h_{i-1}\vec{v}_i.$$

where $h_{i-1} = \frac{2}{3^i}$.

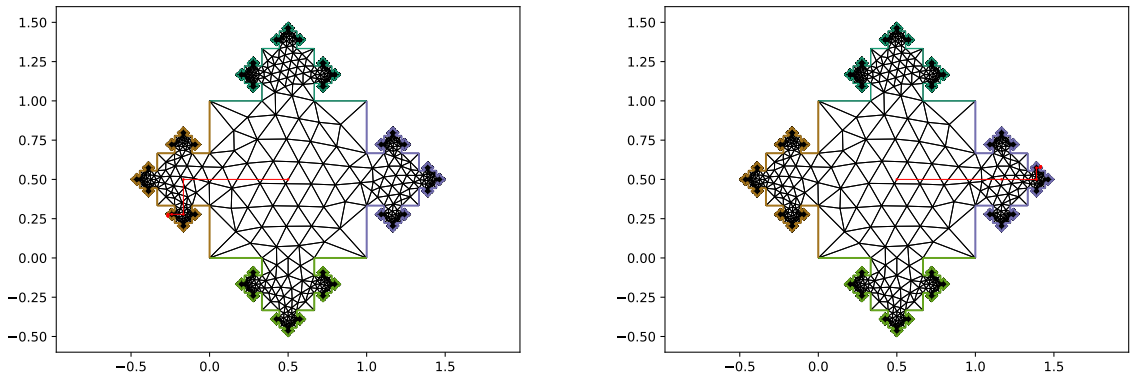


Figure 16: Two random path on the unit square with 8 snowflake iterations.

Let $dx_i = (1/3)^i$. Then we would like to approximate $u(\mathbf{x}_i)$ as a function of dx_i in the form

$$u(\mathbf{x}_i) = c(dx_i)^\alpha$$

By taking the log on both ends,

$$\log u(\mathbf{x}_i) = \log c + \alpha \log(dx_i), \quad 0 \le i \le n.$$

The coefficients c and α are found by least squares approximation.

After running 500 random path, we get:

	Mean	Standard deviation
c	0.047939	0.0053037
α	2.1734	0.095348

(a) $c = 0.051$, $\alpha = 2.03708$ (b) $c = 0.0519$, $\alpha = 2.15278$

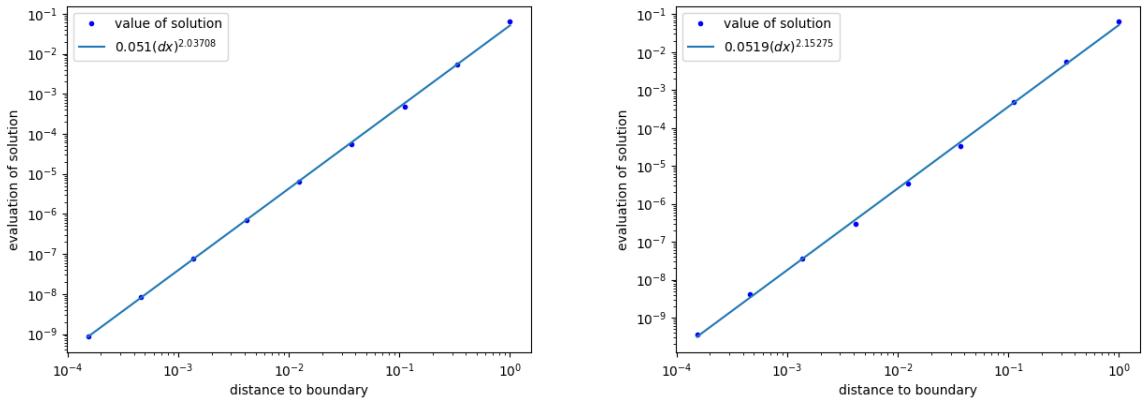


Figure 17: On unit square with 8 snowflake iterations. Result of different random path