Notes on Laplacian on domains with fractal boundary

Qile Yan

January 17, 2024

1 Problem setting

1.1 fractal boundary with a few steps of Koch snowflake

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

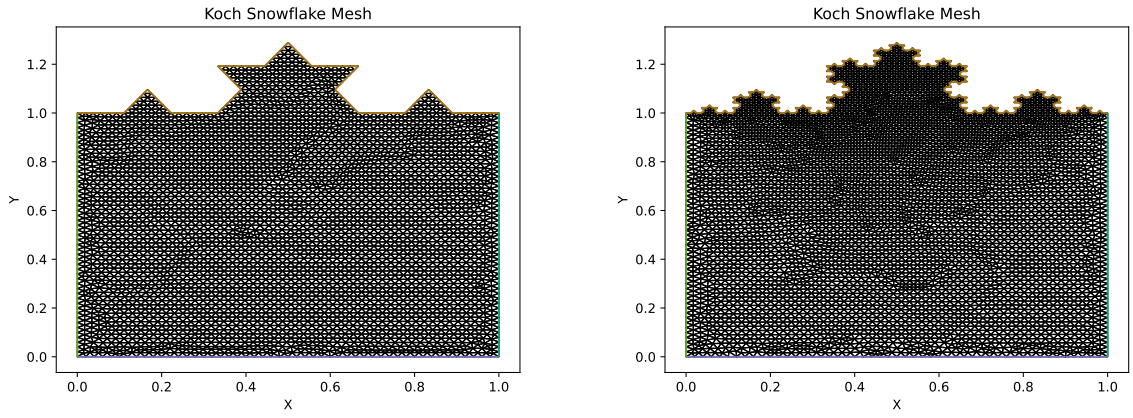


Figure 1: Unit square with the top edge replaced by a Koch snowflake with n iterations.

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) $n = 3$

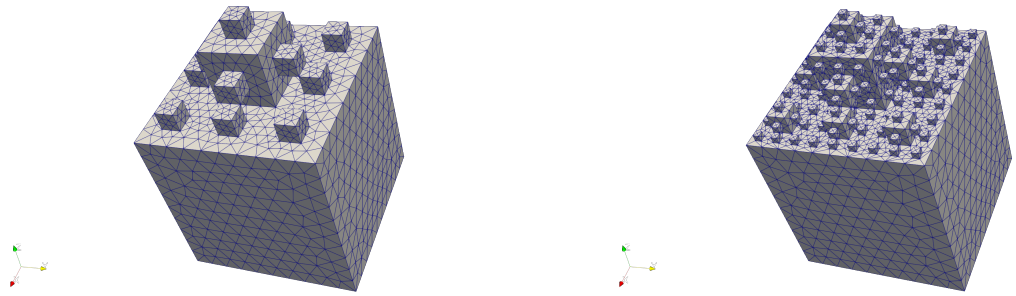


Figure 2: Unit cube with the top surface replaced by a Koch snowflake with n iterations.

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(a) the unit square in which the top edge has been replaced by a prefractal.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

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

Let

$$\Phi_\Lambda = \frac{1}{Z_0 + Z(\Lambda)}.$$

Then $Z(0) = 0$, $Z(\Lambda) = 1/\Phi_\Lambda - 1/Z_0$.

We need to see how $Z(\Lambda)$ depends on Λ:

(1) $\Lambda \ll l$, $Z(\Lambda) \approx \Lambda$.

(2) $l < \Lambda < L_p$, $Z(\Lambda) \approx \Lambda^{1/dim_{frac}}$.

(3) $\Lambda \gg L_p$, $Z(\Lambda) \approx \Lambda$.

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

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

Installation of firedrake:

1. To have mpi setting:

   python3 firedrake-install –mpiexec=mpiexec –mpicc=mpicc –mpicxx=mpicxx –mpif90=mpif90

2. To have netgen suit such that we can refine mesh adaptively:

   firedrake-update –netgen

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

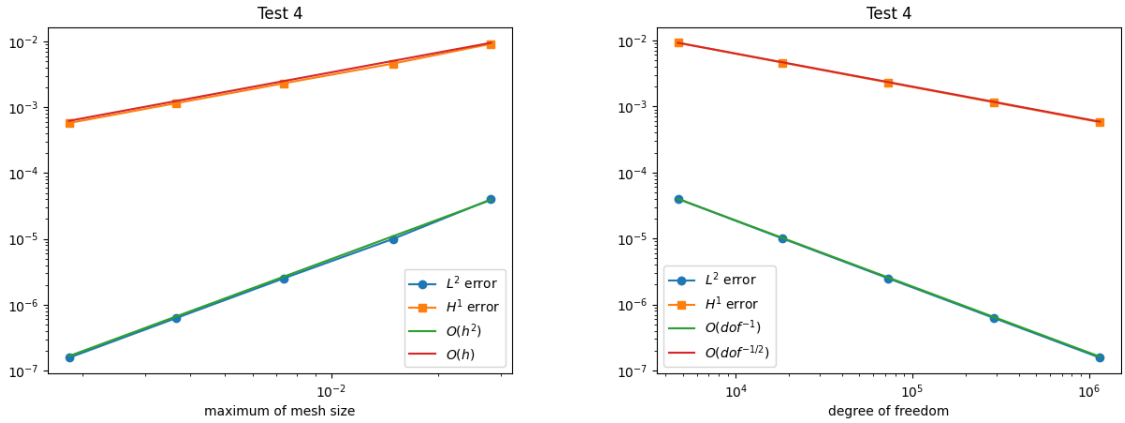


Figure 3: Unit square with snowflake ($n = 4$ iterations): Solution $u = x^2 + y + 2$.

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' ".

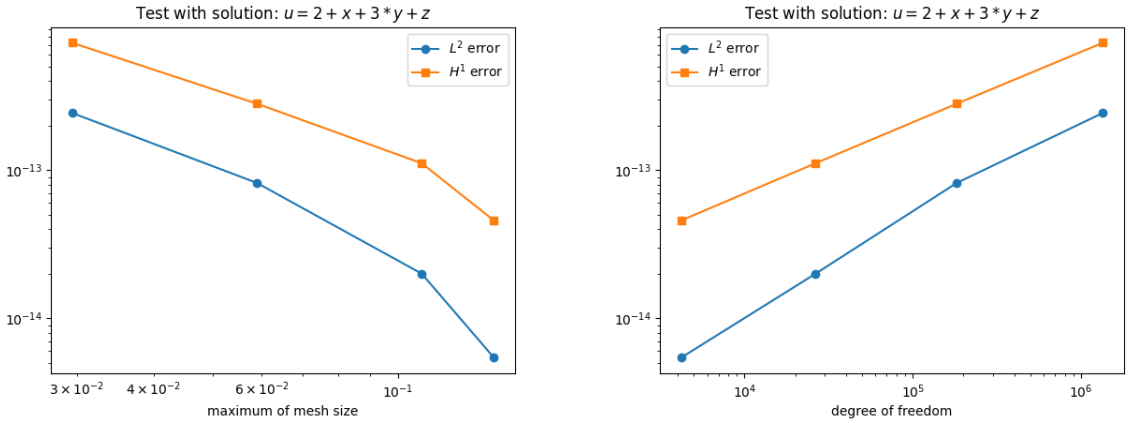
From the first row of Fig. 4 and Fig. 5, we can see that for linear solution u the only round off error appears in the case when LU factorisation is used. But the error from using Krylov subspace is around $5 * 10^{-7}$ to $5 * 10^{-6}$. On the other hand, the Krylov subspace method solve the linear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

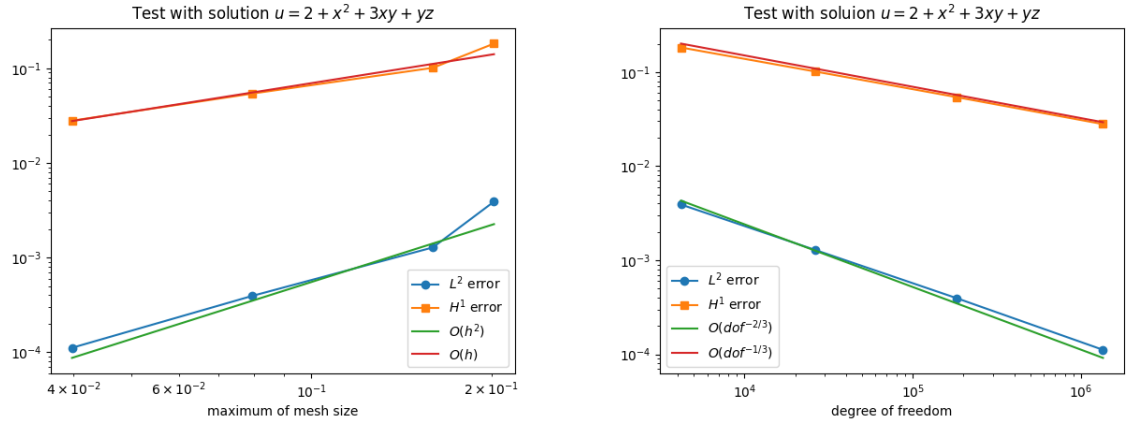
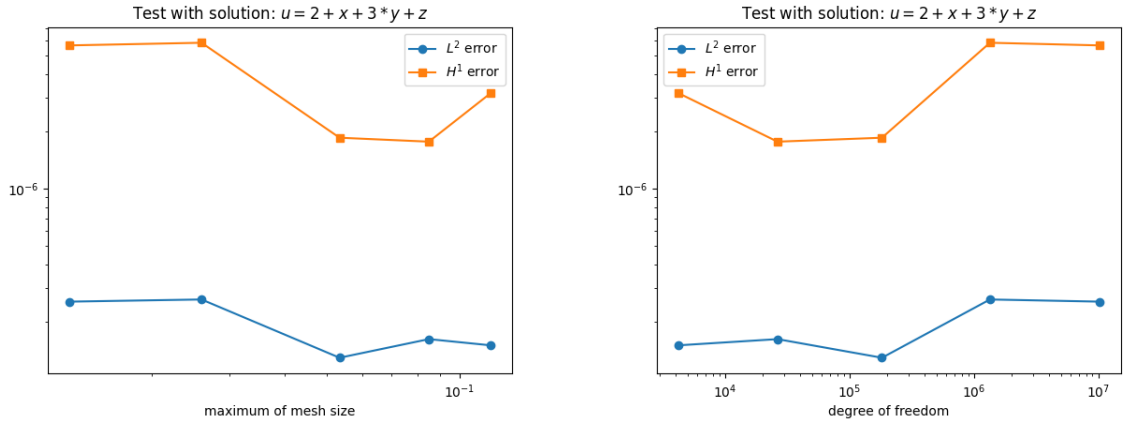


Figure 4: Unit cube: The linear system is solved by LU factorisation.

(a) Error vs mesh size

(b) Error vs degree of freedom



(c) Error vs mesh size

(d) Error vs degree of freedom

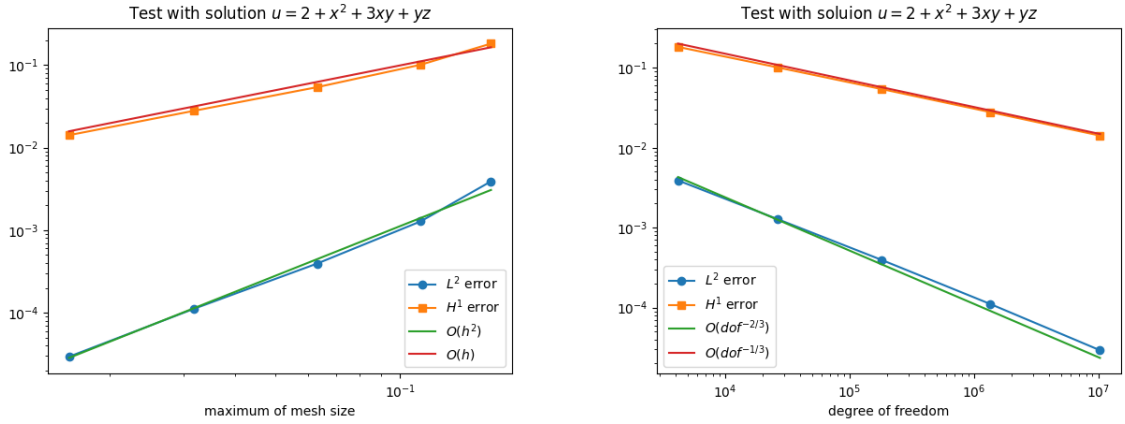
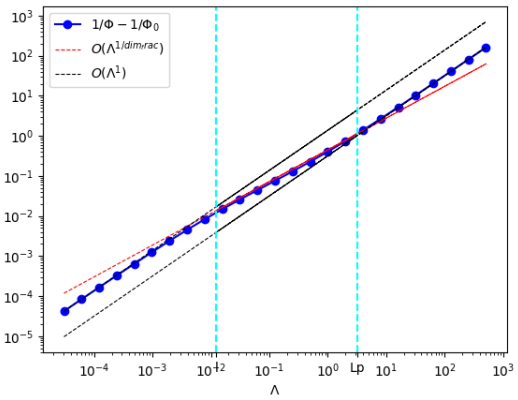


Figure 5: Unit cube. The linear system is solved by Krylov subspace methods

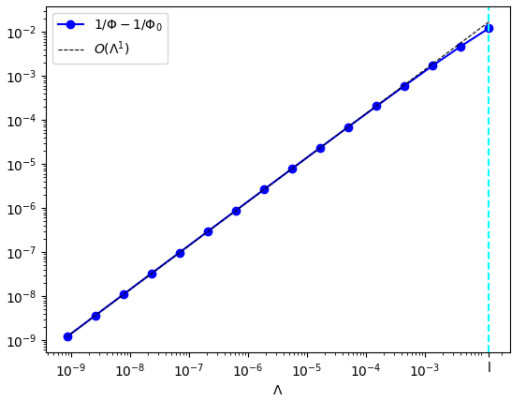
3 Ω is unit square and D is constant

Now, we solve the PDE (1.1) on 2D unit square adaptively using firedrake. In Fig. 6, we plot the function $Z_\Lambda = 1/\Phi_\Lambda - 1/\Phi_0$ as a function of Λ where flux $\Phi = \int_{top} -D\frac{\partial u}{\partial n} d\sigma$

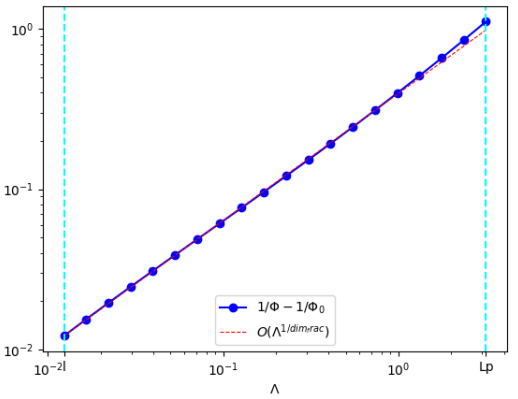
(a) $0 < \Lambda < \infty$



(b) $0 < \Lambda < l$



(c) $l < \Lambda < L_p$



(d) $L_p < \Lambda < \infty$

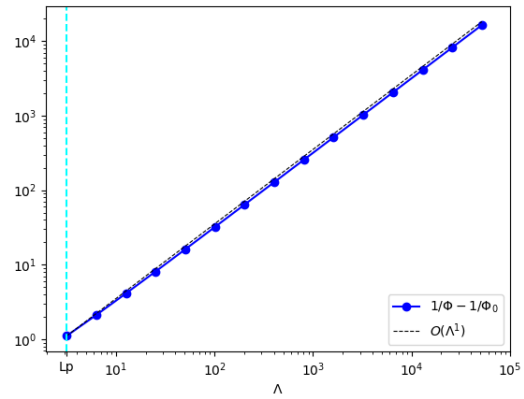
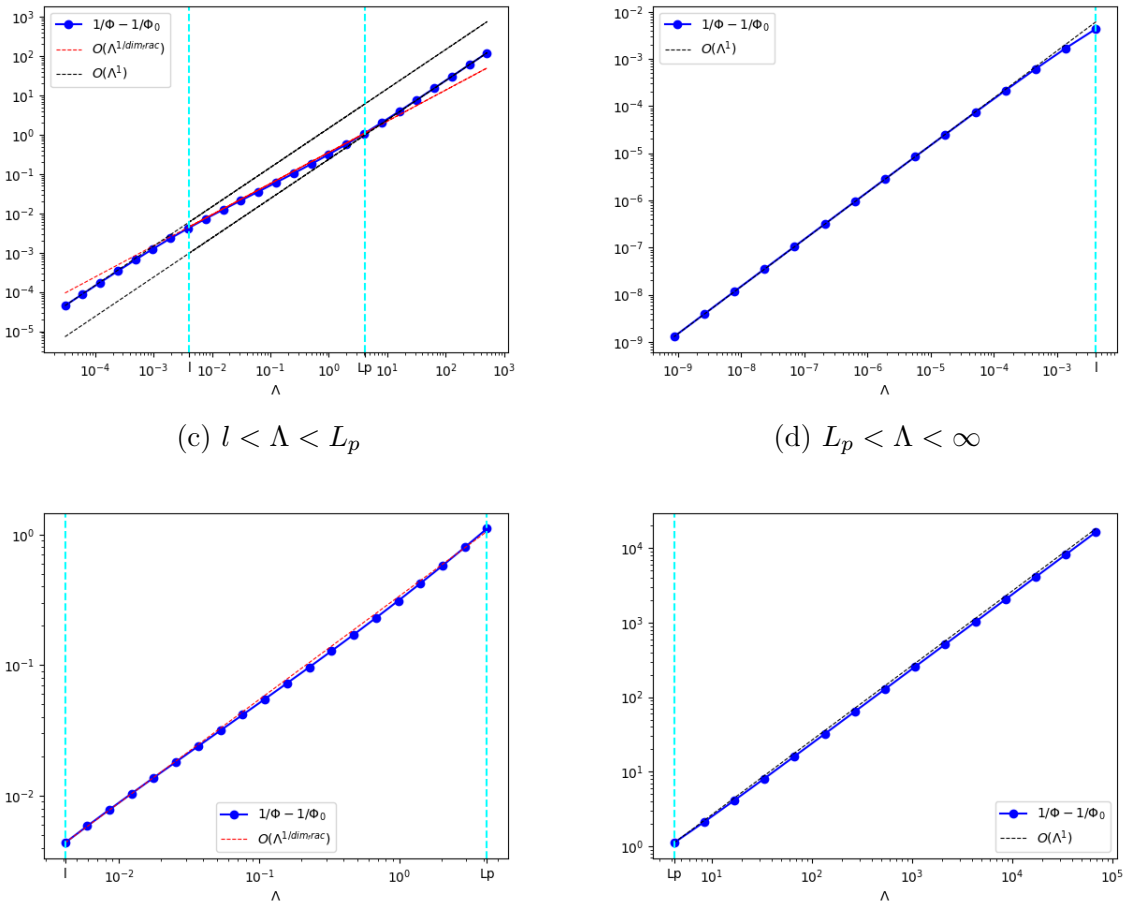


Figure 6: On 2d unit square with snowflake $n = 4$ iterations. $Z_\Lambda = 1/\Phi_\Lambda - 1/\Phi_0$ vs Λ when $D = 1$. $L_p = \left(\frac{4}{3}\right)^n$. $l = \left(\frac{1}{3}\right)^n$. $dim_{frac} = \frac{\log 4}{\log 3}$. Python script: main_flux.py.

(a) $0 < \Lambda < \infty$ (b) $0 < \Lambda < l$



(c) $l < \Lambda < L_p$ (d) $L_p < \Lambda < \infty$

Figure 7: On 2d unit square with snowflake $n = 5$ iterations. $Z_\Lambda = 1/\Phi - 1/\Phi_0$ vs Λ when $D = 1$. $L_p = \left(\frac{4}{3}\right)^n$. $l = \left(\frac{1}{3}\right)^n$. $dim_{frac} = \frac{\log 4}{\log 3}$. Python script: main_flux.py.

(a) $n = 1$ (b) $n = 2$ (c) $n = 3$

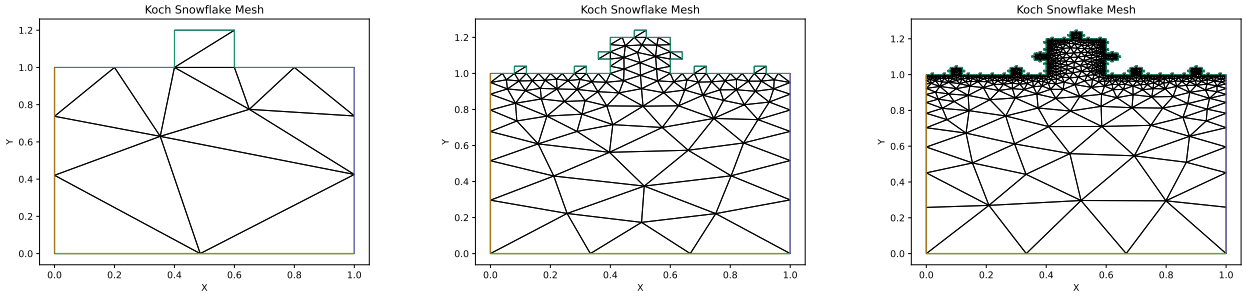
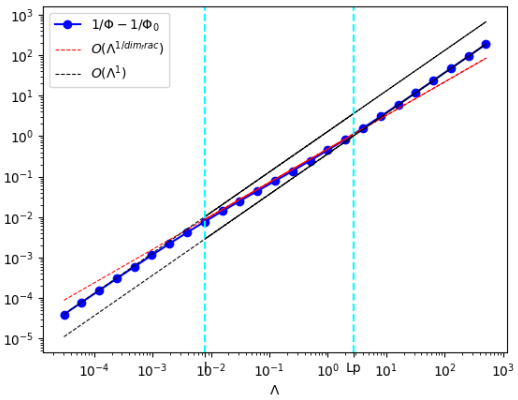
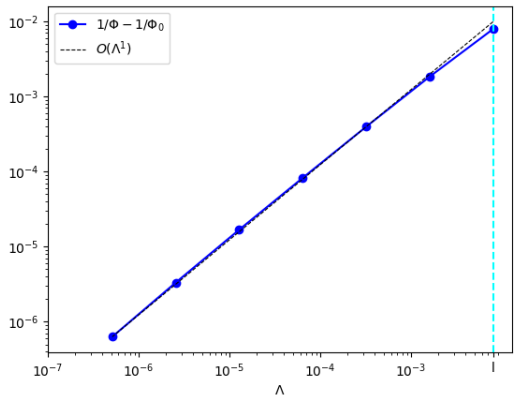


Figure 8: On 2d unit square where the top edge is replaced by a square koch snowflake. $L_p = \left(\frac{7}{5}\right)^n$. $l = \left(\frac{1}{5}\right)^n$.

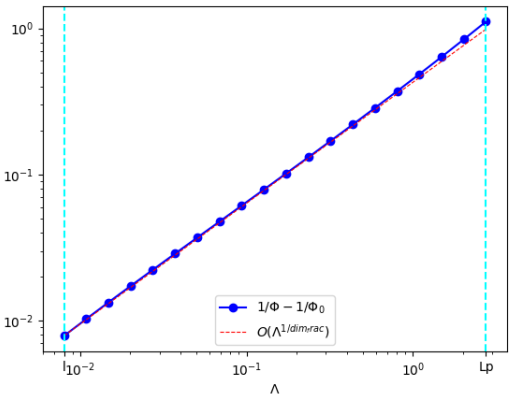
(a) $0 < \Lambda < \infty$



(b) $0 < \Lambda < l$



(c) $l < \Lambda < L_p$



(d) $l_p < \Lambda < \infty$

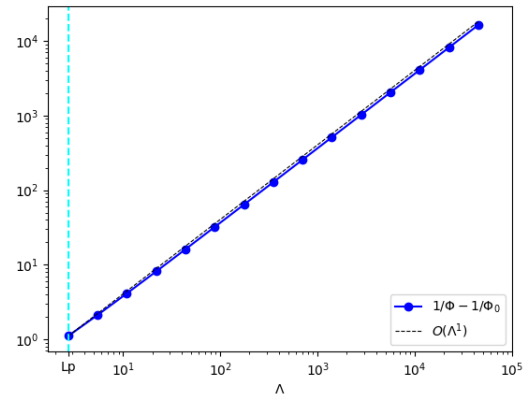
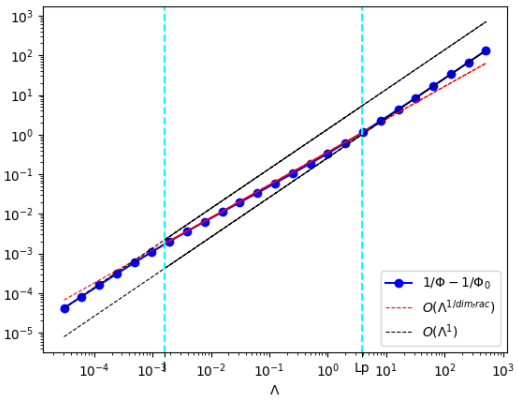
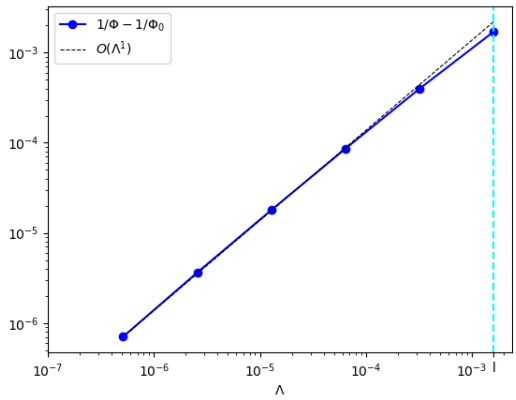


Figure 9: On 2d unit square with $n = 3$ square snowflake as in Fig. 8. $Z_\Lambda = 1/\Phi - 1/\Phi_0$ vs Λ when $D = 1$. $L_p = \left(\frac{7}{5}\right)^n$. $l = \left(\frac{1}{5}\right)^n$. $dim_{frac} = \frac{\log 7}{\log 5}$. Python script: main_flux.py.

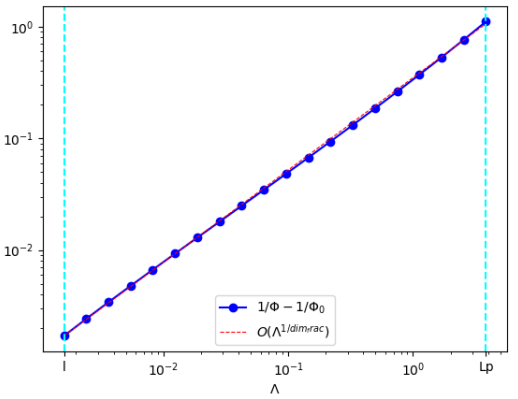
(a) $0 < \Lambda < \infty$



(b) $0 < \Lambda < l$



(c) $l < \Lambda < L_p$



(d) $l_p < \Lambda < \infty$

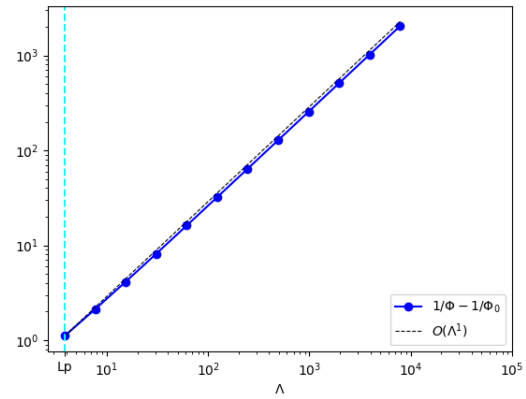


Figure 10: On 2d unit square with $n = 4$ square snowflake as in Fig. 8. $Z_\Lambda = 1/\Phi - 1/\Phi_0$ vs Λ when $D = 1$. $L_p = \left(\frac{7}{5}\right)^n$. $l = \left(\frac{1}{5}\right)^n$. $dim_{frac} = \frac{\log 7}{\log 5}$. Python script: main_flux.py.

4 Ω is unit cube and D is constant

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

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

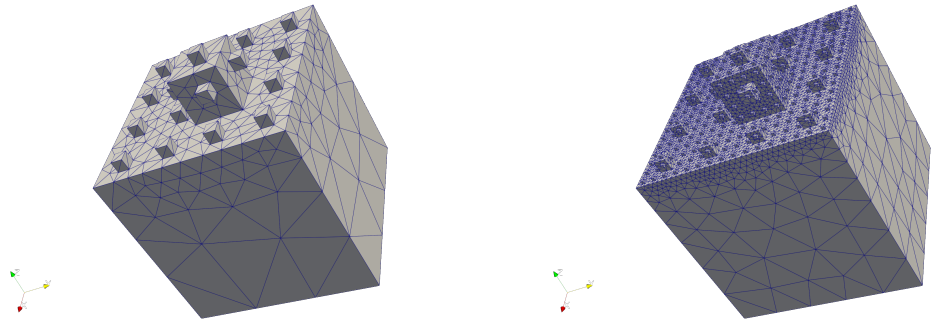
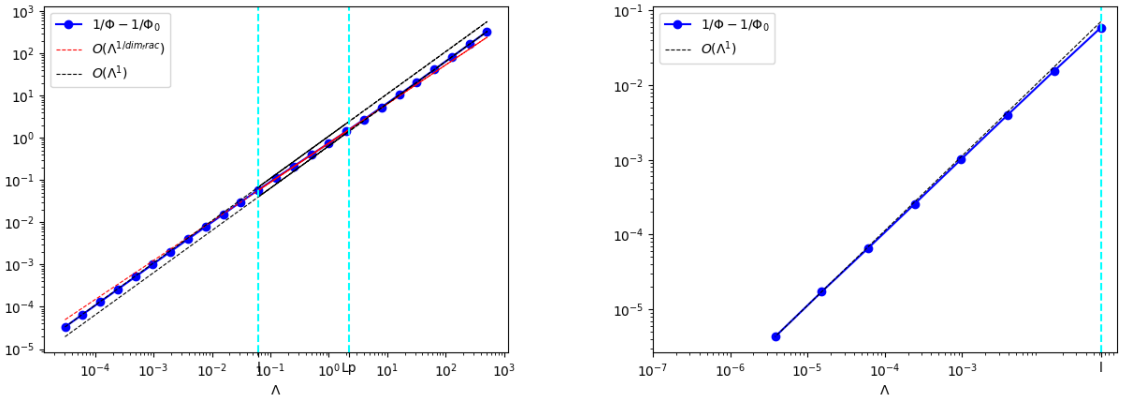


Figure 11: On 3d unit cube where the top is replaced by a square koch snowflake. $L_p = \left(\frac{6}{4}\right)^n$. $l = \left(\frac{1}{4}\right)^n$.

(a) $0 < \Lambda < \infty$ (b) $0 < \Lambda < l$



(c) $l < \Lambda < L_p$ (d) $l_p < \Lambda < \infty$

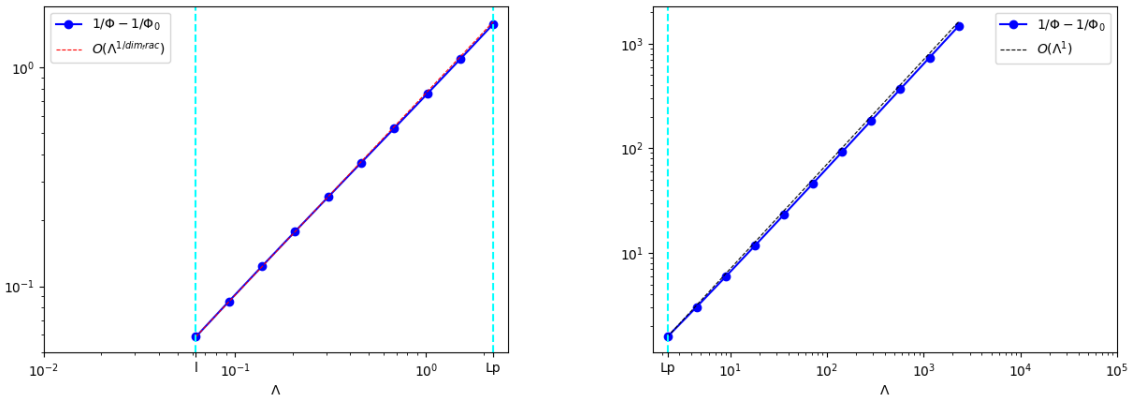
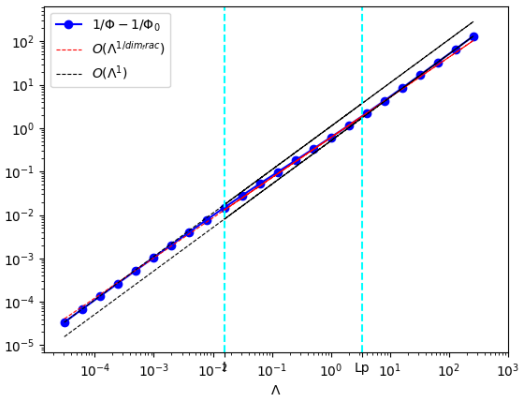
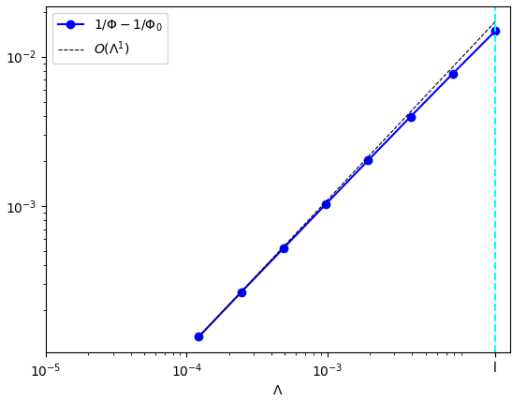


Figure 12: On 3d unit cube with $n = 2$ snowflake in Fig. 11. $Z_\Lambda = 1/\Phi - 1/\Phi_0$ vs Λ when $D = 1$. $L_p = \left(\frac{6}{4}\right)^n$. $l = \left(\frac{1}{4}\right)^n$. $dim_{frac} = \frac{\log 20}{\log 16}$. Python script: main_flux.py.

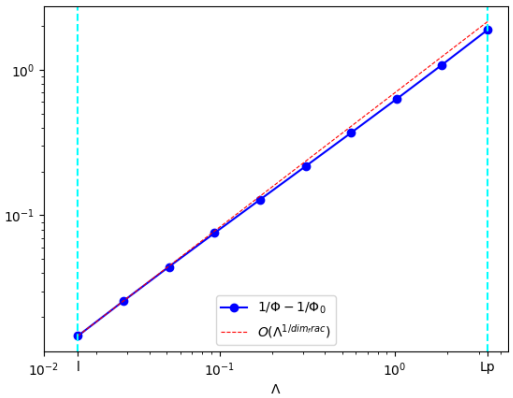
(a) $0 < \Lambda < \infty$



(b) $0 < \Lambda < l$



(c) $l < \Lambda < L_p$



(d) $l_p < \Lambda < \infty$

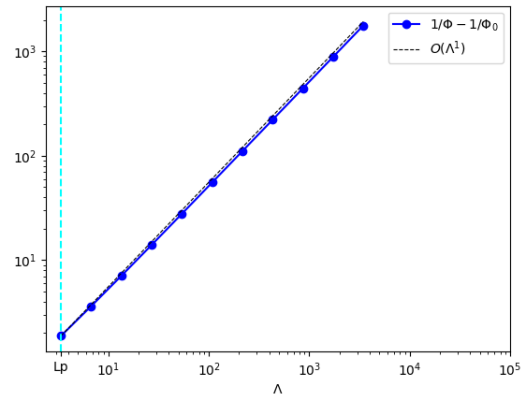


Figure 13: On 3d unit cube with $n = 3$ snowflake in Fig. 11. $Z_\Lambda = 1/\Phi - 1/\Phi_0$ vs Λ when $D = 1$. $L_p = \left(\frac{6}{4}\right)^n$. $l = \left(\frac{1}{4}\right)^n$. $dim_{frac} = \frac{\log 20}{\log 16}$. Python script: main_flux.py.