Course Project of Fast Algorithms for PDEs

A Simple Implementation of Boundary Integral Equations to Solve Laplace Equations

杨浩伦

2024-12-31

Table of Contents

1 Mathematical Formulation

In this project, we are interested in solving the interior Dirichlet problem for the Laplace equation[1] in $\mathbb{R}^2$:

$$\begin{cases} \Delta u(x) = 0, & x \in \Omega \subset \mathbb{R}^2, \\ u(x) = f(x), & x \in \Gamma \subset \mathbb{R}^2. \end{cases} \tag{1}$$

We look for a solution in the form of a double layer potential:

$$u(x) = D[\sigma](x) = \int_\Gamma \frac{\partial G(x,y)}{\partial n_y} \sigma(y) \mathrm{d}s_y, \tag{2}$$

where the fundamental solution G, in this specific case, is the following function

$$G(x,y) = -\frac{1}{2\pi} \ln ||x - y||. \tag{3}$$

From the jump relation, we know that

$$\lim_{z \in \Omega, z \to x} D[\sigma](z) = -\frac{1}{2}\sigma(x) + D[\sigma](x), \quad x \in \Gamma, \tag{4}$$

combined with the boundary equation, we obtain the following integral equation

$$-\frac{1}{2}\sigma(x) + D[\sigma](x) = f(x), \quad x \in \Gamma. \tag{5}$$

Now, consider the boundary as a parameterized curve $x(t) = (\xi(t), \eta(t))$, where t is the parameter between $[0, 2\pi)$. The tangent vector is $(\xi'(t), \eta'(t))$, hence the outer normal vector is $n = \frac{1}{\sqrt{\xi'(t)^2 + \eta'(t)^2}}(\eta'(t), -\xi'(t))$. After a change of variable, the integral equation (5) becomes

$$-\frac{1}{2}\sigma(x(t)) + \int_0^{2\pi} \frac{\eta'(s)\left(\xi(t) - \xi(s)\right) - \xi'(s)\left(\eta(t) - \eta(s)\right)}{2\pi\left(\left(\xi(t) - \xi(s)\right)^2 + \left(\eta(t) - \eta(s)\right)^2\right)} \sigma(x(s)) \mathrm{d}s = f(x(t)). \tag{6}$$

Remark: *the factor* $\frac{1}{\sqrt{\xi'(s)^2 + \eta'(s)^2}}$ *cancels because there is an extra norm factor in the length element* $ds_y = \sqrt{\xi'(s)^2 + \eta'(s)^2} ds$. *In addition, the kernel in* (6) *may seem singular at* $s = t$. *However, in the smooth boundary case, the singularity vanishes mathematically,*

[1] We only consider the region with a smooth boundary.

that is,

$$\lim_{s\to t}\frac{\eta'(s)\left(\xi(t)-\xi(s)\right)-\xi'(s)\left(\eta(t)-\eta(s)\right)}{2\pi\left(\left(\xi(t)-\xi(s)\right)^2+\left(\eta(t)-\eta(s)\right)^2\right)}=\frac{\xi''(t)\eta'(t)-\eta''(t)\xi'(t)}{4\pi\left(\xi'(t)^2+\eta'(t)^2\right)}. \tag{7}$$

But numerically, the rounding error may cause serious problems. Therefore, in practice, we need to derive the explicit expression for the kernel for a specific problem.

To solve for the density σ, we use the Nyström method. First, we discretize the integral in (6) by trapezoidal rule

$$-\frac{1}{2}\sigma_i+h\sum_{j=1}^{N}\frac{\eta'(t_j)\left(\xi(t_i)-\xi(t_j)\right)-\xi'(t_j)\left(\eta(t_i)-\eta(t_j)\right)}{2\pi\left(\left(\xi(t_i)-\xi(t_j)\right)^2+\left(\eta(t_i)-\eta(t_j)\right)^2\right)}\sigma_j=f_i, \tag{8}$$

for $i=1,2,\cdots,N$. The nodes $\{t_i\}$ are taken on $[0,2\pi)$ with equal distance h. And the point-wise approximation $\sigma_i\approx\sigma(x(t_i)), f_i=f(x(t_i))$. This forms a system of linear equations $\left(-\frac{1}{2}I+D\right)\sigma=F$, from which we can obtain the approximation of σ on the given nodes. Note that D is a dense matrix. If we want to approximate σ on an arbitrary node $t=t^*$, we can use the Nyström interpolation, that is, utilize the BIE (6) again and by trapezoidal rule:

$$\sigma(x(t^*))\approx 2\left(h\sum_{j=1}^{N}K(t^*,t_j)\sigma_j-f(x(t^*))\right). \tag{9}$$

Assume we have obtained the approximation of σ on M quadrature nodes by (9), we can then compute the double layer potential in (2) and get the solution u at the target point $x=(x_1,x_2)$:

$$u(x)\approx h_M\sum_{j=1}^{M}\frac{\eta'(t_j)\left(x_1-\xi(t_j)\right)-\xi'(t_j)\left(x_2-\eta(t_j)\right)}{2\pi\left(\left(x_1-\xi(t_j)\right)^2+\left(x_2-\eta(t_j)\right)^2\right)}\sigma_j. \tag{10}$$

This is a naive BIE algorithm that doesn't account for the singularity in the double layer potential when x is close to the boundary. Improvements can be made to resolve this issue. Furthermore, the evaluation of (10) can be accelerated by FMM.

2 A Naive Implementation

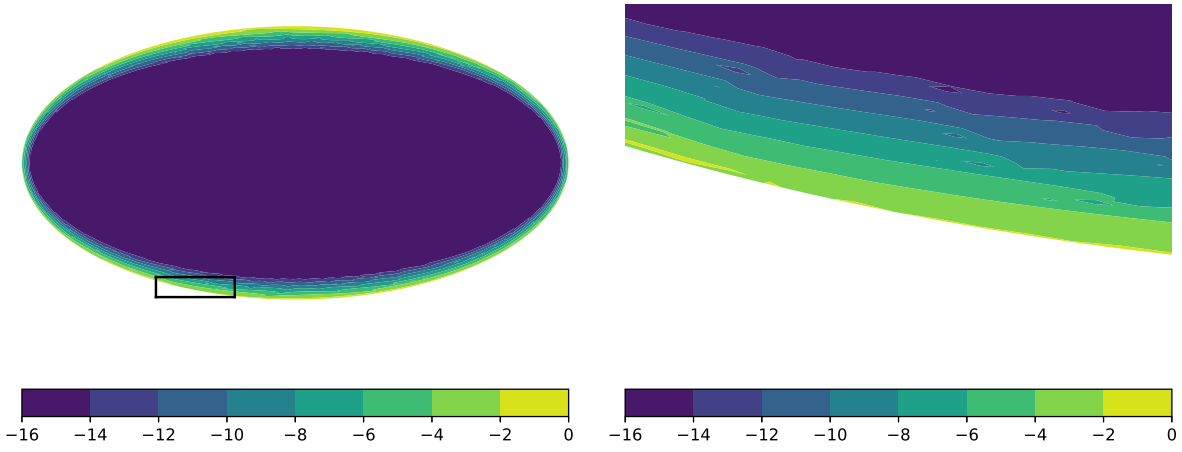
2.1 Elliptic Boundary

First, consider a simple case with a elliptic boundary with semi-major axis a and semi-minor axis b. The parameterized equation is $x(t) = (\xi(t), \eta(t)) = (a\cos t, b\sin t)$. Substitute this into equation (6), we will find that the kernel is

$$K(t,s) = -\frac{ab}{4\pi\left(a^2\sin^2\frac{t+s}{2} + b^2\cos^2\frac{t+s}{2}\right)}. \tag{11}$$

Note that this formula holds even when $t = s$. The infinitesimal terms in the numerator and the denominator cancel, leaving no singularity in the kernel, which saves us from the trouble of considering the case when $t = s$.

Consider the function $u(x_1, x_2) = e^{x_2}\cos x_1$, which satisfies the laplace equation. Use the Dirichlet boundary condition. The number of Nyström nodes (nodes used to evaluate the density in (8)) is $N = 256$. The number of quadrature nodes (nodes used to evaluate the DLP) is $M = 1.5N = 384$. And the number of target points in Ω is $256^2 = 65,536$. I use Python and this algorithm takes $\sim 2.3s$ on my Macbook. The error plot in logarithmic scale is shown in figure 1.



(a) Error in log scale $\log_{10}|u - u^*|$ (b) Zoom in the boxed area

Figure 1: Error plot of the elliptic boundary problem

When x is far from the boundary, the precision is $\sim 10^{-16}$. But as x approaches the boundary, the error grows exponentially, until it becomes $\sim 10^0$. This is expected since we haven't done anything to deal with the singularity in the DLP.

2.2 Boundary Not So Trivial

Now, let's look at a boundary not so trivial. In polar coordinates, the boundary is described by $r(\theta) = 1 + 0.3\cos(3(\theta + 0.3\sin\theta)), \theta \in [0, 2\pi)$. In this case, the expression for $(\xi^{(k)}(t), \eta^{(k)}(t)), k = 0, 1, 2$ is more complicated than before, and the expression for the kernel is long and tiresome. Therefore, instead of deriving an expression as (11), we use the primal formula (6), (7).

We still consider the solution $u(x_1, x_2) = e^{x_2}\cos x_1$ and Dirichlet boundary condition. The number of Nyström nodes is $N = 256$. The number of quadrature points is still $M = 256$, and the number of target points is $256^2 = 65,536$. The implementation takes $\sim 9.0s$. The error plot is shown in figure 2. Note that the precision is satisfying when x is far from the boundary, but the same issue occurs when x approaches the boundary.

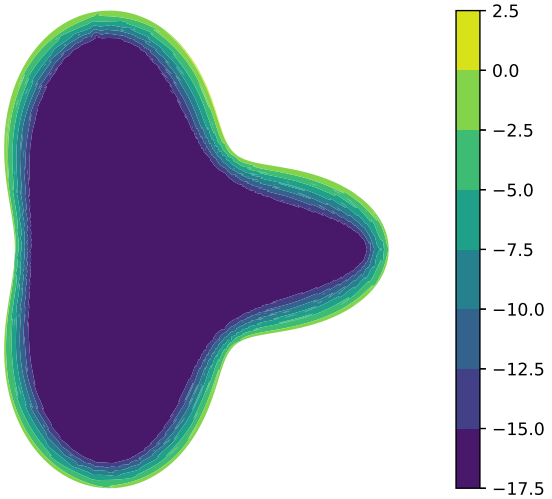


Figure 2: Error plot in log scale $\log_{10}|u - u^*|$ of the problem with boundary not so trivial

3 Some Improvement

3.1 Fast Multipole Method

The fast multipole method is able to reduce the computational cost when computing the N-body interactions, in our case, the double layer potential. The key idea is to expand the potential in the following way,

$$\sum_{j=1}^{N} q_j \ln(x_i - y_j) = \ln(x_i - y_0)\sum_{j=1}^{N} q_j - \sum_{k=1}^{P}\frac{1}{k}\left(\frac{1}{x_i - y_0}\right)^k \sum_{j=1}^{N} q_j (y_j - y_0)^k. \tag{12}$$

By truncating the infinite summation to the first P terms, we are able to reduce the computational cost, while keeping the computation accurate, given that x_i and $\{y_j\}$ are well-separated.

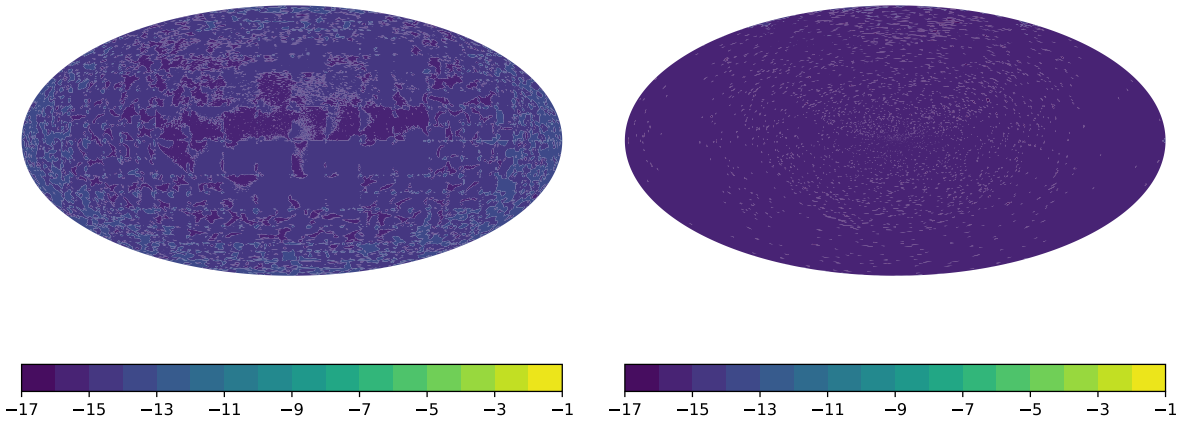
In our case, we are interested in computing the gradient of laplace kernel.

$$u(x) = h_M \sum_{j=1}^{M} -\frac{1}{2\pi} \left(\nabla_{y_j} \ln ||x - y_j|| \cdot n_j\right) \sigma_j. \tag{13}$$

The implementation of FMM is rather complicated. Here, I use the codes in https://github.com/dbstein/flexmm to achieve our goal. There are several things that we need to be careful with when we are using these codes. **First**, the author used `numba` library in Python to accelerate the computation. **Second**, because of the special structures of the codes, the function `FMM.evaluate_to_points` returns the two components of the weighted gradient in (13) simultaneously, and they share the same weight (input), which is not what we desired since the two components of n_j are clearly not identical. So we have to call this function twice, each with different weights. From the first time we draw the first component, and from the second time we draw the second, and sum them up to get the potential. Clearly, this is not the most efficient way and we can do something to improve it. But for now, let's just live with it and see how things turn out. **Third**, we need more than 256 Nyström nodes and 384 quadrature nodes to see the effect of FMM. The tree building process in FMM will cause extra efforts. When the nodes are few, it is often more efficient and accurate to simply use direct summation.

With FMM, let's go back to the first example where the boundary is an ellipse. Now, we set the number of Nyström nodes to $N = 1024$, and the number of quadrature nodes to $M = 10N = 10,240$. The number of target points in Ω is $1024^2 = 1,048,576$. There are some extra parameters to determine for FMM: the number of truncated terms $P = 150$; the maximum number of points in each leaf node (the boxes that don't have children) is 50. The results are shown in figure 3.

This implementation with FMM takes $\sim 27.5s$ to finish, while the one without FMM takes $\sim 240.0s$. In this setting, the algorithm with FMM is more than 8 times faster than the naive algorithm. Observing the error plot, we can see that the error in figure 3a isn't as great as in figure 3b. As x moves closer to the boundary, the color becomes lighter and lighter. The corresponding error changes from $\sim 10^{-16}$ to $\sim 10^{-14}$ and further to $\sim 10^{-12}$, while the error in figure 3b is $\sim 10^{-16}$ most of time. Even so, the error when using FMM is still acceptable in most cases. It helps us reduce the computational effort at the cost



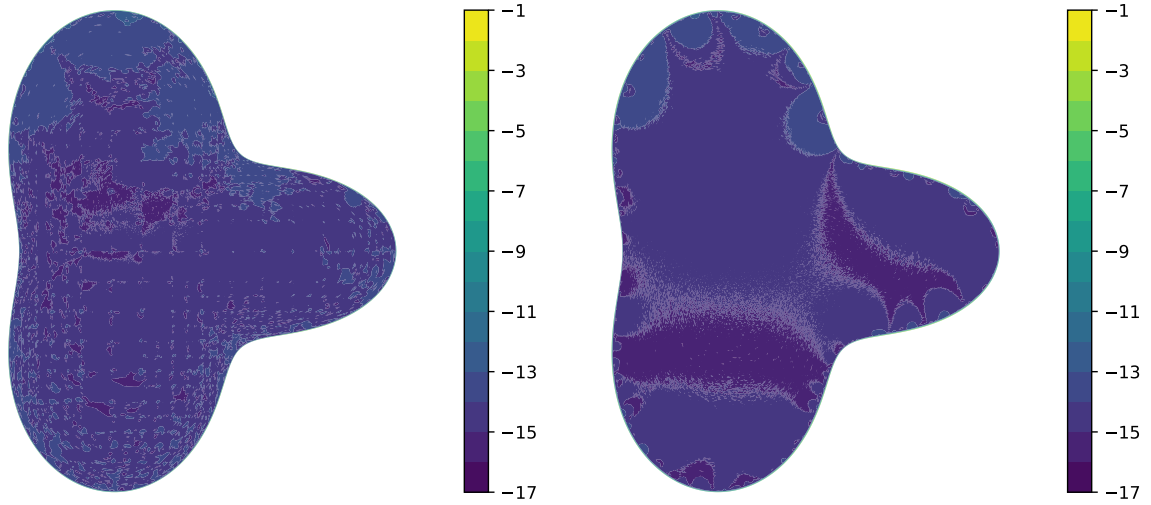
(a) Error in log scale $\log_{10}|u-u^*|$ using FMM to compute DLP

(b) Error in log scale $\log_{10}|u-u^*|$ for naive implementation

Figure 3: Error plot of the elliptic boundary problem in a larger scale $N = 1,024$, $M = 10,240$ with 1024^2 target points

of some precision because we have truncated P terms. This is a tradeoff between the computational cost and precision. If we want it to be more accurate, just set larger P, and it's just going to take more time for finish.

Now, let us implement the second example in section 2.2 again with the help of FMM. We set the number of Nyström nodes to $N = 512$, and the number of quadrature nodes to $M = 10N = 5,120$. The number of target points in Ω is $512^2 = 262,144$. The number of truncated terms and the maximum number of points in each leaf node are still 150 and 50 respectively.



(a) Error in log scale $\log_{10}|u-u^*|$ using FMM to compute DLP

(b) Error in log scale $\log_{10}|u-u^*|$ for naive implementation

Figure 4: Error plot of the untrival boundary problem in a larger scale $N = 512$, $M = 5,120$ with 512^2 target points

The algorithm with FMM takes $\sim 21.5s$, while the other one without FMM takes $\sim 209.3s$.

3.2 Quadrature by Expansion

As we have seen in figure 1 and 2, when x approaches the boundary, the error grows exponentially. This is due to the near-singularity of the double layer potential

$$u(x) = \int_\Gamma -\frac{1}{2\pi} \left(\nabla_y \ln ||x-y|| \cdot n_y\right) \sigma(y) \mathrm{d}s_y. \tag{14}$$

Discretizing it by trapezoidal rule will lead to unbounded error. Quadrature by expansion (QBX) is a method to deal with this kind problem. If we use a complex function to describe the boundary $\Gamma : Z(s) = \xi(s) + i\eta(s), 0 \le s < 2\pi$. Then, for the target point $x = (x_1, x_2)$, consider the following Cauchy integral

$$v(z) = -\frac{1}{2\pi i} \int_\Gamma \frac{\sigma(y)}{y-z} \mathrm{d}y, \qquad \text{where } z = x_1 + ix_2, \tag{15}$$

and claim that the DLP is $u(x) = \mathrm{Re}(v(z))$. To see this, we use the parametrization of Γ mentioned above, and replace $y = \xi(s) + i\eta(s)$ in the integral. We get

$$v(z) = -\frac{1}{2\pi i} \int_0^{2\pi} \frac{\widetilde{\sigma}(s)}{(\xi(s) - x_1) + i(\eta(s) - x_2)} (\xi'(s) + i\eta'(s)) \mathrm{d}s. \tag{16}$$

After removing the complex part in the denominator and some simplification, we will find that

$$\mathrm{Re}(v(z)) = \frac{1}{2\pi} \int_0^{2\pi} \frac{\xi'(s)(\eta(s) - x_2) - \eta'(s)(\xi(s) - x_1)}{(\eta(s) - x_2)^2 + (\xi(s) - x_1)^2} \widetilde{\sigma}(s) \mathrm{d}s, \tag{17}$$

which is exactly the double layer potential after a change of variable.

Remark: This formulation is something I was confused about in my presentation. I was wondering where does the directional derivative come from in (15). And now I've figured it out!

Define the box near the boundary using the parametrization of the complex plane (let's ignore the details and cut to the chase). The width of each box is $5h$, where h is the distance of two adjacent Nyström nodes. Each box has a center z_0, around which we can define the Taylor expansion

$$v(z) = \sum_{m=1}^{\infty} c_m (z - z_0)^m, \quad \forall z \in B(z_0), \tag{18}$$

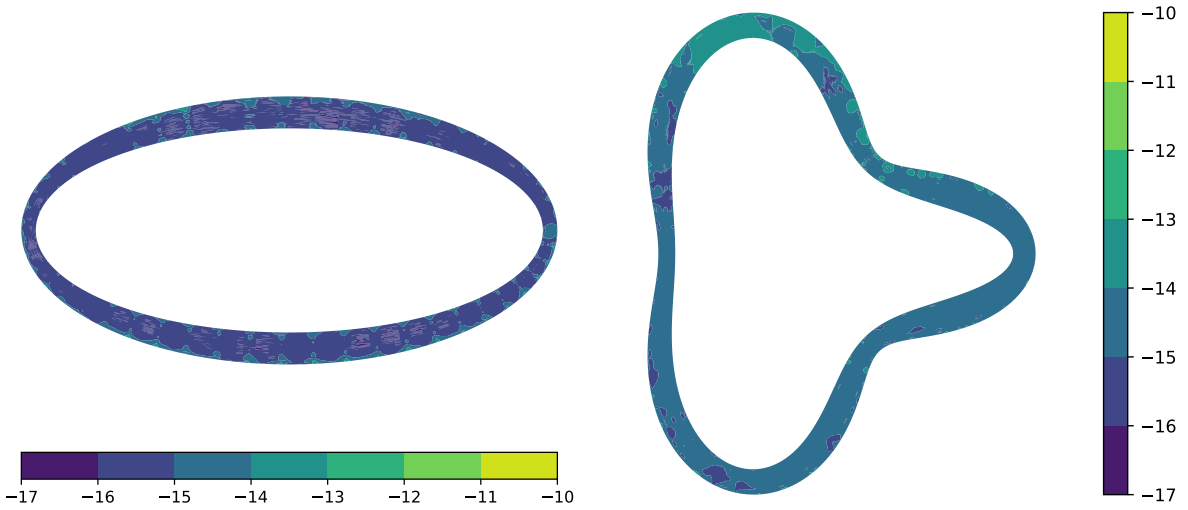
where $B(z_0)$ is the box which z_0 is in. And recall the special property of the Cauchy integral. It's expansion coefficients are

$$c_m = \frac{i}{2\pi}\int_\Gamma \frac{\sigma(y)}{(y-z_0)^{m+1}}\mathrm{d}s_y = \frac{i}{2\pi}\int_0^{2\pi} \frac{\widetilde{\sigma}(s)}{(Z(s)-z_0)^{m+1}}Z'(s)\mathrm{d}s. \tag{19}$$

Use trapezoidal rule to evaluate (19) and truncate the Taylor expansion to the first p terms.

$$\begin{cases} \hat{c}_m = \frac{i}{M}\sum_{j=1}^{M} \frac{Z'(s_j)\sigma_j}{(Z(s_j)-z_0)^{m+1}}, \\ \hat{v}(z) = \sum_{m=1}^{p} \hat{c}_m (z-z_0)^m. \end{cases} \tag{20}$$

We use this technique to solve the two Dirichlet problems mentioned above. We use the same setting in two problems: the number of Nyström nodes is 256, the number of quadrature nodes is $M = 6N = 1,536$, the number of boxes along the boundary is $[N/5] = 51$, and the number of truncated terms is $p = 10$. Here we are only interested in the target points close to the boundary, i.e. the points in the boxes. For the target points far from the boundary, we just use the naive algorithm (even better with FMM) and save the computation here. The error plots are shown in figure 5.



(a) Error in log scale $\log_{10}|u-u^*|$ using QBX near the elliptic boundary

(b) Error in log scale $\log_{10}|u-u^*|$ using QBX near the nontrivial boundary

Figure 5: Error plot of the two Dirichlet problems using QBX near the boundary and ignore those far from the boundary

The error near the boundary is well controlled. At first glance, you may feel the color is relatively light compared with figure 1 and 2. But notice that I have used a colorbar with a new range! Combined with the naive algorithm we mentioned earlier, we have successfully solved these equations to a satisfying precision.

4 Another Method

Now we try to use another approach to solve the two problems mentioned above. I choose Finite Element Method, which is a powerful method and frequently used in industry. The basic formulation of the FEM is as follows. Assume V is the linear space where the solution resides. And V_h is a finite dimensional subspace of V, with basis $\{\phi_1, \phi_2, \cdots, \phi_M\}$. We seek a bilinear functional $a(\cdot, \cdot)$ and a linear functional L such that

$$a(u_h, v) = L(v), \quad \forall v \in V_h, \tag{21}$$

where $u_h = \sum_{j=1}^{M} \alpha_j \phi_j$. Thus, it forms a system of linear equations

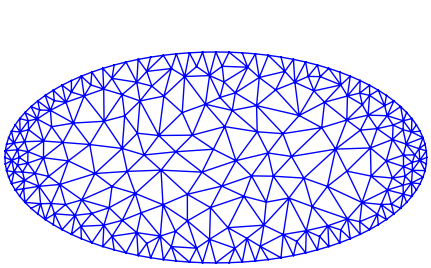
$$\sum_{j=1}^{M} \alpha_j a(\phi_j, \phi_i) = L(\phi_i), \quad i = 1, 2, \cdots, M, \tag{22}$$

from which we can solve for $\{\alpha_j\}$ and find u_h.

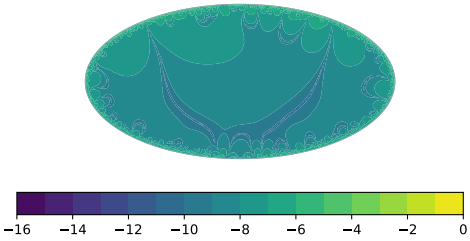
For the 2D Laplace equation $\Delta u = 0$ with Dirichlet BC $u|_{\partial\Omega} = f$. Its weak form is

$$\int_\Omega \nabla u \cdot \nabla v \mathrm{d}x\mathrm{d}y = \int_\Gamma \frac{\partial u}{\partial n} v \mathrm{d}s. \tag{23}$$

Then, we need to form a triangulation of the domain, as shown in figure 6a. And the basis function is the piece-wise linear function. Under a certain setting, we get the results in figure 6b. The overall error is $\sim 10^{-7}$, which is acceptable but not as good as the solution from the BIE method. But it's not fair to make such comparison, since these two methods are fundamentally different and have very different parameters. Changing the parameters and the setting will significantly affect the precision of the solution. There is one thing that's worth mentioning. The error along and near the boundary is well controlled. Because of the special formulation of the FEM, there is no such issues as the singular quadrature near the boundary.



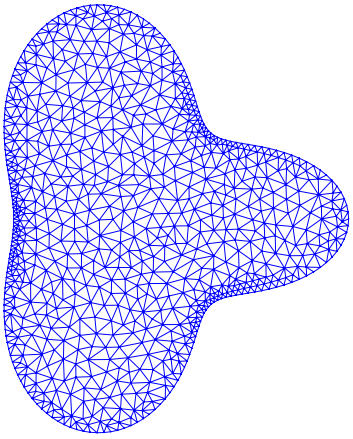
(a) Triangulation of the elliptic domain



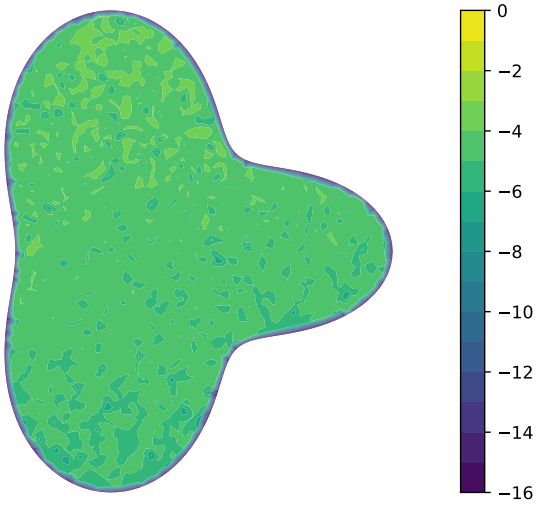
(b) Error in log scale $\log_{10}|u - u^*|$ using FEM for the problem with elliptic boundary

Figure 6: The results using FEM to solve the Dirichlet problem with the elliptic boundary

The triangulation and the error plot of the second example are shown in figure 7.



(a) Triangulation of the nontrivial domain



(b) Error in log scale $\log_{10}|u - u^*|$ using FEM for the problem with nontrivial boundary

Figure 7: The results using FEM to solve the Dirichlet problem with the nontrivial boundary

5 Summary

In this work, we are trying to solve a 2D interior Dirichlet problem for the Laplace equation with boundary integral equation method. First we went over the mathematical formulation of this method. Understanding how this method actually works is essential for us to improve the algorithm. There are two key points. One, discretize an integral to solve for the density along the boundary. Two, evaluate the double layer potential to find the solution u. With these two principles, We implemented this algorithm from scratch and without special techniques. We call it naive implementation, from which we got the solutions for two Dirichlet problems, one with an elliptic boundary, and the other one

with a non-trivial boundary (See figure 1 and 2). We observed that the error is controlled at nearly machine precision when x is far from the boundary, but exponentially grows when x approaches boundary. We used FMM to accelerate the computation of double layer potential. With more source points and target points, it ran almost $8 \sim 10$ times faster than the naive implementation, while the error is larger, but still acceptable (See figure 3 and 4). Then, we used QBX to deal with the near-singularity of the double layer potential, which indeed significantly reduced the error when x is close to the boundary (See figure 5). Finally, we used FEM to solve the same problems, just for your information (See figure 6 and 7). The comparison, as I've stated before, was not really fair since it has a radical difference from the BIE method.

6 Codes

I've uploaded the codes and figures for all the work I've done above to the github. The link is BIE_project. It's my first time to upload a github repository. Sorry for the rough editing.