Malmhedn's Theorem

Tigran Tadevosyan

May 2019

Abstract

In this document I would try to give a brief introduction to Malmheden's theorem. Latter in the document a python implantation for the algorithm is given.

Contents

1 Introduction

1.1 Definition of Problem

Malmeden's algorithm is an elegant way of solving Dirichlet problem for Laplace equation in an n-dimensional ball. In a figure below you can see the formulation of the problem.

$$\begin{cases} \Delta u(c) = 0, & c \in B(c_0, r) \\ u(c) = \varphi(c), & c \in \partial B(c_0, r) \end{cases}$$

With the help of the Malmeden's algorithm we can find the value for that harmonic function u with any valid boundary condition function φ for ball centered at c_0 and radius r.

1.2 Definition of Malmeden's Theorem

Let's take an n-dimensional ball with radius r. Let's define a function v as the linear interpolation of values obtained by the boundary condition function φ on the border of that ball. If we calculate the average of the integrate of that function over the whole boundary we will obtain the get the harmonic function inside that ball which satisfies the boundary condition which means the solution of the Dirichlet problem.

2 A Proof Malmheden's theorem

For simplicity let's look at the case of Dirichlet problem inside a disk.

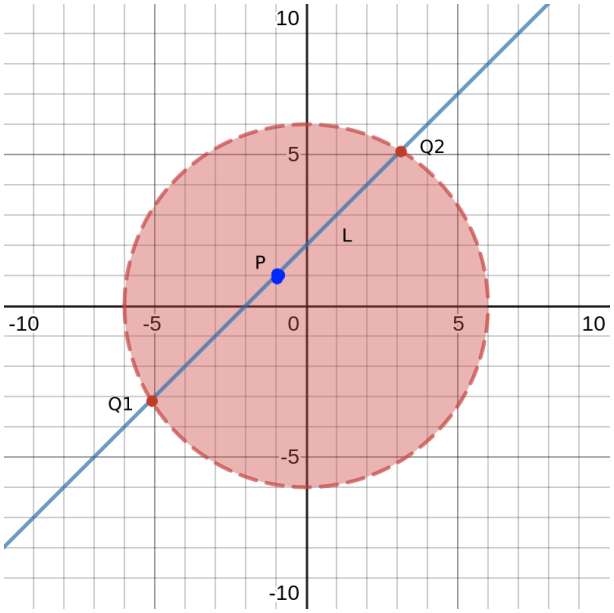

Figure 1: Disk

Let's define Ω as a disk with a boundary Γ. After drawing a chord L in Ω it will intersect Γ in two points' Q_1 and Q_2. Our boundary function φ will obtain values φ_1 and φ_2 at Q_1

and Q_2. Let's define yet another function called υ that will calculate the value of a point inside a disk with linear interpolation. For example υ can interpolate any value on chord L from values φ_1 and φ_2. If we integrate the function υ over the angle θ from 0 to 2π and divide the result by the 2π we will obtain the average of linear interpolation and from the Malmheden's theorem it should be the harmonic function we are seeking. So if we prove that the function we obtain by averaging that integration is harmonic we will prove Malmheden's theorem.

$$\upsilon(P) = \frac{r_1 f_2 + r_2 f_1}{r_1 + r_2}$$

This is what the υ will look like where P is the point we are trying to interpolate and r_1 and r_2 are the distances of P from Q_1 and Q_2 respectively. If we turn to polar coordinates we obtain $\upsilon(P, \theta)$ where θ is the angle of chord makes with the positive direction of $x - axis$. Let's calculate the u function with everything we already got.

$$u(P) = \frac{1}{2\pi} \int_0^{2\pi} \upsilon(P, \theta) dx$$

Obviously we obtain that $r_1(\theta) = r_2(\theta + \pi)$ because it is the same chord with opposite direction. So we can say that $f_1(\theta) = f_2(\theta + \pi)$. From that we can update υ function.

$$\upsilon(P, \theta) = 2\frac{r_2 f_1}{r_1 + r_2}$$

Let's define α as the angle between inner normal n to Γ at Q_1 and the chord L. Turns out that if we compare slight changes in arclength s on γ and arclength of the circle of radius $r1$ centered at P we will obtain that $r_1 d\theta = \cos(\alpha) ds$. So after multiplying the top of function υ from last iteration with $\cos(\alpha) ds$ and the bottom $r_1 d\theta$ after some modifications we obtain.

$$u(P) = \frac{1}{\pi} \int_0^c \frac{\cos(\alpha)}{r_1} f_1 ds - \frac{1}{\pi} \int_0^c \frac{\cos(\alpha)}{r_1 + r_2} f_1 ds$$

Where c is the length of Γ.

The second integral is just a constant depending on the radius of the disk and boundary function. If we replace $\frac{\cos(\alpha)}{r_1}$ by $\frac{\partial \log|P - Q_1|}{\partial n_{Q_1}}$ because they are equal we will obtain that the first integral is the "double layer" potential of f_1 which is a harmonic functions. So that means our function u is a harmonic function that satisfies the boundary condition, so it is the solution to Dirichlet problem.

3 Algorithm Implementation

For the implementation of the algorithm I have use python 3 and PyQT4 for UI. All the codes are available on github with this link.

The script containing the computation of the Dirichlet problem is in the *computer.py*. The function that makes the computation is name *compute*, it takes as arguments x and y coordinates of the point inside disk needed to calculate, the r radius of that disk and *bdry*

the boundary function for the calculation. This function fill only calculate the value for the two dimensional case and for a disk centered at the $(0, 0)$. By using the integrate function from the *scipy* library the *calculate* function is calculation the *integrable* function over θ form 0 to 2π and divides it by 2π to get the average value. The *integrable* function takes as arguments coordinates of the function needed to be calculated, radius of the disk and the boundry function and it returns a function with parameter θ so we can integrate that function over θ. The function *integrable* is construction the integrable function by calculating the $(x1, y1, x2, y2)$ coordinates of the intersection of the chord L with the $x - axis$ which makes θ angle with it. There is a separate function called *getIntersectionPoints* for it which gets the linear equation $y = ax+b$ for the chord where $a = \tan(\theta)$ and b is can be substituted from $b = y-ax$ after calculating a. After that it using the equation $r^2 = x^2+y^2$ it calculates the coordinate's of the intersections on Γ. After that we calculate distances between point needed to calculate and the first and second intersections and store them in variables r_1 and r_2 respectively. We do that by using the function *distanceBetweenPoints* which just calculates $\sqrt{x1 - x2)^2 + (y1 - y2)}$. After that we calculate values f_1 and f_2, which are the values of baundry function at the intersection of the cord. The last step is the return of the linear interpolation $\frac{r_1 f_2 + r_2 f_1}{r1+r2}$. As mentioned before we pass that function back to the *compute* function which integrates it.

After that with the UI created with PyQt user can input (x, y) coordinates, radius r and select boundary function from list of available ones and calculate the value of the solution of the Diriclet problem at that point. Morover there is an additional functionality that allows user to draw 2D garph of that function for any fixed x or y. The last piece of software is the availability of drawing the 3D graph of the function with the use of *opengl* and *pyqtgraph* librares.

The main error of this implementations are that the calculation of the tan and integral are approximations with the use of *numpy* library and that value for every point of that function is taking place separately so we don't get the harmonic function we are seeking we just approximate the value at the points we need(Tho I would like to mention that the approximation is very good, it's max error is about 0.001 form my experience). Because we have to calculate every value of that harmonic function separately and with so many steps it may take a bit of time which can be noticed while drawing the 3D graph of the harmonic function especially with calculations heavy boundary function.

4 Ending

4.1 Interesting Remarks

Years after the discover of the Malmheden's theorem, other mathematician tried to extend it to all convex regions and even star shaped boundaries, but it was proved that if the Malmheden's algorithm solves the Dirichlet problem that the boundary region must be a n-dimensional ball. In this paper only the disk case was proved but it is important to notice that Malmheden's theorems is proved to work for any n-dimensional ball. Later it was proved that Malmheden's theorem also works for poly-harmonic function, for example it solves this

Dirichlet problem for the bi-harmonic case:

$$\begin{cases} \Delta^2 u = 0, \\ u = \varphi, \\ \nabla u = \nabla \varphi \end{cases}$$

4.2 Conclusion

Malmheden's theorem give an elegant geometric solution to Dirichlet problem, which is much faster and easier to apply than for example Poisson Kernel method. It give as more inside into how harmonic and poly-harmoc functions behave.