Neural networks for boundary value problems

Hoang Trung Hieu
Supervisor: Izsák Ferenc

Abstract

The purpose of this work is to introduce two approaches to solve boundary value problem. We use mesh-free methods which are based on the relation between fundamental solutions and can be implemented by neural networks. In the first approach, we construct a direct approximation of analytical solution as a linear combination of fundamental solutions based at outer points. In the second approach, we try to find a mapping between fundamental solutions at boundary points and interior points. The first one is memory-efficient and the learning data in the second one is independent of the boundary conditions. Both of these methods work efficiently giving small errors in a variety of domain's shape and in case of different boundary conditions. Moreover, the different neural networks can lead to the same accuracy.

1 Introduction

In the last decade, the development of the practice and software tools for neural networks made it as one of the most powerful tools in applied mathematics. In this way, it is a natural attempt, to apply this tool somehow for the solution of the major problems in numerical analysis. An important class of these problems is the numerical solution of PDE's. Accordingly, some related works were published in the last years. The main research direction was to mimic the geometry of the domain and the corresponding finite element or finite difference discretization, called the *physics informed neural networks*. At the level of the linear solvers, their motivation was the multigrid method and related convolutional neural networks were constructed. This is extended to a number of PDE's and a whole library of neural networks was prepared for computing [13].

At the same time, some theories and a number of promising numerical experiments are available for mesh-less methods, where one can bypass the construction of a computational grid. The main benefit of using a related approach, the *boundary element methods* is that only the boundary has to be discretized. In the conventional finite difference and finite element methods, the domain is also discretized. At the same time, when solving partial differential equations with inhomogeneous conditions, the boundary element methods are tend to be less efficient because integral reformulations generally involve a domain integral. The evaluation of this term may consume the majority of the computation time. For further details and well-developed convergence analysis, we refer to [16].

Similarly to boundary element methods, *the method of fundamental solutions* [3] can be used for both homogeneous and inhomogeneous elliptic partial differential. This approach became popular because of its simplicity and rapid convergence. This was verified only in a simple case, for a disk domain in [17] and extended to other similar domains in [8]. At the same time, for a general domain Ω a convergence rate in Sobolev norms has still not been derived.

We start from this approach and make it even more efficient by applying the armoury of machine learning. In particular, we use neural networks with different structures for linear elliptic problems. To be more specific, we consider the numerical solution of well-posed boundary value problems for the Laplace's equation

$$\Delta u = 0 \text{ in } \Omega, \tag{1}$$

and for Helmholtz equations

$$\Delta u \pm k^2 u = 0 \text{ in } \Omega, \tag{2}$$

where Ω is a bounded simply connected set with boundary $\partial\Omega$.

The paper is divided into five sections. In section 2, in order to solve the Laplace equation, we introduce the first approach mainly based on the method of fundamental solution for solving Laplace's equation with Dirichlet, Neumann or mixed type boundary conditions. And we modify the learning data of neural networks by approximating fundamental solutions at

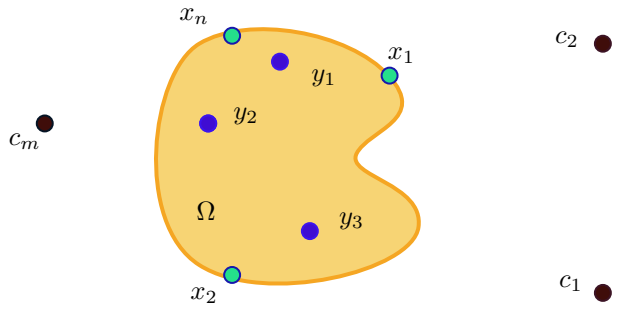

Figure 1: Inner, Boundary, Outer points

interior points by a linear combination of fundamental solutions on boundary points. In section 3, the Helmholtz equations are investigated by both above method. Section 4 lists meaningful results about the convergence analysis for both Laplace and Helmholtz equation, analytical and non-analytical solution. The accuracy, the efficiency, the optimal way to tune parameters of both approaches can be seen in section 5 through different examples. We also give a comparison of advantages and disadvantages in this section.

2 Solving Laplace equations by two approaches

2.1 The first approach

2.1.1 The Laplace's equation with Dirichlet boundary conditions

Let us consider the Laplace equation

$$\begin{cases} \Delta u = 0 & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases} \tag{3}$$

for the unknown function $u \in H^1(\Omega)$ on the domain $\Omega \subset \mathbb{R}^d$ with Dirichlet boundary conditions given by $g \in H^{\frac{1}{2}}(\partial\Omega)$.

In the sense of distribution, the fundamental solution is a solution of the first equation only in the entire space and its form is

$$\phi : \mathbb{R}^d \setminus \{\mathbf{0}\}, \quad \phi(x) = \begin{cases} -\frac{1}{2\pi}\log(|x|) & \text{for } d = 2 \\ \frac{1}{(d-2)\omega_n |x|^{d-2}} & \text{for } d \geq 3 \end{cases}.$$

Take n arbitrary boundary points $x_1, x_2, \ldots, x_n \in \partial\Omega$ and m arbitrary points $c_1, c_2, \ldots, c_m \in \Omega^C$, which will be called the outer points, see Figure 1. The fundamental solution at associated with the point c_j in case of $d = 2$ is

$$\phi_{c_j}(x) = -\frac{1}{2\pi}\log(|x - c_j|). \tag{4}$$

We will approximate the solution of (3) as a linear combination of the functions $\{\phi_{c_j}\}_{j=1}^m$. This method was first proposed in [12], whenever for its convergence on an arbitrary domain still no rigorous proof was developed.

A main motivation of this approximation is the fact that according to the theory of boundary integral equations [16], there is a unique function $\mathcal{G} \in H^{-\frac{1}{2}}(\partial\Omega)$ such that

$$u(x) = \int_{\partial\Omega} \phi_0(x - y)\mathcal{G}(y)\, \mathrm{d}y. \tag{5}$$

If this could be approximated with an appropriate sum, we would get a sum of some terms $\phi_{c_j^*}$. At the same time, on the boundary point c_j^*, the function $\phi_{c_j^*}$ becomes singular. Therefore, we rather choose instead the outer point c_j close to c_j^*.

In any case, if we take more outer points, we can enhance accuracy of the approximation. In concrete terms, we are looking for the approximation

$$u(x) \approx \sum_{j=1}^m a_j \phi_{c_j}(x) \quad (+\text{constant}) \tag{6}$$

where the coefficients a_j will be determined by the boundary condition such that

$$\sum_{j=1}^m a_j \phi_{c_j}(x_i) \approx g(x_i). \tag{7}$$

In the language of neural networks, we are looking for the parameters $a_1, a_2, \ldots, a_m$ such that the corresponding linear mapping in (6) delivers a minimal error in (7). Accordingly:

- The input of training data includes n vectors of length m: $(\phi_{c_1}(x_i), \cdots, \phi_{c_m}(x_i))$, where $i = 1, \cdots, n$.
- The output of training data are boundary function at the point x_i : $g(x_i)$, where $i = 1, \ldots, n$).
- Weights in the most simply case (no hidden layer) are a_j and in the general case are components to get a_j.
- The loss function is least square loss with regularization.

Using the gradient-based methods and tuning the parameters properly to minimize the mean squared error we obtain the desired parameters $a_1, a_2, \ldots, a_m$. This model gives us the approximation of u at any inner point, let say y_k, then

$$u(y_k) \approx \sum_{j=1}^m a_j \phi_{c_j}(y_k).$$

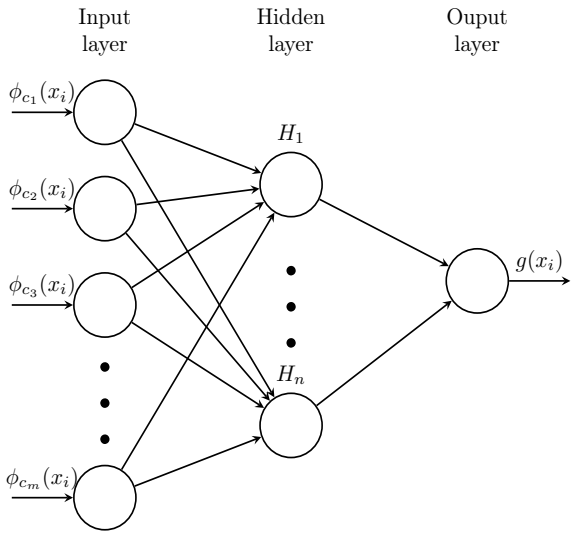

Figure 2: Neural network structure of the first approach

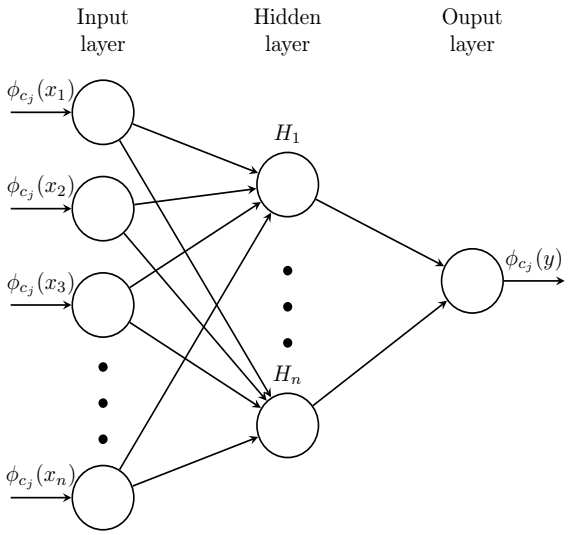

Figure 3: Neural network structure of the second approach

2.1.2 The Laplace's equation with Neumann boundary conditions

The general interior Neumann problem for Laplace's equation in Ω can be formulated as follows:

$$\begin{cases} \Delta u = 0 \quad \text{in} \quad \Omega \\ \frac{\partial u}{\partial \mathbf{n}} = f_{\mathbf{n}} \quad \text{on} \quad \partial\Omega \end{cases} \tag{8}$$

where $\mathbf{n}$ is a unit outward normal vector to the boundary $\partial\Omega$ and $f_{\mathbf{n}}$ is a prescribed function defined on the Lipschitz boundary $\partial\Omega$ of the domain Ω.

Again, we are looking for an approximation in form (6). At the same time, we have to replace the Dirichlet boundary conditions with Neumann type boundary conditions such that we obtain

$$\frac{\partial}{\partial \mathbf{n}} u(x_i) \approx \sum_{j=1}^{m} a_j \frac{\partial}{\partial \mathbf{n}} \phi_{c_j}(x_i). \tag{9}$$

Here using $c = [c_1, c_2]$ for a generic outer point and $x = [x_1, x_2]$ for a generic inner point we have

$$\frac{\partial}{\partial x_1}\phi_c(x) = -\frac{1}{2\pi}\frac{x_1 - c_1}{(x_1 - c_1)^2 + (x_2 - c_2)^2}$$
$$\frac{\partial}{\partial x_2}\phi_c(x) = -\frac{1}{2\pi}\frac{x_2 - c_2}{(x_1 - c_1)^2 + (x_2 - c_2)^2}.$$

Accordingly, the new neural network should be altered in terms of the inputs and outputs as follows:

- Training data input includes vectors $\left(\frac{\partial}{\partial \mathbf{n}}\phi_{c_1}(x_i), \cdots, \frac{\partial}{\partial \mathbf{n}}\phi_{c_m}(x_i)\right)$ where $i = 1, \cdots, n$.
- Training data output are boundary function at the point x_i : $f_{\mathbf{n}}(x_i)$ $(i = 1, \cdots, n)$.

After training, we obtain the weights in the approximation (6).

2.1.3 The Laplace's equation with mixed boundary conditions

The equation (3) is said to satisfy a mixed boundary condition if consisting $\partial\Omega$ of two disjoint parts, Γ_1 and Γ_2, such that $\partial\Omega = \Gamma_1 \cup \Gamma_2$, and u satisfies the following equations:

$$u|_{\Gamma_1} = g \quad \text{and} \quad \left.\frac{\partial u}{\partial \mathbf{n}}\right|_{\Gamma_2} = f. \tag{10}$$

In this case, the neural network is a combination of the two networks above: the Dirichlet input-output training data is: $(\phi_{c_1}(x_i), \cdots, \phi_{c_m}(x_i))$ and $g(x_i)$ where $x_i \in \Gamma_1$ and the Neumann input-output training data includes $\left(\frac{\partial}{\partial \mathbf{n}}\phi_{c_1}(x_i), \cdots, \frac{\partial}{\partial \mathbf{n}}\phi_{c_m}(x_i)\right)$ and $f_{\mathbf{n}}(x_i)$ where $x_i \in \Gamma_2$.

2.2 The second approach

2.2.1 The Laplace's equation with Dirichlet boundary conditions

We discuss another approach to design the neural network which uses only the fundamental solutions as learning data. Here, for each outer point c_j,

- The input of training data is a vector of length n: $\left(\phi_{c_j}(x_1), \cdots, \phi_{c_j}(x_n)\right)$.
- The output of training data is $\phi_{c_j}(y)$ where y is an interior point where we want to estimate the solution.

Here we assume that fundamental solution at any inner point can be written as linear combination of funda-

mental solution at boundary points as

$$\phi_{c_j}(y) \approx \sum_{i=1}^{n} b_i \phi_{c_j}(x_i). \tag{11}$$

Using approximation (6), we get

$$\begin{aligned} u(y) &\approx \sum_{j=1}^{m} a_j \phi_{c_j}(y) \\ &\approx \sum_{j=1}^{m} a_j \sum_{i=1}^{n} b_i \phi_{c_j}(x_i) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_j b_i \phi_{c_j}(x_i) \\ &\approx \sum_{i=1}^{n} b_i u(x_i). \end{aligned}$$

Hence, the numerical solution at the inner point y can be estimated by linear combination of this function at the boundary points.

2.2.2 The Laplace's equation with Neumann boundary conditions

We assume that at the source point c_j, the fundamental solution of the any points can be approximated by a linear combination of derivative of fundamental solution of the boundary points:

$$\phi_{c_j}(y) \approx \sum_{i=1}^{n} b_i \frac{\partial}{\partial \mathbf{n}} \phi_{c_j}(x_i).$$

Using approximation (6), we get

$$\begin{aligned} u(y) &\approx \sum_{j=1}^{m} a_j \phi_{c_j}(y) \\ &\approx \sum_{j=1}^{m} a_j \sum_{i=1}^{n} b_i \frac{\partial}{\partial \mathbf{n}} \phi_{c_j}(x_i) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_j b_i \frac{\partial}{\partial \mathbf{n}} \phi_{c_j}(x_i) \\ &\approx \sum_{i=1}^{n} b_i \frac{\partial}{\partial \mathbf{n}} u(x_i). \end{aligned}$$

Accordingly, the structure of neural network is the following:

- Training data input includes vectors $\left(\frac{\partial}{\partial \mathbf{n}}\phi_{c_j}(x_1), \cdots, \frac{\partial}{\partial \mathbf{n}}\phi_{c_j}(x_n)\right)$ where $j = 1, \cdots, m$.
- The output of training data is $\phi_{c_j}(y)$ where y is an inner point.

2.2.3 The Laplace's equation with mixed boundary conditions

Let us consider the mixed boundary value problem with the conditions in (10). The second approach is estimating the fundamental solution at an inner point by both fundamental solution and derivative of fundamental solution at boundary points

$$\phi_{c_j}(y) \approx \sum_{x_i \in \Gamma_1} b_i \phi_{c_j}(x_i) + \sum_{x_i' \in \Gamma_2} b_i' \frac{\partial}{\partial \mathbf{n}} \phi_{c_j}(x_i'). \tag{12}$$

Note that by the linearity of the original problems (with any kind of boundary conditions), there should be such parameters b_i, b_i'.

Using approximation (6) again, we get

$$u(y) \approx \sum_{x_i \in \Gamma_1} b_i u(x_i) + \sum_{x_i' \in \Gamma_2} b_i' \frac{\partial}{\partial \mathbf{n}} u(x_i').$$

Hence, the output of training data in the second method is always the fundamental solution $\phi_{c_j}(y)$ and the input data in mixed boundary problem is

$$\left(\phi_{c_j}(x_1), \cdots, \phi_{c_j}(x_n), \frac{\partial}{\partial \mathbf{n}}\phi_{c_j}(x_1'), \cdots, \frac{\partial}{\partial \mathbf{n}}\phi_{c_j}(x_n')\right),$$

where $x_1, \cdots, x_n$ are boundary points generated on Γ_1 and $x_1', \cdots, x_n'$ are generated on Γ_2.

3 Solving Helmholtz-typed equations

Let us consider the two types of boundary homogeneous Helmholtz-type equation, including the monotone type

$$\begin{cases} \Delta^2 u - k_1^2 u = 0 \quad \text{on } \Omega \\ u|_{\Gamma_1} = g \quad \text{and} \quad \frac{\partial u}{\partial \mathbf{n}}\big|_{\Gamma_2} = f. \end{cases} \tag{13}$$

and the oscillatory type

$$\begin{cases} \Delta^2 u + k_2^2 u = 0 \quad \text{on } \Omega \\ u|_{\Gamma_1} = g \quad \text{and} \quad \frac{\partial u}{\partial \mathbf{n}}\big|_{\Gamma_2} = f. \end{cases} \tag{14}$$

where $\Omega \subset \mathbb{R}^n$ is an open set and $\partial\Omega$ consists of two disjoint parts, Γ_1 and Γ_2, such that $\partial\Omega = \Gamma_1 \cup \Gamma_2$. The fundamental solution of the modified Helmholtz(13) and the original Helmholtz (14) is given by (resp.)

$$\phi_-(x,y) = \begin{cases} \frac{1}{2\pi} K_0(k_1\|x-y\|) & \text{in 2D case} \\ \dfrac{e^{-k_1\|x-y\|}}{4\pi\|x-y\|} & \text{in 3D case} \end{cases} \tag{15}$$

and

$$\phi_+(x,y) = \begin{cases} \frac{i}{4} H_0^{(1)}(k_2\|x-y\|) & \text{in 2D case} \\ \dfrac{e^{-ik_2\|x-y\|}}{4\pi\|x-y\|} & \text{in 3D case} \end{cases} \tag{16}$$

where K_0 is the modified Bessel function of the second kind of order zero and H_0^1 is Hankel function of the first (See Appendix ..). In practice, the function u is a real-valued function. Hence, we can assume the

fundamental solution of oscillatory typed problem is a real value function, namely

$$\phi_+(x,y) = \begin{cases} \frac{-1}{4} Y_0(k_2\|x-y\|) & \text{in 2D case} \\ \dfrac{\cos(-k_2\|x-y\|)}{4\pi\|x-y\|} & \text{in 3D case} \end{cases} \tag{17}$$

where Y_0 is the Bessel function of the second kind of order zero kind of order zero.
Using the method of fundamental solution (MFS), we estimate the solution as a data-dependent linear representation

$$u(x) \approx \sum_{j=1}^{m} a_j \phi_{c_j}(x) \tag{18}$$

where
From this, we construct the neural network design to find the above coefficient from
the Dirichlet input-output training data is: $(\phi_{c_1}(x_i), \cdots, \phi_{c_m}(x_i))$ and $g(x_i)$ where $x_i \in \Gamma_1$ and the Neumann input-output training data includes $\left(\frac{\partial}{\partial \mathbf{n}}\phi_{c_1}(x_i), \cdots, \frac{\partial}{\partial \mathbf{n}}\phi_{c_m}(x_i)\right)$ and $f_{\mathbf{n}}(x_i)$ where $x_i \in \Gamma_2$.
the output of training data in the second method is always the fundamental solution $\phi_{c_j}(y)$ and the input data in mixed boundary problem is

$$\left(\phi_{c_j}(x_1), \cdots, \phi_{c_j}(x_n), \frac{\partial}{\partial \mathbf{n}}\phi_{c_j}(x_1'), \cdots, \frac{\partial}{\partial \mathbf{n}}\phi_{c_j}(x_n')\right),$$

where $x_1, \cdots, x_n$ are boundary points generated on Γ_1 and $x_1', \cdots, x_n'$ are generated on Γ_2.

4 Convergence analysis

Katsurada and Okamoto ([9]), and Fairweather and Karageorghis ([5]) gave an error bound for the Dirichlet problem for the Laplace equation on the circle $(0, \rho)$, boundary function is analytical, solution is analytically harmonic continuable to the whole plane:

$$\|u - u_M\|_{L^\infty(\Omega)} \leq C \left(\frac{\rho}{R}\right)^M \tag{19}$$

where R is the radius of exterior circle. This shows that the MFS is exponentially convergent with respect to increasing M, or R.
Betcke ([2]) obtained the same results for Helmholtz operator $\Delta + k^2 I$ and Balakrishnan and Ramachandran ([1]); Barnett and Betcke ([2]); Bogomolny ([3]) and for the modified Helmholtz operator $\Delta - k^2 I$.
The estimate 19 was proved by Katsurada ([7], [8]) for boundary domain is a circle and an analytical Jordan curve, respectively.
In case u is not analytically continuable to the whole plane, but rather only up to an extension $B(0; r_0)$ Kitagawa ([10], [11]) proved:

$$\|u - u_M\|_{L^\infty(\Omega)} \leq \|u\|_{L^\infty(\partial B(0,r_0))} \left(\frac{2}{1-\frac{\rho}{R}}\right)\left[(1 + A(R,p))\left(\frac{\rho}{r_0}\right)^{M/3} + 4\left(\frac{\rho}{R}\right)^{M/3}\right]$$

where $A(R, \rho)$ is some constant between 1 and 2. The price to pay for this excellent exponential convergence is that the condition number of the coefficient matrix of the resulting MFS system of equations grows exponentially with respect to M.

5 Numerical experiments

In this section, we dig deep into how these methods work on specific problems, how to opt the position of collocation points, source points and how to tune parameters properly. In particular, we focus to the following questions.

- What is the optimal number (or rather: the ratio) of the boundary and outer points?
- What is the optimal distance of the outer points from the boundary?
- What is the optimal distribution of the boundary and outer points?
- Does the two approaches deliver similar accuracy?
- With an optimal choice of all parameters, which convergence rate can be achieved?
- What is the setup and the parameters in a neural network used in the computations?

5.1 Mixed boundary condition problem on Amoeba-like domain

Let consider the Laplace equation 3 on the Amoeba-like domain where the boundary points have the position of

$$\left(e^{\sin\theta}\sin^2(2\theta) + e^{\cos\theta}\cos^2(2\theta)\right)(\cos\theta, \sin\theta).$$

The boundary conditions are given by the Dirichlet boundary condition if $0 \leq \theta < \pi$ and the Neumann boundary condition if that $0 \leq \theta < 2\pi$ such that the analytical solution is $u(x,y) = \cos(x)\cosh(y) + \sin(x)\sinh(y)$. Figure 4 is obtained by running on 60 boundary points, 90 outer points, $\epsilon = 0.1$ and 10000 epochs. We also implemented the second method on this problem. Using 50 boundary points and 200 outer points, $\epsilon = 0.1$ with 90000 epochs, the result of some points is showed in Table 1.

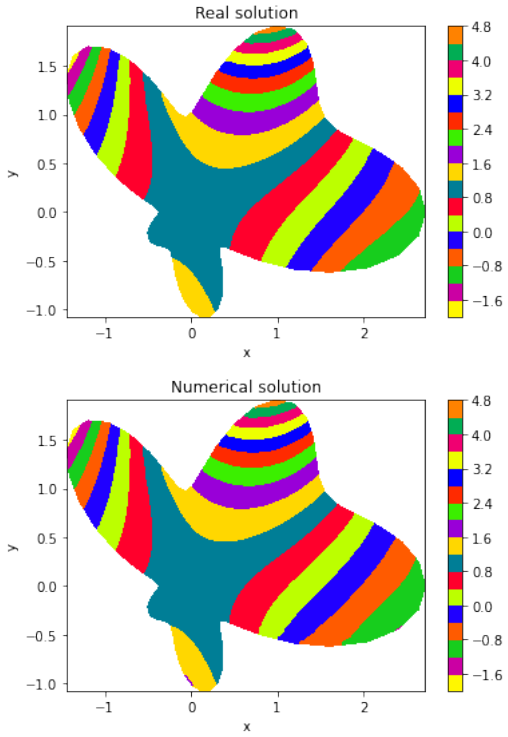

Figure 4: The analytical and numerical solution on Amoeba-like domain

Collocation points	$(0,0)$	$(1,1)$	$(-1,1)$
Analytical solution	1	1.8226277	-0.155167
Numerical solution	1.0056722	1.823955	-0.15932

Table 1: The numerical solution of some points on the mixed boundary problem using the second approach

Figure 5: The outer and boundary points for example 1

5.2 Dirichlet boundary condition on the unit square

Example 1. *In a unit square* $\Omega = \{x, y \mid 0 < x < 1, 0 < y < 1\}$ *let us consider the following equation*

$$\begin{cases} \Delta u = 0 & \text{for } (x,y) \in \Omega \\ u(x,0) = 0,\ u(x,1) = \sin(\pi x) & \text{for } 0 < x < 1 \\ u(0,y) = 0,\ u(1,y) = 0 & \text{for } 0 < y < 1. \end{cases} \tag{20}$$

The analytical solution of this problem is:

$$u(x,y) = \frac{1}{e^{\pi} - e^{-\pi}} \sin(\pi x) \left(e^{\pi y} - e^{-\pi y}\right).$$

Firstly, we implement the first method by choosing the position of outer points on a square which is obtained by a magnification of the unit square (see Figure 5). We define the magnification factor ϵ as the distance between two parallel sides of each square.

In Figure 6 (left figure), we use the following parameters: 32 boundary points, 8192 outer points, magnification factor: $\epsilon = 0.1$, learning rate $= 0.1$, epoch $=1000$, 1 hidden layer with size of 100. Loss function after training: 0.0003839250421151519.

The role of the number of boundary points and source points is important. In Table 2, we run on different values of n(the number of boundary points) and m (the number of boundary points) with $\epsilon = 0.2$. We observe that the more outer points we generate, the more accuracy we get for the Frobenius norm. The number of outer points should exceed the number of boundary points because it is easier to represent a few boundary points by a linear combination of a lot of fundamental solutions. If we choose more boundary points, the loss function is not vanished because of non-singularity when finding weight matrix, it leads to the predicted vector has same all entries. But also note that the solution we get after running a neural network is not unique, so we may get results far away from the true solution but have small losses. Luckily, in this example, the divergence between the two solutions are insignificant.

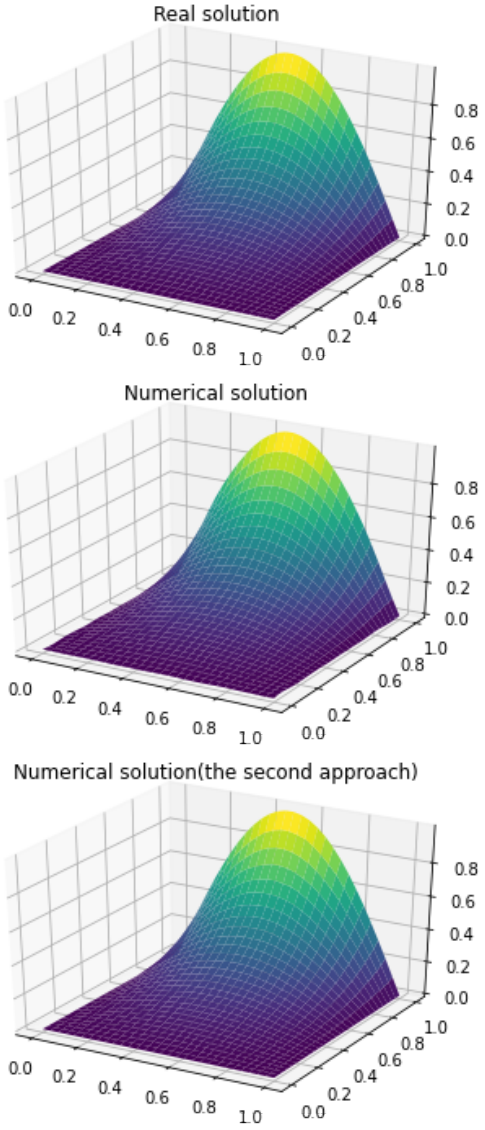

Figure 6: Analytical and Numerical solutions are solved by the first approach (left) and the second approach (right) of example 1

The number of boundary and outer points	Error (Frobenius norm)
$n = 16, m = 64$	0.2461622809658327
$n = 32, m = 64$	0.3182853004347026
$n = 32, m = 128$	0.21974143048835057
$n = 16, m = 128$	0.18014076148115063
$n = 16, m = 512$	0.10919759728510069
$n = 16, m = 2048$	0.07990843843706859
$n = 32, m = 8192$	0.04762830527750495

Table 2: The effect of the number of boundary points and outer points (the first approach)

The distance between two squares can change the error slightly, but the numerical solution is close to the analytical solution in general. According to the Figure 7 (left), ϵ should be lies between 0.05 and 0.1 to get optimal numerical solution. Increasing the number of epochs will also decrease the error (see Figure 7 -right)

Now we turn into the second approach. We want to plot the whole function in this example to compare it to the first method. A heuristic solution is forming the output of training data is a vector computing the fundamental solution at grid points for an outer point. However, in practice, we expect to get the value of every grid points precisely, not only the network in general. It requires the complexity of neural networks and time to train the model. It is an obstacle when tuning the parameters, so our technique is to train each point in grid point separately until it closes to the analytical solution at this point and then apply these parameters to all other points. Figure 6(right) illustrates the numerical solution of the second approach with 320 boundary points, 60 source points and $\epsilon = 0.025$.

However, the purpose of using both approaches is not to estimate every point on the domain. It measures specified source points without using discretization on this domain (i.e.the mesh-free method). In Table 3, we use different values of (the number of boundary points, the number of outer points). It shows that the second method performs moderately well even though the number of boundary points is smaller or larger or equal to the number of outer points.

Similar to the first method, the loss decreases if the auxiliary boundary is closer to the frontier(or ϵ is smaller) which can be seen in Table 4.

5.3 Different neural networks and comparisons

Table 6 and 7 illustrates the results after implementing in different set up of neural networks for both methods. The more complex neural networks we use in the first approach, the more accuracy we get with the

Collocation points	True solution	$(100, 100)$	$(400, 400)$	$(1000, 1000)$	$(1000, 100)$	$(100, 1000)$
$(0.2, 0.2)$	0.034124	0.03115696	0.0338244	0.03507766	0.03429693	0.03419725
$(0.5, 0.5)$	0.199268	0.2008082	0.20086792	0.20041794	0.19841382	0.19588123
$(0.7, 0.7)$	0.311947	0.3144576	0.3115651	0.3125463	0.3116747	0.3061932
$(0.2, 0.7)$	0.226643	0.22474924	0.22370611	0.22674346	0.22672111	0.2225104
$(0.7, 0.2)$	0.046969	0.04574379	0.04649694	0.04676051	0.04662241	0.04642019

Table 3: The different choices of the number of boundary points and outer points, $\epsilon = 0.15$(the second approach)

Collocation points	True solution	$\epsilon = 0.1$	$\epsilon = 0.15$	$\epsilon = 0.2$	$\epsilon = 0.025$
$(x, y) = (0.2, 0.2)$	0.034124	0.0347167	0.03429693	0.03295259	0.03433963
$(x, y) = (0.5, 0.5)$	0.199268	0.19874202	0.19841382	0.2012808	0.19995686
$(x, y) = (0.7, 0.7)$	0.311947	0.3124724	0.3116747	0.3126797	0.3119331
$(x, y) = (0.2, 0.7)$	0.226643	0.22637717	0.22672111	0.22672312	0.2267626
$(x, y) = (0.7, 0.2)$	0.046969	0.04597505	0.04662241	0.04752169	0.04735678

Table 4: The different choices of magnification factor (the second approach), $n = 1000, m = 100$

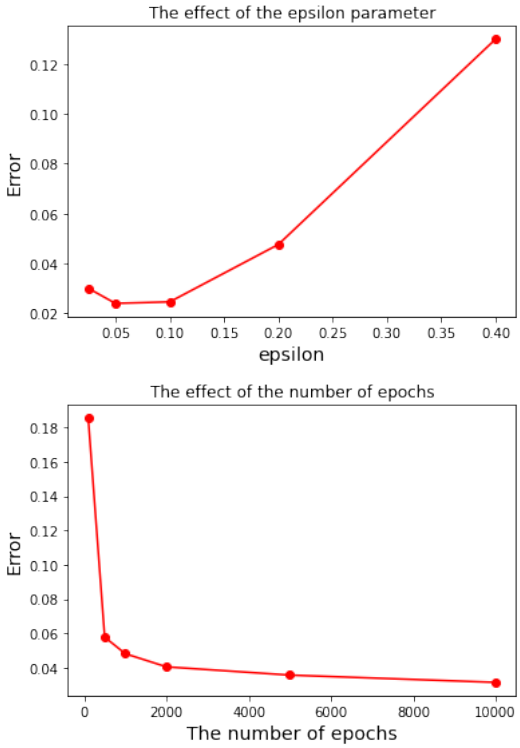

Figure 7: The effect of ϵ and the number of epochs(the first approach)

same number of epochs. But also note that if we use complex neural networks, there are more unknown parameters that cost the computation time. For example 1, we recommend using the neural network with one hidden layer of "small" size. The second approach, otherwise, does not affect by the structure of neural networks. We can achieve a reasonable good numerical solution using a simple no hidden layer network and increasing the number of epochs.

To sum up, both methods are required a large enough number of outer points because both use the approximation (6). In practice, if the number of exterior points is more than 30 and the ratio between the number of boundary and outer points is small (<1), we will get a "good" numerical solution. However, the second approach also necessitates a large enough number of boundary points to guarantee that the approximation (11) or (12) has a small error. It avoids the error term from approximation (6) directly and works for any ratio between the number of boundary and outer points. Hence, the second method is applicable for the problem which generating the equidistant outer points ineffectively, while the first method is an incredible estimation when the auxiliary domain is formed effectively.

In a nutshell, with the optimal choice of all parameters, both methods operate accurately with low error on the "smooth" boundary curve (see Table 5).

5.4 The Helmholtz problem on the sphere

In the first example, the three-dimensional eigenvalue problem for the Laplace operator problem is selected to divulge the competence of the proposed MFS. The monotone typed Helmholtz is given with Dirichlet

Position of inner point	Analytical solution	Approach 2	Approach 1
$(x, y) = (0.2, 0.2)$	0.034124989219717516	0.03417236	0.03415351
$(x, y) = (0.5, 0.5)$	0.19926840766919335	0.1992962	0.19928508
$(x, y) = (0.7, 0.7)$	0.311947830115096	0.31381178	0.31188422
$(x, y) = (0.2, 0.7)$	0.22664336509760868	0.22663322	0.22663559
$(x, y) = (0.7, 0.2)$	0.04696901819829289	0.04670608	0.04691111

Table 5: The approximation of some interior points using both methods

Neural network	Error
Linear Regression, no hidden layer, epochs = 1000	0.048206
Neural network, 1 hidden layer of size 100, epochs = 1000	0.026957
Neural network, 1 hidden layer of size 500, epochs = 1000	0.026286
Neural network, 2 hidden layers of size 100, epochs = 1000	0.022334
CNN with 2 fully connected layers, epochs = 10000	0.017556

Table 6: Different neural network structures for Example 1.

boundary conditions as follow

$$\begin{cases} \Delta^2 u - 3u = 0 & \text{on } \Omega \\ u(x, y, z) = \sinh(x + y + z) & \text{on } \partial\Omega \end{cases} \tag{21}$$

where $\Omega = \{(x, y, z) | x^2 + y^2 + z^2 < 1\}$.
Before injecting the training data into neural network design, we consider the distribution on the boundary and auxiliary boundary surface such that points are evenly distributed. We list four common distributions below.

a) The standard equiangular spherical coordinates distribution forming N_1 lines of latitude and N_2 lines of longitude.

b) There are $6N^2 + 6N + 2$ points generated in the sphere which has positions

$$\left(\sin\left(\frac{n\pi}{2N}\right)\sin\left(\frac{k\pi}{3n}\right), \sin\left(\frac{n\pi}{2N}\right)\cos\left(\frac{k\pi}{3n}\right), \cos\left(\frac{n\pi}{2N}\right)\right)$$

where $k = 1, \cdots, 6n$ and $n = 1, \cdots N$.

c) The Fibonacci lattice is expressed as a sequence of N points with coordinates

$$\sin(k\pi(3 - \sqrt{5}))\sqrt{1 - \left(1 - \left(\frac{k}{N-1}\right)^2\right)^2},$$
$$1 - \left(\frac{k}{N-1}\right)^2,$$
$$\cos(k\pi(3 - \sqrt{5}))\sqrt{1 - \left(1 - \left(\frac{k}{N-1}\right)^2\right)^2}$$

where k runs from 0 to $N - 1$

d) (Golden spiral method)

$$\left(1 - 2\frac{k}{N-1}\right)\sin(k\pi(1 + \sqrt{5})),$$
$$\left(1 - 2\frac{k}{N-1}\right)\cos(k\pi(1 + \sqrt{5})),$$
$$\pm\sqrt{1 - \left(1 - 2\frac{k}{N-1}\right)^2}$$

where $k = 0, \cdots, N - 1$

For comparison, we generate $256 = 16 \times 16$ boundary points and $1444 = 38 \times 38$ outer points for the standard equiangular coordinates. For the (b) distribution, 254 ($N = 6$) points was created in boundary and 1442($N = 15$) points in the pseudo-boundary. We both set 255 boundary points and 1443 outer points on Fibonacci lattice and spiral distribution.

References

[1] Balakrishnan, K. and Ramachandran, P. A. *The method of fundamental solutions for linear diffusion-reaction equations.* Math. Comput. Modelling, 31:221-237. 2000.

[2] Barnett, A. H. and Betcke, T. *Stability and convergence of the method of fundamental solutions for Helmholtz problems on analytic domains* J. Comput. Phys., 227:7003-7026, 2008

[3] Bogomolny, A., *Fundamental solutions method for elliptic boundary value problems.* SIAM J. Numer. Anal., 22, 644-659, 1985.

[4] Cheng, A.H.D and Hong, Y.: *An overview of the method of fundamental solutions-Solvability,*

Neural network	$(0,0)$	$(1,1)$	$(2,0)$	$(-1,1)$
Linear Regression, no hidden layer, epochs = 60000	1.001674	1.822747	-0.4326400	-0.1545439
Neural network, 1 hidden layer of size 100, epochs = 90000	0.998864	1.820469	-0.4302049	-0.1552322
Neural network, 2 hidden layers of size 100, epochs = 90000	1.000284	1.821235	-0.4321503	-0.1545273
CNN with 2 fully connected layers, epochs = 300000	1.002132	1.820958	-0.4312253	-0.1556389
Analytical solution	1	1.8226277	-0.4161468	-0.1551676

Table 7: Different neural network structures for Amoeba-like domain problem (Section 4.5)

Collocation points	Error(dis.1)	Error(dis.2)	Error(dis.3)	Error(dis.4)
$(0.6, 0.6, 0.1)$	-0.00768697	0.00203109	0.00108302	-1.1682*10^{-5}
$(0.3, 0.3, 0.3)$	-0.02142298	0.00585616	0.00275302	-0.00031912
$(0, 0, 0)$	-0.02814355	0.00775653	0.00363088	-0.000422
$(-0.6, 0.7, -0.3)$	-0.00023824	0.00023739	-0.00033593	-0.00058134

Table 8: Different nodal distribution leads to different error distribution

uniqueness, convergence, and stability, Engineering Analysis with Boundary Elements 120, pp. 118-152, 2020.

[5] Fairweather, G. and Karageorghis, A. *The method of fundamental solutions for elliptic boundary value problems.* Adv. Comput. Math., 9:69-95, 1998

[6] Jaeyoun O., Huiqing Z., Zhuojia F., *An adaptive method of fundamental solutions for solving the Laplace equation*, Computers and Mathematics with Applications, 77(7), 1828-1840, 2019.

[7] Katsurada, M. *A mathematical study of the charge simulation method II.* J. Fac. Sci. Univ. Tokyo Sect. IA Math., 36:135-162. 1989.

[8] Katsurada, M. *Asymptotic error analysis of the charge simulation method in a Jordan region with an analytic boundary.* J. Fac. Sci., Univ. of Tokyo, Sect. 1A, Math. 37, 635-657, 1990

[9] Katsurada, M. and Okamoto, H. *The collocation points of the fundamental solution method for the potential problem.* Comput. Math. Appl., 31:123-137. 1996.

[10] Kitagawa, T. (1988). *On the numerical stability of the method of fundamental solution applied to the Dirichlet problem.* Japan. J. Appl. Math. Appl., 5:123- 133.

[11] Kitagawa, T. (1991). *Asymptotic stability of the fundamental solution method.* J. Comput. Appl. Math., 38:263-269

[12] Kupradze, V. D. and Aleksidze, M. A., *The method of functional equations for the approximate solution of certain boundary value problems.* Z. Vycho. Mat., 1964, 4, 633-715.

[13] Lu, L., Meng, X., Mao, Z. and Karniadakis, G.E. *Deepxde: A deep learning library for solving differential equations.* SIAM Review 63(1), 208-228, 2021.

[14] Meng, X. and Karniadakis, G.E. *A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems.* Journal of Computational Physics 401, 109020, 2019.

[15] Özbay, A., Hamzehloo, A., Laizet, S., Tzirakis, P., Rizos, G., Schuller, B. *Poisson CNN: Convolutional neural networks for the solution of the Poisson equation on a Cartesian mesh.* Data-Centric Engineering, 2, E6. doi:10.1017/dce.2021.7, 2021.

[16] Sauter, S. and Schwab, Ch. (2010). Boundary Element Methods. Springer Series in Computational Mathematics, vol. 39, Springer, 2010.

[17] Smyrlis, Y.-S. and Karageorghis, A. *Numerical analysis of the MFS for certain harmonic problems.* Mathematical Modelling and Numerical Analysis, 38, 495-517, 2004.