A hybrid neural-network and finite-difference method for solving Poisson equation with jump discontinuities on interfaces

Wei-Fan Hu\(^1\(^4\), Te-Sheng Lin\(^2\(^4\), Yu-Hau Tseng\(^3\), and Ming-Chih Lai\(^2\)

\(^1\)Department of Mathematics, National Central University, Taoyuan 32001, Taiwan
\(^2\)Department of Applied Mathematics, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan
\(^3\)Department of Applied Mathematics, National University of Kaohsiung, Kaohsiung 81148, Taiwan
\(^4\)National Center for Theoretical Sciences, National Taiwan University, Taipei 10617, Taiwan

October 12, 2022

Abstract

In this work, a new hybrid neural-network and finite-difference method is developed for solving Poisson equation in a regular domain with jump discontinuities on an embedded irregular interface. Since the solution has low regularity across the interface, when applying finite difference discretization to this problem, an additional treatment accounting for the jump discontinuities must be employed at grid points near the interface. Here, we aim to elevate such an extra effort to ease our implementation. The key idea is to decompose the solution into two parts: singular (non-smooth) and regular (smooth) parts. The neural network learning machinery incorporating given jump conditions finds the singular solution, while the standard finite difference method is used to obtain the regular solution with associated boundary conditions. Regardless of the interface geometry, these two tasks only require a supervised learning task of function approximation and a fast direct solver of the Poisson equation, making the hybrid method easy to implement and efficient. The two- and three-dimensional numerical results show that the present hybrid method preserves second-order accuracy for the solution and its derivatives, and it is comparable with the traditional immersed interface method in the literature.

keywords: Neural networks, sharp interface method, fast direct solver, elliptic interface problem
1 Introduction

In this note, we aim to solve a $d$-dimensional ($d = 2$ or $3$) elliptic interface problem defined in a regular domain $\Omega \subset \mathbb{R}^d$, which is separated by an embedded interface $\Gamma$ such that the subdomains inside and outside the interface are denoted by $\Omega^-$ and $\Omega^+$, respectively. Along the interface $\Gamma$, there exists jump discontinuities that the solution must be satisfied. With the associated boundary condition, the problem takes the form

$$\Delta u(x) = f(x), \quad x \in \Omega^- \cup \Omega^+, \tag{1}$$

$$[u(x)] = \gamma(x), \quad [\partial_n u(x)] = \rho(x), \quad x \in \Gamma, \tag{2}$$

$$u(x) = g(x), \quad x \in \partial \Omega. \tag{3}$$

Here, the jump $[\cdot]$ indicates the quantity approaching from $\Omega^+$ side minus the one from $\Omega^-$ side; the shorthand $\partial_n u$ represents the normal derivative $\nabla u \cdot n$ in which $n$ is the normal vector pointing from $\Omega^-$ to $\Omega^+$. Notice that, here the underlying differential equation is subject to the Dirichlet-type boundary condition for illustration purpose, while other types of boundary condition (Neumann or Robin) will not change the main ingredients presented here. Since the Poisson equation is considered in Eq. (1), we simply call the above problem as the Poisson interface problem hereafter.

As seen from Eqs. (1)-(3), the solution and its partial derivatives have jumps across the interface. So, when applying the finite difference discretization to this problem, an additional treatment accounting for those jump discontinuities must be employed at the grid points near the interface. Over the past few decades, different discretization methodologies have been successfully developed to capture those jump conditions sharply or to improve the overall numerical accuracy, such as the immersed interface method (IIM) \cite{1,2,3,4}, ghost fluid method (GFM) \cite{5,6}, Voronoi interface method \cite{7}, to name a few. Other approaches of solving interface problems using finite element or finite volume methods can be found in \cite{8} and the references therein.

On the other hand, much attention has recently been paid to applying deep neural networks (DNNs) to solve elliptic interface problems, rather than using traditional numerical methods to solve such problems. Despite the success of the two mainstream deep learning approaches (Physics-Informed Neural Networks (PINNs) \cite{9} and the deep Ritz method \cite{10}) in solving partial differential equations with smooth solutions, learning methods based on these two frameworks for solving elliptic interface problems with jump discontinuities remain to be improved. The main and intrinsic difficulty may be attributed to the fact that the usual activation functions used in DNNs are generally smooth; thus, DNN function approximators seem to be incapable of representing discontinuous functions. To approximate such discontinuous solutions (or functions), piecewise DNNs \cite{11}, interfaced neural networks \cite{12}, and deep unfitted Nitsche method \cite{13} were proposed to tackle such elliptic interface problems. In those methods, multiple independent networks need to be established and linked with each other by imposing the jump conditions. The resulting
prediction errors in their test examples reach the magnitude $O(10^{-3})$ to $O(10^{-4})$ in relative $L^2$ norm. Moreover, training these DNN models comes at the cost of having to train a separate neural network in each subdomain independently. Until very recently, the authors of this note proposed a Discontinuity Capturing Shallow Neural Network (DCSNN) [14] that allows a single network to represent piecewise smooth functions via a simple augmentation technique. The network is completely shallow (one hidden layer), so the resulting number of trainable parameters is moderate (only a few hundred) and attains prediction accuracy as low as $O(10^{-7})$ in relative $L^2$ norm for all tests in both 2D and 3D elliptic interface problems. Note that the above neural network methods are all completely mesh-free, but their convergence still requires further investigation.

In this work, we propose a novel hybrid method that combines neural network learning machinery and traditional finite difference methods to solve the Poisson interface problem \((\ref{eq:poisson})-\text{(3)}\). The entire computation only comprises supervised learning of function approximation and a fast direct solver of the Poisson equation, which can be easily and directly implemented regardless of interface geometry. The rest of the note is organized as follows. In Section 2, we present the methodology and address some features, including error analysis of the hybrid scheme. Numerical results are given in Section 3, followed by some concluding remarks and future works in Section 4.

2 Hybrid neural-network and finite-difference methodology

By taking advantage of the machine learning techniques, our goal is to design an easy-to-implement fast solver for the Poisson interface problem \((\ref{eq:poisson})-(\text{3})\). To this end, we propose a novel hybrid method that exploits the advantages of neural network learning machinery and traditional finite difference method. As we can see from the jump conditions in \((\ref{eq:jump})\), the solution $u$ is non-smooth across the interface. Thus, we start by decomposing the solution into

$$u(x) = v(x) + w(x), \quad (4)$$

where $v$ and $w$ represent the singular (non-smooth) and regular (smooth) parts of $u$, respectively. More precisely, we require $w$ to be fairly smooth over the entire domain $\Omega$ so that the zero jumps $[w] = [\partial_n w] = [\Delta w] = 0$ on the interface are all satisfied. Now the singular solution $v$ is responsible for having all discontinuities across the interface; hereby, we construct this discontinuous function by assuming

$$v(x) = \begin{cases} V(x) & x \in \Omega^-, \\ 0 & x \in \Omega^+, \end{cases} \quad (5)$$

where $V$ is a smooth function to be found. Using the above definition and plugging the decomposition \((\ref{eq:decomp})\) into the jump conditions \((\ref{eq:jump})\) and differential equation \((\ref{eq:poisson})\), the unknown
function $\mathcal{V}$ must satisfy the following constraints along the interface:

$$
\mathcal{V}(x) = -\gamma(x), \quad \partial_n \mathcal{V}(x) = -\rho(x), \quad \Delta \mathcal{V}(x) = -[f(x)], \quad x \in \Gamma.
$$

(6)

Note that this function is not unique, in the sense that there exist infinitely many functions defined in the domain $\Omega$ that satisfy the restrictions (6). To find $\mathcal{V}$, we leverage the power of function expressibility of neural networks. Here, we simply employ a shallow (one hidden layer) fully-connected feedforward neural network to approximate $\mathcal{V}$, and learn the function via the supervised learning model. Specifically, given a dataset with $M$ training data points \( \{x_i^\Gamma \}_{i=1}^M \) and the target outputs $\gamma(x_i^\Gamma)$, $\rho(x_i^\Gamma)$ and $\lfloor f(x_i^\Gamma) \rfloor$, we find $\mathcal{V}(x)$ by minimizing the following mean squared error loss consisting of the residuals of conditions in Eq. (6):

$$
\text{Loss}(p) = \frac{1}{M} \sum_{i=1}^{M} \left[ (\mathcal{V}(x_i^\Gamma; p) + \gamma(x_i^\Gamma))^2 + (\partial_n \mathcal{V}(x_i^\Gamma; p) + \rho(x_i^\Gamma))^2 + (\Delta \mathcal{V}(x_i^\Gamma; p) + \lfloor f(x_i^\Gamma) \rfloor)^2 \right],
$$

(7)

where $p$ collects all trainable parameters (weights and biases) in the network. To train the above loss model, we adopted the Levenberg-Marquardt (LM) method \[15\], a full-batch optimization algorithm which is particularly efficient for least squares losses. We should also mention that the partial derivatives of the target function $\mathcal{V}(x)$ in the loss function (7) can be computed easily by automatic differentiation \[16\].

Once $\mathcal{V}$ is available, we can obtain $w$ by solving the following Poisson equation:

$$
\Delta w(x) = \Delta u(x) - \Delta v(x) = \begin{cases} 
   f(x) - \Delta \mathcal{V}(x) & x \in \Omega^-,
   \\
   f(x) & x \in \Omega^+,
\end{cases}
$$

(8)

$$
w(x) = g(x), \quad x \in \partial \Omega.
$$

(9)

Notice that, using the last jump constraint for $\mathcal{V}$ in Eq. (6), one can immediately see that the right-hand side function of (8) is continuous on the entire domain. Moreover, $w$ is accompanied with exactly the same boundary condition as the solution $u$ since $v$ is designed to vanish in $\Omega^+$. As a result, $w$ has sufficient regularity (recall the requirement $[w] = [\partial_n w] = [\Delta w] = 0$) and satisfies the Poisson equation in the regular domain $\Omega$ that can be simply and efficiently solved using the well-developed public software Fishpack \[17\] or any fast Poisson solver. Of course, other traditional numerical methods, such as finite volume or finite element methods, can also be used to find the solution $w$. Additionally, we remark that both the right-hand sides of Eq. (5) and the Poisson equation (8) involve the categorization of $x$ in $\Omega^-$ or $\Omega^+$, which can be easily done with the assistance of a level set function for which the zero level set represents the interface $\Gamma$.

Let us summarize the proposed hybrid neural-network and finite-difference method for solving the Poisson interface problem (1)-(3) as follows.
Step 1 With given training dataset and sufficient number of neurons (or trainable parameters) used in the network, find the neural network function $V$ by minimizing the loss function (7) using the LM optimizer. Compute $V$ at those finite difference discretization grid points in $\Omega^-$ and then obtain the singular part of the solution, $v$, at grid points using Eq. (5).

Step 2 Evaluate $\Delta V$ at the grid points in $\Omega^-$, and solve the Poisson equation (8) by discretizing the Laplace operator using the standard five-point Laplacian and applying a fast direct solver to obtain the regular part of the solution, $w$, at those grid points.

Step 3 Recover the numerical solution $u = v + w$ at the grid points.

We conclude this section by introducing several features of the proposed method, as follows. (i) One can immediately deduce that the source of numerical error comes from the network approximation (optimization and network approximation error) and the finite difference approximation (local truncation error). The solution accuracy clearly depends on these two approximations. (ii) It is straightforward to implement the present method when multiple embedded interfaces are considered. (iii) With only one time training, the obtained network function $V$ is defined in a continuous sense so that it can be used in Step 2 for any grid resolutions. (iv) The advantage of traditional grid-based methods is that the boundary conditions are exactly satisfied. The present hybrid method shares the same advantage thanks to the design of Eq. (5). In contrast, most modern deep learning approaches (e.g., PINNs [9], DCSNN [14], and shallow Ritz method [18]) adopt a penalty term in the loss function to enforce the boundary condition, leading to an inevitable prediction error along the domain boundary. (v) The right-hand side function of the Poisson equation for $w$ (see Eq. (8)) is continuous, but, in general, has discontinuities in its derivatives across the interface. Under the finite difference discretization in Step 2, the local truncation error gives first-order accuracy for grid points adjacent to the interface. This can be easily derived by $[w] = [\partial_n w] = [\Delta w] = 0$ on the interface as described in [2]. However, the present numerical evidence shows that the overall accuracy for solving $w$ is still second-order. (vi) The proposed hybrid algorithm is easy to implement and efficient. It comprises a supervised learning task (for $V$) and a fast direct Poisson solver (for $w$), and there are already many well-developed and efficient packages for both tasks.

3 Numerical results

In this section, we check the accuracy of the proposed method by performing two numerical tests, including solving two- and three-dimensional Poisson interface problems. In each test, the neural net function $V$ is simply represented via a shallow network with sigmoid activation function, in which only a single hidden layer is employed. Thanks to the shallow
network structure, it only needs to train a moderate number of parameters (a few hundred parameters used throughout all numerical examples), so learning this network function is efficient and can be done in seconds on iMac (2021). Since all the computational domains considered in the following problems are regular (square in 2D and cube in 3D), to solve the regular part \( w \) we set up a uniform Cartesian grid layout with the same mesh size \( h \) in each spatial direction.

**Example 1** We start by solving a two-dimensional Poisson interface problem and compare the results with the ones obtained by the 2D IIM [2]. The problem is defined in the square domain \( \Omega = [-1,1]^2 \) in which the embedded interface is an ellipse given by \( \Gamma : (x/0.8)^2 + (y/0.2)^2 = 1 \). The exact solution is chosen as

\[
ue(x, y) = \begin{cases} 
\exp(x) \cos(y) & \text{if } (x, y) \in \Omega^-, \\
\exp(x^2) \cos(y) & \text{if } (x, y) \in \Omega^+. 
\end{cases}
\]

so the corresponding right-hand side \( f(x, y) \), and the jump information \( \gamma(x, y), \rho(x, y) \) and \( [f(x, y)] \) used in the loss function can be calculated accordingly. In this example, the network for \( V(x, y) \) is equipped with 40 neurons in the hidden layer and is trained using 200 randomly sampled training points on the interface \( \Gamma \).

In the left panel of Fig. 1 we report the mesh refinement study for maximum norm error \( \| u - u_e \|_\infty \) as a function of mesh size \( h \), where \( u \) denotes the numerical solution. One can see that the results obtained by the present hybrid method (solid blue line with circular markers) and IIM (solid red line with triangular markers) are almost equally well, and both achieve second-order convergence rate. We then use the computed solution to find \( \nabla u = (\partial_x u, \partial_y u) \) simply by applying standard central difference for the regular part \( w \) and auto differentiation for the network solution part \( v \). As can be seen in the right panel of Fig. 1 the gradient of numerical solution attains second-order convergence too.

As discussed in the previous section, the induced numerical error comes from both neural network approximation and finite difference approximation. Although not shown here, the final loss value in (7) is about \( 10^{-13} \sim 10^{-14} \), which leads to the predictive accuracy of the target function \( V \) and its Laplacian are of magnitude \( 10^{-6} \sim 10^{-7} \) roughly. Thus, we can conclude that the error is mainly dominated by the second-order finite difference approximation error when \( h^2 \gtrsim 10^{-6} \). To verify our error estimation, we have run more refining numerical tests with \( h^2 < 10^{-6} \). As expected, the resulting error cannot be reduced when refining the mesh width \( h \) with higher resolutions (not shown here).

**Example 2** Next, we proceed to the three-dimensional Poisson interface problem, in which the interface is an ellipsoid \( \Gamma : (x/0.7)^2 + (y/0.5)^2 + (z/0.3)^2 = 1 \), embedded in the regular cube \( \Omega = [-1,1]^3 \). The exact solution is given by

\[
u_e(x, y, z) = \begin{cases} 
\exp(x + y + z) & \text{if } (x, y, z) \in \Omega^-, \\
\sin(x) \sin(y) \sin(z) & \text{if } (x, y, z) \in \Omega^+. 
\end{cases}
\]
Again, one can obtain \( f, \gamma, \rho, \) and \([f]\) accordingly. We use the same shallow network structure as in the previous example, i.e., we set 40 neurons in one hidden layer and 200 training points on the interface \( \Gamma \) to learn \( V(x, y, z) \). Fig. 2 shows the mesh refinement results for the present method (solid blue line with circular markers) and 3D IIM solver developed in [4] (solid red line with triangular markers), as well as the maximum norm errors for the numerical gradient. Again, one can clearly see that the results of the present hybrid method and IIM are almost identical, and second-order accuracy is achieved for both the numerical solution and its gradient. It is important to point out that, the implementation for learning \( V \) and solving \( w \) in 3D problems is indeed straightforward as in 2D. In contrast, calculating the extra correction terms (incorporating all jump information) in the IIM implementation can be quite tedious in 3D problems, especially when the interface geometry is complex.

4 Conclusion and future work

In this note, we propose a new class of numerical methods to solve an elliptic interface problem whose solution and derivatives are known to have jump discontinuities across an interface. The crucial idea is to decompose the solution into singular (non-smooth) and regular (smooth) parts. The singular part is formed by a neural network representation found by using supervised learning machinery that incorporates all given jump information into the loss function. The regular part, however, is a solution to Poisson equation which can be obtained efficiently by fast direct solver based on finite difference discretization. The numerical experiments for 2D and 3D Poisson interface problems show that the proposed
hybrid neural-network and finite-difference method can achieve second-order accuracy for the solution and its derivatives. Although all illustrated examples consider a single embedded interface only, it is straightforward to implement the hybrid method with multiple interfaces.

As an application, the present hybrid method readily serves as a fast solver for Poisson interface problems involved in the projection step of Navier-Stokes flow problems. Our future work aims to extend the present methodology for solving variable-coefficient elliptic interface problems in regular or even irregular domains.

Acknowledgement

W.-F. Hu, T.-S. Lin, Y.-H. Tseng, and M.-C. Lai acknowledge the supports by National Science and Technology Council, Taiwan, under the research grants 111-2115-M-008-009-MY3, 111-2628-M-A49-008-MY4, 111-2115-M-390-002, and 110-2115-M-A49-011-MY3, respectively. W.-F. Hu and T.-S. Lin also acknowledge the supports by National Center for Theoretical Sciences, Taiwan.

References

[1] R. J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal., 31 (1994) 1019–1044.
[2] M.-C. Lai and H.-C. Tseng, A simple implementation of the immersed interface methods for Stokes flows with singular forces, Computers & Fluids, 37 (2008) 99–106.

[3] W.-F. Hu, M.-C. Lai, Y.-N. Young, A hybrid immersed boundary and immersed interface method for electrohydrodynamic simulations, J. Comput. Phys., 282 (2015) 47–61.

[4] S.-H. Hsu, W.-F. Hu, M.-C. Lai, A coupled immersed interface and grid based particle method for three-dimensional electrohydrodynamic simulations, J. Comput. Phys., 398 (2019) 108903.

[5] X.-D. Liu, R. P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson’s equation on irregular domains, J. Comput. Phys., 160 (2000) 151–178.

[6] R. Egan, F. Gibou, xGFM: Recovering convergence of fluxes in the ghost fluid method, J. Comput. Phys., 409 (2020) 109351.

[7] A. Guittet, M. Lepilliez, S. Tanguy, F. Gibou, Solving elliptic problems with discontinuities on irregular domains—the Voronoi interface method, J. Comput. Phys., 298 (2015) 747–765.

[8] Z. Li, K. Ito, The Immersed Interface Method, SIAM, 2006.

[9] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys., 378 (2019) 686–707.

[10] W. E., B. Yu, The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat., 6 (2018) 1–12.

[11] C. He, X. Hu, L. Mu, A mesh-free method using piecewise deep neural network for elliptic interface problems, J. Comput. Appl. Math., 412 (2022) 114358.

[12] S. Wu, B. Lu, INN: Interfaced neural networks as an accessible meshless approach for solving interface PDE problems, J. Comput. Phys., 470 (2022) 111588.

[13] H. Guo, X. Yang, Deep unfitted Nitsche method for elliptic interface problems, Commun. Comput. Phys., 31 (2022) 1162–1179.

[14] W.-F. Hu, T.-S. Lin, M.-C. Lai, A discontinuity capturing shallow neural network for elliptic interface problems, J. Comput. Phys., 469 (2022) 111576.

[15] D. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, SIAM J. Appl. Math., 11 (1963) 431–441.
[16] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res., 18 (2018) 1–43.

[17] J. Adams, P. Swarztrauber, R. Sweet. Fishpack - a package of fortran subprograms for the solution of separable elliptic partial differential equations [online] (1980).

[18] M.-C. Lai, C.-C. Chang, W.-S. Lin, W.-F. Hu, T.-S. Lin, A shallow Ritz method for elliptic problems with singular sources, J. Comput. Phys., 469 (2022) 111547.