An Extended Deep Learning Method for the Navier Equation

Amattouch Mohamed Ridouan

***Abstract*—This article deals with the Navier equation for simulating the elastic behavior of solids. We propose to solve this equation using a combination of domain decomposition method and meshless method trained with artificial neural networks. The domain decomposition technique, which is employed to solve the Navier partial differential equation, is the Schwarz wave relaxation method. This method involves a parallel implementation of the meshless method and will be trained using a specific neural network approach. Finally, we present several numerical test cases to validate the effectiveness of our methods.**

***Index Terms*—Navier equations, schwarz domain decomposition method, Artificial intelligence methods.**

I. Introduction

Linear elasticity plays a crucial role in modeling the deformation behavior of materials under mechanical loads. In the absence of body forces, the vector equation of equilibrium, often referred to as the Navier equations, governs the displacement field. Specifically, for the displacement vector u, the Navier equations are given by:

$$\begin{cases} div(\nabla \mathbf{u}) + \frac{1}{1-2\nu}\Delta \mathbf{u} &= 0 \\ + \, Boundary \, Conditions \end{cases} \tag{1}$$

Here, $\mathbf{u}$ represents the displacement vector, and ν denotes the Poisson ratio. In practical scenarios, accurately computing the displacement field and integrating the deformation and constraint fields is often more suitable. To address these equations, a widely adopted approach is the finite element method, which has been in elasticity models, by a number of studies([1], [2], [3] and [4]).

While it is effective for simple geometries like cylinders and tubes, finite element methods encounter challenges in complex geometries, requiring fine mesh discretization for convergence. However, memory processing limitations hinder their application in such cases. Spectral methods (e.g., [5], [6] and [7], [8]) offer an alternative, that offers increased accuracy for certain scenarios. However, there are challenges associated with these methods in selecting appropriate collocation points or basis functions for complex domains.

A novel attempt to address domains with intricate geometries is attained through meshless methods. These methods, such as meshless schemes [9], [10], [11], [22], employ a distribution of nodes to approximate solutions for partial differential equations [23]. Meshless methods have been studied by a number of scholars; in fact, the meshless method has the advantage that its precision of interpolation is not affected substantially by the distribution of the nodal [23]. Unlike finite element and volume methods, they exclusively rely on node distributions. However, handling boundary conditions remains a challenge. As a solution, [12] proposed converting the boundary problem into an optimization problem, leveraging metaheuristic or evolutionary algorithms to find optimal solutions.

Recent advancements have seen the application of artificial neural networks, such as the Physics-Informed Neural Network (PINN) [13], [14], [21], [24], to solve PDEs. This approach was further extended in [15], incorporating domain decomposition methods to expedite convergence. Despite their promise, these methods often demand an extensive number of collocation points, leading to slow and inefficient convergence.

In this article, we propose a methodology to alleviate the data burden associated with collocation points while solving the Navier equations. Our approach formulates problem 1 as a global optimization task, akin to the work in [12], by approximating solutions within a Fourier basis. Importantly, we employ node meshes exclusively at the boundary, rather than across the entire domain. To enhance convergence, we integrate the Schwarz domain decomposition method, employing fractional transmission conditions at sub-domain interfaces. This results in a reformulated optimization problem, which we train to determine the general solution.

The structure of this article is as follows: we commence by presenting our domain decomposition method in both 2D and 3D, considering Cartesian (and can be reformulated to cylindrical coordinates). Subsequently, we introduce our meshless method, featuring an artificial neural network solver. Finally, we present numerical results, comparing them against analytical solutions. Through simulation tests, we demonstrate the effectiveness of our proposed method.

II. The Waveform Relaxation Domain Decomposition Method

A. Domain Decomposition in 2D

The equation 1 can be expressed in 2D as follows:

$$\begin{cases} 2\frac{1-\nu}{1-2\nu}\frac{\partial^2 u}{\partial x^2} + \frac{1}{1-2\nu}\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y} &= 0 \quad on \ \Omega \\ 2\frac{1-\nu}{1-2\nu}\frac{\partial^2 v}{\partial y^2} + \frac{1}{1-2\nu}\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 u}{\partial x \partial y} &= 0 \quad on \ \Omega \\ + BC \ on \ \partial\Omega \end{cases} \tag{2}$$

Here, $\mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix}$, and Ω represents our boundary domain.

In the following sections, we present the principles of our Schwarz domain decomposition method. To illustrate, let's consider a scenario where our domain Ω is divided into two sub-domains: Ω_1 and Ω_2, separated by an interface Γ (refer to Figure 1).

By conducting calculations (for detailed derivations, refer to [16], [17], [12]), we find that the eigenvalues of the

Manuscript received January 17, 2024; revised June 18, 2024.

M. R. Amattouch is a Professor of mathematics at the Department of Mathematics, Mechanics, Cryptography and Numerical analysis University Hassan II, Faculty of Science and Technics of Mohamedia, Morocco (e-mail: mohamedridouan.amattouch@fstm.ac.ma).

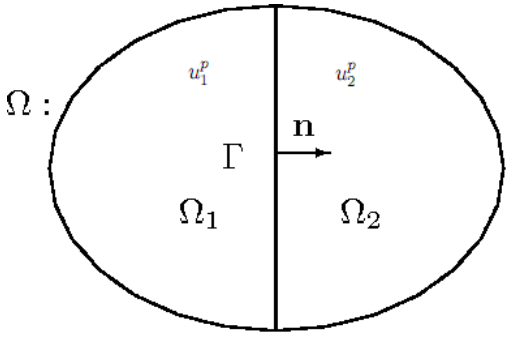


Fig. 1. Domain split into two sub-domains

operator associated with equations 2 are given by:

$$\lambda^{\pm} = -ik \pm k\sqrt{\frac{\nu(3-\nu)}{(1-2\nu)^2}}$$

It's important to note that, in general, $1-2\nu > 0$ ($-1 < \nu < 0.5$ for most materials). This insight leads us to develop a novel domain decomposition method. We establish two sequences, u_1^p and u_2^p (p=1,2), as follows: Starting with initial functions u_1^0 and u_2^0 defined on Ω_1 and Ω_2 respectively, we iteratively compute u_1^p and u_2^p by solving auxiliary problems:

$$\begin{cases} L(u_1^{p+1}) = 0 \quad on \quad \Omega_1 \\ BC\ for\ u_1^{p+1} \quad on \quad \partial\Omega \\ B_1(u_1^{p+1}) = B_2(u_2^p) \quad on \quad \Gamma \end{cases} \tag{3}$$

And

$$\begin{cases} L(u_2^{p+1}) = 0 \quad on \quad \Omega_2 \\ BC\ for\ u_2^{p+1} \quad on \quad \partial\Omega \\ B_2(u_2^{p+1}) = B_1(u_1^p) \quad on \quad \Gamma \end{cases} \tag{4}$$

Where

$$L(\mathbf{u}) = \begin{pmatrix} 2\frac{1-\nu}{1-2\nu}\frac{\partial^2 u}{\partial x^2} + \frac{1}{1-2\nu}\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y} \\ 2\frac{1-\nu}{1-2\nu}\frac{\partial^2 v}{\partial y^2} + \frac{1}{1-2\nu}\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 u}{\partial x \partial y} \end{pmatrix} \tag{5}$$

$$B_1(u) = \begin{pmatrix} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \\ \frac{\partial^2 u}{\partial y} \end{pmatrix} \tag{6}$$

And

$$B_2(u) = \begin{pmatrix} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \\ -\frac{\partial u}{\partial y} \end{pmatrix} \tag{7}$$

We prove the convergence of our algorithms 3 and 4 within two iterations. These algorithms are independent and can be executed in parallel. We also extend this method to splitting into multiple sub-domains. In Figure 2, we illustrate the division of the domain into 2n sub-domains Ω_i, where Γ_i denotes the interface between the sub-domains Ω_i and Ω_{i+1}.

For $i = 1$ to $2n$, we construct sequences u_i^p (p=1,2) as follows: Starting with initial functions u_i^0 defined on Ω_i, we iteratively compute u_1^p and u_2^p by solving auxiliary problems:

$$\begin{cases} L(u_i^{p+1}) = 0 \quad on \quad \Omega_i \\ BC\ for\ u_i^{p+1} \quad on \quad \partial\Omega \cap \Omega_i \\ B_1(u_i^{p+1}) = B_2(u_{i+1}^p) \quad on \quad \Gamma_i \\ B_2(u_i^{p+1}) = B_1(u_{i-1}^p) \quad on \quad \Gamma_{i-1} \end{cases} \tag{8}$$

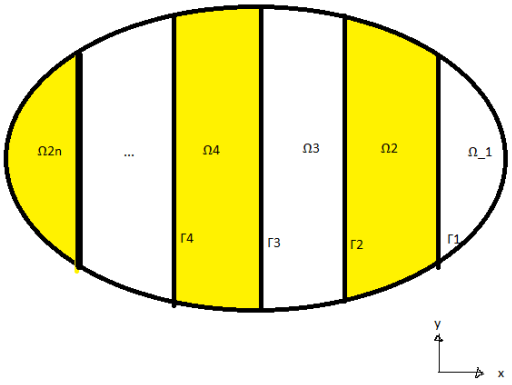


Fig. 2. Domain split into 2n sub-domains

B. Domain Decomposition in 3D Cartesian Coordinate

The equations 1 can be extended to 3D as follows:

$$\begin{cases} 2\frac{1-\nu}{1-2\nu}\frac{\partial^2 u}{\partial x^2} + \frac{1}{1-2\nu}\left(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) \\ \quad + \frac{\partial^2 v}{\partial x \partial y} + \frac{\partial^2 w}{\partial x \partial z} = 0 \quad On\ \Omega \\ 2\frac{1-\nu}{1-2\nu}\frac{\partial^2 v}{\partial y^2} + \frac{1}{1-2\nu}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2}\right) \\ \quad + \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 w}{\partial y \partial z} = 0 \quad On\ \Omega \\ 2\frac{1-\nu}{1-2\nu}\frac{\partial^2 w}{\partial z^2} + \frac{1}{1-2\nu}\left(\frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial x^2}\right) \\ \quad + \frac{\partial^2 v}{\partial z \partial y} + \frac{\partial^2 u}{\partial x \partial z} = 0 \quad On\ \Omega \\ \quad + BC \quad On\ \partial\Omega \end{cases} \tag{9}$$

Here, $\mathbf{u} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$. In the following sections, we outline the principles of our Schwarz domain decomposition method for 3D Cartesian coordinates. To illustrate this, let's consider a scenario where our domain Ω is divided into two sub-domains: Ω_1 and Ω_2, separated by an interface Γ (refer to Figure 3).

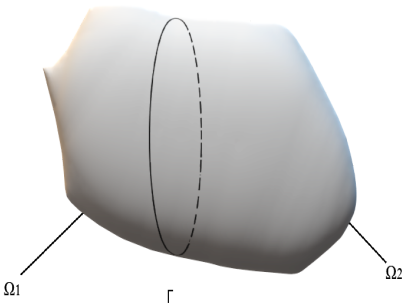


Fig. 3. Domain split into two sub-domains

Through computations, we determine the eigenvalues of

the operator associated with equations 9 as follows:

$$\lambda^{\pm} = -i(k+l) \pm \sqrt{\Delta \frac{\nu(3-\nu)}{(1-2\nu)^2}}$$

Where

$$\Delta = -(k+l)^2 + 8\frac{1-\nu}{(1-2\nu)^2}(k^2+l^2)$$

Here, k (respectively l) is the frequency associated with the variable y (respectively z). We can further express Δ as

$$\Delta = -(k+l)^2 + \gamma(k^2+l^2) = (c_1 l + c_2 k)(c_3 l + k)$$

With

$$c_1 = -1 + \sqrt{1+(\gamma-1)^2}, \quad c_2 = \gamma - 1,$$

And

$$c_3 = \frac{\gamma - 1}{-1 + \sqrt{1+(\gamma-1)^2}}$$

It's important to note that $\Delta > 0$ in general ($-1 < \nu < 0.5 < \frac{\sqrt{3}}{2}$). Based on these insights, we develop a novel domain decomposition method. We establish two sequences, u_1^p and u_2^p (p=1,2), as follows: Starting with initial functions u_1^0 and u_2^0 defined on Ω_1 and Ω_2 respectively, we iteratively compute u_1^p and u_2^p by solving auxiliary problems:

$$\begin{cases} L(u_1^{p+1}) = 0 \quad on \quad \Omega_1 \\ BC\ for\ u_1^{p+1} \quad on \quad \partial\Omega \\ B_1(u_1^{p+1}) = B_2(u_2^p) \quad on \quad \Gamma \end{cases} \tag{10}$$

And

$$\begin{cases} L(u_2^{p+1}) = 0 \quad on \quad \Omega_2 \\ BC\ for\ u_2^{p+1} \quad on \quad \partial\Omega \\ B_2(u_2^{p+1}) = B_1(u_1^p) \quad on \quad \Gamma \end{cases} \tag{11}$$

Where L is the operator associated with equation 9

$$B_1(u) = \frac{\partial u}{\partial x} - D^{\frac{1}{2}}_{c_1\mathbf{k}+c_2\mathbf{j}} D^{\frac{1}{2}}_{c_3\mathbf{k}+\mathbf{j}}(u) \tag{12}$$

And

$$B_2(u) = \frac{\partial u}{\partial x} + D^{\frac{1}{2}}_{c_1\mathbf{k}+c_2\mathbf{j}} D^{\frac{1}{2}}_{c_3\mathbf{k}+\mathbf{j}}(u) \tag{13}$$

Here, $D^{\frac{1}{2}}_{c_1\mathbf{k}+c_2\mathbf{j}}$ (and $D^{\frac{1}{2}}_{c_3\mathbf{k}+\mathbf{j}}$) represents the Caputo derivative of a function f in the direction $c_1\mathbf{k}+c_2\mathbf{j}$ (and $c_3\mathbf{k}+\mathbf{j}$). It's worth noting that the Fourier transform of $D^{\frac{1}{2}}_{c_1\mathbf{k}+c_2\mathbf{j}}$ (and $D^{\frac{1}{2}}_{c_3\mathbf{k}+\mathbf{j}}$) can be expressed as

$$\mathbf{F}(D^{\frac{1}{2}}_{c_1\mathbf{k}+c_2\mathbf{j}})(u)(k,l) = \sqrt{i(c_1k+c_2l)}\mathbf{F}(u)(k,l)$$

and

$$\mathbf{F}(D^{\frac{1}{2}}_{c_3\mathbf{k}+\mathbf{j}})(u)(k,l) = \sqrt{i(c_3k+l)}\mathbf{F}(u)(k,l)$$

Consequently, we demonstrate that the two algorithms 10 and 11 converge within two iterations. The same approach is extended to constructing a multiple domain decomposition method in 3D as presented for 2D in section II-A.

III. Artificial Neural Intelligence Method

In this section, we seek an approximation $\mathbf{u}_{approx}$ for the solution of problems 3 and 10 in the Fourier basis.

In 2D, we seek coefficients a_{ij}^l, b_{ij}^l, c_{ij}^l, d_{ij}^l, for $i,j = 1,\ldots,N$ and $l = 1,2$, such that:

$$\begin{aligned} u_{approx} = \sum_{i=0}^{N}\sum_{j=0}^{N} a_{ij}^1 \cos(i\pi\frac{x}{L})\cos(j\pi\frac{y}{L}) \\ + b_{ij}^1 \cos(i\pi\tfrac{x}{L})\sin(j\pi\tfrac{y}{L}) \\ + c_{ij}^1 \sin(i\pi\tfrac{x}{L})\cos(j\pi\tfrac{y}{L}) + d_{ij}^1 \sin(i\pi\tfrac{x}{L})\sin(j\pi\tfrac{y}{L}) \end{aligned}$$

And similarly for v_{approx}. For simplicity, we can write:

$$u_{approx} = \sum_{i=0}^{M} \mathbf{a_i^1}\Theta_i v_{approx} = \sum_{i=0}^{M} \mathbf{a_i^2}\Theta_i$$

Where $(\Theta_i)_i$ represents the Fourier basis.

The coefficients $\mathbf{u}_{approx}$ minimize the objective function:

$$\begin{aligned} \epsilon = \|L(\mathbf{u}_{approx})\| + \|B_1(\mathbf{u}_{approx}) - B_2(u_{prec})\| + \\ \|BC(\mathbf{u}_{approx})\| = \epsilon_1 + \epsilon_2 + \epsilon_3 \end{aligned}$$

Here, $\mathbf{u_{prec}} = \begin{pmatrix} \sum_{i=0}^{M} \mathbf{ap_i^1}\Theta_i \\ \sum_{i=0}^{M} \mathbf{ap_i^2}\Theta_i \end{pmatrix}$

$\mathbf{u_{prec}}$ represents the initial condition (which can be null) or the last obtained value from 4.

For the first and second parts of the objective function (ϵ_1 and ϵ_2), we define the norms as:

$$\|\mathbf{u}_{approx}\| = \sum_{i=0}^{M} (\mathbf{a_i^1})^2 + (\mathbf{a_i^2})^2$$

By hand calculation, we determine $L(\mathbf{u}_{approx})$ and $B_1(\mathbf{u}_{approx}) - B_2(u_{prec})$ and compute the two norms:

$$\epsilon_1 = \|L(\mathbf{u}_{approx})\| \quad and \quad \epsilon_2 = \|B_1(\mathbf{u}_{approx}) - B_2(u_{prec})\|$$

It's important to note that the norms ϵ_1 and ϵ_2 are quadratic functions of the coefficients a_i and are convex, ensuring that the stochastic gradient can be effectively used and converge.

The norm ϵ_3 associated with the boundary condition is approximated as follows: We discretize the boundary of the sub-domain Ω_i with distributed nodes (refer to Figures 4 and 5).

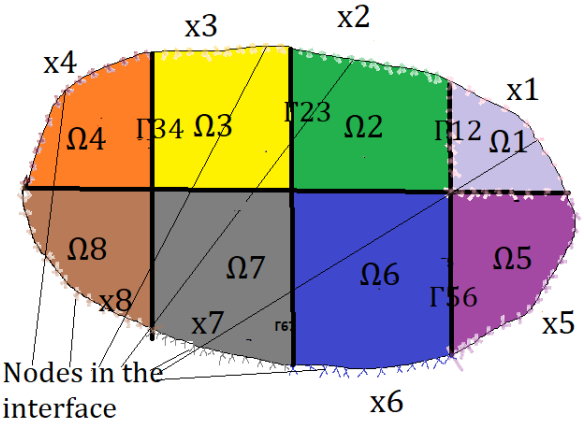


Fig. 4. Nodes on the boundary of the domain in 2D

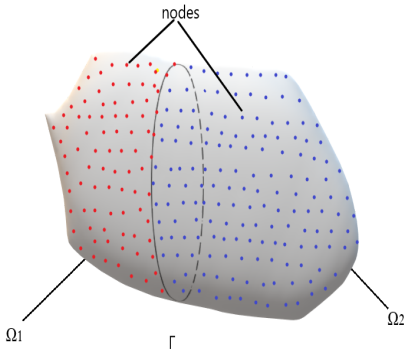


Fig. 5. Nodes on the boundary of the domain in 3D

After defining the nodes x_i^j $(j = 1, \ldots, NN)$ on the boundary of the sub-domain Ω_i, we compute the norm ϵ_3:

$$\epsilon_3 = \sum_{i=0}^{NN} |BC(\mathbf{u}_{approx})(x_j)|$$

The boundary condition BC consists of general Dirichlet, Neumann conditions, or other differential conditions, typically written as $BC(\mathbf{u}_{approx}) = 0$ in equations 3 or 10.

This entire loss function ϵ is then used to train an Artificial Neural Network (ANN), as described in [15] and [18]. The method is summarized in Figure 6. The training data x_i is quasi-uniformly selected and contains the common interface points in the sub-domain Ω_i. The loss function is defined subdomain-wise and includes the interface conditions to stitch the sub-domains together at the boundary.

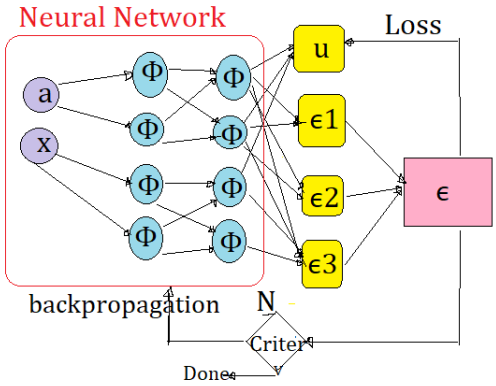


Fig. 6. Schematic of our ANN method employed in a subdomain

The optimization problem of this ANN is guaranteed to be convergent with the gradient descent method due to the convex nature of the loss function.

This extended result underscores a well-founded approach of integrating boundary conditions into the ANN training process for PDEs. Building a complete loss function with a diversity of boundary conditions and including classic optimization techniques sets up a very promising framework for a variety of problems in science and engineering.

IV. Numerical Simulations

To evaluate the efficiency of our proposed method, we compare the simulation results of our equation with established finite element benchmarks like FreeFEM++ [19] and FEniCS [20]. We provide four examples of different domains for demonstration.

First Example: Consider a polygonal domain Ω (see Figure 7). The bottom and right boundaries Γ_D have fixed conditions, while a linear loading f is applied to the left upper edge of the plate Γ_N.

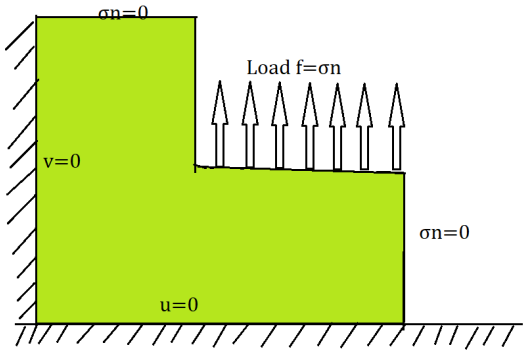


Fig. 7. Domain Ω with load f

The problem is governed by the same boundary value formulation as before:

$$\begin{cases} div(\nabla \mathbf{u}) + \frac{1}{1-2\nu}\Delta \mathbf{u} = 0 \quad on \quad \Omega \\ \sigma.n = 0 \quad on \quad \Gamma_D \\ u = 0 \quad on \quad \Gamma_N \end{cases} \tag{14}$$

where g represents the surface loading, and the Dirichlet boundary condition is applied to the bottom part of the boundary Γ_D. The Young's modulus $E = 69000Pa$, Poisson's ratio $\nu = 0.346$, and the load is $f = 200MPa$.

To evaluate the performance of our computational method, we divide the computational domain into three subdomains. The interfaces between these subdomains are represented by two lines. Within each subdomain, the network architecture comprises six hidden layers, utilizing the hyperbolic tangent (tanh) activation function. We use 100 interface points on each interface and 700 boundary points.

The simulation results for the predicted solution and a classic FreeFem++ simulation are presented in Figure 8, demonstrating that the two simulations match closely. The L_2 error between our method's solution and the finite element solution obtained using FreeFem++ (in the P_2 space) is $3.37\ 10^{-3}$.

To test the convergence of our algorithm, we define the interface loss error, expressed as follows:

$$MSE = \sum_{interfacej} \sum_{pointi \in interfacej} \|u_i^{j1} - u_i^{j1}\|$$

Here, u^{j1} denotes the solution in subdomain j1 and u^{j2} denotes the solution in subdomain j2, which shares interface j with j1. The interface loss error in our simulation is $1.33\ 10^{-5}$.

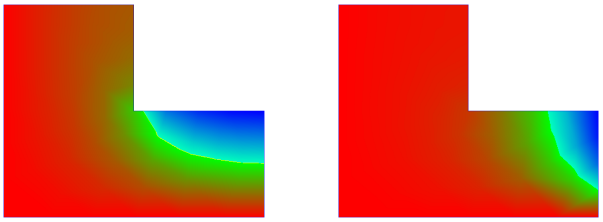


Fig. 8. Displacement coordinates

Second Example: Consider the same polygonal domain Ω (see Figure 9). The bottom and right boundaries Γ_D have fixed conditions, and a linear loading f is applied to the left edge of the plate Γ_N.

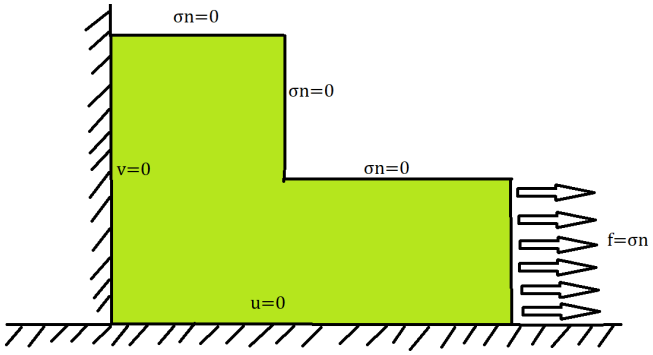


Fig. 9. Domain Ω with load f

The problem is again governed by the same boundary value formulation:

$$\begin{cases} div(\nabla \mathbf{u}) + \frac{1}{1-2\nu}\Delta \mathbf{u} &= 0 \quad on \quad \Omega \\ \sigma.n &= 0 \quad on \quad \Gamma_D \\ u &= 0 \quad on \quad \Gamma_N \end{cases} \tag{15}$$

The Young's modulus $E = 69000Pa$, Poisson's ratio $\nu = 0.346$, and the load is $f = 200MPa$.
We divide again the computational domain into three subdomains. The interfaces between these subdomains are represented by two lines. Within each subdomain, the network architecture comprises six hidden layers, utilizing the hyperbolic tangent (tanh) activation function. We use 100 interface points on each interface and 700 boundary points.

The simulation results for the predicted solution and a classic FreeFem++ simulation are presented in Figure 10, demonstrating that the two simulations match closely. The L_2 error between our method's solution and the finite element solution obtained using FreeFem++ (in the P_2 space) is $3.04\ 10^{-3}$.

To test the convergence of our algorithm, we define the interface loss error, expressed as follows:

$$MSE = \sum_{interfacej} \sum_{pointi \in interfacej} \|u_i^{j1} - u_i^{j1}\|$$

Here, u^{j1} denotes the solution in subdomain j1 and u^{j2} denotes the solution in subdomain j2, which shares interface j with j1. The interface loss error in our simulation is $1.2\ 10^{-5}$.

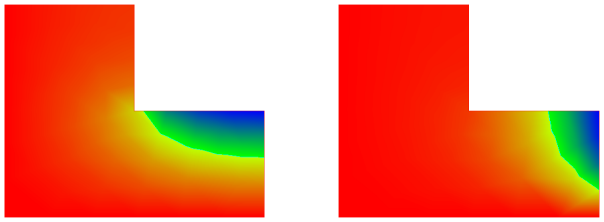


Fig. 10. Displacement coordinates

Third Example: Consider a rectangular domain Ω with a central circular hole of radius R (see Figure 11). The bottom boundary Γ_D has fixed conditions, and a linear loading f is applied to the upper edge of the plate Γ_N. The problem aims to study the shape deformation of the circle and stress concentration around its circumference.

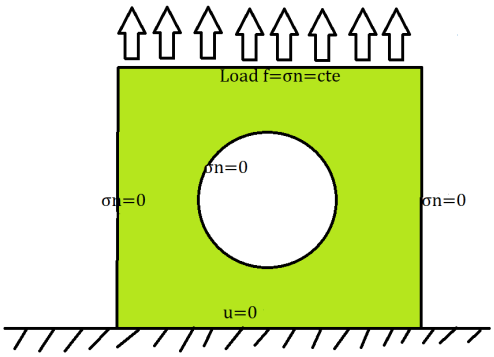


Fig. 11. Domain Ω with load f

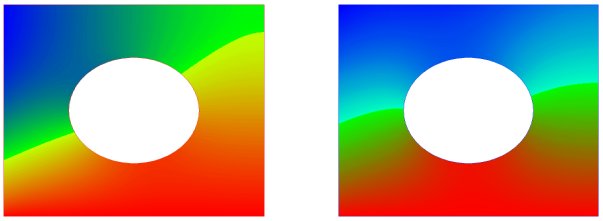
The problem formulation remains the same:

$$\begin{cases} div(\nabla \mathbf{u}) + \frac{1}{1-2\nu}\Delta \mathbf{u} &= 0 \quad on \quad \Omega \\ \sigma.n &= 0 \quad on \quad \Gamma_D \\ u &= 0 \quad on \quad \Gamma_N \end{cases} \tag{16}$$

The Young's modulus $E = 69000Pa$, Poisson's ratio $\nu = 0.346$, and the load is $f = 200MPa$.
To evaluate the performance of our computational method, we divide the computational domain into four subdomains. The interfaces between these subdomains are represented by four lines. Within each subdomain, the network architecture comprises five hidden layers, utilizing the hyperbolic tangent (tanh) activation function. We use 100 interface points on each interface and 1000 boundary points.

The simulation results for the predicted solution and a classic FreeFem++ simulation are presented in Figure 12, demonstrating that the two simulations match closely. The L_2 error between our method's solution and the finite element

solution obtained using FreeFem++ (in the P_2 space) is $1.04\ 10^{-1}$.
The interface loss error in our simulation is $4.47\ 10^{-4}$.



Simulation of u Simulation of v

Fig. 12. Displacement coordinates

Last Example: Now, we extend the domain to 3 dimensions (Figure 13).

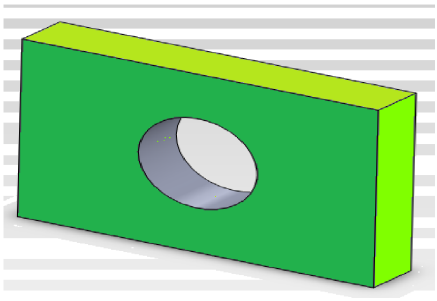


Fig. 13. 3D Domain Ω

We consider the same equation with the same boundary conditions as in problem 16.
To evaluate the performance of our computational method, we divide the computational domain into four subdomains. The interfaces between these subdomains are represented by four surfaces. Within each subdomain, the network architecture comprises seven hidden layers, utilizing the hyperbolic tangent (tanh) activation function. We use 1000 interface points on each interface and 10000 boundary points.

The simulation results for the predicted solution and a classic FreeFem++ simulation are presented in Figure 14, demonstrating that the two simulations match closely. The L_2 error between our method's solution and the finite element solution obtained using FreeFem++ (in the P_2 space) is $9.04\ 10^{-1}$.
The interface loss error in our simulation is $5.46\ 10^{-3}$.

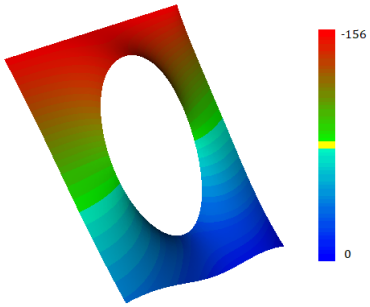


Fig. 14. Simulation of the norm of $\mathbf{u}$ on the domain's surface

The results we obtained for the last examples are similar to the simulations in the Freefem++ Benchmark with efficient accuracy. In fact, the $L^2(\Omega)$ error between the solutions of our methods and the solution obtained by the Freefem++ software is on the order of 0.01 and 0.001. It is worth noting that the Freefem++ software requires a finer mesh to achieve better accuracy, which comes at the cost of increased computation time.

V. Conclusion

In this article, we have introduced a novel approach to solving the elasticity model that combines three distinct methods: a meshless method, a fast domain decomposition method, and an artificial neural network approach. The integration of these methods offers several advantages in terms of efficiency and accuracy.

Our proposed method operates primarily on the boundary of the domain and leverages wave relaxation techniques along with artificial neural networks to accelerate convergence. By focusing on the domain's boundary and employing advanced computational tools, we achieve notable improvements in the efficiency of the solution process.

To validate the effectiveness of our method, we conducted numerical simulations and compared the results with established benchmarks such as FreeFEM. The obtained accuracies for various basic geometries showcased the method's capabilities.

One of the key strengths of our approach is its adaptability to complex simulations involving intricate domains, particularly in higher dimensions. By reducing the need for extensive collocation points and utilizing domain decomposition techniques, our method can efficiently accommodate parallelized computation.

In the future, we intend to extend this approach to more complex scenarios, including applications to the Navier-Stokes equation and other nonlinear mechanical models. The combination of innovative techniques presented here has the potential to significantly enhance the accuracy and efficiency of simulations in various scientific and engineering domains.

References

[1] Fenner,R.T. (1975). Finite element methods for engineers. Editor:Macmillan
[2] Axelsson,O. (1976). A class of iterative methods for finite element equations. Comp. Meth. Appl. Mech. Eng.
[3] Arnold, D.N. and Falk, R.S. (1988). A new mixed formulation for elasticity. Numer. Math., 53:13–30.
[4] Brenner, S.C. and Sung, L. (1992). Linear Finite Element Methods for Planar Linear Elasticity. Mathematics of Computation, Vol. 59, No. 200, pp. 321-338. https://doi.org/10.2307/2153060
[5] Le Louër, Frédérique, L.L. (2014). A high order spectral algorithm for elastic obstacle scattering in three dimensions. Journal of Computational Physics, Volume 279, p. 1-17. doi:10.1016/j.jcp.2014.08.047
[6] Ganesh, M. ; Hawkins, S. C. (2008). A high-order tangential basis algorithm for electromagnetic scattering by curved surfaces. Journal of Computational Physics, Volume 227, Issue 9, p. 4543-4562. https://doi.org/10.1016/j.jcp.2008.01.016
[7] Chen H., Su Y., and Shizgal B.D. (2000). A direct spectral collocation Poisson solver in polar and cylindrical coordinates. J. Comput. Phys. 160:453-469.
[8] Fornberg B. (1987). The pseudospectral method: comparisons with finite differences for the elastic wave equation. Geophysics, 52:483–501.
[9] Larsson, E., Shcherbakov, V., Heryudono, A. (2017). A least squares radial basis function partition of unity method for solving pdes. SIAM J. Sci. Comp. 39(6), A2538–A2563
[10] Piret, C., Dunn, J. (2016). Fast rbf ogr for solving pdes on arbitrary surfaces. AIP Conference Proceedings 1776(1), 070005.

[11] S. N. Atluri and T. Zhu (2000). The Meshless Local Petrov-Galerkin (MLPG) approach for solving problems in elasto-statics. In Computational Mechanics, vol.25,no.2,pp.169-179.
[12] Amattouch M. R. and Belhadj H.(2020). An Heuristic Scheme for a Reaction Advection Diffusion Equation. Heuristics for Optimization and Learning.
[13] Mattey, R. and Ghosh, S. (2022). A Physics Informed Neural Network for Time-Dependent Nonlinear and Higher Order Partial Differential Equations. Computer Methods in Applied Mechanics and Engineering, Volume 390, 114474, ISSN 0045-7825.
[14] Raissi, M., Perdikaris, P. and George Em Karniadakis (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, Volume 378, Pages 686-707.
[15] Jagtap, A.D., Karniadakis, G.E (2020). Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun. Comput. Phys., 28, pp. 2002-2041.
[16] Amattouch, M.R., Nagid, N. and Belhadj, H. (2017). A modified fixed point method for The Perona Malik equation, Journal of Mathematics and System Science 7, 175-185.
[17] Amattouch, M.R. and Belhadj, H. (2017). Combined Optimized Domain Decomposition Method and a Modified Fixed Point Method for Non Linear Diffusion Equation, Applied Mathematics and Information Sciences, 11, No. (1), 201-207.
[18] Jagtap, A.D., Kawaguchi, K. and Karniadakis,G.E. (2020). Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. R. Soc. A 476: 20200334.
[19] Hecht, F. (2012). New development in FreeFem++. Journal of Numerical Mathematics. https://doi.org/10.1515/jnum-2012-0013
[20] Langtangen, H. P. and Logg, A. (2016). Solving PDEs in Python, The FEniCS Tutorial. Simula SpringerBriefs on Computing (SBRIEFSC, volume 3)
[21] Yucheng Fan, and Dong Qiu (2024). On the Evolution of COVID-19 Virus Based on the Prediction Model of Deep Learning and Emotion Analysis, Engineering Letters, vol. 32, no. 2, pp 412-428.
[22] Mohammed Baati, Nada Chakhim, Mohamed Louzar, and Abdellah Lamnii (2024). Numerical Approximation of the One-dimensional Inverse Stefan Problem Using a Meshless Method, Engineering Letters, vol. 32, no. 1, pp112-122.
[23] Xie, H., Zhang, Z., Jiang, Z., and Zhou, J. (2023). Method of Particular Solutions for Second-Order Differential Equation with Variable Coefficients via Orthogonal Polynomials. Journal of Function Spaces,2023, Article ID 9748605, 10 pages. https://doi.org/10.1155/2023/9748605
[24] Antonion, K., Wang, X., Raissi, M., and Joshie, L.(2024). Machine Learning Through Physics–Informed Neural Networks: Progress and Challenges. Academic Journal of Science and Technology, 9(1), 46-49. https://doi.org/10.54097/b1d21816