Boundary Region Reinforcement Physics-Informed Neural Networks for solving Partial Differential Equations

Shengtai Yao[a,b], Weifeng Huang[a], Yang Hu[c], Qiang He[a,*]

[a]*Mechanical Engineering, Tsinghua University, Beijing , 100084, China*
[b]*Applied Mathematics and Statistics, Johns Hopkins University, Baltimore , 21218, MD, United States*
[c]*Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, Beijing , 100044, China*

Abstract

Physics-Informed Neural Networks (PINNs) have emerged as a deep learning framework for solving partial differential equations (PDEs). However, standard approaches often struggle to strictly enforce boundary conditions, limiting their computational accuracy, especially in complex multiphysics coupled systems. To address this challenge, we introduce Boundary Region Reinforcement Physics-Informed Neural Networks (BRR-PINNs), a novel framework designed to strengthen boundary constraint enforcement and significantly improve solution accuracy. We validate the effectiveness of BRR-PINNs on a two-dimensional axisymmetric thermo-elastic coupling problem, involving a circular ring subjected to both thermal and mechanical loads. Experimental results show that our method consistently outperforms several existing approaches, achieving approximately one order of magnitude improvement in accuracy.

Keywords: Partial differential equations, Physics-Informed Neural Networks, Boundary Region Reinforcement, Computational accuracy, AI for science, Thermo-elastic coupling system

1. Introduction

With the continuous advancement of deep learning, its extensive applications in areas such as image recognition, semantic understanding, and autonomous driving rely largely on models trained with vast amounts of data. However, in fields such as engineering applications, there are some research problems in which obtaining a large amount of data is infeasible or prohibitively costly. Raissi et al. [1] proposed Physics-Informed Neural Networks (PINNs), which integrate the physical laws and experimental data into the loss function of the neural network. The entire training process involves minimizing the loss function, ensuring that the resulting model satisfies the given equations and conditions. The advantage is that it requires a small amount of, or even no, labeled experimental data to obtain solutions to partial differential equations (PDEs), due to the incorporation of physical laws into the network. The core of PINNs computational capability lies in automatic differentiation (AD) [2], which enables it to incorporate physical equations into the loss function. AD can be conveniently accessed in commonly used frameworks such as PyTorch and TensorFlow. Raissi et al. [1] proposed two types of problem: the data-driven solution and the data-driven discovery of partial differential equations. They validated the effectiveness of the PINNs method on equations involving the Schrödinger equation, Allen-Cahn equation, and Navier-Stokes equations. Since it has been proposed, PINNs have been applied in various fields, including heat transfer [3, 4] chemical kinetics [5, 6], fluid dynamics [7, 8], solid mechanics [9, 10, 11], systems biology [12], geophysics [13] etc.

After the proposal of PINNs, various variants have been proposed to enhance the model performance and extend its application to different problems. These include B-PINNs [14], based on a Bayesian framework to mitigate over fitting and achieve higher prediction accuracy in noisy data; CAN-PINN [15], which couple AD with numerical differentiation (ND) to obtain greater robustness and efficiency; g-PINNs [16], a gradient-enhanced method, improves

*Corresponding author
Email addresses: syao31@jh.edu (Shengtai Yao), huangwf@mail.tsinghua.edu.cn (Weifeng Huang), yanghu@bjtu.edu.cn (Yang Hu), heqiang@tsinghua.mail.edu.cn (Qiang He)

the smoothness of the obtained solution by adding derivative terms to the loss function, thus enhancing the precision; cPINN [17], designed to solve nonlinear conservation laws in discrete domains, can efficiently lend itself to parallelized computation, etc. The exploration of the computational accuracy, efficiency, and application fields of PINNs is still ongoing.

In traditional finite element methods (FEM), boundary conditions can be exactly imposed at mesh points. However, for mesh-free methods, including PINNs, mesh-free Galerkin methods [18], etc., satisfying boundary conditions is a well-known and pertinent issue [19]. Many studies have shown that errors in boundary conditions significantly affect the convergence and accuracy of the overall solution [20, 21, 22]. Without proper boundary condition constraints, PDEs may have infinitely many solutions; moreover, in some PDEs, such as elliptic partial differential equations, boundary errors can propagate throughout the entire solution domain at an infinite speed [23]. Several efforts have been made to reach a penalty-free, boundary exactly imposed method [19, 21, 24, 25, 26, 27]. Rao et al. [24] proposed a "hard" manner to satisfy initial/boundary conditions (I/BCs), including training three DNNs: I/BCs DNN, Distance function DNN, and General DNN. Sukumar et al. [19] achieved by constructing approximation distance functions (ADFs) derived from R-functions [28], in order to obtain the geometry-trail function for boundary conditions involving Dirichlet, Neumann, and Robin boundary. This type of hard constraint method can obtain more accurate results and has been used to impose boundary conditions, particularly Dirichlet boundaries, in numerous studies [29, 30, 31]. For some specific problems, e.g. beam problems, deep ROLS [32] is proposed with a BCs embedding method, which efficiently embeds the high order BCs based on the "hard" method.

Currently, PINNs method is used in some multi-physics problems, such as thermo-elastic deformation [33, 34, 35], thermo-fluid flows [36, 37, 38], electro-thermal coupling [39, 40], thermo-chemical problems [41, 42], etc. However, in such coupled systems, the governing PDEs typically involve a larger number of equations and coupling terms, resulting in more loss components and additional hyperparameters that must be manually tuned. Training PINNs on these systems is particularly challenging due to the complexity of the loss function, the high-frequency components, and the multiscale features of the underlying physics [43, 44, 45, 46]. Wang et al. [20] attributed this failure to multiscale interactions that induce stiffness in the gradient flow, preventing the optimization process. They further analyzed the training dynamics of PINNs through the lens of the Neural Tangent Kernel (NTK) [47], revealing that different loss components converge at remarkable disparate rates. The “hard” method mentioned earlier may help reduce the number of loss terms; however, it mainly acts on Dirichlet boundary conditions and has limited influence on the inherent difficulties of multi-physics systems. To address these issues, several variational energy-based formulations have been proposed to reduce both the number and the order of governing equations [31, 48]. Other approaches, including mixed formulation PINNs [49, 33], alternate two-stage PINNs (ATPINN) [50], and gradient-enhanced PINNs (g-PINNs) [51], have been developed to improve the accuracy and stability of PINNs when predicting coupled physical fields. In addition, various training strategies, including adaptive learning [52, 20, 47], have been introduced to mitigate the imbalance arising from multiscale features during optimization. Nevertheless, the fundamental reasons for the failure of PINNs and the development of efficient, generalizable solutions remain understudied. We present a simple but effective idea, attributing the failure of PINNs to the incomplete satisfaction of boundary conditions, which may cause the governing system to admit multiple or even infinitely many solutions. Based on this observation, we propose a generalized framework capable of delivering precise predictions, even for complex multiphysics coupled problems.

The popularity of the PINNs method has been steadily increasing in recent years, however, challenges such as computational accuracy and speed still need to be addressed. This paper proposes a novel algorithm, BRR-PINNs, aimed at improving the accuracy of PINNs in fitting boundary conditions, thereby enhancing overall precision, reducing the number of terms in the loss function, and simplifying the tuning of weight parameters. We intend to validate the effectiveness of this algorithm on a two-dimensional axisymmetric thermo-elastic coupling problem, including heat transfer, elastic deformation, and thermo-mechanical coupling deformation calculations.

The structure of the paper is as follows: Section 2 reviews conventional PINNs algorithms and introduces the new BRR-PINNs framework. Section 3 presents the application of the proposed method to a thermo-elastic coupling problem, along with numerical validation, comparative analysis, and an in-depth discussion of the algorithm. Section 4 summarizes the findings and discusses future work. Appendix A describes the setup of the problem, including the PDEs and boundary conditions. Appendix B provides several examples on the implementation of the proposed method. Appendix C contains additional details that were omitted from Section 3. The relevant code and data are available on GitHub: https://github.com/Yaoshengtai/BRR-PINNs.git.

2. Methodology

2.1. PINNs

The core idea of PINNs is to embed physical equations into the loss function of the neural network, utilizing the ability of the neural network to fit the mapping from inputs to outputs of the PDEs. The network training process is essentially the process of solving the equation. Once trained, the network describes the solution function of the PDEs. First, consider the general form of a PDE as shown in (1),

$$\mathcal{N}[u(\mathbf{x},t);\lambda] = 0, \quad \mathbf{x} \in \Omega,\ t \in [0,T], \tag{1}$$

where, u is the latent solution of the PDEs function; $\mathcal{N}$ is a generalized differential operator, which can be linear or nonlinear, parameterized by λ, taking the heat transfer equation as an example, $\mathcal{N} = \frac{\partial}{\partial t} - \lambda\nabla^2$; $\mathbf{x}$ and t represent the spatial coordinates and the temporal coordinates; Ω is the solution domain that is a subset of $\mathbb{R}^d$.

To solve the PDEs, initial conditions and boundary conditions are required to constrain the problem. Use the function $f(\mathbf{x})$ to represent the initial condition, as shown in (2). The boundary condition is represented as $g(\mathbf{x},t)$, shown in (3), where $\partial\Omega$ is the boundary of the domain.

$$u(\mathbf{x},0) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{2}$$

$$u(\mathbf{x},t) = g(\mathbf{x},t), \quad \mathbf{x} \in \partial\Omega,\ t \in [0,T]. \tag{3}$$

Automatic differentiation is a fundamental technique in neural networks. Using the chain rule, it can compute the derivatives of the output variables with respect to the intermediate parameters and the input variables. This enables the formulation of PDEs and conditions within the loss function. After training the network, the resulting function that describes the output in relation to the input satisfies all specified equations and conditions, providing a solution to the given PDEs under the specified conditions. The loss function can be decomposed into a weighted sum of several individual losses, as shown in (4),

$$\mathcal{L} = \omega_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \omega_{\text{ic}}\mathcal{L}_{\text{ic}} + \omega_{\text{bc}}\mathcal{L}_{\text{bc}} + \omega_{\text{data}}\mathcal{L}_{\text{data}}, \tag{4}$$

where, $\mathcal{L}_{\text{PDE}}$, $\mathcal{L}_{\text{ic}}$, $\mathcal{L}_{\text{bc}}$ and $\mathcal{L}_{\text{data}}$ represent the residual loss of the PDE, the initial conditions, the boundary conditions, and the ground truth data; the ω's denote the weight coefficients of each individual loss, and in this work these coefficients are tuned manually. Each loss is expressed as the average loss over the set of collocation points designated for it, as shown in (5)-(8),

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_P}\sum_{k=1}^{N_P}(\mathcal{N}[\hat{u}(\mathbf{x}_P^k, t_P^k \mid \theta);\lambda])^2, \tag{5}$$

$$\mathcal{L}_{\text{ic}} = \frac{1}{N_i}\sum_{k=1}^{N_i}(\hat{u}(\mathbf{x}_i^k, t=0 \mid \theta) - f(\mathbf{x}_i^k))^2, \tag{6}$$

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_b}\sum_{k=1}^{N_b}(\hat{u}(\mathbf{x}_b^k, t_b^k \mid \theta) - g(\mathbf{x}_b^k, t_b^k))^2, \tag{7}$$

$$\mathcal{L}_{\text{data}} = \frac{1}{N_d}\sum_{k=1}^{N_d}(\hat{u}(\mathbf{x}_d^k, t_d^k \mid \theta) - u^k)^2, \tag{8}$$

where, $\hat{u}(\mathbf{x},t \mid \theta)$ represents the output of the neural network with parameters θ when the input is $(\mathbf{x},t)$, $\{\mathbf{x}_P^k, t_P^k\}_{k=1}^{N_P}$ is the set of collocation points designated to measure the residual loss of the PDEs, referred to as $\mathcal{L}_{\text{PDE}}$, defined on the solution domain, i.e., $\{\mathbf{x}_P^k, t_P^k\}_{k=1}^{N_P} \subset \Omega \times [0,T]$; initial condition residual loss $\mathcal{L}_{\text{ic}}$ is measured on the spatial solution domain, i.e., $\{\mathbf{x}_i^k\}_{k=1}^{N_i} \subset \Omega$; boundary condition residual loss $\mathcal{L}_{\text{bc}}$ is measured on the boundary of solution domain $\{\mathbf{x}_b^k, t_b^k\}_{k=1}^{N_b} \subset \partial\Omega \times [0,T]$; the set of collocation points $\{\mathbf{x}_d^k, t_d^k\}_{k=1}^{N_d}$ for ground truth data is all positions in the domain that ground truth is available, u^k is the true value on the point $(\mathbf{x}_d^k, t_d^k)$. By embedding physical information into the loss function using automatic differentiation, we can train the neural network to obtain solutions to the equations.

"Hard" Method. One of the most critical aspects of accurately solving PDEs is the satisfaction of boundary conditions, which has made boundary condition enforcement a popular area of research. In the above-mentioned approach, embedding boundary conditions into the loss function for training is a method also referred to as a "soft" manner [27]. Sukumar et al. [19] proposed using approximate distance functions (ADFs) to exactly impose boundary conditions– commonly referred to as the "hard" method, with research [53] indicating significant advantages compared to the "soft" manner. The use of ADFs for "hard" imposition of Dirichlet boundaries has been used in many research due to its simplicity and accuracy. The ADF for polygon is represented as (9),

$$\mathrm{ADFs}(\mathbf{x}) = \phi(\mathbf{x}) = \left(\frac{1}{\phi_1^m} + \frac{1}{\phi_2^m} + \ldots + \frac{1}{\phi_k^m}\right)^{-\frac{1}{m}}, \tag{9}$$

where, $\{\phi_i\}_{i=1}^k$ is the distance function to the Dirichlet boundary number i from the spatial coordinates $\mathbf{x}$; m is a smooth order, and when m increases, the ADFs approach the exact distance. In our study, m is selected as 1.0. The ADFs is a function equal to 0 on all Dirichlet boundaries and a positive value (approximate distance) on other positions. Accordingly, construct the network's output such that it constantly satisfies all Dirichlet boundaries, as shown in (10),

$$u_s = u_{par} + \phi u_t, \tag{10}$$

where, u_t is the output of the network; u_s is the solution that is used to calculate the loss function; u_{par} is a particular solution that satisfies all Dirichlet boundaries but may not satisfy the other boundary condition and the PDEs. At Dirichlet boundaries, where ADF ϕ equal 0, the solution u_s equals the particular solution u_{par}. Thus, the solution u_s consistently satisfies all Dirichlet boundaries, and the method is called a "hard" method to enforce boundary conditions. It should be noted that each individual distance function in (9) is given based on different types of lines: line segments and curves. For the boundaries of the most commonly used line segments, the distance function is given by (11)-(13) [19],

$$f := f(\mathbf{x}) = \frac{(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)}{L}, \tag{11}$$

$$t := t(\mathbf{x}) = \frac{1}{L}\left[\left(\frac{L}{2}\right)^2 - \left\|\mathbf{x} - \frac{\mathbf{x_2} + \mathbf{x_1}}{2}\right\|^2\right], \tag{12}$$

$$\phi := \phi(\mathbf{x}) = \sqrt{f^2 + \left(\frac{\sqrt{t^2 + f^4} - t}{2}\right)^2}, \tag{13}$$

where, two ends of the line segment: $\mathbf{x_1} = (x_1, y_1)$ and $\mathbf{x_2} = (x_2, y_2)$; L is the length of the segment; ϕ is the ADF from $\mathbf{x} = (x, y)$ to a line segment. When the solution domain is a convex polygon, we can also simplify the calculation using $\phi = f$, because the extension line of the boundary lines of the convex polygon does not pass through the domain.

In summary, the whole process of the PINNs and ADFs based "hard" method is shown in Figure 1. Utilize neural networks to fit the solution functions of equations. Following this, ADFs "hard" method is employed to transform the neural network output, in order to exactly impose Dirichlet boundaries. Subsequently, in the process of training the network, the parameters are updated via backward propagation.

2.2. BRR-PINNs

A clear evidence of the inaccuracy of the PINNs method's results is that the residuals of each equation, including the PDEs and boundary conditions, do not approach zero. If after training the neural network, all equations are very close to zero, such as 10^{-10} or even smaller, then we can confidently assert that the results are correct. However, reality is often less ideal. In practical training, satisfying boundary conditions can be challenging. For Dirichlet boundaries, the exact imposition can be achieved by the "hard" method as described in 2.1. However, for other types of boundary conditions, such as Neumann or Robin boundaries, a "soft" manner of enforcement is still applied. Two of the possible reasons for the difficulty in the training of loss function are as follows:

1. The loss function consists of multiple terms, as shown in (4). During training, there is a situation where a boundary condition with excessively fast training speeds causes the entire process to converge to a wrong local

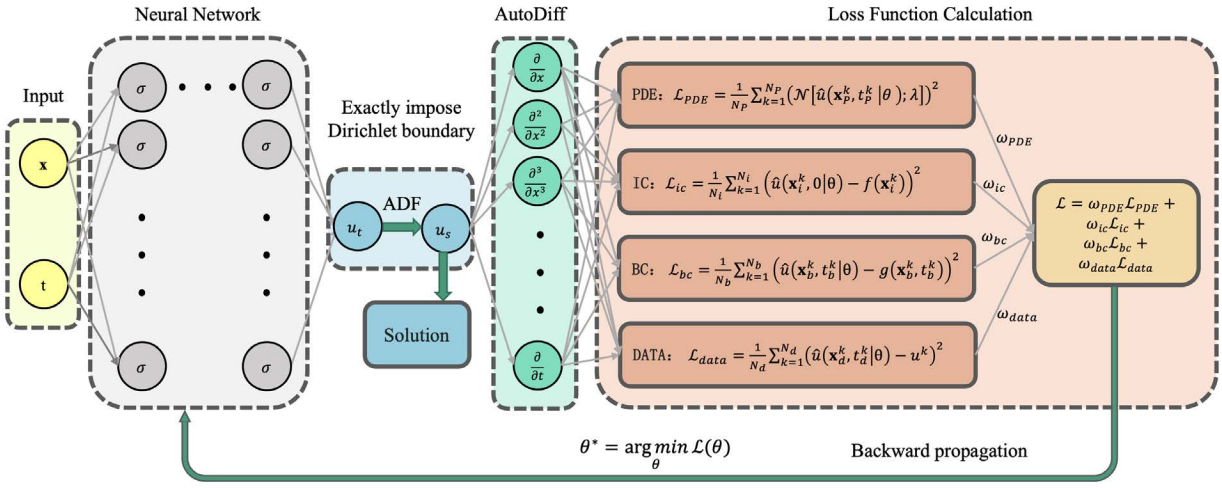


Figure 1: PINNs computational framework and ADFs based "hard" method

minimum, thus preventing other boundary conditions from being satisfied. In relatively simple problems like heat transfer, this issue can be addressed using multi-task learning (MTL) methods. However, in problems with a larger number of boundary conditions and complex equations that exist with significant coupling between them, such as in deformation problems, experimental evidence shows the failure of MTL methods.

2. The satisfaction of boundary condition equations not only relies on points located on the boundary but also involves the influence of interior points. When the output values of the interior points are poorly trained, it can easily affect the satisfaction of the boundary conditions. Moreover, many boundary condition equations are often differential equations, involving gradient terms or even higher-order differentials, which are more susceptible to the influence of interior points. Therefore, achieving the satisfaction of the boundary condition requires a coordinated interaction between the points on the boundary and the interior points, which is challenging in relatively complex problems.

To address the issues identified in the analysis above, a new algorithm is proposed called Boundary Region Reinforcement (BRR). The core of this algorithm involves reducing redundant terms in the loss function and reinforcing the boundary and the nearby interior region. We achieve the former by implementing the "hard" method at Dirichlet boundaries; and achieve the latter by assigning the boundary and nearby collocation points larger weight coefficients during training, thereby prioritizing their learning process.

The first step of the BRR-PINNs algorithm is to formulate PDEs that describe the physical problem. The governing equations represent physical equilibrium, such as thermal equilibrium or force balance. Once the governing equations are established, we identify the target variables to be solved. The number of these variables is typically equal to the number of independent governing equations to ensure that the system is mathematically determined. These target variables governing the physical system (e.g., displacement in solid mechanics), termed as *primitive output variables*, serve as the primary outputs of the neural network. Step 2 aims to deal with boundary conditions. Dirichlet boundary conditions for primitive variables can be directly imposed exactly by the "hard" method. For other types of boundary conditions, *intermediate variables*, which are added to the outputs of the network, are defined to transform them into Dirichlet boundaries. This process generates a set of *connecting equations*, denoted as f_{cnc}, between intermediate and primitive variables. The details of this transformation will be explained later. In step 3, we reinforce the boundary region at these connecting equations, using the boundary region reinforcement function (BRRF), which will be defined below. The complete algorithm flow is depicted in Algorithm 1.

Here, we elaborate on the process of imposing the boundary conditions (Step 2) in detail. Assume a boundary condition, which is not Dirichlet boundary regarding the primitive output variable, $g(\mathbf{u_p}, \mathbf{x}) = 0$, where $\mathbf{u_p}$ is the network output and $\mathbf{x}$ is the spatial coordinate of the network input. We define an intermediate variable $q = g(\mathbf{u_p}, \mathbf{x})$,

Algorithm 1: Boundary Region Reinforcement Physics-Informed Neural Networks (BRR-PINNs)

Data: Collocation points from the solution domain: $\boldsymbol{x}$

Step 1: Formulate the governing PDEs and identify the *primitive output variables* $\boldsymbol{u_p}$.

Step 2: Handle boundary conditions as follows:

  for each primitive output variable u_p^i in $\boldsymbol{u_p}$ **do**

    for each Dirichlet boundary condition for u_p^i **do**

      Compute the distance function ϕ_j;

    end

    Calculate the approximate distance function (ADF): $\phi(\boldsymbol{x}) = \left(\sum_{j=1}^{k} \frac{1}{\phi_j^m}\right)^{-\frac{1}{m}}$;

    Give a particular solution u_{par}^i for u_p^i;

    Impose Dirichlet boundary conditions for u_p^i: $u_p^i = u_{\text{par}}^i + \phi\, u_p^i$;

  end

  Define the set of *intermediate output variables* $\boldsymbol{q}$ for non-Dirichlet boundary conditions;

  Construct the corresponding set of *connecting equations* f_{cnc}.

  for each intermediate output variable q_i in $\boldsymbol{q}$ **do**

    Impose Dirichlet boundary conditions for q_i using the "hard" method;

  end

Step 3: Reinforce the boundary region

$$\mathcal{L}_{\text{cnc}}^i = \omega_{\text{cnc}}^i \,\text{mean}\Big(\left(f_{\text{cnc}}^i(\boldsymbol{x}) \cdot \text{BRRF}(\boldsymbol{x})\right)^2 \Big),$$

Compute the total loss function: $\mathcal{L} = \sum \mathcal{L}_{\text{PDE}}^j + \sum \mathcal{L}_{\text{cnc}}^i$;

Train the network parameters.

return $\boldsymbol{\theta}$

Result: Trained network parameters $\boldsymbol{\theta}$ (i.e., the solution)

therefore, the boundary conditions can be transformed into Dirichlet boundary conditions for the intermediate variable q. By also treating q as an output of the network, we can impose Dirichlet boundary conditions exactly on q. However, since an additional output variable q is introduced, a connecting term needs to be added to the loss function, which describes the relationship between the intermediate output variable q and the primitive output variables u_p as shown in (14),

$$f_{\text{cnc}}(\mathbf{x}) = q - g(\mathbf{u_p}, \mathbf{x}) = 0. \tag{14}$$

This equation holds throughout the domain. Due to the "hard" imposition of the Dirichlet boundary condition on q, q constantly equals zero at this boundary.

This process enables the transfer of boundary conditions to the entire domain, where the boundary region reinforcement can subsequently be applied. And this is why we define the intermediate variables and the connecting equations. Several implementation examples of this process are provided in Appendix B.

The ability of the algorithm to reduce redundant terms in loss function lies in defining intermediate variables to eliminate redundant boundary conditions. For example, the conditions on two different boundaries might be $\partial f/\partial x = 0$ and $\partial f/\partial x = 1$, respectively. By defining the intermediate variable $q = \partial f/\partial x$ (connecting equation), these conditions are transformed into Dirichlet boundary conditions on both boundaries. Then, by giving the particular solution q_{par}, the conditions are exactly imposed, converting two boundary conditions into one connecting equation. In another solid mechanics cases, where the shear stress τ on all free surfaces is zero, the intermediate variable can be defined as the shear stress τ, which takes a value of zero on all free surfaces, thus reducing multiple boundary conditions to a single connecting equation.

Another core aspect of the algorithm lies in boundary region reinforcement (Step 3) through the use of the BRRF function, which enables faster convergence and more accurate computations. The reinforcement of the boundary and

the nearby interior region is achieved by reconstructing the ADFs shown in (9). The ADF ϕ is equal to 0 at the boundaries, and as the distance from the boundaries increases, ϕ gradually increases. Therefore, as shown in (15), the ADF is exponentially transformed such that it reaches its maximum at the boundaries and decreases as the distance from the boundaries increases. We refer to this function as the Boundary Region Reinforcement Function (BRRF).

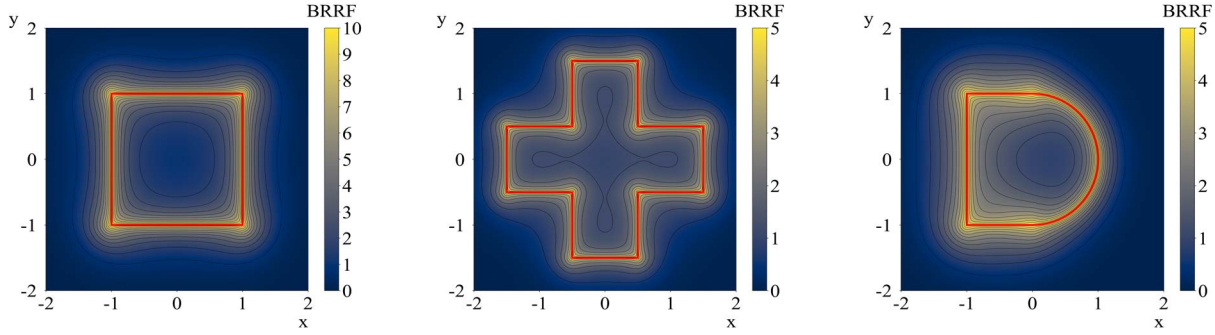
$$\mathrm{BRRF}(\mathbf{x}) = \beta \cdot \exp\left(-\alpha \cdot \mathrm{ADF}(\mathbf{x})\right), \tag{15}$$

where, β and α are adjustable parameters, and $\mathbf{x}$ is the spatial coordinates. And then, let the reinforcement coefficient at the center of the solution domain be 1, meaning no reinforcement is applied. Obtain the value of α and substitute it into (15) to obtain the complete expression of BRRF, as shown in (16),

$$\mathrm{BRRF}(\mathbf{x}) = \beta \cdot \exp\left(\frac{-\ln\beta \cdot \mathrm{ADF}(\mathbf{x})}{\mathrm{ADF}(\mathbf{x}_c)}\right), \tag{16}$$

where, $\mathbf{x}_c$ is the coordinate of the center of the solution domain. The feature of BRRF is that at the boundaries, BRRF equals β; at the center of the domain, BRRF equals 1; as the distance from the boundaries increases, BRRF gradually decreases. Figure 2 shows the distribution of BRRF in domains of different shapes in two-dimensional problems, and the reinforcement coefficient β can be adjusted. The BRRF enhances f_{cnc} by reinforce the boundary region, denoted as $f_{\mathrm{cnc}} \cdot \mathrm{BRRF}$, thus prioritize interior points around the boundaries in the computation.

In summary, the entire process described above is illustrated in Figure 3.



(a) The BRRF distribution of a convex polygon domain (square), reinforcement coefficient $\beta = 10$

(b) The BRRF distribution of a concave polygon domain, reinforcement coefficient $\beta = 5$

(c) The BRRF function distribution over a domain with curved boundaries, $\beta = 5$

Figure 2: In two-dimensional problems, the BRRF distribution over various-shaped domains with red lines indicating the boundaries, including convex polygon domain, concave polygon domain and domain with curved boundaries.

The BRR-PINNs method offers two main advantages:

1. The number of loss function terms is reduced, which leads to simplicity in adjusting the weight parameters in (4), particularly beneficial for problems with multiple and repetitive boundaries. This method enables neural network models to handle more complex geometry and a greater number of boundary conditions effectively.
2. Prioritize interior points near the boundaries by reinforcing the boundary region, leading to more accurate satisfaction of the boundary conditions.

As shown in Table 1, we compare the proposed BRR-PINNs with several existing approaches, including conventional PINNs, the "hard" method, and g-PINNs, in terms of boundary enforcement, the number of loss terms. Suppose that the number of governing PDEs is M_p; the total number of boundary conditions is M_b, among which M_d are Dirichlet-type constraints. Let M_c denote the number of distinct connecting equations. Clearly, $M_c \leq M_b - M_d$. Let N denote the number of sample points. Regarding boundary enforcement, the proposed BRR-PINNs incorporate intermediate-variable transformation and Dirichlet imposition (IVD), together with boundary region reinforcement (BRR).

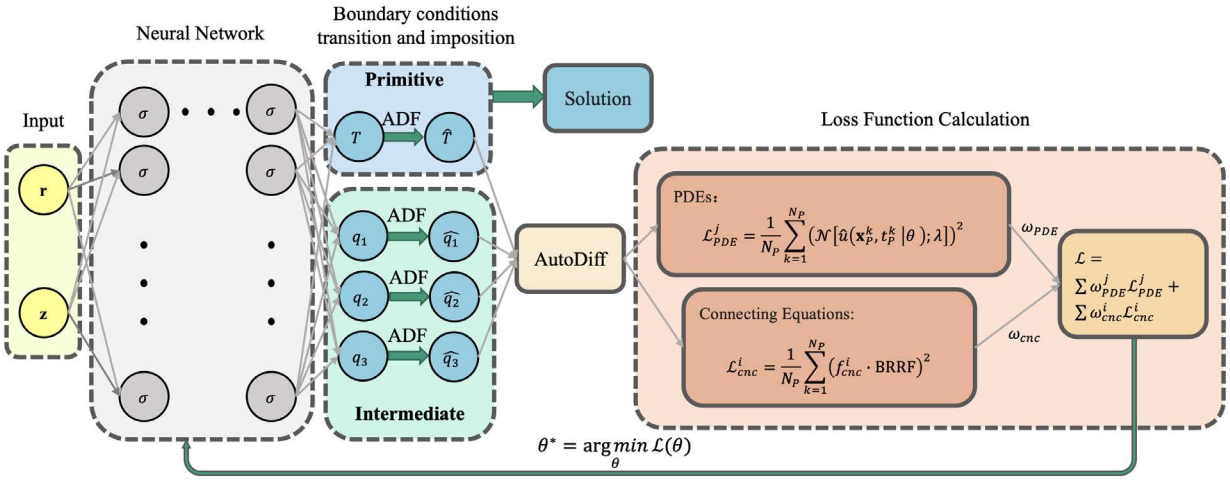


Figure 3: BRR-PINNs computational framework

Method	Boundary Enforcement	# Loss Terms
Conventional PINNs	"Soft" enforce	$M_p + M_b$
"Hard" method PINNs	"Hard" impose (Dirichlet-type)	$M_p + M_b - M_d$
g-PINNs	"Soft" enforce	$2M_p + M_b$
BRR-PINNs	**IVD+BRR**	$\boldsymbol{M_p + M_c}$

Table 1: Comparison of the boundary enforcement and the number of loss terms of different methods

3. Numerical Experiment

This section uses a two-dimensional axisymmetric thermo-elastic coupling system of a circular ring as an example to demonstrate the capabilities of BRR-PINNs. The problem is derived from the sealing ring in a non-contact mechanical seal, with boundary conditions that closely replicate practical operating environments, highlighting the method's applicability to real-world engineering problems. This problem can be divided into three parts: heat transfer, elastic deformation, and thermo-elastic coupling deformation. The detailed description and setup can be found in Appendix A. In this section, we will compare the proposed BRR-PINNs method with several existing methods, including the conventional PINNs, "Hard" method PINNs, and g-PINNs. To evaluate the computational result, a L_2 evaluation metric, as shown in (17),

$$\varepsilon_{rmse} = \sqrt{\frac{\text{mean}(\,|\,\text{Predict} - \text{Exact}\,|^2\,)}{\text{mean}(\,|\,\text{Exact}\,|^2\,)}}, \tag{17}$$

where, "Predict" is the result of PINNs, and "Exact" is the result of FEM. They are both vectors composed of the outputs at the collocation points in the solution domain.

For each method, we follow the hyperparameter-tuning procedure and the stopping criteria described in Appendix C.1. The details of the tuning process and the parameters selected for each method will be provided in Appendix C.2.

3.1. Heat Transfer Problem

The heat transfer problem that we considered is described in detail in Appendix A.1. We compare the memory usage, computational time per epoch, floating point operations (FLOPs) per epoch, and the computational accuracy

of the proposed BRR-PINNs with several existing methods in Table 2, using the experimental setup in Table C.6. As shown in the results, BRR-PINNs require memory usage and computational time comparable to conventional PINNs and "hard" methods, while clearly outperforming g-PINNs, whose higher-order derivatives lead to substantially higher computational cost. More importantly, BRR-PINNs achieve a computational accuracy on the order of $O(10^{-4})$, performing an improvement of approximately one order of magnitude over the other methods.

Method	**GPU Memory**	**Time / epoch**	**FLOPs / epoch**	ε_{rmse}	**Stop epoch**
Conventional PINNs	1.3 GB	0.46 s	2.797e10	1.5e-2	6600
"Hard" method PINNs	1.3 GB	0.56 s	2.801e10	7.1e-3	8200
g-PINNs	6.4 GB	1.33 s	1.113e11	9.9e-2	1200
BRR-PINNs (Ours)	**1.6 GB**	**0.72 s**	**3.428e10**	**6.0e-4**	**8400**

Table 2: Comparison of different methods on the heat transfer problem

As shown in Figure 4, we present the accuracy of the different methods throughout the training process. We can clearly observe the superiority of our proposed method, BRR-PINNs, which achieves an accuracy of the order of $O(10^{-4})$. The resulting temperature-rise distributions from BRR-PINNs and the absolute error with the FEM result can be found in 6. More details of the hyperparameter tuning process, selected parameters, the training process, and the resulting temperature-rise distributions for each method are provided in Appendix C.2.1.

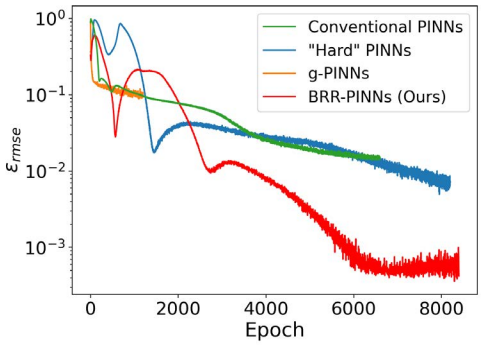


Figure 4: Accuracy comparison of different methods on heat transfer problem

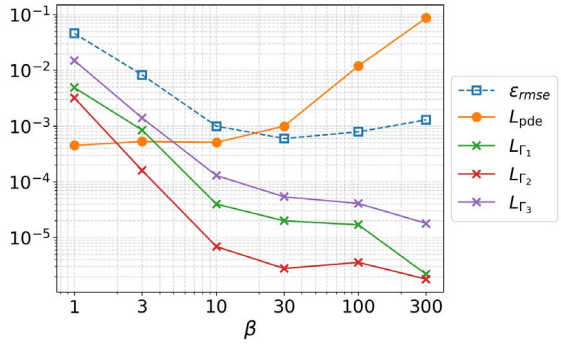
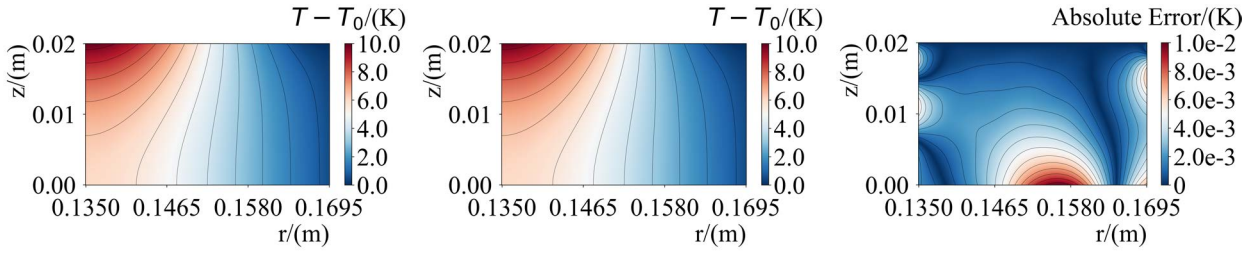


Figure 5: The effect of boundary reinforcement coefficient β in the heat transfer problem



(a) FEM (b) BRR-PINNs (c) Absolute Error

Figure 6: The result from BRR-PINNs

We further analyze the effect of the boundary reinforcement coefficient β on each loss term and the overall accuracy. As shown in Figure 5, increasing β leads to a clear decreasing trend in all loss of boundary conditions, again demonstrating that our algorithm effectively enforces the boundary conditions. However, as β increases, the loss terms

of the governing PDEs get less priority during training, resulting in an observable increase in residuals. This reveals a trade-off between boundary satisfaction and the governing equations. Consequently, the overall error ε_{rmse} first decreases and then increases as β increases. The choice of β should therefore balance these two factors: it should be large enough to significantly reduce the boundary losses, yet not so large that it severely affects the PDEs residuals.

Moreover, when $\beta = 1$, the model simply incorporates the intermediate-variable transformation and Dirichlet imposition without applying boundary region reinforcement. This serves as an ablation case for BRR-PINNs and highlights that the boundary region reinforcement is the core component of the proposed architecture.

3.2. Elastic Deformation Problem

We then consider the elastic deformation problem, described in detail in Appendix A.2. We compare our BRR-PINNs with conventional PINNs and the "Hard" method PINNs. Note that g-PINNs require a significantly larger computational graph for this problem and consequently require far more GPUs memory (larger than 40GB), beyond what is typically available on standard GPUs. Therefore, we do not include g-PINNs in our comparison.

Using the experimental setup in Table C.8, together with the hyperparameter tuning strategy and stopping criteria described in Appendix C.1, the comparison results are presented in Table 3. It is evident that our proposed BRR-PINNs outperform other methods, achieving approximately one order of magnitude improvement in accuracy without increasing too much training complexity, including GPU memory usage or training time per epoch. The accuracy curves during training for displacement in the r-direction and z-direction are shown in Figures 7 and 8, respectively. The predicted displacement maps obtained using the BRR-PINNs, along with the corresponding absolute error distributions, are shown in Figure 9. Additional details including the selected parameters and visualizations of the results from other methods are provided in Appendix C.2.2.

Method	**GPU Memory**	**Time / epoch**	**FLOPs / epoch**	$\varepsilon_{rmse}(u_r)$	$\varepsilon_{rmse}(u_z)$	**Stop epoch**
Conventional PINNs	12.13 GB	0.82 s	3.460e11	3.9e-3	1.1e-2	118k
"Hard" method PINNs	12.49 GB	0.88 s	3.464e11	3.2e-3	6.0e-3	160k
BRR-PINNs (Ours)	**13.59 GB**	**1.04 s**	**3.831e11**	**1.1e-4**	**1.7e-3**	**161k**

Table 3: Comparison of different methods on elastic deformation problem

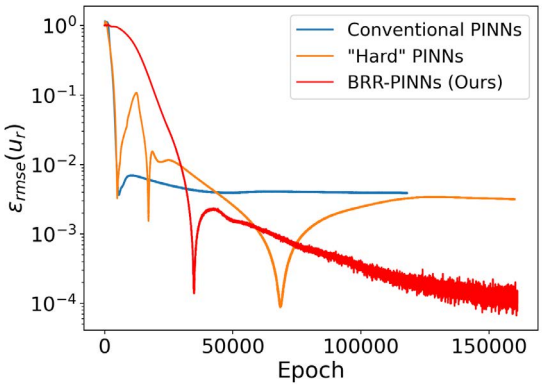


Figure 7: Accuracy comparison of different methods of r-direction displacement on elastic deformation problem

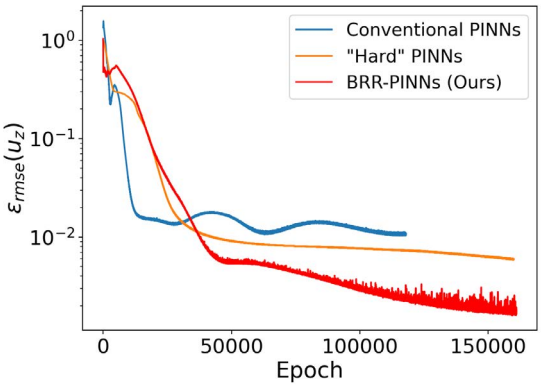
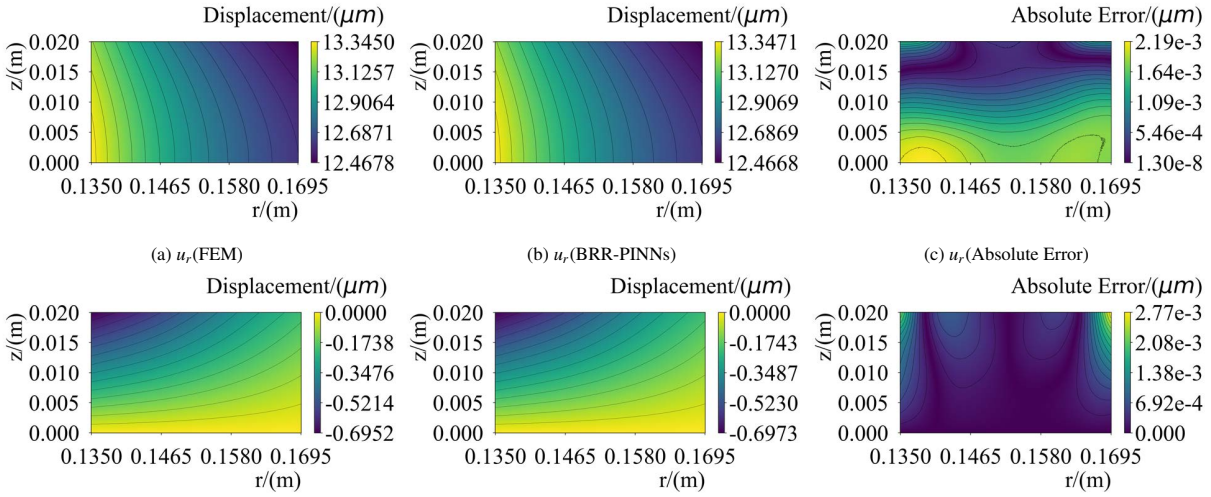


Figure 8: Accuracy comparison of different methods of z-direction displacement on elastic deformation problem

Similarly to the heat transfer problem, the boundary coefficient β exhibits a parallel behavior in the elastic deformation case. As shown in Figures 10 and 11, increasing β leads to higher loss residuals of governing PDEs while significantly reducing boundary condition errors. The overall computational errors first decrease and then increase as β grows, indicating the trade-off involved in selecting β.



(a) u_r(FEM) (b) u_r(BRR-PINNs) (c) u_r(Absolute Error)

(d) u_z(FEM) (e) u_z(BRR-PINNs) (f) u_z(Absolute Error)

Figure 9: FEM solution and BRR-PINNs computed result for the elastic deformation problem (displacement field $[u_r, u_z]$), along with the absolute error between them

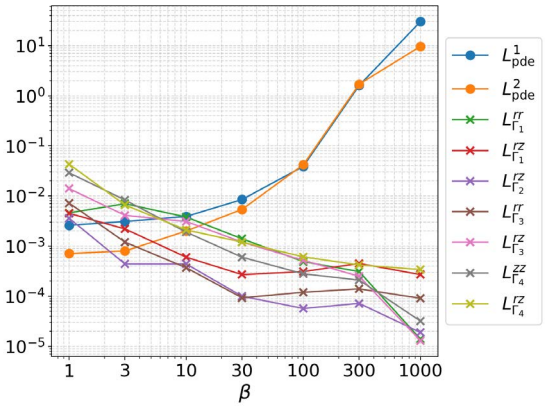


Figure 10: The effect of boundary reinforcement coefficient β on loss terms in the elastic deformation problem

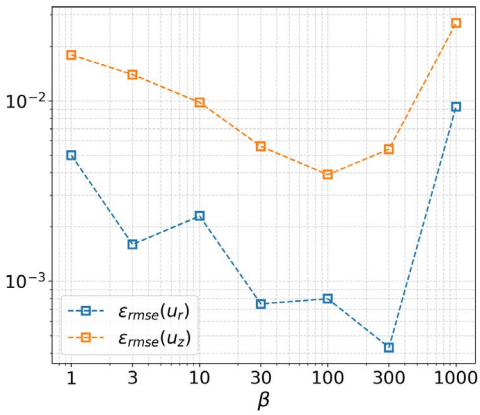


Figure 11: The effect of boundary reinforcement coefficient β on accuracy in the elastic deformation problem

3.3. Thermo-elastic Deformation Problem

Finally, we consider the thermo-elastic coupling deformation problem, described in detail in Appendix A.3. The comparison results are presented in Table 4 and Figures 12 and 13. The predicted displacement maps obtained using the BRR-PINNs are shown in Figure 14. More details can be found in Appendix C.2.3. It is clear that our proposed method is capable of accurately solving multiphysics coupling systems and performs much better than existing methods.

Method	GPU Memory	Time / epoch	FLOPs / epoch	$\varepsilon_{rmse}(u_r)$	$\varepsilon_{rmse}(u_z)$	Stop epoch
Conventional PINNs	14.38GB	0.94s	4.120e11	2.7e-3	5.3e-2	95k
"Hard" method PINNs	14.55GB	1.00s	4.124e11	1.4e-3	1.2e-2	69k
BRR-PINNs (Ours)	**14.26GB**	**1.17s**	**4.701e11**	**1.2e-4**	**1.4e-3**	**102k**

Table 4: Comparison of different methods on thermo-elastic deformation problem

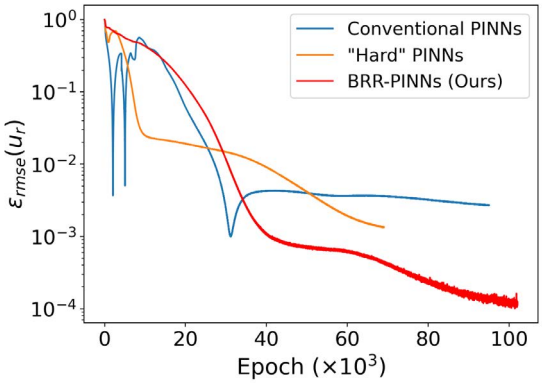


Figure 12: Accuracy comparison of different methods of r-direction displacement on thermo-elastic deformation problem

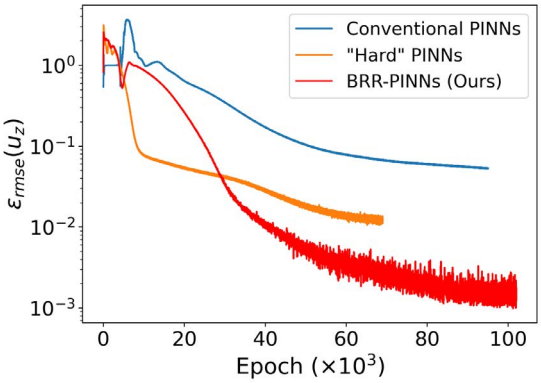
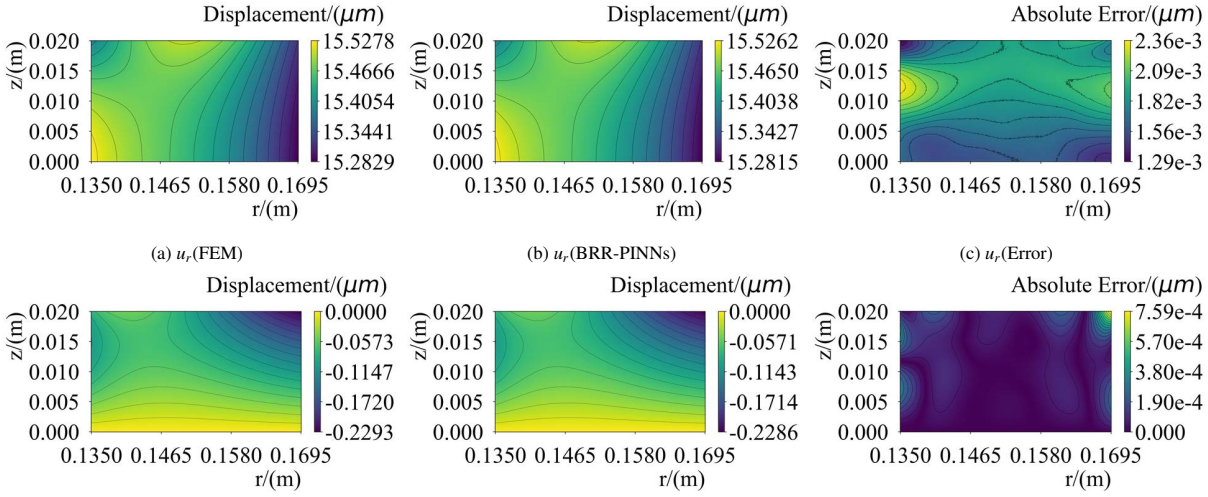


Figure 13: Accuracy comparison of different methods of z-direction displacement on thermo-elastic deformation problem



(a) u_r(FEM) (b) u_r(BRR-PINNs) (c) u_r(Error)

(d) u_z(FEM) (e) u_z(BRR-PINNs) (f) u_z(Error)

Figure 14: FEM solution and BRR-PINNs computed result for the thermo-elastic coupling deformation problem (displacement field $[u_r, u_z]$), along with the absolute error between them

4. Discussion and future work

In summary, we proposed a novel framework, BRR-PINNs, and demonstrated its ability to significantly enhance computational accuracy across three representative problems: the heat transfer problem, the elastic deformation problem, and the thermo-elastic deformation problem. Compared to several existing approaches, including conventional PINNs, "hard" method PINNs, and g-PINNs, our method achieves an improvement of approximately one order of magnitude in precision. Furthermore, we conducted a detailed analysis of the boundary reinforcement coefficient and highlighted the trade-off between reducing the governing PDE residuals and the satisfaction of the boundary conditions.

Intuitively, the failure of conventional PINNs can be attributed to the fact that, in the early stages of training, the model lacks sufficient guidance on where to converge and may become trapped in local minima, ultimately failing to satisfy all governing equations and boundary conditions. Our method mitigates this issue by introducing a framework that prioritizes boundary regions over the interior domain. Experimental results demonstrate that this strategy effectively guides the model toward more accurate solutions. Moreover, the framework naturally extends to complex geometries, as illustrated in Figure 2. These observations indicate that BRR-PINNs hold strong potential for addressing more challenging systems, including complex geometries, higher-dimensional PDEs, and nonlinear PDEs. However, further validation of BRR-PINNs on these sophisticated problems remains an important direction for future work.

Our future work will proceed in several directions. First, as mentioned above, we aim to validate the proposed framework on more complex and challenging problems. Second, we plan to develop a more rigorous mathematical explanation for the phenomena observed in this study and further improve the efficiency and accuracy of the framework. Finally, the current implementation still relies on empirically tuned parameters. An additional research direction is to incorporate multi-task learning techniques or other self-adjusting weighting strategies into the framework, while ensuring the model performance.

References

[1] 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, Journal of Computational physics 378 (2019) 686–707.

[2] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, Journal of machine learning research 18 (153) (2018) 1–43.

[3] N. Zobeiry, K. D. Humfeld, A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications, Engineering Applications of Artificial Intelligence 101 (2021) 104232.

[4] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks for heat transfer problems, Journal of Heat Transfer 143 (6) (2021) 060801.

[5] W. Ji, W. Qiu, Z. Shi, S. Pan, S. Deng, Stiff-pinn: Physics-informed neural network for stiff chemical kinetics, The Journal of Physical Chemistry A 125 (36) (2021) 8098–8106.

[6] M. De Florio, E. Schiassi, R. Furfaro, Physics-informed neural networks and functional interpolation for stiff chemical kinetics, Chaos: An Interdisciplinary Journal of Nonlinear Science 32 (6) (2022).

[7] X. Jin, S. Cai, H. Li, G. E. Karniadakis, Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations, Journal of Computational Physics 426 (2021) 109951.

[8] M. M. Almajid, M. O. Abu-Al-Saud, Prediction of porous media fluid flow using physics informed neural networks, Journal of Petroleum Science and Engineering 208 (2022) 109205.

[9] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Computer Methods in Applied Mechanics and Engineering 379 (2021) 113741.

[10] J. Bai, T. Rabczuk, A. Gupta, L. Alzubaidi, Y. Gu, A physics-informed neural network technique based on a modified loss function for computational 2d and 3d solid mechanics, Computational Mechanics 71 (3) (2023) 543–562.

[11] T. Le-Duc, S. Lee, H. Nguyen-Xuan, J. Lee, A hierarchically normalized physics-informed neural network for solving differential equations: Application for solid mechanics problems, Engineering Applications of Artificial Intelligence 133 (2024) 108400.

[12] A. Yazdani, L. Lu, M. Raissi, G. E. Karniadakis, Systems biology informed deep learning for inferring parameters and hidden dynamics, PLoS computational biology 16 (11) (2020) e1007575.

[13] T. Kadeethum, T. M. Jørgensen, H. M. Nick, Physics-informed neural networks for solving nonlinear diffusivity and biot's equations, PloS one 15 (5) (2020) e0232683.

[14] L. Yang, X. Meng, G. E. Karniadakis, B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data, Journal of Computational Physics 425 (2021) 109913.

[15] P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, Y.-S. Ong, Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method, Computer Methods in Applied Mechanics and Engineering 395 (2022) 114909.

[16] J. Yu, L. Lu, X. Meng, G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse pde problems, Computer Methods in Applied Mechanics and Engineering 393 (2022) 114823.

[17] A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Computer Methods in Applied Mechanics and Engineering 365 (2020) 113028.

[18] I. Babuška, U. Banerjee, J. E. Osborn, Survey of meshless and generalized finite element methods: a unified approach, Acta Numerica 12 (2003) 1–125.

[19] N. Sukumar, A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, Computer Methods in Applied Mechanics and Engineering 389 (2022) 114333.

[20] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM Journal on Scientific Computing 43 (5) (2021) A3055–A3081.

[21] L. Lyu, K. Wu, R. Du, J. Chen, Enforcing exact boundary and initial conditions in the deep mixed residual method, arXiv preprint arXiv:2008.01491 (2020).

[22] J. Chen, R. Du, K. Wu, A comparison study of deep galerkin method and deep ritz method for elliptic problems with different boundary conditions, arXiv preprint arXiv:2005.04554 (2020).

[23] S. Agmon, A. Douglis, L. Nirenberg, Estimates near the boundary for solutions of elliptic partial differential equations satisfying general boundary conditions. i, Communications on pure and applied mathematics 12 (4) (1959) 623–727.

[24] C. Rao, H. Sun, Y. Liu, Physics-informed deep learning for computational elastodynamics without labeled data, Journal of Engineering Mechanics 147 (8) (2021) 04021043.

[25] H. Sheng, C. Yang, Pfnn: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries, Journal of Computational Physics 428 (2021) 110085.

[26] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM Journal on Scientific Computing 43 (6) (2021) B1105–B1132.

[27] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Computer Methods in Applied Mechanics and Engineering 361 (2020) 112732.

[28] V. L. Rvachev, T. I. Sheiko, R-functions in boundary value problems in mechanics (1995).

[29] S. J. Anagnostopoulos, J. D. Toscano, N. Stergiopulos, G. E. Karniadakis, Residual-based attention in physics-informed neural networks, Computer Methods in Applied Mechanics and Engineering 421 (2024) 116805.

[30] A. Aygun, R. Maulik, A. Karakus, Physics-informed neural networks for mesh deformation with exact boundary enforcement, Engineering Applications of Artificial Intelligence 125 (2023) 106660.

[31] J. Wang, Y. Mo, B. Izzuddin, C.-W. Kim, Exact dirichlet boundary physics-informed neural network epinn for solid mechanics, Computer Methods in Applied Mechanics and Engineering 414 (2023) 116184.

[32] K. A. Luong, T. Le-Duc, J. Lee, Deep reduced-order least-square method—a parallel neural network structure for solving beam problems, Thin-Walled Structures 191 (2023) 111044.

[33] A. Harandi, A. Moeineddin, M. Kaliske, S. Reese, S. Rezaei, Mixed formulation of physics-informed neural networks for thermo-mechanically coupled systems and heterogeneous domains, International Journal for Numerical Methods in Engineering 125 (4) (2024) e7388.

[34] M. Raj, P. Kumbhar, R. K. Annabattula, Physics-informed neural networks for solving thermo-mechanics problems of functionally graded material, arXiv preprint arXiv:2111.10751 (2021).

[35] J. Xie, Z. Chai, L. Xu, X. Ren, S. Liu, X. Chen, 3d temperature field prediction in direct energy deposition of metals using physics informed neural network, The International Journal of Advanced Manufacturing Technology 119 (5) (2022) 3449–3468.

[36] R. Laubscher, Simulation of multi-species flow and heat transfer using physics-informed neural networks, Physics of Fluids 33 (8) (2021).

[37] R. Laubscher, P. Rousseau, Application of a mixed variable physics-informed neural network to solve the incompressible steady-state and transient mass, momentum, and energy conservation equations for flow over in-line heated tubes, Applied Soft Computing 114 (2022) 108050.

[38] N. Masclans, F. Vázquez-Novoa, M. Bernades, R. M. Badia, L. Jofre, Thermodynamics-informed neural network for recovering supercritical fluid thermophysical information from turbulent velocity data, International Journal of Thermofluids 20 (2023) 100448.

[39] Y. Ma, X. Xu, S. Yan, Z. Ren, A preliminary study on the resolution of electro-thermal multi-physics coupling problem using physics-informed neural network (pinn), Algorithms 15 (2) (2022) 53.

[40] M. Baldan, P. Di Barba, Energy-based pinns for solving coupled field problems: Concepts and application to the multi-objective optimal design of an induction heater, IET Science, Measurement & Technology (2024).

[41] S. A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, R. Vaziri, Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture, Computer Methods in Applied Mechanics and Engineering 384 (2021) 113959.

[42] T. Praditia, T. Walser, S. Oladyshkin, W. Nowak, Improving thermochemical energy storage dynamics forecast with physics-inspired neural network architecture, Energies 13 (15) (2020) 3873.

[43] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Advances in neural information processing systems 34 (2021) 26548–26560.

[44] A. New, B. Eng, A. C. Timm, A. S. Gearhart, Tunable complexity benchmarks for evaluating physics-informed neural networks on coupled ordinary differential equations, in: 2023 57th Annual Conference on Information Sciences and Systems (CISS), IEEE, 2023, pp. 1–8.

[45] R. Pu, X. Feng, Physics-informed neural networks for solving coupled stokes–darcy equation, Entropy 24 (8) (2022) 1106.

[46] M. Khadijeh, V. Cerqueglini, C. Kasbergen, S. Erkens, A. Varveri, Multistage physics informed neural network for solving coupled multiphysics problems in material degradation and fluid dynamics, Engineering with Computers (2025) 1–31.

[47] S. Wang, X. Yu, P. Perdikaris, When and why pinns fail to train: A neural tangent kernel perspective, Journal of Computational Physics 449 (2022) 110768.

[48] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theoretical and Applied Fracture Mechanics 106 (2020) 102447.

[49] S. Rezaei, A. Harandi, A. Moeineddin, B.-X. Xu, S. Reese, A mixed formulation for physics-informed neural networks as a potential solver for engineering problems in heterogeneous domains: Comparison with finite element method, Computer Methods in Applied Mechanics and Engineering 401 (2022) 115616.

[50] S. Shi, D. Liu, Z. Huo, Simulation of thermoelastic coupling in silicon single crystal growth based on alternate two-stage physics-informed neural network, Engineering Applications of Artificial Intelligence 123 (2023) 106468.

[51] K. Eshkofti, S. M. Hosseini, A gradient-enhanced physics-informed neural network (gpinn) scheme for the coupled non-fickian/non-fourierian diffusion-thermoelasticity analysis: A novel gpinn structure, Engineering Applications of Artificial Intelligence 126 (2023) 106908.

[52] S. Kathane, S. Karagadde, A physics informed neural network (pinn) methodology for coupled moving boundary pdes, arXiv preprint arXiv:2409.10910 (2024).

[53] S. Berrone, C. Canuto, M. Pintore, N. Sukumar, Enforcing dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks, Heliyon 9 (8) (2023).

Appendix A. Problem Setup

In this paper, we consider the heat transfer and deformation problem of a two-dimensional axisymmetric circular ring, with symmetry in the angular direction θ. The physical parameters of the ring are listed in Table A.5.

Physical parameters	
Inner diameter r_1	0.135 m
External diameter r_2	0.1695 m
Height h_1	0.02 m
Poisson ratio μ	0.14
Convective heat transfer coefficient h	2.06e4 $W/(m^2 \cdot K)$
Heat conductivity coefficient k	60 $W/(m \cdot K)$
External temperature T_0	303.15 K
Maximum temperature rise T_m	10 K
Elasticity modulus E	420 GPa
Low pressure P_l	1 MPa
High pressure P_h	10 MPa
Linear thermal expansion coefficient	4e-6 K^{-1}

Table A.5: Physical parameters for the two-dimensional axisymmetric circular ring

Appendix A.1. Heat Transfer Problem

The heat transfer equation in cylindrical coordinates (r, z, θ) is formulated as shown in (A.1),

$$\frac{\partial T}{\partial t} = k\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} + \frac{\partial^2 T}{\partial z^2}\right), \tag{A.1}$$

where, T represents temperature; t denotes the time variable; and k is the thermal conductivity coefficient. With the inclusion of steady-state and symmetry conditions, the heat transfer equation is formulated as shown in (A.2)

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{\partial^2 T}{\partial z^2} = 0. \tag{A.2}$$

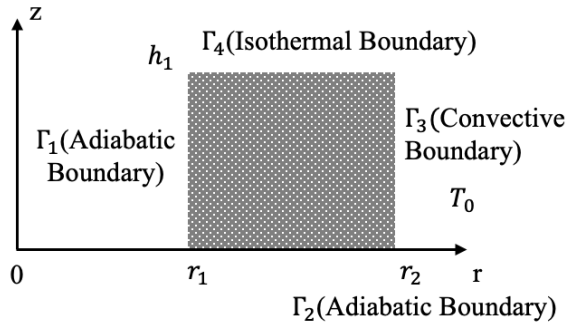


Figure A.15: The boundary conditions of the heat transfer problem

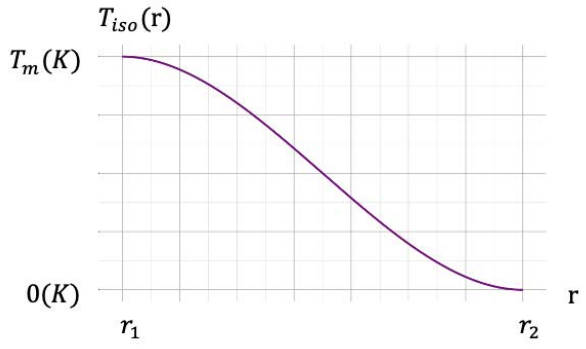


Figure A.16: The temperature rise distribution for the isothermal boundary

The boundary conditions for the heat transfer problem are shown in Figure A.15. The shaded area represents a cross-section of the circular ring, which is exposed to an environment at a temperature of T_0. The inner diameter boundary Γ_1 and the bottom boundary Γ_2 are adiabatic boundaries, the outer diameter boundary Γ_3 is subject to

convective heat transfer and the upper boundary Γ_4, an isothermal boundary, is maintained at a constant temperature. The boundary conditions for these four boundaries are given by (A.3)-(A.6),

$$\frac{\partial T}{\partial r} = 0,\ (r,z) \in \Gamma_1 \tag{A.3}$$

$$\frac{\partial T}{\partial z} = 0,\ (r,z) \in \Gamma_2 \tag{A.4}$$

$$h\,(T - T_0) + k\frac{\partial T}{\partial r} = 0,\ (r,z) \in \Gamma_3 \tag{A.5}$$

$$T - T_0 - T_{\text{iso}}\,(r) = 0,\ (r,z) \in \Gamma_4 \tag{A.6}$$

where, h is the coefficient of convective heat transfer, k is the thermal conductivity coefficient, and T_{iso} is the distribution of constant temperature rise on the boundary Γ_4, and is shown in Figure A.16. In this paper, T_{iso} is constructed by a function such that it satisfies the boundary conditions of the inner and outer diameters simultaneously at both ends. The construction method is shown in (A.7),

$$T_{\text{iso}}(r) = T_m\left(n\left(\frac{r - r_1}{r_2 - r_1}\right)^{n+1} - (n+1)\left(\frac{r - r_1}{r_2 - r_1}\right)^{n} + 1\right), \tag{A.7}$$

where, T_m is the maximum temperature rise; order n is used to control the shape of the curve. In this work, we choose $T_m = 10K$ and $n = 2$. Other physical parameters can be found in Table A.5.

Appendix A.2. Elastic Deformation Problem

Next, we consider the elastic deformation for the two-dimensional axisymmetric circular ring. The problem exhibits symmetry in the angular direction, with shear stresses in the angular direction being zero, i.e., $\sigma_{\theta r} = \sigma_{\theta z} = \sigma_{r\theta} = \sigma_{z\theta} = 0$. The equilibrium equations for the infinitesimal element in cylindrical coordinates (r, z) are given by (A.8)-(A.9),

$$r\frac{\partial \sigma_{rr}}{\partial r} + r\frac{\partial \sigma_{zr}}{\partial z} + \sigma_{rr} - \sigma_{\theta\theta} = 0, \tag{A.8}$$

$$r\frac{\partial \sigma_{zz}}{\partial z} + r\frac{\partial \sigma_{rz}}{\partial r} + \sigma_{rz} = 0, \tag{A.9}$$

where, according to the reciprocal theorem of shear stress, $\sigma_{rz} = \sigma_{zr}$. The relationship between the stress tensor and displacements for linear elastic isotropic materials is given by (A.10)-(A.13),

$$\sigma_{rr} = 2G\left[\frac{1-\mu}{1-2\mu}\frac{\partial u_r}{\partial r} + \frac{\mu}{1-2\mu}\left(\frac{u_r}{r} + \frac{\partial u_z}{\partial z}\right)\right], \tag{A.10}$$

$$\sigma_{\theta\theta} = 2G\left[\frac{1-\mu}{1-2\mu}\frac{u_r}{r} + \frac{\mu}{1-2\mu}\left(\frac{\partial u_z}{\partial z} + \frac{\partial u_r}{\partial r}\right)\right], \tag{A.11}$$

$$\sigma_{zz} = 2G\left[\frac{1-\mu}{1-2\mu}\frac{\partial u_z}{\partial z} + \frac{\mu}{1-2\mu}\left(\frac{\partial u_r}{\partial r} + \frac{u_r}{r}\right)\right], \tag{A.12}$$

$$\sigma_{rz} = G\left(\frac{\partial u_z}{\partial r} + \frac{\partial u_r}{\partial z}\right), \tag{A.13}$$

where, u_r represents the displacement in r direction; u_z is the displacement in z direction; μ is the Poisson's ratio of the material; G is the shear modulus, given by $G = \frac{E}{2(1+\mu)}$, where E is the Young's modulus of the material. These parameters value can be found in Table A.5. Equations A.8-A.13 can fully describe the elastic deformation, and the output variables include displacement field $[u_r, u_z]^T$ and stress tensor $[\sigma_{rr}, \sigma_{\theta\theta}, \sigma_{zz}, \sigma_{rz}]^T$. The boundary conditions for elastic deformation problem are shown in Figure A.17. Boundaries Γ_1, Γ_3 and Γ_4 are imposed pressure perpendicular to their surfaces. Boundary Γ_1 has high pressure, $P_h = 10MPa$; boundary Γ_3 has low pressure, $P_l = 1MPa$; and boundary Γ_4 has a varying pressure distribution as shown in Figure A.18. Since the pressure acts perpendicularly to

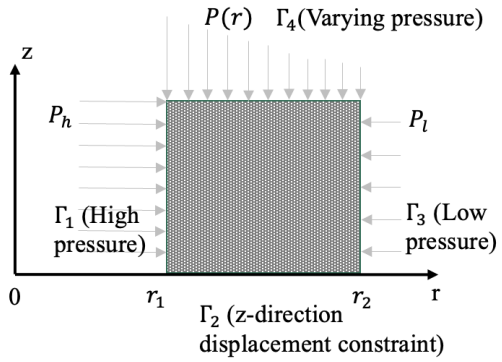


Figure A.17: The boundary conditions of the elastic deformation problem

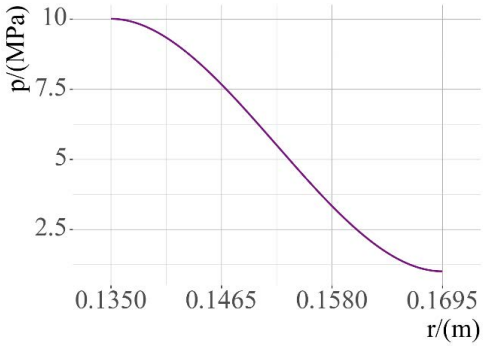


Figure A.18: The varying pressure distribution for the Γ_4 boundary

the surface, there are no components along the surface direction. Boundary Γ_2 is imposed a displacement constraint in z-direction, and similarly, there are no stress components along the surface. On these four boundaries, the stress component $\sigma_{rz} = 0$. The boundary condition equations for the four boundaries are shown in (A.14)-(A.17),

$$\sigma_{rr} = P_h;\ \sigma_{rz} = 0,\ \ (r,z) \in \Gamma_1, \tag{A.14}$$

$$u_z = 0;\ \sigma_{rz} = 0,\ \ (r,z) \in \Gamma_2, \tag{A.15}$$

$$\sigma_{rr} = P_l;\ \sigma_{rz} = 0,\ \ (r,z) \in \Gamma_3, \tag{A.16}$$

$$\sigma_{zz} = P(r);\ \sigma_{rz} = 0,\ \ (r,z) \in \Gamma_4. \tag{A.17}$$

Appendix A.3. Thermo-elastic Deformation Problem

The thermo-elastic deformation equations for a two-dimensional axisymmetric circular ring are analogous to the elastic deformation equations. The equilibrium equations are same as those of elastic deformation, as shown in (A.8)-(A.9). The expression for the stress tensor needs to incorporate the thermal expansion effect, as illustrated in (A.18)-(A.21),

$$\sigma_{rr} = 2G\left[\frac{1-\mu}{1-2\mu}\frac{\partial u_r}{\partial r} + \frac{\mu}{1-2\mu}\left(\frac{u_r}{r} + \frac{\partial u_z}{\partial z}\right)\right] - \gamma\Delta T, \tag{A.18}$$

$$\sigma_{\theta\theta} = 2G\left[\frac{1-\mu}{1-2\mu}\frac{u_r}{r} + \frac{\mu}{1-2\mu}\left(\frac{\partial u_z}{\partial z} + \frac{\partial u_r}{\partial r}\right)\right] - \gamma\Delta T, \tag{A.19}$$

$$\sigma_{zz} = 2G\left[\frac{1-\mu}{1-2\mu}\frac{\partial u_z}{\partial z} + \frac{\mu}{1-2\mu}\left(\frac{\partial u_r}{\partial r} + \frac{u_r}{r}\right)\right] - \gamma\Delta T, \tag{A.20}$$

$$\sigma_{rz} = G\left(\frac{\partial u_z}{\partial r} + \frac{\partial u_r}{\partial z}\right), \tag{A.21}$$

where, γ is the thermal stress coefficient, $\gamma = \frac{\alpha E}{1-2\mu}$, where α is the linear thermal expansion coefficient; ΔT is the temperature rise. It should be noted that ΔT is not a constant; it varies with coordinates (r, z). Therefore, when substituting it into the equilibrium equation, $\frac{\partial \Delta T}{\partial r} \neq 0$. The boundary conditions imposed for thermo-elastic coupling deformation problem are totally the same as elastic deformation, shown in section **??**.

Appendix B. BRR-PINNs Implementation Details

In this section, we provide details on how to implement the BRR-PINNs method, using the example problems defined in Appendix A.

Appendix B.1. Heat Transfer Problem Implementation

The first step of the BRR-PINNs algorithm is to formulate the governing PDEs and determine the primitive output variables. As shown in A.2, there is only one equation and the single primitive variable is the temperature T.

Step 2 addresses the boundary conditions given in (A.3)–(A.6). Among them, only (A.6) is a Dirichlet condition, which can be directly enforced using the "hard" method described in Section 2.1. For the other three boundary conditions, we introduce three intermediate output variables:

$$q_1 = \frac{\partial T}{\partial r}; \quad q_2 = \frac{\partial T}{\partial z}; \quad q_3 = h\,(T - T_0) + k\frac{\partial T}{\partial r}. \tag{B.1}$$

These intermediate variables are defined over the entire domain (r, z), and their corresponding connecting equations are given as follows:

$$\begin{aligned} f^1_{\text{cnc}}(r,z) &= q_1(r,z) - \frac{\partial T(r,z)}{\partial r}, \\ f^2_{\text{cnc}}(r,z) &= q_2(r,z) - \frac{\partial T(r,z)}{\partial z}, \\ f^3_{\text{cnc}}(r,z) &= q_3(r,z) - h\,(T(r,z) - T_0) - k\frac{\partial T(r,z)}{\partial r}. \end{aligned}$$

Using these intermediate variables and connecting equations, the boundary conditions can be reformulated as:

$$q_1(r,z) = 0,\ (r,z) \in \Gamma_1, \tag{B.2}$$

$$q_2(r,z) = 0,\ (r,z) \in \Gamma_2, \tag{B.3}$$

$$q_3(r,z) = 0,\ (r,z) \in \Gamma_3, \tag{B.4}$$

which are Dirichlet boundaries with respect to the intermediate variables $[q_1, q_2, q_3]^\top$, and can therefore be directly imposed using the "hard" method.

Step 3 is to reinforce the boundary regions. The loss term corresponding to each connecting equation can be expressed as:

$$\mathcal{L}^i_{\text{cnc}} = \omega^i_{\text{cnc}}\ \text{mean}\left((f^i_{\text{cnc}} \cdot \text{BRRF})^2\right). \tag{B.5}$$

The output variables of the neural networks is $[T, q_1, q_2, q_3]^\top$, and the loss function is $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \sum_{i=1}^{3} \mathcal{L}^i_{\text{cnc}}$. The network is then trained to minimize this total loss function.

In summary, the entire process described above is illustrated in Figure 3.

Appendix B.2. Elastic Deformation Problem Implementation

The formulation of the elastic deformation problem is presented in (A.8)–(A.13). It is important to emphasize that only (A.8) and (A.9) are the governing PDEs, representing the force balance of the differential elements. In contrast, (A.10)–(A.13) are the constitutive relations that describe the relationship between stress and displacement in specific materials and are distinct from the governing equilibrium equations. Therefore, the primitive variables are u_r and u_z.

The next step addresses the eight boundary conditions presented in (A.14)–(A.17). The condition $u_z = 0$ on the boundary Γ_2 is a standard Dirichlet condition which can be "hard" imposed. The remaining conditions are not Dirichlet-type with respect to primitive variables are u_r and u_z. It is natural to define four intermediate variables, σ_{rr}, $\sigma_{\theta\theta}$, σ_{zz}, σ_{rz}, and (A.10)–(A.13) will serve as the connecting equations. Compared to conventional PINNs, whose loss function consists of $2 + 8 = 10$ terms, the BRR-PINNs method requires only 6 loss terms.

Finally, we proceed the boundary region reinforcement, which is the same as Step 3 of the heat transfer case in Appendix B.1.

Appendix C. Missing Details for Section 3

Appendix C.1. Hyperparameter Selection and Stopping Criteria

Hyperparameter Selection. The success of PINN-based methods relies on careful hyperparameter tuning, including the learning rate, the weights assigned to each loss term, the boundary reinforcement coefficient, and others. In this paper, we adopt an empirical tuning strategy. For each method-conventional PINNs, "hard" method PINNs, g-PINNs, and BRR-PINNs-we follow the tuning procedure outlined below:

- Step 1: Select a learning rate that ensures a stable and consistent decrease of the loss function.
- Step 2 (if applicable): Adjust the boundary reinforcement coefficient β.
- Step 3: Empirically adjust the weights of the individual loss terms according to their training behaviors.

Stopping Criteria. Our stopping criteria is to monitor the whole loss function L, and denote by $L(i)$ the loss value at i-th epoch. We define checkpoints at epochs

$$kW, \quad k = 1, 2, 3, \ldots$$

For each checkpoint kW with $k \geq 2$, compute the average loss over current and previous windows of length W :

$$L_{\text{cur}}(k) := \frac{1}{W} \sum_{i=(k-1)W+1}^{kW} L(i),$$

$$L_{\text{pre}}(k) := \frac{1}{W} \sum_{i=(k-2)W+1}^{(k-1)W} L(i).$$

The relative improvement between two successive windows is defined as

$$\Delta(k) := \frac{L_{\text{pre}}(k) - L_{\text{cur}}(k)}{L_{\text{cur}}(k)}.$$

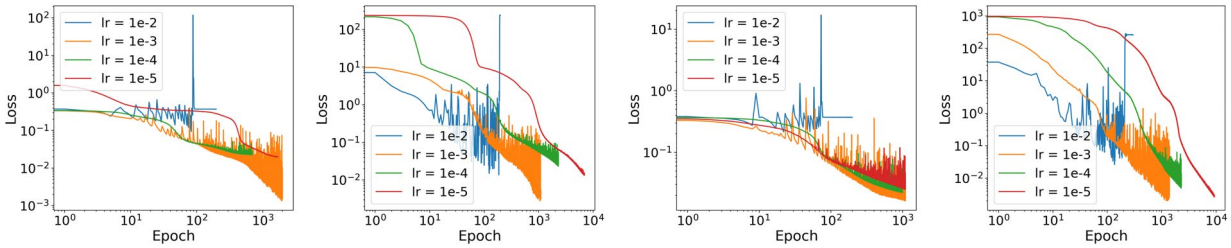
Using $L_{\text{cur}}(k)$ as the denominator makes the measure sensitive to orders of magnitude reductions. For example, when the loss decreases from 0.01 to 0.001 , the metric yields $\Delta(k) = 9$, while the decrease from 0.11 to 0.1 results in $\Delta(k) = 0.1$. We then terminate training once the relative improvement falls below a threshold, i.e., $\Delta(k) \leq \varepsilon$. In this work, we set the stopping threshold to $\varepsilon = 0.01$.

Appendix C.2. Numerical Experiment Details

Appendix C.2.1. Details for Heat Transfer Problem

Experimental Setup	
Neural network	5×64 MLP
Activation function	tanh
Optimizer	Adam
Sampling method	Random per epoch
Collocation points in domain	128×128
Collocation points per boundary	64
Batch size	1024
Checkpoint window size W	100
Stopping threshold ε	0.01
GPU	NVDIA RTX 4090

Table C.6: Experimental setup for the heat transfer problem



(a) Conventional PINNs (b) "Hard" method PINNs (c) g-PINNs (d) BRR-PINNs

Figure C.19: Learning rate tuning for each method

We follow the hyperparameter tuning strategy and the stopping criteria described in Appendix C.1. We first set all loss term weights to 1 and the boundary reinforcement coefficient β to 10, then tune the learning rate for each method. As shown in Figure C.19, we select a learning rate of 1e-5 for all methods. We also note that the oscillations observed in g-PINNs arise from higher-order differentiation rather than from the choice of learning rate.

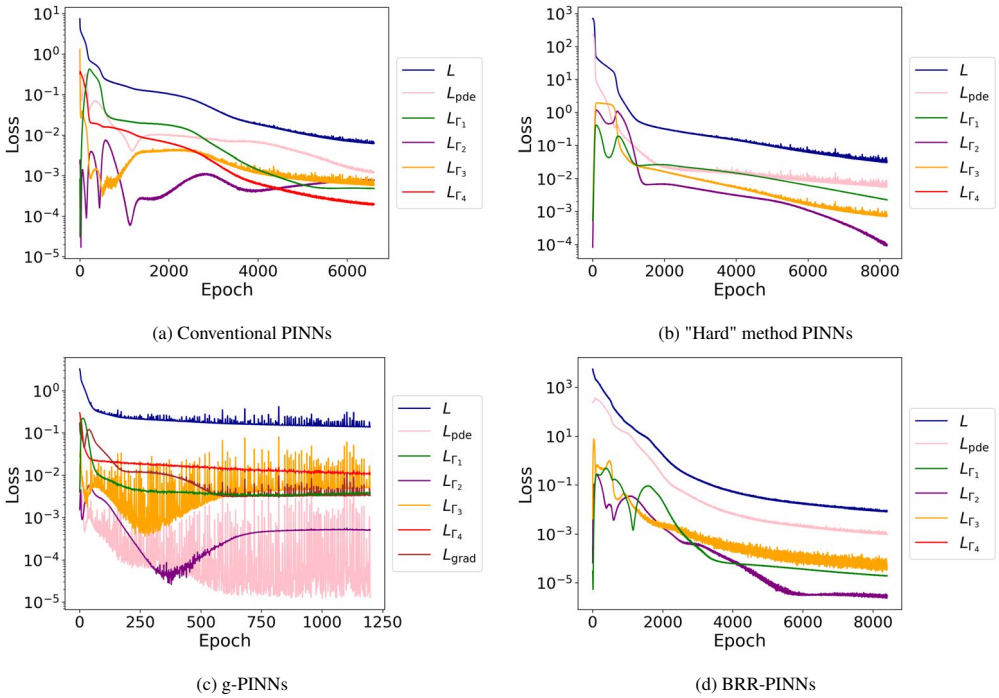
We then tune the other parameters including the loss term weights and boundary reinforce coefficient. We summarize the tuning process in the following Table C.7. The loss value we report is the average of the last $W = 100$ epochs. We note that parameter tuning and selection are based on the behavior of each individual loss term; the errors with respect to the "true" FEM solution are provided solely for reference. The entries in bold in Table C.7 are the set of selected parameters, and we provide the training process for each loss term in Figure C.20.

For a more intuitive comparison, Figure C.21 presents the predicted temperature-rise distributions alongside the absolute error with respect to the "true" FEM solution. It is clearly observed that BRR-PINNs outperform the other methods.

Method	Loss term weights	β	L	L_{pde}	L_{Γ_1}	L_{Γ_2}	L_{Γ_3}	L_{Γ_4}	L_{grad}	Stop epoch	$\varepsilon_{\text{rmse}}$
Conventional PINNs	[1,1,1,1,1,-]	-	1.9e-2	7.8e-4	1.2e-3	1.5e-4	9.5e-5	1.7e-2	-	1500	1.2e-1
	[1,1,1,1,10,-]	-	5.5e-3	1.3e-3	5.1e-4	8.1e-4	1.5e-3	1.4e-4	-	7600	1.4e-2
	[1,1,1,3,10,-]	**-**	**6.4e-3**	**1.2e-3**	**4.9e-4**	**7.9e-4**	**6.3e-4**	**2.0e-4**	**-**	**6600**	**1.5e-2**
"Hard" PINNs	[1,1,1,1,-,-]	-	2.4e-2	6.2e-3	8.9e-3	9.4e-4	7.9e-3	0	-	7400	1.2e-2
	[1,3,1,3,-,-]	-	8.9e-2	2.7e-2	8.9e-3	2.7e-3	1.1e-2	0	-	4100	1.5e-2
	[3,3,1,10,-,-]	**-**	**3.3e-2**	**6.2e-3**	**2.3e-3**	**1.0e-4**	**7.8e-4**	**0**	**-**	**8200**	**7.1e-3**
g-PINNs	[1,1,1,1,1,1]	-	2.6e-2	2.1e-4	1.9e-3	3.7e-4	3.4e-3	2.0e-2	8.9e-4	1200	1.3e-1
	[1,1,1,1,10,1]	-	1.1e-1	2.8e-4	2.3e-2	4.9e-4	2.4e-2	5.6e-3	5.9e-3	1300	7.0e-2
	[1,3,1,1,10,1]	-	1.3e-1	5.1e-4	4.2e-3	3.3e-3	2.7e-2	7.9e-3	3.5e-3	1400	8.0e-2
	[1,3,1,3,10,1]	**-**	**1.5e-1**	**3.3e-4**	**3.7e-3**	**5.2e-4**	**8.3e-3**	**1.1e-2**	**3.7e-3**	**1200**	**9.9e-2**
BRR-PINNs	[1,1,1,1]	1	2.9e-3	4.5e-4	4.9e-3	3.2e-3	1.5e-2	0	-	7700	4.6e-2
	[1,1,1,1]	10	2.8e-3	5.1e-4	4.0e-5	6.9e-6	1.3e-4	0	-	9300	1.0e-3
	[1,1,1,1]	**30**	**8.7e-3**	**1.0e-3**	**2.0e-5**	**2.8e-6**	**5.4e-5**	**0**	**-**	**8400**	**6.0e-4**
	[1,1,1,1]	100	8.5e-2	1.2e-2	1.7e-5	3.6e-6	4.1e-5	0	-	8400	7.9e-4
	[1,1,1,1]	300	2.9e-1	8.7e-2	2.2e-6	1.8e-6	1.8e-5	0	-	9700	1.3e-3
	[1,3,1,3]	**30**	1.3e-2	9.2e-4	6.3e-6	1.7e-5	3.5e-5	0	-	11700	2.2e-3
	[3,1,1,1]	**30**	1.8e-2	1.2e-3	7.7e-6	1.8e-5	1.2e-4	0	-	6100	2.0e-3
	[10,1,1,3]	**30**	5.1e-2	1.1e-3	1.5e-4	4.9e-5	8.5e-5	0	-	8900	3.2e-3

Table C.7: Comparison of memory usage, computational time, and training complexity. For the first three methods, the loss-term weights follow the order $\left[L_{\text{pde}}, L_{\Gamma_1}, \cdots, L_{\Gamma_4}, L_{\text{grad}}\right]$. For BRR-PINNs, the weights are ordered as L_{pde} and the connecting equation losses generated by boundaries $\Gamma_1, \Gamma_2, \Gamma_3$, respectively. Entries in bold highlight the chosen parameter sets along with their corresponding outcomes.

,



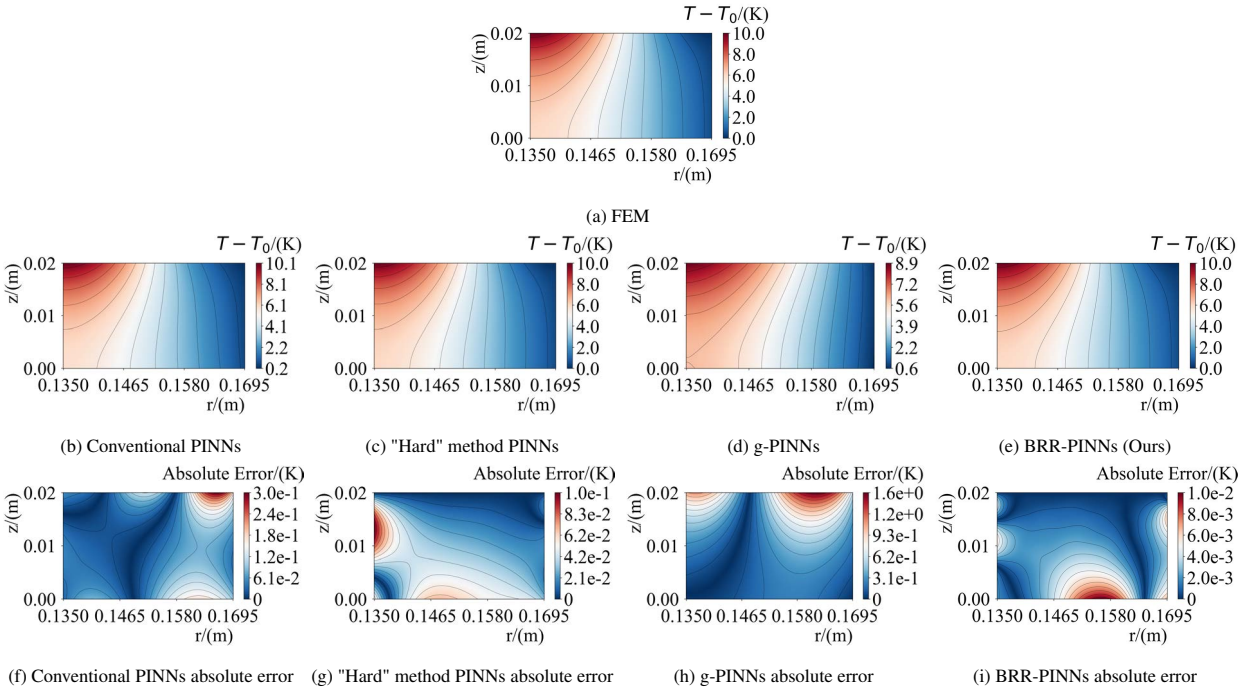
(a) Conventional PINNs

(b) "Hard" method PINNs

(c) g-PINNs

(d) BRR-PINNs

Figure C.20: The training process of the selected model for each method.



(a) FEM

(b) Conventional PINNs

(c) "Hard" method PINNs

(d) g-PINNs

(e) BRR-PINNs (Ours)

(f) Conventional PINNs absolute error

(g) "Hard" method PINNs absolute error

(h) g-PINNs absolute error

(i) BRR-PINNs absolute error

Figure C.21: Comparison of the resulting temperature-rise distributions from several existing methods and BRR-PINNs

Appendix C.2.2. Details for Elastic Deformation Problem

Experimental Setup	
Neural network	5×128 MLP
Activation function	tanh
Optimizer	Adam
Sampling method	Random per epoch
Collocation points in domain	128×128
Collocation points per boundary	128
Batch size	4096
Checkpoint window size W	1000
Stopping threshold ε	0.01
GPU	NVDIA RTX 4090

Table C.8: Experimental setup for the elastic deformation problem

Learning rate	L^1_{PDE}	L^2_{PDE}	$L_{\sigma_{rr}}$	$L_{\sigma_{\theta\theta}}$	$L_{\sigma_{zz}}$	$L_{\sigma_{rz}}$	β
1e-5	1	1	3	1	3	10	100

Table C.9: The selected weight parameters for BRR-PINNs method in the elastic deformation problem

Method	Learning rate	L^1_{PDE}	L^2_{PDE}	$L^{\sigma_{rr}}_{\Gamma_1}$	$L^{\sigma_{rz}}_{\Gamma_1}$	$L^{u_z}_{\Gamma_2}$	$L^{\sigma_{rz}}_{\Gamma_2}$	$L^{\sigma_{rr}}_{\Gamma_3}$	$L^{\sigma_{rz}}_{\Gamma_3}$	$L^{\sigma_{zz}}_{\Gamma_4}$	$L^{\sigma_{rz}}_{\Gamma_4}$
Conventional PINNs	1e-5	1	1	3	10	3	1	1	10	30	1
"Hard" Method PINNs	1e-5	1	1	10	3	-	1	1	1	30	1

Table C.10: The selected weight parameters for conventional PINNs and "hard" method PINNs in the elastic deformation problem

The predicted displacement maps alongside the absolute error with respect to the "true" FEM solution for each method are shown in Figure C.22.

Appendix C.2.3. Details for Thermo-elastic Deformation Problem

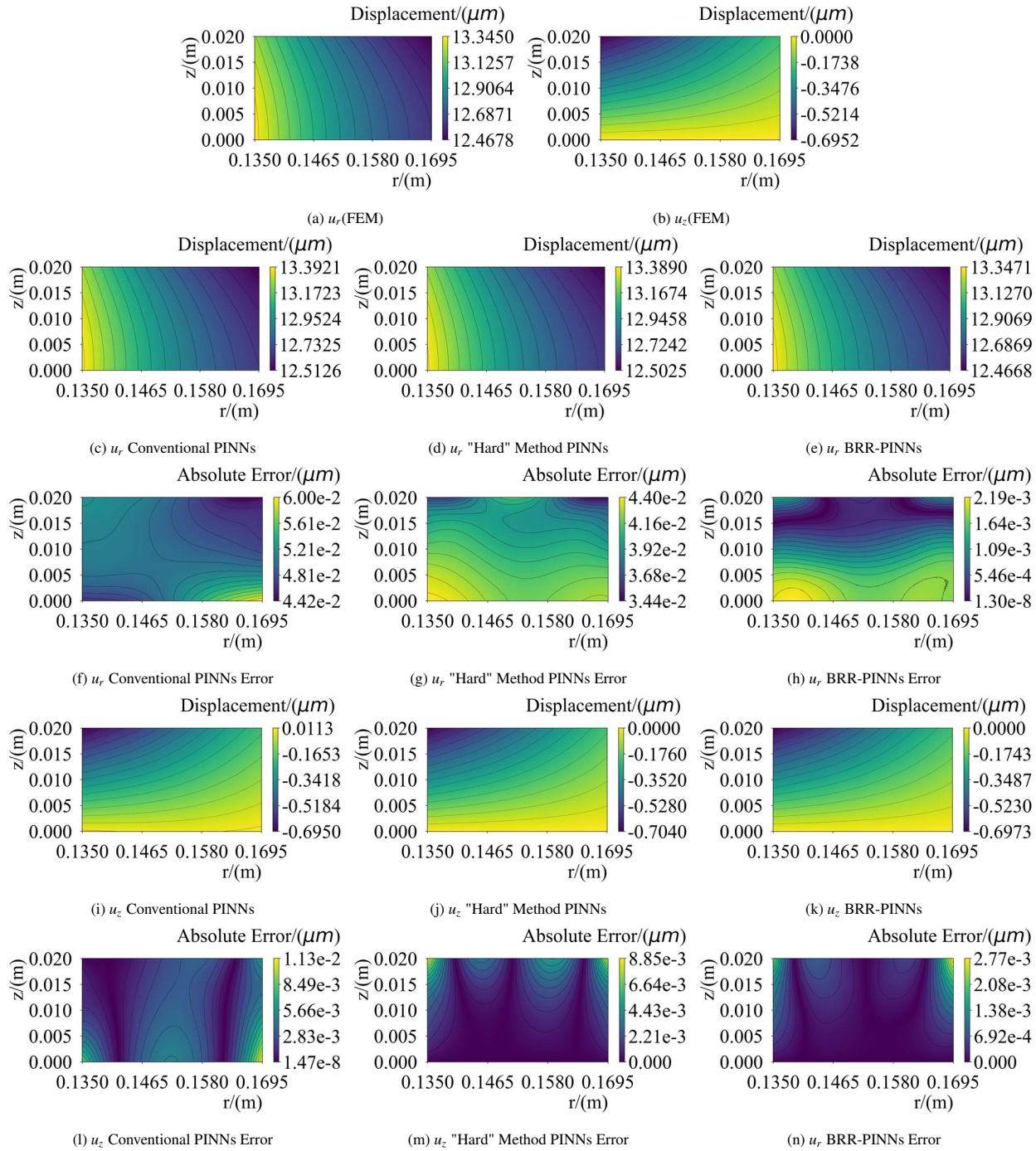
We use the same experimental setup with the case of elastic deformation problem, as shown in Table C.8. The predicted displacement and error maps are shown in Figure C.23.

Learning rate	L^1_{PDE}	L^2_{PDE}	$L_{\sigma_{rr}}$	$L_{\sigma_{\theta\theta}}$	$L_{\sigma_{zz}}$	$L_{\sigma_{rz}}$	β
1e-5	1	1	3	1	3	10	100

Table C.11: The selected weight parameters for BRR-PINNs method in the thermal-elastic deformation problem

Method	Learning rate	L^1_{PDE}	L^2_{PDE}	$L^{\sigma_{rr}}_{\Gamma_1}$	$L^{\sigma_{rz}}_{\Gamma_1}$	$L^{u_z}_{\Gamma_2}$	$L^{\sigma_{rz}}_{\Gamma_2}$	$L^{\sigma_{rr}}_{\Gamma_3}$	$L^{\sigma_{rz}}_{\Gamma_3}$	$L^{\sigma_{zz}}_{\Gamma_4}$	$L^{\sigma_{rz}}_{\Gamma_4}$
Conventional PINNs	1e-5	1	1	1	3	30	1	1	1	10	10
"Hard" Method PINNs	1e-5	1	1	1	10	-	1	1	3	30	10

Table C.12: The selected weight parameters for conventional PINNs and "hard" method PINNs in the thermo-elastic deformation problem



(a) u_r(FEM) (b) u_z(FEM)

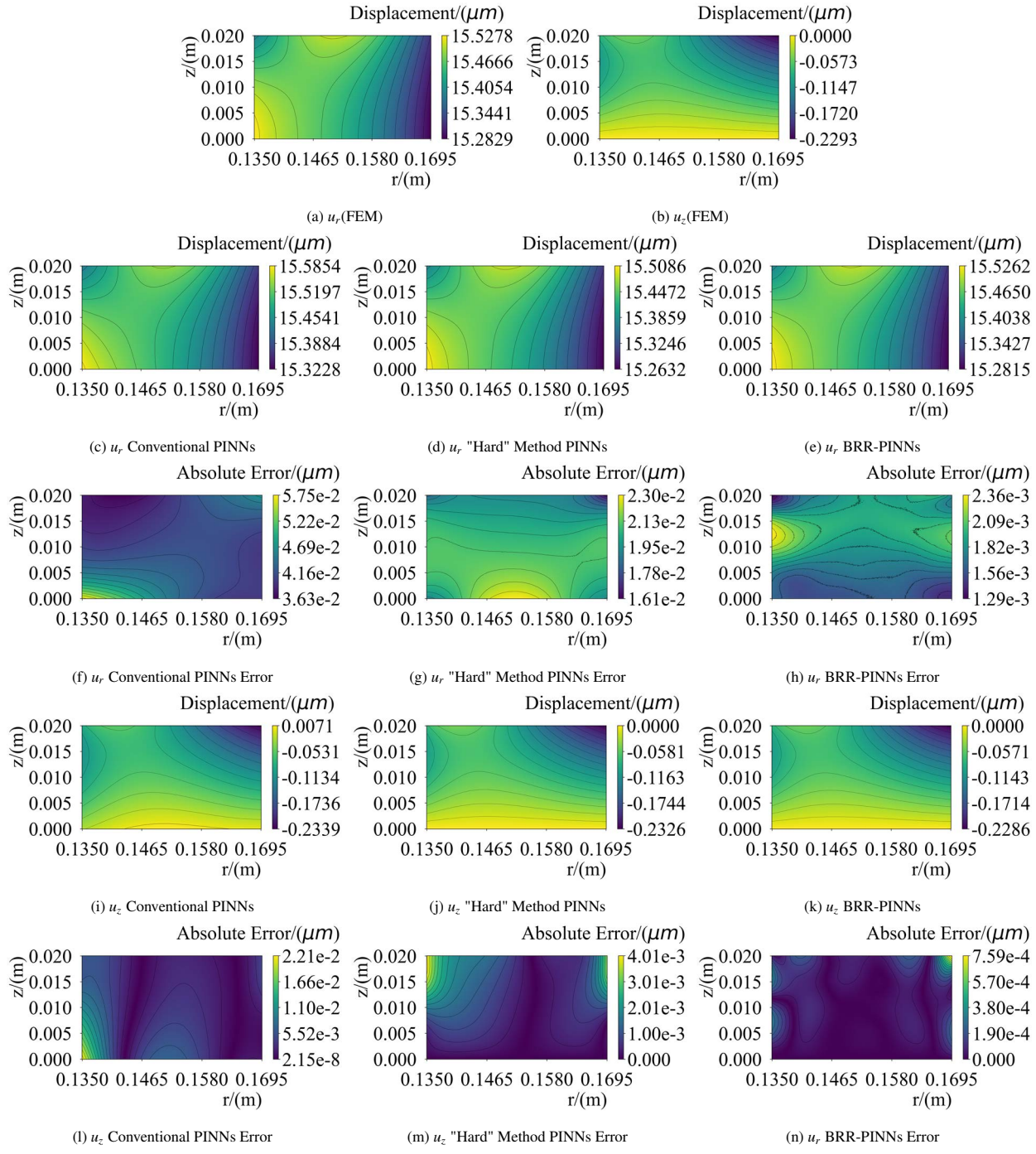
(c) u_r Conventional PINNs (d) u_r "Hard" Method PINNs (e) u_r BRR-PINNs

(f) u_r Conventional PINNs Error (g) u_r "Hard" Method PINNs Error (h) u_r BRR-PINNs Error

(i) u_z Conventional PINNs (j) u_z "Hard" Method PINNs (k) u_z BRR-PINNs

(l) u_z Conventional PINNs Error (m) u_z "Hard" Method PINNs Error (n) u_r BRR-PINNs Error

Figure C.22: Comparison of the resulting displacement maps from several existing methods and BRR-PINNs in the elastic deformation problem



(a) u_r(FEM) (b) u_z(FEM)

(c) u_r Conventional PINNs (d) u_r "Hard" Method PINNs (e) u_r BRR-PINNs

(f) u_r Conventional PINNs Error (g) u_r "Hard" Method PINNs Error (h) u_r BRR-PINNs Error

(i) u_z Conventional PINNs (j) u_z "Hard" Method PINNs (k) u_z BRR-PINNs

(l) u_z Conventional PINNs Error (m) u_z "Hard" Method PINNs Error (n) u_r BRR-PINNs Error

Figure C.23: Comparison of the resulting displacement maps from several existing methods and BRR-PINNs in the thermo-elastic deformation problem