Exploration of Geometry-Aware Physics-Informed Neural Networks for Cardiac Electrophysiology

Ingvild Askim Adde

School of Economics,
Innovation & Technology,
Kristiania University of Applied Sciences,
Norway

Cardiovascular diseases remain a leading cause of mortality worldwide. Despite advances in computational modeling, there is still a lack of efficient and accurate tools for understanding disease mechanisms at a personalized level. Physics-informed neural networks (PINNs) have emerged as a promising framework for accelerating simulations by embedding physical laws into neural networks. However, a major limitation of conventional PINNs is their poor generalizability across varying computational domains, which hinders their clinical applicability. In this project, we address this challenge by proposing two new approaches — Affine-PINN and RefAffine-PINN — that enhance the geometry-awareness of PINNs. These methods enable PINNs to generalize across different geometries without the need for retraining. Our results demonstrate improvements in predictive accuracy and generalization compared to conventional PINNs, thereby advancing the potential for real-time, patient-specific modeling in cardiac electrophysiology. All code is available at https://github.com/ingvildadde/FYS9429_project.

I. INTRODUCTION

Cardiovascular diseases (CVDs) are a leading cause of mortality worldwide, responsible for approximately 30% of global deaths (Mehra, 2007). Among these, cardiac arrhythmias account for 80% of sudden cardiac deaths (SCDs), emphasizing the need for accurate diagnosis and treatment (Dermul and Dierckx, 2024). A crucial part of optimal treatment is the identification of mechanisms responsible for arrhythmias at a personalized level (Katz, 2010; Trayanova and Boyle, 2014). Still, clinical practices largely rely on a 'one-size-fits-all' approach for diagnosis and treatment, leaving patients with sub-optimal outcomes (Arevalo *et al.*, 2016). This highlights a clear need for personalized models to enhance patient care.

Computational modeling offers a promising path toward individualized care by enabling non-invasive *in silico* testing of treatments and predictions of patient responses. Traditional numerical solvers for simulating cardiac electrical activity, such as finite element method (FEM), finite volume method (FVM), and finite difference method (FDM), accurately capture electrophysiological processes but are computationally demanding and struggle with data uncertainty and noise (Dalton *et al.*, 2023; Gao *et al.*, 2021; Karniadakis *et al.*, 2021). Hence, numerical solvers remain impractical for real-time applications in a clinical setting.

In contrast, physics-informed neural networks (PINNs) offer a data-efficient machine learning alternative by embedding physical laws directly into the neural network via partial differential equation (PDE) residuals (Raissi *et al.*, 2019). This integration reduces the risk of overfitting, enhances robustness to noisy data, and enables PINNs to address inverse and ill-posed problems more effectively (Karniadakis *et al.*, 2021). PINNs have shown promise in cardiovascular applications such as estimating electrophysiology (EP) parameters (Herrero Martin *et al.*, 2022), conduction velocity maps (Sahli Costabal *et al.*, 2020), fiber orientation, conductivity tensors (Grandits *et al.*, 2021), effects of anti-arrhythmic drugs on EP parameters (Chiu *et al.*, 2024a), and reconstruction of cardiac action potentials (Chiu *et al.*, 2024b; Ye *et al.*, 2023).

Despite their advantages, PINNs face significant challenges when generalizing to new computational domains (Gao *et al.*, 2021). As PDE solutions depend on boundary and initial conditions, any change in the computational domain typically necessitates retraining, which is time-consuming and inefficient for clinical use. Alternative methods, like physics-informed convolutional neural networks (PI-CNNs) and physics-informed graph neural networks (PI-GNNs), have been explored but either require uniform grid data or complex meshing at inference, making them unsuitable for irregular patient geometries (Dalton *et al.*, 2023; Shimada, 2011).

Furthermore, previous efforts to improve PINN generalization have incorporated shape parameters as additional inputs during training (Costabal *et al.*, 2024; Regazzoni *et al.*, 2022a; Sun *et al.*, 2023). However, choosing these parameters effectively remains challenging, as model performance can vary significantly depending on parameter selection. Recently, neural operators have shown that leveraging principles from nonlinear solid mechanics and incorporating reference domains can enhance generalization across varying computational domains (Yin *et al.*, 2024). This approach introduces a well-defined method to relate computational domains to a reference domain. Despite the promising results of this

concept in neural operators, its application within the context of PINNs remains limited, with the notable exception of Mezzadri et al. (Mezzadri *et al.*, 2023).

In this project, we investigate how to improve the generalization of PINNs across computational domains. We examine a mapping framework that utilizes principles from nonlinear solid mechanics and extends the concepts suggested within neural operators to PINNs. We demonstrate the effectiveness of our approach on cardiac EP problems, including a time-dependent PDE (Aliev-Panfilov model), using 2D domains. The proposed method enables PINNs to make more accurate predictions on new domains without retraining, paving the way for real-time simulations within cardiac EP.

The outline of this report is as follows. First, in Section II, we introduce the Aliev-Panfilov model and how it can be expressed over a reference domain. Next, we present our methods in Section III, including a description of our suggested PINNs. The experiments and results are given in Section IV, while Section V contains a discussion of the results, including limitations and possible improvements.

II. MATHEMATICAL MODELS

We consider a fixed computational domain $\Omega_0 \subset \mathbb{R}^2$ defined as the *reference domain*. The reference domain can be deformed into a new computational domain $\Omega \subset \mathbb{R}^2$ referred to as the *deformed domain*. We assume that there exists a continuous mapping between Ω_0 and Ω such that

$$\Phi := \mathbf{X} \rightarrow \mathbf{x} \tag{1}$$

where $\mathbf{X}$ is a given point in Ω_0 while $\mathbf{x}$ is the associated point in Ω. We define a family of deformed domains as

$$\{\Omega_i \in \mathbb{R}^2\} \in \mathcal{G}, \qquad i = 1, 2, ..., n \tag{2}$$

where $\mathcal{G}$ consists of n domains in 2D. Each domain in $\mathcal{G}$ has a unique mapping Φ_i that relates points in Ω_0 to Ω_i, as defined in (1). For clarity, quantities in Ω_0 will be denoted with uppercase letters while quantities in Ω will be denoted with lowercase letters.

A. Cardiac electrophysiology

The cardiac action potential can be modeled with the monodomain equation, which describes the evolution of the transmembrane potential V over a given computational domain representing the cardiac tissue. When coupled with the Aliev-Panfilov (Aliev and Panfilov, 1996) ionic current model, the equations read

$$\begin{cases} \frac{\partial V}{\partial \tau} = \nabla \cdot (\mathbf{D}\nabla V) - I_{\text{ion}}(V, W) & \text{in } \Omega, \\ \frac{\partial W}{\partial \tau} = \epsilon(V, W)\left(-W - kV\left(V - a - 1\right)\right) & \text{in } \Omega, \\ \mathbf{D}\nabla V \cdot \mathbf{n} = 0 & \text{on } \partial\Omega \end{cases} \tag{3}$$

with $I_{\text{ion}}(V, W) = kV(V-a)(V-1)+VW$ and $\epsilon(V, W) = \left(\epsilon_0 + \frac{\mu_1 W}{V+\mu_2}\right)$. V, W, and τ are dimensionless variables representing the transmembrane potential, recovery variable, and time, respectively. Thus, $V \in [0, 1]$ is given in arbitrary units (AU), while $\tau = 12.9t$ is measured in temporal units (TU) with t given in milliseconds. The tissue conductivity is defined by the diffusion tensor $\mathbf{D}$, while $k, a, \epsilon_0, \mu_1, \mu_2$ are parameters controlling the overall shape and temporal dynamics of V and W. The last line in (3) employs no-flux Neumann boundary condition where $\mathbf{n}$ is the vector normal to the boundary of Ω. Consequently, there is no leakage of V to regions outside of Ω.

B. Mechano-electric formulation of cardiac electrophysiology

When the computational domain Ω is deformed somehow with respect to a reference domain Ω_0, we can utilize principles from non-linear solid mechanics to express the transmembrane potential in Ω over Ω_0. This is a common approach when looking at the coupling between the electric and mechanical activities in the heart, also referred to as the mechano-electric feedback (Bucelli *et al.*, 2023; Nobile *et al.*, 2012; Regazzoni *et al.*, 2022b; Salvador *et al.*, 2022). If the deformation of Ω_0 is time-independent, the equations presented in (3) can be reformulated over Ω_0 as

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{1}{J}\nabla \cdot (J\mathbf{F}^{-1}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V) - I_{\text{ion}}(V, W) & \text{in } \Omega_0, \\ \frac{\partial W}{\partial \tau} = \epsilon(V, W)\left(-W - kV\left(V - a - 1\right)\right) & \text{in } \Omega_0, \\ J\mathbf{F}^{-1}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V \cdot \mathbf{N} = 0 & \text{on } \partial\Omega_0, \end{cases} \tag{4}$$

where $\mathbf{F}$ is the deformation gradient tensor given as $\mathbf{F} = \mathbf{I} + \nabla U$, where U denotes the displacement of Ω with respect to Ω_0. Additionally, J is the deformation Jacobian given as $J = \det(\mathbf{F})$ and $\mathbf{N}$ is the normal vector to the boundary of Ω_0. The reader is referred to Appendix A for a full derivation of (4).

III. METHODS

In the following section, we present our methods used for running the experiments. We start off by presenting the data used for training and testing, before shifting the focus to our PINNs and how they were evaluated.

A. Computational Domains

We defined a $10 \times 10\,\text{cm}$ square as reference domain Ω_0. The reference domain was deformed in various ways through an affine transformation given in its most general form as

$$\mathbf{x} = \mathbf{A}\mathbf{X} + \mathbf{X}^{\mathbf{T}}\mathbf{M}\mathbf{X} \tag{5}$$

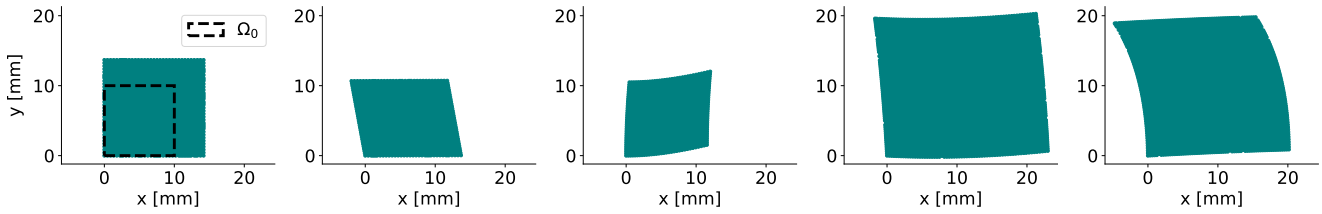


FIG. 1 Example of a deformed domain from each domain family starting from $\mathcal{G}_1$ (left) to $\mathcal{G}_5$ (right). The dashed black lines in the first figure outline the area of the defined reference domain Ω_0.

with

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_4 \end{bmatrix} \tag{6}$$

We created various domain families ($\mathcal{G}_i$) by changing the elements in $\mathbf{A}$ and $\mathbf{M}$. The elements in $\mathbf{A}$ and $\mathbf{M}$ are referred to as the *affine parameters* and is denoted as $\varphi(\Omega) = \{a_1, a_2, a_3, a_4, m_1, m_4\}$. For each deformation, the corresponding deformation gradient $\mathbf{F}$ and deformation Jacobian J were computed at each spatial point. We generated 7 domain families in total. The parameters used for creating family $\mathcal{G}_1$ - $\mathcal{G}_5$ are presented in Table I while Figure 1 shows a corresponding example of a domain from each family. The domain families $\mathcal{G}_1$ - $\mathcal{G}_5$ consisted of 12 domains each. Additionally, two versions of a final domain family $\mathcal{G}_6$ were created by combining the domains from $\mathcal{G}_1$ - $\mathcal{G}_3$. The first version consisted of 36 domains in total (12 domains from $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ were combined), while the second version consisted of 150 domains in total (50 domains from $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ were combined).

B. Synthetic Data Generation

We used *openCARP* (openCARP consortium *et al.*, 2024; Plank* *et al.*, 2021) to generate synthetic data by solving the Aliev-Panfilov model in (3) over the deformed domains using FEM. The synthetic data was used as ground-truth data during training and prediction of our PINNs. We applied a current of $100\,\mu\mathrm{Acm}^{-2}$ for $2\,\mathrm{TU}$ to the nodes located at the left boundary of the domain. The mesh in Ω_0 were created with a maximum element size of 0.05. After applying the affine transformation to Ω_0, we remeshed the deformed domain Ω using a maximum element size of 0.05. This was done to avoid distorted elements which could cause inaccurate FEM solutions. We oriented the fibers along the x-axis, and used isotropic conductivities with $g_{il} = g_{it} = 0.2\,\mathrm{Sm}^{-1}$ and $g_{el} = g_{et} = 1.0\,\mathrm{Sm}^{-1}$. We used a timestep of 1 ms for solving the model, and $C_m = 1\,\mu\mathrm{Fcm}^{-2}$ and $\beta = 0.14\,\mu\mathrm{m}^{-1}$ for the membrane capacitance and surface area to volume ratio, respectively. The simulations were run for $40\,\mathrm{TU}$, and solutions at $\tau < 2\,\mathrm{TU}$ were excluded to remove applied current from the system.

C. Physics-Informed Neural Networks

We developed three different PINNs for predicting the transmembrane potential V over new domains. The ultimate aim of the PINNs was to learn a function such that

$$\mathcal{NN}(\rho;\theta) = \hat{V}(\mathbf{x},\tau) \simeq V(\mathbf{x},\tau)$$

where ρ represents a given set of input parameters, θ are the trainable network parameters (weights and biases), and $\hat{V}$ is the predicted transmembrane potential. The models were trained by minimizing a hybrid loss function defined as

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \mathcal{L}_{phys}(\theta) + \mathcal{L}_{bc}(\theta) + \mathcal{L}_{init}(\theta) \tag{7}$$

where $\mathcal{L}_{data}(\theta)$ is the loss due to known data created in Section III.B, $\mathcal{L}_{phys}(\theta)$ is the loss described by the governing PDE, $\mathcal{L}_{bc}(\theta)$ is the loss described by the boundary condition, while $\mathcal{L}_{init}(\theta)$ is the loss due to the initial condition at $\tau_0 = 2\,\mathrm{TU}$. Thus, each loss term was defined as

$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \left(\hat{V}(\eta_i,\tau_i) - V(\eta_i,\tau_i)\right)^2$$

$$\mathcal{L}_{phys}(\theta) = \frac{1}{N_{phys}} \sum_{j=1}^{N_{phys}} \left(f_V\left(\eta_j,\tau_j\right)^2 + f_W\left(\eta_j,\tau_j\right)^2\right)$$

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{k=1}^{N_{bc}} f_{bc}\left(\eta_k,\tau_k\right)^2$$

$$\mathcal{L}_{init}(\theta) = \frac{1}{N_{init}} \sum_{l=1}^{N_{init}} \left(\hat{V}(\eta_l,\tau_0) - V(\eta_l,\tau_0)\right)^2$$

where $\hat{V}$ is the predicted value made by the PINN and V represents the FEM data used as ground-truth. $\mathcal{N}_{data}$, $\mathcal{N}_{phys}$, $\mathcal{N}_{bc}$, and $\mathcal{N}_{ic}$ give the number of data points used to evaluate each loss term while $f_V\left(\eta_j,\tau_j\right)$ and $f_W\left(\eta_j,\tau_j\right)$ represents the PDE residuals and $f_{bc}\left(\eta_k,\tau_k\right)$ the boundary condition residual. Furthermore, $\eta = \{\mathbf{x},\mathbf{X}\}$ represents the spatial locations used when computing the losses, which differed between the three models. The Basic-PINN and Affine-PINN used $\eta = \mathbf{x}$ and residuals

	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$	$\mathcal{G}_4$	$\mathcal{G}_5$
a_1, a_4	$[1.0, 1.5]$	$[1.0, 1.5]$	1.0	$[1.0, 1.5]$	$[1.0, 1.5]$
a_2, a_3	0.0	$[-0.2, 0.2]$	0.0	$[-0.2, 0.2]$	$[-0.2, 0.2]$
m_1, m_4	0.0	0.0	$[-0.01, 0.02]$	$[-0.01, 0.02]$	$[-0.05, 0.1]$

TABLE I An overview of how the different domain families ($\mathcal{G}_i$) were created with increasing complexity. The affine parameters $(a_1, a_2, a_3, a_4, m_1, m_4)$ were selected randomly from a uniform distribution within the given ranges. Each domain family consisted of 12 domains.

	ρ	f_V, f_W, f_{bc}
Basic-PINN	$\{\mathbf{x}, \tau\}$	Residuals of (3)
Affine-PINN	$\{\mathbf{x}, \tau, \varphi(\Omega)\}$	Residuals of (3)
RefAffine-PINN	$\{\mathbf{X}, \tau, \varphi(\Omega)\}$	Residuals of (4)

TABLE II Overview of how the three PINNs differed. ρ represents the input parameters, f_V, f_W, and f_{bc} represent the residuals used to compute the physics loss and boundary condition loss during training, $\mathbf{x} \in \Omega$ and $\mathbf{X} \in \Omega_0$, while $\varphi(\Omega)$ gives the affine parameters related to the transformation of Ω_0 to Ω. Note that all models take the dimensionless time variable τ as input.

of (3) while the RefAffine-PINN used $\eta = \mathbf{X}$ and residuals from (4) when computing the loss terms. Additionally, the input parameters ρ were different in each PINN. Table II provides an overview of how the three PINNs differed in terms of input parameters and residuals used in the loss function.

Furthermore, we used a fully-connected neural network with 10 hidden layers for all models. Each layer had 25 neurons and employed the *tanh* as activation function. The models predicted $\hat{V}$ and $\hat{W}$ as outputs, as illustrated in Figure 2.

D. Training procedures

We split each domain family into a training set, a validation set, and a test set. The exact train, validation, and test split ratio varied depending on the number of domains available in each $\mathcal{G}_i$. The split ratio used in each experiment is presented in the corresponding subsection of Section IV. Furthermore, we selected $\mathcal{N}_{data} = 14$, $\mathcal{N}_{phys} = 700$, $\mathcal{N}_{bc} = 80$, and $\mathcal{N}_{ic} = 300$ spatial locations from each domain in the training set, and trained the models for 5000 epochs. We used a learning rate of 10^{-3} for the first 100 epochs and a learning rate of 10^{-4} for the remaining epochs. We employed Adam as optimizer (Kingma and Ba, 2014), and each term in the loss function was equally weighted. Furthermore, the Glorot initialization scheme was used for weight initialization (Glorot and Bengio, 2010), while biases were initialized to zero. We employed a batch size of 512 during training to avoid running out of memory locally.

Furthermore, we computed $\mathcal{L}(\theta)$ for each domain in the validation set during training. Since our validation set consisted of multiple different domains, we selected the model state that gave the lowest maximum $\mathcal{L}(\theta)$ over the various validation domains, rather than the lowest average $\mathcal{L}(\theta)$, as the best model. In this way, we ensured that our model generalized well to domains significantly differing from the training set. We computed the loss over the validation domains every 10^{th} epoch using only a subsample of points from each domain to save computational resources.

The models were trained using a NVIDIA HGX H200 at the Kristiania-HPC infrastructure[1] or using a NVIDIA GeForce RTX 4070 Laptop GPU.

E. Algorithms

We implemented all models with *PyTorch* and used `autograd` (Paszke, 2019) to efficiently compute gradients defined in $\mathcal{L}_{phys}(\theta)$ and $\mathcal{L}_{bc}(\theta)$. Additionally, we implemented unit tests to ensure that the formulation of (4) was correctly defined. This was done by comparing the integral of the diffusion term defined in (3) over Ω and (4) over Ω_0. We used a set of predefined functions describing V and solved the integrals both analytically with *SymPy* and numerically with Monte Carlo. The project code is available at `https://github.com/ingvildadde/FYS9429_project`.

F. Evaluation metrics

We employed the relative L_2 error (ε_{L2}) to evaluate each PINN. The metric is given as

$$\varepsilon_{L2} = \frac{\sqrt{\sum_{i=0}^{N_{test}} \sum_{j=0}^{N_{time}} \left(\hat{V}\left(\mathbf{x}_i, \tau_j\right) - V\left(\mathbf{x}_i, \tau_j\right)\right)^2}}{\sqrt{\sum_{i=0}^{N_{test}} \sum_{j=0}^{N_{time}} V\left(\mathbf{x}_i, \tau_j\right)^2}} \quad (8)$$

where N_{test} and N_{time} are the number of spatial test points and time points, respectively. Furthermore, we used activation time (AT) maps and repolarization time (RT) maps to evaluate wavefront differences. The AT

[1] The research presented in this project has received significant support from the Kristiania-HPC infrastructure, financially sponsored by Kristiania University College

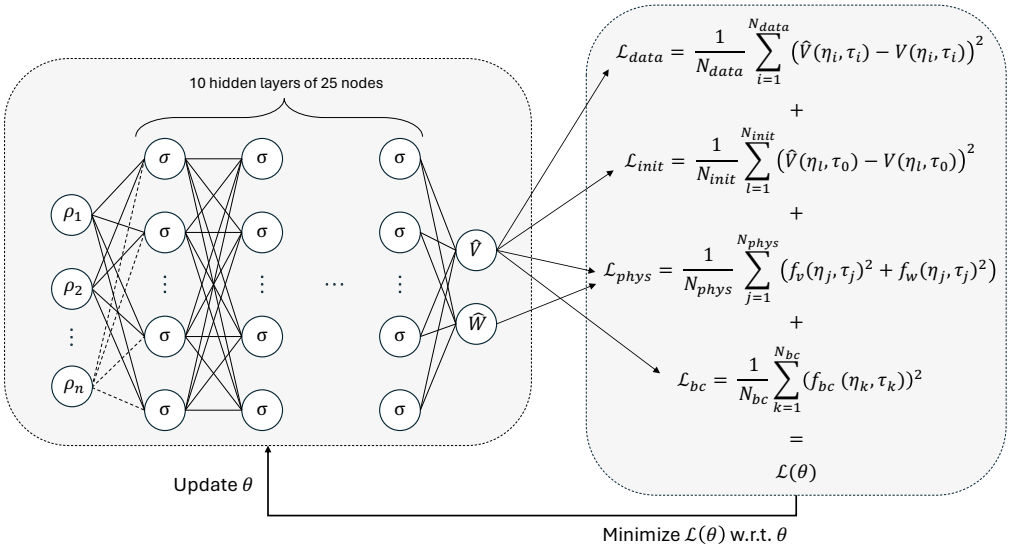


FIG. 2 Overview of the network architecture and optimization process for the PINNs. The input parameters, denoted as $\rho_1, ..., \rho_n$, are fed into the network and sent through 10 hidden layers with 25 nodes in each layer. The network uses $\sigma = tanh$ and gives a predicted value of the transmembrane potential ($\hat{V}$) and recovery variable ($\hat{W}$). The outputs are used in the various loss terms, and the total loss $\mathcal{L}(\theta)$ is computed. The network parameters θ are updated using the Adam optimizer to minimize $\mathcal{L}(\theta)$.

and RT were defined as the time at which

$$\text{AT} := V > 0\,\text{mV}$$
$$\text{RT} := V < -70\,\text{mV}$$

where the RT assumes that the transmembrane potential is in an activated state. Additionally, we created videos of the predicted results to inspect the PINN's performance visually.

IV. RESULTS

In the following section, we present our findings from our experiments using the methods described in Section III. First, we present a simple proof-of-concept experiment to demonstrate the need for new approaches when training a PINN on large or complex deformations. Next, we show how the Affine-PINN and RefAffine-PINN performance changes as the number of training domains increases. Finally, we test our proposed methods in an extrapolation scenario.

A. Experiment I: Proof-of-concept

Our first experiment was a proof-of-concept examination where the models were trained and tested on domains from $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, $\mathcal{G}_4$, and $\mathcal{G}_5$. Each domain family consisted of $n = 12$ domains, with $n_{train} = 4$, $n_{val} = 4$, and $n_{test} = 4$ domains. Figure 3 shows the distribution of ε_{L2} over the test domains in each family, revealing a significantly higher ε_{L2} for the Basic-PINN in $\mathcal{G}_4$ and $\mathcal{G}_5$. Additionally, Table III presents the mean ε_{L2}, denoted as $\bar{\varepsilon}_{L2}$, for each boxplot in Figure 3. Figure 4 presents two examples of how the transmembrane potential evolves with time at a random spatial location in the test domain where the Basic-PINN had lowest ε_{L2} for $\mathcal{G}_2$ in 4(a) and $\mathcal{G}_5$ in 4(b). Figure 4(b) demonstrates how the Basic-PINN is unable to learn sufficiently and make accurate predictions when trained on domains with larger deformations. An example of the loss curves obtained during training of each PINN on domains from the $\mathcal{G}_5$ family is presented in Figure 10 of Appendix B.

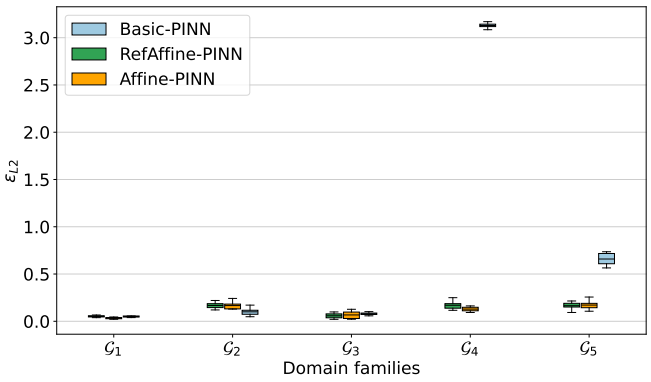
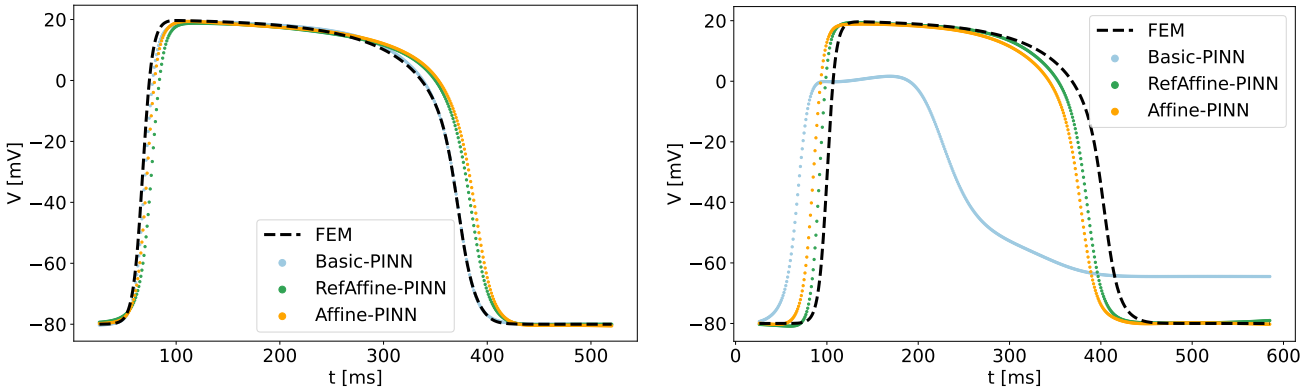


FIG. 3 Boxplots of ε_{L2} obtained during prediction of test set in the given domain families. The horizontal black line in the boxes indicates the mean ε_{L2}.



(a) $\Omega \in \mathcal{G}_2$ (b) $\Omega \in \mathcal{G}_5$

FIG. 4 Predicted transmembrane potential (V) at a random spatial location in the test domain where the Basic-PINN had lowest ε_{L2} for $\mathcal{G}_2$ in (a) and $\mathcal{G}_5$ in (b). The black dashed line represents FEM data used as ground-truth.

	$\bar{\varepsilon}_{L2}$				
	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$	$\mathcal{G}_4$	$\mathcal{G}_5$
Basic-PINN	0.050	**0.100**	0.080	3.130	0.659
Affine-PINN	**0.033**	0.166	0.067	**0.129**	**0.117**
RefAffine-PINN	0.052	0.167	**0.058**	0.169	0.165

TABLE III Mean ε_{L2} error obtained during prediction of test set in the given domain families.

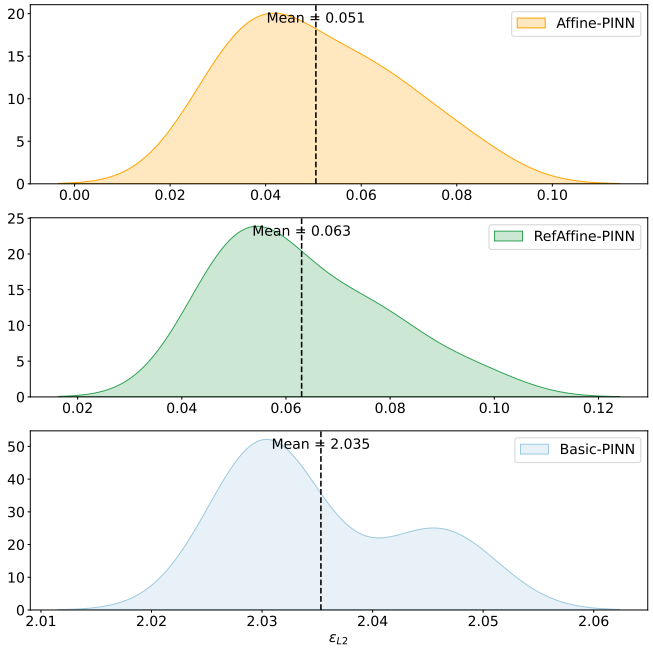


FIG. 5 Density plot of ε_{L2} obtained during prediction of test set in $\mathcal{G}_6$.

B. Experiment II: Different deformation types

In our next experiment, we investigated how the models performed when trained on a dataset containing different deformation types. Thus, we used the $\mathcal{G}_6$ family for training and testing, which consisted of a combination of domains from $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$. Hence, the dataset consisted of a combination of both linear and non-linear deformations with $n = 36$ domains, and $n_{train} = 12$, $n_{val} = 12$, and $n_{test} = 12$ domains. Figure 5 shows a density plot of ε_{L2} for each model when making predictions on the test domains. The plot demonstrates a significantly higher ε_{L2} distribution for the Basic-PINN, suggesting that the Basic-PINN is not able to handle scenarios involving different deformation types.

C. Experiment III: Training set size

For the remaining parts of our result section, we will focus on the Affine-PINN and RefAffine-PINN, and how the two models differed. Consequently, our next experiment investigated how the model performances were affected by the training set size. We created a larger $\mathcal{G}_6$ family with a total number of domains equal to 150. The dataset was balanced, taking 50 domains from $\mathcal{G}_1$, 50 domains from $\mathcal{G}_2$, and 50 domains from $\mathcal{G}_3$. We used 36 domains as a test set and 9 domains as a validation set. The test and validation sets were unchanged during each experiment, while the training set increased in size from 6 to 105 domains. Figure 6(a) shows how the ε_{L2} over the test domains decreases as the number of training samples increases. A similar plot is presented in Figure 6(b), but this time the predictions were made over a new dataset, namely the $\mathcal{G}_5$ family. Figure 6(a) demonstrates that the strongest decrease in ε_{L2} occurs until $n_{train} = 12$, while Figure 6(b) shows an increase in ε_{L2} for $n_{train} > 12$. Based on this result, we selected the models trained with $n_{train} = 12$ as the most optimal models. Figure 7 shows the computed AT maps at the vertices in three different domains from the test set of $\mathcal{G}_6$ using the models trained with $n_{train} = 12$ domains. A similar plot of the RT maps

for the same domains can be found in Figure 11 of Appendix C.

D. Experiment IV: Extrapolation scenario

In our final experiment, we used the model trained on 12 domains in Section IV.C to evaluate the model's extrapolation abilities. Thus, we predicted the transmembrane potential over domains in the $\mathcal{G}_5$ family and created corresponding AT and RT maps. These AT and RT maps are presented for selected domains of $\mathcal{G}_5$ in Figure 8 and 9, respectively. Additionally, videos of the predicted outputs for the domains in Figure 8 and 9 are available in the shared GitHub repository (`https://github.com/ingvildadde/FYS9429_project`).

V. DISCUSSION

In this project, we explored how to make PINNs geometry-aware by relating computational domains to a reference domain. We presented two approaches, namely Affine-PINN and RefAffine-PINN, where both PINNs incorporated affine parameters as additional inputs. Additionally, the RefAffine-PINN included spatiotemporal inputs from the reference domain and employed residuals derived from a modified physics loss, as defined in Equation 4. Our results showed that these two approaches outperformed a conventional PINN, referred to as the Basic-PINN.

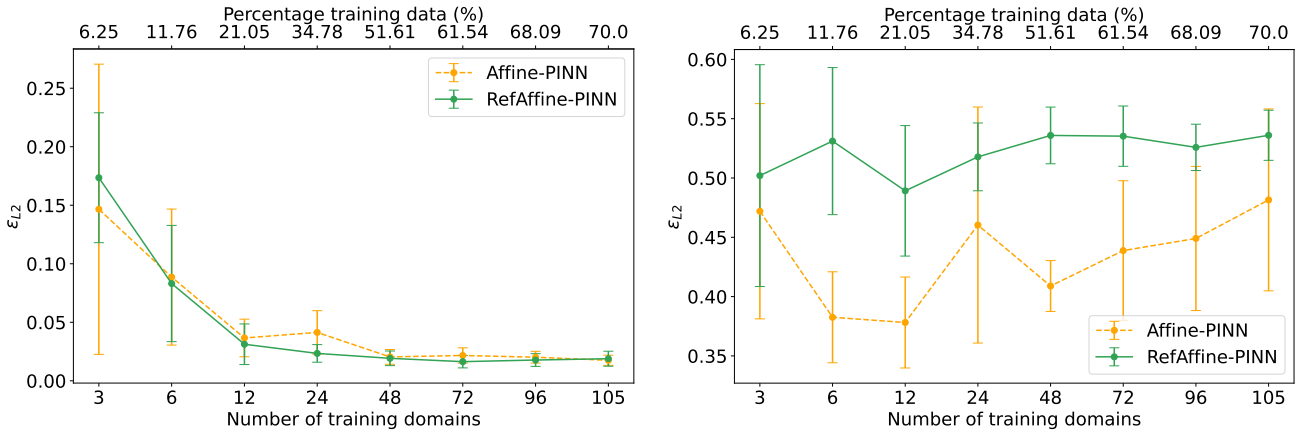
Section IV.A demonstrates that the Basic-PINN struggles to learn effectively when trained on computational domains exhibiting large variations. Figures 3 and 4 illustrate that while the Basic-PINN performs adequately on domains with small variations ($\mathcal{G}_1$–$\mathcal{G}_3$), it fails on larger deformations ($\mathcal{G}_4$ and $\mathcal{G}_5$). Moreover, Figure 5 further highlights this limitation in the context of different deformation types ($\mathcal{G}_6$), where the ε_{L2} error is significantly higher for the Basic-PINN than for the Affine-PINN and RefAffine-PINN. These findings support prior literature (Gao *et al.*, 2021) suggesting that variations in the computational domain can lead to fundamentally different PDE solutions, posing a challenge for conventional PINNs to generalize across varying domains. In contrast, our results show that leveraging a reference domain improves generalizability, as indicated by the significantly lower mean ε_{L2} values for large deformations in Table III and lower error distributions in Figure 5 for the Affine-PINN and RefAffine-PINN.

Figure 6(a) shows that both Affine-PINN and RefAffine-PINN improve when predicting over the test set as the number of training samples increases. In contrast, Figure 6(b) reveals a decrease in ε_{L2} error up to $n_{train} = 12$, followed by a deterioration for larger training sizes. This suggests that the performance gain seen in the interpolation scenario may result from increasing similarity between training and test domains, potentially leading to overfitting. Thus, when applied to extrapolation scenarios, training on many domains may reduce generalization. Furthermore, Figure 7 shows accurate AT predictions by both PINNs when trained on 12 domains and evaluated on test domains in $\mathcal{G}_6$.

Figures 8 and 9 present the AT and RT for predictions on an unseen dataset ($\mathcal{G}_5$), using the PINNs in Section IV.C trained on 12 domains. Figure 8 illustrates a difference in wavefront dynamics between the models: the Affine-PINN exhibits a more inaccurate initial condition, whereas the RefAffine-PINN appears to better capture wave propagation. However, these qualitative observations should be quantified, highlighting the need for improved evaluation metrics for wavefront differences. Despite the promising appearance of the RefAffine-PINN wavefront, discrepancies remain in conduction velocity, as evident from the differences in AT colorbars, where both models lag behind the ground truth. This likely reflects overfitting to smaller deformations ($\mathcal{G}_1$–$\mathcal{G}_3$). Figure 9 shows better RT timing but also indicates disrupted wavefront dynamics, especially for the Affine-PINN model.

Overall, the results confirm that relating computational domains to a reference domain enhances the generalizability of PINNs. Both the Affine-PINN and RefAffine-PINN significantly outperform the Basic-PINN in scenarios involving large or complex deformations. The performance differences between the Affine-PINN and RefAffine-PINN are more subtle. The Affine-PINN has the advantage of simplicity, achieved by adding a few input parameters, while the RefAffine-PINN is more complex but may offer better extrapolation capabilities, as suggested by Figures 8 and 9. Further experimentation is required to thoroughly understand the strengths and limitations of each PINN.

Moreover, this project has several limitations. Except in Section IV.C, the models were trained on a relatively small set of training data, which could simplify the training process. Hence, the method has not been rigorously tested in scenarios where the training data is more complex. Additionally, the experiments were restricted to 2D domains with explicitly known affine parameters, which simplifies real-world problems where deformations often are in 3D and affine parameters are often unknown. Hence, alternative representations of domain deformations should be investigated, such as principal component analysis (PCA) (Yin *et al.*, 2024) or signed distance function (SDF) based methods (Kraus and Tatsis, 2024). Additionally, in this project, we have solely used synthetically generated data. Hence, the problem is a significant simplification of real-world scenarios. Moreover, our study did not aim to optimize training procedures. We used relatively simple network architectures and standard training routines, potentially limiting performance.



(a) ε_{L2} over test domains in $\mathcal{G}_6$.

(b) ε_{L2} over domains in $\mathcal{G}_5$.

FIG. 6 Error bar plot of ε_{L2} as the number of training domains increases. Dots represent the mean ε_{L2} value while the bars represent the standard deviation.

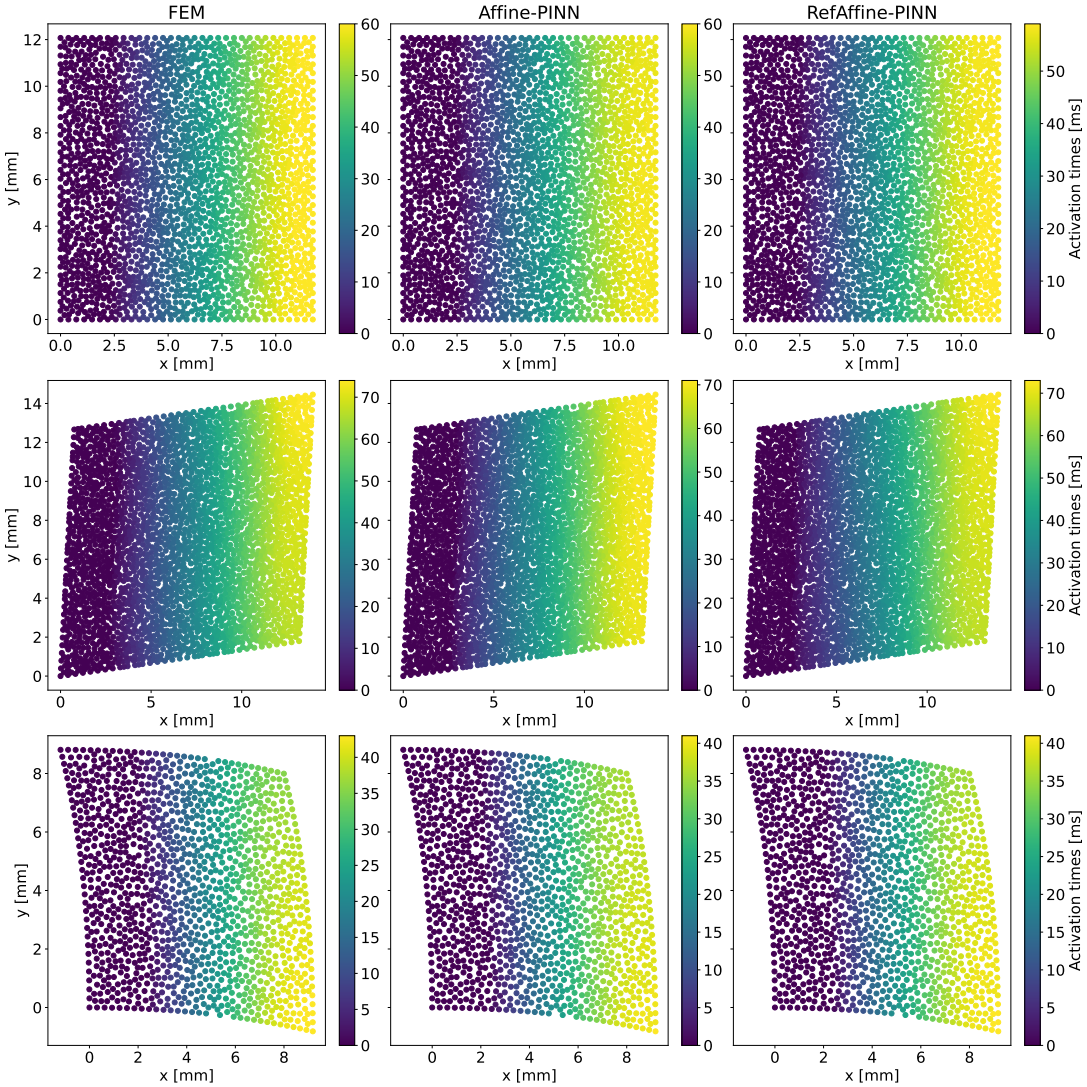


FIG. 7 AT maps from three selected domains in the test set of $\mathcal{G}_6$. The models used for prediction were trained with $n_{train} = 12$ domains.

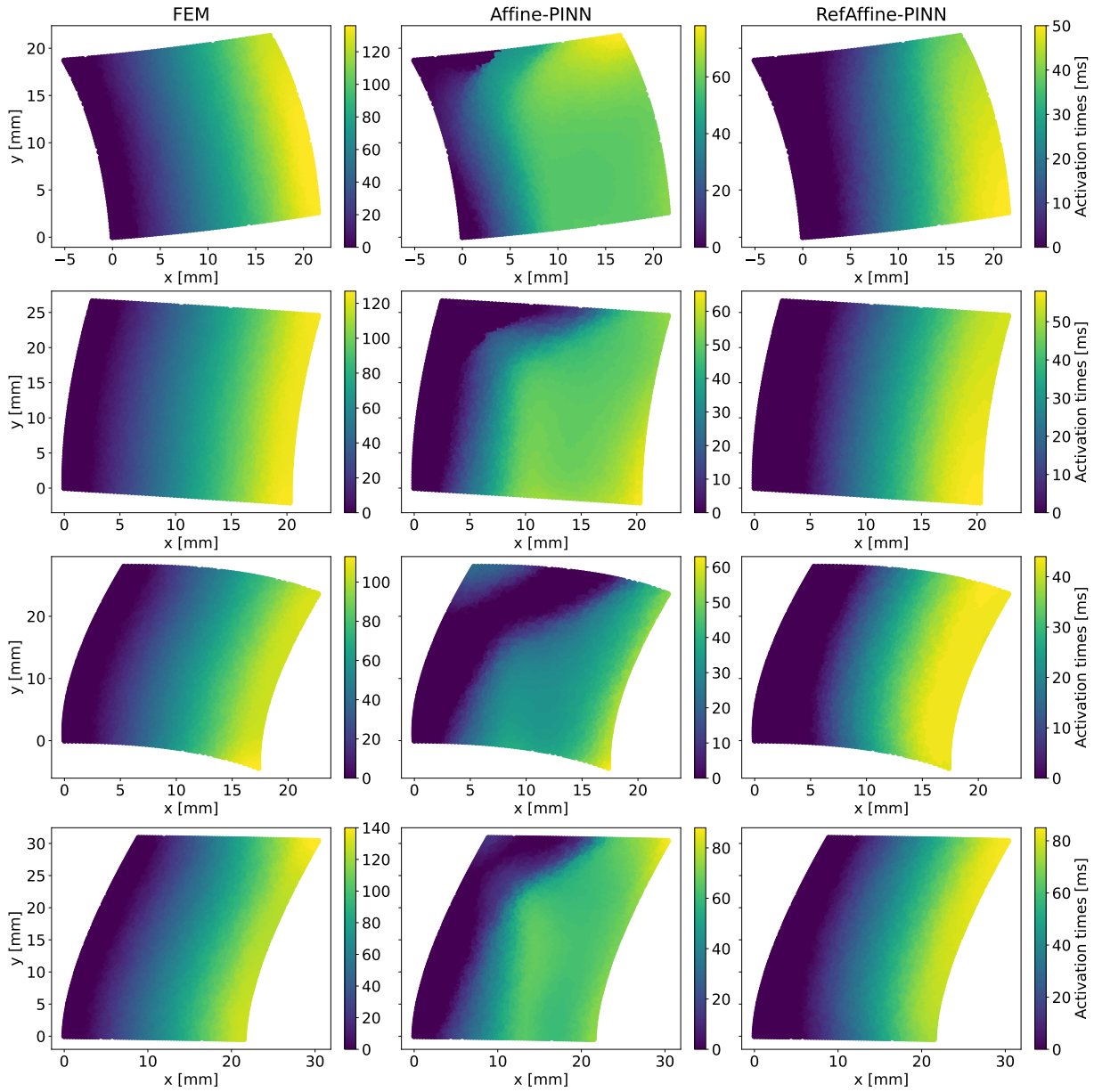


FIG. 8 AT maps from four selected domains in $\mathcal{G}_5$. The models used for prediction were trained on $\mathcal{G}_6$ with $n_{train} = 12$ domains.

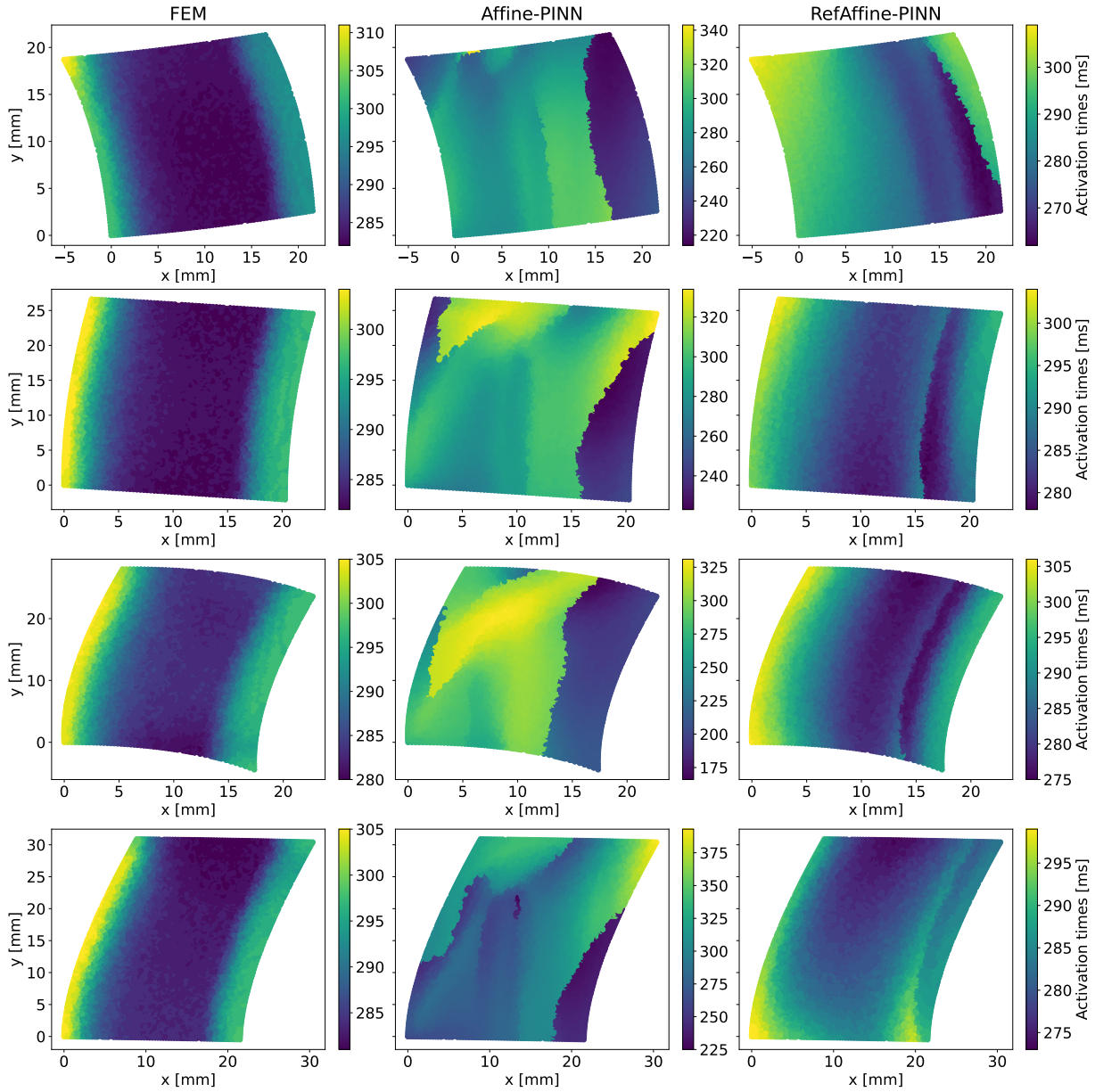


FIG. 9 RT maps from four selected domains in $\mathcal{G}_5$. The models used for prediction were trained on $\mathcal{G}_6$ with $n_{train} = 12$ domains.

Future work should incorporate advanced techniques suggested in the literature, such as resampling of collocation points, dynamically weighted loss functions (Wang *et al.*, 2023), and the L-BFGS optimizer for improved physics convergence (Herrero Martin *et al.*, 2022; Sun *et al.*, 2023). Alternative architectures, beyond fully connected neural networks (FCNNs), could also provide more stable training. Neural operators, for instance, offer an exciting direction, especially when combined with physics-based losses. While prior studies (Yin *et al.*, 2024) have explored this area, most focus on AT and RT rather than the full trajectory of V. Lastly, regularization techniques, such as dropout or noise injection, and incorporation of separate evaluation metrics for wavefront alignment and conduction velocity assessment should be investigated in future work to tackle overfitting and precise evaluation of the methods.

Moving forward, we aim to extend our models to 3D domains, explore complex deformations, and improve the training procedure. We also plan to explore deformation representations suitable for cases where affine parameters are unknown. These steps will be essential for transitioning our approaches from theoretical proof-of-concept to practical tools in scientific and engineering applications.

VI. CONCLUSION

In this project, we explored new approaches for making PINNs geometry-aware. We proposed two new methods, namely the Affine-PINN and RefAffine-PINN, which relate computational domains to a reference domain by introducing affine deformation parameters as additional inputs. In particular, the RefAffine-PINN model also receives spatiotemporal inputs from the reference domain and is trained with a modified physics-informed loss function.

Our results demonstrate that a conventional PINN (Basic-PINN) struggles to learn effectively when faced with computational domains undergoing large or complex deformations. In contrast, our proposed methods outperform the Basic-PINN, achieving significantly lower relative L2 error (ε_{L2}) in such scenarios. These findings indicate that our approaches offer a promising direction for enhancing the geometric awareness of PINNs.

Nevertheless, further development and investigation are required to fully assess the strengths and limitations of these methods. Future work will focus on a more comprehensive evaluation and potential extensions of the proposed methods.

REFERENCES

R. R. Aliev, and A. V. Panfilov (1996), "A simple two-variable model of cardiac excitation," Chaos, Solitons & Fractals **7** (3), 293–301.

H. J. Arevalo, F. Vadakkumpadan, E. Guallar, A. Jebb, P. Malamas, K. C. Wu, and N. A. Trayanova (2016), "Arrhythmia risk stratification of patients after myocardial infarction using personalized heart models," Nature communications **7** (1), 11437.

M. Bucelli, A. Zingaro, P. C. Africa, I. Fumagalli, L. Dede', and A. Quarteroni (2023), "A mathematical model that integrates cardiac electrophysiology, mechanics, and fluid dynamics: Application to the human left heart," International journal for numerical methods in biomedical engineering **39** (3), e3678.

C.-E. Chiu, A. L. Pinto, R. A. Chowdhury, K. Christensen, and M. Varela (2024a), "Characterisation of anti-arrhythmic drug effects on cardiac electrophysiology using physics-informed neural networks," in *2024 IEEE International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5.

C.-E. Chiu, A. Roy, S. Cechnicka, A. Gupta, A. L. Pinto, C. Galazis, K. Christensen, D. Mandic, and M. Varela (2024b), "Physics-informed neural networks can accurately model cardiac electrophysiology in 3d geometries and fibrillatory conditions," arXiv preprint arXiv:2409.12712.

F. S. Costabal, S. Pezzuto, and P. Perdikaris (2024), "δ-pinns: physics-informed neural networks on complex geometries," Engineering Applications of Artificial Intelligence **127**, 107324.

D. Dalton, D. Husmeier, and H. Gao (2023), "Physics-informed graph neural network emulation of soft-tissue mechanics," Computer Methods in Applied Mechanics and Engineering **417**, 116351.

N. Dermul, and H. Dierckx (2024), "Reconstruction of excitation waves from mechanical deformation using physics-informed neural networks," Scientific Reports **14** (1), 16975.

H. Gao, L. Sun, and J.-X. Wang (2021), "Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain," Journal of Computational Physics **428**, 110079.

X. Glorot, and Y. Bengio (2010), "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 9, edited by Y. W. Teh, and M. Titterington (PMLR, Chia Laguna Resort, Sardinia, Italy) pp. 249–256.

T. Grandits, S. Pezzuto, F. S. Costabal, P. Perdikaris, T. Pock, G. Plank, and R. Krause (2021), "Learning atrial fiber orientations and conductivity tensors from intracardiac maps using physics-informed neural networks," in *International Conference on Functional Imaging and Modeling of the Heart* (Springer) pp. 650–658.

C. Herrero Martin, A. Oved, R. A. Chowdhury, E. Ullmann, N. S. Peters, A. A. Bharath, and M. Varela (2022), "Ep-pinns: Cardiac electrophysiology characterisation using physics-informed neural networks," Frontiers in Cardiovascular Medicine **8**, 768419.

G. A. Holzaphel (2000), *Nonlinear Solid Mechanics: A Continuum Approach for Engineering* (Wiley).

G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang (2021), "Physics-informed machine learning," Nature Reviews Physics **3** (6), 422–440.

A. M. Katz (2010), *Physiology of the Heart* (Lippincott Williams and Wilkins).

D. P. Kingma, and J. Ba (2014), "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980.

M. A. Kraus, and K. E. Tatsis (2024), "Sdf-pinns: Joining physics-informed neural networks with neural implicit geometry representation," in *GNI Symposium & Expo on Artificial Intelligence for the Built World 2024*.

R. Mehra (2007), "Global public health problem of sudden cardiac death," Journal of electrocardiology **40** (6), S118–S122.

F. Mezzadri, J. Gasick, and X. Qian (2023), "A framework for physics-informed deep learning over freeform domains," Computer-Aided Design **160**, 103520.

F. Nobile, A. Quarteroni, and R. Ruiz-Baier (2012), "An active strain electromechanical model for cardiac tissue," International journal for numerical methods in biomedical engineering **28** (1), 52–71.

openCARP consortium, C. Augustin, P. M. Boyle, V. Loechner, R. Colin, A. Huppé, M. Gsell, M. Houillon, Y.-L. C. Huang, K. G. Hustad, E. Karabelas, A. Loewe, L. Myklebust, A. Neic, M. Nothstein, G. Plank, A. Prassl, J. Sánchez, G. Seemann, T. Stary, A. Thangamani, N. Tippmann, T. Trevisan Jost, E. Vigmond, E. M. Wülfers, and M. Linder (2024), "openCARP," .

A. Paszke (2019), "Pytorch: An imperative style, high-performance deep learning library," arXiv preprint arXiv:1912.01703.

G. Plank*, A. Loewe*, A. Neic*, C. Augustin, Y.-L. C. Huang, M. Gsell, E. Karabelas, M. Nothstein, J. Sánchez, A. Prassl, G. Seemann*, and E. Vigmond* (2021), "The openCARP simulation environment for cardiac

electrophysiology," Computer Methods and Programs in Biomedicine **208**, 106223.

M. Raissi, P. Perdikaris, and G. E. 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 **378**, 686–707.

F. Regazzoni, S. Pagani, and A. Quarteroni (2022a), "Universal solution manifold networks (usm-nets): non-intrusive mesh-free surrogate models for problems in variable domains," Journal of Biomechanical Engineering **144** (12), 121004.

F. Regazzoni, M. Salvador, P. C. Africa, M. Fedele, L. Dedè, and A. Quarteroni (2022b), "A cardiac electromechanical model coupled with a lumped-parameter model for closed-loop blood circulation," Journal of Computational Physics **457**, 111083.

F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl (2020), "Physics-informed neural networks for cardiac activation mapping," Frontiers in Physics **8**, 42.

M. Salvador, F. Regazzoni, S. Pagani, N. Trayanova, A. Quarteroni, *et al.* (2022), "The role of mechano-electric feedbacks and hemodynamic coupling in scar-related ventricular tachycardia," Computers in Biology and Medicine **142**, 105203.

K. Shimada (2011), "Current Issues and Trends in Meshing and Geometric Processing for Computational Engineering Analyses," Journal of Computing and Information Science in Engineering **11** (2), 021008.

Y. Sun, U. Sengupta, and M. Juniper (2023), "Physics-informed deep learning for simultaneous surrogate modeling and pde-constrained optimization of an airfoil geometry," Computer Methods in Applied Mechanics and Engineering **411**, 116042.

N. A. Trayanova, and P. M. Boyle (2014), "Advances in modeling ventricular arrhythmias: from mechanisms to the clinic," Wiley Interdisciplinary Reviews: Systems Biology and Medicine **6** (2), 209–224.

S. Wang, S. Sankaran, H. Wang, and P. Perdikaris (2023), "An expert's guide to training physics-informed neural networks," arXiv preprint arXiv:2308.08468.

Y. Ye, H. Liu, X. Jiang, M. Toloubidokhti, and L. Wang (2023), "A spatial-temporally adaptive pinn framework for 3d bi-ventricular electrophysiological simulations and parameter inference," in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer) pp. 163–172.

M. Yin, N. Charon, R. Brody, L. Lu, N. Trayanova, and M. Maggioni (2024), "A scalable framework for learning the geometry-dependent solution operators of partial differential equations," Nature Computational Science **4** (12), 928–940.

Appendix A: Derivation of mapped Aliev-Panfilov model

The Aliev-Panfilov model is given as

$$\begin{cases} \frac{\partial V}{\partial \tau} = \nabla \cdot (\mathbf{D}\nabla V) - I_{\text{ion}}(V,W) & \text{in } \Omega, \\ \frac{\partial W}{\partial \tau} = \epsilon(V,W)\left(-W - kV\left(V-a-1\right)\right) & \text{in } \Omega, \\ \mathbf{D}\nabla V \cdot \mathbf{n} = 0 & \text{on } \partial\Omega \end{cases} \tag{A1}$$

as described in Section II.A. We wish to map the PDE in (A1) from a deformed domain (Ω) to a reference domain (Ω_0). This can be achieved by applying the deformation gradient $\mathbf{F}$ and the deformation Jacobian J to quantities in (A1), as well as performing a variable substitution $\mathbf{x} \rightarrow \mathbf{X}$ where $\mathbf{x} \in \Omega$ and $\mathbf{X} \in \Omega_0$. The deformation gradient $\mathbf{F}$ is given as

$$\mathbf{F}(\mathbf{X},t) = \mathbf{I} + \nabla \mathbf{U}(\mathbf{X},t) \tag{A2}$$

where $\mathbf{U}(\mathbf{X},t) = \mathbf{x}(\mathbf{X}) - \mathbf{X}$ describes the displacements between points in Ω and Ω_0, while the deformation Jacobian is given as $J = \det(\mathbf{F})$ (Holzaphel, 2000).

Quantities that do not involve any divergences or gradients can be mapped directly through a volume change defined as

$$\mathrm{d}v = J\mathrm{d}V \tag{A3}$$

where dv and dV are infinitesimally small volume elements in Ω and Ω_0, respectively (Holzaphel, 2000). Gradients of a scalar field ϕ can be mapped as

$$\nabla\phi(\mathbf{x},t) = \mathbf{F}^{-\mathbf{T}}\nabla\phi(\mathbf{X},t) \tag{A4}$$

which is obtained by applying the chain rule to $\nabla\phi(\mathbf{x},t)$. Hence, the mapping of gradients of V in (A1) is given as

$$\nabla V(\mathbf{x},t) = \mathbf{F}^{-\mathbf{T}}\nabla V(\mathbf{X},t) \tag{A5}$$

Finally, the Nanson's formula (Holzaphel, 2000) can be used to map vector elements from Ω to Ω_0. The formula yields

$$\mathrm{d}s\mathbf{n} = J\mathbf{F}^{-\mathbf{T}}\mathrm{d}S\mathbf{N} \tag{A6}$$

where $\mathrm{d}s\mathbf{n}$ and $\mathrm{d}S\mathbf{N}$ give the vector elements of infinitesimally small areas defined in Ω and Ω_0.

In (A1), we encounter gradients and divergences in the divergence term in the first line and in the boundary condition term in the third line. The divergence term in (A1) can be rewritten in integral form as

$$\int_\Omega \nabla \cdot (\mathbf{D}\nabla V)\mathrm{d}\Omega$$

By applying Gauss's divergence theorem, we have that

$$\int_\Omega \nabla \cdot (\mathbf{D}\nabla V)\mathrm{d}\Omega = \int_{\partial\Omega} \mathbf{D}\nabla V \cdot \mathrm{d}s\mathbf{n} \tag{A7}$$

where $\partial\Omega$ is the surface of Ω and $\mathbf{n}$ is the vector normal to the surface. If we utilize the relationship of gradients in (A5) and Nanson's formula in (A6), the divergence term in (A7) can be expressed over Ω_0 as

$$\int_{\partial\Omega} \mathbf{D}\nabla V \cdot \mathrm{d}s\mathbf{n} = \int_{\partial\Omega_0} \mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V \cdot J\mathbf{F}^{-\mathbf{T}}\mathrm{d}S\mathbf{N} \tag{A8}$$

In the 2D case, we have that

$$\mathbf{D} \in \mathbb{R}^{2x2}, \qquad \mathbf{F} \in \mathbb{R}^{2x2}, \qquad \nabla V \in \mathbb{R}^{2x1}.$$

Hence, by assuming that $\mathbf{F}$ is invertible, the terms in (A8) can be reorganized as

$$\int_{\partial\Omega} \mathbf{D}\nabla V \cdot \mathrm{d}s\mathbf{n} = \int_{\partial\Omega_0} J\mathbf{F}^{-\mathbf{1}}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V \cdot \mathrm{d}S\mathbf{N} \tag{A9}$$

Finally, by applying Gauss's divergence theorem again, the divergence term in Ω and Ω_0 can be expressed as

$$\int_\Omega \nabla \cdot (\mathbf{D}\nabla V\mathrm{d}\Omega = \int_{\Omega_0} \nabla \cdot \left(J\mathbf{F}^{-\mathbf{1}}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V\right)\mathrm{d}\Omega_0 \tag{A10}$$

By following the same procedure as in (A7) - (A10) for the boundary condition in (A1) we have that

$$\int_\Omega (\mathbf{D}\nabla V \cdot \mathbf{n})\mathrm{d}\Omega = \int_{\Omega_0} (J\mathbf{F}^{-\mathbf{1}}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V \cdot \mathbf{N})\mathrm{d}\Omega_0 \tag{A11}$$

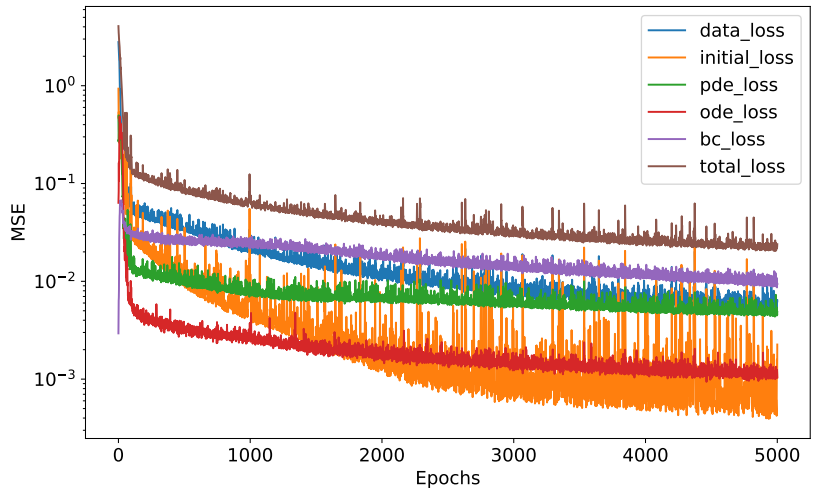
The remaining parts of (A1) do not include any divergences or gradients and can be mapped directly through a volume change as defined in (A3). Consequently, (A1) can be expressed over Ω_0 as

$$\begin{cases} \frac{\partial}{\partial \tau}(JV) = \nabla \cdot (J\mathbf{F}^{-\mathbf{1}}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V) - JI_{\text{ion}}(V,W) & \text{in } \Omega_0, \\ \frac{\partial}{\partial \tau}(JW) = J\epsilon(V,W)\left(-W - kV\left(V-a-1\right)\right) & \text{in } \Omega_0, \\ J\mathbf{F}^{-\mathbf{1}}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V \cdot \mathbf{N} = 0 & \text{on } \partial\Omega_0, \end{cases}$$

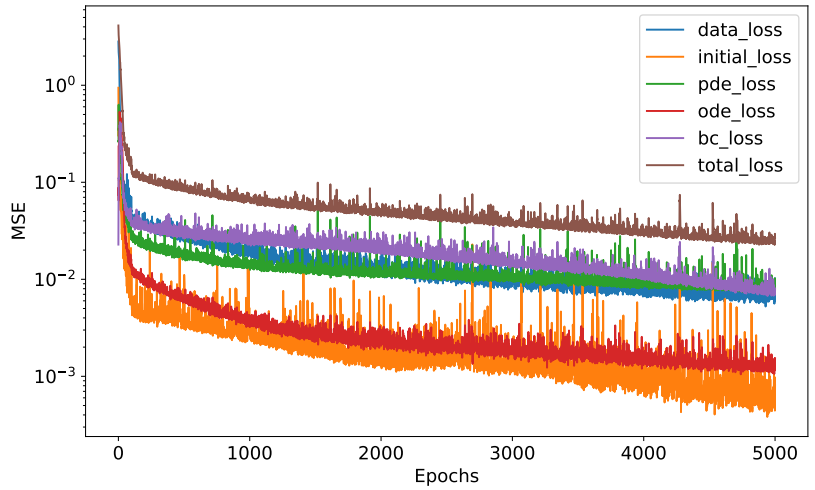
In the case where the deformation of Ω_0 is time-independent, we finally arrive at

$$\begin{cases} \frac{\partial V}{\partial \tau} = \frac{1}{J}\nabla \cdot (J\mathbf{F}^{-\mathbf{1}}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V) - I_{\text{ion}}(V,W) & \text{in } \Omega_0, \\ \frac{\partial W}{\partial \tau} = \epsilon(V,W)\left(-W - kV\left(V-a-1\right)\right) & \text{in } \Omega_0, \\ J\mathbf{F}^{-\mathbf{1}}\mathbf{D}\mathbf{F}^{-\mathbf{T}}\nabla V \cdot \mathbf{N} = 0 & \text{on } \partial\Omega_0, \end{cases}$$

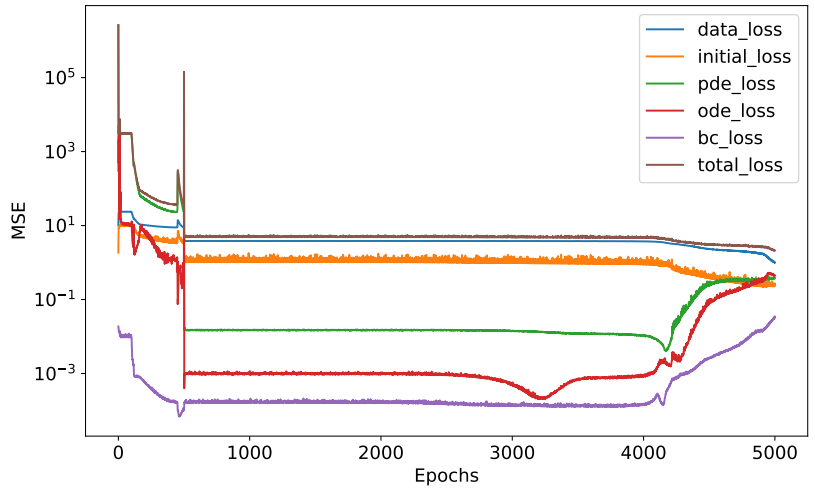
Appendix B: Loss curves example



(a) Affine-PINN



(b) RefAffine-PINN



(c) Basic-PINN

FIG. 10 Training loss curves for the different PINNs when trained on domains from the $\mathcal{G}_5$ family.

Appendix C: Supplementary Results

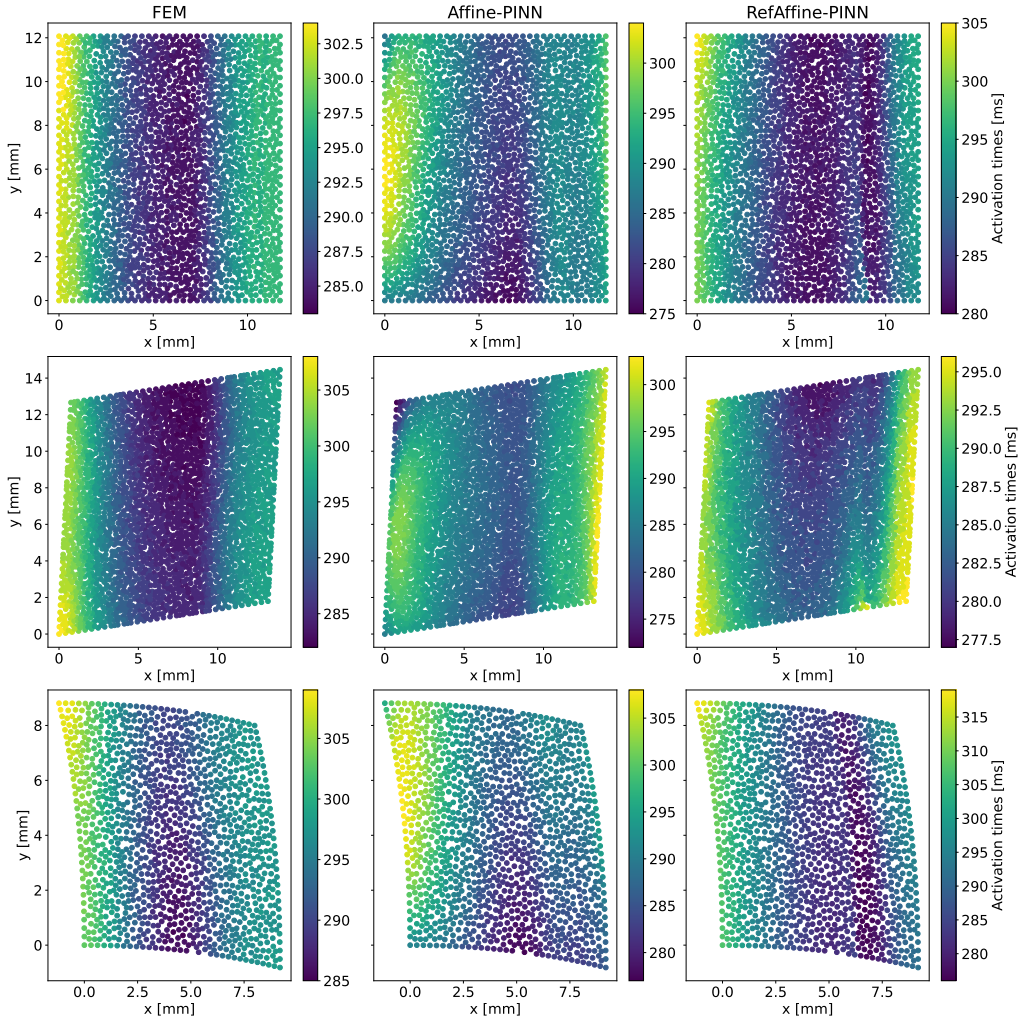


FIG. 11 RT maps from three selected domains in the test set of $\mathcal{G}_6$. The models used for prediction were trained with $n_{train} = 12$ domains.