

Speeding up Collocation-based Robust Variational Physics Informed Neural Networks with Singular Value Decompositions

Tomasz Służalec^{1,2*}, Mateusz Dobija¹, Maciej Sikora¹, Maciej Paszyński¹

¹ AGH University of Krakow, Krakow, Poland

² Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland

Abstract. Physics Informed Neural Networks (PINNs) solve time-dependent Partial Differential Equations (PDEs) by minimizing the residual, initial, and boundary conditions loss functions. The loss functions in PINN are generally not robust. The decrease in the loss function may not correspond to the decrease by the true error (the norm difference between the numerical and exact solution). Following the idea of Robust Variational PINN (RVPINN), we define the robust loss function by employing the weak residuals and the inverse of the Gram matrix as the discrete representative of the Riesz residual. To avoid the expensive integration of the weak residuals, we introduce the collocation method, where we use discrete weak formulations with integration replaced by summations at the collocation points and test functions replaced by Kronecker deltas. With this setup, we derive a sparse Gram matrix as the H_0^1 discrete inner product of the Kronecker delta test functions. The inverse of this Gram matrix can make the loss function robust. Furthermore, to accelerate the training process, we perform a truncated Singular Value Decomposition (SVD) of the inverse of the Gram matrix and use fast matrix-vector multiplication for the reduced rank matrix. The presented strategy reduces the cost of precomputing (before training) from $\mathcal{O}(N^3)$ (inversion of the Gram matrix) to $\mathcal{O}(N^2k)$ (truncated SVD decomposition for $k \ll N$, and the cost of training from $\mathcal{O}(N^2)$ to (Nk) in each iteration of training.

Key words: Collocation method, Robust loss, Physics Informed Neural Networks, Singular Value Decomposition, Non-stationary advection-diffusion

1. INTRODUCTION

Recently, Physics Informed Neural Networks (PINN) have become a very popular method for solving Partial Differential Equations (PDEs). They were introduced by George Karniadakis [1]. PINNs have multiple applications, including nano-optics and nano-materials [2], biomedicine [3], high-speed flows [4], wave propagation problems [5], point-source PDEs [6], and transient problems [7]. The extension of PINNs into weak residuals (Variational PINNs) is presented in [8]. The loss functions of PINNs or VPINNs are generally not robust. They are not equal to the true error (the difference between the exact solution and the numerical solution as provided by the neural network). The extension of VPINN into robust loss functions (Robust VPINNs), which are related to the true error, is presented in [9]. A method to speed up the RVPINN by introducing the collocation method (Collocation-based RVPINN) and discrete weak residuals is presented in [10]. The RVPINN or CRVPINN methods require the inversion of a carefully constructed Gram matrix and the multiplication of the residual loss by the inverse of the Gram matrix. In this paper, we analyze how performing Singular Value Decomposition instead of inversion can speed up the training process. In particular, we do not compute the full inverse of the Gram matrix; instead, we replace it by computing the partial SVD of the Gram matrix and using its inverse to approximate the robust loss function.

The structure of this paper is as follows. In Section 2, we summarize Physics Informed Neural Networks (PINNs)

[11] using the example of the Eriksson-Johnson problem (a 2D stationary advection-diffusion problem). We show that the PINN loss is not robust (it is not related to the true error). In Section 3, we summarize the Collocation-based Robust Variational Physics Informed Neural Networks (CRVPINN) [10] for stationary advection-diffusion problems, and we show that the CRVPINN loss is robust (related to the true error). In the next Section 4, we generalize the idea of CRVPINN to time-dependent advection-diffusion problems. Section 5 discusses the Singular Value Decomposition method to speed up the CRVPINN training. The numerical results are presented in Section 6. We conclude the paper in Section 7. We also present a link to the time-dependent CRVPINN code in Section 8.

2. PHYSICS INFORMED NEURAL NETWORKS FOR STATIONARY ADVECTION-DIFFUSION EQUATIONS

For $\Omega = (0, 1)^2 \subset \mathbb{R}^2$ we seek solution $\Omega \ni (x_1, x_2) \rightarrow u(x_1, x_2) \in \mathcal{R}$ of the model Eriksson-Johnson problem [12], a challenging model problem for verification of numerical methods,

$$\begin{cases} \beta \cdot \nabla u - \varepsilon \Delta u = 0 & \text{in } \Omega \\ u = g & \text{over } \partial\Omega, \end{cases}$$

with $\beta = (1, 0)$, $\varepsilon = \{0.1, 0.01\}$, with g such that

$$\begin{aligned} g(0, x_2) &= \sin(\pi x_2) \text{ for } x_2 \in (0, 1), \\ g(1, x_2) &= 0 \text{ for } x_2 \in (0, 1) \\ g(x_1, 0) &= 0 \text{ for } x_1 \in (0, 1), \\ g(x_1, 1) &= 0 \text{ for } x_1 \in (0, 1). \end{aligned}$$

*e-mail: sluzalec@agh.edu.pl

This problem has a known exact solution

$$u_{\text{exact}}(x, y) = \frac{(e^{r_1(x-1)}) - e^{r_2(x-1)})}{(e^{(-r_1)} - e^{(-r_2)})} \sin(\pi y),$$

$$r_1 = \frac{(1 + \sqrt{(1 + 4\varepsilon^2 \pi^2)})}{(2\varepsilon)}, r_2 = \frac{(1 - \sqrt{(1 + 4\varepsilon^2 \pi^2)})}{(2\varepsilon)}.$$

We define the shift u_{shift} such that

$$u(x_1, x_2) = u_0(x_1, x_2) + u_{\text{shift}}(x_1, x_2),$$

$$u_{\text{shift}}(x_1, x_2) = (1 - x_1) \sin(\pi x_2)$$

$u_0(x_1, x_2) = u(x_1, x_2) - u_{\text{shift}}(x_1, x_2) = 0$ for $(x_1, x_2) \in \partial\Omega$. The shift function is supposed to match the boundary condition on the $x_1 = 0$ boundary of the computational domain. The function $u_{\text{shift}}(0, x_2) = \sin(\pi x_2)$ which is equal to the requested boundary condition. Using the shift, we reformulate the problem with zero Dirichlet b.c.: We seek $\Omega \ni (x_1, x_2) \rightarrow u_0(x_1, x_2) \in \mathcal{R}$, such that

$$\begin{cases} \beta \cdot \nabla u_0 - \varepsilon \Delta u_0 = -\underbrace{\beta \cdot \nabla u_{\text{shift}} + \varepsilon \Delta u_{\text{shift}}}_f & \text{in } \Omega \\ u = 0 & \text{over } \partial\Omega, \end{cases}$$

We define $\Omega = (0, 1)^2$, and we will introduce the discrete domain constructed using the spatial resolution $h = \frac{1}{N}$. The discrete domain

$$\Omega_h = \{(ih, jh) \in \mathbb{R}^2 : 0 \leq i \leq N, 0 \leq j \leq N\}. \quad (1)$$

with the interior

$$\Omega_h^0 = \{(ih, jh) \in \mathbb{R}^2 : 0 < i < N, 0 < j < N\}. \quad (2)$$

In the PINN approach, the neural network $\hat{u}(x, y; \theta)$ represents the solution approximation, namely,

$$\hat{u}(x, y; \theta) = (\mathcal{A}_L \circ \sigma \circ \mathcal{A}_{L-1} \circ \dots \circ \sigma \circ \mathcal{A}_1)(\mathbf{v}), \quad (3)$$

where $\mathbf{v} = [x, y]^T$ is the input vector, $\mathcal{A}_i(\mathbf{z}) = \mathbf{W}_i \mathbf{z} + \mathbf{b}_i$ represents the affine transformation of the i -th layer, and σ is the activation function (sigmoid). Here the $\theta = \{\mathbf{W}_i, \mathbf{b}_i\}$, represents the neural networks parameters.

Remark 1. The points $\{x_{i,j}\}_{i,j}$ employed by the training of the NN for evaluation of the PDE residual are selected from the Ω_h^0 space, and the zero Dirichlet b.c. is enforced strongly.

We can form a vector of residuals

$$\beta \cdot \nabla u_0(x_{i,j}) - \varepsilon \Delta u_0(x_{i,j}) + \beta \cdot \nabla u_{\text{shift}}(x_{i,j}) - \varepsilon \Delta u_{\text{shift}}(x_{i,j}). \quad (4)$$

The θ represents the neural networks parameters. The solution $u_0(x_1, x_2)$ is now represented by the neural network parametrization minus the shift function, namely $u_0(x_1, x_2) = \hat{u}(x_1, x_2) - u_{\text{shift}}(x_1, x_2) = (\mathcal{A}_L \circ \sigma \circ \mathcal{A}_{L-1} \circ \dots \circ \sigma \circ \mathcal{A}_1)([x_1, x_2]^T) - u_{\text{shift}}(x_1, x_2)$. The spatial derivatives are calculated via automatic differentiation.

The numerical results for the training of PINN for the Eriksson-Johnson problem with $\varepsilon = 0.1$ are presented in Figures 1-2. In Figure 1 there is a two orders of magnitude difference between the true error and the PINN loss values. Updating the loss function into the robust will make this difference

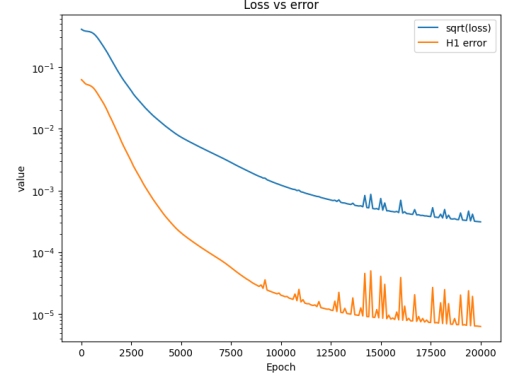


Fig. 1. Convergence of PINN training and comparison between the robust loss and the true error defined as $\|u_{\text{exact}} - u_\theta\|_{H_0^1(\Omega)}$.

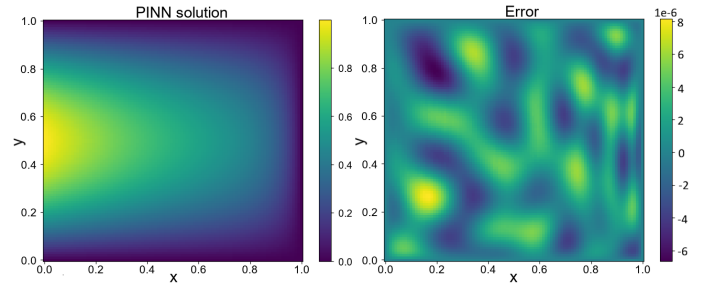


Fig. 2. Left panel: The solution obtained from PINN training. Right panel: The H_0^1 norm difference between the known exact solution and the PINN solution.

much smaller. In Figure 3 the true error and the robust loss are almost identical until the iteration of 5000. Later, they are very close to each other, and this difference is now related with the number of points employed for training. We will explain now how to make the loss function robust.

3. COLLOCATION-BASED ROBUST VARIATIONAL PHYSICS INFORMED NEURAL NETWORKS FOR STATIONARY ADVECTION-DIFFUSION EQUATIONS

Now, we summarize the CRVPINN method introduced in [10] to solve the Eriksson-Johnson model problem in 2D. We define the space of discrete functions over Ω_h

$$L^2(\Omega_h) = \{u: \Omega_h \rightarrow \mathbb{R}\}. \quad (5)$$

We also introduce the discrete inner product

$$(u, v)_{L^2(\Omega_h)} = h^2 \sum_{p \in \Omega_h} u(p)v(p), \quad (6)$$

inducing the norm

$$\|u\|_{L^2(\Omega_h)}^2 = (u, u)_{L^2(\Omega_h)} = h^2 \sum_{p \in \Omega_h} u(p)^2. \quad (7)$$

We also define the functional space over the discrete domain

$$L_0^2(\Omega_h) = \{\Omega_h \ni (x) \rightarrow u(x) \in \mathcal{R}, \sum_{p \in \Omega_h} u^2(p) < \infty; u(x) = 0, x \in \partial\Omega_h\}. \quad (8)$$

Now, for $u: \Omega_h \rightarrow \mathbb{R}$ we denote

$$u_{i,j} = u(x_{i,j}) = u(ih, jh). \quad (9)$$

	$(\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))$	$(\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))$
$(m,n)=(i,j)$	-1	-1
$(m+1,n)=(i,j)$	1	1
$(m,n+1)=(i,j)$	0	0

Table 1. The values of the terms $(\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))$ and $(\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))$ with the differences of Kronecker deltas computed at nodes $x_{i,j}$ forming the Gram matrix entries. Different rows correspond to different value of i, j .

	$(\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))$	$(\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))$
$(m,n)=(i,j)$	-1	-1
$(m+1,n)=(i,j)$	0	0
$(m,n+1)=(i,j)$	1	1

Table 2. The values of the terms $(\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))$ and $(\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))$ with the differences of Kronecker deltas computed at nodes $x_{i,j}$ forming the Gram matrix entries. Different rows correspond to different value of i, j .

We introduce the discrete gradient operators

$$\nabla_+ u_{i,j} = (\nabla_x u_{i,j}, \nabla_y u_{i,j}) = \left(\frac{u_{i+1,j} - u_{i,j}}{h}, \frac{u_{i,j+1} - u_{i,j}}{h} \right), \quad (10)$$

$$\nabla_- u_{i,j} = (\nabla_x u_{i,j}, \nabla_y u_{i,j}) = \left(\frac{u_{i,j} - u_{i-1,j}}{h}, \frac{u_{i,j} - u_{i,j-1}}{h} \right). \quad (11)$$

We define them equal to 0 in the points outside the boundary. Note that the standard 5-point discrete Laplace operator

$$\Delta_h u_{i,j} = \frac{1}{h^2} (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}), \quad (12)$$

can be expressed as $\Delta_h = \nabla_+ \circ \nabla_- = \nabla_- \circ \nabla_+$. Finally, we introduce discrete H_0^1 inner product

$$H_0^1(\Omega_h) = \{ \Omega_h \ni x_{i,j} \rightarrow u_{i,j} \in \mathcal{R}, \quad u \in L_0^2(\Omega_h), \nabla_x u \in L_0^2(\Omega_h), \nabla_y u \in (L_0^2(\Omega_h)) \}, \quad (13)$$

and norm

$$(u, v)_{H_0^1(\Omega_h)} = (\nabla_x u, \nabla_x v)_{L^2(\Omega_h)} + (\nabla_y u, \nabla_y v)_{L^2(\Omega_h)} \quad (14)$$

$$= \sum_{\substack{0 \leq i < N \\ 0 \leq j < N}} (u_{i+1,j} - u_{i,j})(v_{i+1,j} - v_{i,j}) + \sum_{\substack{0 \leq i < N \\ 0 \leq j < N}} (u_{i,j+1} - u_{i,j})(v_{i,j+1} - v_{i,j}). \quad (15)$$

where

$$\|u\|_{H_0^1(\Omega_h)}^2 = (u, u)_{H_0^1(\Omega_h)}. \quad (17)$$

We introduce Kronecker delta test functions $\Omega_h \ni x \mapsto \delta_{i,j}(x) \in \mathbb{R}$

$$\delta_{i,j}(x) = \begin{cases} 1 & \text{for } x = x_{i,j} \\ 0 & \text{otherwise} \end{cases}. \quad (18)$$

We derive now the formula for the Gram matrix using the inner product. The Gram matrix $G_{i,j;k,l} = (\delta_{i,j}, \delta_{k,l})_{H_0^1(\Omega_h)} =$

$$= \sum_{\substack{0 \leq m < N \\ 0 \leq n < N}} (\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))(\delta_{k,l}(x_{m+1,n}) - \delta_{k,l}(x_{m,n})) + \sum_{\substack{0 \leq m < N \\ 0 \leq n < N}} (\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))(\delta_{k,l}(x_{m,n+1}) - \delta_{k,l}(x_{m,n}))$$

For $(i, j) = (k, l)$ we have $G_{i,j;k,l} =$

$$= \sum_{\substack{0 \leq m < N \\ 0 \leq n < N}} (\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))(\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n})) + \sum_{\substack{0 \leq m < N \\ 0 \leq n < N}} (\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))(\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n})) = 4$$

	$(\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))$	$(\delta_{i+1,j}(x_{m+1,n}) - \delta_{i+1,j}(x_{m,n}))$
$(m,n)=(i,j)$	-1	1
$(m+1,n)=(i,j)$	1	0
$(m,n+1)=(i,j)$	0	0

Table 3. The values of the terms $(\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))$ and $(\delta_{i+1,j}(x_{m+1,n}) - \delta_{i+1,j}(x_{m,n}))$ with the differences of Kronecker deltas computed at nodes $x_{i,j}$ forming the Gram matrix entries. Different rows correspond to different value of i, j .

	$(\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))$	$(\delta_{i+1,j}(x_{m,n+1}) - \delta_{i+1,j}(x_{m,n}))$
$(m,n)=(i,j)$	-1	0
$(m+1,n)=(i,j)$	0	0
$(m,n+1)=(i,j)$	1	0

Table 4. The values of the terms $(\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))$ and $(\delta_{i+1,j}(x_{m,n+1}) - \delta_{i+1,j}(x_{m,n}))$ with the differences of Kronecker deltas computed at nodes $x_{i,j}$ forming the Gram matrix entries. Different rows correspond to different value of i, j .

This is illustrated in Tables 1-2.

For $(i+1, j) = (k, l)$ we have $G_{i,j;k,l} =$

$$= \sum_{\substack{0 \leq m < N \\ 0 \leq n < N}} (\delta_{i,j}(x_{m+1,n}) - \delta_{i,j}(x_{m,n}))(\delta_{i+1,j}(x_{m+1,n}) - \delta_{i+1,j}(x_{m,n})) + \sum_{\substack{0 \leq m < N \\ 0 \leq n < N}} (\delta_{i,j}(x_{m,n+1}) - \delta_{i,j}(x_{m,n}))(\delta_{i+1,j}(x_{m,n+1}) - \delta_{i+1,j}(x_{m,n})) = -1$$

This is illustrated in Tables 3-4.

$$G_{i,j;k,l} = \begin{cases} 4 & \text{for } (i, j) = (k, l) \\ -1 & \text{for } (k, l) \in \{(i+1, j), (i-1, j)\} \\ -1 & \text{for } (k, l) \in \{(i, j+1), (i, j-1)\} \end{cases} \quad (19)$$

Remark 2. The inner product selected for the Gram matrix is induced by the norm for which the form $b(u, v)$ of the discrete weak form of the PDE is bounded inf-sup stable.

Theorem 1. For a neural network solution u_θ , the norm for which the form $b(u, v)$ of the discrete weak formulation of the PDE is a bounded inf-sup stable, for u_{exact} the exact solution of the discrete weak formulation, and for the robust loss function $\mathcal{L}_{\text{crvpinn}}(\theta)$

$$\mathcal{L}_{\text{crvpinn}}(\theta) = \text{RES}^T(\theta) \times \mathbf{G}^{-1} \times \text{RES}(\theta) \quad (20)$$

it holds

$$\frac{\sqrt{\mathcal{L}_{\text{crvpinn}}(\theta)}}{\mu} \leq \|u_\theta - u_{\text{exact}}\|_{H_0^1(\Omega_h)} \leq \frac{\sqrt{\mathcal{L}_{\text{crvpinn}}(\theta)}}{\alpha} \quad (21)$$

For the Laplace problems, $\mu = \alpha = 1$.

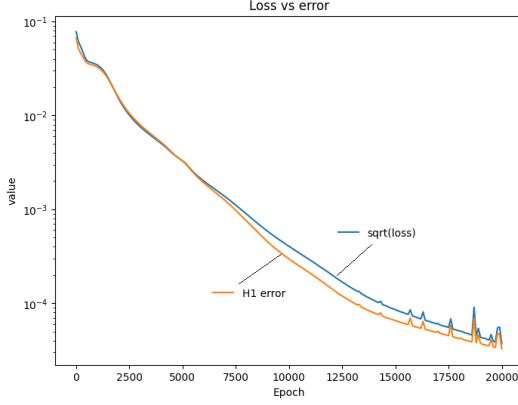


Fig. 3. Convergence of CRVPINN training and comparison between the robust loss and the true error defined as $\|u_{\text{exact}} - u_{\theta}\|_{H_0^1(\Omega)}$.

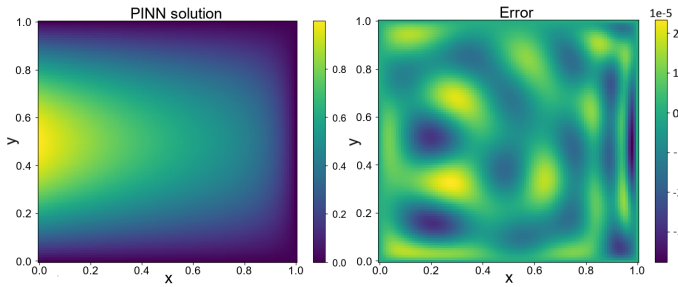


Fig. 4. Numerical results of CRVPINN.

We come back now to the Eriksson-Johnson model problem

1. We derive a discrete weak formulation,

$$\begin{aligned} b(u, v) &= l(v) \quad \forall v \in H_0^1(\Omega_h) \\ b(u, v) &= (\beta_x \nabla_{x+} u, v)_{L^2(\Omega_h)} + (\beta_y \nabla_{y+} u, v)_{L^2(\Omega_h)} + \\ &\quad \varepsilon (\nabla_{x+} u, \nabla_{x+} v)_{L^2(\Omega_h)} \\ l(v) &= (f, v)_{L^2(\Omega_h)}. \end{aligned}$$

The residual of the advection-diffusion problem is

$$r(u, v) = b(u, v) - l(v),$$

where b and l are the corresponding discrete bilinear and linear forms over $(D_{0,h})^2$ and $D_{0,h}$, respectively.

The numerical results for the training of CRVPINN for the Eriksson-Johnson problem with $\varepsilon = 0.1$ are presented in Figures 3-4. This time the robust loss is related to the true error. Small discrepancies result from the discretization error.

4. COLLOCATION-BASED ROBUST VARIATIONAL PHYSICS INFORMED NEURAL NETWORKS FOR TIME-DEPENDENT ADVECTION-DIFFUSION EQUATIONS

In this paper, we generalize this approach into time-dependent formulations. As a model example, we consider the advection-diffusion equations defined over $\Omega \subset \mathbb{R}^2$ and the time interval $I = (0, T)$ with $T > 0$. We seek the pollution concentration scalar field $\Omega \times I \ni (\mathbf{x}, t) \rightarrow u(\mathbf{x}, t) \in \mathcal{R}$ as a result of the pollution generated by a moving vehicle. We solve

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot (\varepsilon \nabla u) + \mathbf{b} \cdot \nabla u = f(\mathbf{x}, t), & \text{in } \Omega \times I, \\ u = 0, & \text{on } \partial\Omega \times I, \\ u(\mathbf{x}, 0) = u_0, & \text{on } \Omega \times \{0\}. \end{cases} \quad (22)$$

In this equation, the ε represents the diffusion coefficient, $\mathbf{b}(\mathbf{x}, t)$ is the advection field that models the vertical temperature gradient (the air movement enforced by the temperature gradient), and u_0 is a zero pollution concentration initial state. The pollution source is modeled by the following formula

$$f(\mathbf{x}, t) = \frac{1}{2} \exp\left(\frac{-x_2^2}{0.32}\right) \frac{1}{2} \left(1 - \cos\left(\pi \frac{(10t - x_1)}{0.8}\right)\right) \quad (23)$$

for $x_2 < 0.2$, and 0 otherwise. It models a moving source of pollution, such as a driving car.

We will derive the discrete weak residual and introduce the test functions as the Kronecker delta functions, following the idea of Collocation-based Robust Variational Physics Informed Neural Networks [10]. We will replace the continuous domain with a discrete domain constructed over the discrete set of collocation points. We will also replace the gradients with discrete finite-difference gradients computed over the collocation points. In order to do that, we first define the set of points:

$$\begin{aligned} \Omega_h \times I_h &:= \{(ih_x, jh_y, kh_t) \in (0, 1)^2 \times (0, T) \\ &\quad : 0 \leq i \leq N_x, 0 \leq j \leq N_y, 0 \leq k \leq N_t\}, \end{aligned} \quad (24)$$

where $h_x = 1/N_x, h_y = 1/N_y, h_t = T/N_t$. We introduce the space

$$D_h := \{u : \Omega_h \times I_h \rightarrow \mathbb{R}\}. \quad (25)$$

We also introduce the discrete inner product and norm

$$(u, v)_h := h_x h_y h_t \sum_{p \in \Omega_h \times I_h} u(p) v(p), \quad \|u\|_h^2 := (u, u)_h, \quad u, v \in D_h. \quad (26)$$

For simplicity, we denote

$$u_{i,j,k} := u(ih_x, jh_y, kh_t), \quad 0 \leq i \leq N_x, 0 \leq j \leq N_y, 0 \leq k \leq N_t, \quad (27)$$

We also introduce a space with zero-Dirichlet boundary conditions at the initial state

$$D_{0,h} = \{u \in D_h : u_{i,j,0} = 0, 0 \leq i \leq N_x, 0 \leq j \leq N_y\}. \quad (28)$$

We introduce a set of Kronecker deltas $\delta_{i,j,k} : \Omega_h \times I_h \rightarrow \mathbb{R}$, $0 \leq i \leq N_x, 0 \leq j \leq N_y, 0 \leq k \leq N_t$, over $\Omega_h \times I_h$,

$$\delta_{i,j,k}(\mathbf{x}, t) = \begin{cases} 1 & \text{if } \mathbf{x} = x_{i,j,k}, \\ 0 & \text{if } \mathbf{x} \neq x_{i,j,k}. \end{cases} \quad (29)$$

We introduce finite difference gradient operations, as given by

$$\nabla_{+} u_{i,j,k} := (\nabla_{x+} u_{i,j,k}, \nabla_{y+} u_{i,j,k}, \nabla_{t+} u_{i,j,k}) := \quad (30)$$

$$\left(\frac{u_{i+1,j,k} - u_{i,j,k}}{h_x}, \frac{u_{i,j+1,k} - u_{i,j,k}}{h_y}, \frac{u_{i,j,k+1} - u_{i,j,k}}{h_t} \right), \quad (31)$$

$$\nabla_{-} u_{i,j,k} := (\nabla_{x-} u_{i,j,k}, \nabla_{y-} u_{i,j,k}, \nabla_{t-} u_{i,j,k}) := \quad (32)$$

$$\left(\frac{u_{i,j,k} - u_{i-1,j,k}}{h_x}, \frac{u_{i,j,k} - u_{i,j-1,k}}{h_y}, \frac{u_{i,j,k} - u_{i,j,k-1}}{h_t} \right), \quad (33)$$

for $0 \leq i \pm 1, j \pm 1 \leq N$. As a result, we can define the following discrete inner product according to these gradient values:

$$\begin{aligned} (u, v)_{\nabla, h} &:= (\nabla_{x+} u, \nabla_{x+} v)_{h_x} + (\nabla_{y+} u, \nabla_{y+} v)_{h_y} + (\nabla_{t+} u, \nabla_{t+} v)_{h_t} \\ &= (\nabla_{x-} u, \nabla_{x-} v)_{h_x} + (\nabla_{y-} u, \nabla_{y-} v)_{h_y} + (\nabla_{t-} u, \nabla_{t-} v)_{h_t}, \end{aligned} \quad (34)$$

$$(35)$$

with the corresponding induced norm

$$k \|u\|_{\nabla, h}^2 := (u, u)_{\nabla, h} = \|\nabla_{x+} u\|_{h_x}^2 + \|\nabla_{y+} u\|_{h_y}^2 + \|\nabla_{t+} u\|_{h_t}^2. \quad (36)$$

We now rewrite the weak formulation into the discrete weak form.

$$\nabla_{t+} u + \beta_x \nabla_{x+} u + \beta_y \nabla_{y+} u - \varepsilon \Delta_h u = f, \quad (37)$$

where $f, \beta_x, \beta_y \in D_{0,h}$ are the source and advection coefficients, and Δ_h is the discrete Laplacian

$$\begin{aligned} \Delta_h u_{i,j,k} &:= \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{h_x^2} + \frac{u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{h_y^2} \\ &\quad + \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{h_t^2}. \end{aligned} \quad (38)$$

Testing with Kronecker Deltas (29) denoted for simplicity as $v \in D_{0,h}$,

$$(\nabla_{t+} u + \beta_x \nabla_{x+} u + \beta_y \nabla_{y+} u - \varepsilon \Delta_h u, v)_h = (f, v)_h, \quad (39)$$

we obtain the following discrete weak variational reformulation: Find $u \in D_{0,h}$ such that:

$$\overbrace{(\nabla_{t+} u + \beta_x \nabla_{x+} u + \beta_y \nabla_{y+} u, v)_h}^{b(u,v)} + \varepsilon \overbrace{(\nabla_{x+} u, \nabla_{x+} v)_h}^{l(v)} = (f, v)_h \quad \forall v \in D_{0,h}, \quad (40)$$

where b and l are the corresponding discrete bilinear and linear forms over $(D_{0,h})^2$ and $D_{0,h}$, respectively.

Following [10], we construct the robust collocation-based loss function as:

$$\text{RES}(u) = \{b(u, r(u)_{i,j}) - l(r(u)_{i,j})\}_{0 < i < N_x, 0 < j < N_y, 0 < k < N_t} \quad (41)$$

$$\mathcal{L}_{\text{crvpinn}}(u) = \text{RES}(u)^T \mathbf{G}^{-1} \text{RES}(u). \quad (42)$$

Here $\mathbf{G}$ is the Gram matrix of the inner product $(\cdot, \cdot)_{\nabla, h}$. We will now construct the Gram matrix by employing the Kronecker delta test functions as follows:

$$\mathbf{G}_{i,j,k;l,m,n} = \frac{1}{h_x h_y h_t} \begin{cases} 6, & \text{if } (l, m, n) = (i, j, k), \\ -1, & \text{if } (l, m, n) = (i \pm 1, j, k), \\ -1, & \text{if } (l, m, n) = (i, j \pm 1, k), \\ -1, & \text{if } (l, m, n) = (i, j, k \pm 1), \\ 0, & \text{otherwise.} \end{cases} \quad (43)$$

For this robust loss (42) we can show (see [10])

Theorem 1 (Robustness). *Let $u \in D_{0,h}$ and let $\mathcal{L}_{\text{crvpinn}}(u)$ be the robust loss (42). Then,*

$$\frac{1}{\mu} \mathcal{L}_{\text{crvpinn}}(u) \leq \|u - u_{\text{exact}}\|_{\nabla, h} \leq \frac{1}{\alpha} \mathcal{L}_{\text{crvpinn}}(u), \quad (44)$$

where u_{exact} denotes the exact solution to the discrete weak problem (40) and μ and α are the boundedness and coercivity constants of b , respectively.

This is not the case for the traditional residual-based loss employed by Physics Informed Neural Networks,

$$\mathcal{L}_{\text{pinn}}(u) = \text{RES}(u)^T \text{RES}(u). \quad (45)$$

5. SINGULAR VALUE DECOMPOSITION TO SPEED UP THE ROBUST LOSS EVALUATION

The inverse of the Gram matrix or the singular value decomposition of the Gram matrix is performed only once at the beginning of training. We, however, have to apply the loss function many times during training (we multiply the inverse of the Gram matrix by the residual vector). The goal of this section is to reduce the cost of this application by performing the singular value decomposition (SVD) of the Gram matrix

$$\text{SVD}(\mathbf{G}) = \mathcal{V}_G^T \mathcal{S}_G \mathcal{U}_G \quad (46)$$

The inverse of the Gram matrix is now given by

$$\mathbf{G}^{-1} = \mathcal{U}_G \mathcal{S}_G^{-1} \mathcal{V}_G^T \quad (47)$$

where $\mathcal{S}_G^{-1}$ is a diagonal matrix and its entries $\mathcal{S}_{Gii}^{-1} = \frac{1}{\mathcal{S}_{Gii}}$ are the inverses of the diagonal elements of $\mathcal{S}_G$. We can now introduce the truncated SVD approximation of the inverse

$$\text{rSVD}(\mathbf{G}^{-1}) = \hat{\mathcal{U}}_G \hat{\mathcal{S}}_G^{-1} \hat{\mathcal{V}}_G^T \quad (48)$$

where $\hat{\mathcal{U}}_G = \mathcal{U}_G[1:N, 1:k]$, $\hat{\mathcal{S}}_G^{-1} = \mathcal{S}_G^{-1}[1:k, 1:k]$, and $\hat{\mathcal{V}}_G = \mathcal{V}_G[1:N, 1:k]$. Then, we employ the approximate loss function

$$\hat{\mathcal{L}}_{\text{crvpinn}} = \text{RES}(u)^T \hat{\mathcal{U}}_G \hat{\mathcal{S}}_G^{-1} \hat{\mathcal{V}}_G^T \text{RES}(u). \quad (49)$$

Actually, we do not need to compute the full SVD decomposition or inversion of the Gram matrix $\mathbf{G}$, which costs $\mathcal{O}(N^3)$. We can use the truncated SVD decomposition, computing k singular values of $\mathbf{G}$ as in (48), which costs $\mathcal{O}(N^2 k)$. We can use it to compute the approximated robust loss as in (49). The computational cost of applying the full loss is the cost of multiplying a residual vector by the inverse of the Gram matrix, which is $\mathcal{O}(N^2)$. The computational cost of applying the approximate loss function is the cost of the multiplications of truncated matrices or, alternatively, the cost of k vector-by-vector multiplications, which is $\mathcal{O}(Nk)$. Thus, we have reduced the cost of computing the loss from quadratic to semi-linear cost.

6. NUMERICAL RESULTS

In the numerical experiments, the implementations of both PINN and CRVPINN methods employ the computations of the gradients via Automatic Differentiation.

We start from selecting a proper number of collocation points for training of CRVPINN method. We test the convergence of CRVPINN training for 12, 25, 50, and 100 points in each direction. The results are presented in Figure 5. Increasing the number of collocation points, the robust loss is related

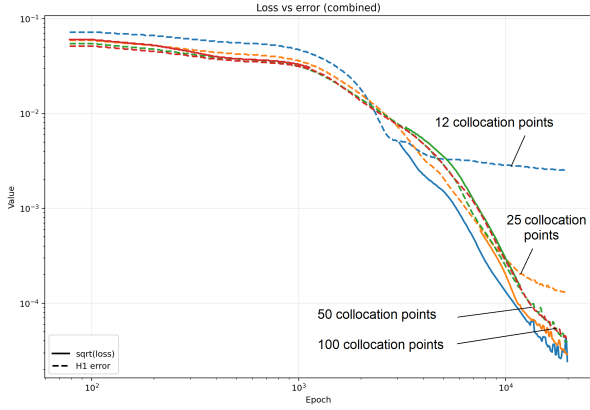


Fig. 5. The convergence of CRVPINN for increasing number of collocation points, from 12, 25, 50, 100, and its comparison with the true error for Erikson-Johnson problem with $\varepsilon = 0.1$.

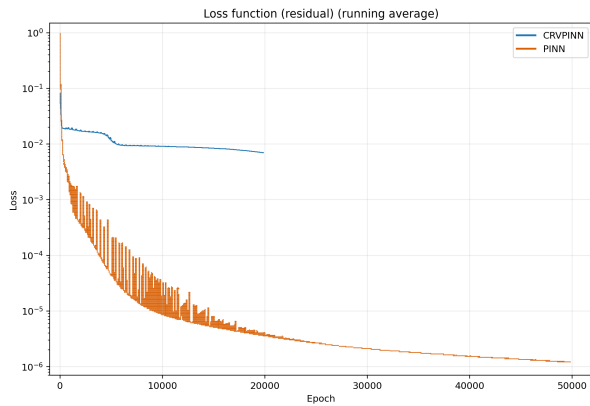


Fig. 6. Convergence of the PINN and CRVPINN training.

with the true error for a larger number of iterations. Based on the numerical experiments we select 25 collocation points in each direction as a baseline for the following numerical experiments.

We compare the convergence of the PINN loss (45) with the convergence of the CRVPINN loss (42) for the inverse of the full Gram matrix in Figure 6. The PINN loss converges to the value of 10^{-6} , but there is no guaranty that the accuracy of the solution is of such high order. The CRVPINN loss, on the contrary, converges to the value of 10^{-2} , which is a good estimator of the true error [10].

The numerical results obtained with this CRVPINN robust loss using the inverse of the full Gram matrix are presented in Figure 7. Pollution is generated by the vehicle moving in the right direction. As expected, it accumulates close to the ground.

Now, we perform tests to check the accuracy and convergence of the approximate loss with $k = 10, 50, 100$ and the full loss of $\mathbf{G}^{-1}$. We run 20,000 iterations of training for a fully connected neural network with 2 layers and 100 neurons per layer. We select $25 \times 25 \times 25$, a total of 15,625 collocation points for training. This also represents the size of the Gram matrix and the dense inverse. The convergence of the residual loss is presented in Figure 8. We can see that the loss function for $k = 10$ converges to the value of 10^{-4} , which is too

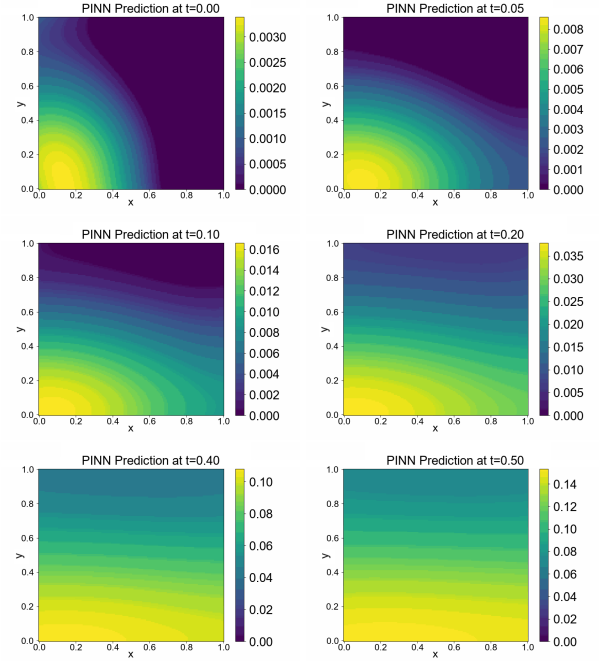


Fig. 7. Snapshots of the simulation of pollution propagation from moving vehicle, using the CRVPINN method.

low (it overestimates the accuracy of the solution). However, for a higher number of singular values $k = 50, 100$, the convergence of the loss appears natural; the loss function converges to a value on the order of 10^{-2} , similar to the convergence of the full loss function. We conclude that $k = 50$ is an appropriate number of singular values since, for $k = 10$, the loss is no longer robust.

How does it influence the accuracy of the numerical results? The exact initial state is zero everywhere. Neither the full loss function nor the approximate loss function learns this zero initial state exactly, but the obtained results are very close to zero. This is presented in Figure 9. The zero initial state is approximated with a maximum value of 0.012 for the full loss. For $k = 10$, the maximum error is 0.0035. For $k = 50$, we obtain the zero initial state approximation with a maximum error of 0.0175. For $k = 100$, this maximum error is 0.030.

How is the middle state approximated with the full loss and the approximate losses? This is illustrated in Figure 10. The full loss provides a uniform distribution of generated pollution, reaching a maximum value of 0.06, as presented in the last panel. All the presented cases approximate the obtained distribution well.

How is the final state approximated with the full loss and the approximate losses? This is illustrated in Figure 11. The full loss provides a uniform distribution of generated pollution, reaching a maximum value of 0.2, as presented in the last panel. Again, all the presented cases approximate the obtained distribution well.

7. CONCLUSIONS

We have shown that performing a truncated SVD decomposition of the inverse of the Gram matrix, while keeping an appro-

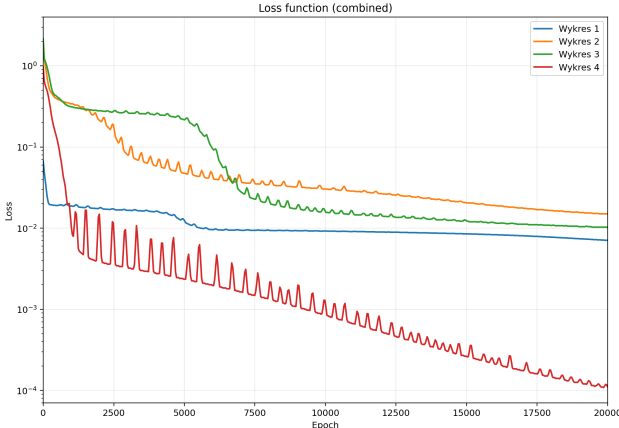


Fig. 8. Residual loss function of CRVPINN for $k = 10$, $k = 50$, and $k = 100$ singular values. Residual loss for full G^{-1} .

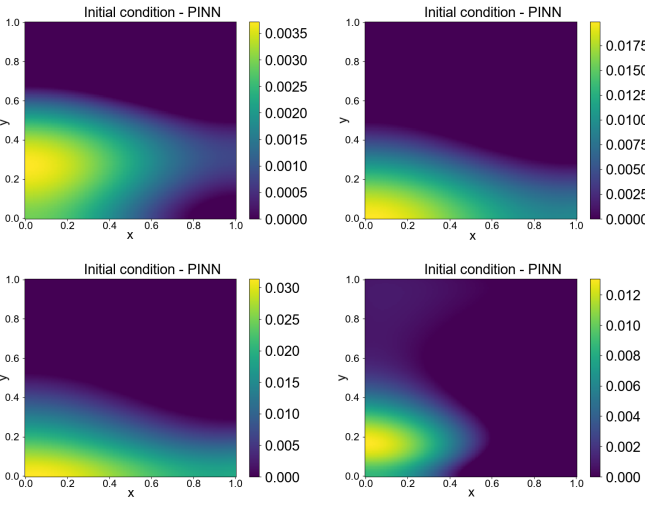


Fig. 9. Initial state trained using $k = 10$, $k = 50$, and $k = 100$ singular values. Initial state for full G^{-1} .

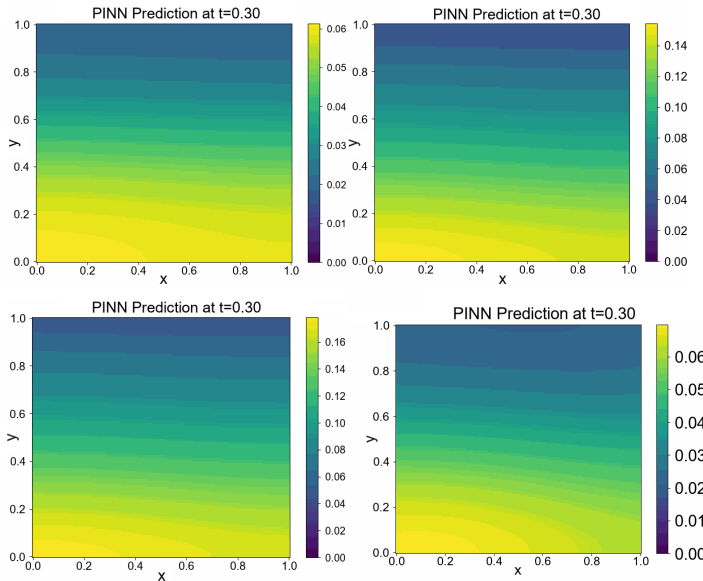


Fig. 10. Snapshot of the simulation, middle time step, for $k = 10$, $k = 50$, and $k = 100$ singular values. Snapshot of the simulation, middle time step, for full G^{-1} .

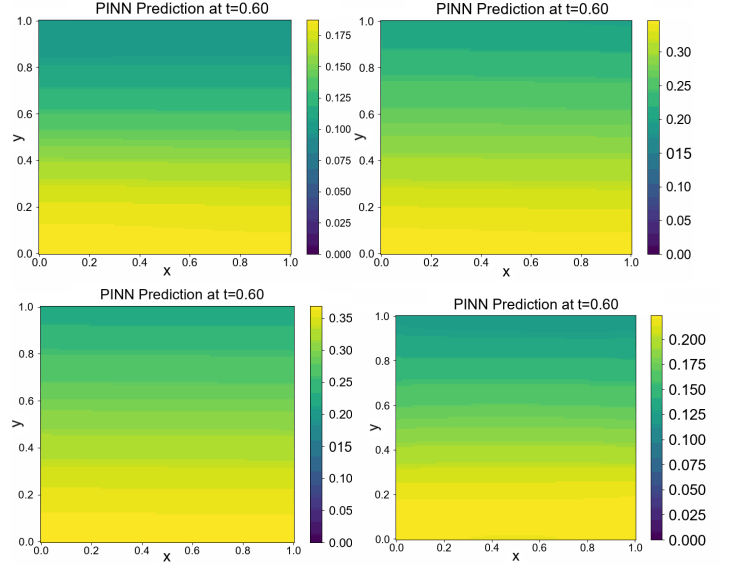


Fig. 11. Snapshot of the simulation, final time step, for $k = 10$, $k = 50$, and $k = 100$ singular values. Snapshot of the simulation, final time step, for full G^{-1} .

appropriate number of singular values (50 in our case), allows us to maintain the quality of the solution and the convergence of the CRVPINN training, and the robustness of the loss function. At the same time, the training of CRVPINN with the truncated SVD approximation of the inverse of the Gram matrix was twice as fast as the original CRVPINN training with the full matrix, reducing the time from 6 seconds per 100 iterations to 3 seconds per 100 iterations (a total of 600 seconds, accounting for 10 minutes instead of the 1200 seconds equivalent to 20 minutes). Even 50 singular values are enough to approximate the solution well. Thus, we recommend this method to speed up the training process.

What does it mean from the point of view of the computational cost? Before training, instead of inverting the Gram matrix, which costs $\mathcal{O}(N^3)$, we can perform truncated SVD, recovering the $k = 50$ largest singular values, which costs $\mathcal{O}(N^2k)$. Next, during the training, instead of full Gram matrix-vector multiplication for the robust loss, which costs $\mathcal{O}(N^2)$ in each iteration, we can perform $k = 50$ vector times vector multiplications, which costs kN . Here, $N = 25^3 = 15,625$ and $k = 50$.

Future work may involve replacing the truncated SVD decomposition of the full matrix with recursive truncated decompositions, as it has already been employed to speed up the GMRES solver computations in [13].

8. CODE

The Python code for the training of time-dependent advection-diffusion equations is available at Google Colab notebook¹.

¹<https://colab.research.google.com/drive/1YQc25IvsNb5meOmojuikWfOs-TrTfGxc>

ACKNOWLEDGEMENTS

This work has been supported by the National Science Centre, Poland grant no. 2025/57/B/ST6/00058. Research project supported by the program “Excellence initiative - research university” for the AGH University of Science and Technology. The authors are grateful for the support from the funds that the Polish Ministry of Science and Higher Education assigned to AGH University of Krakow.

REFERENCES

- [1] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, “Physics-informed neural networks (PINNs) for fluid mechanics: A review,” *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021.
- [2] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro, “Physics-informed neural networks for inverse problems in nano-optics and metamaterials,” *Optics Express*, vol. 28, no. 8, pp. 11 618–11 633, 2020.
- [3] M. Alber, A. B. Tépole, W. R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W. W. Lytton, P. Perdikaris, L. Petzold, and E. Kuhl, “Integrating machine learning and multiscale modeling-perspectives, challenges, and opportunities in the biological biomedical, and behavioral sciences,” *NPJ Digital Medicine*, vol. 2, 2019.
- [4] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, “Physics-informed neural networks for high-speed flows,” *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.
- [5] M. Rasht-Behesht, C. Huber, K. Shukla, and G. E. Karniadakis, “Physics-informed neural networks (pinns) for wave propagation and full waveform inversions,” *Journal of Geophysical Research: Solid Earth*, vol. 127, no. 5, p. e2021JB023120, 2022.
- [6] X. Huang, H. Liu, B. Shi, Z. Wang, K. Yang, Y. Li, M. Wang, H. Chu, J. Zhou, F. Yu, B. Hua, B. Dong, and L. Chen, “A universal pinns method for solving partial differential equations with a point source,” *Proceedings of the Fourteen International Joint Conference on Artificial Intelligence (IJCAI-22)*, pp. 3839–3846, 2022.
- [7] P. Maczuga, M. Sikora, M. Skoczeń, P. Rożnawski, F. Thuszcz, M. Szubert, M. Łoś, W. Dzwiniel, K. Pingali, and M. Paszyński, “Physics informed neural network code for 2d transient problems (pinn-2dt) compatible with google colab,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.03755>
- [8] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, “hp-vpinns: Variational physics-informed neural networks with domain decomposition,” *Computer Methods in Applied Mechanics and Engineering*, vol. 374, p. 113547, 2021.
- [9] S. Rojas, P. Maczuga, J. Muñoz-Matute, D. Pardo, and M. Paszyński, “Robust variational physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 425, p. 116904, 2024.
- [10] M. Łoś, T. Służalec, P. Maczuga, A. Vilkha, C. Uriarte, and M. Paszyński, “Collocation-based robust variational physics-informed neural networks (crvpinns),” *Computers & Structures*, vol. 316, p. 107839, 2025.
- [11] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving non-linear partial differential equations,” *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.
- [12] K. Eriksson and C. Johnson, “Adaptive finite element methods for parabolic problems i: A linear model problem,” *SIAM J. Num. Anal.*, vol. 28, no. 1, pp. 43–77, 1991.
- [13] T. Służalec, M. Dobija, A. Paszyńska, I. Muga, M. Łoś, and M. Paszyński, “Automatic stabilization of finite-element simulations using neural networks and hierarchical matrices,” *Computer Methods in Applied Mechanics and Engineering*, vol. 411, p. 116073, 2023.