

Article

Not peer-reviewed version

Machine Learning Method for Approximate Solutions for Reaction-Diffusion Equations with Multivalued Interaction Functions

[Pavlo O. Kasyanov](#) ^{*}, [Oleksiy V. Kapustyan](#), [Liudmyla B. Levenchuk](#), [Vladyslav R. Novykov](#)

Posted Date: 17 December 2024

doi: [10.20944/preprints202407.2340.v2](https://doi.org/10.20944/preprints202407.2340.v2)

Keywords: reaction-diffusion equations; multivalued interaction functions; machine learning; physics-informed neural networks; approximate solutions



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Machine Learning Method for Approximate Solutions for Reaction-Diffusion Equations with Multivalued Interaction Functions

Pavlo O. Kasyanov ^{1,*}, Oleksiy V. Kapustyan ^{2,†}, Liudmyla B. Levenchuk ^{3,‡}, and Vladyslav R. Novykov ^{4,‡}

¹ Institute for Applied System Analysis, National Technical University of Ukraine 'Igor Sikorsky Kyiv Polytechnic Institute', Beresteiskiy Ave., 37, build. 35, 03056, Kyiv, Ukraine; kasyanov.pavlo@l.kpi.ua

² Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska Street, City of Kyiv, Ukraine; 01601; and Institute for Applied System Analysis, National Technical University of Ukraine 'Igor Sikorsky Kyiv Polytechnic Institute', Beresteiskiy Ave., 37, build. 35, 03056, Kyiv, Ukraine; kapustyan@knu.ua

³ National Technical University of Ukraine 'Igor Sikorsky Kyiv Polytechnic Institute', Beresteiskiy Ave., 37, build. 35, 03056, Kyiv, Ukraine; levenchuk.liudmyla@l.kpi.ua

⁴ National Technical University of Ukraine 'Igor Sikorsky Kyiv Polytechnic Institute', Beresteiskiy Ave., 37, build. 1, 03056, Kyiv, Ukraine; vlad.novykov@gmail.com

* Correspondence: kapustyan@knu.ua

† These authors contributed equally to this work.

Abstract: This paper presents machine learning methods for approximate solutions of reaction-diffusion equations with multivalued interaction functions. This approach addresses the challenge of finding all possible solutions for such equations, which often lack uniqueness. The proposed method utilizes physics-informed neural networks (PINNs) to approximate generalized solutions.

Keywords: reaction-diffusion equations; multivalued interaction functions; machine learning; physics-informed neural networks; approximate solutions

1. Introduction

In this paper we establish machine learning methods for approximate solutions of classes of reaction-diffusion equations with multivalued interaction functions allowing for non-unique solutions of the Cauchy problem. The relevance of this problem is primarily due to the lack of methods for finding all solutions for such mathematical objects. Therefore, there is an expectation that another approximate method will provide us with yet another solution for these problems. In addition, methods for approximate solutions of nonlinear systems with partial derivatives without uniqueness are mostly theoretical and are used primarily in qualitative research [1,2]. The availability of computational power for parallel computations and the creation of open-source software libraries such as PyTorch [3] have stimulated a new wave of development in IT and artificial intelligence methods. Sample-based methods for approximate solutions of such problems were first proposed in [4]. To date, such systems with smooth nonlinearities have been qualitatively and numerically studied. There is a need to develop a methodology for approximating generalized solutions of nonlinear differential-operator systems without uniqueness using recurrent neural networks, sample-based methods, and variations of the Monte Carlo method.

Let $T, \nu > 0$, and $u_0 : \mathbb{R}^2 \rightarrow \mathbb{R}$ be a sufficiently smooth function. We consider the problem:

$$\frac{\partial u}{\partial t}(x, t) \in \nu \nabla^2 u(x, t) - f(u(x, t)), \quad (x, t) \in \mathbb{R}^2 \times [0, T], \quad (1)$$

with initial conditions:

$$u(x, 0) = u_0(x), \quad x \in \mathbb{R}^2, \quad (2)$$

where $f : \mathbb{R} \rightarrow \mathbb{R}$ is a function satisfying the condition of at most linear growth:

$$\exists C > 0 : |f(s)| \leq C(1 + |s|) \quad \text{for each } s \in \mathbb{R}.$$

We note that such nonlinearities appear in impulse feedback control problems, etc. [5–9]. Moreover, the global attractor for solutions of Problem 1 may be a nontrivial set in the general case and can have arbitrarily large fractal dimension. The convergence rate of solutions to the attractor may not be exponential; see [10–14] and references therein.

For a fixed $u_0 \in C_0^\infty(\mathbb{R}^2)$ let $\Omega \subset \mathbb{R}^2$ be a bounded domain with sufficiently smooth boundary and $\text{supp } u_0 \subset \Omega$. According to [1] (see the book and references therein), there exists a weak solution $u = u(x, t) \in L^2(0, T; H_0^1(\Omega))$ with $\frac{\partial u}{\partial t} \in L^2(0, T; H^{-1}(\Omega))$, of Problem (1)–(2) in the following sense:

$$-\int_0^T \int_{\Omega} u(x, t)v(x)\eta_t(t)dxdt + \nu \int_0^T \int_{\Omega} (\nabla u(x, t) \cdot \nabla v(x) + d(x, t)v(x))\eta(t)dt = 0, \quad (3)$$

for all $v \in C_0^\infty(\Omega)$, $\eta \in C_0^\infty(0, T)$, where $d : \mathbb{R} \times [0, T] \rightarrow \mathbb{R}$ be a measurable function such that

$$d(x, t) \in f(u(x, t)) \quad \text{for a.e. } (x, t) \in \mathbb{R}^2 \times (0, T). \quad (4)$$

Such inclusions with multivalued nonlinearities appear in problems of climatology (Budyko–Sellers Model), chemical kinetics (Belousov–Zhabotinsky equations), biology (Lotka–Volterra systems with diffusion), quantum mechanics (FitzHugh–Nagumo system), engineering and medicine (several syntheses and impulse control problems); see [1,2] and references therein.

The main goal of this paper is to develop an algorithm for approximation of solutions for classes of reaction-diffusion equations with multivalued interaction functions allowing for non-unique solutions of the Cauchy problem (1)–(2) via the so-called physics-informed neural networks (PINNs); [15–17] and references therein.

2. Methodology of Approximate Solutions for Reaction-Diffusion Equations with Multivalued Interaction Functions

Fix an arbitrary $T > 0$, and a sufficiently smooth function $u_0 : \mathbb{R}^2 \rightarrow \mathbb{R}$. We approximate the function f by the following Lipschitz functions satisfying the condition of at most linear growth (*Pasch-Hausdorff envelopes*):

$$f_k(x) := \inf_{x' \in \mathbb{R}} \{f(x') + k|x - x'|\}, \quad x \in \mathbb{R}; \quad (5)$$

see [18] and references therein. For a fixed $k = 1, 2, \dots$, consider the problem:

$$\frac{\partial u_k}{\partial t}(x, t) = \nu \nabla^2 u_k(x, t) - f_k(u_k(x, t)), \quad (x, t) \in \mathbb{R}^2 \times [0, T], \quad (6)$$

with initial conditions:

$$u_k(x, 0) = u_0(x), \quad x \in \mathbb{R}^2. \quad (7)$$

According to [2] and references therein, for each $k = 1, 2, \dots$ Problem (6)–(7) has an unique solution $u_k \in C^{2,1}(\mathbb{R}^2 \times [0, T])$. Moreover, [19] implies that each convergent subsequence $\{u_{k_l}\}_{l=1,2,\dots} \subset \{u_k\}_{k=1,2,\dots}$ of corresponding solutions to Problem (6)–(7) weakly converges to a solution u of Problem (1)–(2) in the space

$$W := \{z \in L^2(0, T; H_0^1(\Omega)) : \frac{\partial z}{\partial t} \in L^2(0, T; H^{-1}(\Omega))\} \quad (8)$$

endowed with the standard graph norm, where $\Omega \subset \mathbb{R}^2$ is a bounded domain with sufficiently smooth boundary and $\text{supp } u_0 \subset \Omega$.

Further, to simplify the conclusions we consider as f the following function:

$$f(s) := \begin{cases} \{0\}, & s < 0; \\ [0, 1], & s = 0; \\ \{1\}, & s > 0. \end{cases} \quad (9)$$

We approximate it by the following Lipschitz functions:

$$f_k(s) := \begin{cases} 0, & s < 0; \\ ks, & s \in [0, \frac{1}{k}); \\ 1, & s \geq \frac{1}{k}, \end{cases} \quad k = 1, 2, \dots \quad (10)$$

Thus, the **first step** of the algorithm is to replace the function f in Problem (1)–(2) with f_k considering Problem (6)–(7) for sufficiently large k .

Let us now consider Problem (6)–(7) for sufficiently large k . Theorem 16.1.1 from [15] allows us to reformulate Problem (6)–(7) as an infinite dimensional stochastic optimization problem over a certain function space. More exactly, let $\mathbf{t} \in C([0, T]; (0, \infty))$, $\xi \in C(\mathbb{R}^2; (0, \infty))$, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, let $\mathcal{T} : \Omega \rightarrow [0, T]$ and $\mathcal{X} : \Omega \rightarrow \mathbb{R}^2$ be independent random variables. Assume for all $A \in \mathcal{B}([0, T])$, $B \in \mathcal{B}(\mathbb{R}^2)$ that

$$\mathbb{P}(\mathcal{T} \in A) = \int_A \mathbf{t}(t) dt \quad \text{and} \quad \mathbb{P}(\mathcal{X} \in B) = \int_B \xi(x) dx.$$

Note that $f_k : \mathbb{R} \rightarrow \mathbb{R}$ be Lipschitz continuous, and let $\mathcal{L}_k : C^{2,1}(\mathbb{R}^2 \times [0, T], \mathbb{R}) \rightarrow [0, \infty]$ satisfy for all $v = (v(x, t))_{(x, t) \in \mathbb{R}^2 \times [0, T]} \in C^{2,1}(\mathbb{R}^2 \times [0, T])$ that

$$\mathcal{L}_k(v) = \mathbb{E} \left[|v(\mathcal{X}, 0) - u_0(\mathcal{X})|^2 + \left| \frac{\partial v}{\partial t}(\mathcal{X}, \mathcal{T}) - \nu \nabla^2 v(\mathcal{X}, \mathcal{T}) + f_k(v(\mathcal{X}, \mathcal{T})) \right|^2 \right].$$

Theorem 16.1.1 from [15] implies that the following two statements are equivalent:

1. It holds that $\mathcal{L}_k(u_k) = \inf_{v \in C^{2,1}(\mathbb{R}^2 \times [0, T])} \mathcal{L}_k(v)$.
2. It holds $u_k \in C^{2,1}(\mathbb{R}^2 \times [0, T])$ is the solution of Problem (6)–(7).

Thus, the **second step** of the algorithm is to reduce the regularized Problem (6)–(7) to the infinite dimensional stochastic optimization problem in $C^{2,1}(\mathbb{R}^2 \times [0, T])$:

$$\begin{cases} \mathcal{L}_k(v) \rightarrow \min, \\ v \in C^{2,1}(\mathbb{R}^2 \times [0, T]). \end{cases} \quad (11)$$

However, due to its infinite dimensionality, the optimization problem (11) is not yet suitable for numerical computations. Therefore, we apply the **third step**, the so-called Deep Galerkin Method (DGM) [20], that is, we transform this infinite dimensional stochastic optimization problem into a finite dimensional one by incorporating artificial neural networks (ANNs); see [15, 20] and references therein.

Let $a : \mathbb{R} \rightarrow \mathbb{R}$ be differentiable, let $h \in \mathbb{N}, l_1, l_2, \dots, l_h, \mathfrak{d} \in \mathbb{N}$ satisfy $\mathfrak{d} = 4l_1 + [\sum_{k=2}^h l_k(l_{k-1} + 1)] + l_h + 1$, and let $\mathcal{L}_{k,h} : \mathbb{R}^{\mathfrak{d}} \rightarrow [0, \infty)$ satisfy for all $\theta \in \mathbb{R}^{\mathfrak{d}}$ that

$$\begin{aligned} \mathcal{L}_{k,h}(\theta) &= \mathcal{L}_k(\mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta,3}) \\ &= \mathbb{E} \left[\left| \mathcal{N}_{\mathfrak{M}_{a,l_1}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta,3}(\mathcal{X}, 0) - u_0(\mathcal{X}) \right|^2 \right. \\ &\quad + \left. \left| \frac{\partial \mathcal{N}_{\mathfrak{M}_{a,l_1}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta,3}}{\partial t}(\mathcal{X}, \mathcal{T}) - \nu \nabla^2 \mathcal{N}_{\mathfrak{M}_{a,l_1}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta,3}(\mathcal{X}, \mathcal{T}) \right. \right. \\ &\quad \left. \left. + f_k \left(\mathcal{N}_{\mathfrak{M}_{a,l_1}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta,3}(\mathcal{X}, \mathcal{T}) \right) \right|^2 \right], \end{aligned} \quad (12)$$

where $\mathfrak{M}_{\psi,d}$ is the d -dimensional version of a function ψ , that is,

$$\mathfrak{M}_{\psi,d} : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

is the function which satisfies for all $x = (x_k)_{k \in \{1, 2, \dots, d\}} \in \mathbb{R}^d$, $y = (y_k)_{k \in \{1, 2, \dots, d\}} \in \mathbb{R}^d$ with $\forall k \in \{1, 2, \dots, d\} : y_k = \psi(x_k)$ that

$$\mathfrak{M}_{\psi,d}(x) = y;$$

for each $\mathfrak{d}, L \in \mathbb{N}, l_0, l_1, \dots, l_L \in \mathbb{N}, \theta \in \mathbb{R}^{\mathfrak{d}}$ satisfying $\mathfrak{d} \geq \sum_{k=1}^L l_k(l_{k-1} + 1)$, and for a function $\Psi_k : \mathbb{R}^{l_k} \rightarrow \mathbb{R}^{l_k}, k \in \{1, 2, \dots, L\}$, we denote by $\mathcal{N}_{\Psi_1, \Psi_2, \dots, \Psi_L}^{\theta, l_0} : \mathbb{R}^{l_0} \rightarrow \mathbb{R}^{l_L}$ the realization function of the fully-connected feedforward artificial neural network associated to θ with $L + 1$ layers with dimensions (l_0, l_1, \dots, l_L) and activation functions $(\Psi_1, \Psi_2, \dots, \Psi_L)$, defined as:

$$\begin{aligned} \mathcal{N}_{\Psi_1, \Psi_2, \dots, \Psi_L}^{\theta, l_0}(x) &= (\Psi_L \circ \mathcal{A}_{l_L, l_{L-1}}^{\theta, \sum_{k=1}^{L-1} l_k(l_{k-1} + 1)} \circ \Psi_{L-1} \circ \mathcal{A}_{l_{L-1}, l_{L-2}}^{\theta, \sum_{k=1}^{L-2} l_k(l_{k-1} + 1)} \circ \dots \\ &\quad \dots \circ \Psi_2 \circ \mathcal{A}_{l_2, l_1}^{\theta, l_1(l_0 + 1)} \circ \Psi_1 \circ \mathcal{A}_{l_1, l_0}^{\theta, 0})(x), \end{aligned}$$

for all $x \in \mathbb{R}^{l_0}$; and for each $\mathfrak{d}, m, n \in \mathbb{N}, s \in \mathbb{N}_0 := \mathbb{N} \cup \{0\}, \theta = (\theta_1, \theta_2, \dots, \theta_{\mathfrak{d}}) \in \mathbb{R}^{\mathfrak{d}}$ satisfying $\mathfrak{d} \geq s + mn + m$, the affine function $\mathcal{A}_{s, m, n}^{\theta}$ from \mathbb{R}^n to \mathbb{R}^m associated to (θ, s) , is defined as

$$\mathcal{A}_{s, m, n}^{\theta}(x) = \begin{pmatrix} \theta_{s+1} & \theta_{s+2} & \dots & \theta_{s+n} \\ \theta_{s+n+1} & \theta_{s+n+2} & \dots & \theta_{s+2n} \\ \theta_{s+2n+1} & \theta_{s+2n+2} & \dots & \theta_{s+3n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{s+(m-1)n+1} & \theta_{s+(m-1)n+2} & \dots & \theta_{s+mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \theta_{s+mn+1} \\ \theta_{s+mn+2} \\ \vdots \\ \theta_{s+mn+m} \end{pmatrix}$$

for all $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$.

The **final step** in the derivation involves approximating the minimizer of $\mathcal{L}_{k,h}$ using stochastic gradient descent optimization methods [15]. Let $\xi \in \mathbb{R}^{\mathfrak{d}}, J \in \mathbb{N}, (\gamma_n)_{n \in \mathbb{N}} \subseteq [0, \infty)$, for each $n \in \mathbb{N}, j \in \{1, 2, \dots, J\}$ let $\mathfrak{T} : \Omega \rightarrow [0, T]$ and $\mathfrak{X}_{n,j} : \Omega \rightarrow \mathbb{R}^2$ be random variables. Let for each $n \in \mathbb{N}, j \in \{1, 2, \dots, J\}, A \in \mathcal{B}([0, T]), B \in \mathcal{B}(\mathbb{R}^2)$

$$\mathbb{P}(\mathcal{T} \in A) = \mathbb{P}(\mathfrak{T}_{n,j} \in A) \quad \text{and} \quad \mathbb{P}(\mathcal{X} \in B) = \mathbb{P}(\mathfrak{X}_{n,j} \in B). \quad (13)$$

Let $\ell_{k,h} : \mathbb{R}^d \times \mathbb{R}^2 \times [0, T] \rightarrow \mathbb{R}$ is defined as

$$\begin{aligned} \ell_{k,h}(\theta, x, t) = & \left| \mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta, 3}(x, 0) - u_0(x) \right|^2 \\ & + \left| \frac{\partial \mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta, 3}}{\partial t}(x, t) - \nu \nabla^2 \mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta, 3}(x, t) \right|^2 \\ & + f_k \left(\mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\theta, 3}(x, t) \right)^2, \end{aligned} \quad (14)$$

for each $\theta \in \mathbb{R}^d$, $x \in \mathbb{R}^2$, $t \in [0, T]$, and let $\Theta = (\Theta_n)_{n \in \mathbb{N}_0} : \mathbb{N}_0 \times \Omega \rightarrow \mathbb{R}^d$ satisfy for all $n \in \mathbb{N}$ that

$$\Theta_0 = \xi \quad \text{and} \quad \Theta_n = \Theta_{n-1} - \gamma_n \left[\frac{1}{J} \sum_{j=1}^J (\nabla_{\theta} \ell_{k,h})(\Theta_{n-1}, \mathfrak{T}_{n,j}, \mathfrak{X}_{n,j}) \right]. \quad (15)$$

Ultimately, for sufficiently large $k, h, n \in \mathbb{N}$, the realization $\mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\Theta_n, 3}$ is chosen as an approximation:

$$\mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\Theta_n, 3} \approx u$$

of the unknown solution u of (1)–(2) in the space W defined in (8).

So, the following theorem is justified.

Theorem 1. Let $T > 0$, and $u_0 \in C_0^\infty(\mathbb{R}^2)$. Then the sequence of $\{\mathcal{N}_{\mathfrak{M}_{a,l_1}, \mathfrak{M}_{a,l_2}, \dots, \mathfrak{M}_{a,l_h}, \text{id}_{\mathbb{R}}}^{\Theta_n, 3}\}_{k,h,n}$ defined in (14)–(15) has an accumulation point in the weak topology of W defined in (8). Moreover, each partial limit of the sequence in hands is weakly converges in W to the solution of Problem (1)–(2) in the sense of (3)–(4).

Proof. According to Steps 1–4 above, to derive PINNs, we approximate u in the space W defined in (8) by a deep ANN $\mathcal{N}_\theta : \mathbb{R}^2 \times [0, T] \rightarrow \mathbb{R}$ with parameters $\theta \in \mathbb{R}^d$ and minimize the empirical risk associated to $\mathcal{L}_k(v)$ over the parameter space \mathbb{R}^d . More precisely, we approximate the solution u of (1)–(2) by \mathcal{N}_{θ^*} where

$$\theta^* \in \arg \min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \left[|\mathcal{N}_\theta(X_i, 0) - u_0(X_i)|^2 + \left| \frac{\partial \mathcal{N}_\theta}{\partial t}(X_i, T_i) - \nu \nabla^2 \mathcal{N}_\theta(X_i, T_i) + f_k(\mathcal{N}_\theta(X_i, T_i)) \right|^2 \right]$$

for a suitable choice of training data $\{(X_i, T_i)\}_{i=1}^n$. Here $n \in \mathbb{N}$ denotes the number of training samples and the pairs (X_i, T_i) , $i \in \{1, 2, \dots, n\}$, denote the realizations of the random variables X and T .

Analogously, to derive DGMs, we approximate u by a deep Galerkin method (DGM) $\mathcal{G}_\theta : \mathbb{R}^2 \times [0, T] \rightarrow \mathbb{R}$ with parameters $\theta \in \mathbb{R}^d$ and minimize the empirical risk associated to $\mathcal{L}_{k,h}(v)$ over the parameter space \mathbb{R}^d . More precisely, we approximate the solution u of (1)–(2) by \mathcal{G}_{θ^*} , where

$$\theta^* \in \arg \min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \left[|\mathcal{G}_\theta(X_i, 0) - u_0(X_i)|^2 + \left| \frac{\partial \mathcal{G}_\theta}{\partial t}(X_i, T_i) - \nu \nabla^2 \mathcal{G}_\theta(X_i, T_i) + f_k(\mathcal{G}_\theta(X_i, T_i)) \right|^2 \right]$$

for a suitable choice (please, see the third (final) step above for details) of training data $\{(X_i, T_i)\}_{i=1}^n$. Here $n \in \mathbb{N}$ denotes the number of training samples and the pairs (X_i, T_i) , $i \in \{1, 2, \dots, n\}$, denote the realizations of the random variables X and T . \square

The empirical risk minimization problems for PINNs and DGMs are typically solved using SGD or variants thereof, such as Adam [15]. The gradients of the empirical risk with respect to the parameters θ can be computed efficiently using automatic differentiation, which is commonly available

in deep learning frameworks such as TensorFlow and PyTorch. We provide implementation details and numerical simulations for PINNs and DGMs in the next section.

3. Numerical Implementation

Let us present a straightforward implementation of the method as detailed in the previous Section for approximating a solution $u \in W$ of Problem (1)–(2) with $\nu = 0.005$, and the initial condition $u_0(x) := \psi(x_1^2 + x_2^2)$, where

$$\psi(s) := \begin{cases} \sin(8\pi \exp(1 - \frac{3}{3-s})), & s \in [0, 3); \\ 0, & \text{otherwise,} \end{cases} \quad (16)$$

$(x_1, x_2) \in \mathbb{R}^2$. Let $k = 0.01$. This implementation follows the original proposal by [16], where 20.000 realizations of the random variable $(\mathcal{X}, \mathcal{T})$ are first chosen. Here, \mathcal{T} is uniformly distributed over $[0, 3]$, and \mathcal{X} follows a normal distribution in \mathbb{R}^2 with mean $0 \in \mathbb{R}^2$ and covariance $4I_2 \in \mathbb{R}^{2 \times 2}$. A fully connected feed-forward ANN with 4 hidden layers, each containing 50 neurons, and employing the Swish activation function is then trained. The training process uses batches of size 256, sampled from the 20.000 preselected realizations of $(\mathcal{X}, \mathcal{T})$. Optimization is carried out using the Adam SGD method. A plot of the resulting approximation of the solution u after 20.000 training steps is shown in Figure 1.

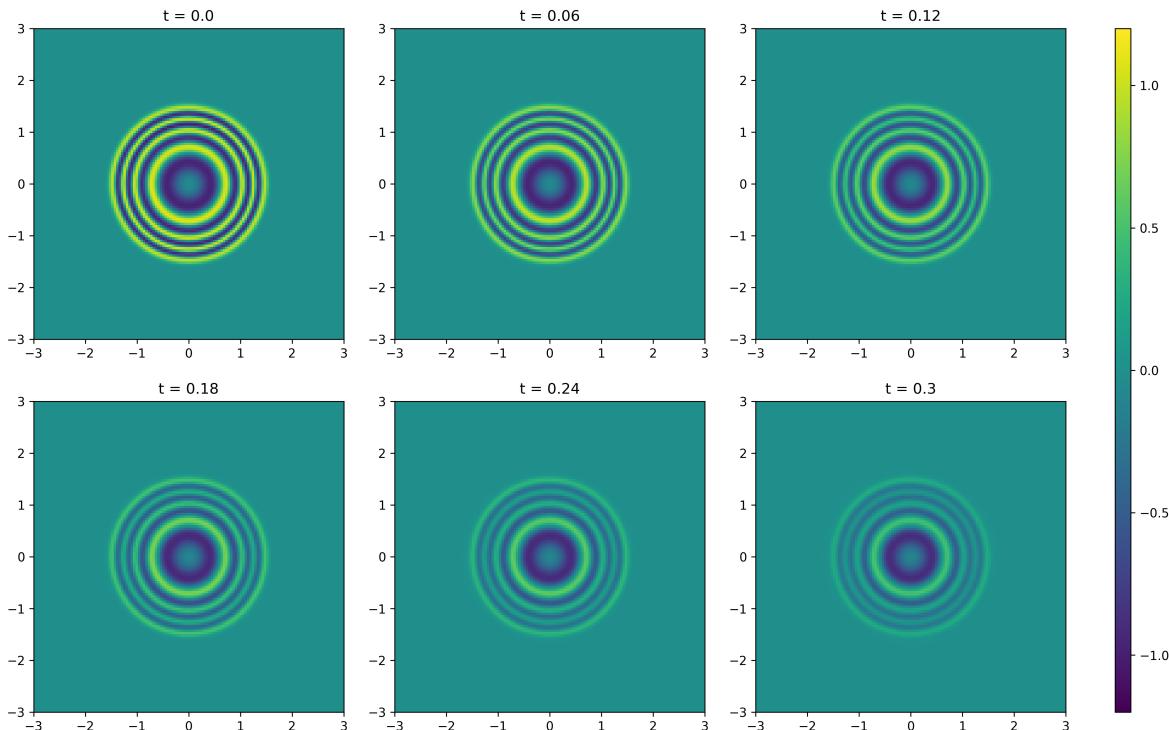


Figure 1. Plots for the functions $[-3, 3]^2 \ni x \mapsto U(x, t) \in \mathbb{R}$, where $t \in \{0, 0.06, 0.12, 0.18, 0.24, 0.3\}$ and $U \in C^{2,1}(\mathbb{R}^2 \times [0, 3])$ is an approximation of the solution u of Problem (1)–(2) with $\nu = 0.005$, $u_0(x) := \psi(x_1^2 + x_2^2)$, where ψ is defined in (16), computed by means of the PINN method as implemented in Source code Listing 1.

```
1 import os
2 import torch
3 import matplotlib.pyplot as plt
4 from torch.autograd import grad
5 from matplotlib.gridspec import GridSpec
6 from matplotlib.cm import ScalarMappable
7
8 dev = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
9
10 T = {\color{black}0.3} # the time horizon
11 M = 20000 # the number of training samples
12 k = {\color{black}10} # the parameter
13
14 torch.manual_seed(0)
15
16 x_data = torch.randn(M, 2).to(dev) * 2
17 t_data = torch.rand(M, 1).to(dev) * T
18
19 # The initial value
20 def phi(x):
21     x1, x2 = x[:, 0], x[:, 1]
22     r_squared = x1.square() + x2.square()
23     mask = r_squared < 3
24     result = torch.zeros_like(r_squared)
25     exponent = 1 - 3 / (3 - r_squared)
26     result[mask] = (8 * torch.pi * torch.exp(exponent[mask])).sin()
27     return result.unsqueeze(1)
28
29 # The interaction function
30
31 def fk(u, k):
32     return torch.where(u > 1/k, torch.tensor(1.0, device=u.device),
33                         torch.where(u < 0, torch.tensor(0.0, device=u
34 .device), u * k))
35
36 N = torch.nn.Sequential(
37     torch.nn.Linear(3, 50), torch.nn.SiLU(),
38     torch.nn.Linear(50, 50), torch.nn.SiLU(),
39     torch.nn.Linear(50, 50), torch.nn.SiLU(),
40     torch.nn.Linear(50, 50), torch.nn.SiLU(),
41     torch.nn.Linear(50, 1),
42 ).to(dev)
43
44 optimizer = torch.optim.Adam(N.parameters(), lr=1e-3)
45
46 J = 256 # the batch size
47
48 for i in range(20000):
49     if i % 100 == 0:
50         print(f"Iteration {i}")
```

```
50     # Choose a random batch of training samples
51     indices = torch.randint(0, M, (J,))
52     x = x_data[indices, :]
53     t = t_data[indices, :]
54
55     x1, x2 = x[:, 0:1], x[:, 1:2]
56
57     x1.requires_grad_()
58     x2.requires_grad_()
59     t.requires_grad_()
60
61     optimizer.zero_grad()
62
63     # Denoting by u the realization function of the ANN, compute
64     # u(0, x) for each x in the batch
65     u0 = N(torch.hstack((torch.zeros_like(t), x)))
66     # Compute the loss for the initial condition
67     initial_loss = (u0 - phi(x)).square().mean()
68
69     # Compute the partial derivatives using automatic
70     # differentiation
71     u = N(torch.hstack((t, x1, x2)))
72     ones = torch.ones_like(u)
73     u_t = grad(u, t, ones, create_graph=True)[0]
74     u_x1 = grad(u, x1, ones, create_graph=True)[0]
75     u_x2 = grad(u, x2, ones, create_graph=True)[0]
76     ones = torch.ones_like(u_x1)
77     u_x1x1 = grad(u_x1, x1, ones, create_graph=True)[0]
78     u_x2x2 = grad(u_x2, x2, ones, create_graph=True)[0]
79
80     # Compute the loss for the PDE
81     Laplace = u_x1x1 + u_x2x2
82     pde_loss = (u_t - (0.005 * Laplace - fk(u, k))).square().mean()
83
84     # Compute the total loss and perform a gradient step
85     loss = initial_loss + pde_loss
86     loss.backward()
87     optimizer.step()
88
89     if i % 100 == 0:
90         print(f"Loss at iteration {i}: {loss.item()}")
91
92     ### Function to plot the solution at different times
93     def plot_solution(i):
94         mesh = 128
95         a, b = -3, 3
96         x, y = torch.meshgrid(
97             torch.linspace(a, b, mesh),
98             torch.linspace(a, b, mesh),
99             indexing="xy")
```

```

100
101     )
102     x = x.reshape((mesh * mesh, 1)).to(dev)
103     y = y.reshape((mesh * mesh, 1)).to(dev)
104     t = torch.full((mesh * mesh, 1), i * T / 5).to(dev)
105     z = N(torch.cat((t, x, y), 1))
106     z = z.detach().cpu().numpy().reshape((mesh, mesh))
107     return i, z
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129

```

Listing 1. Modified version of source code from Section 16.3 of [15].

4. Conclusions

In this paper, we presented a novel machine learning methodology for approximating solutions to reaction-diffusion equations with multivalued interaction functions, a class of equations characterized by non-unique solutions. The proposed approach leverages the power of physics-informed neural networks (PINNs) to provide approximate solutions, addressing the need for new methods in this domain.

Our methodology consists of four key steps:

1. **Approximation of the Interaction Function:** We replaced the multivalued interaction function with a sequence of Lipschitz continuous functions, ensuring the problem becomes well-posed.
2. **Formulation of the Optimization Problem:** The regularized problem was reformulated as an infinite-dimensional stochastic optimization problem.
3. **Application of Deep Galerkin Method (DGM):** We transformed the infinite-dimensional problem into a finite-dimensional one by incorporating artificial neural networks (ANNs).
4. **Optimization and Approximation:** Using stochastic gradient descent (SGD) optimization methods, we approximated the minimizer of the empirical risk, yielding an approximation of the unknown solution.

The numerical implementation demonstrated the effectiveness of the proposed method. We used a fully connected feed-forward ANN to approximate the solution of a reaction-diffusion equation with specific initial conditions. The results showed that the PINN method could approximate solutions accurately, as evidenced by the visual plots.

The key contributions of this paper are as follows:

- **Development of a Machine Learning Framework:** We established a robust framework using PINNs to tackle reaction-diffusion equations with multivalued interaction functions.
- **Handling Non-Uniqueness:** Our method addresses the challenge of non-unique solutions, providing a practical tool for approximating generalized solutions.
- **Numerical Validation:** We provided a detailed implementation and numerical validation, demonstrating the practical applicability of the proposed approach.

Future work could explore the extension of this methodology to other classes of partial differential equations with multivalued interaction functions, as well as further optimization and refinement of the neural network architectures used in the approximation process. The integration of more advanced machine learning techniques and the exploration of their impact on the accuracy and efficiency of the solutions also present promising avenues for research.

Author Contributions: All the authors contributed equally to this work.

Funding: "This research was funded by EIT Manufacturing asbl, 0123U103025, grant: "EuroSpaceHub - increasing the transfer of space innovations and technologies by bringing together the scientific community, industry and startups in the space industry". The second and the third authors were partially supported by NRU project No. 2023.03/0074 "Infinite-dimensional evolutionary equations with multivalued and stochastic dynamics". The authors thank the anonymous reviewers for their suggestions, which have improved the manuscript..

Institutional Review Board Statement: The authors have nothing to declare.

Informed Consent Statement: The authors have nothing to declare.

Data Availability Statement: Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Conflicts of Interest: The author have no relevant financial or non-financial interests to disclose.

References

1. Zgurovsky, M.Z.; Mel'nik, V.S.; Kasyanov, P.O. *Evolution Inclusions and Variation Inequalities for Earth Data Processing I: Operator Inclusions and Variation Inequalities for Earth Data Processing*; Vol. 24, Springer Science & Business Media, 2010.
2. Zgurovsky, M.Z.; Kasyanov, P.O. *Qualitative and quantitative analysis of nonlinear systems*; Springer, 2018.
3. Paszke, A.; Sam, G.; Chintala.; Soumith.; Chanan, G. PyTorch, 2016. Accessed on June 5, 2024.
4. Rust, J. Using randomization to break the curse of dimensionality. *Econometrica: Journal of the Econometric Society* **1997**, pp. 487–516.
5. Denkowski, Z.; Migórski, S.; Schaefer, R.; Telega, H. Inverse problem for the prelinear filtration of ground water. *Computer Assisted Methods in Engineering and Science* **2023**, 3, 97–107.
6. Eikmeier, A. On the existence of solutions to multivalued differential equations. PhD thesis, Dissertation, Berlin, Technische Universität Berlin, 2022, 2023.
7. Papageorgiou, N.S.; Zhang, J.; Zhang, W. Solutions with sign information for noncoercive double phase equations. *The Journal of Geometric Analysis* **2024**, 34, 14.
8. Liu, Y.; Liu, Z.; Papageorgiou, N.S. Sensitivity analysis of optimal control problems driven by dynamic history-dependent variational-hemivariational inequalities. *Journal of Differential Equations* **2023**, 342, 559–595.
9. Peng, Z.; Gamorski, P.; Migórski, S. Boundary optimal control of a dynamic frictional contact problem. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* **2020**, 100, e201900144.
10. Cintra, W.; Freitas, M.M.; Ma, T.F.; Marín-Rubio, P. Multivalued dynamics of non-autonomous reaction-diffusion equation with nonlinear advection term. *Chaos, Solitons & Fractals* **2024**, 180, 114499.

11. Freitas, M.M.; Cintra, W. Multivalued random dynamics of reaction-diffusion-advection equations driven by nonlinear colored noise. *Communications on Pure and Applied Analysis* **2024**, pp. 0–0.
12. Zhao, J.C.; Ma, Z.X. GLOBAL ATTRACTOR FOR A PARTLY DISSIPATIVE REACTION-DIFFUSION SYSTEM WITH DISCONTINUOUS NONLINEARITY. *Discrete & Continuous Dynamical Systems-Series B* **2023**, 28.
13. Gu, A.; Wang, B. Random attractors of reaction-diffusion equations without uniqueness driven by nonlinear colored noise. *Journal of Mathematical Analysis and Applications* **2020**, 486, 123880.
14. Zhang, P.; Gu, A. Attractors for multi-valued lattice dynamical systems with nonlinear diffusion terms. *Stochastics and Dynamics* **2022**, 22, 2140013.
15. Jentzen, A.; Kuckuck, B.; von Wurstemberger, P. Mathematical introduction to deep learning: methods, implementations, and theory. *arXiv preprint arXiv:2310.20360* **2023**.
16. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics* **2019**, 378, 686–707.
17. Beck, C.; Hutzenthaler, M.; Jentzen, A.; Kuckuck, B. An overview on deep learning-based approximation methods for partial differential equations. *Discrete and Continuous Dynamical Systems-B* **2023**, 28, 3697–3746.
18. Feinberg, E.A.; Kasyanov, P.O.; Royset, J.O. Epi-Convergence of Expectation Functions under Varying Measures and Integrands. *Journal of Convex Analysis* **2023**, 30, 917–936.
19. Zgurovsky, M.Z.; Kasyanov, P.O.; Kapustyan, O.V.; Valero, J.; Zadoianchuk, N.V. *Evolution Inclusions and Variation Inequalities for Earth Data Processing III: Long-Time Behavior of Evolution Inclusions Solutions in Earth Data Analysis*; Vol. 27, Springer Science & Business Media, 2012.
20. Sirignano, J.; Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics* **2018**, 375, 1339–1364.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.