Partial Differential Equations and Applications (2023) 4:51
https://doi.org/10.1007/s42985-023-00244-0

ORIGINAL PAPER

Deep learning approximations for non-local nonlinear PDEs with Neumann boundary conditions

Victor Boussange[1,2] · **Sebastian Becker**[3] · **Arnulf Jentzen**[4,5] · **Benno Kuckuck**[5] · **Loïc Pellissier**[1,2]

Received: 6 June 2022 / Accepted: 23 May 2023 / Published online: 1 December 2023

Abstract

Nonlinear partial differential equations (PDEs) are used to model dynamical processes in a large number of scientific fields, ranging from finance to biology. In many applications standard local models are not sufficient to accurately account for certain non-local phenomena such as, e.g., interactions at a distance. Non-local nonlinear PDE models can accurately capture these phenomena, but traditional numerical approximation methods are infeasible when the considered non-local PDE is high-dimensional. In this article we propose two numerical methods based on machine learning and on Picard iterations, respectively, to approximately solve non-local nonlinear PDEs. The proposed machine learning-based method is an extended variant of a deep learning-based splitting-up type approximation method previously introduced in the literature and utilizes neural networks to provide approximate solutions on a subset of the spatial domain of the solution. The Picard iterations-based method is an extended variant of the so-called full history recursive multilevel Picard approximation scheme previ-

---

✉ Arnulf Jentzen
ajentzen@cuhk.edu.cn

Victor Boussange
bvictor@ethz.ch

Sebastian Becker
sebastian.becker@math.ethz.ch

Benno Kuckuck
bkuckuck@uni-muenster.de

Loïc Pellissier
loic.pellissier@usys.ethz.ch

[1] Unit of Land Change Science, Swiss Federal Research Institute for Forest, Snow and Landscape (WSL), Zürcherstrasse 111, Birmensdorf 8903, Switzerland

[2] Landscape Ecology, Institute of Terrestrial Ecosystems, Department of Environmental Systems Science, ETH Zürich, Universitätstrasse 16, Zurich 8092, Switzerland

[3] Risklab, Department of Mathematics, ETH Zürich, Rämistrasse 101, Zurich 8092, Switzerland

[4] School of Data Science and Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, 2001 Longxiang Road, Longgang District, Shenzhen 518172, Guangdong Province, China

[5] Applied Mathematics: Institute for Analysis and Numerics, University of Münster, Einsteinstraße 62, 48149 Münster, Germany

ously introduced in the literature and provides an approximate solution for a single point of the domain. Both methods are mesh-free and allow non-local nonlinear PDEs with Neumann boundary conditions to be solved in high dimensions. In the two methods, the numerical difficulties arising due to the dimensionality of the PDEs are avoided by (i) using the correspondence between the expected trajectory of reflected stochastic processes and the solution of PDEs (given by the Feynman–Kac formula) and by (ii) using a plain vanilla Monte Carlo integration to handle the non-local term. We evaluate the performance of the two methods on five different PDEs arising in physics and biology. In all cases, the methods yield good results in up to 10 dimensions with short run times. Our work extends recently developed methods to overcome the curse of dimensionality in solving PDEs.

Keywords Non-local · Partial differential equation · PDE · Deep learning · Neural networks · Neumann boundary condition · Reflected Brownian motion

Mathematics Subject Classification Primary 35R09; Secondary 65M75 · 45K05 · 35K20 · 65C05 · 65M22 · 68T07

1 Introduction

In this article, we derive numerical schemes to approximately solve high-dimensional non-local nonlinear partial differential equations (PDEs) with Neumann boundary conditions. Such PDEs have been used to describe a variety of processes in physics, engineering, finance, and biology, but can generally not be solved analytically, requiring numerical methods to provide approximate solutions. However, traditional numerical methods are for the most part computationally infeasible for high-dimensional problems, calling for the development of novel approximation methods.

The need for solving non-local nonlinear PDEs has been expressed in various fields as they provide a more general description of the dynamical systems than their local counterparts [1–3]. In physics and engineering, non-local nonlinear PDEs are found, e.g., in models of Ohmic heating production [4], in the investigation of the fully turbulent behavior of real flows [5], in phase field models allowing non-local interactions [6–9], or in phase transition models with conservation of mass [10, 11]; see [1] for further references. In finance, non-local PDEs are used, e.g., in jump-diffusion models for the pricing of derivatives where the dynamics of stock prices are described by stochastic processes experiencing large jumps [3, 12–18]. Penalty methods for pricing American put options such as in Kou's jump-diffusion model [19, 20], considering large investors where the agent policy affects the assets prices [15, 21], or considering default risks [22, 23] can further introduce nonlinear terms in non-local PDEs. In economics, non-local nonlinear PDEs appear, e.g., in evolutionary game theory with the so-called replicator-mutator equation capturing continuous strategy spaces [24–28] or in growth models where consumption is non-local [29]. In biology, non-local nonlinear PDEs are used, e.g., to model processes determining the interaction and evolution of organisms. Examples include models of morphogenesis and cancer evolution [30–32], models of gene regulatory networks [33], population genetics models with the non-local Fisher–Kolmogorov–Petrovsky–Piskunov (Fisher–KPP) equations [34–40], and quantitative genetics models where populations are structured on a phenotypic and/or a geographical space [41–48]. In such models, Neumann boundary conditions are used, e.g., to model the effect of the borders of the geographical domain on the movement of the organisms.

Real world systems such as those just mentioned may be of considerable complexity and accurately capturing the dynamics of these systems may require models of high dimensionality [47], leading to complications in obtaining numerical approximations. For example, the number of dimensions of the PDEs may correspond in finance to the number of financial assets (such as stocks, commodities, exchange rates, and interest rates) in the involved portfolio; in evolutionary dynamics, to the dimension of the strategy space; and in biology, to the number of genes modelled [33] or to the dimension of the geographical or the phenotypic space over which the organisms are structured. Standard approximation methods for PDEs such as finite difference approximation methods, finite element methods, spectral Galerkin approximation methods, and sparse grid approximation methods all suffer from the so called *curse of dimensionality* [49], meaning that their computational costs increase exponentially in the number of dimensions of the PDE under consideration.

Numerical methods exploiting stochastic representations of the solutions of PDEs can in some cases overcome the curse of dimensionality. Specifically, simple Monte Carlo averages of the associated stochastic processes have been proposed a long time ago to solve high-dimensional linear PDEs, such as, e.g., Black–Scholes and Kolmogorov PDEs [50, 51]. Recently, two novel classes of methods have proved successful in dealing with high-dimensional nonlinear PDEs, namely deep learning-based and full history recursive multilevel Picard approximation methods (in the following we will abbreviate *full history recursive multilevel Picard* by MLP). The explosive success of deep learning in recent years across a wide range of applications [52] has inspired a variety of neural network-based approximation methods for high-dimensional PDEs; see [53–59] for a survey of this field of research. One class of such methods is based on reformulating the PDE as a stochastic learning problem through suitable Feynman–Kac formulas, cf., e.g., [60–65]. In particular, the *deep splitting* scheme introduced in [65] relies on splitting the differential operator into a linear part (which is reformulated using a suitable Feynman–Kac formula) and a nonlinear part and in that sense belongs to the class of splitting-up methods [66–68]. The PDE approximation problem is then decomposed along the time axis into a sequence of separate learning problems. The deep splitting approximation scheme has proved capable of computing reasonable approximations to the solutions of nonlinear PDEs in up to 10000 dimensions. On the other hand, the MLP approximation method, introduced in [69–71], utilizes the Feynman–Kac formula to reformulate the PDE problem as a fixed point equation. It further reduces the complexity of the numerical approximation of the time integral through a multilevel Monte Carlo approach. However, neither the deep splitting nor the MLP method can, until now, account for non-localness and Neumann boundary conditions.

The goal of this article is to overcome these limitations and thus we generalize the deep splitting method and the MLP approximation method to approximately solve non-local nonlinear PDEs with Neumann boundary conditions. We handle the non-local term by a plain vanilla Monte Carlo integration and address Neumann boundary conditions by constructing reflected stochastic processes. While the MLP method can, in one run, only provide an approximate solution at a single point $x \in D$ of the spatial domain $D \subseteq \mathbb{R}^d$ where $d \in \mathbb{N} = \{1, 2, \dots\}$, the machine learning-based method can in principle provide an approximate solution on a full subset of the spatial domain D (however, cf., e.g., [72–74] for results on limitations on the performance of such approximation schemes). We use both methods to solve five non-local nonlinear PDEs arising in models from biology and physics and cross-validate the results of the simulations. We manage to solve the non-local nonlinear PDEs with reasonable accuracy in up to 10 dimensions.

For an account of classical numerical methods for solving non-local PDEs, such as finite differences, finite elements, and spectral methods, we refer the reader to the recent survey [2].

Several machine-learning based schemes for solving non-local PDEs can also be found in the literature. In particular, the *physics-informed neural network* and *deep Galerkin* approaches [75, 76], based on representing an approximation of the whole solution of the PDE as a neural network and using automatic differentiation to do a least-squares minimization of the residual of the PDE, have been extended to fractional PDEs and other non-local PDEs [77–81]. While some of these approaches use classical methods susceptible to the curse of dimensionality for the non-local part [77, 78], mesh-free methods suitable for high-dimensional problems have also been investigated [79–81].

The literature also contains approaches that are more closely related to the machine learning-based algorithm presented here. Frey and Köck [82, 83] propose an approximation method for non-local semilinear parabolic PDEs with Dirichlet boundary conditions based on and extending the deep splitting method in [65] and carry out numerical simulations for example PDEs in up to 4 dimensions. Castro [84] proposes a numerical scheme for approximately solving non-local nonlinear PDEs based on [64] and proves convergence results for this scheme. Finally, Gonon and Schwab [85] provide theoretical results showing that neural networks with ReLU activation functions have sufficient expressive power to approximate solutions of certain high-dimensional non-local linear PDEs without the curse of dimensionality.

There is a more extensive literature on machine learning-based methods for approximately solving standard PDEs without non-local terms but with various boundary conditions, going back to early works by Lagaris et al. [86, 87] (see also [88]), which employed a grid-based method based on least-squares minimization of the residual and shallow neural networks to solve low-dimensional ODEs and PDEs with Dirichlet, Neumann, and mixed boundary conditions. More recently, approximation methods for PDEs with Neumann (and other) boundary conditions have been proposed using, e.g., physics-informed neural networks [78, 89, 90], the *deep Ritz* method (based on a variational formulation of certain elliptic PDEs) [91–93], or adversarial networks [94].

The remainder of this article is organized as follows. Section 2 discusses a special case of the proposed machine learning-based method, in order to provide a readily comprehensible exposition of the key ideas of the method. Section 3 discusses the general case, which is flexible enough to cover a larger class of PDEs and to allow more sophisticated optimization methods. Section 4 presents our extension of the MLP approximation method to non-local nonlinear PDEs, which we use to obtain reference solutions in Sect. 5. Section 5 provides numerical simulations for five concrete examples of (non-local) nonlinear PDEs.

2 Machine learning-based approximation method in a special case

In this section, we present in Framework 2.11 in Sect. 2.3 below a simplified version of our general machine learning-based algorithm for approximating solutions of non-local nonlinear PDEs with Neumann boundary conditions proposed in Sect. 3 below. This simplified version applies to a smaller class of non-local heat PDEs, specified in Sect. 2.1 below. In Sect. 2.10 we present some elementary results related to the reflection of straight lines on the boundaries of a suitable subset $D \subseteq \mathbb{R}^d$ where $d \in \mathbb{N}$. These serve to elucidate and justify the notations introduced in Framework 2.10 which in turn will be used to describe time-discrete reflected stochastic processes that are employed in our approximations throughout the rest of the article. The simplified algorithm described in Sect. 2.3 below is limited to using neural networks of a particular architecture that are trained using plain vanilla stochastic gradient descent, whereas the full version proposed in Framework 3.1 in Sect. 3.2 below is formulated in such a way that it encompasses a wide array of neural network architectures and more sophisticated training methods, in particular Adam optimization, minibatches, and batch normalization. Stripping away some of these more intricate aspects of the full algorithm is intended to exhibit more acutely the central ideas in the proposed approximation method.

The simplified algorithm described in this section as well as the more general version proposed in Framework 3.1 in Sect. 3.2 below are based on the deep splitting method introduced in Beck et al. [65], which combines operator splitting with a previous deep learning-based approximation method for Kolmogorov PDEs [62]; see also Beck et al. [53, Sections 2 and 3] for an exposition of these methods.

2.1 Partial differential equations (PDEs) under consideration

Let $T \in (0, \infty)$, $d \in \mathbb{N}$, let $D \subseteq \mathbb{R}^d$ be a sufficiently regular closed set, let[1] $\mathbf{n}\colon \partial_D \to \mathbb{R}^d$ be a suitable outer unit normal vector field associated to D, let $g \in C(D, \mathbb{R})$, let $\nu_x\colon \mathcal{B}(D) \to [0,1]$, $x \in D$, be probability measures, let $f\colon \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ be measurable, let $u = (u(t,x))_{(t,x)\in[0,T]\times D} \in C^{1,2}([0,T]\times D, \mathbb{R})$ have at most polynomially growing partial derivatives, assume[2] for every $t \in (0,T]$, $x \in \partial_D$ that $\langle \mathbf{n}(x), (\nabla_x u)(t,x)\rangle = 0$, and assume for every $t \in [0,T]$, $x \in D$ that $u(0,x) = g(x)$, $\int_D |f(u(t,x), u(t,\mathbf{x}))|\,\nu_x(d\mathbf{x}) < \infty$, and

$$\big(\tfrac{\partial}{\partial t} u\big)(t,x) = (\Delta_x u)(t,x) + \int_D f(u(t,x), u(t,\mathbf{x}))\,\nu_x(d\mathbf{x}). \tag{1}$$

Our goal in this section is to approximately calculate under suitable hypotheses the solution $u\colon [0,T]\times D \to \mathbb{R}$ of the PDE in (1).

2.2 Reflection principle for the simulation of time discrete reflected processes

Lemma 2.1 *Let $d \in \mathbb{N}$, let $D \subseteq \mathbb{R}^d$, let $x, y, \mathfrak{c} \in \mathbb{R}^d$ satisfy*

$$\mathfrak{c} = x + \big[\inf(\{r \in [0,1]\colon x + r(y-x) \notin D\} \cup \{1\})\big](y-x), \tag{2}$$

and assume $\mathfrak{c} \notin \{x, y\}$. Then $\mathfrak{c} \in \partial_D$.

Proof of Lemma 2.1 Throughout this proof let $t \in [0,1]$ satisfy

$$t = \inf(\{r \in [0,1]\colon x + r(y-x) \notin D\} \cup \{1\}). \tag{3}$$

Observe that (2) and (3) imply that

$$\mathfrak{c} = x + t(y-x). \tag{4}$$

This and the assumption that $\mathfrak{c} \notin \{x, y\}$ show that

$$t \notin \{0, 1\}. \tag{5}$$

---

[1] Throughout this article we denote for every topological space $(X, \mathcal{X})$ and every set $D \subseteq X$ by ∂_D the set which satisfies $\partial_D = \{x \in X\colon (\forall\, U \in \mathcal{X}\colon (x \in U \to (U \nsubseteq D \wedge U \nsubseteq X\backslash D)))\}$.

[2] Throughout this article we denote by $\langle\cdot,\cdot\rangle\colon \big(\bigcup_{n\in\mathbb{N}}(\mathbb{R}^n\times\mathbb{R}^n)\big) \to \mathbb{R}$ and $\|\cdot\|\colon \big(\bigcup_{n\in\mathbb{N}}\mathbb{R}^n\big) \to \mathbb{R}$ the functions which satisfy for every $n \in \mathbb{N}$, $v = (v_1, \ldots, v_n)$, $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$ that $\langle v, w\rangle = \sum_{i=1}^n v_i w_i$ and $\|v\| = \sqrt{\langle v, v\rangle} = \big[\sum_{i=1}^n |v_i|^2\big]^{1/2}$.

Combining this with (3) ensures that for every $s \in [0, t)$ it holds that

$$x + s(y - x) \in D. \tag{6}$$

Next observe that (3) and (5) imply that for every $\varepsilon \in (0, \infty)$ there exists $s \in [t, t + \varepsilon)$ such that

$$x + s(y - x) \notin D. \tag{7}$$

Combining this with (4), (5) and (6) shows that $\mathfrak{c} \in \partial_D$. The proof of Lemma 2.1 is thus complete. □

Lemma 2.2 *Let $d \in \mathbb{N}$, $x, n \in \mathbb{R}^d$ and assume $\|n\| = 1$. Then*

(i) *it holds that $\|n - x\| = \|n + x - 2\langle x, n\rangle n\|$ and*
(ii) *it holds that $\|x - 2\langle x, n\rangle n\| = \|x\|$.*

Proof of Lemma 2.2 Throughout this proof let $y \in \mathbb{R}^d$ satisfy

$$y = x - \langle x, n\rangle n. \tag{8}$$

Observe that the assumption that $\|n\| = 1$ implies that

$$\langle y, n\rangle = \langle x, n\rangle - \langle x, n\rangle\langle n, n\rangle = \langle x, n\rangle - \langle x, n\rangle\|n\|^2 = 0. \tag{9}$$

This ensures that for every $a, b \in \mathbb{R}$ it holds that

$$\begin{aligned} \|an + by\|^2 &= \langle an + by, an + by\rangle \\ &= \langle an, an\rangle + \langle an, by\rangle + \langle by, an\rangle + \langle by, by\rangle \\ &= a^2\langle n, n\rangle + 2ab\langle y, n\rangle + b^2\langle y, y\rangle \\ &= a^2\|n\|^2 + b^2\|y\|^2. \end{aligned} \tag{10}$$

Hence, we obtain that

$$\begin{aligned} \|n + x - 2\langle x, n\rangle n\|^2 &= \|n + y - \langle x, n\rangle n\|^2 \\ &= \|(1 - \langle x, n\rangle)n + y\|^2 \\ &= (1 - \langle x, n\rangle)^2\|n\|^2 + \|y\|^2 \\ &= \|(1 - \langle x, n\rangle)n - y\|^2 \\ &= \|n - y - \langle x, n\rangle n\|^2 \\ &= \|n - x\|^2. \end{aligned} \tag{11}$$

This proves item (i). Next note that (10) implies that

$$\|x - 2\langle x, n\rangle n\|^2 = \|y - \langle x, n\rangle n\|^2 = \langle x, n\rangle^2\|n\|^2 + \|y\|^2 = \|y + \langle x, n\rangle n\|^2 = \|x\|^2. \tag{12}$$

This demonstrates item (ii). The proof of Lemma 2.2 is thus complete. □

Lemma 2.3 *Let $d \in \mathbb{N}$, let $D \subseteq \mathbb{R}^d$ be a closed set, let $\mathfrak{c}\colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ satisfy for every $x, y \in \mathbb{R}^d$ that*

$$\mathfrak{c}(x, y) = x + \big[\inf(\{r \in [0, 1] \colon x + r(y - x) \notin D\} \cup \{1\})\big](y - x), \tag{13}$$

let $\mathbf{n}\colon \partial_D \to \{x \in \mathbb{R}^d \colon \|x\| = 1\}$ *and* $\mathscr{R}\colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$ *satisfy for every* $x, y \in \mathbb{R}^d$ *that*

$$\mathscr{R}(x, y) = \begin{cases} (x, y) & \colon \mathfrak{c}(x, y) = x \\ \big(\mathfrak{c}(x, y), y - 2\big\langle y - \mathfrak{c}(x, y), \mathbf{n}(\mathfrak{c}(x, y))\big\rangle \mathbf{n}(\mathfrak{c}(x, y))\big) & \colon \mathfrak{c}(x, y) \notin \{x, y\} \\ (y, y) & \colon \mathfrak{c}(x, y) = y, \end{cases} \tag{14}$$

and let $x_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$, *and* $y_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0$, *satisfy for every* $n \in \mathbb{N}$ *that*

$$(x_n, y_n) = \mathscr{R}(x_{n-1}, y_{n-1}) \tag{15}$$

(cf. Lemma 2.1). Then

(i) *it holds for every* $n \in \mathbb{N}$ *that* $x_n = \mathfrak{c}(x_{n-1}, y_{n-1})$,
(ii) *it holds for every* $n, k \in \mathbb{N}_0$ *with* $x_n \in D$ *that* $x_{n+k} \in D$,
(iii) *it holds for every* $n, k \in \mathbb{N}$ *with* $\mathfrak{c}(x_{n-1}, y_{n-1}) \in \{x_{n-1}, y_{n-1}\}$ *that* $(x_{n+k}, y_{n+k}) = (x_n, y_n)$,
(iv) *it holds for every* $n \in \mathbb{N}$ *that* $\|x_n - y_{n-1}\| = \|x_n - y_n\|$,
(v) *it holds that*

$$\sum_{i=0}^{\infty} \|x_i - x_{i+1}\| \leq \|x_0 - y_0\|. \tag{16}$$

Proof of Lemma 2.3 First, observe that (14) and (15) imply item (i). Moreover, note that (13) ensures that for every $v, w \in \mathbb{R}^d$ with $v \neq w$ and $\mathfrak{c}(v, w) = w$ it holds that

$$\inf(\{r \in [0, 1] \colon v + r(w - v) \notin D\} \cup \{1\}) = 1. \tag{17}$$

The assumption that D is a closed set hence shows that for every $v, w \in \mathbb{R}^d$ with $v \neq w$ and $\mathfrak{c}(v, w) = w$ it holds that $w \in D$. Combining this and the fact that for every $v \in \mathbb{R}^d$ it holds that $\mathfrak{c}(v, v) = v$ with Lemma 2.1 and the assumption that D is a closed set proves that for every $v \in D$, $w \in \mathbb{R}^d$ it holds that

$$\mathfrak{c}(v, w) \in D. \tag{18}$$

Item (i) and induction hence establish item (ii). Next observe that (14) and (15) imply that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) = x_{n-1}$ it holds that

$$(x_n, y_n) = \mathscr{R}(x_{n-1}, y_{n-1}) = (x_{n-1}, y_{n-1}). \tag{19}$$

Induction therefore ensures that for every $n \in \mathbb{N}$, $k \in \mathbb{N}_0$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) = x_{n-1}$ it holds that

$$(x_{n+k}, y_{n+k}) = (x_n, y_n) = (x_{n-1}, y_{n-1}). \tag{20}$$

In addition, note that (14) and (15) show that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) = y_{n-1}$ it holds that

$$(x_n, y_n) = \mathscr{R}(x_{n-1}, y_{n-1}) = (y_{n-1}, y_{n-1}). \tag{21}$$

Hence, we obtain that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) = y_{n-1}$ it holds that

$$\mathfrak{c}(x_n, y_n) = \mathfrak{c}(y_{n-1}, y_{n-1}) = y_{n-1} = x_n. \tag{22}$$

This and (20) demonstrate that for every $n, k \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) = y_{n-1}$ it holds that

$$(x_{n+k}, y_{n+k}) = (x_n, y_n). \tag{23}$$

Combining this with (20) establishes item (iii). In the next step we observe that (14) and (15) ensure that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \in \{x_{n-1}, y_{n-1}\}$ it holds that $y_n = y_{n-1}$. Therefore, we obtain that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \in \{x_{n-1}, y_{n-1}\}$ it holds that

$$\|x_n - y_n\| = \|x_n - y_{n-1}\|. \tag{24}$$

Next note that Lemma 2.1, item (ii) in Lemma 2.2 (applied for every $v \in \mathbb{R}^d$, $w \in \{u \in \mathbb{R}^d : \mathfrak{c}(v, u) \notin \{v, u\}\}$ with $d \curvearrowleft d$, $x \curvearrowleft w - \mathfrak{c}(v, w)$, $n \curvearrowleft \mathbf{n}(\mathfrak{c}(v, w))$ in the notation of Lemma 2.2), and (14) show that for every $v, w, u, z \in \mathbb{R}^d$ with $\mathfrak{c}(v, w) \notin \{v, w\}$ and $(u, z) = \mathscr{R}(v, w)$ it holds that

$$\|z - \mathfrak{c}(v, w)\| = \|w - \mathfrak{c}(v, w) - 2\langle w - \mathfrak{c}(v, w), \mathbf{n}(\mathfrak{c}(v, w))\rangle \mathbf{n}(\mathfrak{c}(v, w))\| = \|w - \mathfrak{c}(v, w)\|. \tag{25}$$

This, item (i), and (15) prove that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that

$$\|x_n - y_n\| = \|y_n - \mathfrak{c}(x_{n-1}, y_{n-1})\| = \|y_{n-1} - \mathfrak{c}(x_{n-1}, y_{n-1})\| = \|x_n - y_{n-1}\|. \tag{26}$$

Combining this with (24) establishes item (iv). Moreover, observe that item (i) and (13) ensure that for every $n \in \mathbb{N}$ there exists $t \in [0, 1]$ such that $x_n = x_{n-1} + t(y_{n-1} - x_{n-1})$. This and item (iv) show that for every $n \in \mathbb{N}$ there exists $t \in [0, 1]$ such that

$$\begin{aligned}
&\|x_{n-1} - x_n\| + \|x_n - y_n\| \\
&= \|x_{n-1} - x_n\| + \|x_n - y_{n-1}\| \\
&= \|x_{n-1} - (x_{n-1} + t(y_{n-1} - x_{n-1}))\| + \|(x_{n-1} + t(y_{n-1} - x_{n-1})) - y_{n-1}\| \\
&= \|-t(y_{n-1} - x_{n-1})\| + \|(t - 1)(y_{n-1} - x_{n-1})\| \\
&= t\|y_{n-1} - x_{n-1}\| + (1 - t)\|y_{n-1} - x_{n-1}\| \\
&= \|x_{n-1} - y_{n-1}\|.
\end{aligned} \tag{27}$$

Combining this with induction proves that for every $n \in \mathbb{N}$ it holds that

$$\|x_0 - y_0\| = \left(\sum_{i=1}^{n} \|x_{i-1} - x_i\|\right) + \|x_n - y_n\|. \tag{28}$$

This implies item (v). The proof of Lemma 2.3 is thus complete. □

Proposition 2.4 *Let $d \in \mathbb{N}$, let $\mathfrak{c} \colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ satisfy for every $x, y \in \mathbb{R}^d$ that*

$$\mathfrak{c}(x, y) = x + \big[\inf(\{r \in [0, 1] \colon \|x + r(y - x)\| > 1\} \cup \{1\})\big](y - x), \tag{29}$$

let $\mathscr{R} \colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathscr{R}(x, y) = \begin{cases} (x, y) & : \mathfrak{c}(x, y) = x \\ \big(\mathfrak{c}(x, y), y - 2\big\langle y - \mathfrak{c}(x, y), \mathfrak{c}(x, y)\big\rangle \mathfrak{c}(x, y)\big) & : \mathfrak{c}(x, y) \notin \{x, y\} \\ (y, y) & : \mathfrak{c}(x, y) = y, \end{cases} \tag{30}$$

and let $x_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0$, and $y_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0$, satisfy for every $n \in \mathbb{N}$ that

$$(x_n, y_n) = \mathscr{R}(x_{n-1}, y_{n-1}). \tag{31}$$

Then there exists $N \in \mathbb{N}_0$ such that for every $n \in \mathbb{N}_0 \cap [N, \infty)$ it holds that

$$(x_n, y_n) = (x_N, y_N). \tag{32}$$

Proof of Proposition 2.4 Throughout this proof let $t_n \in [0,1]$, $n \in \mathbb{N}_0$, satisfy for every $n \in \mathbb{N}_0$ that

$$t_n = \inf(\{r \in [0,1] \colon \|x_n + r(y_n - x_n)\| > 1\} \cup \{1\}). \tag{33}$$

Note that Lemma 2.1 (applied for every $n \in \{m \in \mathbb{N} \colon \mathfrak{c}(x_{m-1}, y_{m-1}) \notin \{x_{m-1}, y_{m-1}\}\}$ with $d \curvearrowleft d$, $D \curvearrowleft \{x \in \mathbb{R}^d \colon \|x\| \leq 1\}$, $x \curvearrowleft x_{n-1}$, $y \curvearrowleft y_{n-1}$, $\mathfrak{c} \curvearrowleft \mathfrak{c}(x_{n-1}, y_{n-1})$ in the notation of Lemma 2.1) implies that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that

$$\|\mathfrak{c}(x_{n-1}, y_{n-1})\| = 1. \tag{34}$$

Next observe that item (i) in Lemma 2.1, (29), and (33) show that for every $n \in \mathbb{N}$ it holds that

$$x_n = \mathfrak{c}(x_{n-1}, y_{n-1}) = x_{n-1} + t_{n-1}(y_{n-1} - x_{n-1}). \tag{35}$$

This and (30) ensure that for every $n \in \mathbb{N}$, $t \in \mathbb{R}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that

$$\begin{aligned}
&x_n + t(y_n - x_n)\\
&\quad = x_n + t\big(y_{n-1} - 2\big\langle y_{n-1} - \mathfrak{c}(x_{n-1}, y_{n-1}), \mathfrak{c}(x_{n-1}, y_{n-1})\big\rangle \mathfrak{c}(x_{n-1}, y_{n-1}) - x_n\big)\\
&\quad = x_n + t(y_{n-1} - x_n) - 2t\big\langle y_{n-1} - \mathfrak{c}(x_{n-1}, y_{n-1}), \mathfrak{c}(x_{n-1}, y_{n-1})\big\rangle \mathfrak{c}(x_{n-1}, y_{n-1})\\
&\quad = x_n + t(y_{n-1} - x_n) - 2\langle t(y_{n-1} - x_n), x_n\rangle x_n.
\end{aligned} \tag{36}$$

Combining this, (34), and (35) with item (i) in Lemma 2.2 (applied for every $t \in \mathbb{R}$, $n \in \{m \in \mathbb{N} \colon \mathfrak{c}(x_{m-1}, y_{m-1}) \notin \{x_{m-1}, y_{m-1}\}\}$ with $d \curvearrowleft d$, $x \curvearrowleft t(y_{n-1} - x_n)$, $n \curvearrowleft x_n$ in the notation of Lemma 2.2) establishes that for every $n \in \mathbb{N}$, $t \in \mathbb{R}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that

$$\|x_n + t(y_n - x_n)\| = \|x_n - t(y_{n-1} - x_n)\|. \tag{37}$$

In the next step we note that (33) implies that for every $n \in \mathbb{N}_0$, $t \in [0, t_n]$ with $t_n > 0$ it holds that

$$\|x_n + t(y_n - x_n)\| \leq 1. \tag{38}$$

The fact that for every $s \in [0,1)$, $t \in [0, \frac{s}{1-s}]$ it holds that $s - t(1-s) \in [0,s]$ and (35) hence ensure that for every $n \in \mathbb{N}$, $t \in [0,1]$ with $t_{n-1} \in (0,1)$ and $t \leq \frac{t_{n-1}}{1-t_{n-1}}$ it holds that

$$\begin{aligned}
&\|x_n - t(y_{n-1} - x_n)\|\\
&\quad = \|x_{n-1} + t_{n-1}(y_{n-1} - x_{n-1}) - t(y_{n-1} - x_{n-1} - t_{n-1}(y_{n-1} - x_{n-1}))\|\\
&\quad = \|x_{n-1} + (t_{n-1} - t(1 - t_{n-1}))(y_{n-1} - x_{n-1})\| \leq 1.
\end{aligned} \tag{39}$$

Combining this with (37) shows that for every $n \in \mathbb{N}$, $t \in [0,1]$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ and $t \leq \frac{t_{n-1}}{1-t_{n-1}}$ it holds that

$$\|x_n + t(y_n - x_n)\| \leq 1. \tag{40}$$

Hence, we obtain that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that

$$t_n \geq \min\big\{1, \tfrac{t_{n-1}}{1-t_{n-1}}\big\}. \tag{41}$$

Furthermore, note that (35) implies that for every $n \in \mathbb{N}_0$ with $\mathfrak{c}(x_n, y_n) \notin \{x_n, y_n\}$ it holds that

$$x_n \neq y_n \qquad \text{and} \qquad t_n \in (0,1). \tag{42}$$

Item (iv) in Lemma 2.3, (35), and (41) therefore establish that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ and $\mathfrak{c}(x_n, y_n) \notin \{x_n, y_n\}$ it holds that

$$\begin{aligned} \|x_{n+1} - x_n\| &= \|t_n(y_n - x_n)\| \\ &\geq \tfrac{t_{n-1}}{1-t_{n-1}} \|y_n - x_n\| \\ &= \tfrac{t_{n-1}}{1-t_{n-1}} \|y_{n-1} - x_n\| \\ &= \tfrac{t_{n-1}}{1-t_{n-1}} \|y_{n-1} - x_{n-1} - t_{n-1}(y_{n-1} - x_{n-1})\| \\ &= \|t_{n-1}(y_{n-1} - x_{n-1})\| \\ &= \|x_n - x_{n-1}\|. \end{aligned} \tag{43}$$

Moreover, observe that (42) ensures that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that

$$\|x_n - x_{n-1}\| = t_{n-1}\|y_{n-1} - x_{n-1}\| > 0. \tag{44}$$

Combining this and (43) with item (v) in Lemma 2.3 demonstrates that there exists $n \in \mathbb{N}_0$ such that $\mathfrak{c}(x_n, y_n) \in \{x_n, y_n\}$. Combining this with item (iii) in Lemma 2.3 establishes that there exists $N \in \mathbb{N}_0$ such that for every $n \in \mathbb{N}_0 \cap [N, \infty)$ it holds that

$$(x_n, y_n) = (x_N, y_N). \tag{45}$$

The proof of Proposition 2.4 is thus complete. □

Lemma 2.5 *Let $d \in \mathbb{N}$, for every $k \in \{1, 2, \dots, d\}$ let $\mathfrak{a}_k^0 \in \mathbb{R}$, $\mathfrak{a}_k^1 \in (\mathfrak{a}_k^0, \infty)$, let $D = [\mathfrak{a}_1^0, \mathfrak{a}_1^1] \times [\mathfrak{a}_2^0, \mathfrak{a}_2^1] \times \cdots \times [\mathfrak{a}_d^0, \mathfrak{a}_d^1] \subseteq \mathbb{R}^d$, let $e_1 = (1, 0, 0, \dots, 0)$, $e_2 = (0, 1, 0, \dots, 0)$, $\dots$, $e_d = (0, \dots, 0, 0, 1) \in \mathbb{R}^d$, let $\mathbf{n}\colon \partial_D \to \mathbb{R}^d$ satisfy for every $x = (x_1, \dots, x_d) \in \partial_D$, $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\}\colon x_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ that*

$$\mathbf{n}(x) = \begin{cases} -e_k & : x_k = \mathfrak{a}_k^0 \\ e_k & : x_k = \mathfrak{a}_k^1, \end{cases} \tag{46}$$

let $\mathfrak{c}\colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathfrak{c}(x, y) = x + \big[\inf(\{r \in [0, 1]\colon x + r(y - x) \notin D\} \cup \{1\})\big](y - x), \tag{47}$$

let $\mathscr{R}\colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathscr{R}(x, y) = \begin{cases} (x, y) & : \mathfrak{c}(x, y) = x \\ \big(\mathfrak{c}(x, y), y - 2\langle y - \mathfrak{c}(x, y), \mathbf{n}(\mathfrak{c}(x, y))\rangle \mathbf{n}(\mathfrak{c}(x, y))\big) & : \mathfrak{c}(x, y) \notin \{x, y\} \\ (y, y) & : \mathfrak{c}(x, y) = y, \end{cases} \tag{48}$$

for every $a \in \mathbb{R}$, $b \in (a, \infty)$ let $r_{a,b}\colon \mathbb{R} \to \mathbb{Z}$ satisfy for every $x \in (-\infty, a)$, $y \in (b, \infty)$ that

$$|r_{a,b}(2a - x)| < |r_{a,b}(x)| \quad \text{and} \quad |r_{a,b}(2b - x)| < |r_{a,b}(y)|, \tag{49}$$

let $\mathfrak{r}\colon \mathbb{R}^d \to \mathbb{N}_0$ satisfy for every $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ that

$$\mathfrak{r}(x) = \sum_{k=1}^{d} |r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(x_k)|, \tag{50}$$

and let $u \in \mathbb{R}^d$, $x = (x_1, \dots, x_d)$, $y = (y_1, \dots, y_d)$, $c = (c_1, \dots, c_d)$, $v = (v_1, \dots, v_d) \in \mathbb{R}^d$ satisfy $c = \mathfrak{c}(x, y) \notin \{x, y\}$ (cf. Lemma 2.1). Then

(i) *it holds that* $c \in \partial_D = \{z = (z_1, \dots, z_d) \in D \colon (\exists\, k \in \{1, 2, \dots, d\} \colon z_k \in \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\})\}$,

(ii) *it holds for every* $i, k \in \{1, 2, \dots, d\}$ *with* $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ *that*

$$v_i = \begin{cases} y_i & \colon i \neq k \\ 2c_k - y_k & \colon i = k, \end{cases} \tag{51}$$

(iii) *it holds that*

$$\mathfrak{c}(u, v) = u \quad \text{or} \quad \mathfrak{r}(v) < \mathfrak{r}(y), \tag{52}$$

and

(iv) *it holds for every* $k \in \{1, 2, \dots, d\}$ *with* $\{k\} = \{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ *that*

$$x_k \neq y_k, \qquad \mathfrak{c}(u, v) \neq u, \qquad \text{and} \qquad \mathfrak{r}(v) < \mathfrak{r}(y). \tag{53}$$

Proof of Lemma 2.5 Throughout this proof let $t \in [0, 1]$ satisfy

$$t = \inf(\{r \in [0, 1] \colon x + r(y - x) \notin D\} \cup \{1\}). \tag{54}$$

Note that Lemma 2.1 and the assumption that $c \notin \{x, y\}$ imply item (i). Next observe that (46) ensures that for every $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\}\}$ it holds that

$$y - \langle y - c, \mathbf{n}(c)\rangle \mathbf{n}(c) = y - 2\langle y - c, e_k\rangle e_k = y - 2(y_k - c_k)e_k. \tag{55}$$

This, (48), and the assumption that $c \notin \{x, y\}$ show that for every $i, k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ it holds that

$$v_i = \begin{cases} y_i & \colon i \neq k \\ y_k - 2(y_k - c_k) & \colon i = k \end{cases} = \begin{cases} y_i & \colon i \neq k \\ 2c_k - y_k & \colon i = k. \end{cases} \tag{56}$$

Hence, we obtain item (ii). Next note that (47) implies that for every $k \in \{1, 2, \dots, d\}$ with $x_k = y_k$ it holds that $x_k = c_k = y_k$. Item (ii) therefore demonstrates that for every $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ and $x_k = y_k$ it holds that

$$v = y. \tag{57}$$

Moreover, observe that the assumption that $c = \mathfrak{c}(x, y) \notin \{x, y\}$, (47), and (54) ensure that

$$t \in (0, 1). \tag{58}$$

The fact that $D \subseteq \mathbb{R}^d$ is a closed, convex set, (54), and the assumption that $c \neq y$ hence proves that for every $s \in (t, 1]$ it holds that

$$x + s(y - x) \notin D. \tag{59}$$

In addition, note that (48) implies that

$$u = c. \tag{60}$$

This, (57), (58), and (59) show that for every $k \in \{1, 2, \dots, d\}$, $s \in (0, 1]$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ and $x_k = y_k$ it holds that

$$\begin{aligned} u + s(v - u) &= c + s(y - c) \\ &= x + t(y - x) + s(y - x - t(y - x)) \\ &= x + (t + s(1 - t))(y - x) \notin D. \end{aligned} \tag{61}$$

Hence, we obtain that for every $k \in \{1, 2, \ldots, d\}$ with $k = \min\{l \in \{1, 2, \ldots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ and $x_k = y_k$ it holds that

$$\mathfrak{c}(u, v) = u. \tag{62}$$

Next observe that (47) and (58) ensure that for every $k \in \{1, 2, \ldots, d\}$ with $x_k \neq y_k$ it holds that

$$x_k < c_k < y_k \qquad \text{or} \qquad y_k < c_k < x_k. \tag{63}$$

Furthermore, note that (58) implies that

$$x \in D. \tag{64}$$

This and (63) show that for every $k \in \{1, 2, \ldots, d\}$ with $x_k \neq y_k$ and $c_k = \mathfrak{a}_k^0$ it holds that

$$y_k < c_k = \mathfrak{a}_k^0. \tag{65}$$

Combining this with item (ii), (49), and (50) proves that for every $k \in \{1, 2, \ldots, d\}$ with $k = \min\{l \in \{1, 2, \ldots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$, $x_k \neq y_k$, and $c_k = \mathfrak{a}_k^0$ it holds that

$$\begin{aligned} \mathfrak{r}(v) &= \left[\sum_{i=1}^{d} |r_{\mathfrak{a}_i^0, \mathfrak{a}_i^1}(y_i)|\right] - |r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(y_k)| + |r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(2\mathfrak{a}_k^0 - y_k)| \\ &< \sum_{i=1}^{d} |r_{\mathfrak{a}_i^0, \mathfrak{a}_i^1}(y_i)| = \mathfrak{r}(y). \end{aligned} \tag{66}$$

In addition, observe that (64) and (63) imply that for every $k \in \{1, 2, \ldots, d\}$ with $x_k \neq y_k$ and $c_k = \mathfrak{a}_k^1$ it holds that

$$\mathfrak{a}_k^1 = c_k < y_k. \tag{67}$$

Combining this with item (ii), (49), and (50) shows that for every $k \in \{1, 2, \ldots, d\}$ with $k = \min\{l \in \{1, 2, \ldots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$, $x_k \neq y_k$, and $c_k = \mathfrak{a}_k^1$ it holds that

$$\begin{aligned} \mathfrak{r}(v) &= \left[\sum_{i=1}^{d} |r_{\mathfrak{a}_i^0, \mathfrak{a}_i^1}(y_i)|\right] - |r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(y_k)| + |r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(2\mathfrak{a}_k^1 - y_k)| \\ &< \sum_{i=1}^{d} |r_{\mathfrak{a}_i^0, \mathfrak{a}_i^1}(y_i)| = \mathfrak{r}(y). \end{aligned} \tag{68}$$

This and (66) establish that for every $k \in \{1, 2, \ldots, d\}$ with $k = \min\{l \in \{1, 2, \ldots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ and $x_k \neq y_k$ it holds that

$$\mathfrak{r}(v) < \mathfrak{r}(y). \tag{69}$$

Combining this with (62) proves item (iii). Next note that the fact that for every $z = (z_1, \ldots, z_d) \in D$, $k \in \{1, 2, \ldots, d\}$ with $z_k \notin \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\}$ it holds that $z_k \in (\mathfrak{a}_k^0, \mathfrak{a}_k^1)$ implies that for every $k \in \{1, 2, \ldots, d\}$ with $c_k \notin \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\}$ there exists $T \in (0, \infty)$ such that for every $t \in [0, T]$ it holds that

$$c_k + t(v_k - c_k) \in [\mathfrak{a}_k^0, \mathfrak{a}_k^1]. \tag{70}$$

Moreover, observe that (47) demonstrates that for every $k \in \{1, 2, \ldots, d\}$ it holds that

$$x_k \leq c_k \leq y_k \qquad \text{or} \qquad y_k \leq c_k \leq x_k. \tag{71}$$

This and (64) show that for every $k \in \{1, 2, \dots, d\}$ with $c_k \in \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\}$ it holds that

$$y_k \le c_k = \mathfrak{a}_k^0 \qquad \text{or} \qquad \mathfrak{a}_k^1 = c_k \le y_k. \tag{72}$$

This and item (ii) ensure that for every $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ and $c_k = \mathfrak{a}_k^0$ it holds that

$$v_k = 2\mathfrak{a}_k^0 - y_k \ge \mathfrak{a}_k^0. \tag{73}$$

In addition, observe that item (ii) and (72) demonstrate that for every $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ and $c_k = \mathfrak{a}_k^1$ it holds that

$$v_k = 2\mathfrak{a}_k^1 - y_k \le \mathfrak{a}_k^1. \tag{74}$$

Combining this and (73) proves that for every $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ there exists $T \in (0, \infty)$ such that for every $t \in [0, T]$ it holds that

$$c_k + t(v_k - c_k) \in [\mathfrak{a}_k^0, \mathfrak{a}_k^1]. \tag{75}$$

This and (70) establish that for every $k \in \{1, 2, \dots, d\}$ with $\{k\} = \{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ there exists $T \in (0, \infty)$ such that for every $t \in [0, T]$ it holds that

$$c + t(v - c) \in D. \tag{76}$$

Moreover, note that item (ii) and the assumption that $c \neq y$ show that $v \neq \mathfrak{c}(x, y)$. Combining this and (76) with (47) and (60) shows that for every $k \in \{1, 2, \dots, d\}$ with $\{k\} = \{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ it holds that

$$u = c \neq \mathfrak{c}(c, v) = \mathfrak{c}(u, v). \tag{77}$$

This and (62) imply that for every $k \in \{1, 2, \dots, d\}$ with $\{k\} = \{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ it holds that

$$x_k \neq y_k. \tag{78}$$

Combining this with (69) proves that for every $k \in \{1, 2, \dots, d\}$ with $\{k\} = \{l \in \{1, 2, \dots, d\} \colon c_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ it holds that

$$\mathfrak{r}(v) < \mathfrak{r}(y). \tag{79}$$

This, (77), and (78) establish item (iv). The proof of Lemma 2.5 is thus complete. □

Lemma 2.6 *Let* $\lceil \cdot \rfloor \colon \mathbb{R} \to \mathbb{Z}$ *satisfy for every* $k \in \mathbb{N}$, $x \in [-k - 1/2, -k + 1/2)$, $y \in [-1/2, 1/2]$, $z \in (k - 1/2, k + 1/2]$ *that*

$$\lceil x \rfloor = -k, \qquad \lceil y \rfloor = 0, \qquad \textit{and} \qquad \lceil z \rfloor = k, \tag{80}$$

let $a \in \mathbb{R}$, $b \in (a, \infty)$, *and let* $r \colon \mathbb{R} \to \mathbb{Z}$ *satisfy for every* $x \in \mathbb{R}$ *that*

$$r(x) = \left\lceil \tfrac{x-a}{b-a} - \tfrac{1}{2} \right\rfloor. \tag{81}$$

Then it holds for every $x \in (-\infty, a)$, $y \in (b, \infty)$ *that*

$$|r(2a - x)| = |r(x)| - 1 \qquad \textit{and} \qquad |r(2b - y)| = |r(y)| - 1. \tag{82}$$

Proof of Lemma 2.6 Observe that (80) and the fact that for every $x \in (0, b - a]$ it holds that $k + \frac{x}{b-a} - \frac{1}{2} \in (k - \frac{1}{2}, -k + \frac{1}{2}]$ ensure that for every $k \in \mathbb{N}$, $x \in (0, b - a]$ it holds that

$$r(a + k(b - a) + x) = \left\lceil k + \tfrac{x}{b-a} - \tfrac{1}{2} \right\rfloor = k. \tag{83}$$

Moreover, note that (80) and the fact that for every $x \in (0, b-a]$ it holds that $-k+\frac{1}{2}-\frac{x}{b-a} \in [-k-\frac{1}{2}, -k+\frac{1}{2})$ imply that for every $k \in \mathbb{N}$, $x \in (0, b-a]$ it holds that

$$r(b-k(b-a)-x) = \left\lceil 1-k-\tfrac{x}{b-a}-\tfrac{1}{2}\right\rfloor = \left\lceil -k+\tfrac{1}{2}-\tfrac{x}{b-a}\right\rfloor = -k. \tag{84}$$

Next observe that (83) and (84) show that for every $k \in \mathbb{N}_0$, $x \in (0, b-a]$ it holds that

$$\begin{aligned} &|r(2a-(a-k(b-a)-x))| = |r(a+k(b-a)+x)| = |k| = k \\ &\quad = |-(k+1)|-1 = |r(b-(k+1)(b-a)-x)|-1 = |r(a-k(b-a)-x)|-1. \end{aligned} \tag{85}$$

This and the fact that for every $x \in (-\infty, a)$ there exist $k \in \mathbb{N}_0$, $y \in (0, b-a]$ such that $x = a-k(b-a)-y$ prove that for every $x \in (-\infty, a)$ it holds that

$$|r(2a-x)| = |r(x)|-1. \tag{86}$$

Furthermore, note that (83) and (84) ensure that for every $k \in \mathbb{N}_0$, $x \in (0, b-a]$ it holds that

$$\begin{aligned} &|r(2b-(b+k(b-a)+x))| = |r(b-k(b-a)-x)| = |-k| = k \\ &\quad = |r(a+(k+1)(b-a)+x)|-1 = |r(b+k(b-a)+x)|-1. \end{aligned} \tag{87}$$

This and the fact that for every $x \in (b, \infty)$ there exist $k \in \mathbb{N}_0$, $y \in (0, b-a]$ such that $x = b+k(b-a)+y$ demonstrate that for every $x \in (b, \infty)$ it holds that

$$|r(2b-x)| = |r(x)|-1. \tag{88}$$

This and (86) establish (82). The proof of Lemma 2.6 is thus complete. □

Corollary 2.7 *Let $d \in \mathbb{N}$, for every $k \in \{1, 2, \dots, d\}$ let $\mathfrak{a}_k^0 \in \mathbb{R}$, $\mathfrak{a}_k^1 \in (\mathfrak{a}_k^0, \infty)$, let $D = [\mathfrak{a}_1^0, \mathfrak{a}_1^1] \times [\mathfrak{a}_2^0, \mathfrak{a}_2^1] \times \cdots \times [\mathfrak{a}_d^0, \mathfrak{a}_d^1] \subseteq \mathbb{R}^d$, let $e_1 = (1, 0, 0, \dots, 0)$, $e_2 = (0, 1, 0, \dots, 0)$, $\dots$, $e_d = (0, \dots, 0, 0, 1) \in \mathbb{R}^d$, let $\mathbf{n}\colon \partial_D \to \mathbb{R}^d$ satisfy for every $x = (x_1, \dots, x_d) \in \partial_D$, $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\}\colon x_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ that*

$$\mathbf{n}(x) = \begin{cases} -e_k & : x_k = \mathfrak{a}_k^0 \\ e_k & : x_k = \mathfrak{a}_k^1, \end{cases} \tag{89}$$

let $\mathfrak{c}\colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathfrak{c}(x, y) = x + \left[\inf(\{r \in [0, 1]\colon x + r(y-x) \notin D\} \cup \{1\})\right](y-x), \tag{90}$$

let $\mathscr{R}\colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathscr{R}(x, y) = \begin{cases} (x, y) & : \mathfrak{c}(x, y) = x \\ \big(\mathfrak{c}(x, y), y - 2\langle y - \mathfrak{c}(x, y), \mathbf{n}(\mathfrak{c}(x, y))\rangle \mathbf{n}(\mathfrak{c}(x, y))\big) & : \mathfrak{c}(x, y) \notin \{x, y\} \\ (y, y) & : \mathfrak{c}(x, y) = y, \end{cases} \tag{91}$$

let $x_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0$, and $y_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0$, satisfy for every $n \in \mathbb{N}$ that

$$x_0 \in D \quad \text{and} \quad (x_n, y_n) = \mathscr{R}(x_{n-1}, y_{n-1}), \tag{92}$$

and let $N \in \mathbb{N} \cup \{\infty\}$ satisfy

$$N = \min(\{n \in \mathbb{N}\colon \mathfrak{c}(x_{n-1}, y_{n-1}) \in \{x_{n-1}, y_{n-1}\}\} \cup \{\infty\}) \tag{93}$$

(cf. Lemma 2.1). Then

(i) *it holds that* $N < \infty$ *and*
(ii) *it holds for every* $n \in \mathbb{N} \cap [N, \infty)$ *that* $(x_n, y_n) = (x_N, y_N)$.

Proof of Corollary 2.7 Throughout this proof let $\lceil \cdot \rfloor \colon \mathbb{R} \to \mathbb{Z}$ satisfy for every $k \in \mathbb{N}$, $u \in [-k - 1/2, -k + 1/2)$, $v \in [-1/2, 1/2]$, $w \in (k - 1/2, k + 1/2]$ that

$$\lceil u \rfloor = -k, \qquad \lceil v \rfloor = 0, \qquad \text{and} \qquad \lceil w \rfloor = k, \tag{94}$$

for every $a \in \mathbb{R}$, $b \in (a, \infty)$ let $r_{a,b} \colon \mathbb{R} \to \mathbb{Z}$ satisfy for every $z \in \mathbb{R}$ that

$$r_{a,b}(z) = \left\lceil \tfrac{z-a}{b-a} - \tfrac{1}{2} \right\rfloor, \tag{95}$$

and let $\mathfrak{r} \colon \mathbb{R}^d \to \mathbb{N}_0$ satisfy for every $z = (z_1, \dots, z_d) \in \mathbb{R}^d$ that

$$\mathfrak{r}(z) = \sum_{k=1}^{d} |r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(z_k)|. \tag{96}$$

Observe that Lemma 2.6 shows that for every $a, b, w, z \in \mathbb{R}$ with $w < a < b < z$ it holds that

$$|r_{a,b}(2a - w)| < |r_{a,b}(w)| \qquad \text{and} \qquad |r_{a,b}(2b - z)| < |r_{a,b}(z)|. \tag{97}$$

Item (iii) in Lemma 2.5 and (92) therefore establish that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ and $\mathfrak{c}(x_n, y_n) \notin \{x_n, y_n\}$ it holds that

$$\mathfrak{r}(y_n) < \mathfrak{r}(y_{n-1}). \tag{98}$$

The fact that for every $z \in \mathbb{R}^d$ it holds that $\mathfrak{r}(z) \in \mathbb{N}_0$ and (93) hence imply that

$$N < \infty. \tag{99}$$

Moreover, note that item (iii) in Lemma 2.3 ensures that for every $n \in \mathbb{N} \cap [N, \infty)$ it holds that

$$(x_n, y_n) = (x_N, y_N). \tag{100}$$

The proof of Corollary 2.7 is thus complete.

Lemma 2.8 *Let* $\lfloor \cdot \rfloor \colon \mathbb{R} \to \mathbb{Z}$ *satisfy for every* $k \in \mathbb{Z}$, $x \in [k, k + 1)$ *that*

$$\lfloor x \rfloor = k, \tag{101}$$

let $a \in \mathbb{R}$, $b \in (a, \infty)$, *and let* $\mathscr{R} \colon \mathbb{R} \to [0, 1]$ *and* $\mathscr{S} \colon \mathbb{R} \to [a, b]$ *satisfy for every* $x \in \mathbb{R}$ *that*

$$\mathscr{R}(x) = \begin{cases} x - \lfloor x \rfloor & \colon \lfloor x \rfloor \in \{2n \colon n \in \mathbb{Z}\} \\ \lfloor x \rfloor - x + 1 & \colon \lfloor x \rfloor \in \{2n + 1 \colon n \in \mathbb{Z}\} \end{cases} \quad \text{and} \quad \mathscr{S}(x) = (b - a)\mathscr{R}\big(\tfrac{x-a}{b-a}\big) + a. \tag{102}$$

Then

(i) *it holds for every* $k \in \mathbb{Z}$, $x \in [0, b - a]$ *that*

$$\mathscr{S}(a + 2k(b - a) + x) = a + x = \mathscr{S}(b - (2k + 1)(b - a) - x) \tag{103}$$

and

(ii) *it holds for every* $x \in \mathbb{R}$ *that* $\mathscr{S}(2a - x) = \mathscr{S}(x) = \mathscr{S}(2b - x)$.

Proof of Lemma 2.8 First, observe that (101) and (102) imply that for every $k \in \mathbb{Z}$, $x \in [0, 1)$ it holds that

$$\mathscr{R}(2k + x) = (2k + x) - 2k = x. \tag{104}$$

Combining this with (102) shows that for every $k \in \mathbb{Z}$, $x \in [0, b - a)$ it holds that

$$\mathscr{S}(a + 2k(b - a) + x) = (b - a)\mathscr{R}\big(2k + \tfrac{x}{b-a}\big) + a = a + x. \tag{105}$$

Furthermore, note that (101) and (102) ensure that for every $k \in \mathbb{Z}$, $x \in [0, 1)$ it holds that

$$\mathscr{R}(2k + 1 + x) = 2k + 1 - (2k + 1 + x) + 1 = 1 - x. \tag{106}$$

This and (102) imply that for every $k \in \mathbb{Z}$ it holds that

$$\mathscr{S}(a + 2k(b - a) + (b - a)) = (b - a)\mathscr{R}(2k + 1) + a = a + (b - a). \tag{107}$$

Combining this with (105) shows that for every $k \in \mathbb{Z}$, $x \in [0, b - a]$ it holds that

$$\mathscr{S}(a + 2k(b - a) + x) = a + x. \tag{108}$$

Next observe that (106) and (102) prove that for every $k \in \mathbb{Z}$, $x \in (0, b - a]$ it holds that

$$\begin{aligned}\mathscr{S}(b - (2k + 1)(b - a) - x) &= (b - a)\mathscr{R}\big(1 - (2k + 1) - \tfrac{x}{b-a}\big) + a\\ &= (b - a)\mathscr{R}\big(2(-k - 1) + 1 + \big(1 - \tfrac{x}{b-a}\big)\big) + a\\ &= (b - a)\big(1 - \big(1 - \tfrac{x}{b-a}\big)\big) + a\\ &= a + x.\end{aligned} \tag{109}$$

Moreover, note that (104) and (102) imply that for every $k \in \mathbb{Z}$ it holds that

$$\mathscr{S}(b - (2k + 1)(b - a)) = (b - a)\mathscr{R}(1 - (2k + 1)) + a = (b - a)\mathscr{R}(-2k) + a = a. \tag{110}$$

Combining this with (109) demonstrates that for every $k \in \mathbb{Z}$, $x \in [0, b - a]$ it holds that

$$\mathscr{S}(b - (2k + 1)(b - a) - x) = a + x. \tag{111}$$

This and (108) establish item (i). In the next step we observe that item (i) shows that for every $k \in \mathbb{Z}$, $x \in [0, b - a]$ it holds that

$$\begin{aligned}\mathscr{S}(2a - (a + 2k(b - a) + x)) &= \mathscr{S}(b - (2k + 1)(b - a) - x)\\ &= a + x = \mathscr{S}(a + 2k(b - a) + x).\end{aligned} \tag{112}$$

Furthermore, note that item (i) ensures that for every $k \in \mathbb{Z}$, $x \in [0, b - a]$ it holds that

$$\begin{aligned}\mathscr{S}(2a - (b - (2k + 1)(b - a) - x)) &= \mathscr{S}(a + 2k(b - a) + x)\\ &= a + x = \mathscr{S}(b - (2k + 1)(b - a) - x).\end{aligned} \tag{113}$$

Moreover, observe that item (i) implies that for every $k \in \mathbb{Z}$, $x \in [0, b - a]$ it holds that

$$\begin{aligned}\mathscr{S}(2b - (a + 2k(b - a) + x)) &= \mathscr{S}(b - (2k - 1)(b - a) - x)\\ &= a + x = \mathscr{S}(a + 2k(b - a) + x).\end{aligned} \tag{114}$$

In addition, note that item (i) shows that for every $k \in \mathbb{Z}$, $x \in [0, b - a]$ it holds that

$$\begin{aligned}\mathscr{S}(2b - (b - (2k + 1)(b - a) - x)) &= \mathscr{S}(a + (2k + 2)(b - a) + x)\\ &= a + x = \mathscr{S}(b - (2k + 1)(b - a) - x).\end{aligned} \tag{115}$$

Combining this, (112), (113), and (114) with the fact that for every $y \in \mathbb{R}$ there exist $k \in \mathbb{Z}$, $x \in [0, b-a]$ such that $y \in \{a + 2k(b-a) + x, b - (2k+1)(b-a) - x\}$ establishes item (ii). The proof of Lemma 2.8 is thus complete. □

Corollary 2.9 *Let $d \in \mathbb{N}$, for every $k \in \{1, 2, \ldots, d\}$ let $\mathfrak{a}_k^0 \in \mathbb{R}$, $\mathfrak{a}_k^1 \in (\mathfrak{a}_k^0, \infty)$, let $D = [\mathfrak{a}_1^0, \mathfrak{a}_1^1] \times [\mathfrak{a}_2^0, \mathfrak{a}_2^1] \times \cdots \times [\mathfrak{a}_d^0, \mathfrak{a}_d^1] \subseteq \mathbb{R}^d$, let $e_1 = (1, 0, 0, \ldots, 0)$, $e_2 = (0, 1, 0, \ldots, 0)$, $\ldots$, $e_d = (0, \ldots, 0, 0, 1) \in \mathbb{R}^d$, let $\mathbf{n} \colon \partial_D \to \mathbb{R}^d$ satisfy for every $x = (x_1, \ldots, x_d) \in \partial_D$, $k \in \{1, 2, \ldots, d\}$ with $k = \min\{l \in \{1, 2, \ldots, d\} \colon x_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ that*

$$\mathbf{n}(x) = \begin{cases} -e_k & \colon x_k = \mathfrak{a}_k^0 \\ e_k & \colon x_k = \mathfrak{a}_k^1, \end{cases} \tag{116}$$

let $\mathfrak{c} \colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathfrak{c}(x, y) = x + \big[\inf(\{r \in [0, 1] \colon x + r(y - x) \notin D\} \cup \{1\})\big](y - x), \tag{117}$$

let $\mathscr{R} \colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathscr{R}(x, y) = \begin{cases} (x, y) & \colon \mathfrak{c}(x, y) = x \\ \big(\mathfrak{c}(x, y), y - 2\langle y - \mathfrak{c}(x, y), \mathbf{n}(\mathfrak{c}(x, y))\rangle \mathbf{n}(\mathfrak{c}(x, y))\big) & \colon \mathfrak{c}(x, y) \notin \{x, y\} \\ (y, y) & \colon \mathfrak{c}(x, y) = y, \end{cases} \tag{118}$$

let $\lfloor \cdot \rfloor \colon \mathbb{R} \to \mathbb{Z}$ satisfy for every $k \in \mathbb{Z}$, $z \in [k, k+1)$ that $\lfloor z \rfloor = k$, let $\mathfrak{r} \colon \mathbb{R} \to [0, 1]$ satisfy for every $z \in \mathbb{R}$ that

$$\mathfrak{r}(x) = \begin{cases} z - \lfloor z \rfloor & \colon \lfloor z \rfloor \in \{2n \colon n \in \mathbb{Z}\} \\ \lfloor z \rfloor - z + 1 & \colon \lfloor z \rfloor \in \{2n + 1 \colon n \in \mathbb{Z}\}, \end{cases} \tag{119}$$

for every $a, b \in \mathbb{R}$ let $r_{a,b} \colon \mathbb{R} \to [a, b]$ satisfy for every $z \in \mathbb{R}$ that $r_{a,b}(z) = (b - a)\mathfrak{r}\big(\frac{z-a}{b-a}\big) + a$, let $\mathscr{S} \colon \mathbb{R}^d \to D$ satisfy for every $z = (z_1, \ldots, z_d) \in \mathbb{R}^d$ that

$$\mathscr{S}(z) = \big(r_{\mathfrak{a}_1^0, \mathfrak{a}_1^1}(z_1), r_{\mathfrak{a}_2^0, \mathfrak{a}_2^1}(z_2), \ldots, r_{\mathfrak{a}_d^0, \mathfrak{a}_d^1}(z_d)\big), \tag{120}$$

let $x_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0$, and $y_n \in \mathbb{R}^d$, $n \in \mathbb{N}_0$, satisfy for every $n \in \mathbb{N}$ that

$$x_0 \in D \backslash \partial_D \quad \text{and} \quad (x_n, y_n) = \mathscr{R}(x_{n-1}, y_{n-1}), \tag{121}$$

and assume for every $n \in \mathbb{N}$ that

$$\mathfrak{c}(x_n, y_n) \notin \{z = (z_1, \ldots, z_d) \in \mathbb{R}^d \colon |\{k \in \{1, 2, \ldots, d\} \colon z_k \in \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\}\}| > 1\} \tag{122}$$

(cf. Lemma 2.1). Then there exists $N \in \mathbb{N}$ such that for every $n \in \mathbb{N} \cap [N, \infty)$ it holds that

$$y_n = \mathscr{S}(y_0). \tag{123}$$

Proof of Corollary 2.9 Throughout this proof let $N \in \mathbb{N} \cup \{\infty\}$ satisfy

$$N = \min(\{n \in \mathbb{N} \colon \mathfrak{c}(x_{n-1}, y_{n-1}) \in \{x_{n-1}, y_{n-1}\}\} \cup \{\infty\}). \tag{124}$$

Note that Corollary 2.7 proves that

(A) it holds that $N < \infty$ and
(B) it holds for every $n \in \mathbb{N} \cap [N, \infty)$ that $(x_n, y_n) = (x_N, y_N)$.

Observe that Lemma 2.1 and (122) ensure that for every $n \in \mathbb{N}_0$ with $\mathfrak{c}(x_n, y_n) \notin \{x_n, y_n\}$ it holds that

$$\mathfrak{c}(x_n, y_n) \in \{z = (z_1, \dots, z_d) \in \mathbb{R}^d : |\{k \in \{1, 2, \dots, d\} : z_k \in \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\}\}| = 1\}. \tag{125}$$

Item (iv) in Lemma 2.5 therefore establishes that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that

$$\mathfrak{c}(x_n, y_n) \neq x_n. \tag{126}$$

Furthermore, note that the assumption that $x \in D \backslash \partial_D$ and (117) imply that

$$\mathfrak{c}(x_0, y_0) \neq x_0. \tag{127}$$

Combining this and (126) with (124) and item (A) shows that

$$\mathfrak{c}(x_{N-1}, y_{N-1}) = y_{N-1}. \tag{128}$$

The fact that D is a closed set and (117) therefore demonstrate that $y_{N-1} \in D$. This, (118), and items (A) and (B) prove that for every $n \in \mathbb{N} \cap [N, \infty)$ it holds that

$$y_n = y_N = y_{N-1} \in D. \tag{129}$$

In the next step we observe that item (ii) in Lemma 2.8 establishes that for every $k \in \{1, 2, \dots, d\}$, $z \in \mathbb{R}$ it holds that

$$r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(2\mathfrak{a}_k^0 - z) = r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(z) \quad \text{and} \quad r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(2\mathfrak{a}_k^1 - z) = r_{\mathfrak{a}_k^0, \mathfrak{a}_k^1}(z). \tag{130}$$

Item (ii) in Lemma 2.5 and (121) therefore imply that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \notin \{x_{n-1}, y_{n-1}\}$ it holds that $\mathscr{S}(y_n) = \mathscr{S}(y_{n-1})$. Combining this and the fact that for every $n \in \mathbb{N}$ with $\mathfrak{c}(x_{n-1}, y_{n-1}) \in \{x_{n-1}, y_{n-1}\}$ it holds that $y_n = y_{n-1}$ with induction demonstrates that for every $n \in \mathbb{N}$ it holds that

$$\mathscr{S}(y_n) = \mathscr{S}(y_0). \tag{131}$$

In addition, note that item (i) in Lemma 2.8 ensures that for every $a \in \mathbb{R}$, $b \in (a, \infty)$, $z \in [a, b]$ it holds that $r_{a,b}(z) = z$. Hence, we obtain that for every $z \in D$ it holds that

$$\mathscr{S}(z) = z. \tag{132}$$

Combining this with (129) and (131) establishes that for every $n \in \mathbb{N} \cap [N, \infty)$ it holds that

$$y_n = \mathscr{S}(y_n) = \mathscr{S}(y_0). \tag{133}$$

The proof of Corollary 2.9 is thus complete. □

Framework 2.10 (Reflection principle for the simulation of time discrete reflected processes) *Let $d \in \mathbb{N}$, let $D \subseteq \mathbb{R}^d$ be a sufficiently regular closed set, let $\mathbf{n} \colon \partial_D \to \mathbb{R}^d$ be a suitable outer unit normal vector field associated to D, let $\mathfrak{c} \colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ satisfy for every $x, y \in \mathbb{R}^d$ that*

$$\mathfrak{c}(x, y) = x + \big[\inf(\{r \in [0, 1] : x + r(y - x) \notin D\} \cup \{1\})\big](y - x), \tag{134}$$

let $\mathscr{R} \colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$ satisfy for every $x, y \in \mathbb{R}^d$ that

$$\mathscr{R}(x, y) = \begin{cases} (x, y) & : \mathfrak{c}(x, y) = x \\ \big(\mathfrak{c}(x, y), y - 2\langle y - \mathfrak{c}(x, y), \mathbf{n}(\mathfrak{c}(x, y))\rangle \mathbf{n}(\mathfrak{c}(x, y))\big) & : \mathfrak{c}(x, y) \notin \{x, y\} \\ (y, y) & : \mathfrak{c}(x, y) = y, \end{cases} \tag{135}$$

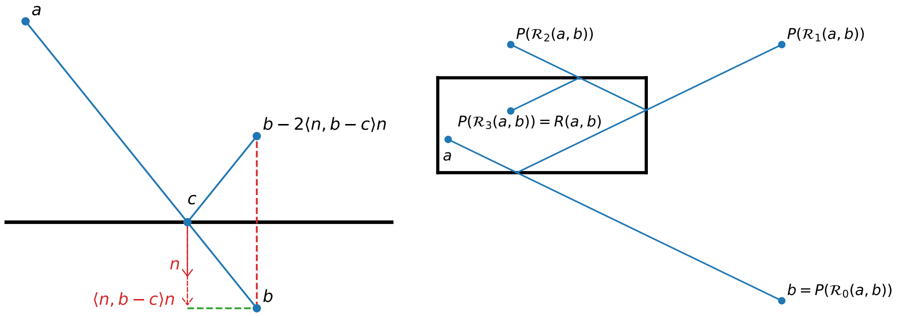

a
$b-2\langle n, b-c\rangle n$
c
n
$\langle n, b-c\rangle n$
b
$P(\mathcal{R}_2(a,b))$
$P(\mathcal{R}_1(a,b))$
$P(\mathcal{R}_3(a,b)) = R(a,b)$
a
$b = P(\mathcal{R}_0(a,b))$

Fig. 1 Two illustrations for Framework 2.10. The diagram on the left shows the reflection of a line segment from a to b on a hyperplane with unit normal vector n. The diagram on the right illustrates the recursive definition of the functions $\mathcal{R}_n \colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$, $n \in \mathbb{N}_0$, and $R \colon (\mathbb{R}^d)^2 \to \mathbb{R}$ defined in Framework 2.10 in the case where $d = 2$ and $D \subseteq \mathbb{R}^d$ is a closed rectangle

let $P \colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ *satisfy for every* $x, y \in \mathbb{R}^d$ *that* $P(x, y) = y$, *let* $\mathcal{R}_n \colon (\mathbb{R}^d)^2 \to (\mathbb{R}^d)^2$, $n \in \mathbb{N}_0$, *satisfy for every* $n \in \mathbb{N}_0$, $x, y \in \mathbb{R}^d$ *that* $\mathcal{R}_0(x, y) = (x, y)$ *and* $\mathcal{R}_{n+1}(x, y) = \mathscr{R}(\mathcal{R}_n(x, y))$, *and let* $R \colon (\mathbb{R}^d)^2 \to \mathbb{R}^d$ *satisfy for every* $x, y \in \mathbb{R}^d$ *that*

$$R(x, y) = \lim\nolimits_{n\to\infty} P(\mathcal{R}_n(x, y)) \tag{136}$$

(cf. Lemma 2.1).

Note that in Framework 2.10 above, for every $d \in \mathbb{N}$, every closed set $D \subseteq \mathbb{R}^d$ with a smooth boundary ∂_D, every suitable outer unit normal vector field $\mathbf{n} \colon \partial_D \to \mathbb{R}^d$, and every $a \in D\backslash\partial_D$, $b \in \mathbb{R}^d\backslash D$ we have

(i) that $\mathfrak{c}(a, b)$ is the point closest to a on the line segment from a to b which lies on ∂_D and
(ii) that $\mathscr{R}(a, b)$ is the reflection of b on the hyperplane through $\mathfrak{c}(a, b)$ tangent to ∂_D (cf. Fig. 1).

In view of this, we can, roughly speaking, think of $R(a, b)$ as the point where a particle that travels a length of $\|b - a\|$ starting in a going towards b and getting reflected every time it hits the boundary of D, ends up; cf. Fig. 1.

Also, observe that in Framework 2.10 above, we have used the vague notions of "sufficiently regular set" and "suitable outer unit normal vector field". Corollaries 2.7 and 2.9 and Proposition 2.4 above provide more information on two particular special cases of the construction in Framework 2.10. More specifically, we consider the conditions satisfied in (at least) the following two cases:

(a) The case where $D = \{x \in \mathbb{R}^d \colon \|x\| = 1\}$ and where $\mathbf{n} \colon \partial_D \to \mathbb{R}^d$ satisfies for every $x \in \partial_D = \{x \in \mathbb{R}^d \colon \|x\| = 1\}$ that $\mathbf{n}(x) = x$. Note that in this case, Proposition 2.4 demonstrates that the limit in (136) exists.
(b) The case where for every $k \in \{1, 2, \dots, d\}$ there exist $\mathfrak{a}_k^0 \in \mathbb{R}$, $\mathfrak{a}_k^1 \in (\mathfrak{a}_k^0, \infty)$ such that $D = [\mathfrak{a}_1^0, \mathfrak{a}_1^1] \times [\mathfrak{a}_2^0, \mathfrak{a}_2^1] \times \cdots \times [\mathfrak{a}_d^0, \mathfrak{a}_d^1]$, where $e_1 = (1, 0, 0, \dots, 0)$, $e_2 = (0, 1, 0, \dots, 0), \dots, e_d = (0, \dots, 0, 0, 1) \in \mathbb{R}^d$, and where $\mathbf{n} \colon \partial_D \to \mathbb{R}^d$, satisfies for every $x = (x_1, \dots, x_d) \in \partial_D = \{x = (x_1, \dots, x_d) \in D \colon (\exists k \in \{1, 2, \dots, d\} \colon x_k \in \{\mathfrak{a}_k^0, \mathfrak{a}_k^1\})\}$, $k \in \{1, 2, \dots, d\}$ with $k = \min\{l \in \{1, 2, \dots, d\} \colon x_l \in \{\mathfrak{a}_l^0, \mathfrak{a}_l^1\}\}$ that

$$\mathbf{n}(x) = \begin{cases} -e_k & : x_k = \mathfrak{a}_k^0 \\ e_k & : x_k = \mathfrak{a}_k^1. \end{cases} \tag{137}$$

Note that in this case, Corollary 2.7 demonstrates that the limit in (136) exists and Corollary 2.9 shows how the function R can be practically computed (up to a zero set).

2.3 Description of the proposed approximation method in a special case

Framework 2.11 (Special case of the machine learning-based approximation method) *Assume Framework 2.10, let* $T, \gamma \in (0,\infty)$*,* $N, M, K \in \mathbb{N}$*,* $g \in C^2(\mathbb{R}^d, \mathbb{R})$*,* $\mathfrak{d}, \mathfrak{h} \in \mathbb{N}\backslash\{1\}$*,* $t_0, t_1, \ldots, t_N \in [0,T]$ *satisfy* $\mathfrak{d} = \mathfrak{h}(N+1)d(d+1)$ *and*

$$0 = t_0 < t_1 < \cdots < t_N = T, \tag{138}$$

let $\tau_0, \tau_1, \ldots, \tau_N \in [0,T]$ *satisfy for every* $n \in \{0, 1, \ldots, N\}$ *that* $\tau_n = T - t_{N-n}$*, let* $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ *be measurable, let* $(\Omega, \mathcal{F}, \mathbb{P}, (\mathcal{F}_t)_{t\in[0,T]})$ *be a filtered probability space, let* $\xi^m : \Omega \to \mathbb{R}^d$*,* $m \in \mathbb{N}$*, be i.i.d.* $\mathcal{F}_0$/$\mathcal{B}(\mathbb{R}^d)$*-measurable random variables, let* $W^m : [0,T] \times \Omega \to \mathbb{R}^d$*,* $m \in \mathbb{N}$*, be i.i.d. standard* $(\mathcal{F}_t)_{t\in[0,T]}$*-Brownian motions, for every* $m \in \mathbb{N}$ *let* $\mathcal{Y}^m : \{0, 1, \ldots, N\} \times \Omega \to \mathbb{R}^d$ *be the stochastic process which satisfies for every* $n \in \{0, 1, \ldots, N-1\}$ *that* $\mathcal{Y}^m_0 = \xi^m$ *and*

$$\mathcal{Y}^m_{n+1} = R\big(\mathcal{Y}^m_n, \mathcal{Y}^m_n + \sqrt{2}(W^m_{\tau_{n+1}} - W^m_{\tau_n})\big), \tag{139}$$

let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}^d$ *satisfy for every* $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ *that*

$$\mathcal{L}(x) = \left(\frac{\exp(x_1)}{\exp(x_1)+1}, \ldots, \frac{\exp(x_d)}{\exp(x_d)+1}\right), \tag{140}$$

for every $\theta = (\theta_1, \ldots, \theta_{\mathfrak{d}}) \in \mathbb{R}^{\mathfrak{d}}$*,* $k, l, v \in \mathbb{N}$ *with* $v + l(k+1) \le \mathfrak{d}$ *let* $A^{\theta,v}_{k,l} : \mathbb{R}^k \to \mathbb{R}^l$ *satisfy for every* $x = (x_1, \ldots, x_k) \in \mathbb{R}^k$ *that*

$$A^{\theta,v}_{k,l}(x) = \left(\theta_{v+kl+1} + \left[\sum_{i=1}^{k} x_i\, \theta_{v+i}\right], \ldots, \theta_{v+kl+l} + \left[\sum_{i=1}^{k} x_i\, \theta_{v+(l-1)k+i}\right]\right), \tag{141}$$

let $\mathbb{V}_n : \mathbb{R}^{\mathfrak{d}} \times \mathbb{R}^d \to \mathbb{R}$*,* $n \in \{0, 1, \ldots, N\}$*, satisfy for every* $n \in \{1, 2, \ldots, N\}$*,* $\theta \in \mathbb{R}^{\mathfrak{d}}$*,* $x \in \mathbb{R}^d$ *that* $\mathbb{V}_0(\theta, x) = g(x)$ *and*

$$\begin{aligned}&\mathbb{V}_n(\theta, x) = \\ &\big(A^{\theta,(\mathfrak{h}n+\mathfrak{h}-1)d(d+1)}_{d,1} \circ \mathcal{L} \circ A^{\theta,(\mathfrak{h}n+\mathfrak{h}-2)d(d+1)}_{d,d} \circ \cdots \circ \mathcal{L} \circ A^{\theta,(\mathfrak{h}n+1)d(d+1)}_{d,d} \circ \mathcal{L} \circ A^{\theta,\mathfrak{h}nd(d+1)}_{d,d}\big)(x),\end{aligned} \tag{142}$$

let $\nu_x : \mathcal{B}(D) \to [0,1]$*,* $x \in D$*, be probability measures, for every* $x \in D$ *let* $Z^{n,m}_{x,k} : \Omega \to D$*,* $k, n, m \in \mathbb{N}$*, be i.i.d. random variables which satisfy for every* $A \in \mathcal{B}(D)$ *that* $\mathbb{P}(Z^{1,1}_{x,1} \in A) = \nu_x(A)$*, let* $\Theta^n : \mathbb{N}_0 \times \Omega \to \mathbb{R}^{\mathfrak{d}}$*,* $n \in \{0, 1, \ldots, N\}$*, be stochastic processes, for every* $n \in \{1, 2, \ldots, N\}$*,* $m \in \mathbb{N}$ *let* $\phi^{n,m} : \mathbb{R}^{\mathfrak{d}} \times \Omega \to \mathbb{R}$ *satisfy for every* $\theta \in \mathbb{R}^{\mathfrak{d}}$*,* $\omega \in \Omega$ *that*

$$\begin{aligned}\phi^{n,m}(\theta, \omega) = \Bigg[&\mathbb{V}_n\big(\theta, \mathcal{Y}^m_{N-n}(\omega)\big) - \mathbb{V}_{n-1}\big(\Theta^{n-1}_M(\omega), \mathcal{Y}^m_{N-n+1}(\omega)\big) \\ &- \tfrac{(t_n - t_{n-1})}{K}\left[\sum_{k=1}^{K} f\big(\mathbb{V}_{n-1}(\Theta^{n-1}_M(\omega), \mathcal{Y}^m_{N-n+1}(\omega)), \mathbb{V}_{n-1}(\Theta^{n-1}_M(\omega), Z^{n,m}_{\mathcal{Y}^m_{N-n+1}(\omega),k}(\omega))\big)\right]\Bigg]^2,\end{aligned} \tag{143}$$

for every $n \in \{1, 2, \dots, N\}$, $m \in \mathbb{N}$ *let* $\Phi^{n,m} \colon \mathbb{R}^{\mathfrak{d}} \times \Omega \to \mathbb{R}^{\mathfrak{d}}$ *satisfy for every* $\theta \in \mathbb{R}^{\mathfrak{d}}$, $\omega \in \Omega$ *that* $\Phi^{n,m}(\theta, \omega) = (\nabla_\theta \phi^{n,m})(\theta, \omega)$, *and assume for every* $n \in \{1, 2, \dots, N\}$, $m \in \mathbb{N}$ *that*

$$\Theta^n_m = \Theta^n_{m-1} - \gamma\, \Phi^{n,m}(\Theta^n_{m-1}). \tag{144}$$

As indicated in Sect. 2.1 above, the algorithm described in Framework 2.11 computes an approximation for a solution of the PDE in (1), i.e., a function $u \in C^{1,2}([0, T] \times D, \mathbb{R})$ which has at most polynomially growing derivatives, which satisfies for every $t \in (0, T]$, $x \in \partial_D$ that $\langle \mathbf{n}(x), (\nabla_x u)(t, x)\rangle = 0$ and which satisfies for every $t \in [0, T]$, $x \in D$ that $u(0, x) = g(x)$, $\int_D |f(u(t, x), u(t, \mathbf{x}))|\, \nu_x(d\mathbf{x}) < \infty$, and

$$\left(\tfrac{\partial}{\partial t} u\right)(t, x) = (\Delta_x u)(t, x) + \int_D f(u(t, x), u(t, \mathbf{x}))\, \nu_x(d\mathbf{x}). \tag{145}$$

Let us now add some explanatory comments on the objects and notations employed in Framework 2.11 above. The algorithm in Framework 2.11 decomposes the time interval $[0, T]$ into N subintervals at the times $t_0, t_1, t_2, \dots, t_N \in [0, T]$ (cf. (138)). For every $n \in \{1, 2, \dots, N\}$ we aim to approximate the function $\mathbb{R}^d \ni x \mapsto u(t_n, x) \in \mathbb{R}$ by a suitable (realization function of a) fully-connected feedforward neural network. Each of these neural networks is an alternating composition of $\mathfrak{h} - 1$ affine linear functions from $\mathbb{R}^d$ to $\mathbb{R}^d$ (where we think of $\mathfrak{h} \in \mathbb{N}\backslash\{1\}$ as the *length* or *depth* of the neural network), of $\mathfrak{h} - 1$ instances of a d-dimensional version of the standard logistic function, and finally of an affine linear function from $\mathbb{R}^d$ to $\mathbb{R}$. Every such neural network can be specified by means of $(\mathfrak{h} - 1)(d^2 + d) + d + 1 \le \mathfrak{h} d(d + 1)$ real parameters and so $N + 1$ of these neural networks can be specified by a parameter vector of length $\mathfrak{d} = \mathfrak{h}(N + 1)d(d + 1) \in \mathbb{N}$. Note that $\mathcal{L} \colon \mathbb{R}^d \to \mathbb{R}^d$ in Framework 2.11 above denotes the d-dimensional version of the standard logistic function (cf. (140)) and for every $k, l, v \in \mathbb{N}$, $\theta = (\theta_1, \dots, \theta_{\mathfrak{d}}) \in \mathbb{R}^{\mathfrak{d}}$ with $v + kl + l \le \mathfrak{d}$ the function $A^{\theta, v}_{k,l} \colon \mathbb{R}^k \to \mathbb{R}^l$ in Framework 2.11 denotes an affine linear function specified by means of the parameters $\theta_{v+1}, \theta_{v+2}, \dots, \theta_{v+kl+l}$ (cf. (141)). Furthermore, observe that for every $n \in \{1, 2, \dots, N\}$, $\theta \in \mathbb{R}^{\mathfrak{d}}$ the function

$$\mathbb{R}^d \ni x \mapsto \mathbb{V}_n(\theta, x) \in \mathbb{R} \tag{146}$$

denotes a neural network specified by means of the parameters $\mathfrak{h} n d(d + 1) + 1, \mathfrak{h} n d(d + 1) + 2, \dots, (\mathfrak{h} n + \mathfrak{h} - 1)d(d + 1) + d + 1$.

The goal of the optimization algorithm in Framework 2.11 above is to find a suitable parameter vector $\theta \in \mathbb{R}^{\mathfrak{d}}$ such that for every $n \in \{1, 2, \dots, N\}$ the neural network $\mathbb{R}^d \ni x \mapsto \mathbb{V}_n(\theta, x) \in \mathbb{R}$ is a good approximation for the solution $\mathbb{R}^d \ni x \mapsto u(t_n, x) \in \mathbb{R}$ to the PDE in (145) at time t_n. This is done by performing successively for each $n \in \{1, 2, \dots, N\}$ a plain vanilla stochastic gradient descent (SGD) optimization on a suitable loss function (cf. (144)).

Observe that for every $n \in \{1, 2, \dots, N\}$ the stochastic process $\Theta^n \colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^{\mathfrak{d}}$ describes the successive estimates computed by the SGD algorithm for the parameter vector that represents (via $\mathbb{V}_n \colon \mathbb{R}^{\mathfrak{d}} \times \mathbb{R}^d \to \mathbb{R}$) a suitable approximation to the solution $\mathbb{R}^d \ni x \mapsto u(t_n, x) \in \mathbb{R}$ of the PDE in (145) at time t_n. Next note that $M \in \mathbb{N}$ in Framework 2.11 above denotes the number of gradient descent steps taken for each $n \in \{1, 2, \dots, N\}$ and that $\gamma \in (0, \infty)$ denotes the learning rate employed in the SGD algorithm. Moreover, observe that for every $n \in \{1, 2, \dots, N\}$, $m \in \{1, 2, \dots, M\}$ the function $\phi^{n,m} \colon \mathbb{R}^{\mathfrak{d}} \times \Omega \to \mathbb{R}$ denotes the loss function employed in the mth gradient descent step during the approximation of the solution of the PDE in (145) at time t_n (cf. (143)). The loss functions employ a family of i.i.d. time-discrete stochastic processes $\mathcal{Y}^m \colon \{0, 1, \dots N\} \times \Omega \to \mathbb{R}^d$, $m \in \mathbb{N}$, which we think of as discretizations of suitable reflected Brownian motions (cf. (139)). In addition, for every $n \in \{1, 2, \ldots, N\}$, $m \in \{1, 2, \ldots, M\}$, $x \in D$ the loss function $\phi^{n,m} \colon \mathbb{R}^{\mathfrak{d}} \times \Omega \to \mathbb{R}$ employs a family of i.i.d. random variables $Z^{n,m}_{x,k} \colon \Omega \to D$, $k \in \mathbb{N}$, which are used for the Monte Carlo approximation of the non-local term in the PDE in (145) whose solution we are trying to approximate. The number of samples used in these Monte Carlo approximations is denoted by $K \in \mathbb{N}$ in Framework 2.11 above.

Finally, for sufficiently large $N, M, K \in \mathbb{N}$ and sufficiently small $\gamma \in (0, \infty)$ the algorithm in Framework 2.11 above yields for every $n \in \{1, 2, \ldots, N\}$ a (random) parameter vector $\Theta^n_M \colon \Omega \to \mathbb{R}^{\mathfrak{d}}$ which represents a function $\mathbb{R}^d \times \Omega \ni (x, \omega) \mapsto \mathbb{V}_n(\Theta^n_M(\omega), x) \in \mathbb{R}$ that we think of as providing for every $x \in D$ a suitable approximation

$$\mathbb{V}_n(\Theta^n_M, x) \approx u(t_n, x). \tag{147}$$

3 Machine learning-based approximation method in the general case

In this section we describe in Framework 3.1 in Sect. 3.2 below the full version of our deep learning-based method for approximating solutions of non-local nonlinear PDEs with Neumann boundary conditions (see Sect. 3.1 for a description of the class of PDEs our approximation method applies to), which generalizes the algorithm introduced in Framework 2.11 in Sect. 2.3 above and which we apply in Sect. 5 below to several examples of non-local nonlinear PDEs.

3.1 PDEs under consideration

Let $T \in (0, \infty)$, $d \in \mathbb{N}$, let $D \subseteq \mathbb{R}^d$ be a closed set with sufficiently smooth boundary ∂_D, let $\mathbf{n} \colon \partial_D \to \mathbb{R}^d$ be an outer unit normal vector field associated to D, let $g \colon D \to \mathbb{R}$, $\mu \colon D \to \mathbb{R}^d$, and $\sigma \colon D \to \mathbb{R}^{d \times d}$ be continuous, let $\nu_x \colon \mathcal{B}(D) \to [0, 1]$, $x \in D$, be probability measures, let $f \colon [0, T] \times D \times D \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be measurable, let $u = (u(t, x))_{(t,x) \in [0,T] \times D} \in C^{1,2}([0, T] \times D, \mathbb{R})$ have at most polynomially growing partial derivatives, assume for every $t \in [0, T]$, $x \in \partial_D$ that $\langle \mathbf{n}(x), (\nabla_x u)(t, x) \rangle = 0$, and assume for every $t \in [0, T]$, $x \in D$ that $u(0, x) = g(x)$, $\int_D \left| f\big(t, x, \mathbf{x}, u(t, x), u(t, \mathbf{x})\big) \right| \nu_x(d\mathbf{x}) < \infty$, and

$$\begin{aligned} \big(\tfrac{\partial}{\partial t} u\big)(t, x) = \int_D & f\big(t, x, \mathbf{x}, u(t, x), u(t, \mathbf{x})\big) \, \nu_x(d\mathbf{x}) \\ & + \big\langle \mu(x), (\nabla_x u)(t, x) \big\rangle + \tfrac{1}{2} \operatorname{Trace}\big(\sigma(x)[\sigma(x)]^* (\operatorname{Hess}_x u)(t, x)\big). \end{aligned} \tag{148}$$

Our goal is to approximately calculate under suitable hypotheses the solution $u \colon [0, T] \times D \to \mathbb{R}$ of the PDE in (148).

3.2 Description of the proposed approximation method in the general case

Framework 3.1 (General case of the machine learning-based approximation method) *Assume Framework 2.10, let* $T \in (0, \infty)$, $N, \varrho, \mathfrak{d}, \varsigma \in \mathbb{N}$, $(M_n)_{n \in \mathbb{N}_0} \subseteq \mathbb{N}$, $(K_n)_{n \in \mathbb{N}} \subseteq \mathbb{N}$, $(J_m)_{m \in \mathbb{N}} \subseteq \mathbb{N}$, $t_0, t_1, \ldots, t_N \in [0, T]$ *satisfy*

$$0 = t_0 < t_1 < \ldots < t_N = T, \tag{149}$$

let $\tau_0, \tau_1, \ldots, \tau_N \in [0, T]$ *satisfy for every* $n \in \{0, 1, \ldots, N\}$ *that* $\tau_n = T - t_{N-n}$, *let* $\nu_x \colon \mathcal{B}(D) \to [0, 1]$, $x \in D$, *be probability measures, for every* $x \in D$ *let* $Z^{n,m,j}_{x,k} \colon \Omega \to D$,

$k, n, m, j \in \mathbb{N}$, be i.i.d. random variables which satisfy for every $A \in \mathcal{B}(D)$ that $\mathbb{P}(Z^{1,1,1}_{x,1} \in A) = \nu_x(A)$, let $f\colon [0,T] \times D \times D \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be measurable, let $(\Omega, \mathcal{F}, \mathbb{P}, (\mathcal{F}_t)_{t\in[0,T]})$ be a filtered probability space, for every $n \in \{1, 2, \dots, N\}$ let $W^{n,m,j}\colon [0,T] \times \Omega \to \mathbb{R}^d$, $m, j \in \mathbb{N}$, be i.i.d. standard $(\mathcal{F}_t)_{t\in[0,T]}$-Brownian motions, for every $n \in \{1, 2, \dots, N\}$ let $\xi^{n,m,j}\colon \Omega \to \mathbb{R}^d$, $m, j \in \mathbb{N}$, be i.i.d. $\mathcal{F}_0$/$\mathcal{B}(\mathbb{R}^d)$-measurable random variables, let $H\colon [0,T]^2 \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ be a function, for every $j \in \mathbb{N}$, $\mathbf{s} \in \mathbb{R}^{\varsigma}$, $n \in \{0, 1, \dots, N\}$ let $\mathbb{V}^{j,\mathbf{s}}_n\colon \mathbb{R}^{\mathfrak{d}} \times \mathbb{R}^d \to \mathbb{R}$ be a function, for every $n \in \{1, 2, \dots, N\}$, $m, j \in \mathbb{N}$ let $\mathcal{Y}^{n,m,j}\colon \{0, 1, \dots, N\} \times \Omega \to \mathbb{R}^d$ be a stochastic process which satisfies for every $k \in \{0, 1, \dots, N-1\}$ that $\mathcal{Y}^{n,m,j}_0 = \xi^{n,m,j}$ and

$$\mathcal{Y}^{n,m,j}_{k+1} = H\big(\tau_{k+1}, \tau_k, \mathcal{Y}^{n,m,j}_k, W^{n,m,j}_{\tau_{k+1}} - W^{n,m,j}_{\tau_k}\big), \tag{150}$$

let $\Theta^n\colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^{\mathfrak{d}}$, $n \in \{0, 1, \dots, N\}$, be stochastic processes, for every $n \in \{1, 2, \dots, N\}$, $m \in \mathbb{N}$, $\mathbf{s} \in \mathbb{R}^{\varsigma}$ let $\phi^{n,m,\mathbf{s}}\colon \mathbb{R}^{\mathfrak{d}} \times \Omega \to \mathbb{R}$ satisfy for every $\theta \in \mathbb{R}^{\mathfrak{d}}$, $\omega \in \Omega$ that

$$\begin{aligned}\phi^{n,m,\mathbf{s}}(\theta, \omega) = & \frac{1}{J_m} \sum_{j=1}^{J_m} \Bigg[\mathbb{V}^{j,\mathbf{s}}_n\big(\theta, \mathcal{Y}^{n,m,j}_{N-n}(\omega)\big) - \mathbb{V}^{j,\mathbf{s}}_{n-1}\big(\Theta^{n-1}_{M_{n-1}}(\omega), \mathcal{Y}^{n,m,j}_{N-n+1}(\omega)\big) \\ & - \tfrac{(t_n - t_{n-1})}{K_n} \Bigg[\sum_{k=1}^{K_n} f\Big(t_{n-1}, \mathcal{Y}^{n,m,j}_{N-n+1}(\omega), Z^{n,m,j}_{\mathcal{Y}^{n,m,j}_{N-n+1}(\omega),k}(\omega), \\ & bV^{j,\mathbf{s}}_{n-1}\big(\Theta^{n-1}_{M_{n-1}}(\omega), \mathcal{Y}^{n,m,j}_{N-n+1}(\omega)\big), \mathbb{V}^{j,\mathbf{s}}_{n-1}\big(\Theta^{n-1}_{M_{n-1}}(\omega), Z^{n,m,j}_{\mathcal{Y}^{n,m,j}_{N-n+1}(\omega),k}(\omega)\big)\Big)\Bigg]\Bigg]^2,\end{aligned} \tag{151}$$

for every $n \in \{1, 2, \dots, N\}$, $m \in \mathbb{N}$, $\mathbf{s} \in \mathbb{R}^{\varsigma}$ let $\Phi^{n,m,\mathbf{s}}\colon \mathbb{R}^{\mathfrak{d}} \times \Omega \to \mathbb{R}^{\mathfrak{d}}$ satisfy for every $\omega \in \Omega$, $\theta \in \{\vartheta \in \mathbb{R}^{\mathfrak{d}}\colon (\mathbb{R}^{\mathfrak{d}} \ni \eta \mapsto \phi^{n,m,\mathbf{s}}(\eta, \omega) \in \mathbb{R})$ is differentiable at $\vartheta\}$ that

$$\Phi^{n,m,\mathbf{s}}(\theta, \omega) = (\nabla_\theta \phi^{n,m,\mathbf{s}})(\theta, \omega), \tag{152}$$

let $\mathcal{S}^n\colon \mathbb{R}^{\varsigma} \times \mathbb{R}^{\mathfrak{d}} \times (\mathbb{R}^d)^{\{0,1,\dots,N\}\times\mathbb{N}} \to \mathbb{R}^{\varsigma}$, $n \in \{1, 2, \dots, N\}$, be functions, for every $n \in \{1, 2, \dots, N\}$, $m \in \mathbb{N}$ let $\psi^n_m\colon \mathbb{R}^{\varrho} \to \mathbb{R}^{\mathfrak{d}}$ and $\Psi^n_m\colon \mathbb{R}^{\varrho} \times \mathbb{R}^{\mathfrak{d}} \to \mathbb{R}^{\varrho}$ be functions, and for every $n \in \{1, 2, \dots, N\}$ let $\mathbb{S}^n\colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^{\varsigma}$ and $\Xi^n\colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^{\varrho}$ be stochastic processes which satisfy for every $m \in \mathbb{N}$ that

$$\mathbb{S}^n_m = \mathcal{S}^n\big(\mathbb{S}^n_{m-1}, \Theta^n_{m-1}, (\mathcal{Y}^{n,m,i}_k)_{(k,i)\in\{0,1,\dots,N\}\times\mathbb{N}}\big), \tag{153}$$

$$\Xi^n_m = \Psi^n_m(\Xi^n_{m-1}, \Phi^{n,m,\mathbb{S}^n_m}(\Theta^n_{m-1})), \qquad \text{and} \qquad \Theta^n_m = \Theta^n_{m-1} - \psi^n_m(\Xi^n_m). \tag{154}$$

In the setting of Framework 3.1 above we think under suitable hypotheses for sufficiently large $N \in \mathbb{N}$, sufficiently large $(M_n)_{n\in\mathbb{N}_0} \subseteq \mathbb{N}$, sufficiently large $(K_n)_{n\in\mathbb{N}} \subseteq \mathbb{N}$, every $n \in \{0, 1, \dots, N\}$, and every $x \in D$ of $\mathbb{V}^{1,\mathbb{S}^n_{M_n}}_n(\Theta^n_{M_n}, x)\colon \Omega \to \mathbb{R}$ as a suitable approximation

$$\mathbb{V}^{1,\mathbb{S}^n_{M_n}}_n(\Theta^n_{M_n}, x) \approx u(t_n, x) \tag{155}$$

of $u(t_n, x)$ where $u = (u(t,x))_{(t,x)\in[0,T]\times\mathbb{R}^d} \in C^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R})$ is a function with at most polynomially growing derivatives which satisfies for every $t \in (0,T]$, $x \in \partial_D$ that $\langle \mathbf{n}(x), (\nabla_x u)(t,x) \rangle = 0$ and which satisfies for every $t \in [0,T]$, $x \in D$ that $u(0,x) = g(x)$, $\int_D \big| f\big(t, x, \mathbf{x}, u(t,x), u(t,\mathbf{x})\big) \big| \, \nu_x(d\mathbf{x}) < \infty$, and

$$\begin{aligned}\big(\tfrac{\partial}{\partial t} u\big)(t,x) = & \int_D f\big(t, x, \mathbf{x}, u(t,x), u(t,\mathbf{x})\big) \, \nu_x(d\mathbf{x}) \\ & + \big\langle \mu(x), (\nabla_x u)(t,x) \big\rangle + \tfrac{1}{2} \operatorname{Trace}\big(\sigma(x)[\sigma(x)]^* (\operatorname{Hess}_x u)(t,x)\big)\end{aligned} \tag{156}$$

(cf. (148)). Compared to the simplified algorithm in Framework 2.11 above, the major new elements introduced in Framework 3.1 are the following:

(a) The numbers of gradient descent steps taken to compute approximations for the solution of the PDE at the times t_n, $n \in \{1, 2, \ldots, N\}$, are allowed to vary with n, and so are specified by a sequence $(M_n)_{n\in\mathbb{N}_0} \subseteq \mathbb{N}$ in Framework 3.1 above.

(b) The numbers of samples used for the Monte Carlo approximation of the non-local term in the approximation for the solution of the PDE at the times t_n, $n \in \{1, 2, \ldots, N\}$, are allowed to vary with n, and so are specified by a sequence $(K_n)_{n\in\mathbb{N}_0} \subseteq \mathbb{N}$ in Framework 3.1 above.

(c) The approximating functions $\mathbb{V}_n^{j,\mathbf{s}}$, $(j, \mathbf{s}, n) \in \mathbb{N} \times \mathbb{R}^{\varsigma} \times \{0, 1, \ldots, N\}$, in Framework 3.1 above are not specified concretely in order to allow for a variety of neural network architectures. For the concrete choice of these functions employed in our numerical simulations, we refer the reader to Sect. 5.

(d) For every $m \in \{1, 2, \ldots, M\}$ the loss function used in the mth gradient descent step may be computed using a minibatch of samples instead of just one sample (cf. (151)). The sizes of these minibatches are specified by a sequence $(J_m)_{m\in\mathbb{N}} \subseteq \mathbb{N}$.

(e) Compared to Framework 2.11 above, the more general form of the PDEs considered in this section (cf. (156)) requires more flexibility in the definition of the time-discrete stochastic processes $\mathcal{Y}^{n,m,j} \colon \{0, 1, \ldots, N\} \times \Omega \to \mathbb{R}^d$, $(n, m, j) \in \{1, 2, \ldots, N\} \times \mathbb{N} \times \mathbb{N}$, which are specified in Framework 3.1 above in terms of the Brownian motions $W^{n,m,j} \colon [0, T] \times \Omega \to \mathbb{R}^d$, $(n, m, j) \in \{1, 2, \ldots, N\} \times \mathbb{N} \times \mathbb{N}$, via a function $H \colon [0, T]^2 \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ (cf. (150)). We refer the reader to (181) in Sect. 5.1 below, (183) in Sect. 5.2 below, (185) in Sect. 5.3 below, (206) in Sect. 5.4 below, and (211) in Sect. 5.5 below for concrete choices of H in the approximation of various example PDEs.

(f) For every $n \in \{1, 2, \ldots, N\}$, $m \in \mathbb{N}$ the optimization step in (154) in Framework 3.1 above is specified generically in terms of the functions $\psi_m^n \colon \mathbb{R}^{\varrho} \to \mathbb{R}^{\mathfrak{d}}$ and $\Psi_m^n \colon \mathbb{R}^{\varrho} \times \mathbb{R}^{\mathfrak{d}} \to \mathbb{R}^{\varrho}$ and the random variable $\Xi_m^n \colon \Omega \to \mathbb{R}^{\varrho}$. This generic formulation covers a variety of SGD based optimization algorithms such as Adagrad [95], RMSprop, or Adam [96]. For example, in order to implement the Adam optimization algorithm, for every $n \in \{1, 2, \ldots, N\}$, $m \in \mathbb{N}$ the random variable Ξ_m^n can be used to hold suitable first and second moment estimates (see (175) and (176) in Sect. 5 below for the concrete specification of these functions implementing the Adam optimization algorithm).

(g) The processes $\mathbb{S}^n \colon \mathbb{N}_0 \times \Omega \to \mathbb{R}^{\varsigma}$, $n \in \{1, 2, \ldots, N\}$, and functions $\mathcal{S}^n \colon \mathbb{R}^{\varsigma} \times \mathbb{R}^{\mathfrak{d}} \times (\mathbb{R}^d)^{\{0,1,\ldots,N\}\times\mathbb{N}} \to \mathbb{R}^{\varsigma}$, $n \in \{1, 2, \ldots, N\}$, in Framework 3.1 above can be used to implement batch normalization; see [97] for details. Loosely speaking, for every $n \in \{1, 2, \ldots, N\}$, $m \in \mathbb{N}$ the random variable $\mathbb{S}_m^n \colon \Omega \to \mathbb{R}^{\varsigma}$ then holds mean and variance estimates of the outputs of each layer of the approximating neural networks related to the minibatches that are used as inputs to the neural networks in computing the loss function at the corresponding gradient descent step.

4 Multilevel Picard approximation method for non-local PDEs

In this section we introduce in Framework 4.1 in Sect. 4.1 below our extension of the full history recursive multilevel Picard approximation method for approximating solutions of non-local nonlinear PDEs with Neumann boundary conditions. The MLP method was first introduced in E et al. [71] and Hutzenthaler et al. [69] and later extended in a number of directions; see E et al. [98] and Beck et al. [53] for recent surveys. We also refer the reader to Becker et al. [99] and E et al. [70] for numerical simulations illustrating the performance of MLP methods across a range of example PDE problems.

In Sect. 4.2 below, we will specify five concrete examples of (non-local) nonlinear PDEs and describe how Framework 4.1 can be specialized to compute approximate solutions to these example PDEs. These computations will be used in Sect. 5 to obtain reference values to compare the deep learning-based approximation method proposed in Sect. 3 above against.

4.1 Description of the proposed approximation method

Framework 4.1 (Multilevel Picard approximation method) *Assume Framework 2.10, let* $c, T \in (0, \infty)$, $\mathfrak{I} = \bigcup_{n \in \mathbb{N}} \mathbb{Z}^n$, $f \in C([0,T] \times D \times D \times \mathbb{R} \times \mathbb{R}, \mathbb{R})$, $g \in C(D, \mathbb{R})$, $u \in C([0,T] \times D, \mathbb{R})$, *assume* $u|_{[0,T) \times D} \in C^{1,2}([0,T) \times D, \mathbb{R})$, *let* $\nu_x \colon \mathcal{B}(D) \to [0,1]$, $x \in D$, *be probability measures, for every* $x \in D$ *let* $Z^{\mathfrak{i}}_x \colon \Omega \to D$, $\mathfrak{i} \in \mathfrak{I}$, *be i.i.d. random variables, assume for every* $A \in \mathcal{B}(D)$, $\mathfrak{i} \in \mathfrak{I}$ *that* $\mathbb{P}(Z^{\mathfrak{i}}_x \in A) = \nu_x(A)$, *for every* $r_1 \in [-\infty, \infty)$, $r_2 \in [r_1, \infty]$ *let* $\phi_{r_1,r_2} \colon \mathbb{R} \to \mathbb{R}$ *satisfy for every* $y \in \mathbb{R}$ *that*

$$\phi_{r_1,r_2}(y) = \min\{r_2, \max\{r_1, y\}\}, \tag{157}$$

let $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space, let* $\mathcal{V}^{\mathfrak{i}} \colon \Omega \to (0,1)$, $\mathfrak{i} \in \mathfrak{I}$, *be independent* $\mathcal{U}_{(0,1)}$-*distributed random variables, let* $V^{\mathfrak{i}} \colon [0,T] \times \Omega \to [0,T]$, $\mathfrak{i} \in \mathfrak{I}$, *satisfy for every* $t \in [0,T]$, $\mathfrak{i} \in \mathfrak{I}$ *that*

$$V^{\mathfrak{i}}_t = t + (T-t)\mathcal{V}^{\mathfrak{i}}, \tag{158}$$

let $W^{\mathfrak{i}} \colon [0,T] \times \Omega \to \mathbb{R}^d$, $\mathfrak{i} \in \mathfrak{I}$, *be independent standard Brownian motions, assume* $(\mathcal{V}^{\mathfrak{i}})_{\mathfrak{i} \in \mathfrak{I}}$ *and* $(W^{\mathfrak{i}})_{\mathfrak{i} \in \mathfrak{I}}$ *are independent, let* $\mu \colon \mathbb{R}^d \to \mathbb{R}^d$ *and* $\sigma \colon \mathbb{R}^d \to \mathbb{R}^{d \times d}$ *be globally Lipschitz continuous, for every* $x \in \mathbb{R}^d$, $\mathfrak{i} \in \mathfrak{I}$, $t \in [0,T]$ *let* $X^{x,\mathfrak{i}}_t = (X^{x,\mathfrak{i}}_{t,s})_{s \in [t,T]} \colon [t,T] \times \Omega \to \mathbb{R}^d$ *be a stochastic process with continuous sample paths, let* $(K_{n,l,m})_{n,l,m \in \mathbb{N}_0} \subseteq \mathbb{N}$, *for every* $\mathfrak{i} \in \mathfrak{I}$, $n, M \in \mathbb{N}_0$, $r_1 \in [-\infty, \infty)$, $r_2 \in [r_1, \infty]$ *let* $U^{\mathfrak{i}}_{n,M,r_1,r_2} \colon [0,T] \times \mathbb{R}^d \times \Omega \to \mathbb{R}^k$ *satisfy for every* $t \in [0,T]$, $x \in \mathbb{R}^d$ *that*

$$\begin{aligned} U^{\mathfrak{i}}_{n,M,r_1,r_2}(t,x) = \Bigg[\sum_{l=0}^{n-1} \tfrac{(T-t)}{M^{n-l}} \sum_{m=1}^{M^{n-l}} \tfrac{1}{K_{n,l,m}} \sum_{k=1}^{K_{n,l,m}} \Bigg[f\Big(V^{(\mathfrak{i},l,m)}_t, X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t}, Z^{(\mathfrak{i},l,m,k)}_{X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t}}, \\ \phi_{r_1,r_2}\Big(U^{(\mathfrak{i},l,m)}_{l,M,r_1,r_2}\big(V^{(\mathfrak{i},l,m)}_t, X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t} \big) \Big), \phi_{r_1,r_2}\Big(U^{(\mathfrak{i},l,m)}_{l,M,r_1,r_2}\big(V^{(\mathfrak{i},l,m)}_t, Z^{(\mathfrak{i},l,m,k)}_{X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t}} \big) \Big) \Big) \\ - \mathbb{1}_{\mathbb{N}}(l)\, f\Big(V^{(\mathfrak{i},l,m)}_t, X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t}, Z^{(\mathfrak{i},l,m,k)}_{X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t}}, \phi_{r_1,r_2}\Big(U^{(\mathfrak{i},l,-m)}_{\max\{l-1,0\},M,r_1,r_2}\big(V^{(\mathfrak{i},l,m)}_t, X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t} \big) \Big), \\ \phi_{r_1,r_2}\Big(U^{(\mathfrak{i},l,-m)}_{\max\{l-1,0\},M,r_1,r_2}\big(V^{(\mathfrak{i},l,m)}_t, Z^{(\mathfrak{i},l,m,k)}_{X^{x,(\mathfrak{i},l,m)}_{t,V^{(\mathfrak{i},l,m)}_t}} \big) \Big) \Big) \Bigg] \Bigg] + \frac{\mathbb{1}_{\mathbb{N}}(n)}{M^n} \Bigg[\sum_{m=1}^{M^n} g\big(X^{x,(\mathfrak{i},0,-m)}_{t,T} \big) \Bigg], \end{aligned} \tag{159}$$

assume for every $t \in [0,T)$, $x \in \partial_D$ *that* $\langle \mathbf{n}(x), (\nabla_x u)(t,x) \rangle = 0$, *and assume for every* $t \in [0,T)$, $x \in D$ *that* $\|u(t,x)\| \le c(1 + \|x\|^c)$, $u(T,x) = g(x)$, *and*

$$\begin{aligned} \big(\tfrac{\partial}{\partial t} u\big)(t,x) + \tfrac{1}{2} \operatorname{Trace}\big(\sigma(x)[\sigma(x)]^* (\operatorname{Hess}_x u)(t,x) \big) + \langle \mu(x), (\nabla_x u)(t,x) \rangle \\ + \int_D f(t, x, \mathbf{x}, u(t,x), u(t,\mathbf{x}))\, \nu_x(d\mathbf{x}) = 0. \end{aligned} \tag{160}$$

4.2 Examples for the approximation method

Example 4.2 (Fisher–KPP PDEs with Neumann boundary conditions) In this example we specialize Framework 4.1 to the case of certain Fisher–KPP PDEs with Neumann boundary conditions (cf., e.g., Bian et al. [36] and Wang et al. [40]).

Assume Framework 4.1, let $\epsilon \in (0, \infty)$, assume for every $t \in [0, T]$, $x, \mathbf{x} \in D$, $y, \mathbf{y} \in \mathbb{R}$, $v \in \mathbb{R}^d$ that $g(x) = \exp\bigl(-\frac{1}{4}\|x\|^2\bigr)$, $\mu(x) = (0, \ldots, 0)$, $\sigma(x)v = \epsilon v$, and $f(t, x, \mathbf{x}, y, \mathbf{y}) = y(1 - y)$, and assume that for every $x \in \mathbb{R}^d$, $\mathfrak{i} \in \mathfrak{I}$, $t \in [0, T]$, $s \in [t, T]$ it holds $\mathbb{P}$-a.s. that

$$X^{x,\mathfrak{i}}_{t,s} = R\biggl(x, x + \int_t^s \mu\bigl(X^{x,\mathfrak{i}}_{t,r}\bigr)\,\mathrm{d}r + \int_t^s \sigma\bigl(X^{x,\mathfrak{i}}_{t,r}\bigr)\,\mathrm{d}W^{\mathfrak{i}}_r\biggr) = R(x, x + \epsilon(W^{\mathfrak{i}}_s - W^{\mathfrak{i}}_t)). \tag{161}$$

The solution $u\colon [0, T] \times D \to \mathbb{R}$ of the PDE in (160) then satisfies that for every $t \in [0, T)$, $x \in \partial_D$ it holds that $\langle \mathbf{n}(x), (\nabla_x u)(t, x)\rangle = 0$ and that for every $t \in [0, T)$, $x \in D$ it holds that $u(T, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\bigl(\tfrac{\partial}{\partial t}u\bigr)(t, x) + \tfrac{\epsilon^2}{2}(\Delta_x u)(t, x) + u(t, x)\bigl(1 - u(t, x)\bigr) = 0. \tag{162}$$

Example 4.3 (Non-local competition PDEs) In this example we specialize Framework 4.1 to the case of certain non-local competition PDEs (cf., e.g., Doebeli and Ispolatov [47], Berestycki et al. [38], Perthame and Génieys [37], and Génieys et al. [42]).

Assume Framework 4.1, let $\mathfrak{s}, \epsilon \in (0, \infty)$, assume for every $x \in \mathbb{R}^d$, $A \in \mathcal{B}(\mathbb{R}^d)$ that $\nu_x(A) = \pi^{-d/2}\mathfrak{s}^{-d} \int_A \exp\bigl(-\mathfrak{s}^{-2}\|x - \mathbf{x}\|^2\bigr)\,\mathrm{d}\mathbf{x}$, assume for every $t \in [0, T]$, $v, x, \mathbf{x} \in \mathbb{R}^d$, $y, \mathbf{y} \in \mathbb{R}$ that $g(x) = \exp(-\frac{1}{4}\|x\|^2)$, $\mu(x) = (0, \ldots, 0)$, $\sigma(x)v = \epsilon v$, and $f(t, x, \mathbf{x}, y, \mathbf{y}) = y(1 - \mathbf{y}\pi^{d/2}\mathfrak{s}^d)$, and assume that for every $x \in \mathbb{R}^d$, $\mathfrak{i} \in \mathfrak{I}$, $t \in [0, T]$, $s \in [t, T]$ it holds $\mathbb{P}$-a.s. that

$$X^{x,\mathfrak{i}}_{t,s} = x + \int_t^s \mu\bigl(X^{x,\mathfrak{i}}_{t,r}\bigr)\,\mathrm{d}r + \int_t^s \sigma\bigl(X^{x,\mathfrak{i}}_{t,r}\bigr)\,\mathrm{d}W^{\mathfrak{i}}_r = x + \epsilon(W^{\mathfrak{i}}_s - W^{\mathfrak{i}}_t). \tag{163}$$

The solution $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (160) then satisfies that for every $t \in [0, T)$, $x \in \mathbb{R}^d$ it holds that $u(T, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\bigl(\tfrac{\partial}{\partial t}u\bigr)(t, x) + \tfrac{\epsilon^2}{2}(\Delta_x u)(t, x) + u(t, x)\left(1 - \int_{\mathbb{R}^d} u(t, \mathbf{x})\,\exp\bigl(-\tfrac{\|x-\mathbf{x}\|^2}{\mathfrak{s}^2}\bigr)\,\mathrm{d}\mathbf{x}\right) = 0. \tag{164}$$

Example 4.4 (Non-local sine-Gordon PDEs) In this example we specialize Framework 4.1 to the case of certain non-local sine-Gordon type PDEs (cf., e.g., Hairer and Shen [9], Barone et al. [6], and Coleman [8]).

Assume Framework 4.1, let $\mathfrak{s}, \epsilon \in (0, \infty)$, assume for every $x \in \mathbb{R}^d$, $A \in \mathcal{B}(\mathbb{R}^d)$ that $\nu_x(A) = \pi^{-d/2}\mathfrak{s}^{-d} \int_A \exp\bigl(-\mathfrak{s}^{-2}\|x - \mathbf{x}\|^2\bigr)\,\mathrm{d}\mathbf{x}$, assume for every $t \in [0, T]$, $v, x, \mathbf{x} \in \mathbb{R}^d$, $y, \mathbf{y} \in \mathbb{R}$ that $g(x) = \exp(-\frac{1}{4}\|x\|^2)$, $\mu(x) = (0, \ldots, 0)$, $\sigma(x)v = \epsilon v$, and $f(t, x, \mathbf{x}, y, \mathbf{y}) = \sin(y) - \mathbf{y}\pi^{d/2}\mathfrak{s}^d$, and assume that for every $x \in \mathbb{R}^d$, $\mathfrak{i} \in \mathfrak{I}$, $t \in [0, T]$, $s \in [t, T]$ it holds $\mathbb{P}$-a.s. that

$$X^{x,\mathfrak{i}}_{t,s} = x + \int_t^s \mu\bigl(X^{x,\mathfrak{i}}_{t,r}\bigr)\,\mathrm{d}r + \int_t^s \sigma\bigl(X^{x,\mathfrak{i}}_{t,r}\bigr)\,\mathrm{d}W^{\mathfrak{i}}_r = x + \epsilon(W^{\mathfrak{i}}_s - W^{\mathfrak{i}}_t). \tag{165}$$

The solution $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (160) then satisfies that for every $t \in [0, T)$, $x \in \mathbb{R}^d$ it holds that $u(T, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\bigl(\tfrac{\partial}{\partial t}u\bigr)(t, x) + \tfrac{\epsilon^2}{2}(\Delta_x u)(t, x) + \sin(u(t, x)) - \int_{\mathbb{R}^d} u(t, \mathbf{x})\,\exp\bigl(-\tfrac{\|x-\mathbf{x}\|^2}{\mathfrak{s}^2}\bigr)\,\mathrm{d}\mathbf{x} = 0. \tag{166}$$

Example 4.5 (Replicator-mutator PDEs) In this example we specialize Framework 4.1 to the case of certain d-dimensional replicator-mutator PDEs (cf., e.g., Hamel et al. [26]).

Assume Framework 4.1, let $\mathfrak{m}_1, \mathfrak{m}_2, \dots, \mathfrak{m}_d, \mathfrak{s}_1, \mathfrak{s}_2, \dots, \mathfrak{s}_d, \mathfrak{u}_1, \mathfrak{u}_2, \dots, \mathfrak{u}_d \in \mathbb{R}$, let $a \colon \mathbb{R}^d \to \mathbb{R}$ satisfy for every $x \in \mathbb{R}^d$ that $a(x) = -\frac{1}{2}\|x\|^2$, assume for every $x \in \mathbb{R}^d$, $A \in \mathcal{B}(\mathbb{R}^d)$ that $\nu_x(A) = \int_{A \cap [-1/2, 1/2]^d} \mathrm{d}\mathbf{x}$, assume for every $t \in [0, T]$, $v = (v_1, \dots, v_d)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, $\mathbf{x} \in \mathbb{R}^d$, $y, \mathbf{y} \in \mathbb{R}$ that $g(x) = (2\pi)^{-d/2} \big[\prod_{i=1}^d |\mathfrak{s}_i|^{-1/2}\big] \exp\big(-\sum_{i=1}^d \frac{(x_i - \mathfrak{u}_i)^2}{2\mathfrak{s}_i}\big)$, $\mu(x) = (0, \dots, 0)$, $\sigma(x)v = (\mathfrak{m}_1 v_1, \dots, \mathfrak{m}_d v_d)$, and

$$f(t, x, \mathbf{x}, y, \mathbf{y}) = y\big(a(x) - \mathbf{y} a(\mathbf{x})\big), \tag{167}$$

and assume that for every $x \in \mathbb{R}^d$, $\mathfrak{i} \in \mathfrak{I}$, $t \in [0, T]$, $s \in [t, T]$ it holds $\mathbb{P}$-a.s. that

$$X_{t,s}^{x,\mathfrak{i}} = x + \int_t^s \mu\big(X_{t,r}^{x,\mathfrak{i}}\big)\, \mathrm{d}r + \int_t^s \sigma\big(X_{t,r}^{x,\mathfrak{i}}\big)\, \mathrm{d}W_r^{\mathfrak{i}} = x + \sigma(0)(W_s^{\mathfrak{i}} - W_t^{\mathfrak{i}}). \tag{168}$$

The solution $u \colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (160) then satisfies that for every $t \in [0, T)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that $u(T, x) = (2\pi)^{-d/2} \big[\prod_{i=1}^d |\mathfrak{s}_i|^{-1/2}\big] \exp\big(-\sum_{i=1}^d \frac{(x_i - \mathfrak{u}_i)^2}{2\mathfrak{s}_i}\big)$ and

$$\big(\tfrac{\partial}{\partial t} u\big)(t, x) + u(t, x)\left(a(x) - \int_{[-1/2, 1/2]^d} u(t, \mathbf{x})\, a(\mathbf{x})\, \mathrm{d}\mathbf{x}\right) + \sum_{i=1}^d \tfrac{1}{2} |\mathfrak{m}_i|^2 \big(\tfrac{\partial^2}{\partial x_i^2} u\big)(t, x) = 0. \tag{169}$$

Example 4.6 (Allen–Cahn PDEs with conservation of mass) In this example we specialize Framework 4.1 to the case of certain Allen–Cahn PDEs with cubic nonlinearity, conservation of mass, and no-flux boundary conditions (cf., e.g., Rubinstein and Sternberg [10]).

Assume Framework 4.1, let $\epsilon \in (0, \infty)$, assume for every $x \in D$, $A \in \mathcal{B}(D)$ that $\nu_x(A) = \int_A \mathrm{d}\mathbf{x}$, assume for every $t \in [0, T]$, $x, \mathbf{x} \in D$, $y, \mathbf{y} \in \mathbb{R}$, $v \in \mathbb{R}^d$ that $g(x) = \exp(-\frac{1}{4}\|x\|^2)$, $\mu(x) = (0, \dots, 0)$, $\sigma(x)v = \epsilon v$, and $f(t, x, \mathbf{x}, y, \mathbf{y}) = y - y^3 - (\mathbf{y} - \mathbf{y}^3)$, and assume that for every $x \in \mathbb{R}^d$, $\mathfrak{i} \in \mathfrak{I}$, $t \in [0, T]$, $s \in [t, T]$ it holds $\mathbb{P}$-a.s. that

$$X_{t,s}^{x,\mathfrak{i}} = R\left(x, x + \int_t^s \mu\big(X_{t,r}^{x,\mathfrak{i}}\big)\, \mathrm{d}r + \int_t^s \sigma\big(X_{t,r}^{x,\mathfrak{i}}\big)\, \mathrm{d}W_r^{\mathfrak{i}}\right) = R(x, x + \epsilon(W_s^{\mathfrak{i}} - W_t^{\mathfrak{i}})). \tag{170}$$

The solution $u \colon [0, T] \times D \to \mathbb{R}$ of the PDE in (160) then satisfies that for every $t \in [0, T)$, $x \in \partial_D$ it holds that $\langle \mathbf{n}(x), (\nabla_x u)(t, x) \rangle = 0$ and that for every $t \in [0, T)$, $x \in D$ it holds that $u(T, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\big(\tfrac{\partial}{\partial t} u\big)(t, x) + \tfrac{\epsilon^2}{2} (\Delta_x u)(t, x) + u(t, x) - [u(t, x)]^3 - \int_{[-1/2, 1/2]^d} u(t, \mathbf{x}) - [u(t, \mathbf{x})]^3\, \mathrm{d}\mathbf{x} = 0. \tag{171}$$

5 Numerical simulations

In this section we illustrate the performance of the machine learning-based approximation method proposed in Framework 3.1 in Sect. 3.2 above by means of numerical simulations for five concrete (non-local) nonlinear PDEs; see Sects. 5.1, 5.2, 5.3, 5.4, and 5.5 below. In each of these numerical simulations we employ the general machine learning-based approximation method proposed in Framework 3.1 with certain 4-layer neural networks in conjunction with the Adam optimizer (cf. (175) and (176) in Framework 5.1 below and Kingma and Ba [96]).

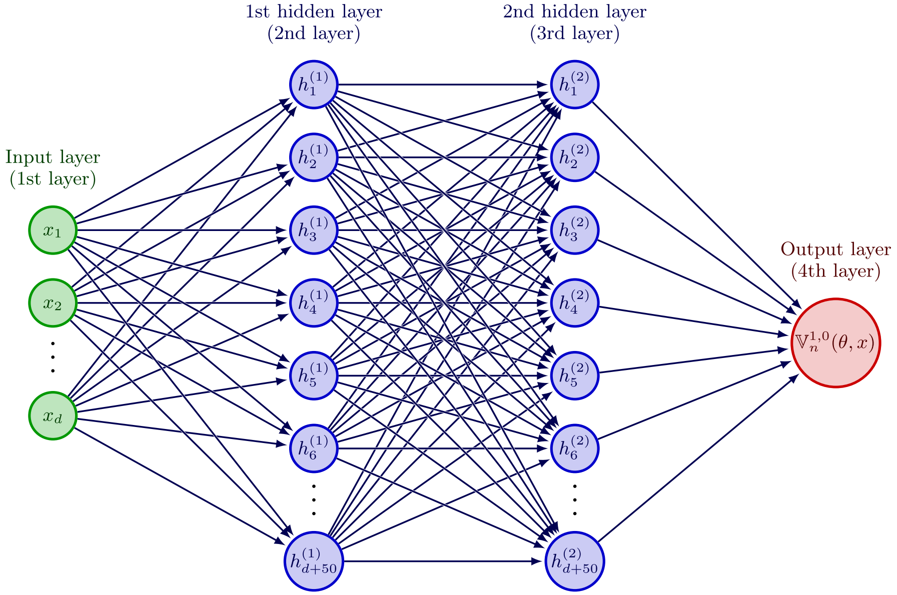


Fig. 2 Graphical illustration of the neural network architecture used in the numerical simulations. In Sects. 5.1, 5.2, 5.3, 5.4, and 5.5 we employ neural networks with 4 layers (corresponding to 3 affine linear transformations in the neural networks) with d neurons on the input layer (corresponding to a d-dimensional input layer), with $d + 50$ neurons on the 1st hidden layer (corresponding to a $(d + 50)$-dimensional 1st hidden layer), with $d + 50$ neurons on the 2nd hidden layer (corresponding to a $(d + 50)$-dimensional 2nd hidden layer), and with 1 neuron on the output layer (corresponding to a 1-dimensional output layer) in the numerical simulations

More precisely, in each of the numerical simulations in Sects. 5.1, 5.2, 5.3, 5.4, and 5.5 the functions $\mathbb{V}_n^{j,\mathbf{s}} \colon \mathbb{R}^{\mathfrak{d}} \times \mathbb{R}^d \to \mathbb{R}$ with $n \in \{1, 2, \ldots, N\}$, $j \in \{1, 2, \ldots, 8000\}$, $\mathbf{s} \in \mathbb{R}^{\varsigma}$ are implemented as N fully-connected feedforward neural networks. These neural networks consist of 4 layers (corresponding to 3 affine linear transformations in the neural networks) where the input layer is d-dimensional (with d neurons on the input layer), where the two hidden layers are both $(d + 50)$-dimensional (with $d + 50$ neurons on each of the two hidden layers), and where the output layer is 1-dimensional (with 1 neuron on the output layer). We refer to Fig. 2 for a graphical illustration of the neural network architecture used in the numerical simulations in Sects. 5.1, 5.2, 5.3, 5.4, and 5.5.

As activation functions just in front of the two hidden layers we employ, in Sects. 5.1, 5.2, 5.3, and 5.4 below, multidimensional versions of the hyperbolic tangent function

$$\mathbb{R} \ni x \mapsto (e^x + e^{-x})^{-1}(e^x - e^{-x}) \in \mathbb{R}, \tag{172}$$

and we employ, in Sect. 5.5 below, multidimensional versions of the ReLU function

$$\mathbb{R} \ni x \mapsto \max\{x, 0\} \in \mathbb{R}. \tag{173}$$

In addition, in Sects. 5.1, 5.2, and 5.4 we use the square function and $\mathbb{R} \ni x \mapsto x^2 \in \mathbb{R}$ as activation function just in front of the output layer and in Sects. 5.3 and 5.5 we use the identity function $\mathbb{R} \ni x \mapsto x \in \mathbb{R}$ as activation function just in front of the output layer. Furthermore, we employ Xavier initialization to initialize all neural network parameters; see Glorot and Bengio [100] for details. We did not employ batch normalization in our simulations.

Each of the numerical experiments presented below was performed with the JULIA library HIGHDIMPDE.JL on a NVIDIA TITAN RTX GPU with 1350 MHz core clock and 24 GB GDDR6 memory with 7000 MHz clock rate where the underlying system consisted of an AMD EPYC 7742 64-core CPU with 2TB memory running JULIA 1.7.2 on Ubuntu 20.04.3.

The algorithms proposed in Framework 3.1 and Sect. 4.1 have been implemented as a JULIA library named HIGHDIMPDE, which belongs to the SCIML software ecosystem. The library is listed in the official registry of general JULIA packages, and its source code can be accessed at https://github.com/SciML/HighDimPDE.jl. The exact source code we used for the simulations in this section is available from two sources: It is included (along with the source code of the library) in the arXiv version of this article available at https://arxiv.org/abs/2205.03672 where the source files can be downloaded by choosing "Other sources" under "Download" and then following the link "Download sources" (or using the URL https://arxiv.org/e-print/2205.03672). Additionally, the specific JULIA source code used for the simulations in this section can be found in a dedicated Github repository located at https://github.com/vboussange/HighDimPDE_examples.

Framework 5.1 *Assume Framework 3.1, assume* $\mathfrak{d} = (d+50)(d+1) + (d+50)(d+51) + (d+51)$*, let* $\varepsilon, \beta_1, \beta_2 \in \mathbb{R}$*,* $(\gamma_m)_{m\in\mathbb{N}} \subseteq (0,\infty)$ *satisfy* $\varepsilon = 10^{-8}$*,* $\beta_1 = \frac{9}{10}$*, and* $\beta_2 = \frac{999}{1000}$*, let* $g\colon D \to \mathbb{R}$*,* $\mu\colon D \to \mathbb{R}^d$*, and* $\sigma\colon D \to \mathbb{R}^{d\times d}$ *be continuous, let* $u = (u(t,x))_{(t,x)\in[0,T]\times D} \in C^{1,2}([0,T]\times D, \mathbb{R})$ *have at most polynomially growing partial derivatives, assume for every* $t \in (0,T]$*,* $x \in \partial_D$ *that* $\langle \mathbf{n}(x), (\nabla_x u)(t,x)\rangle = 0$*, assume for every* $t \in [0,T]$*,* $x \in D$*,* $j \in \mathbb{N}$*,* $\mathbf{s} \in \mathbb{R}^{\varsigma}$ *that* $u(0,x) = g(x) = \mathbb{V}_0^{j,\mathbf{s}}(\theta,x)$*,* $\int_D \bigl| f\bigl(t,x,\mathbf{x},u(t,x),u(t,\mathbf{x})\bigr)\bigr| \,\nu_x(\mathrm{d}\mathbf{x}) < \infty$*, and*

$$\bigl(\tfrac{\partial}{\partial t}u\bigr)(t,x) = \tfrac{1}{2}\operatorname{Trace}\bigl(\sigma(x)[\sigma(x)]^*(\operatorname{Hess}_x u)(t,x)\bigr) + \bigl\langle \mu(x), (\nabla_x u)(t,x)\bigr\rangle + \int_D f\bigl(t,x,\mathbf{x},u(t,x),u(t,\mathbf{x})\bigr)\,\nu_x(\mathrm{d}\mathbf{x}), \tag{174}$$

assume for every $m \in \mathbb{N}$*,* $i \in \{0,1,\ldots,N\}$ *that* $J_m = 8000$*,* $t_i = \frac{iT}{N}$*, and* $\varrho = 2\mathfrak{d}$*, and assume for every* $n \in \{1,2,\ldots,N\}$*,* $m \in \mathbb{N}$*,* $x = (x_1,\ldots,x_{\mathfrak{d}})$*,* $y = (y_1,\ldots,y_{\mathfrak{d}})$*,* $\eta = (\eta_1,\ldots,\eta_{\mathfrak{d}}) \in \mathbb{R}^{\mathfrak{d}}$ *that*

$$\Xi_0^n(x,y,\eta) = 0, \quad \Psi_m^n(x,y,\eta) = \bigl(\beta_1 x + (1-\beta_1)\eta, \beta_2 y + (1-\beta_2)((\eta_1)^2,\ldots,(\eta_{\mathfrak{d}})^2)\bigr), \tag{175}$$

and

$$\psi_m^n(x,y) = \left(\left[\sqrt{\tfrac{|y_1|}{1-(\beta_2)^m}} + \varepsilon\right]^{-1} \frac{\gamma_m x_1}{1-(\beta_1)^m}, \ldots, \left[\sqrt{\tfrac{|y_{\mathfrak{d}}|}{1-(\beta_2)^m}} + \varepsilon\right]^{-1} \frac{\gamma_m x_{\mathfrak{d}}}{1-(\beta_1)^m}\right). \tag{176}$$

In Sects. 5.1, 5.2, 5.3, 5.4, and 5.5 below, we use the machine learning-based approximation method in Framework 5.1 to approximately calculate the solutions to a variety of PDEs with a non-local term and/or Neumann boundary conditions in dimension $d \in \{1,2,5,10\}$ and with time horizon $T \in \{1/5, 1/2, 1\}$. We evaluate these approximations by comparing them to a reference solution. Specifically, in Sects. 5.1, 5.2, 5.3, and 5.5 we use the multilevel Picard approximation method presented in Framework 4.1 to compute an approximation for $u(T,(0,\ldots,0))$ which we use as the reference solution to approximate the unknown exact solution $u(T,(0,\ldots,0))$. For the PDEs treated in Sect. 5.4 we can compute the exact solution directly and use this exact solution as the reference solution. In all cases, we run the simulation 5 times, for each value of d and for each value of T, thus obtaining 5 independent realizations of the random variable $\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0, \ldots, 0))$. We use these to approximately compute and report, in Tables 1, 2, 3, 4, and 7 below, the mean

$$\mathbb{E}\big[\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0, \ldots, 0))\big], \tag{177}$$

the standard deviation

$$\Big(\mathbb{E}\big[|\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0, \ldots, 0)) - \mathbb{E}[\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0, \ldots, 0))]|^2\big]\Big)^{1/2}, \tag{178}$$

the relative L^1-approximation error

$$\mathbb{E}\bigg[\tfrac{|\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0,\ldots,0)) - u(T,(0,\ldots,0))|}{|u(T,(0,\ldots,0))|}\bigg], \tag{179}$$

the standard deviation

$$\Bigg(\mathbb{E}\bigg[\bigg|\tfrac{|\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0,\ldots,0)) - u(T,(0,\ldots,0))|}{|u(T,(0,\ldots,0))|} - \mathbb{E}\bigg[\tfrac{|\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0,\ldots,0)) - u(T,(0,\ldots,0))|}{|u(T,(0,\ldots,0))|}\bigg]\bigg|^2\bigg]\Bigg)^{1/2} \tag{180}$$

of the relative L^1-approximation error, as well as the average of the runtimes of the 5 independent runs. This runtime is measured from the start of the training of the first neural network until the end of the training of the final neural network (note that the time required for the evaluation of the neural networks is negligible).

5.1 Fisher–KPP PDEs with Neumann boundary conditions

In this subsection we use the machine learning-based approximation method in Framework 5.1 to approximately calculate the solutions of certain Fisher–KPP PDEs with Neumann boundary conditions (cf., e.g., Bian et al. [36] and Wang et al. [40]).

Assume Framework 5.1, let $\epsilon \in (0, \infty)$ satisfy $\epsilon = \frac{1}{10}$, assume $d \in \{1, 2, 5, 10\}$, $D = [-1/2, 1/2]^d$, $T \in \{1/5, 1/2, 1\}$, $N = 10$, $K_1 = K_2 = \ldots = K_N = 1$, and $M_1 = M_2 = \ldots = M_N = 500$, assume for every $n, m, j \in \mathbb{N}$, $\omega \in \Omega$ that $\xi^{n,m,j}(\omega) = (0, \ldots, 0)$, assume for every $m \in \mathbb{N}$ that $\gamma_m = 10^{-2}$, and assume for every $s, t \in [0, T]$, $x, \mathbf{x} \in D$, $y, \mathbf{y} \in \mathbb{R}$, $v \in \mathbb{R}^d$ that $g(x) = \exp(-\frac{1}{4}\|x\|^2)$, $\mu(x) = (0, \ldots, 0)$, $\sigma(x)v = \epsilon v$, $f(t, x, \mathbf{x}, y, \mathbf{y}) = y(1 - y)$, and

$$H(t, s, x, v) = R(x, x + \mu(x)(t - s) + \sigma(x)v) = R(x, x + \epsilon v) \tag{181}$$

(cf. (139) and (150)). The solution $u\colon [0, T] \times D \to \mathbb{R}$ of the PDE in (174) then satisfies that for every $t \in (0, T]$, $x \in \partial_D$ it holds that $\langle \mathbf{n}(x), (\nabla_x u)(t, x)\rangle = 0$ and that for every $t \in [0, T]$, $x \in D$ it holds that $u(0, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\big(\tfrac{\partial}{\partial t}u\big)(t, x) = \tfrac{\epsilon^2}{2}(\Delta_x u)(t, x) + u(t, x)\big(1 - u(t, x)\big). \tag{182}$$

In (182) the function $u\colon [0, T] \times D \to \mathbb{R}$ models the proportion of a particular type of alleles in a biological population spatially structured over D. For every $t \in [0, T]$, $x \in \mathbb{R}^d$ the number $u(t, x) \in \mathbb{R}$ describes the proportion of individuals with a particular type of alleles located at position $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ at time $t \in [0, T]$.

For each choice of $d \in \{1, 2, 5, 10\}$ and $T \in \{1/5, 1/2, 1\}$ we approximate the unknown value $u(T, (0, \ldots, 0))$ by computing 5 independent realizations of $\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0, \ldots, 0))$. We report in Table 1 approximations of the mean and the standard deviation of this random variable as well as approximations of the relative L^1-approximation error and the uncorrected

Table 1 Numerical simulations for the approximation method in Framework 3.1 in the case of the Fisher–KPP PDEs with Neumann boundary conditions in (182) in Sect. 5.1

d	T	N	Mean of the approx. method	Standard deviation of the approx. method	Reference value	Relative L^1-approx. error	Standard deviation of the error	Average runtime in seconds
1	1/5	10	0.9995949	0.0000156	0.9995965	0.0000120	0.0000082	15.818
2	1/5	10	0.9990286	0.0004035	0.9991810	0.0001973	0.0003788	16.333
5	1/5	10	0.9979573	0.0000567	0.9979502	0.0000423	0.0000325	16.085
10	1/5	10	0.9958883	0.0000236	0.9959512	0.0000632	0.0000237	16.232
1	1/2	10	0.9992189	0.0000115	0.9992664	0.0000475	0.0000115	16.120
2	1/2	10	0.9985348	0.0000603	0.9984853	0.0000637	0.0000408	16.393
5	1/2	10	0.9962728	0.0000687	0.9962840	0.0000517	0.0000395	16.324
10	1/2	10	0.9925187	0.0001019	0.9924206	0.0000988	0.0001027	16.336
1	1	10	0.9991205	0.0000484	0.9989034	0.0002174	0.0000485	16.221
2	1	10	0.9982253	0.0000424	0.9980188	0.0002069	0.0000425	16.822
5	1	10	0.9956714	0.0000491	0.9956116	0.0000600	0.0000493	16.533
10	1	10	0.9912150	0.0001519	0.9908484	0.0003700	0.0001533	16.520

sample standard deviation of the approximation error (see the discussion below Framework 5.1 for more details). For the latter two statistics, the reference value which is used as an approximation for the unknown value $u(T, (0, \dots, 0))$ of the exact solution of the PDE in (182) is computed via the MLP approximation method for non-local nonlinear PDEs in Framework 4.1 as described in Example 4.2 (cf. Beck et al. [101, Remark 3.3]). More precisely, we apply Example 4.2 with $d \in \{1, 2, 5, 10\}$, $T \in \{1/5, 1/2, 1\}$, $D = [-1/2, 1/2]^d$, $\epsilon = \frac{1}{10}$ in the notation of Example 4.2 and we use the mean of 5 independent realizations of $U^{(0)}_{5,5,0,\infty}(0, (0, \dots, 0))$ as the reference value.

5.2 Non-local competition PDEs

In this subsection we use the machine learning-based approximation method in Framework 5.1 to approximately calculate the solutions of certain non-local competition PDEs (cf., e.g., Doebeli and Ispolatov [47], Berestycki et al. [38], Perthame and Génieys [37], and Génieys et al. [42]).

Assume Framework 5.1, let $\mathfrak{s}, \epsilon \in (0, \infty)$ satisfy $\mathfrak{s} = \epsilon = \frac{1}{10}$, assume $d \in \{1, 2, 5, 10\}$, $D = \mathbb{R}^d$, $T \in \{1/5, 1/2, 1\}$, $N = 10$, $K_1 = K_2 = \cdots = K_N = 1$, and $M_1 = M_2 = \cdots = M_N = 500$, assume for every $n, m, j \in \mathbb{N}$, $\omega \in \Omega$ that $\xi^{n,m,j}(\omega) = (0, \dots, 0)$, assume for every $m \in \mathbb{N}$ that $\gamma_m = 10^{-2}$, and assume for every $s, t \in [0, T]$, $v, x, \mathbf{x} \in \mathbb{R}^d$, $y, \mathbf{y} \in \mathbb{R}$, $A \in \mathcal{B}(\mathbb{R}^d)$ that $\nu_x(A) = \pi^{-d/2}\mathfrak{s}^{-d} \int_A \exp\big(-\mathfrak{s}^{-2}\|x - \mathbf{x}\|^2\big)\, d\mathbf{x}$, $g(x) = \exp(-\frac{1}{4}\|x\|^2)$, $\mu(x) = (0, \dots, 0)$, $\sigma(x)v = \epsilon v$, $f(t, x, \mathbf{x}, y, \mathbf{y}) = y(1 - \mathbf{y}\mathfrak{s}^d \pi^{d/2})$, and

$$H(t, s, x, v) = x + \mu(x)(t - s) + \sigma(x)v = x + \epsilon v \tag{183}$$

(cf. (139) and (150)). The solution $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (174) then satisfies that for every $t \in [0, T]$, $x \in \mathbb{R}^d$ it holds that $u(0, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\big(\tfrac{\partial}{\partial t}u\big)(t, x) = \tfrac{\epsilon^2}{2}(\Delta_x u)(t, x) + u(t, x)\bigg(1 - \int_{\mathbb{R}^d} u(t, \mathbf{x}) \exp\big(-\tfrac{\|x - \mathbf{x}\|^2}{\mathfrak{s}^2}\big)\, d\mathbf{x}\bigg). \tag{184}$$

Table 2 Numerical simulations for the approximation method in Framework 3.1 in the case of the non-local competition PDEs in (184) in Sect. 5.2

d	T	N	Mean of the approx. method	Standard deviation of the approx. method	Reference value	Relative L^1-approx. error	Standard deviation of the error	Average runtime in seconds
1	1/5	10	1.1736135	0.0000097	1.1750579	0.0012292	0.0000083	13.333
2	1/5	10	1.2096596	0.0000269	1.2118434	0.0018021	0.0000222	13.740
5	1/5	10	1.2159404	0.0000304	1.2180500	0.0017319	0.0000250	13.390
10	1/5	10	1.2129205	0.0000767	1.2152996	0.0019576	0.0000631	13.363
1	1/2	10	1.4695143	0.0000762	1.4782081	0.0058813	0.0000515	13.271
2	1/2	10	1.5948946	0.0000863	1.6153017	0.0126336	0.0000535	13.741
5	1/2	10	1.6186081	0.0003448	1.6354169	0.0102780	0.0002109	13.679
10	1/2	10	1.6088821	0.0000915	1.6248389	0.0098205	0.0000563	13.645
1	1	10	2.0494258	0.0000557	2.0684438	0.0091944	0.0000269	13.464
2	1	10	2.4684410	0.0006871	2.5772161	0.0422064	0.0002666	14.100
5	1	10	2.5609298	0.0005965	2.6627727	0.0382469	0.0002240	13.705
10	1	10	2.5310452	0.0011283	2.6514871	0.0454243	0.0004255	13.585

In (184) the function $u\colon [0,T]\times\mathbb{R}^d \to \mathbb{R}$ models the evolution of a population characterized by a set of d biological traits under the combined effects of selection, competition, and mutation. For every $t \in [0,T]$, $x \in \mathbb{R}^d$ the number $u(t,x) \in \mathbb{R}$ describes the number of individuals with traits $x = (x_1,\dots,x_d) \in \mathbb{R}^d$ at time $t \in [0,T]$.

For each choice of $d \in \{1,2,5,10\}$ and $T \in \{1/5, 1/2, 1\}$ we approximate the unknown value $u(T,(0,\dots,0))$ by computing 5 independent realizations of $\mathbb{V}_N^{1,0}(\Theta^N_{M_N},(0,\dots,0))$. We report in Table 2 approximations of the mean and the standard deviation of this random variable as well as approximations of the relative L^1-approximation error and the uncorrected sample standard deviation of the approximation error (see the discussion below Framework 5.1 for more details). For the latter two statistics, the reference value which is used as an approximation for the unknown value $u(T,(0,\dots,0))$ of the exact solution of the PDE in (184) is computed via the MLP approximation method for non-local nonlinear PDEs in Framework 4.1 as described in Example 4.3 (cf. Beck et al. [101, Remark 3.3]). More precisely, we apply Example 4.3 with $d \in \{1,2,5,10\}$, $T \in \{1/5, 1/2, 1\}$, $D = \mathbb{R}^d$, $\mathfrak{s} = \epsilon = \frac{1}{10}$ in the notation of Example 4.3 and we use the mean of 5 independent realizations of $U^{(0)}_{5,5,0,\infty}(0,(0,\dots,0))$ as the reference value.

5.3 Non-local sine-Gordon type PDEs

In this subsection we use the machine learning-based approximation method in Framework 5.1 to approximately calculate the solutions of non-local sine-Gordon type PDEs (cf., e.g., Hairer and Shen [9], Barone et al. [6], and Coleman [8]).

Assume Framework 5.1, let $\mathfrak{s}, \epsilon \in (0,\infty)$ satisfy $\mathfrak{s} = \epsilon = \frac{1}{10}$, assume $d \in \{1,2,5,10\}$, $D = \mathbb{R}^d$, $T \in \{1/5, 1/2, 1\}$, $N = 10$, $K_1 = K_2 = \dots = K_N = 1$, and $M_1 = M_2 = \dots = M_N = 500$, assume for every $n,m,j \in \mathbb{N}$, $\omega \in \Omega$ that $\xi^{n,m,j}(\omega) = (0,\dots,0)$, assume for every $m \in \mathbb{N}$ that $\gamma_m = 10^{-3}$, and assume for every $s,t \in [0,T]$, $v,x,\mathbf{x} \in \mathbb{R}^d$, $y,\mathbf{y} \in \mathbb{R}$, $A \in \mathcal{B}(\mathbb{R}^d)$ that $\nu_x(A) = \pi^{-d/2}\mathfrak{s}^{-d}\int_A \exp\big(-\mathfrak{s}^{-2}\|x-\mathbf{x}\|^2\big)\,\mathrm{d}\mathbf{x}$, $g(x) = \exp(-\frac{1}{4}\|x\|^2)$,

Table 3 Numerical simulations for the approximation method in Framework 3.1 in the case of the non-local sine-Gordon PDEs in (186) in Sect. 5.3

d	T	N	Mean of the approx. method	Standard deviation of the approx. method	Reference value	Relative L^1-approx. error	Standard deviation of the error	Average runtime in seconds
1	1/5	10	1.1362996	0.0000110	1.1367379	0.0003856	0.0000097	12.377
2	1/5	10	1.1678523	0.0000105	1.1685355	0.0005847	0.0000090	12.807
5	1/5	10	1.1731904	0.0000270	1.1739435	0.0006415	0.0000230	12.642
10	1/5	10	1.1704623	0.0000237	1.1717243	0.0010771	0.0000202	12.487
1	1/2	10	1.3513988	0.0000170	1.3537606	0.0017446	0.0000126	12.515
2	1/2	10	1.4393396	0.0000260	1.4421450	0.0019453	0.0000181	12.885
5	1/2	10	1.4546689	0.0000184	1.4588427	0.0028610	0.0000126	12.404
10	1/2	10	1.4473806	0.0000648	1.4535572	0.0042493	0.0000446	12.348
1	1	10	1.7114100	0.0000522	1.7128565	0.0008445	0.0000304	12.296
2	1	10	1.9019937	0.0000758	1.9079758	0.0031353	0.0000397	12.957
5	1	10	1.9364048	0.0000398	1.9437583	0.0037831	0.0000205	12.572
10	1	10	1.9222919	0.0000647	1.9265107	0.0021899	0.0000336	12.461

$\mu(x) = (0, \ldots, 0)$, $\sigma(x)v = \epsilon v$, $f(t, x, \mathbf{x}, y, \mathbf{y}) = \sin(y) - \mathbf{y}\pi^{d/2}\mathfrak{s}^d$, and

$$H(t, s, x, v) = x + \mu(x)(t - s) + \sigma(x)v = x + \epsilon v \tag{185}$$

(cf. (139) and (150)). The solution $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (174) then satisfies that for every $t \in [0, T]$, $x \in \mathbb{R}^d$ it holds that $u(0, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\big(\tfrac{\partial}{\partial t}u\big)(t, x) = \tfrac{\epsilon^2}{2}(\Delta_x u)(t, x) + \sin(u(t, x)) - \int_{\mathbb{R}^d} u(t, \mathbf{x}) \exp\big(-\tfrac{\|x-\mathbf{x}\|^2}{\mathfrak{s}^2}\big)\, d\mathbf{x}. \tag{186}$$

For each choice of $d \in \{1, 2, 5, 10\}$ and $T \in \{1/5, 1/2, 1\}$ we approximate the unknown value $u(T, (0, \ldots, 0))$ by computing 5 independent realizations of $\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0, \ldots, 0))$. We report in Table 3 approximations of the mean and the standard deviation of this random variable as well as approximations of the relative L^1-approximation error and the uncorrected sample standard deviation of the approximation error (see the discussion below Framework 5.1 for more details). For the latter two statistics, the reference value which is used as an approximation for the unknown value $u(T, (0, \ldots, 0))$ of the exact solution of the PDE in (186) is computed via the MLP approximation method for non-local nonlinear PDEs in Framework 4.1 as described in Example 4.4 (cf. Beck et al. [101, Remark 3.3]). More precisely, we apply Example 4.4 with $d \in \{1, 2, 5, 10\}$, $T \in \{1/5, 1/2, 1\}$, $D = \mathbb{R}^d$, $\mathfrak{s} = \epsilon = \frac{1}{10}$ in the notation of Example 4.4 and we use the mean of 5 independent realizations of $U^{(0)}_{5,5,-\infty,\infty}(0, (0, \ldots, 0))$ as the reference value.

5.4 Replicator-mutator PDEs

In this subsection we use both the machine learning-based approximation method in Framework 5.1 and the MLP approximation method in Framework 4.1 to approximately calculate the solutions of certain replicator-mutator PDEs describing the dynamics of a phenotype distribution under the combined effects of selection and mutation (cf., e.g., Hamel et al. [26]). The solutions to the PDEs we are trying to approximate in this subsection can be represented explicitly (cf., e.g., Lemma 5.2 below). The gives us the opportunity to compare the performance of the two approximation methods presented here and it also allows us to evaluate the capability of the machine learning-based approximation method in Framework 5.1 to produce approximations on an entire subset of the domain.

Exact solutions. The following results, Lemma 5.2 below, give an explicit representation of the exact solutions of certain replicator-mutator PDEs. In (190) the function $u\colon [0,T]\times\mathbb{R}^d\to\mathbb{R}$ models the evolution of the phenotype distribution of a population composed of a set of d biological traits under the combined effects of selection and mutation. For every $t\in[0,T]$, $x\in\mathbb{R}^d$ the number $u(t,x)\in\mathbb{R}$ describes the number of individuals with traits $x=(x_1,\dots,x_d)\in\mathbb{R}^d$ at time $t\in[0,T]$. The function a models a quadratic fitness function.

Lemma 5.2 *Let $d\in\mathbb{N}$, $\mathfrak{u}_1,\mathfrak{u}_2,\dots,\mathfrak{u}_d\in\mathbb{R}$, $\mathfrak{m}_1,\mathfrak{m}_2,\dots,\mathfrak{m}_d,\mathfrak{s}_1,\mathfrak{s}_2,\dots,\mathfrak{s}_d\in(0,\infty)$, let $a\colon\mathbb{R}^d\to\mathbb{R}$ satisfy for every $x\in\mathbb{R}^d$ that $a(x)=-\frac{1}{2}\|x\|^2$, for every $i\in\{1,2,\dots,d\}$ let $\mathfrak{S}_i\colon[0,\infty)\to(0,\infty)$ and $\mathfrak{U}_i\colon[0,\infty)\to\mathbb{R}$ satisfy for every $t\in[0,\infty)$ that*

$$\mathfrak{S}_i(t)=\mathfrak{m}_i\left[\frac{\mathfrak{m}_i\sinh(\mathfrak{m}_i t)+\mathfrak{s}_i\cosh(\mathfrak{m}_i t)}{\mathfrak{m}_i\cosh(\mathfrak{m}_i t)+\mathfrak{s}_i\sinh(\mathfrak{m}_i t)}\right]$$
$$\text{and}\quad \mathfrak{U}_i(t)=\frac{\mathfrak{m}_i\mathfrak{u}_i}{\mathfrak{m}_i\cosh(\mathfrak{m}_i t)+\mathfrak{s}_i\sinh(\mathfrak{m}_i t)}, \tag{187}$$

and let $u\colon[0,\infty)\times\mathbb{R}^d\to\mathbb{R}$ satisfy for every $t\in[0,\infty)$, $x=(x_1,\dots,x_d)\in\mathbb{R}^d$ that

$$u(t,x)=(2\pi)^{-d/2}\left[\prod_{i=1}^{d}|\mathfrak{S}_i(t)|^{-1/2}\right]\exp\left(-\sum_{i=1}^{d}\frac{(x_i-\mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right). \tag{188}$$

Then

(i) *it holds that $u\in C^{1,2}([0,\infty)\times\mathbb{R}^d,\mathbb{R})$,*
(ii) *it holds for every $x=(x_1,\dots,x_d)\in\mathbb{R}^d$ that*

$$u(0,x)=(2\pi)^{-d/2}\left[\prod_{i=1}^{d}|\mathfrak{s}_i|^{-1/2}\right]\exp\left(-\sum_{i=1}^{d}\frac{(x_i-\mathfrak{u}_i)^2}{2\mathfrak{s}_i}\right), \tag{189}$$

and

(iii) *it holds for every $t\in[0,\infty)$, $x=(x_1,\dots,x_d)\in\mathbb{R}^d$ that*

$$\left(\tfrac{\partial}{\partial t}u\right)(t,x)=u(t,x)\left(a(x)-\int_{\mathbb{R}^d}u(t,\mathbf{x})\,a(\mathbf{x})\,\mathrm{d}\mathbf{x}\right)+\sum_{i=1}^{d}\tfrac{1}{2}|\mathfrak{m}_i|^2\left(\tfrac{\partial^2}{\partial x_i^2}u\right)(t,x). \tag{190}$$

Proof of Lemma 5.2 First, note that the fact that for every $i\in\{1,2,\dots,d\}$ it holds that $\mathfrak{S}_i\in C^\infty([0,\infty),(0,\infty))$, the fact that for every $i\in\{1,2,\dots,d\}$ it holds that $\mathfrak{U}_i\in C^\infty([0,\infty),\mathbb{R})$, and (188) establish item (i). Moreover, observe that the fact that for every $i\in\{1,2,\dots,d\}$ it holds that $\mathfrak{S}_i(0)=\mathfrak{s}_i$, the fact that for every $i\in\{1,2,\dots,d\}$ it holds that $\mathfrak{U}_i(0)=\mathfrak{u}_i$, and (187) prove item (ii). Next note that (188) ensures that for every $t\in[0,\infty)$, $x=(x_1,\dots,x_d)\in\mathbb{R}^d$ it holds that

$$u(t,x)=\prod_{i=1}^{d}\left[(2\pi\mathfrak{S}_i(t))^{-1/2}\exp\left(-\frac{(x_i-\mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right]. \tag{191}$$

The product rule hence implies that for every $t \in [0, \infty)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned}
&\big(\tfrac{\partial}{\partial t} u\big)(t, x) \\
&= \frac{\partial}{\partial t}\left(\prod_{i=1}^{d}\left[(2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{(x_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right]\right) \\
&= \sum_{i=1}^{d}\left[\left[\prod\nolimits_{j\in\{1,\dots,d\}\setminus\{i\}}\left((2\pi \mathfrak{S}_j(t))^{-1/2} \exp\left(-\frac{(x_j - \mathfrak{U}_j(t))^2}{2\mathfrak{S}_j(t)}\right)\right)\right]\right. \\
&\qquad \left.\cdot\left[\frac{\partial}{\partial t}\left((2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{(x_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right)\right]\right].
\end{aligned} \tag{192}$$

The chain rule, the product rule, and (191) therefore show that for every $t \in [0, \infty)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned}
&\big(\tfrac{\partial}{\partial t} u\big)(t, x) \\
&= \sum_{i=1}^{d}\left[\left[\prod\nolimits_{j\in\{1,\dots,d\}\setminus\{i\}}\left((2\pi \mathfrak{S}_j(t))^{-1/2} \exp\left(-\frac{(x_j - \mathfrak{U}_j(t))^2}{2\mathfrak{S}_j(t)}\right)\right)\right]\right. \\
&\quad \cdot\left[\left(\frac{\partial}{\partial t}\left((2\pi \mathfrak{S}_i(t))^{-1/2}\right)\right) \exp\left(-\frac{(x_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right. \\
&\quad \left.\left. + (2\pi \mathfrak{S}_i(t))^{-1/2}\left(\frac{\partial}{\partial t}\left(-\frac{(x_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right) \exp\left(-\frac{(x_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right]\right] \\
&= \sum_{i=1}^{d}\left[\left[\prod\nolimits_{j\in\{1,\dots,d\}\setminus\{i\}}\left((2\pi \mathfrak{S}_j(t))^{-1/2} \exp\left(-\frac{(x_j - \mathfrak{U}_j(t))^2}{2\mathfrak{S}_j(t)}\right)\right)\right]\right. \\
&\quad \cdot\left[-(2\pi \mathfrak{S}_i(t))^{-1/2}\left[\frac{\big(\tfrac{\partial}{\partial t}\mathfrak{S}_i\big)(t)}{2\mathfrak{S}_i(t)}\right] \exp\left(-\frac{(x_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right. \\
&\quad +(2\pi \mathfrak{S}_i(t))^{-1/2}\left(\frac{2\big(\tfrac{\partial}{\partial t}\mathfrak{U}_i\big)(t)(x_i - \mathfrak{U}_i(t))}{2\mathfrak{S}_i(t)}\right. \\
&\quad \left.\left.\left.+\frac{(x_i - \mathfrak{U}_i(t))^2\big(\tfrac{\partial}{\partial t}\mathfrak{S}_i\big)(t)}{2|\mathfrak{S}_i(t)|^2}\right) \exp\left(-\frac{(x_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right]\right] \\
&= u(t, x)\left[\sum_{i=1}^{d}\left(\frac{-\big(\tfrac{\partial}{\partial t}\mathfrak{S}_i\big)(t)}{2\mathfrak{S}_i(t)} + \frac{2\mathfrak{S}_i(t)\big(\tfrac{\partial}{\partial t}\mathfrak{U}_i\big)(t)(x_i - \mathfrak{U}_i(t)) + (x_i - \mathfrak{U}_i(t))^2\big(\tfrac{\partial}{\partial t}\mathfrak{S}_i\big)(t)}{2|\mathfrak{S}_i(t)|^2}\right)\right].
\end{aligned} \tag{193}$$

Moreover, observe that (187), the chain rule, and the product rule ensure that for every $i \in \{1, \dots, d\}$, $t \in [0, \infty)$ it holds that

$$\begin{aligned}
\big(\tfrac{\partial}{\partial t}\mathfrak{U}_i\big)(t) &= \frac{\partial}{\partial t}\left(\frac{\mathfrak{m}_i \mathfrak{u}_i}{\mathfrak{m}_i \cosh(\mathfrak{m}_i t) + \mathfrak{s}_i \sinh(\mathfrak{m}_i t)}\right) \\
&= -|\mathfrak{m}_i|^2 \mathfrak{u}_i\left[\frac{\mathfrak{m}_i \sinh(\mathfrak{m}_i t) + \mathfrak{s}_i \cosh(\mathfrak{m}_i t)}{[\mathfrak{m}_i \cosh(\mathfrak{m}_i t) + \mathfrak{s}_i \sinh(\mathfrak{m}_i t)]^2}\right] \\
&= -\mathfrak{S}_i(t)\mathfrak{U}_i(t)
\end{aligned} \tag{194}$$

and

$$\begin{aligned}
&\big(\tfrac{\partial}{\partial t}\mathfrak{S}_i\big)(t)\\
&= \frac{\partial}{\partial t}\left(\mathfrak{m}_i\left[\frac{\mathfrak{m}_i\sinh(\mathfrak{m}_i t)+\mathfrak{s}_i\cosh(\mathfrak{m}_i t)}{\mathfrak{m}_i\cosh(\mathfrak{m}_i t)+\mathfrak{s}_i\sinh(\mathfrak{m}_i t)}\right]\right)\\
&= |\mathfrak{m}_i|^2\left[\frac{\mathfrak{m}_i\cosh(\mathfrak{m}_i t)+\mathfrak{s}_i\sinh(\mathfrak{m}_i t)}{\mathfrak{m}_i\cosh(\mathfrak{m}_i t)+\mathfrak{s}_i\sinh(\mathfrak{m}_i t)}\right]-|\mathfrak{m}_i|^2\left[\frac{\mathfrak{m}_i\sinh(\mathfrak{m}_i t)+\mathfrak{s}_i\cosh(\mathfrak{m}_i t)}{\mathfrak{m}_i\cosh(\mathfrak{m}_i t)+\mathfrak{s}_i\sinh(\mathfrak{m}_i t)}\right]^2\\
&= |\mathfrak{m}_i|^2-|\mathfrak{S}_i(t)|^2.
\end{aligned} \tag{195}$$

Combining this with (193) implies that for every $i \in \{1,2,\dots,d\}$, $t \in [0,\infty)$ it holds that

$$\begin{aligned}
\big(\tfrac{\partial}{\partial t}u\big)(t,x) &= \frac{u(t,x)}{2}\sum_{i=1}^{d}\left[\frac{-\big[|\mathfrak{m}_i|^2-|\mathfrak{S}_i(t)|^2\big]}{\mathfrak{S}_i(t)}\right.\\
&\quad\left.+\frac{2|\mathfrak{S}_i(t)|^2\mathfrak{U}_i(t)(\mathfrak{U}_i(t)-x_i)+(x_i-\mathfrak{U}_i(t))^2(|\mathfrak{m}_i|^2-|\mathfrak{S}_i(t)|^2)}{|\mathfrak{S}_i(t)|^2}\right]\\
&= \frac{u(t,x)}{2}\sum_{i=1}^{d}\Bigg[|\mathfrak{m}_i|^2\left(\left(\frac{x_i-\mathfrak{U}_i(t)}{\mathfrak{S}_i(t)}\right)^2-\frac{1}{\mathfrak{S}_i(t)}\right)\\
&\quad+\mathfrak{S}_i(t)+2\big(|\mathfrak{U}_i(t)|^2-\mathfrak{U}_i(t)\,x_i\big)-\big(|x_i|^2-2\mathfrak{U}_i(t)\,x_i+|\mathfrak{U}_i(t)|^2\big)\Bigg]\\
&= \frac{u(t,x)}{2}\sum_{i=1}^{d}\left[|\mathfrak{m}_i|^2\left(\left(\frac{x_i-\mathfrak{U}_i(t)}{\mathfrak{S}_i(t)}\right)^2-\frac{1}{\mathfrak{S}_i(t)}\right)+\mathfrak{S}_i(t)+|\mathfrak{U}_i(t)|^2-|x_i|^2\right].
\end{aligned} \tag{196}$$

Furthermore, note that (191) and the product rule show that for every $i \in \{1,2,\dots,d\}$, $t \in [0,\infty)$, $x=(x_1,\dots,x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned}
\big(\tfrac{\partial}{\partial x_i}u\big)(t,x) &= \frac{\partial}{\partial x_i}\left[\prod_{j=1}^{d}\left[(2\pi\,\mathfrak{S}_j(t))^{-1/2}\exp\left(-\frac{(x_j-\mathfrak{U}_j(t))^2}{2\mathfrak{S}_j(t)}\right)\right]\right]\\
&= \left[\frac{\partial}{\partial x_i}\left[(2\pi\,\mathfrak{S}_i(t))^{-1/2}\exp\left(-\frac{(x_i-\mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)}\right)\right]\right]\\
&\quad\cdot\prod_{j\in\{1,2,\dots,d\}\setminus\{i\}}\left[(2\pi\,\mathfrak{S}_j(t))^{-1/2}\exp\left(-\frac{(x_j-\mathfrak{U}_j(t))^2}{2\mathfrak{S}_j(t)}\right)\right]\\
&= -u(t,x)\left(\frac{x_i-\mathfrak{U}_i(t)}{\mathfrak{S}_i(t)}\right)=u(t,x)\left(\frac{\mathfrak{U}_i(t)-x_i}{\mathfrak{S}_i(t)}\right).
\end{aligned} \tag{197}$$

The product rule therefore assures that for every $i \in \{1,2,\dots,d\}$, $t \in [0,\infty)$, $x=(x_1,\dots,x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned}
&\big(\tfrac{\partial^2}{\partial x_i^2}u\big)(t,x) = \frac{\partial}{\partial x_i}\left(u(t,x)\left(\frac{\mathfrak{U}_i(t)-x_i}{\mathfrak{S}_i(t)}\right)\right)\\
&= \big(\tfrac{\partial}{\partial x_i}u\big)(t,x)\left(\frac{\mathfrak{U}_i(t)-x_i}{\mathfrak{S}_i(t)}\right)-\frac{u(t,x)}{\mathfrak{S}_i(t)}=u(t,x)\left[\left(\frac{x_i-\mathfrak{U}_i(t)}{\mathfrak{S}_i(t)}\right)^2-\frac{1}{\mathfrak{S}_i(t)}\right].
\end{aligned} \tag{198}$$

Hence, we obtain that for every $t \in [0, \infty)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\sum_{i=1}^{d} \left[\tfrac{1}{2} |\mathfrak{m}_i|^2 \big(\tfrac{\partial^2}{\partial x_i^2} u \big)(t, x) \right] = \frac{u(t, x)}{2} \sum_{i=1}^{d} \left[|\mathfrak{m}_i|^2 \left(\left(\frac{x_i - \mathfrak{U}_i(t)}{\mathfrak{S}_i(t)} \right)^2 - \frac{1}{\mathfrak{S}_i(t)} \right) \right]. \tag{199}$$

Next observe that (191) and Fubini's theorem ensure that for every $t \in [0, \infty)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned}
& u(t, x) \left(a(x) - \int_{\mathbb{R}^d} u(t, \mathbf{x})\, a(\mathbf{x})\, \mathrm{d}\mathbf{x} \right) \\
&= u(t, x) \left(-\frac{1}{2} \left[\sum_{i=1}^{d} |x_i|^2 \right] - \int_{\mathbb{R}^d} -\frac{1}{2} \left[\sum_{i=1}^{d} |\mathbf{x}_i|^2 \right] u(t, \mathbf{x})\, \mathrm{d}\mathbf{x} \right) \\
&= \frac{u(t, x)}{2} \left(- \left[\sum_{i=1}^{d} |x_i|^2 \right] \right. \\
&\quad + \sum_{i=1}^{d} \left[\int_{\mathbb{R}} |\mathbf{x}_i|^2 \, (2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{(\mathbf{x}_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)} \right) \mathrm{d}\mathbf{x}_i \right. \\
&\quad \left. \left. \cdot \left(\prod_{j \in \{1, 2, \dots, d\} \setminus \{i\}} \int_{\mathbb{R}} (2\pi \mathfrak{S}_j(t))^{-1/2} \exp\left(-\frac{(\mathbf{x}_j - \mathfrak{U}_j(t))^2}{2\mathfrak{S}_j(t)} \right) \mathrm{d}\mathbf{x}_j \right) \right] \right).
\end{aligned} \tag{200}$$

This and the fact that for every $i \in \{1, 2, \dots, d\}$, $t \in [0, \infty)$ it holds that

$$\int_{\mathbb{R}} (2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{(x - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)} \right) \mathrm{d}x = 1 \tag{201}$$

imply that for every $t \in [0, \infty)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned}
& u(t, x) \left(a(x) - \int_{\mathbb{R}^d} u(t, \mathbf{x})\, a(\mathbf{x})\, \mathrm{d}\mathbf{x} \right) \\
&= \frac{u(t, x)}{2} \sum_{i=1}^{d} \left[-|x_i|^2 + \int_{\mathbb{R}} |\mathbf{x}_i|^2 \, (2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{(\mathbf{x}_i - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)} \right) \mathrm{d}\mathbf{x}_i \right].
\end{aligned} \tag{202}$$

Next observe that the integral transformation theorem demonstrates that for every $i \in \{1, 2, \dots, d\}$, $t \in [0, \infty)$ it holds that

$$\begin{aligned}
& \int_{\mathbb{R}} x^2 \left[(2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{(x - \mathfrak{U}_i(t))^2}{2\mathfrak{S}_i(t)} \right) \right] \mathrm{d}x \\
&= \int_{\mathbb{R}} (x + \mathfrak{U}_i(t))^2 \left[(2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{x^2}{2\mathfrak{S}_i(t)} \right) \right] \mathrm{d}x \\
&= \int_{\mathbb{R}} x^2 \left[(2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{x^2}{2\mathfrak{S}_i(t)} \right) \right] \mathrm{d}x \\
&\quad + \int_{\mathbb{R}} |\mathfrak{U}_i(t)|^2 \left[(2\pi \mathfrak{S}_i(t))^{-1/2} \exp\left(-\frac{x^2}{2\mathfrak{S}_i(t)} \right) \right] \mathrm{d}x \\
&= \mathfrak{S}_i(t) + |\mathfrak{U}_i(t)|^2.
\end{aligned} \tag{203}$$

Combining this with (202) ensures that for every $t \in [0, \infty)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$u(t, x)\left(a(x) - \int_{\mathbb{R}^d} u(t, \mathbf{x})\, a(\mathbf{x})\, d\mathbf{x}\right) = \frac{u(t, x)}{2} \sum_{i=1}^{d} \left(\mathfrak{S}_i(t) + |\mathfrak{U}_i(t)|^2 - |x_i|^2\right). \tag{204}$$

This and (199) demonstrate that for every $t \in [0, \infty)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$\begin{aligned} &u(t, x)\left(a(x) - \int_{D} u(t, \mathbf{x})\, a(\mathbf{x})\, d\mathbf{x}\right) + \sum_{i=1}^{d} \tfrac{1}{2} |\mathfrak{m}_i|^2 \big(\tfrac{\partial^2}{\partial x_i^2} u\big)(t, x) \\ &= \frac{u(t, x)}{2} \sum_{i=1}^{d} \left[|\mathfrak{m}_i|^2 \left(\left(\frac{x_i - \mathfrak{U}_i(t)}{\mathfrak{S}_i(t)} \right)^2 - \frac{1}{\mathfrak{S}_i(t)} \right) + \mathfrak{S}_i(t) + |\mathfrak{U}_i(t)|^2 - |x_i|^2 \right]. \end{aligned} \tag{205}$$

Combining this with (196) proves item (iii). The proof of Lemma 5.2 is thus complete. □

Machine-learning based approximations. Assume Framework 5.1, let $\mathcal{D} \subseteq \mathbb{R}^d$, $\mathfrak{m}_1, \mathfrak{m}_2, \dots, \mathfrak{m}_d, \mathfrak{s}_1, \mathfrak{s}_2, \dots, \mathfrak{s}_d, \mathfrak{u}_1, \mathfrak{u}_2, \dots, \mathfrak{u}_d, \mathfrak{t} \in \mathbb{R}$ satisfy for every $k \in \{1, 2, \dots, d\}$ that $\mathfrak{m}_k = \frac{1}{10}$, $\mathfrak{s}_k = \frac{1}{20}$, $\mathfrak{u}_k = 0$, and $\mathfrak{t} = \frac{1}{50}$, assume $d \in \{1, 2, 5, 10\}$, $D = \mathbb{R}^d$, $T \in \{1/10, 1/5, 1/2\}$, $N = 10$, $K_1 = K_2 = \dots = K_N = 1$, let $a \in C(\mathbb{R}^d, \mathbb{R})$, $\delta \in C(\mathbb{R}^d, (0, \infty))$ satisfy for every $x \in \mathbb{R}^d$ that $a(x) = -\frac{1}{2}\|x\|^2$, and assume for every $s, t \in [0, T]$, $v = (v_1, \dots, v_d)$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, $\mathbf{x} \in \mathbb{R}^d$, $y, \mathbf{y} \in \mathbb{R}$, $A \in \mathcal{B}(\mathbb{R}^d)$ that $\nu_x(A) = \int_{A \cap \mathcal{D}} \delta(\mathbf{x})\, d\mathbf{x}$, $g(x) = (2\pi)^{-d/2} \big[\prod_{i=1}^{d} |\mathfrak{s}_i|^{-1/2}\big] \exp\big(-\sum_{i=1}^{d} \frac{(x_i - \mathfrak{u}_i)^2}{2\mathfrak{s}_i}\big)$, $\mu(x) = (0, \dots, 0)$, $\sigma(x)v = (\mathfrak{m}_1 v_1, \dots, \mathfrak{m}_d v_d)$, $f(t, x, \mathbf{x}, y, \mathbf{y}) = y(a(x) - \mathbf{y} a(\mathbf{x})[\delta(\mathbf{x})]^{-1})$, and

$$H(t, s, x, v) = x + \mu(x)(t - s) + \sigma(x)v = x + (\mathfrak{m}_1 v_1, \dots, \mathfrak{m}_d v_d) \tag{206}$$

(cf. (139) and (150)). The solution $u\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (174) then satisfies that for every $t \in [0, T]$, $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ it holds that

$$u(0, x) = (2\pi)^{-d/2} \left[\prod_{i=1}^{d} |\mathfrak{s}_i|^{-1/2}\right] \exp\left(-\sum_{i=1}^{d} \frac{(x_i - \mathfrak{u}_i)^2}{2\mathfrak{s}_i}\right) \tag{207}$$

and

$$\big(\tfrac{\partial}{\partial t} u\big)(t, x) = u(t, x)\left(a(x) - \int_{\mathcal{D}} u(t, \mathbf{x})\, a(\mathbf{x})\, d\mathbf{x}\right) + \sum_{i=1}^{d} \tfrac{1}{2} |\mathfrak{m}_i|^2 \left(\tfrac{\partial^2}{\partial x_i^2} u\right)(t, x). \tag{208}$$

For Table 4, we approximate for each choice of $d \in \{1, 2, 5, 10\}$ and $T \in \{1/5, 1/2, 1\}$ the value $u(T, (0, \dots, 0))$ by computing 5 independent realizations of $\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, (0, \dots, 0))$. We report in Table 4 approximations of the mean and the standard deviation of this random variable as well as approximations of the relative L^1-approximation error and the uncorrected sample standard deviation of the approximation error (see the discussion below Framework 5.1 for more details). For the latter two statistics, the reference value $u(T, (0, \dots, 0))$ has been calculated using Lemma 5.2 above which provides an explicit representation of the exact solution of the PDE in (208).

In Table 5 we report statistics on the capability of the approximation method in Framework 3.1 to approximate the solution of the PDE in (208) on an entire subset of the domain. More specifically, for Table 5, we approximate for each choice of $d \in \{1, 2, 5, 10\}$ and $T \in \{1/10, 1/5, 1/2\}$ the function $[-1, 1]^d \ni x \mapsto u(T, x) \in \mathbb{R}$ by computing 5 independent

Table 4 Numerical simulations for the approximation method in Framework 3.1 in the case of the replicator-mutator PDEs in (208) in Sect. 5.4 where we approximate the value of $u(T, (0, \dots, 0))$, assuming for every $n, m, j \in \mathbb{N}$, $\mathbf{x} \in \mathbb{R}^d$ that $\mathcal{D} = \mathbb{R}^d$, $\xi^{n,m,j} = 0$, $\gamma_m = 10^{-3}$, $M_n = 1000$, and $\delta(\mathbf{x}) = (2\pi)^{-d/2} \mathrm{t}^{-d} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\mathrm{t}^2}\right)$

d	T	N	Mean of the approx. method	Standard deviation of the approx. method	Reference value	Relative L^1-approx. error	Standard deviation of the error	Average runtime in seconds
1	1/5	10	1.7497161	0.0009179	1.7582066	0.0048291	0.0005220	31.815
2	1/5	10	3.0576093	0.0069599	3.0912904	0.0108955	0.0022514	32.886
5	1/5	10	16.3761745	0.0078223	16.8015567	0.0253180	0.0004656	32.821
10	1/5	10	268.3223816	0.0628488	282.2923073	0.0494874	0.0002226	32.542
1	1/2	10	1.6998393	0.0029625	1.7222757	0.0130272	0.0017201	32.425
2	1/2	10	2.8902346	0.0072711	2.9662336	0.0256214	0.0024513	33.448
5	1/2	10	14.2120520	0.0376394	15.1535149	0.0621284	0.0024839	32.820
10	1/2	10	201.3921326	0.2044270	229.6290127	0.1229674	0.0008902	32.745
1	1	10	1.6310535	0.0079685	1.6692252	0.0228679	0.0047738	32.464
2	1	10	2.6409563	0.0039165	2.7863129	0.0521681	0.0014056	33.519
5	1	10	11.3346539	0.0359433	12.9590963	0.1253515	0.0027736	33.046
10	1	10	128.0593475	0.5314444	167.9381766	0.2374614	0.0031645	33.062

Table 5 Numerical simulations for the approximation method in Framework 3.1 in the case of the replicator-mutator PDEs in (208) in Sect. 5.4 where we approximate the function $[-1, 1]^d \ni x \mapsto u(T, x) \in \mathbb{R}$, assuming for every $n, m, j \in \mathbb{N}$, $\mathbf{x} \in \mathbb{R}^d$ that $\mathcal{D} = [-1, 1]^d$, $\xi^{n,m,j} = 0$, $\gamma_m = 10^{-2}\mathbb{1}_{[0,1000]}(m) + 10^{-3}\mathbb{1}_{[1001,2000]}(m)$, $M_n = 2000$, and $\delta(\mathbf{x}) = 2^{-d}$

d	T	N	L^2-approx. error	Std. dev. error	avg. runtime (s)
1	1/10	10	0.0048685	0.0014896	56.143
2	1/10	10	0.0128065	0.0052835	58.438
5	1/10	10	0.0288090	0.0024520	57.517
10	1/10	10	0.0094485	0.0087283	57.756
1	1/5	10	0.0062907	0.0056264	57.715
2	1/5	10	0.0105398	0.0021021	59.511
5	1/5	10	0.0377786	0.0090177	58.430
10	1/5	10	0.0124844	0.0054922	58.513
1	1/2	10	0.0088032	0.0025974	58.530
2	1/2	10	0.0165290	0.0048038	61.922
5	1/2	10	0.0564803	0.0122768	61.148
10	1/2	10	0.0449148	0.0467653	61.746

realizations of $[-1, 1]^d \ni x \mapsto \mathbb{V}_N^{1,0}(\Theta_{M_N}^N, x) \in \mathbb{R}$. We report in Table 4 an estimate of the L^2-approximation error

$$\mathbb{E}\Big[\Big(\textstyle\int_{[-1,1]^d} |\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, x) - u(T, x)|^2 \, \mathrm{d}x\Big)^{1/2}\Big] \tag{209}$$

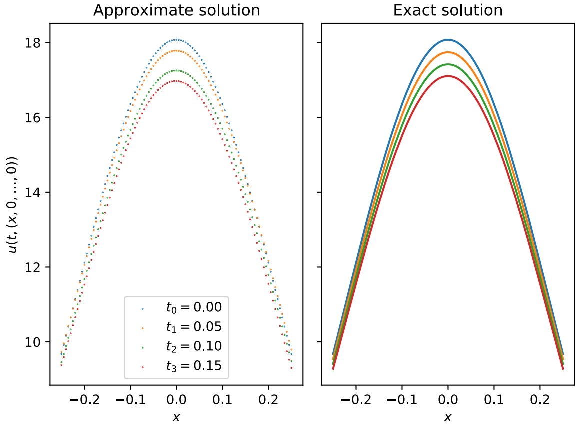


Fig. 3 Plot of a machine learning-based approximation of the solution of the replicator-mutator PDE in (208) in the case $d = 5$, $T = 1/2$, and $\mathcal{D} = \mathbb{R}^d$. The left-hand side shows a plot of $[-1/4, 1/4] \ni x \mapsto \mathbb{V}_n^{1,0}(\Theta_{M_n}^n(\omega), (x, 0, \ldots, 0)) \in \mathbb{R}$ for $n \in \{0, 1, 2, 3\}$ and one realization $\omega \in \Omega$ where the functions $\mathbb{R}^d \ni x \mapsto \mathbb{V}_n^{1,0}(\Theta_{M_n}^n(\omega), x) \in \mathbb{R}$ for $n \in \{0, 1, 2, 3\}$, $\omega \in \Omega$ were computed via Framework 5.1 as an approximation of the solution of the PDE in (208) with $d = 5$, $T = 1/2$, and $\mathcal{D} = [-1/2, 1/2]^d$ where we take $M_1 = M_2 = \cdots = M_N = 2000$, $\gamma_1 = \gamma_2 = \cdots = \gamma_{2000} = 1/200$, and $\delta = \mathbb{1}_{\mathbb{R}^d}$ and where we take $\xi^{n,m,j} \colon \Omega \to \mathbb{R}^d$, $n, m, j \in \mathbb{N}$, to be independent $\mathcal{U}_{[-1/2,1/2]^d}$-distributed random variables. The right-hand side of Fig. 3 shows a plot of $[-1/4, 1/4] \ni x \mapsto u(t, (x, 0, \ldots, 0)) \in \mathbb{R}$ for $t \in \{0, 0.05, 0.1, 0.15\}$ where u is the exact solution of the PDE in (208) with $d = 5$, $T = 1/2$, and $\mathcal{D} = \mathbb{R}^d$

and of the standard deviation

$$\left(\mathbb{E}\left[\left(\left(\int_{[-1,1]^d} |\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, x) - u(T, x)|^2 \, \mathrm{d}x\right)^{1/2} - \mathbb{E}\left[\left(\int_{[-1,1]^d} |\mathbb{V}_N^{1,0}(\Theta_{M_N}^N, x) - u(T, x)|^2 \, \mathrm{d}x\right)^{1/2}\right]\right)^2\right]\right)^{1/2} \tag{210}$$

of the error. The integrals appearing in (209) and (210) have been computed by means of a standard Monte Carlo approximation using the exact solution of the PDE as provided by Lemma 5.2 above.

Figure 3 provides a graphical illustration of an approximation to a solution of the PDEs in (208) computed by means of the the method in Framework 5.1 compared to the exact solution. More specifically, in Fig. 3 we use the machine learning-based method in Framework 5.1 to approximate the solution $u \colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (208) above with $d = 5$, $T = 1/2$, and $\mathcal{D} = \mathbb{R}^d$. The right-hand side of Fig. 3 shows a plot of $[-1/4, 1/4] \ni x \mapsto u(t, (x, 0, \ldots, 0)) \in \mathbb{R}$ for $t \in \{0, 0.05, 0.1, 0.15\}$ where u is the exact solution of the PDE in (208) with $d = 5$, $T = 1/2$, and $\mathcal{D} = \mathbb{R}^d$ computed via (188) in Lemma 5.2 below. The left-hand side of Fig. 3 shows a plot of $[-1/4, 1/4] \ni x \mapsto \mathbb{V}_n^{1,0}(\Theta_{M_n}^n(\omega), (x, 0, \ldots, 0)) \in \mathbb{R}$ for $n \in \{0, 1, 2, 3\}$ and one realization $\omega \in \Omega$ where the functions $\mathbb{R}^d \ni x \mapsto \mathbb{V}_n^{1,0}(\Theta_{M_n}^n(\omega), x) \in \mathbb{R}$ for $n \in \{0, 1, 2, 3\}$, $\omega \in \Omega$ were computed via Framework 5.1 as an approximation of the solution of the PDE in (208) with $d = 5$, $T = 1/2$, and $\mathcal{D} = [-1/2, 1/2]^d$. For the approximation, we take $M_1 = M_2 = \cdots = M_N = 2000$, $\gamma_1 = \gamma_2 = \cdots = \gamma_{2000} = 1/200$, and $\delta = \mathbb{1}_{\mathbb{R}^d}$ and we take $\xi^{n,m,j} \colon \Omega \to \mathbb{R}^d$, $n, m, j \in \mathbb{N}$, to be independent $\mathcal{U}_{[-1/2,1/2]^d}$-distributed random variables. Note that the solution of the PDE in (208) in the case $\mathcal{D} = [-R, R]^d$ with $R \in (0, \infty)$ sufficiently large is a good approximation of the solution $u \colon [0, T] \times \mathbb{R}^d \to \mathbb{R}$ of the PDE in (208) in the case $\mathcal{D} = \mathbb{R}^d$ since we have that for every $t \in [0, T]$ the value $u(t, x)$ of the solution u of the PDE in (208) in the case $\mathcal{D} = \mathbb{R}^d$ sufficiently quickly tends to 0 as $\|x\|$ tends to ∞.

MLP approximations.

For Table 6, we use the approximation method presented in Framework 4.1 to compute approximations of the PDEs in (208) by applying Example 4.5 with $\mathfrak{m}_1 = \mathfrak{m}_2 = \cdots = \mathfrak{m}_d = \frac{1}{10}$, $\mathfrak{s}_1 = \mathfrak{s}_2 = \cdots = \mathfrak{s}_d = \frac{1}{20}$, and $\mathfrak{u}_1 = \mathfrak{u}_2 = \cdots = \mathfrak{u}_d = 0$. We compute 5 independent realizations of $U^{(0)}_{5,5,0,\infty}(0, (0, \ldots, 0))$ as approximations of the value at $(0, x)$ of the solution u of the PDE in (166) in Example 4.5 above (note that this is the value at $(T, x) \in [0, T] \times \mathbb{R}^d$ of the solution u of the PDE in (208) above; cf., e.g., Beck et al. [101, Remark 3.3]). Based on these realizations, we report in Table 6 approximations of the mean and the standard deviation of the random variable $U^{(0)}_{5,5,0,\infty}(0, (0, \ldots, 0))$ as well as approximations of the relative L^1-approximation error and the uncorrected sample standard deviation of the approximation error. For the latter two statistics, the reference value has been calculated using Lemma 5.2 above which provides an explicit representation of the exact solution of the PDE in (208) above (and hence of the exact solution of the PDE in (166) in Example 4.5 above).

Comparison of the two methods.

In Fig. 4 below we compare the performance of the machine learning-based approximation method in Framework 5.1 to the MLP approximation method in Framework 4.1 when computing approximations to the solution of the PDE in (208) with $d = 5$, $\mathcal{D} = [-1, 1]^d$, and $T = 0.2$. A straightforward runtime comparison between these two methods is difficult, as the machine learning-based method is highly parallelizable, with our simulations running to a large part on the GPU, whereas the MLP approximation method cannot be parallelized as easily and in our simulations is performed entirely on the CPU. As a consequence, comparisons of the runtimes to a large degree reflect the relative performance of the GPU compared to the CPU of the test system. In Fig. 4 we instead use the number of evaluations of one-dimensional standard normal random variables as a proxy for the complexity and we plot this measure against the relative L^1-approximation error of the approximation method. For the MLP approximation method, we obtained the four different measurements by computing 5 realizations each of $U^{(0)}_{n,n,0,\infty}(0, (0, \ldots, 0))$ for $n \in \{2, 3, 4, 5\}$. For the machine learning-based approximation method in Framework 5.1, we similarly obtained the four different measurements by computing 5 realizations each of $\mathbb{V}^{1,0}_N(\Theta^N_{M_N}, (0, \ldots, 0))$ varying the number of time steps (with $N \in \{1, 2, 3, 4\}$), the number of neurons in each of the two hidden layers (with the number of neurons in each hidden layer chosen from $\{10, 20, 30, 40\}$), and the batch sizes (with $\forall\, k \in \mathbb{N}\colon J_k = J_1 \in \{10^1, 10^2, 10^3, 10^4\}$).

5.5 Allen–Cahn PDEs with conservation of mass

In this subsection we use the machine learning-based approximation method in Framework 5.1 to approximately calculate the solutions of certain Allen–Cahn PDEs with cubic nonlinearity, conservation of mass, and no-flux boundary conditions (cf., e.g., Rubinstein and Sternberg [10]).

Assume Framework 5.1, let $\epsilon \in (0, \infty)$ satisfy $\epsilon = \frac{1}{10}$, assume $d \in \{1, 2, 5, 10\}$, $D = [-1/2, 1/2]^d$, $T \in \{1/5, 1/2, 1\}$, $N = 10$, $K_1 = K_2 = \cdots = K_N = 5$, and $M_1 = M_2 = \cdots = M_N = 500$, assume $\xi^{n,m,j}$, $n, m, j \in \mathbb{N}$, are independent $\mathcal{U}_D$-distributed random variables, assume for every $m \in \mathbb{N}$ that $\gamma_m = 10^{-2}$, and assume for every $s, t \in [0, T]$, $x, \mathbf{x} \in D$,

Table 6 Numerical simulations for the approximation method in Framework 4.1 in the case of the replicator-mutator PDE in (208) in Sect. 5.4 (cf. Beck et al. [101, Remark 3.3] and Example 4.5 applied with $\mathfrak{m}_1 \curvearrowleft \frac{1}{10}, \mathfrak{m}_2 \curvearrowleft \frac{1}{10}, \dots, \mathfrak{m}_d \curvearrowleft \frac{1}{10}, \mathfrak{s}_1 \curvearrowleft \frac{1}{20}, \mathfrak{s}_2 \curvearrowleft \frac{1}{20}, \dots, \mathfrak{s}_d \curvearrowleft \frac{1}{20}, \mathfrak{u}_1 \curvearrowleft 0, \mathfrak{u}_2 \curvearrowleft 0, \dots, \mathfrak{u}_d \curvearrowleft 0$)

d	T	N	Mean of the approx. method	Standard deviation of the approx. method	Reference value	Relative L^1-approx. error	Standard deviation of the error	Average runtime in seconds
1	1/5	5	1.7567937	0.0004338	1.7582066	0.0008036	0.0002467	0.106
2	1/5	5	3.0850846	0.0016500	3.0912904	0.0020075	0.0005338	0.098
5	1/5	5	16.7011275	0.0220393	16.8015567	0.0059774	0.0013117	0.111
10	1/5	5	277.0042463	1.2390263	282.2923073	0.0187326	0.0043892	0.129
1	1/2	5	1.7191921	0.0009846	1.7222757	0.0017904	0.0005717	0.083
2	1/2	5	2.9487786	0.0068868	2.9662336	0.0058846	0.0023217	0.089
5	1/2	5	14.9121278	0.0220142	15.1535149	0.0159294	0.0014527	0.152
10	1/2	5	215.1817154	3.1947288	229.6290127	0.0629158	0.0139126	0.134
1	1	5	1.6602874	0.0020790	1.6692252	0.0053545	0.0012455	0.087
2	1	5	2.7528992	0.0121563	2.7863129	0.0119921	0.0043629	0.091
5	1	5	12.3874104	0.1425919	12.9590963	0.0441146	0.0110032	0.113
10	1	5	148.6160063	4.8691581	167.9381766	0.1150553	0.0289938	0.131

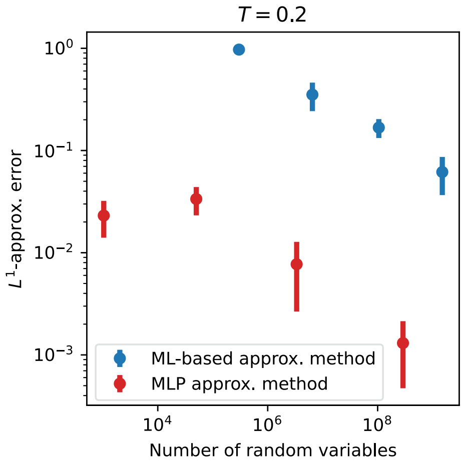


Fig. 4 Comparison of the machine learning-based method in Framework 3.1 and the MLP approximation method in Framework 2.10 when computing approximations to the solution of the PDE in (208) with $d = 5$, $\mathcal{D} = [-1, 1]^d$, and $T = 0.2$

$y, \mathbf{y} \in \mathbb{R}$, $v \in \mathbb{R}^d$, $A \in \mathcal{B}(D)$ that $\nu_x(A) = \int_A d\mathbf{x}$, $g(x) = \exp(-\frac{1}{4}\|x\|^2)$, $\mu(x) = (0, \dots, 0)$, $\sigma(x)v = \epsilon v$, $f(t, x, \mathbf{x}, y, \mathbf{y}) = y - y^3 - \big(\mathbf{y} - \mathbf{y}^3\big)$, and

$$H(t, s, x, v) = R(x, x + \mu(x)(t - s) + \sigma(x)v) = R(x, x + \epsilon v) \tag{211}$$

(cf. (139) and (150)). The solution $u \colon [0, T] \times D \to \mathbb{R}$ of the PDE in (174) then satisfies that for every $t \in (0, T]$, $x \in \partial_D$ it holds that $\langle \mathbf{n}(x), (\nabla_x u)(t, x)\rangle = 0$ and that for every $t \in [0, T]$, $x \in D$ it holds that $u(0, x) = \exp(-\frac{1}{4}\|x\|^2)$ and

$$\big(\tfrac{\partial}{\partial t} u\big)(t, x) = \tfrac{\epsilon^2}{2}(\Delta_x u)(t, x) + u(t, x) - [u(t, x)]^3 - \int_{[-1/2, 1/2]^d} u(t, \mathbf{x}) - [u(t, \mathbf{x})]^3 \, d\mathbf{x}. \tag{212}$$

For each choice of $d \in \{1, 2, 5, 10\}$ and $T \in \{1/5, 1/2, 1\}$ we approximate the unknown value $u(T, (0, \dots, 0))$ by computing 5 independent realizations of $\mathbb{V}^{1,0}_N(\Theta^N_{M_N}, (0, \dots, 0))$. We report in Table 7 approximations of the mean and the standard deviation of this random variable as well as approximations of the relative L^1-approximation error and the uncorrected sample standard deviation of the approximation error (see the discussion below Framework 5.1 for more details). For the latter two statistics, the reference value which is used as an approximation for the unknown value $u(T, (0, \dots, 0))$ of the exact solution of the PDE in (212) is computed via the MLP approximation method for non-local nonlinear PDEs in Framework 4.1 as described in Example 4.6 (cf. Beck et al. [101, Remark 3.3]). More precisely, we apply Example 4.6 with $d \in \{1, 2, 5, 10\}$, $T \in \{1/5, 1/2, 1\}$, $D = [-1/2, 1/2]^d$ in the notation of Example 4.6 and we use the mean of 5 independent realizations of $U^{(0)}_{5,5,0,\infty}(0, (0, \dots, 0))$ as the reference value.

5.6 Influence of hyperparameters

For both the machine learning-based approximation method and the MLP approximation method proposed above, several hyperparameters need to be chosen. More precisely, for the machine learning-based approximation method described in Framework 3.1, we need, e.g., to decide on the number of discrete time steps used in the approximation (denoted by $N \in \mathbb{N}$ in Framework 3.1 above), the numbers of samples used in the Monte Carlo approximation of the nonlocal term (denoted by $(K_n)_{n \in \mathbb{N}} \subseteq \mathbb{N}$ in Framework 3.1 above), the optimization

Table 7 Numerical simulations for the approximation method in Framework 3.1 in the case of the Allen–Cahn PDEs with conservation of mass in (212) in Sect. 5.5

d	T	N	Mean of the approx. method	Standard deviation of the approx. method	Reference value	Relative L^1-approx. error	Standard deviation of the error	Average runtime in seconds
1	$1/5$	10	0.9935298	0.0013204	0.9933120	0.0009695	0.0008076	24.426
2	$1/5$	10	0.9911092	0.0043262	0.9870172	0.0050432	0.0029841	28.312
5	$1/5$	10	0.9909120	0.0085060	0.9708144	0.0207017	0.0087617	31.648
10	$1/5$	10	0.9739443	0.0117640	0.9522296	0.0228040	0.0123542	36.040
1	$1/2$	10	0.9861730	0.0006471	0.9865877	0.0004706	0.0006119	24.926
2	$1/2$	10	0.9769603	0.0026599	0.9728481	0.0042269	0.0027342	30.445
5	$1/2$	10	0.9539644	0.0035275	0.9443930	0.0101350	0.0037352	33.436
10	$1/2$	10	0.9298821	0.0068851	0.9033037	0.0294236	0.0076221	38.802
1	1	10	0.9815766	0.0014325	0.9733318	0.0084707	0.0014717	24.999
2	1	10	0.9662454	0.0001834	0.9572384	0.0094093	0.0001916	31.398
5	1	10	0.9216945	0.0012842	0.9119894	0.0106418	0.0014081	36.176
10	1	10	0.8626075	0.0035230	0.8645918	0.0038279	0.0022071	40.781

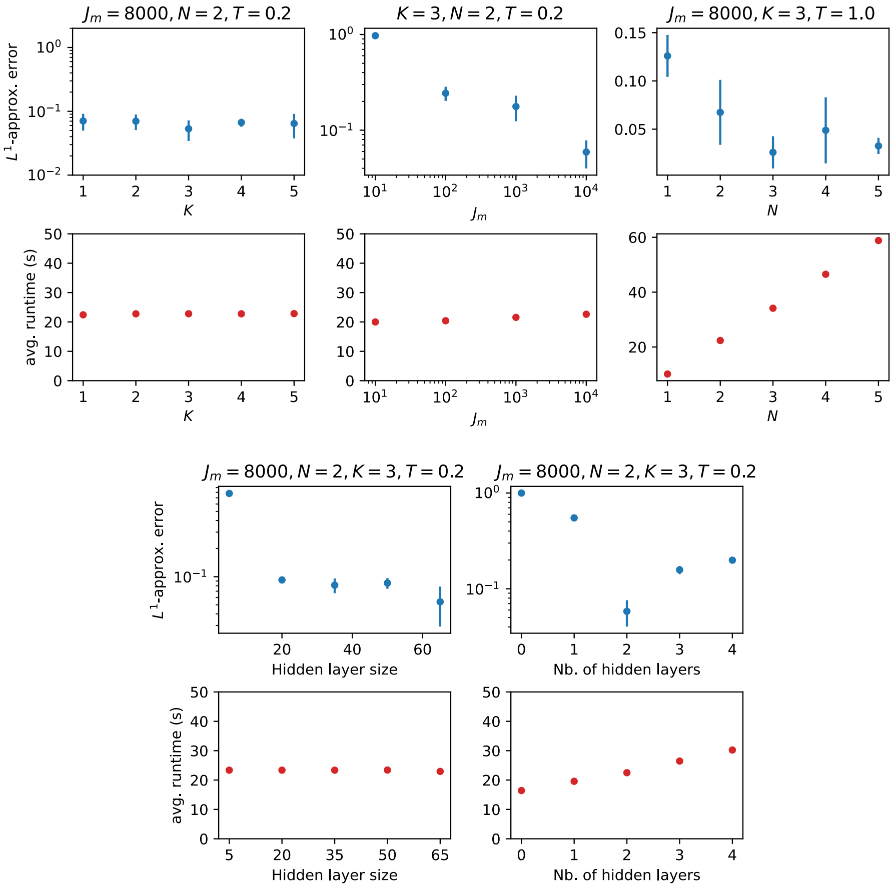
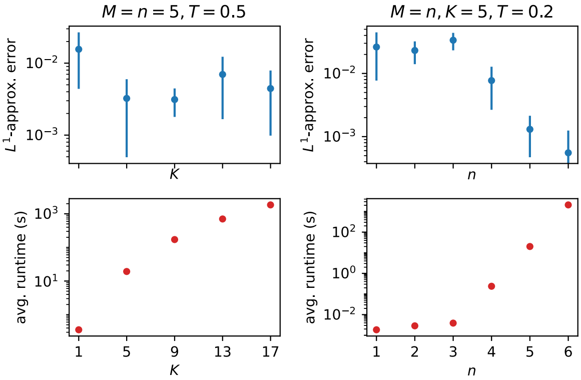


Fig. 5 Influence of various hyperparameters on the performance of the machine learning-based approximation method

algorithm used (represented by the functions $\Psi^n_m : \mathbb{R}^\varrho \times \mathbb{R}^\vartheta \to \mathbb{R}^\varrho, n \in \{1, 2, \ldots, N\}, m \in \mathbb{N}$, in Framework 3.1 above), the batch sizes used in the optimization algorithm (denoted by $(J_m)_{m\in\mathbb{N}} \subseteq \mathbb{N}$ in Framework 3.1 above), and the architecture of the employed neural networks (i.e., in the case of fully connected feedforward neural networks, the number of hidden layers and the number of neurons in each hidden layer). For the MLP approximation method introduced in Framework 4.1 above, we need, e.g., to decide on the numbers of samples used in the Monte Carlo approximation of the nonlocal term (denoted by $(K_{n,l,m})_{n,l,m\in\mathbb{N}_0} \subseteq \mathbb{N}$ in Framework 4.1 above) and which iteration to use as an approximation of the unknown PDE solution (i.e., for which $n, M \in \mathbb{N}_0$ to compute $U^{(0)}_{n,M,r_1,r_2}(t, x)$ in (159) in Framework 4.1 above).

In Figs. 5 and 6 we examine the influence of some of these choices on the accuracy and runtime of the proposed approximation methods when computing approximate solutions to the replicator-mutator PDEs in (208) where $d = 5$, where $\mathcal{D} = [-1, 1]^d$, and where for every $\mathbf{x} \in \mathbb{R}^d$ it holds that $\delta(\mathbf{x}) = 2^{-d}$. More specifically, in Fig. 5 we show the effect on the L^1-approximation error and on the runtime of the machine learning-based approximation method in Framework 5.1 when varying, respectively (from left to right and top to bottom),

Fig. 6 Influence of various hyperparameters on the performance of the MLP approximation method



the number of samples used in the Monte Carlo approximation of the nonlocal term, the batch size, the number of time steps, the number of neurons in each of the hidden layers, and the number of hidden layers. In Fig. 6 we show the effect on the L^1-approximation error and on the runtime of the MLP approximation method in Framework 4.1 (cf. Example 4.5) when varying, respectively (from left to right), the number of samples used to compute the Monte Carlo approximation of the nonlocal term and the number of iteration steps.

Acknowledgements We gratefully acknowledge the helpful suggestions by an anonymous referee. This project has been partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) in the frame of the priority programme SPP 2298 'Theoretical Foundations of Deep Learning' – Project no. 464123384. In addition, this project has been partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2044-390685587, Mathematics Münster: Dynamics – Geometry – Structure.

Declarations

Competing interest statement The authors declare that they have no competing interests regarding this manuscript.

References

1. Kavallaris, N.I., Suzuki, T.: Non-local Partial Differential Equations for Engineering and Biology. Mathematics for Industry (Tokyo), vol. **31**, p. 300. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67944-0
2. D'Elia, M., Du, Q., Glusa, C., Gunzburger, M., Tian, X., Zhou, Z.: Numerical methods for nonlocal and fractional models. Acta Numer. **29**, 1–124 (2020). https://doi.org/10.1017/S096249292000001X
3. Sunderasan, S.: Financial Modeling. In: Long-Term Investments, pp. 33–51. Routledge India, London (2020). https://doi.org/10.4324/9780367817909-3
4. Lacey, A.A.: Thermal runaway in a non-local problem modelling Ohmic heating. I. Model derivation and some special cases. Eur. J. Appl. Math. **6**(2), 127–144 (1995). https://doi.org/10.1017/S095679250000173X

5. Caglioti, E., Lions, P.-L., Marchioro, C., Pulvirenti, M.: A special class of stationary flows for two-dimensional Euler equations: a statistical mechanics description. II. Comm. Math. Phys. **174**(2), 229–260 (1995). https://doi.org/10.1007/BF02099602
6. Barone, A., Esposito, F., Magee, C.J., Scott, A.C.: Theory and applications of the sine-Gordon equation. La Rivista del Nuovo Cimento **1**(2), 227–267 (1971). https://doi.org/10.1007/BF02820622
7. Gajewski, H., Zacharias, K.: On a nonlocal phase separation model. J. Math. Anal. Appl. **286**(1), 11–31 (2003). https://doi.org/10.1016/S0022-247X(02)00425-0
8. Coleman, S.: Quantum sine-Gordon equation as the massive Thirring model. In: Bosonization, vol. 1, pp. 128–137. World Scientific, Singapore (1994). https://doi.org/10.1142/9789812812650_0013
9. Hairer, M., Shen, H.: The dynamical sine-Gordon model. Comm. Math. Phys. **341**(3), 933–989 (2016). https://doi.org/10.1007/s00220-015-2525-3
10. Rubinstein, J., Sternberg, P.: Nonlocal reaction–diffusion equations and nucleation. IMA J. Appl. Math. **48**(3), 249–264 (1992). https://doi.org/10.1093/imamat/48.3.249
11. Stoleriu, I.: Non-local models for solid-solid phase transitions. ROMAI J. **7**(1), 157–170 (2011)
12. Merton, R.C.: Option pricing when underlying stock returns are discontinuous. J. Financ. Econ. **3**(1–2), 125–144 (1976). https://doi.org/10.1016/0304-405X(76)90022-2
13. Chan, T.: Pricing contingent claims on stocks driven by Lévy processes. Ann. Appl. Probab. **9**(2), 504–528 (1999). https://doi.org/10.1214/aoap/1029962753
14. Kou, S.G.: A jump-diffusion model for option pricing. Manag. Sci. **48**(8), 1086–1101 (2002). https://doi.org/10.1287/mnsc.48.8.1086.166
15. Abergel, F., Tachet, R.: A nonlinear partial integro-differential equation from mathematical finance. Discr. Contin. Dyn. Syst. **27**(3), 907–917 (2010). https://doi.org/10.3934/dcds.2010.27.907
16. Benth, F.E., Karlsen, K.H., Reikvam, K.: Optimal portfolio selection with consumption and nonlinear integro-differential equations with gradient constraint: a viscosity solution approach. Finance Stoch. **5**(3), 275–303 (2001). https://doi.org/10.1007/PL00013538
17. Cruz, J.M.T.S., Ševčovič, D.: On solutions of a partial integro-differential equation in Bessel potential spaces with applications in option pricing models. Jpn. J. Ind. Appl. Math. **37**(3), 697–721 (2020). https://doi.org/10.1007/s13160-020-00414-2
18. Cont, R., Tankov, P.: Financial Modelling with Jump Processes. Chapman & Hall/CRC Financial Mathematics Series, Boca Raton (2004)
19. Huang, J., Cen, Z., Le, A.: A finite difference scheme for pricing American put options under Kou's jump-diffusion model. J. Funct. Spaces Appl. (2013). https://doi.org/10.1155/2013/651573
20. Gan, X., Yang, Y., Zhang, K.: A robust numerical method for pricing American options under Kou's jump-diffusion models based on penalty method. J. Appl. Math. Comput. **62**(1–2), 1–21 (2020). https://doi.org/10.1007/s12190-019-01270-1
21. Amadori, A.L.: Nonlinear integro-differential evolution problems arising in option pricing: a viscosity solutions approach. Differ. Integr. Equ. **16**(7), 787–811 (2003)
22. Pham, H.: Continuous-time Stochastic Control and Optimization with Financial Applications. Stochastic Modelling and Applied Probability, vol. 61. Springer, Berlin (2009). https://doi.org/10.1007/978-3-540-89500-8
23. Henry-Labordère, P.: Counterparty Risk Valuation: A Marked Branching Diffusion Approach. arXiv:1203.2369 (2012)
24. Oechssler, J., Riedel, F.: Evolutionary dynamics on infinite strategy spaces. Econ. Theory **17**(1), 141–162 (2001). https://doi.org/10.1007/PL00004092
25. Kavallaris, N.I., Lankeit, J., Winkler, M.: On a degenerate nonlocal parabolic problem describing infinite dimensional replicator dynamics. SIAM J. Math. Anal. **49**(2), 954–983 (2017). https://doi.org/10.1137/15M1053840
26. Hamel, F., Lavigne, F., Martin, G., Roques, L.: Dynamics of adaptation in an anisotropic phenotype-fitness landscape. Nonlinear Anal. Real World Appl. **54**, 103107 (2020). https://doi.org/10.1016/j.nonrwa.2020.103107
27. Alfaro, M., Carles, R.: Replicator-mutator equations with quadratic fitness. Proc. Am. Math. Soc. **145**(12), 5315–5327 (2017). https://doi.org/10.1090/proc/13669
28. Alfaro, M., Veruete, M.: Evolutionary branching via replicator-mutator equations. J. Dynam. Differ. Equ. **31**(4), 2029–2052 (2019). https://doi.org/10.1007/s10884-018-9692-9
29. Banerjee, M., Petrovskii, S.V., Volpert, V.: Nonlocal reaction–diffusion models of heterogeneous wealth distribution. Mathematics **9**(4), 351 (2021). https://doi.org/10.3390/math9040351
30. Lorz, A., Lorenzi, T., Hochberg, M.E., Clairambault, J., Perthame, B.: Populational adaptive evolution, chemotherapeutic resistance and multiple anti-cancer therapies. ESAIM Math. Model. Numer. Anal. **47**(2), 377–399 (2013). https://doi.org/10.1051/m2an/2012031

31. Chen, L., Painter, K., Surulescu, C., Zhigun, A.: Mathematical models for cell migration: A non-local perspective. Philos. Trans. R. Soc. B **375**(1807), 20190379 (2020). https://doi.org/10.1098/rstb.2019.0379
32. Villa, C., Chaplain, M.A.J., Lorenzi, T.: Evolutionary dynamics in vascularised tumours under chemotherapy: Mathematical modelling, asymptotic analysis and numerical simulations. Vietnam J. Math. **49**(1), 143–167 (2021). https://doi.org/10.1007/s10013-020-00445-9
33. Pájaro, M., Alonso, A.A., Otero-Muras, I., Vázquez, C.: Stochastic modeling and numerical simulation of gene regulatory networks with protein bursting. J. Theoret. Biol. **421**, 51–70 (2017). https://doi.org/10.1016/j.jtbi.2017.03.017
34. Fisher, R.A.: The wave of advance of advantageous genes. Ann. Eugen. **7**(4), 355–369 (1937). https://doi.org/10.1111/j.1469-1809.1937.tb02153.x
35. Hamel, F., Nadirashvili, N.: Travelling fronts and entire solutions of the Fisher-KPP equation in $\mathbb{R}^N$. Arch. Ration. Mech. Anal. **157**(2), 91–163 (2001). https://doi.org/10.1007/PL00004238
36. Bian, S., Chen, L., Latos, E.A.: Global existence and asymptotic behavior of solutions to a nonlocal Fisher-KPP type problem. Nonlinear Anal. **149**, 165–176 (2017). https://doi.org/10.1016/j.na.2016.10.017
37. Perthame, B., Génieys, S.: Concentration in the nonlocal Fisher equation: The Hamilton–Jacobi limit. Math. Model. Nat. Phenom. **2**(4), 135–151 (2007). https://doi.org/10.1051/mmnp:2008029
38. Berestycki, H., Nadin, G., Perthame, B., Ryzhik, L.: The non-local Fisher-KPP equation: Travelling waves and steady states. Nonlinearity **22**(12), 2813–2844 (2009). https://doi.org/10.1088/0951-7715/22/12/002
39. Houchmandzadeh, B., Vallade, M.: Fisher waves: An individual-based stochastic model. Phys. Rev. E **96**(1), 012414 (2017). https://doi.org/10.1103/PhysRevE.96.012414
40. Wang, F., Xue, L., Zhao, K., Zheng, X.: Global stabilization and boundary control of generalized Fisher/KPP equation and application to diffusive SIS model. J. Differ. Equ. **275**, 391–417 (2021). https://doi.org/10.1016/j.jde.2020.11.031
41. Burger, R., Hofbauer, J.: Mutation load and mutation-selection-balance in quantitative genetic traits. J. Math. Biol. **32**(3), 193–218 (1994). https://doi.org/10.1007/BF00163878
42. Génieys, S., Volpert, V., Auger, P.: Pattern and waves for a model in population dynamics with nonlocal consumption of resources. Math. Model. Nat. Phenom. **1**(1), 65–82 (2006). https://doi.org/10.1051/mmnp:2006004
43. Berestycki, H., Jin, T., Silvestre, L.: Propagation in a non local reaction diffusion equation with spatial and genetic trait structure. Nonlinearity **29**(4), 1434–1466 (2016). https://doi.org/10.1088/0951-7715/29/4/1434
44. Nordbotten, J.M., Stenseth, N.C.: Asymmetric ecological conditions favor Red-Queen type of continued evolution over stasis. Proc. Natl. Acad. Sci. U.S.A. **113**(7), 1847–1852 (2016). https://doi.org/10.1073/pnas.1525395113
45. Nordbotten, J.M., Levin, S.A., Szathmáry, E., Stenseth, N.C.: Ecological and evolutionary dynamics of interconnectedness and modularity. Proc. Natl. Acad. Sci. U.S.A. **115**(4), 750–755 (2018). https://doi.org/10.1073/pnas.1716078115
46. Roques, L., Bonnefon, O.: Modelling population dynamics in realistic landscapes with linear elements: A mechanistic-statistical reaction–diffusion approach. PLoS ONE **11**(3), 0151217 (2016). https://doi.org/10.1371/journal.pone.0151217
47. Doebeli, M., Ispolatov, I.: Complexity and diversity. Science **328**(5977), 494–497 (2010). https://doi.org/10.1126/science.1187468
48. Nordbotten, J.M., Bokma, F., Hermansen, J.S., Stenseth, N.C.: The dynamics of trait variance in multi-species communities. R. Soc. Open Sci. **7**(8), 200321 (2020). https://doi.org/10.1098/rsos.200321
49. Bellman, R.: Dynamic Programming. Princeton Landmarks in Mathematics. Princeton University Press, Princeton (2010)
50. Metropolis, N., Ulam, S.: The Monte Carlo method. J. Am. Stat. Assoc. **44**(247), 335–341 (1949). https://doi.org/10.1080/01621459.1949.10483310
51. Bauer, W.F.: The Monte Carlo method. J. Soc. Ind. Appl. Math. **6**(4), 438–451 (1958). https://doi.org/10.1137/0106028
52. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
53. Beck, C., Hutzenthaler, M., Jentzen, A., Kuckuck, B.: An overview on deep learning-based approximation methods for partial differential equations. Discr. Contin. Dyn. Syst. Ser. B **28**(6), 3697–3746 (2023). https://doi.org/10.3934/dcdsb.2022238

54. Weinan, E., Han, J., Jentzen, A.: Algorithms for solving high dimensional PDEs: From nonlinear Monte Carlo to machine learning. Nonlinearity **35**(1), 278–310 (2022). https://doi.org/10.1088/1361-6544/ac337f
55. Blechschmidt, J., Ernst, O.G.: Three ways to solve partial differential equations with neural networks—a review. GAMM-Mitteilungen **44**(2), 202100006 (2021). https://doi.org/10.1002/gamm.202100006
56. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nat. Rev. Phys. **3**(6), 422–440 (2021). https://doi.org/10.1038/s42254-021-00314-5
57. Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F.: Scientific machine learning through physics-informed neural networks: Where we are and what's next. J. Sci. Comput. **92**(3), 88 (2022). https://doi.org/10.1007/s10915-022-01939-z
58. Yunus, R.B., Abdul Karim, S.A., Shafie, A., Izzatullah, M., Kherd, A., Hasan, M.K., Sulaiman, J.: An overview on deep learning techniques in solving partial differential equations. In: Abdul Karim, S.A. (ed.) Intelligent Systems Modeling and Simulation II, pp. 37–47. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-04028-3_4
59. Huang, S., Feng, W., Tang, C., Lv, J.: Partial differential equations meet deep neural networks: A survey. arXiv:2211.05567 (2022)
60. E, W., Han, J., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**(4), 349–380 (2017). https://doi.org/10.1007/s40304-017-0117-6
61. Han, J., Jentzen, A., E, W.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. U.S.A. **115**(34), 8505–8510 (2018). https://doi.org/10.1073/pnas.1718942115
62. Beck, C., Becker, S., Grohs, P., Jaafari, N., Jentzen, A.: Solving the Kolmogorov PDE by means of deep learning. J. Sci. Comput. **88**, 73–28 (2021). https://doi.org/10.1007/s10915-021-01590-0
63. Chan-Wai-Nam, Q., Mikael, J., Warin, X.: Machine learning for semi linear PDEs. J. Sci. Comput. **79**(3), 1667–1712 (2019). https://doi.org/10.1007/s10915-019-00908-3
64. Huré, C., Pham, H., Warin, X.: Deep backward schemes for high-dimensional nonlinear PDEs. Math. Comp. **89**, 1547–1579 (2020). https://doi.org/10.1090/mcom/3514
65. Beck, C., Becker, S., Cheridito, P., Jentzen, A., Neufeld, A.: Deep splitting method for parabolic PDEs. SIAM J. Sci. Comput. **43**(5), 3135–3154 (2021). https://doi.org/10.1137/19M1297919
66. Cox, S., Neerven, J.: Pathwise Hölder convergence of the implicit-linear Euler scheme for semi-linear SPDEs with multiplicative noise. Numer. Math. **125**(2), 259–345 (2013). https://doi.org/10.1007/s00211-013-0538-4
67. Gyöngy, I., Krylov, N.: On the splitting-up method and stochastic partial differential equations. Ann. Probab. **31**(2), 564–591 (2003). https://doi.org/10.1214/aop/1048516528
68. Hochbruck, M., Ostermann, A.: Explicit exponential Runge–Kutta methods for semilinear parabolic problems. SIAM J. Numer. Anal. **43**(3), 1069–1090 (2005). https://doi.org/10.1137/040611434
69. Hutzenthaler, M., Jentzen, A., Kruse, T., Nguyen, T.A., Wurstemberger, P.: Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. Proc. A. **476**(2244), 20190630 (2020). https://doi.org/10.1098/rspa.2019.0630
70. E, W., Hutzenthaler, M., Jentzen, A., Kruse, T.: On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. J. Sci. Comput. **79**(3), 1534–1571 (2019). https://doi.org/10.1007/s10915-018-00903-0
71. E, W., Hutzenthaler, M., Jentzen, A., Kruse, T.: Multilevel Picard iterations for solving smooth semilinear parabolic heat equations. Partial Differ. Equ. Appl. **2**(6), 80 (2021). https://doi.org/10.1007/s42985-021-00089-5
72. Heinrich, S.: Monte Carlo complexity of global solution of integral equations. J. Complex. **14**(2), 151–175 (1998). https://doi.org/10.1006/jcom.1998.0471
73. Heinrich, S., Sindambiwe, E.: Monte Carlo complexity of parametric integration. J. Complex. **15**(3), 317–341 (1999). https://doi.org/10.1006/jcom.1999.0508
74. Grohs, P., Voigtlaender, F.: Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces. Found. Comput. Math. (2023). https://doi.org/10.1007/s10208-023-09607-w
75. 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. J. Comput. Phys. **378**, 686–707 (2019). https://doi.org/10.1016/j.jcp.2018.10.045
76. Sirignano, J., Spiliopoulos, K.: DGM: A deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018). https://doi.org/10.1016/j.jcp.2018.08.029
77. Pang, G., Lu, L., Karniadakis, G.E.: fPINNs: Fractional physics-informed neural networks. SIAM J. Sci. Comput. **41**(4), 2603–2626 (2019). https://doi.org/10.1137/18M1229845

78. Lu, L., Meng, X., Mao, Z., Karniadakis, G.E.: DeepXDE: A deep learning library for solving differential equations. SIAM Rev. **63**(1), 208–228 (2021). https://doi.org/10.1137/19M1274067
79. Guo, L., Wu, H., Yu, X., Zhou, T.: Monte Carlo fPINNs: Deep learning method for forward and inverse problems involving high dimensional fractional partial differential equations. Comput. Methods Appl. Mech. Engrg. **400**, 115523 (2022). https://doi.org/10.1016/j.cma.2022.115523
80. Al-Aradi, A., Correia, A., Jardim, G., de Freitas Naiff, D., Saporito, Y.: Extensions of the deep Galerkin method. Appl. Math. Comput. **430**, 127287 (2022). https://doi.org/10.1016/j.amc.2022.127287
81. Yuan, L., Ni, Y.-Q., Deng, X.-Y., Hao, S.: A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. J. Comput. Phys. **462**, 111260 (2022). https://doi.org/10.1016/j.jcp.2022.111260
82. Frey, R., Köck, V.: Deep neural network algorithms for parabolic PIDEs and applications in insurance and finance. Computation **10**(11), 201 (2022). https://doi.org/10.3390/computation10110201
83. Frey, R., Köck, V.: Convergence analysis of the deep splitting scheme: the case of partial integro-differential equations and the associated FBSDEs with jumps. arXiv:2206.01597 (2022)
84. Castro, J.: Deep learning schemes for parabolic nonlocal integro-differential equations. Partial Differ. Equ. Appl. **3**, 77 (2022). https://doi.org/10.1007/s42985-022-00213-z
85. Gonon, L., Schwab, C.: Deep ReLU neural networks overcome the curse of dimensionality for partial integrodifferential equations. Anal. Appl. (Singap.) **21**(1), 1–47 (2023). https://doi.org/10.1142/S0219530522500129
86. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**(5), 987–1000 (1998). https://doi.org/10.1109/72.712178
87. Lagaris, I.E., Likas, A.C., Papageorgiou, D.G.: Neural-network methods for boundary value problems with irregular boundaries. IEEE Trans. Neural Netw. **11**(5), 1041–1049 (2000). https://doi.org/10.1109/72.870037
88. McFall, K.S., Mahan, J.R.: Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. IEEE Trans. Neural Netw. **20**(8), 1221–1233 (2009). https://doi.org/10.1109/TNN.2009.2020735
89. Sukumar, N., Srivastava, A.: Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. Comput. Methods Appl. Mech. Engrg. **389**, 114333 (2022). https://doi.org/10.1016/j.cma.2021.114333
90. Wang, S., Perdikaris, P.: Deep learning of free boundary and Stefan problems. J. Comput. Phys. (2020). https://doi.org/10.1016/j.jcp.2020.109914
91. E, W., Yu, B.: The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**(1), 1–12 (2018). https://doi.org/10.1007/s40304-018-0127-z
92. Liao, Y., Ming, P.: Deep Nitsche method: Deep Ritz method with essential boundary conditions. Commun. Comput. Phys. **29**(5), 1365–1384 (2021). https://doi.org/10.4208/cicp.OA-2020-0219
93. Chen, J., Du, R., Wu, K.: A comparison study of deep Galerkin method and deep Ritz method for elliptic problems with different boundary conditions. Commun. Math. Res. **36**(3), 354–376 (2020). https://doi.org/10.4208/cmr.2020-0051
94. Zang, Y., Bao, G., Ye, X., Zhou, H.: Weak adversarial networks for high-dimensional partial differential equations. J. Comput. Phys. **411**, 109409 (2020). https://doi.org/10.1016/j.jcp.2020.109409
95. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**(61), 2121–2159 (2011)
96. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
97. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. **37**, pp. 448–456. PMLR, Lille, France (2015). https://proceedings.mlr.press/v37/ioffe15.html
98. E, W., Han, J., Jentzen, A.: Algorithms for solving high dimensional PDEs: From nonlinear Monte Carlo to machine learning. Nonlinearity **35**(1), 278–310 (2021). https://doi.org/10.1088/1361-6544/ac337f
99. Becker, S., Braunwarth, R., Hutzenthaler, M., Jentzen, A., Wurstemberger, P.: Numerical simulations for full history recursive multilevel Picard approximations for systems of high-dimensional partial differential equations. Commun. Comput. Phys. **28**(5), 2109–2138 (2020). https://doi.org/10.4208/cicp.OA-2020-0130
100. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. **9**, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (2010). https://proceedings.mlr.press/v9/glorot10a.html

101. Beck, C., E, W., Jentzen, A.: Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. arXiv:1709.05963 (2017)