Parallel-In-Time Solutions with Extreme Learning Machines

Marta M. Betcke[1], Lisa Maria Kreusser[2], Davide Murari[3]

[1]Department of Computer Science, University College London.
[2]Department of Mathematical Sciences, University of Bath.
[3]Department of Mathematical Sciences, Norwegian University of Science and Technology.

Contributing authors: m.betcke@ucl.ac.uk; lmk54@bath.ac.uk; davide.murari@ntnu.no;

Abstract

This paper considers one of the fundamental parallel-in-time methods for the solution of ordinary differential equations, Parareal, and extends it by adopting a neural network as a coarse propagator. We provide a theoretical analysis of the convergence properties of the proposed algorithm and show its effectiveness for several examples, including Lorenz and Burgers' equations. In our numerical simulations, we further specialize the underpinning neural architecture to Extreme Learning Machines (ELMs), a $\mathbf{2}-$layer neural network where the first layer weights are drawn at random rather than optimized. This restriction substantially increases the efficiency of fitting ELM's weights in comparison to a standard feedforward network without negatively impacting the accuracy, as demonstrated in the SIR system example.

Keywords: Parareal, Physics Informed Neural Networks, Extreme Learning Machines, Mathematics of Deep Learning.

1 Introduction

In this paper, we consider initial value problems expressed as a system of first-order ordinary differential equations (ODEs). This wide class of problems arises in many social and natural sciences applications, including semi-discretized, time-dependent partial differential equations. We express a generic system of such differential equations as

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)) \in \mathbb{R}^d, \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \tag{1}$$

which will be our reference problem. Here, $'$ denotes the derivative with respect to the time variable. To guarantee the existence and uniqueness of its solutions, we assume that $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$ is a Lipschitz-continuous vector field and $t \in [0, T]$ for some $T > 0$. Solving an initial value problem like (1) analytically is generally not a possibility, and hence one needs to rely on numerical approximations to the solution curve $t \mapsto \mathbf{x}(t)$. Numerical techniques rely on introducing a time discretization $0 < t_1 < \cdots < t_N = T$ of the interval $[0, T]$, with steps $\Delta t_n = t_{n+1} - t_n$, and computing approximations $\mathbf{x}_n$ of the solution $\mathbf{x}(t_n)$ at the nodes t_n, i.e., $\mathbf{x}_n \approx \mathbf{x}(t_n)$. A popular and established option is provided by one-step methods, such as Runge–Kutta schemes, which relate $\mathbf{x}_{n+1}$ to $\mathbf{x}_n$ in terms of a map $\varphi_{\mathcal{F}}^{\Delta t_n}$ of the form $\mathbf{x}_{n+1} = \varphi_{\mathcal{F}}^{\Delta t_n}(\mathbf{x}_n)$. Collocation methods are a subset of Runge–Kutta methods [1, Section II.7] with particular relevance to this paper. These methods aim to approximate the solution on each interval $[t_n, t_{n+1}]$ with a real polynomial $\tilde{\mathbf{x}}$ of a sufficiently high degree and coefficients in $\mathbb{R}^d$. The updated solution is then computed as $\mathbf{x}_{n+1} = \varphi_{\mathcal{F}}^{\Delta t_n}(\mathbf{x}_n)$ evaluating the polynomial at $t = t_{n+1}$ as $\mathbf{x}_{n+1} = \tilde{\mathbf{x}}(t_{n+1}) \approx \mathbf{x}(t_{n+1})$. To determine the coefficients of the polynomial $\tilde{\mathbf{x}}(t)$, one needs to solve the system of algebraic equations $\tilde{\mathbf{x}}'(t_{n,c}) = \mathcal{F}(\tilde{\mathbf{x}}(t_{n,c}))$ for a set of C collocation points $t_n \leq t_{n,1} < t_{n,2} < \cdots < t_{n,C} \leq t_{n+1}$.

As initial value problems define causal processes, many time-stepping schemes are sequential by nature, in the sense that to compute $\mathbf{x}_{n+1}$, one has to compute $\mathbf{x}_n$ first. Nonetheless, multiple successful approaches such as Parareal [2], PFASST [3], and MGRIT [4] have introduced some notion of parallel-in-time solution of initial value problems (1), see for instance [5] for an overview of existing methods.

In this work, we build upon the Parareal algorithm [2]. The speedup in Parareal is achieved by coupling a fine time step integrator with a coarse step integrator. In each iteration, the coarse integrator updates the initial conditions of initial value problems on time subintervals, which can be solved in parallel and only entail fine step time integration over a short time. The elegance and strong theoretical grounding of the idea (see [6, 7], for instance) led to a number of variants of the Parareal algorithm, including combinations of Parareal with neural networks [8–10].

In recent years, solving differential equations with machine learning approaches gained in popularity; see, for instance, [11] for a review. For learned methods to become staple solvers, understanding their properties and ensuring they reproduce the qualitative behavior of the solutions is paramount. The problem of convergence and generalization for neural network-based PDE solvers has been considered in [12–14], for instance. An analysis of the approximation properties of neural networks in the context of PDE solutions is provided in [15, 16]. In the context of ODEs, there is an increasing interest in developing deep neural networks to learn time-stepping schemes unrestricted by constraints of the local Taylor series, including approaches based on flow maps [17], model order reduction [18], and spectral methods [19].

In the context of combining Parareal with neural networks, Parareal with a physics-informed neural network as a coarse propagator was suggested in [9]. In [8], the authors introduced a parallel (deep) neural network based on parallelizing the forward propagation following similar principles to those behind Parareal. In [10], it was proposed to learn a coarse propagator by parameterizing its stability function and optimizing the associated weights to minimize an analytic convergence factor of the Parareal method for parabolic PDEs.

Neural networks are generally considered as a composition of parametric maps whose weights are all optimized so that a task of interest is solved with sufficient accuracy. The common choice of the optimization procedure is gradient-based algorithms, which start from a random set of initial weights and update them iteratively until the stopping criterion has been reached. A class of neural networks where some of the weights are not updated at all is often called Extreme Learning Machines (ELMs) [20, 21]. Despite their seemingly reduced capability of approximating functions, these neural networks retain most of the approximation results of more conventional neural networks. For example, as derived in [20, Theorem 2.1], ELMs with two layers and H hidden neurons, where only the last layer is optimized while all other weights are independently sampled from any interval according to any continuous probability distribution, can interpolate with probability one any set of H distinct input-to-output pairs. Their expressivity properties, see e.g. [22, 23], make them suitable for the approximation of solutions of ODEs which were successfully considered in [24–28], yielding accurate approximations in a fraction of the training time when compared to more conventional networks.

1.1 Contributions

In this work, we build a hybrid numerical method based on the Parareal framework, where an ELM constitutes the coarse time stepping scheme. We first derive an a-posteriori error estimate for general neural network-based solvers of ODEs. This theoretical result allows us to replace the coarse integrator of the Parareal method with an ELM while preserving its convergence guarantees. The ELMs are trained online during the Parareal iterations. There are several benefits to the proposed procedure. First, our hybrid approach comes with theoretical guarantees and allows us to solve a differential equation such that the produced solution is accurate to a certain degree. Additionally, using ELMs rather than a more conventional neural network leads to a significant speedup in the algorithm without sacrificing its capabilities. Indeed, as we show for the SIR problem, using ELMs leads to about half of the computational time of the other method, even without accounting for the offline training phase of the more conventional network. Further, we demonstrate the effectiveness of the proposed approach, together with the timings of the components of the algorithm, and apply it to several examples in Section 6.

1.2 Outline

The outline of the paper is as follows. We start with introducing the Parareal algorithm and its convergence properties in Section 2. Section 3 presents the theoretical derivation of an a-posteriori error estimate for neural network-based solvers. This result relies on a non-linear variation of the constants formula, also called the Gröbner-Alekseev Theorem. In Section 4, we propose a hybrid algorithm combining the Parareal framework with the ELM-based coarse propagator. We study the convergence properties of this hybrid algorithm in Section 5. The effectiveness of the proposed method is tested in Section 6 on the benchmark dynamical systems studied in [6] with the addition of the SIR and ROBER problems. We conclude with the summary and analysis of the obtained results in Section 7.

2 Parareal method

This section introduces the Parareal algorithm [2] and presents a convergence result needed for our derivations.

2.1 The method

The Parareal algorithm builds on two one-step methods that we call $\varphi_F^{\Delta t}, \varphi_C^{\Delta t} : \mathbb{R}^d \to \mathbb{R}^d$, denoting the fine and coarse integrators with timestep Δt, respectively. There are multiple options to design such maps, one being to use the same one-step method but with finer or coarser timesteps, e.g.,

$$\varphi_F^{\Delta t} := \varphi_C^{\Delta t/M} \circ \cdots \circ \varphi_C^{\Delta t/M} = \left(\varphi_C^{\Delta t/M}\right)^M, \ M \in \mathbb{N}.$$

This strategy motivates the subscripts of the two maps since these methods rely on a fine and a coarse mesh. Another option to define $\varphi_F^{\Delta t}$ and $\varphi_C^{\Delta t}$ is to use methods of different orders, hence different levels of accuracy with the same timestep Δt. Regardless of how we define these two methods, the map $\varphi_F^{\Delta t}$ is more expensive to evaluate than $\varphi_C^{\Delta t}$. The goal of the Parareal algorithm is to get an approximate solution $\{\mathbf{x}_n\}_{n=0}^N$ over the mesh $t_0 = 0 < t_1 < \cdots < t_N = T$, $\Delta t_n = t_{n+1} - t_n$, with the same degree of accuracy as the one obtained with $\varphi_F^{\Delta t_n}$ but in a shorter time. This is achieved by transforming (1) into a collection of initial value problems on a shorter time interval by using $\varphi_C^{\Delta t_n}$. This zeroth iterate of the method consists of finding intermediate initial conditions $\mathbf{x}_n^0$ by integrating (1) with $\varphi_C^{\Delta t_n}$ to get

$$\mathbf{x}_0^0 = \mathbf{x}_0, \ \mathbf{x}_{n+1}^0 = \varphi_C^{\Delta t_n}\left(\mathbf{x}_n^0\right), \ n = 0, \dots, N-1, \ \Delta t_n = t_{n+1} - t_n,$$

and define the N initial value problems on the subintervals

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)), \\ \mathbf{x}(t_n) = \mathbf{x}_n^0, \end{cases} \qquad t \in [t_n, t_{n+1}], \ n = 0, \dots, N-1. \tag{2}$$

These problems can now be solved in parallel using the fine integrator $\varphi_F^{\Delta t_n}$, which constitutes the parallel step in all successive Parareal iterates. A predictor-corrector scheme is used to iteratively update the initial conditions on the subintervals $[t_n, t_{n+1}]$. Parareal iteration $i+1$ reads

$$\mathbf{x}_{n+1}^{i+1} = \varphi_F^{\Delta t_n}\left(\mathbf{x}_n^i\right) + \varphi_C^{\Delta t_n}\left(\mathbf{x}_n^{i+1}\right) - \varphi_C^{\Delta t_n}\left(\mathbf{x}_n^i\right), \ n = 0, \ldots, N-1. \tag{3}$$

A common choice of a stopping criterion is $\max_{n=1,\ldots,N} \|\mathbf{x}_n^{i+1} - \mathbf{x}_n^i\|_2 < \varepsilon$ for some tolerance $\varepsilon > 0$. The parallel speedup is achieved if this criterion is met with far less iterates than the number of time intervals N.

2.2 Interpretation of the correction term

Following [6], we provide the interpretation of (3) as an approximation of the Newton step for matching the exact flow at the time discretization points $t_0 = 0, \ldots, t_N = T$. We consider

$$\mathcal{H}(\mathbf{y}) := \begin{bmatrix} \mathbf{y}_0 - \mathbf{x}_0 \\ \mathbf{y}_1 - \phi_{\mathcal{F}}^{\Delta t_0}(\mathbf{y}_0) \\ \mathbf{y}_2 - \phi_{\mathcal{F}}^{\Delta t_1}(\mathbf{y}_1) \\ \vdots \\ \mathbf{y}_N - \phi_{\mathcal{F}}^{\Delta t_{N-1}}(\mathbf{y}_{N-1}) \end{bmatrix} = 0, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix} \in \mathbb{R}^{d \cdot (N+1)},$$

where $\phi_{\mathcal{F}}^{\Delta t}(\mathbf{x}_n)$ with $\mathbf{x}_n \in \mathbb{R}^d$ is the exact solution $\mathbf{x}(\Delta t)$ of the initial value problem

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)), \\ \mathbf{x}(0) = \mathbf{x}_n. \end{cases}$$

Linearizing $\mathcal{H}$ at the ith iterate, $\mathbf{x}^i$, equating it to 0 and solving for the $i+1$st iterate, $\mathbf{x}^{i+1}$, we arrive at the Newton update

$$\mathbf{x}_{n+1}^{i+1} = \phi_{\mathcal{F}}^{\Delta t_n}\left(\mathbf{x}_n^i\right) + \partial_{\mathbf{x}}\left(\phi_{\mathcal{F}}^{\Delta t_n}\right)\left(\mathbf{x}_n^i\right)\left(\mathbf{x}_n^{i+1} - \mathbf{x}_n^i\right), \quad n = 0, \ldots, N-1$$

for the solution of the system $\mathcal{H}(\mathbf{y}) = 0$. The idea behind Parareal is then to approximate the unknown $\phi_{\mathcal{F}}^{\Delta t_n}(\mathbf{x}_n^i)$ with $\varphi_F^{\Delta t_n}(\mathbf{x}_n^i)$, and the first order term with

$$\partial_{\mathbf{x}}\left(\phi_{\mathcal{F}}^{\Delta t_n}\right)\left(\mathbf{x}_n^i\right)\left(\mathbf{x}_n^{i+1} - \mathbf{x}_n^i\right) \approx \varphi_C^{\Delta t_n}\left(\mathbf{x}_n^{i+1}\right) - \varphi_C^{\Delta t_n}\left(\mathbf{x}_n^i\right),$$

which yields (3).

2.3 Convergence

Convergence of the Parareal iterations was proven in [6] under the assumption that the fine integrator $\varphi_F^{\Delta t}$ and the exact flow map $\phi_{\mathcal{F}}^{\Delta t}$ coincide.

Theorem 1 (Theorem 1 in [6]). *Let us consider the initial value problem* (1) *and partition the time interval* $[0,T]$ *into* N *intervals of size* $\Delta t = T/N$ *using a grid of nodes* $t_n = n\Delta t$. *Assume that the fine integrator* $\varphi_F^{\Delta t}$ *coincides with the exact flow map* $\phi_{\mathcal{F}}^{\Delta t}$, *i.e.* $\varphi_F^{\Delta t} = \phi_{\mathcal{F}}^{\Delta t}$. *Furthermore, suppose that there exist* $p \in \mathbb{N}$, *a set of continuously differentiable functions* $c_{p+1}, c_{p+2}, \cdots$, *and* $\alpha > 0$ *such that*

$$\begin{aligned} &\varphi_F^{\Delta t}(\mathbf{x}) - \varphi_C^{\Delta t}(\mathbf{x}) = c_{p+1}(\mathbf{x})(\Delta t)^{p+1} + c_{p+2}(\mathbf{x})(\Delta t)^{p+2} + \cdots, \quad \textit{and} \\ &\left\|\varphi_F^{\Delta t}(\mathbf{x}) - \varphi_C^{\Delta t}(\mathbf{x})\right\|_2 \leq \alpha(\Delta t)^{p+1} \end{aligned} \tag{4}$$

for every $\mathbf{x} \in \mathbb{R}^d$, *and also that there exists* $\beta > 0$ *such that*

$$\left\|\varphi_C^{\Delta t}(\mathbf{x}) - \varphi_C^{\Delta t}(\mathbf{y})\right\|_2 \leq (1 + \beta\Delta t)\left\|\mathbf{x} - \mathbf{y}\right\|_2 \tag{5}$$

for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. *Then there exists a positive constant* γ *such that, at the* $i-$*th iterate of the Parareal method, the following bound holds*

$$\left\|\mathbf{x}(t_n) - \mathbf{x}_n^i\right\|_2 \leq \frac{\alpha}{\gamma}\frac{\left(\gamma(\Delta t)^{p+1}\right)^{i+1}}{(i+1)!}(1+\beta\Delta t)^{n-i-1}\prod_{j=0}^{i}(n-j).$$

This result guarantees that as the iteration progresses, the method provides an increasingly accurate solution. Furthermore, when $i = n$, the last product on the right-hand side vanishes, which corresponds to the worst-case scenario of the sequential solution, a.k.a. at the nth iterate, the above idealized Parareal method replicates the analytical solution for the time subintervals up to t_n.

We take advantage of this convergence result in Section 4, constructing the coarse propagator as a neural network satisfying the assumptions of Theorem 1.

3 A-posteriori error estimate for neural network-based solvers

We aim to design a hybrid parallel-in-time solver for (1) based on the Parareal algorithm. This procedure consists of the Parareal iteration where the coarse propagator $\varphi_C^{\Delta t}$ is replaced by a neural network. In Section 4, we will focus on a particular class of neural networks, called Extreme Learning Machines (ELMs). For now, however, we do not specify the structure of the neural network and define it as a map $\mathcal{N}_\theta : [0, \Delta t] \times \mathbb{R}^d \to \mathbb{R}^d$, parametrized by weights θ, and satisfying the initial condition of the ODE, $\mathcal{N}_\theta(0; \mathbf{x}_0) = \mathbf{x}_0$.

In the classical Parareal iteration, the coarse propagator $\varphi_C^{\Delta t_n}$ is a map satisfying $\mathbf{x}(t_{n+1}) \approx \varphi_C^{\Delta t_n}(\mathbf{x}(t_n))$, where $\mathbf{x}(t)$ solves (1). The coarse propagator balances the cost versus accuracy of the approximation, with the sweet spot yielding optimal parallel speedup. With this in mind, we design our replacement to be a continuous function of time and to allow longer steps than commonly taken by single-step numerical methods as employed by $\varphi_C^{\Delta t_n}$. Motivated by collocation methods [1, Chapter II.7], we choose the weights of the neural network $\mathcal{N}_\theta$ so that it satisfies the differential equation (1) at some collocation points in the interval $[0, \Delta t]$. More explicitly, given a set $\{t_1, \dots, t_C\} \subset [0, \Delta t]$, we look for a set of weights θ minimizing the loss function

$$\mathcal{L}(\theta, \mathbf{x}_0) := \sum_{c=1}^{C} \left\|\mathcal{N}_\theta'(t_c; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(t_c; \mathbf{x}_0))\right\|_2^2. \tag{6}$$

Consistent with our convention, in (6) $'$ denotes the time derivative, i.e., the derivative with respect to the first component.

In the following, we propose an error analysis for the approximate solution $\mathcal{N}_\theta$. This error analysis allows us to provide a-posteriori theoretical guarantees on both, the accuracy of the network $\mathcal{N}_\theta(\Delta t; \mathbf{x}_0)$ as a continuous approximation of the solution, as well as its potential as a replacement of $\varphi_C^{\Delta t}(\mathbf{x}_0)$ while keeping intact the convergence guarantees of Parareal. We focus on a practical error estimate based on quadrature rules. For an, albeit less practical, alternative estimate based on defect control see Appendix A.

Assumption 1. *Assume that the collocation points $\{t_1, \dots, t_C\} \subset [0, \Delta t]$, with $t_1 < \dots < t_C$, define a Lagrange quadrature rule exact up to order p for some given $p \geq 1$, i.e., there is a set of weights $\rho_1, \dots, \rho_C$ for which*

$$\int_0^{\Delta t} f(t)\,\mathrm{d}t = \sum_{c=1}^{C} \rho_c f(t_c) =: \mathcal{I}_p(f; 0, \Delta t), \ \forall f \in \mathbb{P}_{p-1}, \tag{7}$$

where $\mathbb{P}_{p-1}$ is the set of real polynomials of degree $p-1$.

For a set of collocation points satisfying Assumption 1 and any scalar $p-$times continuously differentiable function $f \in \mathcal{C}^p(\mathbb{R}, \mathbb{R})$, it holds [29, Chapter 9]

$$\left|\mathcal{I}_p(f; 0, \Delta t) - \int_0^{\Delta t} f(t)\,\mathrm{d}t\right| \leq \kappa(\Delta t)^{p+1} \max_{\xi \in [0, \Delta t]} \left|f^{(p)}(\xi)\right|, \ \kappa > 0, \tag{8}$$

where $f^{(p)}$ is the derivative of f of order p.

We can now formulate a quadrature-based a-posteriori error estimate for the *continuous* approximation $\mathcal{N}_\theta(t; \mathbf{x}_0)$ that only requires the defect to be sufficiently small at the collocation points.

Theorem 2 (Quadrature-based a-posteriori error estimate)**.** *Let $\mathbf{x}(t)$ be the solution of the initial value problem (1) with $\mathcal{F} \in \mathcal{C}^{p+1}(\mathbb{R}^d, \mathbb{R}^d)$. Suppose that Assumption 1 on the C collocation points $0 \leq t_1 < \dots < t_C \leq \Delta t$ is satisfied and assume that $\mathcal{N}_\theta(\cdot; \mathbf{x}_0) : [0, \Delta t] \to \mathbb{R}^d$ is smooth and satisfies the collocation conditions up to some error of magnitude ε, i.e.*

$$\left\|\mathcal{N}_\theta'(t_c; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(t_c; \mathbf{x}_0))\right\|_2 \leq \varepsilon, \ c = 1, \dots, C. \tag{9}$$

Then, there exist two constants $\alpha, \beta > 0$ such that, for all $t \in [0, \Delta t]$,

$$\left\|\mathbf{x}(t) - \mathcal{N}_\theta(t; \mathbf{x}_0)\right\|_2 \leq \alpha(\Delta t)^{p+1} + \beta\varepsilon. \tag{10}$$

The proof of Theorem 2 is based on the Gröbner-Alekseev formula [1, Theorem 14.5] that we now state for completeness.

Theorem 3 (Gröbner-Alekseev)**.** *For $\mathcal{F} \in \mathcal{C}^1(\mathbb{R}^d, \mathbb{R}^d)$ and $\mathcal{G} : \mathbb{R}^d \to \mathbb{R}^d$ consider the solutions $\mathbf{x}(t)$ and $\mathbf{y}(t)$ of the two ODEs*

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)), \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \qquad \begin{cases} \mathbf{y}'(t) = \mathcal{F}(\mathbf{y}(t)) + \mathcal{G}(\mathbf{y}(t)), \\ \mathbf{y}(0) = \mathbf{x}_0, \end{cases}$$

assuming they both have a unique solution. For any times $0 \leq s \leq t$, let $\phi_{\mathcal{F}}^{s,t}(\mathbf{y}(s))$ be the exact solution of the initial value problem

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)), \\ \mathbf{x}(s) = \mathbf{y}(s). \end{cases}$$

Then, for any $t \geq 0$, one has

$$\mathbf{y}(t) = \mathbf{x}(t) + \int_0^t \left. \frac{\partial \phi_{\mathcal{F}}^{s,t}(\mathbf{x}_0)}{\partial \mathbf{x}_0} \right|_{\mathbf{x}_0 = \mathbf{y}(s)} \mathcal{G}(\mathbf{y}(s)) \mathrm{d}s. \tag{11}$$

We now prove the a-posteriori error estimate in Theorem 2 using Theorem 3.

Proof of Theorem 2. Let $\mathbf{x}(t)$ be the solution of the initial value problem (1). Further note that $t \mapsto \mathcal{N}_\theta(t; \mathbf{x})$ satisfies the initial value problem

$$\begin{cases} \mathcal{N}_\theta'(t; \mathbf{x}_0) = \mathcal{F}(\mathcal{N}_\theta(t; \mathbf{x}_0)) + \left[\mathcal{N}_\theta'(t; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(t; \mathbf{x}_0))\right], \\ \mathcal{N}_\theta(0; \mathbf{x}_0) = \mathbf{x}_0. \end{cases}$$

Setting $\mathcal{G}(\mathcal{N}_\theta(t; \mathbf{x}_0)) = \mathcal{N}_\theta'(t; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(t; \mathbf{x}_0))$, from (11) we obtain

$$\begin{aligned} \|\mathbf{x}(t) - \mathcal{N}_\theta(t; \mathbf{x}_0)\|_2 &= \left\| \int_0^t \left. \frac{\partial \phi_{\mathcal{F}}^{s,t}(\mathbf{y}_0)}{\partial \mathbf{y}_0} \right|_{\mathbf{y}_0 = \mathcal{N}_\theta(s; \mathbf{x}_0)} \mathcal{G}(\mathcal{N}_\theta(s; \mathbf{x}_0)) \, \mathrm{d}s \right\|_2 \\ &\leq \delta \left\| \int_0^t \left[\mathcal{N}_\theta'(s; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(s; \mathbf{x}_0))\right] \mathrm{d}s \right\|_2, \end{aligned}$$

where $0 < \delta < \infty$ bounds the norm of the Jacobian matrix of $\phi_{\mathcal{F}}^{s,t}$ for $0 \leq s \leq t \leq \Delta t$ by virtue of $\mathcal{F} \in \mathcal{C}^1(\mathbb{R}^d, \mathbb{R}^d)$. Approximating the integral with the quadrature and subsequently bounding the residual at the collocation points, we obtain

$$\begin{aligned} \|\mathbf{x}(t) - \mathcal{N}_\theta(t; \mathbf{x}_0)\|_2 &\leq \delta \left(\left\| \sum_{c=1}^C \rho_c \left[\mathcal{N}_\theta'(t_c; \mathbf{x}_0) - \mathcal{F}(\mathcal{N}_\theta(t_c; \mathbf{x}_0))\right] \right\|_2 + \bar{\kappa}(\Delta t)^{p+1} \right) \\ &\leq \delta \left(\varepsilon \sum_{c=1}^C |\rho_c| + \bar{\kappa}(\Delta t)^{p+1} \right), \end{aligned}$$

where $t \in [0, \Delta t]$ and

$$\bar{\kappa} := \kappa \cdot \left(\max_{t\in[0,\Delta t]} \left\| \frac{d^p}{dt^p} \left[\mathcal{N}_\theta' \left(t; \mathbf{x}_0\right) - \mathcal{F}\left(\mathcal{N}_\theta \left(t; \mathbf{x}_0\right)\right)\right] \right\|_2 \right) > 0,$$

the right-hand side of (8). To conclude the proof we set $\alpha = \bar{\kappa}\delta$, $\beta = \delta \sum_{c=1}^{C} |\rho_c|$. □

While for the proof it suffices that δ is finite, more practical bounds based on the one-sided Lipschitz constant of the vector field can be obtained. We derive such a bound in Appendix B.

Given $\beta\varepsilon \ll (\Delta t)^{p+1}$, Theorem 2 implies that the approximation provided by the neural network is as accurate as the one provided by a pth order one-step method with step size Δt. This result allows us to replace the coarse integrator $\varphi_C^{\Delta t}$ with a neural network-based solver maintaining the convergence properties of Parareal.

4 Parareal method based on Extreme Learning Machines

The theoretical results presented in Section 3 hold for generic continuous approximate solutions, particularly those provided by any neural network $\mathcal{N}_\theta$. We now restrict the neural architecture to Extreme Learning Machines (ELMs) which allow a more efficient, hence faster, solution of the optimization problem (6) as we will highlight in Section 6.

4.1 Architecture design

ELMs are feedforward neural networks composed of two layers, with trainable parameters confined to the outermost layer. We draw the weights of the first layer from the continuous uniform distribution $\mathcal{U}(\underline{\omega}, \overline{\omega})$, for a lower bound $\underline{\omega}$ and an upper bound $\overline{\omega}$ which are set to -1 and 1, respectively, in all our experiments. We then aim to approximate the solution of (1) at a time t with the parametric function

$$\begin{aligned}\mathcal{N}_\theta \left(t; \mathbf{x}_0\right) &= \mathbf{x}_0 + \sum_{h=1}^{H} \mathbf{w}_h \left(\varphi_h\left(t\right) - \varphi_h\left(0\right)\right) = \mathbf{x}_0 + \sum_{h=1}^{H} \mathbf{w}_h \left(\sigma\left(a_h t + b_h\right) - \sigma\left(b_h\right)\right) \\ &= \mathbf{x}_0 + \theta^\top \left(\sigma\left(\mathbf{a}\, t + \mathbf{b}\right) - \sigma\left(\mathbf{b}\right)\right), \ \mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_H \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_H \end{bmatrix} \in \mathbb{R}^H, \ \theta = \begin{bmatrix} \mathbf{w}_1^\top \\ \vdots \\ \mathbf{w}_H^\top \end{bmatrix} \in \mathbb{R}^{H\times d},\end{aligned} \tag{12}$$

by training the weights collected in the matrix θ. Here, $\varphi_h(t) = \sigma(a_h t + b_h) \in \mathbb{R}$, $h = 1, \ldots, H$, is a given set of basis functions with $a_h, b_h \sim \mathcal{U}(\underline{\omega}, \bar{\omega})$, and $\sigma \in \mathcal{C}^\infty(\mathbb{R}, \mathbb{R})$ a smooth activation function. In the numerical experiments, we always choose $\sigma(\cdot) = \tanh(\cdot)$. The architecture in (12) satisfies the initial condition of (1), i.e., $\mathcal{N}_\theta\left(0; \mathbf{x}_0\right) = \mathbf{x}_0$. In addition, $t \mapsto \mathcal{N}_\theta\left(t; \mathbf{x}_0\right)$ and σ have the same regularity.

4.2 Algorithm design

Our method closely mimics the Parareal algorithm, but with the network (12) deployed as a coarse propagator in the Parareal update (3). While in the classical Parareal algorithm the coarse propagator $\varphi_C^{\Delta t_n}$ is assumed to be known for all sub-intervals and Parareal iterations, we do not make this assumption in our approach. Instead, we determine individual weights for the update of each of the initial conditions featuring in the update (3) to allow for a better adaptation to the local behavior of the approximated solution. Furthermore, our neural coarse integrator $\varphi_C^{\Delta t_n}$ is not known ahead of time but is recovered and changing in the course of the Parareal iteration. Learning the coarse integrator involves training an ELM on each of the sub-intervals to solve the ODE (1) at a set of fixed collocation points in the sub-interval. This procedure would be prohibitively expensive for generic neural networks trained with gradient-based methods. However, for ELMs, estimating the matrix θ in (12) is considerably cheaper and comparable with classical collocation approaches, striking a balance between the computational efficiency, desirable behavior, and flexibility of the integrator. Finally, in Section 5 we demonstrate that our approach is provably convergent.

4.3 Training strategy

The neural coarse propagator for solution (1) on the time interval $[0, T]$ is obtained by splitting the interval into N sub-intervals, $\Delta t_n = t_{n+1} - t_n$, $t_0 = 0 < t_1 < \cdots < t_N = T$, and training N individual ELMs in sequence. On the nth sub-interval $[t_n, t_{n+1}]$ we train an ELM of the form (12) to solve the ODE system (1) approximately on this sub-interval. The initial condition at time t_n is obtained by the evaluation of the previous Parareal correction step. Since the vector field $\mathcal{F}$ in (1) does not explicitly depend on the time variable, we can restrict our presentation to a solution on a sub-interval $[0, \Delta t_n]$

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)), \\ \mathbf{x}(0) = \mathbf{x}_n^i, \end{cases} \tag{13}$$

where the superscript i refers to ith Parareal iterate.

To train an ELM (12) on the sub-interval $[0, \Delta t_n]$, we introduce C collocation points $0 \leq t_{n,1} < \cdots < t_{n,C} \leq \Delta t_n$, where the subscript n keeps track of the interval length Δt_n emphasizing the independent choice of collocation points on each sub-interval. For a given initial condition $\mathbf{x}_n^i$, we find a matrix θ_n^i such that $\mathcal{N}_{\theta_n^i}$ approximately satisfies (13) for all $t_{n,c}$, $c = 1, \ldots, C$, by solving the optimization problem

$$\theta_n^i = \arg \min_{\theta \in \mathbb{R}^{H \times d}} \sum_{c=1}^{C} \left\| \mathcal{N}_\theta' \left(t_{n,c}; \mathbf{x}_n^i \right) - \mathcal{F} \left(\mathcal{N}_\theta \left(t_{n,c}; \mathbf{x}_n^i \right) \right) \right\|_2^2 . \tag{14}$$

This hybrid Parareal method returns approximations of the solution at the time nodes $t_0, ..., t_N$, which we call $\mathbf{x}_0, ..., \mathbf{x}_N$. Furthermore, since the ELMs on sub-intervals are smooth functions of time, one could also access a piecewise smooth approximation of the curve $[0,T] \ni t \mapsto \mathbf{x}(t)$ by evaluating the individual ELMs upon convergence of the Parareal iteration

$$\tilde{\mathbf{x}}\left(t\right) = \mathcal{N}_{\theta_n}\left(t - t_n;\mathbf{x}_n\right),\ t \in \left[t_n, t_{n+1}\right),\ n = 0,\dots,N-1. \tag{15}$$

Here, θ_n and $\mathbf{x}_n$ are the weight matrix and the initial condition at the time t_n in the final Parareal iteration. Note that even though the points $\mathbf{x}_n^i$ are updated in each Parareal iteration (3), they do not tend to change drastically, and we can initialize θ_n^{i+1} in (14) with the previous iterate θ_n^i to speedup convergence. We terminate the Parareal iteration when either the maximum number of iterations is reached, or the difference between two consecutive iterates satisfies a given tolerance.

Algorithm 1 Hybrid Parareal algorithm based on ELMs

```
1:  Inputs : $\mathbf{x}_0$, tol, max_it
2:  error $\leftarrow$ tol $+ 1$, $i \leftarrow 1$, $\mathbf{x}_0^0 \leftarrow \mathbf{x}_0$
3:  for $n = 0$ to $N-1$ do                                        ▷ Zeroth iterate
4:      Find $\theta_n^0 = \arg\min_{\theta\in\mathbb{R}^{H\times d}} \sum_{c=1}^{C} \left\|\mathcal{N}_\theta'\left(t_{n,c};\mathbf{x}_n^0\right) - \mathcal{F}\left(\mathcal{N}_\theta\left(t_{n,c};\mathbf{x}_n^0\right)\right)\right\|_2^2$
5:      $\mathbf{x}_{n+1}^0 \leftarrow \mathcal{N}_{\theta_n^0}\left(\Delta t_n;\mathbf{x}_n^0\right)$, $\mathbf{x}_{n+1}^{S,-1} \leftarrow \mathcal{N}_{\theta_n^0}\left(\Delta t_n;\mathbf{x}_n^0\right)$
6:  end for
7:  while $i <$ max_it and error $>$ tol do
8:      error $\leftarrow 0$
9:      for $n = 0$ to $N-1$ do                      ▷ Fine integrator, Parallel For Loop
10:         $\mathbf{x}_{n+1}^F \leftarrow \varphi_F^{\Delta t_n}\left(\mathbf{x}_n^{i-1}\right)$
11:     end for
12:     $\mathbf{x}_0^i \leftarrow \mathbf{x}_0$
13:     for $n = 0$ to $N-1$ do
14:         Find $\theta_n^i = \arg\min_{\theta\in\mathbb{R}^{H\times d}} \sum_{c=1}^{C} \left\|\mathcal{N}_\theta'\left(t_{n,c};\mathbf{x}_n^i\right) - \mathcal{F}\left(\mathcal{N}_\theta\left(t_{n,c};\mathbf{x}_n^i\right)\right)\right\|_2^2$
15:         $\mathbf{x}_{n+1}^S \leftarrow \mathcal{N}_{\theta_n^i}\left(\Delta t_n;\mathbf{x}_n^i\right)$         ▷ Next coarse approximation
16:         $\mathbf{x}_{n+1}^i \leftarrow \mathbf{x}_{n+1}^F + \mathbf{x}_{n+1}^S - \mathbf{x}_{n+1}^{S,-1}$                ▷ Parareal correction
17:         $\mathbf{x}_{n+1}^{S,-1} \leftarrow \mathbf{x}_{n+1}^S$
18:         error$\leftarrow \max\left\{\text{error}, \left\|\mathbf{x}_{n+1}^i - \mathbf{x}_{n+1}^{i-1}\right\|_2\right\}$
19:     end for
20:     $i \leftarrow i + 1$
21: end while
22: return $\left\{\mathbf{x}_0^{i-1},\dots,\mathbf{x}_N^{i-1}\right\}$, $\left\{\theta_0^{i-1},\dots,\theta_{N-1}^{i-1}\right\}$
```

4.4 Implementation details

Our hybrid Parareal method is described in Algorithm 1 and the Python code can be found in the associated GitHub repository[1]. The zeroth iterate of the method, starting in line 3, only relies on ELMs to get intermediate initial conditions $\mathbf{x}_n^0$, $n =$

[1] https://github.com/davidemurari/ELMHybridParareal

$0, \dots, N-1$. These initial conditions are then used to solve with the fine integrators $\varphi_F^{\Delta t_n}$ the N initial value problems in parallel, see line 10. These approximations are subsequently updated in the Parareal correction step of line 16.

The Algorithm 1 relies on solving a *nonlinear* optimization problem in lines 4 and 14 to update the weights θ_n^i. For all systems studied in Section 6 but the Burgers' equation, we use the Levenberg–Marquardt algorithm [30, Chapter 10]. For Burgers' equation, we rely on the Trust Region Reflective algorithm [31] to exploit the sparsity of the Jacobian matrix. The optimization algorithms are implemented with the `least_squares` method of `scipy.optimize`. In both cases, we provide an analytical expression of the Jacobian of the loss function with respect to the weight θ, derived in Appendix C. Additionally, all the systems but the ROBER problem are solved on a uniform grid, i.e., $t_n = nT/N$. For the ROBER problem, we work with a non-uniform grid, refined in $[0, 1]$, to capture the spike in the solution occurring at small times.

As common in neural network-based approaches for solving differential equations, see, e.g. [28], we opt for C equispaced collocation points in each time interval. We also tested Lobatto quadrature points in the Lorenz example in subsection 6.3. In all experiments, we set $C = 5$ and the number of hidden neurons $H = C = 5$ to match.

5 Convergence of the ELM-based Parareal method

In this section, we study the convergence properties of Algorithm 1. Following the Parareal analysis in Theorem 1 we only need to consider the time interval $[0, \Delta t]$ and collocation points $0 < t_1 < \cdots < t_C < \Delta t$ satisfying Assumption 1.

We write our solution ansatz, (12), and its time derivative evaluated at the collocation points as the matrices

$$\tilde{\mathbf{X}}_\theta(\mathbf{x}, \Delta t) = \begin{bmatrix} \mathcal{N}_\theta(t_1; \mathbf{x})^\top \\ \vdots \\ \mathcal{N}_\theta(t_C; \mathbf{x})^\top \end{bmatrix} \in \mathbb{R}^{C \times d}, \quad \tilde{\mathbf{X}}'_\theta(\mathbf{x}, \Delta t) = \begin{bmatrix} \mathcal{N}'_\theta(t_1; \mathbf{x})^\top \\ \vdots \\ \mathcal{N}'_\theta(t_C; \mathbf{x})^\top \end{bmatrix} \in \mathbb{R}^{C \times d},$$

and shorthand the evaluation of the vector field $\mathcal{F}$ on the rows of the matrix $\mathbf{X} \in \mathbb{R}^{C \times d}$

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} \mathcal{F}\left(\mathbf{X}^\top \mathbf{e}_1\right)^\top \\ \vdots \\ \mathcal{F}\left(\mathbf{X}^\top \mathbf{e}_C\right)^\top \end{bmatrix} \in \mathbb{R}^{C \times d},$$

with $\mathbf{e}_1, \dots, \mathbf{e}_C \in \mathbb{R}^C$ the canonical basis of $\mathbb{R}^C$.

We further rewrite the ansatz as $\tilde{\mathbf{X}}_\theta(\mathbf{x}, \Delta t) = \mathbf{1}_C \mathbf{x}^\top + (\mathbf{H} - \mathbf{H}_0)\theta$, where $\mathbf{1}_C = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^\top \in \mathbb{R}^C$,

$$\mathbf{H} = \begin{bmatrix} \sigma\left(\mathbf{a}^\top t_1 + \mathbf{b}^\top\right) \\ \vdots \\ \sigma\left(\mathbf{a}^\top t_C + \mathbf{b}^\top\right) \end{bmatrix} \in \mathbb{R}^{C \times H}, \quad \mathbf{H}' = \begin{bmatrix} \sigma'\left(\mathbf{a}^\top t_1 + \mathbf{b}^\top\right) \odot \mathbf{a}^\top \\ \vdots \\ \sigma'\left(\mathbf{a}^\top t_C + \mathbf{b}^\top\right) \odot \mathbf{a}^\top \end{bmatrix} \in \mathbb{R}^{C \times H},$$

with $\mathbf{a}^\top = \begin{bmatrix} a_1 & \cdots & a_H \end{bmatrix}$, $\mathbf{b}^\top = \begin{bmatrix} b_1 & \cdots & b_H \end{bmatrix}$, and $\sigma \in \mathcal{C}^\infty(\mathbb{R}, \mathbb{R})$ evaluated componentwise while $\odot$ denotes the componentwise product. As for the experiments, we restrict ourselves to the case $C = H$ for which one can prove, see [20, Theorem 2.1], that with probability one $\mathbf{H}$ and $\mathbf{H}'$ are invertible for $\mathbf{a}, \mathbf{b}$ drawn from any continuous probability distribution. Finally, $\mathbf{H}_0 = \mathbf{1}_C \sigma\left(\mathbf{b}^\top\right) \in \mathbb{R}^{C\times H}$ in $\tilde{\mathbf{X}}_\theta\left(\mathbf{x}, \Delta t\right)$, accounts for the initial condition.

Theorem 4 (Existence and regularity of the solution)**.** *For the loss function*

$$\mathcal{L}(\theta, \mathbf{x}) := \left\| \tilde{\mathbf{X}}'_\theta\left(\mathbf{x}, \Delta t\right) - \mathbf{F}\left(\tilde{\mathbf{X}}_\theta\left(\mathbf{x}, \Delta t\right)\right)\right\|_F^2 \tag{16}$$

with $\mathcal{N}_\theta$ in $\tilde{\mathbf{X}}_\theta$ defined as in (12), σ a smooth $1-$Lipschitz activation function, and a choice of step size

$$\Delta t \in \left(0, \left(\left\|\left(\mathbf{H}'\right)^{-1}\right\|_2 \mathrm{Lip}(\mathcal{F})\sqrt{C}\|\mathbf{a}\|_2\right)^{-1}\right), \tag{17}$$

there exists a unique Lipschitz continuous function $\mathbb{R}^d \ni \mathbf{x} \mapsto \theta(\mathbf{x}) \in \mathbb{R}^{H\times d}$ such that $\mathcal{L}(\theta(\mathbf{x}), \mathbf{x}) = 0$ for every $\mathbf{x} \in \mathbb{R}^d$.

We remark that the loss function in (16) using the Frobenius norm $\|\cdot\|_F$ is a reformulation of (6) in a matrix form. We now prove Theorem 4 using a parameterized version of Banach Contraction Theorem presented in [32, Lemma 1.9].

Proof. The requirement $\mathcal{L}(\theta, \mathbf{x}) = 0$ implies that the ansatz $\tilde{\mathbf{X}}_\theta\left(\mathbf{x}, \Delta t\right) = \mathbf{1}_C\mathbf{x}^\top + (\mathbf{H} - \overline{\mathbf{H}})\theta$ and its derivative, $\tilde{\mathbf{X}}'_\theta\left(\mathbf{x}, \Delta t\right) = \mathbf{H}'\theta$, satisfy the ODE (13), $\tilde{\mathbf{X}}'_\theta\left(\mathbf{x}, \Delta t\right) = \mathbf{F}\left(\tilde{\mathbf{X}}_\theta\left(\mathbf{x}, \Delta t\right)\right)$, which can be equivalently written as $\mathbf{H}'\theta = \mathbf{F}\left(\mathbf{1}_C\mathbf{x}^\top + \left(\mathbf{H} - \overline{\mathbf{H}}\right)\theta\right)$. We introduce the fixed point map

$$T\left(\theta, \mathbf{x}\right) = \left(\mathbf{H}'\right)^{-1}\mathbf{F}\left(\mathbf{1}_C\mathbf{x}^\top + \left(\mathbf{H} - \overline{\mathbf{H}}\right)\theta\right) \in \mathbb{R}^{H\times d},$$

and, when not differently specified, we denote with $\mathrm{Lip}(f)$ the Lipschitz constant of a Lipschitz continuous function f with respect to the ℓ^2 norm. Since $\mathrm{Lip}(\sigma) \leq 1$, we have

$$\left\|\mathbf{H} - \overline{\mathbf{H}}\right\|_F^2 = \sum_{c=1}^C \left\|\sigma\left(\mathbf{a}^\top t_c + \mathbf{b}^\top\right) - \sigma\left(\mathbf{b}^\top\right)\right\|_2^2 \leq C\|\mathbf{a}\|_2^2(\Delta t)^2$$

as $t_c \in (0, \Delta t)$. Furthermore,

$$\|\mathbf{F}(\mathbf{X}) - \mathbf{F}(\mathbf{Y})\|_F^2 = \sum_{c=1}^C \left\|\mathcal{F}(\mathbf{X}^\top\mathbf{e}_c) - \mathcal{F}(\mathbf{Y}^\top\mathbf{e}_c)\right\|_2^2 \leq \mathrm{Lip}\left(\mathcal{F}\right)^2\|\mathbf{X} - \mathbf{Y}\|_F^2$$

for any $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{C\times d}$. Setting $\ell_\theta = \left\|\left(\mathbf{H}'\right)^{-1}\right\|_2 \mathrm{Lip}(\mathcal{F})\sqrt{C}\|\mathbf{a}\|_2\Delta t$ we conclude that $T(\cdot, \mathbf{x})$ is Lipschitz continuous with constant $\ell_\theta < 1$ for Δt satisfying (17), as

$$\|T(\theta_2, \mathbf{x}) - T(\theta_1, \mathbf{x})\|_F \leq \left\|\left(\mathbf{H}'\right)^{-1}\right\|_2 \mathrm{Lip}(\mathcal{F})\sqrt{C}\|\mathbf{a}\|_2\Delta t\left\|\theta_2 - \theta_1\right\|_F = \ell_\theta\left\|\theta_2 - \theta_1\right\|_F$$

for any $\theta_1, \theta_2 \in \mathbb{R}^{H\times d}$. We note that the $2-$norm of $(\mathbf{H}')^{-1}$ can be used since for any pair of matrices $\mathbf{A}, \mathbf{B}$ of compatible dimensions, it holds $\|\mathbf{AB}\|_F \leq \|\mathbf{A}\|_2 \|\mathbf{B}\|_F$. Furthermore, $T(\theta, \cdot)$ is Lipschitz continuous with Lipschitz constant given by $\ell_{\mathbf{x}} = \left\|(\mathbf{H}')^{-1}\right\|_2 \mathrm{Lip}\left(\mathcal{F}\right)\sqrt{C}$, since

$$\begin{aligned}\|T\left(\theta, \mathbf{x}_2\right) - T\left(\theta, \mathbf{x}_1\right)\|_F &\leq \left\|(\mathbf{H}')^{-1}\right\|_2 \mathrm{Lip}\left(\mathcal{F}\right) \left\|\mathbf{1}_C \left(\mathbf{x}_2 - \mathbf{x}_1\right)^\top\right\|_F, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d \\ &= \left\|(\mathbf{H}')^{-1}\right\|_2 \mathrm{Lip}\left(\mathcal{F}\right)\sqrt{C}\left\|\mathbf{x}_2 - \mathbf{x}_1\right\|_2 = \ell_{\mathbf{x}} \left\|\mathbf{x}_2 - \mathbf{x}_1\right\|_2.\end{aligned}$$

By [32, Lemma 1.9], we can hence conclude that, provided Δt satisfies (17), there is a well-defined Lipschitz continuous function $\theta : \mathbb{R}^d \to \mathbb{R}^{H\times d}$, with

$$\mathrm{Lip}\left(\theta\right) \leq \frac{\ell_{\mathbf{x}}}{1 - \ell_\theta} = \frac{\left\|(\mathbf{H}')^{-1}\right\|_2 \mathrm{Lip}\left(\mathcal{F}\right)\sqrt{C}}{1 - \left\|(\mathbf{H}')^{-1}\right\|_2 \mathrm{Lip}(\mathcal{F})\sqrt{C}\|\mathbf{a}\|_2 \Delta t},$$

such that $\theta(\mathbf{x}) = T(\theta(\mathbf{x}), \mathbf{x})$ for every $\mathbf{x} \in \mathbb{R}^d$, or equivalently $\mathcal{L}(\theta(\mathbf{x}), \mathbf{x}) = 0$. □

Proposition 1 (Convergence of the hybrid Parareal method)**.** *Consider the initial value problem* (1) *with* $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$ *a smooth Lipschitz continuous vector field. Suppose that the time interval* $[0, T]$ *is partitioned into* N *intervals of size* $\Delta t = T/N$ *such that* Δt *satisfies* (17) *and choose the* C *collocation points* $0 \leq t_1 < \dots < t_C \leq \Delta t$ *to satisfy the Assumption 1. Let* σ *be a smooth* $1-$*Lipschitz function. Then for the coarse integrator* $\varphi_C^{\Delta t}(\mathbf{x}) = \mathbf{x} + \theta(\mathbf{x})^T\left(\sigma(\mathbf{a}\Delta t + \mathbf{b}) - \sigma(\mathbf{b})\right)$ *with* $\theta(\mathbf{x})$ *as in Theorem 4 and the fine integrator* $\varphi_F^{\Delta t} = \phi_{\mathcal{F}}^{\Delta t}$, *there exist positive constants* α, γ, β *such that, at the* $i-$*th iterate of the hybrid Parareal method, the following bound holds*

$$\left\|\mathbf{x}(t_n) - \mathbf{x}_n^i\right\|_2 \leq \frac{\alpha}{\gamma} \frac{\left(\gamma(\Delta t)^{p+1}\right)^{i+1}}{(i+1)!} \left(1 + \beta\Delta t\right)^{n-i-1} \prod_{j=0}^{i} \left(n - j\right). \tag{18}$$

Proof. The proof is based on showing that our network satisfies assumptions (4) and (5) of Theorem 1. Theorem 4 guarantees that, for Δt satisfying (17), $\mathbf{x} \mapsto \theta(\mathbf{x})$ is Lipschitz continuous with Lipschitz constant $\mathrm{Lip}\left(\theta\right)$. Further noting that $\|\sigma(\mathbf{a}\Delta t + \mathbf{b}) - \sigma(\mathbf{b})\|_2 \leq \|\mathbf{a}\|_2 \Delta t$ as $\mathrm{Lip}(\sigma) \leq 1$ we can write

$$\begin{aligned}\left\|\varphi_C^{\Delta t}\left(\mathbf{x}_2\right) - \varphi_C^{\Delta t}\left(\mathbf{x}_1\right)\right\|_2 &\leq \left\|\mathbf{x}_2 - \mathbf{x}_1\right\|_2 + \|\mathbf{a}\|_2 \,\Delta t \,\mathrm{Lip}\left(\theta\right)\left\|\mathbf{x}_2 - \mathbf{x}_1\right\|_2 \\ &= \left(1 + \beta\Delta t\right)\left\|\mathbf{x}_2 - \mathbf{x}_1\right\|_2\end{aligned}$$

for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, where $\beta := \mathrm{Lip}\left(\theta\right)\|\mathbf{a}\|_2$, and hence condition (5) is satisfied. Given that $\theta(\mathbf{x}) = T(\theta(\mathbf{x}), \mathbf{x})$, as guaranteed by Theorem 4, satisfies the collocation conditions exactly, Theorem 2 ensures that there exists $\alpha > 0$ such that $\left\|\varphi_F^{\Delta t}(\mathbf{x}) - \varphi_C^{\Delta t}(\mathbf{x})\right\|_2 \leq \alpha\left(\Delta t\right)^{p+1}$. Because of the smoothness of $\mathcal{F}$ and σ, one can also Taylor expand in time and guarantee the existence of continuously differentiable functions $c_{p+1}, c_{p+2}, \ldots$ such that

$$\phi_{\mathcal{F}}^{\Delta t}(\mathbf{x}) - \mathcal{N}_\theta\left(\Delta t; \mathbf{x}\right) = c_{p+1}(\mathbf{x})(\Delta t)^{p+1} + c_{p+2}(\mathbf{x})(\Delta t)^{p+2} + \cdots .$$

This allows concluding that (4) holds, and the hybrid Parareal satisfies (18). □

As for the classical Parareal method, at the nth iterate, our hybrid Parareal method with the exact fine integrator replicates the analytical solution at the time instants $t_0, \ldots, t_n$.

In practice, as presented in the previous section, we do not have access to the function $\mathbf{x} \mapsto \theta(\mathbf{x})$, but we only approximate its value at the points involved in the hybrid Parareal iterates, i.e., $\theta_n^i \approx \theta(\mathbf{x}_n^i)$. Let us denote by $\hat{\theta} : \mathbb{R}^d \to \mathbb{R}^{H \times d}$ the function approximating θ, so that $\theta_n^i = \hat{\theta}(\mathbf{x}_n^i)$. This function is typically provided by a convergent iterative method minimizing (16). Under the smoothness assumptions of Proposition 1 and supposing the map $x \mapsto \hat{\theta}(x)$ is Lipschitz continuous, i.e., the adopted optimization method depends regularly on the parameter $\mathbf{x} \in \mathbb{R}^d$, the convergence in Proposition 1 also holds for the approximate case. To see this, note that condition (4) also holds for the approximate case as long as $\mathcal{F}$ is smooth enough and the collocation conditions are solved sufficiently accurately. In practice, based on (9), it suffices to have

$$\max_{c=1,\ldots,C} \left\| \left(\tilde{\mathbf{X}}'_{\theta_n^i}\left(\mathbf{x}, \Delta t\right) - \mathbf{F}\left(\tilde{\mathbf{X}}_{\theta_n^i}\left(\mathbf{x}, \Delta t\right)\right) \right)^\top \mathbf{e}_c \right\|_2 \leq \tilde{\alpha}\left(\Delta t\right)^{p+1}$$

for an $\tilde{\alpha} > 0$, and every $n = 0, \ldots, N-1$ and iterate i. Furthermore, assumption (5) follows from the Lipschitz regularity of the approximate function $\hat{\theta}$.

6 Numerical results

This section collects several numerical tests that support our theoretical derivations. We consider six dynamical systems, four of which come from the experimental section in [6], to which we add the SIR model and the ROBER problem. We assume that, for each of these systems, a single initial value problem is of interest and explore how ELM-based coarse propagators perform for that initial value problem. For the one-dimensional Burgers' equation, we consider a semi-discretization with centered finite differences and provide the experimental results for different initial conditions, imposing homogeneous Dirichlet boundary conditions on the domain $[0, 1]$.

The chosen fine integrators are classical Runge–Kutta methods with a smaller timestep than the coarse one Δt. More explicitly, we assume that the coarse timestep Δt is a multiple of the fine timestep δt and one coarse integrator step Δt, corresponds to $\Delta t/\delta t$ steps of the size δt of the fine integrator. In all experiments, we use equispaced time collocation points, and for the Lorenz system, we also use Lobatto points. For stiff problems such as Burgers' and ROBER's, we use the implicit Euler method (IE), with update $\mathbf{x}_{n+1} = \mathbf{x}_n + \delta t \mathcal{F}(\mathbf{x}_{n+1})$, as a fine integrator, while for the others we found Runge-Kutta (RK4),

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{\delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

with

$$k_1 = \mathcal{F}\left(\mathbf{x}_n\right), \; k_2 = \mathcal{F}\left(\mathbf{x}_n + \delta t \frac{k_1}{2}\right), \; k_3 = \mathcal{F}\left(\mathbf{x}_n + \delta t \frac{k_2}{2}\right), \; k_4 = \mathcal{F}\left(\mathbf{x}_n + \delta t k_3\right).$$

to provide accurate solutions with moderately small step sizes. We specify the adopted timesteps in the dedicated sections below.

The purpose of this paper is to demonstrate that our hybrid Parareal method based on ELMs is theoretically motivated and practically effective, rather than the high-performance implementation. Thus, most of our experiments are run on a single processor where the parallel speedup would result from parallel execution of the fine integrators on the sub-intervals, in proportion to the number of cores used. To demonstrate the principle in hardware we run the ROBER's problem on five processors available to us and compare to the serial application of the fine integrator, however Parareal benefits will scale up with the problem size and number of cores. For Burgers' equation, we again use five processors for convenience since this allows us to do 100 repeated experiments faster.

In all plots, the label "para" refers to the hybrid methodology with neural networks as coarse propagators, while "ref" to the reference solution, obtained by the sequential application of the fine solver. We always plot the piecewise smooth Hybrid Parareal approximant constructed as (15). We run the Hybrid Parareal until the difference between two consecutive iterates was at most $\texttt{tol} = 10^{-4}$. As a safeguard, we put a hard limit, $\texttt{max\_it} = 20$, on the iteration number, which was, however, not triggered in any of our experiments. All experiments were run on a MacBook Pro 2020 with Intel i5 processor and all the computational times were averaged over 100 runs per experiment. For each experiment, we report an average time per update of the coarse integrator on a sub-interval, which is also averaged over the number of sub-intervals. We measure the timing when computing the zeroth iterate in lines 3-6 of Algorithm 1 to isolate the effects of warm starts used in Parareal update in later iterations. We also report a total average time to compute the solution, including the above mentioned coarse integrator updates along with possibly parallel execution of the fine step integrators.

6.1 SIR

The SIR model is one of the simplest systems considered in mathematical biology to describe the spread of viral infections. SIR consists of three coupled ODEs for

$\mathbf{x} = [x_1, x_2, x_3]^\top$ with parameters $\beta = 0.1$, and $\gamma = 0.1$:

$$\begin{cases} x_1'(t) = -\beta x_1(t)\, x_2(t)\,, \\ x_2'(t) = \beta x_1(t)\, x_2(t) - \gamma x_2(t)\,, \\ x_3'(t) = \gamma x_2(t)\,, \\ \mathbf{x}(0) = \begin{bmatrix} 0.3 \; 0.5 \; 0.2 \end{bmatrix}^\top . \end{cases} \tag{19}$$

We use this example to compare two different types of coarse propagators, the ELM-based approach with a neural operator-type *flow map* trained to approximate the solutions of the dynamical system described by (19) for initial conditions in the compact set $\Omega = [0, 1]^3$, and times in $[0, 1]$, see also [33, 34]. Given that the Parareal method needs to evaluate the coarse propagator on several initial conditions, the learned flow map is the most natural neural network-based alternative, while a standard Physics Informed Neural Network, which needs to be fitted for each initial condition, would be computationally too expensive. Both coarse propagators use the same coarse timestep $\Delta t = 1$, while the fine solver timestep is $\delta t = 10^{-2}$. The piecewise smooth approximations computed with both methods are plotted in Figure 1. We report the corresponding timings in Table 1.

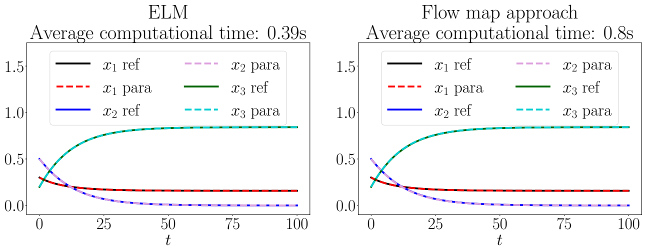


Fig. 1: SIR: Hybrid Parareal solution with (left) an ELM-based coarse propagator, (right) flow map coarse propagator.

Timing breakdown	ELM	Flow
Offline training phase	0s	∼20 minutes
Average cost coarse step in the zeroth iterate	0.0009773s	0.0002729s
Total	0.3940s	0.8047s

Table 1: SIR: Computational time for the ELM and flow map based Hybrid Parareal variants on a single core.

The ELM-based approach took an average of 0.3940 seconds to converge to the final solution over 100 repeated experiments, while the flow map approach took an average time of 0.8047 seconds. The reason behind the higher cost of the flow map approach is that ELMs minimize the residual more accurately than the flow map approach since they are trained for the specific initial conditions of interest, leading to a faster convergence of the Parareal method. If the offline training phase is accounted for, about 20 more minutes must be considered for the flow map approach, while no offline training is required for the ELM-based approach. The offline training cost depends on the chosen architecture and training strategy. These details are provided in Appendix D.

Given the reported results, it is clear that while both methods are comparable in terms of accuracy, the distribution of the costs is considerably different. The flow map approach has a high training cost and a low evaluation cost but is also less accurate hence needing more Parareal iterations. On the other hand, the ELM strategy, having no offline training phase and yielding more accurate solutions and hence needing fewer Parareal iterations, saves substantial time. For this reason, we will only focus on the ELM-based approach in the following experiments.

6.2 ROBER

The ROBER problem is a prototypical stiff system of coupled ODEs with parameters $k_1 = 0.04$, $k_2 = 3 \cdot 10^7$, and $k_3 = 10^4$,

$$\begin{cases} x_1'(t) = -k_1 x_1(t) + k_3 x_2(t) x_3(t), \\ x_2'(t) = k_1 x_1(t) - k_2 x_2^2(t) - k_3 x_2(t) x_3(t), \\ x_3'(t) = k_2 x_2^2(t), \\ \mathbf{x}(0) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top. \end{cases} \tag{20}$$

As ROBER's solution spikes for short times, the usual approach is to discretize the time non-uniformly. Therefore we choose the coarse step size to be $\Delta t = 10^{-2}$ for times in $[0, 1]$ and $\Delta t = 3$ for times in $[1, 100]$. The fine integrator timestep is $\delta t = 10^{-4}$. We remark that ROBER's problem is commonly solved using a variable step-size method, for example, based on an embedded Runge-Kutta method [1, Section II.4]. Fixing the step size allows us to understand how the proposed hybrid method performs on stiff equations without extra complication of step adaptivity. A variable step Parareal method (regardless if the coarse propagator is learned or classical), would involve adaptivity in both coarse and fine step and is beyond scope of this work.

Timing breakdown	ELM	Sequential IE, δt
Average cost coarse step in the zeroth iterate	0.001881s	
Average cost to produce the solution	179.8280s	263.2613s

Table 2: ROBER: Computational time for Hybrid Parareal using five cores versus sequential application of IE with fine step δt.

We report the obtained approximate solutions in Figure 2 and the timings in Table 2. In these experiments, the fine integrators were executed in parallel on five cores. Thus, the total average time to compute the solution reflects the parallel speed up, albeit for a small number of cores. Given this stiff problem requires an implicit fine integrator, we expect the computational costs of the update of the coarse integrator, 0.001881s, and one step of the fine integrator, 0.000263s, to be closer than when using

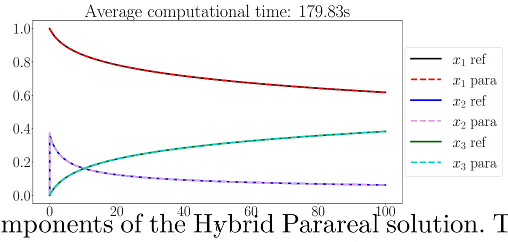


Fig. 2: ROBER: Components of the Hybrid Parareal solution. To plot all components on the same scale, $\mathbf{x}_2$ was scaled by a factor of 10^4.

an explicit scheme as it is the case in the remaining examples. Additionally, to cover one coarse step, the fine integrator needs to perform at least 100 steps, given our choices for δt and Δt. These respective costs help to optimally balance the choice of the number of sub-intervals versus the number of fine steps in each sub-interval, along with practical considerations like the number of cores available.

6.3 Lorenz

For weather forecasts, real-time predictions are paramount, rendering parallel-in-time solvers highly relevant in this context. Lorenz's equations

$$\begin{cases} x_1'(t) = -\sigma x_1(t) + \sigma x_2(t), \\ x_2'(t) = -x_1(t)\, x_3(t) + r x_1(t) - x_2(t), \\ x_3'(t) = x_1(t)\, x_2(t) - b x_3(t), \\ \mathbf{x}(0) = \begin{bmatrix} 20 & 5 & -5 \end{bmatrix}^\top, \end{cases} \tag{21}$$

describe one simple model for weather prediction. Different parameter values give rise to considerably different trajectories for this system. We set $\sigma = 10$, $r = 28$, and $b = 8/3$ to have chaotic behavior. We compute an approximate solution up to time $T = 10$, using ELMs as a coarse propagator with $\Delta T = T/250$ and RK4 with step $\delta t = T/14500$ as a fine integrator.

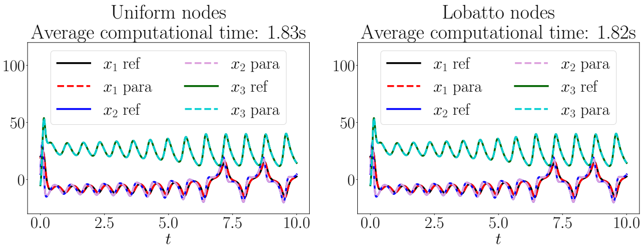


Fig. 3: Lorenz: Hybrid Parareal solution with (left) uniform collocation points, (right) Lobatto collocation points.

To show that the algorithm is not overly sensitive to the choice of the collocation points, we repeated the simulations using Lobatto collocation points. The qualitative behavior of the produced solutions for one choice of trained weights is reported in Figure 3 and the corresponding timings in Table 3. Although the Lorenz system is chaotic, the proposed hybrid solver provides an accurate approximate solution on the considered interval. Additionally, the average cost of one evaluation of the coarse ELM-based integrator does not appear to depend strongly on the system's complexity but mostly on its dimension. Indeed, the average cost of one ELM evaluation is comparable with the one for the SIR problem, see Table 1.

Timing breakdown	Uniform	Lobatto
Average cost coarse step in the zeroth iterate	0.0009430s	0.0009371s
Average cost to produce the solution	1.8312s	1.8184s

Table 3: Lorenz: Computational time for the ELM-based Hybrid Parareal with uniform and Lobatto nodes on a single core.

6.4 Arenstorf orbit

The three-body problem is a well-known problem in physics that pertains to the time evolution of three bodies interacting because of their gravitational forces. Changing the ratios between the masses, their initial conditions, and velocities, can starkly alter the system's time evolution, and many configurations have been thoroughly studied. One of them is the stable Arenstorf orbit, which arises when one of the masses is negligible and the other two masses orbit in a plane. The equations of motion for this specific instance of the three-body problem are

$$\begin{cases} x_1^{''}(t) = x_1(t) + 2x_2'(t) - b\frac{x_1+a}{D_1} - a\frac{x_1'(t)-b}{D_2}, \\ x_2^{''}(t) = x_2(t) - 2x_1'(t) - b\frac{x_2(t)}{D_1} - a\frac{x_2(t)}{D_2}, \\ \begin{bmatrix} x_1(0) & x_1'(0) & x_2(0) & x_2'(0) \end{bmatrix}^\top = \begin{bmatrix} 0.994 & 0 & 0 & v_2^0 \end{bmatrix}^\top, \end{cases} \tag{22}$$

$$D_1 = \left((x_1(t) + a)^2 + x_2(t)^2 \right)^{3/2}, \quad D_2 = \left((x_1(t) - b)^2 + x_2(t)^2 \right)^{3/2},$$

$v_2^0 = -2.00158510637908252240537862224$, $a = 0.12277471$, and $b = 1 - a$. This configuration leads to a periodic orbit of period 17.06521656015796 [1]. In practice, we transform (22) into a first order system via the velocity variables $v_1(t) := x_1'(t)$ and $v_2(t) := x_2'(t)$. We include the plot of the obtained solution for time up to $T = 17$ and timesteps $\Delta t = T/125$, and $\delta t = T/80000$, in Figure 4 and the timings in Table 4.

This experiment serves to illustrate the benefits of using a Parareal-like correction of the neural network-based solution. Indeed, the approximate solution for short times does not accurately follow the correct trajectory. One possible remedy would be to restrict the step size Δt as was done for the ROBER's problem. However, even for this larger time step choice, after just one step Δt, the Parareal correction resets the initial condition for the next interval bringing the solution back onto the stable orbit. Thus, not relying solely on a network-based solution allows us to compute an accurate solution for the later times, even though initially the solution departs the orbit.

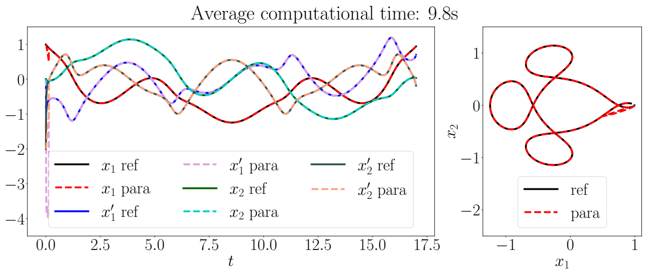


Fig. 4: Arenstorf: Components of the Hybrid Parareal solution (left), and the orbit of the initial condition (right).

Timing breakdown	ELM
Average cost coarse step in the zeroth iterate	0.001912s
Average cost to produce the solution	9.7957s

Table 4: Arenstorf: Computational time for Hybrid Parareal using a single core.

6.5 Viscous Burgers' equation

Most of the systems considered up to now are low-dimensional. A natural way to test the method's performance on higher-dimensional systems is to work with spatial semi-discretizations of PDEs, where the mesh over which the spatial discretization is defined determines the system's dimension. We consider the one-dimensional Burgers' equation

$$\begin{cases} \partial_t u\left(x,t\right) + u\left(x,t\right)\partial_x u\left(x,t\right) = \nu\partial_{xx} u\left(x,t\right), \quad x \in \Omega = [0,1], \\ u\left(x,0\right) = \sin\left(2\pi x\right), \quad x \in \Omega, \\ u\left(0,t\right) = u\left(1,t\right) = 0, \quad t \geq 0. \end{cases} \tag{23}$$

In this section, we only report the results for the initial condition in Equation (23), but we include results for two more choices of initial conditions in Appendix F. All the experiments were run on five cores. In all tests we work with viscosity parameter $\nu = 1/50$, a uniform spatial grid of 51 points in $\Omega = [0,1]$ and coarse and fine step sizes $\Delta t = 1/50$ and $\delta t = 1/500$, respectively. The spatial semi-discretization with centered finite differences writes

$$\begin{cases} \mathbf{u}'\left(t\right) = -\mathbf{u}\left(t\right) \odot \left(D_1\mathbf{u}\left(t\right)\right) + \nu D_2\mathbf{u}\left(t\right), \\ \mathbf{u}\left(0\right) = \sin\left(2\pi\mathbf{x}\right) \in \mathbb{R}^{51}, \end{cases}$$

where $\mathbf{x} = \begin{bmatrix} x_0 & x_1 & \dots & x_{50} \end{bmatrix}^\top$, $\mathbf{x}_i = i\Delta x$, $\Delta x = 1/50$, $i = 0, \dots, 50$, $\odot$ is the componentwise product, and $D_1, D_2 \in \mathbb{R}^{51\times 51}$ are the centered finite difference matrices of first and second order, respectively, suitably corrected to impose the homogeneous Dirichlet boundary conditions on $t \mapsto \mathbf{u}(t)$.

We report the qualitative behavior of the solutions in Figure 5. Subfigure (a) tracks the solution at ten equally spaced time instants in the interval $[0, T = 1]$. Subfigure

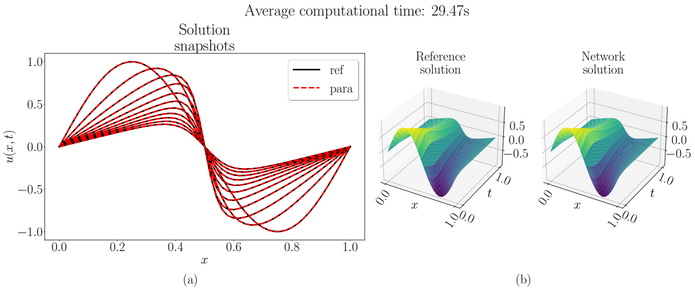


Fig. 5: Burgers: Snapshots of the solution obtained with Hybrid Parareal (left), comparison of the solution surfaces between Hybrid Parareal and the fine integrator applied serially (right). Solution corresponding to $u_0(x) = \sin(2\pi x)$.

(b) shows the solution surfaces obtained with the IE method on the left and the hybrid Parareal for one set of trained parameters on the right. We include the timings in Table 5. We observe that the cost of the presented hybrid Parareal method grows with the dimensionality d of the problem. However, we remark that for each of the 51 components of the solution, we adopted only $H = 5$ coefficients.

Timing breakdown	ELM
Average cost coarse step in the zeroth iterate	0.2098s
Average cost to produce the solution	29.4740s

Table 5: Burgers: Computational time for Hybrid Parareal using five cores, and initial condition $u_0(x) = \sin(2\pi x)$.

7 Conclusions and future extensions

In this manuscript, we proposed a hybrid parallel-in-time algorithm to solve initial value problems using a neural network as a coarse propagator within the Parareal framework. We derived an a-posteriori error estimate for generic neural network-based approximants. Based on these theoretical results we defined a hybrid Parareal algorithm involving ELMs as coarse propagators which inherits the theoretical guarantees of the Parareal algorithm.

We compared our hybrid Parareal solver based on ELMs with one based on the flow map approach on the SIR problem. We demonstrated that our approach led to lower computational costs and no offline training phase. We reserve the judgment of flow map performance. However, we also tested it for other examples, including the Brusselator, where we noticed that the offline training phase can be very intricate because one has to first identify a forward invariant subset Ω of $\mathbb{R}^d$.

The most promising extension of this work is to include a mechanism allowing for time-adaptivity in the algorithm, i.e., for coarsening or refinement of the temporal grid based on the local behavior of the solution. It would also be interesting to test our approach on higher-dimensional systems with high-performance computing hardware.

Acknowledgments

The authors would like to thank the Isaac Newton Institute for Mathematical Sciences for support and hospitality during the programme Mathematics of deep learning (MDL) when work on this paper was undertaken. This work was supported by: EPSRC grant number EP/R014604/1. This work was partially supported by a grant from the Simons Foundation (DM).

References

[1] Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I, Nonstiff Problems. Springer, ??? (1993)

[2] Lions, J.-L., Maday, Y., Turinici, G.: A ”parareal” in time discretization of PDE’s. Comptes Rendus de l’Académie des Sciences - Series I - Mathematics **332**, 661–668 (2001)

[3] Emmett, M., Minion, M.L.: Toward an efficient parallel in time method for partial differential equations. Communications in Applied Mathematics and Computational Science **7**(1), 105–132 (2012)

[4] Falgout, R.D., Friedhoff, S., Kolev, T.V., Maclachlan, S.P., Schroder, J.B.: Parallel Time Integration with Multigrid. SIAM Journal on Scientific Computing **36**(6), 635–661 (2014)

[5] Gander, M.J.: 50 Years of Time Parallel Time Integration. In: Carraro, T., Geiger, M., Körkel, S., Rannacher, R. (eds.) Multiple Shooting and Time Domain Decomposition Methods, pp. 69–113. Springer, Cham (2015)

[6] Gander, M.J., Hairer, E.: Nonlinear Convergence Analysis for the Parareal Algorithm. In: Langer, U., Discacciati, M., Keyes, D.E., Widlund, O.B., Zulehner, W. (eds.) Domain Decomposition Methods in Science and Engineering XVII, pp. 45–56. Springer, Berlin, Heidelberg (2008)

[7] Gander, M.J., Vandewalle, S.: Analysis of the Parareal Time-Parallel Time-Integration Method. SIAM Journal on Scientific Computing **29**(2), 556–578 (2007)

[8] Lee, Y., Park, J., Lee, C.-O.: Parareal Neural Networks Emulating a Parallel-in-Time Algorithm. IEEE Transactions on Neural Networks and Learning Systems, 1–12 (2022)

[9] Ibrahim, A.Q., Götschel, S., Ruprecht, D.: Parareal with a Physics-Informed Neural Network as Coarse Propagator. In: Cano, J., Dikaiakos, M.D., Papadopoulos, G.A., Pericàs, M., Sakellariou, R. (eds.) Euro-Par 2023: Parallel Processing, pp. 649–663. Springer, Cham (2023)

[10] Jin, B., Lin, Q., Zhou, Z.: Learning Coarse Propagators in Parareal Algorithm. arXiv preprint arXiv:2311.15320 (2023)

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

[12] Mishra, S., Molinaro, R.: Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. IMA Journal of Numerical Analysis **42**(2), 981–1022 (2022)

[13] Doumèche, N., Biau, G., Boyer, C.: Convergence and error analysis of PINNs. arXiv preprint arXiv:2305.01240 (2023)

[14] De Ryck, T., Mishra, S.: Error analysis for physics-informed neural networks (PINNs) approximating Kolmogorov PDEs. Advances in Computational Mathematics **48**(6), 79 (2022)

[15] Opschoor, J.A., Petersen, P.C., Schwab, C.: Deep ReLU networks and high-order finite element methods. Analysis and Applications **18**(05), 715–770 (2020)

[16] Kutyniok, G., Petersen, P., Raslan, M., Schneider, R.: A Theoretical Analysis of Deep Neural Networks and Parametric PDEs. Constructive Approximation **55**(1), 73–125 (2022)

[17] Liu, Y., Kutz, J.N., Brunton, S.L.: Hierarchical deep learning of multiscale differential equation time-steppers. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **380**(2229), 20210200 (2022)

[18] Regazzoni, F., Dedè, L., Quarteroni, A.: Machine learning for fast and reliable solution of time-dependent differential equations. Journal of Computational Physics **397**, 108852 (2019)

[19] Lange, H., Brunton, S.L., Kutz, J.N.: From Fourier to Koopman: Spectral Methods for Long-term Time Series Prediction. J. Mach. Learn. Res. **22**(1) (2021)

[20] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: Theory and applications. Neurocomputing **70**(1-3), 489–501 (2006)

[21] Huang, G., Huang, G.-B., Song, S., You, K.: Trends in extreme learning machines: A review. Neural Networks **61**, 32–48 (2015)

[22] Rahimi, A., Recht, B.: Uniform Approximation of Functions with Random Bases. In: 2008 46th Annual Allerton Conference on Communication, Control, and Computing, pp. 555–561 (2008)

[23] Rahimi, A., Recht, B.: Weighted Sums of Random Kitchen Sinks: Replacing

minimization with randomization in learning. Advances in neural information processing systems **21** (2008)

[24] Fabiani, G., Galaris, E., Russo, L., Siettos, C.: Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs. Chaos: An Interdisciplinary Journal of Nonlinear Science **33**(4), 043128 (2023)

[25] Mortari, D., Johnston, H., Smith, L.: High accuracy least-squares solutions of nonlinear differential equations. Journal of computational and applied mathematics **352**, 293–307 (2019)

[26] Schiassi, E., De Florio, M., D'Ambrosio, A., Mortari, D., Furfaro, R.: Physics-Informed Neural Networks and Functional Interpolation for Data-Driven Parameters Discovery of Epidemiological Compartmental Models. Mathematics **9**(17), 2069 (2021)

[27] Dwivedi, V., Srinivasan, B.: Physics Informed Extreme Learning Machine (PIELM)–A rapid method for the numerical solution of partial differential equations. Neurocomputing **391**, 96–118 (2020)

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

[29] Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics vol. 37. Springer, ??? (2006)

[30] Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, ??? (1999)

[31] Branch, M.A., Coleman, T.F., Li, Y.: A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems. SIAM Journal on Scientific Computing **21**(1), 1–23 (1999)

[32] Bullo, F.: Contraction Theory for Dynamical Systems, 1.1 edn. Kindle Direct Publishing, ??? (2023)

[33] Flamant, C., Protopapas, P., Sondak, D.: Solving Differential Equations Using Neural Network Solution Bundles. arXiv preprint arXiv:2006.14372 (2020)

[34] Wang, S., Perdikaris, P.: Long-time integration of parametric evolution equations with physics-informed deeponets. Journal of Computational Physics **475**, 111855 (2023)

[35] Söderlind, G.: On nonlinear difference and differential equations. BIT Numerical Mathematics **24**, 667–680 (1984)

[36] Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I,

Nonstiff Problems. Springer, ??? (1993)

[37] Desoer, C.A., Vidyasagar, M.: Feedback Systems: Input–Output Properties. SIAM, ??? (2009)

[38] Magnus, J.R., Neudecker, H.: Matrix Differential Calculus with Applications in Statistics and Econometrics. John Wiley & Sons, ??? (2019)

[39] Ault, S., Holmgreen, E.: Dynamics of the Brusselator. Math 715 Projects (Autumn 2002) **2** (2003)

Appendix A A-posteriori error estimate based on the defect

We now derive an alternative a-posteriori estimate for network-based approximate solutions based on defect control.

Lemma 1. *Consider the initial value problem* (1)*, given by*

$$\begin{cases} \mathbf{x}'(t) = \mathcal{F}(\mathbf{x}(t)), \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases}$$

where $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$ *is continuously differentiable and admits a unique solution. Let* $\mathbf{y} : \mathbb{R} \to \mathbb{R}^d$ *satisfy*

$$\begin{cases} \mathbf{y}'(t) = \mathcal{F}(\mathbf{y}(t)) + \mathbf{d}(t), \quad \mathbf{d} : \mathbb{R} \to \mathbb{R}^d, \\ \mathbf{y}(0) = \mathbf{x}_0. \end{cases}$$

Then $\mathbf{z}(t) := \mathbf{y}(t) - \mathbf{x}(t)$ *satisfies the linear differential equation*

$$\begin{cases} \mathbf{z}'(t) = A(t)\,\mathbf{z}(t) + \mathbf{d}(t), \\ \mathbf{z}(0) = 0, \end{cases} \tag{A1}$$

where

$$A(t) = \int_0^1 \mathrm{D}\,\mathcal{F}(\mathbf{x}(t) + s\mathbf{z}(t))\mathrm{d}s$$

and $\mathrm{D}\,\mathcal{F}$ *is the Jacobian matrix of* $\mathcal{F}$*.*

Proof. To prove the lemma, it suffices to highlight that

$$\mathcal{F}(\mathbf{y}(t)) - \mathcal{F}(\mathbf{x}(t)) = \int_0^1 \frac{d}{ds}\mathcal{F}(\mathbf{x}(t) + s\mathbf{z}(t))\,\mathrm{d}s = A(t)\,\mathbf{z}(t).$$

□

The solution to the linear problem (A1) satisfies the following bound:

Lemma 2 (Theorem 1 in [35])**.** *Let* $\mathbf{z}(t)$ *solve the initial value problem in* (A1)*. Suppose that* $\|\mathbf{d}(t)\|_2 \leq \varepsilon$ *for* $t \geq 0$*. Then,*

$$\|\mathbf{z}(t)\|_2 \leq \varepsilon \int_0^t \exp\left(\int_s^t \mu_2\left(A\left(\tau\right)\right) \mathrm{d}\tau\right) \mathrm{d}s,$$

where

$$\mu_2(A) = \lambda_{\max}\left(\frac{A + A^\top}{2}\right)$$

is the logarithmic $2-$*norm of* A*.*

For the proof of this lemma, see [36, Theorem 10.6].

As we are interested in solving (1), we set $\mathbf{y}(t) := \mathcal{N}_\theta\left(t; \mathbf{x}_0\right)$ and introduce the defect function

$$\mathbf{d}\left(t\right) := \mathcal{N}_\theta'\left(t; \mathbf{x}_0\right) - \mathcal{F}\left(\mathcal{N}_\theta\left(t; \mathbf{x}_0\right)\right).$$

We remark that the definition of $\mathbf{d}$ is of the same form as the loss (6). If it was known that $\|\mathbf{d}(t)\|_2 \leq \varepsilon$ for a tolerance $\varepsilon > 0$ and all $t \in [0, \Delta t]$, then by Lemma 2 we could conclude that

$$\left\|\mathbf{x}\left(t\right) - \mathcal{N}_\theta\left(t; \mathbf{x}_0\right)\right\|_2 \leq \varepsilon \int_0^t \exp\left(\int_s^t \mu_2\left(A\left(\tau\right)\right) \mathrm{d}\tau\right) \mathrm{d}s, \quad t \in \left[0, \Delta t\right].$$

Given that the solution $\mathbf{x}(t)$ is unknown, $A(\tau)$ and its logarithmic norm cannot be computed exactly. Thus, for a more practical error estimate, we introduce an assumption on the existence of a compact subset $\Omega \subset \mathbb{R}^d$ such that $\mathbf{x}(t) + s\mathbf{z}(t) \in \Omega$ for $(s, t) \in [0, 1] \times [0, \Delta t]$. Then, we can proceed with the inequality chain as

$$\left\|\mathbf{x}\left(t\right) - \mathcal{N}_\theta\left(t; \mathbf{x}_0\right)\right\|_2 \leq \varepsilon \int_0^t e^{M(t-s)} \mathrm{d}s = \varepsilon \frac{e^{Mt} - 1}{M}, \tag{A2}$$

where $M := \max_{\mathbf{z} \in \Omega} \mu_2(\mathrm{D}\,\mathcal{F}(\mathbf{z})) \in \mathbb{R}$. Note that the right-hand side of (A2) is non-negative for all $t \geq 0$. In particular, (A2) implies that a neural network $\mathcal{N}_\theta$ can be employed to approximate the solution of (1) which is as accurate as a classical coarse solver $\varphi_C^{\Delta t}$ provided the norm of the defect $\|\mathbf{d}(t)\|_2$ is sufficiently small.

Appendix B Bound on the norm of the sensitivity matrix

In this appendix, we provide a practical bound for the norm of the Jacobian of the flow map of a vector field $\mathcal{F}$, assumed to be continuously differentiable with respect to the initial condition. For this, we differentiate the initial value problem (1), given by

$$\begin{cases} \frac{d}{dt}\phi_{\mathcal{F}}^{s,t}\left(\mathbf{x}_0\right) = \mathcal{F}\left(\phi_{\mathcal{F}}^{s,t}\left(\mathbf{x}_0\right)\right) \in \mathbb{R}^d, \\ \phi_{\mathcal{F}}^{s,s}\left(\mathbf{x}_0\right) = \mathbf{x}_0, \end{cases} \tag{B3}$$

with respect to $\mathbf{x}_0$ and obtain

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial \phi_{\mathcal{F}}^{s,t}(\mathbf{x}_0)}{\partial \mathbf{x}_0}\right) = \mathrm{D}\,\mathcal{F}\left(\phi_{\mathcal{F}}^{s,t}\left(\mathbf{x}_0\right)\right) \frac{\partial \phi_{\mathcal{F}}^{s,t}(\mathbf{x}_0)}{\partial \mathbf{x}_0} \in \mathbb{R}^{d\times d}, \\ \frac{\partial \phi_{\mathcal{F}}^{s,s}(\mathbf{x}_0)}{\partial \mathbf{x}_0} = I_d, \end{cases} \tag{B4}$$

where $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix. Equation (B4) is generally known as the variational equation of (B3). This ODE is a non-autonomous linear differential equation in the unknown matrix $\partial_{\mathbf{x}_0}\phi_{\mathcal{F}}^{s,t}(\mathbf{x}_0)$. In practice, (B4) should be solved jointly with (B3). However, for the purpose of bounding the Euclidean norm $\|\partial_{\mathbf{x}_0}\phi_{\mathcal{F}}^{s,t}(\mathbf{x}_0)\|_2$, it is not necessary to solve them. Following [37, Chapter 2], we assume that $\phi_{\mathcal{F}}^{s,t}(\mathbf{x}_0) \in \Omega$ for $\Omega \subset \mathbb{R}^d$ compact and all $0 \leq s \leq t \leq \Delta t$. This is not a restrictive assumption on compact time intervals given the assumed regularity for $\mathcal{F}$. Then, one can get

$$\begin{aligned} \left\|\partial_{\mathbf{x}_0}\phi_{\mathcal{F}}^{s,t}\left(\mathbf{x}_0\right)\right\|_2 &\leq \left\|\partial_{\mathbf{x}_0}\phi_{\mathcal{F}}^{s,s}\left(\mathbf{x}_0\right)\right\|_2 \exp\left(\int_s^t \mu_2\left(\mathrm{D}\,\mathcal{F}\left(\phi_{\mathcal{F}}^{s,s'}\left(\mathbf{x}_0\right)\right)\right)\mathrm{d}s'\right) \\ &= \exp\left(\int_s^t \mu_2\left(\mathrm{D}\,\mathcal{F}\left(\phi_{\mathcal{F}}^{s,s'}\left(\mathbf{x}_0\right)\right)\right)\mathrm{d}s'\right) \leq \exp\left(M\Delta t\right), \end{aligned}$$

where $M = \max_{\mathbf{z}\in\Omega}\mu_2(D\mathcal{F}(\mathbf{z}))$. We conclude that the constant δ in the proof of Theorem 2 can be set to $\exp(M\Delta t)$, with M positive or negative depending on $\mathcal{F}$.

Appendix C The Jacobian matrix of the loss function

In this subsection, we consider the loss function (16) and its gradient. Note that (16) can be expressed as (6) which in turn can be related to the solution of the non-linear matrix equation

$$\tilde{\mathbf{X}}'_\theta\left(\mathbf{x}, \Delta t\right) = \mathbf{F}\left(\tilde{\mathbf{X}}_\theta\left(\mathbf{x}, \Delta t\right)\right).$$

More explicitly, we have

$$\begin{aligned} &\tilde{\mathbf{X}}_\theta\left(\mathbf{x}, \Delta t\right) = \mathbf{1}_C\mathbf{x}^\top + \left(\mathbf{H} - \overline{\mathbf{H}}\right)\theta,\ \mathbf{1}_C = \begin{bmatrix}1 & \dots & 1\end{bmatrix}^\top \in \mathbb{R}^C, \\ &\tilde{\mathbf{X}}'_\theta\left(\mathbf{x}, \Delta t\right) = \mathbf{H}'\theta. \end{aligned}$$

To minimize the loss function (6), we need the Jacobian of the matrix-valued function $\mathbf{G}_\theta(\mathbf{x}, \Delta t) = \tilde{\mathbf{X}}'_\theta(\mathbf{x}, \Delta t) - \mathbf{F}(\tilde{\mathbf{X}}_\theta(\mathbf{x}, \Delta t))$. As $\mathbf{G}_\theta$ is a matrix-valued function with matrix inputs, we rely on the vectorization operator, denoted by vec, using the machinery of matrix-calculus introduced, for example, in [38]. We hence compute $\frac{\partial \mathrm{vec}(\mathbf{G}_\theta(\mathbf{x},\Delta t))}{\partial \mathrm{vec}(\theta)} \in \mathbb{R}^{Cd\times Hd}$, given by

$$\frac{\partial \mathrm{vec}\left(\mathbf{G}_\theta\left(\mathbf{x}, \Delta t\right)\right)}{\partial \mathrm{vec}\left(\theta\right)} = I_d \otimes \mathbf{H}' - \frac{\partial \mathrm{vec}\left(\mathbf{F}\left(\tilde{\mathbf{X}}_\theta\left(\mathbf{x}, \Delta t\right)\right)\right)}{\partial \mathrm{vec}\left(\theta\right)}$$

$$
\begin{aligned}
&= I_d \otimes \mathbf{H}' - \frac{\partial \mathrm{vec}\left(\mathbf{F}\left(\mathbf{X}\right)\right)}{\partial \mathrm{vec}\left(\mathbf{X}\right)}\Big|_{\mathbf{X}=\tilde{\mathbf{X}}_\theta(\mathbf{x},\Delta t)} \frac{\partial \mathrm{vec}\left(\tilde{\mathbf{X}}_\theta\left(\mathbf{x},\Delta t\right)\right)}{\partial \mathrm{vec}\left(\theta\right)} \\
&= I_d \otimes \mathbf{H}' - \frac{\partial \mathrm{vec}\left(\mathbf{F}\left(\mathbf{X}\right)\right)}{\partial \mathrm{vec}\left(\mathbf{X}\right)}\Big|_{\mathbf{X}=\tilde{\mathbf{X}}_\theta(\mathbf{x},\Delta t)} \left(I_d \otimes \left(\mathbf{H} - \overline{\mathbf{H}}\right)\right),
\end{aligned}
$$

where $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix, $\otimes$ is the Kronecker product, and vec stacks the columns of the input matrix into a column vector. The Jacobian of $\mathbf{F}$ in the last line depends on the vector field $\mathcal{F}$, while the other terms do not.

Most of the dynamical systems we consider in the numerical experiments in Section 6 are of low dimension. For this reason, for all the cases but Burgers' equation, we assemble the Jacobian case by case, following this construction. For Burgers' equation, we instead implement it as a linear operator, specifying its action and the action of its transpose onto input vectors. For the Burgers' equation, we have

$$
\mathcal{F}\left(\mathbf{u}\right) = -\mathbf{u} \odot \left(\mathbf{D}_1\mathbf{u}\right) + \nu\mathbf{D}_2\mathbf{u} \in \mathbb{R}^d,
$$

and hence $\mathbf{F}(\mathbf{X}) = -\mathbf{X} \odot (\mathbf{X}\mathbf{D}_1^\top) + \nu\mathbf{X}\mathbf{D}_2^\top \in \mathbb{R}^{C\times d}$. This expression implies that

$$
\frac{\partial \mathrm{vec}\left(\mathbf{F}\left(\mathbf{X}\right)\right)}{\partial \mathrm{vec}\left(\mathbf{X}\right)} = -\mathrm{diag}\left(\mathrm{vec}\left(\mathbf{X}\mathbf{D}_1^\top\right)\right) - \mathrm{diag}\left(\mathrm{vec}\left(\mathbf{X}\right)\right)\left(\mathbf{D}_1 \otimes I_C\right) + \nu\mathbf{D}_2 \otimes I_C.
$$

Appendix D Details on the network for the flow map approach

In this section, we provide details on the network for the flow map approach required for the comparison of the training costs presented in Table 1. The network used for the coarse propagator is based on the parametrization

$$
\begin{aligned}
\mathbf{z} := \left[\mathbf{x}_0^\top, t\right]^\top &\mapsto \tanh\left(\mathbf{A}_0\mathbf{z} + \mathbf{a}_0\right) =: \mathbf{h}_1 \in \mathbb{R}^{10}, \\
\mathbf{h}_\ell &\mapsto \tanh\left(\mathbf{A}_\ell\mathbf{h}_\ell + \mathbf{a}_\ell\right) =: \mathbf{h}_{\ell+1} \in \mathbb{R}^{10},\ \ell = 1, \cdots, 4, \\
\mathbf{h}_5 &\mapsto \mathbf{x}_0 + \left(1 - e^{-t}\right)\mathbf{P}\mathbf{h}_5 =: \mathcal{N}_\theta\left(t;\mathbf{x}_0\right) \in \mathbb{R}^3,
\end{aligned}
$$

where $\theta = \{\mathbf{A}_\ell, \mathbf{a}_\ell, \mathbf{P}\}_{\ell=0}^4$. To train the network, implemented with PyTorch, we use the Adam optimizer for 10^5 epochs, with each epoch consisting of minimizing the ODE residual over 500 different randomly sampled collocation points $(t^i, \mathbf{x}_0^i) \in [0,1]\times[0,1]^3$.

Appendix E Experiment for Brusselator's equation

This section collects numerical experiments for the Brusselator, which is a system of two scalar differential equations modeling a chain of chemical reactions [39]. The equations write

$$\begin{cases} x_1'(t) = A + x_1^2(t)\, x_2(t) - (B+1)\, x_1(t), \\ x_2'(t) = Bx_1(t) - x_1^2(t)\, x_2(t), \\ \mathbf{x}(0) = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top, \end{cases} \tag{E5}$$

where we choose the parameters $A = 1$, $B = 3$. In this setting, one can prove to have a limit cycle in the dynamics. We simulate this system on the time interval $[0, T = 12]$, with a fine timestep $\delta t = T/640$ and a coarse one of size $\Delta T = T/32$. We repeat the simulation 100 times, reporting the average cost of one coarse timestep in Table E1, together with the average total cost of the hybrid Parareal solver. Figure E1 shows the approximate solution and a reference solution. We also remark that, as desired, the hybrid method recovers the limit cycle.

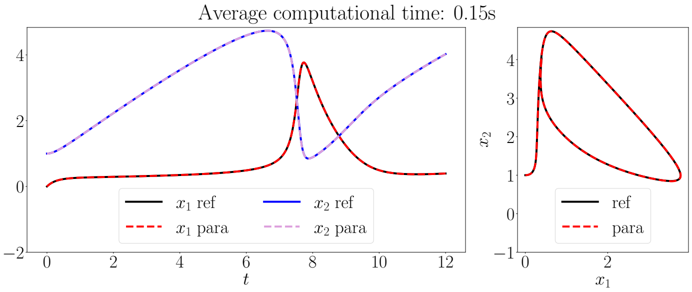


Fig. E1: Qualitative accuracy of the predicted solutions using Parareal with ELM.

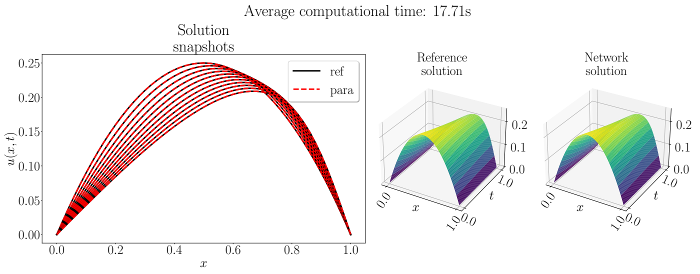
Timing breakdown	ELM
Average cost coarse step in the zeroth iterate	0.001012s
Average cost to produce the solution	0.1469s

Table E1: Brusselator: Computational time for Hybrid Parareal using a single core.

Appendix F Additional experiments for Burgers' equation

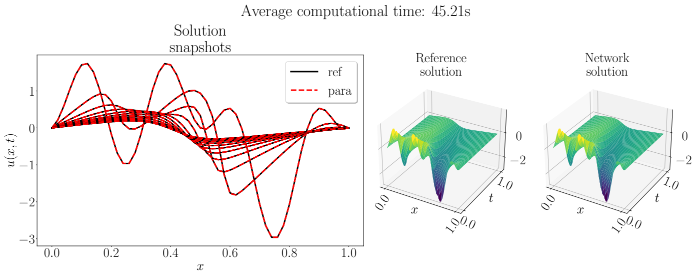
Timing breakdown	ELM
Average cost coarse step in the zeroth iterate	0.1695s
Average cost to produce the solution	17.7069s

Table F2: Burgers: Computational time for Hybrid Parareal using five cores, and initial condition $u_0(x) = x(1-x)$.



(a) (b)

Fig. F2: Burgers: Snapshots of the solution obtained with Hybrid Parareal (left), comparison of the solution surfaces between Hybrid Parareal and the fine integrator applied serially (right). Solution corresponding to $u_0(x) = x(1-x)$.



(a) (b)

Fig. F3: Burgers: Snapshots of the solution obtained with Hybrid Parareal (left), comparison of the solution surfaces between Hybrid Parareal and the fine integrator applied serially (right). Solution corresponding to $u_0(x) = \sin(2\pi x) + \cos(4\pi x) - \cos(8\pi x)$.

Timing breakdown	ELM
Average cost coarse step in the zeroth iterate	0.3356s
Average cost to produce the solution	45.2056s

Table F3: Burgers: Computational time for Hybrid Parareal using five cores, and initial condition $u_0(x) = \sin(2\pi x) + \cos(4\pi x) - \cos(8\pi x)$.

In this section, we report the simulation results for the Burgers' equation with two more initial conditions. The setup of the network and the partition of the time domain are the same as for the initial condition included in Section 6.5. In Figure F2, we work with the initial condition $u_0(x) = x(1-x)$, while in Figure F3 with $u_0(x) = \sin(2\pi x) + \cos(4\pi x) - \cos(8\pi x)$. The timings are included in Tables F2 and F3, respectively. As expected, the time to obtain the full solution grows with the complexity of the initial condition. Indeed, there are about 10 seconds of difference between the fastest, corresponding to the quadratic initial condition in Figure F3, to the second fastest, the one with $u_0(x) = \sin(2\pi x)$, and the slowest in Figure F3. The reason behind this observed behavior is that, for more complicated solutions, the coarse predictions need to be corrected with the Parareal correction step more often, and the optimization problems to solve to get the coarse propagator get more expensive.