# Nearest Neighbors GParareal: Improving Scalability of Gaussian Processes for Parallel-in-Time Solvers

**Guglielmo Gattiglio**, Lyudmila Grigoryeva, Massimiliano Tamborrino
Department of Statistics, University of Warwick, Coventry, UK guglielmo.gattiglio@warwick.ac.uk

## Motivation

Why is time parallelization for ODEs and PDEs important?

- In plasma physics and other fields, space parallelization reaches saturation on modern supercomputers leaving time parallelization as the only avenue for improvement [1].
- Simulations of molecular dynamics often involve averages over very long trajectories of stochastic dynamics [2].

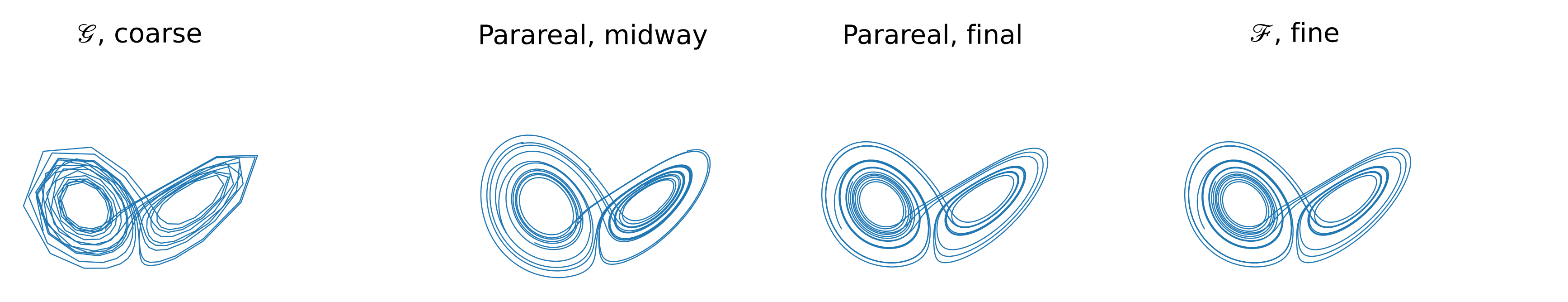

Figure 1: Visualization of Parareal evolution on chaotic Lorenz attractor.

## Existing Approaches: Parareal and GParareal

Consider a system of $d \in \mathbb{N}$ ODEs (and similarly for PDEs)

$$\frac{du}{dt} = h(u(t), t) \text{ on } t \in [t_0, t_N], \text{ with } u(t_0) = u^0, \tag{1}$$

where $h : \mathbb{R}^d \times [t_0, t_N] \to \mathbb{R}^d$ is a smooth multivariate function, $u : [t_0, t_N] \to \mathbb{R}^d$ is the time dependent vector solution, and $u^0 \in \mathbb{R}^d$ are the initial values at $t_0$. Parareal [3] solves (1) by dividing the timespan $[t_0, t_N]$ into $N$ initial value problems

$$\frac{du_i}{dt} = h(u_i(t \mid U_i), t), \quad t \in [t_i, t_{i+1}], \quad u_i(t_i) = U_i, \text{ for } i = 0, ..., N-1.$$

and solving them in parallel. To ensure continuity, the initial conditions $U_i$ are iteratively updated every Parareal iterations $k$

$$U_i^k = \mathscr{G}\left(U_{i-1}^k\right) + \mathscr{F}\left(U_{i-1}^{k-1}\right) - \mathscr{G}\left(U_{i-1}^{k-1}\right), \quad i = 1, \ldots, N-1, \tag{2}$$

where $\mathscr{F}$ and $\mathscr{G}$ are numerical solvers. $\mathscr{F}$ is slow (hours, days), accurate, and always executed in *parallel*. $\mathscr{G}$ is fast (seconds), inaccurate, and used to build the approximate solution *sequentially*. See Figure 2.

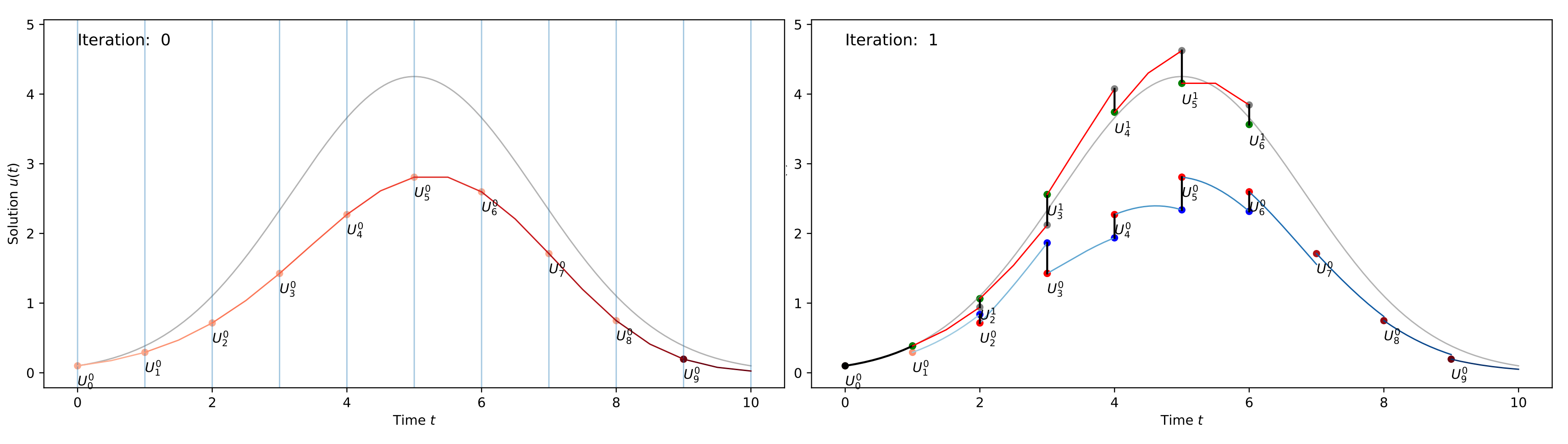

Figure 2: Parareal. Gray line, truth. Left, approximate initial solution $\mathscr{G}$ at iteration $k = 0$. Right, parallel evaluations of $\mathscr{F}$ (blue) and sequential evaluations of $\mathscr{G}$ in (2) (red lines) for iteration $k = 1$. The green, gray, blue and red dots are $U_i^k$, $\mathscr{G}\left(U_{i-1}^k\right)$, $\mathscr{F}\left(U_{i-1}^{k-1}\right)$, and $\mathscr{G}\left(U_{i-1}^{k-1}\right)$ from (2).

In (2), Parareal uses data coming from the previous iteration $k-1$. GParareal [4] changes (2) to use current iteration information

$$U_i^k = \mathscr{F}\left(U_{i-1}^k\right) = (\mathscr{F} - \mathscr{G} + \mathscr{G})\left(U_{i-1}^k\right) = (\mathscr{F} - \mathscr{G})\left(U_{i-1}^k\right) + \mathscr{G}\left(U_{i-1}^k\right). \tag{3}$$

However, this would require a *serial* computation of $\mathscr{F}\left(U_{i-1}^k\right)$, defeating parallelization. Instead, Gaussian processes (GPs) are used to predict the correction term $\mathscr{F} - \mathscr{G}$ using the known (updated) initial condition $U_{i-1}^k$. At iteration $k > 0$, the GPs are trained on the dataset $D_k$ composed of pairs of inputs $U$ and outputs $y = (\mathscr{F} - \mathscr{G})(U)$:

$$D_k = \{(U_i^{j-1}, (\mathscr{F} - \mathscr{G})(U_i^{j-1})), i = 0, ..., N-1, j = 1, ..., k\}.$$

The GP posterior mean is used to predict $(\mathscr{F} - \mathscr{G})\left(U_{i-1}^k\right)$ in (3) as

$$m_{D_k}(U_{i-1}^k) = K(U_{i-1}^k, \mathbf{U})^T [K(\mathbf{U}, \mathbf{U}) + \sigma_n^2 I]^{-1} \mathbf{y}, \tag{4}$$

where $K(\mathbf{U}, \mathbf{U})$ is the covariance matrix. The matrix inversion in (4) is **computationally expensive** with a $O((Nk)^3)$ cost, cubic in the size of the dataset $D_k$ at iteration $k$. This negatively affects speed-up (Figure 4) and makes GParareal unattractive for bigger $N$s. More intervals $N$ allows for greater parallellization, hence the need for an alternative approach.

## Nearest Neighbors GParareal New!

To overcome the computational bottleneck of GParareal, we reduce the dataset size, thereby decreasing the matrix inversion cost. As seen in Figure 3, *most points are far from the prediction point* $U_{i-1}^k$.

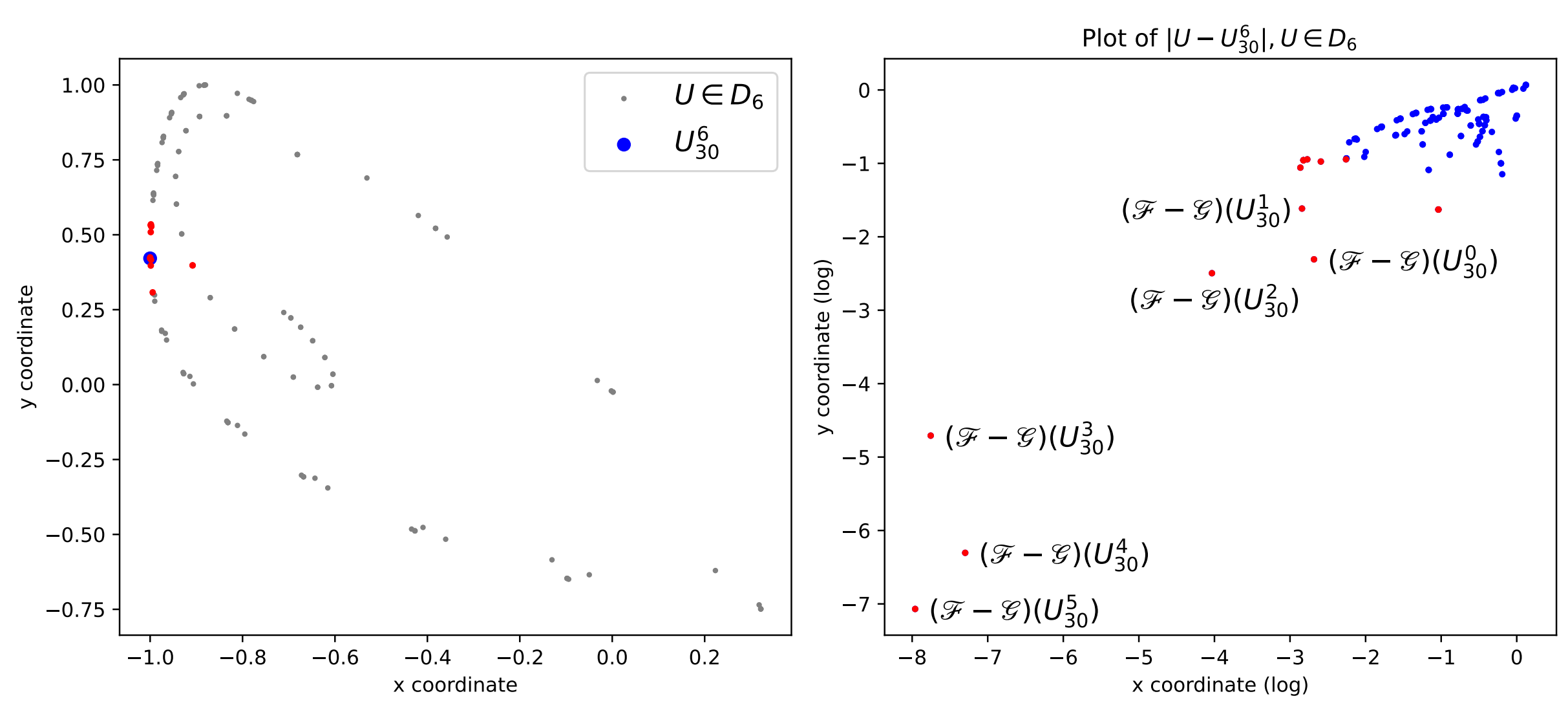

Figure 3: Left, a scatterplot of $D_6$ with the test observation $U_{i-1}^k = U_{30}^6$ (blue). In red are the $m = 15$ nearest neighbors to $U_{30}^6$. Right, plot of the absolute (log) distance coordinate-wise between $U \in \mathbf{U}$ and $U_{30}^6$.

Hence, for a prediction at $U_{i-1}^k$ we can train the GP on a subset $D_{i-1,k} \subset D_k$ consisting of only $m$ points, which we choose as the **nearest neighbors** to $U_{i-1}^k$ (red dots in Figure 3). This is known in the literature as a nearest-neighbors GP (NNGP). Provided $m$ is small, the computational complexity is *sensibly favorable* as the NNGP training cost is $O(Nm^3 + N\log(kN))$ at iteration $k$. The log component comes from data sorting operations.

Figure 4 shows the NN-GParareal speed-up, the ratio of running $\mathscr{F}$ sequentially as opposed to using Parareal

$$\text{Speed-up} = T_{serial}/T_{parall},$$

where $T_{\cdot}$ indicates wallclock runtime. The upper bound to the speed-up achievable by any algorithm converging in $K$ steps is $K/N$. For low values of $N$, GParareal and NN-GParareal provide sensible speed-up. However, as the number of cores increases, GParareal performs worse than Parareal or fails to converge within reasonable time limits.

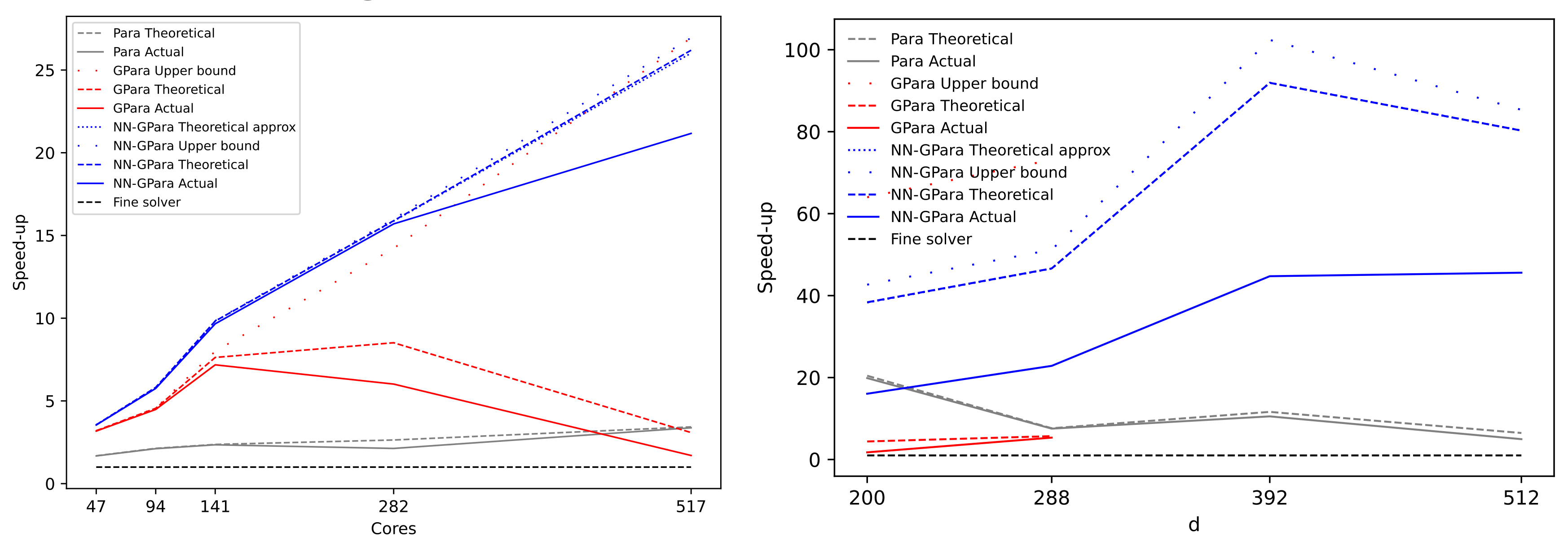

Figure 4: Speed-up for a 2D Hopf bifurcation ODE [5] (left) and FitzHugh-Nagumo 2D PDE [6] (right). Right, $N = 512$.

## References

(1) D. Samaddar, D. P. Coster et al., *Comput. Phys. Commun.*, 2019, **235**, 246–257.
(2) O. Gorynina, F. Legoll et al., *arXiv preprint arXiv:2212.10508*, 2022.
(3) J.-L. Lions, Y. Maday et al., *Comptes Rendus de l'Academie des Sci. I-Mathematics*, 2001, **332**, 661–668.
(4) K. Pentland, M. Tamborrino et al., *Stat. Comput.*, 2023, **33**, 23.
(5) R. Seydel, *Practical bifurcation and stability analysis*, Springer Science & Business Media, 2009, vol. 5.
(6) B. Ambrosio and J.-P. Françoise, *Philos. Trans. Royal Soc. A: Math. Phys. Eng. Sci.*, 2009, **367**, 4863–4875.