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

Guglielmo Gattiglio
University of Warwick

January 23, 2024

Joint work with:
Lyudmila Grigoryeva, University of St. Gallen
Massimiliano Tamborrino, University of Warwick

Parareal (Lions et al., 2001).

- Theory
- Sketch of the procedure
- Computational cost

GParareal (Pentland et al., 2023).

- Intuition
- Empirical results
- Computational cost

Nearest Neighbor GParareal New!

- Intuition
- Empirical results
- Computational cost

Introduction

In this talk, we consider machine-learning-based approaches to speed up Parareal (Lions et al., 2001), a parallel-in-time solver for ODEs and PDEs. Why is time parallelization important?

- ▶ Space parallelization has been a widely use technique for solving PDEs on multiple processors.
- ▶ In plasma physics and other fields, these traditional techniques often reach saturation on modern supercomputers, thus leaving time parallelization as the only avenue for improvement (Samaddar et al., 2019).
- ▶ Simulations of molecular dynamics often involve averages over very long trajectories of stochastic dynamics. Space parallelization is thus useless to reduce the wall clock time requirements (Gorynina et al., 2022)

Parareal (Lions et al. (2001))

Parareal (Lions et al., 2001)

Consider a system of $d \in \mathbb{N}$ ODEs

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

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.

We partition the time domain into N sub-intervals of equal length

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

and we enforce the continuity conditions at each t_i, namely

$$U_0 = u^0, U_i = u_{i-1}(t_i | U_{i-1}), \text{ for } \quad i = 1, \ldots, N.$$

Parareal (Lions et al., 2001)

We solve this system of $N + 1$ equations

$$U_0 = u^0, U_i = u_{i-1}\left(t_i | U_{i-1}\right), \text{ for } \quad i = 1, \ldots, N.$$

using Newton-Rapson, which yields the following iterative strategy:

$$U_0^{k+1} = u^0, \tag{1}$$

$$U_i^{k+1} = u_{i-1}\left(t_i | U_{i-1}^k\right) + \frac{\partial u_{i-1}}{\partial U_{i-1}}\left(t_i | U_{i-1}^k\right)\left[U_{i-1}^{k+1} - U_{i-1}^k\right]. \tag{2}$$

Assume that we have available a fine and a coarse numerical integrator, $\mathscr{F}$ and $\mathscr{G}$ respectively,

- ▶ $\mathscr{F}$ is accurate but computationally expensive, infeasible to run sequentially over $[t_0, t_N]$. Parallel computation over $[t_i, t_{i+1}]$ is possible.
- ▶ $\mathscr{G}$ is less accurate but cheap to execute.

They can be the same solver with different time steps, or different numerical schemes.

Parareal (Lions et al., 2001)

Looking at the expression (2) from the previous slide,

$$U_i^{k+1} = u_{i-1}\left(t_i|U_{i-1}^k\right) + \frac{\partial u_{i-1}}{\partial U_{i-1}}\left(t_i|U_{i-1}^k\right)\left[U_{i-1}^{k+1} - U_{i-1}^k\right], \quad (2)$$

we can approximate the first term using the fine solver, $\mathscr{F}\left(U_{i-1}^k\right)$, and the derivative in the second term by finite differences using

$$\mathscr{G}\left(U_{i-1}^{k+1}\right) - \mathscr{G}\left(U_{i-1}^k\right).$$

Since the update is sequential in time we cannot use $\mathscr{F}$ for approximating the derivative as $\mathscr{F}\left(U_{i-1}^{k+1}\right)$ is not known.

By (2), the starting points are iteratively updated using the predictor-corrector rule

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

Parareal (Lions et al., 2001)

Some comments:

- ▶ The computation of the fine solver over $[t_i, t_{i+1}]$ can be parallelized once all the starting points U_i^0 have been serially computed, normally using $\mathscr{G}$.

- ▶ The stopping criterion of this algorithm is chosen as

$$||U_i^k - U_i^{k-1}||_\infty < \epsilon, \quad \forall i \leq N, \tag{4}$$

  where $|| \cdot ||_\infty$ is the infinity norm, which guarantees that the initial conditions have stabilized.
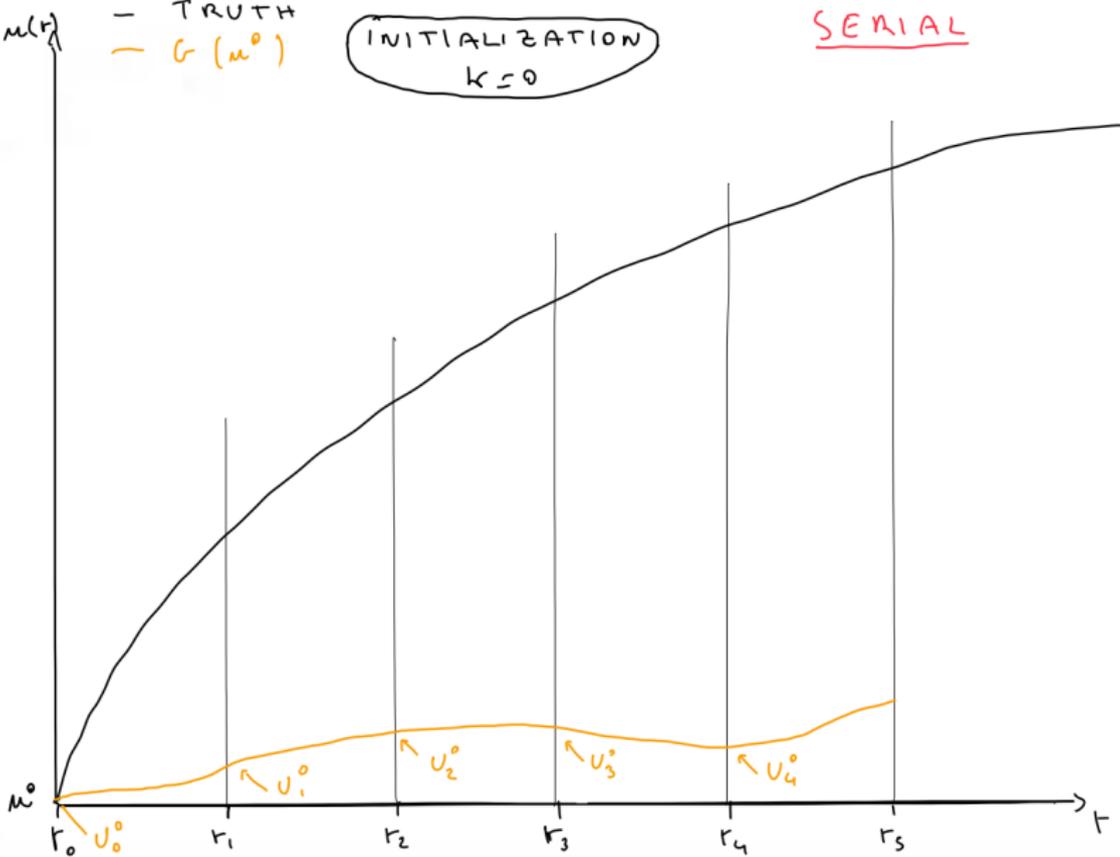
- ▶ Let K be the total number of Parareal iterations to convergence. In the worst-case scenario, $K = N$ and the solution trivially converges to that of the fine solver.
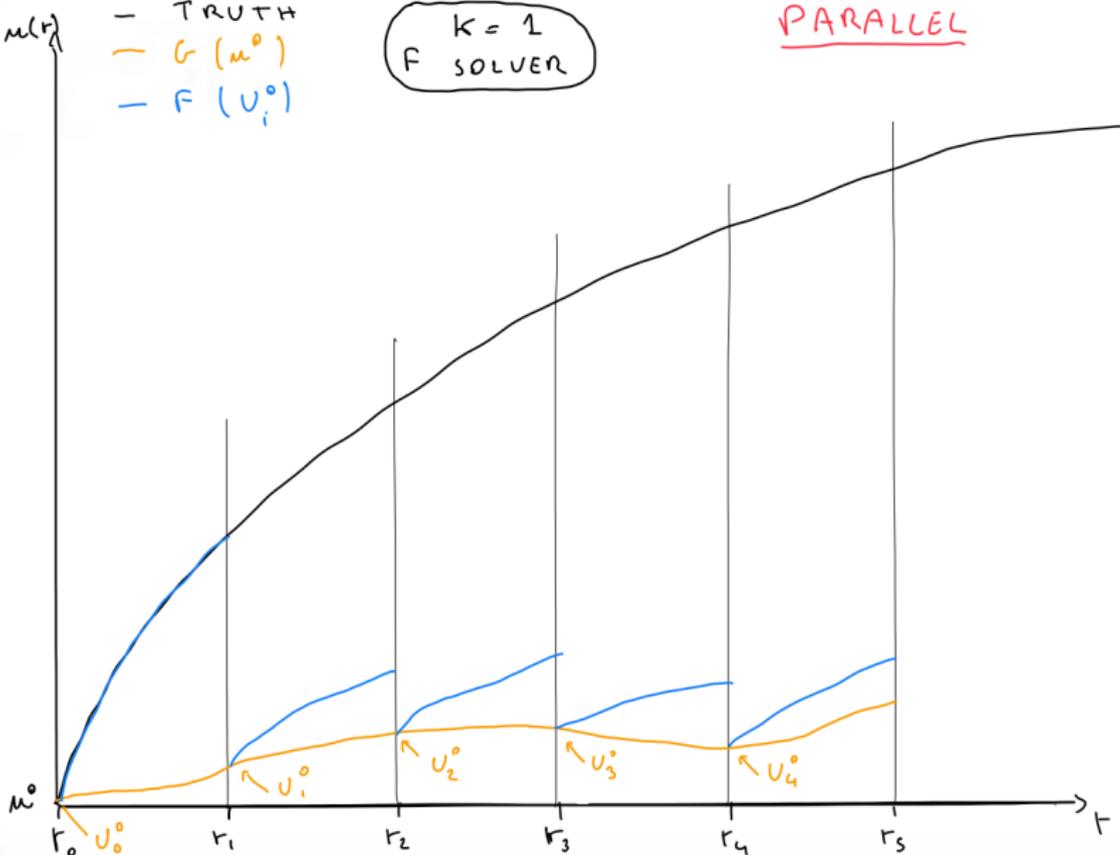
Consider solving the following ODE with initial condition $u^0 = 0$, and $N = 5$
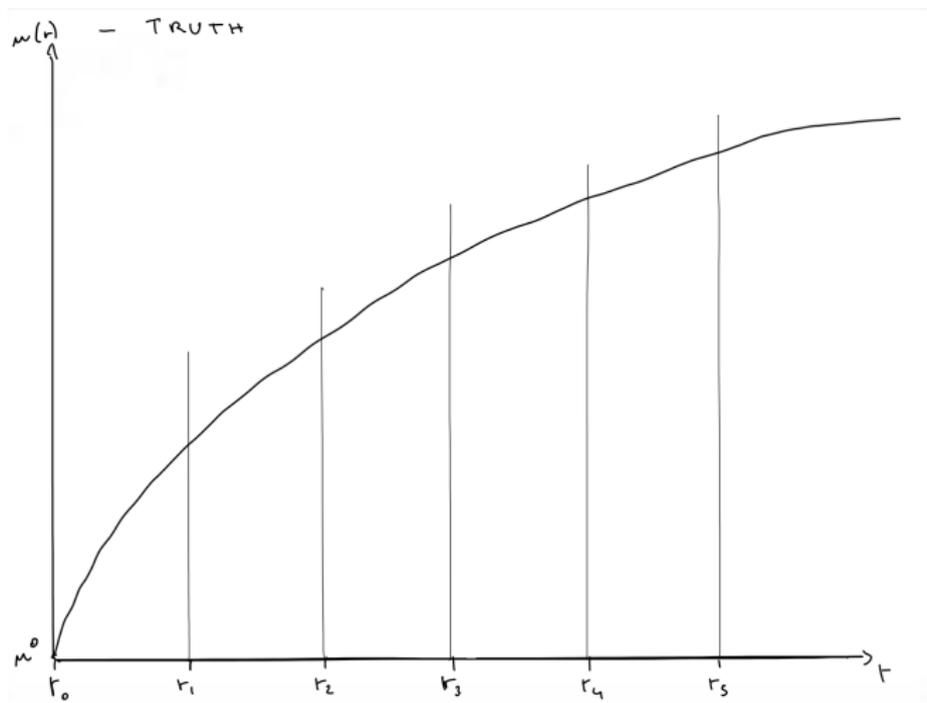
Parareal - Sketch of behavior 2/8
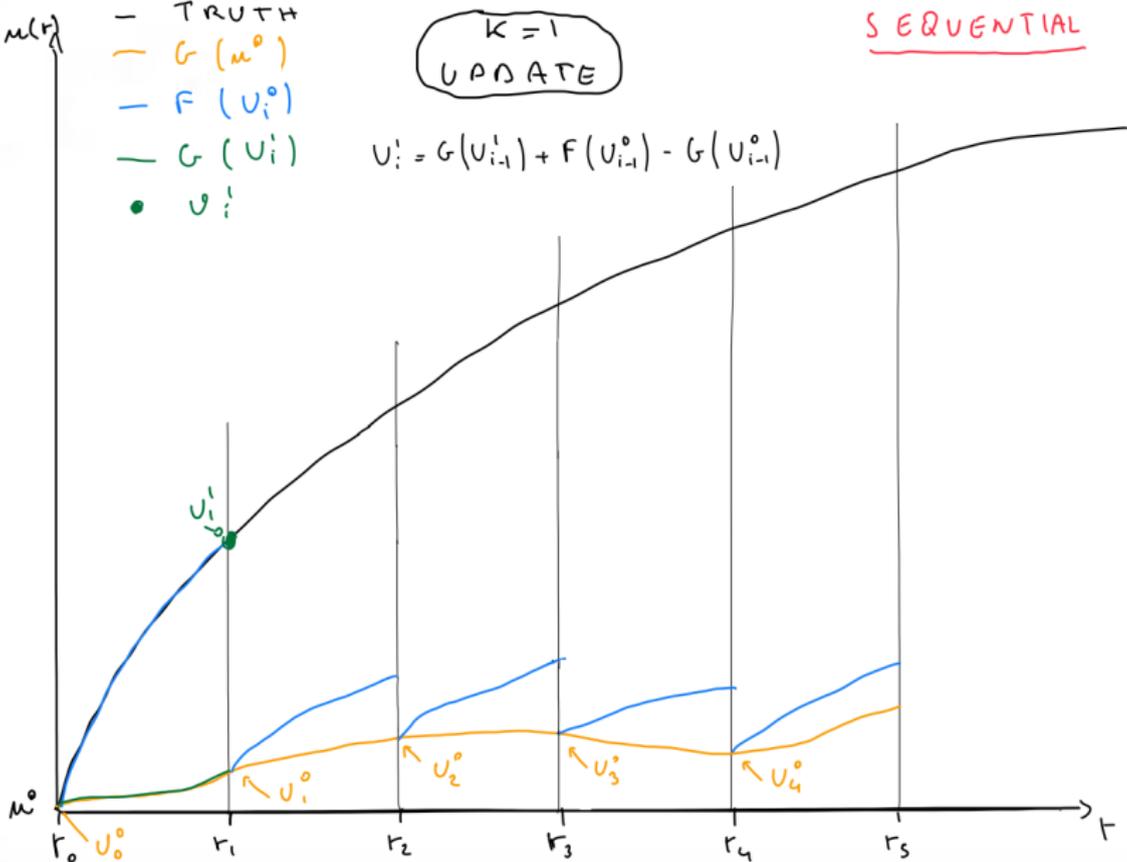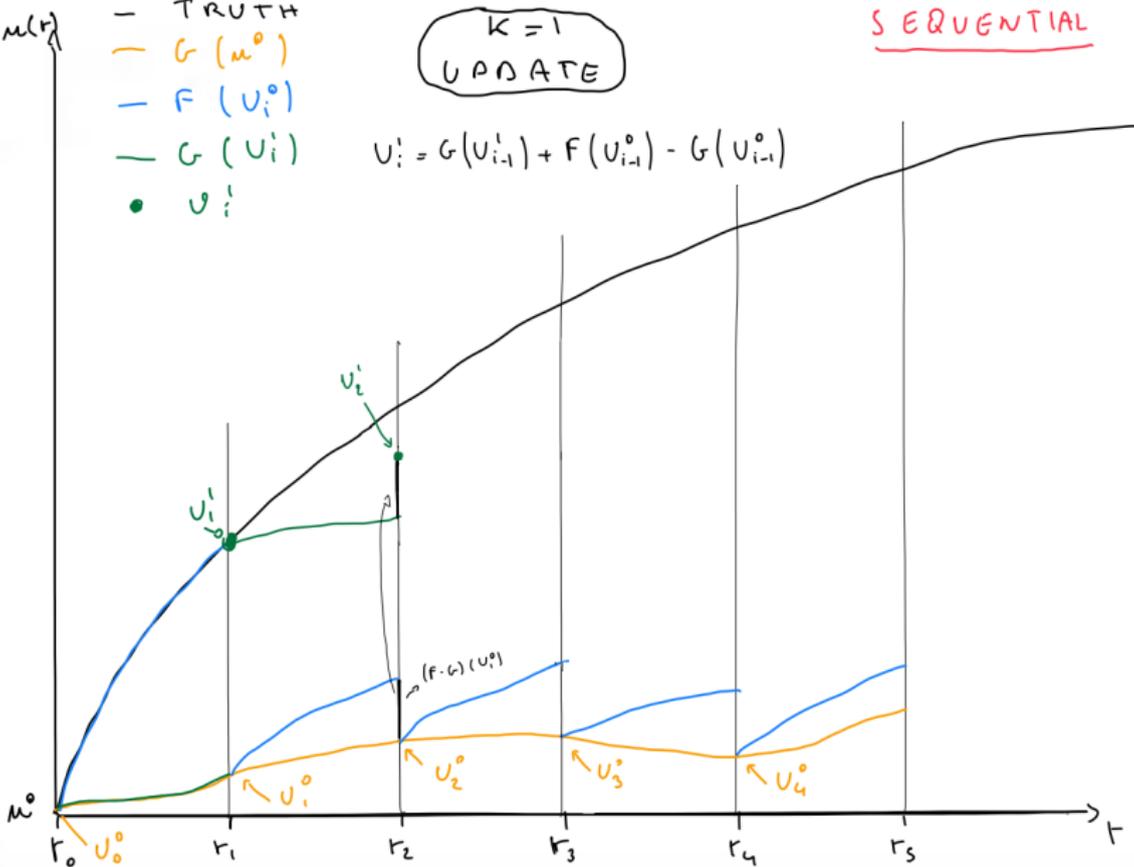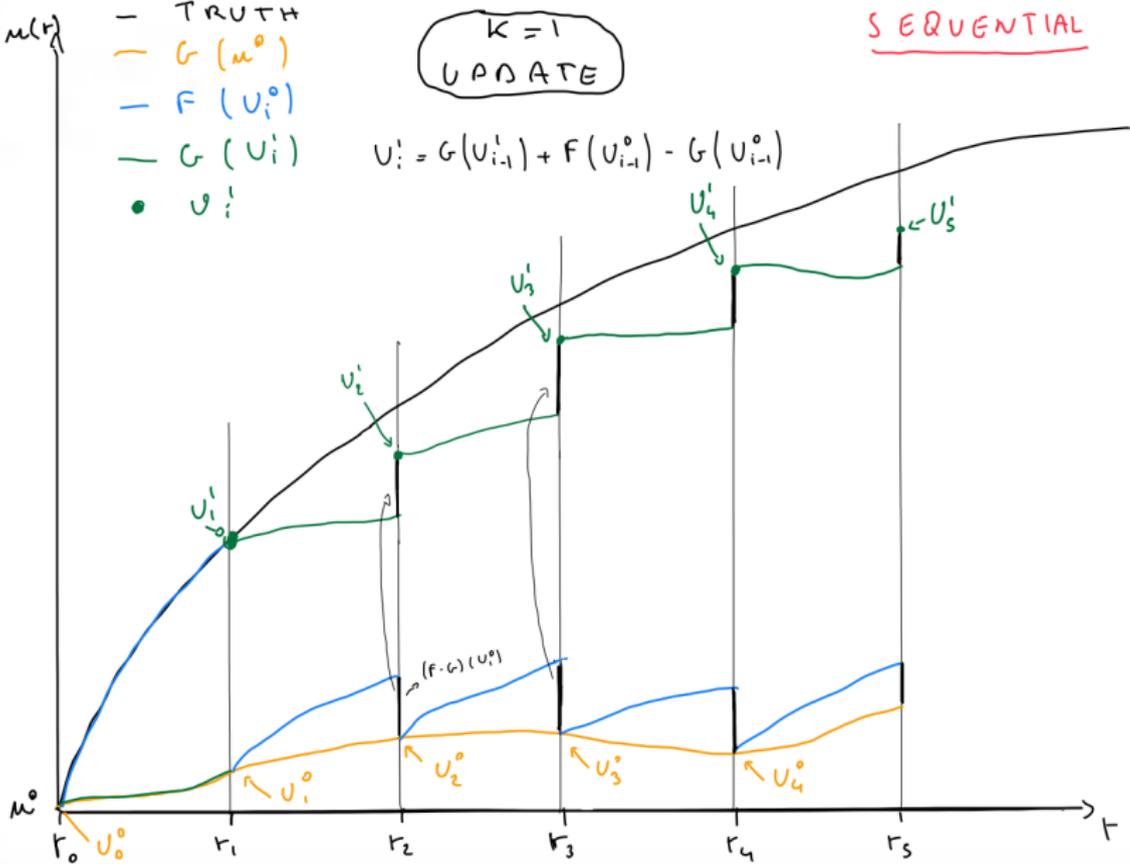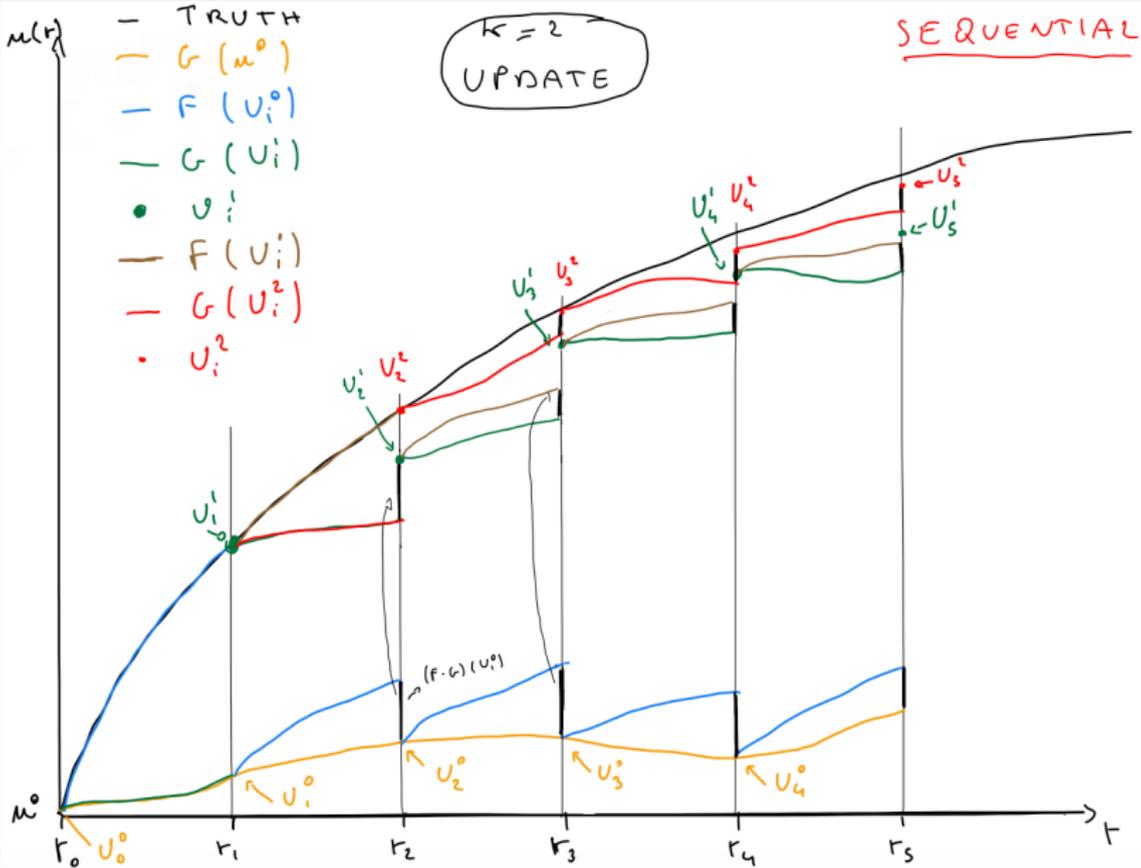
Parareal - Sketch of behavior 4/8
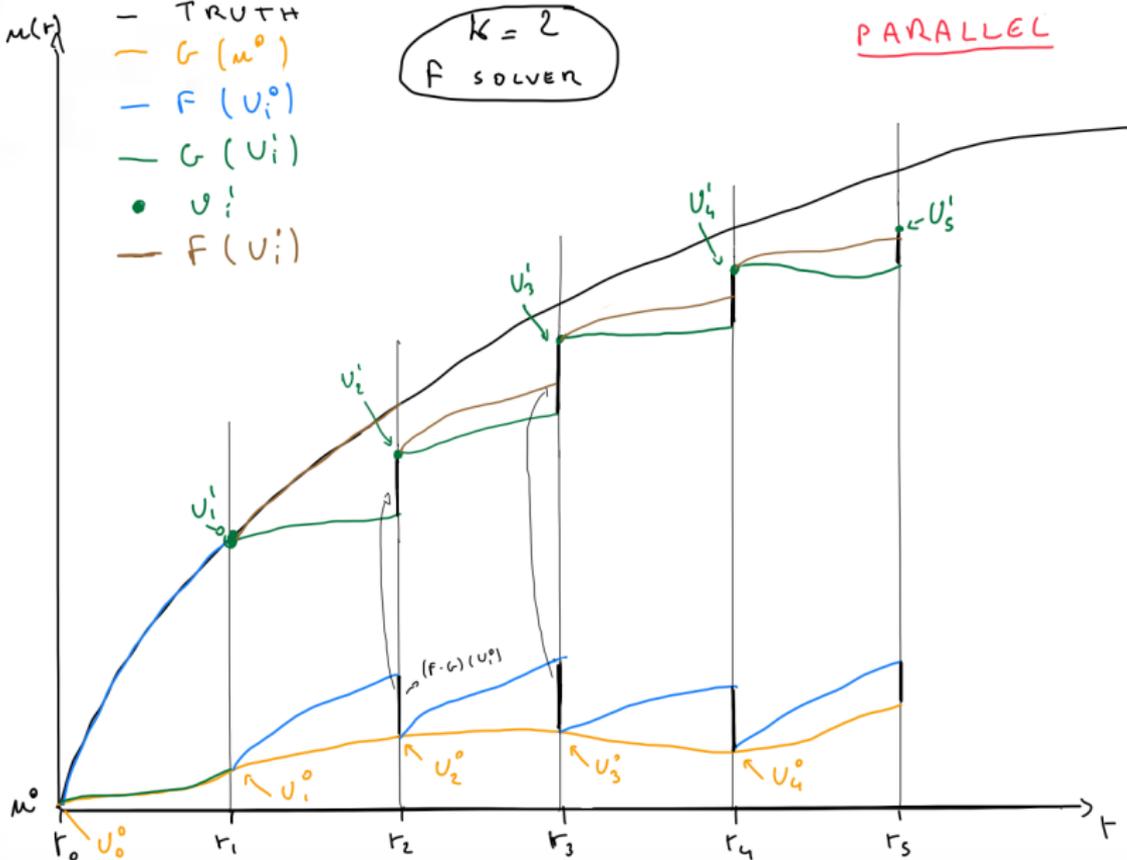
Parareal - Computational cost

Assume that running $\mathscr{F}$ over one interval $[t_i, t_{i+1}]$ takes $T_\mathscr{F}$ time, and similarly for $\mathscr{G}$. The cost of the *serial* procedure is

$$T_{Serial} = N T_\mathscr{F}.$$

The cost of Parareal, assuming it converges in K_{Para} iterations, is

$$T_{Para} \approx N T_\mathscr{G} + \sum_{k=1}^{K_{Para}} (T_\mathscr{F} + (N-k) T_\mathscr{G})$$

$$= K_{Para} T_\mathscr{F} + (K_{Para} + 1)(N - K_{Para}/2) T_\mathscr{G}$$

While the parallel speed-up, compared to the (serial) fine solver:

$$S_{Para} = \frac{T_{Serial}}{T_{Para}} \approx \left[\frac{K_{Para}}{N} + (K_{Para} + 1) \left(1 - \frac{K_{Para}}{2N} \right) \frac{T_\mathscr{G}}{T_\mathscr{F}} \right]^{-1}.$$

Parareal is faster when $K_{Para} < N$ and $T_\mathscr{G}/T_\mathscr{F} << 1$.

How can we improve on this? Keep $\mathscr{G}$ fixed and reduce K_{Para}.

GPareal (Pentland et al. (2023))

GParareal (Pentland et al. (2023))

Pentland et al. (2023) change the update criterion of the initial conditions, resulting in a new technique called GParareal.

Parareal uses information calculated during the previous iteration k,

$$U_i^{k+1} = \mathscr{G}\left(U_{i-1}^{k+1}\right) + \mathscr{F}\left(U_{i-1}^k\right) - \mathscr{G}\left(U_{i-1}^k\right), \quad i = 1, \ldots, N, \text{ (3)}$$

GParareal uses information from the current iteration $k + 1$,

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

This would require a serial computation of $\mathscr{F}\left(U_{i-1}^{k+1}\right)$. Instead, a Gaussian process is used to infer the first term from data.

GParareal (Pentland et al. (2023))

System	Parareal	GParareal*	GParareal
FitzHugh–Nagumo (FHN)	11	5	5
Rossler	18	13	13
Hopf	19	10	10
Brusselator	19	NA	20
Lorenz	15	NA	11
Double Pendulum	15	10	10

Comparison of performance for common ODE systems in the literature, described in Slides 57-63.

GParareal* refers to the original approach (Pentland et al., 2023), while GParareal refers to our implementation.

'NA' stands for not available as not considered by the reference. The results have been produced using accuracy $\epsilon = 5e^{-7}$.

Review: Gaussian Processes

Consider a dataset $D = \{(x, y)_i\}_{i=1}^n$ where

$$y_i = f(x_i) + \epsilon \in \mathbb{R}, \quad x_i \in \mathbb{R}^d,$$

and ϵ an additive i.i.d. Gaussian noise of variance σ_n^2. We want to learn the function f. We use Gaussian processes for this.

Definition 1 (Gaussian Process (GP))

A Gaussian process is a collection of random variables any finite number of which have a joint Gaussian distribution. It is uniquely identified by the mean function and the covariance function.

Here we take:

▶ Mean function $m(x) = 0$.

▶ Covariance function $Cov(f(x), f(x')) = k(x, x')$, where the kernel $k(\cdot, \cdot)$ is the squared exponential

$$k(x, x') = \exp(\|x - x'\|_2^2 / \sigma_s^2). \tag{5}$$

Review: Gaussian Processes

The observational noise σ_n and the kernel bandwidth σ_s control the performance of the method upon prediction. They are learned from the data by maximizing the marginal log-likelihood

$$\log p(\mathbf{y}|\mathbf{x}) = -\frac{1}{2}(\mathbf{y}^T(K(\mathbf{x},\mathbf{x}) + \sigma_n^2 I_n)^{-1}\mathbf{y} + \log|K(\mathbf{x},\mathbf{x})| + n\log 2\pi),$$

where $|\cdot|$ is the determinant.

After having trained the model, we can use it to make a prediction at a new point x^*, conditional on the observed data D. This can be obtained through the posterior distribution $y|x^*$, which is normal with mean

$$K(x^*,\mathbf{x})\left[K(\mathbf{x},\mathbf{x}) + \sigma_n^2 I\right]^{-1}\mathbf{y},$$

where I_n is the identity matrix of size n, and $K(\mathbf{x},\mathbf{x})$ is the kernel matrix of size $n \times n$ having $K(\mathbf{x},\mathbf{x})_{i,j} = k(x_i, x_j)$. Note the need to invert the kernel matrix $K(\cdot,\cdot)$, at a computational cost of $O(n^3)$.

GParareal - Computational cost

The runtime cost of GParareal is

$$T_{GPara*} \approx NT_{\mathscr{G}} + \sum_{k=1}^{K_{GPara*}} \left(T_{\mathscr{F}} + (N - k)T_{\mathscr{G}} + T_{GP*}(k) \right)$$

$$= K_{GPara*}T_{\mathscr{F}} + (K_{GPara*} + 1)(N - K_{GPara*}/2)T_{\mathscr{G}} + T_{GP*},$$

where

$$T_{GP*} := \sum_{k=1}^{K_{GPara*}} T_{GP*}(k),$$

and $T_{GP*}(k)$ is the wallclock time expended in using the model at iteration k.

Given the cubic cost of matrix inversion for fitting a GP, and the dataset size of the order $O(kN)$ by iteration k, we have

$$T_{GP*} = \sum_{k=1}^{K_{GPara*}} O(k^3 N^3) = O(K_{GPara*}^4 N^3).$$

GParareal - Computational cost

The speed-up is

$$S_{GPara*} \approx \left[\frac{K_{GPara*}}{N} + (K_{GPara*} + 1) \left(1 - \frac{K_{GPara*}}{2N} \right) \frac{T_{\mathscr{G}}}{T_{\mathscr{F}}} + \frac{T_{GP*}}{N T_{\mathscr{F}}} \right]^{-1}.$$

When $K_{Para} = K_{GPara*}$, to achieve the same speed-up S_{Para}, we require the total cost of the GP to be negligible compared to that of the serial procedure.

Finally, note that the maximum speed-up achievable by any parallel procedure that converges in K iterations, given by

$$S_{UB} = \frac{K}{N}.$$

GParareal - Performance - Hopf bifurcations

To showcase the empirical performance of GParareal, consider a non-linear model for the study of Hopf bifurcations (Seydel, 2009, pg. 72; also Slide 59), defined by the following equations

$$\frac{du_1}{dt} = -u_2 + u_1\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad \frac{du_2}{dt} = u_1 + u_2\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad (6)$$

where we note the dependence on time. In practice, we add time as an additional coordinate yielding a $d = 3$ autonomous system.
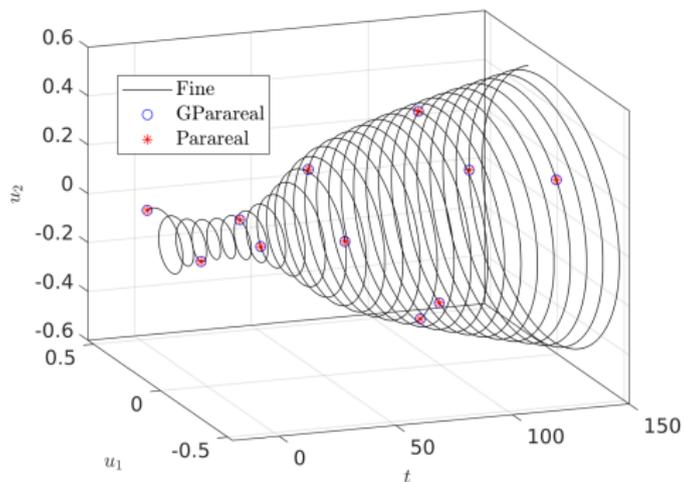
GParareal - Performance - Hopf bifurcations
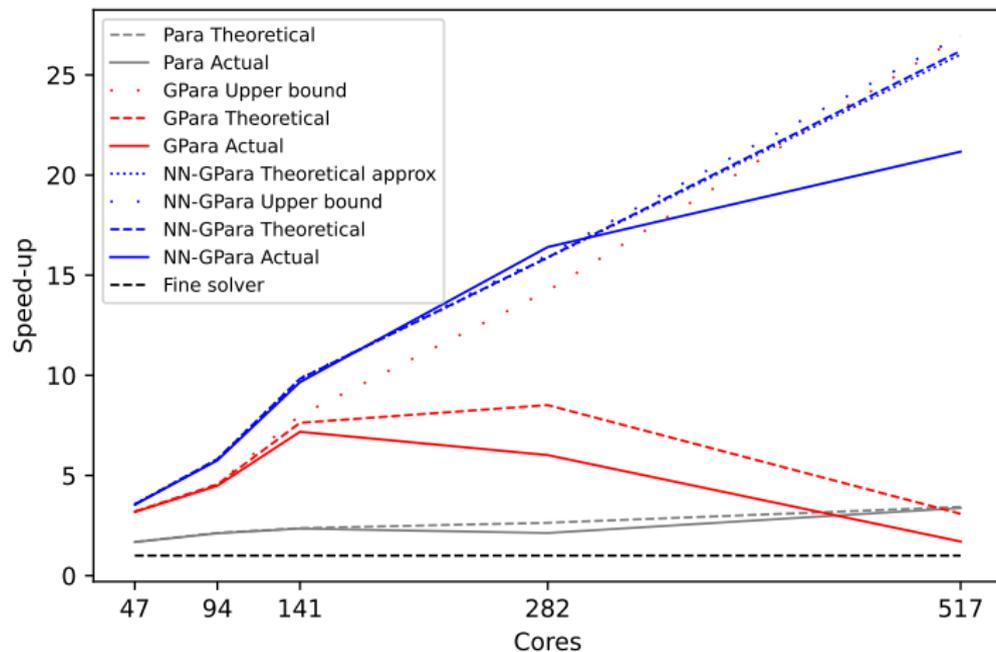
GParareal - Improvements

How can we improve? Maintain $K \leq K_{GPara*}$ while reducing T_{GP*}.

The GP cost comes from the sample size $O(Nk)$ by iteration k.
Can we reduce the sample size without affecting performance?

Yes, we can fit the model using a small subset consisting of the *nearest neighbors* to the prediction point. This is sufficient to smooth locally because very few points are empirically close in Euclidean distance.

Nearest Neighbor GParareal
(NN-GParareal)

NN-GParareal - Key Points

- Whereas GParareal trains the GP once per iteration k using the full dataset D, NN-GParareal is re-trained every time a prediction is made and it uses a subset $D' \subset D$ of the dataset D, with cardinality $|D'| = m$.

- Empirically, a fixed small value of $m \in \{15, ..., 20\}$ is sufficient for comparable performance to training on the whole D.

- Empirically, choosing the m observations to be the nearest neighbors (NN) of the prediction point in Euclidean distance has at least the same performance as other reasonable approaches.

- This model is known as nearest neighbor Gaussian process (NNGP) in the literature.

- Re-training at every prediction makes the GP globally non-stationary, without the need to change the kernel.

NN-GParareal - Performance

System	Parareal	GParareal*	GParareal	NN-GParareal
FitzHugh–Nagumo	11	**5**	**5**	**5**
Rossler	18	13	13	**12**
Hopf	19	10	10	**9**
Brusselator	19	NA	20	**17**
Lorenz	15	NA	11	**9**
Double Pendulum	15	**10**	**10**	**10**

Comparison of performance for common ODE systems in the literature, described in Slides 57-63.

GParareal* refers to the original approach (Pentland et al., 2023), while GParareal refers to our implementation.

'NA' stands for not available as not considered by the reference. The results have been produced using accuracy $\epsilon = 5e^{-7}$.

NN-GParareal - Computational Cost

Assuming NN-GParareal converges in K_{NN} iterations, we have

$$T_{NN-GPara} \approx NT_{\mathscr{G}} + \sum_{k=1}^{K_{NN}} \left(T_{\mathscr{F}} + (N-k)T_{\mathscr{G}} + T_{NNGP}(k) \right)$$
$$= K_{NN}T_{\mathscr{F}} + (K_{NN}+1)(N - K_{NN}/2)T_{\mathscr{G}} + T_{NNGP},$$

where $T_{NNGP} := \sum_{k=1}^{K_{NN}} T_{NNGP}(k)$, and $T_{NNGP}(k)$ is the cost of using the model during iteration k,

$$T_{NNGP}(k) = (N-k)T_{NNGP}^m,$$

T_{NNGP}^m is the cost of using the model to make a single prediction, including training. It is virtually constant across k and can be easily estimated beforehand. This follows from the constant matrix size $m \times m$ to be inverted.

NN-GParareal - Computational Cost

The speed-up for NN-GParareal is

$$S_{NN\text{-}GPara} \approx \left[\frac{K_{NN}}{N} + (K_{NN}+1)\left(1 - \frac{K_{NN}}{2N}\right)\frac{T_{\mathscr{G}}}{T_{\mathscr{F}}} + \frac{K_{NN}T_{NNGP}^m}{NT_{\mathscr{F}}}(N - (K_{NN}+1)/2) \right]^{-1}$$

The speed-up doesn't immediately clarify whether this model is cheaper than a normal GP. However, for a small, fixed m, the computational complexity is loglinear in N

$$T_{NNGP} = \sum_{k=1}^{K_{NN}}(N-k)T_{NNGP}^m = \sum_{k=1}^{K_{NN}}(N-k)[O(m^3) + O(\log(kn))]$$
$$= O(K_{NN}Nm^3) + O(K_{NN}N\log(K_{NN}N)).$$

The log term comes from the nearest neighbor computation.

NN-GParareal - Performance - Hopf bifurcations

We explore the performance of Parareal and its variants on a high-dimensional system. We use the two-dimensional, non-linear FitzHugh-Nagumo PDE model (Ambrosio and Françoise, 2009). See also Slide 67.

It represents a set of cells constituted by a small nucleus of pacemakers near the origin immersed among an assembly of excitable cells. The simpler FHN ODE system only considers one cell and its corresponding spike generation behavior.

We discretize both spatial dimensions using finite difference and $\tilde{d}$ equally spaced points, yielding an ODE with $d = 2\tilde{d}^2$ dimensions.

We consider $\tilde{d} = 10, 12, 14, 16$, corresponding to $d = 200, 288, 392, 512$, and set $N = 512$.

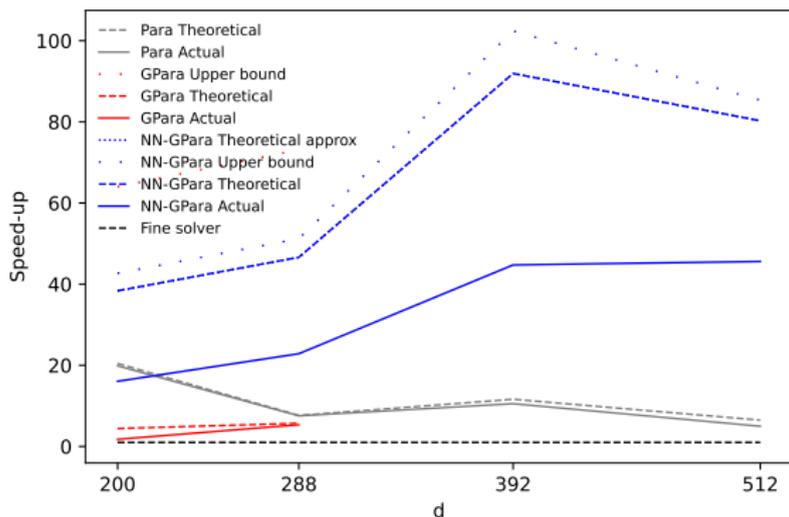# NN-GParareal - Performance - FitzHugh-Nagumo PDE

Figure: Plot of speed-ups for Parareal and its variants for the FitzHugh-Nagumo PDE model. The speed-ups are computed according to the formulas above. For $N = 256, 512$, GParareal failed to converge within the computational time budget.

References I

📄 Ambrosio, Benjamin and Jean-Pierre Françoise (2009). "Propagation of bursting oscillations". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1908, pp. 4863–4875.

📄 Danby, J.M. Anthony (1997). *Computer modeling: from sports to spaceflight... from order to chaos.*

📄 Deneubourg, Jean-Louis and Simon Goss (1989). "Collective patterns and decision-making". In: *Ethology Ecology & Evolution* 1.4, pp. 295–311.

📄 Fornberg, Bengt (1988). "Generation of finite difference formulas on arbitrarily spaced grids". In: *Mathematics of computation* 51.184, pp. 699–706.

📄 Gilpin, William (2021). "Chaos as an interpretable benchmark for forecasting and data-driven modelling". In: *arXiv preprint arXiv:2110.05266*.

References II

📄 Gorynina, Olga et al. (2022). "Combining machine-learned and empirical force fields with the parareal algorithm: application to the diffusion of atomistic defects". In: *arXiv preprint arXiv:2212.10508*.

📄 Kauffman, Stuart A. (1993). *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA.

📄 Lefever, René and Grégoire Nicolis (1971). "Chemical instabilities and sustained oscillations". In: *Journal of theoretical Biology* 30.2, pp. 267–284.

📄 Lions, Jacques-Louis, Yvon Maday, and Gabriel Turinici (2001). "Résolution d'EDP par un schéma en temps pararéel". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 332.7, pp. 661–668.

📄 Lorenz, Edward N. (1963). "Deterministic nonperiodic flow". In: *Journal of atmospheric sciences* 20.2, pp. 130–141.

References III

Nagumo, Jinichi, Suguru Arimoto, and Shuji Yoshizawa (1962). "An active pulse transmission line simulating nerve axon". In: *Proceedings of the IRE* 50.10, pp. 2061–2070.

Pentland, Kamran et al. (2023). "GParareal: a time-parallel ODE solver using Gaussian process emulation". In: *Statistics and Computing* 33.1, p. 23.

Rasmussen, Steen et al. (1990). "The coreworld: Emergence and evolution of cooperative structures in a computational chemistry". In: *Physica D: Nonlinear Phenomena* 42.1-3, pp. 111–134.

Rössler, Otto E. (1976). "An equation for continuous chaos". In: *Physics Letters A* 57.5, pp. 397–398.

Samaddar, Debasmita et al. (2019). "Application of the parareal algorithm to simulations of ELMs in ITER plasma". In: *Computer Physics Communications* 235, pp. 246–257.

Seydel, Rüdiger (2009). *Practical bifurcation and stability analysis*. Vol. 5. Springer Science & Business Media.

References IV

Thomas, René (1999). "Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis," labyrinth chaos"'". In: *International Journal of Bifurcation and Chaos* 9.10, pp. 1889–1905.

Vecchia, Aldo V. (1988). "Estimation and model identification for continuous spatial processes". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 50.2, pp. 297–312.

Yang, Lu et al. (2023). "Learning Dynamical Systems from Data: A Simple Cross-Validation Perspective, Part V: Sparse Kernel Flows for 132 Chaotic Dynamical Systems". In: *arXiv preprint arXiv:2301.10321*.

We run Parareal and its variants using a variable number of intervals N over $t \in [-20, 500]$. In the table:

- K is the number of iterations to convergence.
- $\mathscr{F}$ and $\mathscr{G}$ are the cost per iteration of the fine (accurate, slow) and coarse (less accurate, fast) respectively.
- 'Model' is the cost of training and inference for the learner used.
- 'Total' is the overall running time.
- 'Speed-up' is the empirical speed-up, the ratio of the serial solver ($\mathscr{F}$) to the parallel algorithm.

All entries are in seconds.

GParareal - Performance - Hopf bifurcation

NN-GParareal: our contribution. It trains the GP on a fixed, small subset of the data to drastically reduce the cost. More details later.

Non-linear Hopf bifurcation model, $N = 32$

Model	K	$T_\mathcal{G}$	$T_\mathcal{F}$	Model	Total	Speed-up
Fine	-	-	-	-	3.51e+04	1
Parareal	19	3.70e-02	1.09e+03	1.96e-03	2.08e+04	1.69
GParareal	10	6.24e-02	1.09e+03	6.40e+00	1.09e+04	3.21
NN-GParareal	9	1.53e-02	1.09e+03	4.71e+00	9.80e+03	**3.58**

Non-linear Hopf bifurcation model, $N = 64$

Model	K	$T_\mathcal{G}$	$T_\mathcal{F}$	Model	Total	Speed-up
Fine	-	-	-	-	3.51e+04	1
Parareal	30	3.42e-02	5.50e+02	6.37e-03	1.65e+04	2.13
GParareal	14	5.31e-02	5.52e+02	5.55e+01	7.78e+03	4.52
NN-GParareal	11	3.38e-02	5.50e+02	1.05e+01	6.06e+03	**5.80**

GParareal - Performance - Hopf bifurcation

Non-linear Hopf bifurcation model, $N = 128$

Model	K	$T_{\mathscr{G}}$	$T_{\mathscr{F}}$	Model	Total	Speed-up
Fine	-	-	-	-	3.51e+04	1
Parareal	54	5.29e-02	2.72e+02	2.36e-02	1.47e+04	2.40
GParareal	16	8.49e-02	2.73e+02	4.31e+02	4.79e+03	7.33
NN-GParareal	13	6.68e-02	2.72e+02	3.28e+01	3.56e+03	**9.86**

Non-linear Hopf bifurcation model, $N = 256$

Model	K	$T_{\mathscr{G}}$	$T_{\mathscr{F}}$	Model	Total	Speed-up
Fine	-	-	-	-	3.51e+04	1
Parareal	97	8.88e-02	1.96e+02	7.63e-02	1.90e+04	1.85
GParareal	18	1.36e-01	1.37e+02	3.24e+03	5.72e+03	6.15
NN-GParareal	16	1.19e-01	1.36e+02	1.08e+02	2.28e+03	**15.42**

Non-linear Hopf bifurcation model, $N = 512$

Model	K	$T_{\mathscr{G}}$	$T_{\mathscr{F}}$	Model	Total	Speed-up
Fine	-	-	-	-	3.51e+04	1
Parareal	149	1.55e-01	6.80e+01	1.97e-01	1.02e+04	3.46
GParareal	19	2.41e-01	6.95e+01	1.88e+04	2.01e+04	1.75
NN-GParareal	19	2.50e-01	7.06e+01	2.68e+02	1.62e+03	**21.75**

NN-GParareal - Intuition

Each plot is the prediction error incurred by the model across intervals i and iterations k. The blue line is that of Parareal, while the gray one is of 1-nearest neighbor, a learning model that predicts using exclusively the value of the closest observation.
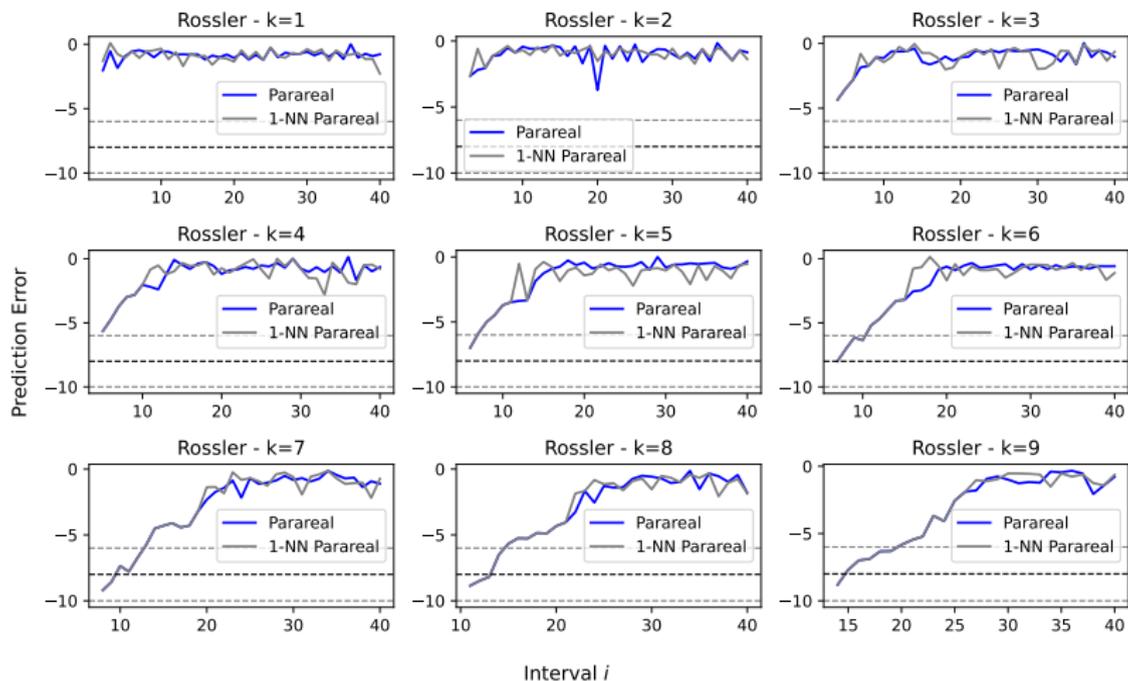
NN-GPararreal - Intuition
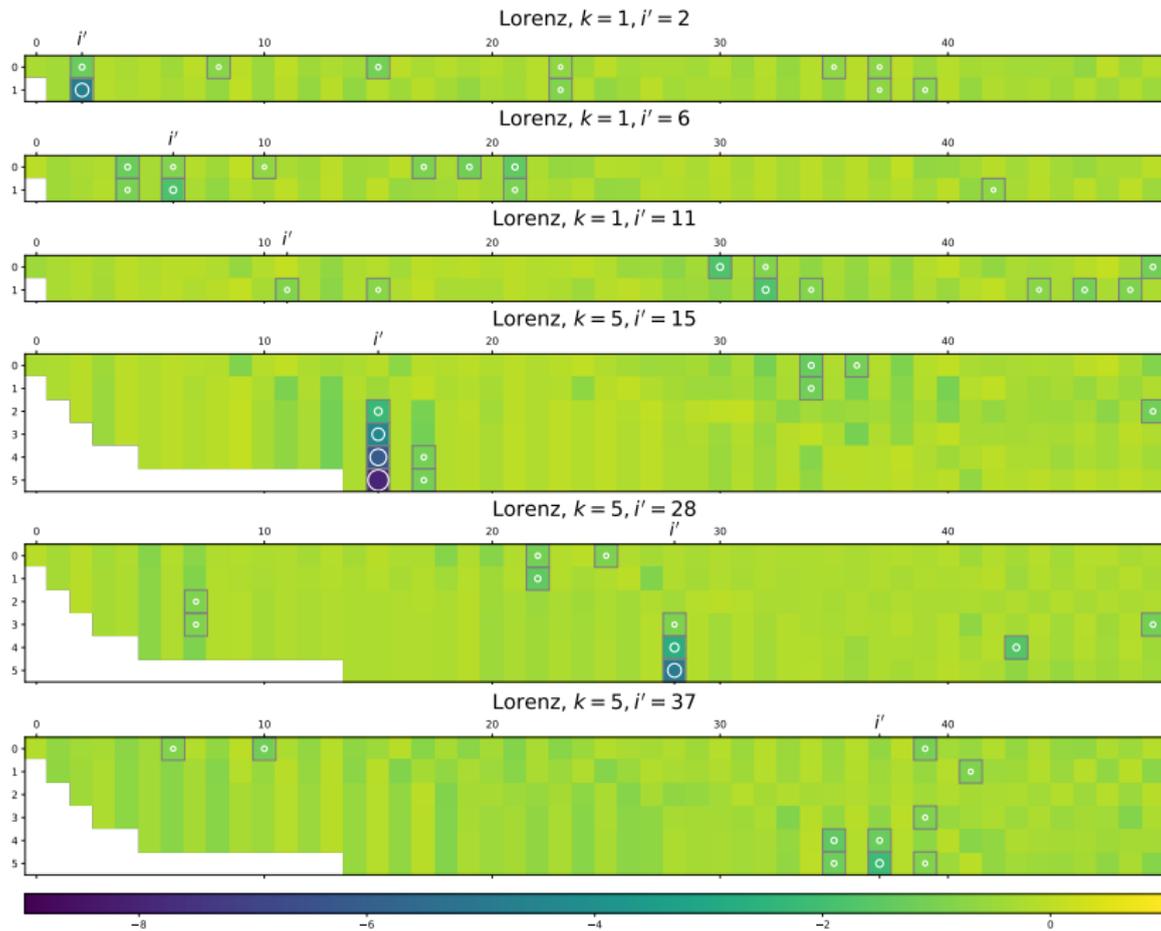
NN-GParareal - Choosing the data subset - Heuristics

System	NN	Col + rnd	Col only	Row + col	Row-major	Col-major
FHN	5	8	8	8	10	7
Rossler	12	14	17	17	21	16
Hopf	10	10	10	10	30	10
Brusselator	17	19	Exc	Exc	Exc	20
Lorenz	10	13	13	13	12	13
Double Pendulum	10	11	15	12	13	13

Table: Simulation results for heuristic choices of data subset for NN-GParareal. 'Exc' failed to converge.

- ▶ *NN*. The nearest neighbors. Taken as reference.
- ▶ *Col + rnd*. Take the complete history (a column) $(x_{j,i})_{j=1}^{k}$ up to m, and distribute any remaining neighbors $m - k$ randomly.
- ▶ *Col only*. Take the complete history. Note that this sets $m = k$.
- ▶ *Row + col*. Expand radially by striking a trade-off between previous iterations (column entries) and nearby intervals (row entries).
- ▶ *Row-major*. Give priority to nearby intervals, thus expanding horizontally across columns first.
- ▶ *Column-major*. Give priority to previous iterations, thus expanding vertically across rows first.

NN-GParareal - Choosing the data subset - Learning

We enrich the observation $x_{k,i}$ by including information about the current interval i and iteration k, obtaining $\mathbf{z}_{k,i} = (x_{k,i}, i, k)$. We employ the following compositional kernel:

$$
\begin{aligned}
K(\mathbf{z}_1, \mathbf{z}_2) &= K((x_1, i_1, k_1), (x_2, i_2, k_2)) \\
&= 10^{\sigma_v} K_1(\mathbf{z}_1, \mathbf{z}_2) K_2(\mathbf{z}_1, \mathbf{z}_2) K_3(\mathbf{z}_1, \mathbf{z}_2),
\end{aligned}
$$

with

$$
\begin{aligned}
K_1(\mathbf{z}_1, \mathbf{z}_2) &= \exp\{-0.5 \cdot 10^{-\sigma_s} \|x_1 - x_2\|_2^2\}, \\
K_2(\mathbf{z}_1, \mathbf{z}_2) &= \exp\{-0.5 \cdot 10^{-\sigma_i} \|i_1 - i_2\|_2^2\}, \\
K_3(\mathbf{z}_1, \mathbf{z}_2) &= \exp\{-0.5 \cdot 10^{-\sigma_k} \|k_1 - k_2\|_2^2\},
\end{aligned}
$$

where σ_v captures the standard deviation of the process, and $\sigma_s, \sigma_i, \sigma_k$ control the relative importance of the spatial, temporal-across-intervals and temporal-across-iterations dimensions.

As proposed by Vecchia (1988), we can rank observations based on the kernel score with respect to the prediction point $\mathbf{z}^*$, $k(\cdot, \mathbf{z}^*)$. Then, choose the top m. Since the optimal σs depend on m, we follow the iterative procedure:

1. Propose a random subset of size m, D'
2. Compute the optimal σs given D'
3. Rank the observations using $k(\cdot, \cdot)$ based on the optimal θs
4. Propose a new subset from the top m observations
5. Repeat steps 2-4 until the subsets stabilize

This procedure is computationally expensive due to its iterative nature.

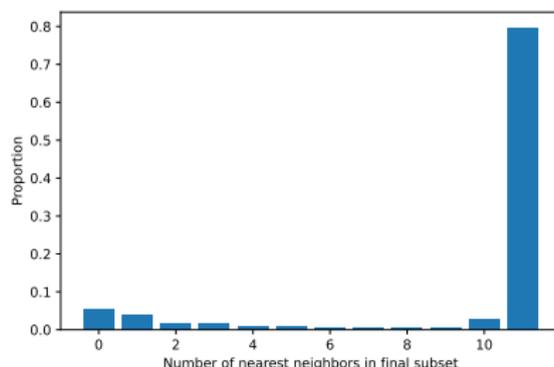# NN-GParareal - Choosing the data subset - Learning

Figure: Visualization of the converged subset selection for the NN-GParareal with time extension algorithm run on Lorenz. Histogram of the distribution of the percentage of points in each converged subset that match the nearest neighbors, aggregating over.
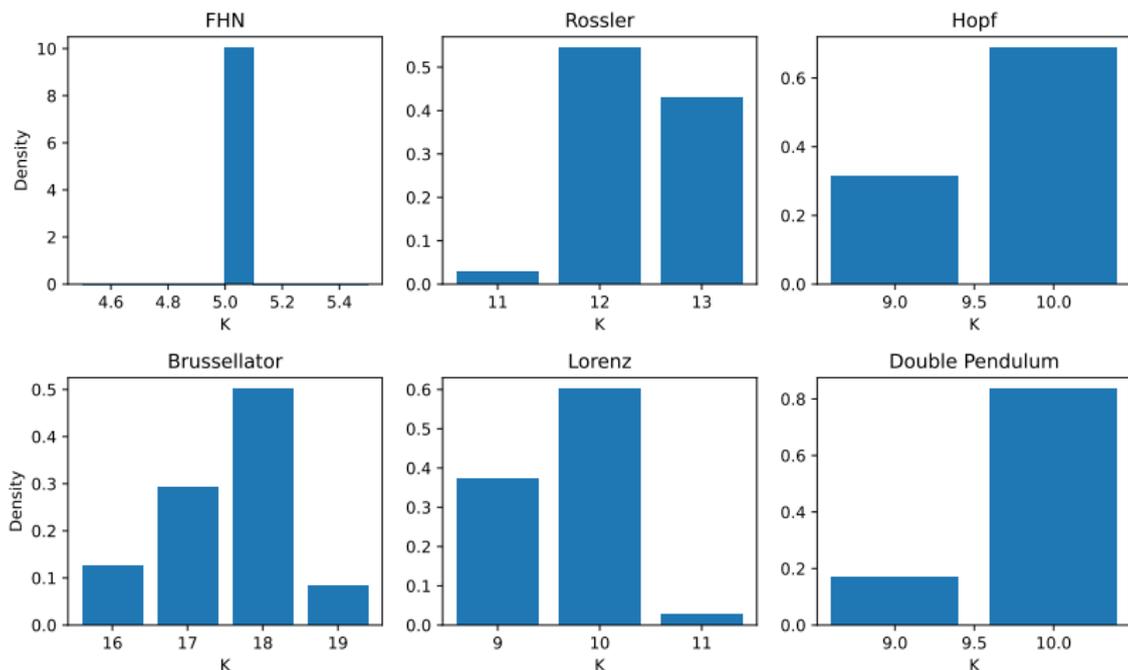
Choosing m

Figure: Histogram of K for several systems obtained using seven values of m, between 10 and 20, and five random seeds. Note how concentrated the empirical distributions are, guaranteeing consistent performance regardless of the value of m and the specific execution.

NN-GParareal - More results: Thomas Labyrinth

Finally, we consider Thomas Labyrinth (Gilpin, 2021), a chaotic system reportedly difficult to learn by a variety of kernel methods (Yang et al., 2023). For $N = 256$ and $N = 512$ GParareal failed to converge within 48 hours, intermediate results have been placed instead. This doesn't affect the conclusions.

NN-GParareal - More results: Thomas Labyrinth
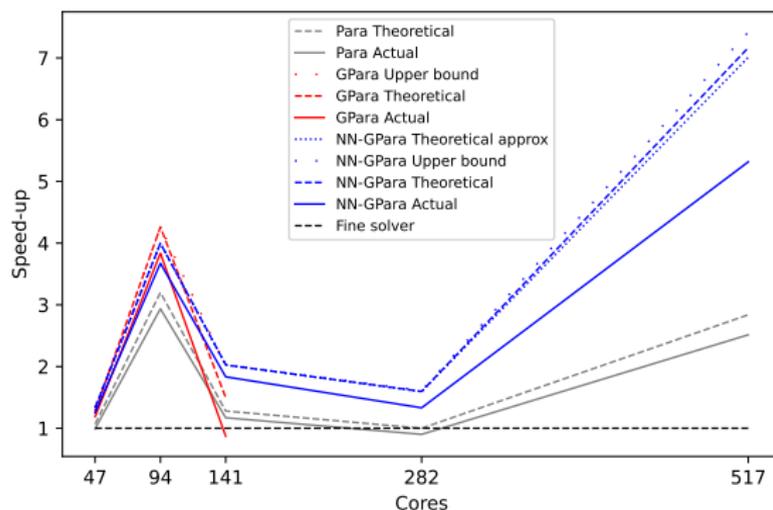

Figure: Plot of speed-ups for Parareal and its variants for Thomas labyrinth. The speed-ups are computed according to the formulas in Section **??**. GParareal failed to converge within the time limit for $N = 256, 512$, hence the missing data.

NN-GParareal - More results: Thomas Labyrinth

Figure: GParareal's percentage of converged intervals (red line) and the training time per iteration (blue line) for Thoams labyrinth. The aggregated model cost across k is shown in the title. Note that $N = 256, 512$ failed to converge within the computational time budget. The model cost per iteration is increasing with k.
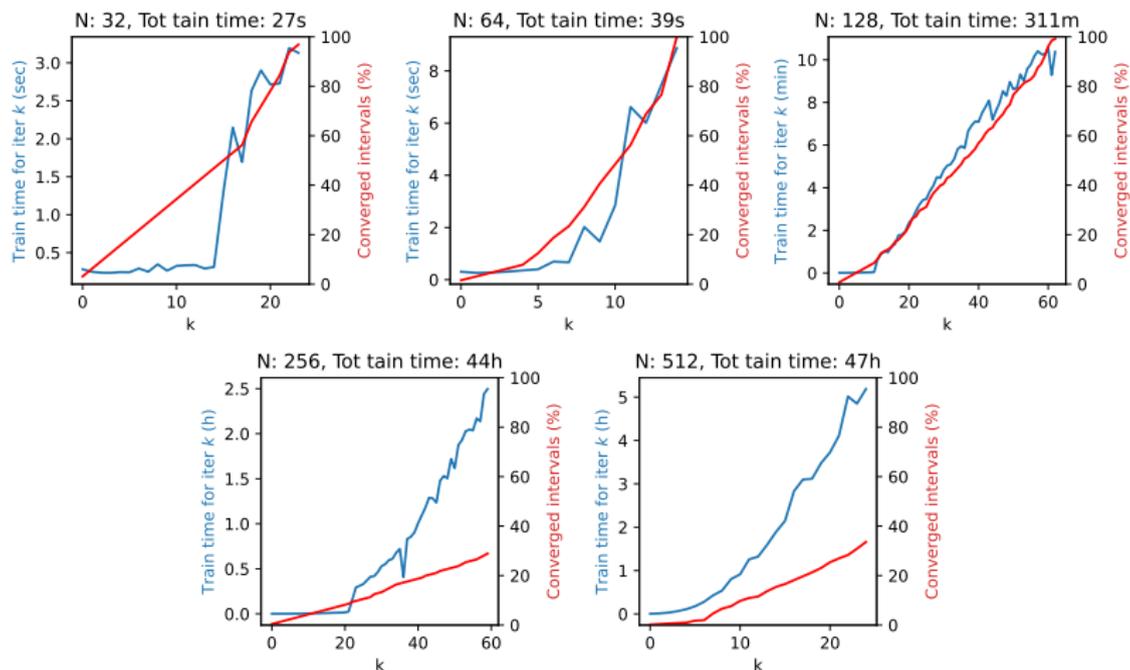
NN-GParareal - More results: Thomas Labyrinth
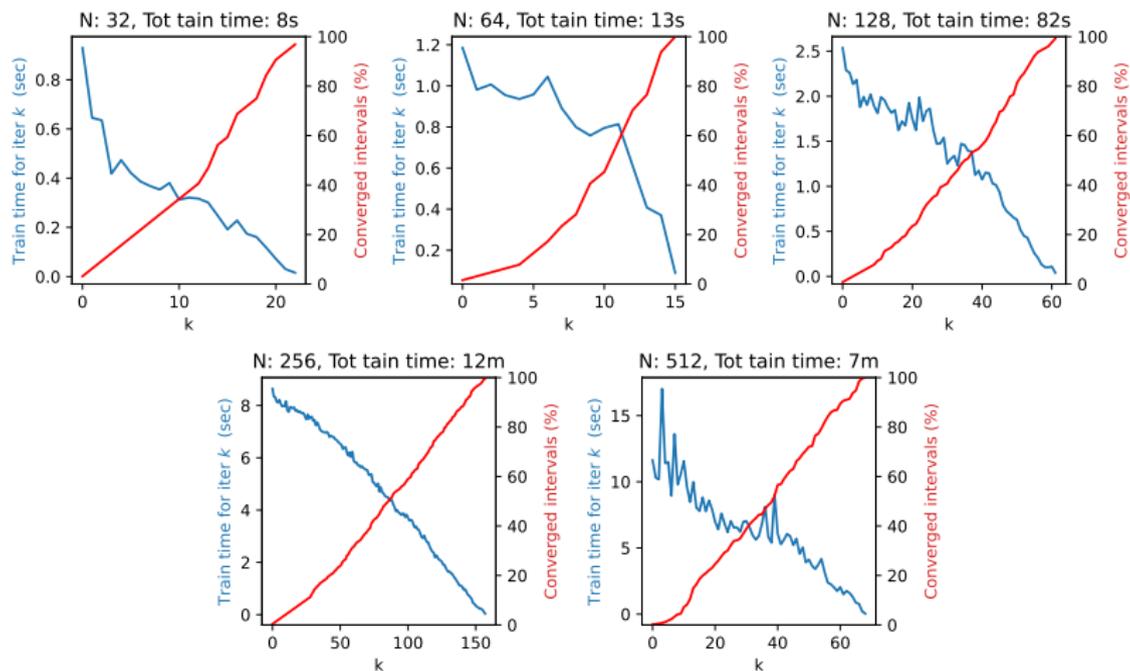
Figure: NN-GParareal's percentage of converged intervals (red line) and the training time per iteration (blue line) for Thoams labyrinth. The aggregated model cost across k is shown in the title. The model cost per iteration is decreasing with k.

NN-GParareal - More results: Thomas Labyrinth

Figure: NN-GParareal, empirical distribution of K across values of m for Thomas labyrinth. 200 independent runs for each m have been carried out. The shaded area indicates better performance than GParareal.
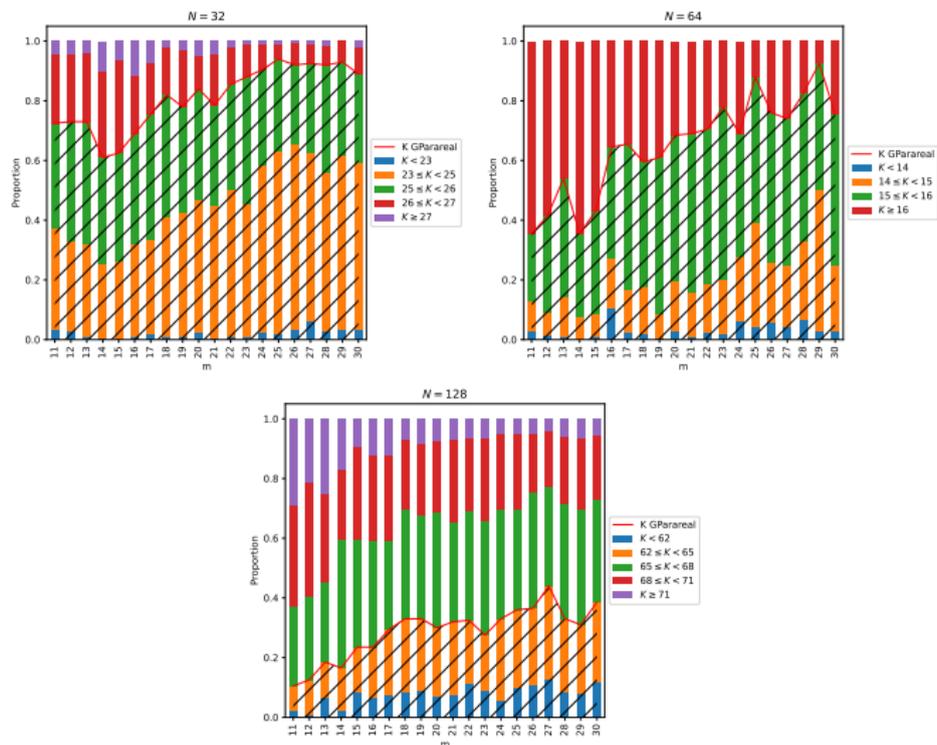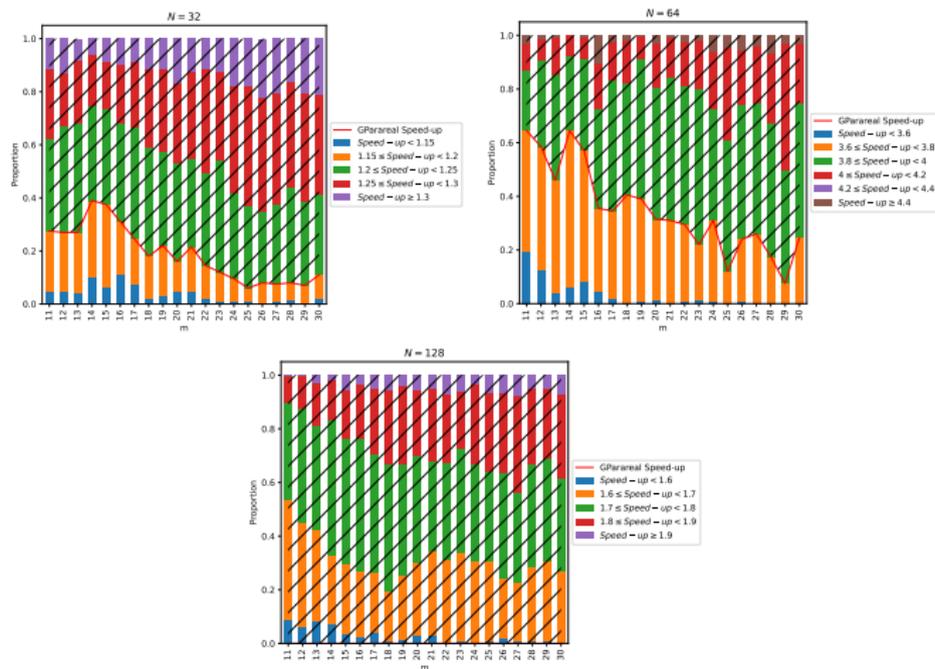
Figure: NN-GParareal, empirical distribution of the speed-up across values of *m* for Thomas labyrinth. 200 independent runs for each *m* have been carried out. The shaded area indicates better performance than GParareal.

ODE/PDE Systems

Systems: FitzHugh–Nagumo

The FitzHugh-Nagumo (FHN) is a model for an animal nerve axon (Nagumo et al., 1962). It is a reasonably easy system to learn, does not exhibit chaotic behavior, and is commonly used throughout the literature. It is described by the following equations

$$\frac{\mathrm{d}u_1}{\mathrm{d}t} = c\left(u_1 - \frac{u_1^3}{3} + u_2\right), \quad \frac{\mathrm{d}u_2}{\mathrm{d}t} = -\frac{1}{c}\left(u_1 - a + bu_2\right),$$

with $(a, b, c) = 0.2, 0.2, 3$. We integrate over $t \in [0, 40]$ using $N = 40$ intervals, taking $u_0 = (-1, 1)$ as the initial condition. We use Runge-Kutta 2 with 160 steps for the coarse solver $\mathscr{G}$, and Ruge-Kutta 4 with $1.6e^5$ steps for the fine solver $\mathscr{F}$. This is the same setting as Pentland et al. (2023), which allows almost direct comparison, although our system is the normalized version of the above, which also applies to u_0. We use a (normalized) error $\epsilon = 5e^{-7}$.

Systems: Rossler

The Rossler is a model for turbulence (Rössler, 1976)

$$\frac{\mathrm{d}u_1}{\mathrm{d}t} = -u_2 - u_3, \quad \frac{\mathrm{d}u_2}{\mathrm{d}t} = u_1 + \hat{a}u_2, \quad \frac{\mathrm{d}u_3}{\mathrm{d}t} = \hat{b} + u_3\left(u_1 - \hat{c}\right).$$

When $(\hat{a}, \hat{b}, \hat{c}) = (0.2, 0.2, 5.7)$, it exhibits chaotic behavior. This configuration is commonly used throughout the literature. We integrate over $t \in [0, 340]$ using $N = 40$ intervals, taking $u_0 = (0, -6.78, 0.02)$ as initial condition. We use Runge-Kutta 1 with $9e^4$ steps for the coarse solver $\mathscr{G}$, and Ruge-Kutta 4 with $4.5e^8$ steps for the fine solver $\mathscr{F}$. This is the same setting as Pentland et al. (2023), although, like above, we use the normalized version and set a normalized $\epsilon = 5e^{-7}$.

Systems: Non-linear Hopf bifurcation

This is a non-linear model for the study of Hopf bifurcations, see Seydel (2009, pg. 72) for a detailed explanation. The model is defined by the following equations

$$\frac{du_1}{dt} = -u_2 + u_1\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad \frac{du_2}{dt} = u_1 + u_2\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad (7)$$

where we note the dependence on time. To counter that, we add time as an additional coordinate, thus yielding a $d = 3$ system. We integrate over $t \in [-20, 500]$ using $N = 32$ intervals, taking $u_0 = (0.1, 0.1, 500)$ as initial condition. We use Runge-Kutta 1 with 2048 steps for the coarse solver $\mathscr{G}$, and Ruge-Kutta 8 with $5.12e^5$ steps for the fine solver $\mathscr{F}$. This is the same setting as Pentland et al. (2023), although, like above, we use the normalized version and set a normalized $\epsilon = 5e^{-7}$.

Systems: Brusselator

The Brusselator models an autocatalytic chemical reaction
(Lefever and Nicolis, 1971). It is a stiff, non-linear ODE, and the
following equations govern it

$$\frac{du_1}{dt} = A + u_1^2 u_2 - (B+1)u_1,$$
$$\frac{du_2}{dt} = Bu_1 - u_1^2 u_2,$$

where $(A, B) = (1, 3)$. We integrate over $t \in [0, 100]$ using $N = 32$
intervals, taking $u_0 = (1, 3.7)$ as initial condition. We use
Runge-Kutta 4 with $2.5e^2$ steps for the coarse solver $\mathscr{G}$, and
Ruge-Kutta 4 with $2.5e^4$ steps for the fine solver $\mathscr{F}$. We use the
normalized version and set a normalized $\epsilon = 5e^{-7}$.

Systems: Double pendulum

This is a model for a double pendulum, adapted from Danby (1997). It consists of a simple pendulum of mass m and rod length ℓ connected to another simple pendulum of equal mass m and rod length ℓ, acting under gravity g. The model is defined by the following equations

$$\frac{\mathrm{d}u_1}{\mathrm{d}t} = u_3,$$

$$\frac{\mathrm{d}u_2}{\mathrm{d}t} = u_4,$$

$$\frac{\mathrm{d}u_3}{\mathrm{d}t} = \frac{-u_3^2 f_1(u_1, u_2) - u_4^2 \sin(u_1 - u_2) - 2\sin(u_1) + \cos(u_1 - u_2)\sin(u_2)}{f_2(u_1, u_2)},$$

$$\frac{\mathrm{d}u_4}{\mathrm{d}t} = \frac{2u_3^2 \sin(u_1 - u_2) + u_4^2 f_1(u_1, u_2) + 2\cos(u_1 - u_2)\sin(u_1) - 2\sin(u_2)}{f_2(u_1, u_2)},$$

where

$$f_1(u_1, u_2) = \sin(u_1 - u_2)\cos(u_1 - u_2),$$
$$f_2(u_1, u_2) = 2 - \cos^2(u_1 - u_2).$$

Systems: Double pendulum

In the above, m, ℓ, and g have been scaled out of the system by letting $\ell = g$. The variables u_1 and u_2 measure the angles between each pendulum and the vertical axis, while u_3 and u_4 measure the corresponding angular velocities.

The system exhibits chaotic behavior and is commonly used in the literature. Based on the initial condition, it can be difficult to learn.

We integrate over $t \in [0, 80]$ using $N = 32$ intervals, taking $u_0 = (-0.5, 0, 0, 0)$ as initial condition. We use Runge-Kutta 1 with 3104 steps for the coarse solver $\mathcal{G}$, and Ruge-Kutta 8 with $2.17e^5$ steps for the fine solver $\mathcal{F}$. This is a similar setting as Pentland et al. (2023, Figure 4.10), although, like above, we use the normalized version and set a normalized $\epsilon = 5e^{-7}$.

Systems: Lorenz

The Lorenz system is a simplified model for weather prediction Lorenz, 1963. With the following parameters, it is a chaotic system governed by the equations

$$
\frac{du_1}{dt} = \gamma_1 \left(u_2 - u_1 \right),
$$
$$
\frac{du_2}{dt} = \gamma_2 u_1 - u_1 u_3 - u_2,
$$
$$
\frac{du_3}{dt} = u_1 u_2 - \gamma_3 u_3,
$$

with $(\gamma_1, \gamma_2, \gamma_3) = (10, 28, 8/3)$. We integrate over $t \in [0, 18]$ using $N = 50$ intervals, taking $u_0 = (-15, -15, 20)$ as initial condition. We use Runge-Kutta 4 with $3e^2$ steps for the coarse solver $\mathscr{G}$, and Ruge-Kutta 4 with $2.25e^4$ steps for the fine solver $\mathscr{F}$. We use the normalized version and set a normalized $\epsilon = 5e^{-7}$.

Systems: Thomas labyrinth

Thomas (1999) has proposed a particularly simple three-dimensional system representative of a large class of auto-catalytic models that occur frequently in chemical reactions (Rasmussen et al., 1990), ecology (Deneubourg and Goss, 1989), and evolution (Kauffman, 1993). It is described by the following equations

$$\begin{cases} \frac{dx}{dt} = b\sin y - ax, \\ \frac{dy}{dt} = b\sin z - ay, \\ \frac{dz}{dt} = b\sin x - az, \end{cases} \quad (8)$$

where $(a, b) = (0.5, 10)$. We integrate over $t \in [0, 10]$ for $N = 32, 64$, $t \in [0, 40]$ for $N = 128$, and $t \in [0, 100]$ for $N = 256, 512$ intervals. Following Gilpin (2021), we take

$$u_0 = (4.6722764, 5.2437205e^{-10}, -6.4444208e^{-10})$$

as initial condition, for which the system exhibits chaotic dynamics. Further, we use Runge-Kutta 1 with $10N$ steps for the coarse solver $\mathcal{G}$ and Ruge-Kutta 4 with $1e^9$ steps for the fine solver $\mathcal{F}$.

Systems: Viscous Burgers' equation

The viscous Burgers' equation is a fundamental PDE describing convection-diffusion occurring in various areas of applied mathematics. It is one-dimensional and defined as

$$v_t = \nu v_{xx} - v v_x \quad (x, t) \in (-L, L) \times (t_0, t_N], \tag{9}$$

with initial condition $v(x, t_0) = v_0(x), x \in [-L, L]$, and boundary conditions

$$v(-L, t) = v(L, t), \quad v_x(-L, t) = v_x(L, t), \quad t \in [t_0, T_N].$$

In the above, ν is the diffusion coefficient. We discretize the spatial domain using finite difference (Fornberg, 1988) and $d + 1$ equally spaced points $x_{j+1} = x_j + \Delta x$, where $\Delta x = 2L/d$ and $j = 0, ..., d$.

Systems: Viscous Burgers' equation

In the numerical experiments, we consider two values for the time horizon, $t_N = 5$ and $t_N = 5.9$, with $t_0 = 0$. We set $N = d = 128$ and take $L = 1$ and $\nu = 1/100$. The discretization and finite difference formulation imply that it is equivalent to solving a d-dimensional system of ODEs.

We take $v_0(x) = 0.5(\cos(\frac{9}{2}\pi x) + 1)$ as the initial condition. We use Runge-Kutta 1 with $4N$ steps for the coarse solver $\mathscr{G}$ and Ruge-Kutta 8 with $5.12e^6$ steps for the fine solver $\mathscr{F}$.

We use the normalized version with a normalized $\epsilon = 5e^{-7}$.

Systems: FitzHugh-Nagumo PDE

The two-dimensional, non-linear FitzHugh-Nagumo PDE model (Ambrosio and Françoise, 2009) is an extension of the ODE system in Slide 57. It represents a set of cells constituted by a small nucleus of pacemakers near the origin immersed among an assembly of excitable cells. The simpler FHN ODE system only considers one cell and its corresponding spike generation behavior.

It is defined as

$$
\begin{aligned}
v_t &= a\nabla^2 v + v - v^3 - w - c, \quad (x, t) \in (-L, L)^2 \times (t_0, t_N] \\
w_t &= \tau\left(b\nabla^2 w + v - w\right),
\end{aligned}
\tag{10}
$$

with initial conditions

$$
v(x, t_0) = v_0(x), w(x, t_0) = w_0(x), \quad x \in [-L, L],
$$

Systems: FitzHugh-Nagumo PDE

and boundary conditions

$$v((x, -L), t) = v((x, L), t)$$
$$v((-L, y), t) = v((L, y), t)$$
$$v_y((x, -L), t) = v_y((x, L), t)$$
$$v_x((-L, y), t) = v_x((L, y), t), \quad t \in [t_0, t_N].$$

The boundary conditions for w are equivalent and not repeated. We discretize both spatial dimensions using finite difference and $\tilde{d}$ equally spaced points, yielding an ODE with $d = 2\tilde{d}^2$ dimensions.

Systems: FitzHugh-Nagumo PDE

In the numerical experiments, we consider four values for $\tilde{d} = 10, 12, 14, 16$, corresponding to $d = 200, 288, 392, 512$. We set $N = 512$, $L = 1$, $t_0 = 0$, and take $v_0(x), w(0)$ randomly sampled from $[0,1]^d$ as the initial condition.

We use Ruge-Kutta 8 with 10^8 steps for the fine solver $\mathscr{F}$. We use the normalized version with a normalized $\epsilon = 5e^{-7}$.

The time span and coarse solvers depend on $\tilde{d}$, Table 2 describes their relation. This is to provide a realistic experiment where the user would need to adjust the coarse solver based on $t_N - t_0$.

Systems: FitzHugh-Nagumo PDE

d	$\mathscr{G}$	$\mathscr{G}$ steps	t_N
200	RK2	$3N$	$t_N = 150$
288	RK2	$12N$	$t_N = 550$
392	RK2	$25N$	$t_N = 950$
512	RK4	$25N$	$t_N = 1100$

Table: Simulation setup for the two-dimensional FitzHugh-Nagumo PDE. Adjusting the coarse solver based on the time horizon t_N makes the simulation more realistic.