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

Guglielmo Gattiglio
University of Warwick

June 7, 2024

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

Parareal [LMT01].

- Sketch of the procedure
- Computational cost

GParareal [Pen+23].

- Empirical results
- Computational cost

Nearest Neighbor GParareal New!

- Empirical results
- Computational cost

Random Weights Neural Networks Parareal Another talk

- Replace Gaussian process with neural networks
- Faster runtime
- No uncertainty quantification on the prediction

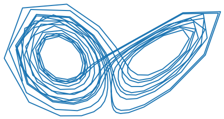
Introduction

In this talk, we consider machine-learning-based approaches to speed up Parareal [LMT01], a parallel-in-time solver for ODEs and PDEs. Why is time parallelization important?

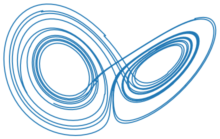
- Space parallelization has been a widely use technique for solving PDEs on multiple processors.
- In plasma physics and other fields, these traditional techniques often reach saturation on modern supercomputers, thus leaving time parallelization as the only avenue for improvement [Sam+19].
- Simulations 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 [Gor+22]

Solving Lorenz

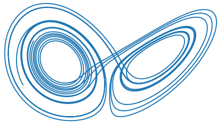
$\mathscr{G}$, coarse



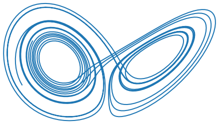
Parareal, midway



Parareal, final

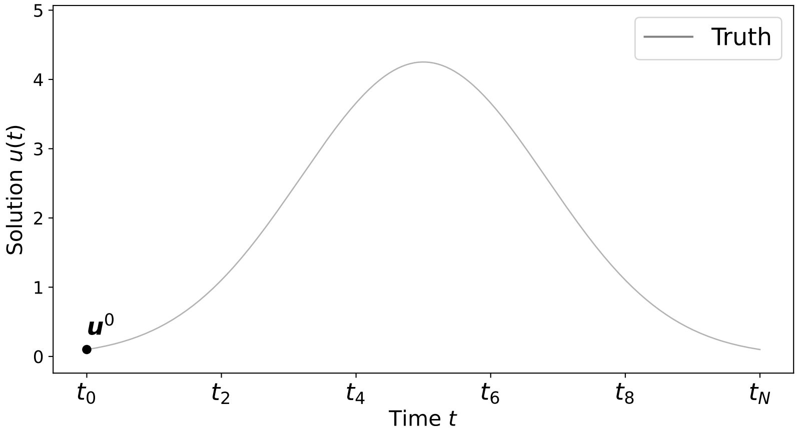


$\mathscr{F}$, fine



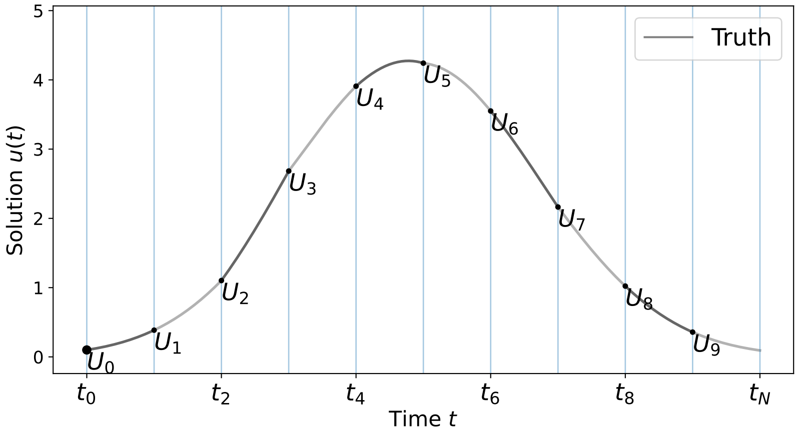
Parareal (Lions et al. [LMT01])

Parareal - Sketch of behavior - 1D System



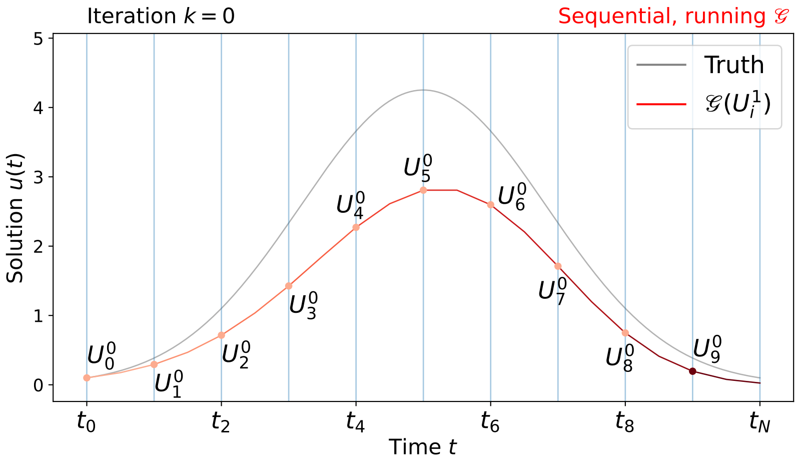
We wish to solve the d-dimensional ODE $\frac{d\boldsymbol{u}}{dt} = h(\boldsymbol{u}(t), t)$ on $t \in [t_0, t_N]$, with $\boldsymbol{u}(t_0) = \boldsymbol{u}^0, \quad N \in \mathbb{N}$, and $\boldsymbol{u}^0 \in \mathbb{R}^d$; here $d = 1$.

Parareal - Sketch of behavior - 1D System



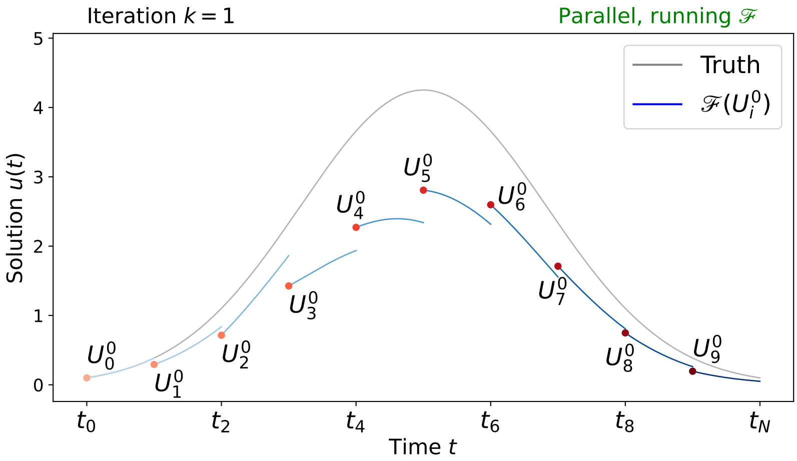
We divide the time interval in N sub-intervals and assign one processor for each. The initial conditions $\boldsymbol{U}_i$, $i = 1, \dots, N-1$ are unknown and need to be estimated.

Parareal - Sketch of behavior - 1D System



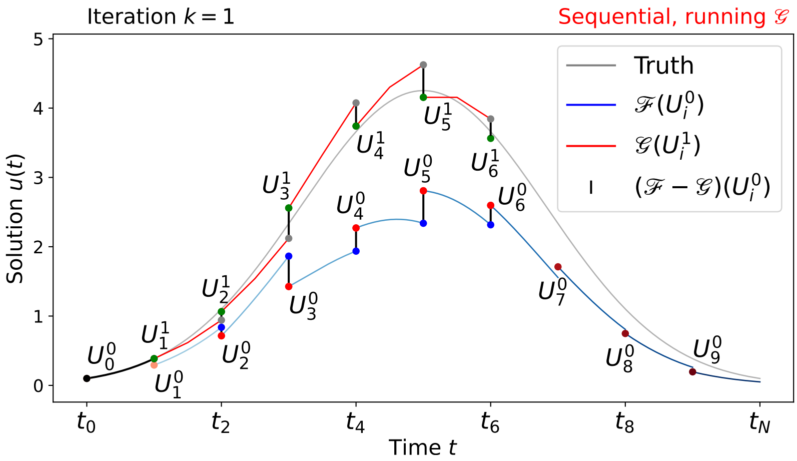
We run a cheap (fast), inaccurate coarse solver $\mathscr{G}$ to provide approximate initial conditions $\boldsymbol{U}_i^k$ for iteration $k = 0$ (initialization) and all intervals $i = 1, \ldots, N-1$.

Parareal - Sketch of behavior - 1D System



Once *some* initial conditions are available, we can run a precise (slow) fine solver $\mathscr{F}$ in *parallel* over the N processors, each started from an initial condition U_i^0.

Parareal - Sketch of behavior - 1D System



The initial conditions are then *sequentially* updated to satisfy a continuity condition, using the Parareal predictor-corrector rule

$$\boldsymbol{U}_i^k = \mathscr{G}(\boldsymbol{U}_{i-1}^k) + \mathscr{F}(\boldsymbol{U}_{i-1}^{k-1}) - \mathscr{G}(\boldsymbol{U}_{i-1}^{k-1})$$

A generic Parareal algorithm

The Parareal predictor-corrector rule can be generalized to a) use data from the current iteration $k+1$ and b) account for different ways of computing the discrepancy $\mathscr{F} - \mathscr{G}$

$$\boldsymbol{U}_i^k = \mathscr{G}(\boldsymbol{U}_{i-1}^k) + \widehat{f}(\boldsymbol{U}_{i-1}^k), \tag{1}$$

where $\widehat{f} : \mathbb{R}^d \to \mathbb{R}^d$ models $\mathscr{F} - \mathscr{G}$. Parareal uses

$$\widehat{f}_{\mathrm{Para}}(\boldsymbol{U}_{i-1}^k) = (\mathscr{F} - \mathscr{G})(\boldsymbol{U}_{i-1}^{k-1}).$$

To evaluate Parareal's performance we use the *parallel speed-up* $S_{\mathrm{alg}} := T_{\mathrm{Serial}}/T_{\mathrm{alg}}$, where

- T_{Serial} is the cost of running $\mathscr{F}$ *sequentially* over $[t_0, t_N]$
- T_{alg} is the runtime ofthe parallel procedure (Parareal)

A generic speed-up calculation

Assume that running $\mathscr{F}$ over one interval $[t_i, t_{i+1}]$ takes $T_{\mathscr{F}}$ time, and similarly for $\mathscr{G}$, and let K_{alg} be the iterations to convergence,

$$S_{\mathrm{alg}} \approx \left(\frac{K_{\mathrm{alg}}}{N} + \text{serial cost} \left(\frac{T_{\mathscr{G}}}{T_{\mathscr{F}}} \right) + \frac{T_{\mathrm{mdl}}}{N T_{\mathscr{F}}} \right)^{-1} \leq \left(\frac{K_{\mathrm{alg}}}{N} \right)^{-1} := S^*_{\mathrm{alg}},$$

where T_{mdl} is the overall cost of evaluating $\widehat{f}$.

Parareal has $T_{\mathrm{Para}} \in O(1)$ and is efficient 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}.

GParareal (Pentland et al. [Pen+23])

GParareal (Pentland et al. [Pen+23])

GParareal stores all the discrepancies $(\mathscr{F} - \mathscr{G})$ into a dataset $\mathcal{D}_k$ of cardinality $|\mathcal{D}_k| = NK$ by iteration k

$$\mathcal{D}_k := \{(\boldsymbol{U}_{i-1}^{j}, (\mathscr{F} - \mathscr{G})(\boldsymbol{U}_{i-1}^{j})), \ \ i = 1, \dots, N, \ \ j = 0, \dots, k-1\},$$

and models each coordinate $s = 1, \dots, d$ of $\widehat{f}(\boldsymbol{U}_{i-1}^{k})$ through a Gaussian process (GP), using the posterior mean μ

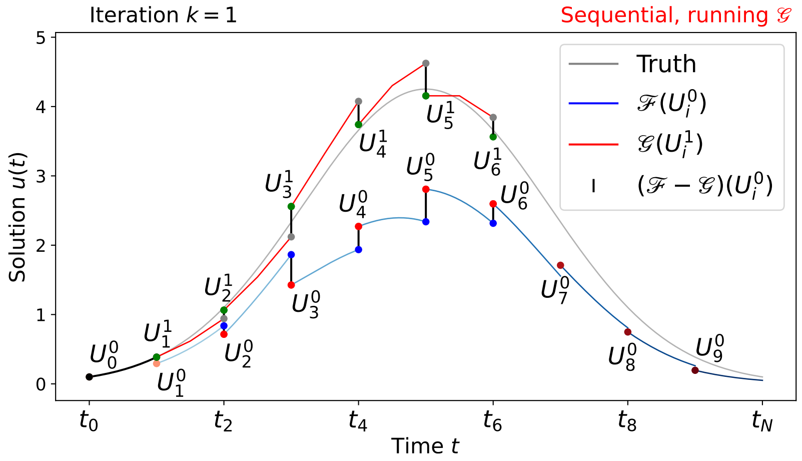
$$\widehat{f}_{\mathrm{GP}}(\boldsymbol{U}_{i-1}^{k})_s = \mu_{\mathcal{D}_k}^{(s)}(\boldsymbol{U}_{i-1}^{k}) \in \mathbb{R},$$

thereby training d different models, one per ODE coordinate.

Since evaluating μ involves inverting a $|\mathcal{D}_k| \times |\mathcal{D}_k|$ matrix, $T_{\mathrm{GP}}(k)$ scales as $O((d/N \vee 1)k^3 N^3)$, with $\vee$ the minimum operator, giving

$$T_{\mathrm{GP}} = \sum_{k=1}^{K_{\mathrm{GPara}}} T_{\mathrm{GP}}(k) \in O\left((d/N \vee 1)K_{\mathrm{GPara}}^4 N^3\right) > T_{\mathrm{Para}} \in O(1)$$

Recall



$$S_{\text{alg}} \approx \left(\frac{K_{\text{alg}}}{N} + \text{sequential cost} \left(\frac{T_{\mathscr{G}}}{T_{\mathscr{F}}} \right) + \frac{T_{\text{mdl}}}{N T_{\mathscr{F}}} \right)^{-1} \leq \left(\frac{K_{\text{alg}}}{N} \right)^{-1}$$

GParareal (Pentland et al. [Pen+23])

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

Comparison of performance for common ODE systems in the literature, described in Slides 37-43.

GParareal - Performance - Hopf bifurcations

To showcase the empirical performance of GParareal, consider a non-linear model for the study of Hopf bifurcations ([Sey09, pg. 72]; also Slide 39), defined by the following equations

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

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

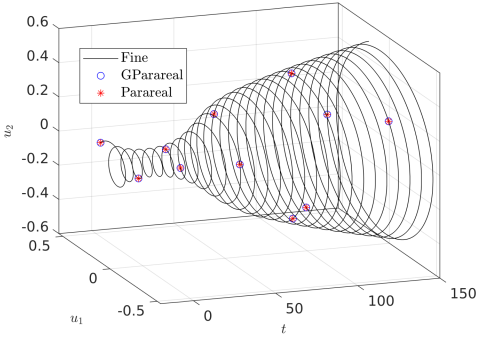
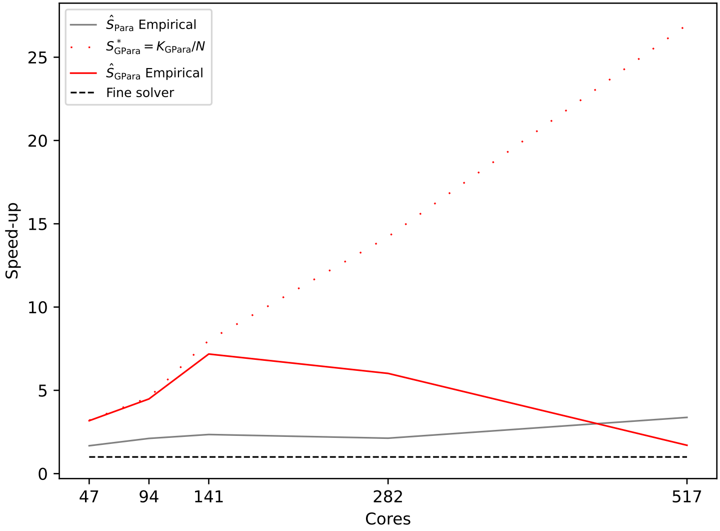


Image taken from Pentland et al. [Pen+23]

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 poin ts are empirically close in Euclidean distance.

Nearest Neighbor (nn) GParareal (nnGParareal)

nnGParareal

Whereas GParareal trains the GP once per iteration k on $\mathcal{D}_k$, nnGParareal is re-trained every time a prediction $\widehat{f}(\boldsymbol{U}_{i-1}^k)$ is made, on a subset $\mathcal{D}_{i-1,k} \subset \mathcal{D}_k$ with cardinality $|\mathcal{D}_{i-1,k}| = m$

$$\mathcal{D}_{i-1,k} := \{m \text{ nearest neighbors (nn) of } \boldsymbol{U}_{i-1}^k\}.$$

This is known as a nearest neighbor Gaussian process (nnGP).

Using the reduced dataset, nnGParareal models $\widehat{f}$ as

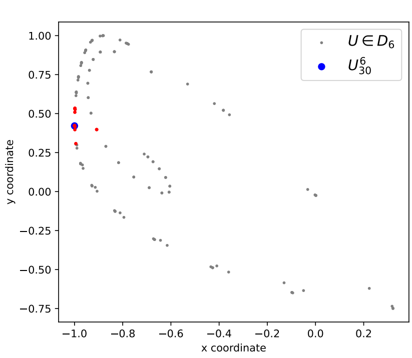
$$\widehat{f}_{\mathrm{GP}}(\boldsymbol{U}_{i-1}^k)_s = \mu_{\mathcal{D}_{i-1,k}}^{(s)}(\boldsymbol{U}_{i-1}^k),$$

at a cost of (assuming K_{nnGPara} iterations to convergence)

$$T_{\mathrm{nnGP}} \in O\left((d/N \vee 1)K_{\mathrm{nnGPara}}N(m^3 + \log(K_{\mathrm{nnGPara}}N)\right),$$

loglinear in N, instead of cubic as GParareal.

nnGParareal - visualizing $\mathcal{D}_{i-1,k}$



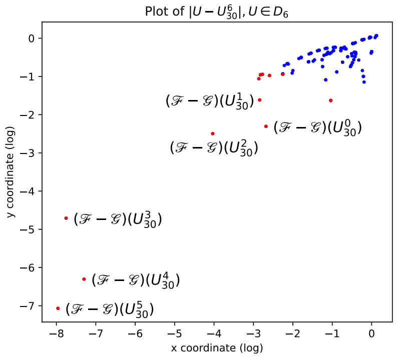


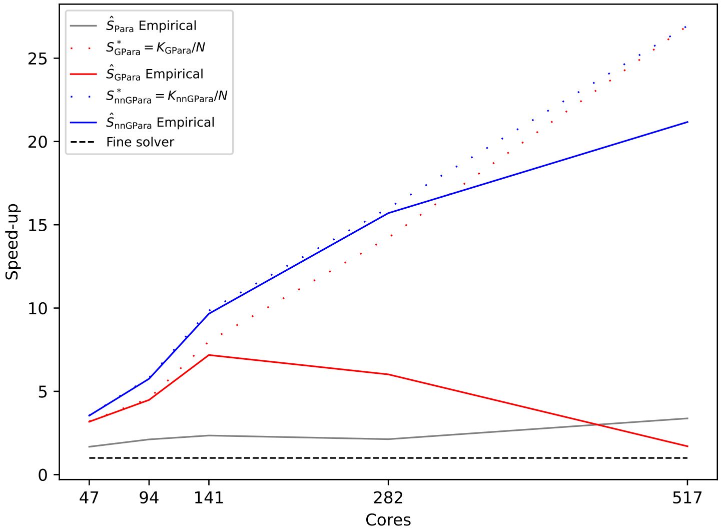
Figure: Visualization of the dataset for the two-dimensional Brusselator system (40) Left, a scatterplot of the observation **U** accumulated by iteration 6 (gray), with the test observation U_{30}^6 (blue). In red are the $m = 15$ nearest neighbors to the test observations. Note how far most points are from U_{30}^6. The right plot makes this easier to see by displaying the absolute (log) distance coordinate-wise between $U \in \mathbf{U}$ and U_{30}^6.

nnGParareal - Performance

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

Comparison of performance for common ODE systems in the literature, described in Slides 37-43.

nnGParareal - Performance - Hopf bifurcations



nnGParareal - Performance - FitzHugh-Nagumo PDE

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 [AF09]. See also Slide 47.

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 $N = 512$.

nnGParareal - Performance - FitzHugh-Nagumo PDE

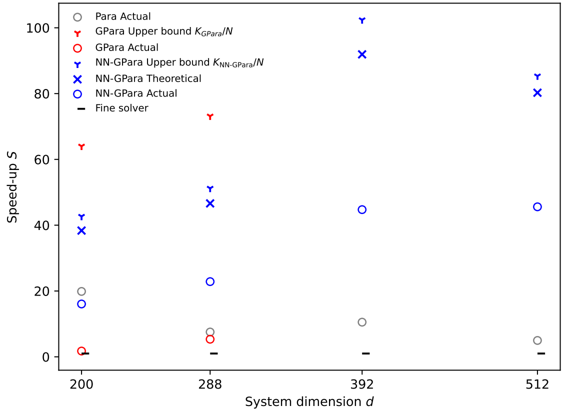


Figure: Plot of speed-ups for Parareal and its variants for the FitzHugh-Nagumo PDE model. The speed-ups are computed according to the formulas above. For $N = 256, 512$, GParareal failed to converge within the computational time budget of 48 hours.

Recap

GParareal:

- Pro: Accelerated convergence compared to Parareal.
- Con: Infeasible for moderate numbers of processors N and ODE dimension d, limiting applicability beyond toy examples.
- Con: It requires one model per ODE dimension.
- Con: Hyperparameter optimization via log-likelihood maximization is very expensive. Usually non-convex.

Recap

Nearest-Neighbors GParareal:

- Pro: Achieves drastic data reduction maintaining or improving performance.
- Pro: The model is re-trained for every prediction, partially relaxing the stationarity assumption.
- Pro: Reduced computational complexity from cubic to loglinear. Verified empirical scalability in N and d (limited).
- Pro: The algorithm runtime can be estimated beforehand.
- Con: It requires one nnGP per ODE dimension.
- Con: Hyperparameter optimization via log-likelihood maximization is still expensive. Usually non-convex.

Wrapping up

Random weights neural networks Parareal:

- All NN-GParareal advantages, plus
- Pro: One model learns all d coordinates
- Pro: No hyperparameter optimization needed, training about x1000 faster than GPs
- Pro: Convex optimization problem with closed-form solution
- Pro: Universal approximation guarantees
- Pro: Drastically improved scalability in d, up to 10^5 spatial discretization points
- Con: Doesn't compute prediction uncertainty.
- Con: Overall performance still affected by $\mathscr{G}$.

Further research question:

- Include uncertainty estimation for the algorithm's solution (probabilistic numerics).

Thank you for listening, any questions?

arxiv.org/abs/2405.12182

Scan the QR code for a link to the paper!

References I

[AF09] Benjamin Ambrosio and Jean-Pierre Françoise. "Propagation of bursting oscillations". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1908 (2009), pp. 4863–4875.

[Dan97] J.M. Anthony Danby. *Computer modeling: from sports to spaceflight... from order to chaos.* 1997.

[DG89] Jean-Louis Deneubourg and Simon Goss. "Collective patterns and decision-making". In: *Ethology Ecology & Evolution* 1.4 (1989), pp. 295–311.

[For88] Bengt Fornberg. "Generation of finite difference formulas on arbitrarily spaced grids". In: *Mathematics of computation* 51.184 (1988), pp. 699–706.

References II

[Gil21] William Gilpin. "Chaos as an interpretable benchmark for forecasting and data-driven modelling". In: *arXiv preprint arXiv:2110.05266* (2021).

[Gor+22] Olga Gorynina et al. "Combining machine-learned and empirical force fields with the parareal algorithm: application to the diffusion of atomistic defects". In: *arXiv preprint arXiv:2212.10508* (2022).

[Kau93] Stuart A. Kauffman. *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA, 1993.

[LN71] René Lefever and Grégoire Nicolis. "Chemical instabilities and sustained oscillations". In: *Journal of theoretical Biology* 30.2 (1971), pp. 267–284.

References III

[LMT01] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. "Résolution d'EDP par un schéma en temps pararéel". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 332.7 (2001), pp. 661–668.

[Lor63] Edward N. Lorenz. "Deterministic nonperiodic flow". In: *Journal of atmospheric sciences* 20.2 (1963), pp. 130–141.

[NAY62] Jinichi Nagumo, Suguru Arimoto, and Shuji Yoshizawa. "An active pulse transmission line simulating nerve axon". In: *Proceedings of the IRE* 50.10 (1962), pp. 2061–2070.

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

References IV

[Ras+90] Steen Rasmussen et al. "The coreworld: Emergence and evolution of cooperative structures in a computational chemistry". In: *Physica D: Nonlinear Phenomena* 42.1-3 (1990), pp. 111–134.

[Rös76] Otto E. Rössler. "An equation for continuous chaos". In: *Physics Letters A* 57.5 (1976), pp. 397–398.

[Sam+19] Debasmita Samaddar et al. "Application of the parareal algorithm to simulations of ELMs in ITER plasma". In: *Computer Physics Communications* 235 (2019), pp. 246–257.

[Sey09] Rüdiger Seydel. *Practical bifurcation and stability analysis*. Vol. 5. Springer Science & Business Media, 2009.

References V

[Tho99] René Thomas. “Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis,” labyrinth chaos””. In: *International Journal of Bifurcation and Chaos* 9.10 (1999), pp. 1889–1905.

ODE/PDE Systems

Systems: FitzHugh–Nagumo

The FitzHugh-Nagumo (FHN) is a model for an animal nerve axon [NAY62]. 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. [Pen+23], 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 [Rös76]

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

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. [Pen+23], 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 [Sey09, pg. 72] for a detailed explanation. The model is defined by the following equations

$$\frac{du_1}{dt} = -u_2 + u_1(\frac{t}{T} - u_1^2 - u_2^2), \quad \frac{du_2}{dt} = u_1 + u_2(\frac{t}{T} - u_1^2 - u_2^2), \quad (3)$$

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. [Pen+23], although, like above, we use the normalized version and set a normalized $\epsilon = 5e^{-7}$.

Systems: Brusselator

The Brusselator models an autocatalytic chemical reaction [LN71]. It is a stiff, non-linear ODE, and the following equations govern it

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

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 [Dan97]. 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

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

where

$$\begin{aligned}
f_1\left(u_1, u_2\right) &= \sin\left(u_1 - u_2\right)\cos\left(u_1 - u_2\right), \\
f_2\left(u_1, u_2\right) &= 2 - \cos^2\left(u_1 - u_2\right).
\end{aligned}$$

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 $\mathscr{G}$, and Ruge-Kutta 8 with $2.17e^5$ steps for the fine solver $\mathscr{F}$. This is a similar setting as Pentland et al. [Pen+23, 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 [Lor63]. With the following parameters, it is a chaotic system governed by the equations

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

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 [Tho99] has proposed a particularly simple three-dimensional system representative of a large class of auto-catalytic models that occur frequently in chemical reactions [Ras+90], ecology [DG89], and evolution [Kau93]. It is described by the following equations

$$\begin{cases} \frac{dx}{dt} = b \sin y - ax, \\ \frac{dy}{dt} = b \sin z - ay, \\ \frac{dz}{dt} = b \sin x - az, \end{cases} \tag{4}$$

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 [Gil21], 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 $\mathscr{G}$ and Ruge-Kutta 4 with $1e^9$ steps for the fine solver $\mathscr{F}$.

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

$$v_t = \nu v_{xx} - v v_x \quad (x,t) \in (-L, L) \times (t_0, t_N], \tag{5}$$

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 [For88] 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 [AF09] is an extension of the ODE system in Slide 37. 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

$$\begin{aligned} v_t &= a\nabla^2 v + v - v^3 - w - c, \quad (x,t) \in (-L,L)^2 \times (t_0, t_N] \\ w_t &= \tau\left(b\nabla^2 w + v - w\right), \end{aligned} \tag{6}$$

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

$$\begin{aligned}
v((x,-L),t) &= v((x,L),t) \\
v((-L,y),t) &= v((L,y),t) \\
v_y((x,-L),t) &= v_y((x,L),t) \\
v_x((-L,y),t) &= v_x((L,y),t), \quad t \in [t_0, t_N].
\end{aligned}$$

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 1 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.