

NN-GParareal: Improving Scalability of GParareal Using Nearest Neighbors

Guglielmo Gattiglio
University of Warwick

January 16, 2024

Under the supervision of:
Lyudmila Grigoryeva, University of St. Gallen
Massimiliano Tamborrino, University of Warwick

- ▶ Parareal
- ▶ Sketch of the Parareal algorithm
- ▶ Intermezzo: Gaussian process refresher
- ▶ GParareal
- ▶ NN-GParareal **New!**
- ▶ Empirical results

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 used 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)

Introduction

An example of the systems we wish to solve: Thomas Labyrinth, a chaotic ODE that will be considered later in the presentation.



ThomasLabyrinth

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

- ▶ $h : \mathbb{R}^d \times [t_0, t_N] \rightarrow \mathbb{R}^d$ is a smooth multivariate function,
- ▶ $u : [t_0, t_N] \rightarrow \mathbb{R}^d$ 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

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

and we enforce the continuity conditions at each t_i , namely

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

Parareal (Lions et al., 2001)

We solve this system of $N + 1$ equations

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

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

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

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

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

- ▶ \mathcal{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.
- ▶ \mathcal{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, $\mathcal{F} \left(U_{i-1}^k \right)$, and the derivative in the second term by finite differences using

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

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

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

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

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 \mathcal{G} .
- ▶ The stopping criterion of this algorithm is chosen as

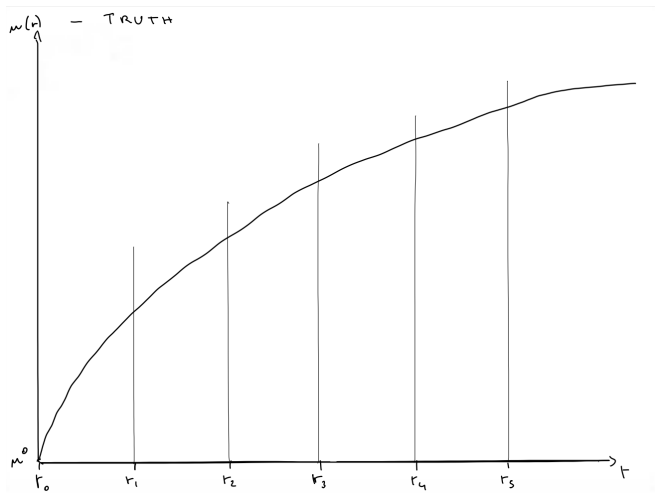
$$\|U_i^k - U_i^{k-1}\|_\infty < \epsilon, \quad \forall i \leq N, \quad (4)$$

where $\|\cdot\|_\infty$ is the infinity norm, which guarantees that the initial conditions have stabilized.

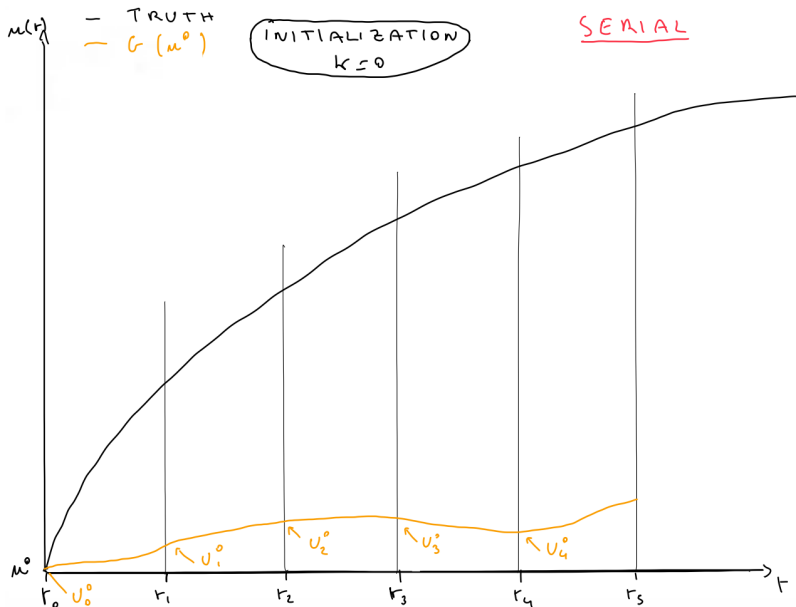
- ▶ Let K be the total number of Parareal iterations to convergence. In the worst-case scenario, $K = N$ and the solution trivially converges to that of the fine solver.

Parareal - Sketch of behavior 1/13

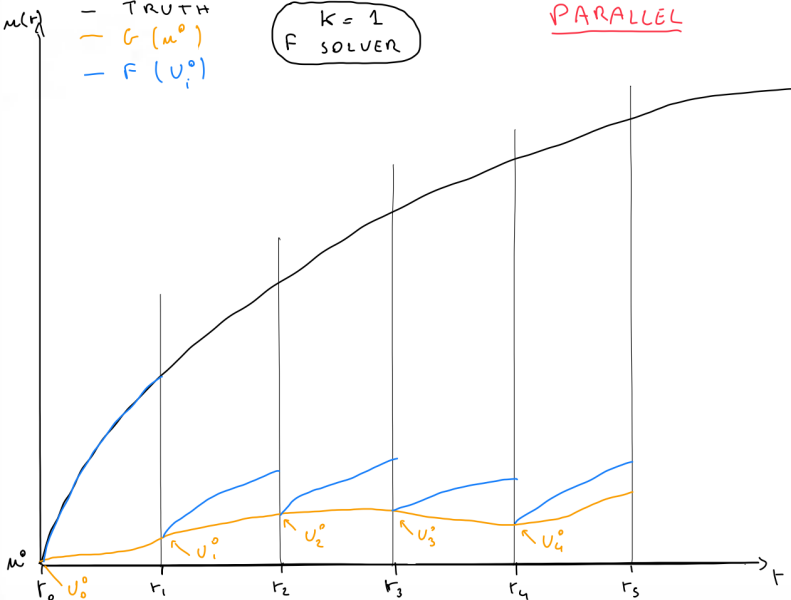
Consider solving the following ODE with initial condition $u^0 = 0$, and $N = 5$



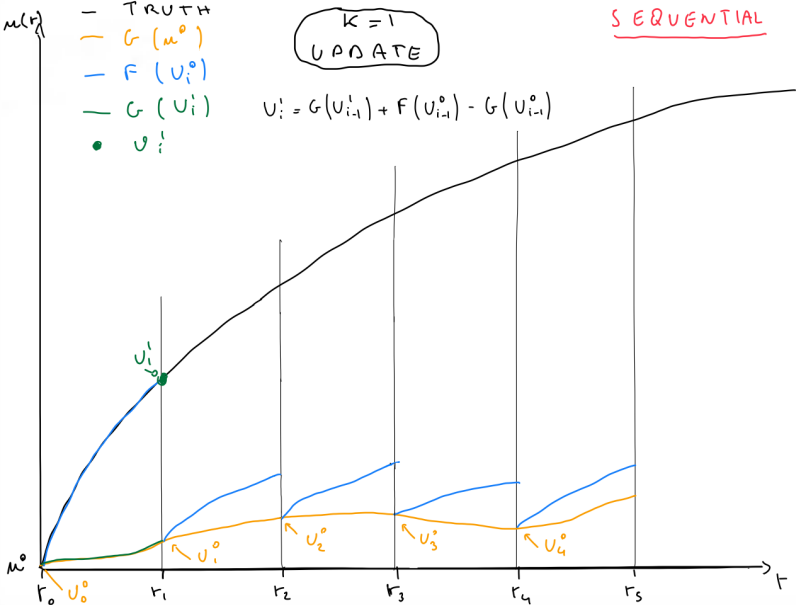
Parareal - Sketch of behavior 2/13



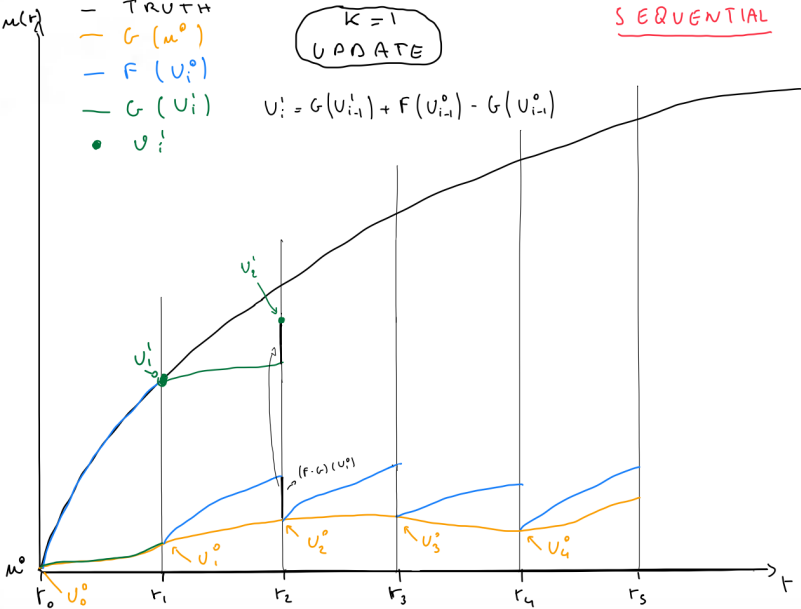
Parareal - Sketch of behavior 3/13



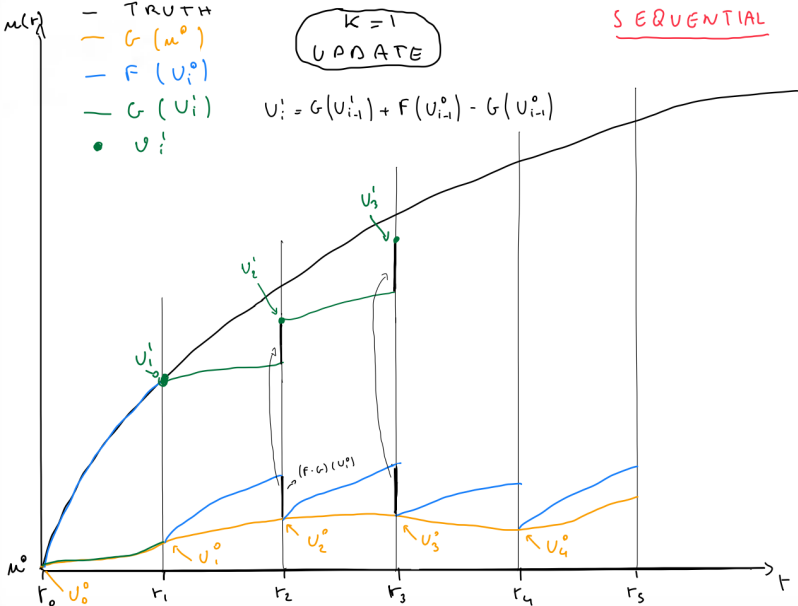
Parareal - Sketch of behavior 4/13



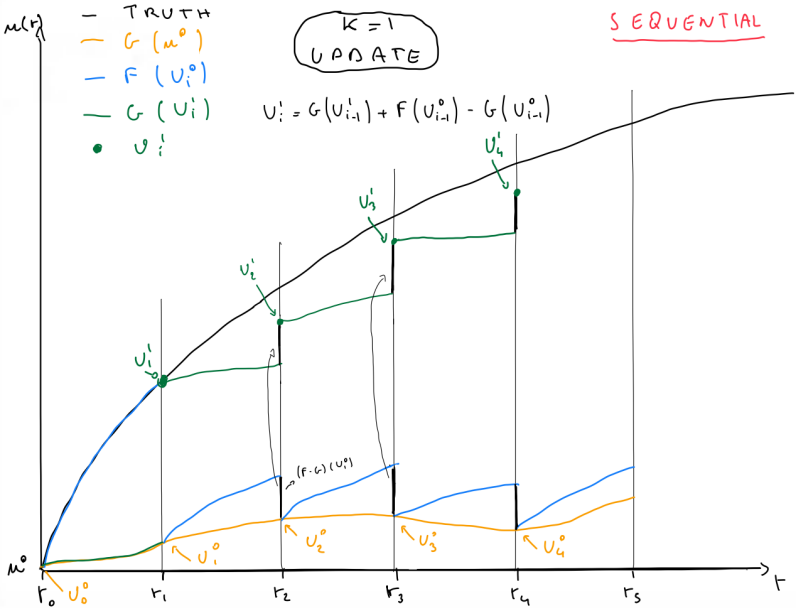
Parareal - Sketch of behavior 5/13



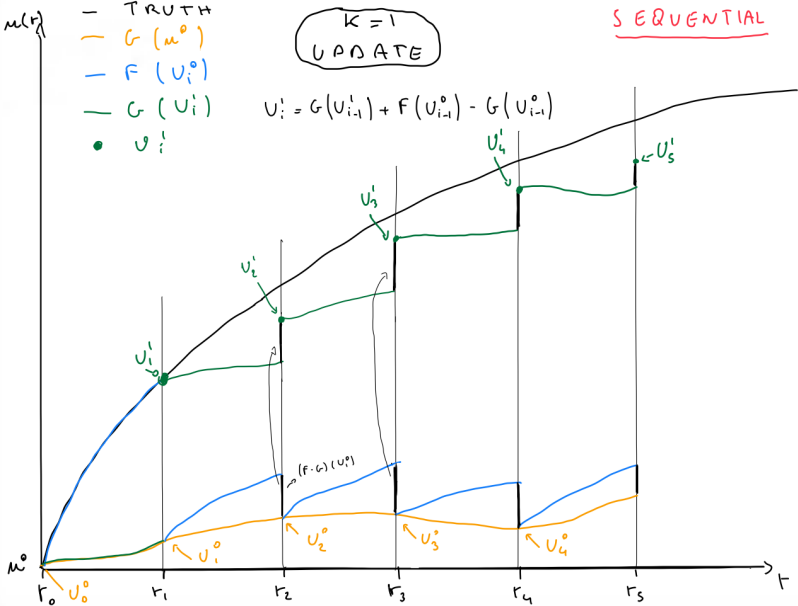
Parareal - Sketch of behavior 6/13



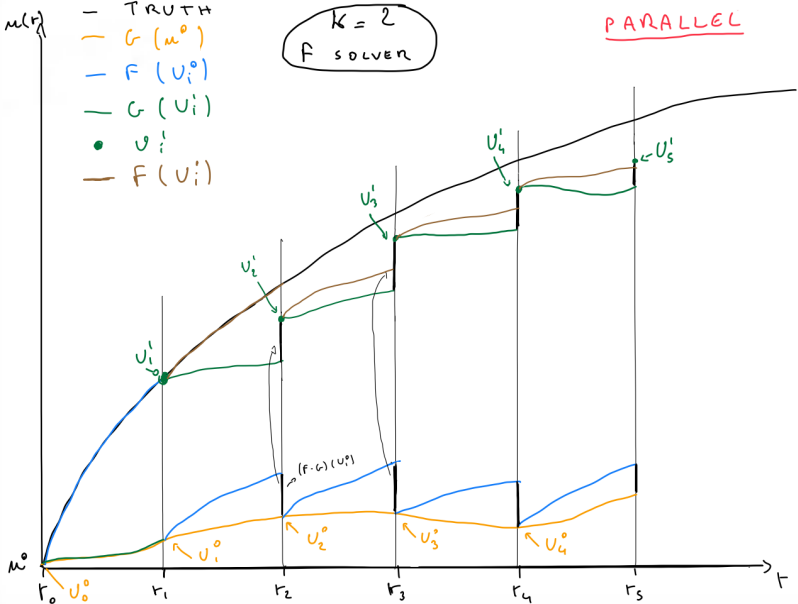
Parareal - Sketch of behavior 7/13



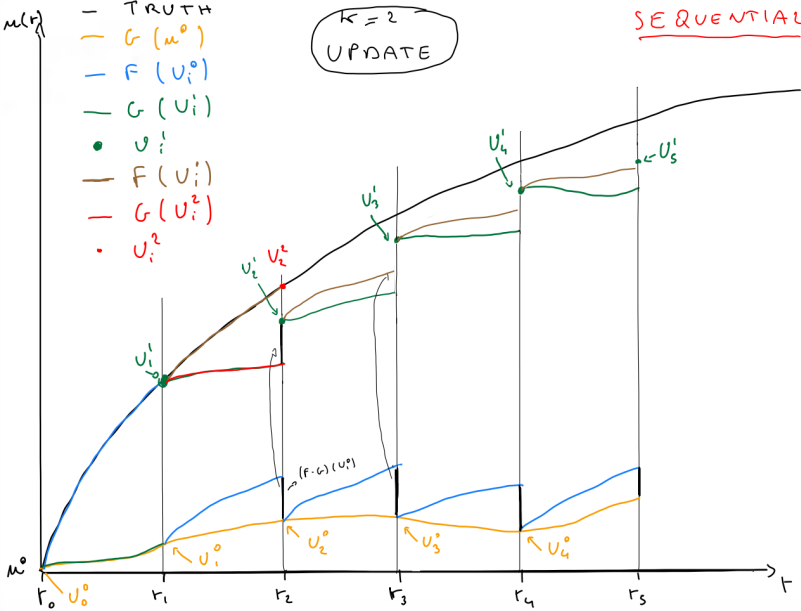
Parareal - Sketch of behavior 8/13



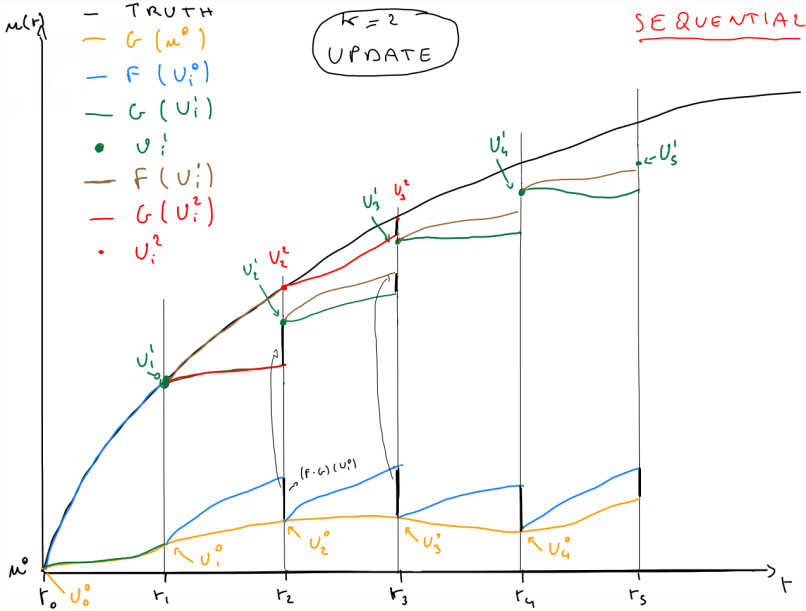
Parareal - Sketch of behavior 9/13



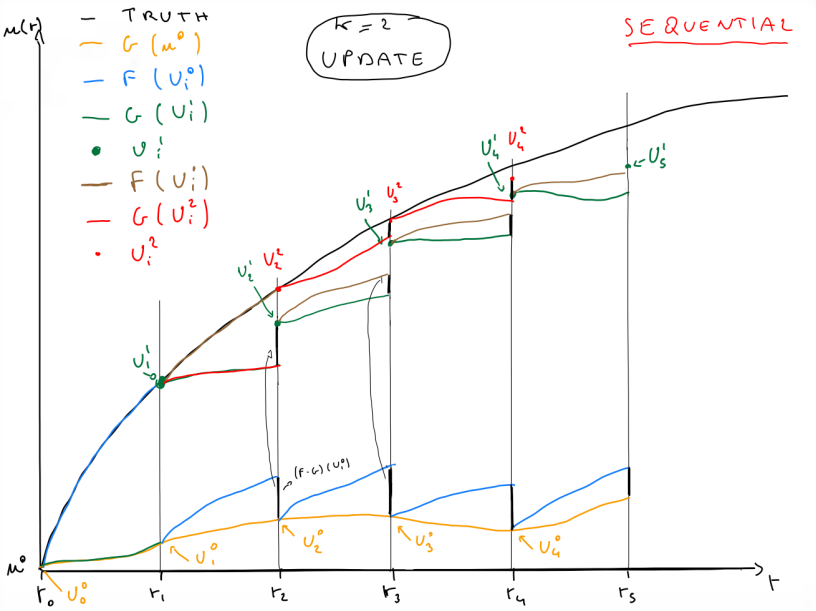
Parareal - Sketch of behavior 10/13



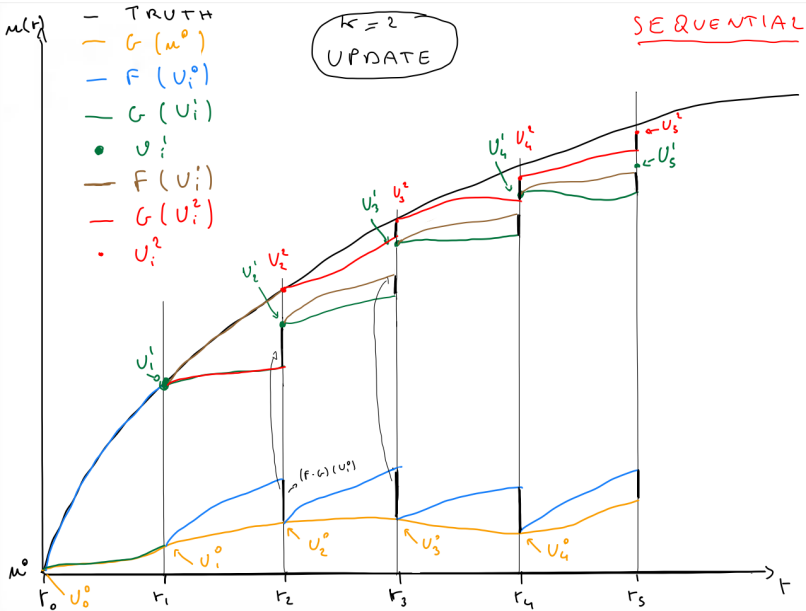
Parareal - Sketch of behavior 11/13



Parareal - Sketch of behavior 12/13



Parareal - Sketch of behavior 13/13



GParareal (Pentland et al. (2023))

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 (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 $\text{Cov}(f(x), f(x')) = k(x, x')$, where the kernel $k(\cdot, \cdot)$ is the squared exponential

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

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 \mathbf{x}^* , conditional on the observed data D . This can be obtained through the posterior distribution $y|\mathbf{x}^*$, which is normal with mean

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

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 $O(n^3)$.

We can use GPs to improve the Parareal update (3).

GParareal (Pentland et al. (2023))

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

The original algorithm computes the correction based on information calculated during the previous iteration k ,

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

Conversely, Gparareal uses information from the current iteration $k + 1$,

$$\begin{aligned} U_i^{k+1} &= \mathcal{F} \left(U_{i-1}^{k+1} \right) = (\mathcal{F} - \mathcal{G} + \mathcal{G}) \left(U_{i-1}^{k+1} \right) \\ &= (\mathcal{F} - \mathcal{G}) \left(U_{i-1}^{k+1} \right) + \mathcal{G} \left(U_{i-1}^{k+1} \right). \end{aligned}$$

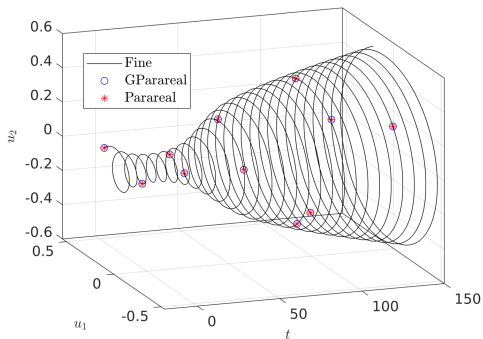
However, this would require a serial computation of $\mathcal{F} \left(U_{i-1}^{k+1} \right)$. Instead, a Gaussian process is used to infer the first term from data.

GParareal - Advantages and Disadvantages

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

$$\frac{du_1}{dt} = -u_2 + u_1\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad \frac{du_2}{dt} = u_1 + u_2\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad (5)$$

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



GParareal - Advantages and Disadvantages

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.
- ▶ \mathcal{F} and \mathcal{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 (\mathcal{F}) to the parallel algorithm.

All entries are in seconds.

GParareal - Advantages and Disadvantages

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	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	19	2.78e-02	1.09e+03	1.85e-03	2.07e+04	1.61
GParareal	10	1.52e-02	1.09e+03	6.64e+00	1.09e+04	3.06
NN-GParareal	9	1.53e-02	1.09e+03	4.71e+00	9.80e+03	3.40

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

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	30	3.40e-02	5.49e+02	6.12e-03	1.65e+04	2.02
GParareal	14	5.25e-02	5.50e+02	5.64e+01	7.75e+03	4.30
NN-GParareal	11	3.38e-02	5.50e+02	1.05e+01	6.06e+03	5.50

GParareal - Advantages and Disadvantages

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

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	54	5.36e-02	2.71e+02	2.23e-02	1.47e+04	2.28
GParareal	16	8.39e-02	2.72e+02	5.48e+02	4.90e+03	6.81
NN-GParareal	13	6.68e-02	2.72e+02	3.28e+01	3.56e+03	9.36

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

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	97	8.93e-02	1.36e+02	7.05e-02	1.32e+04	2.53
GParareal	18	1.47e-01	1.36e+02	4.83e+03	7.27e+03	4.59
NN-GParareal	16	1.19e-01	1.36e+02	1.08e+02	2.28e+03	14.64

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

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	149	1.59e-01	6.81e+01	1.99e-01	1.02e+04	3.28
GParareal	19	2.73e-01	6.91e+01	3.99e+04	4.13e+04	0.81
NN-GParareal	19	2.50e-01	6.93e+01	2.91e+02	1.61e+03	20.69

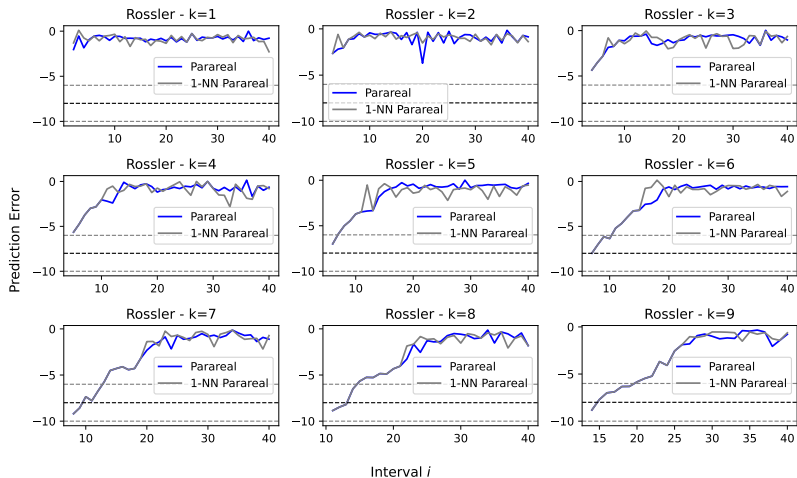
NN-GParareal

NN-GParareal - Key Points

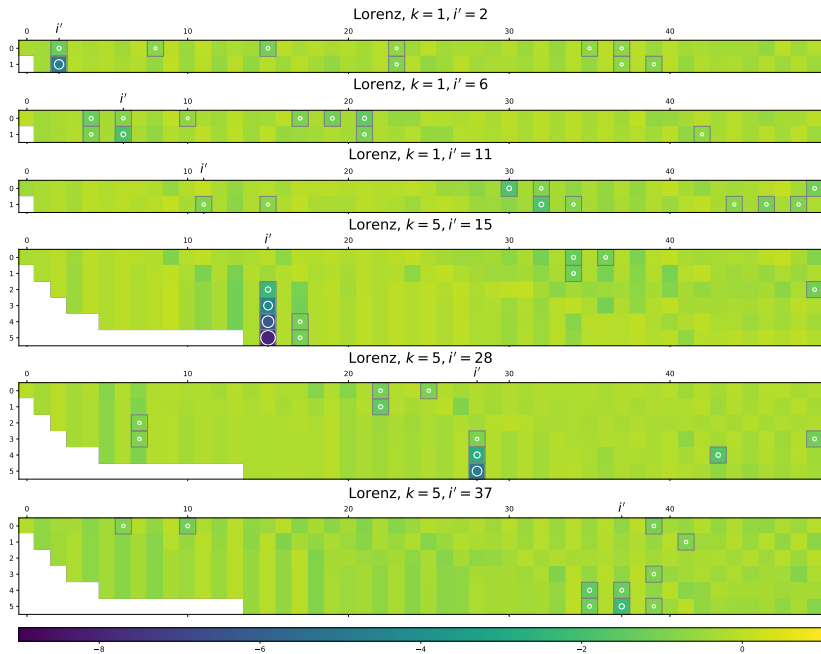
- ▶ Whereas GParareal trains the GP once per iteration k using the full dataset D , NN-GParareal is re-trained every time a prediction is made and it uses a subset $D' \subset D$ of the dataset D , with cardinality $|D'| = m$.
- ▶ Empirically, a fixed small value of $m \in \{15, \dots, 20\}$ is sufficient 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. We chose this among the others for its simplicity and intuition (next slide).
- ▶ This model is known as nearest neighbor Gaussian process (NNGP) in the literature.
- ▶ The computational cost incurred by the GP at iteration k is $O(k^3 N^3)$, while that of NNGP is $O(Nm^3 + N \log(kN))$.
- ▶ Re-training at every prediction makes the GP globally non-stationary, without the need to change the kernel.

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-GParareal - Intuition



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.

Thomas Labyrinth, $N = 32$

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	30	1.82e-02	1.04e+03	2.52e-03	3.13e+04	1.07
GParareal	26	2.15e-02	1.04e+03	4.09e+01	2.71e+04	1.23
NN-GParareal	24	2.19e-02	1.04e+03	7.49e+00	2.50e+04	1.33

Thomas Labyrinth, $N = 64$

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	20	3.59e-02	5.42e+02	3.85e-03	1.08e+04	3.08
GParareal	14	4.84e-02	5.42e+02	2.55e+01	7.62e+03	4.38
NN-GParareal	16	4.27e-02	5.59e+02	1.24e+01	8.96e+03	3.72

NN-GParareal - More results: Thomas Labyrinth

Thomas Labyrinth, $N = 128$

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	100	3.87e-02	2.74e+02	3.15e-02	2.74e+04	1.22
GParareal	71	4.83e-02	2.70e+02	3.06e+04	4.99e+04	0.67
NN-GParareal	63	4.55e-02	2.72e+02	8.83e+01	1.72e+04	1.94





Thomas Labyrinth, $N = 256$

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	256	6.92e-02	1.39e+02	1.52e-01	3.56e+04	0.94
GParareal	> 46	1.64e-01	1.41e+02	> 1.66e+05	> 1.73e+05	< 0.19
NN-GParareal	159	8.24e-02	1.37e+02	4.05e+02	2.21e+04	1.51




Thomas Labyrinth, $N = 512$

Model	K	\mathcal{G}	\mathcal{F}	Model	Total	Speed-up
Fine	-	-	-	-	3.34e+04	1
Parareal	180	1.34e-01	7.00e+01	2.10e-01	1.27e+04	2.64
GParareal	> 25	3.17e-01	8.86e+01	> 1.71e+05	> 1.73e+05	< 0.19
NN-GParareal	69	1.69e-01	7.84e+01	4.67e+02	5.89e+03	5.66

References I

-  Gilpin, William (2021). “Chaos as an interpretable benchmark for forecasting and data-driven modelling”. In: *arXiv preprint arXiv:2110.05266*.
-  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*.
-  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.
-  Pentland, Kamran et al. (2023). “GParareal: a time-parallel ODE solver using Gaussian process emulation”. In: *Statistics and Computing* 33.1, p. 23.

References II

-  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.
-  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*.

NN-GParareal - More results: Thomas Labyrinth

GParareal

