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

Guglielmo Gattiglio
University of Warwick

January 31, 2024

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

Parareal (Lions et al., 2001).

- Theory Done
- Sketch of the procedure Done
- Computational cost

GParareal (Pentland et al., 2023).

- Intuition Done
- Empirical results
- Computational cost

Nearest Neighbor GParareal New!

- Intuition
- Empirical results
- Computational cost

Parareal

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

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

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

$$\begin{aligned} T_{Para} &\approx NT_{\mathscr{G}} + \sum_{k=1}^{K_{Para}} (T_{\mathscr{F}} + (N-k)T_{\mathscr{G}}) \\ &= K_{Para}T_{\mathscr{F}} + (K_{Para}+1)(N - K_{Para}/2)T_{\mathscr{G}} \end{aligned}$$

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}.

GParareal

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 24-30.

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 $\epsilon = 5e^{-7}$.

GPararealreal - Computational cost

The runtime cost of GPararealreal is

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

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

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

GPararealreal - Computational cost

The speed-up is

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

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

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

GPararealal - Performance - Hopf bifurcations

To showcase the empirical performance of GPararealal, consider a non-linear model for the study of Hopf bifurcations (Seydel, 2009, pg. 72; also Slide 26), defined by the following equations

$$\frac{du_1}{dt} = -u_2 + u_1\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad \frac{du_2}{dt} = u_1 + u_2\left(\frac{t}{T} - u_1^2 - u_2^2\right), \quad (1)$$

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

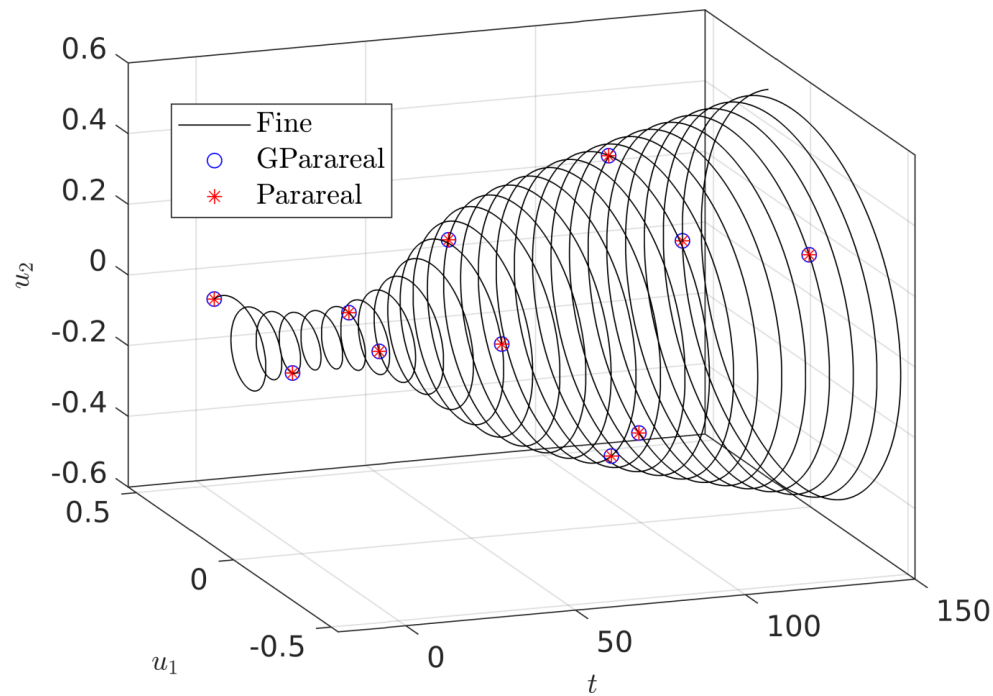
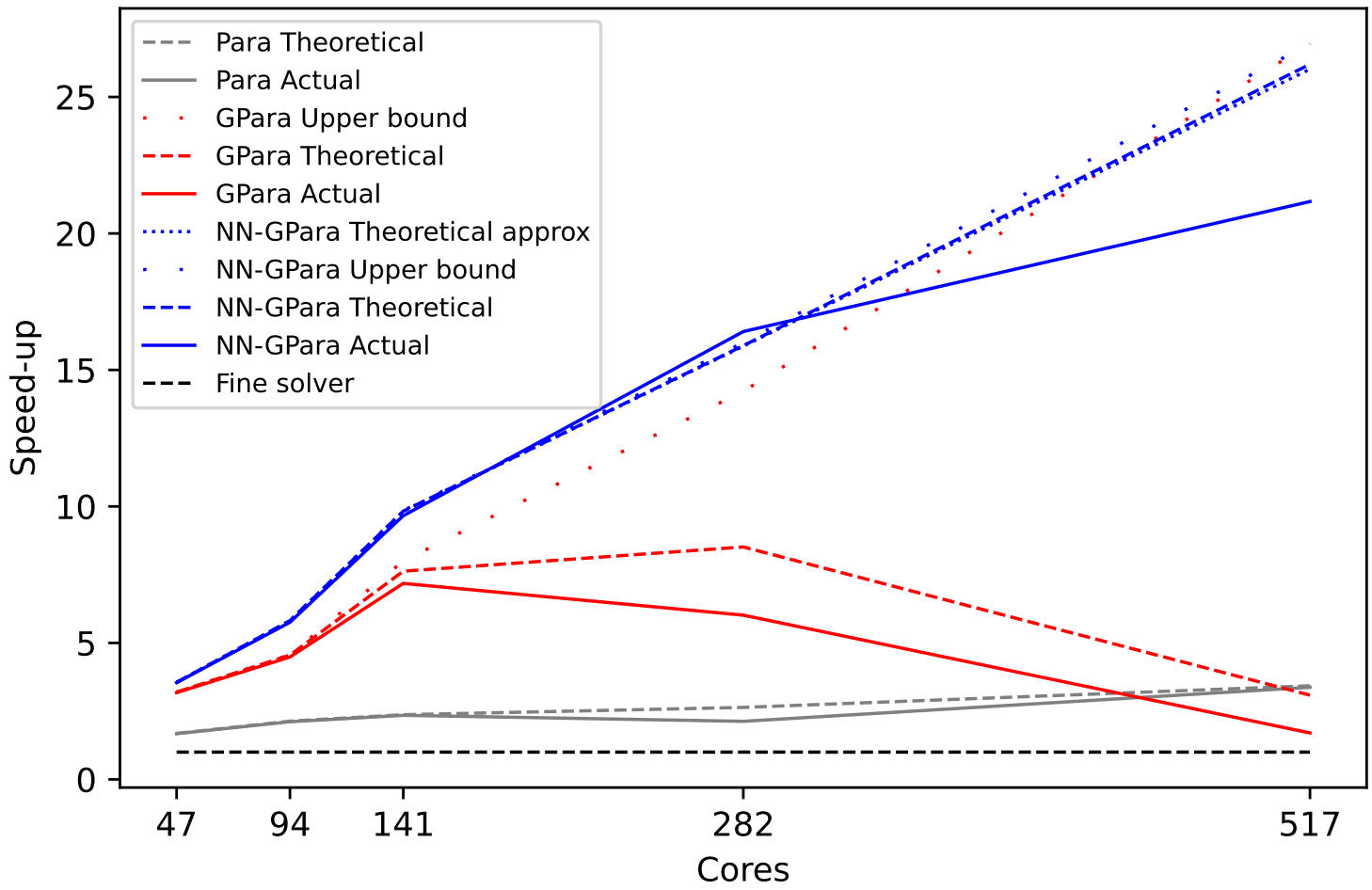


Image taken from Pentland et al. (2023)

GPararealal - Performance - Hopf bifurcations



GPararealreal - 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 GPararealal (NN-GPararealal)

NN-GPareareal - Key Points

- Whereas GPareareal trains the GP once per iteration k using the full dataset D, NN-GPareareal 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 24-30.

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-GPararealreal - Computational Cost

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

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

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,

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

T^{m}_{NNGP} 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-GPararealal - Computational Cost

The speed-up for NN-GPararealal is

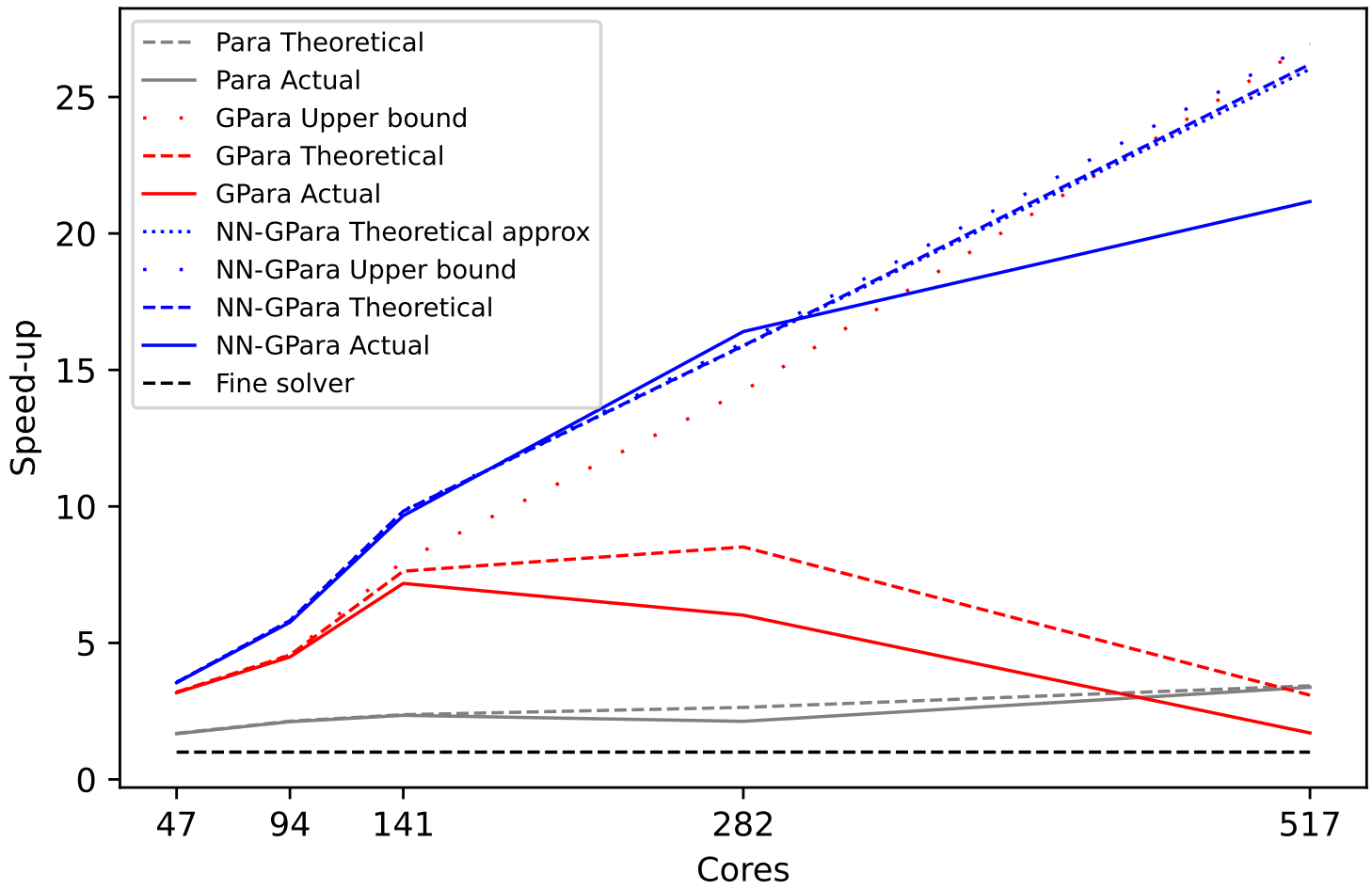
$$S_{NN\text{-}GPara} \approx \left[\frac{K_{NN}}{N} + (K_{NN}+1)\left(1 - \frac{K_{NN}}{2N}\right)\frac{T_{\mathscr{G}}}{T_{\mathscr{F}}} + \frac{K_{NN}\, T^m_{NNGP}}{N T_{\mathscr{F}}}(N - (K_{NN}+1)/2) \right]^{-1}.$$

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

$$\begin{aligned} T_{NNGP} = \sum_{k=1}^{K_{NN}} (N-k)\, T^m_{NNGP} &= \sum_{k=1}^{K_{NN}} (N-k)[O(m^3) + O(\log(kn))] \\ &= O(K_{NN} N m^3) + O(K_{NN} N \log(K_{NN} N)). \end{aligned}$$

The log term comes from the nearest neighbor computation.

NN-GPararealal - Performance - Hopf bifurcations



NN-GParareal - 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 (Ambrosio and Françoise, 2009). See also Slide 34.

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

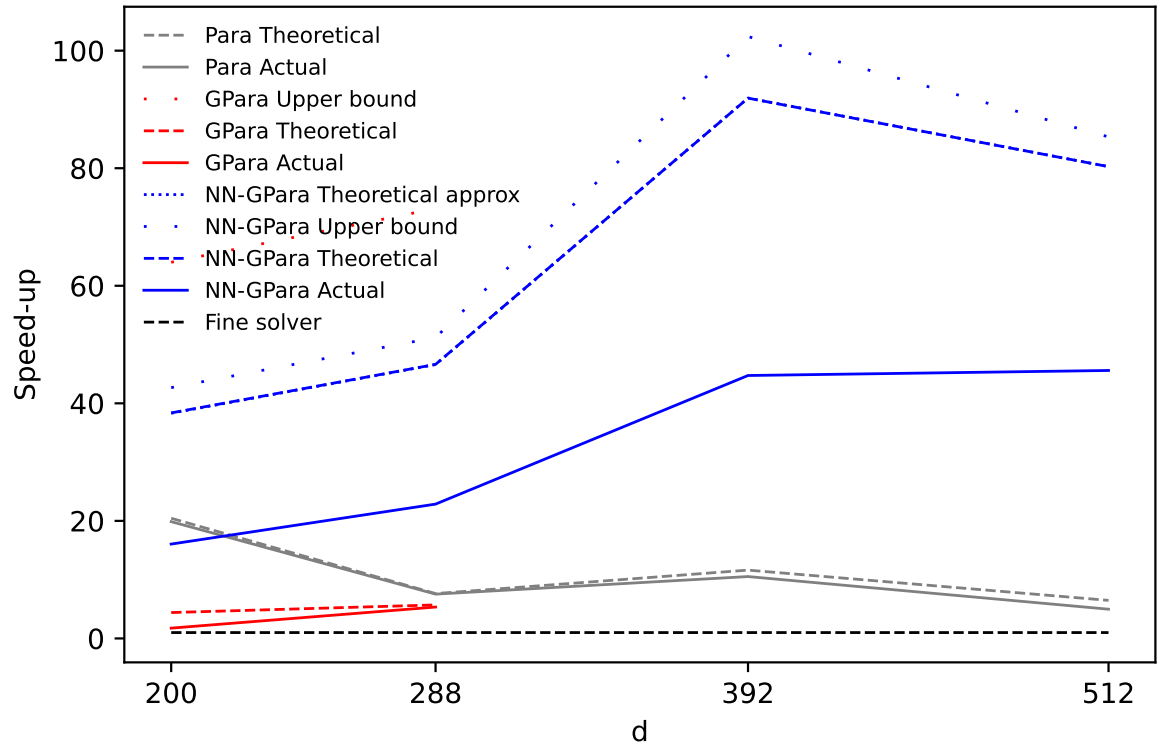


Figure: Plot of speed-ups for Parareal and its variants for the FitzHugh-Nagumo PDE model. The speed-ups are computed according to the formulas above. For $N = 256, 512$, GParareal failed to converge within the computational time budget.

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

- 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.
- 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.

References III

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

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)

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

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

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

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 $\mathscr{G}$, and Ruge-Kutta 8 with $2.17e^5$ steps for the fine solver $\mathscr{F}$. This is a similar setting as Pentland et al. (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

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

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 (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

$$\begin{cases} \frac{dx}{dt} = b \sin y - ax, \\ \frac{dy}{dt} = b \sin z - ay, \\ \frac{dz}{dt} = b \sin x - az, \end{cases} \tag{3}$$

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 $\mathscr{G}$ 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

$$v_t = \nu v_{xx} - v v_x \quad (x,t) \in (-L, L) \times (t_0, t_N], \tag{4}$$

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 24. 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

$$\begin{aligned} v_t &= a\nabla^2 v + v - v^3 - w - c, \quad (x,t) \in (-L,L)^2 \times (t_0, t_N] \\ w_t &= \tau\left(b\nabla^2 w + v - w\right), \end{aligned} \tag{5}$$

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.