GParareal: A time-parallel ODE solver using Gaussian process emulation

K. Pentland[1], M. Tamborrino[2], T. J. Sullivan[1,3], J. Buchanan[4], and L. C. Appel[4]

[1]Mathematics Institute, University of Warwick, UK
[2]Department of Statistics, University of Warwick, UK
[3]Alan Turing Institute, British Library, London, UK
[4]Culham Centre for Fusion Energy, UKAEA, Oxfordshire, UK

Motivation

In this work, we seek numerical solutions $\boldsymbol{U}_j \approx \boldsymbol{u}(t_j)$ to a system of $d \in \mathbb{N}$ (nonlinear) ordinary differential equations (ODEs)

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(\boldsymbol{u}(t), t) \quad \text{over} \quad t \in [t_0, t_J], \quad \text{with} \quad \boldsymbol{u}(t_0) = \boldsymbol{u}^0, \tag{1}$$

on the mesh $\boldsymbol{t} = (t_0, \ldots, t_J)$, where $t_{j+1} = t_j + \Delta T$ for fixed $\Delta T = (t_J - t_0)/J$. In particular, we are interested in initial value problems (IVPs) (1) in which any combination of:

(i) the interval of integration, $[t_0, t_J]$

(ii) the number of mesh points, $J+1$

(iii) the wallclock time to evaluate the vector field, $\boldsymbol{f}$

is so **large** that simulating $\boldsymbol{U}_j$ **takes hours, days, or even weeks** using **sequential** numerical methods (e.g. Runge-Kutta).

The **aims of this work** are to:

- develop a time-parallel algorithm (**GParareal**) that iteratively locates a numerical solution to (1) using a **Gaussian process (GP) emulator**.
- **train** the emulator using **fine** and **coarse** resolution **acquisition** and **legacy** solution data.
- illustrate that GParareal can converge in **fewer iterations** than the classic **parareal algorithm**, thus achieving **additional parareal speed-up**.

Parareal

Parareal[1] is a well established time-parallel numerical method for solving a variety of IVPs - including **fusion plasma dynamics**[2]. It locates a solution $\boldsymbol{U}_j^k$ to (1) (on $\boldsymbol{t}$) in $k \in \{1, \ldots, J\}$ iterations, yielding a **fixed parallel speed up** (roughly J/k) compared to a serial numerical integrator.

Setup

- **Discretise** problem (1) into J sub-problems on J sub-intervals - **assiging one processor to each** ($J = 5$ in Fig. 1).
- Choose **two sequential numerical integrators** to carry out integration from t_j to t_{j+1} :
  - ➔ $\mathcal{F}$ – **fine integrator** with slow execution but high accuracy.
  - ➔ $\mathcal{G}$ – **coarse integrator** with fast execution but low accuracy.

Algorithm

```
Goal:    Locate $\mathbf{U}_j = \mathcal{F}(\mathbf{U}_{j-1})$ without running $\mathcal{F}$ $J$ times sequentially.
Step 1:  Calculate initial guesses using $\mathcal{G}$ serially: $\mathbf{U}_j^0 = \mathcal{G}(\mathbf{U}_{j-1}^0)$.
Step 2:  For $k = 1$ to $J$
    (i)    Propagate solutions on each sub-interval using $\mathcal{F}$ in parallel, calculating $\mathcal{F}(\mathbf{U}_{j-1}^{k-1})$.
    (ii)   Sequentially calculate $\mathcal{G}(\mathbf{U}_{j-1}^k)$, then use the predictor-corrector (PC):
               $\mathbf{U}_j^k = \underbrace{\mathcal{G}(\mathbf{U}_{j-1}^k)}_{\text{Prediction}} + \underbrace{\mathcal{F}(\mathbf{U}_{j-1}^{k-1}) - \mathcal{G}(\mathbf{U}_{j-1}^{k-1})}_{\text{Correction}}$.    (2)
    (iii)  If the tolerance $\|\mathbf{U}_j^k - \mathbf{U}_j^{k-1}\|_\infty < \varepsilon$ is met for all $j$, break the loop and return $\mathbf{U}_j^k$.
           Else continue iterations for the unconverged $t_j$.
```

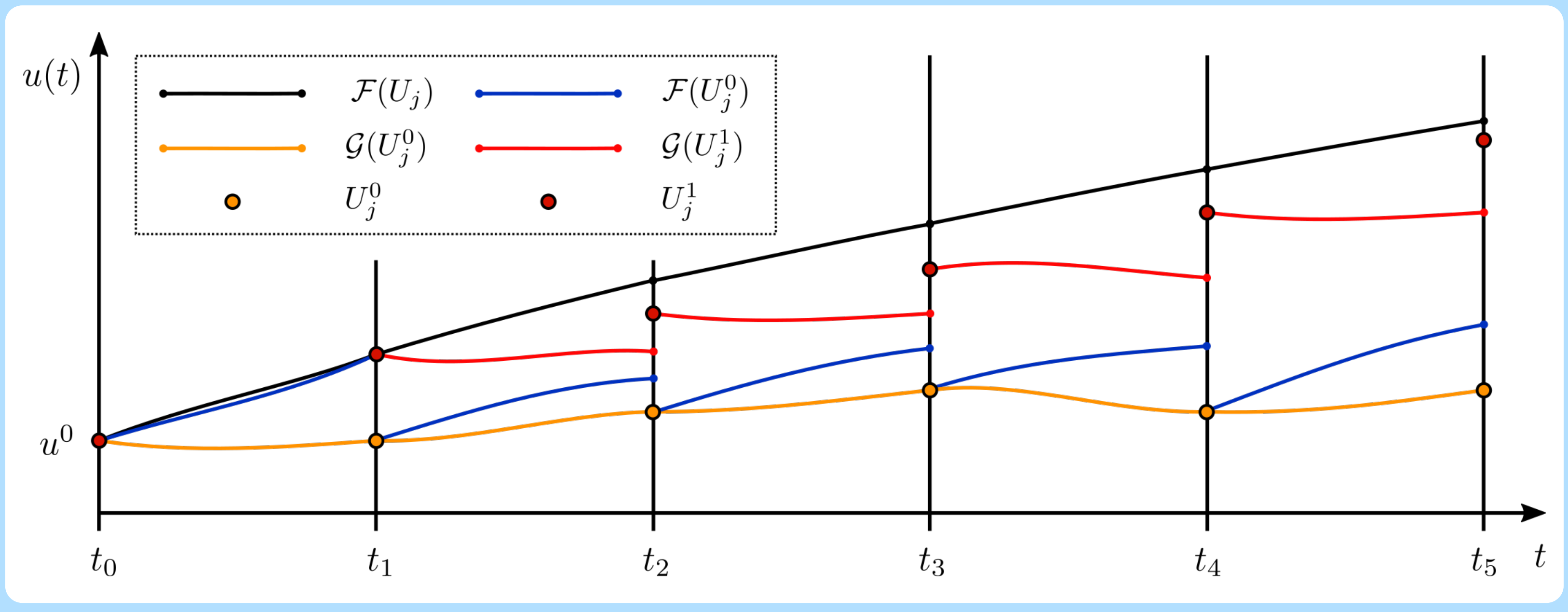

Fig. 1: First iteration of parareal. Exact "fine" solution (black), first coarse solves (yellow), first parallel fine solves (blue), second coarse solves (red), and the PC (2) solutions (red dots).

Gaussian process emulation

What is a GP emulator?

- A **statistical model** built from **multivariate Gaussian distributions** that emulates (predicts) an **unknown function** using a finite number of **known evaluations** of the function.

How does it work?

- **Goal**: model an unknown, expensive to evaluate, function $H(u)$ (solid black line, Fig. 2).
- We only know H at a few evaluation points (red dots).
- A GP emulator conditions a **Gaussian prior** over H on these known points to obtain a **Gaussian prediction** over H at any input location $u \in \mathbb{R}$:

$$H(u) \sim \mathcal{N}(\mu(u), K(u,u)),$$

where μ and K are known **mean** (solid blue) and **covariance** (shaded light blue) functions.

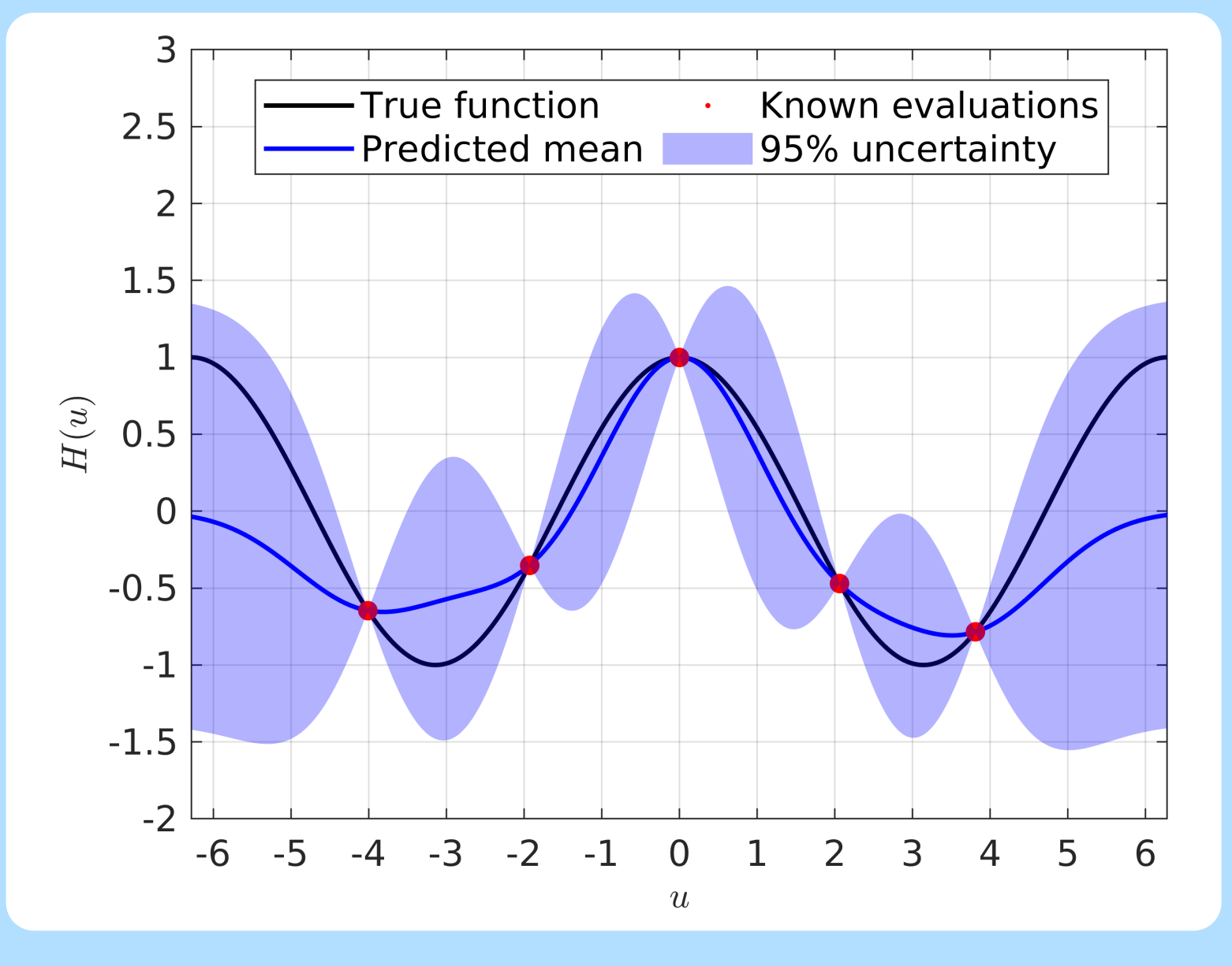

Fig. 2: A GP emulator in action.

GParareal

Idea: Avoid wasting valuable data by modelling the **correction term** in parareal (2) using a GP emulator **trained on all evaluations** of $\mathcal{F}$ and $\mathcal{G}$ (plus **legacy solution data** if available)[3].

We start by re-writing (2) as

$$\boldsymbol{U}_j^k = \overbrace{\mathcal{F}(\underbrace{\boldsymbol{U}_{j-1}^k}_{\text{known}})}^{\text{unknown}} = (\mathcal{F} - \mathcal{G} + \mathcal{G})(\boldsymbol{U}_{j-1}^k) = \underbrace{\mathcal{G}(\boldsymbol{U}_{j-1}^k)}_{\text{Prediction}} + \underbrace{(\mathcal{F} - \mathcal{G})(\boldsymbol{U}_{j-1}^k)}_{\text{Correction}}. \tag{3}$$

We can query the (trained) emulator to obtain

$$(\mathcal{F} - \mathcal{G})(\boldsymbol{U}_{j-1}^k) \sim \mathcal{N}\big(\mu(\boldsymbol{U}_{j-1}^k), K(\boldsymbol{U}_{j-1}^k, \boldsymbol{U}_{j-1}^k)\big), \tag{4}$$

allowing us to approximate (3) by

$$\boldsymbol{U}_j^k \approx \mathcal{G}(\boldsymbol{U}_{j-1}^k) + \underbrace{\mathbb{E}[(\mathcal{F} - \mathcal{G})(\boldsymbol{U}_{j-1}^k)]}_{\text{Expected value of (4)}}. \tag{5}$$

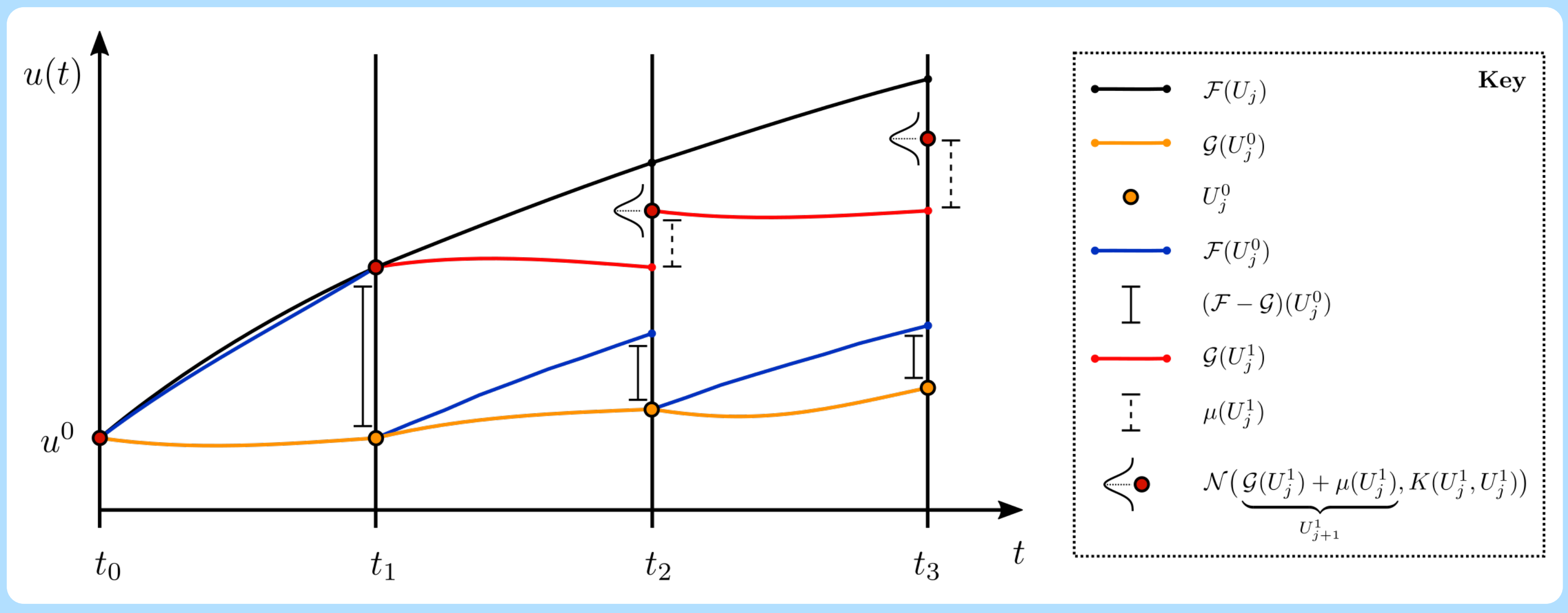

Fig. 3: First iteration of GParareal. Exact solution (black), first coarse solves (yellow), first parallel fine solves (blue), residual between these solves (solid bars), second coarse runs (red), correction from the GP (dashed bars), and the refined solutions from (5) (red dots).

Test problem: FitzHugh-Nagumo model

We test GParareal on the FitzHugh-Nagumo model

$$\frac{du_1}{dt} = 3\left(u_1 - \frac{u_1^3}{3} + u_2\right), \quad \frac{du_2}{dt} = -\frac{1}{3}\left(u_1 - \frac{1}{5} + \frac{u_2}{5}\right), \quad \text{over} \quad t \in [0, 40], \tag{6}$$

using 2nd ($\mathcal{G}$) and 4th ($\mathcal{F}$) order Runge-Kutta solvers on $J = 40$ processors.

In Fig. 4, we solve (6) over a **range of initial values** to comparare the convergence rate k of both parareal and GParareal. We find that GParareal:

- converges in **significantly fewer iterations** k **than parareal** for all tested initial values.
- locates a solution in **faster wallclock time than parareal** (results shown in paper[3]).
- **converges for initial values where parareal fails** (i.e. when coarse solutions blow up).

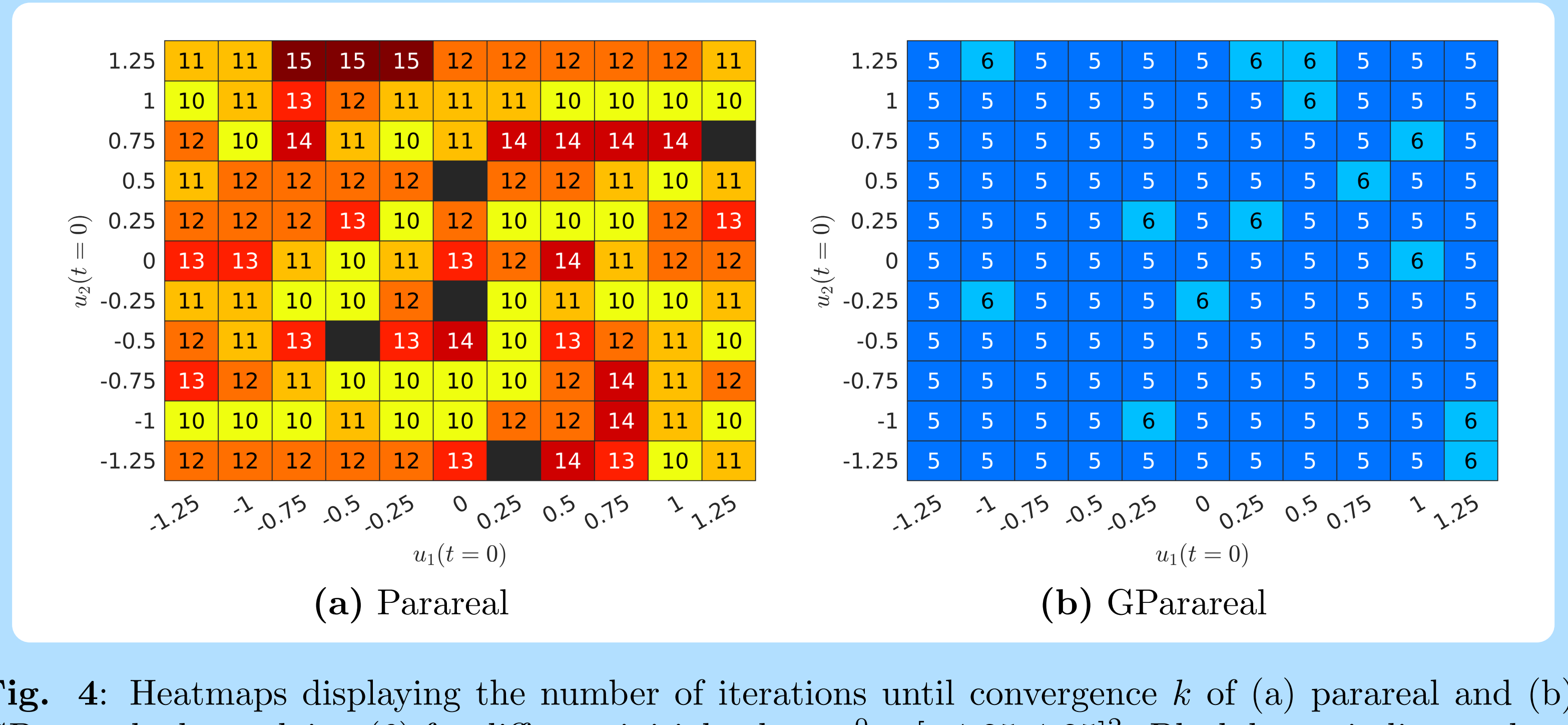

Fig. 4: Heatmaps displaying the number of iterations until convergence k of (a) parareal and (b) GParareal when solving (6) for different initial values $\boldsymbol{u}^0 \in [-1.25, 1.25]^2$. Black boxes indicate where parareal returned a **NaN** value during simulation.

Conclusions and future work

- GParareal can converge in **faster wallclock time than parareal**, locating solutions that **maintain accuracy** (wrt parareal), **even for chaotic systems** (results shown in paper[3]).
- **Future work:** implement GParareal with a **more efficient GP emulator** so that it **scales for much larger ODE systems**.
- **Longer term:** Develop a truly **probabilistic time-parallel numerical method** that returns a distribution[4] rather than point estimates over the solution.

Acknowledgements and References

The authors acknowledge funding from CCFE, EPSRC (EP S022244/1), the EUROfusion consortium, the Euratom programme (no. 633053), and the Deutsche Forschungsgemeinschaft (415980428).

[1] Lions *et al.*. 2001. *Résolution d'EDP par un schéma en temps pararéel.* Comptes Rendus de l'Academie des Sciences - Series I: Mathematics. DOI: `10.1016/S0764-4442(00)01793-6`.

[2] Samaddar *et al.*. 2010. *Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm.* J. of Comp. Phys. DOI: `10.1016/j.jcp.2010.05.012`.

[3] Pentland *et al.*. 2022. *GParareal: a time-parallel ODE solver using Gaussian process emulation.* Pre-print at: `arXiv:2201.13418`.

[4] Pentland *et al.*. 2021. *Stochastic parareal: an application of probabilistic methods to time-parallelisation.* Pre-print at: `arXiv:2106.10139`.