UKAEA-CCFE-PR(22)56

K. Pentland, M. Tamborrino, D. Samaddar, L. C. Appel

Stochastic parareal: an application of probabilistic methods to time-parallelisation

Enquiries about copyright and reproduction should in the first instance be addressed to the UKAEA Publications Officer, Culham Science Centre, Building K1/0/83 Abingdon, Oxfordshire, OX14 3DB, UK. The United Kingdom Atomic Energy Authority is the copyright holder.

The contents of this document and all other UKAEA Preprints, Reports and Conference Papers are available to view online free at scientific-publications.ukaea.uk/

Stochastic parareal: an application of probabilistic methods to time-parallelisation

K. Pentland, M. Tamborrino, D. Samaddar, L. C. Appel

This is a preprint of a paper submitted for publication in
SIAM Journal on Scientific Computing

Stochastic parareal: an application of probabilistic methods to time-parallelisation

Kamran Pentland*[1], Massimiliano Tamborrino[2], D. Samaddar[3], and L. C. Appel[3]

[1]*Mathematics Institute, University of Warwick, Coventry, CV4 7AL, United Kingdom*
[2]*Department of Statistics, University of Warwick, Coventry, CV4 7AL, United Kingdom*
[3]*Culham Centre for Fusion Energy, Culham Science Centre, Abingdon, Oxfordshire, OX14 3DB, United Kingdom*

July 8, 2022

Abstract

Parareal is a well-studied algorithm for numerically integrating systems of time-dependent differential equations by parallelising the temporal domain. Given approximate initial values at each temporal sub-interval, the algorithm locates a solution in a fixed number of iterations using a predictor-corrector, stopping once a tolerance is met. This iterative process combines solutions located by inexpensive (coarse resolution) and expensive (fine resolution) numerical integrators. In this paper, we introduce a *stochastic parareal* algorithm aimed at accelerating the convergence of the deterministic parareal algorithm. Instead of providing the predictor-corrector with a deterministically located set of initial values, the stochastic algorithm samples initial values from dynamically varying probability distributions in each temporal sub-interval. All samples are then propagated in parallel using the expensive integrator. The set of sampled initial values yielding the most continuous (smoothest) trajectory across consecutive sub-intervals are fed into the predictor-corrector, converging in fewer iterations than the deterministic algorithm with a given probability. The performance of the stochastic algorithm, implemented using various probability distributions, is illustrated on low-dimensional systems of ordinary differential equations (ODEs). We provide numerical evidence that when the number of sampled initial values is large enough, stochastic parareal converges almost certainly in fewer iterations than the deterministic algorithm, maintaining solution accuracy. Given its stochastic nature, we also highlight that multiple simulations of stochastic parareal return a distribution of solutions that can represent a measure of uncertainty over the ODE solution.

Keywords. parareal, time-parallel integration, probabilistic numerics, sampling-based solver, multivariate copulas

1 Introduction

In its most basic form, parallel computing is the process by which an algorithm is partitioned into a number of sub-problems that can be solved simultaneously without prior knowledge of each other. More widespread

*Corresponding author: kamran.pentland@warwick.ac.uk parallelism is becoming increasingly necessary in many different fields to reduce the computational burden and thus overcome the physical limitations (i.e. space, power usage, clock speeds, cooling, and financial costs) arising on machine hardware. Complex models in science often involve solving large systems of ordinary or partial differential equations (ODEs or PDEs) which, in the case of spatio-temporal PDEs, can be parallelised using existing domain decomposition methods. We refer to [1] for an overview. Although very efficient for high dimensional systems, these methods are reaching scale-up limits, and wallclock speeds often bottleneck due to the temporal integration. For instance, modern algorithms used to simulate Edge Localised Modes in turbulent fusion plasmas can take anywhere between 100-200 days to integrate over a time interval of just one second [2].

These sequential bottlenecks in time have motivated the research and development of *time-parallel* integration methods for systems of ODEs and PDEs over the last 55 years or so (see [3, 4, 5] for reviews). These methods provide a way to integrate initial value problems (IVPs) over long time intervals where solutions would be unobtainable using existing sequential algorithms. One approach to integrate in a non-sequential manner, similar to spatial parallelisation, is to discretise the time interval into N sub-intervals upon which N sub-problems are solved in parallel using existing numerical methods. The solution at a given time step, however, depends upon the solution at the previous step(s). This creates a problem prior to the parallel integration as $N-1$ of the N initial values (from which to begin integration in each sub-interval) are unknown *a priori*. Existing algorithms that attempt to locate these $N-1$ initial values by direct or iterative means have been collectively referred to as multiple shooting methods [6, 7, 8, 9, 10].

Our focus is on one multiple shooting method in particular: *parareal* [8]. This algorithm uses a combination of coarse and fine grained (in time) numerical integrators to provide parallel speedup, and has been tested successfully on problems spanning molecular [11] and fluid dynamics [12, 13, 14], to stochastic differential equations [15, 16] and fusion plasma simulations [2, 17]. The popularity of parareal is due to its relatively straightforward implementation and demonstrable effectiveness in speeding up the integration of IVPs. The algorithm iteratively locates a numerical solution at each sub-interval using a predictor-corrector scheme, derived by discretising the Newton-Raphson method. The algorithm stops once a tolerance is met with convergence guaranteed under certain mathematical conditions – more detail on the algorithm is provided in Section 2. Much work has gone into analysing the numerical convergence of this method [18, 19, 20, 21], combining it with spatial decomposition techniques [22], and developing variants that utilise idle processors [23] and adaptive time-stepping [24]. However, given fixed fine and coarse integrators, the method itself is strictly deterministic and little work has gone into trying to reduce the number of iterations $k_d \in \{1,\ldots,N\}$ that parareal takes to solve a particular IVP. Our primary focus is on reducing k_d and henceforth we refer to it as the 'convergence rate'. In scenarios where parareal struggles to converge in $k_d \ll N$ iterations, reducing k_d by even a few iterations, say to $k_s < k_d$, can lead to significant parallel gains (roughly a factor k_d/k_s), enabling faster numerical integration.

The purpose of this work is to extend the parareal algorithm using probabilistic methods to converge in fewer than k_d iterations for a given system of ODEs. To achieve this, we introduce a *stochastic* parareal algorithm. Instead of integrating forward in time from a single initial value (given by the predictor-corrector) in each sub-interval, a pre-specified probability distribution (see sampling rules in Section 3.2) is used to generate M candidate initial values that are *simultaneously* integrated forward in time in parallel. At each sub-interval, one optimal initial value (from the M samples) is selected sequentially such that the most continuous trajectory is chosen across consecutive sub-intervals. These initial values are then fed directly into the predictor-corrector – the idea being that this stochastically generated set of values provides a better guess to the solution than those found purely deterministically. For example, suppose running the predictor-corrector with initial values $\boldsymbol{x}^{(0)}$ yields convergence in k_d iterations, generating the sequence of solutions $\{\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k_d)}\}$. Instead of starting with $\boldsymbol{x}^{(0)}$, suppose we sample initial values from a probability distribution and choose some "better" starting point which is close to, say, $\boldsymbol{x}^{(i)}$ for some $i \in \{1,\ldots,k_d-1\}$. Then the sequence generated by the predictor-corrector would instead be approximately $\{\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i+1)}, \ldots, \boldsymbol{x}^{(k_d)}\}$, converging in $k_d - i$ iterations. Therefore, given a fixed number of samples M, the stochastic parareal algorithm will converge in fewer than k_d iterations with some non-zero probability.

This idea of propagating multiple initial values in each sub-interval in parallel is not new. In the first major work proposing the time-parallel integration of IVPs, Nievergelt [9] discussed choosing M initial values deterministically. The method for determining the solution from this ensemble of trajectories was to combine two of the M samples in each sub-interval using an interpolation coefficient determined from the preceding sub-interval. Whilst an excellent first incursion into the field, for nonlinear problems this direct method suffered from interpolation errors and questions remained on how to efficiently scale this up to systems of ODEs. We generalise the original idea of Nievergelt by combining it with the parareal algorithm, generating the M initial values at each sub-interval using probability distributions based on information known about the solutions at the current iteration.

The paper is organised as follows. In Section 2, we recall the parareal algorithm and its properties from a multiple shooting viewpoint, including a known modification that will enable us to extend the algorithm to incorporate stochastic sampling. In Section 3, we introduce the stochastic parareal algorithm. Following an explanation of its key features, we elucidate how a variety of different sampling rules can be flexibly interchanged in order to carry out the sampling. In Section 4, we conduct numerical experiments to illustrate the performance of stochastic parareal against its deterministic counterpart using the different sampling rules. Findings are presented for three ODE systems of increasing complexity, with two additional examples given in the Appendix. Results indicate that for sufficiently many samples, stochastic parareal almost certainly beats the convergence rate of the deterministic algorithm and generates (stochastic) solutions of comparable numerical accuracy. For the multivariate ODEs, results show that performance is improved by generating correlated, as opposed to uncorrelated, random samples. In Section 5, conclusions are drawn and avenues for future research are discussed.

2 The parareal algorithm

Following previously outlined descriptions of the parareal algorithm [20, 8], henceforth referred to as 'parareal' or $\mathcal{P}$, consider a system of $d \in \mathbb{N}$ ODEs

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}\big(\mathbf{u}(t), t\big) \quad \text{on} \quad t \in [T_0, T_N], \quad \text{with} \quad \mathbf{u}(T_0) = \mathbf{u}^0, \tag{1}$$

where $\mathbf{f} : \mathbb{R}^d \times [T_0, T_N] \to \mathbb{R}^d$ is a smooth multivariate (possibly nonlinear) function, $\mathbf{u} : [T_0, T_N] \to \mathbb{R}^d$ the time-dependent vector solution, and $\mathbf{u}^0 \in \mathbb{R}^d$ the initial values at T_0. Decompose the time domain into N sub-intervals such that $[T_0, T_N] = [T_0, T_1] \cup \cdots \cup [T_{N-1}, T_N]$, with each sub-interval taking fixed length $\Delta T := T_n - T_{n-1}$ for $n = 1, \ldots, N$ (see Figure 1 for a schematic).

Having partitioned the time domain, N smaller sub-problems

$$\frac{d\mathbf{u}_n}{dt} = \mathbf{f}\big(\mathbf{u}_n(t|\mathbf{U}_n), t\big) \quad \text{on} \quad t \in [T_n, T_{n+1}], \quad \text{with} \quad \mathbf{u}_n(T_n) = \mathbf{U}_n, \tag{2}$$

for $n = 0, \ldots, N-1$ can theoretically be solved in parallel on N processors, henceforth denoted $P_1, \ldots, P_N$. Each solution $\mathbf{u}_n(t|\mathbf{U}_n)$ is defined over $[T_n, T_{n+1}]$ *given* the initial values $\mathbf{U}_n \in \mathbb{R}^d$ at $t = T_n$. Note however that only the initial value $\mathbf{U}_0 = \mathbf{u}_0(T_0) = \mathbf{u}^0$ is known, whereas the rest ($\mathbf{U}_n$ for $n \geq 1$) need to be determined before (2) can be solved in parallel. These initial values must satisfy the continuity conditions

$$\mathbf{U}_0 = \mathbf{u}^0 \quad \text{and} \quad \mathbf{U}_n = \mathbf{u}_{n-1}(T_n|\mathbf{U}_{n-1}) \quad \text{for} \quad n = 1, \ldots, N, \tag{3}$$

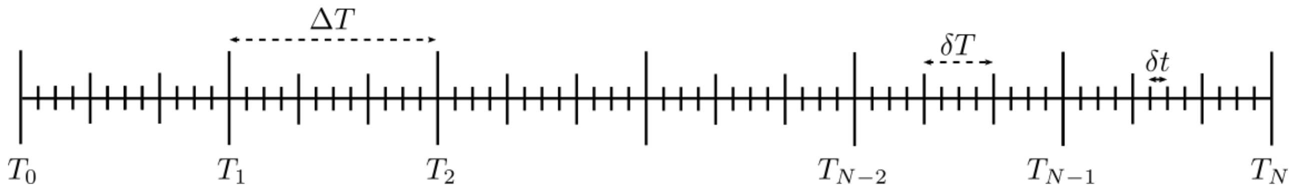


Figure 1: Schematic of the time domain decomposition. Three levels of temporal discretisation are shown: sub-intervals (size ΔT), coarse intervals (size δT), and fine intervals (size δt). Note how the discretisations align with one another and that $\delta t \ll \delta T \leq \Delta T$ in our implementation.

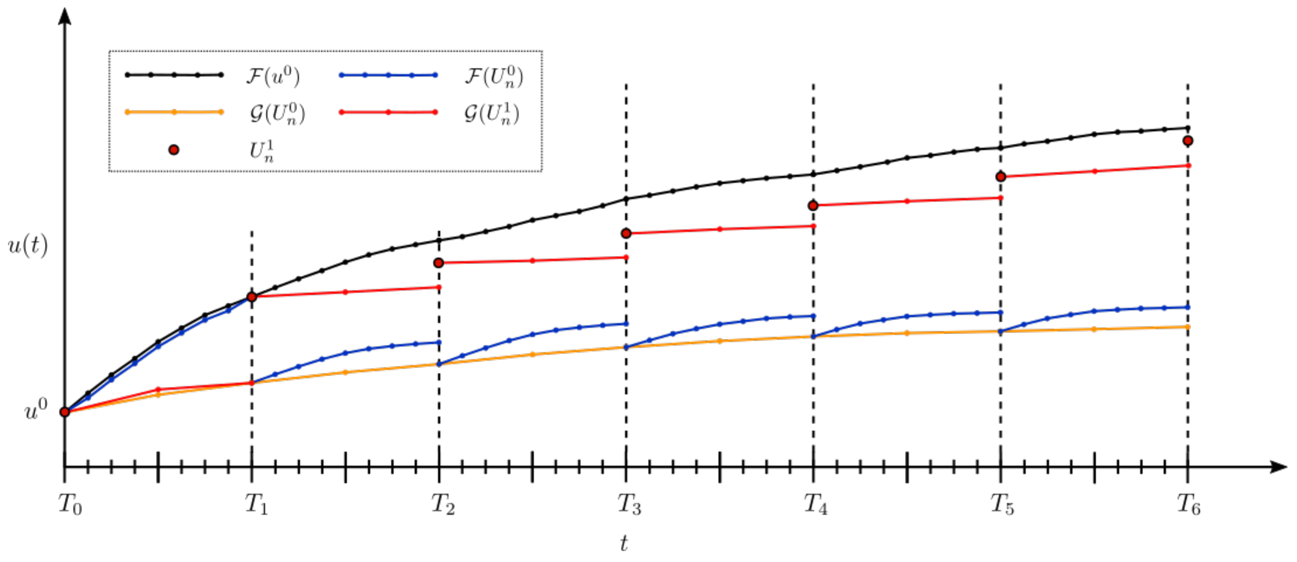


Figure 2: First iteration of the parareal algorithm to numerically evaluate the solution of a scalar ODE, either available or obtained via the fine solver (black line). The first runs of $\mathcal{G}$ and $\mathcal{F}$ are given in yellow and blue respectively; and the second run of $\mathcal{G}$ in red. The red dots represent the solution after applying the predictor-corrector (5).

which form a (nonlinear) system of $N+1$ equations that ensure solutions match at each T_n $\forall n \geq 1$. System (3) is solved for $\mathbf{U}_n$ using the Newton-Raphson method to form the iterative system

$$\mathbf{U}_0^{k+1} = \mathbf{u}^0, \tag{4a}$$

$$\mathbf{U}_n^{k+1} = \mathbf{u}_{n-1}(T_n|\mathbf{U}_{n-1}^k) + \frac{\partial \mathbf{u}_{n-1}}{\partial \mathbf{U}_{n-1}}(T_n|\mathbf{U}_{n-1}^k)\Big[\mathbf{U}_{n-1}^{k+1} - \mathbf{U}_{n-1}^k\Big], \tag{4b}$$

for $n = 1,\dots,N$, where $k = 0,1,2,\dots$ is the iteration number. This system contains the unknown solutions $\mathbf{u}_n$ and their partial derivatives, which even if known, would be computationally expensive to calculate.

To solve (4) and evaluate $\mathbf{u}_n$ at discrete times, $\mathcal{P}$ utilises two numerical integrators. The first is a numerically fast *coarse integrator* $\mathcal{G}$ that integrates over $[T_n, T_{n+1}]$ using initial values $\mathbf{U}_n$. The second is a *fine integrator* $\mathcal{F}$ that runs much slower compared to $\mathcal{G}$ but has much greater numerical accuracy, chosen such that it integrates over the same interval $[T_n, T_{n+1}]$ with initial values $\mathbf{U}_n$. In our implementation, the difference between fast and slow integration is guaranteed by setting the time steps for $\mathcal{G}$ and $\mathcal{F}$ as δT and δt respectively with $\delta t \ll \delta T$ (recall Figure 1). The key principle is that, if used to integrate (1) over $[T_0, T_N]$ serially, $\mathcal{F}$ would take an infeasible amount of computational time, hence the need to use $\mathcal{P}$ in the first place. Therefore $\mathcal{G}$ is permitted to run serially across multiple sub-intervals rapidly, whereas the slower solver $\mathcal{F}$ is only permitted to run in parallel on sub-intervals. This is a strict requirement for running $\mathcal{P}$, else numerical speedup will not be achieved.

Given that (4a) is known *a priori* for all k, Lions et al. [8] approximate the first term in equation (4b) using the fine solver $\mathcal{F}(\mathbf{U}_{n-1}^k)$ and the second term by a coarse finite difference of the derivative using $\mathcal{G}(\mathbf{U}_{n-1}^{k+1}) - \mathcal{G}(\mathbf{U}_{n-1}^k)$. One could instead approximate the derivative using a fine finite difference $\mathcal{F}(\mathbf{U}_{n-1}^{k+1}) - \mathcal{F}(\mathbf{U}_{n-1}^k)$, however (4b) then becomes a sequential calculation using just $\mathcal{F}$ – exactly what we wish to avoid. Assuming $\mathcal{G}$ meets the conditions required for (4b) to converge (see Section 2.1), using the coarse approximation becomes reasonable and enables the parallel computation of (4b) [20]. The result is that an initial guess for the initial values $\mathbf{U}_n^0$ (found using $\mathcal{G}$) is improved at successive parareal iterations k using the predictor-corrector update rule

$$\mathbf{U}_n^{k+1} = \mathcal{G}(\mathbf{U}_{n-1}^{k+1}) + \mathcal{F}(\mathbf{U}_{n-1}^k) - \mathcal{G}(\mathbf{U}_{n-1}^k) \quad \text{for} \quad n = 1,\dots,N. \tag{5}$$

Pseudocode for an implementation of $\mathcal{P}$ is given in Algorithm 1 alongside a graphical description of the first iteration in Figure 2. During the 'zeroth' iteration, $\mathcal{G}$ is applied to $\mathbf{u}^0$, generating initial values $\hat{\mathbf{U}}_n^0$ $\forall n \in \{1,\dots,N\}$

sequentially (lines 1-6). Immediately following, $\mathcal{F}$ is run in parallel from these initial values to generate a more accurate set of fine solution states $\tilde{\mathbf{U}}_n^0$ (lines 8-11). Next, $\mathcal{G}$ is run in the first time sub-interval, from the known initial value, to 'predict' the solution $\hat{\mathbf{U}}_1^1$ at T_1. It is subsequently 'corrected' using rule (5), and the predictor-corrector solution $\mathbf{U}_1^1$ is found at T_1. This prediction and correction process (lines 12-16) repeats sequentially until $\mathbf{U}_n^1$ is found $\forall n \in \{1,\dots,N\}$. After checking if the stopping criteria has been met (lines 17-21), the algorithm either starts the next iteration or stops.

Algorithm 1 Parareal ($\mathcal{P}$)

```
1: Set counters $k = I = 0$ and define $\mathbf{U}_n^k$, $\hat{\mathbf{U}}_n^k$ and $\tilde{\mathbf{U}}_n^k$ as the predictor-corrector, coarse, and fine
   solutions at the $n^{th}$ time step and $k^{th}$ iteration respectively (note that $\mathbf{U}_0^k = \hat{\mathbf{U}}_0^k = \tilde{\mathbf{U}}_0^k = \mathbf{u}^0\ \forall k$).
2: // Calculate initial values at the start of each sub-interval $T_n$ by running $\mathcal{G}$ serially on processor $P_1$.
3: for $n = 1$ to $N$ do
4:     $\hat{\mathbf{U}}_n^0 = \mathcal{G}(\hat{\mathbf{U}}_{n-1}^0)$
5:     $\mathbf{U}_n^0 = \hat{\mathbf{U}}_n^0$
6: end for
7: for $k = 1$ to $N$ do
8:     // Propagate the predictor-corrector solutions (from iteration $k-1$) on each sub-interval by
       running $\mathcal{F}$ in parallel on processors $P_{I+1},\dots,P_N$.
9:     for $n = I+1$ to $N$ do
10:        $\tilde{\mathbf{U}}_n^{k-1} = \mathcal{F}(\mathbf{U}_{n-1}^{k-1})$
11:    end for
12:    // Propagate the predictor-corrector solution (at iteration $k$) with $\mathcal{G}$ on any available processor.
       Then correct this value using coarse and fine solutions obtained during iteration $k-1$ (this step
       *cannot* be carried out in parallel).
13:    for $n = I+1$ to $N$ do
14:        $\hat{\mathbf{U}}_n^k = \mathcal{G}(\mathbf{U}_{n-1}^k)$
15:        $\mathbf{U}_n^k = \tilde{\mathbf{U}}_n^{k-1} + \hat{\mathbf{U}}_n^k - \hat{\mathbf{U}}_n^{k-1}$
16:    end for
17:    // Check whether the stopping criterion is met, saving all solutions up to time step $T_I$ before the
       next iteration. If tolerance is met for all time steps, the algorithm stops.
18:    $I = \max_{n \in \{I+1,\dots,N\}} \|\mathbf{U}_i^k - \mathbf{U}_i^{k-1}\|_\infty < \varepsilon\ \forall i < n$
19:    if $I == N$ then
20:        return $k$, $\mathbf{U}^k$
21:    end if
22: end for
```

2.1 Stopping criteria and other properties

The algorithm is said to have converged up to time T_I if

$$\|\mathbf{U}_n^k - \mathbf{U}_n^{k-1}\|_\infty < \varepsilon \quad \forall n \le I, \tag{6}$$

for some small fixed tolerance ε – noting that $\|\cdot\|_\infty$ denotes the usual infinity norm [21, 17]. Taking the relative, instead of absolute, errors in (6) would also be appropriate, however, in our numerical experiments the results (not reported) did not change. Once $I = N$, we say $\mathcal{P}$ has taken $k_d = k$ iterations to converge, yielding a solution with numerical accuracy of the order of the $\mathcal{F}$ solver. In its original formulation, $\mathcal{P}$ iteratively improves the solution across *all* time steps, regardless of whether they have converged or not, up until the tolerance has been met for all T_n. The modified formulation presented here however only iterates on the solutions for the unconverged sub-intervals [23, 17]. This has no effect on the convergence rate k_d and becomes especially important once we introduce the stochastic modifications in Section 3.

It should be clear that *at least* one sub-interval will converge during each iteration k, as $\mathcal{F}$ will be run directly from a converged initial value. Therefore it will take at most $k_d = N$ iterations for $\mathcal{P}$ to converge, equivalent to running $\mathcal{F}$ over the N sub-intervals serially (yielding zero parallel speedup). To achieve significant parallel speedup, multiple sub-intervals need to converge during an iteration so that $k_d \ll N$. Assuming $\mathcal{G}$ takes a negligible amount of time to run compared to the parallel components (along with any other serial computations), an approximate upper bound on the speedup achieved by $\mathcal{P}$ is N/k_d.

One challenge in attempting to minimise k_d, hence the overall runtime of $\mathcal{P}$, is to identify optimal solvers $\mathcal{F}$ and $\mathcal{G}$ for implementation. Whereas $\mathcal{F}$ is assumed to have high-accuracy and be computationally expensive to run, $\mathcal{G}$ must be chosen such that it runs significantly faster (usually orders of magnitude) than $\mathcal{F}$ whilst being sufficiently numerically accurate to converge in as few iterations as possible. Usually $\mathcal{G}$ is chosen such that its solutions have lower numerical accuracy, coarser temporal resolution, and/or reduced physics/coarser spatial resolution (when solving PDEs). There is currently no rigorous method for choosing the two solvers, however they should be chosen such that the ratio between the time taken to run $\mathcal{F}$ over $[T_n, T_{n+1}]$ and the time taken to run $\mathcal{G}$ over the same interval is large.

In this paper, we fix both $\mathcal{F}$ and $\mathcal{G}$ as explicit[1] fourth-order Runge-Kutta methods (henceforth RK4) for ease of implementation. More detailed analysis on the mathematical conditions required for $\mathcal{P}$ to converge, i.e. for the numerical solution $\mathbf{U}_n$ to approach the fine solution, can be found in [25, 20, 8, 22].

3 A stochastic parareal algorithm

In this section, we introduce the stochastic parareal algorithm, an extension of parareal that incorporates randomness and utilises its well-studied deterministic convergence properties to locate a solution in $k_s < k_d$ iterations. A summary of additional notation required to describe the algorithm, henceforth referred to as $\mathcal{P}_s$, is provided in Table 1.

The idea behind $\mathcal{P}_s$ is to sample M vectors of initial values $\boldsymbol{\alpha}^k_{n,1}, \dots, \boldsymbol{\alpha}^k_{n,M}$, at each unconverged sub-interval T_n, in the neighbourhood of the current predictor-corrector solution $\mathbf{U}^k_n$ from a given probability distribution (with sufficiently broad support) and propagate them all in parallel using $\mathcal{F}$. Given a sufficient number of samples is taken, one will be closer (in the Euclidean sense) to the true root that equation (5) is converging toward. Among them, we select an optimal $\hat{\boldsymbol{\alpha}}^k_n$ by identifying which samples generate the most continuous trajectory, at the fine resolution, in phase space across $[T_0, T_N]$. Therefore, at each iteration, we stochastically “jump" toward more accurate initial values and then feed them into the predictor-corrector (5). For increasing values of M, the convergence rate k_s will decrease, satisfying $k_s < k_d$ with probability one – shown numerically in Section 4.

3.1 The algorithm

Suppose again we aim to solve system (1), adopting the same conditions, properties, and notation as discussed in Section 2. $\mathcal{P}_s$ follows the first iteration ($k = 1$) of $\mathcal{P}$ identically – see line 1 of Algorithm 2. This is because information about initial values at the different temporal resolutions (i.e. results from $\mathcal{F}$ and $\mathcal{G}$) are required to construct the appropriate probability distributions for sampling. Following the convergence check, we assume (for the purposes of explaining the stochastic iterations) that only the first sub-interval $[T_0, T_1]$ converged during $k = 1$, leaving $N - 1$ unconverged sub-intervals. At this point we know the most up-to-date predictor-corrector solutions $\mathbf{U}^1_n\ \forall n \in \{1, \dots, N\}$ and the stochastic iterations can begin (henceforth $k = 2$).

[1]When solving the stiff ODE systems in Section 4, the initial values at $t = 0$ are chosen such that explicit methods work. For greater numerical stability, one could alternatively use implicit solvers at extra computational cost.

Table 1: Additional notation used to describe the stochastic parareal algorithm.

Notation	Description
n	Index of the time sub-interval T_n, $n = 0,\dots,N$.
k	Iteration number of $\mathcal{P}$ and $\mathcal{P}_s$, $k = 1,\dots,N$.
k_s	Total iterations taken for stochastic parareal to stop and return a solution, $k_s \in \{1,\dots,N\}$.
M	Number of random samples taken at each T_n.
Φ_n^{k-1}	The d-dimensional probability distribution used to sample initial values at time T_n and iteration k.
$\boldsymbol{\alpha}_{n,m}^{k-1}$	The m^{th} d-dimensional sample from distribution Φ_n^{k-1} at time T_n and iteration k, $m = 1,\dots,M$.
$\hat{\boldsymbol{\alpha}}_n^{k-1}$	The selected d-dimensional sample from $\boldsymbol{\alpha}_{n,1}^{k-1},\dots,\boldsymbol{\alpha}_{n,M}^{k-1}$ at time T_n and iteration k.
$\boldsymbol{\mu}_n^{k-1}$	The d-dimensional vector of marginal means at time T_n and iteration k.
$\boldsymbol{\sigma}_n^{k-1}$	The d-dimensional vector of marginal standard deviations at time T_n and iteration k.
$\rho_{n_{i,j}}^{k-1}$	Pairwise correlations coefficients between i^{th} and j^{th} components of the M fine resolution propagated samples at time T_n and iteration k.
$\mathbf{R}_n^{k-1}$	The symmetric positive semi-definite $d \times d$ correlation matrix with elements $\rho_{n_{i,j}}^{k-1}$ at time T_n and iteration k.
$\mathbb{I}$	The $d \times d$ identity matrix.
Σ_n^{k-1}	The symmetric positive semi-definite $d \times d$ covariance matrix at time T_n and iteration k.

At any unconverged T_n $(n > 1)$, we sample M vectors of initial values, denoted $\boldsymbol{\alpha}_{n,m}^{k-1}$ for $m = 1,\dots,M$. The first sample is fixed as the predictor-corrector $\mathbf{U}_n^{k-1}$, to ensure that $\mathcal{P}_s = \mathcal{P}$ when $M = 1$. The other $M-1$ initial values are sampled from a pre-specified d-dimensional probability distribution Φ_n^{k-1} with finite marginal means $\boldsymbol{\mu}_n^{k-1} = (\mu_{n_1}^{k-1},\dots,\mu_{n_d}^{k-1})^{\mathrm{T}}$, marginal standard deviations $\boldsymbol{\sigma}_n^{k-1} = (\sigma_{n_1}^{k-1},\dots,\sigma_{n_d}^{k-1})^{\mathrm{T}}$, and correlation structure given by the matrix $\mathbf{R}_n^{k-1}$. These quantities depend upon the information available at iteration $k-1$, i.e. a combination of $\mathbf{U}_n^{k-1}$, $\mathcal{F}(\mathbf{U}_n^{k-1})$, $\mathcal{G}(\mathbf{U}_n^{k-1})$ and $\mathcal{G}(\mathbf{U}_n^{k-2})$, see Section 3.2. The correlation matrix $\mathbf{R}_n^{k-1}$ is introduced to take into account the dependence between components of the ODE system (lines 3-9). The elements of $\mathbf{R}_n^{k-1}$, for $k \geq 3$, are defined using the Pearson correlation coefficient

$$\rho_{n_{i,j}}^{k-1} = \frac{\sum_{m=1}^{M}(x_m^{(i)} - \bar{x}^{(i)})(x_m^{(j)} - \bar{x}^{(j)})}{\sqrt{\sum_{m=1}^{M}(x_m^{(i)} - \bar{x}^{(i)})^2}\sqrt{\sum_{m=1}^{M}(x_m^{(j)} - \bar{x}^{(j)})^2}}, \quad i,j \in \{1,\dots,d\}, \tag{7}$$

where

$$x_m^{(i)} = \mathcal{F}(\boldsymbol{\alpha}_{n-1,m}^{k-2})(i), \qquad \bar{x}^{(i)} = \frac{1}{M}\sum_{m=1}^{M} x_m^{(i)}, \tag{8}$$

and $\mathcal{F}(\boldsymbol{\alpha}_{n-1,m}^{k-2})(i)$ denotes the ith element of $\mathcal{F}(\boldsymbol{\alpha}_{n-1,m}^{k-2})$. The coefficients $\rho_{n_{i,j}}^{k-1}$ in (7) are essentially the estimated pairwise correlation coefficients of the M d-dimensional fine resolution propagations of the sampled initial values at T_n from the previous iteration, i.e. $\mathcal{F}(\boldsymbol{\alpha}_{n-1,1}^{k-2}),\dots,\mathcal{F}(\boldsymbol{\alpha}_{n-1,M}^{k-2})$. Note that other types of linear correlation coefficient could also be chosen. Since each $\mathcal{F}(\boldsymbol{\alpha}_{n-1,m}^{k-2})$ is not available at iteration $k = 2$, we set $\mathbf{R}_n^{k-1} = \mathbb{I}$ for $k = 2$, i.e. we sample from a multivariate distribution with uncorrelated components.

Following this, the sampling and subsequent propagation using $\mathcal{F}$ can begin in parallel (lines 10-22). Given the solution between $[T_0, T_1]$ has converged, $\mathcal{F}$ will run from the converged initial value at T_1, with sampling starting from T_2 onward (see Figure 3). *All* sampled initial values are then propagated forward in parallel using $\mathcal{F}$, requiring at least $M(N-2)+1$ processors (M samples times $N-2$ unconverged sub-intervals plus running $\mathcal{F}$ once in $[T_1, T_2]$).

Of the M sampled initial values at each T_n $(n > 1)$, only one is retained, denoted by $\hat{\boldsymbol{\alpha}}_n^{k-1}$, chosen such that it minimises the Euclidean distance between the fine solution and the sampled values (lines 23-28). To do this,

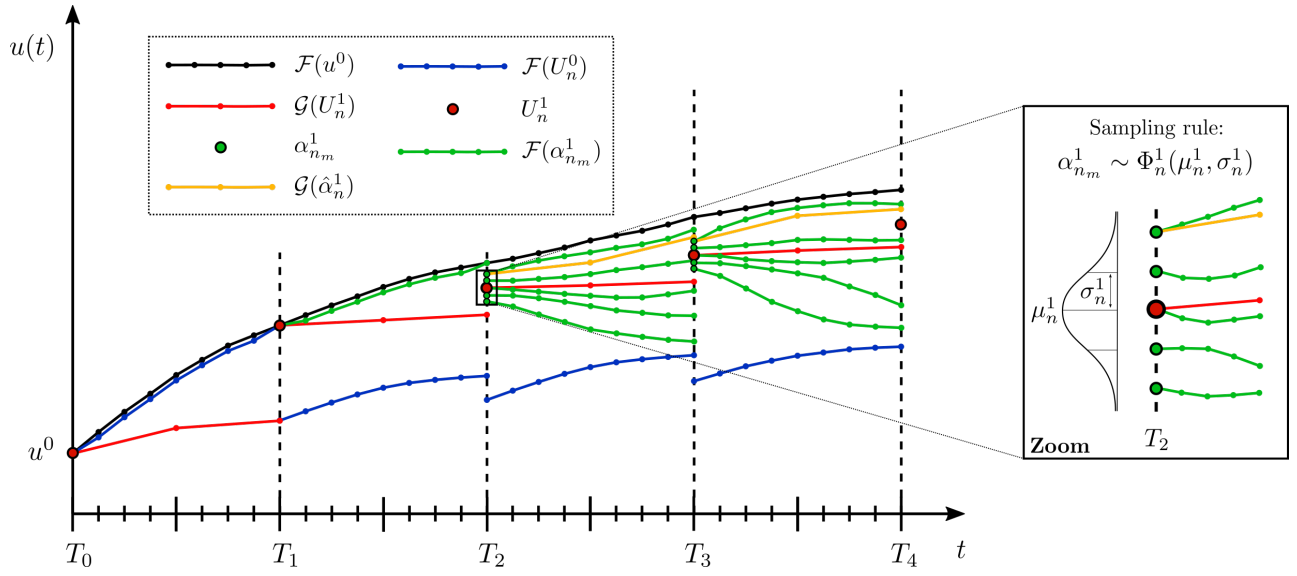


Figure 3: An illustration of the sampling and propagation process within $\mathcal{P}_s$ following iteration $k = 1$. The fine solution is given in black, the $k = 0$ fine solutions in blue, the $k = 1$ coarse solutions in red, and the $k = 1$ predictor-corrector solutions as red dots. With $M = 5$, four samples $\alpha_{n,m}^1$ (green dots) are taken at T_2 and T_3 from distributions with means U_2^1 and U_3^1, and some finite standard deviations respectively. These values, along with U_2^1 and U_3^1 themselves, are propagated in parallel forward in time using $\mathcal{F}$ (green lines). The optimally chosen samples $\hat{\alpha}_n^1$ (refer to text for how these are chosen) are then propagated forward in time using $\mathcal{G}$ (yellow lines).

start from the converged initial values at T_2 given by the fine solver: $\mathcal{F}(\mathbf{U}_1^{k-1})$. Calculate the Euclidean distance between $\mathcal{F}(\mathbf{U}_1^{k-1})$ and each of the M samples $\boldsymbol{\alpha}_{2,1}^{k-1},\dots,\boldsymbol{\alpha}_{2,M}^{k-1}$. The sample minimising this distance is chosen as $\hat{\boldsymbol{\alpha}}_2^{k-1}$. Repeat for later T_n, minimising the distance between $\mathcal{F}(\hat{\boldsymbol{\alpha}}_{n-1}^{k-1})$ and one of the samples $\boldsymbol{\alpha}_{n,1}^{k-1},\dots,\boldsymbol{\alpha}_{n,M}^{k-1}$. This process must be run sequentially and relies on the modification to $\mathcal{P}$ made in Section 2 – that solutions are not altered once converged. Referring again to Figure 3, the corresponding coarse trajectories of these optimally chosen samples $\hat{\boldsymbol{\alpha}}_n^{k-1}$ must also be calculated to carry out the predictor-corrector step (lines 29-32).

At this point, the set of initial values $\{\hat{\boldsymbol{\alpha}}_2^{k-1},\dots,\hat{\boldsymbol{\alpha}}_{N-1}^{k-1}\}$ has been selected from the ensemble of random samples, effectively replacing the previously found $\mathbf{U}_n^{k-1}$. The coarse and fine propagations of these values are now used in the predictor-corrector (lines 33-37) such that

$$\mathbf{U}_n^k = \begin{cases} \mathcal{G}(\mathbf{U}_{n-1}^k) + \mathcal{F}(\mathbf{U}_{n-1}^{k-1}) - \mathcal{G}(\mathbf{U}_{n-1}^{k-1}) & \text{for} \quad n = 2, \\ \mathcal{G}(\mathbf{U}_{n-1}^k) + \mathcal{F}(\hat{\boldsymbol{\alpha}}_{n-1}^{k-1}) - \mathcal{G}(\hat{\boldsymbol{\alpha}}_{n-1}^{k-1}) & \text{for} \quad n = 3,\dots,N. \end{cases} \tag{9}$$

Using the same stopping criteria (6) from $\mathcal{P}$ (lines 38-42), the algorithm either stops or runs another stochastic parareal iteration.

As a final remark, instead of minimising the distance between $\mathcal{F}(\hat{\boldsymbol{\alpha}}_{n-1}^{k-1})$ and one of the samples $\boldsymbol{\alpha}_{n,1}^{k-1},\dots,\boldsymbol{\alpha}_{n,M}^{k-1}$, one could think about doing some sort of interpolation to choose a more optimal point than the M samples. In this setting, however, this is not possible because we require not just the exact starting condition, which would be the optimally chosen sample, but also its value having been propagated by $\mathcal{F}$ (which we only have for the M samples).

3.2 Sampling rules

The probability distributions Φ_n^{k-1} incorporate different combinations of available information about the initial values at different temporal resolutions, i.e. the coarse, fine, and predictor-corrector initial values $\mathcal{G}(\mathbf{U}_{n-1}^{k-1})$,

Algorithm 2 Stochastic parareal ($\mathcal{P}_s$)

```
1: Run $\mathcal{P}$ (Algorithm 1) until the end of iteration $k = 1$.
2: for $k = 2$ to $N$ do
3:   // Calculate correlation matrices (only if $d > 1$).
4:   $\mathbf{R}_n^{k-1} = \mathbb{I} \ \forall n$
5:   if $k \geq 3$ then
6:     for $n = I+1$ to $N-1$ do
7:       Calculate $\mathbf{R}_n^{k-1}$ using $\mathcal{F}(\boldsymbol{\alpha}_{n-1,1}^{k-2}),\dots,\mathcal{F}(\boldsymbol{\alpha}_{n-1,M}^{k-2})$ (recall (7)).
8:     end for
9:   end if
10:  // Initial value sampling and propagation. Both line 11 and the nested for loop below (lines
     12–22) run in parallel on $P_1,\dots,P_{M(N-1-I)+1}$, i.e. all runs of $\mathcal{F}$ must be in parallel.
11:  $\tilde{\mathbf{U}}_{I+1}^{k-1} = \mathcal{F}(\mathbf{U}_I^{k-1})$ // propagate converged initial value at $T_I$ on $P_1$
12:  for $n = I+1$ to $N-1$ do
13:    for $m = 1$ to $M$ do
14:      if $m == 1$ then
15:        $\boldsymbol{\alpha}_{n,1}^{k-1} = \mathbf{U}_n^{k-1}$ // first ‘sample’ is fixed to the predictor-corrector
16:        $\tilde{\mathbf{U}}_{n+1,1} = \mathcal{F}(\boldsymbol{\alpha}_{n,1}^{k-1})$ // temporarily store propagated values
17:      else
18:        $\boldsymbol{\alpha}_{n,m}^{k-1} \sim \Phi_n^{k-1}$ // sample initial value randomly, see Section 3.2
19:        $\tilde{\mathbf{U}}_{n+1,m} = \mathcal{F}(\boldsymbol{\alpha}_{n,m}^{k-1})$
20:      end if
21:    end for
22:  end for
23:  // Select the most continuous fine trajectory from the ensemble sequentially.
24:  for $n = I+1$ to $N-1$ do
25:    $J = \underset{j\in\{1,\dots,M\}}{\operatorname{argmin}} \|\boldsymbol{\alpha}_{n,j}^{k-1} - \tilde{\mathbf{U}}_n^{k-1}\|_2$
26:    $\hat{\boldsymbol{\alpha}}_n^{k-1} = \boldsymbol{\alpha}_{n,J}^{k-1}$ // store optimal initial value
27:    $\tilde{\mathbf{U}}_{n+1}^{k-1} = \tilde{\mathbf{U}}_{n+1,J}$ // store most optimal fine trajectories
28:  end for
29:  // Run $\mathcal{G}$ from the optimal samples (can run in parallel).
30:  for $n = I+1$ to $N-1$ do
31:    $\hat{\mathbf{U}}_{n+1}^{k-1} = \mathcal{G}(\hat{\boldsymbol{\alpha}}_n^{k-1})$
32:  end for
33:  // Predict and correct the initial values.
34:  for $n = I+1$ to $N$ do
35:    $\hat{\mathbf{U}}_n^k = \mathcal{G}(\mathbf{U}_{n-1}^k)$
36:    $\mathbf{U}_n^k = \tilde{\mathbf{U}}_n^{k-1} + \hat{\mathbf{U}}_n^k - \hat{\mathbf{U}}_n^{k-1}$
37:  end for
38:  // Check whether the stopping criterion is met.
39:  $I = \max_{n\in\{I+1,\dots,N\}} \|\mathbf{U}_i^k - \mathbf{U}_i^{k-1}\|_\infty < \varepsilon \ \forall i < n$
40:  if $I == N$ then
41:    return $k$, $\mathbf{U}^k$
42:  end if
43: end for
```

$\mathcal{G}(\mathbf{U}_{n-1}^{k-2})$, $\mathcal{F}(\mathbf{U}_{n-1}^{k-2})$, and $\mathbf{U}_n^{k-1}$ respectively. This information is used to define the marginal means and standard deviations in the 'sampling rules' outlined in the following subsections. Note how the distributions vary with time T_n and k, so that the accuracy of the distributions increase as the initial values in $\mathcal{P}_s$ are iteratively improved. Using Gaussian and copula distributions, we analyse the performance of different sampling rules within $\mathcal{P}_s$ in Section 4. This will give us a more comprehensive understanding on whether the choice of distribution family Φ_n^{k-1} or the parameters $\boldsymbol{\mu}_n^{k-1}$, $\boldsymbol{\sigma}_n^{k-1}$, and $\mathbf{R}_n^{k-1}$ have the greatest impact on the convergence rate k_s. In all tested cases, we observe that taking correlated samples significantly improves the performance of $\mathcal{P}_s$ compared to the independent setting.

3.2.1 Multivariate Gaussian

First, we consider perturbing the initial values using stochastic "noise", i.e. considering errors compared to the true initial values to be normally distributed, a standard method for modelling uncertainty. The initial values $\boldsymbol{\alpha}_{n,m}^{k-1}$ are sampled from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_n^{k-1}, \Sigma_n^{k-1})$, where $(\Sigma_n^{k-1})_{i,j} = \rho_{n_{i,j}}^{k-1}\sigma_{n_i}^{k-1}\sigma_{n_j}^{k-1}$ is the $d \times d$ positive semi-definite covariance matrix. As marginal means $\boldsymbol{\mu}_n^{k-1}$ we choose either the fine resolution values $\mathcal{F}(\mathbf{U}_{n-1}^{k-2})$ (prior to correction) or the predictor-corrector values $\mathbf{U}_n^{k-1}$. For the marginal standard deviations, we choose $\boldsymbol{\sigma}_n^{k-1} = |\mathcal{G}(\mathbf{U}_{n-1}^{k-1}) - \mathcal{G}(\mathbf{U}_{n-1}^{k-2})|$ as they are of the order of the corrections made by the predictor-corrector and each marginal decreases toward zero as the algorithm converges (as expected). For the correlation coefficients, $\rho_{n_{i,j}}^{k-1}$, we calculate the linear correlation between the $\mathcal{F}$ propagated samples using Pearson's method – recall (7). Note that $|\cdot|$ denotes the component-wise absolute value. Testing revealed that alternative marginal standard deviations $|\mathbf{U}_n^{k-1} - \mathbf{U}_n^{k-2}|$ and $|\mathcal{F}(\mathbf{U}_{n-1}^{k-2}) - \mathcal{G}(\mathbf{U}_{n-1}^{k-2})|$ did not span sufficiently large distances around $\boldsymbol{\mu}_n^{k-1}$ in order for sampling to be efficient, i.e they required much higher sampling to perform as well as $\boldsymbol{\sigma}_n^{k-1} = |\mathcal{G}(\mathbf{U}_{n-1}^{k-1}) - \mathcal{G}(\mathbf{U}_{n-1}^{k-2})|$ (results not shown). The samples $\boldsymbol{\alpha}_{n,m}^{k-1} \sim \mathcal{N}(\boldsymbol{\mu}_n^{k-1}, \Sigma_n^{k-1})$ are taken according to the following sampling rules:

$$\textbf{Rule 1}: \ \boldsymbol{\mu}_n^{k-1} = \mathcal{F}(\mathbf{U}_{n-1}^{k-2}) \text{ and } \boldsymbol{\sigma}_n^{k-1} = |\mathcal{G}(\mathbf{U}_{n-1}^{k-1}) - \mathcal{G}(\mathbf{U}_{n-1}^{k-2})|.$$
$$\textbf{Rule 2}: \ \boldsymbol{\mu}_n^{k-1} = \mathbf{U}_n^{k-1} \text{ and } \boldsymbol{\sigma}_n^{k-1} = |\mathcal{G}(\mathbf{U}_{n-1}^{k-1}) - \mathcal{G}(\mathbf{U}_{n-1}^{k-2})|.$$

Note that a linear combination of the rules or taking half the samples from each appears to work well, with performance similar to the individual rules themselves (results not shown).

3.2.2 Multivariate copula

Alternatively, we consider errors that may have a different (possibly non-Gaussian) dependency structure. We consider another multivariate distribution, known as a copula, with uniformly distributed marginals scaled such that they have the same value as the marginal means and standard deviations in Section 3.2.1.

A copula $\mathcal{C} : [0,1]^d \to [0,1]$ is a joint cumulative distribution function, of a d-dimensional random vector, with uniform marginal distributions [26]. Sklar's theorem [27] states that any multivariate cumulative distribution function with continuous marginal distributions can be written in terms of d uniform marginal distributions and a copula that describes the correlation structure between them. Whilst there are numerous families of copulas, we consider the symmetric t-copula, $\mathcal{C}^t$, the copula underlying the multivariate t-distribution, which depends on the parameter ν and the correlation matrix $\mathbf{R}_n^{k-1}$. We fix $\nu = 1$ so that samples have a higher probability of being drawn toward the edges of the $[0,1]^d$ hypercube, see [26]. Note that $\nu \to \infty$ can be thought of as sampling from the Gaussian copula.

Correlated samples $\boldsymbol{\chi} = (\chi_1, \dots, \chi_d)^{\mathrm{T}} \sim \mathcal{C}^t$ that are generated in $[0,1]^d$ then need to be re-scaled such that each marginal is uniformly distributed in an interval $[x_i, y_i] \subset \mathbb{R}$, with mean μ_i and standard deviation σ_i for $i \in \{1, \dots, d\}$. By definition, a marginal uniform distribution on $[x_i, y_i]$ has mean $(x_i + y_i)/2$ and variance $(y_i - x_i)^2/12$ which we set to μ_i and σ_i^2 respectively. Solving these equations, we find that the desired marginals are uniform distributions on $[\mu_i - \sqrt{3}\sigma_i, \mu_i + \sqrt{3}\sigma_i]$. Thus, setting $2\sqrt{3}\sigma_i\chi_i + \mu_i - \sqrt{3}\sigma_i$ guarantees that the generated samples $\boldsymbol{\chi} \sim \mathcal{C}^t$ have the same marginal means μ_i and standard deviations σ_i as the Gaussian distributions. This allows us to compare performance results in Section 4. The t-copula sampling rules (**Rule 3** and **Rule 4**) are therefore defined component-wise as $\boldsymbol{\alpha}_{n,m}^{k-1}(i) = 2\sqrt{3}\sigma_i\chi_i + \mu_i - \sqrt{3}\sigma_i$, for $i \in \{1,\dots,d\}$, with $\boldsymbol{\chi} \sim \mathcal{C}^t$ and parameters μ_i and σ_i chosen to be identical to Rule 1 and Rule 2 respectively.

Note that this copula construction can be extended to other families of copulas (e.g. the Gaussian copula) and to copulas with different marginal distributions (e.g. Gaussian, t, or logistic marginals). Therefore, sampling from the Gaussian multivariate distributions in Section 3.2.1 can be considered a special case of these more general copulas.

3.3 Convergence

Independent simulations of $\mathcal{P}_s$ will return a numerical solution $\boldsymbol{U}_n^k$ which, along with k_s, vary stochastically – since the optimal initial values chosen will vary between simulations. Given the randomness in these quantities, we can discuss the convergence of $\mathcal{P}_s$ in two ways.

Firstly, consider convergence in terms of minimising the random variable k_s by studying $\mathbb{P}(k_s < k_d)$. Given there are no analytical results for $\mathcal{P}$ guaranteeing that $k_d < N$ for any given problem, proving that $\mathbb{P}(k_s < k_d) = 1$, or at least $\mathbb{E}(k_s) = \sum_{k=1}^{N} k\mathbb{P}(k_s = k) < k_d$ seems to be analytically intractable. However, we can qualitatively discuss $\mathbb{P}(k_s < k_d)$ and $\mathbb{E}(k_s)$ with respect to the number of samples M. Consider the following cases:

- $M = 1$
  Running $\mathcal{P}_s$ is equivalent to running $\mathcal{P}$, hence the convergence of $\mathcal{P}_s$ follows from that of $\mathcal{P}$ and therefore $\mathbb{E}(k_s) = k_s = k_d$.
- $1 < M < \infty$
  For finitely many samples, we estimate the discrete probability distributions $\mathbb{P}(k_s = k)$ and observe, in all numerical experiments (see Section 4), that $\mathbb{P}(k_s < k_d) \to 1$ for $M \approx 10$. Moreover, we observe that $\mathbb{P}(k_s < k_d)$ increases and $\mathbb{E}(k_s)$ decreases for increasing M – with $\mathbb{E}(k_s) < k_d$ for all values of M tested.
- $M \to \infty$
  If we were able to take infinitely many samples, $\mathcal{P}_s$ effectively samples every possible value in the support of Φ, i.e. every initial value that has a non-zero probability of being sampled from Φ. Therefore, if Φ has infinite support, e.g. the Gaussian distribution, all possible initial values in $\mathbb{R}^d$ are sampled and propagated, hence the fine solution will be recovered almost surely in $\mathbb{E}(k_s) = k_s = 2$ iterations. Note this is the smallest value k_s can take to converge assuming convergence does not occur following the first iteration – although the algorithm could be modified such that convergence occurs almost surely in $k_s = 1$ iterations as $M \to \infty$. In Section 4.1, we illustrate this property numerically by taking a large number of samples for a single realisation of $\mathcal{P}_s$.

In the scenario that $\mathcal{P}_s$ converges in $k_s = N$ iterations (irrespective of the value of M), it will return the fine solution just as $\mathcal{P}$ does when $k_d = N$ (having propagated the exact initial value at T_0 sequentially N times using $\mathcal{F}$).

Secondly, consider convergence in terms of the stochastic solution U_n^k (9) approaching (in the mean-square sense) the fine solution U_n as k increases. In the numerical experiments in Section 4, we did not observe a case where $\mathcal{P}_s$ fails to converge to the fine solution. In fact, we observe tight confidence intervals on the numerical errors between U_n^k and U_n upon multiple realisations of $\mathcal{P}_s$ (see Figure 7 and Figure 9). We may expect $\mathcal{P}_s$ to fail to converge (i.e. solutions blow up) in cases in which $\mathcal{P}$ also fails – typically this means that a more accurate coarse solver is required for both algorithms. It should be noted, however, that because $\mathcal{P}_s$ samples M initial values, only M' out of the M propagated solutions may blow up in each time sub-interval, due to the poor coarse solver, and so $\mathcal{P}_s$ could in fact locate a solution in cases where $\mathcal{P}$ cannot – although this is not tested here.

3.4 Computational complexity

At each iteration, $\mathcal{P}_s$ runs the fine solver more frequently than $\mathcal{P}$, albeit still in parallel, and therefore requires a larger number of processors. The first iteration of $\mathcal{P}_s$ requires N processors, however once sampling begins in $k \geq 2$, it requires at most $M(N - I - 1) + 1$ processors – assuming I sub-intervals converge during $k = 1$. Whilst

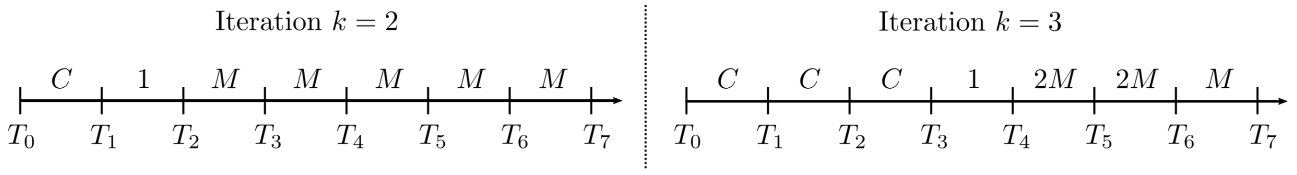


Figure 4: Illustration of a possible processor configuration if two sub-intervals were to converge between iterations two and three of $\mathcal{P}_s$. The letter C denotes a converged sub-interval, the number 1 denotes a sub-interval where we propagate the converged initial value from the preceding sub-interval using $\mathcal{F}$, and the letter M denotes the number of samples taken in an unconverged sub-interval.

we assume processors are in abundance, this number scales directly with M and so it is important to keep M to a minimum if limited processing power is available in practice.

As the stochastic iterations progress, the number of processors required, i.e. $M(N - I - 1) + 1$, decreases as the number of converged sub-intervals, I, increases – leaving a growing number of processors idle. Each additional sub-interval that converges leaves M idle processors that we can re-assign to do additional sampling and propagation. We assign each set of M idle processors to the earliest unconverged sub-interval with the least number of samples, ensuring all processors are working at all times to explore the solution space for the true initial values (see Figure 4 for an illustration). We do not explicitly write the pseudocode for re-assigning the idle processors in Algorithm 2 (lines 10-22) to avoid additional complexity – the process is, however, implemented in the numerical experiments in Section 4[2].

In terms of timings, an iteration of $\mathcal{P}_s$ takes approximately the same wallclock time as one of $\mathcal{P}$. In this regard, we assume that the extra serial costs in $\mathcal{P}_s$, e.g. correlation estimation, selecting optimal samples, and the extra $\mathcal{G}$ runs, take negligible wallclock time when compared to a single run of $\mathcal{F}$. Therefore, the wallclock time for $\mathcal{P}_s$ will be lower than for $\mathcal{P}$ if $k_s < k_d$. This comes at a cost of requiring $\mathcal{O}(MN)$ processors rather than N to solve the problem. However, it should be highlighted that if $\mathcal{P}_s$ converges in even one less iteration than $\mathcal{P}$, we avoid an extra run of $\mathcal{F}$ which may save a large amount of wallclock time.

4 Numerical results

In this section, we compare the numerical performance of $\mathcal{P}$ and $\mathcal{P}_s$ on systems of one, two, and three ODEs of varying complexity[3]. Both algorithms use RK4 methods to carry out integration, with $\mathcal{G}$ using time step δT and $\mathcal{F}$ using time step δt, the latter at least 75 times smaller than the former. We quantify the performance of $\mathcal{P}_s$ by estimating the distributions of k_s for each sampling rule and by measuring the accuracy of the stochastic solutions against those obtained serially with $\mathcal{F}$. Since a limited number of processors were available for these experiments, the results are based not on calculating wallclock runtimes but on comparing the convergence rates k_d and k_s – which are independent of the number of processors used. Additional results for these test cases, as well as two further test problems, are given in the Appendix.

4.1 Scalar nonlinear equation

First, we consider the nonlinear ODE

$$\frac{du_1}{dt} = \sin(u_1)\cos(u_1) - 2u_1 + e^{-t/100}\sin(5t) + \ln(1+t)\cos(t), \tag{10}$$

[2]To increase efficiency further we also attempted to store previously sampled and propagated fine trajectories to use in future iterations of the algorithm, however, they did not improve performance. This was because only the most recent samples were ever chosen in each iteration (results not shown).

[3]All algorithms were coded in MATLAB and simulations were run using HPC facilities at the University of Warwick. Samples code for both $\mathcal{P}$ and $\mathcal{P}_s$ can be found in the public repository at https://github.com/kpentland/StochasticParareal.

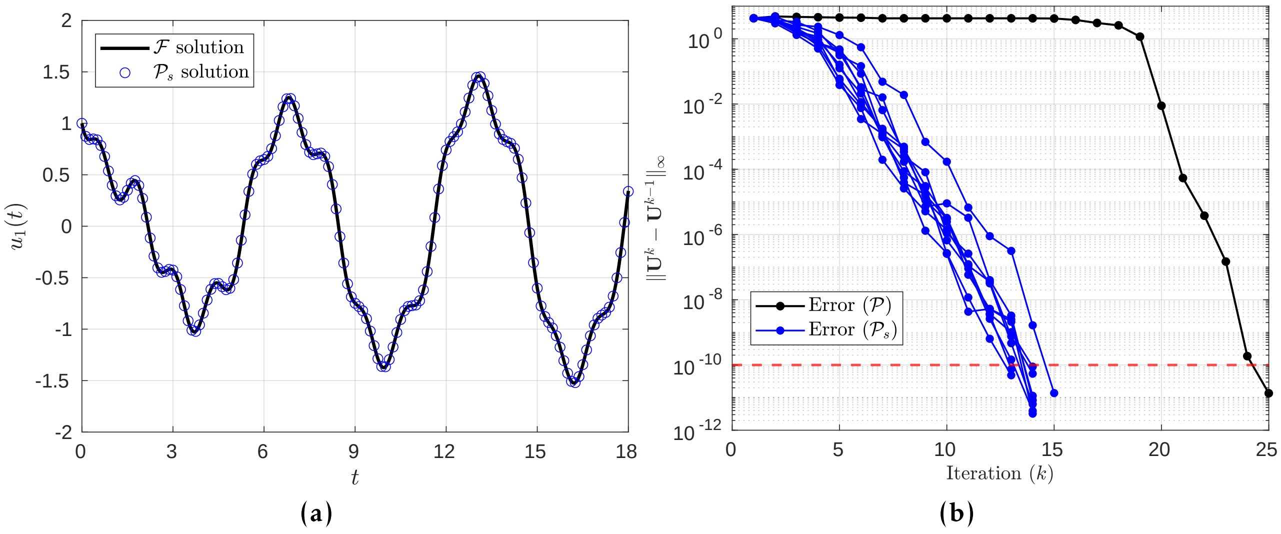


Figure 5: (a) Numerical solution of (10) over $[0,18]$ using $\mathcal{F}$ serially and a single realisation of $\mathcal{P}_s$. Note that only a subset of the fine times steps of the $\mathcal{P}_s$ solution are shown for clarity. (b) Errors at successive iterations of $\mathcal{P}$ (black line) and ten independent realisations of $\mathcal{P}_s$ (blue lines). Horizontal dashed red line represents the stopping tolerance $\varepsilon = 10^{-10}$. Note that both panels use $\mathcal{P}_s$ with sampling rule 1 and $M = 3$.

with initial value $u_1(0) = 1$ [7]. Discretise the time interval $t \in [0,100]$ using $N = 40$ sub-intervals, coarse time steps $\delta T = 100/80$, and fine time steps $\delta t = 100/8000$. Numerical solutions to (10) are shown on the interval $[0,18]$ in Figure 5a. Deterministically, $\mathcal{P}$ locates a solution in $k_d = 25$ iterations using error tolerance $\varepsilon = 10^{-10}$. $\mathcal{P}_s$ converges in a varying number of iterations k_s, with $\mathbb{P}(k_s < k_d) = 1$ – see Figure 5b for the convergence of ten independent simulations using $M = 3$. From this plot, we can see that by taking just three samples, $\mathcal{P}_s$ reduces the number of iterations by almost a factor of two – from 25 to approximately 14 (on average).

When M is increased above one, $\mathcal{P}_s$ begins generating stochastic solutions that converge in a varying number of iterations k_s. In order to accurately compare k_d with the discrete random variable k_s, we run 2000 independent simulations of $\mathcal{P}_s$ to estimate the distribution of k_s for a given M. Upon estimating these distributions, it was found that $\mathbb{P}(k_s < 25) = 1$ for each of the four sampling rules (for all $M > 1$), meaning that by doubling the number of processors we can beat parareal with probability one. The estimated distributions of k_s, using sampling rule 1 (the other rules perform similarly), as a function of M are given in Figure 6a. The stacked bars represent the estimated discrete probability of a simulation converging in a given number of iterations. The results show $\mathcal{P}_s$ converging in just five iterations in the best case – demonstrating $\mathcal{P}_s$ has the potential to yield significant parallel speedup, given sufficiently many samples are drawn. Figure 6b emphasises the power of the stochastic method, showing that the estimated expected value $\mathbb{E}(k_s)$ decreases as M increases, with the estimated standard deviation $sd(k_s) = \sqrt{\sum_{k=1}^{N}(k - \mathbb{E}(k_s))^2 \mathbb{P}(k_s = k)}$ decreasing too. The improved performance of $\mathcal{P}_s$ as M increases reflects what was discussed in Section 3.3. In particular, we ran a single realisation of $\mathcal{P}_s$ with $M = 10^6$, observing that $\mathcal{P}_s$ converged in four iterations (result not shown), confirming that k_s continues to decrease for increasing M. By looking at Figure 6b, we see that sampling rule 1 yields the lowest expected values of k_s for small values of M, with all sampling rules performing similarly for large M.

To verify the accuracy of the stochastic solutions, we plot the difference between the mean of 2000 independent realisations of $\mathcal{P}_s$ and the serially calculated $\mathcal{F}$ solution in Figure 7. Also shown is the confidence interval given by two standard deviations of the stochastic solutions (which is at most $\mathcal{O}(10^{-11})$) and the error generated by $\mathcal{P}$. Accuracy is maintained with respect to the fine solution across the time interval, even more so than the $\mathcal{P}$ solution. See Appendix A for numerical results of $\mathcal{P}_s$ applied to a *stiff* scalar nonlinear ODE. In this case, the stiffness of the equation demands a higher value of M to improve k_s – something we also observe for the Brusselator in Section 4.2.

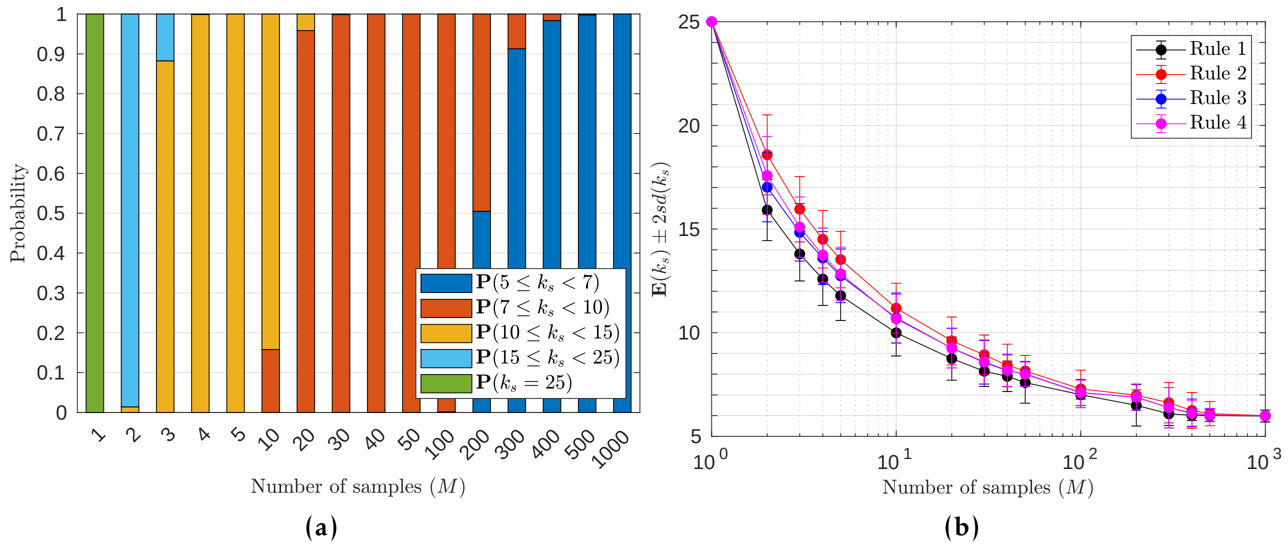


Figure 6: (a) Estimated discrete distributions of k_s as a function of M for sampling rule 1. (b) Estimated expectation of k_s as a function of M, calculated using estimated distributions of k_s for each sampling rule with error bars representing ± two standard deviations $sd(k_s)$. Distributions in both panels are estimated by simulating 2000 independent realisations of $\mathcal{P}_s$ for each M.

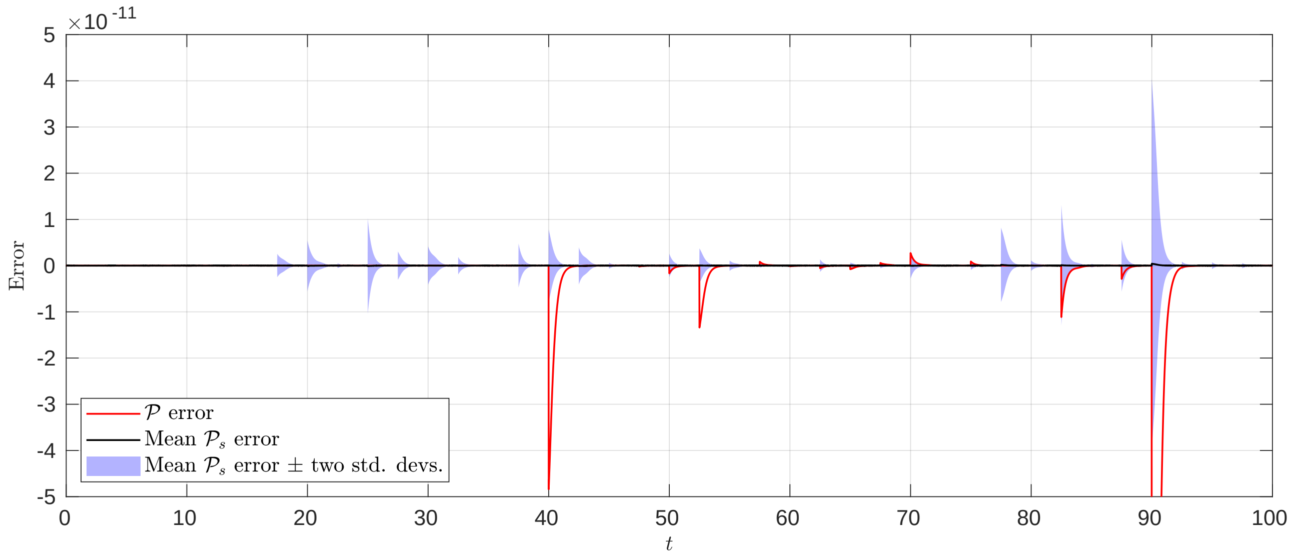


Figure 7: Errors of $\mathcal{P}$ (red) and mean $\mathcal{P}_s$ (black) solutions against the serial $\mathcal{F}$ solution over time. The mean error is obtained by running 2000 independent realisations of $\mathcal{P}_s$ with sampling rule 2 and $M = 4$ – the confidence interval representing the mean ± two standard deviations is shown in light blue.

4.2 The Brusselator system

Next, consider the Brusselator system

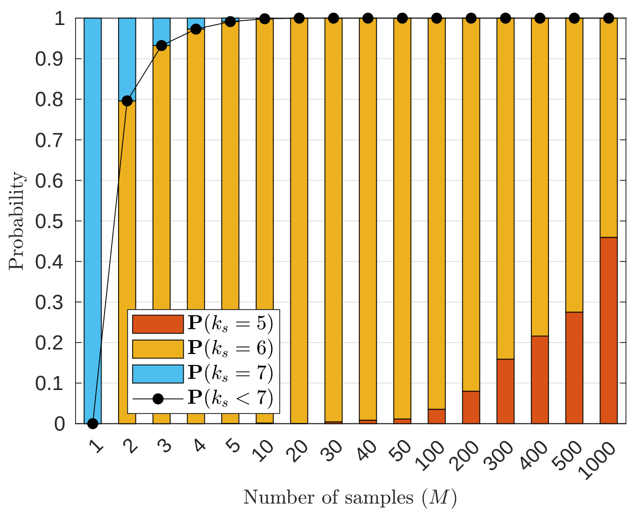
$$\frac{du_1}{dt} = A + u_1^2 u_2 - (B+1)u_1, \tag{11a}$$

$$\frac{du_2}{dt} = Bu_1 - u_1^2 u_2, \tag{11b}$$

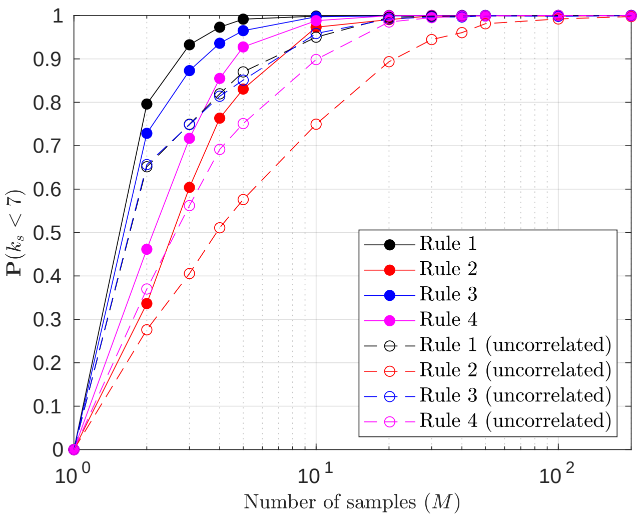
a pair of stiff nonlinear ODEs that model an auto-catalytic chemical reaction [28]. Using parameters $(A,B) = (1,3)$, trajectories of the system exhibit oscillatory behaviour in phase space, approaching a limit cycle (as $t \to \infty$) that contains the unstable fixed point $(1,3)^{\mathrm{T}}$. Now that $d > 1$, we use bivariate distributions to sample the initial values – meaning we can compare the effects of including or excluding the correlations between variables. System (11) is solved using initial values $\mathbf{u}(0) = (1, 3.07)^{\mathrm{T}}$ over time interval $t \in [0, 15.3]$ with $N = 25$, $\delta T = 15.3/25$, and $\delta t = 15.3/2500$ [29]. The numerical solution to (11) in phase space and convergence of the successive errors are reported in Appendix B. With these parameters and a tolerance of $\varepsilon = 10^{-6}$, $\mathcal{P}$ takes $k_d = 7$ iterations to stop and return a numerical solution.

The estimated distributions of k_s for sampling rule 1 are given in Figure 8a. Even though $\mathcal{P}$ takes just $k_d = 7$ iterations to stop, we observe that $\mathcal{P}_s$ can still reach the desired tolerance in 5 or 6 iterations – albeit requiring larger values of M. We believe this is due to the stiffness of the system and poor accuracy of the explicit $\mathcal{G}$ solver – results presented for the stiff ODE in Appendix A appear to confirm this. Using adaptive time-stepping methods could be a way to reduce the value of M needed to converge in fewer k_s [24]. The solid lines in Figure 8b show that, using sampling rules 1 or 3, $\mathcal{P}_s$ only requires $M \approx 10$ to beat parareal almost certainly, i.e. to guarantee that $\mathbb{P}(k_s < 7) \to 1$. Sampling rules 1 and 3 outperform 2 and 4 in this particular system. Note, however, the stark decrease in performance if instead uncorrelated samples are generated within $\mathcal{P}_s$ (dashed lines). This demonstrates the importance of accounting for the dependence between variables in nonlinear systems such as (11). Observe again, in Figure 9, how the mean $\mathcal{P}_s$ solutions attain equivalent, or better, accuracy than the $\mathcal{P}$ solutions, with standard deviations at most $\mathcal{O}(10^{-6})$.

Further results of $\mathcal{P}_s$ applied to (11) and an additional two-dimensional nonlinear system are presented in

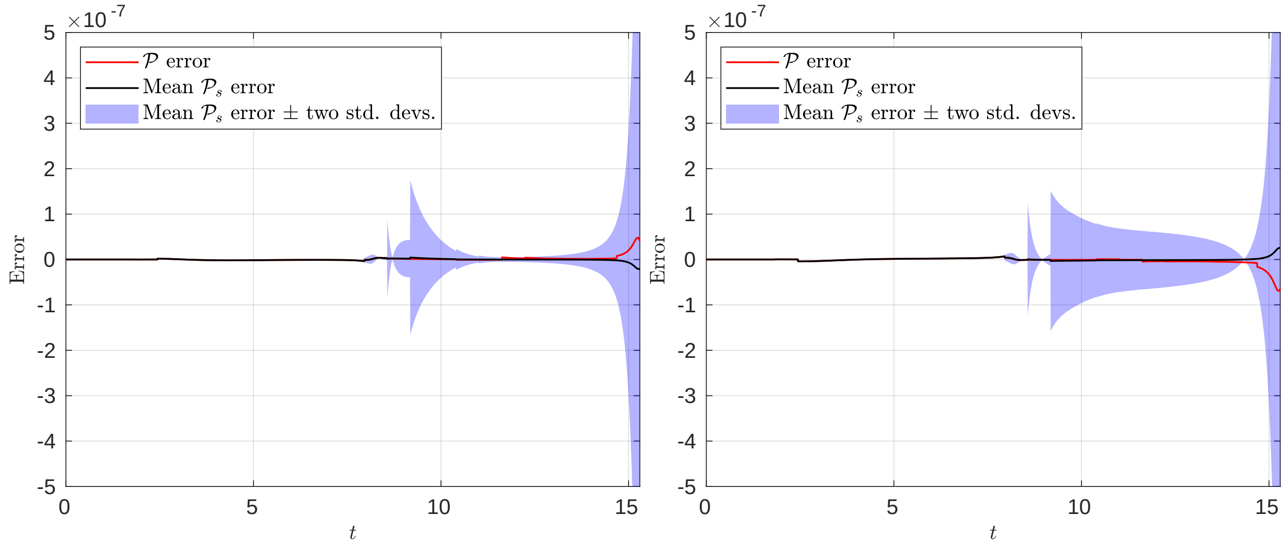


(a)



(b)

Figure 8: (a) Estimated discrete probabilities of k_s as a function of M for sampling rule 1. (b) Estimated probability that the convergence rate k_s is smaller than $k_d = 7$ as a function of M for the sampling rules with (solid lines) and without (dashed lines) correlations. Distributions were estimated by simulating 2000 independent realisations of $\mathcal{P}_s$ for each M.



(a) **(b)**

Figure 9: Errors of $\mathcal{P}$ (red) and mean $\mathcal{P}_s$ (black) solutions against the $\mathcal{F}$ solution. The mean error is obtained by running 2000 independent realisations of $\mathcal{P}_s$ with sampling rule 4 with $M = 200$ – the confidence interval representing the mean ± two standard deviations is shown in light blue. Panel (a) displays errors for the u_1 component of the solution whilst (b) displays the u_2 component.

Appendix B and Appendix C respectively. Observe again that for the less stiff system in Appendix C, we require less sampling to improve k_s compared to the Brusselator – further highlighting the demand for higher M in stiff systems to improve the convergence rate.

4.3 The Lorenz system

Finally, we consider the Lorenz system

$$\frac{du_1}{dt} = \gamma_1(u_2 - u_1), \tag{12a}$$

$$\frac{du_2}{dt} = \gamma_2 u_1 - u_1 u_3 - u_2, \tag{12b}$$

$$\frac{du_3}{dt} = u_1 u_2 - \gamma_3 u_3, \tag{12c}$$

a simplified model for weather prediction [30]. With the parameters $(\gamma_1, \gamma_2, \gamma_3) = (10, 28, 8/3)$, (12) exhibits chaotic behaviour where trajectories with initial values close to one another diverge exponentially. This will test the robustness of $\mathcal{P}_s$, as small numerical differences between initial values will mean that errors can grow rapidly as time progresses. We solve (12) using initial values $\mathbf{u}(0) = (-15, -15, 20)^{\mathrm{T}}$ over the interval $[0, 18]$, discretised using $N = 50$ sub-intervals and time steps $\delta T = 18/250$ and $\delta t = 18/18750$. With a tolerance of $\varepsilon = 10^{-8}$, $\mathcal{P}$ takes $k_d = 20$ iterations to converge.

Running $\mathcal{P}_s$ to compare the performance of the sampling rules, we see again in Figure 10a that taking correlated samples is much more efficient than not and that only $M \approx 10$ samples are required to beat parareal with probability one. For the chaotic trajectories generated by (12), sampling close to the predictor-corrector, rules 2 and 4, yields superior performance compared to rules 1 and 3 for small values of M. Figure 10b displays estimated distributions for varying M using sampling rule 2 – yielding a best convergence rate $k_s = 16$ for approximately 25% of runs with $M = 1000$. These results demonstrate the robustness of $\mathcal{P}_s$ and that the sampling

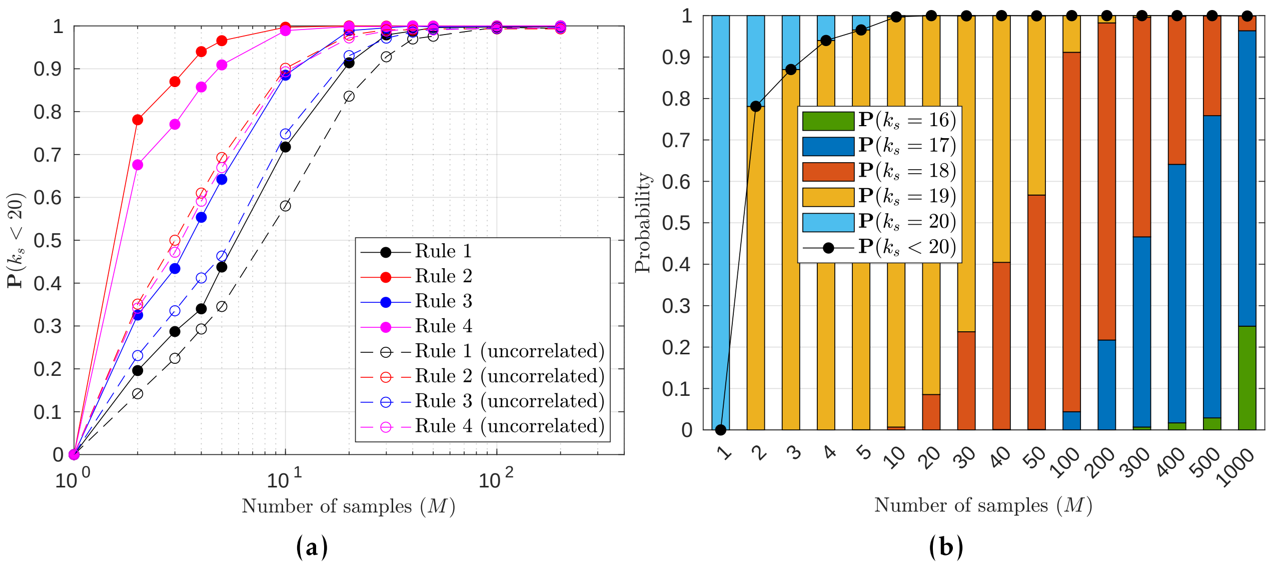


Figure 10: (a) Estimated probability that the convergence rate k_s is smaller than $k_d = 20$ as a function of M for the sampling rules with (solid lines) and without (dashed lines) correlations. Distributions were estimated by simulating 2000 independent realisations of $\mathcal{P}_s$ for each M. (b) Estimated discrete probabilities of k_s as a function of M for sampling rule 2.

and propagation process is not impeded by the exponential divergence of trajectories. Additional results of $\mathcal{P}_s$ applied to (12) are given in Appendix D.

5 Conclusions

In this paper, we have extended the parareal algorithm using probabilistic methods to develop a stochastic parareal algorithm for solving systems of ODEs in a time-parallel manner. Instead of passing deterministically calculated initial values into parareal's predictor-corrector, stochastic parareal selects more accurate values from a randomly sampled set, at each temporal sub-interval, to converge in fewer iterations. In Section 4, we compared performance against the deterministic parareal algorithm on several low-dimensional ODE systems of increasing complexity by calculating the estimated convergence rate distributions (upon multiple independent realisation of stochastic parareal) with increasing numbers of random samples M. By taking just $M \approx 10$ (correlated) samples, the estimated probability of converging sooner than parareal approached one in all test cases. Similarly, we observed numerical convergence toward the fine (exact) solution with accuracy of similar order to parareal and, in the spirit of probabilistic numerics [31, 32], obtained a measure of uncertainty over the ODE solution upon multiple realisations of the algorithm.

The probability that stochastic parareal converges faster than standard parareal depends on a number of factors: the complexity of the problem being solved, the number of time sub-intervals (N), the accuracy of the coarse integrator ($\mathcal{G}$), the number of random samples (M), and the type of sampling rule in use. Sampling rules 1 and 3 (sampling close to the fine solutions) outperformed rules 2 and 4 (sampling close to the predictor-corrector solutions) for the ODE systems in Section 4.1, Section 4.2, and SM1. The reverse was true, however, for the systems in Section 4.3 and SM3, making it difficult to determine an optimal rule for a general ODE system. To overcome having to choose a particular sampling rule, one could linearly combine different rules or even sample from multiple rules simultaneously. We would suggest sampling from probability distributions with infinite support, i.e. the Gaussians (rules 1 and 2), so that samples can be taken anywhere in $\mathbb{R}^d$ with non-zero probability. Having finite support may have created difficulty for the uniform marginal t-copulas (rules 3 and 4) because samples could only be taken in a finite hyperrectangle in $\mathbb{R}^d$ – problematic if the exact solution were to lay outside of this space.

When solving stiff ODEs (see Section 4.2 and SM1), results indicated that stochastic parareal demanded increasingly high sampling to converge sooner than parareal than for non-stiff systems. For example, we observe that when taking $M = 100$ samples in the non-stiff scalar ODE in Figure 6, the expected convergence rate decreases from 25 to 7 whereas for the stiff scalar ODE in SM1, the rate only drops from 8 to 6. A similar observation can be made in the two-dimensional test cases in Section 4.2 (stiff) and SM3 (non-stiff). These results exemplify the role that system complexity, e.g. stiffness or chaos, plays in the performance of both algorithms. In SM1, stochastic parareal was also shown to perform more efficiently for problems that deterministic parareal itself struggles with, i.e. cases in which the accuracy of the coarse integrator $\mathcal{G}$ is poor. In SM3 it was also observed that, for low sample numbers ($M = 2$), stochastic parareal actually converged in one more iteration than parareal in less than 2.5% of cases. This suggests there may be minimum number of samples required to beat parareal in some situations – something to be investigated with further experimentation.

In summary, we have demonstrated that probabilistic methods and additional processors can, for low-dimensional ODEs, be used to increase the parallel scalability of existing time-parallel algorithms. Next we need to investigate possible improvements and generalisations. For example, determining whether the algorithm scales for larger systems of equations – essential if it is to be used for solving PDE problems. Moreover, whether we can design adaptive sampling rules that do not need to be specified *a priori* to simulation. Finally, we would aim to make use of the whole ensemble of fine propagated trajectories rather than using only one – avoiding the waste of valuable information about the solution at the coarse and fine resolutions. An approach in this direction has recently been proposed [33], making use of ideas from the field of probabilistic numerics to adopt a more Bayesian approach to this problem.

Acknowledgements

This work has partly been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. KP is funded by the Engineering and Physical Sciences Research Council through the MathSys II CDT (grant EP/S022244/1) as well as the Culham Centre for Fusion Energy. For the purpose of open access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising. The authors would also like to acknowledge the University of Warwick Scientific Computing Research Technology Platform for assistance in the research described in this paper and the anonymous reviewers for their insightful comments on improving this manuscript.

References

[1] A. Toselli and O. Widlund. *Domain Decomposition Methods — Algorithms and Theory*. Springer New York, 2005.

[2] D. Samaddar, D. P. Coster, X. Bonnin, L. A. Berry, W. R. Elwasif, and D. B. Batchelor. Application of the parareal algorithm to simulations of ELMs in ITER plasma. *Comput. Phys. Commun.*, 235:246–257, 2019.

[3] K. Burrage. *Parallel and sequential methods for ordinary differential equations*. The Clarendon Press Oxford University Press, Burlington, MA, USA, 1995.

[4] M. J. Gander. 50 Years of Time Parallel Time Integration. In *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113. Springer International Publishing, 2015.

[5] B. W. Ong and J. B. Schroder. Applications of time parallelization. *Comput. Vis. Sci.*, 23:1–11, 2020.

[6] A. Bellen and M. Zennaro. Parallel algorithms for initial-value problems for difference and differential equations. *J. Comput. Appl. Math.*, 25:341–350, 1989.

[7] P. Chartier and B. Philippe. A parallel shooting technique for solving dissipative ODE's. *Computing*, 51:209–236, 1993.

[8] J. L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps pararéel. *C. R. Math. Acad. Sci. Paris - Series I: Math.*, 332:661–668, 2001.

[9] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Communications of the ACM*, 7:731–733, 1964.

[10] P. Saha, J. Stadel, and S. Tremaine. A Parallel Integration Method for Solar System Dynamics. *The Astronomical Journal*, 114:409, 1997.

[11] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah. Parallel-in-time molecular-dynamics simulations. *Phys. Rev. E*, 66:4, 2002.

[12] P. F. Fischer, F. Hecht, and Y. Maday. A parareal in time semi-implicit approximation of the navier-stokes equations. *Lect. Notes Comput. Sci. Eng.*, 40:433–440, 2005.

[13] I. Garrido, M. S. Espedal, and G. E. Fladmark. A convergent algorithm for time parallelization applied to reservoir simulation. *Lect. Notes Comput. Sci. Eng.*, 40:469–476, 2005.

[14] J. M.F. Trindade and J. C.F. Pereira. Parallel-in-time simulation of two-dimensional, unsteady, incompressible laminar flows. *Numerical Heat Transfer, Part B: Fundamentals*, 50:25–40, 2006.

[15] S. Engblom. Parallel in time simulation of multiscale stochastic chemical kinetics. *Multiscale Model. Simul.*, 8:46–68, 2009.

[16] F. Legoll, T. Lelièvre, K. Myerscough, and G. Samaey. Parareal computation of stochastic differential equations with time-scale separation: a numerical convergence study. *Comput. Vis. Sci.*, 23:1–18, 2020.

[17] D. Samaddar, D. E. Newman, and R. Sánchez. Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm. *J. Comput. Phys.*, 229:6558–6573, 2010.

[18] G. Bal. On the convergence and the stability of the parareal algorithm to solve partial differential equations. *Lect. Notes Comput. Sci. Eng.*, 40:425–432, 2005.

[19] G. Bal and Y. Maday. A "Parareal" Time Discretization for Non-Linear PDE's with Application to the Pricing of an American Put. In *Recent Developments in Domain Decomposition Methods*, pages 189–202. Springer, Berlin, Heidelberg, 2002.

[20] M. J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM J. Sci. Comput.*, 29:556–578, 2007.

[21] Y. Maday and G. Turinici. A parareal in time procedure for the control of partial differential equations. *C. R. Math. Acad. Sci. Paris*, 335:387–392, 2002.

[22] Y. Maday and G. Turinici. The parareal in time iterative solver: A further direction to parallel implementation. *Lect. Notes Comput. Sci. Eng.*, 40:441–448, 2005.

[23] W. R. Elwasif, S. S. Foley, D. E. Bernholdt, L. A. Berry, D. Samaddar, D. E. Newman, and R. Sanchez. A dependency-driven formulation of parareal: Parallel-in-time solution of PDEs as a many-task application. In *MTAGS'11 - Proceedings of the 2011 ACM International Workshop on Many Task Computing on Grids and Supercomputers, Co-located with SC'11*, pages 15–24, New York, New York, USA, 2011. ACM Press.

[24] Y. Maday and O. Mula. An adaptive parareal algorithm. *J. Comput. Appl. Math.*, 377:112915, 2020.

[25] M. J. Gander and E. Hairer. Nonlinear convergence analysis for the parareal algorithm. *Lect. Notes Comput. Sci. Eng.*, 60:45–56, 2008.

[26] R. B. Nelsen. *An Introduction to Copulas*. Springer New York, 2006.

[27] A Sklar. Fonctions de Répartition à n Dimensions et Leurs Marges. *Publications de L'Institut de Statistique de L'Université de Paris*, 8:229–231, 1959.

[28] R. Lefever and G. Nicolis. Chemical instabilities and sustained oscillations. *J. Theoret. Biol.*, 30:267–284, 1971.

[29] L.N. Trefethen, A. Birkisson, and T. Driscoll. *Exploring ODEs*. SIAM, Philadelphia, USA, 2017.

[30] E. N. Lorenz. Deterministic Nonperiodic Flow. *J. Atmos. Sci.*, 20:130–141, 1963.

[31] P. Hennig, M. A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proc. R. Soc. Math. Phys. Eng. Sci.*, 471:20150142, 2015.

[32] C. J. Oates and T. J. Sullivan. A modern retrospective on probabilistic numerics. *Stat. Comput.*, 29:1335–1351, 2019.

[33] K. Pentland, M. Tamborrino, T. J. Sullivan, J. Buchanan, and L. C. Appel. GParareal: A time-parallel ODE solver using Gaussian process emulation, 2022. arXiv:2201.13418.

[34] M. W. Hirsch, S. Smale, and R. L. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press, 3rd edition, 2013.

Appendix

In the following sections we illustrate some additional results using stochastic parareal.

A Scalar Bernoulli equation

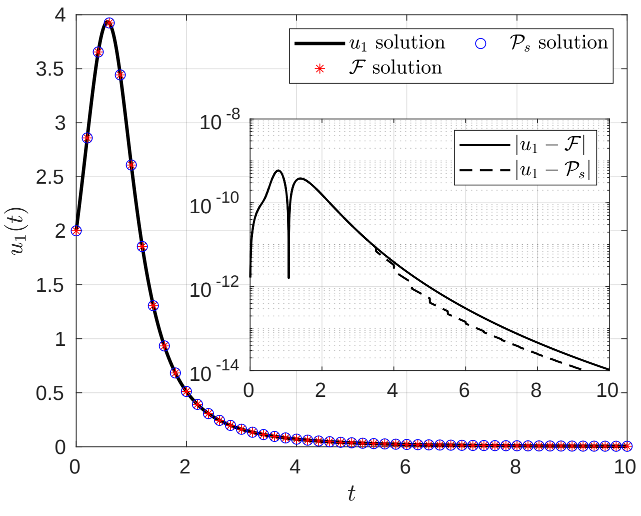
Consider the nonlinear non-autonomous Bernoulli equation

$$\frac{du_1}{dt} = \frac{2}{1+t}u_1 - t^2 u_1^2, \tag{13}$$

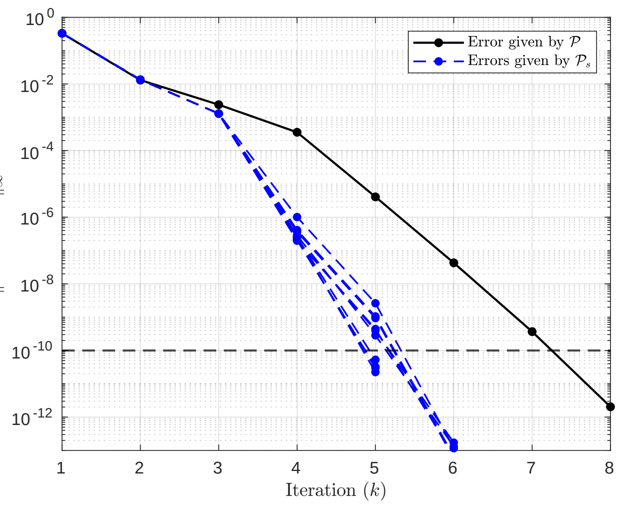
with initial value $u_1(0) = 2$ on $t \in [0, 10]$. We discretise using $N = 20$ sub-intervals and time steps $\delta T = 10/20$ and $\delta t = 10/2000$. Equation (13) permits the analytical solution $u_1(t) = (1+t)^2/(t^5/5 + t^4/2 + t^3/3 + 1/2)$, tending to zero as $t \to \infty$. Observe the spacing between equidistant time steps of the true $\mathcal{F}$ solution and the $\mathcal{P}_s$ solution to (13) in Figure 11a, highlighting the stiffness of the solution at early times. Given the stopping tolerance $\varepsilon = 10^{-10}$, observe in Figure 11b how $\mathcal{P}$ converges in $k_d = 8$ iterations deterministically whilst $\mathcal{P}_s$ converges in $k_s \in \{5, 6\}$ iterations for each of the ten independent realisations shown with $M = 1000$.

Figure 12a shows the estimated distributions of k_s using sampling rule 1. As expected, if $M = 1$, convergence is deterministic (i.e. $\mathcal{P}_s = \mathcal{P}$) and hence $\mathbb{P}(k_s = 8) = 1$. As M increases however, $\mathbb{P}(k_s = 8)$ decreases rapidly to zero whilst $\mathbb{P}(k_s < 8)$ increases from zero to one with just $M \approx 5$ samples. This demonstrates that $\mathcal{P}_s$ requires very few samples to begin converging in fewer iterations than $\mathcal{P}$, and that k_s assumes low values with increasing probabilities for increasing M. The stiffness of (13) appears to demand much larger values of M to continually reduce k_s when compared to the non-stiff scalar example in subsection 4.1.

In Figure 12b we report the expected values of k_s for each sampling rule. This shows that for low values of M, using either a Gaussian or t-copula sampling rule makes little difference to performance. More interestingly, rules 1 and 3, centred around the fine solutions, outperform rules 2 and 4, which are centred around the predictor-corrector. Further testing revealed that varying the fine time steps within $\mathcal{P}_s$ had little impact on these



(a)



(b)

Figure 11: (a) Analytical solution u_1 of (13) plotted against the serial $\mathcal{F}$ solution and a single realisation of $\mathcal{P}_s$. Note that only a subset of the fine times steps of the numerical solutions are shown for clarity. Inset: displays the numerical errors of $\mathcal{F}$ and $\mathcal{P}_s$ compared to u_1. (b) Errors at successive iterations of $\mathcal{P}$ (black lines) and ten independent realisations of $\mathcal{P}_s$ (blue lines). The horizontal dashed black line represents the tolerance $\varepsilon = 10^{-10}$. Both panels use $\mathcal{P}_s$ with sampling rule 3 and (a) $M = 30$ and (b) $M = 1000$.

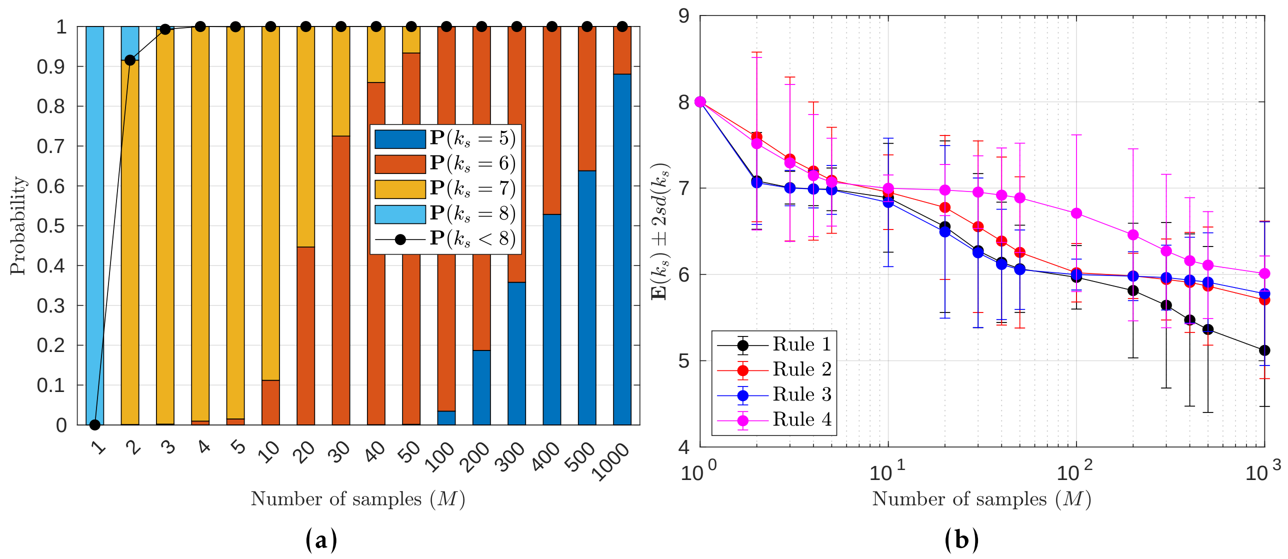


Figure 12: (a) Estimated discrete distributions of k_s as a function of M for sampling rule 1. (b) Estimated expectation of k_s as a function of M, calculated using estimated distributions of k_s for each sampling rule with error bars representing $\pm$ two standard deviations $sd(k_s)$. Distributions were estimated by simulating 2000 independent realisations of $\mathcal{P}_s$ for each M.

probabilities. On the contrary, Figure 13 shows how increasing the number of coarse steps in the interval $[0, 10]$ from 20 to 60 drastically decreases the probabilities of $\mathcal{P}_s$ converging sooner than $\mathcal{P}$. Increasing the number of coarse steps increases the accuracy of the $\mathcal{G}$ solver, hence $\mathcal{P}$ reaches the stopping tolerance in fewer iterations k_d. This result suggests that whilst $\mathcal{P}_s$ can still converge faster than $\mathcal{P}$ by using more samples, it works more efficiently for particular problems where k_d is relatively large (i.e when $\mathcal{P}$ obtains low rates of parallel speed up).

Finally, we calculated errors between the mean $\mathcal{P}_s$ solution with respect to the $\mathcal{F}$ solution (Figure 14a) and the analytical solution u_1 (Figure 14b). In both cases, we observe how the mean solution attains comparable accuracy to $\mathcal{P}$ and both the fine and analytical solutions.

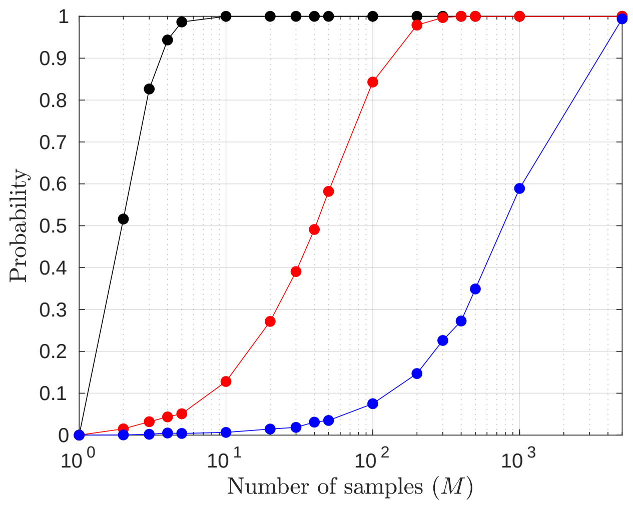


Figure 13: Estimated probabilities that the convergence rate k_s is smaller than k_d as a function of M using sampling rule 3. Each curve shows $\mathbb{P}(k_s < 8)$ (black), $\mathbb{P}(k_s < 5)$ (red), and $\mathbb{P}(k_s < 4)$ (blue) using coarse steps $\delta T = 10/20, 10/40, 10/60$ respectively - noting that $\mathcal{P}$ converges in $k_d = 8, 5, 4$ iterations for these coarse steps respectively. As before, 2000 independent realisations of $\mathcal{P}_s$ were run for each M.

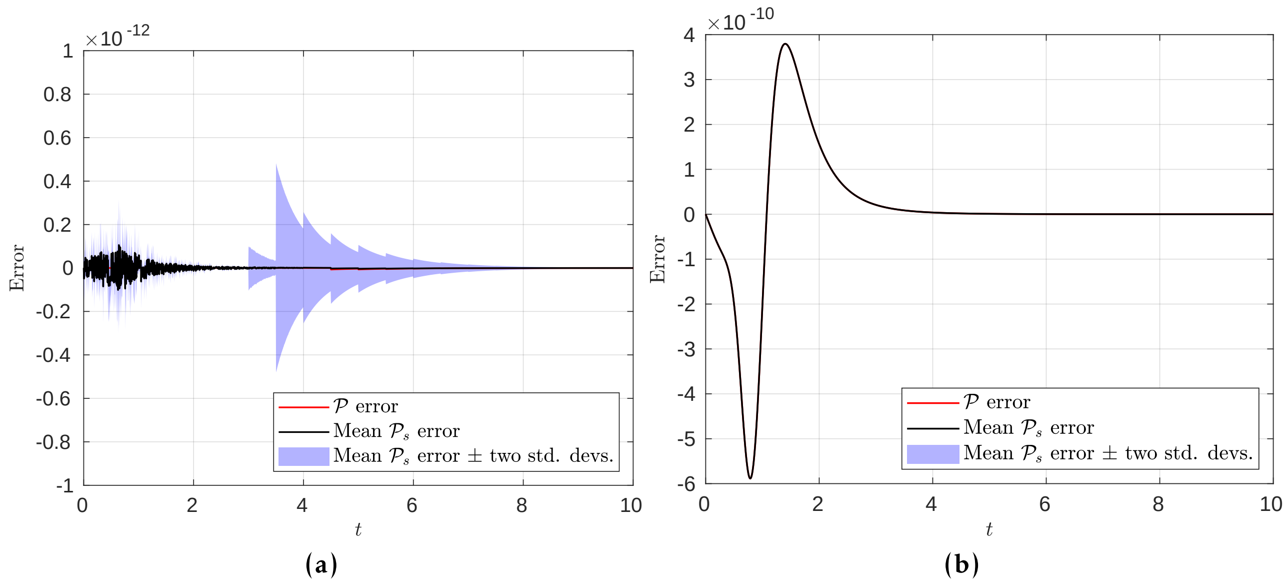


Figure 14: Errors of $\mathcal{P}$ (red) and mean $\mathcal{P}_s$ (black) solutions against (a) the $\mathcal{F}$ solution and (b) the analytical solution u_1. The mean error is obtained by running 2000 independent realisations of $\mathcal{P}_s$ with sampling rule 1 with $M = 10$ – the confidence interval representing the mean ± two standard deviations is shown in light blue.

B The Brusselator system

The additional results here complement subsection 4.2. The numerical solutions to system (4.2) in the phase plane are given in 15a alongside the errors at successive iterations of five runs of $\mathcal{P}_s$ in 15b. In Figure 16 we report the expected value of k_s as a function of M for each of the sampling rules. This result suggests that larger values of M are required to reduce $\mathbb{E}(k_s)$ even further for this stiff system.

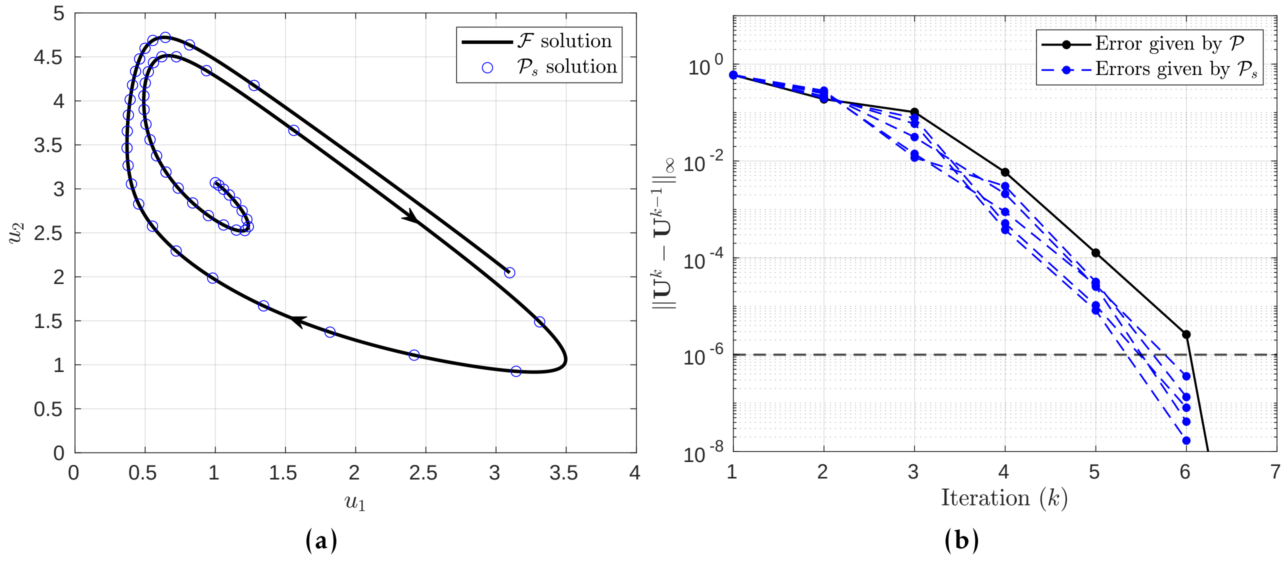


Figure 15: (a) Numerical solution of system (4.2) in the phase plane using $\mathcal{F}$ and $\mathcal{P}_s$. Note again that only a subset of the fine times steps of the (mean) $\mathcal{P}_s$ solution are shown for clarity. (b) Errors at successive iterations of $\mathcal{P}$ (black lines) and five independent realisations of $\mathcal{P}_s$ (blue lines). Horizontal dashed black line represents the tolerance $\varepsilon = 10^{-6}$. Both panels run $\mathcal{P}_s$ with sampling rule 1 and $M = 10$.

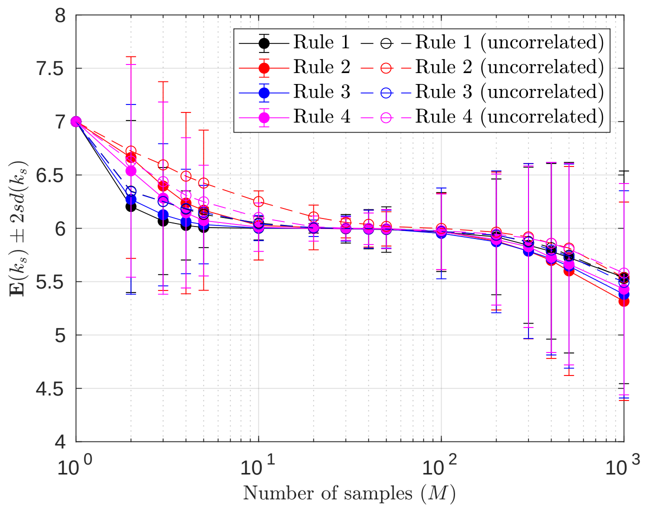


Figure 16: Estimated expectation of k_s as a function of M, calculated using estimated distributions of k_s for each sampling rule with error bars representing ± two standard deviations $sd(k_s)$ shown for correlated (solid lines) sampling rules. Uncorrelated sampling rules are shown with dashed lines without error bars. Distributions are estimated by simulating 2000 independent realisations of $\mathcal{P}_s$ for each M.

C 'Square limit cycle' system

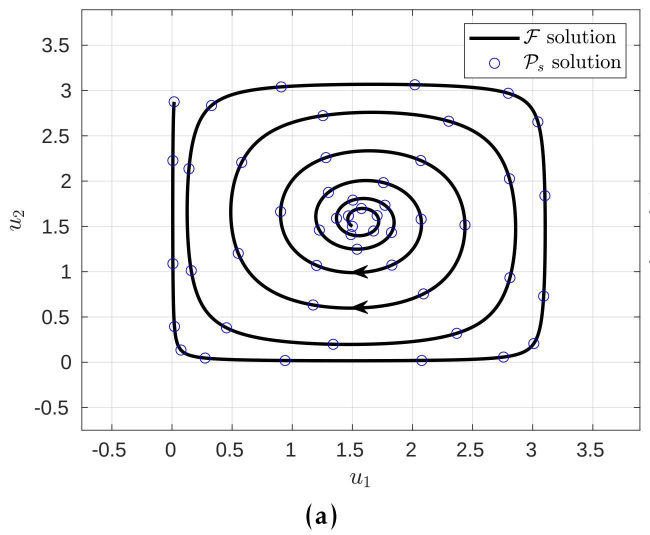
Consider the system

$$\frac{du_1}{dt} = -\sin(u_1)\left(\frac{\cos(u_1)}{10} + \cos(u_2)\right), \tag{14a}$$

$$\frac{du_2}{dt} = -\sin(u_2)\left(\frac{\cos(u_2)}{10} - \cos(u_1)\right), \tag{14b}$$

whose solutions, for initial values within the box $[0,\pi]\times[0,\pi]$, converge toward a square-shaped limit cycle on the edges of the box (see Figure 17a) [34]. The system is solved on $t \in [0,60]$, starting at $\mathbf{u}(0) = (3/2, 3/2)^{\mathrm{T}}$, using $N = 30$ sub-intervals and time steps $\delta T = 60/30$ and $\delta t = 60/3000$. As shown in Figure 17b, $\mathcal{P}$ takes $k_d = 20$ iterations to converge with tolerance $\varepsilon = 10^{-8}$ whilst the ten realisations of $\mathcal{P}_s$ shown take between $17 \leq k_s \leq 19$.

Contrary to the previous two-dimensional test problem, Figure 18a shows that sampling close to the predictor-corrector values (rules 2 and 4) yield lower expected values of k_s. In this case, the bivariate Gaussian outperforms the t-copula, however the reverse is true for rules 1 and 3. Results using uncorrelated samples have been shown to generate inferior performance hence are not shown here. The detailed distributions of k_s in Figure 18b, using sampling rule 2, show a best performance of $k_s = 14$ with 100 samples and $\mathbb{P}(k_s < 20) = 1$ for $M \approx 10$. They do, however, reveal that in a limited number of cases, using two samples, $\mathcal{P}_s$ can in fact take more than $k_d = 20$ iterations to converge ($k_s = 21$ in this small fraction of realisations). This suggests that there may be a minimum number of samples required to beat the convergence rate of parareal for some systems of equations – something to be investigated in future work.



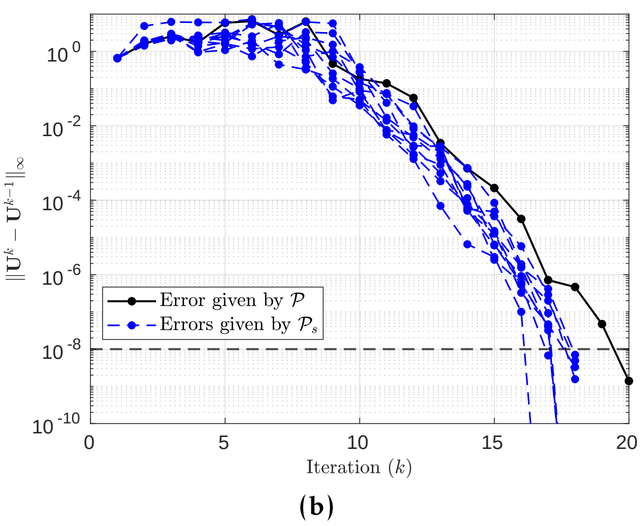
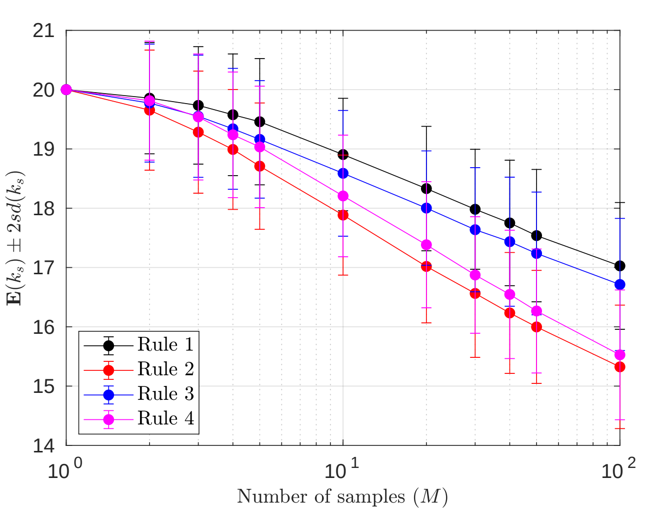


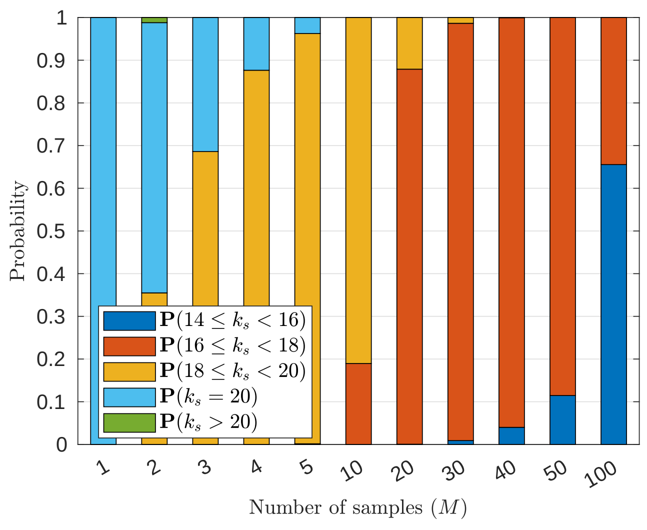
Figure 17: (a) Numerical solution of (14) over $[0,60]$ using $\mathcal{F}$ serially and $\mathcal{P}_s$. Note again that only a subset of the fine times steps of the (mean) $\mathcal{P}_s$ solution are shown for clarity. (b) Errors at successive iterations of $\mathcal{P}$ and ten independent realisations of $\mathcal{P}_s$. Dashed black line represents the tolerance $\varepsilon = 10^{-8}$. Note that both panels use $\mathcal{P}_s$ with sampling rule 2 and $M = 20$.

D The Lorenz system

The additional results here complement subsection 4.3. Figure 19 displays $\mathbb{E}(k_s)$ against M, indicating that for the Lorenz system, generating correlated samples close to the predictor-corrector solutions (sampling rules 2 and 4) yield the lowest expected values of k_s. In Figure 20, we plot the absolute errors between the mean $\mathcal{P}_s$ solution and the fine solver. Notice that accuracy is maintained with respect to the parareal solution in this chaotic system, even as the errors grow with increasing time.



(a)



(b)

Figure 18: (a) Estimated expectation of k_s as a function of M, calculated using estimated distributions of k_s for each sampling rule with error bars representing ± two standard deviations $sd(k_s)$. (b) Estimated discrete distributions of k_s as a function of M for sampling rule 2. Distributions were calculated by simulating 2000 independent realisations of $\mathcal{P}_s$ for each M.

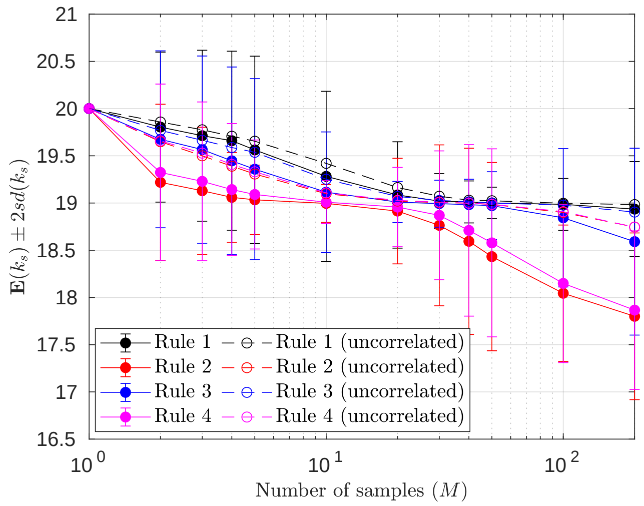


Figure 19: Estimated expectation of k_s as a function of M, calculated using estimated distributions of k_s for each sampling rule with error bars representing ± two standard deviations $sd(k_s)$ shown for correlated (solid lines) sampling rules. Uncorrelated sampling rules are shown with dashed lines without error bars. Distributions are estimated by simulating 2000 independent realisations of $\mathcal{P}_s$ for each M.

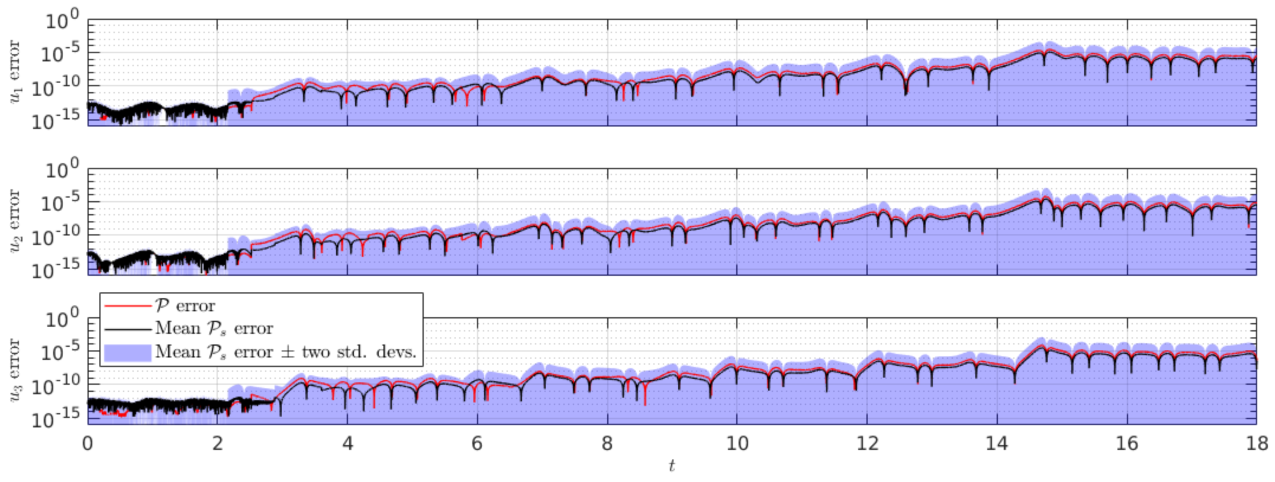


Figure 20: Absolute errors of $\mathcal{P}$ (red) and mean $\mathcal{P}_s$ (black) solutions against the $\mathcal{F}$ solution in each component: u_1 (top), u_2 (middle), and u_3 (bottom). The mean error is obtained by running 2000 independent realisations of $\mathcal{P}_s$ with sampling rule 2 with $M = 500$ – the confidence interval representing the mean $\pm$ two standard deviations is shown in light blue.