PARAREAL: AN APPLICATION OF PROBABILISTIC METHODS TO TIME PARALLELISATION

Haoran Ni, Jacques Bara, Jimmy McKendrick, Kamran Pentland, Yijie Zhou

Supervisors: Massimiliano Tamborrino (Warwick), Debasmita Samaddar, Lynton Appel (UKAEA)

Mathematics for Real-World Systems CDT, University of Warwick

Parallelism and Parareal

Parallel computing is increasingly relied upon for computationally intensive tasks in fields ranging from molecular chemistry [1] to turbulent plasma dynamics [4]. Current techniques rely heavily on spatial parallelisation methods that often struggle with serial time integration. With nuclear fusion-based plasma simulations in mind, some exciting, alternative approaches that offer significant gains in efficiency and speed up are known as time parallelisation algorithms.

Our focus, the Parareal algorithm ($\mathcal{P}$), is a multigrid, multiple-shooting method that solves a system of differential equations [3]. It utilises both coarse ($\mathcal{G}$) and fine ($\mathcal{F}$) numerical solvers successively across N time sub-intervals (see below). The aims of this project are to:

- Adapt the current predictor-corrector update rule (1) used in parareal by using probabilistic techniques instead.
- Analyse whether this leads to increased convergence, decreased computation time and improved solution accuracy towards a fine solution.

The Parareal Algorithm

The algorithm requires N compute processors, one per time sub-interval defined (i.e. $N = 5$ in Fig. 1), and solvers $\mathcal{G}$ and $\mathcal{F}$ use time steps $\Delta T \gg \Delta t$ respectively.

1. Set a counter $k = 0$ and define $\mathbf{U}_n^k$ as the solution of the system of equations at n^{th} time step and k^{th} iteration.
2. Calculate initial guess $\mathbf{U}_n^0$ at the start of each sub-interval T_n using $\mathcal{G}$ serially such that $\mathbf{U}_n^0 = \mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^0)\ \forall n \leq N$.
3. For $k = 1$ to N:
   (a) Running $\mathcal{F}$ in parallel, propagate the solution on each sub-interval using initial values previously calculated: $\mathcal{F}(T_{n-1}, T_n, \mathbf{U}_{n-1}^{k-1})\ \forall n \leq N$.
   (b) Iteratively, at each time step n, calculate $\mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^k)$, then predict (and correct) the solution using the update rule (1).

$$\mathbf{U}_n^k = \mathcal{F}(T_{n-1}, T_n, \mathbf{U}_{n-1}^{k-1}) + \mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^k) - \mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^{k-1}) \tag{1}$$

   (c) If error tolerance (2) is met at all time steps, break the loop and return $\mathbf{U}^k$.

$$\|\mathbf{U}_n^k - \mathbf{U}_n^{k-1}\|_\infty < \epsilon \tag{2}$$

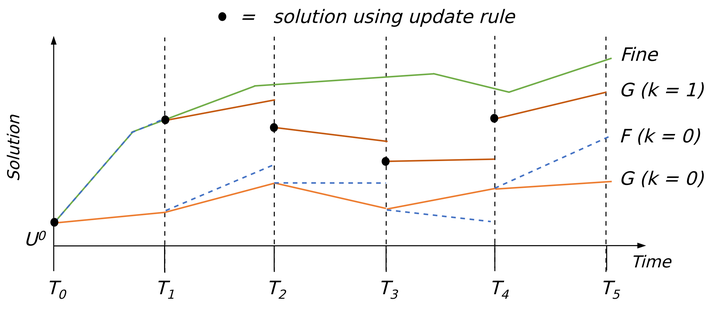

Fig. 1: Schematic of the parareal algorithm in action. After initialisation, $\mathcal{G}$ calculates the initial guess, $\mathcal{G}(k = 0)$, which $\mathcal{F}$ uses to integrate in parallel and obtain $\mathcal{F}(k = 0)$. Following this, $\mathcal{G}(k = 1)$ is calculated sequentially, then rule (1) is used to update the solutions. The process repeats, eventually converging toward the fine solution.

The Brusselator

To test the implementation, convergence and accuracy of $\mathcal{P}$, we examined the Brusselator (3), a stiff two-dimesional ODE system that models coupled chemical reactions. Using $N = 28$ sub-intervals, 140 coarse step, 7,000 fine steps and parameter values $A = 1$ and $B = 3$ from [2] (to yield a limit cycle) we obtain the results in Fig. 2.

$$\frac{dx}{dt} = A + x^2y - (B+1)x, \quad \frac{dy}{dt} = Bx - x^2y \quad \text{with} \quad (x, y)(0) = (0, 1). \tag{3}$$

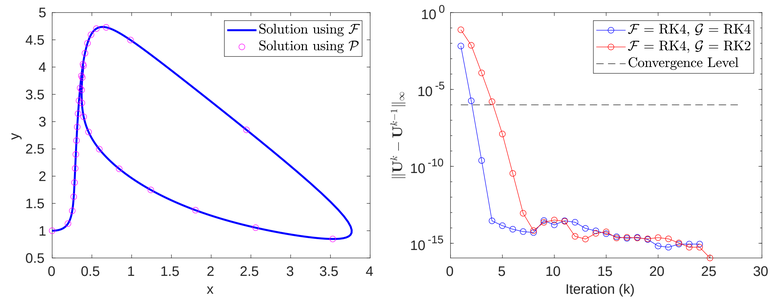

Fig. 2: (a) Solutions to the Brusselator running both $\mathcal{P}$, after three iterations, and $\mathcal{F}$, where $\mathcal{F}$ and $\mathcal{G}$ are fourth-order Runge-Kutta solvers. (b) Convergence of $\mathcal{P}$, using error tolerance $\epsilon = 10^{-6}$ in (2), and two different coarse Runge-Kutta solvers $\mathcal{G}$.

The Lorenz System

Next we investigate the Lorenz system (4) in its chaotic state ($\sigma = 10$, $\rho = 28$, $\beta = 8/3$) as its trajectories diverge drastically under the smallest of perturbations. This becomes more significant when we alter parareal's update rule (see next section). Implementing $\mathcal{P}$ again, we obtain the results in Fig. 3.

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z \quad \text{with} (x, y, z)(0) = (20, 5, -5). \tag{4}$$

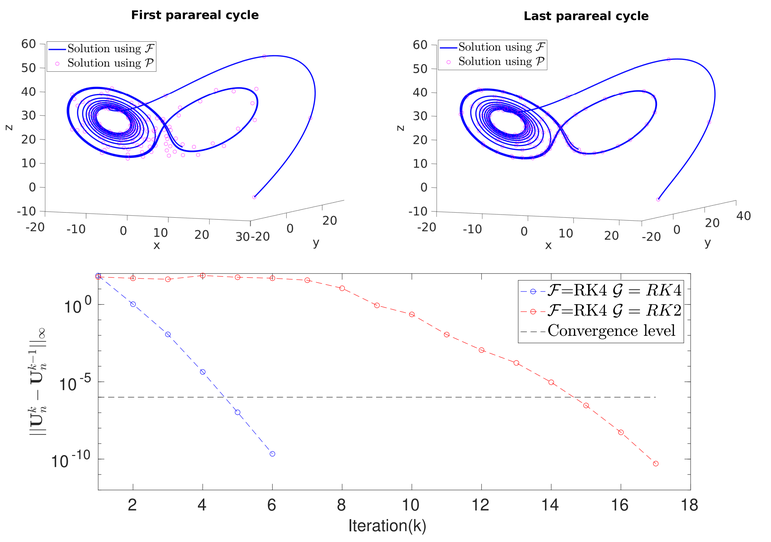

Fig. 3: (Top) Solutions to the Lorenz system after the first and final (fifth) iterations using fourth-order Runge-Kutta solvers. (Bottom) Convergence of $\mathcal{P}$ using different coarse solvers as before.

Probabilistic Update Rules

In order to improve the convergence rate of parareal, we propose modifying the update rule (1) using probabilistic methods. Two new 'sampling rules' we aim to test, alongside others, use Gaussian samples to update the solution:

$$\mathbf{U}_n^k = \mathcal{F}(T_{n-1}, T_n, \mathbf{U}_{n-1}^{k-1}) + \Big[\mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^k) - \mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^{k-1})\Big]\xi, \tag{5}$$

$$\mathbf{U}_n^k = \mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^k) + \Big[\mathcal{F}(T_{n-1}, T_n, \mathbf{U}_{n-1}^{k-1}) - \mathcal{G}(T_{n-1}, T_n, \mathbf{U}_{n-1}^{k-1})\Big]\xi. \tag{6}$$

The intuition behind this is that during an update cycle, a stochastic update rule could (on average) converge more quickly to a solution than a purely deterministic one. The updated solution $\mathbf{U}_n^k$ is set equal to the previous solution (either from the last fine solve or the current coarse solve) plus a white noise term with standard deviation equal to the difference in solutions (between consecutive $\mathcal{G}$ or between $\mathcal{F}$ and $\mathcal{G}$).

In our implementation we sample m numbers from $\mathcal{N}(0, 1)$ and select the smallest absolute value, ξ, for the update rule. For an optimal value of m we expect that the stochastic parareal algorithm will converge after k_s iterations, with $k_s < k_d$, where k_d is the number of iterations taken by the deterministic parareal. Further analysis needs to be carried out to determine if optimal values of m, for different sampling rules, exist and whether ensemble solutions from the stochastic parareal are accurate compared to the fine solver $\mathcal{F}$.

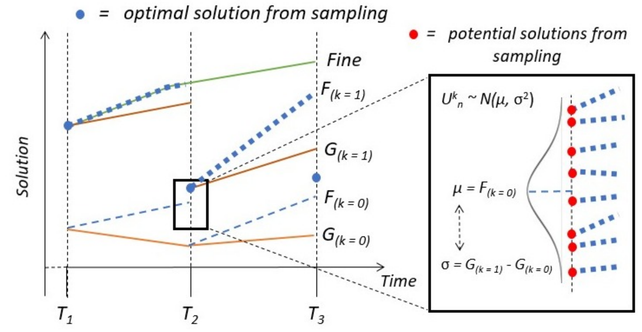

Fig. 4: Schematic of parareal generating m stochastic solutions, based upon sampling rule (5), at time step T_2.

Note: A public-access repository implementing parareal in MATLAB has been created at: https://github.com/kpentland/Parareal-MATLAB.

Acknowledgements and References

The authors would like to acknowledge the aforementioned supervisors, funding from EPSRC and the University of Warwick Research Technology Platform (SCRTP) for the help and assistance in creating this poster.

[1] L. Baffico et al. "Parallel-in-time molecular-dynamics simulations". In: *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* 66.5 (Nov. 2002), p. 4. ISSN: 1063651X. DOI: 10.1103/PhysRevE.66.057701.

[2] Martin J. Gander and Ernst Hairer. "Nonlinear convergence analysis for the parareal algorithm". In: *Lecture Notes in Computational Science and Engineering*. Vol. 60. Springer, Berlin, Heidelberg, 2008, pp. 45–56. ISBN: 9783540751984. DOI: 10.1007/978-3-540-75199-1_4.

[3] Jacques Louis Lions, Yvon Maday, and Gabriel Turinici. "Résolution d'EDP par un schéma en temps «pararéel »". In: *Comptes Rendus de l'Academie des Sciences - Series I: Mathematics* 332.7 (Apr. 2001), pp. 661–668. ISSN: 07644442. DOI: 10.1016/S0764-4442(00)01793-6.

[4] D. Samaddar, D. E. Newman, and R. Sánchez. "Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm". In: *Journal of Computational Physics* 229.18 (Sept. 2010), pp. 6558–6573. ISSN: 10902716. DOI: 10.1016/j.jcp.2010.05.012.