HAL
open science

An Adaptive Parareal Algorithm

Yvon Maday, Olga Mula

▶ **To cite this version:**

Yvon Maday, Olga Mula. An Adaptive Parareal Algorithm. Journal of Computational and Applied Mathematics, 2020, 10.1016/j.cam.2020.112915 . hal-01781257v2

HAL Id: hal-01781257

https://hal.science/hal-01781257v2

Submitted on 26 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An Adaptive Parareal Algorithm

Y. Maday, O. Mula

Abstract

In this paper, we consider the problem of accelerating the numerical simulation of time dependent problems by time domain decomposition. The available algorithms enabling such decompositions present severe efficiency limitations and are an obstacle for the solution of large scale and high dimensional problems. Our main contribution is the improvement of the parallel efficiency of the parareal in time method. The parareal method is based on combining predictions made by a numerically inexpensive solver (with coarse physics and/or coarse resolution) with corrections coming from an expensive solver (with high-fidelity physics and high resolution). At convergence, the parareal algorithm provides a solution that has the fine solver's high-fidelity physics and high resolution In the classical version of parareal, the fine solver has a fixed high accuracy which is the major obstacle to achieve a competitive parallel efficiency. In this paper, we develop an adaptive variant of the algorithm that overcomes this obstacle. Thanks to this, the only remaining factor impacting performance becomes the cost of the coarse solver. We show both theoretically and in a numerical example that the parallel efficiency becomes very competitive when the cost of the coarse solver is small.

1 Introduction

Solving complex models with high accuracy and within a reasonable computing time has motivated the search for numerical schemes that exploit efficiently parallel computing architectures. In this paper, the model consists of a Partial Differential Equation (PDE) set on a domain $\mathcal{D}$. In this context, one of the main ideas to parallelize a simulation is to break the problem into subproblems defined over subdomains of a partition of $\mathcal{D}$. The domain can potentially have high dimensionality and be composed of different variables like space, time, velocity or even more specific variables for some problems. While there exist algorithms with very good scalability properties for the decomposition of the spatial variable in elliptic and saddle-point problems (see [29] or [30] for an overview), the same cannot be said for the decomposition of time of even simple systems of ODEs. This is despite the fact that research on time domain decomposition is currently very active and has by now a history of at least 50 years (back to at least [28]) during which several algorithms have been explored (see [14] for an overview). As a consequence, time domain decomposition is to date only a secondary option when it comes to deciding what algorithm/method distributes the tasks in a parallel cluster.

The main goal of this work is to address this efficiency limitation in the framework of one particular scheme: the parareal in time algorithm. The method was first introduced in [19] and has been well accepted by the community because it is easily applicable to a relatively large spectrum of problems. (Some specific difficulties are nevertheless encountered on certain types of PDEs as reported in, e.g., [8, 12] for hyperbolic systems or [4, 7] for hamiltonian problems). Another ingredient for its success is that, even though its scalability properties are limited, they are in general competitive in comparison with other methods. Without entering into very specific details of the algorithm at this stage, we can summarize the procedure by saying that

we build iteratively a sequence to approximate the exact solution of the problem by a predictor-corrector algorithm. At every iteration, predictions are made by a solver which has to be as numerically inexpensive as possible since it is run on the full time interval. It usually involves coarse physics and/or coarse resolution. Corrections involve an expensive solver with high-fidelity physics and high resolution which is propagated in parallel over small time subdomains. In the classical version of parareal, the fine solver has a fixed high accuracy across all iterations. It is set to the one that we would use to solve the dynamics at the desired accuracy with a purely sequential solver. It is well-known that this point is the major obstacle to achieve better parallel efficiency. In this paper, we propose an adaptive variant where the accuracy of the fine solver is increased across the iterations. Our main goal is to show that this new point of view overcomes the obstacle of the cost of the fine solver and that the only remaining factor limiting high performance becomes the cost of the coarse solver. We refer to, e.g., [6] for contributions on the lowering of the cost of that coarse solver.

We present in section 2 the new adaptive point of view. This requires to formulate an idealized version of the parareal algorithm in an infinite dimensional function space where the fine propagations are replaced by the exact ones (section 2.2). Since this scheme is obviously not implementable in practice, we formulate a feasible "perturbed" version that involves approximations of the exact propagations at increasing accuracy across the iterations (section 2.3). The accuracies are tightened in such a way that the feasible adaptive algorithm converges at the same rate as the ideal one and with a near-minimal numerical cost. The identified tolerances involve quantities that are difficult to estimate in practice. In addition, they may not be optimal because they are derived from a theoretical convergence analysis based on abstract conditions for the coarse and fine solvers. We bridge this gap between theory and actual implementation by proposing practical guidelines to set these tolerances. We next explain in section 2.4 how the new formulation invites to use adaptive schemes not only in the time variable, but also in other variables that may be involved in the dynamics. The performance of the algorithm could also be enhanced by re-using informations from previous iterations in order to limit the cost of internal solvers. The techniques for this will strongly depend on the nature of the specific problem. We discuss common situations in Appendix A. We close section 2.5 by listing the main advantages of the new framework and how the classical parareal paradigm can be formulated with the optics of the new standpoint.

The parallel performance of the adaptive scheme is difficult to predict a priori but in section 3 we carry a discussion where we show that it will always be superior to the classical approach. In the idealized situation where the cost of the coarse solver and communication delays are negligible, we show that the algorithm would exhibit a very high parallel efficiency.

Finally, in section 4, we illustrate the performance of the algorithm in a numerical example: the Brusselator system. It is a stiff ODE system which is challenging because it does not allow to use a very inexpensive coarse solver. We show that the adaptive parareal algorithm systematically performs around 2.5 times better than the classical one. In addition, we confirm in the example that the only remaining obstacle to achieve a very competitive performance is the cost of the coarse solver. Due to the nature of the algorithm, it is not clear how to overcome this limitation and addressing this issue is deferred to future works.

We conclude this introduction by some bibliographical remarks. To the best of our knowledge, the current abstract and broad formulation of an adaptive version of parareal has never been proposed in the literature. However, previous works have instantiated in a variety of particular applications the idea of re-using information from previous parareal iterations and this is the main point of contact between this work and previous contributions. Among the most relevant ones stand the coupling of the parareal algorithm with spatial domain decomposition (see [22, 16, 1]), the combination of the parareal algorithm with iterative high order methods

in time like spectral deferred corrections (see [26, 23, 25]) and, in a similar spirit, applications of the parareal algorithm to solve optimal control problems (see [22, 21]). In appendix A, we briefly explain in what sense the two first applications can be seen as particular instances of the current approach and how our viewpoint could help to give them more solid theoretical foundations. Finally, we identify another relevant scenario where efficiency could be enhanced by reusing information from previous iterations: the solution of time-dependent problems involving internal iterative schemes at every time step. This idea was first proposed in [27] and a more complete analysis will be proposed in a forthcoming work.

2 An adaptive parareal algorithm

In this section, after introducing some preliminary notations in section 2.1, we formulate an ideal parareal scheme on an infinite dimensional functional setting (section 2.2). We then present feasible realizations involving a fine solver whose accuracy is adaptively increased across the iterations (section 2.3). We prove that the feasible adaptive algorithm converges at the same rate as the ideal one provided that the tolerances of the fine solver are increased at certain rate which will be discussed. Finally, we discuss how the new paradigm can be realized thanks to adaptive schemes and/or the re-use of information from previous steps (section 2.4 and appendix A). In section 2.5, we connect the new adaptive formulation with the classical parareal algorithm and list the main advantages of the new standpoint.

2.1 Setting and preliminary notations

Let $\mathbb{U}$ be a Banach space of functions defined over a domain $\Omega \subset \mathbb{R}^d$ ($d \geq 1$), e.g. $\mathbb{U} = L^2(\Omega)$. Let

$$\mathcal{E} : [0,T] \times [0,T] \times \mathbb{U} \to \mathbb{U}$$

be a propagator, that is, an operator such that, for any given time $t \in [0,T]$, $s \in [0, T-t]$ and any function $w \in \mathbb{U}$, $\mathcal{E}(t,s,w)$ takes w as an initial value at time t and propagates it at time $t+s$. We assume that $\mathcal{E}$ satisfies the semi group property

$$\mathcal{E}(r, t-r, w) = \mathcal{E}(s, t-s, \mathcal{E}(r, s-r, w)), \quad \forall w \in \mathbb{U}, \forall (r,s,t) \in [0,T]^3, \ r < s < t.$$

We further assume that $\mathcal{E}$ is implicitly defined through the solution $u \in \mathcal{C}^1([0,T], \mathbb{U})$ of the time-dependent problem

$$u'(t) + \mathcal{A}(t, u(t)) = 0, \quad t \in [0,T], \tag{1}$$

where $\mathcal{A}$ is an operator from $[0,T] \times \mathbb{U}$ into $\mathbb{U}$ with adequate regularity we shall detail latter. Then, given any $w \in \mathbb{U}$, $\mathcal{E}(t,s,w)$ denotes the solution to (1) at time $t+s$ with initial condition w at time $t \geq 0$. In our problem of interest, we study the evolution given by (1) when the initial condition is $u(0) \in \mathbb{U}$. Note that $\mathcal{E}$ could also be associated to a discretized version of the evolution equation or be defined through an operator that is not necessary related to an evolution equation (see [13]).

Since, in general, the problem does not have an explicit solution, we seek to approximate it at a given target accuracy. For any initial value $w \in \mathbb{U}$, any $t \in [0,T[$, $s \in [0, T-t]$ and any $\zeta > 0$ we denote by $[\mathcal{E}(t,s,w); \zeta]$ an element of $\mathbb{U}$ that approximates $\mathcal{E}(t,s,w)$ such that we have

$$\|\mathcal{E}(t,s,w) - [\mathcal{E}(t,s,w); \zeta]\| \leq \zeta \, s \, (1 + \|w\|), \tag{2}$$

where, here and in the following, $\|\cdot\|$ denotes the norm in $\mathbb{U}$. Any realization of $[\mathcal{E}(t,s,w); \zeta]$ involves three main ingredients:

i) a numerical scheme to discretize the time dependent problem (1) (e.g. an Euler scheme in time),

ii) a certain expected error size associated with the choice of the discretization (e.g. error associated with the time step size of the Euler scheme),

iii) a numerical implementation to solve the resulting discrete systems (e.g. conjugate gradient, Newton method, SSOR, ...).

In the following, we will use the term *solver* to denote a particular choice for i), ii) and iii). Given a solver $\mathcal{S}$, we will use the same notation as for the exact propagator $\mathcal{E}$ to express that $\mathcal{S}(t,s,w)$ is an approximation of $\mathcal{E}(t,s,w)$ with a certain accuracy ζ. In other words, we can write $\mathcal{S}(t,s,w) = [\mathcal{E}(t,s,w);\zeta]$.

2.2 An idealized version of the parareal algorithm

Let be given a decomposition of the time interval $[0,T]$ into $\underline{N}$ subintervals $[T_N, T_{N+1}]$, $N = 0,\dots,\underline{N}-1$. Without loss of generality, we will take them of uniform size $\Delta T = T/\underline{N}$ which means that $T_N = N\Delta T$ for $N = 0,\dots,\underline{N}$. For a given target accuracy $\eta > 0$, the primary goal of the parareal in time algorithm is to build an approximation $\tilde{u}(T_N)$ of $u(T_N)$ such that

$$\max_{1\le N\le \underline{N}} \|u(T_N) - \tilde{u}(T_N)\| \le \eta. \tag{3}$$

The classical way to achieve this is to set

$$\tilde{u}(T_N) = \mathcal{S}_{\text{seq}}(0, T_N, u(0)) = [\mathcal{E}(0, T_N, u(0)); \zeta], \quad 1 \le N \le \underline{N},$$

where $\mathcal{S}_{\text{seq}}$ is some sequential solver in $[0,T]$ with $\zeta = \eta/(T(1+\|u(0)\|))$ in (2). Since this comes at the cost of solving over the whole time interval $[0,T]$, the main goal of the parareal in time algorithm is to speed up the computing time, while maintaining the same target accuracy η. This is made possible by first decomposing the computations over the time domain. Instead of solving over $[0,T]$, we perform $\underline{N}$ parallel solves over each intervals $(T_N, T_{N+1}]$of size ΔT. We next introduce an idealized version of it which will not be feasible in practice but will be the starting point of subsequent implementable versions. The algorithm relies on the use of a solver $\mathcal{G}$ (known as the coarse solver) with the following properties involving the operator

$$\delta\mathcal{G} := \mathcal{E} - \mathcal{G}.$$

Hypotheses (H): There exists $\varepsilon_{\mathcal{G}},\ C_c,\ C_d > 0$ such that for any function $x,\ y \in \mathbb{U}$ and for any $t \in [0,T[$ and $s \in [0, T-t]$,

$$\mathcal{G}(t,s,x) = [\mathcal{E}(t,s,x), \varepsilon_{\mathcal{G}}] \quad \Leftrightarrow \quad \|\delta\mathcal{G}(t,s,x)\| \le s(1+\|x\|)\varepsilon_{\mathcal{G}} \tag{4a}$$

$$\|\mathcal{G}(t,s,x) - \mathcal{G}(t,s,y)\| \le (1 + C_c s)\|x-y\|, \tag{4b}$$

$$\|\delta\mathcal{G}(t,s,x) - \delta\mathcal{G}(t,s,y)\| \le C_d s \varepsilon_{\mathcal{G}} \|x-y\| \tag{4c}$$

Note that these hypothesis are the classical abstract formulations of the properties of numerical schemes related to stability and accuracy. Hypothesis (4b) is a Lipschitz condition and the quantity $\varepsilon_{\mathcal{G}}$ is a small constant which, in the case of a Euler scheme, would be equal to the time step size.

The idealized version of the algorithm consists in building iteratively a series $(y_k^N)_k$ of approximations of $u(T_N)$ for $0 \leq N \leq \underline{N}$ following the recursive formula

$$\begin{cases} y_0^{N+1} = \mathcal{G}(T_N, \Delta T, y_0^N), & 0 \leq N \leq \underline{N}-1 \\ y_{k+1}^{N+1} = \mathcal{G}(T_N, \Delta T, y_{k+1}^N) + \mathcal{E}(T_N, \Delta T, y_k^N) & \\ \qquad\qquad - \mathcal{G}(T_N, \Delta T, y_k^N), & 0 \leq N \leq \underline{N}-1, \quad k \geq 0, \\ y_0^0 = u(0). & \end{cases} \tag{5}$$

At this point, several comments are in order. The first one is that the computation of y_k^N only requires propagations with $\mathcal{E}$ over intervals of size ΔT. As follows from (5), for a given iteration k, $\underline{N}$ propagations of this size are required, each of them over distinct intervals $[T_N, T_{N+1}]$ of size ΔT, each of them with independent initial conditions. Since they are independent from each other, they can be computed over $\underline{N}$ parallel processors and the original computation over $[0, T]$ is decomposed into parallel computations over $\underline{N}$ subintervals of size ΔT. The second observation is that the algorithm may not be implementable in practice because it involves the exact propagator $\mathcal{E}$. Feasible instantiations consist in replacing $\mathcal{E}(T_N, \Delta T, y_k^N)$ by some approximation $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ with a certain accuracy ζ_k^N which has to be carefully chosen. We will come to this point in the next section. The third observation is to note that, in the current version of the algorithm, for all $N = 0, \ldots, \underline{N}$, the exact solution $u(T_N)$ is obtained after exactly $k = N$ parareal iterations. This number can be reduced when we only look for an approximate solution with accuracy η. Depending on the problem, the final number of iterations $K(\eta)$ can actually be much smaller than $\underline{N}$. The convergence result of theorem 2.1 and its proof are helpful to understand the main mechanisms driving the convergence of the algorithm and explaining its behavior. To present it, we introduce the shorthand notation for the error norm

$$E_k^N := \|u(T_N) - y_k^N\|, \quad k \geq 0, \ 0 \leq N \leq \underline{N},$$

and the quantities

$$\mu = \frac{e^{C_c T}}{C_d} \max_{0 \leq N \leq \underline{N}} (1 + \|u(T_N)\|), \quad \text{and} \quad \tau := C_d T e^{-C_c \Delta T} \varepsilon_{\mathcal{G}}.$$

Theorem 2.1. *If $\mathcal{G}$ and $\delta\mathcal{G}$ satisfy Hypothesis* (4)*, then,*

$$\max_{0 \leq N \leq \underline{N}} \|u(T_N) - y_k^N\| \leq \mu \frac{\tau^{k+1}}{(k+1)!}, \quad \forall k \geq 0. \tag{6}$$

Proof. The proof is in the spirit of existing results from the literature (see [19, 5, 20, 15]) but it is instructive to give it for subsequent developments in the paper. We introduce the following quantities

$$\begin{cases} \alpha & := C_d \varepsilon_{\mathcal{G}} \Delta T \\ \beta & := 1 + C_c \Delta T \\ \gamma & := \Delta T \varepsilon_{\mathcal{G}} \max_{0 \leq N \leq \underline{N}} (1 + \|u(T_N)\|) \end{cases} \tag{7}$$

as shorthand notations for the proof.

If $k = 0$, using definition (5) for y_0^N, we have for $0 \leq N \leq \underline{N} - 1$,

$$\begin{aligned} E_0^{N+1} &= \|y_0^{N+1} - u(T_{N+1})\| \\ &= \|\mathcal{G}(T_N, \Delta T, y_0^N) - \mathcal{E}(T_N, \Delta T, u(T_N))\| \\ &\leq \|\mathcal{G}(T_N, \Delta T, y_0^N) - \mathcal{G}(T_N, \Delta T, u(T_N))\| + \|\mathcal{G}(T_N, \Delta T, u(T_N)) - \mathcal{E}(T_N, \Delta T, u(T_N))\| \\ &\leq (1 + C_c \Delta T) E_0^N + \Delta T \varepsilon_{\mathcal{G}} (1 + \|u(T_N)\|) \\ &\leq \beta E_0^N + \gamma, \end{aligned}$$

where we have used (4a) and (4b) to derive the second to last inequality.

For $k \geq 1$, starting from (5), we have

$$\begin{aligned} y_k^{N+1} - u(T_{N+1}) &= \mathcal{G}(T_N, \Delta T, y_k^N) + \mathcal{E}(T_N, \Delta T, y_{k-1}^N) - \mathcal{G}(T_N, \Delta T, y_{k-1}^N) - \mathcal{E}(T_N, \Delta T, u(T_N)) \\ &= \mathcal{G}(T_N, \Delta T, y_k^N) - \mathcal{G}(T_N, \Delta T, u(T_N)) + \delta\mathcal{G}(T_N, \Delta T, y_{k-1}^N) - \delta\mathcal{G}(T_N, \Delta T, u(T_N)). \end{aligned}$$

Taking norms and using (4b), (4c), we derive

$$E_k^{N+1} \leq \beta E_k^N + \alpha E_{k-1}^N,$$

Following [15], we consider the sequence $(e_k^N)_{N,k\geq 0}$ defined recursively as follows. For $k = 0$,

$$e_0^N = \begin{cases} 0, & \text{if } N = 0 \\ \beta e_0^{N-1} + \gamma, & \text{if } N \geq 1 \end{cases} \tag{8}$$

and for $k \geq 1$,

$$e_k^N = \begin{cases} 0, & \text{if } N = 0 \\ \alpha e_{k-1}^{N-1} + \beta e_k^{N-1}, & \text{if } N \geq 1. \end{cases} \tag{9}$$

Since $E_k^N \leq e_k^N$ for $k \geq 0$ and $N = 0, \ldots, \underline{N}$, we analyze the behavior of (e_k^N) to derive a bound for E_k^N. For this, we consider the generating function

$$\rho_k(\xi) = \sum_{N\geq 0} e_k^N \xi^N.$$

From (8) and (9) we get

$$\begin{cases} \rho_k(\xi) = \alpha\xi\rho_{k-1}(\xi) + \beta\xi\rho_k(\xi), & k \geq 1 \\ \rho_0(\xi) = \gamma\frac{\xi}{1-\xi} + \beta\xi\rho_0(\xi), \end{cases}$$

from which we derive

$$\rho_k(\xi) = \gamma\alpha^k \frac{\xi^{k+1}}{(1-\xi)} \frac{1}{(1-\beta\xi)^{k+1}}, \quad k \geq 0.$$

Since, $\beta \geq 1$, we can bound the term $(1-\xi)$ in the denominator by $(1-\beta\xi)$. Next, using the binomial expansion

$$\frac{1}{(1-\beta\xi)^{k+2}} = \sum_{j\geq 0} \binom{k+1+j}{j} \beta^j \xi^j \tag{10}$$

and identifying the term in ξ^N in the expansion, we derive the bound

$$e_k^N \leq \gamma\alpha^k \beta^{N-k-1} \binom{N}{k+1}.$$

Hence, using definition (7) for α, β and γ,

$$E_k^N \leq e_k^N \leq \frac{(1+C_c\Delta T)^{N-k-1} \max_{0\leq N\leq \underline{N}}(1+||u(T_N)||)}{C_d(k+1)!} \left[C_d\varepsilon_{\mathcal{G}} e^{-C_c\Delta T} T_N\right]^{k+1},$$

which ends the proof of the lemma. □

Note that at least one step is not sharp in the above proof: it is the step where $1-\xi$ is replaced by $1-\beta\xi$. Note also that τ is the quantity driving convergence and its speed. It follows that a sufficient condition to converge is to use a coarse solver such that

$$\tau < 1 \quad \Leftrightarrow \quad \varepsilon_{\mathcal{G}} < \frac{e^{C_c\Delta T}}{C_d T}.$$

This gives a certain limitation on the definition of the coarse solver since it imposes some minimal accuracy. In the following, we will work under this assumption for $\varepsilon_{\mathcal{G}}$.

2.3 Feasible realizations of the parareal algorithm

Feasible versions of algorithm (5) involve approximations of $\mathcal{E}(T_N, \Delta T, y_k^N)$ with a certain accuracy ζ_k^N. This leads to consider algorithms of the form

$$\begin{cases} y_0^{N+1} = \mathcal{G}(T_N, \Delta T, y_0^N), & 0 \le N \le \underline{N} - 1 \\ y_{k+1}^{N+1} = \mathcal{G}(T_N, \Delta T, y_{k+1}^N) + [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N] & \\ \qquad\qquad - \mathcal{G}(T_N, \Delta T, y_k^N), & 0 \le N \le \underline{N} - 1,\ k \ge 0, \\ \quad y_0^0 = u(0). & \end{cases} \tag{11}$$

Since no feasible version will converge at a better rate than (6), we analyze here what is the minimal accuracy ζ_k^N that preserves it. A result in this direction is given in the following theorem. It requires to introduce the quantity

$$\nu_p := \frac{\max_{0 \le N \le \underline{N}}(1 + \|y_p^N\|)}{\max_{0 \le N \le \underline{N}}(1 + \|u(T_N)\|)}, \quad \forall p \ge 0.$$

which tends to 1 as $p \to \infty$.

Theorem 2.2. *Let $\mathcal{G}$ and $\delta\mathcal{G}$ satisfy Hypothesis* (4)*. Let $k \ge 0$ be any given positive integer. If for all $0 \le p < k$ and all $0 \le N < \underline{N}$, the approximation $[\mathcal{E}(T_N, \Delta T, \zeta_p^N)]$ has accuracy*

$$\zeta_p^N \le \zeta_p := \frac{\varepsilon_{\mathcal{G}}^{p+2}}{(p+1)!\,\nu_p}, \tag{12}$$

then the $(y_k^N)_N$ of the feasible parareal scheme (11) *satisfy*

$$\max_{0 \le N \le \underline{N}} \|u(T_N) - y_k^N\| \le \mu \frac{\tilde{\tau}^{k+1}}{(k+1)!}, \tag{13}$$

with

$$\tilde{\tau} := \left(1 + C_d T e^{-C_c \Delta T}\right) \varepsilon_{\mathcal{G}}.$$

Let us make a couple of remarks before giving the proof of the theorem. First, comparing (6) and (13), we see that the feasible parareal algorithm converges at a rate close to the ideal one in the sense that it only deviates by a factor

$$\frac{\tilde{\tau}}{\tau} = \frac{\tau + \varepsilon_{\mathcal{G}}}{\tau} = 1 + \frac{e^{C_c \Delta T}}{C_d T}.$$

For a given problem with fixed C_c, C_d and T, this deviation depends on the size of ΔT, which is itself driven by the number of processors $\underline{N}$. The minimal accuracy to update the realization of $\mathcal{E}(T_N, \Delta T, y_k^N)$ is given by (12). As a final remark, we note that we can also interpret both convergence bounds (6) and (13) in terms of $\varepsilon_{\mathcal{G}}$. With this viewpoint, the error at iteration k is bounded by $C(k)\varepsilon_{\mathcal{G}}^{k+1}/(k+1)!$ where $C(k)$ is a factor which grows (only) exponentially with k and is thus "swallowed" by the denominator.

Proof. The proof follows the same lines as the one for theorem 2.1 and E_k^N, α, β, γ are defined exactly as before. In addition, it will be useful to introduce the sequence

$$\begin{cases} g_k & = \zeta_k \Delta T \max_{0 \le N \le \underline{N}}(1 + \|y_k^N\|), \quad \forall k \ge 0 \\ g_{-1} & = \gamma \end{cases}$$

We concentrate on the case $k \geq 1$ since the case $k = 0$ is identical as in theorem 2.1. For $k \geq 1$, using (11), we have

$$\begin{aligned} y_k^{N+1} - u(T_{N+1}) &= \mathcal{G}(T_N, \Delta T, y_k^N) - \mathcal{G}(T_N, \Delta T, u(T_N)) - \mathcal{G}(T_N, \Delta T, y_{k-1}^N) \\ &\quad + \mathcal{G}(T_N, \Delta T, u(T_N)) + [\mathcal{E}(T_N, \Delta T, y_{k-1}^N); \zeta_{k-1}^N] - \mathcal{E}(T_N, \Delta T, u(T_N)) \\ &= \mathcal{G}(T_N, \Delta T, y_k^N) - \mathcal{G}(T_N, \Delta T, u(T_N)) + \delta\mathcal{G}(T_N, \Delta T, y_{k-1}^N) \\ &\quad - \delta\mathcal{G}(T_N, \Delta T, u(T_N)) + [\mathcal{E}(T_N, \Delta T, y_{k-1}^N); \zeta_{k-1}^N] - \mathcal{E}(T_N, \Delta T, y_{k-1}^N). \end{aligned}$$

Taking norms, using (4b), (4c) and the definition (2) applied to $[\mathcal{E}(T_N, \Delta T, y_{k-1}^N); \zeta_{k-1}^N]$, we derive

$$\begin{aligned} E_k^{N+1} &\leq [1 + C_c \Delta T] E_k^N + C_d \Delta T \varepsilon_{\mathcal{G}} E_{k-1}^N + \zeta_{k-1}^N \Delta T \max_{0 \leq N \leq \underline{N}} (1 + \|y_{k-1}^N\|) \\ &\leq \beta E_k^N + \alpha E_{k-1}^N + g_{k-1}. \end{aligned}$$

Similarly to theorem 2.1, we introduce the sequence $(\tilde{e}_k^N)_{N,k \geq 0}$ defined for $k = 0$ as $\tilde{e}_0^N = e_0^N$ for all $N \geq 0$ and for $k \geq 1$,

$$\tilde{e}_k^N = \begin{cases} 0, & \text{if } N = 0 \\ \alpha \tilde{e}_{k-1}^{N-1} + \beta \tilde{e}_k^{N-1} + g_{k-1}, & \text{if } N \geq 1 \end{cases}$$

The associated generating function $\tilde{\rho}_k$ satisfies

$$\begin{cases} \tilde{\rho}_k(\xi) & = \alpha\xi\tilde{\rho}_{k-1}(\xi) + \beta\xi\tilde{\rho}_k(\xi) + g_{k-1}\frac{\xi}{1-\xi}, \quad \forall k \geq 1, \\ \tilde{\rho}_0(\xi) & = \rho_0(\xi) = \frac{\gamma\xi}{(1-\xi)(1-\beta\xi)}. \end{cases}$$

Hence

$$\begin{aligned} \tilde{\rho}_k(\xi) &= \left(\frac{\alpha\xi}{1-\beta\xi}\right) \tilde{\rho}_{k-1}(\xi) + \frac{\xi}{(1-\xi)(1-\beta\xi)} g_{k-1} \\ &= \left(\frac{\alpha\xi}{1-\beta\xi}\right)^k \tilde{\rho}_0(\xi) + \frac{\xi}{(1-\xi)(1-\beta\xi)} \sum_{\ell=0}^{k-1} \left(\frac{\alpha\xi}{1-\beta\xi}\right)^\ell g_{k-1-\ell} \end{aligned}$$

By replacing again at the denominator the factor $(1 - \xi)$ by $(1 - \beta\xi)$ and using the binomial expansion (10), we derive the bound

$$\begin{aligned} \tilde{\rho}_k(\xi) &\leq \gamma\alpha^k\xi^{k+1} \sum_{j \geq 0} \binom{k+1+j}{j} \beta^j \xi^j + \sum_{\ell=0}^{k-1} \alpha^\ell \xi^{\ell+1} g_{k-1-\ell} \sum_{j \geq 0} \binom{\ell+1+j}{j} \beta^j \xi^j, \\ &= \sum_{j \geq 0} \sum_{\ell=0}^{k} \alpha^\ell g_{k-1-\ell} \beta^j \binom{\ell+1+j}{j} \xi^{\ell+1+j} \end{aligned}$$

where we have used that $g_{-1} = \gamma$. The coefficient associated to the term ξ^N above gives the inequality

$$\tilde{e}_k^N \leq \sum_{\ell=0}^{k} \alpha^\ell g_{k-1-\ell} \beta^{N-\ell-1} \binom{N}{\ell+1}, \quad \forall k \geq 1,$$

From the definition of ζ_ℓ, we have that

$$g_\ell \leq \frac{\Delta T \varepsilon_{\mathcal{G}}^{\ell+2}}{(\ell+1)!} \max_{0 \leq N \leq \underline{N}} (1 + \|u(T_N)\|).$$

Therefore, recalling the definition (7) of α, β and γ, we derive

$$\begin{aligned}
\tilde{e}_k^N &\leq \frac{\varepsilon_{\mathcal{G}}^{k+1} \max_{0\leq N\leq \underline{N}}(1+\|u(T_N)\|)}{C_d} \sum_{\ell=0}^{k} (C_d\Delta T)^{\ell+1} \frac{(1+C_c\Delta T)^{N-\ell-1}}{(k-\ell)!} \binom{N}{\ell+1} \\
&\leq \frac{\varepsilon_{\mathcal{G}}^{k+1} \max_{0\leq N\leq \underline{N}}(1+\|u(T_N)\|)}{C_d} \sum_{\ell=0}^{k} \left(C_d T e^{-C_C\Delta T}\right)^{\ell+1} \frac{e^{C_C T}}{(\ell+1)!\,(k-1-\ell)!} \\
&\leq \frac{\max_{0\leq N\leq \underline{N}}(1+\|u(T_N)\|)e^{C_c T}}{C_d(k+1)!} \left((1+C_d T e^{-C_c\Delta T})\varepsilon_{\mathcal{G}}\right)^{k+1},
\end{aligned}$$

which ends the proof since $E_k^N \leq \tilde{e}_k^N$ for $N = 0, \ldots, \underline{N}$. □

2.4 Practical realization of $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$

Since the accuracy ζ_k^N needs to improve with k, the most natural way to build the approximations $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ is with adaptive techniques and with adaptive refinements at every step k. The implementation ultimately rests on the use of a posteriori error estimators. It opens the door to local time step adaptation in the parareal algorithm as well as spatial coarsening or refinement if the problem involves additional spatial variables.

In principle, as ζ_k^N decreases with k, the numerical cost increases in terms of degrees of freedom and also in terms of computing time. This actually reveals the key idea of this new approach which is that we would like that only the last fine solver is expensive and the cost of the previous ones is a small fraction of the cost of the last one (we refer to the next sub-section for a more precise statement). By re-using information from previous iterations, we can limit the cost of internal solvers required in $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ and enhance the speed-up. This depends of course on the nature of the specific problem. We discuss several common situations in Appendix A.

2.5 Connection to the classical formulation of the parareal algorithm and advantages of the current view-point

In the original version of the algorithm, $\mathcal{E}(T_N, \Delta T, y_k^N)$ is approximated with an accuracy $\zeta_k^N = \zeta_{\mathcal{F}}$ which is kept *constant* in N and across the parareal iterations k. This has usually been done by using a solver $\mathcal{F}$ defined in the same spirit as $\mathcal{G}$, but satisfying Hypothesis (4) with a better accuracy $\varepsilon_{\mathcal{F}} < \varepsilon_{\mathcal{G}}$. We have in this case

$$[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_{\mathcal{F}}] = \mathcal{F}(T_N, \Delta T, y_k^N)$$

and we recover the classical algorithm (see [2] and [3])

$$\begin{cases}
y_0^{N+1} = \mathcal{G}(T_N, \Delta T, y_0^N), & 0 \leq N \leq \underline{N}-1 \\
y_{k+1}^{N+1} = \mathcal{G}(T_N, \Delta T, y_{k+1}^N) & \\
\qquad\qquad + \mathcal{F}(T_N, \Delta T, y_k^N) - \mathcal{G}(T_N, \Delta T, y_k^N), & 0 \leq N \leq \underline{N}-1, \quad k \geq 0, \\
y_0^0 = u(0). &
\end{cases}$$

Compared to this classical version of the parareal algorithm, the adaptive approach offers the following important advantages:

1. The algorithm *converges to the exact solution* $u(T_N)$ and not to the solution achieved by the fixed chosen fine solver $\mathcal{F}(0, T_N, u(0))$ (indeed, for any N, $\zeta_k^N \longrightarrow 0$ as $k \longrightarrow \infty$).

2. For a final target accuracy η, the parallel efficiency will always be superior to the classical approach (see section 3).

3. The numerical cost is designed to be near-minimal because we identify the minimal required accuracies at each iteration (see equation (12)). Early iterations use a loose tolerance, thus avoiding unnecessary work due to oversolving, while later iterations use tighter tolerances to deliver accuracy.

4. The dynamical refinements of the fine solver invite to incorporate adaptive solvers with a posteriori error estimators to the parareal scheme.

3 Parallel efficiency

It is difficult to give accurate a priori estimations for the speed-up and efficiency of the method due to its adaptive nature so the actual performance can only be established through relevant examples. In section 4, we give some results for the case of the Brusselator system. Despite this difficulty in estimation, we make some general remarks in this section, which aim primarily at highlighting the relevance of the cost of the coarse solver. The speed-up is defined as the ratio

$$\textbf{speed-up}_{\mathrm{AP/seq}}(\eta, [0,T]) := \frac{\textbf{cost}_{\mathrm{seq}}(\eta, [0,T])}{\textbf{cost}_{\mathrm{AP}}(\eta, [0,T])} \tag{14}$$

between the cost to run a sequential fine solver achieving a target accuracy η with the cost to run an adaptive parareal algorithm providing at the end the same target accuracy η. The parallel efficiency of the method is then defined as the ratio of the above speed up with the number of processor which gives a target of 1 to any parallel solver:

$$\textbf{eff}_{\mathrm{AP/seq}}(\eta, [0,T]) := \frac{\textbf{speed-up}_{\mathrm{AP/seq}}(\eta, [0,T])}{\underline{N}}.$$

Assume that g_k^N and f_k^N are the numerical costs to realize $\mathcal{G}(T_N, \Delta T, y_k^N)$ and $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$. Since the tolerances ζ_k^N decrease with k, we have $f_0^N < \cdots < f_{K(\eta)}^N$. Neglecting the communication delays, the cost of the adaptive solver is

$$\textbf{cost}_{\mathrm{AP}}(\eta, [0,T]) = \sum_{k=0}^{K(\eta)} \sum_{N=0}^{\underline{N}-1} g_k^N + \sum_{k=0}^{K(\eta)-1} \sum_{N=0}^{\underline{N}-1} f_k^N.$$

The classical parareal algorithm involves, at every iteration $k \in \{0, ..., K(\eta)\}$ propagations at the highest accuracy $\zeta_{K(\eta)}^N$. Thus its cost is

$$\textbf{cost}_{\mathrm{CP}}(\eta, [0,T]) = \sum_{k=0}^{K(\eta)} \sum_{N=0}^{\underline{N}-1} g_k^N + K(\eta) \sum_{N=0}^{\underline{N}-1} f_{K(\eta)}^N,$$

from which it directly follows that (at least if, with obvious notation, $K_{AP}(\eta) = K_{CP}(\eta)$)

$$\textbf{eff}_{\mathrm{AP/seq}}(\eta, [0,T]) > \textbf{eff}_{\mathrm{CP/seq}}(\eta, [0,T]).$$

Thus the parallel performance of the adaptive algorithm is at least the one of the classical version. Note that this holds even when communications are not negligible since there is the same amount of information exchange in both algorithms.

We next give a more quantitative statement on an admittedly idealized setting. Assume that the cost of the coarse solve is negligible, that there is no communication delay and that the cost to realize $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ is

$$f_k^N = \Delta T (\zeta_k^N)^{-1/\alpha}. \tag{15}$$

This assumption for the cost is, for instance, reasonable when we use an explicit time-stepping method of order $\alpha > 0$. It would also hold for an implicit method where a direct solver can be used. Note that α could actually depend on k but we stick to this simple model for clarity of exposition.

Proposition 3.1. *If $f_k^N = \Delta T (\zeta_k^N)^{-1/\alpha}$, for some $\alpha > 0$ and if the cost of the coarse solver is negligible with respect to f_k^N for any $k \geq 0$, then*

$$\textbf{\textit{eff}}_{AP/seq}(\eta, [0,T]) = \frac{1 - \tau^{1/\alpha}}{1 - \tau^{K(\eta)/\alpha}} \sim \frac{1}{(1 + \varepsilon_{\mathcal{G}}^{1/\alpha})}.$$

Therefore

$$\textbf{\textit{speed-up}}_{AP/seq}(\eta, [0,T]) \sim \underline{N} \frac{1}{(1 + \varepsilon_{\mathcal{G}}^{1/\alpha})}.$$

Proof. The cost of the scalable adaptive parareal scheme after $K(\eta)$ iterations is

$$\mathbf{cost}_{\mathrm{AP}}(\eta, [0,T]) = \Delta T \sum_{k=0}^{K(\eta)-1} \zeta_k^{-1/\alpha} \tag{16}$$

Since we are in a range where the scheme converges, the quantity $\max_{0 \leq N \leq \underline{N}} \|y_k^N\|$ is bounded and thus there exists $0 < \underline{c} \leq 1 \leq \overline{c}$ such that $\underline{c} \leq \nu_k \leq \overline{c}$ for all $k \geq 0$. We will account for this with the notation $\nu_k \sim 1$. Note that in fact $\underline{c}$ and $\overline{c}$ are close to one. Let us start with the simple case $\alpha = 1$ and denote $\underline{K} = K(\eta) - 1$:

$$\mathbf{cost}_{\mathrm{AP}}(\eta, [0,T]) = \Delta T \sum_{k=0}^{\underline{K}} \zeta_k^{-1} = \Delta T \zeta_{K(\eta)-1}^{-1} \left(1 + \frac{\varepsilon_{\mathcal{G}}}{\underline{K}} + \frac{\varepsilon_{\mathcal{G}}^2}{\underline{K}(\underline{K}-1)} + \cdots + \frac{\varepsilon_{\mathcal{G}}^{\underline{K}-1}}{\underline{K}!} \right)$$

we thus derive

$$\mathbf{cost}_{\mathrm{AP}}(\eta, [0,T]) \leq \Delta T \zeta_{K(\eta)-1}^{-1} (1 + \varepsilon_{\mathcal{G}})$$

In the general case ($\alpha > 1$), the same is true with

$$\mathbf{cost}_{\mathrm{AP}}(\eta, [0,T]) \leq \Delta T \zeta_{K(\eta)-1}^{-1/\alpha} (1 + \varepsilon_{\mathcal{G}}^{1/\alpha})$$

Up to this last factor, the current conclusion is that the global cost of the parareal procedure is equal to the last fine solver on each sub-interval with size ΔT (both the coarse and the previous fine propagations are negligible).

Since the accuracy that is obtained at the end of the parareal procedure (see (13)) is of the same order as the accuracy provided with classical parareal solver (compare with (6)), it follows that if we now take the last target accuracy $\zeta_{K(\eta)-1}^{-1/\alpha}$ of the adaptive algorithm as the accuracy of the fine scheme in the classical parareal algorithm, the cost would be

$$\mathbf{cost}_{\mathrm{CP}}(\eta, [0,T]) = K(\eta) \Delta T \zeta_{K(\eta)-1}^{-1/\alpha},$$

Therefore,

$$\frac{\mathbf{cost}_{\mathrm{AP}}(\eta,[0,T])}{\mathbf{cost}_{\mathrm{CP}}(\eta,[0,T])} \sim \frac{1}{K(\eta)}(1+\varepsilon_{\mathcal{G}}^{1/\alpha}). \tag{17}$$

In addition, we know that when the cost of the coarse solver is negligible,

$$\mathbf{speed\text{-}up}_{\mathrm{CP/seq}}(\eta,[0,T]) = \frac{\mathbf{cost}_{\mathrm{seq}}(\eta,[0,T])}{\mathbf{cost}_{\mathrm{CP}}(\eta,[0,T])} = \frac{\underline{N}}{K(\eta)}, \tag{18}$$

Dividing (18) by (17) yields

$$\mathbf{speed\text{-}up}_{\mathrm{AP/seq}}(\eta,[0,T]) \sim \underline{N}\frac{1}{(1+\varepsilon_{\mathcal{G}}^{1/\alpha})}$$

and

$$\mathbf{eff}_{\mathrm{AP/seq}}(\eta,[0,T]) \sim \frac{1}{(1+\varepsilon_{\mathcal{G}}^{1/\alpha})}.$$

□

In the ideal setting of Proposition 3.1:

- The parallel efficiency of the adaptive parareal algorithm does not depend on *the final number of iterations*. This is in contrast to the classical version whose efficiency decreases with the final number of iterations $K(\eta)$ as $1/K(\eta)$.
- The efficiency behaves like $1 - o(\varepsilon_{\mathcal{G}})$ in the adaptive version, and $o(\varepsilon_{\mathcal{G}})$ rapidly goes to zero with $\varepsilon_{\mathcal{G}}$. As soon as $\varepsilon_{\mathcal{G}}$ becomes negligible with respect to 1, we will be in the range of full scalability.

We emphasize that, obviously, the above idealized setting will never hold in practice, but the result is interesting in its own right since it highlights that the cost of the fine solver is no longer the main obstacle for full scalability in the adaptive setting: the cost of the coarse solver becomes now the major obstruction towards full efficiency.

4 Numerical tests

4.1 Guidelines for a practical implementation

Practical choice of ζ_k^N Formula (12) of the convergence analysis of section 2.3 gives an estimate for ζ_k^N that one could in principle use for the implementation. However, these tolerances may not be optimal because they are derived from a theoretical convergence analysis based on abstract conditions for the coarse and fine solvers. This was confirmed during our numerical tests where we observed that using estimates (12) for ζ_k^N did not deliver satisfactory enough results. This is the reason why it is necessary to devise a practical rule to set ζ_k^N. We have explored the following choice: if η is the final target accuracy, the classical parareal algorithm is usually run with a solver that delivers a slightly higher accuracy, say $\eta/2$. Assume that the classical algorithm converges in $K_{\mathrm{CP}}(\eta) = K$ iterations. We propose to build the tolerances of ζ_k^N in such a way to target that $K_{\mathrm{AP}}(\eta) = K_{\mathrm{CP}}(\eta)$ and such that the cost of the last fine propagation is of the order of the sum of the previous ones. This motivates to set

$$\zeta_k^N = \begin{cases} \varepsilon_{\mathcal{G}}^{1-\frac{k+1}{K}} \left(\frac{\eta}{2}\right)^{\frac{k+1}{K}}, & \text{if } k < K \\ \eta/2, & \text{if } k \geq K. \end{cases}$$

The numerical example of the next section uses these tolerances.

Load balancing: For simplicity of exposition, the algorithm has so far been discussed for $\underline{N}$ subintervals of uniform size ΔT. However, this decomposition may lead to a task imbalance because some time intervals may have more complex dynamics than others, requiring more degrees of freedom, thus more computational time. In order to balance tasks as efficiently as possible, we dynamically adapt the size of the $\underline{N}$ subintervals in a way to have the fine solver propagations as balanced as possible among processors.

4.2 Results for a stiff ODE: the Brusselator system

We consider the brusselator system

$$\begin{cases} x' = A + x^2 y - (B+1)x \\ y' = Bx - x^2 y, \end{cases}$$

with initial condition $x(0) = 0$ and $y(0) = 1$. This is a stiff ODE that models a chain of chemical reactions. It was already studied in a previous work on the parareal algorithm (see [15]). The system has a fixed point at $x = A$ and $y = B/A$ which becomes unstable when $B > 1 + A^2$ and leads to oscillations. We place ourselves in this oscillatory regime by setting $A = 1$ and $B = 3$. The dynamics present large velocity variations in some time subintervals, making the use of adaptive time-stepping schemes particularly desirable for an appropriate treatment of the transient.

For the coarse solver, we set

$$\varepsilon_{\mathcal{G}} = 0.1,$$

and use an explicit Runge Kutta method of order 5 with an adaptive time-stepping (see [9]). For the fine solver, we use the implicit Runge-Kutta method of the Radau IIA family of order 5 with adaptive time-stepping (see [17, 18]). Both integrators are available in the ODE integration library of Scipy[1] which we have used to produce our results.

As already discussed, the target accuracies η_k^N should be ensured by rigorous a posteriori error estimators. These type of estimators is unfortunately not available in the Scipy library and we are not aware of any mainstream library with this capability, so we have built a priori a "chart" mapping accuracies of the solver against certain tolerance parameters of the library. Once the chart is prepared, we run our parareal algorithm and adapt dynamically the parameters of our library solver to obtain desired accuracy η_k^N.

We use formula (14) to compare the speed-up of the classical and adaptive parareal algorithm in terms of the number of operations involved in the numerical solution (communication delays have not been taken into account). For the costs g_k^N and f_k^N, we take into account:

- the number of time steps (which is adaptively increased as we tightned the accuracy),
- the number of right-hand side evaluations,
- for the fine solver, we additionally count the number of evaluations of the Jacobian matrix and of the number of linear system inversions.

In Figure 1, we plot the obtained speed-up for different configurations:

- the final time T varies from 100 to 900,
- the final target accuracy is $\eta = 10^{-6}$ or $\eta = 10^{-8}$,

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html`

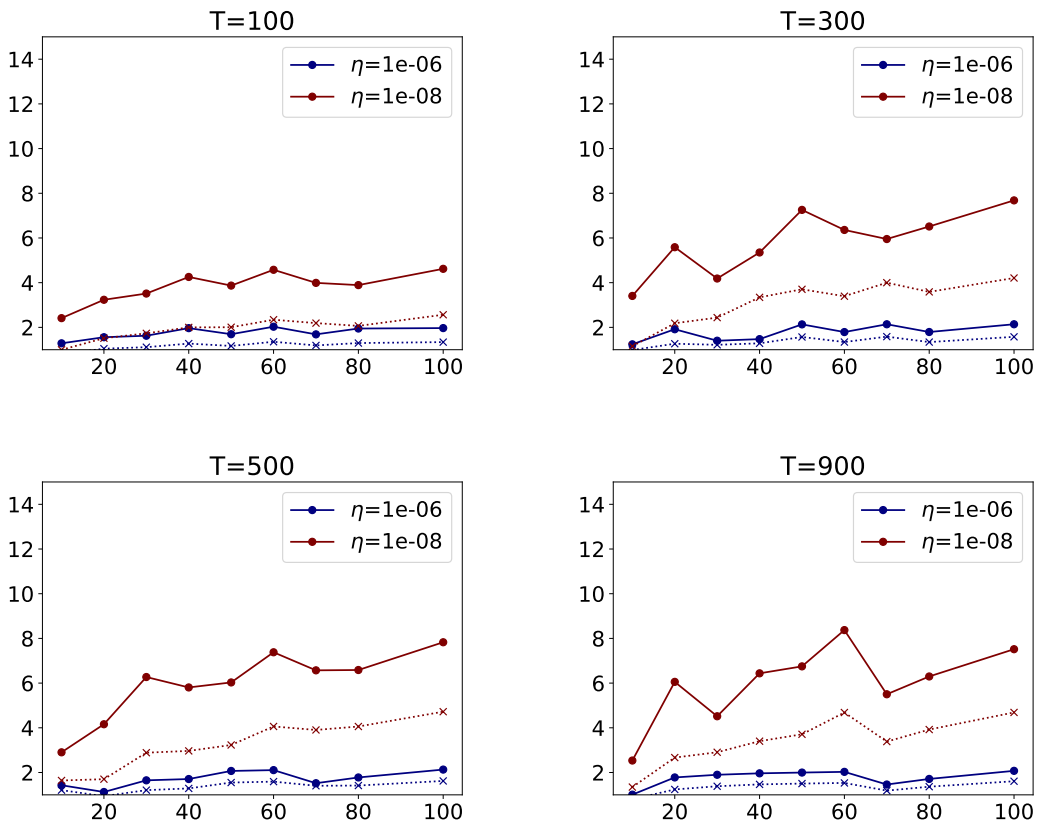

Figure 1: Speed-up as a function of the number of processors $\underline{N}$. Dashed lines: classical parareal. Continuous lines: Adaptive parareal.

- the number of processors $\underline{N}$ varies from 10 to 100.

As anticipated in section 3, the speed-up of the adaptive parareal is always superior to the one of the classical parareal. We observe that the gain is marginal for a moderate accuracy ($\eta = 10^{-6}$) but it is about 2.5 times larger for $\eta = 10^{-8}$. Note that sometimes the speed-up does not increase monotonically as the number of processors $\underline{N}$ increases. Also, the speed-up generally increases with $\underline{N}$ but the increase is rather moderate. We believe that this behavior could be improved with a rigorous a posteriori error estimator.

The values significantly differ from the range of full scalability and we next explain why this is mainly due to the cost of the coarse solver. Since the problem is stiff and we consider relatively long time intervals, it has been necessary to use a sufficiently accurate coarse solver. This explains our choice of an explicit Runge-Kutta scheme of order 5. To illustrate the impact of its cost, let us fix $T = 500$, $\eta = 10^{-8}$ and $\underline{N} = 50$ (other parameters would yield similar conclusions). We compare the speed-up and efficiency when we count or do not count the cost of the coarse solver on Table 1. Obviously, when we do not count the cost of the coarse solver, the performance of both algorithms improves but it is particularly increased in the case of the adaptive version. If the cost of $\mathcal{G}$ was negligible, it would deliver a very satisfactory efficiency of 75.52%. This is five times larger than what the classical parareal would yield. This analysis illustrates that the major obstacle to achieve competitive scalabilities is no longer the cost of the fine solver like in the classical version, but the cost of the coarse propagator.

We next give some insight on the differences in the convergence behavior of both algorithms.

Speed-up	Classical parareal	Adaptive Parareal
With cost $\mathcal{G}$	4.06	7.38
Without cost $\mathcal{G}$	7.38	37.76

Efficiency	Classical parareal	Adaptive Parareal
With cost $\mathcal{G}$	8%	14.76%
Without cost $\mathcal{G}$	14.76%	75.52%

Table 1: Impact of the cost of the coarse solver. Speed-up and efficiency with $\underline{N} = 50$.

We fix $T = 10$, $\eta = 10^{-8}$ and $\underline{N} = 10$ and plot in Figure 2 the convergence history of the parareal solution in terms of:

- the errors of the fine solver at every fine time-step
- the maximum error of the parareal solution at the macro-intervals

$$\max_N \|u(T_N) - y_k^N\|$$

Note that the maximum error in the adaptive scheme steadily decreases to the desired accuracy whereas the error in the classical scheme degrades at iteration $k = 1$ before converging. This type of behavior has been observed for all other configurations and we conjecture that an important difference in accuracy between the coarse and the fine solver at early stages of the algorithm may be the cause. Finally, an inspection of the error of the fine solver shows that the adaptive algorithm succeeds to reduce the error at every time t in a much more uniform way than the classical algorithm.

5 Conclusions and perspectives

The new adaptive formulation of the parareal algorithm opens the door to improve significantly the parallel efficiency of the method provided that the cost of the coarse solver is moderate. The increasing target tolerances which have to be met at each step allows to use online stopping criteria involving *a posteriori* estimators. The developed methodology remains theoretical since we have not quantified the impact of communication delays between processors nor potential memory issues (note however that the load balancing is devised with the purpose of equilibrating tasks and memory). In the framework of the ANR project "Ciné-Para (ANR-15-CE23-0019)", we are working on these issues that are of a different level of theory and involve different collaborators. This will be the topic of another paper. In addition to this, several extensions based on the current findings are subject of ongoing works, in particular, the coupling of the adaptive parareal with adaptive space-time schemes and the coupling of parareal with internal iterative solvers like in the discussion of section A.3 of the appendix.

Acknowledgment

This work was funded by the ANR project "Ciné-Para" (ANR-15-CE23-0019).

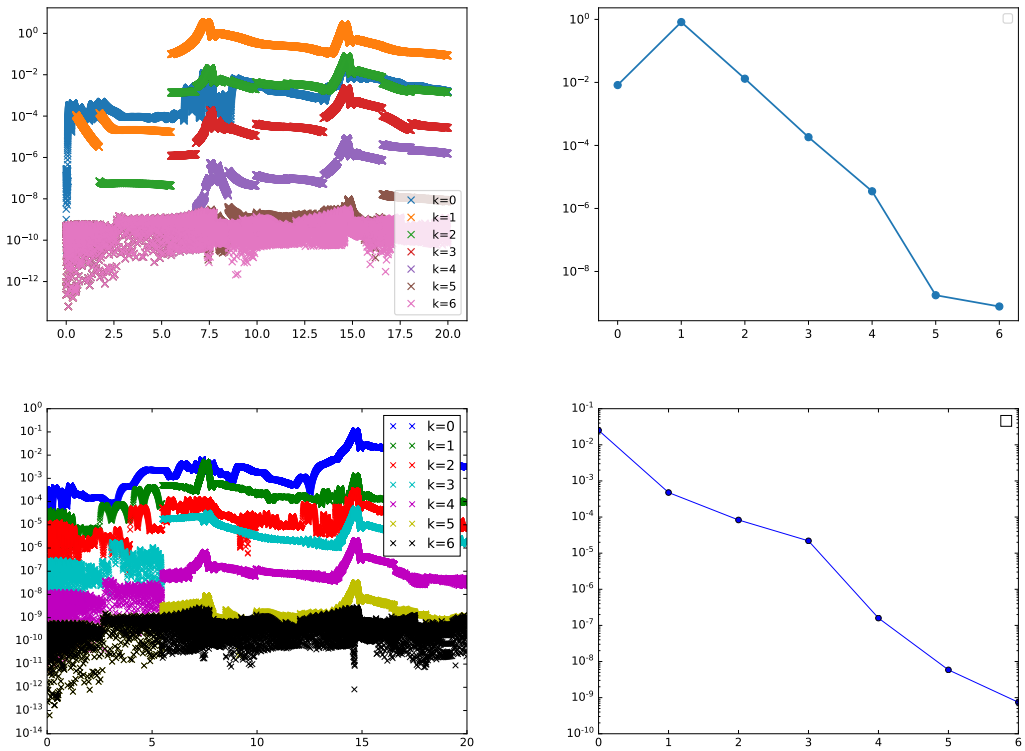

Figure 2: Convergence history of the errors. Top: classical parareal. Bottom: adaptive parareal. Left: Errors for $t \in [0, 10]$. Right: maximum error at each iteration k.

A Enriching the input information with previous iterations

Usually, solvers to realize $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ are built using only $\mathbb{I}_k^N = \{T_N, \Delta T, y_k^N\}$ as input information. We account for this idea with the notation

$$\mathcal{S}(T_N, \Delta T, y_k^N) \xrightarrow[(\mathbb{I}_k^N, \mathbf{cost}_k^N)]{} [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N],$$

and the numerical cost, denoted $\mathbf{cost}_k^N$, increase as ζ_k^N is tightened. In this section, we discuss how to enhance the gain in efficiency of the adaptive algorithm by discussing ways to increase the accuracy of the solver $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ across the iterations while maintaining the cost to realize it as independent as possible from ζ_k^N, N, and k. For this, one possibility is to enrich $\mathbb{I}_k^N$ with data produced during the previous parareal iterations (although it would of course be at the cost of increasing the storage requirements).

Let $\mathbb{P}_k^N$ denote the intermediate information that has been produced at iteration k between $[T_N, T_{N+1}]$ and by $\mathbb{P}_k := \cup_{N=0}^{N-1} \mathbb{P}_k^N$ all the information produced at step k. Using $\widetilde{\mathbb{I}}_k^N = \{\mathbb{I}_k^N, \mathbb{P}_k^{N-1}, \ldots, \mathbb{P}_k^0, \mathbb{P}_{k-1}, \ldots, \mathbb{P}_0\}$, as input information, the idea is to see whether it could be possible to find a solver $\mathcal{S}$ such that

$$\mathcal{S}(T_N, \Delta T, y_k^N) \xrightarrow[(\widetilde{\mathbb{I}}_k^N, \mathbf{cost})]{} [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N] \tag{19}$$

with a constant and small complexity **cost**. Note that using the enriched set of information $\widetilde{\mathbb{I}}_k^N$ means that we want to learn from the previous approximations of $u(T_{N+1})$ given by

$[\mathcal{E}(T_N, \Delta T, y_p^N), \zeta_p^N]$, $0 \leq p \leq k-1$, to start the current algorithm closer to $u(T_{N+1})$. After each parareal iteration, we thus improve the accuracy, without increasing the work for solving because we start from a better input, accumulated from the previous parareal iterations.

In the rest of this section, we describe three relevant scenarios where we can approximate $\mathcal{E}(T_N, \Delta T, y_k^N)$ by trying to build a scheme in the spirit of (19). The first two examples have already been presented in the literature and concern the coupling of parareal with spatial domain decomposition (section A.1) and with iterative high-order time integration schemes (section A.2). In these two cases, there is to date no complete convergence analysis since it remains to show that i) $\mathcal{E}(T_N, \Delta T, y_k^N)$ is approximated with accuracy ζ_k and ii) the **cost** of the solver is really constant through the parareal steps. In addition to these two applications, we mention a third scenario where the convergence analysis can be fully proven. It concerns the solution of time-dependent problems involving internal iterative schemes at every time step. This idea was first analyzed in [27] in a restricted setting. In section A.3, we give the main setting and defer the analysis for a forthcoming paper.

A.1 Parareal coupled with spatial domain decomposition

Here, we consider a solver $\mathcal{S} = \texttt{DDM}$ which involves spatial domain decomposition over $[T_N, T_{N+1}]$ [16, 1]. We assume that DDM involves a time discretization with a small time step $\delta t < \Delta T$. Let $\underline{n}$ be the number of time steps on each interval $[T_N, T_{N+1}]$ so that we have the relations $\Delta T = \underline{n}\delta t$ and $T = \underline{N}\Delta T = \underline{N}\ \underline{n}\delta t$ and the total number of time steps in $[0, T]$ is $\underline{\underline{n}} := \underline{N}\ \underline{n}$. The domain decomposition iterations act on a partition $\Omega = \cup_{l=1}^{L}\Omega_l$ of the domain. For $0 \leq n \leq \underline{n}$, we denote by $u_k^{N,n,j}$ the solution produced by DDM at time $t = T_N + n\delta t$ after $j \geq 0$ domain decomposition iterations. The notation J^* will denote the last iteration (fixed according to some stopping criterion). At $j = 0$, these iterations need to be initialized at the interfaces $\partial\Omega_l$, $1 \leq l \leq L$. The idea explored in, e.g., [16, 1], is to take the values $u_{k-1}^{N,n,J^*}|_{\partial\Omega_l}$ at these interfaces as a starting guess for $0 \leq n \leq \underline{n}$ so that

$$\widetilde{\mathbb{I}}_k^N = \{\mathbb{I}_k^N, \{u_{k-1}^{N,n,J^*}|_{\partial\Omega_l}, 0 \leq n \leq \underline{n}\}\}.$$

From [16, 1], there is numerical evidence that the computations of $\texttt{DDM}(T_N, \Delta T, y_k^N)$ using $\widetilde{\mathbb{I}}_k^N$ yield $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$ after a reduced number of iterations J^* which is independent of k. Thus **cost** would be kept constant and

$$u_k^{N,\underline{n},J^*} = \texttt{DDM}(T_N, \Delta T, y_k^N) \xrightarrow[(\widetilde{\mathbb{I}}_k^N, \mathbf{cost})]{} [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N],$$

where the above "**cost**" is much small than the cost of the fine solver.

A.2 Parareal coupled with iterative high-order time integration schemes

Spectral Deferred Correction (SDC, [10]) is an iterative time integration scheme. Starting from an initial guess of $u(t)$ at discrete points, the method adds successive corrections to this guess. The corrections are found by solving an associated evolution equation. Under certain conditions, the correction at every step increases by one the accuracy order of the time discretization.

We carry here a simplified discussion on how to build $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$ when $\mathcal{S} = \texttt{SDC}$ and connect it to the so-called Parallel Full Approximation Scheme in Space-Time (PFASST, [24, 11]). For a given time interval $[T_N, T_{N+1}]$, let us consider its $\underline{n}+1$ associated Gauss-Lobatto points $\{t_{N,n}\}_{n=0}^{\underline{n}}$ and quadrature weights $\{\omega_n\}_{n=0}^{\underline{n}}$. The Gauss-Lobatto points are such that $t_{N,0} = T_N$ and $t_{N,\underline{n}} = T_{N+1}$. Let us denote $u_k^{N,n,j}$ the approximation of $u(t_{N,n})$ at parareal

iteration k after $j \geq 0$ SDC iterations. Assuming that one uses an implicit time-stepping scheme to solve the corrector equations involved in this method, $u_k^{N,n+1,j}$ is given by

$$\begin{aligned} u_k^{N,n+1,j} &= u_k^{N,n,j} + (t_{N,n+1} - t_{N,n}) \left(\mathcal{A}(t_{N,n+1}, u_k^{N,n+1,j}) - \mathcal{A}(t_{N,n+1}, u_k^{N,n+1,j-1}) \right) \\ &\quad + \sum_{m=0}^{\underline{n}} \omega_m \mathcal{A}(t_{N,m}, u_k^{N,m,j-1}), \quad 1 \leq j,\ 0 \leq n \leq \underline{n} - 1, \\ u_k^{N,0,j} &= y_k^N \\ u_k^{N,n,0} &\text{ given for } 0 \leq n \leq \underline{n}. \end{aligned}$$

To speed-up computations, one of the key elements is the choice of the starting guesses $u_k^{N,n,0}$, $0 \leq n \leq \underline{n}$. Without entering into very specific details, the PFASST algorithm is a particular instantiation of the above scheme when $J^* = 1$ and $u_k^{N,n,0}$ uses information produced at the previous parareal iteration $k-1$. Therefore, PFASST falls into the present framework in the sense that it produces

$$u_k^{N,\underline{n},1} = \texttt{SDC}(T_N, \Delta T, y_k^N)$$

with

$$\widetilde{\mathbb{I}}_k^N = \{T_N, \Delta T, y_k^N, \mathbb{P}_{k-1}\}$$

and it is expected that $u_k^{N,\underline{n},1} = [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$.

An additional component of PFASST is that the algorithm also tries to improve the accuracy of the coarse solver $\mathcal{G}$ using SDC iterations built with $\widetilde{\mathbb{I}}_N^k$. This has not been taken into account in our adaptive parareal algorithm (11).

A.3 Coupling with internal iterative schemes

Any implicit discretization of problem (1) leads to discrete linear or nonlinear systems of equations which are often solved with iterative schemes. When they are involved as internal iterations within the parareal algorithm, one could try to speed them up by building good initial guesses based on information from previous parareal iterations. We illustrate this idea in the simple case where:

- $\mathcal{A}(t, \cdot)$ is a linear differential operator in $\mathbb{U}$ complemented with suitable boundary conditions,
- we use an implicit Euler scheme for the time discretization.

A sequential solution of problem (1) with the implicit Euler scheme goes as follows. At each time $t^{N,n} = T_N + n\delta t$, the solution $u(t^{N,n})$ is approximated by the function $u^{N,n} \in \mathbb{U}$ which is itself the solution to

$$\mathcal{B}(u^{N,n}) = g^{N,n},$$

where $g^{N,n} = u^{N,n-1} + \delta t f(t^{N,n})$ and

$$\mathcal{B}(v) := v + \delta t \mathcal{A}(t^{N,n}, v), \quad \forall v \in \mathbb{U}.$$

Note that $\mathcal{B}$ depends on time but our notation does not account for it in order not to overload the notations.

After discretization of $\mathbb{U}$, the problem classically reduces to solving a linear system of the form

$$\mathbf{B}\bar{u}^{N,n} = \bar{g}^{N,n},$$

for the unknown $\bar{u}^{N,n}$ in some discrete subspace S of $\mathbb{U}$. Usually, the above system is solved either by means of a conjugate gradient method or by a Richardson iteration of the form

$$\begin{cases} \bar{u}^{N,n,j} = (\mathbf{Id} + \omega \mathbf{PB})\bar{u}^{N,n,j-1} + \omega \mathbf{P}\bar{b}, \quad j \geq 1 \\ \bar{u}^{N,n,0} \in S \text{ given.} \end{cases}$$

Here, ω is a suitably chosen relaxation parameter and $\mathbf{P}$ can be seen as a pre-conditioner. The internal iterations j are stopped whenever a certain criterion is met (it could be an a posteriori estimator) and we denote by $J_{N,n}$ their final number. Obviously, $J_{N,n}$ depends on the starting guess for which a usual choice is to take the solution at the previous time, that is

$$\bar{u}^{N,n,0} = \bar{u}^{N,n-1,J_{N,n-1}}.$$

In order to achieve our goal, i.e. maintaning a low cost while increasing the accuracy at each parareal step, we can now reuse information from previous parareal iterations for the starting guess. In [27], two options are explored. The first is

$$\begin{cases} \bar{u}_k^{N,n,0} = \bar{u}_k^{N,n-1,J_{N,n-1,k}}, & \text{if } k = 0 \\ \bar{u}_k^{N,n,0} = \bar{u}_{k-1}^{N,n,J_{N,n,k-1}}, & \text{if } k \geq 1, \end{cases}$$

and the second, less natural choice,

$$\begin{cases} \bar{u}_k^{N,n,0} = \bar{u}_k^{N,n-1,J_{N,n-1,k}}, & \text{if } k = 0 \\ \bar{u}_k^{N,n,0} = \bar{u}_{k-1}^{N,n,J_{N,n,k-1}} + \bar{u}_k^{N,n-1,J_{N,n,k}} - \bar{u}_{k-1}^{N,n-1,J_{N,n-1,k-1}}, & \text{if } k \geq 1. \end{cases}$$

In the first case, we take over the internal iterations at the point where they were stopped in the previous parareal iteration $k-1$. In addition to this, in the second case, the term $u_k^{N,n-1,J_{N,n,k}} - u_{k-1}^{N,n-1,J_{N,n-1,k-1}}$ tries to better take the dynamics of the process into account. Note that the use of solutions that have been produced in the previous parareal iterations is at the expense of additional memory requirements. It might also be at the cost of a certain increase in the complexity locally at certain times. However, [27] shows in a restricted setting that these starting guesses (in particular the second) have interesting potential to enhance the speed-up of the parareal algorithm. A general theory on this aspect will be presented in a forthcoming work.

References

[1] S. Aouadi, D. Q. Bui, R. Guetat, and Y. Maday, *Convergence analysis of the coupled parareal-schwarz waveform relaxation method*, 2019. In preparation.

[2] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, *Parallel-in-time molecular-dynamics simulations*, Physical Review E, 66 (2002), p. 057701.

[3] G. Bal and Y. Maday, *A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put*, Recent developments in domain decomposition methods, 23 (2002), pp. 189–202.

[4] G. Bal and Q. Wu, *Symplectic parareal*, in Domain decomposition methods in science and engineering XVII, Springer, 2008, pp. 401–408.

[5] Bal, G., *Parallelization in time of (stochastic) ordinary differential equations*, 2003. Preprint, http://www.columbia.edu/ gb2030/PAPERS/paralleltime.pdf.

[6] K. Carlberg, L. Brencher, B. Haasdonk, and A. Barth, *Data-driven time parallelism via forecasting*, SIAM Journal on Scientific Computing, 41 (2019), pp. B466–B496.

[7] X. Dai, C. Le Bris, F. Legoll, and Y. Maday, *Symmetric parareal algorithms for hamiltonian systems*, ESAIM: Mathematical Modelling and Numerical Analysis, 47 (2013), pp. 717–742.

[8] X. Dai and Y. Maday, *Stable parareal in time method for first- and second-order hyperbolic systems*, SIAM J. Sci. Comput., 35 (2013), pp. A52–A78.

[9] J. R. Dormand and P. J. Prince, *A family of embedded runge-kutta formulae*, Journal of computational and applied mathematics, 6 (1980), pp. 19–26.

[10] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT Numerical Mathematics, 40 (2000), pp. 241–266.

[11] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Communications in Applied Mathematics and Computational Science, 7 (2012), pp. 105–132.

[12] C. Farhat and M. Chandesris, *Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid–structure applications*, International Journal for Numerical Methods in Engineering, 58 (2003), pp. 1397–1434.

[13] M. Gaja and O. Gorynina, *Parallel in time algorithms for nonlinear iterative methods*, 2018. To appear in ESAIM Proceedings of CEMRACS 2016 – Numerical Challenges in Parallel Scientific Computing.

[14] M. J. Gander, *50 years of time parallel time integration*, in Householder Symposium XIX June 8-13, Spa Belgium, 2015, p. 81.

[15] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm*, in Domain Decomposition Methods in Science and Engineering XVII, Springer, 2008, pp. 45–56.

[16] R. Guetat, *Méthode de parallélisation en temps: Application aux méthodes de décomposition de domaine*, PhD thesis, Paris VI, 2012.

[17] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, vol. 14, 01 1996.

[18] E. Hairer and G. Wanner, *Stiff differential equations solved by radau methods*, Journal of Computational and Applied Mathematics, 111 (1999), pp. 93 – 111.

[19] J. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps pararéel*, C. R. Acad. Sci. Paris, (2001). t. 332, Série I, p. 661-668.

[20] Y. Maday, M. Ronsquist, E., and G. Staff, *The parareal in time algorithm: Basics, stability analysis and more*, in Staff PhD Thesis, see below (2006), pp. 653–663.

[21] Y. Maday, J. Salomon, and G. Turinici, *Monotonic parareal control for quantum systems*, SIAM Journal on Numerical Analysis, 45 (2007), pp. 2468–2482.

[22] Y. Maday and G. Turinici, *The Parareal in Time Iterative Solver: a Further Direction to Parallel Implementation*, in Domain Decomposition Methods in Science and Engineering, Springer Berlin Heidelberg, 2005, pp. 441–448.

[23] M. Minion, *A hybrid parareal spectral deferred corrections method*, Comm. App. Math. and Comp. Sci., 5 (2010).

[24] M. Minion, *A hybrid parareal spectral deferred corrections method*, Communications in Applied Mathematics and Computational Science, 5 (2011), pp. 265–301.

[25] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht, *Interweaving PFASST and parallel multigrid*, SIAM Journal on Scientific Computing, 37 (2015), pp. S244–S263.

[26] M. L. Minion, A. Williams, T. E. Simos, G. Psihoyios, and C. Tsitouras, *Parareal and spectral deferred corrections*, in AIP Conference Proceedings, vol. 1048, 2008, p. 388.

[27] O. Mula, *Some contributions towards the parallel simulation of time dependent neutron transport and the integration of observed data in real time*, PhD thesis, Paris VI, 2014.

[28] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Commun. ACM, 7 (1964), pp. 731–733.

[29] A. Quarteroni and A. Valli, *Domain decomposition methods for partial differential equations*, Von Karman institute for fluid dynamics, 1996.

[30] A. Toselli and O. Widlund, *Domain decomposition methods: algorithms and theory*, vol. 3, Springer, 2005.