A SCALABLE ADAPTIVE PARAREAL ALGORITHM WITH ONLINE STOPPING CRITERION

Yvon Maday, Olga Mula

A SCALABLE ADAPTIVE PARAREAL ALGORITHM WITH ONLINE STOPPING CRITERION*

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 significant improvement of the parallel efficiency of the parareal in time method, an iterative predictor-corrector algorithm. This is achieved by first reformulating the algorithm in a rigorous infinite dimensional functional space setting. We then formulate implementable versions where time dependent subproblems are solved at *increasing accuracy* across the parareal iterations (in opposition to the classical version where the subproblems are solved at a fixed high accuracy). Aside from the important improvement in parallel efficiency and as a natural by product, the new approach provides a rigourous online stopping criterion with a posteriori error estimators and the numerical cost to achieve a certain final accuracy is designed to be near-minimal. We illustrate the gain in efficiency of the new approach on simple numerical experiments. In addition to this, we discuss the potential benefits of reusing information from previous parareal iterations to enhance efficiency even more.

Key words. domain decomposition; parareal in time algorithm; parallel efficiency; convergence rates; inexact fine solver; a posteriori estimators

AMS subject classifications. : 65M12, 65N55, 65Y05, 65Y20.

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. For a given Partial Differential Equation (PDE), one of the main ideas to parallelize a simulation is to break the problem into subproblems defined over subdomains of a partition of the original domain. 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 [25] or [26] 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 [24]) during which several algorithms have been explored (see [12] 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 [15] and has been well accepted by the community because it is easily applicable to a relatively large spectrum of problems. (Some specific difficulties are

*Submitted to the editors April 29, 2018.

Funding: This work was funded by ANR Ciné-Para (ANR-15-CE23-0019).

*Sorbonne Université, Université Paris-Diderot SPC, CNRS, Laboratoire Jacques-Louis Lions, LJLL, F-75005 Paris. (email: maday@ann.jussieu.fr)

†Institut Universitaire de France

‡Division of Applied Mathematics, Brown University, Providence RI, USA.

§Université Paris-Dauphine, PSL Research University, CNRS, UMR 7534, CEREMADE, 75016 Paris, France (email: mula@ceremade.dauphine.fr)

nevertheless encountered on certain types of PDEs as reported in, e.g., [7, 10] for hyperbolic systems or [4, 6] 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. In its classical formulation, subproblems over small time subdomains with the same accurate resolution as the one used to produce, with a sequential solver, the targeted accuracy, are solved at each iteration.

In this paper, we identify these high resolution computations as the main obstruction to achieve full parallel efficiency and propose an algorithm where the accuracy of these computations is increased across the iterations. For this, we start in section 2 by formulating an idealized version of the parareal algorithm in an infinite dimensional function space. This version will not be implementable but its rate of convergence will serve as a benchmark for a subsequent implementable version which is built in such a way that its convergence is kept close to the one of benchmark at a near-minimal numerical cost. This allows to identify the minimal accuracies required at every step in the solution of the subproblems over small time subdomains. Compared to the classical version of the algorithm, the new approach presents several important advantages. First, we prove both theoretically (in section 3) and in simple numerical examples (section 5) that its parallel efficiency is significantly superior to the traditional version and, in some cases, it can even be close to the ideal value of one. Second, our convergence analysis provides an online stopping criterion when the target accuracies are estimated with a posteriori error estimators (this point will be explored more extensively in a forthcoming work). Third, the numerical cost to achieve a certain final accuracy is designed to be near-minimal.

In setion 4.1, we explain how to satisfy in practice the required tolerances with adaptive techniques. In addition to adaptivity, one can expect to gain in efficiency even further by reusing information from previous steps. This idea is actually the main point of contact between this work and previous contributions from the literature which have incorporated it with encouraging results in a variety of contexts. Among the most relevant ones stand the coupling of the parareal algorithm with spatial domain decomposition (see [18, 14, 1]), the combination of the parareal algorithm with iterative high order methods in time like spectral deferred corrections (see [22, 19, 21]) and, in a similar spirit, applications of the parareal algorithm to solve optimal control problems (see [18, 17]). In section 4.2, 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 [23] and a more complete analysis will be proposed in a forthcoming work.

2. A scalable adaptive parareal algorithm.

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 in $[0,T]\times\mathbb{U}$. 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 [11]).

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 $\mathcal{S}(t,s,w) = [\mathcal{E}(t,s,w);\zeta]$ and we shall assume that the numerical complexity to realize $[\mathcal{E}(t,s,w);\zeta]$ is[1]

$$\mathbf{cost}_{\mathcal{S}}(\zeta,s) = s\zeta^{-1/\alpha}, \tag{3}$$

i.e. proportional to the size of the time evolution window. The cost increases when ζ get smaller like $\zeta^{-1/\alpha}$, where $\alpha > 0$ accounts for the order of the numerical scheme which is used to discretize the time variable and in particular the three ingredients above.

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

[1] In fact, $\mathbf{cost}_{\mathcal{S}}(\zeta,s)$ scales proportional to $s\zeta^{-1/\alpha}$ but for simplicity we assume that the proportionality constant is one which can be seen as normalization of the unit cost.

goal of the parareal in time algorithm is to build an approximation $\tilde{u}(T_N)$ of $u(T_N)$ such that

$$\max_{0\leq N\leq \underline{N}} \|u(T_N) - \tilde{u}(T_N)\| \leq \eta. \tag{4}$$

The classical way to achieve this is to set

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

where $\mathcal{S}_{\rm seq}$ is some sequential solver in $[0,T]$ with $\zeta = \eta/(T(1+\|u(0)\|))$ in (2). However, this comes at the cost of solving over the whole time interval $[0,T]$ and we have

$$\mathbf{cost}_{\rm seq}(\eta, [0,T]) = T\left(\frac{\eta}{T(1+\|u(0)\|)}\right)^{-1/\alpha}.$$

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 intervals 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 $s \in [0, T-t]$,

$$\mathcal{G}(t,s,x) = [\mathcal{E}(t,s,x), \varepsilon_{\mathcal{G}}] \quad \Leftrightarrow \quad \|\delta\mathcal{G}(t,s,x)\| \leq s(1+\|x\|)\varepsilon_{\mathcal{G}} \tag{5a}$$

$$\|\mathcal{G}(t,s,x) - \mathcal{G}(t,s,y)\| \leq (1+C_c s)\|x-y\|, \tag{5b}$$

$$\|\delta\mathcal{G}(t,s,x) - \delta\mathcal{G}(t,s,y)\| \leq C_d s \varepsilon_{\mathcal{G}} \|x-y\| \tag{5c}$$

The idealized version of the algorithm then 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

$$\begin{cases} y_0^{N+1} = \mathcal{G}(T_N, \Delta T, y_0^N), & 0 \leq N \leq \underline{N}-1 \\ y_{k+1}^{N+1} = \mathcal{G}(T_N, \Delta T, y_{k+1}^N) + \mathcal{E}(T_N, \Delta T, y_k^N) \\ \qquad\qquad - \mathcal{G}(T_N, \Delta T, y_k^N), & 0 \leq N \leq \underline{N}-1, \quad k \geq 0, \\ y_0^0 = u(0). \end{cases} \tag{6}$$

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 (6), 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 $\underline{N}$ 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 $N = 0, \dots, \underline{N}$, 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. In this case, we can estimate the final number of iterations $K(\eta)$ using the following convergence rate result. To this end, 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 1. *If $\mathcal{G}$ and $\delta\mathcal{G}$ satisfy Hypothesis (5), then,*

$$\max_{0 \leq N \leq \underline{N}} \|u(T_N) - y_k^N\| \leq \mu \frac{\tau^{k+1}}{(k+1)!}, \quad \forall k \geq 0. \tag{7}$$

Proof. The proof is in the spirit of existing results from the literature (see [15, 5, 16, 13]) but it is instructive to give it for subsequent developments in the paper. We introduce the following quantities

$$\begin{cases} \alpha & := C_d \varepsilon_{\mathcal{G}} \Delta T \\ \beta & := 1 + C_c \Delta T \\ \gamma & := \Delta T \varepsilon_{\mathcal{G}} \max_{0 \leq N \leq \underline{N}} (1 + \|u(T_N)\|) \end{cases} \tag{8}$$

as shorthand notations for the proof.

If $k = 0$, using definition (6) 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 (5a) and (5b) to derive the second to last inequality.

For $k \geq 1$, starting from (6), 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 (5b), (5c), we derive

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

Following [13], we consider the sequence $(e_k^N)_{N,k \geq 0}$ defined recursively as follows. For $k = 0$,

$$e_0^N = \begin{cases} 0, & \text{if } N = 0 \\ \beta e_0^{N-1} + \gamma, & \text{if } N \geq 1 \end{cases} \tag{9}$$

and for $k \geq 1$,

$$e_k^N = \begin{cases} 0, & \text{if } N = 0 \\ \alpha e_{k-1}^{N-1} + \beta e_k^{N-1}, & \text{if } N \geq 1. \end{cases} \tag{10}$$

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 (9) and (10) 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

$$\frac{1}{(1-\beta\xi)^{k+2}} = \sum_{j \geq 0} \binom{k+1+j}{j} \beta^j \xi^j \tag{11}$$

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 (8) 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 the only step which is not sharp in the above proof is the step where $1 - \xi$ is replaced by $1 - \beta\xi$. Therefore, bound (7) yields a good estimate of the number $K(\eta)$ of iterations to reach a final accuracy η, namely

$$K(\eta) = \min \left\{ k \geq 0 \ : \ \mu \frac{\tau^{k+1}}{(k+1)!} \leq \eta \right\}.$$

As a result, we achieve (4) with $\tilde{u}(T_N) = y_{K(\eta)}^N$.

Finally, we also note 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.

2.3. Feasible realizations of the parareal algorithm. Feasible versions of algorithm (6) 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

$$
\begin{cases}
y_0^{N+1} = \mathcal{G}(T_N, \Delta T, y_0^N), & 0 \le N \le \underline{N} - 1 \\
y_{k+1}^{N+1} = \mathcal{G}(T_N, \Delta T, y_{k+1}^N) + [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N] \\
\qquad\qquad - \mathcal{G}(T_N, \Delta T, y_k^N), & 0 \le N \le \underline{N} - 1,\ k \ge 0, \\
y_0^0 = u(0).
\end{cases} \tag{12}
$$

Since no feasible version will converge at a better rate than (7), 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

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

THEOREM 2. *Let $\mathcal{G}$ and $\delta\mathcal{G}$ satisfy Hypothesis (5). 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*

$$
\zeta_p^N \le \zeta_p := \frac{\varepsilon_{\mathcal{G}}^{p+2}}{(p+1)!\,\nu_p}, \tag{13}
$$

then the $(y_k^N)_N$ of the feasible parareal scheme (12) satisfy

$$
\max_{0 \le N \le \underline{N}} \|u(T_N) - y_k^N\| \le \mu \frac{\tilde{\tau}^{k+1}}{(k+1)!}, \tag{14}
$$

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 (7) and (14), 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 (13). As a final remark, we note that we can also interpret both convergence bounds (7) and (14) 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 exponentially with k.

Proof. The proof follows the same lines as the one for theorem 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 1. For $k \geq 1$, using (12), 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 (5b), (5c) 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 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 (11), 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 (8) 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. 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 (5) 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])

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

Compared to this classical version of the parareal algorithm, the approach with dynamically updated tolerances (12) 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 $\mathcal{F}(0, T_N, u(0))$.
2. The accuracies $\varepsilon_{\mathcal{G}}$ and ζ_k can be estimated with some appropriate *a posteriori* estimators so it is possible to have an *online stopping criterion*.
3. If one wants to approximate with a final accuracy η, the new approach *balances numerical costs* in a near-optimal way because the accuracy ζ_k^N is rather low for small k and it is increased by a factor of roughly $\varepsilon_{\mathcal{G}}$ at every new step. As will be explained latter, this brings a clear gain in parallel efficiency with respect to the classical algorithm which uses the fine solver at every parareal iteration. An estimation of the actual gain is given in the next section.

3. Parallel efficiency. In this section, we estimate the parallel speed up and efficiency of the method. The speed up is defined as the ratio

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

between the cost to run a sequential fine solver achieving a target accuracy η with the cost to run a scalable adaptive parareal algorithm providing at the end the same target accuracy η.

The 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:

$$\mathbf{eff}_{\mathrm{SA/seq}}(\eta,[0,T]) \coloneqq \frac{\mathbf{speed\text{-}up}_{\mathrm{SA/seq}}(\eta,[0,T])}{\underline{N}},$$

For our application, let us assume that the cost to realize $[\mathcal{E}(T_N,\Delta T,y_k^N),\zeta_k]$ follows the model given in (3), i.e., $\mathbf{cost}_{\mathcal{S}}(\zeta_k,\Delta T)\simeq \Delta T\zeta_k^{-1/\alpha}$ and $\alpha>0$ is linked to the order of the numerical scheme which is used to discretize the time variable. Note that α could actually depend on k, as we could change the nature of the scheme $[\mathcal{E}(T_N,\Delta T,y_k^N);\zeta_k^N]$ when k increases, but we stick to this simple model for clarity of exposition.

LEMMA 3. *If the cost of the coarse solver is negligible with respect to the cost of realizing* $[\mathcal{E}(T_N,\Delta T,y_k^N),\zeta_k]$ *for any* $k\geq 0$*, then*

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

Therefore

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

Before giving the proof of the lemma, several remarks/observations are in order. The fist is about the hypothesis that the cost of the coarse solver is negligible. We claim that this assumption is not as strong as it may appear. Indeed, this cost is negligible in front of, e.g., the last realization of the fine solver (remember that $\mathcal{G}(t,s,x)$ is some $[\mathcal{E}(t,s,x),\varepsilon_{\mathcal{G}}]$). This is due to the fact that the cost of the coarse solver after $K(\eta)$ iterations is

$$\mathbf{cost}(\mathcal{G}) = TK(\eta)(\varepsilon_{\mathcal{G}})^{-1/\alpha} \tag{15}$$

while the cost of the fine solver in the last parareal iteration is

$$\Delta T\zeta_{K(\eta)}^{-1/\alpha} \simeq \frac{T}{N}\left(\frac{(\varepsilon_{\mathcal{G}})^{K(\eta)+1}}{K(\eta)!}\right)^{-1/\alpha} \simeq \mathbf{cost}(\mathcal{G})\left(\frac{1}{NK(\eta)}\left[\frac{K(\eta)!}{(\varepsilon_{\mathcal{G}})^{K(\eta)}}\right]^{1/\alpha}\right) \tag{16}$$

and we note that the last factor between brackets is rapidly much larger than 1.

Another remark that follows from the lemma is that, *regardless of the final number of iterations*, $K(\eta)$, the parallel efficiency will always be larger than $1-o(\varepsilon_{\mathcal{G}})$, where $o(\varepsilon_{\mathcal{G}})$ rapidly goes to zero with $\varepsilon_{\mathcal{G}}$. This is in contrast to the plain parareal algorithm whose efficiency decreases with the final number of iterations $K(\eta)$ as $1/K(\eta)$. In addition, as soon as $\varepsilon_{\mathcal{G}}$ becomes negligible in front of 1, we will be in the range of full scalability.

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

$$\mathbf{cost}_{\mathrm{SA}}(\eta,[0,T]) = \Delta T\sum_{k=0}^{K(\eta)-1}\zeta_k^{-1/\alpha} \tag{17}$$

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 $\underline{K}=K(\eta)-1$:

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

we thus derive

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

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

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

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 propagation are negligible).

Since the accuracy that is obtained at the end of the parareal procedure (see (14)) is of the same order as the accuracy provided with plain parareal solver (see (7)), it follows that if we now take the last target accuracy $\zeta_{K(\eta)-1}^{-1/\alpha}$ of the scalable adaptive algorithm as the accuracy of the fine scheme in the plain parareal algorithm, the cost would be

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

Therefore, the complexity reduction with respect to the scalable adaptive parareal method is

$$\mathbf{reduction}_{\mathrm{SA/PP}}(\eta,[0,T])=\frac{\mathbf{cost}_{\mathrm{SA}}(\eta,[0,T])}{\mathbf{cost}_{\mathrm{PP}}(\eta,[0,T])}\sim\frac{1}{K(\eta)}(1+\varepsilon_{\mathcal{G}}^{1/\alpha})$$

In addition, since

$$\mathbf{reduction}_{\mathrm{PP/seq}}(\eta,[0,T])=\frac{\mathbf{cost}_{\mathrm{PP}}(\eta,[0,T])}{\mathbf{cost}_{\mathrm{seq}}(\eta,[0,T])}=\frac{K(\eta)}{\underline{N}},$$

the scalable adaptive algorithm reduces the complexity by a factor

$$\mathbf{reduction}_{\mathrm{SA/seq}}(\eta,[0,T])=\frac{\mathbf{cost}_{\mathrm{SA}}(\eta,[0,T])}{\mathbf{cost}_{\mathrm{seq}}(\eta,[0,T])}\sim\frac{1}{\underline{N}}(1+\varepsilon_{\mathcal{G}}^{1/\alpha})\sim\frac{1}{\underline{N}}$$

with respect to a sequential solution, as soon as $\varepsilon_{\mathcal{G}}$ is small enough. Therefore

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

and

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

□

4. Realization of $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$. In this section, we list different possibilities to build the approximations $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ in practice.

4.1. Adaptivity. Since the accuracy ζ_k^N should improve with k, the first natural ingredient to realize properly $[\mathcal{E}(T_N, \Delta T, y_k^N), \zeta_k^N]$ is to use adaptive refinement techniques. The implementation of adaptive refinements ultimately rests on the use of a posteriori error estimators and opens the door to an online local time step adaptation procedure in the parareal algorithm. In section 5, we present an application of the algorithm to two ODEs where we use a dyadic adaptive refinement strategy.

4.2. 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

$$\mathcal{S}(T_N, \Delta T, y_k^N) \xrightarrow[(\mathbb{I}_k^N, \Delta T \zeta_k^{-1/\alpha})]{} [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N].$$

In this section, we want to further enhance the gain in efficiency that the scalable adaptivity offers, by proposing 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

$$\mathcal{S}(T_N, \Delta T, y_k^N) \xrightarrow[(\widetilde{\mathbb{I}}_k^N, \mathbf{cost})]{} [\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N] \tag{18}$$

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 $[\mathcal{E}(T_N, \Delta T, y_p^N), \zeta_p^N]$, $0 \le p \le k-1$, 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 (18). The first two examples have already been presented in the literature and concern the coupling of parareal with spatial domain decomposition (section 4.2.1) and with iterative high-order time integration schemes (section 4.2.2). 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 [23] in a restricted setting. In section 4.2.3, we give the main setting and defer the analysis for a forthcoming paper since its full development would exceed the page limitations.

4.2.1. Parareal coupled with spatial domain decomposition. Here, we consider a solver $\mathcal{S} = \texttt{DDM}$ which involves spatial domain decomposition over $[T_N, T_{N+1}]$

[14, 1]. We assume that DDM involves a time discretization with 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 $\underline{\underline{n}} \coloneqq \underline{N}\ \underline{n}$. The domain decomposition iterations act on a partition $\Omega = \cup_{l=1}^{L}\Omega_l$ of the domain. For $0 \le n \le \underline{n}$, we denote by $u_k^{N,n,j}$ the solution produced by DDM at time $t = T_N + n\delta t$ after $j \ge 0$ 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 $\partial\Omega_l,\ 1 \le l \le L$. The idea explored in, e.g., [14, 1], is to take the values $u_{k-1}^{N,n,J^*}|_{\partial\Omega_l}$ at these interfaces as a starting guess for $0 \le n \le \underline{n}$ so that

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

From [14, 1], there is numerical evidence that the computations of $\mathtt{DDM}(T_N, \Delta T, y_k^N)$ 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

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

4.2.2. Parareal coupled with iterative high-order time integration schemes. Spectral Deferred Correction (SDC, [8]) 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 $\mathcal{S} = \mathtt{SDC}$ and connect it to the so-called Parallel Full Approximation Scheme in Space-Time (PFASST, [20, 9]). For a given time interval $[T_N, T_{N+1}]$, we 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 $t_{N,\underline{n}} = T_{N+1}$. We denote $u_k^{N,n,j}$ the approximation of $u(t_{N,n})$ at parareal iteration k after $j \ge 0$ 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

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

To speed-up computations, one of the key elements is the choice of the starting guesses $u_k^{N,n,0},\ 0 \le n \le \underline{n}$. 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

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

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 scalable adaptive parareal algorithm (12).

4.2.3. 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

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

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

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

Here, ω is a suitably chosen relaxation parameter and $\mathbf{P}$ can be seen as a preconditioner. 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 [23], 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, [23] 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.

5. A first numerical example. Some of the quantities involved in the convergence and parallel efficiency analysis are difficult to estimate in actual practice. We study the performance in two extreme cases. The first is a favorable scenario where the adaptive algorithm gives very high parallel efficiency. The second is a more involved example were the performance is degraded by two sources: a computationally expensive coarse solver and a task imbalance in the parallel computation of the approximations $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$.

An example with good efficiency: the circular trajectory: We consider the system

$$\begin{cases} x'(t) = -y(t), \\ y'(t) = x(t), \end{cases}$$

for $t \in [0, 3]$ and with initial condition $x(0) = 0$ and $y(0) = 1$. The trajectory is a circle of radius 1 centered at the origin (see figure 1). This example is admittedly very simple but it has been chosen because there is no subinterval where the dynamics become more involved. For the discussion, we set the target accuracy

$$\eta = 10^{-3}$$

and implement the adaptive parareal algorithm (12) with a number $\underline{N} = 8$ of processors. This choice sets the value $\Delta T = T/\underline{N} = 3/8 = 3.75.10^{-1}$. We work with a coarse solver $\mathcal{G}$ that uses an explicit Euler scheme with uniform time step ΔT. Its accuracy $\varepsilon_{\mathcal{G}}$ has been measured to be $\varepsilon_{\mathcal{G}} = 7.12.10^{-1}$. This estimation was done with a reference solver using an explicit Euler scheme with uniform time step $\delta t_{\text{ref}} = 5.10^{-4}$. It is taken as the reference sequential solution and is used in the following to compute approximation errors. Finer grids have been tried and do not change the results that are presented here.

To build $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$, we use an explicit Euler scheme whose time step δt is refined until the the accuracy ζ_k^N is reached. We have worked with a simple

dyadic refinement strategy: we start from $\delta t = \Delta T$ and then divide δt by two until the accuracy ζ_k^N is reached. More elaborated strategies could of course be considered. The accuracy of the approximation is evaluated by computing the error between the explicit Euler solver and the reference sequential solution at time T_{N+1}. However, note that this error estimation should ideally involve *a posteriori* error techniques in order to have a real online stopping criterion. This point, together with more elaborated refinement strategies, will be addressed in a forthcoming paper.

To satisfy $\eta = 10^{-3}$, we need 6 parareal iterations. This means that the index k ranges from 0 to 5 and $K(\eta) = 5$. It also means that we have built approximations $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$ up to index $K - 1 = 4$. We plot in figure 1 the trajectories and convergence errors. Due to the fact that the dynamics are very homogeneous in the whole interval $[0, T]$, for a given iteration k, the refinement strategy acts identically on all the time slices $[T_N, T_{N+1}]$ to compute $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$. Table 1 displays the time steps δt required at every iteration k and also the numerical cost of each computation. We have taken one step of the Euler explicit method as unit cost.

k	δt to compute $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$	$\mathbf{cost}([\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N])$
0	$\Delta T/2 \approx 1.9.10^{-1}$	2
1	$\Delta T/2^2 \approx 9.4.10^{-2}$	2^2
2	$\Delta T/2^4 \approx 2.3.10^{-2}$	2^4
3	$\Delta T/2^7 \approx 2.9.10^{-3}$	2^7
4	$\Delta T/2^9 \approx 7.3.10^{-4}$	2^9

Table 1: Time steps δt and cost to compute $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$ at each iteration k.

Taking the time to compute one step of the Euler method as our time unit, it follows from the results of the table that the computing time to run the algorithm is

$$\begin{aligned} T_{\mathrm{SA}}(\eta) &= \sum_{k=0}^{5} \sum_{N=0}^{\underline{N}-1} \mathbf{cost}([T_N, T_{N+1}]; \mathcal{G}) + \sum_{k=0}^{4} \mathbf{cost}([\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]) \\ &= \left(\sum_{k=0}^{5} \sum_{N=0}^{7} 1 \right) + (2 + 2^2 + 2^4 + 2^7 + 2^9) \\ &= 710. \end{aligned}$$

To evaluate the speed-up and parallel efficiency of the method, we compare this computing time with the one of a sequential explicit Euler scheme giving accuracy η at the minimal numerical cost. This requires to find the largest time step δt_{seq} that gives η accuracy at times $T_0,\ T_1, \ldots, T_{\underline{N}}$. Searching with a dyadic refinement strategy, we find that $\delta t_{\mathrm{seq}} = \Delta T/2^8$ yields and accuracy 1.44η and the next level $\delta t_{\mathrm{seq}} = \Delta T/2^9$ gives 0.35η so we take this value to carry the discussion. In this setting,

$$\mathbf{speed\text{-}up}_{\mathrm{SA}} = \frac{T_{\mathrm{seq}}(\eta)}{T_{\mathrm{SA}}(\eta)} = 5, \quad \mathbf{eff}_{\mathrm{SA}} = \frac{\mathbf{speed\text{-}up}_{\mathrm{SA}}}{\underline{N}} = 0.65,$$

and the time step of the last parareal iteration corresponds exactly to the time step of the sequential solver, namely $\delta t_{\mathrm{seq}} = \delta t_{\mathrm{SA}} = \Delta T/2^9$ (see table 1). This clearly illustrates the advantage of the proposed methodology: only the last parareal iteration has the fine resolution and expensive numerical cost of the sequential solver, the

previous iterations being at a cost which is almost negligible with respect to the last step. This is in contrast to the classical approach where all steps are done with fine resolution.

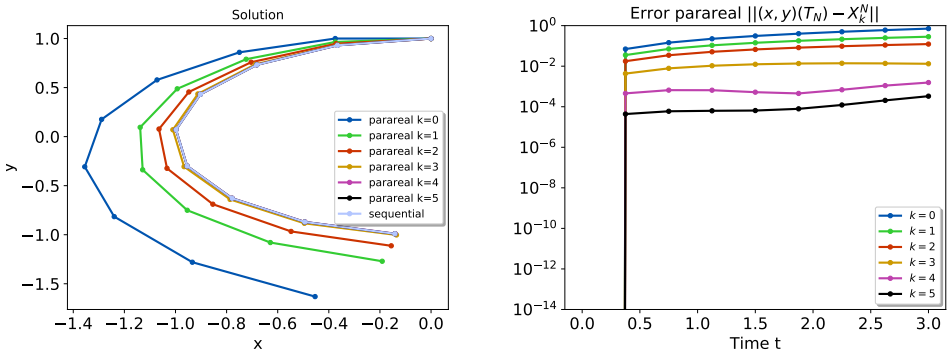

Figure 1: Trajectories and convergence history of the adaptive parareal algorithm

To compare the performance of the adaptive approach with the classical one, we work with a fine solver using an explicit Euler scheme with time step $\delta t = \Delta T/2^9$ at all parareal iterations. The algorithm reaches accuracy η after 5 iterations so we have

$$\begin{aligned} T_{\mathrm{PP}}(\eta) &= \sum_{k=0}^{4}\sum_{N=0}^{\underline{N}-1} \mathbf{cost}([T_N, T_{N+1}]; \mathcal{G}) + \sum_{k=0}^{3} \mathbf{cost}(\mathcal{F}(T_N, \Delta T, y_k^N)) \\ &= \left(\sum_{k=0}^{4}\sum_{N=0}^{7} 1\right) + 4.2^9 = 2088. \end{aligned}$$

As a result,

$$\mathbf{speed\text{-}up}_{\mathrm{PP}} = \frac{T_{\mathrm{seq}}(\eta)}{T_{AP}(\eta)} \approx 1.96, \quad \mathbf{eff}_{\mathrm{PP}} = \frac{\mathbf{speed\text{-}up}_{\mathrm{PP}}}{\underline{N}} \approx 0.25,$$

which means that, in this simple example, the adaptive version is about 2.5 more efficient than the classical implementation.

An example with obstructions: the brusselator system: As a second example, we consider the brusselator system

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

which models a chain of chemical reactions. This ODE was already studied in a previous work on the parareal algorithm (see [13]). 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 which is more challenging for computations because there are large velocity variations in some time subintervals. These dynamical heterogeneities were not present in the previous example, where the velocity module was constant in time. They are however to be expected in most

problems and entail different refinement levels from one subinterval to another. Our example will reveal that, in this case, there is need for a rebalancing strategy to preserve the high parallel efficiency of the algorithm.

We set $A = 1$ and $B = 3$ and the time interval is $[0, T]$ with $T = 18$. The initial condition is $x(0) = 0$ and $y(0) = 1$. The system oscillates around the fixed point with coordinates $(1, 3)$ performs two loops in $[0, 18]$. We show in figure 2 the first loop corresponding to $t \in [0, 12]$ (the second loop is not shown for the sake of clarity). In the figure, the separation between successive points is proportional to the velocity of the system and illustrates its strong variations.

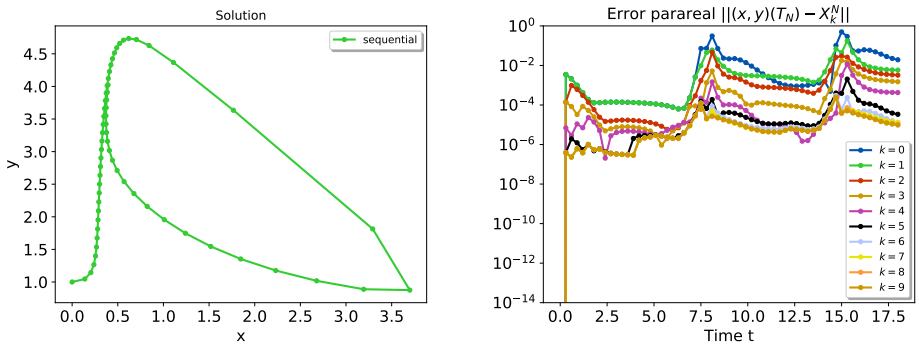

Figure 2: Left: Trajectory of the brusselator system with $A = 1$, $B = 3$ and over $[0, 12]$. Right: Convergence history of the adaptive parareal algorithm in the whole interval $[0, 18]$.

Our discussion follows the same lines as in the previous example. Here we take

$$\eta = 7.10^{-5}$$

and $\underline{N} = 60$ which sets the value $\Delta T = T/\underline{N} = 0.3$. We work with a coarse solver $\mathcal{G}$ that uses an explicit Runge-Kutta scheme of order 4 (RK4) with uniform time step ΔT. Its accuracy $\varepsilon_{\mathcal{G}}$ has been measured to be $\varepsilon_{\mathcal{G}} = 5.0.10^{-1}$. This estimation was done with a reference solver using an RK4 scheme with uniform time step $\delta t_{\text{ref}} = 10^{-5}$. It is taken as the reference sequential solution and is used in the following to compute the approximation errors. As in the previous example, finer time steps δt_{ref} have been tried and produce the same numerical results that are presented here.

To build $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$, we use an explicit RK4 scheme whose time step δt is dyadically refined (starting from $\delta t = \Delta T$) until the the accuracy ζ_k^N is reached.

To satisfy $\eta = 7.10^{-5}$, we need 7 parareal iterations so the index k goes from 0 to 6 and $K(\eta) = 6$. We plot in figure 2 the convergence errors. Since the dynamics are more complex at certain times, the refinements of the time steps δt to compute $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$ will vary depending on the time slices $[T_N, T_{N+1}]$. Table 2 displays for every k the number of time steps δt of different sizes in the $\underline{N} = 60$ intervals $[T_N, T_{N+1}]$.

The difference in the time steps on every time slice leads to an imbalance of the tasks in the algorithm. Indeed, for a given step k, each $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$ is computed in parallel for $N = 0, \ldots, \underline{N}$. Therefore some processors will finish the computation of $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$ later than others due to the heterogeneities in the numerical complexity. As a result, without any rebalancing of the tasks, the parallel

	Number of subintervals $[T_N, T_{N+1}]$ with time step δt				
k	$\delta t = \Delta T$	$\delta t = \Delta T/2$	$\delta t = \Delta T/2^2$	$\delta t = \Delta T/2^3$	$\delta t = \Delta T/2^4$
0	0	54	6	0	0
1	0	54	4	2	0
2	0	52	4	4	0
3	0	46	8	2	4
4	0	30	19	5	6
5	0	30	18	6	6

Table 2: Number ot time steps of different sizes δt at each iteration k.

efficiency is degraded. In the current example, taking the time to compute one step of the RK4 method as our time unit, the computing time to run the algorithm without any rebalancing is

$$\begin{aligned} T_{\mathrm{SA}}(\eta) &= \sum_{k=0}^{6} \sum_{N=0}^{\underline{N}-1} \mathbf{cost}([T_N, T_{N+1}]; \mathcal{G}) + \sum_{k=0}^{5} \max_{0 \le N \le \underline{N}-1} \mathbf{cost}([\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]) \\ &= \sum_{k=0}^{6} \sum_{N=0}^{59} \mathbf{cost}([T_N, T_{N+1}]; \mathcal{G}) + \sum_{k=0}^{5} \max_{0 \le N \le 59} \mathbf{cost}([\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]) \\ &= 7 * 60 + 2^2 + 2.2^3 + 3.2^4 = 488. \end{aligned}$$

In the above formula, the maximum over N comes as a consequence of the task imbalance. We compare this computing time with the one of the best sequential RK4 solver that gives η at the minimal numerical cost. Searching with the dyadic refinement strategy, we find that $\delta t_{\mathrm{seq}} = \Delta T/2^3$ yields an accuracy 2.2η and the next level $\delta t_{\mathrm{seq}} = \Delta T/2^4$ gives 0.14η so we select this value to carry the discussion. With this choice,

$$T_{\mathrm{seq}}(\eta) = 960,$$

and

$$\mathbf{speed\text{-}up}_{\mathrm{SA}} = \frac{T_{\mathrm{seq}}(\eta)}{T_{\mathrm{SA}}(\eta)} \approx 1.96, \quad \mathbf{eff}_{\mathrm{SA}} = \frac{\mathbf{speed\text{-}up}_{\mathrm{SA}}}{\underline{N}} \approx 3.28.10^{-2}$$

The classical implementation of the algorithm corresponds to use a fine solver with an RK4 scheme with $\delta t = \Delta T/2^4$ at all iterations k and all time slices. The algorithm converges in 8 iterations (so $K = 7$), which is one more step than the adaptive algorithm. This behavior is due to the impact that slight variations in the initial conditions on each subinterval $[T_N, T_{N+1}]$ might have. With the classical approach

$$\mathbf{speed\text{-}up}_{\mathrm{PP}} = \frac{T_{\mathrm{seq}}(\eta)}{T_{\mathrm{PP}}(\eta)} = \frac{960}{592} \approx 1.62, \quad \mathbf{eff}_{\mathrm{PP}} = \frac{\mathbf{speed\text{-}up}_{\mathrm{PP}}}{\underline{N}} \approx 2.7.10^{-2}.$$

so the adaptive implementation is slightly more efficient but both algorithms perform rather poorly. The first reason for this is due to the fact that the numerical cost of the coarse solver is not negligible with respect to the cost of the fine propagations (this is in contrast to the circular trajectory). The second reason, as already anticipated, concerns the heterogeneities in the refinements from one subinterval to another in the

adaptive algorithm, which lead to an imbalance in the task of computing the approximations $[\mathcal{E}(T_N, \Delta T, y_k^N); \zeta_k^N]$. This obstruction can be overcome with a rebalancing strategy that both equalizes and minimizes the numerical complexity per processor:

1. If we work with a constant number $\underline{N}$ of processors, we can merge subintervals that do not require high refinements and split others where δt becomes small.
2. If we can add more than the original $\underline{N}$ processors during the process, we can split the subintervals requiring more refinements and treat them with an increasing number of processors.

Note that, in our current example, most of the subintervals only require two levels of refinement. So, for the sake of the argument, let us assume that, in our example, we add a few more processors so that the maximum level of refinement is two ($\delta t \geq \Delta T/2^2$) and that the coarse solver involves an explicit Euler method (each step being four times cheaper than an RK4 step). Then

$$\widetilde{T}_{\mathrm{SA}}(\eta) = 7 * 60/4 + 6 * 2^2 = 129,$$

and

$$\widetilde{\mathbf{speed\text{-}up}}_{\mathrm{SA}} \approx 7.44; \quad \widetilde{\mathbf{eff}}_{\mathrm{SA}} \approx 1.24.10^{-1}$$

so about five times more efficient than the previous results.

Conclusion of the examples and perspectives: The two examples show that the adaptive approach has a lot of potential to improve the parallel efficiency of the algorithm. However, in order to fully benefit from its advantages, it is necessary to couple it with a rebalancing strategy and with *a posteriori* error estimators. Both points will be the subject of a forthcoming paper. The results of the brusselator system also highlight that the obstruction in the algorithm is in principle no longer related to the fine propagations but to the repeated use of a coarse solver whose cost might not be negligible. How to overcome this issue remains for the moment an open question on which research is being done.

6. Conclusions and perspectives. The new formulation of the parareal algorithm opens the door to improve significantly its parallel efficiency based on rigorous mathematical arguments. The increasing target tolerances which have to be met at each parareal step in the approximation of the problems on small time subdomains open also the door to the use of online stopping criteria involving *a posteriori* estimators. This point has not been exploited in the current work and will be the subject of a forthcoming paper. Other extensions based on these findings are currently ongoing.

REFERENCES

[1] S. Aouadi, D. Q. Bui, R. Guetat, and Y. Maday, *Convergence analysis of the coupled parareal-schwarz waveform relaxation method*, 2018. 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] 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.
[7] 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.
[8] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT Numerical Mathematics, 40 (2000), pp. 241–266.
[9] 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.
[10] 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.
[11] 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.
[12] M. J. Gander, *50 years of time parallel time integration*, in Householder Symposium XIX June 8-13, Spa Belgium, 2015, p. 81.
[13] 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.
[14] R. Guetat, *Méthode de parallélisation en temps: Application aux méthodes de décomposition de domaine*, PhD thesis, Paris VI, 2012.
[15] 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.
[16] 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.
[17] Y. Maday, J. Salomon, and G. Turinici, *Monotonic parareal control for quantum systems*, SIAM Journal on Numerical Analysis, 45 (2007), pp. 2468–2482.
[18] 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.
[19] M. Minion, *A hybrid parareal spectral deferred corrections method*, Comm. App. Math. and Comp. Sci., 5 (2010).
[20] M. Minion, *A hybrid parareal spectral deferred corrections method*, Communications in Applied Mathematics and Computational Science, 5 (2011), pp. 265–301.
[21] 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.
[22] 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.
[23] 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.
[24] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Commun. ACM, 7 (1964), pp. 731–733.
[25] A. Quarteroni and A. Valli, *Domain decomposition methods for partial differential equations*, Von Karman institute for fluid dynamics, 1996.
[26] A. Toselli and O. Widlund, *Domain decomposition methods: algorithms and theory*, vol. 3, Springer, 2005.