The parareal in time algorithm

Yvon Maday*

June 7, 2008

Abstract

In this paper we present the current status of a method, first introduced in 2001 authored by J.-L. Lions, Y. Maday and G. Turinici that allows for parallization in time for the simulation of systems of Ordinary Differential Equations or time dependent partial differential equations. Following the same strategy as the one that is used in domain decomposition methods for solving elliptic problems that consists in breaking the domain of computation into subdomains (with overlap or without) and solving iteratively over each subdomain independently using different processors, the "parareal in time" method proposes to break the global problem of time evolution into a series of independent evolution problems on smaller time intervals.

The iterative algorithm is based on a predictor corrector approach that generally converges quite fast, and leads, when very many processors are available, to real time solution procedures. This reasoning led us to name "parareal in time" (parallel in real time) this new algorithm.

Keywords: parallelisation, time dependent problems, predictor-corrector, shooting techniques.

1 Introduction

The use of parallel computers for solving a closed given problem is based on the notion of task decomposition allocated to each processors. This is so for all parallel algorithms and their efficiency is related both to the degree of independence of each sub-task and also on the relative complexity to have them solved on a single processor. Indeed, it has to be large enough compared with the communication between the processors in order to relatively minimize the latter in favour of the former. For instance, for the resolution of partial differential equations using domain decomposition techniques, the (spacial) domain of computation is broken up into (smaller though

*UPMC Univ Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France and Division of Applied Mathematics, Brown University, Providence, RI 02912, USA.

large enough) subdomains over which the original partial differential equation set over each subdomain and complemented with appropriate boundary conditions are solved independently. Most of the domain decomposition algorithms can be cast under the form of a fixed point acting on these unknown internal boundary conditions in order to discover them.

The efficiency of the best domain decomposition algorithms based on an update of the internal boundary conditions is such that it leads to the expected convergence with a number of iterations that is only marginally dependent on both the number of subdomains that are involved in the domain decomposition (thanks to the use of proper coarse grid preconditionners) and the discretization parameters (e.g. mesh size). Since the best resolution procedures for partial differential problems have on one processor a complexity that is larger than proportional to the number of points in the domain of computation, the resolution of large scale problems over P processors using domain decomposition algorithms is often close to P times faster than a single processor resolution leading to a full scalability of the implementation.

This is true, at least, if the local complexity is large enough, i.e. if every processor has enough operations to perform locally without exchanging the datas. With the new parallel platforms that are already available (more than 200 000 processors for the Lawrence Livermore National Laboratory's Terascale Simulation Facility, BlueGene/L in 2007) the domain decomposition cannot be the sole way to decompose the various tasks. For time dependent problems based on partial differential equations, the time direction has not received much attention to be combined with domain decomposition to further break the complexity. This is of course more critical for large systems of ordinary differential equations, where domain decomposition cannot be advocated.

There are many contributions in the literature dealing with parallelisation for time dependent problems. We refer eg to the book of K. Burrage [5] for a synthetic approach on the subject. The various techniques are classically classified into three categories :

- parallelism across the system : this consists in partitioning the "right hand side" in its various components and spread the computation of each of them over different processors. This is the most common and efficient approach when the dimension of the system is large. This is the most common use made of parallel computers in gravitational and molecular simulations, for example. Graph techniques may be used to provide a proper equilibrium among the computational charges over each processor. Another recent alternative in this category is given in [21].

- parallelism across the method : as its name tells, this approach is directly linked to the numerical scheme that is used to solve the equation. The efficiency highly depends on the characteristic of the scheme but the research in this direction has led to design intrinsically new parallel schemes. Runge-Kutta methods [12] and partitioned Runge-Kutta methods offer the possibility of parallelism across the method. The research is still active in this direction.

- parallelism across the time : this approach consists in breaking up the integration interval into sub-intervals and solve concurrently over each sub-intervals. The obvious difficulty being to provide the correct seed value, at the left of each integration sub-interval. Mostly all the techniques in this category is a multishooting technique starting from the precursor work from A. Bellen et M. Zennaro [4] on the parallelism across the steps, leading to the waveform relaxation methods introduced by E. Lelarasmee, A. E. Ruehli and A. L. Sangiovanni-Vincentelli [17]and the multigrid approaches introduced by W. Hackbush [15].

The parareal in time method, that is the object of this paper, and has been presented first in [18] enters in this third category. As it has been explained by M. Gander and S. Vandewalle[10] , it can be interpreted both as a kind of multishooting technique and also as a kind of multigrid approach, even though the leading idea came more from the (spacial) domain decomposition approach with coarse grid to get the full scalability in case many subdomains are used.

2 Presentation of the parareal in time algorithm

2.1 Basics on solution for ODE's

Consider the following evolution system of ordinary differential equation : find $\boldsymbol{u}$ function of time such that for $t \geq 0$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \mathcal{A}(t, \boldsymbol{u}) = 0, \tag{1}$$

where $\mathcal{A}$ is an appropriate operator (linear or nonlinear) from $I\!R \times V$ into V' (V being a Hilbert space, and V' its dual space), depending on $\boldsymbol{u}$ and time. We assume that, for any initial condition $\boldsymbol{v}$ provided at any initial time $t > 0$, there exists a unique solution $\boldsymbol{u}$ to (1) over the interval $[t, t + \tau]$, where τ is large enough, or even τ is any positive real number. It is well known that the solution can be written using the flow map as follows

$$\forall \tau > 0, \quad \boldsymbol{u}(t + \tau) = \mathcal{E}_\tau(t, \boldsymbol{v}), \tag{2}$$

and of course, uniqueness of the solution provides a semigroup, property

$$\forall \mu > 0, \tau > 0, \quad \mathcal{E}_\mu(t + \tau; \mathcal{E}_\tau(t; \boldsymbol{v})) = \mathcal{E}_{\mu+\tau}(t; \boldsymbol{v}). \tag{3}$$

Assume that, asides the complete evolution of $\boldsymbol{u}$, some particular check points $T_0 = t_0 < T_1 < \cdots < T_N$ are provided in the interval of propagation and that the solutions $\boldsymbol{u}(T_n)$ are of particular interest. Let us denote $\Delta T_n = T_n - T_{n-1}$; we have obviously

$$\forall n > 0, \quad \boldsymbol{u}(T_n) = \mathcal{E}_{T_n - T_0}(T_0; \boldsymbol{v}) = \mathcal{E}_{\Delta T_n}(T_{n-1}; \boldsymbol{u}(T_{n-1})) \tag{4}$$

that highlights the inherent sequential nature of the Cauchy problem since $\boldsymbol{u}(T_n)$ is a priori only known after $\boldsymbol{u}(T_{n-1})$ has been determined.

Actually, for applications, the exact flow operator is rarely known and we introduce a precise enough approximation of $\mathcal{E}$, denoted as $\mathcal{F}$ such that, for any given time $t > 0$, any $\tau > 0$, and any $\boldsymbol{v}$, $\mathcal{F}_\tau(t, \boldsymbol{v})$ is an approximation of the solution $\mathcal{E}_\tau(t, \boldsymbol{v})$. This precise approximation can be obtained by the use of any appropriate classical discretization scheme with a small timestep δt. What we are looking at are thus approximations $\left\{\lambda_n\right\}_n$ of $\left\{\boldsymbol{u}(T_n)\right\}_n$ given by $\left\{\lambda_n = \mathcal{F}_{T_n - T_0}(T_0; \boldsymbol{v})\right\}_n$ or again $\left\{\lambda_n = \mathcal{F}_{\Delta T_n}(T_{n-1}; \lambda_{n-1})\right\}_n$, if we assume that the approximated propagator $\mathcal{F}$ also satisfies the semigroup property

$$\forall \mu > 0, \tau > 0, \quad \mu, \tau \in \delta t.\mathbb{Z}, \quad \mathcal{F}_\mu(t + \tau; \mathcal{F}_\tau(t; \boldsymbol{v})) = \mathcal{F}_{\mu+\tau}(t; \boldsymbol{v}). \tag{5}$$

Note that the knowledge of a given λ_n comes after the knowledge λ_{n-1} and requires the succession of $\Delta T_n / \delta t$ steps of size δt required to go from T_{n-1} to T_n.

Note also that, should these seed values λ_n be known, then the complete approximate solution could be recomposed, independently over each interval (T_n, T_{n+1}) by evaluating $\mathcal{F}_\tau(T_n, \lambda_n)$ for any $\tau < T_{n+1} - T_n$, making the possibility of using parallel computers dedicated to each interval.

The idea in the parareal in time algorithm is to propose a sequence $\left\{\lambda_n^k\right\}_n$ that converges to $\left\{\lambda_n\right\}_n$ (rapidly) when $k \longrightarrow \infty$ and that can be built mainly in parallel.

2.2 The parareal in time algorithm

In addition to the datum of $\mathcal{F}$, let us assume that a coarse approximation of the flow is also available. This second approximation — denoted by $\mathcal{G}$ — provides another approximation $\mathcal{G}_\tau(t, \boldsymbol{v})$ of the same solution at time $t + \tau$ of problem (1) with initial condition $\boldsymbol{u}(t) = \boldsymbol{v}$.

This coarse approximation does not need to be as accurate as $\mathcal{F}$ and can be chosen much less expensive e.g. by the use of a scheme with a much larger time step : $\delta T >> \delta t$. Another way to get a less expensive solver is to base the definition of $\mathcal{G}$ on a "reduced physics" denoted by $\hat{\mathcal{A}}$ that is less computer resources demanding than the "exact" physics, described by $\mathcal{A}$ e.g. by getting rid of some large scales. We shall elaborate on this in the sequel (see subsection 3.2).

For the sake of simplicity, we also assume the semigroup property over $\mathcal{G}$:

$$\forall \mu > 0, \tau > 0, \quad \mu, \tau \in \delta T.\mathbb{Z}, \quad \mathcal{G}_\mu(t + \tau; \mathcal{G}_\tau(t; \boldsymbol{v})) = \mathcal{F}_{\mu+\tau}(t; \boldsymbol{v}). \tag{6}$$

The parareal in time algorithm then proposes an approximation of each λ_n, $n = 1, .., N$, by recursively defining the sequence λ_n^k by

$$\lambda_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(T_n; \lambda_n^{k+1}) + \mathcal{F}_{\Delta T}(T_n; \lambda_n^k) - \mathcal{G}_{\Delta T}(T_n; \lambda_n^k). \tag{7}$$

It is an easy matter to realize first that the method is exact after enough iterations, indeed, by induction we obtain that $\lambda_n^n = \lambda_n = \mathcal{F}_{T_n}(T_0; \boldsymbol{u}_0)$ for any $n > 0$; however the convergence of λ_n^k to λ_n goes much faster as we shall see shortly.

Second, this way of presenting the algorithm places it in the category of predictor/corrector algorithms, where the predictor is $\mathcal{G}_{\Delta T}(T_n;\lambda_n^{k+1})$ while the corrector is $\mathcal{F}_{\Delta T}(T_n;\lambda_n^k) - \mathcal{G}_{\Delta T}(T_n;\lambda_n^k)$. This is not the original presentation of the scheme that was made in [18], though these are completely equivalent for linear systems while (7) is more amenable to general (e.g. nonlinear) systems of differential equations. We remark also that the same equation (7) could also be rewritten as

$$\lambda_{n+1}^{k+1} = \mathcal{F}_{\Delta T}(T_n;\lambda_n^k) + \mathcal{G}_{\Delta T}(T_n;\lambda_n^{k+1}) - \mathcal{G}_{\Delta T}(T_n;\lambda_n^k) \tag{8}$$

the interpretation being that λ_{n+1}^{k+1} is obtained as a correction of $\mathcal{F}_{\Delta T}(T_n;\lambda_n^k)$ by a translation of vector $\mathcal{G}_{\Delta T}(T_n;\lambda_n^{k+1}) - \mathcal{G}_{\Delta T}(T_n;\lambda_n^k)$, i.e. the translation that allows to go from $\mathcal{G}_{\Delta T}(T_n;\lambda_n^k)$ to $\mathcal{G}_{\Delta T}(T_n;\lambda_n^{k+1})$ that can be written as :

$$\lambda_{n+1}^{k+1} = \mathcal{T}_{\mathcal{G}_{\Delta T}(T_n;\lambda_n^k)\to\mathcal{G}_{\Delta T}(T_n;\lambda_n^{k+1})}[\mathcal{F}_{\Delta T}(T_n;\lambda_n^k)] \tag{9}$$

with obvious notations.

This presentation may inspire alternative formulations if one want to preserve some quantities of physical meaning. An example of this is provided in [25] where the translation was replaced by a rotation that allowed to maintain the normalisation of the wave function, solution for a Schrödinger equation :

$$\lambda_{n+1}^{k+1} = \mathcal{R}_{\mathcal{G}_{\Delta T}(T_n;\lambda_n^k)\to\mathcal{G}_{\Delta T}(T_n;\lambda_n^{k+1})}[\mathcal{F}_{\Delta T}(T_n;\lambda_n^k)]. \tag{10}$$

Last, but not least, this algorithm is embarrassingly easily implemented in parallel. Indeed, once, for a given k, every computation in n has been done, we know λ_n^k for any n, then the (expensive) evaluation of $\mathcal{F}_{\Delta T}(T_n;\lambda_n^k)$ can be done independently over each processor (assuming you get at least N of them), the corrector can thus be formed and stored in parallel and the — non expensive — sequential propagation at iteration $k+1$ can proceed through the evaluation of the predictor (based on $\mathcal{G}$) and using pre-computed corrector.

2.3 Numerical simulation - first example.

Throughout this paper, the one dimensional periodic Burger's equation will serve as a typical example for illustrating the numerical behavior of the parareal scheme. The problem is to find a function u, with periodic boundary conditions over $(0, 2\pi)$ such that

$$\frac{\partial u}{\partial t} - \nu\frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x} = 0 \tag{11}$$

where the viscosity term $\nu > 0$ will diminish largely as the paper evolves.

We have chosen periodic boundary condition in order to use a very simple discretization in space of Fourier type since our focus here is on the time evolution. We have chosen a second order splitting scheme, in order to be able to use an exact evolution for the diffusion step (working in Fourier space) and a collocation method for the nonlinear convection part based on a second order Runge Kutta explicit scheme.

In this first simulation, the viscosity will be rather large, $\nu = 1$ and the plain parareal scheme is employed. The error between the solution obtained from the use of the fine scheme only over the time range $[0, 1]$ with $\delta t = 10^{-3}$ and Fourier truncation at $N = 64$ and the solution obtained with the parareal scheme after $k + 1$ coarse steps (and thus the use k fine solvers) and $\delta T = \Delta T = 10^{-2}$, are presented in the following table and we see on the lin. /log. plot that the convergence is geometrical anticipating the theorem that is stated in subsection 3.1, see also [11] for superlinear convergence rate.

TABLE 1 : LOGARITHM OF THE ERROR AFTER k ITERATIONS

	k=2	k=3	k=4	k=5	k=6	k=7
Log(error)	-4.00	-5.85	-7.58	-9.24	-10.9	-12.5

After one iteration, the error is already quite small, about the order of the difference between the exact solution (of the truncated Fourier approximation) and the solution provided by the fine scheme (that is of the order of 10^{-4}, with a logarithm equal to -3.96) providing a full efficiency if we neglect the cost of the coarse solver. Indeed, if the implementation of the fine solvers over each interval (T_n, T_{n+1}) is performed in parallel, all the $\Delta T/\delta t$ splitting operations to propagate the solution between T_n and T_{n+1} can be done concurrently, which is $N = T/\Delta T$ times less than the $T/\delta t$ splitting operations to propagate the solution between T_0 and T_N.

In the following, we have run the same experiment with a smaller fine time step $\delta t = 5.10^{-4}$ and the same coarse time step $\delta T = \Delta T = 10^{-2}$. This implementation of the parareal in time algorithm provides an error that is about the same as the pure sequential fine solver (that is equal to $3.\ 10^{-5}$, with a logarithm equal to -4.52) after two iterations.

TABLE 2 : LOGARITHM OF THE ERROR AFTER k ITERATIONS

	k=2	k=3	k=4	k=5	k=6	k=7
Log(error)	-4.00	-5.84	-7.57	-9.23	-10.9	-12.4

2.4 Complexity and parallelism efficiency - part 1

First of all, the implementation of the plain fine solver, regardless of the discretization scheme that is used (explicit Euler, implicit Runge Kutta or whatever..), involves a complexity that is proportional to the number of time steps required between time T_0 and time T_N. The complexity is thus equal to $\frac{T_N - T_0}{\delta t} = \frac{N \Delta T}{\delta t}$ times the elementary cost to perform one step denoted as $\mathcal{C}_{\mathcal{F}}$ (that depends on the solver that is used).

Now, as regards the implementation of the parareal in time scheme, from the remark at the end of the previous subsection, we can perform all computations involving the fine solver in parallel so that, provided we have at least N processors, the complexity over each processors is reduced to $\Delta T/\delta t$, hence N times smaller than the sequential implementation of the method.

In addition, let us denote by $\mathcal{C}_\mathcal{G}$ the cost to perform the coarse propagator over one time step, and by K the number of iteration required to achieve the required convergence of the parareal in time algorithm (meaning K resolutions through the coarse solver and $K-1$ parallel resolution through the fine solver) then the total (cumulated) complexity of the parareal in time algorithm is

$$KN\mathcal{C}_\mathcal{G} + (K-1)N\frac{\Delta T}{\delta t}\mathcal{C}_\mathcal{F}$$

but the local complexity, over each processor (that is responsible for the time to be waited to get the solution) is only

$$KN\mathcal{C}_\mathcal{G} + (K-1)\frac{\Delta T}{\delta t}\mathcal{C}_\mathcal{F} \tag{12}$$

that should be compared with

$$\frac{T_N - T_0}{\delta t}\mathcal{C}_\mathcal{F} = N\frac{\Delta T}{\delta t}\mathcal{C}_\mathcal{F} \tag{13}$$

so that, assuming that the cost of the coarse propagator is (relatively) negligible with respect to the cost of the fine propagator, we deduce that a parallelism efficiency equal to $N/(K-1)$ is achieved.

We shall come back to this first result in subsection 4.3; we can nevertheless already draw a few preliminary conclusions :

- As first noted in [2], K should be small, and the best should be $K = 2$ leading to an almost perfect parallel efficiency. This can be realized as it has been illustrated on the example in Table 1. It can also be obtained, in more complex situations, by having a ΔT small enough so that $N\Delta T$ is still rather small. This leads to a hierarchical implementation of the parareal in time algorithm where after 2 coarse and (only) 1 fine resolutions, over N intervals providing an evolution over the interval $[T_0, T_N^1]$ we should start over the algorithm over the interval $[T_0^2 = T_N^1, T_N^2]$ and proceed. This strategy requires that δt, that should be smaller than ΔT, ends up being quite small and is applicable in rather particular situations only. In the most general case, K will be small but most of the times about 5 so that having P processors speeds up the resolution time only by a factor $P/4$.

- Note that the choice of the coarse operator $\mathcal{G}$ is rather critical for the full efficiency, indeed, it should be close enough to $\mathcal{F}$, but cheap enough so that $KN\mathcal{C}_\mathcal{G}$ can be neglected with respect to $(K-1)\frac{\Delta T}{\delta t}\mathcal{C}_\mathcal{F}$. We shall elaborate on this by providing some guidelines and rules for the choice of this element of the algorithm (see (3.1)).

- Comparing the complexity analysis of the parareal in time algorithm with the similar one in the context of domain decomposition methods reveals one major difference between the parareal in time algorithm and the standard domain

decomposition approaches even though there exists many other similitude. The complexity of the solution procedure in the time direction is directly proportional to the number of steps involved between the initial time and the final time (except if you would like to use a global spectral method in time) whereas the complexity of the solution procedure in the spacial direction is very scarcely proportional to the number of degrees of freedom (say n_{dof}) in the spacial direction : it often scales like $\mathcal{O}(n_{dof}^s)$ with $s > 1$. This implies that even without an effective implementation on parallel processors, the domain decomposition approach may already be a good solution procedure. Indeed, by breaking the global problem into K pieces, the resolution of each subproblem is K^s less expensive than the original one, adding all these resolutions leads to a factor K^{1-s} time the resolution of the original problem for each iterations..... This will never be the case for the (only) use of the parareal in time algorithm. We shall elaborate on this latter (see section 4).

2.5 An algebraic interpretation

Back to the material that was introduced in subsection 2.1, we remind that the discrete problem of fine propagation we want to solve reads : for any n, find $\lambda_n = \mathcal{F}_{T_n - T_0}(T_0; \boldsymbol{v})$. This can equivalently be written as $\lambda_n = \mathcal{F}_{\Delta T}(T_{n-1}; \lambda_{n-1})$ (from now on we assume all $\Delta T_n = \Delta T$) or again, under a convenient matricial form

$$\begin{pmatrix} Id & 0 & ... & 0 \\ -\mathcal{F}_{\Delta T} & Id & 0 & ... \\ 0 & -\mathcal{F}_{\Delta T} & Id & .. \\ 0 & ..0 & -\mathcal{F}_{\Delta T} & Id \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ .. \\ \lambda_{N-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v} \\ 0 \\ .. \\ 0 \end{pmatrix} \tag{14}$$

that can be synthesized, with obvious notations, as follows

$$M \quad \Lambda \quad = \quad F. \tag{15}$$

The sequential nature of the fine scheme clearly appears here since a standard inversion of this lower triangular system involves $\mathcal{O}(N)$ propagations of system

$$\lambda_n = \mathcal{F}_{\Delta T}(T_{n-1}; \lambda_{n-1}).$$

In order to accelerate the solution procedure, through the parareal in time scheme, we have proposed to define a sequence Λ^k that converges toward the exact solution of (15) as $k \to \infty$. The choice of the coarse propagator $\mathcal{G}_{\Delta T}$, allows to define the matrix

$$\widetilde{M} = \begin{pmatrix} Id & 0 & ... & 0 \\ -\mathcal{G}_{\Delta T} & Id & 0 & ... \\ 0 & -\mathcal{G}_{\Delta T} & Id & .. \\ 0 & ..0 & -\mathcal{G}_{\Delta T} & Id \end{pmatrix} \tag{16}$$

and propose the following algebraic formulation of the parareal in time algorithm

$$\Lambda^{k+1} \quad = \quad \Lambda^k \quad + \quad \widetilde{M}^{-1} \quad Res^k \tag{17}$$

where the residual Res^k is defined by $Res^k = F - M\Lambda^k$.

The convergence of the parareal in time algorithm, at least when A is linear positive definite, leads to the following conclusion : to some extent $\widetilde{M}^{-1}$ can be considered as a preconditionner for M, in the sense that the amplification matrix $\widetilde{M}^{-1}M$ is close to Identity.

3 Various ways to reduce the coarse simulation

In this section we are going to propose different ways of reducing the computational cost of the coarse operator while preserving the convergence properties of the parareal in time algorithm. In the first subsection, we present the analysis of the convergence of the algorithm to frame the influence of the coarseness of $\mathcal{G}$.

3.1 Convergence analysis

Let us remind a few further basics on the analysis of solutions for differential equations. The analysis presented in this subsection was first presented in [22], we give it back here for the sake of completeness.

The existence of a propagator $\mathcal{E}$ associated with equation (1) that has been assumed in subsection 2.1 results e.g. from the hypothesis, made on $\mathcal{A}$:

$$|\mathcal{A}(t,x)| \leq C(1+|x|), \quad |\mathcal{A}(t,x) - \mathcal{A}(t,y)| \leq C|x-y| \tag{18}$$

in addition, we have the following stability

$$|\mathcal{E}_\tau(t,x) - \mathcal{E}_\tau(t,y)| \leq (1+C\tau)|x-y|, \tag{19}$$

with the same constant C. In what follows the generic constants will be still denoted by C and only depend on $\mathcal{A}$ and the chosen discretization schemes.

We also make the following hypothesis, denoting by $\delta\mathcal{F}$ (resp. $\delta\mathcal{G}$) the difference $\delta\mathcal{F} = \mathcal{E} - \mathcal{F}$ (resp. $\delta\mathcal{G} = \mathcal{E} - \mathcal{G}$)

$$\forall\tau, \quad |\delta\mathcal{F}_\tau(t,x)| \leq C\tau\eta(1+|x|), \quad |\delta\mathcal{G}_\tau(t,x)| \leq C\tau\varepsilon(1+|x|) \tag{20}$$

and in addition that

$$\forall\tau, \quad |\delta\mathcal{F}_\tau(t,x) - \delta\mathcal{F}_\tau(t,y)| \leq C\tau\eta|x-y|, \quad |\delta\mathcal{G}_\tau(t,x) - \delta\mathcal{G}_\tau(t,y)| \leq C\tau\varepsilon|x-y| \tag{21}$$

with $0 < \eta \leq \varepsilon$. Note that from (19, 21) we get

$$|\mathcal{F}_\tau(t,x) - \mathcal{F}_\tau(t,y)| \leq (1+C\tau)|x-y|, \tag{22}$$

and

$$|\mathcal{G}_\tau(t,x) - \mathcal{G}_\tau(t,y)| \leq (1+C\tau)|x-y|, \tag{23}$$

with a constant still denoted by C. Under these assumption, we have the following result

Theorem 1 *Assume that the discrete propagators $\mathcal{F}$ and $\mathcal{G}$ satisfy (20) and (21). Assume also that $k \leq K$ with some fixed $K \leq N/2$. Then the error between the exact solution and the solution provided by the parareal in time scheme (7) satisfies*

$$|\lambda_n^k - \boldsymbol{u}(T_n)| \leq C(\varepsilon^k + \eta), \quad \forall T_n < T. \tag{24}$$

Proof: Let us recall the definition

$$\lambda_n^k = \mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^k) + \mathcal{F}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}) - \mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}),$$

from which we deduce

$$\begin{aligned}
\lambda_n^k - \mathcal{F}_{T_n - T_0}(T_0, \boldsymbol{v}) &= \mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^k) - \mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}), \\
&\quad + \mathcal{F}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}) - \mathcal{F}_{\Delta T}(T_{n-1}, \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v})) \\
&= \mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^k) - \mathcal{G}_{\Delta T}(T_{n-1}, \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v})) \\
&\quad - \mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}) + \mathcal{G}_{\Delta T}(T_{n-1}, \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v})) \\
&\quad + \mathcal{F}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}) - \mathcal{F}_{\Delta T}(T_{n-1}, \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v}))
\end{aligned}$$

hence

$$\begin{aligned}
|\lambda_n^k - \mathcal{F}_{T_n - T_0}(T_0, \boldsymbol{v})| &\leq |\mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^k) - \mathcal{G}_{\Delta T}(T_{n-1}, \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v}))| \\
&\quad + |\delta\mathcal{G}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}) - \delta\mathcal{G}_{\Delta T}(T_{n-1}, \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v}))| \\
&\quad + |\delta\mathcal{F}_{\Delta T}(T_{n-1}, \lambda_{n-1}^{k-1}) - \delta\mathcal{F}_{\Delta T}(T_{n-1}, \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v}))|.
\end{aligned}$$

From (19), (20) and (21) we obtain

$$\begin{aligned}
|\lambda_n^k - \mathcal{F}_{T_n - T_0}(T_0, \boldsymbol{v})| &\leq (1 + C\Delta T)|\lambda_{n-1}^k - \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v})| \\
&\quad + C(\varepsilon + \eta)\Delta T|\lambda_{n-1}^{k-1} - \mathcal{F}_{T_{n-1}-T_0}(T_0, \boldsymbol{v})|
\end{aligned}$$

and setting

$$\theta_n^k = (1 + C\Delta T)^{k-n} C(\varepsilon + \eta)^{-k} \Delta T^{-k} |\lambda_n^k - \mathcal{F}_{T_n - T_0}(T_0, \boldsymbol{v})|,$$

this can be rewritten as

$$\theta_n^k \leq \theta_{n-1}^k + \theta_{n-1}^{k-1},$$

so that

$$\theta_n^k \leq C \binom{n}{k},$$

or again

$$|\lambda_n^k - \mathcal{F}_{T_n - T_0}(T_0, \boldsymbol{v})| \leq C(\varepsilon + \eta)^k \Delta T^k \binom{n}{k}, \tag{25}$$

so that for any fixed k, we get

$$|\lambda_n^k - \mathcal{F}_{T_n - T_0}(T_0, \boldsymbol{v})| \leq C[T_n - T_0]^k (\varepsilon + \eta)^k \leq C[T_n - T_0]^k \varepsilon^k.$$

The proof follows easily by recalling the classical error bound on the fine propagator derived through Gronwall lemma

$$|\mathcal{F}_{T_n-T_0}(T_0, \boldsymbol{v}) - y(T_n)| \leq C\eta.$$

In the previous applications where e.g. $\mathcal{F}$ and $\mathcal{G}$ are based on a first order Euler scheme, the fine propagator with a time step δt, the assumption (20) and (21) hold respectively with $\eta = \delta t$ and $\varepsilon = \Delta T$. We get here the standard approximation for the parareal-in-time scheme, providing an error of size $\Delta T^k + \delta t$ after k iterations.

The additional feature derived from this analysis is that the dynamics that is used to propagate $\mathcal{G}$ can be degraded with respect to the original $\mathcal{A}(t, y(t))$ present in (1) in order to get an even cheaper coarse propagator.

This degradation can be obtain from a model reduction as is presented in the next subsection. It can be done by coarsening the spatial discretization for the definition of $\mathcal{G}$. The error ε then includes not only the coarse time step, but also this spatial error (represented as a Δx), i.e., $\varepsilon \simeq \Delta T + \Delta x$. This is what is done in the following subsection.

3.2 Model reduction

For large scale problem, the main contribution in determining the complexity $\mathcal{C}_{\mathcal{F}}$ of each steps of the fine solver lies on the description of the model $\mathcal{A}$. In order to decrease the complexity $\mathcal{C}_{\mathcal{G}}$ of the coarse solver, reduced models as the one that are often proposed by the model designers may be used . The idea is to use a simplified physics $\hat{\mathcal{A}}$ instead of $\mathcal{A}$.

The first example that has been implemented (see [1]) in the context of the parareal in time algorithm deals with a differential system for ab initio molecular dynamics simulation. There are many available coarse propagators at our disposal in the literature motivated by physical insight. Very often, these alternative models, of empirical nature depend on coefficients that are fitted to reproduce (to a certain extent) the ab initio models.

In the simulations presented in [1], it appeared that the use of these empirical models for the definition of the coarse propagator allowed to maintain the convergence rate obtained with the plain algorithm based on a coarse propagator with the full physics and where only the time step was larger. It can be argued that nothing is much surprising in that as, in some sense, the online work required on ab initio models to converge on the nonlinear contributions is admittedly replaced by a simpler linear model derived however from an expertise based on a large number of simulations. In some sense, the reduced model is here an exceptional accurate model. In order to refute this claim, we have also performed an additional simulations where the original coefficients in the simplified model were deviated from the empirical ones by rather large values and the convergence was not much affected, proving the robustness of the approach.

A similar use of a reduced model has also been presented in [19] for kinetic situations where chemical reactions involving many scales in time have been solved. In these situations the presence of very rapid reactions is responsible of a stiff character of the simulations that require the use of very small time steps to be stably evolved. A lot of work exist in the literature in this field too, some of empirical nature other of mathematical nature, to propose model reductions based on the existence of a equilibrium manifold where the concentrations in species evolve slowly when some equilibrium has been reached. Note that in such a situation, the fine model only needs to be used at the very beginning of the simulation, the reduced model is indeed sufficient to treat about the slow evolution of the kinetics except if some external sources are present and des-equilibrate the states. These reduced models have been used in [19] to define the coarse propagator with success, for simulations where accidental events occurred from times to times to de-equilibrate the species and deviate their evolutions from the equilibrium manifold.

The two previous examples deal with differential systems. For time dependent partial differential equations, the definition of reduced models is also a field of quite large activity, actually in some cases the approach is in the other way around since, very often, finer models are invented and employed in replacement of previously used coarser ones since the finer can now be implemented and more phenomenon need to be simulated. So in some sense, it is more the definition of refined models that is actually the field of interest. For classical problems, as the one encountered in fluid flows, where this is not the case, some reduced models can also be derived by a "learn as you walk" strategy, where the model is built from the solutions computed from the first fine solutions following a "à la reduced basis method" strategy as is explained in [19] or [8]. In this case the reduction comes from the projection of the solution on a small dimensional domain well suited to represent the state of the solution. We refer to [23] and [14] for definition of the reduced basis in the more general context of parabolic problems and a posteriori estimators a proposed to validate the approximation.

It is not the route that has been followed in [16] where the reduced model is built from an average of the small scales. The strategy is taken from [6] and [7] : the coarse model is based on an effective equation that is constructed in order to model the behavior of the oscillating solutions of a model nonlinear, dispersive convection diffusion equation. This happens when the viscosity parameters is small and leads to highly oscillatory solutions. The use of the more simple effective equation is enough to advance sufficiently well the solution in a parareal algorithm.

3.3 Coarse spacial grid

In this subsection, we propose to take benefit of the presence of an additional discretization in space when the time differential system is derived from the spacial approximation of an unsteady partial differential equation. There are many reasons for this: the first one being the "Courant-Friedrichs-Lewy" (C.F.L.) type condition that needs to be satisfied in order to meet the stability constraint of the global discretiza-

tion, especially when explicit schemes are used. Indeed, the time step of the coarse solver being larger than for the fine solver, there is a need to tune the spacial discretization parameter, so that the stability condition continues to be satisfied.

The second reason is, as in the previous subsection, to try to lower the cost of the coarse solver; this cost being of course monotically increasing with the number n_{dof} of degrees of freedom used in the spacial direction (remember the complexity in $\mathcal{O}(n_{dof}^s)$), $s \geq 1$.

Actually there is a third reason as we shall see at the end of this subsection and that will be more detailed at the end of the paper.

The first numerical implementation of this reduction can be found in [9] for the Navier-Stokes problem. Two types of discretizations were presented in that paper, both of variational type, based either on finite element or on spectral approximations. The variational framework is actually quite handy to coarsen the spacial discretization used for the coarse solver, it suffices indeed to base the coarse solver on a variational subspace X_Δ of the basic one X_δ, i.e. the one that is used for the fine solver. We also need two operators in order to navigate from one space into the other one. The fine propagator indeed carries a solution in X_δ while the coarse one in X_Δ, in order to add them up, an imbedding and an extension operators are required. One, named $\pi_{\mathcal{G}}^{\mathcal{F}}$ maps X_Δ into X_δ, the other one named $\pi_{\mathcal{F}}^{\mathcal{G}}$ maps X_δ into X_Δ. Provided with these operators, the parareal algorithm reads

$$\lambda_{n+1}^{k+1} = \pi_{\mathcal{G}}^{\mathcal{F}} \mathcal{G}_{\Delta T}(T_n; \pi_{\mathcal{F}}^{\mathcal{G}} \lambda_n^{k+1}) + \mathcal{F}_{\Delta T}(T_n; \lambda_n^k) - \pi_{\mathcal{G}}^{\mathcal{F}} \mathcal{G}_{\Delta T}(T_n; \pi_{\mathcal{F}}^{\mathcal{G}} \lambda_n^k). \tag{26}$$

We report here some experiments one our test example of the Burgers' problem that provide the very same conclusions as the ones in [9]. The coarse approximation is obtained by reducing the degree in the Fourier direction. Starting with the very same parameters as in subsection 2.3 : viscosity equal to 1, $\delta t = 10^{-3}$, $\delta T = \Delta T = 10^{-2}$ and Fourier truncation at $N = 64$ for the fine solver we first test $N_{\mathcal{G}} = 32$ for the coarse one, the results are exactly the same as in Table 1, i.e. the results are similar as if the full spectrum was retained in the coarse solver, the results are also the same if we use only $N_{\mathcal{G}} = 16$ in the coarse propagator. This of course comes from the fact that this problem is very nice (because the viscosity is quite large). If the problem is a bit more difficult and requires higher degree in space (this is the case if the viscosity diminishes) the lack of accuracy of the coarse solver is more visible: in the next simulation, we use a viscosity equal to $\nu = .2$ with $\delta t = 2.10^{-4}$, $\delta T = \Delta T = 4.10^{-3}$ and Fourier truncation at $N = 128$ for the fine solver we first compare the use of $N_{\mathcal{G}} = 128$ or $N_{\mathcal{G}} = 64$ for the coarse one, the results are as again the same and reveal that the same error as the one obtained with the fine solver (-3.80) is achieved after 2 iterations of the parareal algorithm. If now we use only $N_{\mathcal{G}} = 32$ for the coarse solver, the results are degraded, and we need $k = 4$ to get the full accuracy.

TABLE 3 : LOGARITHM OF THE ERROR AFTER k ITERATIONS

	k=2	k=3	k=4	k=5	k=6	k=7
N=128	-3.20	-4.47	-5.63	-6.75	-7.85	-8.92
N=64	-3.20	-4.47	-5.63	-6.75	-7.85	-8.92
N=32	-2.35	-2.84	-3.32	-4.31	-4.81	-5.41

that reveals a relative degradation of the convergence rate of the algorithm if the coarse solver does not carry well enough the dynamics, since the error of the fine solver is recovered after 4 iterations (that makes however the fine solver run about 80 times faster than the pure sequential implementation of fine solver).

It is interesting to note that, for a smaller viscosity ($\nu = 0.03$) and the same discretization as above, the parareal algorithm with $N_{\mathcal{G}} = 128$ or $N_{\mathcal{G}} = 64$, is not stable but the use of a smaller $N_{\mathcal{G}} = 32$ in the coarse solver allows to have a monotonic convergence of the parareal algorithm. We shall elaborate on this stability issue further down.

4 More on iterative solvers

After these premises on the method, let us focus now on problems of larger size or larger complexity.

For problems with a larger size, when many processors are available, parallel algorithms are used to break the complexity, this is the case for domain decomposition as we indicated in the introduction and the iterations of the algorithm can be intertwined with the iterations of the parareal algorithm. This is explained in the first subsection in the case of an overlapping domain decomposition. Of course non-overlapping domain decomposition (FETI, Neumann-Neumann...) can also be used as is presented in [26] and into more details in [13].

For problems with higher complexity, either due to nonlinearity or due to the back and forth resolution of a primal and dual problem in case of an optimal control strategy, iterations are also required that can be combined with the ones of the parareal algorithm. We explain this feature in the case of the control problem in the following subsection.

4.1 Coupling with domain decomposition – The overlapping case.

The problem and the decomposition of the domain In what follows, we present a space-time parallel iterative method of Schwarz' type for solving the following problem

$$\begin{array}{rcll} \frac{\partial u}{\partial t} - \Delta u & = & f, & \text{in } \Omega \times [0,T] \\ u(x,0) & = & u_0(x), & \text{in } \Omega, \\ u(x,t) & = & g(t,x), & \text{over } \partial\Omega \times [0,T], \end{array} \tag{27}$$

where Ω is a domain in $I\!R^2$ or $I\!R^3$.

We assume Ω is decomposed into P subdomains that we shall assume to be overlapping to make things easier, i.e. $\Omega = \cup_{p=1}^{P} \Omega^p$. This decomposition leads to the definition of a regular partition of unity associated with this decomposition, i.e. there exists regular enough positive functions χ^p with support in Ω^p such that

$$\forall x \in \Omega, \quad \sum_{p=1}^{P} \chi^p(x) = 1.$$

From the hypothesis of overlapping decomposition, there exists, for each interface $\Gamma^p = \partial\Omega^p \setminus \partial\Omega$ an overlapping decomposition

$$\Gamma^p = \bigcup_{q=1, q\neq p}^{P} \Gamma^{p,q}, \quad \text{where } \Gamma^{p,q} = \Gamma^p \cap \Omega^q$$

to which another regular partition of unity over Γ^p can be associated, i.e. there exists regular enough positive functions $\psi^{p,q}$ with support in $\Gamma^{p,q}$

$$\forall x \in \Gamma^p, \quad \sum_{q=1, q\neq p}^{P} \psi^{p,q}(x) = 1.$$

To make things easy, for the definition of the algorithm, we assume no discretization is used in space (neither for the coarse nor for the fine propagator).

The iterative procedure we are going to define involves a fine and accurate solution (that here, again for simplicity, is assumed to be exact) acting only and independently over each block $\Omega^p \times [T_n, T_{n+1}]$ and a global coarse solution.

The fine solution over $\Omega^p \times [T_n, T_{n+1}]$ at iteration k is denoted as $u_{p,n}^k$. A global function u_n^k is built from the various $\left(u_{p,n}^k\right)_p$ as being an element of $H^1(\Omega)$ defined e.g. by

$$\forall x \in \Omega \times [T_n, T_{n+1}], \quad u_n^k(x,t) = \sum_{p=1}^{P} \chi^p(x) u_{p,n}^k(x,t)$$

for almost each time.

As in the original settings (7), the parareal algorithm with domain decomposition is based on the iterative definition of the snapshots λ_n^k at each time T_n. The algorithm though, cannot be expressed on this entity only as it involves also the familly of u_n^k. It reads

$$\{\lambda_{n+1}^{k+1}, u_n^{k+1}\} = \mathcal{G}_{\Delta T}(T_n; \lambda_n^{k+1}) + \mathcal{F}_{\Delta T}(T_n; \{\lambda_n^k, u_n^k\}) - \mathcal{G}_{\Delta T}(T_n; \lambda_n^k). \tag{28}$$

The coarse propagator is actually quite standard; it consists in the resolution of :

$$\frac{\mathcal{G}_{\Delta T}(T_n; \lambda_n^k) - \lambda_n^k}{\Delta T} - \Delta(\mathcal{G}_{\Delta T}(T_n; \lambda_n^k)) = f(T_{n+1}). \tag{29}$$

where we do not elaborate on this simulation. Note however that it should be based on a coarse spacial mesh (as explained above) and on standard Schwarz' domain decomposition algorithm in order to lower the cost at most.

The fine propagator, as we announced above, actually involves not only the knowledge of λ_n^k but also of u_n^k. The definition of $\mathcal{F}_{\Delta T}(T_n;\{\lambda_n^k,u_n^k\})$ proceeds as follows :

Step one. We propagate the solution over $\Omega^p\times[T_n,T_{n+1}]$ by solving once

$$\begin{array}{rcl}\dfrac{\partial\tilde{u}_{p,n}^{k+1}}{\partial t}-\Delta\tilde{u}_{p,n}^{k+1} &=& f,\quad \text{in } \Omega^p\times[T_n,T_{n+1}]\\ \tilde{u}_{p,n}^{k+1}(x,T_n) &=& \lambda_n^k(x),\quad \text{in } \Omega^p,\\ u_{p,n}^{k+1}(x,t) &=& g(x,t),\quad \text{over } \partial\Omega^p\cap\partial\Omega\times[T_n,T_{n+1}],\\ \tilde{u}_{p,n}^{k+1}(x,t) &=& u_n^k(x,t)+(\lambda_n^k(x)-u_n^k(x,T_n)),\\ && \text{over } \partial\Omega^p\setminus\partial\Omega\times[T_n,T_{n+1}],\end{array}\tag{30}$$

Note that the correction : $\lambda_n^k-u_n^k(.,T_n)$, allows us to have the boundary conditions over $\partial\Omega^p\setminus\partial\Omega$ compatible with the initial condition provided at time T_n for each local problem.

Step two. We now define from the various $\tilde{u}_{p,n}^{k+1}$ an updated boundary condition over each $\partial\Omega^p$ as follows

$$\forall x\in\partial\Omega^p\times[T_n,T_{n+1}],\quad \gamma_{p,n}^{k+1}(x,t)=\sum_{q=1,q\neq p}^{P}\psi^{p,q}(x)\tilde{u}_{q,n}^{k+1}(x,t).$$

Step three. We propagate once more the solution over $\Omega^p\times[T_n,T_{n+1}]$ by solving

$$\begin{array}{rcl}\dfrac{\partial u_{p,n}^{k+1}}{\partial t}-\Delta u_{p,n}^{k+1} &=& f,\quad \text{in } \Omega^p\times[T_n,T_{n+1}]\\ u_{p,n}^{k+1}(x,T_n) &=& \lambda_n^k(x),\quad \text{in } \Omega^p,\\ u_{p,n}^{k+1}(x,t) &=& g(x,t),\quad \text{over } \partial\Omega^p\cap\partial\Omega\times[T_n,T_{n+1}],\\ u_{p,n}^{k+1}(x,t) &=& \gamma_{p,n}^{k+1}(x,t),\quad \text{over } \partial\Omega^p\setminus\partial\Omega\times[T_n,T_{n+1}],\end{array}\tag{31}$$

This allows us to define a new global solution u_n^{k+1} over each $\Omega\times[T_n,T_{n+1}]$ and ends the definition of the fine solver, we set

$$\forall x\in\Omega\times[T_n,T_{n+1}],\quad \mathcal{F}_{\Delta T}(T_n;\{\lambda_n^k,u_n^k\})=u_n^{k+1}(x,t)=\sum_{p=1}^{P}\chi^p(x)u_{p,n}^{k+1}(x,t).$$

The numerical results presented in [26] illustrate well the convergence of this intertwined algorithm. Further simulations are performed in [13].

4.2 Coupling with control

The virtual control paradigm : Let us consider the following linear version of problem (1)

$$\frac{\partial \boldsymbol{u}}{\partial t} + A\boldsymbol{u} = 0 \tag{32}$$

$$\boldsymbol{u}(0) = \boldsymbol{u}^0, \tag{33}$$

Remember that we have decomposed the time interval $[T_0, T_N = T]$ with

$$0 = T_0 < T_1 < ... < T_n < T_{n+1} < ... < T_N = T.$$

We then define the functions $\{\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_n, ..., \boldsymbol{u}_{N-1}\}$ such that $\boldsymbol{u}_n$ is the solution of

$$\frac{\partial \boldsymbol{u}_n}{\partial t} + A\boldsymbol{u}_n = 0 \quad \text{over } (T_n, T_{n+1}), \tag{34}$$

$$\boldsymbol{u}_n(T_n^+) = \lambda_n, \tag{35}$$

for any $n = 1, ..., N-1$ and $\boldsymbol{u}_0$ is defined similarly on the first interval (T_0, T_1) with initial condition $\boldsymbol{u}^0$.

It is obvious to note that the collection of solutions $\{\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_n, ..., \boldsymbol{u}_{N-1}\}$ of the previous problems is connected with the solution $\boldsymbol{u}$ of the original problem (1) (in the sense that $\boldsymbol{u}_n = \boldsymbol{u}_{|(T_n, T_{n+1})}$) if and only if, for any $n = 1, ..., N-1$, $\lambda_n = \boldsymbol{u}(t_n)$.

Our goal now is to describe how the search of the seed values $\{\lambda_n\}_n$ can be cast in a control paradigm. Let us introduce the (virtual) cost functional

$$\widetilde{\mathcal{J}(\Lambda)} = \sum_{n=1}^{N-1} \|\boldsymbol{u}_{n-1}(T_n^-) - \lambda_n\|^2, \tag{36}$$

The minimum of $\widetilde{\mathcal{J}}$ is obviously zero and is achieved from the choice $\lambda_n = \boldsymbol{u}(T_n)$. We shall not use this information however but try instead to develop a link between this frame (minimization of a convex cost functional from virtual control) and the parareal algorithm.

In order to solve this minimization problem, we compute the derivative of $\widetilde{\mathcal{J}(\Lambda)}$

$$\delta\widetilde{\mathcal{J}(\Lambda)}(\delta\Lambda) = \sum_{n=1}^{N-1} (\boldsymbol{u}_{n-1}(T_n^-) - \lambda_n, \delta\boldsymbol{u}_{n-1}(T_n^-) - \delta\lambda_n). \tag{37}$$

where $\delta\boldsymbol{u}_{n-1}$ is the solution of

$$\frac{\partial \delta\boldsymbol{u}_{n-1}}{\partial t} + A\delta\boldsymbol{u}_{n-1} = 0 \quad \text{over } (T_{n-1}, T_n), \tag{38}$$

$$\delta\boldsymbol{u}_{n-1}(T_{n-1}^+) = \delta\lambda_{n-1}, \tag{39}$$

The introduction of an adjoint state is the standard argument for getting an expression of this derivative. Let us introduce the collection p_n, $n = N-2, N-1, ..., 0$ of solutions to

$$\begin{aligned} \frac{\partial p_n}{\partial t} + A^* p_n &= 0 \quad \text{over } (T_n, T_{n+1}), & (40) \\ p_n(T_{n+1}^-) &= (\boldsymbol{u}_n(T_{n+1}^-) - \lambda_{n+1}). & (41) \end{aligned}$$

We multiply now (40) by $\delta \boldsymbol{u}_n$ for any $n = 0, ..., N-2$, we integrate in time between T_n and T_{n+1}, then integrate by parts and use (34) so as to obtain

$$-(p_n(T_{n+1}^-), \delta \boldsymbol{u}_n(T_{n+1}^-)) + (p_n(T_n^+), \delta \boldsymbol{u}_n(T_n^+)) = 0.$$

We derive from (41) that $p_{n-1}(T_n^-) = (\boldsymbol{u}_{n-1}(T_n^-) - \lambda_n)$ so that

$$(\boldsymbol{u}_n(T_{n+1}^-) - \lambda_{n+1}, \delta \boldsymbol{u}_n(T_{n+1}^-)) = (p_n(T_n^+), \delta \boldsymbol{u}_n(T_n^+)).$$

Using this in (37) then yields

$$\delta \widetilde{\mathcal{J}(\Lambda)}(\delta\Lambda) = \sum_{n=0}^{N-1} (p_n(T_n^+), \delta \boldsymbol{u}_n(T_n^+)) - \sum_{n=1}^{N-1} (p_{n-1}(T_n^-), \delta\lambda_n)$$

recalling that $\delta \boldsymbol{u}_n(T_n^+) = \delta\lambda_n$, we get

$$\begin{aligned} \delta \widetilde{\mathcal{J}(\Lambda)}(\delta\Lambda) = & \sum_{n=0}^{N-1} (p_n(T_n^+), \delta \boldsymbol{u}_n(T_n^+)) - \sum_{n=1}^{N-1} (p_{n-1}(T_n^-), \delta\lambda_n), \\ = & \sum_{n=1}^{N-1} (p_n(T_n^+) - p_{n-1}(T_n^-), \delta\lambda_n) \qquad (42) \end{aligned}$$

since $\delta\lambda_0 = 0$. From these computations, we can easily implement a gradient method (or better a conjugate gradient method) an iteration of which reads: Assume p_n^k, λ_n^k are known, then

$$\lambda_n^{k+1} = \lambda_n^k - \rho(p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1.$$

It is quite easy to realize that the speed of convergence of the latter algorithm depends on the number N, of time steps T_n. Another way is to remark that the resolution of $\delta\widetilde{\mathcal{J}}(\Lambda) = 0$ can also be writen as

$$M^* \quad M \quad \Lambda \quad = \quad M^* \quad F$$

and note that the vector of jumps $p_n^k(T_n^+) - p_{n-1}^k(T_n^-)$ in the dual state is exactely equal to the vector $M^* Res^k$ i.e. to the residual of (17) at step k. The reason why the original gradient scheme is slow comes from the fact that the conditionning of M is $\mathcal{O}(N)$. In subsection 2.5 we have produced a good preconditioner for M : it was the matrix $\widetilde{M}^{-1}$ constructed thanks to the coarse corrector. This allows to forsee that $\widetilde{M}^{-1}(\widetilde{M}^{-1})^*$ may be a good preconditionner for $M^*\, M$ so that, a better approach should be the following preconditionned gradient method

$$\lambda_n^{k+1} = \lambda_n^k - \rho[\widetilde{M}^{-1}(\widetilde{M}^{-1})^*](p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1$$

The parareal algorithm for control problem can now be introduced. Lets us consider the following control problem

$$\frac{\partial \boldsymbol{u}}{\partial t} + A\boldsymbol{u} = B\boldsymbol{v} \tag{43}$$

$$\boldsymbol{u}(0) = \boldsymbol{u}^0, \tag{44}$$

where the control $\boldsymbol{v}$ (boundary or distributed) belongs to some space $\mathcal{U}$ and where B is an operator in $\mathcal{L}(\mathcal{U}; V')$. We assume that for any given $\boldsymbol{v}$, this problem is well posed.

We complement this problem with a cost functional to be minimized

$$\mathcal{J}(\boldsymbol{v}) = \frac{1}{2}\int_0^T \|\boldsymbol{v}\|^2_{\mathcal{U}} + \frac{\alpha}{2}\|\boldsymbol{u}(T) - \boldsymbol{u}^T\|^2,$$

where $\alpha > 0$, $\boldsymbol{u}^T$ is a target and the norm is the H norm if, for instance $V \subset H \subset V'$.

Based on the introduction of a virtual control to represent the strategy of parallelisation in time, we propose to modify slightly the cost functional $\mathcal{J}$ as follows

$$\mathcal{J}_\varepsilon(\boldsymbol{v}, \Lambda) = \frac{1}{2}\int_0^T \|\boldsymbol{v}\|^2_{\mathcal{U}} + \frac{\alpha}{2}\|\boldsymbol{u}_{N-1}(T) - \boldsymbol{u}^T\|^2 + \frac{1}{2\varepsilon\Delta T}\sum_{n=1}^{N-1}\|\boldsymbol{u}_{n-1}(T_n^-) - \lambda_n\|^2, \tag{45}$$

where ε is a small parameter.

In order to propose an algorithm for solving this minimization problem, we compute the derivative of $\mathcal{J}_\varepsilon(\boldsymbol{v}, \Lambda)$

$$\begin{aligned}\delta\mathcal{J}_\varepsilon(\boldsymbol{v}, \Lambda)(\delta\boldsymbol{v}, \delta\Lambda) = \; & \textstyle\sum_{n=0}^{N-1}\int_{T_n}^{T_{n+1}}(\boldsymbol{v}_n, \delta\boldsymbol{v}_n)_{\mathcal{U}} + \alpha(\boldsymbol{u}_{N-1}(T) - \boldsymbol{u}^T, \delta\boldsymbol{u}_{N-1}(T)) \\ & + \tfrac{1}{\varepsilon\Delta T}\textstyle\sum_{n=1}^{N-1}(\boldsymbol{u}_{n-1}(T_n^-) - \lambda_n, \delta\boldsymbol{u}_{n-1}(T_n^-) - \delta\lambda_n).\end{aligned} \tag{46}$$

The adjoint state is now defined as follows: let p_{N-1} be the solution over (T_{N-1}, T_N) of

$$\frac{\partial p_{N-1}}{\partial t} + A^* p_{N-1} = 0 \text{ over } (T_{N-1}, T_N), \tag{47}$$

$$p_{N-1}(T) = \alpha(\boldsymbol{u}_{N-1}(T) - \boldsymbol{u}^T), \tag{48}$$

and the collection p_n, $n = N-2, N-1, ..., 0$ of solutions of

$$\frac{\partial p_n}{\partial t} + A^* p_n = 0 \text{ over } (T_n, T_{n+1}) \tag{49}$$

$$p_n(T_{n+1}^-) = \frac{1}{\varepsilon\Delta T}(\boldsymbol{u}_n(T_{n+1}^-) - \lambda_{n+1}). \tag{50}$$

The derivative of $\mathcal{J}_\varepsilon$ can be given, following the same lines as before,

$$\delta\mathcal{J}_\varepsilon(\boldsymbol{v}, \Lambda)(\delta\boldsymbol{v}, \delta\Lambda) = \sum_{n=0}^{N-1}\int_{T_n}^{T_{n+1}}(\boldsymbol{v}_n + B^* p_n, \delta\boldsymbol{v}_n)_{\mathcal{U}} + \sum_{n=1}^{N-1}(p_n(T_n^+) - p_{n-1}(T_n^-), \delta\lambda_n),$$

An iteration of a gradient method thus reads: Assume $\boldsymbol{u}_n^k, p_n^k, \boldsymbol{v}_n^k, \lambda_n^k$ are known, then

$$\begin{aligned} \boldsymbol{v}_n^{k+1} &= \boldsymbol{v}_n^k - \rho(\boldsymbol{v}_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1}) \\ \lambda_n^{k+1} &= \lambda_n^k - \rho(p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1. \end{aligned}$$

that of course cannot have a convergence rate independent of N. The following preconditionned gradient method takes into account the parareal framework

$$\begin{aligned} \boldsymbol{v}_n^{k+1} &= \boldsymbol{v}_n^k - \rho(\boldsymbol{v}_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1}) \\ \lambda_n^{k+1} &= \lambda_n^k - \rho[\widetilde{M}^{-1}(\widetilde{M}^{-1})^*](p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1 \end{aligned}$$

The preliminary numerical results presented in [27] show an impressive speed up for the parallel implementation of this preconditioner in a case of a linear parabolic equation.

Another route to couple parareal algorithm with control strategy is given in the paper [24] where we decompose the control problem over $[0, T]$ on a series of independent control problems over each intervals $[T_n, T_{n+1}]$ with intermediate initial conditions at time T_n and intermediate target values at time T_{n+1} that have to be designed following an iterative procedure. The method has been proposed and implemented first for the control of Schrödinger equation in [24] with a very nice implementation of a harder problem in [28] and proves also to be very successful. This frame with an hyperbolic equation where the adjoint equation is the same as the primal problem makes the design and the analysis of the iterative method to define iteratively the intermediate initial condition and intermediate target values more easy than in the parabolic case. This case is under investigation.

4.3 Complexity and parallelism efficiency - part 2

We have explained in subsection 2.4 that the full efficiency, i.e. divide the simulation time by a factor close to the number of processors used for the implementation of the plain parareal algorithm is only obtained if the cost $\mathcal{C}_\mathcal{G}$ of the coarse solver is negligible with respect to the cost $\mathcal{C}_\mathcal{F}$ of the fine solver, and also if the algorithm converges with $K = 2$ involving only one fine sweep. For complex problem, the first item is easy to meet but the second one is more involved and can lead to conclude that the parareal algorithm should not be used, at least if an alternative does exist (as the domain decomposition that is now fully efficient).

What we have proposed in the previous subsections is to intertwine the parareal iterations with some other iterations. This allows to reduce drastically the cost of the fine solver which makes sense since during the first iterations, where the seed values Λ^k are not yet converged, it is indeed not useful to pay the full price for the resolution of the fine resolution of the problem between time T_N to time T_{n+1}.

Such a strategy leads to a cost

$$KN\mathcal{C}_\mathcal{G} + (K-1)\frac{\Delta T}{\delta t}\widetilde{\mathcal{C}}_\mathcal{F} \tag{51}$$

where $\widetilde{\mathcal{C}}_{\mathcal{F}} << \mathcal{C}_{\mathcal{F}}$ is the coast of a unconverged fine solver, that has still to be compared with the same coast $N\frac{\Delta T}{\delta t}\mathcal{C}_{\mathcal{F}}$ of the original fine solver.

Of course the parallel efficiency is much improved and the full scalability can be obtained. This is what we noticed for the parareal/domain decomposition coupling: indeed, for a given convergence tolerance, we have reported in [26] that the number of iterations required for convergence remains constant if, for a given fine problem with a given δt we half the size of ΔT because we have twice the number of processors. Since the number of iterations remains constant, this means that the global time to wait is divided by 2.

We have not performed much precise numerical analysis of this matter because it is quite involved but it is certainly interesting to do it.

5 Stability issues

5.1 Basic results

We have already reported a problem that appears for the parareal simulation of the Burgers' problem when the viscosity diminishes. This problem has been previously documented in some papers among them [3], [29], [22], [8]. Note that this instability is not exponential, as is most often the case if the CFL condition is not satisfied, but is is polynomial as we shall see further. This only polynomial instability is effective only if K is large, i.e. if the convergence is not achieved for few iterations of the parareal algorithm. This is another reason for using the hierarchical strategy of Bal [2] that we presented in subsection 2.4 that allows to require $K = 2$ only. However, we do need other strategies in order to cure this problem for large K since the number of iterations will grow as we have seen if the parareal algorithm is intertwined with other iterative methods.

In order to understand the roots of the problem, let us sketch the analysis that is presented in [3]. Going back to the convergence analysis in subsection 3.1, we notice in equation (25) the presence of a factor $\binom{n}{k}$ that behaves like n^k and, if multiplied by a factor that is not going to zero may lead to instabilities. This is exactly what may happen, and what happens for lowly dissipative problems with eigenvalues with large imaginary parts.

5.2 A possible way out for convection dominated problems.

As we have just seen, it is the high frequency that are responsible for the polynomial instability. The idea of filtering out the high modes, where it is possible, i.e. in the coarse propagator, is thus quite natural. As we already mentioned in subsection 3.3, it does improve the stability of the parareal algorithm, without reducing the convergence rate. It improves but it does not solve the problem since with such a ingredient the case $\nu = 0.1$ ($N = 512$, $N_{\mathcal{G}} = 32$, $\delta t = 210^{-4}$, $\Delta T = 10^{-2}$) is not stable (blow up after

$k = 3$).

What we propose next is to try to treat as exactly as possible the pure convective part of the operator $\mathcal{A}$. Taking into account the fact that our splitting scheme decouples the convection part and the diffusion part, we propose to replace, in the coarse simulation only, the convection step based on the solution of

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

by two steps, one based on solving

$$\frac{\partial u}{\partial t} + (u-2)\frac{\partial u}{\partial x} = 0$$

followed by one based on the exact resolution of

$$\frac{\partial u}{\partial t} + 2\frac{\partial u}{\partial x} = 0$$

that can be implemented by advancing $u = \sum_{n=-N}^{N} \hat{u}^n e^{inx}$ by $\tilde{u} = \sum_{n=-N}^{N} \hat{u}^n e^{in(x-2t)}$.

The idea is to separate, in the coarse solver, the convection part into an averaged uniform part (here a uniform convection equal to 2 which is the Rankine Hugoniot velocity) that will be treated almost exactly and the difference with this averaged part that is treated through the standard coarse solver. We refer to [20] for further simulations. Note that there is some robustness with respect to this averaged velocity. The simulation with 1.5 behaves similarly.

This strategy is able to stabilize the previous simulation that was unstable with $\nu = 0.1$. We have also been able to simulate the Burgers equation with $\nu = 10^{-2}$ with a fine spectral accuracy based on $N = 1024$, $N_{\mathcal{G}} = 32$, $\delta t = 10^{-4}$ and $\Delta T = 5.10^{-2}$. The simulation is stable and the error at final time ($T = 2$) is monotically decreasing as a function of the iterations.

6 Conclusion

In this paper we have presented the basics of the parareal algorithm and illustrated its behavior. The plain algorithm is already able to achieve important speed up for the restitution of the simulation. The full efficiency though is rarely obtain for the plain algorithm. We have presented some way of intertwining the parareal algorithm with other independent iterative methods. The full efficiency can then be achieved, this is the case for example when the parareal algorithm is coupled with domain decomposition techniques and optimal control tools. Finally, a severe limitation of the parareal algorithm related to a lack of stability of the approach for differential problems with eigenvalues close to imaginary axis (convection dominated problems) can be removed by a combined effect of an appropriate coarse solver and a special treatment of the convective part in the coarse solver also.

Ackowledgements This work is partially supported by the French national Network Research Program Agence Nationale de la Recherche ANR through the project PITAC 2006.

References

[1] L. Baffico, S. Benard, Y. Maday, G. Turinici and G. Zérah, *"Parallel in time molecular dynamics simulations"*, Phys. Rev. E., **66**, 057701, 2002.

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

[3] G. Bal, *"On the convergence and the stability of the parareal algorithm to solve partial differential equations"*, Lecture Notes in Computational Science and Engineering , **40**, 425–432; R. Kornhuber ; R. Hoppe; J. Priaux ; O. Pironneau; O. Widlund; J. Xu (Eds.) 2005.

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

[5] K. Burrage, *"Parallel and sequential methods for ordinary differential equations"*, Numerical Mathematics and Scientific Computation, Oxford Science Publications, The Clarendon Press Oxford University Press, New York, 1995.

[6] A. Chorin, O Hald and R. Kupferman, *"Optimal prediction with memory"*, Phys. D, **166**, 3-4, 239–257, 2002.

[7] A. Chorin, *" Averaging and renormalization for the Korteveg-deVries-Burgers equation"*, Proc. Natl. Acad. Sci. USA, **100** 17, 9674–9679, 2003.

[8] C. Farhat, J. Cortial, C. Dastillung and H. Bavestrello, *"Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses"*, Internat. J. Numer. Methods Engrg., **67**, 5, 697–724, 2006.

[9] P. Fischer, F. Hecht, Y. Maday, *" A parareal in time approximation of the Navier Stokes equations"*, Lecture Notes in Computational Science and Engineering , **40**, 433–440; R. Kornhuber ; R. Hoppe; J. Priaux ; O. Pironneau; O. Widlund; J. Xu (Eds.) 2005.

[10] M. Gander and S. Vandewalle, *" Analysis of the parareal time-parallel time-integration method"*, SIAM J. Sci. Comput., **29**, 2, 556–578, 2007.

[11] M. Gander and S. Vandewalle, *" On the superlinear and linear convergence of the parareal algorithm"*, Domain decomposition methods in science and engineering XVI, Lect. Notes Comput. Sci. Eng., **55**, 291–298, Springer, Berlin, 2007.

[12] C. W. Gear, *" Parallel methods for ordinary differential equations"*, Calcolo, **25**, 1-2, 1–20, 1988.

[13] R. Gueta and Y. Maday, in preparation, 2008.

[14] MA Grepl and AT Patera, *"A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations"*, Mathematical Modelling and Numerical Analysis (M2AN), **39** 1, 157–181, 2005.

[15] W. Hackbusch, *"Parabolic multigrid methods"*, Computing methods in applied sciences and engineering, VI (Versailles, 1983), 189–197, North-Holland, Amsterdam, 1984.
[16] S. M. Kaber and Y. Maday, in preparation, 2008.
[17] E. Lelarasmee, A. E. Ruehli and A. L. Sangiovanni-Vincentelli, *"The waveform relaxation me- thod for time-domain analysis of large scale integrated circuits"*, IEEE Trans. on CAD of IC and Syst., **1**, 131-145, 1982.
[18] J.-L. Lions , Y. Maday and G. Turinici, *"Résolution d'EDP par un schéma en temps pararéel"*, C.R.Acad Sci. Paris Sér. I Math, **332**, 661—668, 2001.
[19] Y. Maday, *"Parareal in time algorithm for kinetic systems based on model reduction"*, in *"High-dimensional partial differential equations in science and engineering"*, CRM Proc. Lecture Notes **41**, 183–194, Amer. Math. Soc. Providence, RI, 2007.
[20] Y. Maday, in preparation, 2008.
[21] Y. Maday and E. M. Ronquist *"Parallelization in time through tensor-product space-time solvers"*, C. R. Math. Acad. Sci. Paris, **346**, 1-2, 113–118, 2008.
[22] Y. Maday , E. M. Ronquist and G. Staff *"The parareal-in-time algorithm: basics, stability and more"*, submitted.
[23] Y. Maday and D.V. Rovas, *"Reduced Basis Output bounds methods for parabilic problems"* , IMA J Appl Math, **26** 3, 423 – 445, 2006 .
[24] Y. Maday, J. Salomon and G. Turinici, *"Monotonic parareal control for quantum systems"* SIAM J. Numer. Anal., **45** 6, 2468–2482, 2007.
[25] Y. Maday and G. Turinici, *"Parallel in time algorithms for quantum control: the parareal time discretization scheme"* Int. J. Quant. Chem., **93** 3, 223–228, 2003.
[26] Y. Maday and G. Turinici, *"The parareal in time iterative solver: a further direction to parallel implementation"* Lecture Notes in Computational Science and Engineering , **40**; R. Kornhuber ; R. Hoppe; J. Priaux ; O. Pironneau; O. Widlund; J. Xu (Eds.) 2004
[27] Y. Maday and G. Turinici, *" A parareal in time procedure for the control of partial differential equations"* C. R. Math. Acad. Sci. Paris , **335**, 387–392, 2002.
[28] J. Salomon, C.M. Dion and G. Turinici, *"Optimal molecular alignment and orientation through rotational ladder climbing"*, J. Chem. Phys., **123**, 14, 144310, 2005.
[29] G. Staff and E. M. Rønquist, *"Stability of the parareal algorithm"* , Lecture Notes in Computational Science and Engineering , **40**, 449–456; R. Kornhuber ; R. Hoppe; J. Priaux ; O. Pironneau; O. Widlund; J. Xu (Eds.) 2005.