PARALLELIZATION IN TIME OF (STOCHASTIC) ORDINARY DIFFERENTIAL EQUATIONS

GUILLAUME BAL *

Abstract. This paper analyzes some properties of the parareal algorithm, which can be used to parallelize the time discretizations of differential equations. The parareal algorithm proceeds as follows. Firstly, a coarse time step is used to solve the equation sequentially on a given time interval. Secondly, a fine discretization is used to solve the evolution equation on each coarse time step. This step may be performed in parallel. Finally, the local errors between the coarse and fine solutions are propagated through the whole time interval by using the sequential coarse algorithm. The latter two steps may be repeated $k-1$ times. The main result is that this two-level parallel algorithm replaces a discretization of order m by a discretization of order km. Convergence is analyzed for ordinary differential equations and for the Euler discretization of stochastic ordinary differential equations. Optimal implementations that maximize speedup or system efficiency are then considered for the two-level as well as more general multi-level parareal algorithms. Some examples of application of the algorithm, such as the integration of equations over long times or the filtering problem, are then analyzed numerically.

Key words. Time discretization, parallelization, ordinary differential equations, stochastic ordinary differential equations, long time integration, filtering problem.

AMS subject classifications. 65L05 65Y05 65C30

1. Introduction.

Evolution equations posed on an interval $[0,T]$ with initial condition at time $t=0$ require the knowledge of the solution at time $\tau>0$ to calculate the solution at a later time $t>\tau$. This obvious fact is an impediment to "classical" parallelizations, where N independent processors would be used to solve the equation on N subsets of the whole interval of time $[0,T]$. Because of computational time and memory issues, parallelization has become an indispensable tool in the solution of large scale problems. Several techniques have thus been developed to address the parallelization of time discretizations. An interesting but somewhat limited technique consists of parallelizing the computations required at every time step for high order methods [9, 18, 21]. Another approach is based on replacing the initial value problem by a boundary value problem that is more amenable to parallelization [1, 2, 8, 16, 22]. The waveform relaxation technique may also be used to parallelize the numerical simulation of ODE systems. It is a parallelism *across* the ODE system, using the notation in [20], obtained by splitting the ODE system into smaller sub-systems. We refer to [9, 11, 20] for details on the method.

We consider here the *parareal* algorithm, which was pioneered in [13], subsequently modified in [6], and used in several applications including control theory [14], molecular dynamics [4], and fluid-structure interactions [10] for instance. The algorithm consists of discretizing an interval of time by using a fine time step, which governs the required final accuracy of the method, and a coarse time step, introduced e.g. to maximize speedup (maximal gain in time) or to optimally use a given number of processors (system efficiency). The algorithm is described as follows. The first time step consists of solving the evolution equation on the coarse discretization iteratively. This step cannot be done in parallel (although the techniques described in the literature cited above could be applied here). The evolution equation is then solved in parallel on each coarse interval. This step is fully parallelizable provided that we have as many

*Department of Applied Physics and Applied Mathematics, Columbia University, New York NY, 10027; gb2030@columbia.edu

processors as we have coarse time intervals. The last step consists of propagating the error between the coarse and fine discretizations through the whole time interval by using one more time the non-parallel coarse discretization. The latter two steps may be repeated sequentially $k-1$ times. The main advantage of the algorithm is that it replaces a scheme of order m on the coarse grid by a scheme of order km on that same grid.

This paper concentrates on the parallelization in time of ordinary differential equations and a simple stochastic ordinary differential equation. Convergence is presented for sufficiently regular ordinary differential equations. It is also interesting to see how the parareal algorithm deals with "stiff" equations. The stochastic ordinary differential equations act as a proxy for such stiff equations. We show that the "paths" of the stochastic solutions are indeed well captured by the parareal algorithm, which shows its robustness. An interesting application, where one is interested in stochastic paths, is optimal filtering. Note that the parareal algorithm is not useful when one is interested in the law of the stochastic solution. Indeed, natural parallelization over stochastic realizations, as in Monte Carlo method, is optimal as far as system efficiency is concerned, and is much more efficient than parallelization based on the parareal algorithm. As in the case of deterministic ordinary differential equations, the advantage of the parareal algorithm is that it allows us to approximate the "path" of the solution faster than with sequentially-based method provided that a sufficient number of processors is available. Note also that although some of the analysis presented here may apply to partial differential equations, high frequency modes generated by spatial discretizations require special treatment, as is shown for instance in [5, 6], and will not be dealt with here.

In its simplest implementation, the parareal algorithm briefly described above involves four main parameters: a fine time step δT that is eventually tied to the accuracy we expect from the algorithm, a coarse time step ΔT, which we can choose freely, a number k of iterations of the algorithm, and a number M of successive times the parareal algorithm is used on intervals of size τ/M to solve an equation on the interval $[0,\tau]$. This paper describes how the parameters should be chosen to maximize speedup and system efficiency of the parareal algorithm. When τ is of order $O(1)$ independent of δT, optimal speedups are obtained for $k=2$. The coarse time step is then chosen as the square root of the fine time step. For evolution on longer times, where τ may be modeled as $(\delta T)^{-\gamma}$, other choices lead to optimal speedups or system efficiencies. With these optimal choices of parameters, the parareal algorithm may be particularly adapted to the solution of small systems of ordinary differential equations over very long times, where other parallelization techniques such as waveform relaxation may not be as efficient.

The organization of the paper is as follows. Section 2 recalls well-known results on ordinary differential equations and introduces the notation used in subsequent sections. The parareal algorithm is analyzed in the framework of ordinary differential equations in section 3. Section 4 generalizes the results to a simple stochastic ordinary differential equation. To simplify the presentation, only the Euler discretization is considered. Section 5 is concerned with the practical implementation of the algorithm either to maximize the speedup provided that a sufficient number of processors is available, or to maximize the relative time where processors are active (system efficiency). The two-level algorithm considered so far is generalized to a multi-level algorithm, which allows for higher maximal speedup. Finally, section 6 presents very simple numerical simulations that indicate where the algorithm may prove valuable.

Among them are the solution of highly oscillatory problems over long intervals of time and the filtering problem, where some quantities of interest are reconstructed from noisy measurements.

2. Ordinary Differential Equations. This section recalls well-known results on the discretization of ordinary differential equations. It also serves as an introduction to the notation used in subsequent sections. We consider a system of ordinary differential equations of the form

$$\begin{aligned} &dX(t) = b(t, X(t))dt, \qquad t \in [0, T] \\ &X(0) = X_0, \end{aligned} \tag{2.1}$$

for $T > 0$. Here, $X(t)$ is a vector in $\mathbb{R}^d$ and b a function from $[0, T] \times \mathbb{R}^d$ to $\mathbb{R}^d$ satisfying the regularity constraints

$$\begin{aligned} &|b(t, x)| \leq C(1 + |x|), \quad x \in \mathbb{R}^d,\, t \in [0, T], \\ &|b(t, x) - b(t, y)| \leq C|x - y|, \quad x, y \in \mathbb{R}^d,\, t \in [0, T], \end{aligned} \tag{2.2}$$

where $|\cdot|$ is any norm on $\mathbb{R}^d$. Here and below, C denotes a positive universal constant. It is a classical result [3] that the above restrictions on $b(t, x)$ ensure existence and uniqueness of a solution to (2.1).

To solve this equation numerically on $[0, T]$, we set a time step $\Delta T > 0$ and a discretization $T^n = n\Delta T$ of the interval $[0, T] = [T^0, T^N]$. We introduce the solution function $g(t, X) = g(t, X; \Delta T)$ defined by

$$g(t, X) = X(t + \Delta T), \tag{2.3}$$

where

$$\begin{aligned} &dX(\tau) = b(\tau, X(\tau))d\tau, \qquad \tau \geq t \\ &X(t) = X. \end{aligned} \tag{2.4}$$

The ordinary differential equation (2.1) is solved numerically by approximating the function g by a discretized version $g_\Delta(t, X)$. We then define $X^0 = X_0$ and iteratively

$$X^{n+1} = g_\Delta(T^n, X^n) \quad \text{ for } 0 \leq n \leq N - 1. \tag{2.5}$$

The accuracy of the approximation is obtained by analyzing

$$X(T^{n+1}) - X^{n+1} = g(T^n, X(T^n)) - g_\Delta(T^n, X^n) \tag{2.6}$$

iteratively. Let us assume for instance that we have a discretization of order $m > 0$ so that

$$\sup_{1 \leq n \leq N; x \in \mathbb{R}^d} |g(T^n, x) - g_\Delta(T^n, x)| \leq C(\Delta T)^{m+1}(1 + |x|). \tag{2.7}$$

This might require additional regularity assumptions on the function b, which we do not dwell on. We deduce from (2.2) the following Lipschitz regularity

$$\sup_{t \in [0, T]; x, y \in \mathbb{R}^d} |g(t, x) - g(t, y)| \leq (1 + C\Delta T)|x - y|. \tag{2.8}$$

From (2.7) and (2.8), we see that a similar result to (2.8) also holds for g_Δ. This implies that $|X^n| \leq C(1+|X_0|)$ for all $0 \leq n \leq N$. We can then use (2.7) and (2.8) to deduce from (2.6) that

$$|X(T^{n+1}) - X^{n+1}| \leq (1+C\Delta T)|X(T^n) - X^n| + C(\Delta T)^{m+1}(1+|X^n|). \tag{2.9}$$

From this we obtain that

$$|X(T^n) - X^n| \leq C(1+C\Delta T)^n n(\Delta T)^{m+1}(1+|X_0|) \leq Cn(\Delta T)^{m+1}(1+|X_0|), \tag{2.10}$$

and, for instance, that

$$|X(T) - X^N| \leq CT(\Delta T)^m(1+|X_0|). \tag{2.11}$$

This concludes the analysis of the iterative algorithm (2.5) to approximately solve the system (2.1).

3. Parallelization in Time. Let us now assume that we have as many processors to calculate fine solutions as we have coarse intervals $[T^n, T^{n+1}]$. Solving the problem accurately on every interval $[T^n, T^{n+1}]$ could then theoretically be done in parallel. Since the solution $X(T^n)$ (or an approximation of it) is required to calculate $X(t)$ on $[T^n, T^{n+1}]$, parallelizing time discretizations is however not straightforward.

As an adaptation of an algorithm pioneered in [13], a two-level parallelization scheme has been proposed in [6] This section analyzes this algorithm to solve general systems of ordinary differential equations. The parallelization scheme is defined as follows. The scheme is initialized by the solution of the iterative algorithm (2.5) of the preceding section

$$X_1^{n+1} = g_\Delta(T^n, X_1^n) \quad \text{for } 0 \leq n \leq N-1, \tag{3.1}$$

with $X_1^0 = X_0$. We then define more accurate solutions as the following two-level iterative process. Let us assume that $k \geq 1$ and that X_k^n is known for $0 \leq n \leq N$. First, we calculate jumps by solving local problems "exactly",

$$S_k^n = g(T^{n-1}, X_k^{n-1}) - X_k^n, \qquad 1 \leq n \leq N. \tag{3.2}$$

The jump S_k^n is the difference between the local "exact" solution at time T^n starting from X_k^{n-1} at T^{n-1} and the predictor X_k^n obtained after k parareal iterations. Second, we propagate these jumps by using the coarse solver

$$X_{k+1}^{n+1} = \sum_{l=1}^{k} S_l^{n+1} + g_\Delta(T^n, X_{k+1}^n), \qquad 0 \leq n \leq N-1, \tag{3.3}$$

with initial condition $X_{k+1}^0 = X_0$.

This is the form implemented in the numerical simulations to calculate X_{k+1}^n from X_k^n, requiring to use a non-parallel coarse discretization to propagate the jumps and a parallel local fine discretization to calculate these jumps. In practice, the local calculations also need to be discretized. To simplify the presentation, we assume that the local calculations are performed exactly. The generalization to the discrete case is straightforward; see section 5.

The above form can be simplified. We indeed verify the following recurrence relation (see Fig.3.1)

$$X_{k+1}^{n+1} = g_\Delta(T^n, X_{k+1}^n) + \delta g(T^n, X_k^n), \tag{3.4}$$

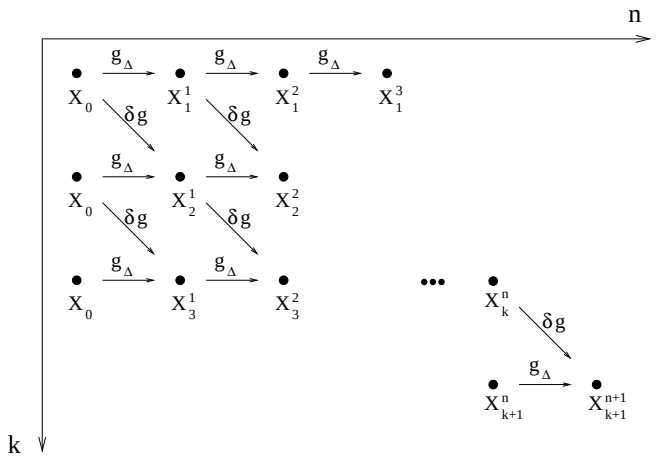

FIG. 3.1. *The recurrence relation (3.4).*

where we have introduced the difference operator

$$\delta g(T^n, X) = g(T^n, X) - g_\Delta(T^n, X). \tag{3.5}$$

This prediction-correction formula was already used in [4, 6]. We can recast the above relation as

$$\begin{aligned} X_{k+1}^{n+1} - X(T^{n+1}) &= [g_\Delta(T^n, X_{k+1}^n) - g_\Delta(T^n, X(T^n))] \\ &+ [\delta g(T^n, X_k^n) - \delta g(T^n, X(T^n))]. \end{aligned} \tag{3.6}$$

Let us assume that δg verifies the following regularity hypothesis

$$\sup_{1\le n\le N; x,y\in\mathbb{R}^d} |\delta g(T^n, x) - \delta g(T^n, y)| \le C(\Delta T)^{m+1}|x-y|. \tag{3.7}$$

This hypothesis means that g_Δ not only is a discretization of order m of g as in (2.7), but also verifies a Lipschitz regularity in its initial condition. Again, this requirement relies on a sufficient regularity of the function $b(t,x)$ and the discretization scheme. We deduce from (2.8) and (3.7) that

$$\sup_{t\in[0,T]; x,y\in\mathbb{R}^d} |g_\Delta(t,x) - g_\Delta(t,y)| \le (1 + C\Delta T)|x-y|. \tag{3.8}$$

Introducing the error between the exact solution and the iterative solution at step k

$$\varepsilon_k^n = X_k^n - X(T^n), \tag{3.9}$$

we deduce from (3.6), (3.7) and (3.8) that

$$|\varepsilon_{k+1}^{n+1}| \le (1 + C\Delta T)|\varepsilon_{k+1}^n| + C(\Delta T)^{m+1}|\varepsilon_k^n|. \tag{3.10}$$

Let us now define

$$\theta_k^n = (1 + C\Delta T)^{k-n} C(\Delta T)^{-k(m+1)}|\varepsilon_k^n|. \tag{3.11}$$

We observe that (3.10) implies

$$\theta_{k+1}^{n+1} \leq \theta_{k+1}^{n} + \theta_{k}^{n}.$$

Since θ_0^n is bounded (above and below) as well as $(1+C\Delta T)^N$, we deduce that

$$|\varepsilon_k^n| = |X_k^n - X(T^n)| \leq C(\Delta T)^{k(m+1)} \binom{n}{k}. \tag{3.12}$$

For $n = N$ and $k = O(1)$, this implies that

$$|X_k^N - X(T)| \leq CT^k (\Delta T)^{km} |X_0|. \tag{3.13}$$

This error is to be compared with (2.11). The iterative scheme (3.4) replaces a discretization of order m by a discretization of order km after $k-1$ iterations, involving k coarse solutions and $k-1$ fine solutions calculated in parallel.

This iterative algorithm allows for substantial gains in computational time provided that a sufficiently large number of processors is available. The analysis of this speedup is postponed to section 5.

4. Parallelization for Stochastic Ordinary Differential Equations. The analysis performed in the preceding section concerns smooth systems of ordinary differential equations. It is unclear how the technique can be adapted to stiff equations. We show in this section that the parareal algorithm still works for a simple discretization of stochastic ordinary differential equations.

We consider here the system of stochastic ordinary differential equations:

$$\begin{aligned} &dX(t) = b(t, X(t))dt + \sigma(t, X(t))dB(t), \qquad t \in [0, T]. \\ &X(0) = X_0, \end{aligned} \tag{4.1}$$

where, b and X are defined as in the preceding section, $B(t)$ is $m-$dimensional Brownian motion defined on a probability space $(\Omega, \mathcal{F}, P)$, and $\sigma(t, X)$ maps $[0, T] \times \mathbb{R}^d$ to $\mathcal{M}_{d\times m}$, the set of $d \times m$ matrices. We assume that b and σ are such that

$$\begin{aligned} &|b(t,x)| + |\sigma(t,x)| \leq C(1+|x|), \quad x \in \mathbb{R}^d,\, t \in [0,T] \\ &|b(t,x) - b(t,y)| + |\sigma(t,x) - \sigma(t,y)| \leq C|x-y|, \quad x, y \in \mathbb{R}^d,\, t \in [0,T]. \end{aligned} \tag{4.2}$$

This implies existence and uniqueness of a strong solution $X(t)$ to the above stochastic ordinary differential equation [19]. We denote by $\mathcal{F}_t$ a filtration adapted to Brownian motion $B(t)$ (see [19]). To simplify the presentation we assume that the dimensions $d = 1$ and $m = 1$. The generalization to higher values of d and m present no theoretical difficulty for the numerical scheme considered here (see [12, 17]).

In integral form, (4.1) between T^n and T^{n+1} is equivalent to

$$\begin{aligned} X(T^{n+1}) &= X(T^n) + \int_{T^n}^{T^{n+1}} b(t, X(t))dt + \int_{T^n}^{T^{n+1}} \sigma(t, X(t))dB(t) \\ &= g(T^n, X(T^n)). \end{aligned} \tag{4.3}$$

To solve (4.1) numerically, we consider in this paper only the explicit Euler scheme, defined by

$$\begin{aligned} X^{n+1} &= X^n + b(T^n, X^n)\Delta T + \sigma(T^n, X^n)(B(T^{n+1}) - B(T^n)) \\ &= g_\Delta(T^n, X^n). \end{aligned} \tag{4.4}$$

The accuracy of this scheme has been analyzed in the literature [12, 15, 17]. We aim at generalizing the analysis of section 3 on the time parallelization scheme to the discretization (4.4) of (4.1). More accurate schemes than the Euler explicit discretization can also be analyzed using the techniques of this paper and of the above references although we do not consider them here.

Let us define the operators

$$\mathcal{L}f = \frac{\partial f}{\partial t} + b\frac{\partial f}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2 f}{\partial x^2}, \qquad \Lambda f = \sigma\frac{\partial f}{\partial x}. \tag{4.5}$$

We deduce from the Itô formula [19] that

$$\begin{aligned} db(t,X(t)) &= \mathcal{L}b(t,X(t))dt + \Lambda b(t,X(t))dB(t) \\ d\sigma(t,X(t)) &= \mathcal{L}\sigma(t,X(t))dt + \Lambda\sigma(t,X(t))dB(t). \end{aligned}$$

We use these relations to recast (4.3) as

$$\begin{aligned} X(T^{n+1}) = X(T^n) &+ b(T^n,X(T^n))\Delta T + \sigma(T^n,X(T^n))(B(T^{n+1})-B(T^n)) \\ &+ \int_{T^n}^{T^{n+1}}\int_{T^n}^{t}\Big(\mathcal{L}b(s,X(s))dsdt + \Lambda b(s,X(s))dB(s)dt\Big) \\ &+ \int_{T^n}^{T^{n+1}}\int_{T^n}^{t}\Big(\mathcal{L}\sigma(s,X(s))dsdB(t) + \Lambda\sigma(s,X(s))dB(s)dB(t)\Big). \end{aligned} \tag{4.6}$$

The local error between (4.3) and (4.4), or equivalently the difference

$$\delta g(T^n,X(T^n)) = g(T^n,X(T^n)) - g_\Delta(T^n,X(T^n)),$$

is then given by the two integral terms in (4.6).

With these generalized definitions for g and g_Δ, we define X_k^n as the solution to the time parallelization scheme (3.1)-(3.3). Recall from (3.6) that

$$\begin{aligned} X_{k+1}^{n+1} - X(T^{n+1}) &= [g_\Delta(T^n,X_{k+1}^n) - g_\Delta(T^n,X(T^n))] \\ &+ [\delta g(T^n,X_k^n) - \delta g(T^n,X(T^n))]. \end{aligned}$$

We now aim at estimating the quantity

$$\eta_k^n = \mathbb{E}[(X_k^n - X(T^n))^2]. \tag{4.7}$$

Here $\mathbb{E}$ denotes expectation with respect to the probability measure P. We have

$$\begin{aligned} \eta_{k+1}^{n+1} \le\ & \mathbb{E}\Big[\big(g_\Delta(T^n,X_{k+1}^n) - g_\Delta(T^n,X(T^n))\big)^2\Big] \\ &+\mathbb{E}\Big[\big(\delta g(T^n,X_k^n) - \delta g(T^n,X(T^n))\big)^2\Big] \\ &+2\Big|\mathbb{E}\Big[\big(g_\Delta(T^n,X_{k+1}^n) - g_\Delta(T^n,X(T^n))\big)\big(\delta g(T^n,X_k^n) - \delta g(T^n,X(T^n))\big)\Big]\Big| \end{aligned} \tag{4.8}$$

Let us define $a(T^n) = b(T^n,X_{k+1}^n) - b(T^n,X(T^n))$ and $\gamma(T^n) = \sigma(T^n,X_{k+1}^n) - \sigma(T^n,X(T^n))$. We compute

$$\begin{aligned} & \mathbb{E}\big[\big(g_\Delta(T^n,X_{k+1}^n) - g_\Delta(T^n,X(T^n))\big)^2\big] \\ =\ & \mathbb{E}\big[\big(X_{k+1}^n - X(T^n) + a(T^n)\Delta T + (\gamma(T^n))(B(T^{n+1})-B(T^n))\big)^2\big] \\ =\ & \mathbb{E}\big[\big(X_{k+1}^n - X(T^n) + a(T^n)\Delta T\big)^2\big] + \Delta T\mathbb{E}\big[\gamma^2(T^n)\big] \\ \le\ & (1+C\Delta T)\mathbb{E}\big[\big(X_{k+1}^n - X(T^n)\big)^2\big] = (1+C\Delta T)\eta_{k+1}^n. \end{aligned}$$

Here we have used (4.2) and $\mathbb{E}[B(T^{n+1}) - B(T^n)|\mathcal{F}_{T^n}] = 0$.

Let us now recast $\delta g(T^n, X_k^n) - \delta g(T^n, X(T^n))$ as

$$\int_{T^n}^{T^{n+1}} \int_{T^n}^{t} \Big(a_1(s) ds dt + a_2(s) dB(s) dt + a_3(s) ds dB(t) + a_4(s) dB(s) dB(t) \Big),$$

where

$$\begin{aligned} a_1(s) &= \mathcal{L}b(s, X(s)) - \mathcal{L}b(s, Y(s)), \qquad a_2(s) = \Lambda b(s, X(s)) - \Lambda b(s, Y(s)) \\ a_3(s) &= \mathcal{L}\sigma(s, X(s)) - \mathcal{L}\sigma(s, Y(s)), \qquad a_4(s) = \Lambda \sigma(s, X(s)) - \Lambda \sigma(s, Y(s)), \end{aligned}$$

where $X(s)$ and $Y(s)$ solve (4.1) with initial conditions given by X_k^n and $X(T^n)$, respectively. We assume that all the functions involved in the definition of the terms $a_i(s)$ are Lipschitz with respect to their second variable:

$$\begin{aligned} |\mathcal{L}b(t,x) - \mathcal{L}b(t,y)| + |\mathcal{L}\sigma(t,x) - \mathcal{L}\sigma(t,y)| + |\Lambda b(t,x) - \Lambda b(t,y)| \\ + |\Lambda \sigma(t,x) - \Lambda \sigma(t,y)| \leq C|x-y|, \quad x, y \in \mathbb{R},\, t \in [0,T], \end{aligned} \tag{4.9}$$

for some constant C. We deduce from this Lipschitz regularity and from the stability of the solution to (4.1) that for all $s \in [T^n, T^{n+1}]$,

$$\mathbb{E}[(a_i(s))^2] \leq C\mathbb{E}[|X_k^n - X(T^n)|^2] = C\eta_k^n. \tag{4.10}$$

Let us also recall that for non-anticipating functions $f(t,\omega)$ and $g(t,\omega)$ (see [19]), we have the Itô isometry

$$\mathbb{E}\Big[\int_t^\tau f(s,\omega) dB(s) \int_t^\tau g(s,\omega) dB(s) \Big] = \mathbb{E}\Big[\int_t^\tau f(s,\omega) g(s,\omega) ds \Big], \quad t < \tau.$$

Several applications of this Itô isometry, the Cauchy-Schwarz inequality, and the above bounds show then that

$$\mathbb{E}[|\delta g(T^n, X_{k+1}^n) - \delta g(T^n, X_k^n)|^2] \leq C(\Delta T)^\alpha \eta_k^n, \tag{4.11}$$

with $\alpha = 2$ in general, $\alpha = 3$ if $a_4 = 0$, and $\alpha = 4$ if $a_2 = a_3 = a_4 = 0$ (as in the deterministic case or the linear case, i.e., b and σ constant).

Interestingly enough, the cross term in (4.8) cannot be estimated satisfactorily from the two other terms and an estimate of the form $(\mathbb{E}[(a+b)^2])^{1/2} \leq (\mathbb{E}[a^2])^{1/2} + (\mathbb{E}[b^2])^{1/2}$ would imply that the Euler scheme does not converge when $\alpha = 2$ above. The reason is that the biggest terms vanish in the expectation of the cross term, given by

$$\begin{aligned} &\mathbb{E}\Big[\big(g_\Delta(T^n, X_{k+1}^n) - g_\Delta(T^n, X(T^n)) \big) \big(\delta g(T^n, X_k^n) - \delta g(T^n, X(T^n)) \big) \Big] \\ &= \mathbb{E}\Big[\Big((X_{k+1}^n - X(T^n)) + \Delta T a(T^n) \Big) \int_{T^n}^{T^{n+1}} \int_{T^n}^{t} a_1(s) ds dt \Big] \\ &+ \mathbb{E}\Big[\gamma(T^n) \Big(\int_{T^n}^{T^{n+1}} dB(t) \Big) \big(\delta g(T^n, X_k^n) - \delta g(T^n, X(T^n)) \big) \Big]. \end{aligned}$$

The first term is bounded by

$$C(\Delta T)^2 \sqrt{\eta_{k+1}^n \eta_k^n} \leq C(\Delta T) \eta_{k+1}^n + C(\Delta T)^3 \eta_k^n,$$

thanks to (4.9), (4.10), and the Cauchy-Schwarz inequality. Using now the relation $(\mathbb{E}[ab])^2 \leq \mathbb{E}[a^2]\mathbb{E}[b^2]$, we obtain for the second term as estimate of the form

$$C(\Delta T)^{1/2}\sqrt{\eta_{k+1}^n}(C\Delta T)^{\alpha/2}\sqrt{\eta_k^n} \leq C(\Delta T)\eta_{k+1}^n + C(\Delta T)^{\alpha}\eta_k^n.$$

The above bounds and (4.8) show that

$$\eta_{k+1}^{n+1} \leq (1 + C\Delta T)\eta_{k+1}^n + C(\Delta T)^{\alpha\wedge 3}\eta_k^n. \tag{4.12}$$

Here, $a \wedge b = \min\{a, b\}$. The same results as (3.12) show that

$$\eta_k^n \leq C(\Delta T)^{k(\alpha\wedge 3)}\binom{n}{k}. \tag{4.13}$$

For $n = N$ and $k \ll N$, this implies that

$$\eta_k^N \leq C(\Delta T)^{k(\alpha\wedge 3-1)}\mathbb{E}[(X_0)^2]. \tag{4.14}$$

We deduce the following estimate at the final time T for the root-mean-square (RMS) error

$$\left(\mathbb{E}[(X_k^N - X(T))^2]\right)^{1/2} \leq C(\Delta T)^{\frac{k(\alpha\wedge 3-1)}{2}}\left(\mathbb{E}[(X_0)^2]\right)^{1/2}. \tag{4.15}$$

When $\alpha \geq 3$, for instance when $\sigma(t)$ is independent of x, we obtain a method of order k after k iterations of the parallel scheme. When $\alpha = 2$, we obtain a method of order $k/2$. The Euler scheme without parallel acceleration is respectively of order 1 when $\alpha \geq 3$ and $1/2$ when $\alpha = 2$. We thus recover similar results to those of section 3.

Note that the Euler discretization is one of the few schemes that does not require to estimate integrals of the form $\int_0^t B_i(s)dB_j(s)$. When $m = 1$ (and $i = j = 1$), such integrals can be calculated explicitly. In higher dimensions however, these integrals may be difficult to estimate (see [17]). An advantage of the parallelization scheme is that it is a high-order scheme without estimating these integrals, provided that we have a sufficiently large number of processors to solve the equations on the fine grid.

The main result of this section is that the good behavior of the parareal algorithm observed in (3.13) extends to the discretization of stochastic ordinary differential equations insofar as (4.15) holds. The latter result means "strong" convergence (in the $L^2(\Omega, \mathcal{F}, P)$ sense) of the discrete process to the continuous one, in the sense that the continuous paths of $X(t)$ are indeed well approximated by X_k^n. If one is interested is stochastic paths, as in the filtering application considered in section 6, then indeed the parareal algorithm may be efficiently used to accelerate convergence. Note however that the parareal algorithm is probably not useful when one is interested in the law of the process $X(t)$. In that case, "weak" convergence estimates of the form $|\mathbb{E}[f(X_k^N)] - \mathbb{E}[f(X(T))]|$ for compactly supported continuous functions $f(\cdot)$ are likely to be much smaller than the bounds in (4.15) (see [12] for instance). Yet numerical simulations of $\mathbb{E}[f(X(T))]$ are usually based on Monte Carlo methods, which are trivially parallelizable over realizations of the stochastic process, and which will thus be much more efficient than parallelizations based on the parareal algorithm

5. Speedup, system efficiency, multi-level parallelization. This section shows in which situations the parallel algorithm introduced in section 3 may be useful and which speedup and system efficiency are to be expected. The speedup is the ratio between the full fine resolution and the parallel algorithm. The system efficiency is the ratio of this speedup with the number of required processors. Ideally, we want the speedup to be as large as possible and the system efficiency as close to 1 as possible.

5.1. Speedup and system efficiency. We assume that the coarse discretization is of order 1 in ΔT (so that $m = 1$). The theory of the preceding sections can be generalized to a fine discretization that is not exact, but rather of order 1 in $\delta T \sim (\Delta T)^k$ for a fixed value of k on the interval $(0, T)$. In the context of section 3, this amounts to replacing the operator g by g_δ (with obvious notation) and observing that the new difference operator $\delta g = g_\delta - g_\Delta$ still satisfies the required regularity hypothesis (3.7), as an application of the triangle inequality.

To calculate the speedup, we assume that the cost of one coarse step and the cost of one fine time step are identical and equal to 1. We denote by τ the final time and want to solve the system of (possibly stochastic) ordinary differential equations on $(0, \tau)$. We want to consider times $\tau = O(1)$ and $\tau \gg 1$. For the Euler discretization of time step δT, we have that the error $\varepsilon^n = |X^n - X(T^n)|$ satisfies

$$\varepsilon^{n+1} \leq (1 + C_0 \delta T)\varepsilon^n + (\delta T)^2 |X^n|.$$

Since $|X^n| < (1 + C_0 \delta T)^n$, we deduce that

$$\varepsilon^n \leq (1 + C_0 \delta T)^n n (\delta T)^2 |X_0|. \tag{5.1}$$

When $n \leq C(\delta T)^{-1}$, we obtain an accuracy of order δT. When $n \gg (\delta T)^{-1}$ however, the term $(1 + C_0 \delta T)^n$ blows up exponentially. To avoid this effect, which is not physical in many instances, we assume that C_0 is arbitrary when $\tau = O(1)$, and that $C_0 = 0$ when $\tau \gg 1$. This allows us to consider the large time solution of problems with no growing modes, for instance for problems such that

$$|g(t, x) - g(t, y)| \leq |x - y| \quad \text{and} \quad |g_\Delta(t, x) - g_\Delta(t, y)| \leq |x - y|.$$

We will consider the case of a linear isometry in the next section. With these restrictions, we introduce the following notation.

- τ is the final time
- M is the number of successive use of the parallel algorithm between 0 and τ (this means that the parallel algorithm is used M times on intervals of size $T = \tau/M$)
- P is the number of processors
- S is the speedup
- E is the system efficiency of the calculation, given by the ratio of the speedup over the number of processors $E = S/P$
- T is the length of the intervals of time on which the parallel algorithm is used and is such that $\tau = MT$.

The accuracy of the fine discretization at the final time (5.1) is given by

$$(\delta T)^2 \frac{\tau}{\delta T} = \tau \delta T.$$

This allows us to get convergence of the numerical simulation for times τ as large as $(\delta T)^{-\alpha}$ for all $0 \leq \alpha < 1$.

The accuracy of the parallel scheme is deduced on each interval of size T from (3.12) and is given by

$$|\varepsilon_k^n| \leq C(T \Delta T)^k.$$

Since the errors made at different time steps do not grow by more than a constant factor thanks to our hypothesis on C_0, the final error after M steps of the parallel

algorithm is given by

$$M(T\Delta T)^k.$$

Both accuracies must be equivalent so that the fine discretization and the parallel algorithm can be compared. This implies that

$$M(T\Delta T)^k \sim \tau\delta T = MT\delta T \quad \text{i.e.,} \quad (T\Delta T)^k \sim T\delta T. \tag{5.2}$$

Here, $a \sim b$ means that $a/b = O(1)$ as δT and ΔT tend to 0.

Let us now calculate the respective time costs of the methods. The fine solution cost is simply

$$\frac{\tau}{\delta T}.$$

The time cost of the parallel algorithm is

$$M\Big[\frac{T}{\Delta T} + (k-1)\Big(\frac{T}{\Delta T} + \frac{\Delta T}{\delta T}\Big)\Big].$$

Notice that the CPU cost over all processors is much higher. This implies that the speedup is

$$S = \frac{\dfrac{T}{\delta T}}{k\dfrac{T}{\Delta T} + (k-1)\dfrac{\Delta T}{\delta T}} = \frac{1}{k\dfrac{\delta T}{\Delta T} + (k-1)\dfrac{\Delta T}{T}}. \tag{5.3}$$

The number of processors is $P = T(\Delta T)^{-1}$ so that the system efficiency is

$$E = \frac{S}{P} = \frac{1}{(k-1) + k\dfrac{T\delta T}{(\Delta T)^2}}. \tag{5.4}$$

We see that the system efficiency cannot be better than $(k-1)^{-1}$, which is the number of iterations of the fine discretization required in the parallel algorithm.

5.2. Maximization of speedup or system efficiency. To fix ideas let us assume that

$$\tau = (\delta T)^{-\beta} \quad \text{for } 0 \le \beta < 1,$$

so that the final accuracy of the calculation at τ is $(\delta T)^{1-\beta}$.

Let us assume moreover that

$$T \sim (\delta T)^{-\gamma} \quad \text{for } \gamma \le \beta,$$

where the constant γ is a free parameter. We deduce from the two contributions on the denominator of (5.3) that the speedup is optimal provided that

$$\frac{\delta T}{\Delta T} = \mu\frac{\Delta T}{T}, \tag{5.5}$$

where μ is a constant of order $O(1)$. Using (5.2), we finally deduce that optimally

$$k = \frac{2(1-\gamma)}{1-3\gamma}, \qquad \text{and} \qquad S \sim (\delta T)^{-(1+\gamma)/2}.$$

We should thus choose γ as large as possible but less than $1/3$, which corresponds to $k = +\infty$. Optimizing the speedup is thus realized as follows. Let us assume that $\beta = (k_0 - 2)/(3k_0 - 2)$ for some $2 \leq k_0 \in \mathbb{N}$ when $\beta < 1/3$ to simplify (and β is arbitrary when $1/3 \leq \beta < 1$). Then we have the optimal values

$$\begin{aligned} &\beta < \frac{1}{3}, \quad S \sim (\delta T)^{-(1+\beta)/2}, \quad k = \frac{2(1-\beta)}{1-3\beta} = k_0, \quad M = 1, \\ &\beta \geq \frac{1}{3}, \quad S \sim (\delta T)^{-2/3}, \qquad k = +\infty, \qquad\qquad M \sim (\delta T)^{1/3-\beta}. \end{aligned} \tag{5.6}$$

When $\beta < 1/3$ there is an optimal value of $k \geq 2$ such that the maximal speedup is attained without restart ($M = 1$). When $\beta > 1/3$ however, the maximal speedup is bounded by $(\delta T)^{-2/3}$ independent of the number of processors P and the algorithm is optimal at $k = \infty$ and should be restarted $(\delta T)^{1/3-\beta}$ times. The optimal number of processors P_o required is of the same order as S here. When $\beta = 0$, i.e. when the final time $\tau = O(1)$, we obtain that

$$S \sim (\delta T)^{-1/2}, \quad k = 2, \quad M = 1, \quad \Delta T \sim (\delta T)^{1/2}. \tag{5.7}$$

The corresponding efficiency of the algorithm is then given by

$$E = \frac{1}{k - 1 + \mu k} < 1.$$

It is not optimal since $k \geq 2$ and $\mu = O(1)$. Instead of maximizing the speedup we can try to maximize the speedup knowing the number of processors $P \ll P_o$. To fix ideas, let us assume that

$$P = \frac{T}{\Delta T} = \frac{1}{(\delta T)^\alpha} \tag{5.8}$$

for some $0 < \alpha < 1/2$. To obtain an efficiency close to 1 we choose $k = 2$ (for $\alpha > 1/2$ we need $k > 2$, which we do not considered here). We deduce from (5.2) that

$$T = (\delta T)^{\frac{1-2\alpha}{3}}, \qquad \Delta T = (\delta T)^{\frac{1+\alpha}{3}}. \tag{5.9}$$

The speedup and system efficiency are then found to be

$$S = \frac{1}{(\delta T)^\alpha} E, \qquad E = \frac{1}{1 + 2(\delta T)^{\frac{2(1-2\alpha)}{3}}}. \tag{5.10}$$

Here, the speedup is therefore roughly equal to the number of processors, which is equivalent to saying that the system efficiency is close to 1. Notice that the speedup is however smaller than in (5.6). The number of successive occurrences of the parallel algorithm is given by $M \sim (\delta T)^{\frac{2\alpha-1}{3}} \tau \gg \tau$. Maximizing the system efficiency requires to restart the parallelization algorithm many times.

5.3. Multi-level parallelization. The maximal speedup obtained in the previous section is proportional to $(\delta T)^{-1/2}$ provided that we have enough processors when $\beta = 0$, which we now assume for simplicity. When the number of processors is larger, the two-level method cannot be used to obtain an optimized speedup. Instead, we need to consider a multi-level algorithm.

The previous section shows that $k = 2$ is optimal for both the maximization of speedup and system efficiency. This actually also holds in the multi-level parallelization so we anticipate that result to simplify. The mechanism to describe three-level parallelization is to apply the two-level procedure to the solution of the fine discretization on every interval of size ΔT. Let us assume that

$$dT \ll \delta T \ll \Delta T \ll T \leq \tau.$$

We now compute the order of the maximal speedup that can be obtained.

The accuracy of the fine solution is again τdT. The accuracy of the parallel solution is also still given by $M(T\Delta T)^2 = \tau T(\Delta T)^2$, where T and ΔT are still unknown. Both accuracies are comparable provided that

$$dT \sim T(\Delta T)^2.$$

The cost of the fine solution is τ/dT, whereas that of the three-level parallel algorithm is

$$M\Big[2\frac{T}{\Delta T} + 2\frac{\Delta T}{\delta T} + \frac{\delta T}{dT}\Big].$$

This implies the following expression for the speedup

$$S = \frac{\dfrac{T}{dT}}{2\dfrac{T}{\Delta T} + 2\dfrac{\Delta T}{\delta T} + \dfrac{\delta T}{dT}}.$$

Upon maximizing the above expression, we find that

$$\Delta T = \sqrt{\delta T T}, \qquad \delta T = \sqrt{4\Delta T dT}.$$

Upon taking $dT = T(\Delta T)^2$, we obtain

$$T = \Big(\frac{dT}{16}\Big)^{1/7}, \quad \Delta T = 4\Big(\frac{dT}{16}\Big)^{3/7}, \quad \delta T = 16^{2/7}(dT)^{5/7}.$$

This implies that the number of processors and the speedup are both of order

$$S \sim (dT)^{-4/7} \gg (dT)^{-1/2}, \qquad P \sim (dT)^{-4/7}.$$

The corresponding system efficiency equals a constant smaller than 1 that we do not reproduce. Provided that the number of processors is less than the above estimate, the system efficiency can be made arbitrarily close to 1 as for the two-level parallel algorithm.

More generally, let us define the multi-level parallelization algorithm as follows. We assume that we have a scale of time steps such that

$$\Delta_m T \ll \Delta_{m-1} T \ll \cdots \ll \Delta_1 T \ll \Delta_0 T \ll \tau.$$

At each scale, the ordinary differential equation is solved by using a two-level parallel algorithm involving the next finer scale.

The accuracy of the fine solution is given by $\tau\Delta_m T$. The accuracy of the parallel algorithm is given by $M(\Delta_0 T \Delta_1 T)^2$. Here again, we have $\tau = M\Delta_0 T$. This implies that

$$\Delta_0 T(\Delta_1 T)^2 \sim \Delta_m T. \tag{5.11}$$

The speedup of the multi-level algorithm is given by

$$S = \frac{\frac{\tau}{\Delta_m T}}{M\Big(2(\frac{\Delta_0 T}{\Delta_1 T} + \cdots + \frac{\Delta_{m-2} T}{\Delta_{m-1} T}\Big) + \frac{\Delta_{m-1} T}{\Delta_m T}\Big)}.$$

At $\Delta_m T$ and $\Delta_{m-2} T$ fixed, we again have that $2\frac{\Delta_{m-2} T}{\Delta_{m-1} T} + \frac{\Delta_{m-1} T}{\Delta_m T}$ is minimized provided that $(\Delta_{m-1} T)^2 \sim \Delta_{m-2} T \Delta_m T$. More generally, S is maximized provided that asymptotically

$$\frac{\Delta_n T}{\Delta_{n+1} T} \sim \frac{\Delta_{n+1} T}{\Delta_{n+2} T}$$

for $0 \leq n \leq m-2$. Upon taking the product of these relations, we find that

$$\frac{\Delta_0 T}{\Delta_n T} \sim \Big(\frac{\Delta_0 T}{\Delta_1 T}\Big)^n, \quad 0 \leq n \leq m. \tag{5.12}$$

We deduce then from the above relation and (5.11) that

$$\Delta_0 T = (\Delta_m T)^{\frac{m-2}{3m-2}}, \qquad \Delta_1 T = (\Delta_m T)^{\frac{m}{3m-2}}.$$

Using (5.12) one more time, we deduce from the relation $nm - (n-1)(m-2) = m + 2(n-1)$ that

$$\Delta_n T = (\Delta_m T)^{\frac{m+2(n-1)}{3m-2}}, \qquad 0 \leq n \leq m.$$

We obtain that the optimal speedup and the number of required processors are then of order

$$S \sim \frac{\Delta_1 T}{\Delta_m T} \sim (\Delta_m T)^{-\frac{2m-2}{3m-2}}, \qquad P \sim (\Delta_m T)^{-\frac{2m-2}{3m-2}}. \tag{5.13}$$

We recover $(\Delta_2 T)^{-1/2} = (\delta T)^{-1/2}$ when $m = 2$ and $(\Delta_3 T)^{-4/7} = (dT)^{-4/7}$ when $m = 3$. Notice that the speedup tends to $(dt)^{-2/3}$ when $m \to \infty$, where dt is the finest available time discretization. The number of occurrences of the multi-level algorithm is $M \sim (dt)^{-1/3}$. So on each interval of size $\Delta_0 T \sim (dt)^{1/3}$, the number of points of the finest discretization and the number of processors are asymptotically equivalent, which is optimal. The multi-level algorithm needs however to be restarted $(dt)^{-1/3}$ times to reach the required accuracy.

6. Examples and numerical simulations. This section presents numerical simulations that illustrate the type of solutions obtained by the parareal algorithm and confirm the theoretical predictions obtained in previous sections. These numerical simulation are performed for extremely simple equations and on a single processor machine using low order schemes to allow for comparison with theory. Since communication between the processors occurs only at the beginning and the end of the coarse steps we do not expect any surprise in the implementation on parallel machines. This however remains to be studied more carefully. We also refer to the existing literature on the parareal algorithm for more challenging applications than the simple equations considered here.

6.1. Exponential function. The simplest example consists of solving the equation

$$dX(t) = X(t)dt, \quad t \in (0,1), \qquad X(0) = 1. \tag{6.1}$$

We obtain that $X(1) = e$ and consider the explicit Euler scheme to solve it:

$$X^{n+1} = (1 + \delta T)X^n = (1 + \delta T)^{n+1}.$$

The advantage of such a simple equation is that we obtain an explicit expression for the solution of the parallel scheme. Assuming that a fine time step dT and a coarse time step ΔT are used, and that the parallel algorithm is restarted M times on intervals of size $1/M$, we obtain after some algebra the expression

$$X_2 = (1 + \Delta T)^{1/\Delta T}\Big(1 + \frac{(1 + dT)^{\Delta T/dT} - (1 + \Delta T)}{M\Delta T(1 + \Delta T)}\Big)^M. \tag{6.2}$$

In order to calculate the speedup of the method, we require that the final errors at time $\tau = 1$ be the same (say 10^{-8}) for the classical explicit scheme and the parallel scheme:

$$|X(1) - X^{1/\delta T}| = |X(1) - X_2| = 10^{-8}. \tag{6.3}$$

Some algebra shows that δT is roughly $(2/e)10^{-8}$. We now have the choice of ΔT and dT in (6.2) provided that the constraint (6.3) remains satisfied. After optimization, we obtain for $M = 1$ that

$$dT = 7.21\,10^{-9}, \quad \Delta T = 9.67\,10^{-5}, \quad P = 10341, \quad S = 3987, \quad E = 0.40,$$

and for $M = 20$ that

$$dT = 7.21\,10^{-9}, \quad \Delta T = 4.35\,10^{-4}, \quad P = 114.9, \quad S = 112.3, \quad E = 0.98.$$

In order to obtain an accuracy of 10^{-8}, the maximal speedup we can expect is of the order of 4000 provided that we have 10^4 processors. If we have access to 115 processors, we can increase the efficiency of the parallelization to 0.98 provided that we use the parallel algorithm on intervals of size $T = \tau/M = 1/20$.

These results are consistent with the theoretical analysis of the preceding section. Similar results would also hold for the examples of application of the parallel algorithm considered below. Since the optimization of the speedup and efficiency presented in section 5 requires the control of constants appearing in front of the order of convergence of the numerical methods, which are not easily estimated, even numerically, we shall focus in the subsequent sections on the numerical analysis of the theory presented in sections 3 and 4, and show that the convergence in $(\Delta T)^k$ is very well observed numerically, even for small values of ΔT.

6.2. Harmonic oscillator. We consider in this section the harmonic oscillator, which solves the following 2×2 system

$$\begin{aligned} dX(t) &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} X(t)dt, \qquad t \in [0,T] \\ X(0) &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \end{aligned} \tag{6.4}$$

The solution $g(t,x)$ corresponding to the system (6.4) is given by

$$g(t,x) = \begin{pmatrix} x_1 \cos \Delta T - x_2 \sin \Delta T \\ x_1 \sin \Delta T + x_2 \cos \Delta T \end{pmatrix}. \tag{6.5}$$

We verify that it is an isometry in the sense that

$$y_1^2 + y_2^2 = x_1^2 + x_2^2$$

for $y = g(t,x)$. This implies that the constant $C_0 = 0$ in (5.1) and that no exponentially growing mores are present in the continuous solution.

We consider the Euler implicit scheme to solve this problem, given by

$$g_\Delta(t,x) = \begin{pmatrix} 1 & \Delta T \\ -\Delta T & 1 \end{pmatrix}^{-1} x. \tag{6.6}$$

For $z = g_\Delta(t,x)$, we easily verify that

$$z_1^2 + z_2^2 = (1 + (\Delta T)^2)^{-1}(x_1^2 + x_2^2).$$

The implicit Euler scheme is therefore quite dissipative and should not be used in practice for long time calculations. We shall see that the parallel algorithm still allows us to use this low-order scheme satisfactorily even for long time calculations.

The parallel algorithm presented in section 3 is considered on an interval of size $T = 2\pi$. Since we have access to the exact solution, the implementation of the algorithm is exactly as given in section 3. We consider the numerical solution for values of $1 \le k \le K = 5$ for discretizations $N = 25$, $N = 50$, $N = 100$ and $N = 200$.

The results of the numerical simulation for $|\varepsilon_k^N|$, the error at final time T between the exact solution and the approximation of order k X_k^N, are given in Tab. 6.1. A plot of the different solutions X_k^n for $\Delta T = T/25$ is given in Fig. 6.1. In the theory

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
$N=25$	$5.53\,10^{-1}$	$1.83\,10^{-1}$	$4.02\,10^{-2}$	$6.29\,10^{-3}$	$7.30\,10^{-4}$
radius		0.33	0.22	0.16	0.12
$N=50$	$3.30\,10^{-1}$	$6.01\,10^{-2}$	$7.35\,10^{-3}$	$6.64\,10^{-4}$	$4.69\,10^{-5}$
radius		0.18	0.12	0.090	0.071
$N=100$	$1.80\,10^{-1}$	$1.72\,10^{-2}$	$1.09\,10^{-3}$	$5.20\,10^{-5}$	$1.69\,10^{-6}$
radius		0.095	0.064	0.048	0.038
$N=200$	$9.44\,10^{-2}$	$4.58\,10^{-3}$	$1.49\,10^{-4}$	$3.60\,10^{-6}$	$6.96\,10^{-8}$
radius		0.049	0.032	0.024	0.019

TABLE 6.1

Errors at final time T of the error $|X(T^N) - X_k^N|$ for several values of k and $\Delta T = T/N$ for the harmonic oscillator of section 6.2.

of section 3, the ratio between the error at level k and the error at level $k+1$ is proportional to ΔT. The radius in Tab. 6.1 should decrease by a factor 2 when the number of coarse time steps doubles. This is very well confirmed numerically.

6.3. Harmonic oscillator over long times. Let us now consider the same harmonic oscillator as in the preceding section, but over a much longer time. The solutions X_1^n, X_2^n, X_6^n, and $X(T^n)$ are represented over the interval $[0, 12\pi]$ for a

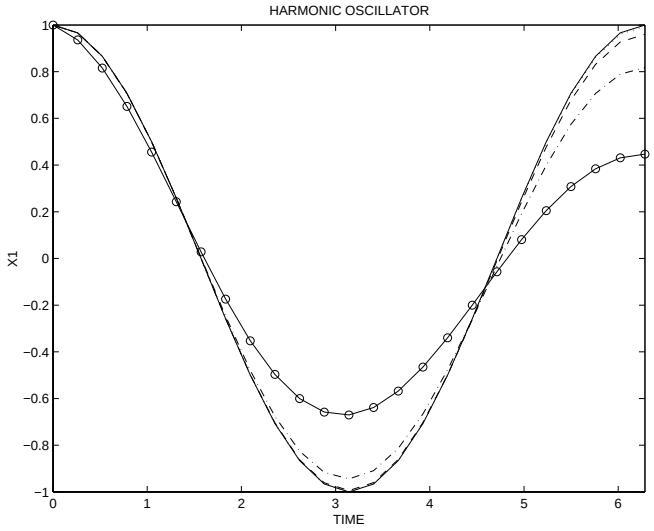

FIG. 6.1. *Exact solution $X(T^n)$ and approximate solutions X_k^n for $1 \leq n \leq N = 25$ and $1 \leq k \leq 4$. X_1^n: solid line with circles. X_2^n: dot-dashed line. X_3^n: dashed line. X_4^n: dotted line. Exact solution $X(T)$: solid line.*

number of discretization $N = 150 = 6 \times 25$ in Fig.6.2. Since the implicit Euler scheme is dissipative, the solution X_1^n is quickly very far off the exact solution. Notice that the algorithm still converges to the exact solution as $k \to \infty$. However, to obtain a reasonable accuracy at $T = 12\pi$, at least 6 iterations of the parallel scheme are necessary, which would require a time discretization of time step $\delta T \sim (\Delta T)^6$, i.e., approximately $6 \times 25^6 \approx 10^9$ discretization points.

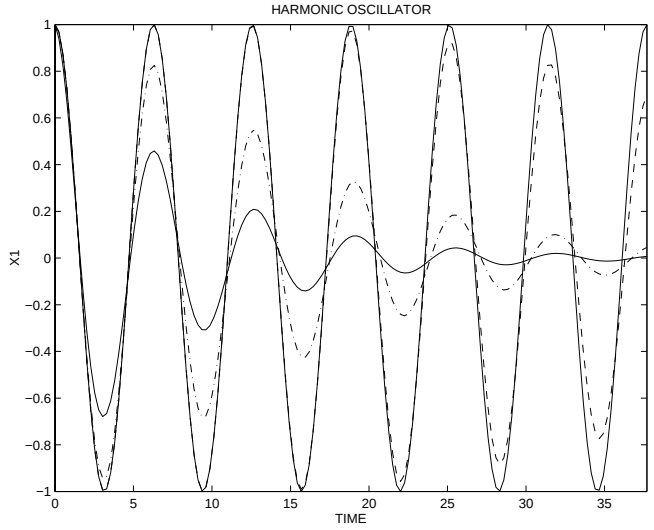

FIG. 6.2. *Exact solution $X(T^n)$ and approximate solutions X_k^n for $1 \leq n \leq 6N = 150$ and $k = 1, 2, 6$. X_1^n: solid line. X_2^n: dash-dotted line. X_6^n: dashed line. Exact solution $X(T)$: solid line.*

M	5	10	20	40
error	0.796	0.400	0.198	0.0983
radius		0.503	0.495	0.497

TABLE 6.2
L^2 error of the discrete solutions on the last period of the computation interval.

Instead, we can use the parallel algorithm on smaller intervals and restart the process until final time. Consider for instance the solution of (6.4) on a domain of size $\tau = 2\pi M$, where $M = N/5$ and N is the number of discretization points of the coarse discretization on $(0, 2\pi)$. We use the parallel algorithm with $k = 2$, which is

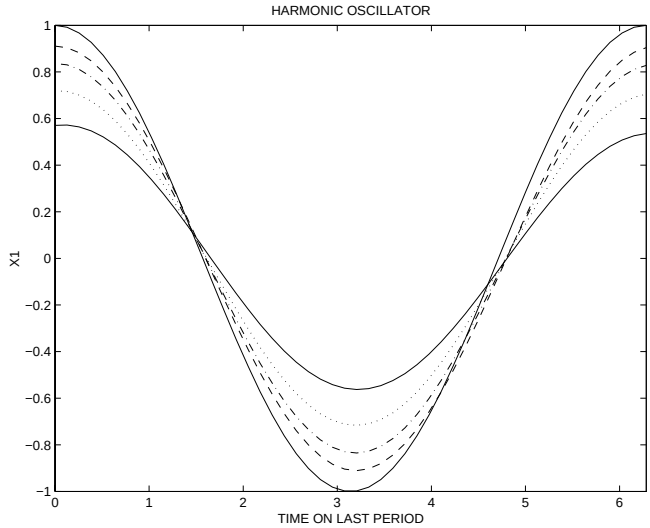

FIG. 6.3. *Exact solution $X(T^n)$ (solid line) and approximate solution X_2^n on the last period of computation $[2\pi(M-1), 2\pi M]$ for different values of M: $M = 5$ (solid line), $M = 10$ (dotted line), $M = 20$ (dash-dotted line), $M = 40$ (dashed line). Notice that the exact solution is periodic and takes therefore the same value on the last period of the computation interval independently of M.*

optimal to maximize system efficiency according to the theory of the preceding section. Since the method with $k = 2$ is of order 2 and the whole domain $(0, \tau)$ is of size N, the accuracy at final time τ should be of order $1/N$ also. This is confirmed by the results presented in Fig. 6.3 and Tab. 6.2. In Fig. 6.3, the exact and approximate solutions are represented on the last loop $(2\pi(M-1), 2\pi M)$ for values of $M = 5$, $M = 10$, $M = 20$, and $M = 40$. In Tab. 6.2 are shown the L^2 error between the exact and approximate solutions on $[2\pi(M-1), 2\pi M]$ for the same values of M as above. The accuracy of the error between the exact solution and the solution of the parallel algorithm on $[2\pi(M-1), 2\pi M]$ is theoretically of order N^{-1}. This is extremely well reproduced numerically as can be seen on the last row of the table.

If we assume that $\delta T = 2\pi/N^2$ and $\Delta T = 2\pi/N$ according to (5.5) with $\mu = 1$ and $T = 2\pi$, the speedup is given by $S = N/2$ provided that we have $P = N$ processors. For $N = 200$, a speedup of order 100 is quite valuable and may allow us to compute the dynamics of oscillatory systems quite accurately even over a large number of periods.

	$N = 25$	$N = 50$	$N = 100$	$N = 200$
$k = 1$	$2.5\,10^{-1}$	$4.3\,10^{-2}$	$6.1\,10^{-3}$	$7.9\,10^{-4}$
radius		0.63	0.72	0.66
$k = 2$	$1.6\,10^{-1}$	$1.8\,10^{-2}$	$1.8\,10^{-3}$	$1.5\,10^{-4}$
radius		0.42	0.50	0.50
$k = 3$	$1.1\,10^{-1}$	$8.9\,10^{-3}$	$6.1\,10^{-4}$	$3.8\,10^{-5}$
radius		0.29	0.35	0.34
$k = 4$	$7.5\,10^{-2}$	$4.5\,10^{-3}$	$2.1\,10^{-5}$	$9.6\,10^{-6}$
radius		0.19	0.25	0.25

TABLE 6.3

RMS error between the exact solution $X(3)$ given by (6.8) and the discrete solutions X_k^N at final time for $k = 1, 2, 3, 4$.

6.4. Geometric Brownian Motion. The last two numerical simulations are devoted to stochastic equations and illustrations of the theoretical result obtained in (4.15). We do not try to maximize speedup and system efficiency here and thus choose the non-optimal $M = 1$. We consider here Geometric Brownian motion, defined as the solution to

$$\begin{aligned} &dX(t) = rX(t)dt + \sigma X(t)dB(t), \qquad t \in [0, T] \\ &X(0) = 1. \end{aligned} \tag{6.7}$$

There is an explicit solution to this equation given by

$$X(t) = \exp\Big(\sigma B(t) + (r - \frac{\sigma^2}{2})t\Big). \tag{6.8}$$

We assume that the final time $T = 3$. We use the Euler scheme given by (4.4) to solve (6.7) and the parallelization scheme defined by (3.4) to obtain more accurate solutions X_k^n for $k = 1, 2, 3, 4$. Notice that we calculate δg exactly since we have access to the exact solution (6.8).

We consider ensemble averages (over $N_r = 1000$ realizations so the results presented have an accuracy of order $\sqrt{N_r} \approx 3\,10^{-2}$) for a number of discretization points $N = 25$, $N = 50$, $N = 100$, and $N = 200$. Since the Euler scheme has an accuracy of order $1/2$ (since $a_4 \neq 0$; see section 4), we expect

$$\varepsilon_k^N = \sqrt{\mathbb{E}[(X(T) - X_k^N)^2]}$$

at final time $T = 3$ to be of order $N^{-k/2}$. The results are reported in Tab. 6.3. They agree very well with theory. The exact ratios of convergence are $1/\sqrt{2} \approx 0.71$ for $k = 1$; $1/2$ for $k = 2$; $1/(2\sqrt{2}) \approx 0.35$ for $k = 3$; and $1/4$ for $k = 4$. We display on Fig. 6.4 an example of convergence of X_k^N to $X(T)$ as k increases although X_1^N is very far from approximating the true solution.

6.5. Filtering problem. The above example shows how the parallel scheme can be used to solve stochastic ordinary differential equations. Let us repeat however that when only statistical averages of the solution are required, such as $\mathbb{E}[f(X(T))]$, it might be better to use the available number of processors to run independent realizations of the random process using the fine time step δT. Indeed, the corresponding speedup is exactly given by the number of processors ($S = P$ and $E = 1$), which beats the expected speedup of the parallel algorithm.

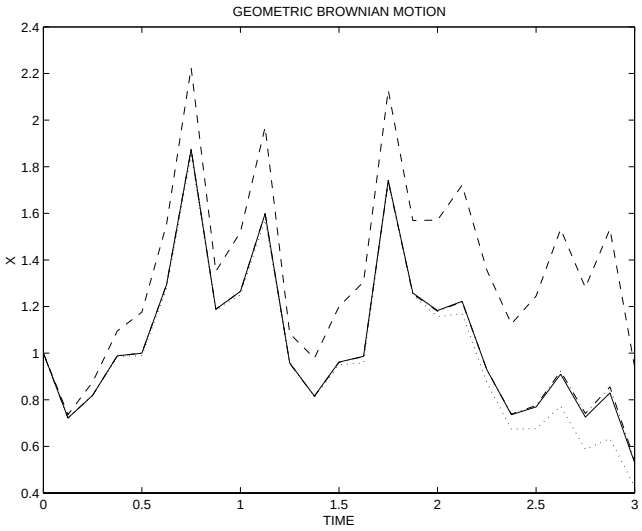

FIG. 6.4. *One realization of geometric Brownian motion on* $(0,3)$ *with* $N = 25$. *Exact solution* $X(T^n)$ *(solid line), approximate solutions* X_1^n *(dashed line),* X_2^n *(dotted line),* X_3^n *(dash-dotted line). The fourth approximation* X_4^n *is indistinguishable from* $X(T^n)$.

One interesting application where the solution of a stochastic equation for *one realization* of the random process matters is the filtering problem. We refer to [19] for a presentation of the filtering problem and consider here the simple example of the noisy observation of a population growth.

We consider the growth model

$$\begin{aligned} &dX(t) = rX(t)dt, \quad t \in [0,T] \\ &X(0) = X_0(\omega), \quad \mathbb{E}[X_0] = b > 0, \quad \mathbb{E}[(X_0 - b)^2] = a^2, \end{aligned} \tag{6.9}$$

where r is a constant growth rate, and a and b are constants. The unknown quantity is therefore the size of the population at time $T = 0$. The filtering problem is as follows. Let us assume that we have access to noisy observations of the population modeled by

$$dZ(t) = X(t)dt + mdB(t), \qquad Z(0) = 0. \tag{6.10}$$

In the absence of noise in the measurements, the population is given by $X(t) = Z'(t)$. In the presence of noise, we want to find the best estimator $\hat{X}(t)$ of $X(t)$ based on the measured realization of the random process $Z(t)$. In mathematical terms, this means that we are looking for the random process

$$\hat{X}(t) = \mathbb{E}[X(t)|\mathcal{G}_t],$$

where $\mathcal{G}_t$ is the σ-algebra generated by $\{Z(s), X(s); s \leq t\}$, and $\mathbb{E}[\cdot]$ is expectation with respect to the probability measure P associated with $B(\cdot)$ and X_0.

This problem can be answered by solving a stochastic ordinary differential equation and an ordinary differential equation of Riccati type [7, 19]. In our context, these

	$N = 25$	$N = 50$
$k = 1$	$4.2\,10^{-2}$	$2.1\,10^{-2}$
radius		0.49
$k = 2$	$1.6\,10^{-3}$	$3.9\,10^{-4}$
radius		0.24

TABLE 6.4

RMS error between the exact solution $X(T)$ for the filtering problem and the discrete solutions X_k^N at final time for $k = 1, 2$, $T = 2$, and $\sigma = 0.6$.

equations are given by

$$\begin{aligned} d\hat{X}(t) &= \Big(r - \frac{S(t)}{m^2}\Big)\hat{X}(t)dt + \frac{S(t)}{m^2}dZ(t), & \hat{X}(0) &= b, \\ dS(t) &= \Big(2rS(t) - \frac{S^2(t)}{m^2}\Big)dt, & S(0) &= a^2. \end{aligned} \tag{6.11}$$

Since the diffusion constant in front of dB in (6.11) is independent of x, we expect the Euler scheme (4.4) to be of order 1 (see section 4). Two iterations of the parallel scheme make then $(\hat{X}_2^N, S_2^N)$ an approximation of order 2 of the exact solution $(\hat{X}(T), S(T))$ (in the sense of (4.15)).

We consider the following example. We assume that $r = 2$, that X_0 is uniformly distributed over $(0, 2)$ so that $b = 1$ and $a^2 = 4/3$. The results of the numerical simulations are given in Tab. 6.4 for $\sigma = 0.6$ and $T = 2$. The graph of the renormalized solution $e^{-rt}\hat{X}(t)$ for one realization of $B(t)$ is given in Fig. 6.5 for $\sigma = .6$, $T = 2$ (left) and $\sigma = 6$, $T = 5$ (right).

To calculate the RMS errors, 500 realizations have been used. Since the exact solution operator $g(T, X)$ is not known in general (although the solution of the Riccati equation is known explicitly), it has been approximated by using a discretization with $O(N^3)$ points. The "exact" solution has been calculated by using a discretization with $O(N^3)$ points.

The error estimates given in Tab. 6.4 are according to theory: the theoretical radius of convergence is $1/2$ for $k = 1$ when the number of discretization points is halved. The corresponding radius is $(1/2)^2$ for $k = 2$. When σ is small, there is little noise in the measurements and the estimate $\hat{X}(t)$ is quickly close to $X(t)$. When σ increases, so does noise in the measurements and the accuracy of $\hat{X}(t)$ decreases.

Acknowledgment. The author would like to thank Yvon Maday and Gabriel Turinici for numerous fruitful discussions on the parallelization of time discretizations. This work was supported in part by NSF grant DMS-0239097 and an Alfred P. Sloan fellowship.

REFERENCES

[1] P. AMODIO AND L. BRUGNANO, *Parallel implementation of block boundary value methods for odes*, J. Comput. Appl. Math., 78 (1997), pp. 197–211.

[2] P. AMODIO AND F. MAZZIA, *Parallel block preconditioning for the solution of boundary value methods*, J. Comput. Appl. Math., 69 (1996), pp. 191–206.

[3] V. I. ARNOL'D, *Ordinary Differential Equations*, Springer-Verlag, Berlin, 1992.

[4] L. BAFFICO, S. BERNARD, T. MADAY, G. TURINICI, AND G. ZÉRAH, *Parallel-in-time molecular-dynamics simulations*, Phys. Rev. E, 66 (2002), p. 057701.

[5] G. BAL, *On the Convergence and the Stability of the Parareal Algorithm to solve Partial Differential Equations*, in Domain Decomposition Methods in Science and Engineering, R.

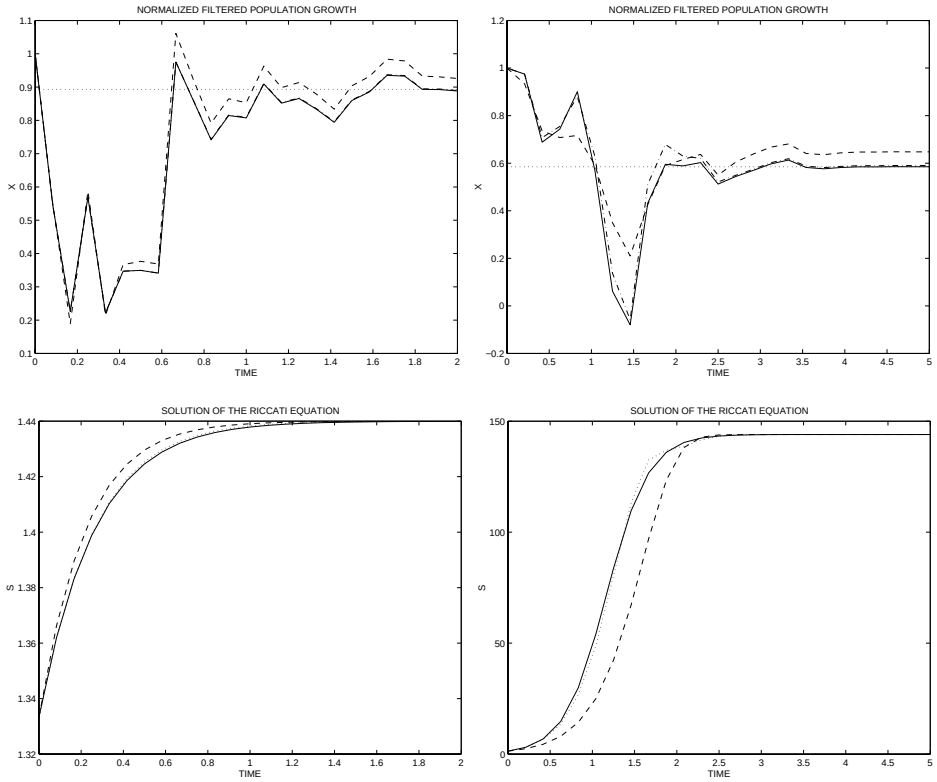

FIG. 6.5. *Typical realizations of $e^{-rt}\hat{X}(t)$ and corresponding solution $S(t)$ for two different values of σ and T. Left figures: $\sigma = 0.6$ and $T = 2$. Right figures: $\sigma = 6$ and $T = 5$. Top figures: Exact solution $e^{-rt}\hat{X}(t)$ (solid line), approximate solutions $e^{-rt}\hat{X}_1^n$ (dashed line) and $e^{-rt}\hat{X}_2^n$ (dash-dotted line). The initial condition X_0 of $X(t)$ is represented as a dotted line. Bottom figures: corresponding solution $s(T)$ and approximations S_k^n.*

Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, J. Xu, eds., Lect. Notes in Comput. Sci. Eng., Springer, Berlin, 40 (2004), pp. 425–432.

[6] 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 (Zürich, 2001), Lect. Notes Comput. Sci. Eng., Springer, Berlin, 23 (2002), pp. 189–202.

[7] A. BENSOUSSAN, *Stochastic Control of Partially Observable Systems*, Cambridge Univ. Press, 1992.

[8] L. BRUGNANO AND D. TRIGIANTE, *Solving differential problems by multistep initial and boundary value methods.*, Stability and Control: Theory, Methods and Applications, 6, Gordon and Breach Science Publishers, Amsterdam, 1998.

[9] K. BURRAGE, *Parallel and Sequential Methods for Ordinary Differential Equations*, Clarendon Press, Oxford, 1995.

[10] C. FARHAT AND M. CHANDESRIS, *Time-decomposed parallel time-integrator: theory and feasibility studies for fluid, structure, and fluid-structure applications*, Int. J. Num. Methods Eng., 58 (2003), pp. 1397-1434.

[11] M. J. GANDER AND A. M. STUART, *Space-time continuous analysis of the waveform relaxation for the heat equation*, SIAM J. Sci. Comput., 19 (1998), pp. 2014–1031.

[12] P. KLOEDEN AND E. PLATEN, *Numerical Solution of Stochastic Differential Equations*, Springer Verlag, Berlin, 1999.

[13] 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 (2000), pp. 661–668.

[14] Y. MADAY AND G. TURINICI, *A parareal in time procedure for the control of partial differential equations*, C.R. Acad. Sci. Paris Sér. I Math., 335 (2002), pp. 387–391.

[15] G. MARUJAMA, *Continuous markov processes and stochastic equations*, Rend. Mat. Circ.

Palermo, Ser. 2, 4 (1955), pp. 48–90.
[16] F. Mazzia, *Boundary value method for the initial value problems: parallel implementation*, Ann. Numer. math., 1 (1994), pp. 439–450.
[17] G. N. Milstein, *Numerical Integration of Stochastic Differential Equations*, Kluwer Academic Publisher, Dordrecht, 1995.
[18] W. L. Miranker and W. Liniger, *Parallel methods for the numerical integration of ordinary differential equations*, Math. Comp., 21 (1967), pp. 303–320.
[19] B. Øksendal, *Stochastic Differential Equations*, Springer-Verlag, Berlin, 2000.
[20] J. Sand and K. Burrage, *A Jacobi waveform relaxation for ODEs*, SIAM J. Sci. Comput., 20 (1998), pp. 534–552.
[21] P. J. van der Houwen, B. P. Sommeijer, and W. A. van der Veen, *Parallel iteration across the steps of high-order runge-kutta methods for nonstiff initial value problems*, J. Comput. Appl. Math., 60 (3) (1995), pp. 309–329.
[22] S. J. Wright, *Stable parallel elimination for boundary value ODEs*, Numer. Math., 67 (1994), pp. 521–535.