The parareal in time algorithm applied to the kinetic neutron diffusion equation

Anne-Marie Baudron, Jean-Jacques Lautard, Yvon Maday, Olga Mula

The parareal in time algorithm applied to the kinetic neutron diffusion equation

A.-M. Baudron[1,3], J.-J. Lautard[1,3], Y. Maday[2,3,4,5], and O. Mula[1,2,3]

Key words: parareal, neutron, diffusion

Introduction

In the framework of nuclear core calculations, the development of efficient tools to run neutron kinetic computations is a field of current active research. While such calculations are crucial for security assessment and the study of new reactor concepts, they present several mathematical and computational issues that still need to be overcome.

The exact model (kinetic transport equation) is indeed far too expensive to be simulated for these purposes and different simplifications (multi group diffusion approximation) have led to more tractable numerical simulations. Nevertheless, on real geometries and despite the use of domain decomposition enabling accelerations of the simulations thanks to parallel architectures [7], there is still need for improvements for applications on regular basis.

In this context, the purpose of this work is to investigate the implementation of the parareal in time algorithm [9] within an industrial solver called MINOS developed at C.E.A. (cf. [4]) following the preliminary analysis [5].

The paper is organized as follows: after the presentation of the neutron diffusion equation in Section 1, the main aspects of the parareal method will be recalled in Section 2. In particular, we will explain the distributed algorithm that has been used in our case from the point of view of the expected speed-up. The performances of the parareal in time algorithm in a numerical application are summarized in section 3 which is followed in Section 4 by a discussion about the convergence behavior observed in our example.

[1] C.E.A, CEA Saclay - DEN/DANS/DM2S/SERMA/LLPR - 91191 Gif-Sur-Yvette CEDEX - France , e-mail: {anne-marie.baudron}{jean-jacques.lautard}{olga.mulahernandez}@cea.fr .[2] UPMC Univ Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France e-mail: maday@ann.jussieu.fr .[3] LRC MANON, Laboratoire de Recherche Conventionnée, CEA/DEN/DANS/DM2S and UPMC-CNRS/LJLL. .[4] Institut Universitaire de France .[5] Brown Univ, Division of Applied Maths, Providence, RI, USA

1 Model

The evolution of the flux ψ of neutrons in a reactor core $\mathscr{R}$ is governed by a kinetic transport PDE whose theoretical properties (existence, uniqueness, positiveness of the solution) have been investigated in e.g. [6] (chapter XXI, section 2, theorem 3). Given the fact that ψ depends on 7 variables, namely the time t, the position within the reactor denoted as $\overrightarrow{r}$, the velocity of the neutrons $\overrightarrow{v} = \sqrt{2E/m}\overrightarrow{\Omega}$ where E stands for the energy of the neutron, $\overrightarrow{\Omega}$ stands for the direction of the velocity and m is the mass of the neutron, it has been proposed in e.g. [6] (chapter XXI, section 5), to simplify the model by first considering the average flux over the angular variables as the unknown: $\phi(t,\overrightarrow{r},E) = \frac{1}{4\pi}\int_{\mathbb{S}_2}\psi(t,\overrightarrow{r},\overrightarrow{\Omega'},E)d\overrightarrow{\Omega'}$. This approach leads to results that are accurate enough in most of the usual cases but the computing time still remains unacceptably long.

Another simplification consists in averaging also in the energy variable. This further approximation, known as the multi-group theory [10], is based on the division of the energy interval into G subintervals ($[E_{min},E_{max}] = [E_G,E_{G-1}]\cup\ldots\cup[E_1,E_0]$) and leads to consider the set $\Phi = \{\phi^g\}_{g\in\{1,G\}}$ as the new unknown solution. In order to take into account the presence of radioactive isotopes (also called precursors) that are important since they emit neutrons with a given delay, the model is complemented with a set of first order ODE's expressing their decays denoted as $\mathbf{C} = \{C_\ell\}_{\ell\in\{1,L\}}$. Since their half-lives have values that vary in a wide range, the resulting system is very stiff and small time steps are required for an accurate approximation in long time intervals.

The set $(\Phi,\mathbf{C})$ is the solution of the following set of multi-group diffusion equations:

$$(*)\begin{cases} \frac{1}{v^g}\frac{\partial\phi^g}{\partial t} - \nabla.\left(D^g\overrightarrow{\nabla}\phi^g\right) + \sigma_t^g\phi^g = \sum\limits_{g'=1}^{G}\mathscr{S}^{gg'}\phi^{g'} + \chi_p^g\sum\limits_{g'=1}^{G}\mathscr{F}^{g'}\phi^{g'} + \sum\limits_{\ell=1}^{L}\chi_\ell^g\lambda_\ell C_\ell \\ \text{over } [0,T]\times\mathscr{R},\forall g\in\{1,G\}, \\ \frac{\partial C_\ell}{\partial t} = -\lambda_\ell C_\ell + \sum\limits_{g'=1}^{G}\mathscr{F}_\ell^{g'}\phi^{g'} \text{ over } [0,T]\times\mathscr{R},\forall\ell\in\{1,L\}, \\ \phi^g = 0, \text{ on } [0,T]\times\partial\mathscr{R} \\ \phi^g(0,.) = \phi_0^g(.);\ C_\ell(0,.) = C_{\ell,0}(.) \text{ on } \mathscr{R} \end{cases}$$

where v^g is the neutron velocity, D^g the diffusion coefficient and σ_t^g the total cross-section in energy group g. χ_p^g is the prompt spectrum in energy group g, χ_ℓ^g the delayed spectrum of precursor ℓ in energy group g and λ_ℓ is the decay constant of precursor ℓ. $\mathscr{F}^g$ and $\mathscr{F}_\ell^g$ denote the prompt and delayed fission operators respectively. $\mathscr{S}^{gg'}$ is the neutron scattering operator from energy g to g' and makes the flux equations be coupled with respect to the energy variable.

2 The parareal algorithm

The unsteady problem $(*)$ can be written in a more compact form:

$$\frac{\partial y}{\partial t} + \mathscr{A}(t;y) = 0\ , t \in [\tau_0, \tau_1]; \tag{1}$$

it is complemented with initial conditions at time $t = \tau_0 : y(\tau_0) = y_0$. The parareal in time algorithm applied to (1) is an iterative technique where, at each iteration a predictor corrector propagation is proposed based on two propagators : a fine one $\mathscr{F}_{\tau_0}^{\tau_1}(y_0)$ that computes an approximation of the solution of (1) at time τ_1 accurately but slowly, and a coarse one $\mathscr{G}_{\tau_0}^{\tau_1}(y_0)$ that computes an other approximation quickly but not so accurately (and not accurately enough). In addition to these two propagators $\mathscr{F}$ and $\mathscr{G}$, the parareal in time algorithm is based on the division of the full interval $[0,T]$ into N sub-intervals $[0,T] = \bigcup_{n=0}^{N-1}[T_n, T_{n+1}]$ that will each be assigned to a processor P_n, assuming that we have N processors at our disposal.

The value $y(T_n)$ is approximated by Y_n^k as k increases with an accuracy that tends to the one achieved by the fine solver (see [9], [2], [3] for further details). It is obtained by the recurrence relation:

$$Y_{n+1}^{k+1} = \mathscr{G}_{T_n}^{T_{n+1}}(Y_n^{k+1}) + \mathscr{F}_{T_n}^{T_{n+1}}(Y_n^k) - \mathscr{G}_{T_n}^{T_{n+1}}(Y_n^k),\ n = 1, ..., N \tag{2}$$

starting from $Y_{n+1}^0 = \mathscr{G}_{T_n}^{T_{n+1}}(Y_n^0)$ In this work, the recently described distributed algorithm (summarized in [1]) has been used for the practical implementation of parareal. It represents an improvement of parareal from the algorithmic point of vue.

The first method of implementation was indeed suggested in [9] and consisted on a master-slave algorithm where the master carried out the coarse propagation in the whole time interval (each slave being in charge of the fine propagations over its assigned time slice and sending $\mathscr{F}_{T_n}^{T_{n+1}}(Y_n^k)$ to the master so that the master computed the parareal corrections (2) $\forall n$). This original algorithm gives rise to two main computing drawbacks: the coarse propagation by the master is a bottleneck in the computation and the memory requirement in the master processor scales linearly with the number of slaves. The distributed algorithm improves both aspects and can easily be implemented via the MPI library: for each processor P_n the fine and the coarse solvers are propagated over $[T_n, T_{n+1}]$ and the parareal correction Y_{n+1}^{k+1} is carried out. The process is repeated until convergence, i.e. $\|Y_n^{k+1} - Y_n^k\| < \eta$, $\forall n$, where η is a given tolerance.

It is easy to realize that this kind of implementation does not change the number of iterations in order the parareal algorithm to converge but it provides better speed-ups than the original master-slave version. This is the reason why the distributed algorithm has been implemented in this study. Indeed, if we do not take into account the communication time between processors, the theoretical speed-ups of the distributed and master-slave algorithms are respectively (see [1]):

$$S_{distrib} = \frac{N}{Nr + k^*(1+r)} \quad ; \quad S_{MS} = \frac{N}{Nr(1+k^*) + k^*} \tag{3}$$

where r is the ratio between the two solution times of the two propagators $\mathscr{G}$ and $\mathscr{F}$ and k^* is the number of parareal iterations needed in order to converge.

3 Numerical simulation

3.1 Definition of the test case:

The parareal algorithm has been implemented with an implicit discretization in time. Note that here we have used the same physical model (diffusion) for both the coarse and the fine solvers (the only difference is the size of the time steps used to solve equation $(*)$ δt for $\mathscr{F}$ and $\Delta t = T_{n+1} - T_n$ for $\mathscr{G}$). At each time step, a Gauss-Seidel iteration is used on the energy groups and the spatial discretization is done with RT-1 finite elements (see [4]).

The geometry and history that have been chosen for the simulation is the so called TWIGL benchmark that represents a rod withdrawal (see [8]). The geometry of the core is three-dimensional. A cross-sectional view of it is specified in FIGURE 1 where only a quarter of it has been represented (the rest can be inferred by symmetry). The first group of rods (yellow) is withdrawn from $t = 0$ ($z = 100$ *cm* measured starting from below) until $t = 26.6$ *s*. ($z = 180$ *cm*) at a constant velocity. The second group of rods (brawn) is inserted from $t = 7.5$ *s*. ($z = 180$ *cm*) until $t = 47.7$ *s*. ($z = 60$ *cm*) and the simulated interval of time is $[0, T]$ with $T = 66.6$ *s*.

Computations have been carried out with $G = 2$ energy groups, $L = 6$ precursors. The coefficients of $(*)$ remain constant in time and only the geometry varies. The fine solver has a fixed time step of $\delta t = 1/6$ *s*.

The scaling has been evaluated with a convergence test associated in which the tolerance η has been fixed to the precision of the numerical scheme (i. e. $\eta \sim 10^{-3}$). With this threshold, convergence has been achieved after only $k^* = 2$, 3 or at most 4 iterations of the parareal in time algorithm.

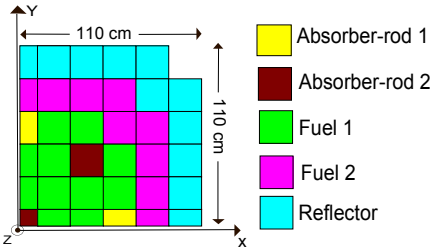

Fig. 1 Cross-sectional view of a quarter of the core in the TWIGL benchmark

3.2 Strong scaling results:

For the strong scaling analysis, the same problem has been solved on an increasing number N of processors. The size of each interval, equal to the time step of the coarse solver, has been reduced from $\Delta t = 50\delta t$ to $\Delta t = 5\delta t$ in order to increase the number of processors. Therefore, as N varies, the ratio r and the number of parareal iterations k^* change. With the computed k^* and using $\delta t/\Delta t$ as an approximation of r, one can infer from formula 3 the optimal speed-up values that can be obtained in our current case with the distributed algorithm (measured speed-ups are of course lower due to the communication time that is not taken into account in formula 3). The values are plotted in FIGURE 2, where the theoretical speed-ups of the master-slave algorithm are also shown in order to compare both methods.

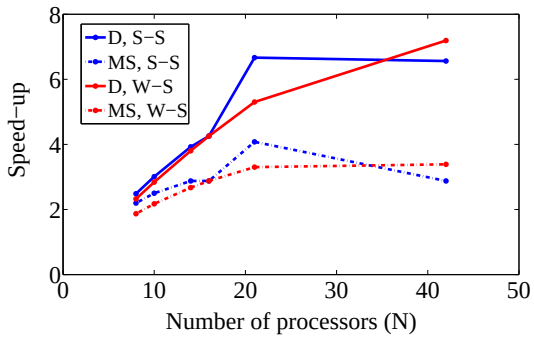

Fig. 2 Optimal speed-ups obtained for the scaling tests (D=Distributed algorithm; MS= Master-Slave algorithm; S-S= Strong Scaling; W-S= Weak Scaling)

As it can be observed, the distributed algorithm performs better for any number N of processors. For a reduced number of processors, the speed-ups are similar because both algorithms increase like N/k^* for N small enough. However, when N becomes significant in formulae 3, the distributed algorithm will behave like N/r and the master-slave method like $N/(r(1+k^*))$, making the distributed algorithm become more perfomant on a wider range of values of N. The performances reach a plateau and even decrease when N becomes very large ($N > 20$ in our case) because the cost of $\mathscr{G}$ becomes equivalent to the cost of $\mathscr{F}$ (r tends to 1).

3.3 Weak scaling results

For this alternative evaluation of the scaling, the same geometry as before has been used. We now consider the case in which the problem has a variable length $T = N\Delta t$

and the time step of the coarse solver Δt is fixed (i.e. the size of the problem linearly increases with the number N of processors). For our computations, the fine and coarse time steps are fixed to $\delta t = 1/6\ s.$ and $\Delta t = 50\delta t$ respectively.

The control rods are inserted and withdrawn periodically with a sequence of motion that creates fluctuations in the total power. With the computed k^*, the optimal speed-ups for the distributed algorithm are plotted in FIGURE 2 and compared to the master-slave model. The most important result here is that the distributed algorithm can effectively speed-up long time calculations as it can be observed. When compared to the master-slave implementation for large values of N, the distributed algorithm has a clear advantage because the increase of k^* has not such a strong negative impact on it than on the master-slave implementation (as it can also be seen in FIGURE 2).

4 About the convergence of parareal in the kinetic neutron diffusion equation

The analysis of the convergence process can be done into two ways, either by looking only at the history of the values at each T_n, $1 \leq n \leq N$, or by looking at the error at each fine discrete time $m\Delta t$:

$$e^k(t_n + m\delta t)_{fine} = \frac{\|\mathscr{F}_{T_n}^{T_n+m\delta t}(\Phi_n^k) - \mathscr{F}_0^{T_n+m\delta t}(\Phi_0)\|_{L^2}}{\|\Phi_0\|_{L^2}} \tag{4}$$
$$\forall n = 1,...N,\ \forall m = 0,1,...\frac{\Delta t}{\delta t},\ \forall k = 0,...,N-1$$

FIGURE 3 illustrates the global convergence history according to formula 4. Above the convergence threshold, we note a surprising behavior of the error over each interval $[T_n, T_{n+1}]$ that is, in most cases, neither linear nor constant despite that $(*)$ is linear. The following analysis will explain that this is due to the presence of the radioactive isotopes.

Under several hypothesis (see the point kinetics approximation in [10]), the kinetic behavior of system $(*)$ can be analysed through a set of first order ODE's of the form:

$$(5)\begin{cases} \frac{d\Phi(t)}{dt} = \alpha\Phi(t) + \sum_{\ell=1}^{L} \lambda_\ell C_\ell(t) \\ \frac{dC_\ell(t)}{dt} = \gamma_\ell \Phi(t) - \lambda_\ell C_\ell(t), \forall \ell = 1,\ldots,L \\ \Phi(0) = \Phi_0,\ C_\ell(0) = C_{\ell,0} \end{cases}$$

where the coefficients are in the range $-0.5 \leq \alpha \leq -6.10^{-3}$, while for any $\ell, 1 \leq \ell \leq L$, $10^{-2} \leq \lambda_\ell \leq 4$ and $3.10^{-3} \leq \gamma_\ell \leq 3,4.10^{-2}$

In order to understand the phenomenon in the simulation of $(*)$ represented in FIGURE 3, let us consider the case where $L = 1$ in (5). Due to linearity, the evolution of the error (e_{fine}) between the parareal fine propagator and the sequential fine one

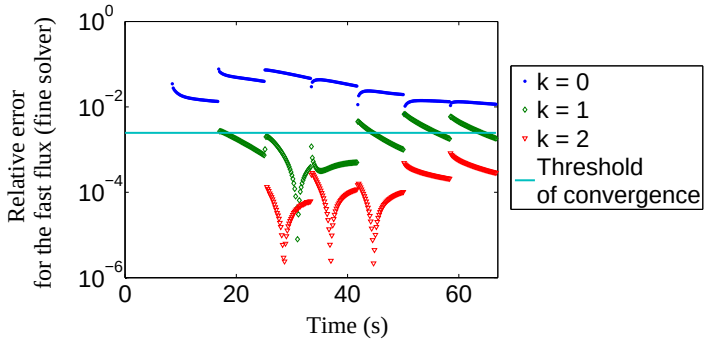

Fig. 3 Example of convergence of the fine parareal solution $\mathscr{F}_{T_n}^{T_n+m\delta t}(\Phi_n^k)$ (TWIGL benchmark, $N = 8$ processors, $\Delta t = 8.3$ *s*.)

follows the same evolution as Φ in (5) over each interval $[T_n, T_{n+1}]$ starting from an initial error $\delta\Phi$ over Φ and δC over $C = C_1$. This system can be solved and the solution is the sum of two exponential behaviors $e^{\mu_- t}$ and $e^{\mu_+ t}$ where $\mu_\pm$ are the two eigenvalues associated with the problem : $\mu_\pm = \frac{(\alpha-\lambda)\pm\sqrt{(\lambda+\alpha)^2+4\lambda\gamma}}{2}$. In the range of values where the physical parameters lie, $\lambda+\alpha$ is not small and we can consider that $\gamma = \frac{(\lambda+\alpha)^2}{4\lambda}(\varepsilon + \circ(\varepsilon))$. In this case, the eigenvalues behave as $\mu_\pm = \frac{\alpha-\lambda\pm|\lambda+\alpha|}{2} \pm \frac{|\lambda+\alpha|}{4}\varepsilon + \circ(\varepsilon)$ where ε is a small quantity, the error $\delta\Phi(t) = \delta\Phi_0 e^{\alpha t} + \frac{\lambda}{\lambda+\alpha}\delta C_0\left(e^{\alpha t} - e^{-\lambda t}\right) + \theta(\delta\Phi_0, \delta C_0, \alpha, \lambda)\varepsilon + \circ(\varepsilon)$, with θ gathering the terms at order ε. At first order, and depending on the values of α and λ, $\delta\Phi$ (and therefore e_{fine}) will present an exponentially decreasing trend (e.g. $\alpha = -0.006$, $\lambda = 4$) or a brief increase followed by a decrease (e.g. $\alpha = -0.5$, $\lambda = 0.01$) as it appears in FIGURE 3.

Conclusion

The results of this study show that the parareal distributed algorithm can effectively speed-up neutron kinetic diffusion calculations. They can certainly be improved by coupling parareal with spatial domain decomposition. A further analysis needs to be done on the impact of the communication time between processors.

An analysis of a surprising behavior of the error within each interval $[T_n, T_{n+1}]$ has also been explained and is a consequence of a special tune of the parameters.

Note also that these results represent the first implementation of the parareal in time algorithm within the industrial solver MINOS so the current results represent as well a successful industrial application of parareal.

These results are encouraging because they open the door to the construction of kinetic transport solvers. Our ongoing study is therefore to explore whether the parareal algorithm can successfully accelerate such calculations.

Acknowledgements

This work was supported in part by the joint research program MANON between CEA-Saclay and University Pierre et Marie Curie-Paris 6.

References

1. Aubanel, E.: Scheduling of tasks in the parareal algorithm. Parallel Computing **37**, 172–182 (2011)
2. Baffico, L., Bernard, S., Maday, Y., Turinici, G., Zérah, G.: Parallel-in-time molecular-dynamics simulations. Phys. Rev. E **66** (2002)
3. Bal, G., Maday, Y.: A parareal time discretization for non-linear PDE's with application to the pricing of an American put. Recent developments in domain decomposition methods **23**, 189–202 (2002)
4. Baudron, A.M., Lautard, J.J.: A simplified P_n solver for core calculation. Nuclear Science and Engineering **155**, 250–263 (2007)
5. Baudron, A.M., Lautard, J.J., Riahi, K., Maday, Y., Salomon, J.: Time-parareal parallel in time integrator solver for time-dependent neutron diffusion equation. Submitted
6. Dautray, R., Lions, J.L.: Analyse mathématique et calcul numérique. Masson, Cambridge, Massachusetts (1984)
7. Jamelot, E., Baudron, A.M., Lautard, J.J.: Domain decomposition for the SPN solver MINOS. Transport Theory and Statistical Physics **41**:7, 495–512 (2012)
8. Langenbuch, S., Maurer, W., Werner, W.: Coarse-mesh flux expansion method for the analysis of space-time effects in large light water reactor cores. Nuclear Science and Engineering **63**, 437–456 (1977)
9. Lions, J., Maday, Y., Turinici, G.: Résolution d'EDP par un schéma en temps pararéel. C. R. Acad. Sci. Paris (2001). T. 332, Série I, p. 661-668
10. Reuss, P.: Précis de neutronique. EDP Sciences, Collection Génie Atomique (2003)