MINARET: Towards a parallel 3D time-dependent neutron transport solver

Jean-Jacques Lautard, Yvon Maday, Olga Mula

MINARET: Towards a parallel 3D time-dependent neutron transport solver

Lautard, J.-J.[a,b], Maday, Y.[b,c,d,e], Mula, O.[a,b,c]

[a]CEA Saclay, DEN/DANS/DM2S/SERMA/LLPR, 91191 Gif-Sur-Yvette CEDEX - France
[b]LRC MANON – CEA/DEN/DANS/DM2S and UPMC-CNRS/LJLL.
[c]Sorbonne Universités, UPMC Univ Paris 06 and CNRS UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France.
[d]Institut Universitaire de France
[e]Division of Applied Mathematics, Brown University, Providence RI, USA.

Abstract

We present in this work a newly developed time-dependent 3D multigroup discrete ordinates (S_n) neutron transport solver to obtain accurate solutions to the kinetic neutron transport equation on reactor cores. The implemented module has been included in a code called MINARET, which is developed at CEA in the framework of the APOLLO3® project. Given that the main obstacle in the use of this type of accurate solver relies in the long computational times that the resolution requires, the MINARET solver has been used in the present work as a support for a study about sequential and parallel acceleration techniques. A particular focus will be given to the parallel acceleration techniques, where we will present results about the parallelization of two of the variables involved in our equation: the angular directions and the time. This last variable has been parallelized by a (time) domain decomposition method called the parareal in time algorithm.

Keywords: MINARET; neutronics; neutron transport; parareal in time algorithm; parallel algorithm; unsteady phenomena

1. Introduction

In the framework of numerical simulations, the advances of computing architectures in the last decades have resulted in a progressive replacement of traditional coarse models by ever-increasing finer ones. Historically speaking, this trend has been possible on a first stage thanks to the advances of computer power capabilities per machine.

In the field of nuclear core calculations and, more particularly, regarding the deterministic resolution of the time-dependent neutron transport equation, the enhanced processor capabilities have led to significant advances. Indeed, the quite recent developments of the 2D time-dependent transport code DORT-TD [1] and its posterior extended three dimensional version TORT-TD (and even its coupling with thermal-hydraulics codes [2]) confirm this fact and have proven that the traditional diffusion [3], improved quasi-static [4] or point kinetics [5] traditional approximations can nowadays start to be overcome. While this represents a significant progress for security assessment, the long computing times of neutron transport remain still nowadays the main obstacle to face in order to definitely use transport on a regular basis. In this framework, the present work is a contribution to show that the resolution of the time dependent neutron transport equation in realistic 3D geometries is feasible in a reasonable amount of time by the use of modern parallel computer architectures together with innovative numerical schemes.

For this purpose, we start by presenting in sections 2 and 3 the newly developed time dependent 3D multigroup discrete ordinates neutron transport code that has been included in the MINARET solver (which is a tool developed at CEA in the framework of the APOLLO3® project, [6]). In particular, in section 2 some properties and notations of the equation (that depends on time, energy, angular and space variables) will be recalled and section 3 will be devoted to the set of discretizations applied to each of the variables together with the numerical iterative schemes that have been implemented.

We will after come to the sequential and parallel acceleration techniques that we have explored to accelerate the computations. Our numerical results will be related to a 3D test case described in section 4. In section 5, we will present the two traditional sequential accelerations that have been included (the Chebyshev extrapolation and the

Diffusion Synthetic Acceleration) and provide some numerical evidences of their efficiency. We will also explain how extra speed-ups can be obtained by the choice of an appropriate starting guess in the outer iterations.

The remaining sections will be devoted to the main contribution of the present work, which consists in the acceleration of MINARET by the parallelization of some of the present variables. Section 6 deals with the technique implemented to parallelize the angular variable (following the previous work of [7]). In section 7, a study about the additional speed-up that can be obtained if this angular parallelization is coupled with the parallelization of the time variable will be presented. This task has been performed by a domain decomposition method called the parareal in time algorithm (see [8] [9] [10] for its theoretical foundations) and has been implemented following the preliminary analysis of [11] [12] where this technique was successfully applied to the time-dependent neutron diffusion equation. We will propose an extension of the classical theoretical expected speed-up formulae (summarized in, e.g. [13]) in order to take into account the coupling of parareal with other iterative methods in a more realistic way. The comparison of these formulae with the results obtained with MINARET in a numerical test case will highlight the potentialities of parareal to accelerate long time transport computations thanks to a negligible impact of the communication time between processors.

The methods used to parallelize the angular and temporal variables are not the only possibility to exploit concurrency: the parallelization of the energy variable seems to be a promising field to explore to accelerate calculations as outlined in [14]. Regarding the spatial variables however, because of the hyperbolic nature of the space in the transport equation, the spatial domain decomposition techniques existing for elliptic problems (like the ones explored in [15] for the steady-state neutron SP_n equations) cannot be applied and the efficient decomposition of the spatial domain in the transport equation represents still nowadays a difficult problem. It will nevertheless be discussed in section 7.4 how the parareal in time algorithm could be used for the parallelization of the spatial variables in transport equations (as an alternative to other domain decomposition methods explored for this kind of problem like the works of [16] [17]).

2. The time-dependent neutron transport equation

The evolution of the angular flux ψ of neutrons in a reactor core $\mathcal{R}$ is governed by a linear Bolzman equation whose terms physically express a balance between the free neutrons that are created and that disappear in the core. We will consider here the three-dimensional case ($\mathcal{R} \subset \mathbb{R}^3$) where ψ depends on 7 variables, namely the time $t \in [0, T]$, the position within the reactor denoted as $\boldsymbol{r} \in \mathcal{R}$, the velocity of the neutrons $\boldsymbol{v} = \sqrt{2E/m}\,\boldsymbol{\omega}$ where $E \in [E_{\min}, E_{\max}]$ stands for the energy of the neutron, $\boldsymbol{\omega} = \frac{\boldsymbol{v}}{|\boldsymbol{v}|} \in \mathbb{S}_2$ stands for the direction of the velocity and m is the mass of the neutron. We will have $\boldsymbol{v} \in \mathcal{V}$, where $\mathcal{V} = \mathbb{S}_2 \times [E_{\min}, E_{\max}]$ is a compact subset of $\mathbb{R}^3$. The fission chain reaction that takes place inside the core leads to the presence of some radioactive isotopes that emit neutrons with a given delay (we refer to them as precursors of delayed neutrons). This phenomenon must be taken into account in our balance equation, hence the coupling of the Bolzman equation with a set of first order ODE's expressing the evolution in $\mathcal{R}$ of the precursors' concentration that will be denoted as $\mathbf{C} = \{C_\ell\}_{\ell \in \{1,\dots,L\}}$.

The set $(\psi, \mathbf{C})$ is thus the solution to the following initial value problem over the domain $\mathcal{D} = \{(t, \boldsymbol{r}, \boldsymbol{\omega}, E) \in [0, T] \times \mathcal{R} \times \mathbb{S}_2 \times [E_{\min}, E_{\max}]\}$:

$$\begin{cases} \frac{1}{|\boldsymbol{v}|}\partial_t \psi(t, \boldsymbol{r}, \boldsymbol{\omega}, E) + (L - H - F - Q)\psi(t, \boldsymbol{r}, \boldsymbol{\omega}, E) = 0 \\ \partial_t C_\ell(t, \boldsymbol{r}) = -\lambda_\ell C_\ell(t, \boldsymbol{r}) \\ \qquad\qquad + \int_{E'=E_{\min}}^{E_{\max}} \beta_\ell(t, \boldsymbol{r}, E')(\nu\sigma_f)(t, \boldsymbol{r}, E')\phi(t, \boldsymbol{r}, E')dE', \ \forall \ell \in \{1, \dots, L\}, \end{cases} \tag{1}$$

where $\phi(t, \boldsymbol{r}, E) = \int_{\mathbb{S}_2} \psi(t, \boldsymbol{r}, \boldsymbol{\omega}', E)d\boldsymbol{\omega}'$ is the scalar flux and the following operator notations have been used:

- $L\psi(t, \boldsymbol{r}, \boldsymbol{\omega}, E) = (\boldsymbol{\omega}.\nabla + \sigma_t(t, \boldsymbol{r}, E))\,\psi(t, \boldsymbol{r}, \boldsymbol{\omega}, E)$ is the advection operator,
- $H\psi(t, \boldsymbol{r}, \boldsymbol{\omega}, E) = \int_{\mathbb{S}_2}\int_{E'=E_{\min}}^{E_{\max}} \sigma_s(t, \boldsymbol{r}, \boldsymbol{\omega}' \to \boldsymbol{\omega}, E' \to E)\psi(t, \boldsymbol{r}, \boldsymbol{\omega}', E')dE'd\boldsymbol{\omega}'$ is the scattering operator,

- $F\psi(t,\boldsymbol{r},\boldsymbol{\omega},E) = \dfrac{\chi_p(t,\boldsymbol{r},E)}{4\pi}\displaystyle\int_{E'=E_{\min}}^{E_{\max}} (1-\beta(t,\boldsymbol{r},E'))\,(\nu\sigma_f)(t,\boldsymbol{r},E')\phi(t,\boldsymbol{r},E')dE'$ is the prompt fission operator,
- $Q\psi(t,\boldsymbol{r},\boldsymbol{\omega},E) = \sum\limits_{\ell=1}^{L} \lambda_\ell\chi_{d,\ell}(t,\boldsymbol{r},E)C_\ell(t,\boldsymbol{r})$ is the delayed fission source.

In the enlisted terms, $\sigma_t(t,\boldsymbol{r},E)$ denotes the total cross-section and $\sigma_s(t,\boldsymbol{r},\boldsymbol{\omega}'\to\boldsymbol{\omega},E'\to E)$ is the scattering cross-section from energy E' and direction $\boldsymbol{\omega}'$ to energy E and direction $\boldsymbol{\omega}$. $\sigma_f(t,\boldsymbol{r},E)$ is the fission cross-section. $\nu(t,\boldsymbol{r},E)$ is the average number of neutrons emitted and $\chi_p(t,\boldsymbol{r},E)$ and $\chi_{d,\ell}(t,\boldsymbol{r},E)$ are respectively the prompt spectrum and the delayed spectrum of precursor ℓ. λ_ℓ and $\beta_\ell(t,\boldsymbol{r},E)$ are the decay constant and the delayed neutron fraction of precursor ℓ respectively and $\beta(t,\boldsymbol{r},E) = \sum\limits_{\ell=1}^{L}\beta_\ell(t,\boldsymbol{r},E)$.

We will work with initial conditions ψ^0 and $C_{\ell,0}$ at $t=0$ and vacuum boundary conditions over $\partial\mathcal{R}$, i.e.

$$\begin{cases}\psi = 0, \text{ on } [0,T]\times\partial\mathcal{R}_-\times\mathbb{S}_2\times\mathbb{R}_+\\ \psi(0,.) = \psi^0(.), \text{ on } \mathcal{R}\times\mathbb{S}_2\times\mathbb{R}_+\\ C_\ell(0,.) = C_{\ell,0}(.) \text{ on } \mathcal{R},\end{cases}$$

where $\partial\mathcal{R}_- := \{\boldsymbol{r}\in\partial\mathcal{R}\,|\,\boldsymbol{\omega}.\vec{n} < 0\}$ denotes the part of the boundary where the angular flux is incoming. The knowledge of the initial conditions ψ^0 and $C_{\ell,0}$ is a complex issue in itself. In nuclear safety computations like the ones we are interested in, it is of special interest to analyze transients starting from a stable state of the core. The derivation of this state is related to an eigenvalue problem whose foundations are very well established. We refer to [5] and [18] for physical and mathematical aspects of it and to [19] for numerical details about its computation in the MINARET solver.

Remark 2.1 (A diffusion problem as an approximation to the transport equation (1)). *Under some given physical hypothesis (see, e.g. [5] and [18]), the angular mean value $\phi(t,\boldsymbol{r},E) = \int_{\mathbb{S}_2}\psi(t,\boldsymbol{r},\boldsymbol{\omega}',E)d\boldsymbol{\omega}'$ of the angular flux $\psi(t,\boldsymbol{r},\boldsymbol{\omega},E)$ satisfies a diffusion equation that has the advantage of being much less computationally expensive than the transport equation from the memory storage and from the computational time point of view. Although the present work deals with the resolution of the transport equation (1), the existence of such surrogate approximation will be used in our case in some acceleration techniques.*

3. Discretization and implementation in the MINARET solver

With the exception of some simple cases (see [5] for further references) where problem (1) can exactly be solved, the resolution of (1) needs to be numerically addressed and requires discretizations and approximations of the involved variables. The MINARET solver uses traditional discretization techniques and this section briefly explains them by putting special stress on the iterative numerical schemes that have been implemented.

We start by discretizing the energy variable and deriving the multigroup version of equation (1). The strategy is based on the division of the energy interval into G subintervals: $[E_{\min},E_{\max}] = [E_G,E_{G-1}]\cup\cdots\cup[E_1,E_0]$. For $1\le g\le G$, we denote by ψ^g the approximation of ψ in the subinterval $[E_g,E_{g-1}]$. Further, let $[0,T] = \bigcup_{n=0}^{N-1}[t_n,t_{n+1}]$ be a division of the full time interval and $\Delta T_{n+1} = t_{n+1}-t_n$. An Euler-backward scheme for the time variable is then applied. Let $\psi^{g,n}(\boldsymbol{r},\boldsymbol{\omega})$ be the approximation of $\psi(t,\boldsymbol{r},\boldsymbol{\omega},E)$ at time $t=t_n$ and for $E\in[E_g,E_{g-1}]$. Given $\{\psi^{g,n}(\boldsymbol{r},\boldsymbol{\omega})\}_{g=1}^G$, the set of unknowns $\{\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega})\}_{g=1}^G$ for the time t_{n+1} is the solution of the following set of coupled source problems:

$$\begin{cases}\text{Find over } \mathcal{R}\times\mathbb{S}_2 \text{ the angular flux } \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) \text{ that is the solution of:}\\ \left(L^g - H^g - \tilde{F}^g - \tilde{Q}^g\right)\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \tilde{S}^{g,n}(\boldsymbol{r},\boldsymbol{\omega}), \quad \forall g\in\{1,\dots,G\}.\end{cases} \tag{2}$$

The following notations have been used:

- $\tilde{S}^{g,n}(\boldsymbol{r},\boldsymbol{\omega}) := \dfrac{\psi^{g,n}(\boldsymbol{r},\boldsymbol{\omega})}{V^g \Delta T_{n+1}}$, where V^g is the average velocity of the neutrons whose energy belong to the interval $[E_g, E_{g-1}]$. Note that for the computation of $\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega})$, the term $\tilde{S}^{g,n}(\boldsymbol{r},\boldsymbol{\omega})$ is known and is a source for the equation.

- $L^g \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \left(\boldsymbol{\omega}.\nabla + \left(\sigma_t^{g,n+1}(\boldsymbol{r}) + \dfrac{1}{V^g \Delta T_{n+1}}\right)\right)\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega})$

- $H^g \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \sum_{g'=1}^{G} H^{g'\to g}\psi^{g',n+1}(\boldsymbol{r},\boldsymbol{\omega})$, with
  $H^{g'\to g}\psi^{g',n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \displaystyle\int_{\mathbb{S}_2} \sigma_s^{g'\to g,n+1}(\boldsymbol{r},\boldsymbol{\omega}'\to\boldsymbol{\omega})\psi^{g',n+1}(\boldsymbol{r},\boldsymbol{\omega}')d\boldsymbol{\omega}'$.

- $\tilde{F}^g \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \dfrac{\chi_p^{g,n+1}(\boldsymbol{r})}{4\pi}\sum_{g'=1}^{G}\left(1-\beta^{g',n+1}(\boldsymbol{r})\right)(\nu\sigma_f)^{g',n+1}(\boldsymbol{r})\phi^{g',n+1}(\boldsymbol{r})$, where $\phi^{g,n+1}(\boldsymbol{r}) = \displaystyle\int_{\mathbb{S}_2}\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega})d\boldsymbol{\omega}'$, $\forall g \in \{1,\dots,G\}$.

- $\tilde{Q}^g \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \sum_{\ell=1}^{L}\lambda_\ell \chi_{d,\ell}^{g,n+1}(\boldsymbol{r})C_\ell^{n+1}(\boldsymbol{r})$

The coefficients $\sigma_t^{g,n+1}(\boldsymbol{r})$, $\sigma_s^{g'\to g,n+1}(\boldsymbol{r},\boldsymbol{\omega}'\to\boldsymbol{\omega})$, $(\nu\sigma_f)^{g,n+1}$, $\chi_p^{g,n+1}(\boldsymbol{r})$ and $\chi_{d,\ell}^{g,n+1}(\boldsymbol{r})$ correspond to energy average values in $[E_g, E_{g-1}]$ at time $t = t_n$ of the coefficients $\sigma_t(t,\boldsymbol{r},E)$, $\sigma_s(t,\boldsymbol{r},\boldsymbol{\omega}'\to\boldsymbol{\omega},E'\to E)$, $\sigma_f(t,\boldsymbol{r},E)$, $\nu(t,\boldsymbol{r},E)$, $\chi_p(t,\boldsymbol{r},E)$ and $\chi_{d,\ell}(t,\boldsymbol{r},E)$. We also have $\beta^{g,n+1}(\boldsymbol{r}) = \sum_{\ell=1}^{L}\beta_\ell^{g,n+1}(\boldsymbol{r})$. The Euler backward scheme applied to the precursors' equation provides $C_\ell^{n+1}(\boldsymbol{r})$ for any $\ell \in \{1,\dots,L\}$:

$$C_\ell^{n+1}(\boldsymbol{r}) = \frac{1}{1+\lambda_\ell \Delta T_{n+1}}C_\ell^n(\boldsymbol{r}) + \frac{\Delta T_{n+1}}{1+\lambda_\ell\Delta T_{n+1}}\sum_{g'=1}^{G}\beta_\ell^{g',n+1}(\boldsymbol{r})(\nu\sigma_f)^{g',n+1}(\boldsymbol{r})\phi^{g',n+1}(\boldsymbol{r}). \tag{3}$$

The insertion of (3) into (2) finally yields the set of source problems:

$$\begin{cases}\text{Find over } \mathcal{R}\times\mathbb{S}_2 \text{ and } \forall g \text{ the angular flux } \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) \text{ that is the solution of:}\\ \left(L^g - H^g - F^g\right)\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = S^{g,n}(\boldsymbol{r},\boldsymbol{\omega}),\end{cases} \tag{4}$$

where:

- $F^g\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \sum_{g'=1}^{G}F^{g',g}\psi^{g',n+1}$ and
  $$F^{g',g}\psi^{g',n+1} = \left(\frac{\chi_p^{g,n+1}(\boldsymbol{r})}{4\pi}\left(1-\beta^{g',n+1}(\boldsymbol{r})\right) + \sum_{\ell=1}^{L}\frac{\lambda_\ell\beta_\ell^{g'}\chi_{d,\ell}^{g}\Delta T_{n+1}}{1+\lambda_\ell\Delta T_{n+1}}\right)(\nu\sigma_f)^{g',n+1}(\boldsymbol{r})\phi^{g',n+1}(\boldsymbol{r}),$$

- $S^{g,n}(\boldsymbol{r},\boldsymbol{\omega}) := \dfrac{\psi^{g,n}(\boldsymbol{r},\boldsymbol{\omega})}{V^g\Delta T_{n+1}} + \dfrac{1}{1+\lambda_\ell\Delta T_{n+1}}C_\ell^n(\boldsymbol{r})$.

Because of the coupling in the energy variable, system (4) is iteratively solved with a numerical method that we will call "generalized Gauss-Seidel scheme" (these are the so called "outer iterations" in neutronics). The scheme goes as follows: let $\psi_{(M)}^{g,n+1}$ be the approximation of $\psi^{g,n+1}$ at iteration number M. If we denote

$$\boldsymbol{\psi}^{\boldsymbol{n+1}} = \begin{pmatrix}\psi^{1,n+1}\\ \psi^{2,n+1}\\ \vdots\\ \psi^{G,n+1}\end{pmatrix} \quad ; \quad \boldsymbol{S}^{\boldsymbol{n}} = \begin{pmatrix}S^{1,n}\\ S^{2,n}\\ \vdots\\ S^{G,n}\end{pmatrix}, \tag{5}$$

then the scheme reads:

$$\begin{cases} M_{G,G}\boldsymbol{\psi}^{n+1}_{(M+1)} = N_{G,G}\boldsymbol{\psi}^{n+1}_{(M)} + \boldsymbol{S}^{n} \\ \boldsymbol{\phi}^{n+1}_{(M=0)} \text{ given,} \end{cases} \tag{6}$$

where $A_{G,G} = M_{G,G} - N_{G,G}$, with

$$M_{G,G} = \begin{pmatrix} L^1 - H^{1\to 1} & 0 & \cdots & 0 \\ -H^{1\to 2} & L^2 - H^{2\to 2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -H^{1\to G} & -H^{2\to G} & \cdots & L^G - H^{G\to G} \end{pmatrix} \tag{7}$$

and

$$N_{G,G} = \begin{pmatrix} F^{1,1} & H^{2\to 1} + F^{2,1} & \cdots & H^{G\to 1} + F^{G,1} \\ F^{1,2} & F^{2,2} & \cdots & H^{G\to 2} + F^{G,2} \\ \vdots & \vdots & \ddots & \vdots \\ F^{1,G} & F^{2,G} & \cdots & F^{G,G} \end{pmatrix}. \tag{8}$$

Note that the difference between this scheme and a traditional Gauss-Seidel lies in the "explicit" treatment of the fission terms $F^{g',g}$ for $g' \leq g$.

Remark 3.1. *Despite the fact that we look for angular fluxes, in (6) the initial guess $\boldsymbol{\phi}^{n+1}_{(M=0)}$ corresponds to flux moments. This is due to the fact that our iterative scheme is built such that one does not need to give initial angular flux guesses but only flux moments.*

It has been proven in chapter 1 of [20] (theorem 1.6.1) that this scheme converges for small enough time steps. In MINARET, the iterations are performed until the average error in the scalar flux

$$e^{n+1}_{outer}(M+1) := \frac{\sum_{g=1}^{G} \int_{\mathcal{R}} |\phi^{g,n+1}_{(M+1)} - \phi^{g,n+1}_{(M)}| \phi^{g,n+1}_{(M+1)} d\boldsymbol{r'}}{\sum_{g=1}^{G} \int_{\mathcal{R}} \phi^{g,n+1}_{(M+1)} \phi^{g,n+1}_{(M+1)} d\boldsymbol{r'}} \tag{9}$$

goes below a given convergence threshold ε_{outer}. The convergence property implies that the choice of $\boldsymbol{\phi}^{n+1}_{(M=0)}$ will have an impact on the number of iterations required to achieve a given tolerance ε_{outer} but not in the convergence itself. The most usual choice is to take $\boldsymbol{\phi}^{n+1}_{(M=0)} = \boldsymbol{\phi}^{n}_{(\infty)}$, where the subscript ∞ denotes converged values of the flux. As will be outlined in section 5, there might be cleverer choices than $\boldsymbol{\phi}^{n}_{(\infty)}$ to minimize the number of iterations required to converge.

For a given iteration M and a given energy group g, the problem to be solved reads:

$$\begin{aligned} &(L^g - H^{g\to g})\psi^{g,n+1}_{(M+1)}(\boldsymbol{r},\boldsymbol{\omega}) \\ &\quad = \sum_{g'<g} H^{g'\to g}\psi^{g',n+1}_{(M+1)} + \sum_{g'>g} H^{g'\to g}\psi^{g',n+1}_{(M)} + \sum_{g'=1}^{G} F^{g',g}\psi^{g',n+1}_{(M)} + S^{g,n}(\boldsymbol{r},\boldsymbol{\omega}), \end{aligned} \tag{10}$$

which is a monoenergetic problem of the form:

$$\boldsymbol{\omega}.\nabla\psi(\boldsymbol{r},\boldsymbol{\omega}) + \sigma_t(\boldsymbol{r})\psi(\boldsymbol{r},\boldsymbol{\omega}) - \int_{\mathbb{S}_2} \sigma_s(\boldsymbol{r},\boldsymbol{\omega'}\to\boldsymbol{\omega})\psi(\boldsymbol{r},\boldsymbol{\omega'})d\boldsymbol{\omega'} = q(\boldsymbol{r},\boldsymbol{\omega}),\ \forall(\boldsymbol{r},\boldsymbol{\omega}) \in \mathcal{R}\times\mathbb{S}_2, \tag{11}$$

where the terms σ_t, σ_s, q must be understood as generic notations whose definition must be coherent with equation (10). Since equation (11) is integral in the angular variable and differential in space, a second numerical scheme is performed (called "inner or source iterations"). If $\psi^{g,n+1}_{(M,m)}$ is the approximation of $\psi^{g,n+1}_{(M)}$ at the $m-th$ inner iteration, then $\psi^{g,n+1}_{(M,m+1)}$ is the solution of:

$$L^g \psi^{g,n+1}_{(M,m+1)}(\boldsymbol{r},\omega) = H^{g\to g}\psi^{g,n+1}_{(M,m)}(\boldsymbol{r},\omega) + \tilde{S}(\boldsymbol{r},\omega), \tag{12}$$

with

$$\tilde{S}(\boldsymbol{r},\omega) = \sum_{g'<g} H^{g'\to g}\psi^{g',n+1}_{(M+1)} + \sum_{g'>g} H^{g'\to g}\psi^{g',n+1}_{(M)} + \sum_{g'=1}^{G} F^{g',g}\psi^{g',n+1}_{(M)} + S^{g,n}(\boldsymbol{r},\omega).$$

It has been shown in [20] (section 1.6.1.2 of chapter 1) that this strategy is equivalent to a Richardson scheme. The iterations are performed until the relative error

$$e^{g,n+1}_{inner}(m+1) := \frac{\|\phi^{g,n+1}_{(M,m+1)} - \phi^{g,n+1}_{(M,m)}\|_{L^2(\mathcal{R})}}{\|\phi^{g,n+1}_{(M,1)} - \phi^{g,n+1}_{(M,0)}\|_{L^2(\mathcal{R})}} \tag{13}$$

goes below a given convergence threshold ε_{inner}.

The angular discretization of equation (11) has been performed with the discretes ordinates of order n technique (S_n), i.e., problem (11) is solved for a discrete number of directions $\{\omega_d\}_{d=1}^{D}$, where $D = n(n+2)$. The scattering operator is computed in practice by a standard expansion in Legendre polynomials of arbitrary order and the integrals in the angular variable are effectively computed by a quadrature formula involving the points and weights of the level-symmetric rule.

The space variables are treated with discontinuous Galerkin finite elements of arbitrary order and the order can be spatially adapted. The three-dimensional spatial mesh is "partially unstructured" in the sense that it is built by extrusion of an initial two-dimensional unstructured mesh (we refer to [21] for further details on MINARET's mesh generator).

Algorithm 1 summarizes the described two-stage nested iterative strategy implemented in MINARET.

Algorithm 1 The iterative strategy implemented in MINARET

```
1:  for $t_n = \Delta T$ to $N\Delta T$ do
2:      While(not converge) do (generalized GS iterations – see equation (6))
3:      for $g = 1$ to $G$ do
4:          Update fission operator
5:          Update scattering (except self-scattering)
6:          While(not converge) do (source iterations)
7:          for $\omega = \omega_1$ to $\omega_D$ do
8:              Update self-scattering
9:              Solve spatial problem (12) for $\omega$
10:         end for
11:         Diffusion Synthetic Acceleration
12:         End While
13:     end for
14:     Chebyshev Extrapolation
15:     End While
16: end for
```

4. Definition of the numerical test cases

We briefly explain in this section the two test cases that will be used to illustrate the numerical performances of the acceleration methods that are going to be discussed in the remaining of this paper.

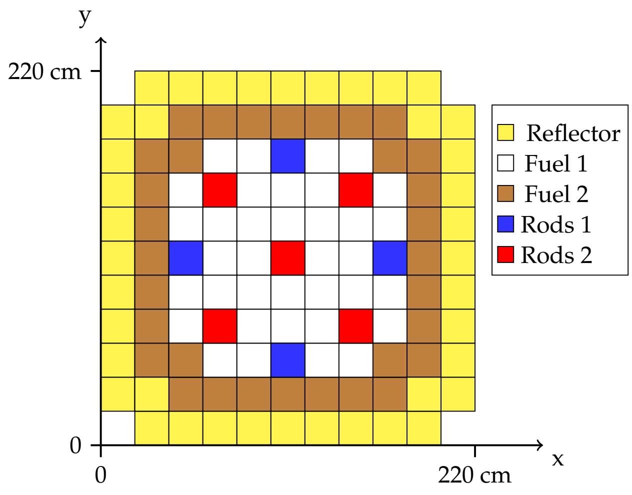
The first test case (denoted below as "case A") corresponds to the so called TWIGL benchmark and it represents a rod withdrawal (see [22]). The geometry of the core is three-dimensional and the domain is $\mathcal{R} = \{(x,y,z) \in \mathbb{R}^3 | \, 0 \leq$

$x \leq 220\ cm;\ 0 \leq y \leq 220\ cm;\ 0 \leq z \leq 200\ cm;\ \}$. A cross-sectional view at the height $z = 180\ cm$ is specified in table 1a. The first group of rods (blue) is withdrawn from $t = 0$ ($z = 100\ cm$ measured starting from below) until $t = 26.6\ s.$ ($z = 180\ cm$) at a constant speed. The second group of rods (red) 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 $T = 70\ s$ (see table 1b). The evolution of the power is also represented in table 1b.

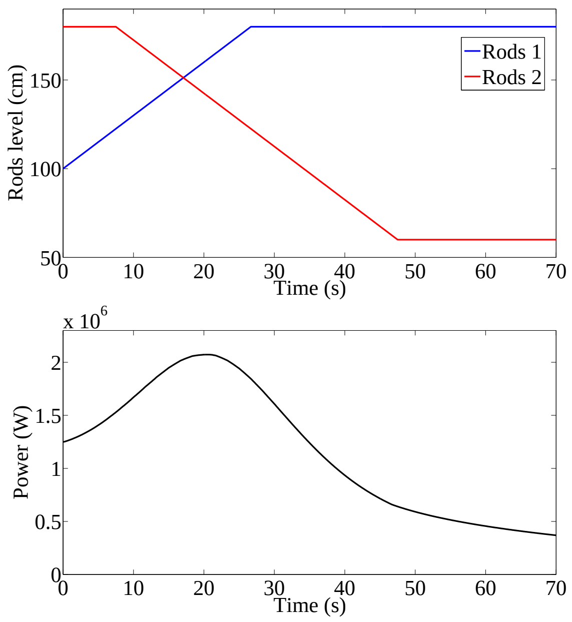
The second test case (denoted below as "case B") uses the same geometry as the TWIGL benchmark but an oscillatory sequence of motion of the rods has been devised so that power fluctuations are produced. The simulated interval of time is $[0, T]$ with $T = 250\ s$ (see table 2 for the details).

Both tests have been carried out with $G = 2$ energy groups, $L = 6$ precursors and vacuum boundary conditions. All the computations that will be presented hereafter have been obtained in a cluster of 38 nodes of 16 Gb memory, each one composed of 8 cores of 2814 MHz speed.

Remark 4.1. *In the TWIGL benchmark from the literature ([22]), calculations are done in a quarter of a core with reflective boundary conditions in the inner parts of the core. In our case, the full geometry has been computed in order to be coherent with case B that has no spatial symmetries.*

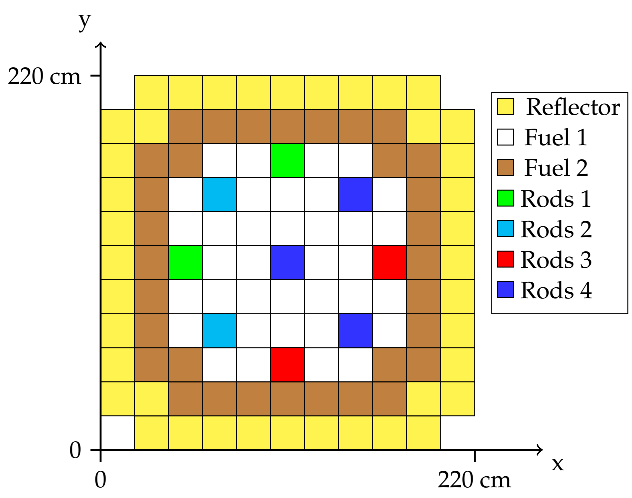


(a) Cross-sectional view of the core ($z = 180\ cm$).

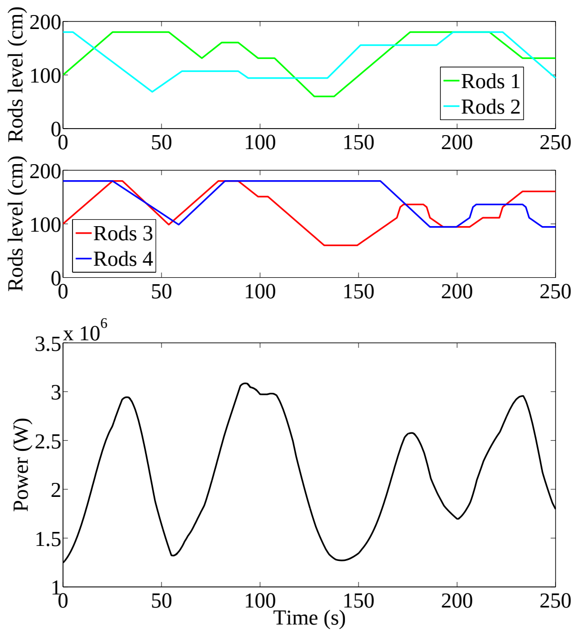


(b) Rods' vertical motion and power evolution.

Table 1: Case A (TWIGL benchmark).



(a) Cross-sectional view of the core ($z = 180\ cm$).



(b) Rods' vertical motion and power evolution.

Table 2: Case B.

5. Sequential acceleration techniques

The convergence of the iterative resolution of the multigroup problem given in (6) is often extremely slow and acceleration methods are required in order to obtain reasonable computing times.

Two traditional sequential accelerations have been included in MINARET. The first one is the Chebychev extrapolation in the outer iterations. It consists on adding a linear combination of the fluxes after each Gauss-Seidel iteration:

$$\begin{cases} M_{G,G}\boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M+1/2)}} = N_{G,G}\boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M)}} + \boldsymbol{S^n} \\ \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M+1)}} = \alpha_{M+1}\left(\boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M+1/2)}} - \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M-1)}}\right) + \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M-1)}},\ M \geq 1 \\ \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M=0)}} \text{ given.} \end{cases} \tag{14}$$

We refer to [23] for the exact form of the coefficients (α_M) and the theoretical foundations of this acceleration scheme.

Remark 5.1. *Note that the scheme (14) is well-defined thanks to the fact that* $N_{G,G}\boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M)}}$ *requires the only knowledge of* $\boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M+1)}}$ *(see the definition of* $N_{G,G}$ *in equation (8)).*

The second acceleration scheme is the so-called diffusion synthetic acceleration (DSA) that has been added for the convergence of the inner iterations. It reads:

$$\begin{cases} L^g\psi^{g,n+1}_{(M,m+1/2)}(\boldsymbol{r},\boldsymbol{\omega}) = H^{g\to g}\psi^{g,n+1}_{(M,m)}(\boldsymbol{r},\boldsymbol{\omega}) + \tilde{S}(\boldsymbol{r},\boldsymbol{\omega}), \forall g \in \{1,\dots,G\} \\ \psi^{g,n+1}_{(M,m+1)}(\boldsymbol{r},\boldsymbol{\omega}) = \psi^{g,n+1}_{(M,m+1/2)}(\boldsymbol{r},\boldsymbol{\omega}) + e^{g,n+1}_{(M,m+1)}(\boldsymbol{r}), \end{cases} \tag{15}$$

where $e^{g,n+1}_{(M,m+1)}$ is the solution of the diffusion problem:

$$\begin{cases} -\operatorname{div}\left(\dfrac{1}{3\left(\sigma_t^g(\boldsymbol{r})+\frac{1}{V^g\Delta T_{n+1}}\right)}\nabla e^{g,n+1}_{(M,m+1)}(\boldsymbol{r})\right) \\ \qquad +\left(\sigma_t^g(\boldsymbol{r})+\frac{1}{V^g\Delta T_{n+1}}-\sigma_s^{g,g}(\boldsymbol{r})\right)e^{g,n+1}_{(M,m+1)}(\boldsymbol{r}) \\ \qquad =\sigma_s(\boldsymbol{r})\left(\phi^{g,n+1}_{(M,m+\frac{1}{2})}(\boldsymbol{r})-\phi^{g,n+1}_{(M,m)}(\boldsymbol{r})\right), \quad \forall \boldsymbol{r}\in\mathcal{R} \\ e^g_{(m+1)}(\boldsymbol{r})=0, \quad \forall \boldsymbol{r}\in\partial\mathcal{R}. \end{cases} \tag{16}$$

DSA is an acceleration scheme because it acts as a preconditioner of transport to solve equation (12). We refer to [24] (sections I.D and II.B) and [19] for more theoretical details about this method. In MINARET, the spatial resolution of the DSA problem (16) is discretized with discontinuous Galerkin finite elements of the same order than the ones employed in problem (12). The discretized DSA problem is iteratively solved with a conjugate gradient method preconditioned by SSOR. If r_i denotes the residual at the i-th iteration, the DSA iterations are performed until the ratio

$$\frac{||r_i||_{L^2(\mathcal{R})}}{||r_0||_{L^2(\mathcal{R})}} \tag{17}$$

goes below a given convergence threshold ε_{DSA}.

We illustrate the performances in MINARET of both acceleration methods through some numerical results obtained for the test case A.

To begin with, table 3 lists the number of outer iterations M_{outer}, inner iterations N_{inner} and DSA iterations N_{DSA} required to perform a propagation from time 0 to time $5/3$ s. in an S_4 calculation. We also provide the exact computing times obtained in our cluster. The convergence criteria associated to the errors (9), (13) and (17) have been fixed to $\varepsilon_{outer} = 10^{-5}$, $\varepsilon_{inner} = 10^{-1}$ and $\varepsilon_{DSA} = 10^{-2}$. The product $M_{outer}N_{inner}$ is also given as an estimation of the complexity of the resolution (the complexity added by the DSA can be neglected in a first approach in a sequential calculation). As the first case of table 3 shows, it is clear that the solver needs acceleration techniques in order to converge in a reasonable time. While the inclusion of the Chebyshev extrapolation (case 2) already represents a dramatic improvement in the computing time (by reducing about 10 times the number of outer iterations), this performance can still be improved by another factor of about 10 if the Chebyshev extrapolation is coupled with DSA in the inner iterations (case 3). This is achieved thanks to the reduction of the number of inner iterations.

Case	Chebyshev	DSA	M_{outer}	N_{inner}	N_{DSA}	$M_{outer}N_{inner}$	Computing time (s)
1	No	No	678	29784	0	$\approx 2000.10^4$	7510
2	Yes	No	67	2900	0	$\approx 19.10^4$	736.5
3	Yes	Yes	59	345	1557	$\approx 2.10^4$	87.67

Table 3: An illustration of the impact on the speed-up performances of the Chebyshev extrapolation and the DSA.

Another factor of about 3 can further be obtained if the initial guess $\boldsymbol{\phi}^{n+1}_{(M=0,N=0)}$ of the outer iterations is well chosen. The classical choice is

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)}, \quad \forall g\in\{1,\dots,G\}. \tag{18}$$

This option is reasonable because for small time steps one could conjecture that the system does not change very much from t_n to t_{n+1}. Other possibilities that exploit the information of the previous time steps have been explored (these are at the cost of storing additional information). One can first try a linear extrapolation of the flux:

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} + \frac{t_{n+1}-t_n}{t_n-t_{n-1}}\left(\phi^{g,n}_{(\infty)}-\phi^{g,n-1}_{(\infty)}\right), \quad \forall g\in\{1,\dots,G\}. \tag{19}$$

However, according to the point kinetics approximation, the behavior of the flux is rather exponential and another idea would be an exponential extrapolation:

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} \exp^{\frac{t_{n+1}-t_n}{t_n-t_{n-1}} \ln\left(\frac{\phi^{g,n}_{(\infty)}}{\phi^{g,n-1}_{(\infty)}}\right)}, \quad \forall g \in \{1, \dots, G\}. \tag{20}$$

Another interesting option is to use the diffusion approximation to build a two-level propagation scheme. The idea goes as follows: the computation of the solution with the diffusion approximation can be obtained very quickly in comparison with the transport solution. For a given time t_{n+1}, we can therefore compute the solution at t_{n+1} coming from the diffusion (denoted here as $\tilde{\phi}^{g,n+1}_{(\infty)}$) and use it as a starting guess to compute $\phi^{g,n+1}$:

$$\phi^{g,n+1}_{(M=0,N=0)} = \tilde{\phi}^{g,n+1}_{(\infty)}, \quad \forall g \in \{1, \dots, G\}.. \tag{21}$$

As will be illustrated in the numerical results, this is a bad choice whose main problem is that the diffusion solution has a different orbit than the transport one, hence the degraded computing times (the transport solver needs to correct the orbit and converge to the transport solution). However, since the diffusion approximation seems to present the good trend, one can conjecture that

$$\frac{\phi^{g,n+1} - \phi^{g,n}}{t_{n+1} - t_n} \approx \frac{\tilde{\phi}^{g,n+1} - \tilde{\phi}^{g,n}}{t_{n+1} - t_n}.$$

In this case, we can try:

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} + \tilde{\phi}^{g,n+1}_{(\infty)} - \tilde{\phi}^{g,n}_{(\infty)}, \quad \forall g \in \{1, \dots, G\}. \tag{22}$$

The numerical results will show that this is a good starting guess. Furthermore, if we suppose that the trend is exponential as point kinetics suggests, an interesting initial guess could be:

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} \exp^{\frac{t_{n+1}-t_n}{t_n-t_{n-1}} \ln\left(\frac{\tilde{\phi}^{g,n+1}_{(\infty)}}{\tilde{\phi}^{g,n}_{(\infty)}}\right)}, \quad \forall g \in \{1, \dots, G\}. \tag{23}$$

Starting guess	Formula
A (traditional)	$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)}$
B (linear extrapolation)	$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} + \frac{t_{n+1} - t_n}{t_n - t_{n-1}} \left(\phi^{g,n}_{(\infty)} - \phi^{g,n-1}_{(\infty)}\right)$
C (exponential extrapolation)	$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} \exp^{\frac{t_{n+1}-t_n}{t_n-t_{n-1}} \ln\left(\frac{\phi^{g,n}_{(\infty)}}{\phi^{g,n-1}_{(\infty)}}\right)}$
D (plain multilevel)	$\phi^{g,n+1}_{(M=0,N=0)} = \tilde{\phi}^{g,n+1}_{(\infty)}$
E (multilevel linear)	$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} + \tilde{\phi}^{g,n+1}_{(\infty)} - \tilde{\phi}^{g,n}_{(\infty)}$
F (multilevel exponential)	$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} \exp^{\frac{t_{n+1}-t_n}{t_n-t_{n-1}} \ln\left(\frac{\tilde{\phi}^{g,n+1}_{(\infty)}}{\tilde{\phi}^{g,n}_{(\infty)}}\right)}$

Table 4: List of the explored starting guesses.

We summarize all the options in table 4. Their perfomances have been tested in "case A" with a constant time-step of 5/3 s. In figure 1 we plot the computing times per time step as well as the cumulative ones. We also plot M_{outer} and $M_{outer}N_{inner}N_{DSA}$. From these figures, it seems thus clear that the use of a multilevel scheme outperforms the rest of the approaches provided that we do a linear or exponential extrapolation. The computing times are reduced by a factor of about 3 with this strategy. Options B and C provide a more moderate gain compared to the traditional case A. As it can be observed from the figures, the speed up comes from the reduction of the number of outer iterations M_{outer}, which results in a dramatic reduction of the total number of iterations $M_{outer}N_{inner}N_{DSA}$.

Once these sequential acceleration techniques have been implemented, very few gain in the speed-up can be obtained by adding other sequential techniques to the code and, if additional speed-ups are required, it is necessary to explore efficient parallelization techniques. We therefore devote the rest of the paper to the analysis of the parallelization of the angular and the temporal variables.

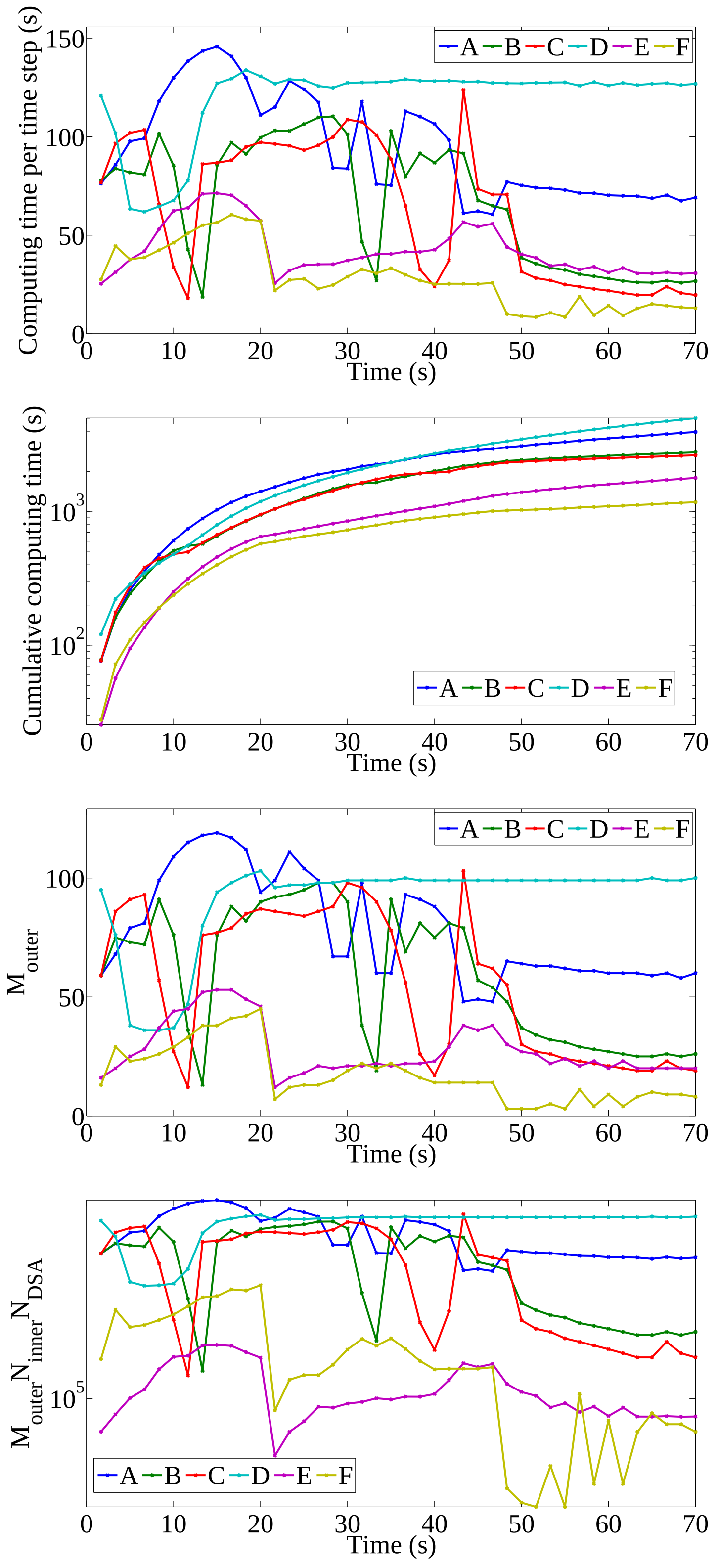


Figure 1: Performances of the initial guesses.

6. Parallelization of the angular variable

The numerical scheme outlined in algorithm 1 shows that, for a given energy group g and a given inner iteration m, the set of angular fluxes $\{\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)\}_{d=1}^D$ is computed by a loop over the S_n directions (lines 7 to 10 of algorithm 1). For each $\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)$, the spatial problem 12 is solved and it is decoupled from the spatial problem of the other unknowns $\psi(\boldsymbol{r}, \boldsymbol{\omega}_{d'})$ $(d' \neq d)$. The loop of lines 7 to 10 is therefore an embarrassingly parallel task that can be performed concurrently on several processors by uniformly distributing the set of angular fluxes to be treated among the different processors.

From an implementation point of view, the distribution of the tasks is performed in MINARET in a master-slave fashion with the MPI library. This implementation strategy has the important advantage of alleviating the memory storage per processor in comparison with a sequential implementation because each processor stores only the angular fluxes $\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)$ of its assigned directions (and the moments of the flux are only stored by the master). Thanks to this fact, MINARET can address time-depend calculations involving a large number of directions and leading to HPC problems (we refer to [7] for similar results on this topic but for the steady state case).

At the end of each inner iteration, the master gathers all the angular fluxes $\{\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)\}_{d=1}^D$. After computing the scalar flux $\phi(\boldsymbol{r})$, it performs the diffusion synthetic acceleration.

Table 5 and figure 2a show the numerical performances of this implementation in a strong scaling test regarding the angular variable: the test case A has been performed for a fixed number of directions $D = 24$ with an increasing number of processors N that treat the loop over the directions. There is a trade-off between:

- the number of directions assigned to each processor
- the spatial complexity for the calculation of an unknown $\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)$ (resolution of problem 12)
- the computing time required to perform the DSA step (that is run sequentially)

For a reduced number of processors, the algorithm has excellent scalability properties ($N \leq 8$). The behavior is degraded for larger values of N because the amount of work assigned to each processor decreases. The time to perform the loop on the angular directions is therefore reduced whereas the time to do the DSA remains constant because it is not parallelized: the DSA becomes a bottleneck. This issue could be overcome by its parallelization with domain decomposition methods or multigrid techniques like in the works of [25] and [26] respectively.

As a consequence of all this factors, in order not to lose much efficiency, there is a minimum number of directions $\boldsymbol{\omega}_d$ that need to be treated by each processor. In the present case, the most reasonable choice according to this criterion seems to be to assign $N/D = 4$ directions per processor (see table 5).

It is also desirable that the number of processors N is a divisor of the total number of directions D in order to have an uniform distribution of the tasks between processors. This is indeed a source of inefficiency as illustrated in table 5 for the case $N = 10$ (some processors will treat 3 directions and others only 2).

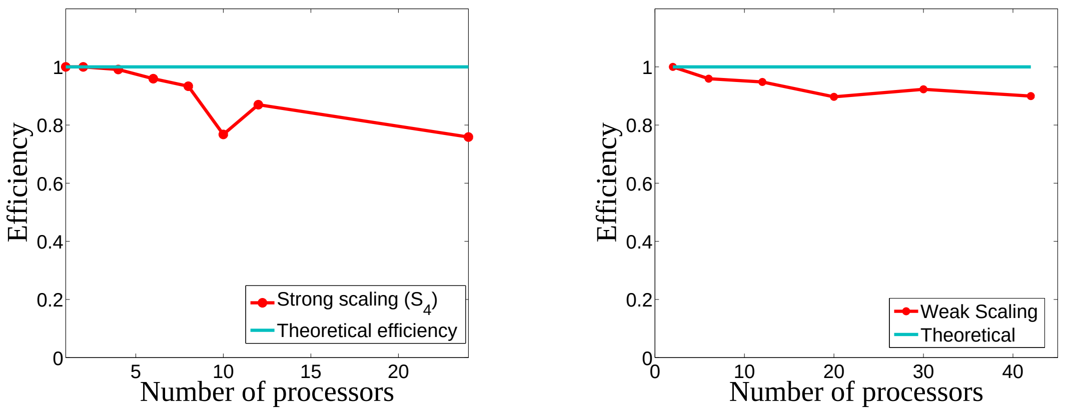
With the "optimal" number of $N/D = 4$ directions per processor being fixed, a weak scaling test has been performed where the angular S_n approximation is increased (D increases) by incrementing the number N of processors. The results are summarized in table 6 and figure 2b where it can be noticed that the efficiency is almost not degraded as N increases. This is a numerical proof that shows that, provided that we have enough processors at our disposal, extremely precise S_n approximations can be performed without increasing the total computing time in comparison with lower S_n approximations.

D	24	24	24	24	24	24	24	24
N	1	2	4	6	8	10	12	24
D/N_{proc}	24	12	6	4	3	2 or 3	2	1
Efficiency	1	1	0.99	0.96	0.933	0.77	0.87	0.76

Table 5: Efficiency in the strong scaling test for the angular variable (case A)

S_n approx	2	4	6	8	10	12
N_{dir}	8	24	48	80	120	168
N_{proc}	2	6	12	20	30	42
N_{dir}/proc	4	4	4	4	4	4
Efficiency	1	0.96	0.95	0.90	0.922	0.90

Table 6: Efficiency in the weak scaling test for the angular variable (case A).



(a) Strong scaling. **(b)** Weak scaling.

Figure 2: Efficiency in the parallelization of the angular variable (case A)

7. Parallelization of the time variable

As has been outlined in the previous section, an efficient technique for the acceleration of the resolution of the time dependent neutron transport equation is the parallelization of the angular variable. Its performances seem to be only slightly degraded in weak scaling cases, which implies that arbitrary high S_n orders can be addressed in a reasonable time. The most usual case, however, is to fix the S_n angular accuracy in coherence with the accuracy fixed for other variables (like, e.g., the spatial variable in which the accuracy is given by the finite element polynomial approximation). For this reason, the number of allocated processors to efficiently accelerate a given calculation is upper bounded and, if we have more processors at our disposal and wish additional speed-ups, the parallelization of other variables needs to be addressed. In this context, it is interesting to consider the extra speed-up that can bring the parallelization of the temporal variable. In the present case, this task has been adressed by a domain decomposition technique: the parareal in time algorithm. This section is organized as follows: after a brief recall of the basics of the parareal in time algorithm, an extension of the traditional theoretical speed-up formula will be proposed in order to properly take into account our particular case in which parareal is coupled with other iterative techniques at each time propagation. Finally, an analysis of the performances of the method for the resolution of transport transients with MINARET will be presented. The implemented results consider the parallelization of the time without coupling it with the parallelization of the angle. They are nevertheless representative enough of the accelerations that could be obtained in addition to the ones provided by the angular parallelization.

7.1. The parareal in time algorithm

The unsteady problem (1) can be written in a more compact form:

$$\frac{\partial y}{\partial t} + \mathcal{A}(t; y) = 0 \, , t \in [0, T]; \tag{24}$$

it is complemented with initial conditions: $y(t = 0) = y_0$.

We assume that we have two propagators to solve (24): a fine one $\mathcal{F}_{\tau_0}^{\tau_1}(y(\tau_0))$ that, starting from time $\tau_0 \in [0, T]$ with the value $y(\tau_0)$, computes an approximation of the solution of (24) at time $\tau_1 \in [\tau_0, T]$ accurately but slowly, and a coarse one $\mathcal{G}_{\tau_0}^{\tau_1}(y_0)$ that computes an other approximation quickly but not so accurately (and not accurately enough). The fine propagator $\mathcal{F}$ can, e. g., perform the propagation of the phenomenon from τ_0 to τ_1 with small time steps δt with very accurate physics described by $\mathcal{A}$. On the other hand, the coarse approximation $\mathcal{G}$ 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 $\Delta T \gg \delta t$ or by treating ”reduced physics” (i.e. by simplifying $\mathcal{A}$ into a less computer resources demanding operator).

In addition to these two propagators $\mathcal{F}$ and $\mathcal{G}$, 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 parareal in time algorithm applied to (24) is an iterative technique where, at each iteration k, the value $y(T_n)$ is approximated by Y_n^k with an accuracy that tends to the one achieved by the fine solver when k increases. Y_n^k is obtained by the recurrence relation:

$$Y_{n+1}^{k+1} = \mathcal{G}_{T_n}^{T_{n+1}}(Y_n^{k+1}) + \mathcal{F}_{T_n}^{T_{n+1}}(Y_n^k) - \mathcal{G}_{T_n}^{T_{n+1}}(Y_n^k), \ n = 0, ..., N-1 \tag{25}$$

starting from $Y_{n+1}^0 = \mathcal{G}_{T_n}^{T_{n+1}}(Y_n^0)$.

From formula (25), it can first of all be seen by recursion that the method is exact after enough iterations. Indeed, for any $n > 0$, $Y_n^n = \mathcal{F}_0^{T_n}(y_0)$. However, convergence of Y_n^k to $\mathcal{F}_0^{T_n}(y_0)$ goes much faster than this as will be illustrated in our numerical example. Second, by the recurrence formula (25), the parareal in time algorithm can be cast in the category of predictor corrector algorithms, where the predictor is $\mathcal{G}_{T_n}^{T_{n+1}}(Y_n^{k+1})$ while the corrector is $\mathcal{F}_{T_n}^{T_{n+1}}(Y_n^k) - \mathcal{G}_{T_n}^{T_{n+1}}(Y_n^k)$ (we refer to [27] for a detailed discussion about the several possible interpretations of the parareal method).

7.2. Algorithmics and theoretical speed-up

While the main results about the convergence properties of the method were studied in depth a decade ago (see, e.g. [8] [9] [10]), more recent efforts ([28] [13] [29] [30]) focus on the algorithmics to implement it in order to improve the speed-up provided by the original algorithm suggested in [8]. It consisted on a master-slave type of implementation where the master carried out the coarse propagation in the whole time interval $[0; T]$, each slave being in charge of the fine propagations over its assigned time slice and sending $\mathcal{F}_{T_n}^{T_{n+1}}(Y_n^k)$ to the master so that the master computed the parareal corrections of equation 25, $\forall n$. 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.

A remedy to both drawbacks is a distributed algorithm that was suggested in [13]: 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. A graphical description of the master-slave and distributed algorithms is shown in figures 3a and 3b in the ideal case where each processor is identical and the communication time is negligible.

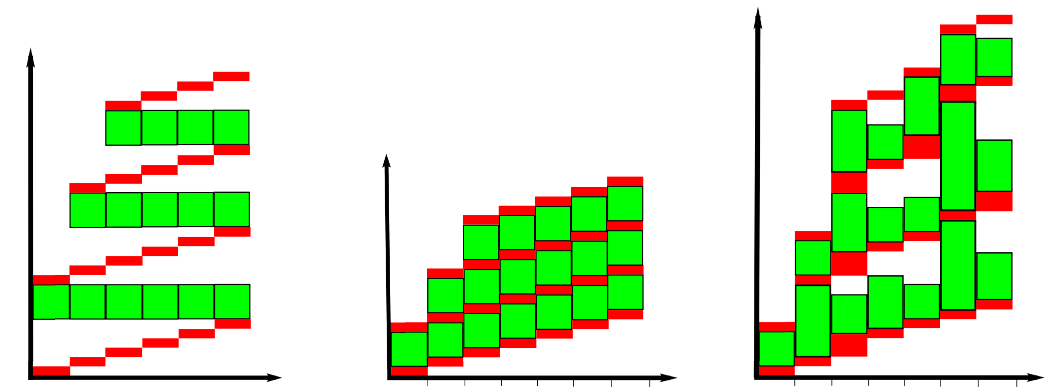
It is easy to realize that the distributed implementation does not change the number of iterations in order the parareal algorithm to converge but it provides better speed-ups than the original master-slave version (see formula (26) below). This is the reason why the distributed algorithm has been implemented in this study.

To the best of the authors knowledge, the theoretical analysis for the maximum attainable speed-up provided by the parareal algorithm in different types of algorithms has always been made under the assumptions that the computational cost of the fine and the coarse solvers is identical from one processor to another and that the communication time is negligible. Under these hypothesis, the maximum speed-up for the master-slave (S_{MS}) and distributed algorithms (S_D) are respectively (see [13]):

$$S_{MS} = \frac{T_{seq}}{T_{para,MS}} = \frac{N}{Nr(1+k^*) + k^*} \quad ; \quad S_D = \frac{T_{seq}}{T_{para,D}} = \frac{N}{Nr + k^*(1+r)}, \tag{26}$$

where k^* is the number of parareal iterations needed in order to converge and $r = \dfrac{T_\mathcal{G}}{T_\mathcal{F}}$, $T_\mathcal{G}$ and $T_\mathcal{F}$ are the computational costs of the coarse and fine propagators per processor. Note that $S_D > S_{MS}$ for any k^*, r and $N > 1$.

In the case that the fine and the coarse propagators solve each time step with an iterative numerical method, it is possible that the cost of the fine and the coarse solvers dramatically vary form one processor to another depending on



(a) A balanced master-slave scheme.

(b) A balanced distributed scheme.

(c) A distributed scheme with task imbalance.

Figure 3: Two different algorithms to implement the parareal in time method (a-b) and an illustration of the imbalance in the tasks (c) observed when the parareal algorithm is coupled with other iterative schemes for each time step propagation (in the example, $k^* = 3$ and $N = 7$ processors).

the numerical complexity of the events that take place in each time slice ΔT (and this complexity cannot be predicted a priori). Figure 3c illustrates this fact. Formulae (26) need therefore to be extended to the broader case in which the computational costs $T_{\mathcal{G}} = T_{\mathcal{G}}(k, p)$ and $T_{\mathcal{F}} = T_{\mathcal{F}}(k, p)$ depend on the processor p and the parareal iteration k. It is easy to show that a more adequate formula for the speed-up in this case is:

$$\begin{cases} \tilde{S}_D &= \dfrac{T_{seq}}{\tilde{T}_{para,D}} \\ &= \dfrac{T_{seq}}{\sum\limits_{p=0}^{N-1} T_{\mathcal{G}}(0,p) + \sum\limits_{k=1}^{k^*} \max\limits_{p\in\{0,\dots,N-1\}} \left(T_{\mathcal{G}}(k,p) + T_{\mathcal{F}}(k,p)\right)}, \\ \tilde{S}_{MS} &= \dfrac{T_{seq}}{\tilde{T}_{para,MS}} \\ &= \dfrac{T_{seq}}{\sum\limits_{p=0}^{N-1} T_{\mathcal{G}}(0,p) + \sum\limits_{k=1}^{k^*} \left(\sum\limits_{p=0}^{N-1} T_{\mathcal{G}}(k,p) + \max\limits_{p\in\{0,\dots,N-1\}} T_{\mathcal{F}}(k,p)\right)}, \end{cases} \tag{27}$$

where the communication time between processors has been neglected. Note that in the generalized formulae (27), we also find that $\tilde{S}_D > \tilde{S}_{MS}$ since we have $\tilde{T}_{para,MS} - \tilde{T}_{para,D} \geq \sum\limits_{k=1}^{k^*} \sum\limits_{p\neq p^*(k)} T_{\mathcal{G}}(k,p) > 0$, where $T_{\mathcal{G}}(k,p^*) := \max\limits_{p\in\{0,\dots,N-1\}} T_{\mathcal{G}}(k,p)$.

Remark 7.1. *Slightly better speed-ups than the ones provided by the distributed algorithm can be achieved with the event-based parareal algorithm suggested by [30] which, in turn, represents a major improvement from the processor utilization point of view. The algorithm exploits the fact that the coarse and fine propagations can be considered as a collection of tasks that can be treated by a processor as soon as their initial conditions are fulfilled. Once the task is performed, the processor treats the following task, if any, leading to an optimization of the processor utilization. However, since the present work focuses essentially on the feasibility and attainable speed-ups of parareal applied to equation (4), the distributed algorithm has been selected for its simpler implementation.*

7.3. Numerical application

The parareal algorithm has been applied to the resolution of the test cases A and B. An S_4 transport propagator has been used as the fine solver whereas two coarse solvers have been tried out:

- an S_4 transport propagator (the only difference with the fine solver is the size of the time steps used: δt for $\mathcal{F}$ and $\Delta t = T_{n+1} - T_n > \delta t$ for $\mathcal{G}$),
- a diffusion propagator.

All calculations have been evaluated with a convergence test (for the parareal iterations) in which the tolerance η has been fixed to the precision of the numerical scheme (i. e. $\eta \sim 10^{-3}$). The tolerance in the convergence for the outer and inner iterations has been fixed to $\varepsilon_{outer} = 10^{-5}$ and $\varepsilon_{inner} = 10^{-1}$. With this thresholds, parareal convergence has been achieved after only $k^* = 2$, 3 or at most 4 iterations of the parareal in time algorithm.

In the following subsections, after giving a numerical proof of the convergence of the parareal algorithm in our case of study, some results about measured speed-ups will be presented.

7.3.1. A numerical proof of the convergence

Figure 4 illustrates that parareal effectively converges in the particular case where both propagators use S_4 transport to solve test case A. The fine solver has a time step $\delta t = 5/3$ *s*. and the coarse one $\Delta t = 4\delta t$. The points represent the errors

$$e^k(T_n) = \frac{||\Phi_n^k - \mathcal{F}_0^{T_n}(\Phi_0)||_{L^2}}{||\mathcal{F}_0^{T_n}(\Phi_0)||_{L^2}}, \ \forall n \in \{0, 1, ...N\}, \ k \in \{0, 1, 2\} \tag{28}$$

between the parareal scalar flux Φ_n^k and the sequential fine solution $\mathcal{F}_0^{T_n}(\Phi_0)$.

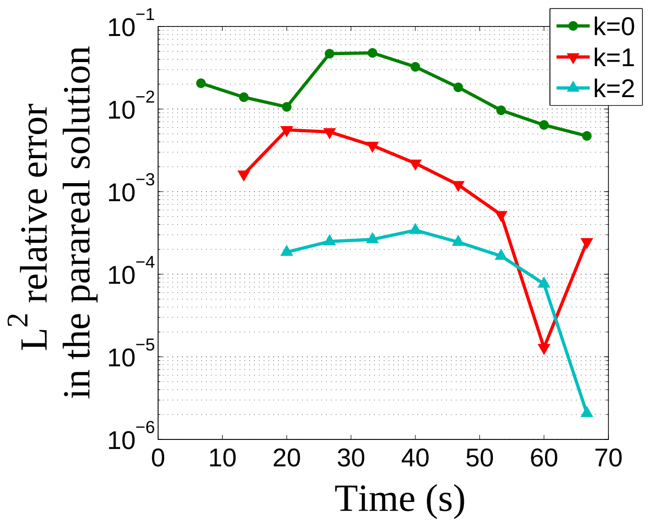


Figure 4: An example of the numerical convergence of the parareal algorithm in our neutron transport case.

7.3.2. Speed-up performances

In the following strong and weak scaling tests, the fine solver has a fixed time step of $\delta t = 1/12$ *s*.

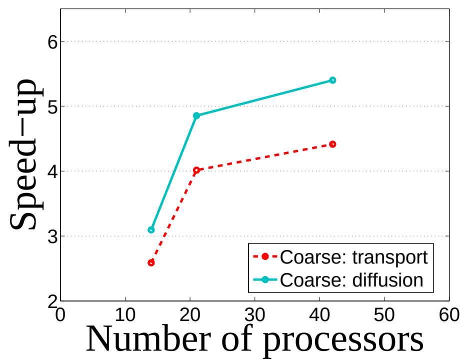
Strong scaling results:. For the strong scaling analysis, the test case A has been solved with MINARET on an increasing number N of processors. The size of each sub-interval $T_{n+1} - T_n$ is constant for all n and equal to the time step of the coarse solver Δt. In order to increase the number of processors solving the transient in the fixed time interval [0; 70 *s*.], the coarse time step has been reduced from $\Delta t = 60\delta t$ to $\Delta t = 20\delta t$.

The measured speed-ups are plotted in figure 5a and are in perfect agreement with the theoretical formula $\tilde{S}_D$. It can therefore be infered that the communication time between processors is negligible in our case and the obtained results are optimal (regarding the fixed convergence thresholds $\eta = 10^{-3}$, $\varepsilon_{outer} = 10^{-5}$ and $\varepsilon_{inner} = 10^{-1}$). Another interesting element to note is that one gets better speed-ups with a coarse diffusion propagator. This result seems reasonable because diffusion propagations are faster than transport ones.

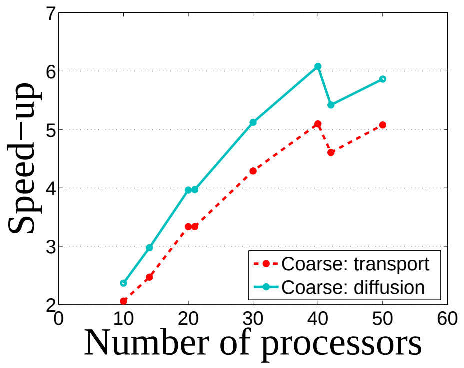
We also observe that for a reduced number of processors, the speed-up increases linearly until it reaches a plateau for more than 21 processors. This is due to the fact that, for large values of N, the size of the sub-intervals $\Delta t = T_{n+1} - T_n$ decreases. As a result, the size of the problem addressed by each processor decreases and we reach a point in which the addition of more processors does not improve any longer the performances.

Weak scaling results. For this alternative evaluation of the scaling, we will focus on the test case B. We now consider the case in which the time step of the coarse solver Δt is fixed to $60\delta t$ and the transient has a variable length $T(N) = N\Delta t$ (i.e. the size of the problem linearly increases with the number N of processors). As an example, for $N = 14$, transient B will be solved in the time interval $[0, 70\ s.]$, whereas when $N = 42$ the time interval will be $[0; 210\ s.]$.

The measured speed-ups are plotted in figure 5b and, like in the strong scaling case, they are in perfect agreement with the theoretical formula $\tilde{S}_D$. The most important result here is that the distributed algorithm can effectively speed-up long time calculations: the global trend for the speed-up is to increase linearly with the number of processors. The discontinuity in the trend observed between a number of processors $N = 41$ and 42 comes from the fact that, due to the increasing size of the interval $[0, T(N)]$, the number of parareal iterations k^* raises from 3 to 4 at this stage.



(a) Strong scaling results (test case A).



(b) Weak scaling results (test case B).

Figure 5: Parareal scaling results.

7.4. A parareal in space and energy algorithm?

In this part, we will discuss about the possibility to use the parareal algorithm to parallelize the space and energy variables.

The method was originally suggested for the time variable but it is quite straightforward to realize that the variable t of equation 24 is "dummy" in the sense that it could also represent a spatial variable: parareal provides also a method to parallelize 1D advection equations. The extension to 3D spatial advective problems like the current one (see equation 12) seems therefore theoretically possible: for each angular unknown flux $\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)$, the spatial mesh could be divided in a manner that is coherent with the direction of propagation $\boldsymbol{\omega}_d$. Each part of the mesh could be assigned to a different processor that would perform the fine propagation (i.e. the transport propagation of $\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)$). The coarse solver could consist in a diffusion approximation of the original equation 1. This idea is, however, not the first attempt to parallelize hyperbolic spatial problems. There exists indeed several references on this topic and we refer to, e.g., [16] [17] for interesting developments on this issue.

If we now observe the multigroup problem (equation 4) or the outline of the resolution of a transient in algorithm 1, it can be seen that, for a given time step, the energy groups are solved through a loop that could in turn be also parallelized by the parareal in time algorithm: the coarse solver would propagate a reduced number of energy groups while the fine solver would propagate the problem for all the energy groups.

Conclusion

The developments presented in this paper have shown on a first stage how the MINARET solver has been extended to address time dependent problems. Such computations usually involve extremely large numbers of unknowns and acceleration techniques are required in order to run the calculations in a reasonable time. To address this issue, several sequential and parallel acceleration methods have been explored:

The two sequential accelerations included in MINARET are classical (the Chebyshev extrapolation and the diffusion synthetic acceleration) but it has been shown by a concrete example that they are essential in making the outer and inner iterative schemes converge in a reasonable time (the computing times are reduced by a factor of about 100 from the initial one). It has further been noted that one can still reduce the computing time by the use of a multilevel scheme that involves diffusion propagations and an exponential extrapolation formula.

Regarding parallel accelerations, it has first been explained how the parallelization of the angular directions can efficiently speed-up calculations. Its excellent scalability for a reduced number N of processors is degraded as N grows. This is due to the sequential computation of the DSA: since its computing time remains constant with N, its contribution to the global computing time becomes more and more significant as N raises because the time to perform the loop on the angular directions decreases with N. This problem could be solved by parallelizing the DSA by domain decomposition techniques.

Provided that we have enough processors at our disposal, the parallelization in the angular directions could be coupled with the parallelization of the time variable by the parareal in time algorithm. The efficiency of this method is much lower than the performances provided by other parallelization techniques, but this is due to the difficult task of parallelizing a variable that is sequential by nature. It has nevertheless been illustrated that the method can provide additional speed-ups for the computation of –long time– neutron transport transients. Two types of coarse solvers have been explored: one that does not degrade the original S_n transport model and another that uses the diffusion approximation. The results obtained with the diffusion coarse solver are slightly higher. This is due to the fact that diffusion propagations are performed much faster than the transport ones and because the number of required parareal iterations is not degraded in comparison with the other case.

A loss in the performances of the parareal algorithm has been detected because it has been coupled with a generalized Gauss-Seidel iterative techniques in the propagation of each time step. In the same spirit as the works of Maday and Turinici in [31] or Minion in [28], [29] where the parareal in time algorithm has already been coupled with spatial domain decomposition and spectral deferred corrections, a way to improve the present results could consist in enhancing the coupling between the parareal in time algorithm and the outer iterations of the multigroup problem 4.

Acknowledgments

The authors would like to thank A.M. Baudron for fruitful discussions.

This work was supported in part by the joint research program MANON between CEA-Saclay and University Pierre et Marie Curie-Paris 6. The authors are also indebted to AREVA-NP and EDF for their financial support.

[1] A. Pautz, A. Birkhofer, DORT-TD: A transient neutron transport code with fully implicit time integration, Nucl. Sci. Eng. 145 (2003) 299–319.

[2] A. Saubert, A. Sureda, J. Bader, J. Lapins, M. Buck, E. Laurien, The 3-D time-dependent transport code TORT-TD and its coupling with the 3D thermal-hydraulic code ATTICA3D for HTGR applications, Nuclear Engineering and Design 251 (2012) 173–180.

[3] S. Chauvet, A. Nachaoui, A.-M. Baudron, J.-J. Lautard, A multi-scale approach for the neutronic kinetics equation using the mixed dual solver minos, in: Joint International Conference on Mathematics and Computation and Supercomputing in Nuclear Applications, 2007.

[4] E. Girardi, P. Guérin, S. Dulla, P. Ravetto, Comparison of direct and quasi-static methods for neutron kinetic calculations with the EDF R&D COCAGNE code, in: PHYSOR, Advances in Reactor Physics, 2012.

[5] P. Reuss, Précis de neutronique, EDP Sciences, Collection Génie Atomique, 2003.

[6] H. Golfier, R. Lenain, C. Calvin, J.-J. Lautard, A.-M. Baudron, P. Fougeras, P. Magat, E. Martinolli, Y. Dutheillet, APOLLO3: a common project of CEA, AREVA and EDF for the development of a new deterministic multi-purpose code for core physics analysis, in: International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, 2009.

[7] E. Jamelot, J. Dubois, J.-J. Lautard, C. Calvin, A.-M. Baudron, High performance 3D neutron transport on petascale and hybrid architectures within APOLLO3 code, in: International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, 2011.

[8] J. Lions, Y. Maday, G. Turinici, Résolution d'EDP par un schéma en temps pararéel, C. R. Acad. Sci. ParisT. 332, Série I, p. 661-668.
[9] L. Baffico, S. Bernard, Y. Maday, G. Turinici, G. Zérah, Parallel-in-time molecular-dynamics simulations, Phys. Rev. E 66.
[10] G. Bal, Y. Maday, A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put, Recent developments in domain decomposition methods 23 (2002) 189–202.
[11] A.-M. Baudron, J.-J. Lautard, K. Riahi, Y. Maday, J. Salomon, Parareal in time 3D numerical solver for the LWR Benchmark neutron diffusion transient model, Submitted.
[12] A.-M. Baudron, J.-J. Lautard, Y. Maday, O. Mula, The parareal in time algorithm applied to the kinetic neutron diffusion equation, in: 21st International Conference on Domain Decomposition Methods, 2012.
[13] E. Aubanel, Scheduling of tasks in the parareal algorithm, Parallel Computing 37 (2011) 172–182.
[14] R. Slaybaugh, T. Evans, G. Davidson, P. Wilson, Rayleigh quotient iteration in 3d, deterministic neutron transport, in: PHYSOR, Advances in Reactor Physics, 2012.
[15] E. Jamelot, A.-M. Baudron, J.-J. Lautard, Domain Decomposition for the SP_N Solver MINOS, Transp. Theory Stat. Phys. 41 (7) (2012) 495–512.
[16] L. Gastaldi, On a domain decomposition for the transport equation: theory and finite element approximation, IMA J. Numer. Anal. 14 (1) (1994) 111–135.
[17] F. Golse, S. Jin, C. D. Levermore, A domain decomposition analysis for a two-scale linear transport problem, ESAIM, Math. Model. Numer. Anal. 37 (2003) 869–892.
[18] R. Dautray, J.-L. Lions, Analyse mathématique et calcul numérique, Masson, 1984.
[19] J.-Y. Moller, Éléments finis courbes et accélération pour le transport de neutrons, Ph.D. thesis, Université Henri Poincaré (2012).
[20] O. Mula, Some contributions towards the parallel simulation of time dependent neutron transport and the integration of observed data in real time, Ph.D. thesis, Paris VI (2014).
[21] J.-Y. Moller, J.-J. Lautard, Minaret, a deterministic neutron transport solver for nuclear core calculations, in: International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, 2011.
[22] S. Langenbuch, W. Maurer, W. Werner, Coarse-mesh flux expansion method for the analysis of space-time effects in large light water reactor cores, Nucl. Sci. Eng. 63 (1977) 437–456.
[23] R. S. Varga, Matrix iterative analysis, Springer series in computational mathematics, Springer Verlag, Berlin, Heidelberg, Paris, 2000.
[24] M. L. Adams, E. W. Larsen, Fast iterative methods for discrete-ordinates particle transport calculations, Progress in Nuclear Energy 40(1) (2002) 3 – 159.
[25] P. F. Antonietti, B. Ayuso, Schwarz domain decomposition preconditioners for discontinuous galerkin approximations of elliptic problems: non-overlapping case, ESAIM, Math. Model. Numer. Anal. 41 (2007) 21–54.
[26] A. Napov, Y. Notay, An algebraic multigrid method with guaranteed convergence rate, SIAM J. Sci. Comput. 34 (2) (2012) A1079–A1109.
[27] M. Gander, S. Vandewalle, Analysis of the parareal time-parallel time-integration method, SIAM J. Sci. Comput. 29 (2) (2007) 556–578.
[28] M. Minion, A hybrid parareal spectral deferred corrections method, Comm. App. Math. and Comp. Sci. 5 (2).
[29] M. Emmet, M. Minion, Toward an efficient parallel in time method for partial differential equations, Comm. App. Math. and Comp. Sci. 1 (1).
[30] L. Berry, W. Elwasif, J. Reynolds-Barredo, D. Samaddar, R. Sanchez, D. Newman, Event-based parareal: A data-flow based implementation of parareal, J. Comput. Phys. 231 (17) (2012) 5945 – 5954.
[31] Y. Maday, G. Turinici, The Parareal in Time Iterative Solver: a Further Direction to Parallel Implementation, in: Domain Decomposition Methods in Science and Engineering, Springer Berlin Heidelberg, 2005, pp. 441–448.