MINARET or the quest towards the use of time-dependent neutron transport solvers for nuclear core calculations on a regular basis

Jean-Jacques Lautard, Yvon Maday, Olga Mula

MINARET OR THE QUEST TOWARDS THE USE OF TIME-DEPENDENT NEUTRON TRANSPORT SOLVERS FOR NUCLEAR CORE CALCULATIONS ON A REGULAR BASIS

J.-J. LAUTARD[†‡], Y. MADAY[§¶‖‡], AND O. MULA[†‡§]

Abstract. The present paper deals with the resolution of the time-dependent neutron transport equation that is involved in the field of nuclear safety studies. Through the presentation of the newly implemented kinetic module in the MINARET solver [24] (developed at CEA in the framework of the APOLLO3® project), we aim first of all at presenting a brief and comprehensive overview of the most widespread resolution techniques employed nowadays in neutron transport industrial codes. Given that the main obstacle in the use of this type of accurate solver on a regular basis relies in the long computing times, MINARET has been used in the present work as a support to rigorously quantify the efficiency of the most common sequential and parallel acceleration techniques that are currently used in this field. An important part of the paper will be devoted to study the performances of an acceleration method that has never been considered before in the resolution of this equation, which is the parallelization of the time variable. In this regard, the parareal in time algorithm (a domain decomposition method for the time variable, [20]) has been implemented to explore its potentialities in this particular application.

Key words. MINARET; neutronics; neutron transport; parareal in time algorithm; time domain decomposition; core calculations

AMS subject classifications.

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. In the field of nuclear core calculations and, more particularly, regarding the resolution of the time-dependent neutron transport equation, this development of the computing capabilities has led to significant advances. Indeed, codes like PARTISN [3], TORT-TD [29] and the recently developped time-dependent module in MINARET (that will be presented in this paper) confirm this fact and have proven that the traditional diffusion [13], improved quasi-static [17] or point kinetics [28] traditional approximations can now start to be overcome. While this represents a significant progress for security assessment (approximations like diffusion are indeed not accurate enough in the study of fast transients or fast reactors), the long computing times of transport solvers in realistic 3D core geometries represent the main obstacle to face in order to definitely use this type of accurate solvers on a regular basis. In this framework, the present work is a contribution towards showing that these computing times can be very significantly reduced through acceleration methods that do not alter the main structure of the solvers. Towards this end, it will be explained how the traditional (sequential and parallel) acceleration methods employed in this field can be coupled efficiently and in a "non-intrusive" manner with other innovative acceleration methods that involve modern parallel architectures. For this reason, a non negligible part of the paper will be devoted to recall the current construction of these solvers as well as the classical acceleration strategies in order to better justify how new techniques can be added without modifying their main construction. A special effort has been done to provide tractable results about the performances of all the methods that will be discussed. These come from computations obtained with MINARET in a well-known benchmark test case of the literature [19]. The results will illustrate at the same time the work that has been done to optimize the implementation of MINARET.

Hence, the paper is organized as follows: we will start by recalling in section 2 the equation under consideration. Then, in section 3, we will present an overview of the most widespread discretization and resolution strategies that are employed in the field of nuclear safety calculations. This will be done through the particular example of the newly developed time-depend module of the discrete ordinates MINARET solver (we refer also to other codes like PARTISN and TORT-TD for similar approaches). Then, section 4 will be devoted to a simple validation study of MINARET in order to prove the validity of the results presented in the following sections.

We will after discuss the sequential and parallel acceleration techniques that are classically employed in these codes or that are currently the subject of active research and that do not require changes in the resolution scheme introduced in section 3. Section 5 will recall the two most traditional sequential accelerations (the Chebyshev extrapolation and the Diffusion Synthetic Acceleration). The combination of both strategies leads to speed-ups of a factor of around 100 as computations with MINARET will show. We will also explain how extra speed-ups can be obtained by the choice of an appropriate starting guess in the so-called outer iterations. In particular, the idea of involving diffusion computations in the starting guess had never been tested in the literature and seems to outperform the classical choices (a speed-up of a factor of about 3 has been noted).

The remaining part of the paper is devoted to parallel acceleration techniques. In section 6, we recall the usual method to parallelize the angular variable in discrete ordinates codes and illustrate its performances. The limitations of this strategy have motivated the exploration of other sources of concurrency in the litterature such as the parallelization of the spatial and energetic variables. Regarding the spatial variable, the so-called KBA algorithm was proposed in the late nineties (see [7]) and seems to be the most efficient strategy known so far. Its efficient implementation on unstructured meshes represents one of the main difficulties to face as the works of [27] show. Also, the parallelization of the energy variable seems to be a promising field to explore as outlined in [30] and some recent contributions such as [2] even aim at efficiently parallelizing jointly some of these three variables (energy, space and angle). However, among all these already existing approaches, the parallelization of the time variable seems to be missing and, for this reason, in section 7, a study about the additional speed-up that can be obtained with such a strategy will be presented. This task has been performed by a domain decomposition method called the parareal in time algorithm (see [20] [6] [8] for its theoretical foundations). The implementation is following some preliminary analysis made in [11] [10] where the strategy was successfully applied to the time-dependent neutron diffusion equation. We will propose an extension of the classical theoretical expected speed-up formulae from the literature (see, e.g. [5] for a summary on this topic) in order to take into account the fact that, in the present case, the parareal algorithm is being coupled with other iterative methods. The limitations regarding the maximum attainable speed-up will be discussed at this point. Despite this fact, a comparison between measured speed-up performances with MINARET and the theoretical optimal values will highlight the negligible impact of the communication time between processors and will illustrate the capacity of the method to accelerate long time transients. We will end up by outlining a strategy that is currently being developed to improve the performances of the parareal algorithm by coupling the method with the outer iterations in a more efficient manner. The results will be the published on a sequel of this paper.

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, $\boldsymbol{\omega} = \dfrac{\boldsymbol{v}}{|\boldsymbol{v}|} \in \mathbb{S}_2$ 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 $\mathcal{D} = \{(t, \boldsymbol{r}, \boldsymbol{\omega}, E) \in [0,T] \times \mathcal{R} \times \mathbb{S}_2 \times [E_{\min}, E_{\max}]\}$:

$$
\left\{
\begin{array}{l}
\dfrac{1}{|\boldsymbol{v}|}\partial_t \psi(t,\boldsymbol{r},\boldsymbol{\omega},E) + (L - H - F - Q)\psi(t,\boldsymbol{r},\boldsymbol{\omega},E) = 0 \\
\partial_t C_\ell(t,\boldsymbol{r}) = -\lambda_\ell C_\ell(t,\boldsymbol{r}) \\
\qquad\qquad + \displaystyle\int_{E'=E_{\min}}^{E_{\max}} \beta_\ell(t,\boldsymbol{r},E')(\nu\sigma_f)(t,\boldsymbol{r},E')\phi(t,\boldsymbol{r},E')dE',\ \forall \ell \in \{1,\dots,L\},
\end{array}
\right. \tag{2.1}
$$

where $\phi(t,\boldsymbol{r},E) = \displaystyle\int_{\mathbb{S}_2} \psi(t,\boldsymbol{r},\boldsymbol{\omega'},E)d\boldsymbol{\omega'}$ 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,
- $H\psi(t,\boldsymbol{r},\boldsymbol{\omega},E) = \displaystyle\int_{\mathbb{S}_2}\int_{E'=E_{\min}}^{E_{\max}} \sigma_s(t,\boldsymbol{r},\boldsymbol{\omega'} \to \boldsymbol{\omega}, E' \to E)\psi(t,\boldsymbol{r},\boldsymbol{\omega'},E')dE'd\boldsymbol{\omega'}$ 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,
- $Q\psi(t,\boldsymbol{r},\boldsymbol{\omega},E) = \displaystyle\sum_{\ell=1}^{L} \lambda_\ell \chi_{d,\ell}(t,\boldsymbol{r},E)C_\ell(t,\boldsymbol{r})$ is the delayed fission source. Note that this term depends on ψ because C_ℓ depends on it in vertue of the ODE in time that they satisfy.

In the enlisted terms:

- $\sigma_t(t,\boldsymbol{r},E)$ denotes the total cross-section,
- $\sigma_s(t,\boldsymbol{r},\boldsymbol{\omega'} \to \boldsymbol{\omega}, E' \to E)$ 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,
- $\chi_p(t,\boldsymbol{r},E)$ 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,
- $\beta(t,\boldsymbol{r},E) = \displaystyle\sum_{\ell=1}^{L} \beta_\ell(t,\boldsymbol{r},E)$.

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 $\partial\mathcal{R}_- := \{\boldsymbol{r} \in \partial\mathcal{R} \mid \boldsymbol{\omega}.\overrightarrow{n} < 0\}$ 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 [28] and [14] for physical and mathematical aspects of it and to [23] for numerical details about its computation in the MINARET solver.

REMARK 2.1 (A diffusion problem as an approximation to the transport equation (2.1)). *Under some given physical hypothesis (see, e.g. [28] and [14]), 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 (2.1), the existence of such surrogate approximation will be used in our case in some acceleration techniques.*

3. Classical discretization and implementation schemes in neutronics: an example through the MINARET solver. With the exception of some simple cases (see [28] for further references) where problem (2.1) can exactly be solved, the resolution of (2.1) needs to be numerically addressed and requires discretizations and approximations of the involved variables. We will present here the classical discretizations and numerical schemes that are used in the field of core calculations. This will be done through the particular case of the MINARET solver, whose kinetic module has never been presented before, but that can be taken as a representative example due to its similarities with other solvers like PARTISN or TORT-TD.

We start by discretizing the energy variable and deriving the multigroup version of equation (2.1). 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 $1 \leq g \leq G$, we denote by ψ^g the approximation of ψ in the subinterval $[E_g, E_{g-1}]$. Let us introduce a series of time steps $\delta t_n > 0$ and a serie of times t_n such that $t_0 = 0$, $t_N = T$ and $t_n - t_{n-1} = \delta t_n$. 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:

$$\begin{cases} \text{Find over } \mathcal{R} \times \mathbb{S}_2 \text{ the angular flux } \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) \text{ such that} \\ \left(L^g - H^g - \tilde{F}^g - \tilde{Q}^g\right)\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \tilde{S}^{g,n}(\boldsymbol{r},\boldsymbol{\omega}), \quad \forall g \in \{1,\ldots,G\}. \end{cases} \tag{3.1}$$

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.

- $L^g\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \left(\boldsymbol{\omega}.\nabla + \left(\sigma_t^{g,n+1}(\boldsymbol{r}) + \dfrac{1}{V^g\delta t_{n+1}}\right)\right)\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega})$
- $H^g\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \sum\limits_{g'=1}^{G} H^{g'\to g}\psi^{g',n+1}(\boldsymbol{r},\boldsymbol{\omega})$, with
  $H^{g'\to g}\psi^{g',n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \displaystyle\int_{\mathbb{S}_2} \sigma_s^{g'\to g,n+1}(\boldsymbol{r},\boldsymbol{\omega'}\to\boldsymbol{\omega})\psi^{g',n+1}(\boldsymbol{r},\boldsymbol{\omega'})d\boldsymbol{\omega'}.$
- $\tilde{F}^g\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \dfrac{\chi_p^{g,n+1}(\boldsymbol{r})}{4\pi}\sum\limits_{g'=1}^{G}\left(1-\beta^{g',n+1}(\boldsymbol{r})\right)(\nu\sigma_f)^{g',n+1}(\boldsymbol{r})\phi^{g',n+1}(\boldsymbol{r})$,
  where $\phi^{g,n+1}(\boldsymbol{r}) = \displaystyle\int_{\mathbb{S}_2}\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega})d\boldsymbol{\omega'},\ \forall g\in\{1,\dots,G\}.$
- $\tilde{Q}^g\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \sum\limits_{\ell=1}^{L}\lambda_\ell\chi_{d,\ell}^{g,n+1}(\boldsymbol{r})C_\ell^{n+1}(\boldsymbol{r})$

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 $\beta^{g,n+1}(\boldsymbol{r}) = \sum\limits_{\ell=1}^{L}\beta_\ell^{g,n+1}(\boldsymbol{r})$. The Euler backward scheme applied to the precursors' equation provides $C_\ell^{n+1}(\boldsymbol{r})$ for any $\ell\in\{1,\dots,L\}$:

$$\begin{aligned} C_\ell^{n+1}(\boldsymbol{r}) &= \frac{1}{1+\lambda_\ell\delta t_{n+1}}C_\ell^n(\boldsymbol{r}) \\ &\quad + \frac{\delta t_{n+1}}{1+\lambda_\ell\delta t_{n+1}}\sum_{g'=1}^{G}\beta_\ell^{g',n+1}(\boldsymbol{r})(\nu\sigma_f)^{g',n+1}(\boldsymbol{r})\phi^{g',n+1}(\boldsymbol{r}). \end{aligned} \tag{3.2}$$

The insertion of (3.2) into (3.1) finally yields the set of source problems:

$$\begin{cases} \text{Find over } \mathcal{R}\times\mathbb{S}_2 \text{ and } \forall g \text{ the angular flux } \psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) \text{ such that:} \\ \left(L^g - H^g - F^g\right)\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = S^{g,n}(\boldsymbol{r},\boldsymbol{\omega}), \end{cases} \tag{3.3}$$

where:

- $F^g\psi^{g,n+1}(\boldsymbol{r},\boldsymbol{\omega}) = \sum\limits_{g'=1}^{G}F^{g',g}\psi^{g',n+1}$ 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 (3.3) is iterativelysolved over the index g 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

$$\boldsymbol{\psi^{n+1}} = \begin{pmatrix} \psi^{1,n+1} \\ \psi^{2,n+1} \\ \vdots \\ \psi^{G,n+1} \end{pmatrix} \quad ; \quad \boldsymbol{\phi^{n+1}} = \begin{pmatrix} \phi^{1,n+1} \\ \phi^{2,n+1} \\ \vdots \\ \phi^{G,n+1} \end{pmatrix} \quad ; \quad \boldsymbol{S^{n}} = \begin{pmatrix} S^{1,n} \\ S^{2,n} \\ \vdots \\ S^{G,n} \end{pmatrix}, \tag{3.4}$$

then the scheme reads:

$$\begin{cases} M_{G,G}\boldsymbol{\psi^{n+1}_{(M+1)}} = N_{G,G}\boldsymbol{\psi^{n+1}_{(M)}} + \boldsymbol{S^{n}} \\ \boldsymbol{\psi^{n+1}_{(M=0)}} \text{ given,} \end{cases} \tag{3.5}$$

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

$$M_{G,G} = \begin{pmatrix} L^1 - H^{1\to 1} & 0 & \cdots & 0 \\ -H^{1\to 2} & L^2 - H^{2\to 2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -H^{1\to G} & -H^{2\to G} & \cdots & L^G - H^{G\to G} \end{pmatrix} \tag{3.6}$$

and

$$N_{G,G} = \begin{pmatrix} F^{1,1} & H^{2\to 1} + F^{2,1} & \cdots & H^{G\to 1} + F^{G,1} \\ F^{1,2} & F^{2,2} & \cdots & H^{G\to 2} + F^{G,2} \\ \vdots & \vdots & \ddots & \vdots \\ F^{1,G} & F^{2,G} & \cdots & F^{G,G} \end{pmatrix}. \tag{3.7}$$

System (3.5) is lower-triangular and is simply solved by forward substitution. 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 $g' \leq g$. This choice has been made for implementation purposes to minimize the number of computations: the fission terms are computed with the previous iterate so that their value remains fixed in the iterative resolution of (3.5) that is going to be introduced in (3.11).

It has been proven in chapter 1 of [25] (theorem 1.6.1) that this scheme converges for small enough time steps. In MINARET, the iterations are performed until the average residual error in the scalar flux

$$e^{n+1}_{outer}(M+1) := \left(\frac{\sum_{g=1}^{G} \| \phi^{g,n+1}_{(M+1)} - \phi^{g,n+1}_{(M)} \|^2_{L^2(\mathcal{R})}}{\sum_{g=1}^{G} \| \phi^{g,n+1}_{(M+1)} \|^2_{L^2(\mathcal{R})}} \right)^{1/2} \tag{3.8}$$

goes below a given convergence threshold ε_{outer} and $\boldsymbol{\psi^{n+1}_{(\infty)}}$ and $\boldsymbol{\phi^{n}_{(\infty)}}$ will denote the converged angular and scalar fluxes.

For $g = 1, \ldots, G$ and for a given iteration M, the problem to be solved in the forward substitutions to invert system (3.5) reads:

$$\begin{aligned} &(L^g - H^{g\to g})\psi^{g,n+1}_{(M+1)}(\boldsymbol{r}, \boldsymbol{\omega}) \\ &= \sum_{g'<g} H^{g'\to g}\psi^{g',n+1}_{(M+1)} + \sum_{g'>g} H^{g'\to g}\psi^{g',n+1}_{(M)} + \sum_{g'=1}^{G} F^{g',g}\psi^{g',n+1}_{(M)} + S^{g,n}(\boldsymbol{r}, \boldsymbol{\omega}), \end{aligned} \tag{3.9}$$

which is a monoenergetic equation of the form

$$\boldsymbol{\omega}.\nabla\psi(\boldsymbol{r},\boldsymbol{\omega}) + \sigma_t(\boldsymbol{r})\psi(\boldsymbol{r},\boldsymbol{\omega}) - \int_{\mathbb{S}_2} \sigma_s(\boldsymbol{r},\boldsymbol{\omega}' \to \boldsymbol{\omega})\psi(\boldsymbol{r},\boldsymbol{\omega}')d\boldsymbol{\omega}' = q(\boldsymbol{r},\boldsymbol{\omega}), \tag{3.10}$$

for all $(\boldsymbol{r},\boldsymbol{\omega}) \in \mathcal{R}\times\mathbb{S}_2$. The terms σ_t, σ_s, q must be understood as generic notations whose definition must be coherent with equation (3.9). Since equation (3.10) is integral in the angular variable and differential in space, a second numerical scheme is performed (called "inner or source iterations"). If $\psi^{g,n+1}_{(M+1,m)}$ is the approximation of $\psi^{g,n+1}_{(M+1)}$ at the $m-th$ inner iteration, then $\psi^{g,n+1}_{(M+1,m+1)}$ is the solution of:

$$L^g\psi^{g,n+1}_{(M+1,m+1)}(\boldsymbol{r},\boldsymbol{\omega}) = H^{g\to g}\psi^{g,n+1}_{(M+1,m)}(\boldsymbol{r},\boldsymbol{\omega}) + \tilde{S}(\boldsymbol{r},\boldsymbol{\omega}), \quad \forall g \in \{1,\dots,G\}, \tag{3.11}$$

with

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

It has been shown in [25] (section 1.6.1.2 of chapter 1) that this strategy is equivalent to a Richardson scheme. The iterations are performed until the relative residual

$$e^{g,n+1}_{inner}(m+1) := \frac{\|\phi^{g,n+1}_{(M,m+1)} - \phi^{g,n+1}_{(M,m)}\|_{L^2(\mathcal{R})}}{\|\phi^{g,n+1}_{(M,1)} - \phi^{g,n+1}_{(M,0)}\|_{L^2(\mathcal{R})}} \tag{3.12}$$

goes below a given convergence threshold ε_{inner}.

The angular discretization of equation (3.11) has been performed with the discrete ordinates of order n technique (S_n), i.e., problem (3.11) is solved for a discrete number D of directions $\{\boldsymbol{\omega}_d\}_{d=1}^D$. The scattering terms $H^{g'\mapsto g}$ are computed in practice by assuming that the scatering cross sections are isotropic, i.e., that

$$\sigma_s(\boldsymbol{r},\boldsymbol{\omega}' \mapsto \boldsymbol{\omega}) = \sigma_s(\boldsymbol{r},\boldsymbol{\omega}'.\boldsymbol{\omega}),$$

This assumption allows to make an expansion of the terms $H^{g'\mapsto g}$ in Legendre polynomials (the indexes g, n, M and m have momentarily been removed to ease the reading). The expansion is truncated at a given order L so that

$$\int_{\mathbb{S}_2} \sigma_s(\boldsymbol{r},\boldsymbol{\omega}' \to \boldsymbol{\omega})\psi(\boldsymbol{r},\boldsymbol{\omega}')d\boldsymbol{\omega}' \approx \sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi}\sigma_{s,\ell}(\boldsymbol{r}) \int_{\mathbb{S}_2} \psi(\boldsymbol{r},\boldsymbol{\omega}')P_\ell(\boldsymbol{\omega}.\boldsymbol{\omega}')d\boldsymbol{\omega}',$$

where P_ℓ is the Legendre polynomial of degree ℓ and $\sigma_{s,\ell} = 2\pi\int_{-1}^{1}\sigma_{s,\ell}(\boldsymbol{r},\mu)P_\ell(\mu)d\mu$. The use of the addition theorem $P_\ell(\boldsymbol{\omega}'.\boldsymbol{\omega}) = \frac{4\pi}{2\ell+1}\sum_{m=-\ell}^{\ell} Y_{\ell,m}(\boldsymbol{\omega}')Y_{\ell,m}(\boldsymbol{\omega})$, where $Y_{\ell,m}(\boldsymbol{\omega})$ is the spherical harmonic of index (ℓ,m), eventually leads to

$$\int_{\mathbb{S}_2} \sigma_s(\boldsymbol{r},\boldsymbol{\omega}' \to \boldsymbol{\omega})\psi(\boldsymbol{r},\boldsymbol{\omega}')d\boldsymbol{\omega}' \approx \sum_{\ell=0}^{L} \sigma_{s,\ell}(\boldsymbol{r}) \sum_{m=-\ell}^{\ell} \phi_{\ell,m}(\boldsymbol{r})Y_{\ell,m}(\boldsymbol{\omega}),$$

which is an expansion that depends only on the flux moments that are defined as

$$\phi_{\ell,m}(\boldsymbol{r}) = \int_{\mathbb{S}_2} \psi(\boldsymbol{r},\boldsymbol{\omega}')Y_{\ell,m}(\boldsymbol{\omega}')d\boldsymbol{\omega}'. \tag{3.13}$$

Note that $\phi_{\ell,m}(\boldsymbol{r}) = \phi(\boldsymbol{r})$ for $\ell = m = 0$. More importantly, the fact that only the flux moments are required to compute the scattering term implies that the evaluation of the term $N_{G,G}\boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M)}}$ in system (3.5) does not need the knowledge of the angular fluxes $\boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M)}}$, but only the knowledge of flux moments $\boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m\ (M)}}$ (in the case where $\ell = m = 0$, only $\boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M)}}$ is needed). We can therefore re-write system (3.5) as

$$\left\{ \begin{array}{l} M_{G,G}\boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M+1)}} = \widetilde{N}_{G,G}\boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M)}} + \boldsymbol{S^n} \\ \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M=0)}} \text{ given,} \end{array} \right. \tag{3.14}$$

where $\widetilde{N}_{G,G}$ denotes the same operator as $N_{G,G}$, but after having made the observation that the entries are flux moments.

The integrals in the angular variable of relation (3.13) are computed with a quadrature formula. In the case of MINARET, the level-symmetric quadrature rule has been used so that $D = n(n+2)$ in our case.

The treatment of the spatial variable is the point where the neutronic solvers most differ. In the case of MINARET, discontinuous Galerkin finite elements of arbitrary order have been implemented (however, the results presented in this paper involve only degree 1 polynomials – see remark 3.1). 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 [24] for further details on MINARET's mesh generator and also for a study made on the Jules Horowitz reactor, whose cylindrical geometry cannot be treated accurately on a cartesian grid and thus requires non cartesian grids. In addition to this, the finite element order can be locally adapted depending on the complexity of the phenomena involved in each region, which makes the solver be well-suited for the study of accidents.

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

REMARK 3.1. *The advantages of the hp adapativity of the MINARET solver will not be illustrated in the present work as our major goal is to provide tractable results about acceleration methods. We refer to [24] as a first overview of these capabilities and to future papers in which MINARET will be used for accidental studies.*

4. Definition of the numerical test cases and validation of the MINARET solver. 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") represents a rod withdrawal and corresponds to a well-known benchmark in the literature of neutronics called TWIGL (see [19]). The geometry of the core is three-dimensional and the domain is $\mathcal{R} = \{(x,y,z) \in \mathbb{R}^3 |\ 0 \leq x \leq 220 \text{ cm};\ 0 \leq y \leq 220 \text{ cm};\ 0 \leq z \leq 200 \text{ cm};\ \}$. 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 results provided by the benchmark do not correspond to measured values, but to diffusion calculations, which is the reason why this test case was originally proposed for the validation of diffusion solvers. However, to the best of the authors knowledge, a specific kinetic benchmark for transport solvers does not exist in the literature.

Algorithm 1 The iterative strategy implemented in MINARET

```
1: for t_n = ΔT to NΔT do
2:     While(not converge) do (generalized GS iterations – see equation (3.5))
3:     Update fission operator
4:     for g = 1 to G do
5:         Update scattering (except self-scattering)
6:         While(not converge) do (source iterations)
7:         for ω = ω_1 to ω_D do
8:             Update self-scattering
9:             Solve spatial problem (3.11) for ω
10:        end for
11:        Diffusion Synthetic Acceleration (see section 5)
12:        End While
13:    end for
14:    Chebyshev Extrapolation (see section 5)
15:    End While
16: end for
```

It will therefore not be possible to study through this example the performances of MINARET in situations where diffusion is no longer accurate enough. Despite this fact, the benchmark will allow to

- do a simple validation of the solver's kinetic module by comparing our calculations with a diffusion solver (called MINOS; see, e.g., [9]) and with the original results presented in [19],
- provide tractable performance results.

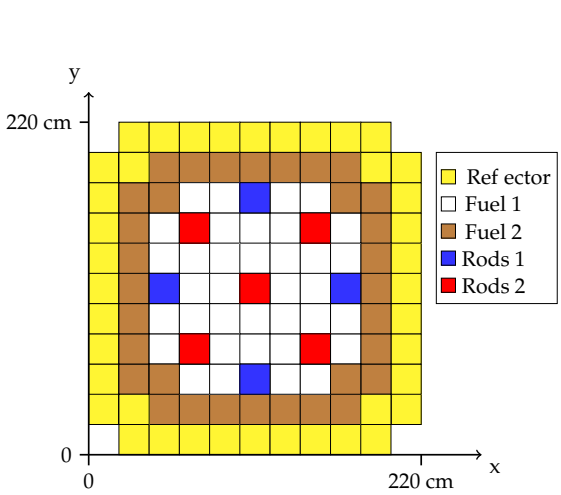
Regarding the validation, table 1b shows the power evolution computed with MINARET, MINOS and the original values from the diffusion benchmark given in [19]. As it can be noted, the MINARET solver presents the correct trend, which will be taken as an indication that the resolution has correctly been implemented.

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).

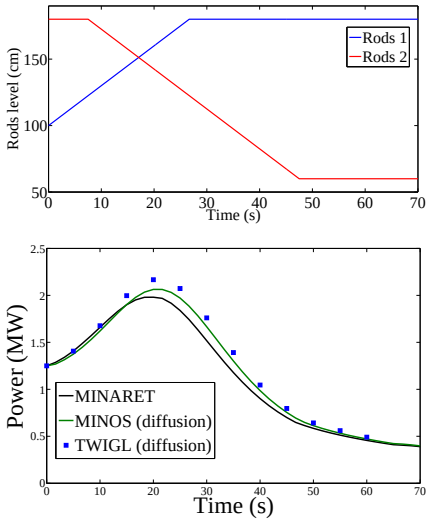
Both tests have been carried out with $G = 2$ energy groups, $L = 6$ precursors and vacuum boundary conditions. The degree of the discontinuous finite elements is P1 and the spatial mesh size is of order 20 cm. All the computations that will be presented hereafter have been obtained in a cluster of 38 nodes of 16 Gb memory, each one composed of 8 cores of 2814 MHz speed.

REMARK 4.1. *In the original TWIGL benchmark from the literature ([19]), 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.*

REMARK 4.2. *We insist on the fact that the test cases that have been considered will not illustrate the advantages of using transport rather than the classical diffusion approximation. They do not show the potentialities of MINARET's unstructured mesh either but we emphasize again that our choice has been determined with the aim of illustrating the algorithmic efficiency of our implementation and not the discretiza-*

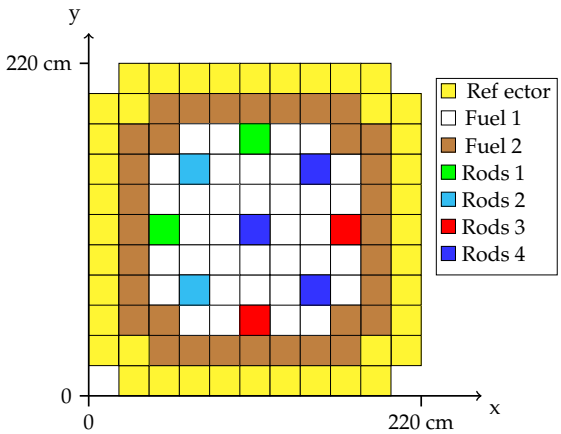


(a) Cross-sectional view of the core ($z = 180\ cm$).

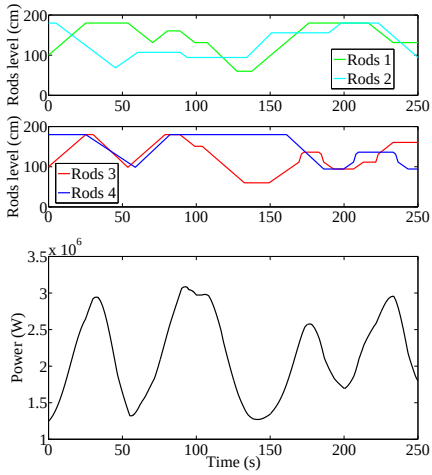


(b) Rods' vertical motion and power evolution.

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.

tion per se; we thus provide results that allow to quantify the effect of the acceleration strategies. The other aspects about approximation rate and capabilities will be illustrated in subsequent works with the solver.

5. Sequential acceleration techniques. The convergence of the iterative resolution of the multigroup problem given in (3.5) is often extremely slow and acceleration

methods are required in order to obtain reasonable computing times.

We will recall in this section the two main traditional sequential accelerations. Their efficiency will be illustrated by some computations with MINARET. The first strategy is the Chebychev extrapolation in the outer iterations. It consists in adding a linear combination of the fluxes after each Gauss-Seidel iteration:

(5.1)
$$\begin{cases} M_{G,G}\boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M+1)}} = \widetilde{N}_{G,G}\boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M)}} + \boldsymbol{S^n} \\ \widetilde{\boldsymbol{\phi}}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M+1)}}(\boldsymbol{r}) = \displaystyle\int_{\mathbb{S}_2} \boldsymbol{\psi}^{\boldsymbol{n+1}}_{\boldsymbol{(M+1)}}(\boldsymbol{r},\boldsymbol{\omega'}) Y_{\ell,m}(\boldsymbol{\omega'}) d\boldsymbol{\omega'} \\ \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M+1)}} = \alpha_{M+1}\left(\widetilde{\boldsymbol{\phi}}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M+1)}} - \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M-1)}}\right) + \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M-1)}},\ M \geq 1 \\ \boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{\ell,m,(M=0)}} \text{ given } (0 \leq \ell \leq L \text{ and } -\ell \leq m \leq \ell). \end{cases}$$

We refer to [31] for the exact form of the coefficients (α_M) and the theoretical foundations of this acceleration scheme.

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:

(5.2)
$$\begin{cases} L^g \widetilde{\psi}^{g,n+1}_{(M,m+1)}(\boldsymbol{r},\boldsymbol{\omega}) = H^{g\to g}\psi^{g,n+1}_{(M,m)}(\boldsymbol{r},\boldsymbol{\omega}) + S(\boldsymbol{r},\boldsymbol{\omega}), \forall g \in \{1,\dots,G\} \\ \widetilde{\phi}^{g,n+1}_{(M,m+1)}(\boldsymbol{r}) = \displaystyle\int_{\mathbb{S}_2} \widetilde{\psi}^{g,n+1}_{(M,m+1)}(\boldsymbol{r},\boldsymbol{\omega'}) d\boldsymbol{\omega'} \\ \psi^{g,n+1}_{(M,m+1)}(\boldsymbol{r},\boldsymbol{\omega}) = \widetilde{\psi}^{g,n+1}_{(M,m+1)}(\boldsymbol{r},\boldsymbol{\omega}) + e^{g,n+1}_{(M,m+1)}(\boldsymbol{r}). \end{cases}$$

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

(5.3)
$$\begin{cases} -\operatorname{div}\left(\dfrac{1}{3\left(\sigma_t^g(\boldsymbol{r}) + \frac{1}{V^g \Delta T_{n+1}}\right)} \nabla e^{g,n+1}_{(M,m+1)}(\boldsymbol{r})\right) \\ \qquad + \left(\sigma_t^g(\boldsymbol{r}) + \frac{1}{V^g \Delta T_{n+1}} - \sigma_s^{g,g}(\boldsymbol{r})\right) e^{g,n+1}_{(M,m+1)}(\boldsymbol{r}) \\ \qquad = \sigma_s(\boldsymbol{r})\left(\widetilde{\phi}^{g,n+1}_{(M,m+1)}(\boldsymbol{r}) - \phi^{g,n+1}_{(M,m)}(\boldsymbol{r})\right), \quad \forall \boldsymbol{r} \in \mathcal{R} \\ e^g_{(m+1)}(\boldsymbol{r}) = 0, \quad \forall \boldsymbol{r} \in \partial\mathcal{R}, \end{cases}$$

where $\phi^{g,n+1}_{(M,m)}(\boldsymbol{r}) = \displaystyle\int_{\mathbb{S}_2} \psi^{g,n+1}_{(M,m)}(\boldsymbol{r},\boldsymbol{\omega'}) d\boldsymbol{\omega'}$.

DSA is an acceleration scheme because it acts as a preconditioner of transport to solve equation (3.11). We refer to [1] (sections I.D and II.B) and [23] for more theoretical details about this method. To preserve the total amount of neutrons, the spatial resolution of the DSA problem (5.3) is discretized with discontinuous Galerkin finite elements of the same degree as the one used for the transport problem (5.2). 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

(5.4)
$$\frac{\|r_i\|_{L^2(\mathcal{R})}}{\|r_0\|_{L^2(\mathcal{R})}}$$

goes below a given convergence threshold ε_{DSA}.

We illustrate the performances of both acceleration methods through some numerical results obtained with MINARET 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 (i.e, a calculation in which the S_n discrete ordinates of order n technique is set to $n = 4$). We also provide the exact computing times obtained in our cluster. The convergence criteria associated to the errors (3.8), (3.12) and (5.4) 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 $\boldsymbol{\phi}^{\boldsymbol{n+1}}_{\boldsymbol{(M=0,N=0)}}$ of the outer iterations is well chosen (we will discuss here the case where $L = 0$, but note that it could very easily be generalized to any type of order L in the scatering expansion with Legendre polynomials). The classical choice is

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)}, \quad \forall g \in \{1,\dots,G\}. \tag{5.5}$$

This option is reasonable because for small time steps one can 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:

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} + \frac{t_{n+1}-t_n}{t_n-t_{n-1}}\left(\phi^{g,n}_{(\infty)} - \phi^{g,n-1}_{(\infty)}\right), \quad \forall g \in \{1,\dots,G\}. \tag{5.6}$$

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

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} \exp^{\frac{t_{n+1}-t_n}{t_n-t_{n-1}} \ln\left(\frac{\phi^{g,n}_{(\infty)}}{\phi^{g,n-1}_{(\infty)}}\right)}, \quad \forall g \in \{1,\dots,G\}. \tag{5.7}$$

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 coming from the diffusion (denoted here as $\tilde{\phi}^{g,n+1}_{(\infty)}$) and use it as a predictor of the transport solution by setting:

$$\phi^{g,n+1}_{(M=0,N=0)} = \tilde{\phi}^{g,n+1}_{(\infty)}, \quad \forall g \in \{1,\dots,G\}.. \tag{5.8}$$

As will be illustrated in the numerical results, this is a bad choice whose main problem is that the diffusion solution has a different trajectory than the transport one, hence the degraded computing times (the transport solver has to correct the trajectory and then 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:

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} + \tilde{\phi}^{g,n+1}_{(\infty)} - \tilde{\phi}^{g,n}_{(\infty)}, \quad \forall g \in \{1,\dots,G\}. \tag{5.9}$$

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:

$$\phi^{g,n+1}_{(M=0,N=0)} = \phi^{g,n}_{(\infty)} \exp^{\frac{t_{n+1}-t_n}{t_n - t_{n-1}} \ln\left(\frac{\tilde{\phi}^{g,n+1}_{(\infty)}}{\tilde{\phi}^{g,n}_{(\infty)}}\right)}, \quad \forall g \in \{1,\dots,G\}. \tag{5.10}$$

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 the TWIGL benchmark case 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 starting guess 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 $M_{outer}N_{inner}N_{DSA}$.

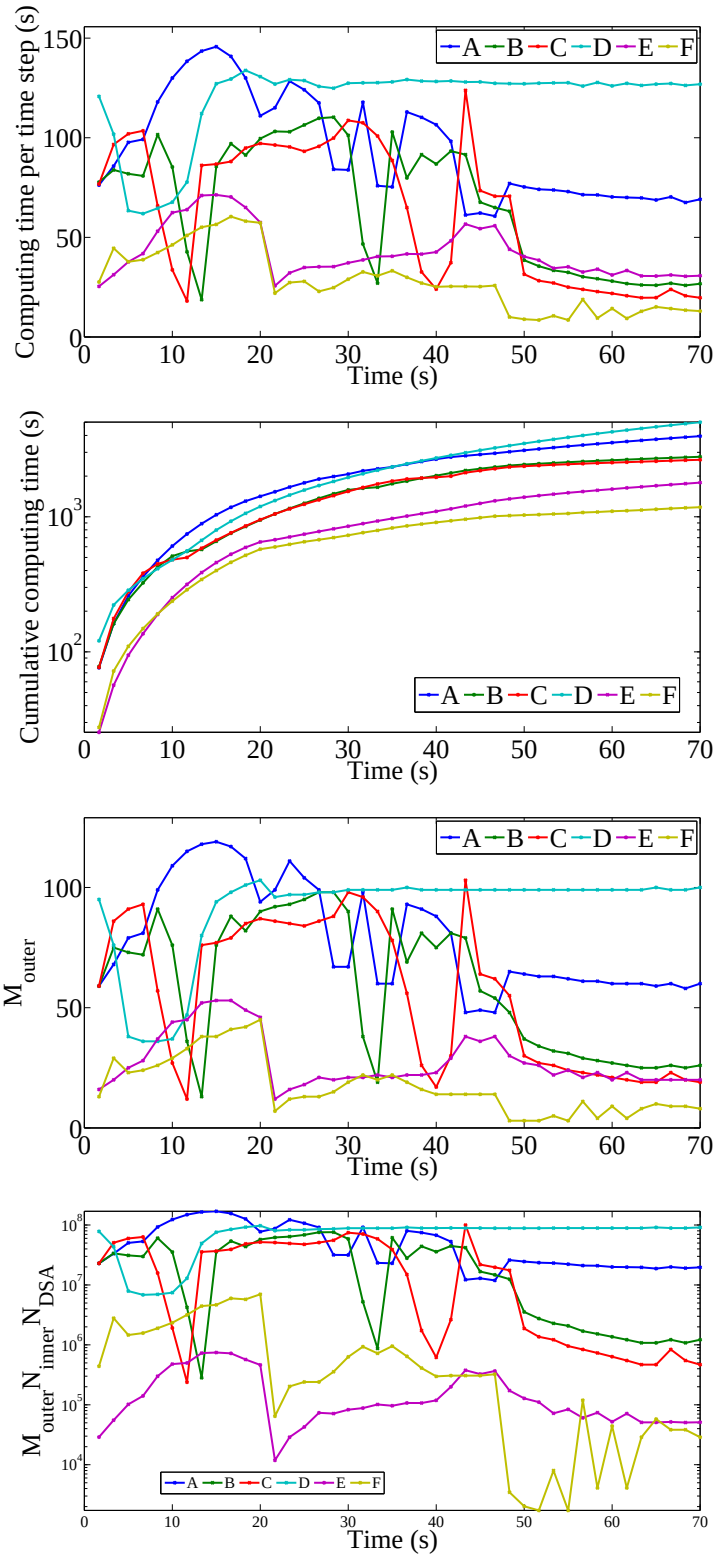


Fig. 1: Performances of the initial guesses.

Note however that the starting guesses have been tried in a case in which diffusion is a good approximation to transport and it is therefore legitimate to wonder whether the starting guesses involving diffusion computations will still be efficient in other contexts. We believe that their performances should not be very degraded as long as the time step remains sufficently small, but computations in a more "transport-oriented" test case are required to prove this claim.

6. Parallelization of the angular variable. We now turn to parallel acceleration techniques and start by recalling in this section the most classical and straightforward strategy employed in discrete ordinates codes:

The numerical scheme outlined in algorithm 1 puts in evidence that, for a given energy group g and a given inner iteration m, each angular flux $\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)$ is computed independently (see lines 7 to 10 of algorithm 1) from 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 a master-slave fashion, that has the important advantage of alleviating the memory storage per processor in comparison with a sequential implementation. Indeed, 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, time-depend calculations involving a large number of directions and leading to HPC problems can be addressed. However, at the end of each inner iteration, the master gathers the angular fluxes $\{\psi(\boldsymbol{r}, \boldsymbol{\omega}_d)\}_{d=1}^{D}$ to compute the scalar flux $\phi(\boldsymbol{r})$ and then it performs the diffusion synthetic acceleration. As a consequence, as shown in, e.g., [18] this part becomes a bottleneck for large numbers of processors because of the communication time and this isssue is one of the reasons that motivates to couple the parallelization of this variable with other strategies. Moreover, apart from this communication problem for large numbers of processors, the efficiency of this method is also 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 (3.11))
- the computing time required to perform the DSA step (that is run sequentially in MINARET)

This trade-off is illustrated on table 5 and figure 2a where the numerical performances of this implementation in a strong scaling test are shown: 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. As it can be observed, 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 spatial domain decomposition methods or multigrid techniques like in the works of [4] and [26] respectively.

As a consequence of all these 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, we perform a weak scaling test in which the angular S_n approximation is increased (D increases) while incrementing the number N of processors. The results are summarized in table 6 and figure 2b where it can be noticed that, for a relatively reduced amount of processors, the efficiency is almost not degraded as N increases. As anticipated before, a degradation would be observed for large N because of significant communication times, but it is important to recall here that the S_n angular accuracy should be chosen in coherence with the accuracy of the other of the variables (like, e.g., the spatial variable whose accuracy is given by the finite element polynomial approximation in our case). For this reason, and also for the time communication issues arising for large N, there is a limit in the total number of processors that is worth devoting to the parallelization of the angular variable. Thus, it is interesting to consider other parallelization strategies as a source of additional speed-up. As outlined in the introduction, significant efforts have been invested in previous works for the parallelization of the spatial variable with the KBA algorithm and promising results have also been obtained regarding the parallelization of the energy. As a complementary approach to all these methods, we propose to explore the additional speed-ups that the parallelization of the time variable could bring to the resolution of our equation.

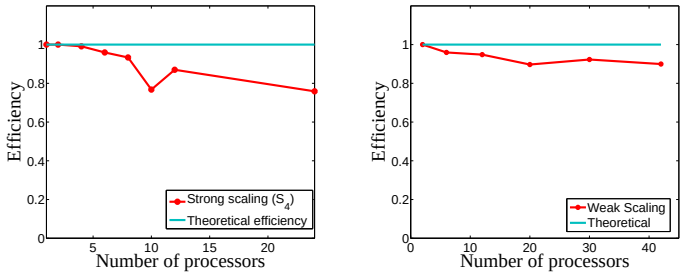
D	24	24	24	24	24	24	24	24
N	1	2	4	6	8	10	12	24
D/N_{proc}	24	12	6	4	3	2 or 3	2	1
Efficiency	1	1	0.99	0.96	0.933	0.77	0.87	0.76

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).

7. Parallelization of the time variable. 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 rest of the variables. They are nevertheless representative



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

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

enough of the accelerations that could be obtained in addition to the ones provided by the other methods.

7.1. The parareal in time algorithm. The unsteady problem (2.1) can be written in a more compact form:

$$\frac{\partial y}{\partial t} + \mathcal{A}(t; y) = 0 \ , t \in [0, T]; \tag{7.1}$$

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

We assume that we have two propagators to solve (7.1): 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 (7.1) 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 another 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 (7.1) 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:

$$Y_{n+1}^{k+1} = \mathcal{G}_{T_n}^{T_{n+1}}(Y_n^{k+1}) + \mathcal{F}_{T_n}^{T_{n+1}}(Y_n^k) - \mathcal{G}_{T_n}^{T_{n+1}}(Y_n^k), \ n = 0, ..., N-1 \tag{7.2}$$

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

From formula (7.2), 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 (7.2), the parareal in time algorithm can be cast into the category of predictor corrector algorithms, where the predictor

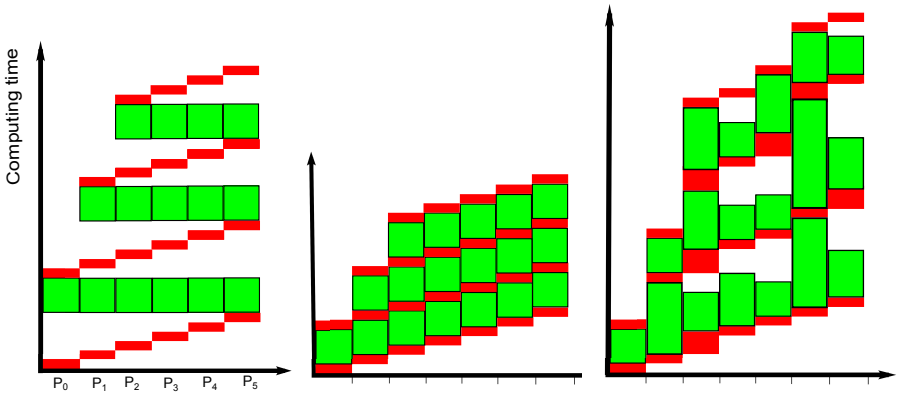
is $\mathcal{G}_{T_n}^{T_{n+1}}(Y_n^{k+1})$ while the corrector is $\mathcal{F}_{T_n}^{T_{n+1}}(Y_n^k) - \mathcal{G}_{T_n}^{T_{n+1}}(Y_n^k)$ (we refer to [16] for a detailed discussion about the several possible interpretrations of the parareal method).

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. [20] [6] [8]), more recent efforts ([22] [5] [15] [12]) focus on the algorithmics to implement it in order to improve the speed-up provided by the original algorithm suggested in [20]. It consists of a master-slave type of implementation: the master carries out the coarse propagation in the whole time interval $[0;T]$, while each slave is in charge of fine propagations over its assigned time slice. Each slave sends $\mathcal{F}_{T_n}^{T_{n+1}}(Y_n^k)$ to the master so that the master computes the parareal corrections of equation (7.2), for all 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 [5]: 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.

$$\|Y_n^{k+1} - Y_n^k\| < \eta,$$

for all n and 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.



(a) A balanced master-slave scheme. (b) A balanced distributed scheme. (c) An imbalanced distributed scheme.

Fig. 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 and the red/green colors represent the coarse/fine propagations).

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 (7.3) below). This is the reason why the distributed algorithm has been implemented in this study.

REMARK 7.1. *An improvement of the distributed algorithm from the processor utilization point of view is the so-called event-based parareal algorithm suggested by [12]. It 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 (3.3), the distributed algorithm has been selected for its simpler implementation.*

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 (resp. coarse) solver does not vary from one processor to another. It is also assumed that the communication time is negligible. Under these two hypothesis, the maximum speed-up for the master-slave (S_{MS}) and distributed algorithms (S_D) are respectively (see [5]):

$$S_{MS} = \frac{T_{seq}}{T_{para,MS}} = \frac{N}{Nr(1+k^*)+k^*}; \qquad S_D = \frac{T_{seq}}{T_{para,D}} = \frac{N}{Nr+k^*(1+r)}, \tag{7.3}$$

where k^* is the total 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 (like in equation (2.1)), it is possible that the cost of the fine and the coarse solvers dramatically vary form one processor to another depending on 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 (7.3) 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

$$\begin{cases} \tilde{S}_D &= \dfrac{T_{seq}}{\tilde{T}_{para,D}} \\ &= \dfrac{T_{seq}}{\sum\limits_{p=0}^{N-1} T_{\mathcal{G}}(0,p) + \sum\limits_{k=1}^{k^*} \max\limits_{p\in\{0,\dots,N-1\}} \left(T_{\mathcal{G}}(k,p) + T_{\mathcal{F}}(k,p)\right)}, \\ \tilde{S}_{MS} &= \dfrac{T_{seq}}{\tilde{T}_{para,MS}} \\ &= \dfrac{T_{seq}}{\sum\limits_{p=0}^{N-1} T_{\mathcal{G}}(0,p) + \sum\limits_{k=1}^{k^*} \left(\sum\limits_{p=0}^{N-1} T_{\mathcal{G}}(k,p) + \max\limits_{p\in\{0,\dots,N-1\}} T_{\mathcal{F}}(k,p)\right)}, \end{cases} \tag{7.4}$$

where the communication time between processors has been neglected. Note that in the generalized formulae (7.4), 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 $T_{\mathcal{G}}(k,p^*) := \max\limits_{p\in\{0,\dots,N-1\}} T_{\mathcal{G}}(k,p)$. More importantly, there is an upper bound for these speed-ups given by N/k^* so that

$$\tilde{S}_{MS} < \tilde{S}_D < \frac{N}{k^*}. \tag{7.5}$$

Because of the upper bound N/k^*, the algorithm turns out to be intrinsically less efficient than other parallelization techniques (like, e.g, spatial domain decomposition) in which one can theoretically expect to reduce by N the computing time with N processors. It could be stated that this lack of efficiency is due to the intrinsic difficulty that the parallelization of the time variable represents, because time is sequential by nature (see however section 7.4 for an current developement to improve this aspect). For this reason, the parareal in time algorithm should be employed to get additional speed-ups when other more efficient parallelization techniques reach saturation.

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.

In all the computations, we will consider that convergence in the parareal scheme is achieved whenever the residual

$$r^k = \max_{n=0,\dots,N} \left(\frac{\sum_{g=1}^{G} \|\Phi_k^{g,n} - \Phi_{k-1}^{g,n}\|^2_{L^2(\mathcal{R})}}{\sum_{g=1}^{G} \|\Phi_k^{g,n}\|^2_{L^2(\mathcal{R})}} \right)^{1/2}, \quad k \geq 1,$$

between the current parareal scalar flux solutions $(\phi_k^{g,n})_{\substack{0\leq n\leq N\\ 1\leq g\leq G}}$ and the previous ones $\left(\phi_{k-1}^{g,n}\right)_{\substack{0\leq n\leq N\\ 1\leq g\leq G}}$ go below a threshold η. Its value has been fixed to the precision of the numerical scheme that has been evaluated to be of order 10^{-3}. Hence, $\eta = 10^{-3}$. 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 through one particular example that parareal effectively converges. The fine and coarse solvers are:

- Fine solver: S_4 transport with $\delta t = 1/12$ s.
- Coarse solver: Diffusion with $\Delta t = 60\delta t = 5$ s.

The points represent the errors

$$e^k(T_n) = \frac{\sum_{g=1}^{G} \|\phi_k^{g,n} - \phi_{(\infty)}^{g,n}\|^2_{L^2(\mathcal{R})}}{\sum_{g=1}^{G} \|\phi_{(\infty)}^{g,n}\|^2_{L^2(\mathcal{R})}}, \ \forall n \in \{0,1,...N\}, \ k \in \{0,1,2\} \tag{7.6}$$

between the parareal scalar fluxes $\Phi_k^{g,n}$ and the sequential fine solution $\phi_{(\infty)}^{g,n}$.

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.

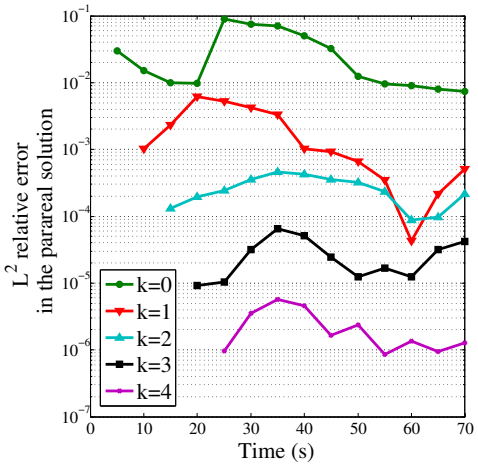


Fig. 4: An example of the numerical convergence of the parareal algorithm in our neutron transport case.

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 $\Delta t = 20\delta t$.

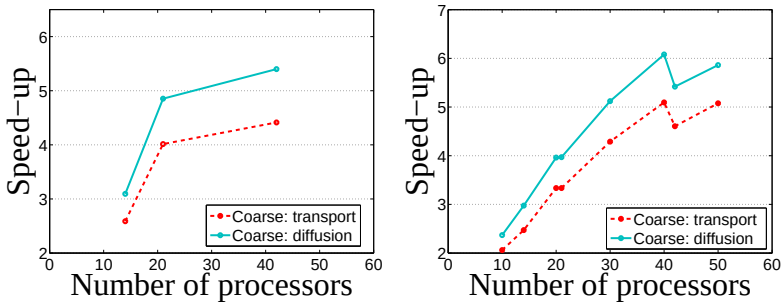
The measured speed-ups S_{measured} are plotted in figure 5a. The evaluation of the maximum attainable speed-up formula $\tilde{S}_D$ can easily be derived thanks to the evaluation of the involved terms given in (7.4). When comparing the measured values with the theoretical ones, we obtain that $S_{\text{measured}} = \tilde{S}_D$ so that the communication time between processors is negligible in our case and the obtained results are optimal.

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 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 with the number of processors. The discontinuity in the trend observed between a number of processors



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

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

Fig. 5: Scaling results with the pararel in time algorithm.

$N = 41$ 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.

7.4. Improvement of the efficiency through a degraded fine solver. In addition to the intrinsic lack of full efficiency of the plain parareal scheme (see relation (7.5) and the results obtained in section 7.3), the example of neutron transport also shows that, in general, the cost of the propagations of the fine solver (and also of the coarse one) can vary from one time t_n to another time $t_{n'}$ and this imbalance leads to a loss in the performances of the parareal scheme (see figure 3c). In the present case, this is due to the fact that the resolution of each time step is performed by iterative techniques (our so-called outer and inner iterations). As a result, for a given convergence criterion in the outer and inner iterations, the cost of the fine propagations $\mathcal{F}_{T_n}^{T_{n+1}}$ can dramatically vary from one processor to another depending on the numerical complexity that takes place in each time slice $[T_n; T_{n+1}]$ to which processor P_n is assigned (notice, by the way, that this situation is very general and can occur in the resolution of other PDEs).

A strategy that is currently being studied is to degrade the fine solver $\mathcal{F}$ and build a solver $\widetilde{\mathcal{F}}_J$ that performs a reduced number J of outer iteration at each time step δt instead of making $M_{n,\varepsilon_{outer}} \gg J$ outer iterations until convergence to ε_{outer} of the residual error (see formula (3.8)). As a consequence, instead of converging "locally" at every parareal iteration, the convergence is expected to be achieved "globally" across the parareal iterations through the modified parareal scheme:

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

This development is in the same spirit as the works of Maday and Turinici in [21] or Minion in [22], [15], where the idea of degrading the fine propagator and coupling it with other iterative methods such as spatial domain decomposition and spectral deferred corrections has already been tested. The main novelty of our proposed approach is that the degradation is purely algebraic and the main difficulty is of course to prove the convergence of the degraded parareal scheme. Preliminary results in this respect can be found in chapter 3 of [25] in which the neutron diffusion equation has been adressed as an example. The study of this idea will be the topic of two subsequent papers: in the first one we will provide theoretical results of convergence and

provide numerical results in a simple case. On a second stage, we plan to implement the scheme in the MINARET solver to carry a study about the real gain in efficiency that the method could provide.

7.5. A parareal in space and energy algorithm?. The parareal method was originally suggested for the time variable but it is quite straightforward to realize that the variable t of equation (7.1) 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 (3.11)) 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 (2.1). Furthermore, if we now observe the multigroup problem (equation (3.3)) 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. An analysis of traditional and innovative acceleration techniques that can be included in time-dependent neutron transport solvers without altering their main construction has been presented. A particular effort has been made to provide tractable numerical results about the performances of the methods. This has been achieved by the presentation computational results derived in a classical benchmark test case from the literature and obtained with the newly developed time-depend module of the MINARET solver.

Hence, it has been shown that the inclusion of the classical Chebyshev extrapolation and diffusion synthetic acceleration reduce the computing times by a factor of about 100. It has further been noted that one can still reduce the computing time by a factor of about 3 with the use of an appropriate starting guess for the outer iterations that involves diffusion propagations.

Regarding parallel accelerations, it has first been explained how the parallelization of the angular directions can efficiently speed-up calculations under several conditions (the efficiency of an S_{12} calculation involving 168 directions is 90%). However, the parallelization of other variables is justified because there are limitations in the amount of processors that one should devote to this strategy for discretization and communication reasons. For this reason, it is interesting to explore complementary strategies such as 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 (e.g., a transient of 210 s. can be accelerated by a factor of 6 with 50 processors). In an ongoing work, a parareal scheme that involves a degraded fine solver with truncated outer iterations is being analyzed in an attempt to improve the efficiency of the method (see chapter 3 of [25] for preliminary results).

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

REFERENCES

[1] M. L. Adams and E. W. Larsen, *Fast iterative methods for discrete-ordinates particle transport calculations*, Progress in Nuclear Energy, 40(1) (2002), pp. 3 – 159.

[2] P Adams, M. L. Adams, W. D. Hawkins, T. Smith, L. Rauchwerger, and N. M. Amato, *Provably optimal parallel transport sweeps on regular grids*, in International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, 2013.

[3] R. E. Alcouffe and R.S. Baker, *Time-dependent neutron transport on parallel architectures using PARTISN*, in Radiation Protection and shielding division Topical Conference, 1998.

[4] P. F. Antonietti and B. Ayuso, *Schwarz domain decomposition preconditioners for discontinuous galerkin approximations of elliptic problems: non-overlapping case*, ESAIM, Math. Model. Numer. Anal., 41 (2007), pp. 21–54.

[5] E. Aubanel, *Scheduling of tasks in the parareal algorithm*, Parallel Computing, 37 (2011), pp. 172–182.

[6] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, *Parallel-in-time molecular-dynamics simulations*, Phys. Rev. E, 66 (2002).

[7] R.S. Baker and K. R. Koch, *An S_n algorithm for the massively parallel CM-200 computer*, Nucl. Sci. Eng., 128 (1998).

[8] G. Bal and Y. Maday, *A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put*, Recent developments in domain decomposition methods, 23 (2002), pp. 189–202.

[9] A.-M. Baudron and J.-J. Lautard, *MINOS: a simplified P_N solver for core calculation*, Nucl. Sci. Eng., 155(2) (2007), pp. 250–263.

[10] A.-M. Baudron, J.-J. Lautard, Y. Maday, and O. Mula, *The parareal in time algorithm applied to the kinetic neutron diffusion equation*, in 21st International Conference on Domain Decomposition Methods, 2012.

[11] A.-M. Baudron, J.-J. Lautard, Y. Maday, K. Riahi, and J. Salomon, *Parareal in time 3D numerical solver for the LWR Benchmark neutron diffusion transient model*, Journal of Computational Physics, 279 (2014), pp. 67–79.

[12] L.A. Berry, W. Elwasif, J.M. Reynolds-Barredo, D. Samaddar, R. Sanchez, and D.E. Newman, *Event-based parareal: A data-flow based implementation of parareal*, J. Comput. Phys., 231 (2012), pp. 5945 – 5954.

[13] S. Chauvet, A. Nachaoui, A.-M. Baudron, and 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.

[14] R. Dautray and J.-L. Lions, *Analyse mathématique et calcul numérique*, Masson, 1984.

[15] M. Emmet and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Comm. App. Math. and Comp. Sci., 1 (2012).

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

[17] E. Girardi, P. Guérin, S. Dulla, and 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.

[18] E. Jamelot, J. Dubois, J.-J. Lautard, C. Calvin, and 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.

[19] S. Langenbuch, W. Maurer, and 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), pp. 437–456.

[20] J.L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps pararéel*, C. R. Acad. Sci. Paris, (2001). t. 332, Série I, p. 661-668.

[21] Y. Maday and G. Turinici, *The Parareal in Time Iterative Solver: a Further Direction to Parallel Implementation*, in Domain Decomposition Methods in Science and Engineering, Springer Berlin Heidelberg, 2005, pp. 441–448.

[22] M. Minion, *A hybrid parareal spectral deferred corrections method*, Comm. App. Math. and

Comp. Sci., 5 (2010).

[23] J.-Y. Moller, *Éléments finis courbes et accélération pour le transport de neutrons*, PhD thesis, Université Henri Poincaré , 2012.

[24] J.-Y. Moller and 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.

[25] O. Mula, *Some contributions towards the parallel simulation of time dependent neutron transport and the integration of observed data in real time*, PhD thesis, Paris VI, 2014.

[26] A. Napov and Y. Notay, *An algebraic multigrid method with guaranteed convergence rate*, SIAM J. Sci. Comput., 34 (2012), pp. A1079–A1109.

[27] S.D. Pautz, *An algorithm for parallel S_N sweeps on unstructured meshes*, Nucl. Sci. Eng., 34 (2002), pp. 111–136.

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

[29] A. Saubert, A. Sureda, J. Bader, J. Lapins, M. Buck, and 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), pp. 173–180.

[30] R.N. Slaybaugh, T.M. Evans, G.G. Davidson, and P.P. Wilson, *Rayleigh quotient iteration in 3d, deterministic neutron transport*, in PHYSOR, Advances in Reactor Physics, 2012.

[31] R. S. Varga, *Matrix iterative analysis*, Springer series in computational mathematics, Springer Verlag, Berlin, Heidelberg, Paris, 2000.