10.18605/2175-7275/cereus.v14n1p349-366

ARTIGO ORIGINAL

<< Recebido em: 27/02/2022 Aceito em: 06/04/2022. >>

REVISTA CEREUS

ISSN: 2175-7275

Parallelization of a coarse-mesh neutron transport method applied in the solution of one-dimensional neutron shielding problems

Paralelização de um método malha grossa de transporte de nêutrons applicado na solução de problemas unidimensionais de transporte de nêutrons

Rafael Barbosa Libotte[1], Hermes Alves Filho[2], Fernando Carvalho da Silva[3]

ABSTRACT

In this work, a coarse-mesh method used in the solution of neutron shielding problems in slab geometry is parallelized, using the memory shared environment OpenMP. The neutron transport problem is modelled using the linearized Boltzmann equation considering the energy multigroup theory and the discrete ordinate formulation (S_N). The numerical method, named Modified Spectral Deterministic (MSD), uses the neutrons transport intranodal general solution alongside an iterative process to calculate the outgoing neutron angular fluxes on the nodal interfaces. Two model-problems are solved, comparing the iterative process execution time using different number of threads. The numerical results, which achieved around 30% and 50% better execution time, were generated using the programming language C++.

Keywords: Parallel Programing. OpenMP. Neutron transport theory. Neutron shielding problems. Spectral-nodal methods.

RESUMO

Neste trabalho, um método de malha grossa usado na solução de problemas de blindagem de nêutrons em geometria *slab* foi paralelizado, usando o ambiente de memoria compartilhada OpenMP. O problema de transporte de nêutrons é modelado segundo a equação linearizada de Boltzmann, considerando a teoria mutigrupos de energia e a formulação das ordenadas discretas (S_N). O método numérico, de nome Modified Spectral Deterministic (MSD), usa a solução intranodal geral da equação de transporte de nêutrons e um processo iterativo que calcula os fluxos angulares de nêutrons emergentes nas interfaces nodais. Dois problemas-modelo são resolvidos, comparando os tempos de execução do processo iterativo usando diferentes numeros de *threads*. Os resultados numéricos, os quais atingiram cerca de 30% e 50% de melhora nos tempos de execução, foram gerados usando a linguagem de programação C++.

Palavras-chave: Programação paralela. OpenMP. Teoria de tranasporte de nêutrons. Problemas de blindagem de nêutrons. Métodos espectro-nodais.

[1] Instituto Politécnico - Universidade do Estado do Rio de Janeiro - Nova Friburgo, RJ, Brasil.

E-mail: rafaellibotte@hotmail.com

https://orcid.org/0000-0001-8864-7906

[2] Instituto Politécnico - Universidade do Estado do Rio de Janeiro - Nova Friburgo, RJ, Brasil.

E-mail: halves@iprj.uerj.br

https://orcid.org/0000-0001-6905-0039

[3] Instituto Politécnico - Universidade do Estado do Rio de Janeiro - Nova Friburgo, RJ, Brasil.

E-mail: fernando@con.ufrj.br

https://orcid.org/0000-0001-9181-7582

1. INTRODUCTION

Nowadays, the growing sophistication of engineering problems, being on the development of new technologies or the enhancement of existing ones, created a demand for computational algorithms that are capable of achieving precise results in reduced times. These problems require a big effort to calculate and control a large number of variables, due to the elaboration of robust mathematical models, resultant of physical phenomena analysis.

The neutron transport problem, for shielding calculations (fixed-source), can be modelled with the linearized Boltzmann equation (Lewis & Miller, 1993). The analytical solution of this equation is highly complex (Case & Zweifel, 1967). Therefore, some numerical approaches were developed in order to obtain solutions for these problems. Among the different types of numerical solutions, we can cite fine-mesh methods, e.g. Diamond Difference (DD) and Constant Nodal (CN) (Lewis & Miller, 1993), and coarse-mesh methods, e.g. Modified Spectral Deterministic (MSD) (Libotte, 2021.

In the field of numerical methods, some modifications in its algorithms can be made so these can present better performance in the execution and solution of a model-problem. Among these, the parallelization of sub-routines that compose the MSD can be cited. In this work, the OpenMP (Dagum, 1998) interface was used to perform this task. A model-problem was simulated and the execution time of the algorithm with different number of threads was compared.

Hereafter, the sections that compose this work will be shown. In Section 2, it is briefly presented the mathematical modelling of the neutron transport problem, for shielding calculations, in the discrete ordinate formulation (S_N). In Section 3, it is shown the equations that compose the spectral-nodal method MSD, used in the solution of the problem. In Section 4, it is shown the sub-routines that were parallelized on the computational algorithm, and the directives used for it. In Section 5, the numerical results of a performance test in the solution if a model-problem is shown. Lastly, in Section 6, it is presented the conclusions and future perspectives for the continuity of this work.

2. MATHEMATICAL MODELLING

In the mathematical modelling of a neutron shielding problem, the linearized Boltzmann equation (Lewis & Miller, 1993) can be used. This equation, in the energy multigroup formulation, in a one-dimensional geometry domain of dimension H, stationary, with fixed external source and in the discrete ordinate formulation, is shown in Eq. (1)

$$\mu_m \frac{d}{dx}\psi_{m,g}(x) + \sigma_{T,g}(x)\psi_{m,g}(x) = \sum_{g'=1}^{G} \frac{\sigma_{S0}^{g'\to g}(x)}{2} \sum_{n=1}^{N} \omega_n \psi_{n,g'}(x) + Q_g(x), \qquad (1)$$
$$m = 1:N,\ g = 1:G,\ x \in [0,\ H].$$

Where μ_m and ω_m represents respectively the roots of the Legendre polynomial angular set of order N and the weights of the Gauss-Legendre quadrature, $\psi_{m,g}$ represents the neutron angular fluxes of group g among the G discretized energy groups, $\sigma_{T,g}$ and $\sigma_{S0}^{g'\to g}$ represents respectively the macroscopic total cross section of group g and the macroscopic scattering cross section from group g' to group g, and Q_g represents an external isotropic neutron source of group g.

In this work, two types of boundary conditions were approached, which are presented in the following form:

- **Prescribed**

$$\psi_{m,g}(x) = \begin{cases} b_{m,g}, & \text{if } x = 0 \text{ and } \mu_m > 0,\ g = 1:G, \\ c_{m,g}, & \text{if } x = H \text{ and } \mu_m < 0,\ g = 1:G. \end{cases} \qquad (2)$$

- **Reflective**

$$\begin{cases} \psi_{m,g}(0) = \psi_{m+N/2,g}(0), \text{ for } \mu_m > 0,\ g = 1:G, \\ \psi_{m,g}(H) = \psi_{m-N/2,g}(H), \text{ for } \mu_m < 0\ g = 1:G. \end{cases} \qquad (3)$$

2.1 Spatial Discretization

Now, consider an arbitrary spatial grid Γ defined in a one-dimensional domain of width H, an shown in Figure 1. The grid is composed of J spatial nodes Γ_j of width h_j with uniform physical-material parameters. Using Eq. (1) in an arbitrary node Γ_j, it assumes the form:

$$\mu_m \frac{d}{dx}\psi_{m,g}(x) + \sigma_{T,g,j}\psi_{m,g}(x) = \sum_{g'=1}^{G} \frac{\sigma_{S0,j}^{g'\to g}}{2} \sum_{n=1}^{N} \omega_n \psi_{n,g'}(x) + Q_{g,j}, \qquad (4)$$
$$m = 1:N,\ g = 1:G,\ x \in \Gamma_j.$$

In Figure 1, it is also showed the representation of the neutron angular fluxes ($\psi_{m,g}$) on the Γ_j node-edges, for a quadrature order $N = 4$.

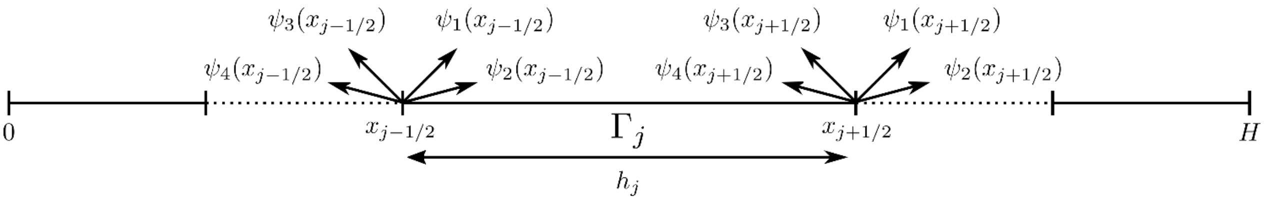


Figure 1: Spatial grid Γ_j in a one-dimensional domain of width H.

The Eq. (4) has an intranodal analytical solution, in the form

$$\psi_{m,g}(x) = \psi^h_{m,g}(x) + \psi^p_{m,g},\ m = 1:N,\ g = 1:G,\ x \in \Gamma_j. \tag{5}$$

The particular solution ψ^p can be calculated using the system of equations (Barros, 1990)

$$\sigma_{T,g,j}\psi^p_{m,g} - \sum_{g'=1}^{G} \frac{\sigma^{g'\to g}_{S0,j}}{2} \sum_{n=1}^{N} \omega_n \psi^p_{n,g'} = Q_{g,j},\ m = 1:N,\ g = 1:G. \tag{6}$$

For the homogeneous solution (ψ^h), let us consider the expression (Barros, 1990)

$$\psi^h_{m,g}(x) = a_{m,g}(\vartheta)e^{-(x-x_{j-1/2})/\vartheta},\ m = 1:N,\ g = 1:G. \tag{7}$$

Substituting Eq. (7) on the homogeneous part of Eq. (4), we obtain the following eigenvalue problem

$$\frac{\delta_{m,n}\sigma_{T,g,j}a_{m,g}(\vartheta)}{\mu_m} - \sum_{g'=1}^{G} \frac{\sigma^{g'\to g}_{S0,j}}{2\mu_m} \sum_{n=1}^{N} \omega_n a_{n,g'}(\vartheta) = \frac{1}{\vartheta}a_{m,g}(\vartheta),\ m = 1:N,\ g = 1:G. \tag{8}$$

The Eq. (8) generates NG real and symmetric eigenvalues ϑ and NG real eigenvectors $a(\vartheta)$. Thus, the intranodal general solution in Γ_j of Eq. (4) has the form

$$\psi_{m,g}(x) = \sum_{l=1}^{NG} \alpha_l a_{m,g}(\vartheta_l)e^{-(x-x_{j-1/2})/\vartheta_l} + \psi^p_{m,g},\ m = 1:N,\ g = 1:G, \tag{9}$$

where α_l represents constants to be determined.

3. MÉTODO MODIFIED SPECTRAL DETERMINISTIC (MSD)

The MSD method is a coarse-mesh formulation, that uses the intranodal general solution of the S_N neutron transport equation, represented by Eq. (9) and an iterative process that uses the spatial balance S_N equations to calculate the outgoing angular neutron fluxes on the node-edges.

The first step to develop this method is the application of the average operator on Eq. (4), resulting in the S_N spatial balance equation, which assumes the form

$$\frac{\mu_m}{h_j}\left(\psi_{m,g,j+1/2}-\psi_{m,g,j-1/2}\right)+\sigma_{T,g,j}\overline{\psi}_{m,g,j}=\sum_{g'=1}^{G}\frac{\sigma_{S0,j}^{g'\to g}}{2}\sum_{n=1}^{N}\omega_n\overline{\psi}_j+Q_{g,j}, \qquad m=1:N,\ g=1:G. \tag{10}$$

Where, by definition, we have the following expression

$$\overline{\psi}_{m,g,j}\equiv\frac{1}{h_j}\int_{x_{j-1/2}}^{x_{j+1/2}}\psi_{m,g}(x)\mathrm{d}x \tag{11}$$

Substituting the $\psi_{m,g}(x)$ from Eq. (9) in Eq. (11), we have

$$\overline{\psi}_{m,g,j}=\begin{cases}\sum_{l=1}^{NG}\frac{-1}{h_j}\alpha_l\vartheta a_{m,g}(\vartheta_l)\left(e^{-h_j/\vartheta_l}-1\right), & \text{se } \vartheta_l>0,\ m=1:N,\ g=1:G,\\ \sum_{l=1}^{NG}\frac{-1}{h_j}\alpha_l\vartheta_l a_{m,g}(\vartheta_l)\left(1-e^{h_j/\vartheta_l}\right), & \text{se } \vartheta_l<0,\ m=1:N,\ g=1:G.\end{cases} \tag{12}$$

After some algebraic manipulations in Eq. (10), we obtain the following formulations (Libotte, 2021). For the outgoing angular neutron fluxes on the right node-edge ($\mu_m>0$), we obtain the expression

$$\psi_{m,g,j+1/2}=\frac{hj}{\mu_m}\left(-\sigma_{T,g,j}\overline{\psi}_{m,g,j}+SS_{g,j}+Q_{g,j}\right)+\psi_{m,g,j-1/2}, \qquad m=1:N/2,\ g=1:G, \tag{13}$$

For the neutron angular fluxes on the left node-edge ($\mu_m<0$), we obtain the expression

$$\psi_{m,g,j-1/2} = \frac{hj}{|\mu_m|}\left(-\sigma_{T,g,j}\overline{\psi}_{m,g,j} + SS_{g,j} + Q_{g,j}\right) + \psi_{m,g,j+1/2},$$
$$m = N/2+1:N,\ g=1:G, \tag{14}$$

where the scattering source is given by the equation

$$\max_{j=1:J+1,\ g=1:G}\left|\frac{\phi^{i}_{g,j-1/2} - \phi^{i-1}_{g,j-1/2}}{\phi^{i-1}_{g,j-1/2}}\right| \times 100\% < \xi. \tag{15}$$

After this step, it is possible to start the spatial grid sweeping process, where the outgoing neutron angular fluxes are calculated on the interfaces of every Γ_j spatial node. Using the initial estimates of the incoming neutron angular fluxes, a set of α parameters is calculated using Eq. (9) (Oliva, 2018), here some numerical adjustments are made to make this calculation more stable (Silva, 2018). With these values, the average neutron angular fluxes of the first node can be calculated using Eq. (12). Now, the outgoing neutron angular fluxes on the right of the first node can be calculated with Eq. (13), and on the left with Eq. (14) (Libotte, 2021). On Figure 2, it is shown a representation of the first step of the iterative process, where the black solid arrows represent the boundary conditions, the dotted grey ones are the initial estimates of the neutron angular fluxes, and the dashed red ones are the new estimates of the outgoing neutron angular fluxes of the first node.

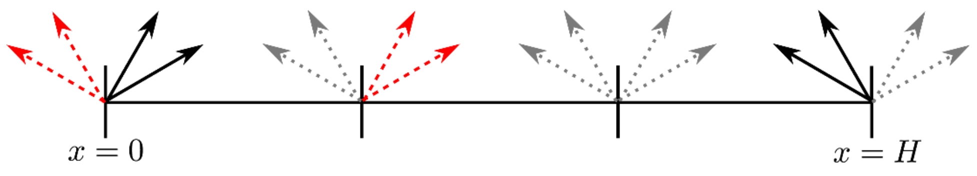


Figure 2: MSD sweeping.

From the new neutron angular fluxes estimates, the same process is made on the other nodes, calculating the outgoing neutron angular fluxes on every node, until all the grid is swept. After obtaining these values, the neutrons scalar fluxes on the nodal interfaces can be calculated, using the equation

$$\phi_g(x) = \frac{1}{2}\sum_{n=1}^{N}\psi_{n,g}(x)\omega_n. \tag{16}$$

With the neutron scalar fluxes calculated in all nodal interfaces, the stopping criterion for the iterative process can be verified according to the equation

$$\max_{j=1:J+1,\ g=1:G} \left| \frac{\phi^{i}_{g,j-1/2} - \phi^{i-1}_{g,j-1/2}}{\phi^{i-1}_{g,j-1/2}} \right| \times 100\% < \xi. \tag{17}$$

Thereby, the iterative process can be interrupted when the maximum difference of the neutron scalar flux in a point of the domain between two subsequent iterations is smaller than a preestablished precision parameter ξ.

4. PARALLELIZATION PROCESS

In this section, it is briefly shown the directives used in the parallelization process of the MSD method using the shared memory programming interface OpenMP. On MSDs iterative process, some sub-routines can be parallelized in order to reduce its execution time. Therefore, hereafter some parts of the algorithm that are being performed in parallel are shown.

On the parallelization of a sub-routine, initially the parallel programming environment is initialized, with the directive $\#pragma\_omp\_parallel\{\ldots\}$, which comprehends all the parallel portion of the algorithm (IBM, 2021). In order to execute a for loop, it is used the directive $\#pragma\_omp\_parallel\ \ for$ (Microsoft, 2019). In Algorithm 1, it is shown the parallelization process to calculate the α parameters in a region of the domain. According to Eq. (9). Thereunto, at first the number of threads T to be used in the parallel programming environment must be defined, using the directive $omp\_set\_num\_threads\ (T)$. Inside the loop, the work is divided between the threads to execute the algorithm (Microsoft, 2019).

Algorithm 1: Parallelization of the α parameters inside a region k calculations.

```
$for\ (i \leq G)\ do$
    $\#pragma\ omp\ parallel\ for$
    $for(j \leq N/2)\ do$
        $\psi_{temp,j,i} = \psi_{j,i,k} - \psi_{p,j,i,k}$
    $end$
    $\#pragma\ omp\ parallel\ for$
    $for(N/2 \leq j \leq N)\ do$
        $\psi_{temp,j,i} = \psi_{j,i,k+1} - \psi_{p,j,i,k}$
    $end$
    $\alpha_k = A \times \psi_{temp}$
$end$
```

Output: $\boldsymbol{\alpha_k}$ parameters.

The second sub-routine that was parallelized in this work, corresponds to the average angular neutron fluxes calculations, using Eq. (12). In this step, beyond the already shown directives, it is also used the directive $\#omp\ reduction\ operation(var)$. This one reduces private variables of each of the T threads into a variable var using an operator $operation$ (Microsoft, 2019). In Figure 3, it is shown an example of this directive usage. On the OpenMP Environment block, it is represented the start of the parallel computation environment, then the work inside the for loops is done for the threads $1,2, \ldots, T$, at last, with the directive $reduction$, the private partial results in each thread is summed (as seen the + sign in the figure) into the variable aux.

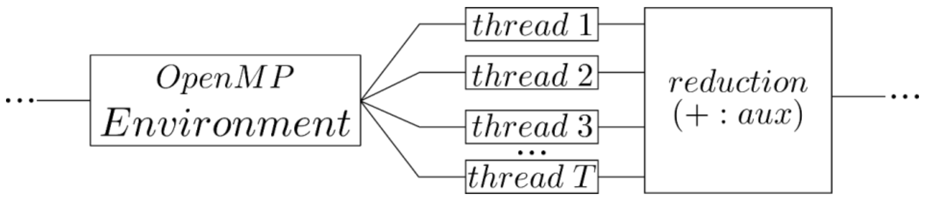


Figure 3: $reduction$ directive representation inside OpenMP environment.

In Algorithm 2, it is shown a parallelized sub-routine to calculate the average neutron angular fluxes inside a region.

Algorithm 2: Parallelization of the average neutron angular fluxes calculations.

```
$for\ (i \leq NG) do$
    $\#pragma\ omp\ parallel\ for\ reduction\ (+: \bar{\psi}_{i,k})$
    $for\ (j \leq NG)\ do$
        $if\ \left(\vartheta_{j,k} > 0\right)\ then$
            $\bar{\psi}_{i,k} = \bar{\psi}_{i,k} + \alpha_{j,k} V_{i,j,k}\, \vartheta_{j,k} \left(e^{-\frac{h_k}{\vartheta_{j,k}}} - 1\right)$
        $else$
            $\bar{\psi}_{i,k} = \bar{\psi}_{i,k} + \alpha_{j,k} V_{i,j,k}\, \vartheta_{j,k} \left(1 - e^{\frac{h_k}{\vartheta_{j,k}}}\right)$
        $end$
    $end$
$\bar{\psi}_{i,k} = -\frac{1}{h_k}\, \bar{\psi}_{i,k} + \psi_{p.i.k}$
$end$
```

Output: Average neutron angular fluxes ($\bar{\psi}_k$).

The third and last parallelized sub-routine is the scattering source calculations inside a region. On this one, as in the average neutron angular fluxes calculations, the $reduction$ directive was used to group the partial scattering sources private variables in each thread in a single output. In order to parallelize more than one for loop at once, the $collapse(num)$ directive is used. In this directive the number num is set to specify the number of loops to be associated in the multi-threading environment (OpenMP, 2022). Algorithm 3 shows how this sub-routine was parallelized for this work.

Algorithm 3: Parallelization of the scattering source calculations.

```
$for\ (i \leq G)\ do$
    $\#pragma\ omp\ parallel\ for\ reduction\ (+: SS_{i,k})\ collapse\ (2)$
    $for\ (j \leq G)\ do$
        $for\ (l \leq N)\ do$
            $SS_{i,k} = SS_{i,k} + \frac{\sigma_{S0,k,j,i}}{2}\ \omega_l\ \bar{\psi}_{l,j,k}$
        $end$
    $end$
$end$
```

Output: Scattering source (SS_k)

5. NUMERICAL RESULTS

In this section, the numerical results of two model-problems using the method MSD are shown. Performance tests were performed, comparing the execution time of the algorithm using different number of threads.

5.1 Model problem 1

The first model-problem, consists in a $100\ cm$ domain composed by 4 regions and 3 material zones. This problem has 2 energy groups, and reflective boundary condition on the left and vacuum on the right, which is a particular kind of prescribed condition with incoming neutron angular fluxes $\psi(H)_{m,g} = 0$. The geometry and the physical-material parameters, other than the macroscopic scattering cross-sections are shown in Figure 4.

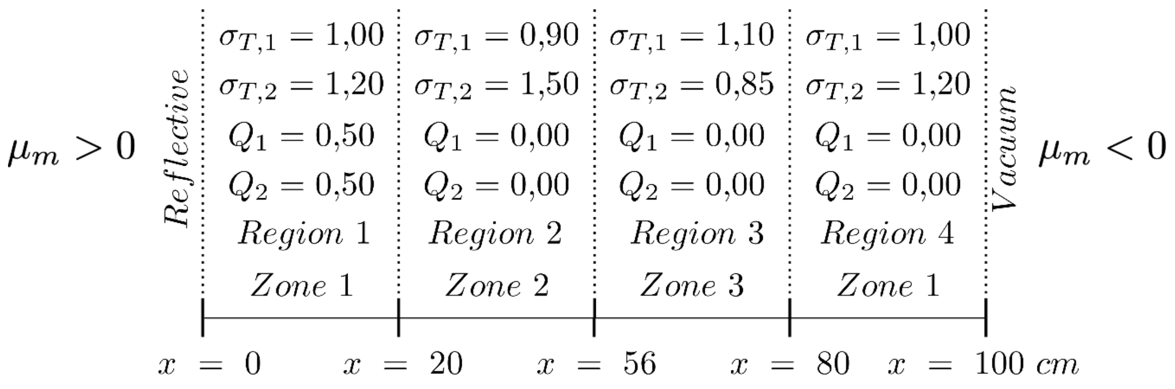


Figure 4: Model-problem 1.

On Table 1, the macroscopic scattering cross sections of the 3 material zones are shown.

Table 1: Macroscopic scattering cross-sections (cm^{-1}) of the model-problem.

	Zone 1		Zone 2		Zone 3	
	$g = 1$	$g = 2$	$g = 1$	$g = 2$	$g = 1$	$g = 2$
$\sigma_{S0}^{1\to g}$	0.90	0.20	0.75	0.30	0.95	0.60
$\sigma_{S0}^{2\to g}$	0.05	0.80	0.10	0.99	0.00	0.20

At first, a precision test was done on this model-problem. The numerical algorithm was executed with different quadrature orders sets, and different number of threads. The results for the scalar neutron fluxes in the region interfaces can be seen in Table 2, alongside its reference, using the DD method with a mesh with 10000 nodes in the first and fourth regions, 18000 in the second one, and 12000 in the third one.

Table 2: Numerical results of the scalar neutron fluxes ($cm^{-2}s^{-1}$) for model-problem 1.

	N	g	$0\ cm$	$20\ cm$	$56\ cm$	$80\ cm$	$100\ cm$
MSD	128	1	7.49902	3.51559	1.87411×10^{-7}	2.72813×10^{-14}	1.60032×10^{-18}
		2	4.99943	2.55791	1.37085×10^{-7}	3.45837×10^{-14}	8.38927×10^{-19}
	256	1	7.99022	3.51559	1.87411×10^{-7}	2.72813×10^{-14}	1.60031×10^{-18}
		2	4.99943	2.55790	1.37085×10^{-7}	3.45837×10^{-14}	8.38926×10^{-19}
	512	1	7.49902	3.51559	1.87411×10^{-7}	2.72813×10^{-14}	1.60031×10^{-18}
		2	4.99943	2.55790	1.37085×10^{-7}	3.45837×10^{-14}	8.38926×10^{-19}
	1024	1	7.49903	3.51559	1.87412×10^{-7}	2.72813×10^{-14}	1.60031×10^{-18}
		2	4.99943	2.55790	1.37085×10^{-7}	3.45837×10^{-14}	8.38927×10^{-19}
DD	128	1	7.49902	3.51559	1.87411×10^{-7}	2.72813×10^{-14}	1.60030×10^{-18}
		2	4.99943	2.55790	1.37085×10^{-7}	3.45836×10^{-14}	8.38917×10^{-19}
	256	1	7.49902	3.51559	1.87411×10^{-7}	2.72813×10^{-14}	1.60029×10^{-18}
		2	4.99943	2.55790	1.37085×10^{-7}	3.45836×10^{-14}	8.38916×10^{-19}
	512	1	7.49902	3.51559	1.87411×10^{-7}	2.72813×10^{-14}	1.60029×10^{-18}
		2	4.99943	2.55790	1.37085×10^{-7}	3.45836×10^{-14}	8.38916×10^{-19}
	1024	1	7.49902	3.51559	1.87411×10^{-7}	2.72813×10^{-14}	1.60028×10^{-18}
		2	4.99943	2.55790	1.37085×10^{-7}	3.45836×10^{-14}	8.38916×10^{-19}

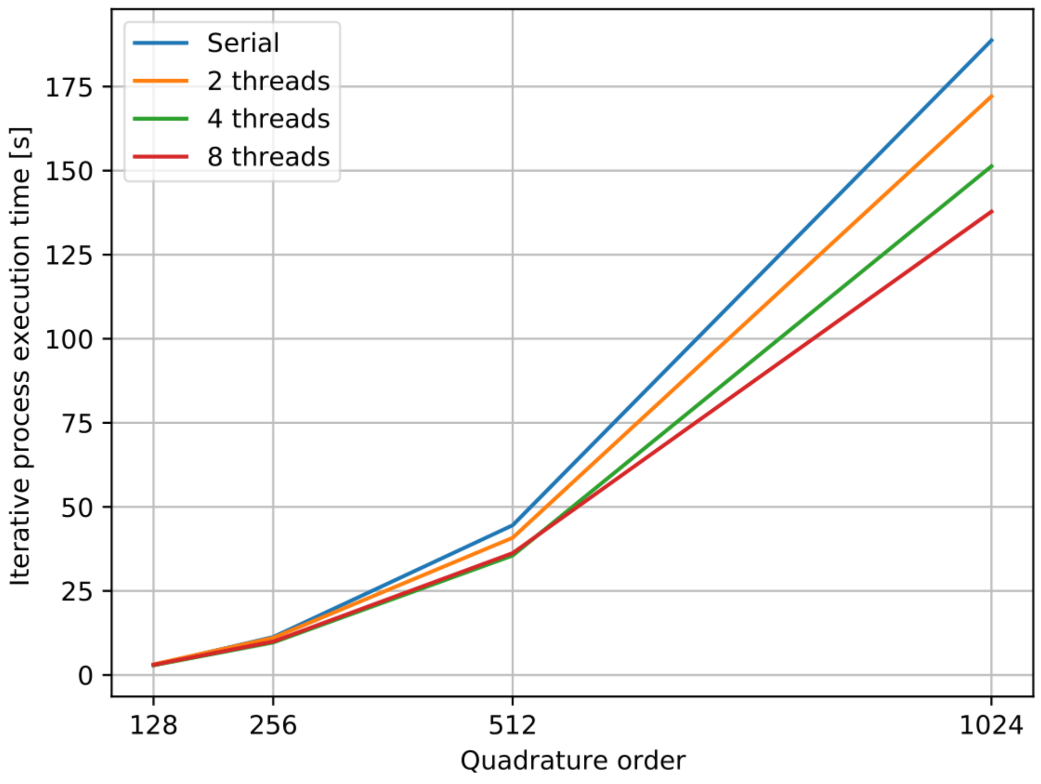
On the performance test, the algorithm was executed 50 times for each set of quadrature order and number of threads, and calculated the average execution time of the iterative process. Furthermore, a SpeedUp test was also made comparing the average execution times with 8 threads and the serial algorithm, as shown on Eq. (18). These results are shown in Table 3 and Figure 5.

$$SpeedUp = \frac{time_{8\ threads}}{time_{serial}} \tag{18}$$

Table 3: Model-problem 1 performance test numerical results.

	Average execution time $\pm$ Standard Deviation (s)				
N	$Serial$	$2\ threads$	$4\ threads$	$8\ threads$	$SpeedUp$
128	2.921300 $\pm$ 0.031304	3.150640 $\pm$ 0.036935	2.883300 $\pm$ 0.020201	3.014140 $\pm$ 0.018164	1.031
256	11.286360 $\pm$ 0.049192	11.047820 $\pm$ 0.104257	9.665620 $\pm$ 0.042579	9.546120 $\pm$ 0.052533	0.845
512	44.524640 $\pm$ 0.172734	40.814040 $\pm$ 0.224167	35.529580 $\pm$ 0.373609	36.248780 $\pm$ 0.872558	0.814
1024	188.748100 $\pm$ 0.508784	172.082360 $\pm$ 0.379614	151.311860 $\pm$ 1.741442	137.814820 $\pm$ 0.436255	0.730

Figure 5: Model-problem 1 performance test.

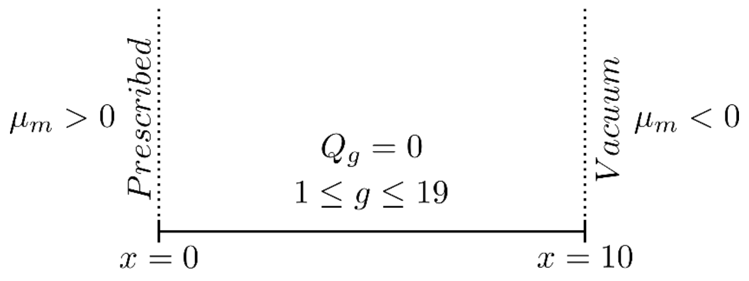


When analyzing the presented results in the performance test, it can be seen the decrease on the average execution time of the iterative process for the executions with Gauss-Legendre angular quadrature set order (N) equal $256, 512$ and 1024, when compared the 8 threads execution with the serial one

5.2 Model problem 2

The second model-problem, studies a $10\ cm$ homogeneous region, adapted from (Garvia & Siewert, 1981), with prescribed boundary conditions on the left with value $\psi_{m,g}(0) = 1\ cm^{-2}\ s^{-1}$ $(g = 1 : G$ and $m = 1{:}\,N/2)$, and vacuum on the right, without external neutron sources. The geometry, of this model-problem can be seen in Figure 6.

Figure 6: Model-problem 2.



The physical-material parameters are modelled with 19 energy groups, and are shown in Table 4

Table 4: Model-problem 2 physical-material data (cm^{-1}).

		$\sigma_{S0}^{g'\to g}$								
g	$\sigma_{T,g}$	g'=1	g'=2	g'=3	g'=4	g'=5	g'=6	g'=7	g'=8	g'=9
1	0.479679	0.029548	0	0	0	0	0	0	0	0
2	0.506558	0.059021	0.036618	0	0	0	0	0	0	0
3	0.533669	0.042185	0.055039	0.034362	0	0	0	0	0	0
4	0.560541	0.040667	0.052174	0.067969	0.041937	0	0	0	0	0
5	0.591740	0.039584	0.049683	0.063496	0.082364	0.052292	0	0	0	0
6	0.628809	0.039291	0.047935	0.059660	0.075624	0.101615	0.066948	0	0	0
7	0.665745	0.026723	0.031681	0.038304	0.047248	0.061780	0.087389	0.058025	0	0
8	0.701391	0.028002	0.032357	0.038040	0.045595	0.057782	0.079256	0.111110	0.071593	0
9	0.744831	0.030571	0.034431	0.039258	0.045470	0.055287	0.072442	0.097964	0.134869	0.090422
10	0.800421	0.035421	0.039042	0.043256	0.048329	0.055937	0.068787	0.087715	0.115277	0.166326
11	0.877230	0.044612	0.048593	0.052814	0.057356	0.063406	0.072554	0.085180	0.103204	0.137025
12	0.997659	0.060829	0.068782	0.074618	0.080320	0.086842	0.094663	0.103120	0.113162	0.130579
13	1.161747	0.000166	0.006341	0.016966	0.030045	0.048561	0.075192	0.091718	0.096676	0.102197
14	1.390244	0	0	0	0	0	0.000177	0.015297	0.045544	0.088399
15	1.854291	0	0	0	0	0	0	0	0	0
16	2.714682	0	0	0	0	0	0	0	0	0
17	4.407603	0	0	0	0	0	0	0	0	0
18	7.209129	0	0	0	0	0	0	0	0	0
19	11.24335	0	0	0	0	0	0	0	0	0

		$\sigma_{S0}^{g'\to g}$								
	g'=10	G' = 11	G'=12	G'=13	G'=14	G'=15	G'=16	G'=17	G'=18	G'=19
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0

6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0.117541	0	0	0	0	0	0	0	0	0
11	0.208661	0.158391	0	0	0	0	0	0	0	0
12	0.168081	0.267149	0.223428	0	0	0	0	0	0	0
13	0.110830	0.134363	0.229258	0.204165	0	0	0	0	0	0
14	0.176181	0.176181	0.201602	0.324419	0.273854	0	0	0	0	0
15	0.001544	0.066969	0.206765	0.333976	0.430387	0.383716	0	0	0	0
16	0	0	0	0.054499	0.263053	0.446045	0.399119	0	0	0
17	0	0	0	0	0	0.196781	0.651724	0.564511	0	0
18	0	0	0	0	0	0	0.033657	0.483268	0.459418	0
19	0	0	0	0	0	0	0	0.090778	0.724763	1.217800

At first, a precision test was performed. In this case, the scalar fluxes on the nodal interfaces were calculated using different Gaussian-Quadrature angular quadrature set orders (N) on the execution of the algorithm. The results for the scalar fluxes of groups 1, 10 and 19 are shown in Table 5, as well as a reference using method DD with 50000 nodes on the homogeneous region.

Table 5: Numerical results of the scalar neutron fluxes ($cm^{-2}s^{-1}$) for model-problem 2.

		MSD		DD	
N	G	0 cm	10 cm	0 cm	10 cm
32	1	5.079466×10^{-1}	7.049451×10^{-4}	5.079466×10^{-1}	7.049451×10^{-4}
	10	1.336406×10^{-2}	1.449657×10^{-4}	1.336406×10^{-2}	1.449657×10^{-4}
	19	4.178121×10^{-6}	6.877874×10^{-8}	4.178121×10^{-6}	6.877873×10^{-8}
64	1	5.079466×10^{-1}	7.049306×10^{-4}	5.079466×10^{-1}	7.049306×10^{-4}
	10	1.335576×10^{-2}	1.449500×10^{-4}	1.335576×10^{-2}	1.449500×10^{-4}
	19	4.182466×10^{-6}	6.885076×10^{-8}	4.182466×10^{-6}	6.885073×10^{-8}
128	1	5.079466×10^{-1}	7.049270×10^{-4}	5.079466×10^{-1}	7.049270×10^{-4}
	10	1.335325×10^{-2}	1.449465×10^{-4}	1.335325×10^{-2}	1.449465×10^{-4}
	19	4.183692×10^{-6}	6.887082×10^{-8}	4.183692×10^{-6}	6.887081×10^{-8}
256	1	5.079466×10^{-1}	7.049260×10^{-4}	5.079466×10^{-1}	7.049260×10^{-4}
	10	1.335252×10^{-2}	1.449457×10^{-4}	1.335252×10^{-2}	1.449457×10^{-4}
	19	4.184032×10^{-6}	6.887638×10^{-8}	4.184032×10^{-6}	6.887636×10^{-8}

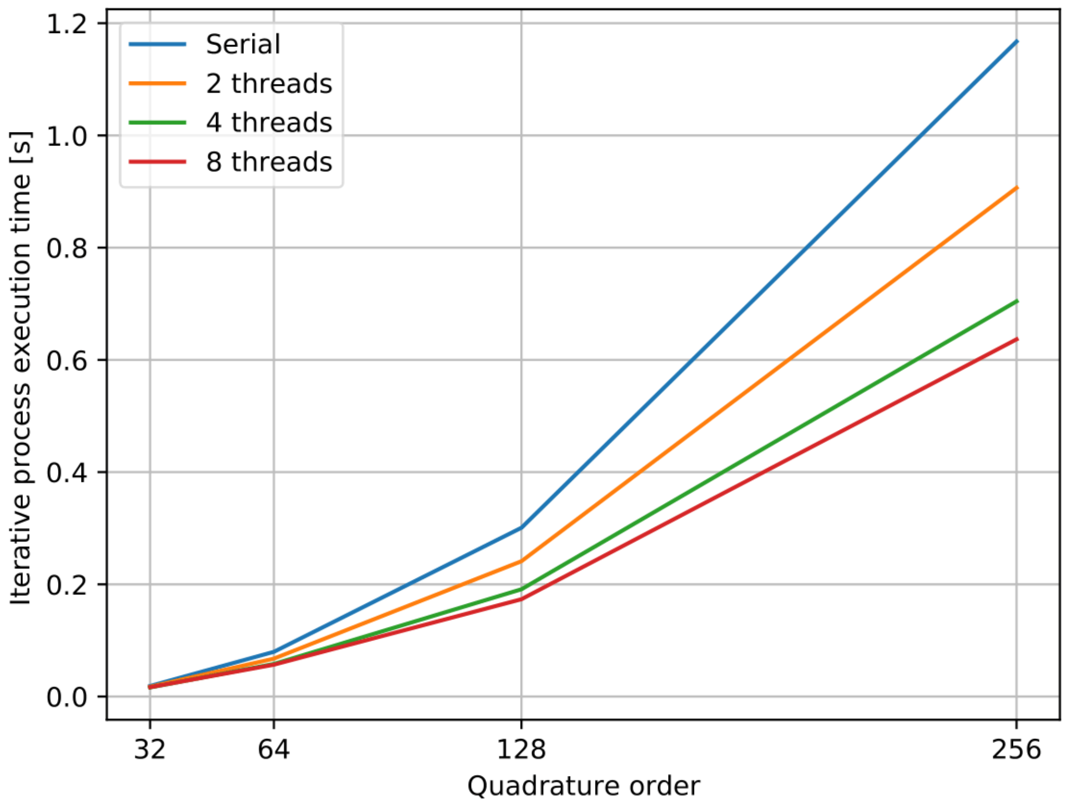
As in model-problem 1, 50 executions were made measuring the iterative process execution time, for each Gauss-Legendre order and number of threads set. The numerical results for this performance test are shown in Table 6, and the graphical results in Figure 7.

Table 6: Model-problem 2 performance test numerical results.

	Average execution time $\pm$ Standard Deviation (s)				
N	$Serial$	$2\ threads$	$4\ threads$	$8\ threads$	$SpeedUp$

32	0.018891 $\pm$ 0.000162	0.017399 $\pm$ 0.002774	0.016173 $\pm$ 0.002436	0.016558 $\pm$ 0.003178	0.876
64	0.079696 $\pm$ 0.001011	0.067611 $\pm$ 0.004633	0.058071 $\pm$ 0.003173	0.056747 $\pm$ 0.002344	0.712
128	0.300789 $\pm$ 0.000971	0.241159 $\pm$ 0.002428	0.191193 $\pm$ 0.002090	0.173445 $\pm$ 0.004194	0.576
256	1.167353 $\pm$ 0.038469	0.906578 $\pm$ 0.004714	0.704312 $\pm$ 0.010237	0.636645 $\pm$ 0.009315	0.545

Figure 7: Model-problem 2 performance test.



According to the results obtained in this performance test, the same behavior of the first model-problem can be seen. Because of the time spent dividing the work throughout the threads, problems with higher complexity tends to present faster execution time when comparing multi-thread executions with serial ones. It can be seen on the problems executed with 64, 128 and 256 gaussian quadrature orders. For the $N = 32$ problem, the fastest average execution time for the iterative process is found when used 4 threads, due to the smaller dimension matrixes used in the solution.

6. CONCLUSIONS AND FUTURE PERSPECTIVES

In this work, a parallelization process was performed on a spectral-nodal method applied in the solution of neutron shielding problems using the multigroup transport equation on the discrete ordinate formulation, using the OpenMP API. The numerical results of 2 model-problems were obtained using different number of threads and comparing the iterative process execution time.

When analyzing the obtained results, it can be concluded that the increase in the number of threads to solve a model-problem not necessarily decreases the execution time. This phenomenon happens in cases with smaller complexity, for example, the solution of the first model-problem with Gaussian-Legendre quadrature of order $N = 128$, and the second model-problem using $N = 32$. In these cases, the time needed to divide the tasks on the shared memory environment between the logical threads, and the solution using a bigger number of threads can be longer than the serial execution time.

On the numerical results presented in this work, it can be seen an enhancement on the average execution time in most cases. For the first model-problem solved using $N = 1024$, the iterative process was executed with about 27% less time when comparing the 8 thread and the serial solution, and for the second model-problem the multi-threading solution executed the iterative process with $N = 256$ in 54.5% of the serial execution time.

Considering the results achieved in this work, the group intends to extend the parallelization strategies on the solution of two-dimensional cartesian geometry problems, considering a higher anisotropy degree for the scattering phenomenon and an increase in the number of energy groups.

REFERENCES

BARROS, R. C. **A spectral nodal method for the solution of discrete ordinates problems in one- and two-dimensional cartesian geometry.** Tese de doutorado, University of Michigan, UMICH. Ann Arbor, 1990.

CASE, K. M., Zweifel, P. F. **Linear Transport Theory**. Massachusetts: 1 ed. Addison-Wesley, 1967.

DAGUM, L., Menon R. **OpenMP: na industry standard API for shared-memory**

programming. Computational Science & Engineering IEEE, v. 5, n. 1, p. 46-55.

Garcia, R. D. M. Siewert, C. E. **Mutigroup transport theory II: Numerical results.** Nuclear Science and engineering, v. 78, p. 315-323, 1981.

IBM. Using OpenMP directives. Disponível em <https://www.ibm.com/docs/en/xl-c-aix/13.1.0?topic=programs-using-openmp-directives>. Acesso em: 28 ago. 2021.

LEWIS, E. E., Miller, W. F. **Computational methods of neutron transport**. New York: 2 ed. Wiley, 1993.

LIBOTTE, R. B. **Método de malha grossa para solução numérica de problemas de blindagem de nêutrons em geometria unidimenional na formulação de ordenadas discretas com perspectivas a cálculos multidimensional em geometria retangular.** Dissertação de Mestrado, IPRJ/UERJ. Nova Friburgo, 2021.

MICROSOFT. Diretivas (OpenMP). Disponível em <https://docs.microsoft.com/ptbr/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160>. Acesso em: 28 ago. 2021.

OLIVA, A. M. **Método spectral Determinístico para a solução de problemas de transprte de nêutrons usando a formulação de ordenadas disctretas.** Tese de doutorado, IPRJ/UERJ. Nova Friburgo 2018.

OPENMP. OPENMP API Specifications: Version 5.0 November 2018. Disponível em <https://www.openmp.org/spec-html/5.0/openmpsu44.html>. Acesso em: 15 fev. 2022.

SILVA. O. P. **Um método de matriz resposta para cálculos de transporte multigrupos de energia na formulação de ordenadas discretas em meios não-multiplicativos.** Tese de Doutorado, IPRJ/UERJ. Nova Friburgo, 2018.