BJRS
BRAZILIAN JOURNAL OF RADIATION SCIENCES

doi.org/10.15392/2319-0612.2024.2498
202x, V(I) | 01-13 | e2498
Submitted: 2024-06-04
Accepted: 2025-06-17

Efficient Acceleration in Solving the 2D Neutron Diffusion Equation with CUDA: Exploring the Collaborative Practicality of Colab

Pessoa[a*], P.I.O.; Henrice[a], E.

[a] Eletronuclear, 20091-029, Rio de Janeiro, RJ, Brazil.

*Correspondence: ppessoa@eletronuclear.gov.br

Abstract: This paper explores an approach to accelerate the finite difference method applied to solving the two-dimensional (2D) neutron diffusion equation for two energy groups (2G) independent of time. The main innovation lies in the implementation of a performance optimization method, emphasizing the practicality of development in Python using direct browser collaboration through Google Colaboratory (Colab). Utilizing CUDA (Compute Unified Device Architecture) for GPU acceleration, we achieve significant computational performance improvements. The study compares Python implementations using CuPy and NumPy libraries with traditional FORTRAN implementations utilizing the LAPACK library, highlighting the efficiency and precision of GPU-accelerated calculations. Results show that Python with CuPy significantly outperforms NumPy, both in a Colab environment and on a personal desktop computer. This demonstrates the practicality of cloud-based solutions for intensive computations, as the ability to run code directly in the browser through Colab eliminates the need for extensive local hardware resources. The results emphasize the convenience of executing complex simulations without relying on physical computers, promoting greater flexibility and accessibility in computational research. All computational codes are available on GitHub for transparency and reproducibility.

Keywords: Neutron diffusion equation, NumPy, CuPy, Colab.

SBPR

BJRS
BRAZILIAN JOURNAL OF RADIATION SCIENCES

doi.org/10.15392/2319-0612.2024.2498
202x, V(I) | 01-13 | e2498
Submitted: 2024-06-04
Accepted: 2025-06-17

Aceleração eficiente na resolução da equação de difusão de nêutrons 2D com CUDA: explorando a praticidade colaborativa do Colab

Resumo: Este artigo explora uma abordagem para acelerar o método de diferenças finitas aplicado à resolução da equação de difusão de nêutrons bidimensional (2D) para dois grupos de energia (2G) independentes do tempo. A principal inovação está na implementação de um método de otimização de desempenho, enfatizando a praticidade do desenvolvimento em Python utilizando a colaboração direta do navegador através do Google Colaboratory (Colab). Utilizando CUDA (Compute Unified Device Architecture) para aceleração de GPU, alcançamos melhorias significativas de desempenho computacional. O estudo compara implementações Python usando bibliotecas CuPy e NumPy com implementações tradicionais de FORTRAN utilizando a biblioteca LAPACK, destacando a eficiência e precisão dos cálculos acelerados por GPU. Os resultados mostram que Python com CuPy supera significativamente o NumPy, tanto em um ambiente Colab quanto em um computador desktop pessoal. Isso demonstra a praticidade das soluções baseadas em nuvem para cálculos intensivos, já que a capacidade de executar código diretamente no navegador por meio do Colab elimina a necessidade de extensos recursos de hardware locais. Os resultados enfatizam a conveniência de executar simulações complexas sem depender de computadores físicos, promovendo maior flexibilidade e acessibilidade na pesquisa computacional. Todos os códigos computacionais estão disponíveis no GitHub para transparência e reprodutibilidade.

Palavras-chave: Equação de difusão de nêutrons, NumPy, CuPy, Colab.

SBPR

1. INTRODUCTION

This paper explores an approach to accelerate the finite difference method applied in solving the two-dimensional (2D) neutron diffusion equation at two energy groups (2G) and independent of time. The main innovation lies in the implementation of a method for performance optimization, with an emphasis on the practicality of development in Python using direct browser collaboration, through the Google Colaboratory environment (Colab).

The finite difference method [1] is a technique widely used in the simulation of physical phenomena, especially around nuclear reactors. However, the growing need for computational efficiency motivates the search for significant speedups. In this context, CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model created by NVIDIA, which will allow significant increases in computational performance by taking advantage of the power of the graphics-processing unit (GPU) to process data.

Other methods can also be accelerated using the same strategy. Among these, we can mention methods as neutron transport [2], finite differences [1], finite elements [3], nodal [4],[5]. Reconstruction methods [6],[7] are also on this list, highlighting the reconstruction methods that use finite differences [8] and finite elements [9],[10]. Research [15],[16] into Accelerator Driven System (ADS) type reactor can also be investigated.

The choice to implement the code in Python stands out for its accessibility and flexibility [11]. Python's CuPy library [12] is used for GPU acceleration and the NumPy library [13] is also used for testing in the work. Running directly in the browser through Colab provides development convenience and the ability to run code on any device with internet access. This democratizes access to the acceleration of complex calculations, making research more inclusive and collaborative.

To evaluate the effectiveness of the proposed method, we will compare the results and computational times obtained using Python in the browser and FORTRAN, the latter representing the conventional approach that uses the LAPACK library [14] to solve the system arising from finite difference discretization. Furthermore, a personal computer will be used as a comparison parameter, which highlights the importance of choosing hardware in optimizing performance.

This study not only seeks to improve computational efficiency in solving complex problems, but also seeks to democratize access to such advances. The intersection between Python, CUDA and online collaboration represents a significant step towards a more accessible and participatory community in scientific research and the simulation of physical phenomena.

2. METHODOLOGY

This study begins with the 2D neutron diffusion equation for two energy groups and in Cartesian geometry:

$$-D_1\nabla^2\phi_1(x,y) + \Sigma_{r,1}\phi_1(x,y) = \frac{1}{k}\sum_{g'=1}^{2} \nu\Sigma_{f,g'}\phi_{g'}(x,y) \tag{1}$$

and

$$-D_2\nabla^2\phi_2(x,y) + \Sigma_{r,2}\phi_2(x,y) = \Sigma_{2,1}\phi_1(x,y). \tag{2}$$

The solution of equations (1) and (2) provides the detailed flux distribution for the problem addressed, where k is the multiplication factor and $D_g^n, \Sigma_{r,g}^n, \nu\Sigma_{f,g}^n$ and $\Sigma_{g',g}^n$ the group constants that characterize the homogeneous region of the pin cell.

The discretization of equations 1 and 2 by finite differences centered on the interface leads to the following solution:

$$a_1^{i-1,j}\phi_1^{i-1,j} + a_1^{i+1,j}\phi_1^{i+1,j} + a_1^{i,j}\phi_1^{i,j} + a_1^{i,j-1}\phi_1^{i,j-1} + a_1^{i,j+1}\phi_1^{i,j+1} = \frac{1}{k}\left(f_1^{i,j}\phi_1^{i,j} + f_2^{i,j}\phi_2^{i,j}\right) \quad (3)$$

and

$$a_2^{i-1,j}\phi_2^{i-1,j} + a_2^{i+1,j}\phi_2^{i+1,j} + a_2^{i,j}\phi_2^{i,j} + a_2^{i,j-1}\phi_2^{i,j-1} + a_2^{i,j+1}\phi_2^{i,j+1} = q^{i,j}\phi_1^{i,j}. \quad (4)$$

The terms $a_g^{i-1,j}$, $a_g^{i+1,j}$, $a_g^{i,j}$, $a_g^{i,j-1}$, $a_g^{i,j+1}$, $f_1^{i,j}$, $f_2^{i,j}$ and $q^{i,j}$ are written as a function of the group constants and can be different depending on the region of the problem contour. We can completely represent the equations 3 and 4 by ordering the fluxes and forming vectors and matrices:

$$\boldsymbol{\phi_g} \equiv \left[\phi_g^{1,1}, \phi_g^{2,1}, \ldots, \phi_g^{N,1}, \ldots, \phi_g^{N,N}\right]^T \quad (5)$$

$$A_g \equiv pentadiagonal\left[a_g^{i,j-1}, a_g^{i-1,j}, a_g^{i,j}, a_g^{i+1,j}, a_g^{i,j+1}\right] \quad (6)$$

$$F_g \equiv diagonal\left[f_g^{1,1}, f_g^{2,1}, \ldots, f_g^{N,1}, \ldots, f_g^{N,N}\right] \quad (7)$$

and

$$Q \equiv diagonal[q^{1,1}, q^{2,1}, \ldots, q^{N,1}, \ldots, q^{N,N}]. \quad (8)$$

With these definitions, we can rewrite equations 3 and 4 in their discretized form:

$$A_1\boldsymbol{\phi_1} = \frac{1}{k}[F_1\boldsymbol{\phi_1} + F_2\boldsymbol{\phi_2}] \quad (9)$$

and

$$A_2\boldsymbol{\phi_2} = Q\boldsymbol{\phi_1}. \quad (10)$$

Grouping further we can rewrite the entire system of equations as

$$A\boldsymbol{\Psi} = \mathbf{s}, \quad (11)$$

with

$$\boldsymbol{\Psi} = [\boldsymbol{\phi_1}, \boldsymbol{\phi_2}]^T; \quad A = \begin{bmatrix} A_1 & 0 \\ -Q & A_2 \end{bmatrix}; \quad F = \begin{bmatrix} F_1 & F_2 \\ 0 & 0 \end{bmatrix}; \quad \boldsymbol{s} = \frac{1}{k}F\boldsymbol{\Psi}. \quad (12)$$

Then, we arrive at a system of linear equations represented by the form in equation (11), where A is the coefficient matrix, $\boldsymbol{\Psi}$ is the neutron flux vector and $\boldsymbol{s}$ is the source vector. Solving this system is essential to obtain information about the distribution of neutrons in the system under study.

3. RESULTS AND DISCUSSIONS

As a first test, we discretize the one-dimensional (1D) neutron diffusion equation for an energy group using finite differences. The 1D problem addressed is a homogeneous slab with 400 cm and zero flux boundary condition. The group constants for this problem are $D = 1.320$, $\Sigma_a = 0.041$, $\nu\Sigma_f = 0.045$ and the homogeneous region divided into 4000 meshes with the intention of making the calculations more difficult by increasing the matrix A (4000x4000) of the problem. The multiplication factor found in all simulations was $k = 1.0975610$. For a second test, we discretized the 2D neutron diffusion equation for two energy groups using finite differences. The problem 2D addressed is homogeneous with zero flux on the boundary, dimensions of $50x50\ cm$ and divided into $50x50$ meshes. The group constants for this problem are in Table 1. The multiplication factor found in all simulations was 0.7294215.

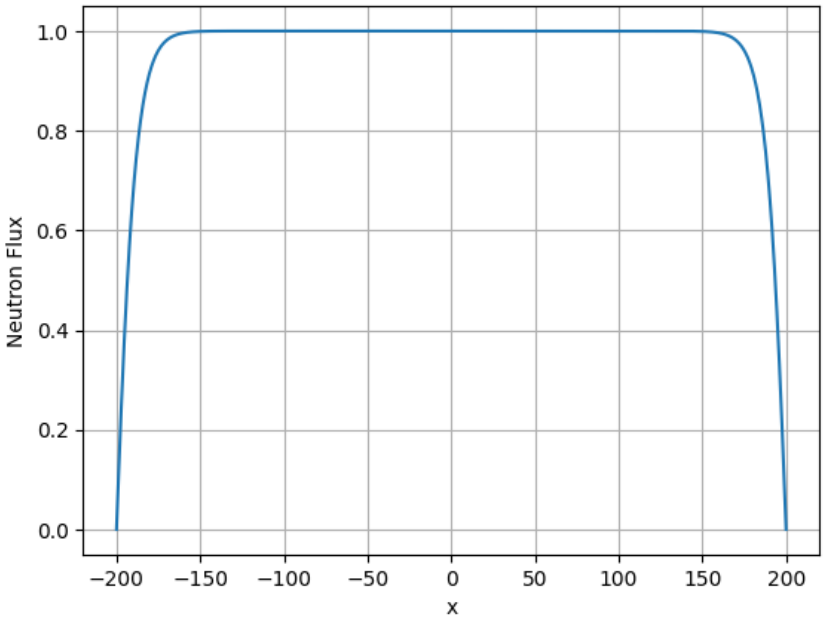
Table 1: Group constants for the 2D problem.

g	$\Sigma_{a,g}(cm^{-1})$	$\nu\Sigma_{f,g}(cm^{-1})$	$D_g(cm)$	$\Sigma_{g,g'}(cm^{-1})$
1	0.0130	0.0065	1.5000	0.0065
2	1.1800	0.2400	0.4000	0.0000

The 1D and 2D problems were run on the different computing environments, platforms and libraries using the group constants provided. Neutron fluxes were calculated and normalized to maximum neutron flux. Thus, Figures 1, 2 and 3 were generated,

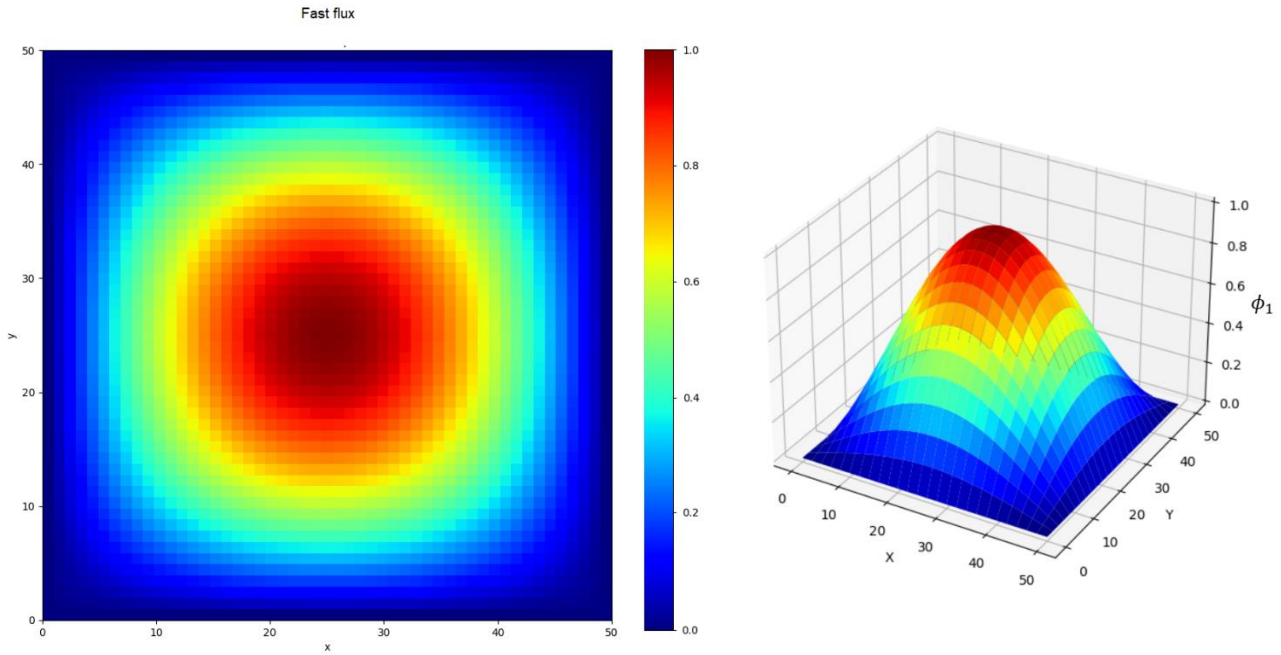
respectively, for the 1D problem and for the fast and thermal fluxes profiles of the 2D problem.

Figure 1: Neutron flux for the 1D problem.



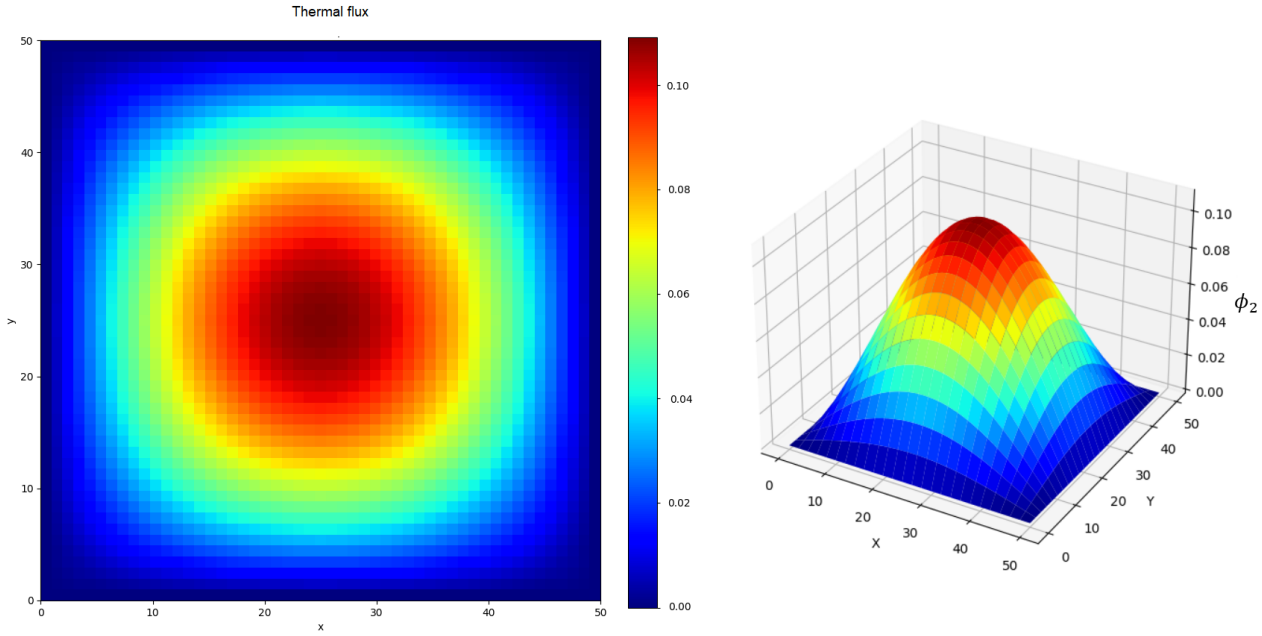
Source : Generated using Python with NumPy and Matplotlib libraries.

Figure 2: Fast neutron flux profiles for the 2D problem.



Source: Generated using Python with NumPy and Matplotlib libraries.

Figure 3: Thermal neutron flux profiles for the 2D problem.



Source: Generated using Python with NumPy and Matplotlib libraries.

Figures 1, 2 and 3 show the neutron fluxes of the 1D and 2D problems, which gives more clarity to the results obtained. It is worth mentioning that the errors associated with the neutron fluxes obtained by different environments and platforms are practically zero, highlighting the precision and reliability of the computational simulations.

Regarding the accuracy associated with mesh refinement, good numerical agreement is observed. In the 1D problem, with 8000 $(h = 0.05\,cm)$, 4000 $(h = 0.1\,cm)$ e 2000 $(h = 0.2\,cm)$ spatial divisions, the scalar neutron flux at a fixed point near the boundary consistently converges to approximately 0.36406, demonstrating high numerical stability. In the 2D case, using 25 $(h = 2{,}0\,cm)$, 50 $(h = 1{,}0\,cm)$ e 100 $(h = 0{,}5\,cm)$ divisions, variations in the scalar neutron flux at the same point remain confined to the third decimal place for both energy groups, while the effective multiplication factor shows differences only in the fourth decimal place.

Although the main goal of the article is to accelerate the solution process, the results obtained show excellent agreement between the analytical and numerical solutions for both problems analyzed. In the 1D problem, the analytical value of the effective multiplication factor is 1.0953856, while the numerical value is 1.0975610, resulting in a relative error of approximately 0.2%. In the 2D problem, the analytical value calculated was 0.7293548, and the corresponding numerical value was 0.7294215, with a relative error of less than 0.01%. These results demonstrate the accuracy of the implemented numerical solutions and validate the reliability of the proposed method.

Table 2 presents the computational times for executing the code in two different environments. In the first environment, a computer equipped with a 3.00GHz Intel Core i5-7400 processor, a GTX 980 Ti GPU with 2816 CUDA cores and 32 GB of RAM, the execution times for the 1D and 2D problems were, respectively, as follows: 1.33 s and 2,42 s for Fortran compiled with Intel Fortran 2020 and LaPACK library [14], 0.98 s and 10,67 for Python with the NumPy library [13] and 0.89 s and 9,65 s for Python with the CuPy library [12]. It is observed that for the 1D problem, on the same hardware, the Fortran implementation was slower compared to the Python version using the NumPy library. However, for the 2D problem, Fortran implementation was much faster than the Python implementation using the NumPy library, around 70% faster. Now, using the CuPy library resulted in shorter processing time for both problems in relation to the NumPy library, but the Fortran implementation proved to be faster.

On the other hand, when using the Colab environment directly from the browser, which has a 2.20GHz Intel Xeon CPU, a T4 GPU with 2560 CUDA cores and 12.7 GB of RAM, the execution times for the 1D and 2D problems were, respectively, 2.89 s and 51,12 for Python with NumPy and 0.79 and 7.51 for implementation with CuPy library. These results show that the CuPy implementation was much more efficient than NumPy.

Table 2: Computational times for 1D and 2D problems.

Methods	$t_1(s)$ 1D problem	$t_2(s)$ 2D problem
Fortran (CPU)	1.33	2.42
Python with NumPy (CPU)	0.98	10.67
Python with CuPy (GTX 980 TI)	0.89	9.65
Python with NumPy (Colab)	2.89	51.12
Python with CuPy (Colab)	0.79	7.51

These results highlight the importance of choosing the programming language and libraries used, as well as the influence of available hardware. Although Fortran has shown good performance, especially compared to pure Python, using libraries like CuPy can offer a significant advantage in terms of computational efficiency, especially when it comes to GPU-intensive calculations. The difference in performance between local hardware and the Colab environment also highlights the importance of considering the characteristics of the execution environment when performing performance analysis.

4. CONCLUSIONS

In summary, our tests showed significant variations in computational performance across different programming languages, libraries, and hardware environments when implementing the neutron diffusion equation. Although Fortran performed better, Python with the CuPy library proved to be efficient and useful, mainly due to the ease of accessing the software and hardware provided directly through the browser Implementation with the CuPy library led to over 70% reduction in execution time compared to the NumPy library on the Colab platform. On the personal computer, the reduction was approximately 10% for the 1D and 2D problems. This demonstrates the effectiveness of GPU acceleration in handling computationally intensive calculations such as neutron diffusion problems.

Furthermore, the Colab environment provided a convenient platform for computational experiments without the need for extensive local hardware resources. The Colab platform can also offer the option to purchase additional computing power. Leveraging Colab's computing power remotely highlights the importance of cloud-based solutions. This accessibility promotes collaboration and innovation in the scientific community.

Looking to the future, it is important to note that this work will be extended to solve the 2D and 2G neutron diffusion equation for heterogeneous problems, which could simulate commercial nuclear reactors. Future research could explore this acceleration technique in other methods in the nuclear area, application in three-dimensional geometries and time-dependent problems.

All computational codes mentioned in this article will be publicly available in the GitHub repository [https://github.com/Nuclear2024/Efficient-Acceleration-in-Solving-the-2D-Neutron-Diffusion-Equation-with-CUDA.git] to ensure collaboration, transparency and reproducibility of this research.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Eletronuclear for granting us the necessary permission to conduct this work. Your collaboration and support were fundamental to the success of this project. We are deeply grateful for your generosity and trust.

REFERENCES

[1] Derstine, K.L. DIF3D: a code to solve one-, two-, and three-dimensional finite-difference diffusion theory problems. **ANL**- Technical Report-82-64. United States, 1984.

[2] Fletcher, J.K. "The solution of the time-independent multi-group neutron transport equation using spherical harmonics". **Annals of Nuclear Energy**, vol. 4, pp. 401–405, 1977.

[3] Araujo, L.M.; Carmo, E.G.D.; Silva, F.C. Galerkin partial least-square SN (GpLS - SN) method for fixed source problems in the neutrons transport theory. **Annals of Nuclear Energy**. vol. 90, pp. 175-190, 2016.

[4] Finnemann, H.; Bennewitz, F.; Wagner, M.R. Interface current techniques for multidimensional reactor calculations. **Atomkernenergie**. vol. 30, pp. 123-128, 1977.

[5] Silva, A.; Pessoa, P.O.; Silva, F.C.; Martinez, A.S. Two-dimensional analytical solution for nodal calculation of nuclear reactors. **Annals of Nuclear Energy**, vol. 100, pp. 76-81, 2016.

[6] Pessoa, P.O.; Silva, F.C.; Martinez, A.S. Analytical method of pin-by-pin reconstruction of the nuclear power density distribution. *In*: INTERNATIONAL NUCLEAR ATLANTIC CONFERENCE – **INAC**. PE, Brazil, 2013.

[7] Pessoa, P.O.; Silva, F.C.; Martinez, A.S. Methods for reconstruction of the density distribution of nuclear power. **Annals of Nuclear Energy**, vol. 83, pp. 76–86, 2015.

[8] Pessoa, P.O.; Silva, F.C.; Martinez, A.S. Finite difference applied to the reconstruction method of the nuclear power density distribution. **Annals of Nuclear Energy**, vol. 92, pp. 378–390, 2016.

[9] Pessoa, P.O.; Araujo, L.M.; Silva, F.C. A strategy for pin power reconstruction based on classic Galerkin variational formulation. **Progress in Nuclear Energy**, vol. 104, pp. 251–263. 2018.

[10] Pessoa, P.O.; Araujo, L.M.; Silva, F.C. Numerical methods applied to pin power reconstruction based on coarse-mesh nodal calculation. **Annals of Nuclear Energy**, vol. 118, pp. 291–312, 2018.

[11] Pessoa, P.O. and Henrice, E. J. Efficient Acceleration in Solving the 2D Neutron Diffusion Equation with CUDA: Exploring the Collaborative Practicality of Colab. In: INTERNATIONAL NUCLEAR ATLANTIC CONFERENCE – INAC. RJ, Brazil, 2024.

[12] Okuta, R.; Unno, Y.; Nishino, D.; Hido, S.; Loomis, C. CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. In: 31ST CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (**NIPS 2017**), Long Beach, CA, USA, 2017.

[13] Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M.H.; Wieser, E.; Brett, M.; Haldane, A.; Del Rio, J.F.; Wiebe, M.; Peterson, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; Oliphant, T.E. Array programming with NumPy. **Nature**, vol. 585, pp. 357-362, 2020.

[14] Anderson, E.; Bai, Z.; Bischof, C.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; Sorensen, D. LAPACK Users' Guide. **Society for Industrial and Applied Mathematics** (SIAM), Philadelphia, PA, 1999.

[15] Henrice, E.; Palma, D.A.P.; Gonçalves, A.C. Online identification of trips caused by the external proton source in an ADS reactor. **Nuclear Engineering and Design**, vol. 383, pp. 111-419, 2021.

[16] Henrice, E.; Palma, D.A.P.; Gonçalves, A.C.; Mesquita, A.Z. Support to the identification of anomalies in an external neutron source using Hurst Exponents. **Progress in Nuclear Energy**, vol. 99, pp. 119-126, 2017.

LICENSE