Mathematical and Computational Applications

MDPI

Article

Block Preconditioning Matrices for the Newton Method to Compute the Dominant λ-Modes Associated with the Neutron Diffusion Equation

Amanda Carreño [1,*], Luca Bergamaschi [2], Angeles Martinez [3], Antoni Vidal-Ferrándiz [1], Damian Ginestar [4] and Gumersindo Verdú [1]

1 Instituto Universitario de Seguridad Industrial, Radiofísica y Medioambiental, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain; anvifer2@upvnet.upv.es (A.V.-F.); gverdu@iqn.upv.es (G.V.)
2 Department of Civil Environmental and Architectural Engineering, University of Padua, Via 8 Febbraio, 2, 35122 Padua, Italy; luca.bergamaschi@unipd.it
3 Department of Mathematics "Tullio Levi-Civita", University of Padua, Via 8 Febbraio, 2, 35122 Padua, Italy; acalomar@math.unipd.it
4 Instituto Universitario de Matemática Multidisciplinar, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain; dginesta@mat.upv.es
* Correspondence: amcarsan@iqn.upv.es

Received: 30 November 2018; Accepted: 10 January 2019; Published: 15 January 2019

Abstract: In nuclear engineering, the λ-modes associated with the neutron diffusion equation are applied to study the criticality of reactors and to develop modal methods for the transient analysis. The differential eigenvalue problem that needs to be solved is discretized using a finite element method, obtaining a generalized algebraic eigenvalue problem whose associated matrices are large and sparse. Then, efficient methods are needed to solve this problem. In this work, we used a block generalized Newton method implemented with a matrix-free technique that does not store all matrices explicitly. This technique reduces mainly the computational memory and, in some cases, when the assembly of the matrices is an expensive task, the computational time. The main problem is that the block Newton method requires solving linear systems, which need to be preconditioned. The construction of preconditioners such as ILU or ICC based on a fully-assembled matrix is not efficient in terms of the memory with the matrix-free implementation. As an alternative, several block preconditioners are studied that only save a few block matrices in comparison with the full problem. To test the performance of these methodologies, different reactor problems are studied.

Keywords: block preconditioner; generalized eigenvalue problem; neutron diffusion equation; modified block Newton method

1. Introduction

The neutron transport equation is a balance equation that describes the behavior of the neutrons inside the reactor core. This equation for three-dimensional problems is an equation defined in a phase space of dimension seven, and this makes the problem very difficult to solve. Thus, some approximations are considered such as the multigroup neutron diffusion equation by relying on the assumption that the neutron current is proportional to the gradient of the neutron flux by means of a diffusion coefficient.

Given a configuration of a nuclear reactor core, its criticality can be forced by dividing the production operator in the neutron diffusion equation by a positive number, λ, obtaining a neutron

balance equation: the λ-modes problem. For the two energy groups approximation and without considering up-scattering, this equation can be written as [1]:

$$\begin{pmatrix} -\vec{\nabla}(D_1\vec{\nabla}) + \Sigma_{a_1} + \Sigma_{12} & 0 \\ -\Sigma_{12} & -\vec{\nabla}(D_2\vec{\nabla}) + \Sigma_{a_2} \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \frac{1}{\lambda} \begin{pmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}, \tag{1}$$

where ϕ_1 and ϕ_2 denote the fast and thermal flux, respectively. The macroscopic cross-sections D_g, Σ_{ag}, and $\nu\Sigma_{fg}$, with $g = 1, 2$, and $\Sigma_{1,2}$, are values that depend on the position.

The largest eigenvalue in magnitude, called the k-effective (or multiplication factor), indicates a measure of the criticality of the reactor, and its corresponding eigenfunction describes the steady-state neutron distribution in the reactor core. Next, dominant eigenvalues and their corresponding eigenfunctions are useful to develop modal methods for the transient analysis.

To make a spatial discretization of Problem (1), a high order continuous Galerkin finite element method is used, leading to a generalized algebraic eigenvalue problem of the form:

$$\begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} \tilde{\phi}_1 \\ \tilde{\phi}_2 \end{pmatrix} = \frac{1}{\lambda} \begin{pmatrix} M_{11} & M_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{\phi}_1 \\ \tilde{\phi}_2 \end{pmatrix}, \tag{2}$$

where the matrix elements are given by:

$$\begin{aligned} L_{ij} = \sum_{e=1}^{N_t} \Bigg(& D_1 \int_{\Omega_e} \vec{\nabla} N_{1i} \cdot \vec{\nabla} N_{1j} \, \mathrm{d}V - D_1 \int_{\Gamma_e} N_{1i} \vec{\nabla} N_{1j} \, \mathrm{d}\vec{S} + D_2 \int_{\Omega_e} \vec{\nabla} N_{2i} \cdot \vec{\nabla} N_{2j} \, \mathrm{d}V \\ & - D_2 \int_{\Gamma_e} N_{2i} \vec{\nabla} N_{2j} \, \mathrm{d}V + (\Sigma_{a1} + \Sigma_{12}) \int_{\Omega_e} N_{1i} N_{1j} \, \mathrm{d}V + \Sigma_{a2} \int_{\Omega_e} N_{2i} N_{2j} \, \mathrm{d}V \\ & - \Sigma_{12} \int_{\Omega_e} N_{2i} N_{1j} \, \mathrm{d}V \Bigg), \\ M_{ij} = \sum_{e=1}^{N_t} \Bigg(& \nu\Sigma_{f1} \int_{\Omega_e} N_{1i} N_{1j} \, \mathrm{d}V + \nu\Sigma_{f2} \int_{\Omega_e} N_{1i} N_{2j} \, \mathrm{d}V \Bigg), \end{aligned}$$

where N_i is the prescribed shape function for the ith node. The vector $\tilde{\phi} = (\tilde{\phi}_1, \tilde{\phi}_2)^{\mathsf{T}}$ is the algebraic vector of the finite weights corresponding to the neutron flux in terms of the shape functions. The shape functions used in this work are Lagrange polynomials. The subdomains Ω_e ($e = 1, \ldots, N_t$) denote the cells in which the reactor domain is divided and where the cross-sections are assumed to be constant. Similarly, Γ_e is the corresponding subdomain surface, which is part of the reactor boundary. More details on the finite element discretization can be found in [2]. For the implementation of the finite element method, the open source finite elements library Deal.II [3] has been used.

In this work, a matrix-free strategy for the blocks of the matrix M and for the non-diagonal blocks of L is developed. In this way, matrix-vector products are computed on the fly in a cell-based interface. For instance, we can consider that a finite element Galerkin approximation that leads to the matrix $M_{1,1}$ takes a vector u as input and computes the integrals of the operator multiplied by trial functions, and the output vector is v. The operation can be expressed as a sum of N_t cell-based operations,

$$v = M_{1,1}u = \sum_{e=1}^{N_t} P_e^T M_{1,1}^e P_e u, \tag{3}$$

where P_e denotes the matrix that defines the location of cell-related degrees of freedom in the global vector and $M_{1,1}^e$ denotes the submatrix of $M_{1,1}$ on finite element e. This sum is optimized through sum-factorization. Details about the implementation are explained in [4]. This strategy greatly reduces the memory used by the matrix elements.

Calculation of the dominant lambda mode has traditionally utilized the classical power iteration method, which although robust, converges slowly for dominance ratios near one, as occurs in some practical problems. Thus, acceleration techniques are needed to improve the convergence of the power iteration method. Some approaches in diffusion theory are, for instance, Chebyshev iteration [5] and Wielandt shift [6]. Alternative approaches to the power iteration method have been studied in an attempt to improve upon the performance of accelerated power iteration methods [7,8]. The subspace iteration method [9], the Implicit Restarted Arnoldi method (IRAM) [10], the Jacobi–Davidson [11], and the Krylov–Schur method [2] implemented in the SLEPclibrary [12] have been used to compute the largest or several dominant eigenvalues for the neutron diffusion equation and their corresponding eigenfunctions. More recently, other Krylov methods have been used to compute these modes for other approximations of the neutron transport equation [7,13]. Usually, applying these kinds of methods requires either transforming the generalized problem (2) into an ordinary eigenvalue problem or applying a shift and invert technique. In both cases, in the solution process, it is necessary to solve numerous linear systems. These systems are not well-conditioned, and they need to be preconditioned. Thus, the time and computational memory needed to compute several eigenvalues become very high.

One alternative is to use a method that does not require solving any linear system, such as the generalized Davidson, used for neutron transport calculations in [14]. Other methods are the block Newton methods that have been shown to be very efficient in the computation of several eigenvalues in neutron diffusion theory. These methods either do not need to solve as many linear systems as the Krylov methods or avoid solving any linear system with some hybridization. One of these Newton methods is the modified block Newton method, which has been considered for the ordinary eigenvalue problem associated with Problem (2) [15] or directly for the generalized eigenproblem (2) [16]. One advantage of these block methods is that several eigenvectors can be approximated simultaneously, and as a consequence, the convergence behavior improves. The convergence of the eigensolvers usually depends on the eigenvalue separation, and if there are clustered or multiple eigenvalues, the methods may have problems finding all the eigenvalues. In practical situations of reactor analysis, the dominance ratio corresponding to the dominant eigenvalues is often near unity, resulting in a slow convergence. In the block methods, this convergence only depends on the separation of the group of target eigenvalues from the rest of the spectrum. Another advantage is that these methods do not require solving as many linear systems as the previous methods. However, these linear systems still need to be preconditioned. Another of this kind of Newton method is the Jacobian-free Newton–Krylov methods that have been studied with traditional methods such as the power iteration used as the preconditioner [17,18] or with a more sophisticated Schwarz preconditioner [19]. In this work, we use the Modified Generalized Block Newton Method (MGBNM) presented in [16], and we propose several ways to precondition the linear systems that need to be solved in this method in an efficient way.

The structure of the rest of the paper is as follows. In Section 2, the modified generalized block Newton method is described. In Section 3, the different preconditioners for the MGBNM are presented. The performance of the preconditioners is presented in Section 4 for two different benchmark problems. Finally, Section 5 synthesizes the main conclusions of this work.

2. The Modified Generalized Block Newton Method

This method was presented by Lösche in 1998 [20] for ordinary eigenvalue problems, and an extension to generalized eigenvalue problems was studied in [16]. Given the partial generalized eigenvalue problem (2) written as:

$$MX = LX\Lambda, \tag{4}$$

where $X \in \mathbb{R}^{n\times q}$ is a matrix with q eigenvectors and $\Lambda \in \mathbb{R}^{q\times q}$ is a diagonal matrix with the q eigenvalues associated, we suppose that the eigenvectors can be factorized as $X = ZS$, where Z is

an orthogonal matrix. Moreover, the biorthogonality condition $W^{\mathsf{T}}Z = I$ is introduced, where W is a fixed matrix. Thus, if we denote $K = S\Lambda S^{-1}$, the problem (4) can be rewritten as:

$$MX = LX\Lambda \Leftrightarrow MZ = LZS\Lambda S^{-1} \Leftrightarrow MZ = LZK.$$

We construct this projection to ensure that the method converges to independent eigenvectors.

Then, the solution of Problem (4) is obtained by solving the non-linear problem:

$$F(Z,\Lambda) := \begin{pmatrix} MZ - LZK \\ W^T Z - I_q \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{5}$$

By applying Newton's method, a new iterated solution arises as:

$$Z^{(k+1)} = Z^{(k)} - \Delta Z^{(k)}, \qquad K^{(k+1)} = K^{(k)} - \Delta K^{(k)}, \tag{6}$$

where $\Delta Z^{(k)}$ and $\Delta K^{(k)}$ are solutions of the system obtained when the equations (6) are substituted into the equations (5), and these are truncated at the first order terms.

The matrix $K^{(k)}$ is not necessarily a diagonal matrix, and as a result, the system is coupled. To avoid this problem, the modified generalized block Newton method (MGBNM) needs to apply the previous two steps. Firstly, the modified Gram–Schmidt process is used to orthonormalize the matrix $Z^{(k)}$. Then, the Rayleigh–Ritz projection method for the generalized eigenvalue problem [21] is applied. Thus, $\Delta Z^{(k)} = (\Delta z_1^{(k)}, \ldots, \Delta z_q^{(k)})$, where $\Delta z_i^{(k)} \in \mathbb{R}^n$ and $\Delta K^{(k)} = (\Delta k_1^{(k)}, \ldots, \Delta k_q^{(k)})$ where $\Delta k_i^{(k)} \in \mathbb{R}$ are obtained from the solutions of the linear systems:

$$\begin{pmatrix} M - \lambda_i^{(k)} L & LZ^{(k)} \\ {Z^{(k)}}^{\mathsf{T}} & 0 \end{pmatrix} \begin{pmatrix} \Delta z_i^{(k)} \\ -\Delta k_i^{(k)} \end{pmatrix} = \begin{pmatrix} Mz_i^{(k)} - Lz_i^{(k)}\lambda_i^{(k)} \\ 0 \end{pmatrix}, \quad i = 1, \ldots, q.$$

The solution of these systems is computed by using the Generalized Minimal Residual method (GMRES) computing the matrix vector products with block matrix multiplications. However, these systems need to be preconditioned (in each iteration and for each eigenvalue) to reduce the condition number of the matrix.

3. Preconditioning

The first choice for a preconditioner is assembling the matrix:

$$A = \begin{pmatrix} M - \lambda_i^{(k)} L & LZ^{(k)} \\ {Z^{(k)}}^{\mathsf{T}} & 0 \end{pmatrix},$$

and constructing the full preconditioner associated with the matrix. We use the ILUT(0) preconditioner since A is a non-symmetric matrix. There are no significant differences if the preconditioner obtained for the matrix associated with the first eigenvalue is used for all eigenvalues in the same iteration because in the matrix, A only changes the value of $\lambda_i^{(k)}$, and usually, the eigenvalues in reactor problems are clustered. This preconditioner is denoted by P.

To devise an alternative preconditioner without the necessity of assembling the matrix A, we write the explicit inverse of A, by using its block structure,

$$A^{-1} = \begin{pmatrix} J^{-1}(I - C_1(C_2^{\mathsf{T}}C_1)^{-1}C_2^{\mathsf{T}}) & J^{-1}C_1(C_2^{\mathsf{T}}C_1) \\ (C_2^{\mathsf{T}}C_1)^{-1}C_2^{\mathsf{T}} & -(C_2^{\mathsf{T}}C_1)^{-1} \end{pmatrix},$$

where:

$$J = M - \lambda_i L, \quad C_1 = LZ, \quad C_2^{\mathsf{T}} = Z^{\mathsf{T}} J^{-1}.$$

We desire a preconditioner for A by suitably approximating A^{-1}. Let us call P_J a preconditioner for J. For instance, $P_J = (LU)^{-1}$, where L, U are the incomplete L and U factors of J. Thus, we can define, after setting $C_2^{\mathsf{T}} = Z^{\mathsf{T}} P_J$, the preconditioner of A as:

$$\hat{P}_J = \begin{pmatrix} P_J(I - C_1(C_2^{\mathsf{T}} C_1)^{-1} C_2^{\mathsf{T}}) & P_J C_1 (C_2^{\mathsf{T}} C_1) \\ (C_2^{\mathsf{T}} C_1)^{-1} C_2^{\mathsf{T}} & -(C_2^{\mathsf{T}} C_1)^{-1} \end{pmatrix}.$$

The previous preconditioner does not need to assemble the entire matrix A, but it needs to assemble the matrix J to build its ILUpreconditioner. Therefore, the next alternative that we propose is using a preconditioner of $-L$ instead of $J = M - \lambda_1 L$. This preconditioner works well because in the discretization process, the L matrix comes from the discretization of the differential matrix that has the gradient operators and the diffusion terms. In addition, in nuclear calculations, λ_1 is near 1.0. Thus, we can build a preconditioner of $-L$ instead of the matrix J. We denote by $\hat{P}_L$ the preconditioner $\hat{P}_J$ where the preconditioner of $-L$ is used to precondition the block J.

Finally, the last alternative is avoiding assembling the matrix L taking advantage of its block structure. For that purpose, we carry out a similar process as the one used for matrix A. We write the explicit form of the inverse of L as:

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1} L_{21} L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}, \tag{7}$$

and substitute the inverses of the blocks by preconditioners. Thus, the preconditioner of L has the following structure:

$$Q_L = \begin{pmatrix} P_{11} & 0 \\ -P_{22} L_{21} P_{11} & P_{22} \end{pmatrix},$$

where P_{11}, P_{22} denote a preconditioner of L_{11} and L_{22}, respectively. The block matrices L_{11}, L_{22} are symmetric and positive definite. Then, we can use as preconditioner the Incomplete Cholesky decomposition (IC(0)). However, the main advantage of this preconditioner is that it permits using a matrix-free implementation that does not require allocating all matrices. We only need to assemble the blocks L_{11} and L_{22} to construct the associated IC(0) preconditioners. The application of $\hat{P}_J$ with $-Q_L$ to precondition J is called $\hat{P}_Q$.

4. Numerical Results

In this section, the performance of the proposed preconditioners has been tested on two different problems: a version of the 3D NEACRPreactor [22] and a configuration of the Ringhals reactor [23]. The neutron diffusion equation in both problems has been discretized using the finite element method presented in Section 1 using Lagrange polynomials of degree three because it is shown in previous works that this degree is necessary to obtain accurate results in similar reactor problems [2]. The number of eigenvalues computed was four for each reactor.

The incomplete lower-upper preconditioner with Level 0 of fill (ILU) has been provided by the PETScpackage [24].

As the modified generalized block Newton method needs an initial approximation of a set of eigenvectors, a multilevel initialization with two meshes was used to obtain this approximation (for more details, see [25]).

The stopping criteria for all solvers has been set equal to 10^{-6} in the global residual error,

$$\varepsilon_{\text{res}} = \max_{i=1,\dots,q} \|Mx_i - \lambda_i Lx_i\|^2 ,$$

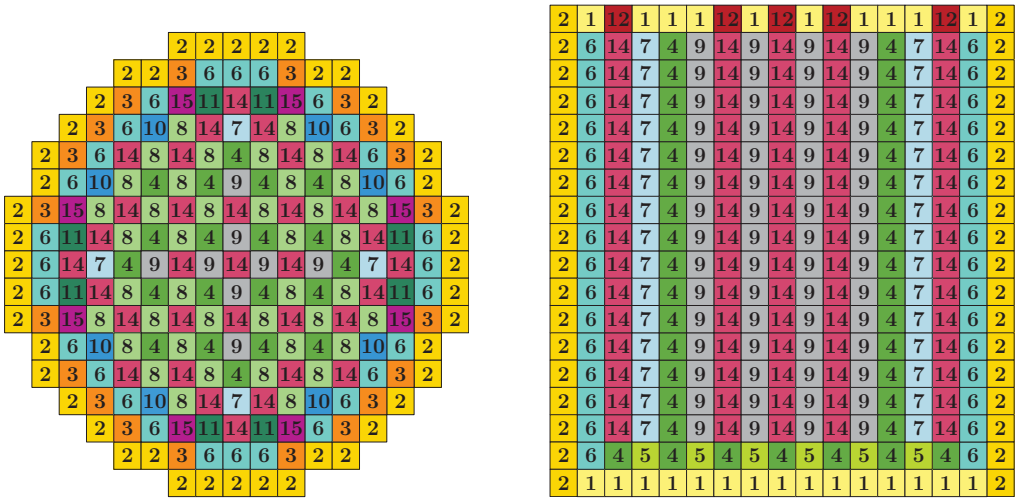
where λ_i is the i^{th} eigenvalue and x_i its associated eigenvector such that $\|x_i\| = 1$.

The modified block Newton method has been implemented using a dynamic tolerance in the residual error of the solution in the linear systems. The tolerance values have been set to $\{10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}, 10^{-8}, \dots\}$.

The methods have been implemented in C++ based on the data structures provided by the library Deal.ii [3] and PETSc [24]. The computer used for the computations was an Intel® Core™i7-4790 @3.60 GHz with 32 Gb of RAM running on Ubuntu GNU/Linux 16.04 LTS.

4.1. NEACRP Reactor

The NEACRP benchmark in a near-critical configuration [22] is chosen to compare the proposed methodology. The reactor core has a radial dimension of 21.606 cm × 21.606 cm per assembly. Axially, the reactor, with a total height of 427.3 cm, is divided into 18 layers with height (from bottom to top): 30.0 cm, 7.7 cm, 11.0 cm, 15.0 cm, 30.0 cm (10 layers), 12.8 cm (two layers), 8.0 cm, and 30.0 cm. Figure 1 shows the reactor geometry and the distribution of the different materials. The cross-sections of materials are displayed in Table 1. The total number of cells of the reactor domain is 3978. Zero flux boundary conditions were set. The spatial discretization of the neutron diffusion equation, by using polynomials of degree three, gave a number of 230,120 degrees of freedom. The mesh built to obtain an initial guess had 1308 cells, and the computational time needed to obtain this approximation was 24 s. The four dominant eigenvalues computed are collected in Table 2. This table shows that the spectrum associated with the problem is clustered with two degenerated eigenvalues. A representation of the fast flux distribution for each mode is displayed in Figure 2.



(**a**) Top view (**b**) Profile view

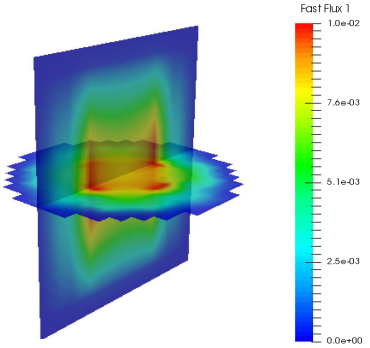
Figure 1. Geometry and distribution of the materials of the NEACRPreactor.

Table 1. Macroscopic cross-section of the NEACRP reactor.

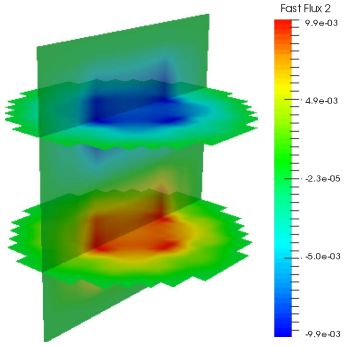
Mat.	D_1 (cm)	D_2 (cm)	Σ_{a1} (cm^{-1})	Σ_{a2} (cm^{-1})	Σ_{12} (cm^{-1})	$\nu\Sigma_{f1}$ (cm^{-1})	$\nu\Sigma_{f2}$ (cm^{-1})
1	5.9264	8.2289×10^{-1}	2.5979×10^{-4}	1.7085×10^{-1}	2.7988×10^{-2}	0.0000	0.0000
2	1.1276	1.7053×10^{-1}	1.1878×10^{-3}	1.9770×10^{-1}	2.3161×10^{-2}	0.0000	0.0000
3	1.1276	1.7053×10^{-1}	1.1878×10^{-3}	1.9770×10^{-1}	2.0081×10^{-2}	0.0000	0.0000
4	1.4624	3.9052×10^{-1}	8.4767×10^{-3}	6.2569×10^{-2}	1.9686×10^{-2}	5.0150×10^{-3}	8.7712×10^{-2}
5	1.4637	3.9485×10^{-1}	8.8225×10^{-3}	6.9978×10^{-2}	1.9436×10^{-2}	5.6085×10^{-3}	1.0424×10^{-1}
6	1.4650	3.9851×10^{-1}	9.1484×10^{-3}	7.6850×10^{-2}	1.9196×10^{-2}	6.1819×10^{-3}	1.1954×10^{-1}
7	1.4641	4.0579×10^{-1}	9.0869×10^{-3}	7.7687×10^{-2}	1.8526×10^{-2}	5.5830×10^{-3}	1.0289×10^{-1}
8	1.4642	4.0946×10^{-1}	9.1738×10^{-3}	8.0302×10^{-2}	1.8223×10^{-2}	5.5741×10^{-3}	1.0232×10^{-1}
9	1.4642	4.1314×10^{-1}	9.2596×10^{-3}	8.2924×10^{-2}	1.7920×10^{-2}	5.5650×10^{-3}	1.0169×10^{-1}
10	1.4653	4.0919×10^{-1}	9.4097×10^{-3}	8.4462×10^{-2}	1.8288×10^{-2}	6.1564×10^{-3}	1.1807×10^{-1}
11	1.4655	4.1277×10^{-1}	9.4956×10^{-3}	8.7030×10^{-2}	1.7986×10^{-2}	6.1474×10^{-3}	1.1744×10^{-1}
12	5.5576	8.7013×10^{-1}	2.7375×10^{-3}	1.9644×10^{-1}	2.4796×10^{-2}	0.0000	0.0000
13	5.6027	8.6371×10^{-1}	2.4169×10^{-3}	1.9313×10^{-1}	2.5209×10^{-2}	0.0000	0.0000
14	1.4389	4.0085×10^{-1}	1.0954×10^{-2}	8.8157×10^{-2}	1.6493×10^{-2}	4.9122×10^{-3}	8.4889×10^{-2}
15	1.4413	4.0665×10^{-1}	1.1578×10^{-2}	1.0250×10^{-1}	1.6054×10^{-2}	6.0593×10^{-3}	1.1626×10^{-1}

Table 2. Eigenvalues for the NEACRP reactor.

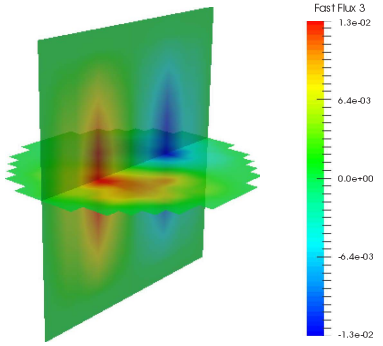
Eigenvalue	Value
1	1.002
2	0.98862
3	0.985406
4	0.985406



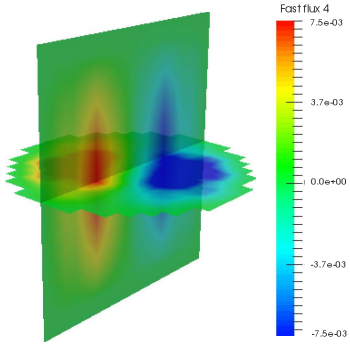
(**a**) 1st mode



(**b**) 2nd mode



(**c**) 3rd mode



(**d**) 4th mode

Figure 2. Fast fluxes' distribution of the NEACRP reactor corresponding to the first four modes.

First, we show the convergence history of the MGBNM to obtain the solution of the eigenvalue problem. Figure 3 shows the number of iterations against the residual error for the NEACRP reactor. It is observed that the MGBNM only needed four iterations to reach a residual error equal to 10^{-6}.

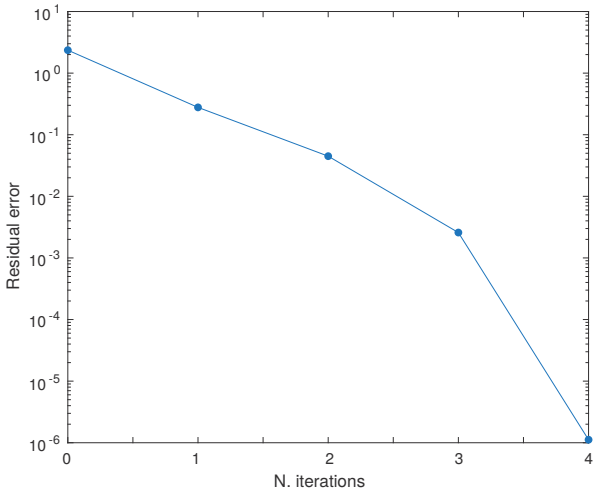


Figure 3. Convergence history of the Modified Generalized Block Newton Method (MGBNM) for the NEACRP reactor.

Table 3 collects the average number of iterations obtained by directly applying the ILU preconditioner of A and the total time that the GMRES method needs to reach the residual error in the linear systems given in Tol($\|b - Ax\|$). The time spent to assemble the matrices and to build the preconditioner (setup time (s)) is also displayed. These data are presented for each iteration and in a total sum. This table shows that the number of iterations is not very high, but the time spent to assemble the matrix and to construct the preconditioner increases the total CPU time considerably. It is necessary to build in each iteration a new preconditioner for A because the columns related to the block Z change considerably in each updating.

Table 3. Data for the preconditioner P for the NEACRP reactor. GMRES, Generalized Minimal Residual.

No. It. MGBNM	Tol ($\|b - Ax\|$)	Mean Its. GMRES	Setup Time (s)	Total Time (s)
1	1×10^{-2}	4.5	12.0	18.0
2	1×10^{-3}	9.75	12.0	20.4
3	1×10^{-5}	20.75	12.0	25.2
4	1×10^{-8}	37.5	12.0	33.2
Total		72.5	48.0	96.8

Table 4 displays these data related to the block preconditioner proposed $\hat{P}_J$ that uses the ILU preconditioner for approximating the inverse of $M - \lambda_1 L$. It is observed that we only needed to assemble the matrix $M - \lambda_1 L$ once in the first iteration to build the preconditioner. This is because we only needed a preconditioner of $M - \lambda_1 L$, and the value of λ_1 was very similar for all iterations. The mean of the number of iterations of the GMRES preconditioned with $\hat{P}_J$ was larger than in the previous case, but the total CPU time of using this block preconditioner was reduced by 26 s with respect to the full preconditioner.

Table 4. Data for the preconditioner $\hat{P}_J$ with ILUfor the NEACRP reactor.

No. It. MGBNM	Tol ($\|b - Ax\|$)	Mean Its GMRES	Setup Time (s)	Total Time (s)
1	1×10^{-2}	8.25	6.6	12.9
2	1×10^{-3}	13.25	-	9.5
3	1×10^{-5}	23.25	-	16.6
4	1×10^{-8}	41.25	-	30.0
Total		86.0	6.6	70.0

Table 5 shows the data related to the block preconditioner $\hat{P}_J$, but in this case, we have used the Geometric Multigrid (GMG) preconditioner to approximate the inverse of $M - \lambda_1 L$. The results show, in comparison with the results of Table 4, that in spite of the total number of iterations and the setup time being much lower for the GMG, the total computational time is much higher. This is due to the application of the GMG preconditioner being more expensive than the application of the ILU preconditioner.

Table 5. Data for the preconditioner $\hat{P}_J$ with the Geometric Multigrid (GMG) for the NEACRP reactor.

No. It. MGBNM	Tol ($\|b - Ax\|$)	Mean Its GMRES	Setup Time (s)	Total Time (s)
1	1×10^{-2}	6.00	2.5	19.8
2	1×10^{-3}	9.75	-	30.8
3	1×10^{-5}	12.75	-	39.4
4	1×10^{-8}	20.50	-	61.3
Total		49.00	2.5	151.3

The next results were obtained by using the block preconditioner, $\hat{P}$, but in these cases, approximating the $(M - \lambda_1 L)^{-1}$ by the ILU preconditioner of $-L$ ($\hat{P}_L$) and by a block preconditioner of $-L$ ($Q_{\hat{L}}$). The most relevant data to compare the preconditioners considered in this work are exposed in Table 6. They were the total iterations of the GMRES, the total setup time, the total time to compute the solution, and the maximum computational memory spent by the matrices. We observe that the number of iterations increased when worse approximations of the inverse of A were considered, but the setup time that each preconditioner needs became smaller. Moreover, the maximum CPU memory was also reduced significantly. In the total CPU times, we observed that the block preconditioner ($\hat{P}$), in all of its versions, improved the times obtained by applying the ILU preconditioner of A directly. Between the possibilities for obtaining a preconditioner of $M - \lambda_1 L$, there were no big differences in the computational times, but there was an important savings of the computational memory. The best results were obtained by $\hat{P}_{\hat{L}}$ if the computational memory consumption was taken into account.

Table 6. Data obtained by using different preconditioners for the NEACRP reactor.

Prec.	Its GMRES	Time Setup	Total Time	Max. CPU mem.
$P^{\mathbf{ILU}}$	72.5	48.0 s	96.8 s	2062 Mb
$\hat{P}_J{}^{\mathrm{ILU}}$	86.0	6.6 s	70.0 s	1418 Mb
$\hat{P}_L{}^{\mathrm{ILU}}$	98.0	4.4 s	73.2 s	787 Mb
$\hat{P}_Q{}^{\mathrm{ILU}}$	100.25	1.8 s	74.4 s	787 Mb

Table 7 shows the timings and the memory spent in the matrix allocation by using the matrix-free technique or without using this strategy. The results show that not only the matrix memory consumption and the time to assemble were reduced, but also the time spent to compute the matrix-vector products. That implies that the matrix-free strategy reduced the total CPU time by about 30%.

Table 7. Data obtained using different matrix implementations for the NEACRP reactor.

	Matrix Memory	Time Matvec Products	Time Assembly	Time Newton	Total Time
Sparse Matrix	787 Mb	27 s	7 s	51 s	74 s
Matrix Free	319 Mb	10 s	4 s	33 s	52 s

Finally, we compared the MGBNM with this methodology against other eigenvalue solvers commonly used in the neutron diffusion computations (Table 8). We show the results by using a different number of computed eigenvalues (No. eigs). In particular, we have chosen for this comparison the generalized Davidson preconditioned with the block Gauss–Seidel preconditioner and the Krylov–Schur method by previously reducing the generalized eigenvalue problem to an ordinary eigenvalue problem as in [2]. To use both methods, the library SLEPc has been used [12]. From the computational times, we can deduce that the MGBNM was twice as fast as the rest of solvers for the computation of one and two eigenvalues, and it was very competitive at computing four eigenvalues.

Table 8. Computational times for the MGBNM with $\hat{P}_Q$, the generalized Davidson method, and the Krylov–Schur method for the NEACRP reactor. eigs, eigenvalues.

No. Eigs	MGBNM	Generalized Davidson	Krylov–Schur
1	14 s	28 s	27 s
2	23 s	39 s	37 s
4	53 s	48 s	52 s

4.2. Ringhals Reactor

For a practical application of the preconditioners in a real reactor, we have chosen the configuration of the Ringhals rector. Particularly, we have chosen the C9 point of the BWRreactor Ringhals I stability benchmark, which corresponds to a point of operation that degenerated in an out-of-phase oscillation [23]. It is composed of 27 planes with 728 cells in each plane. A representation with more detail of its geometry can be observed in Figure 4. The spatial discretization using finite elements of degree three gave 1,106,180 degrees of freedom. The coarse mesh considered to obtain an initial guess for the MGBNM had 6709 cells and the problem associated with this mesh a size of 386,768 degrees of freedom. The computed dominant eigenvalues were 1.00191, 0.995034, 0.992827, and 0.991401. The corresponding fast fluxes are represented in Figure 5.

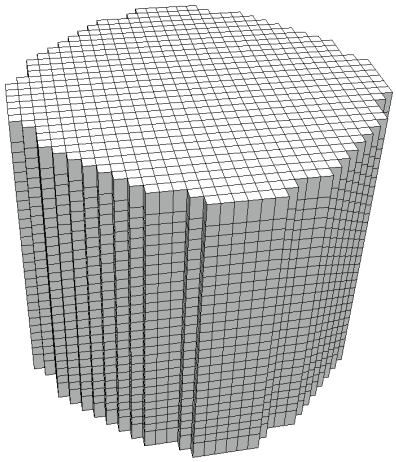
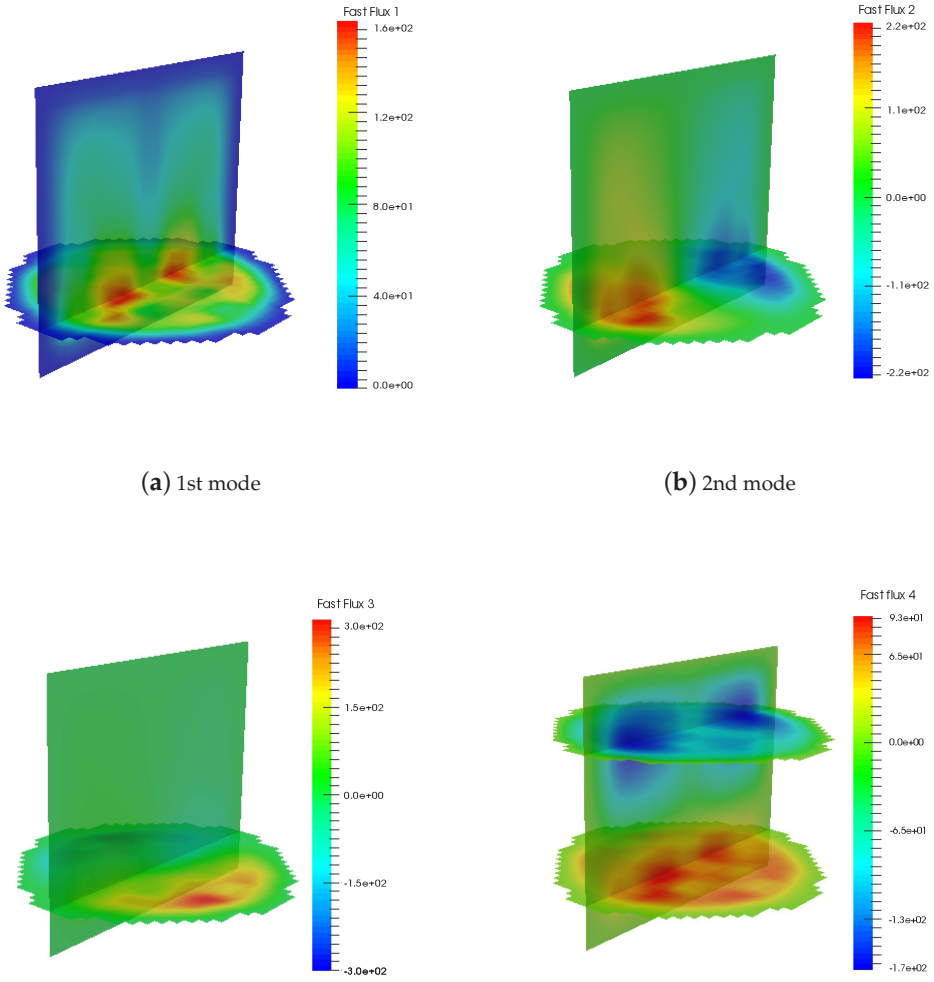


Figure 4. Geometry of the Ringhals reactor.



(**a**) 1st mode

(**b**) 2nd mode

(**c**) 3rd mode

(**d**) 4th mode

Figure 5. Fast fluxes' distribution of the Ringhals reactor corresponding to the first four modes.

The convergence history of the MGBNM associated with the Ringhals reactor is represented in Figure 6. For this reactor, the number of iterations needed to reach the tolerance (10^{-6}) was also equal to four.

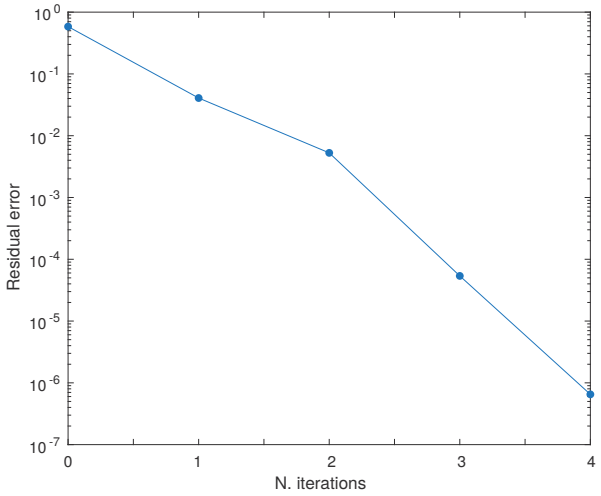


Figure 6. Convergence history of the MGBNM for the Ringhals reactor.

Table 9 collects the average number of iterations for the GMRES method for each iteration of MGBNM, the time to assemble the matrices and build the preconditioners, the total time of the MGBNM to reach the tolerance, and the maximum computational memory requested to assemble the matrices. From this table, similar conclusions as the ones obtained for the previous reactor are deduced. The number of iterations was not reduced, but the total CPU time and the maximum memory decreased considerably. For the Ringhals reactor, the most efficient option, in terms of computational memory, was also to apply the block preconditioner $\hat{P}_Q$. However, as the size of this reactor is much larger, the differences between the preconditioners for computational memory were much higher.

Table 9. Data obtained using different preconditioners for the Ringhals reactor.

Prec.	Its GMRES	Time Setup	Total Time	Max. CPU mem.
P	71.5	155 s	408 s	12.5 Gb
$\hat{P}_J$	81.0	39 s	331 s	9.3 Gb
$\hat{P}_L$	85.2	36 s	348 s	6.2 Gb
$\hat{P}_Q$	88.2	8 s	308 s	3.7 Gb

Finally, in Table 10, we compare the MGBNM with the generalized Davidson method and the Krylov–Schur method from the SLEPc library as in the previous reactor. The results show that the MGBNM was more efficient in terms of the computational time to compute one or a set of the lambda modes than the generalized Davidson and the Krylov–Schur methods.

Table 10. Computational times for the MGBNM with $\hat{P}_Q$, the generalized Davidson method, and the Krylov–Schur method for the Ringhals reactor.

No. Eigs	MGBNM	Generalized Davidson	Krylov–Schur
1	100 s	264 s	324 s
2	207 s	294 s	471 s
4	308 s	317 s	528 s

5. Conclusions

The modified generalized block Newton method (MGBNM) is an efficient eigenvalue solver that has been used to compute the dominant λ-modes associated with the neutron diffusion equation. This problem has been previously discretized by using a high order finite element method. This method

requires solving many linear systems that need to be previously preconditioned. Different block preconditioners have been studied as an alternative to assemble the full matrix and to construct a preconditioner in each iteration. The different preconditioners have been tested in a benchmark reactor problem (NEACRP) and in a realistic reactor problem (Ringhals). The preconditioners proposed in this work break down the setup cost at the price of a slight increase of the number of iterations. The result is a significant reduction of the total CPU time needed to reach convergence and the memory occupancy. Among the implementations studied, it is shown that the best option is the one that uses the block structure of the L matrix. Moreover, this implementation permits implementing the MGBNM with a matrix-free technique, thus greatly reducing the memory consumption. The differences increase when the size of the problem is larger. In comparison with other eigenvalue solvers, such as the generalized Davidson and the Krylov–Schur methods, the numerical results conclude that the MGBNM with this strategy of preconditioning is more efficient in some cases and very competitive in the rest. In future works, the MGBNM with these strategies for the preconditioning will be applied to other approximations of the neutron transport equations as the SP_N equations.

Author Contributions: A.C., L.B., and A.M. developed and investigated the methodology; A.C. and A.V.-F. implemented and validated the methods; D.G. and G.V. supervised the work. All authors discussed the results and contributed to the final manuscript.

Acknowledgments: This work has been partially supported by the Spanish Ministerio de Economía y Competitividad under Projects ENE2014-59442-P, MTM2014-58159-P, and BES-2015-072901.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Stacey, W.M. *Nuclear Reactor Physics*; John Wiley & Sons: Hoboken, NJ, USA, 2018.
2. Vidal-Ferrandiz, A.; Fayez, R.; Ginestar, D.; Verdú, G. Solution of the Lambda modes problem of a nuclear power reactor using an h–p finite element method. *Ann. Nucl. Energy* **2014**, *72*, 338–349. [CrossRef]
3. Bangerth, W.; Hartmann, R.; Kanschat, G. deal.II—A general-purpose object-oriented finite element library. *ACM Trans. Math. Softw.* **2007**, *33*, 24. [CrossRef]
4. Kronbichler, M.; Kormann, K. A generic interface for parallel cell-based finite element operator application. *Comput. Fluids* **2012**, *63*, 135–147. [CrossRef]
5. Hageman, L.A.; Young, D.M. *Applied Iterative Methods*; Courier Corporation: North Chelmsford, MA, USA, 2012.
6. Sutton, T.M. Wielandt Iteration as Applied to the Nodal Expansion Method. *Nucl. Sci. Eng.* **1988**, *98*, 169–173. [CrossRef]
7. Warsa, J.S.; Wareing, T.A.; Morel, J.E.; McGhee, J.M.; Lehoucq, R.B. Krylov subspace iterations for deterministic k-eigenvalue calculations. *Nucl. Sci. Eng.* **2004**, *147*, 26–42. [CrossRef]
8. Allen, E.; Berry, R. The inverse power method for calculation of multiplication factors. *Ann. Nucl. Energy* **2002**, *29*, 929–935. [CrossRef]
9. Verdú, G.; Ginestar, D.; Vidal, V.; Muñoz-Cobo, J. 3D λ-modes of the neutron-diffusion equation. *Ann. Nucl. Energy* **1994**, *21*, 405–421. [CrossRef]
10. Verdú, G.; Miró, R.; Ginestar, D.; Vidal, V. The implicit restarted Arnoldi method, an efficient alternative to solve the neutron diffusion equation. *Ann. Nucl. Energy* **1999**, *26*, 579–593. [CrossRef]
11. Verdú, G.; Ginestar, D.; Miró, R.; Vidal, V. Using the Jacobi–Davidson method to obtain the dominant Lambda modes of a nuclear power reactor. *Ann. Nucl. Energy* **2005**, *32*, 1274–1296. [CrossRef]
12. Hernandez, V.; Roman, J.E.; Vidal, V. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Softw.* **2005**, *31*, 351–362. [CrossRef]
13. Evans, T.M.; Stafford, A.S.; Slaybaugh, R.N.; Clarno, K.T. Denovo: A new three-dimensional parallel discrete ordinates code in SCALE. *Nucl. Technol.* **2010**, *171*, 171–200. [CrossRef]
14. Hamilton, S.P.; Evans, T.M. Efficient solution of the simplified PN equations. *J. Comput. Phys.* **2015**, *284*, 155–170. [CrossRef]
15. González-Pintor, S.; Ginestar, D.; Verdú, G. Updating the Lambda Modes of a nuclear power reactor. *Math. Comput. Model.* **2011**, *54*, 1796–1801. [CrossRef]

16. Carreño, A.; Vidal-Ferràndiz, A.; Ginestar, D.; Verdú, G. Spatial modes for the neutron diffusion equation and their computation. *Ann. Nucl. Energy* **2017**, *110*, 1010–1022. [CrossRef]
17. Gill, D.; Azmy, Y. Newton's method for solving k-eigenvalue problems in neutron diffusion theory. *Nucl. Sci. Eng.* **2011**, *167*, 141–153. [CrossRef]
18. Knoll, D.; Park, H.; Newman, C. Acceleration of k-eigenvalue/criticality calculations using the Jacobian-free Newton-Krylov method. *Nucl. Sci. Eng.* **2011**, *167*, 133–140. [CrossRef]
19. Kong, F.; Wang, Y.; Schunert, S.; Peterson, J.W.; Permann, C.J.; Andrš, D.; Martineau, R.C. A fully coupled two-level Schwarz preconditioner based on smoothed aggregation for the transient multigroup neutron diffusion equations. *Numer. Linear Algebra Appl.* **2018**, *25*, e2162. [CrossRef]
20. Lösche, R.; Schwetlick, H.; Timmermann, G. A modified block Newton iteration for approximating an invariant subspace of a symmetric matrix. *Linear Algebra Appl.* **1998**, *275*, 381–400. [CrossRef]
21. Saad, Y. *Iterative Methods for Sparse Linear Systems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2003.
22. Finnemann, H.; Galati, A. NEACRP 3-D LWR Core Transient Benchmark: Final Specifications. Available online: https://www.oecd-nea.org/science/docs/1991/neacrp-l-1991-335.pdf (accessed on 13 January 2019).
23. Lefvert, T. OECD/NEA Ringhals 1 Stability Benchmark. Available online: http://www.nea.fr/science/docs/1996/nsc-doc96-22.pdf (accessed on 13 January 2019).
24. Balay, S.; Abhyankar, S.; Adams, M.; Brune, P.; Buschelman, K.; Dalcin, L.; Gropp, W.; Smith, B.; Karpeyev, D.; Kaushik, D.; et al. *PETSc Users Manual Revision 3.7*; Technical Report for Argonne National Lab: Argonne, IL, USA, 2016.
25. Carreño, A.; Vidal-Ferràndiz, A.; Ginestar, D.; Verdú, G. Block hybrid multilevel method to compute the dominant λ-modes of the neutron diffusion equation. *Ann. Nucl. Energy* **2018**, *121*, 513–524. [CrossRef]