Hindawi
Computational and Mathematical Methods
Volume 2022, Article ID 3884836, 13 pages
https://doi.org/10.1155/2022/3884836

Research Article

Strategies of Preconditioner Updates for Sequences of Linear Systems Associated with the Neutron Diffusion

A. Carreño,[1] **L. Bergamaschi**,[2] **A. Martínez**,[3] **D. Ginestar**,[4] **A. Vidal-Ferràndiz**,[4] **and G. Verdú**[1]

[1]*ISIRYM, Universitat Politècnica de València, Valencia, Spain*
[2]*DICEA, University of Padua, Padua, Italy*
[3]*DMG, University of Trieste, Trieste, Italy*
[4]*IMM, Universitat Politècnica de València, Valencia, Spain*

Correspondence should be addressed to A. Carreño; amcarsan@iqn.upv.es

Received 9 September 2021; Revised 6 May 2022; Accepted 16 June 2022; Published 26 June 2022

Academic Editor: Higinio Ramos

The time-dependent neutron diffusion equation approximates the neutronic power evolution inside a nuclear reactor core. Applying a Galerkin finite element method for the spatial discretization of these equations leads to a stiff semi-discrete system of ordinary differential equations. For time discretization, an implicit scheme is used, which implies solving a large and sparse linear system of equations for each time step. The GMRES method is used to solve these systems because of its fast convergence when a suitable preconditioner is provided. This work explores several matrix-free strategies based on different updated preconditioners, which are constructed by low-rank updates of a given initial preconditioner. They are two tuned preconditioners based on the bad and good Broyden's methods, initially developed for nonlinear equations and optimization problems, and spectral preconditioners. The efficiency of the resulting preconditioners under study is closely related to the selection of the subspace used to construct the update. Our numerical results show the effectiveness of these methodologies in terms of CPU time and storage for different nuclear benchmark transients, even if the initial preconditioner is not good enough.

1. Introduction

The safety and the design of a nuclear power plant require, among others, fast and accurate codes that simulate the behaviour of the neutrons inside of the core of a nuclear reactor. The time-dependent multigroup neutron diffusion equation approximates the time evolution of the neutronic power. This equation is an approximation of the neutron transport equation that assumes that the neutron current is proportional to the gradient of the scalar neutron flux by means of a diffusion coefficient. This equation can be expressed as [1].

$$\begin{aligned} \mathcal{V}^{-1}\frac{\partial \Phi}{\partial t} + \mathcal{L}\Phi &= (1-\beta)\mathcal{F}\Phi + \sum_{k=1}^{K}\lambda_k^d \chi \mathcal{C}_k, \\ \frac{\partial \mathcal{C}_k}{\partial t} &= \beta_k \mathcal{F}_1 \Phi - \lambda_k^d \mathcal{C}_k, k = 1, \cdots, K, \end{aligned} \tag{1}$$

where for the special case of two energy groups and without considering up-scattering, the continuous operators are given by

$$\mathcal{L} = \begin{bmatrix} -\vec{\nabla}\cdot\left(D_1\vec{\nabla}\right) + \Sigma_{a_1} + \Sigma_{12} & 0 \\ -\Sigma_{12} & -\vec{\nabla}\cdot\left(D_2\vec{\nabla}\right) + \Sigma_{a_2} \end{bmatrix}, \quad \mathcal{F} = \begin{bmatrix} \nu_1\Sigma_{f_1} & \nu_2\Sigma_{f_2} \\ 0 & 0 \end{bmatrix}, \mathcal{V} = \begin{bmatrix} 1/\nu_1 & 0 \\ 0 & 1/\nu_2 \end{bmatrix},$$
$$\mathcal{F}_1 = \begin{bmatrix} \nu_1\Sigma_{f_1} & \nu_2\Sigma_{f_2} \end{bmatrix}, \qquad \chi = [1 \quad 0]^T, \Phi = [\Phi_1 \quad \Phi_2]^T. \tag{2}$$

The vectors Φ_1 and Φ_2 denote the fast and thermal flux, respectively. The vectors $\mathcal{C}_k, k = 1, \cdots, K$ are the concentration of the neutron delayed precursors, where K is the total number of precursor groups. The cross-sections and diffusion coefficients, Σ_{a_g}, Σ_{f_g}, $\Sigma_{12}D_g$, $g = 1, 2$, are functions that depend on the materials of the reactor. $\nu_g, g = 1, 2$, is the average number of neutrons released per fission while $\nu_g, g = 1, 2$, is the neutron velocity associated with the energy group g. The

spectrum of the prompt and the delayed neutrons are denoted by χ. The fraction of the delayed neutrons is β_k such that the total delayed neutron fraction is $\beta = \sum_k^K \beta_k$. Finally, the neutron precursor delayed constants are denoted as λ_k^d. The rest of the coefficients in this work are considered constants related to the neutron delayed precursors. The problem (1) depends on both the position and time; thus, spatial and time discretization must be selected in order to compute the numerical solution. Applying a Galerkin finite element method for the spatial discretization of (1) leads to a stiff semi-discrete system of ordinary differential equations. For time discretization, an implicit scheme is used, which implies solving a linear system of equations for each time step. In a realistic nuclear reactor, the size of the resulting matrices is huge, and Krylov subspace methods are chosen to solve the systems, but these methods may exhibit slow convergence unless a good preconditioner is provided.

In many works, preconditioners for these unstructured linear systems are based on incomplete factorizations of the coefficient matrix [2]. Incomplete LU factorization works well for this type of problem, but it requires the storage of both the full matrix and the preconditioner, and this can be very expensive from the point of view of computational cost and memory requirement. Since the computation of a good preconditioner can be expensive, while solving a sequence of many linear systems, it is advantageous to recycle preconditioners, that is, update a previous preconditioner and reuse the updated version. In, e.g., [3–7] the authors describe updating strategies for incomplete factorizations or approximate inverse preconditioners in the solution of sequences of linear systems. Unfortunately, these techniques do not avoid the need to store the preconditioner. In this work, we focus on updating strategies that can be applied to any initial preconditioner and that can be constructed and applied in a matrix-free regime.

This work explores several matrix-free strategies based on different updated preconditioners, which are constructed by low-rank corrections of a given initial preconditioner. We analyze the performance of two tuned preconditioners based on the bad and good Broyden's methods, initially developed for nonlinear equations and optimization problems, and spectral preconditioners. We study the influence of the selection of the subspace used to construct the update on the efficiency of the resulting preconditioner. The efficiency is measured by both the decrease in the number of iterations needed to reach convergence and the reduction of the overall solution time. We present numerical results showing the effectiveness of these methodologies in terms of CPU time and storage for different nuclear benchmark transients, even when starting from a poor-quality initial preconditioner. Summarizing, the aims of this contribution are (1) to analyze the different low-rank updates, (2) to apply this tool to the sequence of linear systems arising from the discretization of the neutron diffusion equation, and (3) to perform an experimental study on the choice of S_i.

The remaining of this paper is organized as follows. In Section 2, we describe the spatial and time discretizations that gives rise to the sequences of sparse nonsymmetric linear systems. Section 3 describes the different low-rank preconditioner updates we consider while in Section 4, we describe the sequences of low-rank preconditioners and analyze different choices for selecting the basis of the subspace used to construct the low-rank update. In Section 5, we present the numerical experiments and discuss the results. The paper is ended by stating the main conclusions of this study in Section 6.

2. Spatial and Time Discretizations

A high-order finite element method (FEM) is considered for the spatial discretization of the neutron diffusion equation to get a semi-discrete system of ordinary differential equations. This methodology can be applied to any type of reactor geometry. In this work, a continuous Galerkin method with Lagrange polynomials is used. It yields to the system of ordinary differential equations [2]

$$\begin{aligned} V\frac{d\widehat{\Phi}}{dt} + L\bar{\Phi} &= (1-\beta)F\bar{\Phi} + \sum_{p=1}^{N_p}\lambda_p^d XC_p, \\ \frac{dPC_p}{dt} &= \beta_k F_1 \widehat{\Phi} - \lambda_p^d PC_p, p = 1, \cdots, N_p, \end{aligned} \tag{3}$$

where $\widehat{\Phi}$, C, L, F, V, and X are the discrete version of the previous differential operators. The implementation of the FEM has been made by using the structures of the open source library Deal.ii [8]. For a realistic nuclear reactor problem, the system (3) is, in general, stiff. Hence, it is necessary to use implicit methods to avoid artificial stepsize restriction due to stability and not accuracy reasons. In this work, a one-step implicit scheme is used, such that the time domain, $[0, T]$ is divided into several time intervals $[t_n, t_{n+1}]$ with $\Delta t_n = t_{n+1} - t_n$ and the neutron flux at time t_{n+1} can be approximated by solving the linear system [2].

$$T^{n+1}\widehat{\Phi}^{n+1} = \frac{1}{\Delta t_h} V^n \widehat{\Phi}^n + \sum_{p=1}^{N_p} \lambda_p^d e^{-\lambda_p^d \Delta t_h} XC_p^n, \tag{4}$$

where

$$T^{n+1} = \frac{1}{\Delta t_n} V^{n+1} + L^{n+1} - \hat{a}F^{n+1}, \hat{a} = 1 - \beta + \sum_{p=1}^{N_p} \beta_p \left(1 - e^{-\lambda_p^d \Delta t_h}\right). \tag{5}$$

Superindex n denotes the matrices and vectors evaluated at time t_n.

The neutron precursor equations are approximated by

$$PC_p^{n+1} = PC_p^n e^{-\lambda_p^d \Delta t_n} + \frac{\beta_p}{\lambda_p^d}\left(1 - e^{-\lambda_p^d \Delta t_h}\right) F_1^{n+1} \Phi^{n+1}, p = 1, \cdots, N_p. \tag{6}$$

The previous scheme is numerically stable, but due to its implicit character, it requires solving a linear system at each

time step. In a realistic nuclear reactor, the size of the matrices is huge, and Krylov subspace methods are chosen to solve the systems, but, to accelerate convergence, a good preconditioner is mandatory. In many works, preconditioners for these unstructured linear systems are based on incomplete factorizations of the coefficient matrix [2]. Incomplete LU factorization works well for this type of problem, but it requires the storage of the full matrix together with the preconditioner, and this can be very expensive from the point of view of memory requirement.

To alleviate the cost of computing the preconditioners, other alternatives, based on low-rank updates of a given incomplete factorization, have been studied, see, e.g., [3–6], which can be implemented using only vector-matrix products and without requiring additional storage.

In our application, although the matrix operators change at each time step, they are not very different, and the coefficient matrices related to successive time steps share similar spectral properties. Therefore, it is reasonable to explore methodologies to improve the convergence of the Krylov methods by using some information generated when the solutions of the previous linear systems are computed. A technique which takes into account information of the Krylov process to accelerate the iterative solution of subsequent linear systems has been also investigated, e.g., in [9]. In [10], spectral information on the system matrix (which is kept constant throughout the sequence) is extracted and refined from the conjugate gradient solution of previous linear systems and used to deflate the subsequent systems in the sequence.

In this work, an initial preconditioner is updated from this spectral information to improve the convergence of the linear solver along a given transient. This technique has been studied in previous works [3, 11–13]. Moreover, these preconditioners have a matrix-free implementation if the initial preconditioner can also be applied in a matrix-free regime, as for instance the polynomial preconditioner recently resumed in [14], reducing the CPU resources of the matrix allocation considerably. Another strategy to update preconditioners for sequences of linear systems, based on the sparse minimization of the Frobenius norm of a suitable error function, has been recently studied in [15].

We describe in the next section different preconditioner updates for a linear system. Next, the methodology is extended to a sequence of linear systems.

3. Updated Preconditioners for a Linear System

Let us consider a linear system

$$Ax = b, \tag{7}$$

where $A \in \mathbb{R}^{N\times N}$ is a nonsymmetric sparse matrix; $x, b \in \mathbb{R}^N$ is the solution vector and the independent term, respectively. Under suitable conditions, the rate of convergence of Krylov methods such as the GMRES [16] can depend on the distribution of the eigenvalues of A [17]. Even if, in the nonsymmetric instances, this is not directly related to extremal eigenvalues, however, eigenvalues with small modulus are usually responsible of slow convergence of these methods. In this context, we define a left preconditioned system as

$$PAx = Pb. \tag{8}$$

The solution of (8) is the same as that of system (7) if P is nonsingular; thus, P is designed such that the condition number of matrix PA is smaller than the condition number of A.

Theoretically, the optimal preconditioner is $P = A^{-1}$ because the matrix of the left preconditioned system is $A^{-1}A = I$, which has optimal condition number 1, requiring a single iteration for convergence. However, applying the optimal preconditioner is as difficult as solving the original system, and preconditioners that approximate A^{-1} are used. Moreover, we are also interested in obtaining a preconditioner that is cheap to construct and apply.

Alternatively, the system (19) can be also preconditioned on the right as

$$APy = b, x = Py. \tag{9}$$

If the systems are associated with a symmetric and positive definite (SPD) matrix, we can apply the conjugate gradient method, and the convergence does not depend on using right or left preconditioning. However, this behaviour does not generally happen when the GMRES is applied for nonsymmetric matrices. The results can be different if the preconditioner is applied on the left or on the right. Typically, the right preconditioner is preferred since, in this case, the exit test is based on the true residual.

3.1. Tuned Preconditioner. The concept of tuned preconditioner was introduced in [18], and other papers of the same authors, in the framework of iterative eigensolvers. The link between these preconditioner and the quasi-Newton updates, presented in [19], is described in the survey [20]. In this work, we use a block definition of these updates as discussed in [21, 22].

Definition 1. Let P_0 be an initial preconditioner of the matrix A and $S \in \mathbb{R}^{N\times k}$ a matrix with full column rank, a tuned preconditioner is a matrix P obtained by updating P_0 by a low-rank matrix depending on A and S so as to satisfy the relation

$$PAS = S. \tag{10}$$

From the previous definition, the matrix PA has the eigenvalue 1 with (at least) multiplicity k associated with the eigenvectors corresponding to each column of S.

Different types of tuned preconditioners can be developed. The following tuned preconditioners are derived from the quasi-Newton methods developed in numerical optimization. These methods provide an approximation of the inverse of the Jacobians, which is equal to A^{-1} if we consider the problem $F(x) = 0$, for the particular case of $F(x) = Ax - b$ [19, 23]. Therefore, they are good candidates to design preconditioners for Krylov methods. In particular, two

Broyden algorithms are used for this purpose, the Bad-Broyden method and the Good-Broyden method.

3.1.1. The Bad-Broyden Preconditioner. This strategy has been developed in [24], where the preconditioner is designed from rank-1 updates from either the "bad" Broyden's method or Eirola and Nevalinna's method. The Sherman-Morrison formula constructs the sequence of approximations of the inverse of the Jacobian $P_{k+1} \approx A^{-1}$ by a low-rank update of the previous approximate matrix as

$$P_{k+1} = P_k + \frac{(s_k - P_k y_k) y_k^T}{y_k^T y_k}, \tag{11}$$

where $s_k = \Delta x_k = x_{k+1} - x_k$ is the (quasi) Newton step and $y_k = F(x_{k+1}) - F(x_k)$. This is known as the Bad-Broyden variant. To define the preconditioner, y_k is taken as $y_k = As_k$ to avoid the restriction of choosing the vectors s_k as the Newton step. From these assumptions, the sequence would be

$$P_{k+1} = P_k \left(I_N - \frac{\left(As_k s_k^T A^T\right)}{s_k^T A^T A s_k} \right) + \frac{s_k s_k^T A^T}{s_k^T A^T A s_k}. \tag{12}$$

This formula can be generalized to a block rank- k extension if the column vectors of $S = [s_1, \cdots s_k]$ are in the same subspace that the ones that satisfy the $A^T A$-conjugacy, i.e., $s_i^T A^T A s_j = 0$, $i \neq j$, where $i, j = 1 \cdots, k$ [25]. The authors in [24] called this preconditioner limited memory preconditioner by setting $P_0 = I_N$. In our case, we do not take this assumption and P_0 will be an initial preconditioner.

Definition 2. Let $A \in \mathbb{R}^{N \times N}$ be a general nonsingular matrix and $S \in \mathbb{R}^{N \times k}$ of full rank k, with $k \leq N$. The Bad-Broyden preconditioner $P \in \mathbb{R}^{N \times N}$ for nonsymmetric systems is defined as

$$P = P_0 \left(I_N - AS\left(S^T A^T AS\right)^{-1} S^T A^T \right) + S\left(S^T A^T AS\right)^{-1} S^T A^T. \tag{13}$$

Section 4.1 will study the selection of S. This preconditioner satisfies the tuning property (10). The authors [25] show spectral properties of the operator AP (where $P_0 = I_N$).

3.1.2. The Good-Broyden Preconditioner. Additionally, one can update the preconditioner by using the Good-Broyden method [20]. This algorithm updates the approximate inverse of the Jacobian P_{k+1} as

$$P_{k+1} = P_k + \frac{(s_k - P_k y_k) s_k^T P_k}{s_k^T P_k y_k}, \tag{14}$$

where, as before, $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. As in the previous case, choosing $y_k = As_k$ yields

$$P_{k+1} = P_k + \frac{(s_k - P_k A s_k) s_k^T P_k}{s_k^T P_k A s_k}, \tag{15}$$

and the previous expression can be extended to a block rank-k update if the column vectors of $S = [s_1, \cdots, s_k]$ are in the same subspace as the ones that satisfy the $P_k A$-conjugacy, that is, $s_i^T P_k A s_j = 0$, $i \neq j$, where $i, j = 1 \cdots, k$.

Definition 3. Let $A \in \mathbb{R}^{N \times N}$ be a general nonsingular matrix and $S \in \mathbb{R}^{N \times k}$ of full rank k, with $k \leq N$. The Good-Broyden preconditioner $P \in \mathbb{R}^{N \times N}$ for nonsymmetric systems is defined as

$$P = P_0 - (P_0 AS - S)\left(S^T P_0 AS\right)^{-1} S^T P_0. \tag{16}$$

This preconditioner also satisfies the tuning property (10). A selection of S will be studied in Section 4.1.

3.2. Spectral Preconditioners. The spectral preconditioners are based on low-rank corrections obtained from the spectral data associated with the smallest eigenvalues of the preconditioned matrix [26, 27].

We start with a preconditioner such that the preconditioned matrix $P_0 A$ is diagonalizable, that is,

$$P_0 A = V \Lambda V^{-1}, \tag{17}$$

where $\Lambda = \text{diag}\,(\lambda_i)$, where $|\lambda_1| \leq \cdots \leq |\lambda_n|$ are the eigenvalues and $V = (v_i)$ the associated right eigenvectors. Let V_ε be the set of right eigenvectors associated with the set of eigenvalues $|\lambda_i| \leq \varepsilon$.

Definition 4. Let S be such that $A_c = S^T A V_\varepsilon$ has full rank, a left spectral preconditioner is defined as the matrix

$$P = P_0 + V_\varepsilon A_c^{-1} S^T. \tag{18}$$

As proved in [26], it easy to see that the matrix PA maintains the same right eigenvectors V_ε as the matrix $P_0 A$ with eigenvalues $\eta_i = \lambda_i + 1$, showing that the spectral preconditioner shifts the smallest eigenvalues of the preconditioned matrix away from zero. This is expected to improve the convergence of the Krylov methods. To select the matrix S, different options are proposed in Section 4.1. In [26], a suitable choice of S is proved to leave unchanged all the other eigenvalues and eigenvectors.

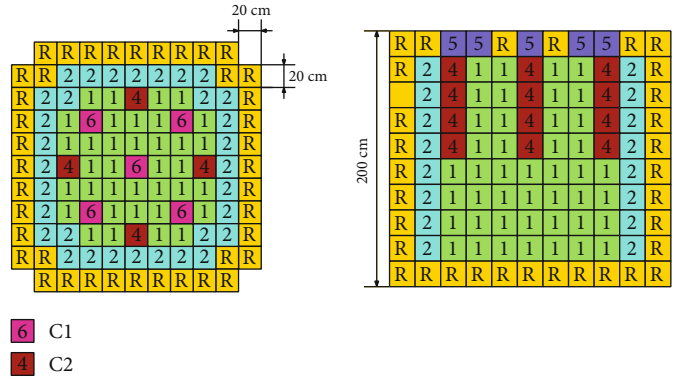
4. Sequence of Preconditioners for Sequences of Linear Systems

Now, let us consider a sequence of linear systems

$$A_i x_i = b_i, 1 \leq i \leq i_{\max}, \tag{19}$$

where $A_i \in \mathbb{R}^{N \times N}$ are nonsymmetric sparse matrices; $x_i \in \mathbb{R}^N$ are the solution vectors, which in this application are the neutron flux in the different time steps; and $b_i \in \mathbb{R}^N$ are the independent terms.

We are interested in computing a sequence of preconditioners P_i for the sequence of linear systems (19). To construct these preconditioners, we use the updated preconditioners of



(a) Radial profile (b) Axial profile

Figure 1: Geometry and materials distribution of the Langenbuch reactor.

Section 3, where the initial preconditioner is kept constant and matrix S is updated along the sequence.

Therefore, the Bad-Broyden preconditioner for the i-th linear system is

$$P_i = P_0\left(I_N - A_iS_i\left(S_i^TA_i^TA_iS_i\right)^{-1}S_i^TA_i^T\right) + S_i\left(S_i^TA_i^TA_iS_i\right)^{-1}S_i^TA_i^T. \tag{20}$$

The Good-Broyden preconditioner is

$$P_i = P_0 - \left(P_0A_iS_i - S_i\right)\left(S_i^TP_0A_iS_i\right)^{-1}S_i^TP_0. \tag{21}$$

Finally, the spectral preconditioner is

$$P_i = P_0 + V_\varepsilon\left(S_i^TA_iV_\varepsilon\right)^{-1}S_i^T. \tag{22}$$

In selecting P_0, two different possibilities are studied in function of the initial preconditioner chosen. If we use the ILU factorization for the initial preconditioner, the ILU of the matrix A_0 is used as P_0 for all systems of the transient. However, if we choose the inverse of the diagonal of the matrix as the initial preconditioner, whose computation is negligible, P_0 corresponds to the inverse of the diagonal matrix A_i.

4.1. Choice of S_i. The selection of S_i will influence the efficiency of the updated preconditioner in terms of both the number of iterations and the cost of applying the preconditioner. The best option is to set the columns of matrix S_i as the eigenvectors of the matrix A_iP_0 or P_0A_i (depending on whether right or left preconditioning is used) associated with the eigenvalues of the smallest modulus. However, in practical computations, this is not feasible, and other approximations are commonly used. A practical solution is to approximate these eigenvectors from the Ritz vectors that are defined as follows:

Definition 5. Let A be a nonsingular matrix and $Q = [q_1, \cdots, q_k]$ be a matrix whose columns form a set of orthonormalized vectors. A Ritz vector of A associated with the subspace span $\{q_1, \cdots, q_k\}$ is defined as the vector $w = Qy$, where y is an eigenvector of Q^TAQ. The eigenvalue θ, corresponding to the eigenvector y, is called Ritz value, and the pair (θ, w) is called the Ritz pair.

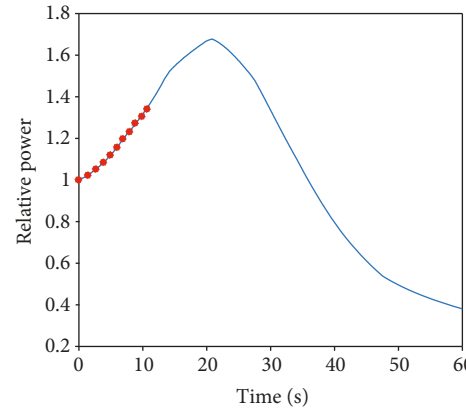


Figure 2: Global power of the Langenbuch transient.

Table 1: Number of iterations of LGMRES to solve the linear problem $Ax = b$ with the different updated preconditioners and choices of S. $P_0 = ILU(A)$.

S	No update	Bad-Broyden	Good-Broyden	Spectral
eigs(P_0A)	45	31	34	32
eigs(A)	45	32	34	35

It seems reasonable to update the initial preconditioners using the Ritz vectors associated with matrix A_iP_0 or P_0A_i. However, numerical results will show that the Ritz vectors associated with matrix A_i can be used instead, without any significant detriment of the convergence rate of the iterative method.

The Ritz vectors can be complex because the matrices A_i, P_0A_i, or A_iP_0 are nonsymmetric. In the case that a Ritz vector (w) with a complex Ritz value (θ) appears, the pair $(\bar{\theta}, \bar{w})$ is also a Ritz pair (A_i, P_0A_i, and A_iP_0 are real). Note that the real and imaginary parts of the Ritz vector w generate the same subspace as the two conjugate Ritz vectors. Therefore,

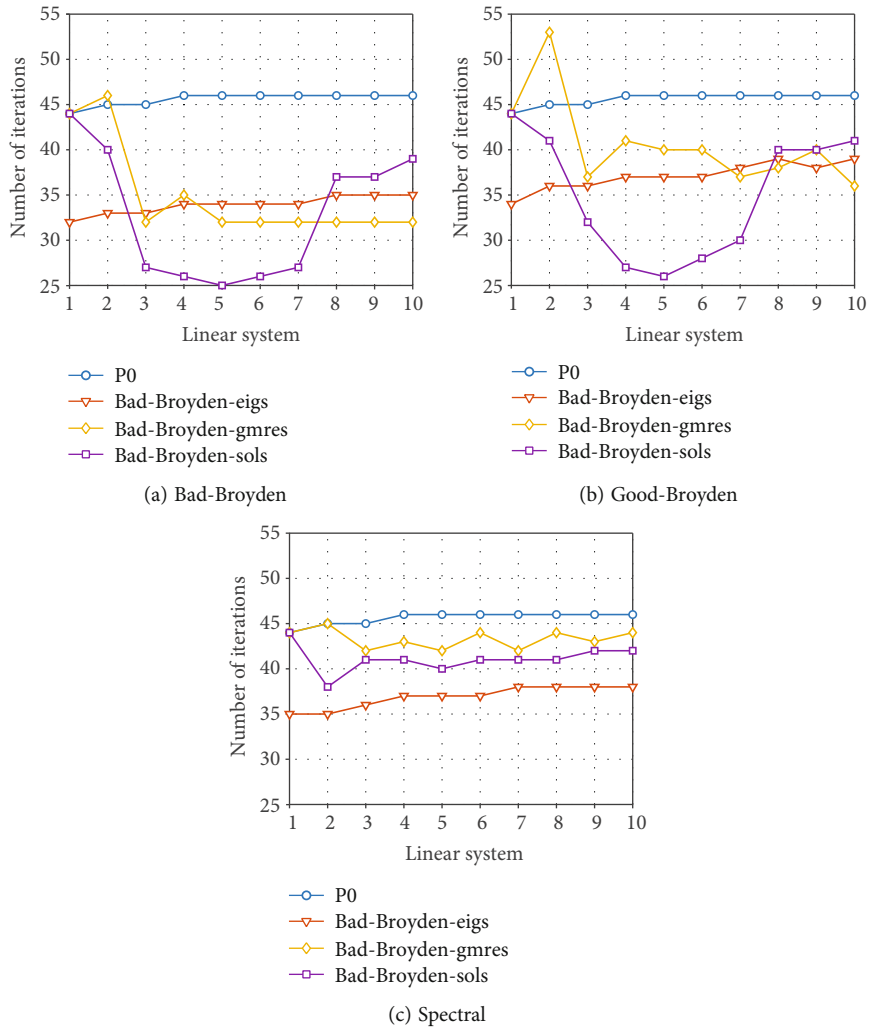


FIGURE 3: Number of iterations of the LGMRES method to solve the sequence of linear systems with the updated preconditioners with rank 5 corrections and P_0 equal to ILU (0).

the columns of S_i are set equal to the real and imaginary parts of the Ritz vector w. Finally, in case any linearly dependent (or nearly) column appears, the last vector is removed.

In the following, we describe two ways to get a set of vectors Q.

If a sequence of linear systems is solved with the GMRES(m) in k iterations, it is assumed that the relation $Q_l^T A_i Q_l = H_l$ holds with $Q_l \in \mathbb{R}^{N\times k}$ and $H_l \in \mathbb{R}^{k\times k}$ $(k \leq m)$. In this case, the Ritz vectors are computed with the basis Q_l constructed during the solution of the previous linear system [28, 9, 10].

Note that if the Ritz vectors associated with the matrix A_i are computed, these are obtained as $W = Q_l Z$, where the columns of the matrix Z are the eigenvectors associated with the smallest eigenvalues of H_l.

4.1.1. Previous Solutions of the Systems. The second option we consider is to use the orthonormal basis Q using solutions of the previous linear systems [29, 20]. In that sense, to solve the i-th linear system, the matrix $\tilde{Q}$ is defined as

$$\tilde{Q} = \left[x_{i-p}, \cdots, x_{i-1}\right]. \tag{23}$$

The column vectors of Q are orthonormal vectors that span the same subspace as $\tilde{Q}$ computed at each iteration before constructing the Ritz vectors.

This selection is based on the reasonable assumption that the components of a solution x_i are in the directions of the leftmost eigenvectors of A_i, because if the independent terms are expanded in terms of the eigenvectors of A_i, $b_i = \sum_{j=1}^{N} \alpha_j u_j^{(i)}$, then $x_k = \sum_{j=1}^{N} (\alpha_j/\lambda_j^{(i)}) u_j^{(i)}$ with the largest weights provided by the smallest eigenvalues.

On the other hand, in our application, two consecutive matrices in the sequence of linear systems display small changes since the time-step is usually considered small. In principle, this property suggests, but does not guarantee, that also eigenvectors of the two consecutive matrices are close. Experimentally, we found that the eigenvectors of A_i closely approximate those eigenvectors of A_{i+1}, for a given i.

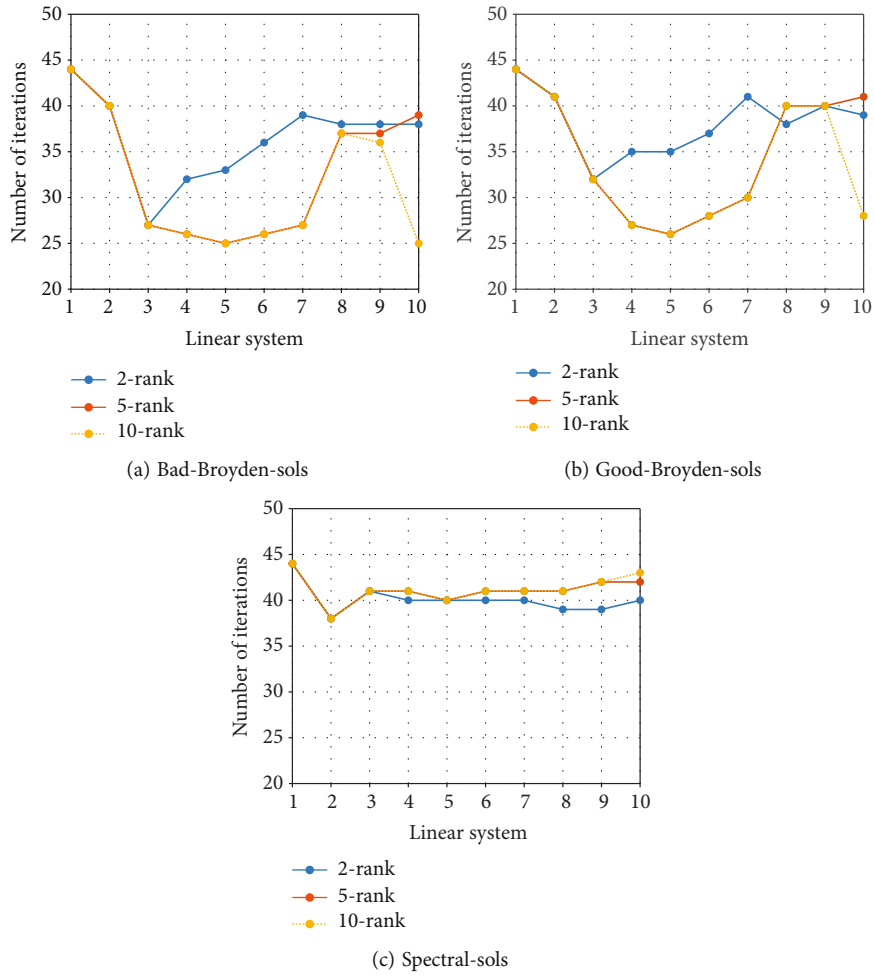


FIGURE 4: Number of iterations of the LGMRES method to solve the sequence of linear systems with the updated preconditioners and different rank corrections where P_0 = is ILU (0).

5. Numerical Results

5.1. 3D Langenbuch Transient. This transient was defined in [30]. It corresponds to an operational transient of the Langenbuch reactor. This reactor is a small LWR composed of 77 fuel assemblies and two types of fuel. Figure 1 shows the geometry of the reactor model, whose spatial discretization is composed of 1170 cells. The spatial discretization of the problem is made with a degree of the polynomials in the finite element method equal to 3 to obtain a problem of size $N = 69440$.

Materials 4 and 6 represent control rods. It has been initiated by the withdrawal of a bank of four partially inserted control rods (C1 in Figure 1) at a rate of 3 cm/s over $0 < t < 26.7$ s. A second bank of control rods (C2 in Figure 1) is inserted at the same rate over $7.5 < t < 47.5$ s. The transient is followed during 60 s. Figure 1(b) represents the axial profile at $t = 0.0$ s. The evolution of global power can be observed in Figure 2.

As a first case, a sequence of 10 linear systems is used to test the performance of the updated preconditioners. They are obtained from the backward difference method (Equation (4)) with $\Delta t = 1.0$ s. The positions corresponding to the different systems are marked in Figure 2 with red dots. GMRES with left preconditioning (LGMRES) is applied to solve the linear systems. The initial vectors in the iterative process are set equal to a vector of zeros for all systems. The tolerances for the stopping criterion have been set to $tol = 10^{-8}$ and the exit test is based on the unpreconditioned residual $\|b_i - A_i x_i\|_2 < tol$. In this problem, MATLAB is used to compute the numerical results.

First, we present the number of iterations to solve the first linear system $A_1 x_1 = b_1$ by using the LGMRES and updated preconditioners from 5 eigenvectors of A and $P_0 A$ associated with the smallest eigenvalues (Table 1). The eigenvalues are computed by using the command eigs of MATLAB. The initial preconditioner P_0 is set equal to the ILU (0) of the matrix A_1 obtained with the MATLAB command ilu. This table shows that all the updated preconditioners with the eigenvectors of $P_0 A_1$ and A improve the results obtained when the initial P_0 (with no update) is used. Interestingly, one can observe that using the eigenvectors of A_1 provides similar results as those obtained when the eigenvectors of $P_0 A_1$ are used.

Table 2: CPU times (s) of LGMRES to solve the sequence of 10 linear systems with the updated preconditioners and P_0 equal ILU (0). "P_i construction" collects the CPU time to construct the ILU (0) preconditioner of the first matrix and the CPU time to compute the subspace S_i. "P_i application" contains the total CPU time in the computation of the 10 linear systems devoted to apply the P_i. "Total" is the total CPU time spent to obtain the solution of the 10 linear systems.

Preconditioner	Rank	Total its.	CPU time (s)			
			P_i construction	P_i application	P_i application per it.	Total
ILU(A_1)	—	456	1.3	8.1	0.018	28.1
Bad-Broyden-sols	2	365	1.4	7.5	0.021	25.6
	5	328	1.5	6.8	0.021	23.3
	10	313	1.5	6.6	0.021	22.5
Good-Broyden-sols	2	356	1.4	7.5	0.021	23.8
	5	321	1.5	7.1	0.021	23.1
	10	308	1.5	7.0	0.022	22.6
Spectral-sols	2	401	1.4	8.1	0.020	26.9
	5	410	1.5	8.1	0.020	27.0
	10	412	1.5	8.2	0.020	27.2

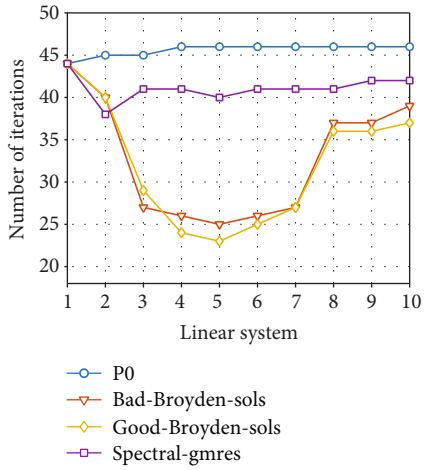


Figure 5: Number of LGMRES iterations to solve the sequence of linear systems with the updated preconditioners, 5-rank corrections and P_0 equal ILU (0).

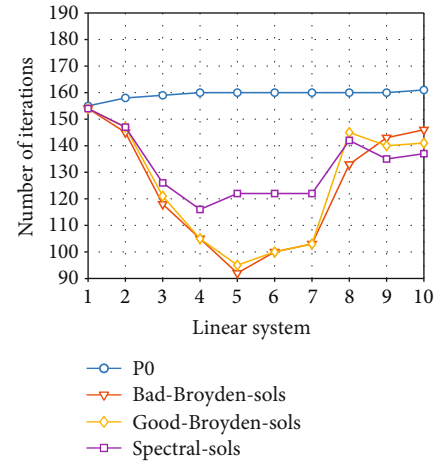


Figure 6: Number of iterations of LGMRES to solve the sequence of linear systems with the updated preconditioners, 5-rank corrections, and P_0 equal to the inverse of the diagonal of A_i.

Now, we test the performance of the updated preconditioners for the sequence of linear systems. In the following, we select the initial preconditioner P_0 as the ILU (0) preconditioner of matrix A_1 for all computations.

First, we test the type of vectors used to construct matrix S_i. Rank-5 corrections are taken in all cases in the updated preconditioners. Figure 3 shows the number of iterations of the LGMRES method needed to reach convergence with different choices of the columns of matrix S, namely, the eigenvectors of the matrix A_i (*eigs*), the Ritz vectors associated with the matrix A_i from the basis obtained from GMRES (*gmres*), or the Ritz vectors associated with matrix A_i and the basis obtained from the solutions of the previous linear systems (*sols*). Each subfigure reports the results obtained for each type of updated preconditioner (good or bad Broyden or spectral). The figures also show the number of iterations obtained if we apply only the preconditioner P_0 kept fixed without updating for all the linear systems in the sequence.

Broyden's preconditioners (Figures 3(a) and 3(b)) with the "exact" eigenvectors (*eigs*) provide a constant reduction of the number of iterations, with respect to using P_0 fixed, from the very first system. In contrast, the selection of the previous solutions (*sols*) and the vectors of the GMRES method (*gmres*) improves the convergence after a few linear systems are solved, even though from the 8th linear system, the improvement deteriorates. The 8th linear system corresponds to the transient at $t = 8$ s. At $t = 7.5$ s, a bank of control rods is inserted to shut down the reactor. Therefore, the subspace obtained with the 1-7th linear systems is not as useful to construct the preconditioner for the 8th linear system as we have observed for the systems 1-7. In the spectral preconditioner (Figure 3(c)), the reduction for all vectors of the number of iterations is more constant. For all updated preconditioners, the selection of the previous solutions (*sols*) speeds up the convergence of the LGMRES more than when the Ritz vectors constructed from GMRES (*gmres*) are used.

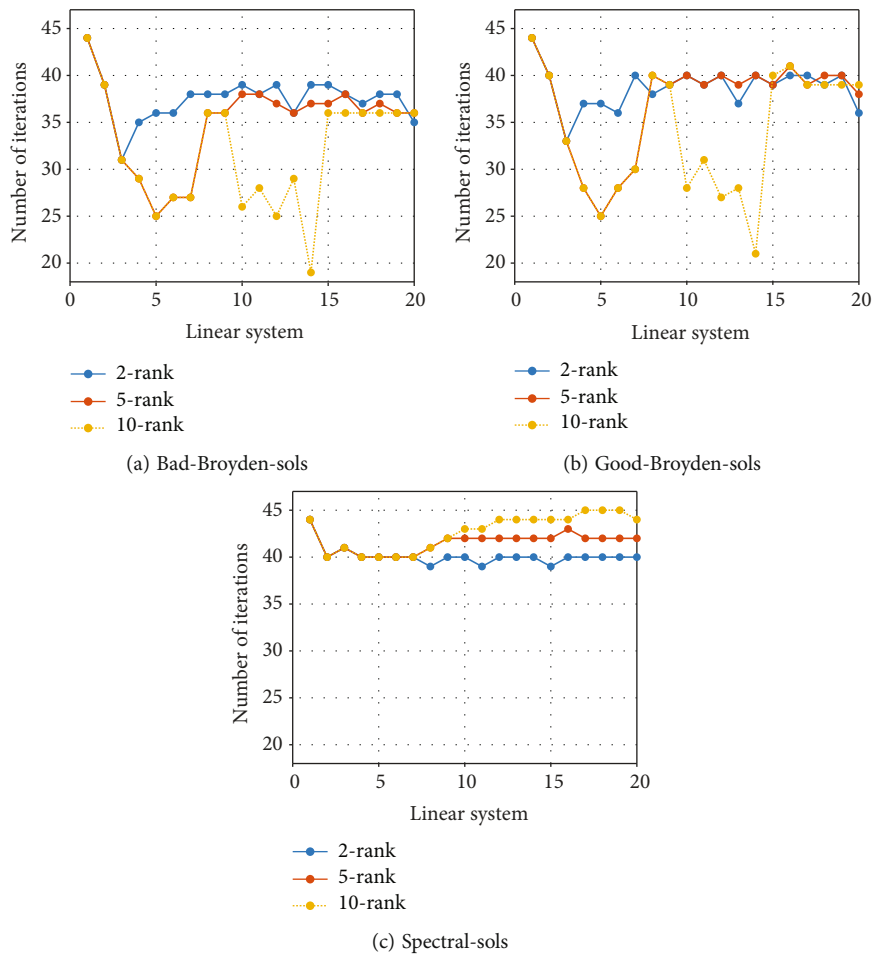


FIGURE 7: Number of iterations of the LGMRES method to solve the sequence of linear systems with the updated preconditioners and different rank corrections where P_0 = is ILU (0).

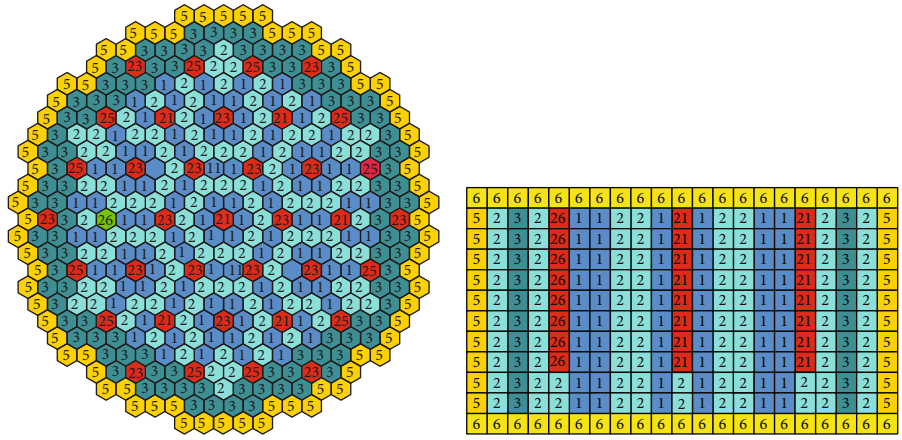
We investigate in Figure 4 the effect of using different maximum ranks (2, 5, 10) to construct the matrix S_i to correct the initial preconditioner. In this experiment, the columns of matrix S_i have been set using the previous solutions. For Broyden's preconditioners, the results show that corrections of maximum rank-5 improve the convergence. Corrections of maximum rank-10 only slighlty reduce the number of iterations because in two cases, the new solutions are linearly dependent from the previous ones, and the rank of the subspaces for these linear systems is quite similar (see Subsection 5.1.1 for an analysis of the effect of the maximum rank on the number of iterations to compute 20 linear systems).

Table 2 shows the total number of iterations and CPU times for solving the sequence of systems with different preconditioners and maximum ranks. We also report the CPU times devoted to construct and apply the preconditioners. We observe that the updated preconditioners are more convenient than using the fixed ILU(A_1) preconditioner, both in terms of number of iterations and CPU time, due also to the fact that the cost per iteration is only slightly increased by the preconditioner update. The best performance is obtained with the Broyden updates. In particular, the Good-Broyden updates are slightly more efficient when considering the number of iterations. This improvement will be more evident for more difficult cases, as documented in Section 5.2.

Figure 5 compares the number of iterations of LGMRES to solve the sequence of linear systems with the updated preconditioners constructed using rank-5 corrections and P_0 equal ILU (0). The Ritz vectors associated with the previous solutions are chosen as set S_i. Broyden's preconditioners (with similar results for both strategies) provide a smaller number of iterations than the spectral preconditioner.

We note that the studied updated preconditioners require only matrix-vector products if the application of the initial preconditioner P_0 relies on this operation. If P_0 is taken as the ILU (0) factorization, then the complete assembly of the matrix A_1 is required. P_0 can be set as the inverse of the diagonal of the matrices A_i to avoid assembling this matrix. The performance of the updated preconditioners with LGMRES when P_0 is equal to the inverse of the diagonal is displayed in Figure 6. This figure shows that the updated preconditioners decrease the number of iterations.

Moreover, note that, for this application, Broyden's preconditioners are better updating strategies than the spectral preconditioner, independently of the type of P_0 used.



(a) Radial profile (b) Axial profile

Figure 8: Geometry and materials distribution of the VVER-440 reactor.

However, one can observe that if the inverse of the diagonal is used, the number of iterations to reach the convergence is much higher than when the ILU (0) factorization is used. Selecting P_0 as a diagonal preconditioner can be suitable for huge problems with high memory demands.

5.1.1. Case 2: A Sequence of 20 Linear Systems. To conclude this section, we study the performance of the updated preconditioners for solving a sequence of 20 linear systems associated with the previous transient. Figure 7 reports the number of iterations obtained with the updated preconditioners by using different maximum ranks (2, 5, 10) to construct the matrix S_i. The columns of matrix S_i are set using the previous solutions. The initial preconditioner used is the ILU (0) of the matrix A_0. For this sequence, we can observe the improvement, in terms of reduction of the number of iterations, provided by the rank-10 Broyden preconditioners. Regarding the spectral preconditioner, increasing the rank does not translate in further reduction of the number of iterations.

5.2. AER-DYN-001 Problem. To test the efficiency of the updated preconditioner in a more realistic nuclear benchmark, the AER-DYN-001 transient is used. This problem was introduced in [31]. It corresponds to an asymmetric control rod ejection accident without feedback in a VVER440 reactor. The core has 250 cm height, with two reflector layers of 25 cm each added, one to the top and the other one to the bottom of the core. The assembly pitch is 14.7 cm. The core is a VVER-440 core type, with 25 fuel elements across the diameter. The disposition of materials together with the initial position of the control rods is shown in Figure 8. For the spatial discretization, the deal. II library cannot handle hexagonal cells, so each hexagon is subdivided into three quadrilaterals to obtain a total number of 15156 cells. Cubic polynomials are used for the finite element method to obtain algebraic problems of size 857882 degrees of freedom.

The transient is defined as follows. The control rod denoted by number 26 is ejected in the first 0.08 s with a velocity 25 m/s. Then, scram is initiated, inserting the safety rods 23 and 25 at $t = 1.0$ s with a velocity 0.25 m/s, so that the bottom position is reached at $t = 11.0$ s. The drop of control rod group 21 is also started at 1.0 s with the same velocity. The transient is followed during 4 s. Figure 9 represents the global power of the transient. For the time discretization, $\Delta t = 0.01$ s is used to obtain a squence of 400 linear systems to be solved. The methodology for this reactor has been implemented in C++ by using Deal.II and PETSc structures. The computer for the calculations is an Intel Core i7-4790 @ 3.60 GHz $\times 8$ processor with 32 Gb of RAM running on Ubuntu GNU/Linux 18.04 LTS.

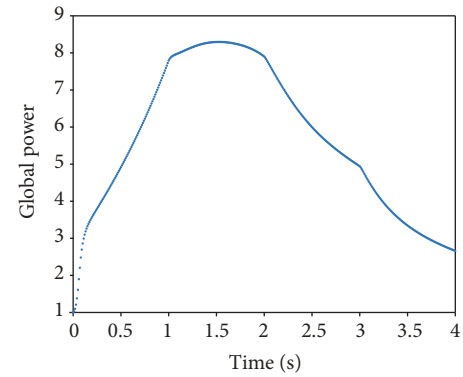
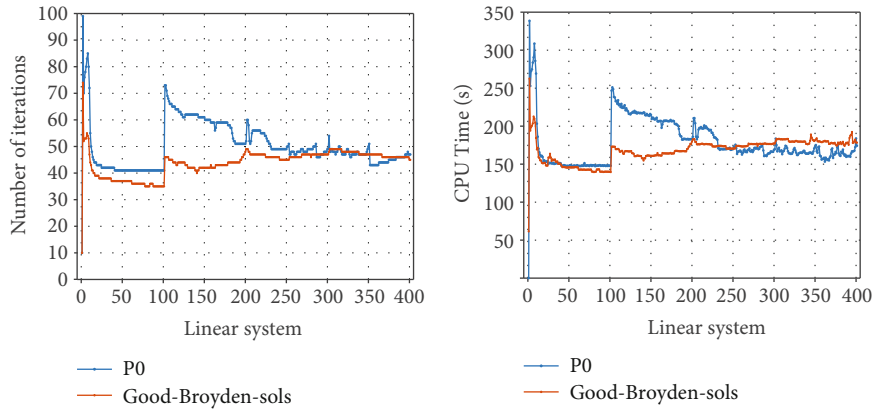


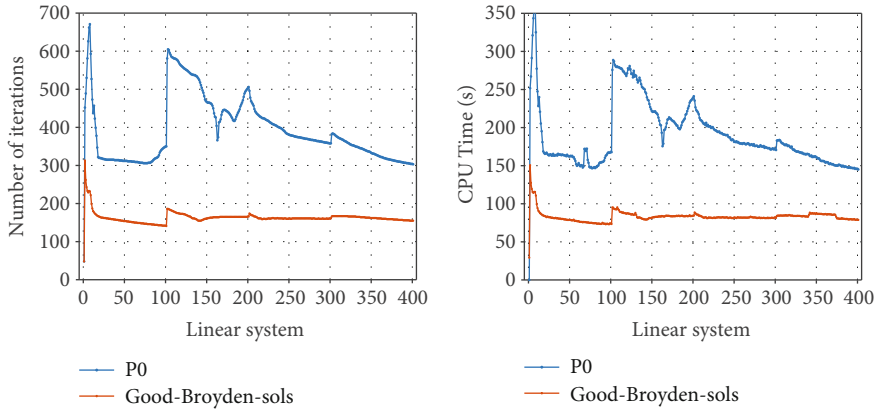
Figure 9: Global power of the AER-DYN-001 transient.

In this problem, we use the Good-Broyden preconditioner, where the columns of S_i are the five Ritz vectors associated with the smallest Ritz values of A_i and the previous solutions. The left GMRES of the PETSc library is applied to solve the linear systems. For the initial preconditioner, first, we use as P_0 the Block Gauss-Seidel preconditioner of A_1 ($BGS(A_1)$), unless the number of iterations of a linear system j is greater than 50. In such case, the new P_0 is computed as the Block Gauss-Seidel preconditioner of the matrix A_j. This preconditioner permits the use of a semi matrix-free implementation of the matrices (see [32] for more details). Figure 10 shows the number of iterations (Figure 10(a)) and the CPU time (Figure 10(b)) needed to reach convergence to the required accuracy for each linear system. The



(a) N. iterations (b) CPU time (s)

Figure 10: Number of iterations and CPU time using the Good-Broyden preconditioner versus the Block Gauss-Seidel preconditioner.



(a) N. iterations (b) CPU time (s)

Figure 11: Number of iterations and CPU time using the Good-Broyden preconditioner versus the Chebyshev preconditioner.

total CPU time of the computation is equal to 19.9 hours (no update) whereas it is equal to 18.7 hours if P_0 is updated by the Good-Broyden formula. In this case, the updated preconditioner does not provide a significant reduction neither of the number of iterations nor the CPU time.

However, as in the previous case, we propose an alternative to set P_0, which can be implemented using only matrix-vector products. Chebyshev polynomial preconditioners are recommended for parallel computations and matrix-free implementations [33]. In this work, five iterations of the Chebyshev preconditioner, whose implementation is provided by the Deal.II library [8], are applied. The application of the Chebyshev method requires estimating the largest eigenvalue of A_i. For that purpose, in this work, we use ten iterations of the GMRES method preconditioned with the inverse diagonal of A_i. Figure 11 shows the number of iterations (on the left) and the CPU time (on the right) that each linear system needs to reach convergence using P_0 either without updating or updated with the Good-Broyden method. The total CPU time is 21.2 hours, when the P_0 is not updated, while it reduces to 9.5 hours, with the Good-Broyden preconditioner. We observe a large improvement in the results if the P_0 (=chebyshev) is corrected.

Table 3: CPU time and mean number of iterations of GMRES to reach convergence with different preconditioners for the AER-DYN-001 transient.

	P_0=BGS(A_1)		P_0=Chebyshev(A_i)	
	No update	Good-Broyden	No update	Good-Broyden
CPU time (h)	19.9	18.6	21.2	9.5
Avg. GMRES iterations	50.3	43.8	391.5	162.6

Now, we compare these results (where $P_0 = chebyshev(A_i)$) with the previous (where $P_0 = BGS(A_1)$). Table 3 collects the total CPU times and the number of iterations that GMRES needed with each preconditioner to reach convergence. If the preconditioners are not updated, the Chebyshev preconditioner is not as efficient as Block Gauss-Seidel in terms of CPU time. However, if we use the Good-Broyden to correct the initial setting, the Chebyshev preconditioner, in addition to its negligible memory requirements, reduces the CPU time by more than a half.

6. Conclusions

This work describes and compares some strategies of updated preconditioners to speed up the computation of the sequence of linear systems obtained from the time discretization of the neutron diffusion equation. In particular, three types of techniques are considered: two tuned-type preconditioners, the Bad-Broyden and the Good-Broyden, and a spectral preconditioner. All of them are based on low-rank corrections of an initial preconditioner. Theoretically, they are constructed by matrix-vector multiplications that involve a set of eigenvectors of the preconditioned matrix. Moreover, based on different subspaces, cheaper options to define these preconditioners are tested.

Numerical results show that the low-rank updates presented in this work accelerate the convergence of the GMRES, with tuned preconditioners a better option than the spectral preconditioner. We found Broyden preconditioners to be less sensitive to the selection of the basis vectors used to construct the update in comparison with the spectral preconditioner. In fact, the vectors forming the basis for the updates for Broyden's preconditioner can be obtained from the solutions to the previous linear systems in the sequence. The cost-free choice proposed in this work, namely, the Ritz vectors extracted from the GMRES iteration, does not always guarantee a sufficient acceleration of the spectral preconditioner. In any case, low-rank corrections are generally enough to obtain efficient preconditioners for our problems.

The CPU times needed to apply the updated preconditioners are higher than those for applying the initial preconditioner only. However, these preconditioners enjoy a matrix-free implementation if the initial preconditioner can be applied in a matrix-free regime such as the Chebyshev preconditioner. In this case, low-rank updated preconditioners allow for great savings in storage and also on the CPU time needed to assemble the matrices of the sequence.

Results on a realistic nuclear benchmark requiring the solution of a sequence of 400 linear systems, each one of size of more than 8.5×10^5 degrees of freedom, show that the Good-Broyden update applied to the matrix-free Chebyshev preconditioner provides an impressive reduction of the CPU time taking 9.5 hours to complete the simulation vs. the 21.2 hours needed without updating.

Data Availability

The data used to support the findings of this study are included or referenced within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work has been partially supported by Spanish Ministerio de Economía y Competitividad under projects ENE2017-89029-P and MTM2017-85669-P. The second and third authors have been supported by the INdAM Research group GNCS project: *Optimization and Advanced Linear Algebra for Problems Governed by PDEs*. Furthermore, this work has been financed by the Generalitat Valenciana under the project PROMETEO/2018/035. We sincerely thank the three anonymous reviewers for helping us to significantly improve the manuscript.

References

[1] W. Stacey, *Nuclear Reactor Physics*, vol. 2, Wiley Online Library, 2007.

[2] A. Vidal-Ferràndiz, R. Fayez, D. Ginestar, and G. Verdú, "Moving meshes to solve the time-dependent neutron diffusion equation in hexagonal geometry," *Journal of Computational and Applied Mathematics*, vol. 291, pp. 197–208, 2016.

[3] D. Bertaccini, "Efficient preconditioning for sequences of parametric complex symmetric linear systems," *Electronic Transactions on Numerical Analysis*, vol. 18, pp. 49–64, 2004.

[4] C. Calgaro, J. P. Chehab, and Y. Saad, "Incremental incomplete LU factorizations with applications," *Numerical Linear Algebra with Applications*, vol. 17, no. 5, pp. 811–837, 2010.

[5] S. Bellavia, D. Bertaccini, and B. Morini, "Nonsymmetric preconditioner updates in Newton–Krylov methods for nonlinear systems," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2595–2619, 2011.

[6] S. Bellavia, V. De Simone, D. Di Serafino, and B. Morini, "Efficient preconditioner updates for shifted linear systems," *SIAM Journal on Scientific Computing*, vol. 33, no. 4, pp. 1785–1809, 2011.

[7] K. Ahuja, B. K. Clark, E. de Sturler, D. M. Ceperley, and J. Kim, "Improved scaling for quantum Monte Carlo on insulators," *SIAM Journal on Scientific Computing*, vol. 33, no. 4, pp. 1837–1859, 2011.

[8] W. Bangerth, R. Hartmann, and G. Kanschat, "deal.II—a general-purpose object-oriented finite element library," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 4, 2007.

[9] M. E. Kilmer and E. de Sturler, "Recycling subspace information for diffuse optical tomography," *SIAM Journal on Scientific Computing*, vol. 27, no. 6, pp. 2140–2166, 2006.

[10] A. Stathopoulos and K. Orginos, "Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics," *SIAM Journal on Scientific Computing*, vol. 32, no. 1, pp. 439–462, 2010.

[11] M. Benzi and D. Bertaccini, "Approximate inverse preconditioning for shifted linear systems," *BIT Numerical Mathematics*, vol. 43, no. 2, pp. 231–244, 2003.

[12] J. D. Tebbens and M. Tůma, "Efficient preconditioning of sequences of nonsymmetric linear systems," *SIAM Journal on Scientific Computing*, vol. 29, no. 5, pp. 1918–1941, 2007.

[13] J. Duintjer Tebbens and M. Tuma, "Preconditioner updates for solving sequences of linear systems in matrix-free environment," *Numerical Linear Algebra with Applications*, vol. 17, no. 6, pp. 997–1019, 2010.

[14] L. Bergamaschi and A. Martínez, "Parallel Newton–Chebyshev polynomial preconditioners for the conjugate gradient method," *Computational and Mathematical Methods*, vol. 3, no. 6, article e1153, 2021.

[15] A. Carr, E. de Sturler, and S. Gugercin, "Preconditioning parametrized linear systems," *SIAM Journal on Scientific Computing*, vol. 43, no. 3, pp. A2242–A2267, 2021.

[16] Y. Saad and M. H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[17] D. Bertaccini and M. K. Ng, "Band-Toeplitz preconditioned GMRES iterations for time-dependent PDEs," *BIT Numerical Mathematics*, vol. 43, no. 5, pp. 901–914, 2003.

[18] M. A. Freitag and A. Spence, "Convergence of inexact inverse iteration with application to preconditioned iterative solves," *BIT Numerical Mathematics*, vol. 47, no. 1, pp. 27–44, 2007.

[19] L. Bergamaschi, R. Bru, A. Martínez, and M. Putti, "Quasi-Newton preconditioners for the inexact Newton method," *Electronic Transactions on Numerical Analysis*, vol. 23, pp. 76–87, 2006.

[20] L. Bergamaschi, "A survey of low-rank updates of preconditioners for sequences of symmetric linear systems," *Algorithms*, vol. 13, no. 4, p. 100, 2020.

[21] S. Gratton, A. Sartenaer, and J. Tshimanga, "On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides," *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 912–935, 2011.

[22] A. Martínez, "Tuned preconditioners for the eigensolution of large SPD matrices arising in engineering problems," *Numerical Linear Algebra with Applications*, vol. 23, no. 3, pp. 427–443, 2016.

[23] L. Bergamaschi, R. Bru, and A. Martínez, "Low-rank update of preconditioners for the inexact Newton method with SPD Jacobian," *Mathematical and Computer Modelling*, vol. 54, no. 7–8, pp. 1863–1873, 2011.

[24] U. Yang and K. Gallivan, "A new family of preconditioned iterative solvers for nonsymmetric linear systems," *Applied Numerical Mathematics*, vol. 19, no. 3, pp. 287–317, 1995.

[25] S. Mercier, *Fast Nonlinear Solvers in Solid Mechanics [PhD Thesis]*, Université de Toulouse, Université Toulouse III-Paul Sabatier, Toulouse, 2015.

[26] B. Carpentieri, I. S. Duff, and L. Giraud, "A class of spectral two-level preconditioners," *SIAM Journal on Scientific Computing*, vol. 25, no. 2, pp. 749–765, 2003.

[27] L. Giraud, S. Gratton, and E. Martin, "Incremental spectral preconditioners for sequences of linear systems," *Applied Numerical Mathematics*, vol. 57, no. 11-12, pp. 1164–1180, 2007.

[28] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, "Recycling Krylov subspaces for sequences of linear systems," *SIAM Journal on Scientific Computing*, vol. 28, no. 5, pp. 1651–1674, 2006.

[29] P. F. Fischer, "Projection techniques for iterative solution of Ax = b with successive right- hand sides," *Computer Methods in Applied Mechanics and Engineering*, vol. 163, no. 1-4, pp. 193–204, 1998.

[30] S. Langenbuch, W. Maurer, and W. Werner, "Coarse-mesh flux-expansion method for the analysis of space-time effects in large light water reactor cores," *Nuclear Science and Engineering*, vol. 63, no. 4, pp. 437–456, 1977.

[31] A. Keresztúri and M. Telbisz, *A three dimensional hexagonal kinetic benchmark problem*, European Nuclear Society., 1992.

[32] A. Vidal-Ferràndiz, A. Carreño, D. Ginestar, and G. Verdú, "A block Arnoldi method for the SPN equations," *International Journal of Computer Mathematics*, vol. 97, no. 1-2, pp. 341–357, 2020.

[33] M. Adams, M. Brezina, J. Hu, and R. Tuminaro, "Parallel multigrid smoothing: polynomial versus Gauss-Seidel," *Journal of Computational Physics*, vol. 188, no. 2, pp. 593–610, 2003.