Multilevel matrix-free preconditioner to solve linear systems associated with a time-dependent SP_N equations

A. Carreño*, A. Vidal-Ferràndiz†, D. Ginestar† and G. Verdú*

* Instituto Universitario de Seguridad Industrial, Radiofísica y Medioambiental (ISIRYM)
Universitat Politècnica de València
Valencia, Spain
e-mail: amcarsan@iqn.upv.es, gverdu@iqn.upv.es

† Instituto Universitario de Matemática Multidisciplinar (IMM)
Universitat Politècnica de València
Valencia, Spain
e-mail: anvifer2@imm.upv.es, dginesta@mat.upv.es

Key words: Simplified spherical harmonics equations, Finite element method, Multilevel preconditioner, Matrix-free implementation

Abstract: *The evolution of the neutronic power inside of a nuclear reactor core can be approximated by means of the diffusive time-dependent simplified spherical harmonics equations (SP_N). For the spatial discretization of these equations, a continuous Galerkin high order finite element method is applied to obtain a semi-discrete system of equations that is usually stiff. A semi-implicit time scheme is used for the time discretization and many linear systems are needed to be solved and previously, preconditioned. The aim of this work is to speed up the convergence of the linear systems solver with a multilevel preconditioner that uses different degrees of the polynomials used in the finite element method. Furthermore, as the matrices that appear in this type of system are very large and sparse, a matrix-free implementation of the preconditioner is developed to avoid the full assembly of the matrices. A benchmark transient tests this methodology. Numerical results show, in comparison with the block Gauss-Seidel preconditioner, an improvement in terms of number of iterations and the necessity of computational resources.*

1 INTRODUCTION

Inside the reactor core, the evolution of the neutronic power can be modelled by means of the multigroup simplified spherical harmonics equations (SP_N). Different time formulations for this approximation of the neutron transport equation can be developed [15]. This work uses a formulation where the partial derivative of the even moments of the flux are neglected such that it can be seen as a generalized diffusive system with derivatives of order two in the space.

Spatially, the problem is discretized by applying a continuous Galerkin high order finite element method. Two sets of time-dependent differential equations are obtained, that for usual reactor systems are stiff. One related to the neutron moments and other related to the delayed neutron precursor concentrations. Therefore, implicit time schemes must be used [7] that require to solve large linear systems at each time-step. Krylov solvers such as the Generalized Minimal Residual (GMRES) [13] have been shown to be very efficient to solve such large sparse linear systems, if they are applied with a reasonable preconditioner.

Different preconditioners can be applied to these linear systems. First, one can apply classical preconditioners based on an incomplete matrix factorization, such as the ILU decomposition or the ICC decomposition. The linear systems associated with the SP_N equations have a block structure that also permits to apply block preconditioners such as the block Jacobi or the block Gauss-Seidel preconditioner [13]. Although these types of preconditioners are efficient, its implementation implies to store the matrices, or one part of them, in memory, which is very demanding.

https://doi.org/10.4995/YIC2021.2021.12510

Recently, multilevel methods are successfully applied to a wide range of problems. The levels are obtained either from different finite element discretizations on the original grid [5], from a hierarchy of coarser meshes [14] or from several levels of energy groups [6]. In this type of problems, the multigrid preconditioner can be applied, but the initial spatial meshes used in the computation are taken as coarse meshes and homogenized cross-sections must be redefined on each cell. That leads to an expensive application of the preconditioner [4]. Using a preconditioner based on several levels of energy groups is good option to integrate problems with a high number of energy groups. In this work, a two-level preconditioner based on different degrees of the polynomials used in the finite element discretization is applied.

On the other hand, the spatial discretization of a realistic nuclear reactor system with a high-order FEM produces huge algebraic matrices that require high demands of computational memory. Thus, a matrix-free technique can be used where the matrices are not allocated in memory and matrix-vector products are computed on the fly by using a cell-based interface. This technique does not only reduce the computational memory, but also it can reduce the matrix-vector multiplication runtimes in some computer architectures [10]. The main inconvenience of this technique is that algorithms to solve linear systems only can use matrix-vector products, since it is very difficult to access to particular elements of the matrices. In this work, a matrix-free implementation of the multilevel preconditioner is provided.

The rest of the paper is organized as follows. Section 2 presents the simplified spherical harmonics equations. Section 3 briefly exposes the finite element method used for the spatial discretization and the backward method used for the time-discretization. Section 4 explains the multilevel preconditioner. Section 5 describes some details about the implementation, in particular, about the matrix-free technique. Section 6 contains the numerical results obtained to test the proposed methodology. Finally, Section 7 collects the main conclusions of this work.

2 SIMPLIFIED P_N EQUATIONS

The diffusive time-dependent simplified harmonics (SP_N) equations can be written as [15]

$$\begin{aligned}\boldsymbol{\mathcal{V}}\frac{\partial}{\partial t}\phi^n - \vec{\nabla}\cdot\left(\frac{n\,(\boldsymbol{\mathcal{S}}^{n-1})^{-1}}{(2n+1)(2n-1)}\vec{\nabla}\left((n-1)\phi^{n-2}+n\phi^n\right) + \frac{(n+1)(\boldsymbol{\mathcal{S}}^{n+1})^{-1}}{(2n+1)(2n+3)}\vec{\nabla}\left((n+1)\phi^n\right.\right.\\ \left.\left.+\,(n+2)\phi^{n+2}\right)\right) + \boldsymbol{\mathcal{S}}^n\phi^n = \delta_{n0}\boldsymbol{\mathcal{F}}\phi^n + \delta_{n0}\sum_{k=1}^{K}\boldsymbol{\mathcal{M}}_k\boldsymbol{\mathcal{C}}_k\,, \qquad n=0,2,\ldots,N-1\,,\end{aligned} \tag{1}$$

and the equations for delayed neutron precursor concentration are

$$\frac{\partial}{\partial t}\boldsymbol{\mathcal{C}}_k = -\lambda_k^d\boldsymbol{\mathcal{C}}_k + \boldsymbol{\mathcal{R}}_k\phi^0\,, \qquad k=1,\ldots,K, \tag{2}$$

where

$$\boldsymbol{\mathcal{S}}^n = \begin{pmatrix}\Sigma_{t1}-\Sigma_{s11}^n & \cdots & -\Sigma_{sG1}^n\\ \vdots & \ddots & \vdots\\ -\Sigma_{s1G}^n & \cdots & \Sigma_{tG}-\Sigma_{sGG}^n\end{pmatrix}, \quad \boldsymbol{\mathcal{F}} = \begin{pmatrix}\chi_1^p(1-\beta^1)\nu_1\Sigma_{f1} & \cdots & \chi_1^p(1-\beta^G)\nu_G\Sigma_{fG}\\ \vdots & \ddots & \vdots\\ \chi_G^p(1-\beta^1)\nu_1\Sigma_{f1} & \cdots & \chi_G^p(1-\beta^G)\nu_G\Sigma_{fG}\end{pmatrix}, \tag{3}$$

$$\boldsymbol{\mathcal{V}} = \begin{pmatrix}1/v_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 1/v_G\end{pmatrix}, \quad \boldsymbol{\mathcal{M}}_k = \begin{pmatrix}\lambda_k^d\chi_1^{d,k}\\ \vdots\\ \lambda_k^d\chi_G^{d,k}\end{pmatrix}, \quad \boldsymbol{\mathcal{R}}_k = \begin{pmatrix}\beta_k^1\nu_1\Sigma_{f1}\\ \vdots\\ \beta_k^G\nu_G\Sigma_{fG}\end{pmatrix}^T, \quad \phi^n = \begin{pmatrix}\phi_1^n\\ \vdots\\ \phi_G^n\end{pmatrix}.$$

The variable $\phi_g^n = \phi_g^n(x,t)$ denotes the nth-moment of the neutron flux for the energy group g ($g = 1, \ldots, G$). $\boldsymbol{\mathcal{C}}_k$ denotes the delayed neutron precursor concentration of group k ($k = 1, \ldots, K$). The value of Σ_t is the total cross-sections that is approximated by the transport cross-section Σ_{tr}. Σ_f is the fission cross-section. The value of Σ_s^n is the nth-component of the scattering cross section in the spherical harmonics expansion. In this work, $\Sigma_s^n = 0, \forall n > 0$ because it is assumed that the scattering is isotropic. ν denotes the mean number of neutrons produced by fission. v_g are the neutron velocities. The spectrum of the prompt and the delayed neutrons are denoted by χ_g^p and $\chi_g^{d,k}$. The fraction of the delayed neutrons is β_k^g such that $\beta^g = \sum_{k=1}^K \beta_k^g$. The neutron precursor delayed constants are λ_k^d.

The linear change of variables proposed in [9] is applied to the Equation (1). In the case of the SP_3 equations, the change of variables is

$$U^1 = \phi^0 + 2\phi^2\,, \quad U^2 = 3\phi^2\,, \tag{4}$$

to obtain a system of the form

$$\mathbb{V}\frac{\partial}{\partial t}U - \vec{\nabla}\cdot\left(\mathbb{D}\vec{\nabla}U\right) + \mathbb{S}U = \mathbb{F}U + \mathbb{C}\,, \qquad U = \left(U^1, U^2\right)^T\,, \tag{5}$$

where

$$\mathbb{D} = \begin{pmatrix} \frac{1}{3}(\boldsymbol{\mathcal{S}}^1)^{-1} & 0 \\ 0 & \frac{1}{7}(\boldsymbol{\mathcal{S}}^3)^{-1} \end{pmatrix}, \quad \mathbb{S}_{ij} = \sum_{m=1}^{2} \mathbf{c}_{ij}^{(\mathbf{m})}\boldsymbol{\mathcal{S}}^m, \tag{6}$$

$$\mathbb{V}_{ij} = \sum_{m=1}^{2} \mathbf{c}_{ij}^{(\mathbf{m})}\boldsymbol{\mathcal{V}}, \quad \mathbb{F}_{ij} = \mathbf{c}_{ij}^{(\mathbf{1})}\boldsymbol{\mathcal{F}}, \quad \mathbb{C}_i = \mathbf{d}_i \sum_{k=1}^{K} \boldsymbol{\mathcal{M}}_k\boldsymbol{\mathcal{C}}_k, \tag{7}$$

and the coefficients matrix, $\mathbf{c}^{(\mathbf{m})}$ and vector $\mathbf{d}$ are

$$\mathbf{c}^{(\mathbf{1})} = \begin{pmatrix} 1 & -\frac{2}{3} \\ -\frac{2}{3} & \frac{4}{9} \end{pmatrix}, \quad \mathbf{c}^{(\mathbf{2})} = \begin{pmatrix} 0 & 0 \\ 0 & \frac{5}{9} \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} 1 \\ -\frac{2}{3} \end{pmatrix}. \tag{8}$$

3 SPATIAL AND TIME DISCRETIZATIONS

A high-order continuous Galerkin Finite Element Method (FEM) for the spatial discretization of the problem (5) is used. The discretization yields an semi-discrete time-dependent problem of the form

$$\mathbf{V}\frac{d}{dt}\tilde{U} + \mathbf{T}\tilde{U} = \mathbf{F}\tilde{U} + \mathbf{d}\sum_{k=1}^{K}\mathbf{M}_k C_k\,, \tag{9}$$

$$\frac{d}{dt}\mathbf{P}\mathbf{C}_k = -\mathbf{L}_k\mathbf{C}_k + \mathbf{R}_k\tilde{\phi}^0, \tag{10}$$

where $\mathbf{V}$, $\mathbf{T}$, $\mathbf{F}$, $\mathbf{M}_k$, $\mathbf{L}_k$ and $\mathbf{R}_k$ are the discretized operator of the $\mathbb{V}$, $-\vec{\nabla}\cdot\mathbb{D}\vec{\nabla} + \mathbb{S}$, $\mathbb{F}$, $\boldsymbol{\mathcal{M}}_k$, $\boldsymbol{\mathcal{L}}_k$ and $\boldsymbol{\mathcal{R}}_k$, respectively. The form of these operators will depend on the formulation. The mass matrix P is not the identity matrix because the basis of the FEM, that is composed of Lagrange polynomials, is not orthonormal. Vectors $\tilde{U}$ and C_k contain the discrete version of the moments U and the delayed neutron precursor concentration $\mathbb{C}_k$. The finite element method is implemented using `deal.II` library [3] and its structures. More details about FEM can be found in [19, 18]. Generally, these matrices are not symmetric, but they have a block structure provided by the different energy groups and neutronic field moments.

Given a configuration of a nuclear reactor, the time-dependent semi-discrete system (9) is generally stiff. Thus, implicit methods are used for its time discretization. In this work, a semi-implicit scheme is used where each type of equation is integrated independently.

The time interval $[0, T]$ is divided into several subintervals $[t_h, t_{h+1}]$ where $\Delta t_h = t_{h+1} - t_h$. The equation for the moments at $t = t_{h+1}$ (Equation (9)) is integrated by applying a backward difference of first order to the time derivative. The other magnitudes are substituted by its value at time t_{h+1}, excluding the concentration of neutron precursors that is substituted by its value at t_h. In this way, the solution of the linear system

$$\left(\frac{1}{\Delta t_h}\mathbf{V}^{h+1} + \mathbf{T}^{h+1} - \mathbf{F}^{h+1}\right) U^{h+1} = \frac{1}{\Delta t_h}\mathbf{V}^h U^h + \mathbf{d}\sum_{k=1}^{K}\mathbf{M}_k^{h+1}\mathbf{C}_k^h\,, \tag{11}$$

gives an approximation of the moments at time t_{h+1}.

This linear system has a size of $(N+1)\times N_{dofs}\times G/2$, where N_{dofs} are the degrees of freedom of the FEM. It is solved with the GMRES method provided by the PETSc library [2]. This work is devoted to study a multilevel preconditioner (described in Section 4) to precondition this system.

The concentration of delayed precursors equation is also integrated by using a one-step implicit scheme. The rest of the magnitudes are substituted by its value at t_{h+1}. Thus, the concentration of precursors can be approximated by solving the linear system

$$\left(\frac{1}{\Delta t_h}\mathbf{P} + \mathbf{L}_k^h\right)\mathbf{C}^{h+1} = \frac{1}{\Delta t_h}\mathbf{P}\mathbf{C}^h + \mathbf{R}_k^h\tilde{\phi}^{0,h+1}\,. \tag{12}$$

The matrices of this system are much smaller than the previous ones with a size equal to N_{dofs}. Thus, it is simply solved with the GMRES method and the ILU(0) preconditioner from the PETSc library [2].

4 MULTILEVEL PRECONDITIONER

The linear systems of Equation (11) need to be preconditioned to integrate the SP$_N$ equations with a reasonable CPU demand. In this work, we study a multilevel preconditioner based on the finite element method. This multilevel preconditioner is based on the classical V-cycle multigrid method [8]. In general, multigrid methods are designed to accelerate the convergence of a simple iterative method (known as smoothing, which reduces the short frequencies error) by a correction obtained when a coarse problem is solved (which is cheaper to solve). To solve this coarse problem, also a smoother and a coarser problem can be used, obtaining in this way a hierarchy of problems, known as levels of the multigrid method [14]. For nuclear reactor computations, using different meshes is not feasible because it does not require very refine meshes, and constructing coarse meshes implies homogenizing the reactor materials at each level. To avoid this problem, smaller problems are defined by considering a degree of the polynomial in the FEM, p^*, smaller than the original value p. The same spatial mesh is considered, but the simplified problem associated has a smaller number of degrees of freedom. This method only makes sense if $p > 1$. For these applications, we use a two-level method with one problem associated with each level because a degree in the FEM equal to 3 is enough to obtain accurate results. However, in other applications where higher degrees in FEM would be necessary, a multilevel preconditioner with more than two levels can be applied following a similar process than the multigrid method.

It is assumed that we want to define a preconditioner to solve the discrete SP$_N$ problem using a degree p in the FEM (first level),

$$\mathbf{A}x = b\,, \tag{13}$$

and we define the smaller problem with degree p^* in the FEM (second level)

$$\mathbf{A}^* x^* = b^* . \tag{14}$$

The two-level preconditioner smooths the iteration error on the original level and correct the iterate by a second level correction in a two-level setting. Recurrent applications on a sequence of levels lead to a multilevel procedure. The multilevel preconditioner can be used without smoothing [12]. In that case, the coarse level solve damps the low-frequency error of the fine problem, but not high-frequency errors. In a more physically sense, the smoother is applied to approximate the finer details.

To apply the multilevel preconditioner, we need to define a restriction operator, $\boldsymbol{\mathcal{R}}$, that interpolates vectors defined on the problem of FEM with degree p into a smaller vector associated with the problem with degree p^*. The values at the nodes of the FEM with degree p (original problem) are interpolated into the nodes associated with FEM with low degree of polynomial. This interpolation is made element by element through a transfer matrix, t^{res}, between the nodes associated with degree p of one element and the nodes associated with degree p^* of such element. The elements of such transfer matrix are given by

$$t^{res}_{ij} = \boldsymbol{\mathcal{N}}^p_j(\eta^*_i), \quad j = 1, \ldots, p+1, \quad i = 1, \ldots, p^*+1, \tag{15}$$

where $\boldsymbol{\mathcal{N}}^p_j$ are the Lagrange shape functions of the expansion which characterize the finite element method of degree p such that $\boldsymbol{\mathcal{N}}^p_j(\eta_k) = \delta_{jk}, \quad k = 1, \ldots, p+1$, being δ_{jk} the Kronecker delta function and η_k the position of the k-th node in the element associated with a degree p. η^*_i denotes the position of the i-th node in the element of the FEM with degree p^*.

In the other way, one can define the prolongation operator, $\boldsymbol{\mathcal{P}}$, also through a interpolation process. However, in this case, the elements of the transfer matrix, t^{pr}, that interpolates the nodes of one element associated with degree p^* into the ones associated with degree p, are given by

$$t^{pr}_{ij} = \boldsymbol{\mathcal{N}}^{p^*}_j(\eta_i), \quad j = 1, \ldots, p^*+1, \quad i = 1, \ldots, p+1, \tag{16}$$

where $\boldsymbol{\mathcal{N}}^{p^*}_i$ are the Lagrange shape functions of the expansion which characterize the finite element method of degree p_* and η_i is the position of the i-th node in the element associated with a degree p. Figure 1 shows a scheme of these operators for a two-dimensional problem.

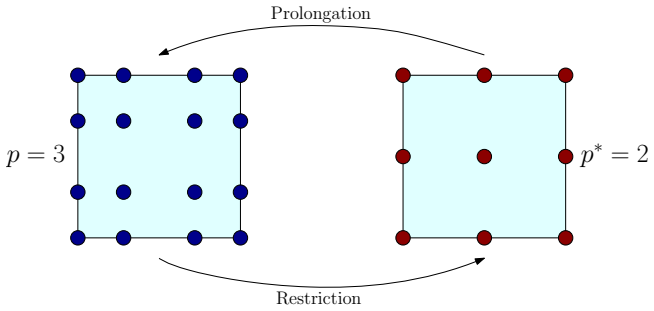


Figure 1: Restriction and prolongation operator of a element associated with degree $p = 3$ and $p^* = 2$ in the FEM.

The smoothing can be done with a Gauss-Seidel iterative method, but it needs to access to the matrix elements [4]. Therefore, Chebyshev polynomial smoothers are recommended for parallel computations and matrix-free implementations [1]. In particular, the implementation of the `deal.II` library is used [3]. The number of iterations for the smoother, its_{ch}, is set to $its_{ch} = 5$. To estimate the largest eigenvalue to apply the Chebyshev polynomial, ten iterations of the GMRES method preconditioned with the inverse diagonal of A are used.

To solve the small problem (14), the GMRES method is applied of the PETSc library [2]. To precondition the coarse problem, two type of strategies are tested. First, the coarse problem is assembled with a semi-matrix-free allocation and the block Gauss-Seidel preconditioner is used to solve this 'small' problem. Second, the matrix of the coarse problem is not assembled and the inverse of the diagonal elements is used to preconditioner the 'small' linear system. Note that to solve this small problem, if $p_* > 1$, we can also use a smaller degree to construct a smaller problem and repeat the process to obtain a three-level preconditioner.

The implementation of the particular case of two-level preconditioner is described in Algorithm 1.

Algorithm 1 Two-level preconditioner

Input: Vector x, matrices $\mathbf{A}$ and $\mathbf{A}^*$.

Output: Vector $y = \mathbf{P}x$ with $\mathbf{P} \approx \mathbf{A}^{-1}$ preconditioner of $\mathbf{A}$.

1: Pre-smooth $\mathbf{A}y = x$ with $its_{ch} = 5$ (Initialize the iterative method with $y_0 = 0$)
2: Restrict the residual $r = \mathbf{A}y - x$ to the second level by $r^* = \mathcal{R}(r)$
3: Solve $\mathbf{A}^* e^* = r^*$
4: Prolongate e^* by $e = \mathcal{P}(e^*)$
5: Correct $y = y + e$
6: Post-smooth $\mathbf{A}y = x$ with $its_{ch} = 5$ (Initialize the iterative method with $y^0 = y$)

The multilevel preconditioner is compared with the block Gauss-Seidel preconditioning (BGS). The number of moments and energy groups of the equations lead to a block structure of the matrices of the linear systems. In the block Gauss-Seidel preconditioner (BGS), each diagonal block is (approximately) solved with the conjugate gradient method and the incomplete Cholesky preconditioner. This preconditioner allows to save only the diagonal blocks and to use a semi-matrix-free implementation of the matrices [17].

5 MATRIX-FREE STRATEGY

A matrix-free strategy for the matrix $\mathbf{A}$ is applied to remove the computational cost of saving the matrices in memory. Nowadays, there are supercomputers without computational memory problems of capacity to solve this type of problem. In practice, they are available for some researching groups. In industrial sectors, such as nuclear engineering, there is a great interest in simulating the behaviour of the reactor without requiring high computational demands. On the other hand, this technique does not only reduce the computational memory, but also it can reduce the matrix-vector multiplication runtimes in some computer architectures [10]. Matrix-vector products are computed on the fly in a cell-based interface. For instance, we can consider a finite element Galerkin approximation, that leads to the matrix A, that takes a vector u as input and computes the integrals of the operator multiplied by trial functions to obtain the output vector is v. The operation can be expressed as a sum of N_c cell-based operations,

$$v = \mathbf{A}u = \sum_{c=1}^{N_c} \mathbf{P}_c^T \mathbf{A}^c \mathbf{P}_c u, \tag{17}$$

where $\mathbf{P}_c$ denotes the matrix that defines the location of cell-related degrees of freedom in the global vector and $\mathbf{A}^c$ denotes the submatrix of $\mathbf{A}$ on cell c. This sum is optimized through *sum-factorization*. Details about the implementation are explained in [10].

This type of implementation does not permit access to the matrix elements, which inabilities to use typical preconditioners such as ILU preconditioner. In this work, two types of allocations are used. First, the full matrix-free implementation (Full-MF) where any element of the

matrices are saved in memory. Second, a semi-matrix-free implementation (Semi-MF) where only the diagonal blocks of the matrices are assembled. The rest are implemented with the matrix-free technique.

6 NUMERICAL RESULTS

This Section tests the performance of the multilevel preconditioner in a transient defined from the movement of two banks of control rods in the tridimensional Langenbuch benchmark reactor [11]. The geometry of the reactor is modelled with 1170 cells. The transient is followed during 30 s.

The methodology has been implemented in C++ based on data structures provided by the libraries `deal.II` [3] and PETSc [2]. It has been incorporated to the open-source neutronic code `FEMFFUSION` [16]. For the computations, a computer with an Intel® Core™ i7-4790 @3.60GHz×8 processor with 32Gb of RAM running on Ubuntu 18.04 has been used.

Numerical results are presented to obtain the solution of the SP_1 and the SP_3 equations. Degree $p = 3$ in the polynomial of the FEM is used for the spatial discretization. Time-step for the backward method is set to $\Delta t_n = 0.1$ s to obtain a sequence of 300 linear systems. The two-level preconditioners are defined from small problems obtained by considering a degree in the FEM of the coarse problem (FEDC) equal to $p^* = 1$ and $p^* = 2$.

Table 1 shows the size of the matrices (Size($\mathbf{A}$)) for the different type of equations and degrees in the finite element method (FED). This Table shows as the full matrix-free implementation largely reduces the CPU memory required by the matrices. This property becomes more relevant as the problem is larger.

Table 1: Size of matrices associated with the linear systems and computational memory for the Full and Semi matrix-free implementations.

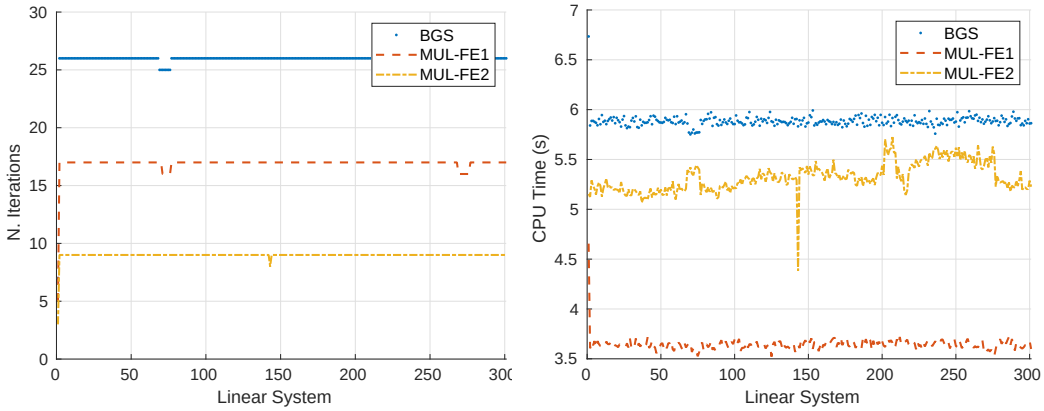
	FED=1			FED=2			FED=3		
Equations	Size (A)	CPU Memory		Size (A)	CPU Memory		Size (A)	CPU Memory	
		Full-MF	Semi-MF		Full-MF	Semi-MF		Full-MF	Semi-MF
SP_1	3080	0.07 MB	1 MB	21 546	0.16 MB	18 MB	69 440	0.34 MB	115 MB
SP_3	6160	0.14 MB	2 MB	43 092	0.23 MB	33 MB	138 880	0.40 MB	214 MB

First, we test the preconditioner used to solve the coarse problem in the multilevel preconditioner. Two types of implementation are compared: the BGS preconditioner with Semi-MF allocation and the inverse diagonal preconditioner with Full-MF allocation. Table 2 shows the mean number of iterations to solve the coarse problems with each preconditioner and the total CPU Time. The results are displayed to compute the sequence associated with the SP_1 equations if FEDC is equal to $p^* = 1$ and $p^* = 2$. Table 2 shows that the application of the BGS preconditioner solves the linear systems with much less iterations than using the inverse of the diagonal as preconditioner. However this preconditioner requires the assembly of the matrices and preconditioner. Numerical results shows that if FEDC is $p^* = 1$, the total CPU time of both implementations is similar. The CPU time of iterations is compensated by the assembling time. However, if FEDC is $p^* = 2$, the CPU time with the inverse diagonal is a bit higher, because in this last case the number of iterations needed by the inverse diagonal is 3 times higher than if BGS preconditioner is applied.

Table 2: Performance of the multilevel preconditioner to compute the sequence associated with the SP_1 equations.

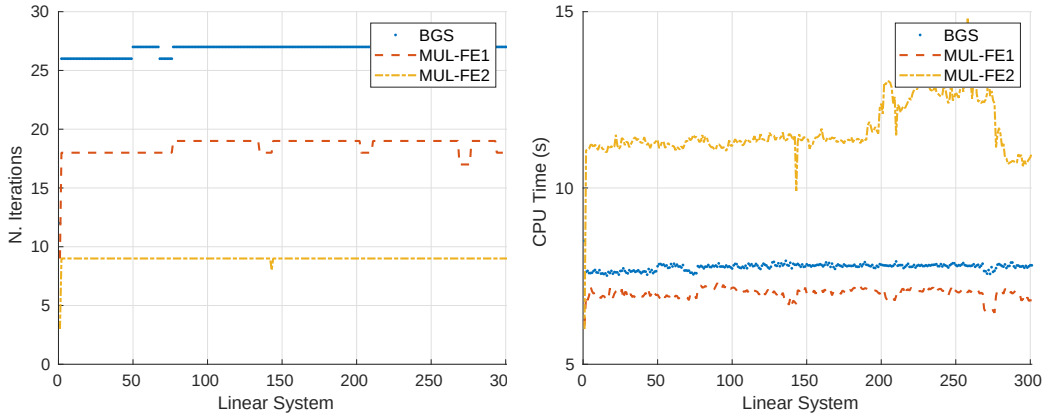
FEDC (p^*)	Type of preconditioner	Mean Number of Iterations	Total CPU time
1	BGS	8.12	1108 s
1	Inverse Diagonal	14.49	1096 s
2	BGS	19.31	1367 s
2	Inverse Diagonal	73.87	1602 s

Now, the multilevel preconditioner is compared with the BGS to solve the sequence of linear systems associated with the SP_1 and SP_3 equations. To apply the BGS and the multilevel preconditioner, the semi-matrix-free implementation and the full matrix-free implementation are used, respectively. The coarse problems in the multilevel preconditioner are solved with a full matrix-free implementation of the matrices and the inverse diagonal to have a full matrix-free implementation for the integration of the equations. Figure 2 shows the number of iterations (left) and the CPU time (right) needed by each type of preconditioner for every system in the sequence of the SP_1 equations. One can observe that the multilevel preconditioner reduces considerably the number of iterations required by the BGS, especially if the coarse problem is defined from a $p^* = 2$. However, in the CPU time, the most efficient preconditioner is the multilevel preconditioner with FEDC equal to $p^* = 1$, because the coarse problems are much smaller than the coarse problems with FEDC equal to $p^* = 2$ (Table 1). Figure 3 displays the results obtained in the sequence of systems associated with the SP_3 equations. Similar conclusions as the ones obtained for the SP_1 equations are obtained, even though the differences between the multilevel and FEDC equal to $p^* = 1$ and the BGS are not as high.



(a) N. iterations (b) CPU Time

Figure 2: Comparison of the multilevel preconditioner where the second level is obtained from $p_* = 1$ (MUL-FE1) and $p_* = 2$ (MUL-FE2), and the BGS preconditioner for the solution of the time-dependent SP_1 equations.



(a) N. iterations (b) CPU Time

Figure 3: Comparison of the multilevel preconditioner where the second level is obtained from $p_* = 1$ (MUL-FE1) and $p_* = 2$ (MUL-FE2), and the BGS preconditioner for the solution of the time-dependent SP_3 equations.

7 CONCLUSIONS

This work has proposed a two-level preconditioner based on different degrees in the polynomials of the finite element method to integrate the SP_N equations.

Numerical results have been shown the competitiveness of this multilevel preconditioner when it is compared with the block Gauss-Seidel preconditioner. Different coarse problems with degree $p^* = 1$ and $p^* = 2$ in the FEM are tested. Coarse problems with degree $p^* = 2$ are more convenient to reduce the number of iterations needed for the linear solver to converge. However, from a CPU time point of view, it is recommended to define the coarse problem from degree equal to $p^* = 1$. On the other hand, this two-level preconditioner can be applied by only using matrix-vector products allowing a full matrix-free implementation that removes the time to assembly the sparse matrices involved in the linear systems and reducing the necessary memory to allocate the matrices.

8 ACKNOWLEDGEMENTS

This work has been partially supported by Spanish Ministerio de Economía y Competitividad under projects ENE2017-89029-P and MTM2017-85669-P. Furthermore, this work has been financed by the Generalitat Valenciana under the project PROMETEO/2018/035.

REFERENCES

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss–Seidel. *Journal of Computational Physics*, 188(2):593–610, 2003.

[2] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, et al. PETSc users manual. 2019.

[3] W. Bangerth, T. Heister, and Kanschat G. `deal.II` *Differential Equations Analysis Library*. `http://www.dealii.org`.

[4] A. Carreño, A. Vidal-Ferràndiz, D. Ginestar, and G. Verdú. Block hybrid multilevel method to compute the dominant λ-modes of the neutron diffusion equation. *Annals of Nuclear Energy*, 121:513–524, 2018.

[5] B. Cockburn, O. Dubois, J. Gopalakrishnan, and S. Tan. Multigrid for an HDG method. *IMA Journal of Numerical Analysis*, 34(4):1386–1425, 2014.

[6] L.R. Cornejo, D.Y. Anistratov, and K. Smith. Iteration methods with multigrid in energy for eigenvalue neutron diffusion problems. *Nuclear Science and Engineering*, 2019.

[7] D. Ginestar, G. Verdú, V. Vidal, R. Bru, J. Marín, and J. L. Munoz-Cobo. High order backward discretization of the neutron diffusion equation. *Annals of Nuclear Energy*, 25(1-3):47–64, 1998.

[8] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[9] S.P. Hamilton and Thomas M. Evans. Efficient solution of the simplified pn equations. *Journal of Computational Physics*, 284:155–170, 2015.

[10] M. Kronbichler and K. Kormann. A generic interface for parallel cell-based finite element operator application. *Computers & Fluids*, 63:135–147, 2012.

[11] 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*, 63(4):437–456, 1977.

[12] J.A. Roberts and B. Forget. Multigroup diffusion preconditioners for multiplying fixed-source transport problems. *Journal of Computational Physics*, 274:455–472, 2014.

[13] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[14] R.S. Sampath and G. Biros. A parallel geometric multigrid method for finite elements on octree meshes. *SIAM Journal on Scientific Computing*, 32(3):1361–1392, 2010.

[15] W.M. Stacey. *Nuclear reactor physics*, volume 2. Wiley Online Library, 2007.

[16] A. Vidal-Ferràndiz, A. Carreño, D. Ginestar, and G. Verdú. `FEMFFUSION`: A finite element method code for the neutron diffusion equation. `https://www.femffusion.imm.upv.es`, 2020.

[17] A. Vidal-Ferràndiz, A. Carreño, D. Ginestar, and G. Verdú. A block arnoldi method for the spn equations. *International Journal of Computer Mathematics*, 97(1-2):341–357, 2020.

[18] A. Vidal-Ferràndiz, R. Fayez, D. Ginestar, and G. Verdú. Solution of the lambda modes problem of a nuclear power reactor using an h–p finite element method. *Annals of Nuclear Energy*, 72:338–349, 2014.

[19] 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*, 291:197–208, 2016.