A matrix-free eigenvalue solver for the multigroup neutron diffusion equation.

A. Carreño[1], A. Vidal-Ferràndiz[1], D. Ginestar[2], and G. Verdú[1]

[1] Instituto Universitario de Seguridad Industrial, Radiofísica y Medioambiental
Universitat Politècnica de València, València, Spain,
amcarsan@iqn.upv.es,anvifer2@upv.es , gverdu@iqn.upv.es
[2] Instituto Universitario de Matemática Multidisciplinar,
Universitat Politècnica de València, València, Spain,
dginesta@mat.upv.es

Abstract. The stationary neutron transport equation describes the neutron population and thus, the generated heat, inside a nuclear reactor core. Obtaining the solution of this equation requires to solve a generalized eigenvalue problem efficiently. The majority of the eigenvalue solvers use the factorization of the system matrices to construct preconditioners, such as the ILU decomposition or the ICC decomposition, to speed up the convergence of the methods. The storage of the involved matrices and incomplete factorization demands high quantities of computational memory although a the sparse format is used. This makes the computational memory the limiting factor for this kind of calculations in some personal computers. In this work, we propose a matrix-free preconditioned eigenvalue solver that does not need to have the matrices allocated in memory explicitly. This method is based on the block inverse-free preconditioned Arnoldi method (BIFPAM) with the innovation that uses a preconditioner that is applied from matrix-vector operations. As well as reducing enormously the computational memory, this methodology removes the time to assembly the sparse matrices involved in the system. A two-dimensional and three-dimensional benchmarks are used to study the performance of the methodology proposed.

Keywords: Neutron diffusion, Eigenvalue problem, Lambda modes, Matrix Free, Block method.

1 Introduction

The simulation of the reactor kinetics is a fundamental objective to ensure safe operation of nuclear reactors. The steady-state neutron transport equation [10] is the equation that describes the neutron flux and then, the generated heat power in every region of the reactor in steady-state.

Different equations have been successfully used to approximate the neutron transport equation. Usually, all eliminate the energy dependence of the equations by means of a multi-group approximation. The dependence on the direction of the neutrons depends on the selected method. In this work, the multigroup

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_68

neutron diffusion equation is chosen. This equation is analogous to the Fick's law for the diffusion of species and the Fourier equations in heat transfer. This equation states, for each energy group $g = 1, \ldots, G$, as

$$-\boldsymbol{\nabla} \cdot (D_g \boldsymbol{\nabla} \phi_g) + \Sigma_{tg}\, \phi_g(\boldsymbol{r}) = \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{s,\, g' \to g}\, \phi'_g(\boldsymbol{r}) + \frac{1}{\lambda} \sum_{g'=1}^{G} \chi_g \nu_g \Sigma_{fg'}\, \phi_{g'} \quad (1)$$

where ϕ_g denotes the neutron flux of the energy group g. The coefficients, D_g, Σ_{tg}, $\Sigma_{fg'}$, χ_g and $\Sigma_{s,\, g' \to g}$ depend of the energy group g and group g'.

The spatial discretization scheme chosen for the diffusion equations is a continuous Galerkin finite element method to obtain an algebraic generalized eigenvalue problem, where the largest eigenvalue λ, also known as k-effective (k_{eff}), shows the criticality of the reactor and its corresponding eigenvector the distribution of the flux in the reactor core. Moreover, it is interesting computing several modes to develop modal methods that allows integrating the time-dependent equation.

Most eigenvalue problems, that arise from the different approximations to deterministic neutron transport equations, have been classically solved with the power iteration method. However, recently the Krylov-Schur method [12], the Generalized Davidson [5] or the block inverse-free preconditioner Arnoldi method (BIFPAM) [2], are becoming increasingly popular for this type of computations. These methods permit to solve the eigenvalue problem faster than the power iteration when the spectral distribution of the eigenvalues is very clustered. The bottleneck of all these methods is the preconditioner used. The eigenvalue solvers can use the factorization of the system matrices to construct preconditioners, such as the ILU decomposition or the ICC decomposition, to speed up the convergence of the methods. However, this type of factorizations demands a high level of computational memory to assemble the matrices and the preconditioners. Henceforth, preconditioners based on matrix-vector product are needed. In this work, we use the BIFPAM with a matrix-free preconditioner based on the block Gauss-Seidel method and the Chebyshev polynomial that does not need to have the matrices constructed explicitly eliminating setup costs for the matrix assembly and reducing storage requirements. Other matrix-free approaches have been studied in [3,4,7].

2 Algebraic eigenvalue problem

The problem (1) is spatially discretized by means of a high-order continuous Galerkin Finite Element Method (FEM), implemented with the help of library `deal.II` [1]. The discretization transforms the differential problem into an algebraic generalized eigenvalue problem of the form

$$Ax = \lambda Bx. \quad (2)$$

More details about the finite element spatial discretization are explained in [12].

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_68

As we are solving an energy multigroup diffusion equations problem, we can take advantage of the block structure of the matrices A and B, where each block is symmetric and positive definite. Most of the blocks in the lower part of matrix A and far from the diagonal in B are zero. For example, the matrices of the C5G7 benchmark studied in Section **??** have the following block structure,

$$\begin{pmatrix} A_{11} & \cdots & A_{17} \\ A_{21} & \cdots & A_{27} \\ A_{31} & \cdots & A_{37} \\ A_{41} & \cdots & A_{47} \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \lambda \begin{pmatrix} B_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ B_{21} & B_{22} & 0 & 0 & 0 & 0 & 0 \\ B_{31} & B_{32} & B_{33} & 0 & 0 & 0 & 0 \\ B_{41} & B_{42} & B_{43} & B_{44} & B_{45} & 0 & 0 \\ 0 & 0 & 0 & B_{54} & B_{55} & B_{56} & 0 \\ 0 & 0 & 0 & 0 & B_{65} & B_{66} & B_{67} \\ 0 & 0 & 0 & 0 & 0 & B_{76} & B_{77} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}. \quad (3)$$

3 Matrix-free strategy

This strategy computes the matrix-vector products on the fly in a cell-based interface. For instance, we can consider that a finite element Galerkin approximation that leads to the block matrix $A_{b,b}$ 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 K cell-based operations,

$$v = A_{b,b}u = \sum_{k=1}^{K} P_k^T A_{b,b}^k P_k u$$

where P_k denotes the matrix that defines the location of cell-related degrees of freedom in the global vector and $A_{b,b}^k$ denotes the submatrix of $A_{b,b}$ on cell k. This sum is optimized through *sum-factorization*. Details about the implementation are explained in [6]. The main difficult of this strategy is to obtain efficient algebraic solvers that only use matrix-vector multiplications.

In this work, three matrix storage schemes are used. The first one, allocated all the block matrices in a compressed sparse row *CRS* way. The second one stores the diagonal block matrices of B in a sparse way to permit the computation of an incomplete LU factorization of these blocks. The rest of the blocks are implemented with the matrix-free operator (*non-diagonal*). Finally, all block matrices are implemented with the matrix-free technique in the *full matrix-free* scheme.

4 Eigenvalues solver

This section is devoted to present the preconditioned block algorithm based on the Block inverse-free preconditioned Arnoldi method (BIFPAM) for finding the q largest eigenvalues in magnitude and their corresponding eigenvectors of the generalized eigenvalue problem (2).

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_68

Given problem (2) and an initial block approximation

$$([\lambda_{0,1}, \lambda_{0,1}, \ldots, \lambda_{0,q}], [x_{0,1}, x_{0,1}, \ldots, x_{0,q}]),$$

one could obtain a new approximation $[x_{1,1}, x_{1,1}, \ldots, x_{1,q}]$ from the union of the p bases of the Krylov subspaces $K_m(A-\lambda_{0,i}B, x_{0,i})$, $1 \leq i \leq q$ of order m, where

$$K_m(A - \lambda_{0,i}B, x_{0,i}) := \text{span}\{x_{0,i}, (A - \lambda_{0,i}B)x_{0,i}, \ldots, (A - \lambda_{0,i}B)^m x_{0,i}\}.$$

Then, the original generalized eigenvalue problem is projected onto these bases. Arnoldi method is used to construct each basis K_m. This process is repeated to obtain the next iterations.

The convergence of this method improves with the application of a preconditioner of the matrix $C_{k,i} = A - \lambda_{k,i}B$. One can use preconditioners that come from the factorization of the matrices involved in the system, but it implies to assemble these matrices. In this work, we use an approximation of the matrix B^{-1} as preconditioner that it is shown in other works that it is more efficient than using a preconditioner of $C_{k,i}$ or $C_{1,1}$. This permits using a block preconditioner (by using the block structure of the matrix B), without assembling any additional matrix, with the advantage that the blocks of this matrix are symmetric and positive definite. This causes an improvement in the implementation of the Algorithm. In particular, we choose the block Gauss-Seidel preconditioner [9] as shown in Algorithm 1.

Algorithm 1 Block Gauss-Seidel preconditioner.

Input: Matrix B and vector $x = [x_1; \ldots; x_{Bl}]$.
Output: Vector $y = [y_1; \ldots; y_{Bl}]$, result of applying the preconditioner of B to x.

1: **for** $b = 1$ to $nblocks$ **do**
2: $t = x_b$
3: **for** $c = 1$ to b **do**
4: Compute $t = t - B_{b,c}y_c$
5: **end for**
6: Solve $B_{b,b}t = y_b$
7: **end for**

The previous algorithm only applies matrix-vector multiplications except in line 6, where some linear systems related with the block diagonal matrices are needed to be solved. In this work, the conjugate gradient (CG) is applied to solve these linear systems preconditioned with a Chebyshev preconditioner [11], provided by the library `deal.II` [1]. The degree of the Chebyshev polynomial has been set to 3. These auxiliary systems of the preconditioner are solved with a low maximum number of iterations (50).

5 Numerical Results

The performance of the proposed matrix-free method is tested with the C5G7 benchmark introduced by the Nuclear Energy Agency (NEA) [8] . In particular,

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_68

we test the 2D configuration version and the 3D configuration version. It consists of a nuclear reactor core with MOX and UO_2 square fuel assemblies surrounded by a moderator region.

First, we start with the 2D configuration benchmark. Figure 1 shows the proposed mesh used to spatially discretize the square regions. The area in gray color represents the neutron fuel region. The degree of the polynomial for the finite element method has been set to 2. The computation using the mesh of Figure 1 requires 28 900 finite element cells and 812 063 degrees of freedom. The number of eigenvalues computed is 4.

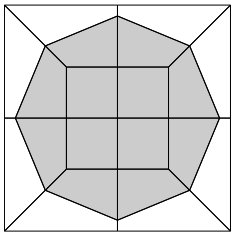

Fig. 1: Mesh considered in the FEM for the fuel pin in the C5G7 benchmark.

We compare the performance of the matrix-free preconditioner with the incomplete LU preconditioner. This last strategy is called 'CG-ILU'. This application needs the assembly of the block matrices B_{gg} to make ILU factorizations. Then, we propose to use the conjugate gradient method with the same number of maximum iterations, but in this case, the Chebyshev preconditioner is applied to solve linear systems. We denote this implementation as 'CG-CHEB'. For this last methodology, we compare three types of matrix-free implementations described in Section 3.

Table 1 displays the computational memory required by the matrix and preconditioner operators, the CPU time to set the matrices and the preconditioners and finally, the total CPU time to reach a residual error in the generalized eigenvalue problem less than 10^{-7} for each methodology. Table 1 shows that the CSR strategy is outperformed by the rest of the methodologies in terms of memory consumption. We can observe that the CG-ILU methodology solves the problem in the fastest way but this implementation does not allow to reduce the computational memory. If the 'full matrix-free' type is considered the computational memory is greatly reduced but the computational time is increased by a factor of 6 respect to 'CG-ILU' and a factor of 2 respect to the 'Non-Diag.' implementation.

In the next, we consider the 3D version of the C5G7 benchmark problem. This problem has the same radial configuration as the two dimensional version, and then, the discretization is used in this direction has been the same than the 2D version (Figure 1). The axial discretization is done by extruding the two dimensional mesh by axial plane. The mesh used has 264 992 finite element cells

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_68

Table 1: Computational results for the 2D-C5G7.

Methodology	Matrix-free Type	Mat. Memory	Prec. Memory	Mat.+Prec. CPU time	Total CPU time
CG-ILU	Non-Diag.	193 Mb	160 Mb	1s	184 s
CG-CHEB	Non-Diag.	193 Mb	-	0.8s	570 s
CG-CHEB	Full Matrix-Free	20 Mb	-	0.1s	1110 s
CG-CHEB	CSR	2375 Mb	-	4.6s	573 s

and 2 343 865 degrees of freedom. The configuration of the finite element method to compute the solution for this case is the same than in the previous case. In this case the number of eigenvalues requested is 1.

Now, we compare the matrix-free strategy presented in this work with the other methodologies described. Table 2 displays the CPU memory and the CPU time required by the matrix and preconditioner to set the matrices and the preconditioners and the total CPU time to reach a residual error in the generalized eigenvalue problem less than 10^{-7}. The same pattern of results described above can be observed in this Table. However, for the 3D case the reduction of the memory is considerable. In this way, if the required memory for the problem is low the authors recommends to use the 'CG-ILU' preconditioner. In this case, the CSR strategy has not been computed due to its high memory demands. However, for personal computers with low memory resources the full matrix-free strategy with the 'CG-CHEB' preconditioner is a reasonable option to solve the problem.

Table 2: Computational results for the 3D-C5G7 with $r = 1$.

Methodology	Matrix-free Type	Mat. Memory	Prec. Memory	Mat.+Prec. CPU time	Total CPU time
CG-ILU	Non-Diag.	13415 Mb	10668 Mb	143s	608 s
CG-CHEB	Non-Diag.	13415 Mb	-	62s	3534 s
CG-CHEB	Full Matrix-Free	2410 Mb	-	4s	7224 s

6 Conclusions

The multigroup neutron diffusion equation has been selected to approximate the neutron transport equation. A finite element method is used to discretize the problem obtaining an algebraic generalized eigenvalue problem. In this work, we propose a matrix-free preconditioned eigenvalue solver that does not need to have the matrices allocated in memory explicitly. This method is based on the BIFPAM

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_68

and uses as preconditioner the Chebyshev process. Moreover, different types of matrix-free implementation for the problem are proposed. The performance of these schemes are tested by using the C5G7 benchmark.

Numerical results concludes that the ILU preconditioner is more efficient than the Chebyshev preconditioner in terms of CPU time, but needs to access to assembled block diagonal matrix elements. However, the Chebyshev preconditioner, although is less efficient, can be implemented without any block matrix assembled and removes the time to assembly the sparse matrices involved in the system. The reduction of the memory in the full matrix-free strategy is greater as the size of the problem increases. It means, if the size of the problem is low, the authors recommends to use the 'CG-ILU' preconditioner. On the other hand, for the personal computers (with low memory resources) the full matrix-free strategy with the 'CG-CHEB' preconditioner is a reasonable option to solve the problem.

Adknowledgements

This work has been partially supported by the Ministerio de Economía y Competitividad under projects ENE2014-59442-P, MTM2014-58159-P and BES-2015-072901 and the Universitat Politècnica de València under project PAID-06-18.

References

1. W. Bangerth, R. Hartmann, and G. Kanschat. Deal.II – a general-purpose object-oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 2007.
2. 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.
3. V. De Simone and D. di Serafino. A matrix-free approach to build band preconditioners for large-scale bound-constrained optimization. *Journal of Computational and Applied Mathematics*, 268:82–92, 2014.
4. J. Gondzio. Matrix-free interior point method. *Computational Optimization and Applications*, 51(2):457–480, 2012.
5. S. P Hamilton and T. M Evans. Efficient solution of the simplified pn equations. *Journal of Computational Physics*, 284:155–170, 2015.
6. M. Kronbichler and K. Kormann. A generic interface for parallel cell-based finite element operator application. *Computers & Fluids*, 63:135–147, 2012.
7. M. Kronbichler and W. A Wall. A performance comparison of continuous and discontinuous galerkin methods with fast multigrid solvers. *SIAM Journal on Scientific Computing*, 40(5):A3423–A3448, 2018.
8. OECD/NEA. Benchmark on deterministic transport calculations without spatial homogenisation. Technical Report NEA/NSC/DOC(2003)16, NEA, 2003.
9. Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 2nd edn. edition, 2003.
10. W. M. Stacey. *Nuclear Reactor Physics*. John Wiley & Sons, Germany, 2007.
11. R. S. Varga. *Matrix iterative analysis*. Springer, Berlin, 2nd edition, 2009.
12. 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. *Ann. Nucl. Energy*, 72:338–349, 2014.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_68