Calculation of multiple eigenvalues of the neutron diffusion equation discretized with a parallelized finite volume method

Alvaro Bernal[a,*], Jose E. Roman[b], Rafael Miró[a], Gumersindo Verdú[a]

[a] *Institute for Industrial, Radiophysical and Environmental Safety, Universitat Politècnica de València, Camí de Vera s/n, Valencia, Spain*
[b] *Department of Information Systems and Computation, Universitat Politècnica de València, Camí de Vera s/n, Valencia, Spain*

Abstract

The spatial distribution of the neutron flux within the core of nuclear reactors is a key factor in nuclear safety. The easiest and fastest way to determine it is by solving the eigenvalue problem of the neutron diffusion equation, which only contains spatial derivatives. The approximation of these derivatives is performed by discretizing the geometry and using numerical methods. In this work, the authors used a finite volume method based on a polynomial expansion of the neutron flux. Once these terms are discretized, a set of matrix equations is obtained, which constitutes the eigenvalue problem. A very effective class of methods for the solution of eigenvalue problems are those based on projection onto a low-dimensional subspace, such as Krylov subspaces. Thus, the SLEPc library was used for solving the eigenvalue problem by means of the Krylov-Schur method, which also uses projection methods of PETSc for solving linear systems. This work includes a complete sensitivity analysis of different issues: mesh, polynomial terms, linear systems solvers and parallelization.

Keywords: `eigenvalue problem`, `neutron diffusion equation`, `finite volume method`, `Krylov subspaces`

*Corresponding author
Email addresses: `abernal@iqn.upv.es` (Alvaro Bernal), `jroman@dsic.upv.es` (Jose E. Roman), `rmiro@iqn.upv.es` (Rafael Miró), `gverdu@iqn.upv.es` (Gumersindo Verdú)

1. Introduction

The spatial distribution of the neutron flux within nuclear reactor core is a key factor in nuclear safety, since it is related to the power. This distribution can be calculated by means of Monte Carlo or deterministic methods. The latter ones solve the integro-differential neutron transport equation and they require typically less computational resources than the former ones. The deterministic method most widely used in Reactor Physics is the neutron diffusion theory, which is a simplification of the neutron transport theory based on Fick's Law [1].

The neutron diffusion equation is a partial differential equation containing temporal and spatial partial derivatives. The spatial distribution can be obtained by using the separation of variables technique for the temporal and spatial terms. Thus, the spatial distribution of the neutron flux is obtained by solving an eigenvalue problem, which only contains spatial derivative terms, and is also the solution of the steady-state. The approximation of the spatial derivative terms is performed by discretizing the geometry and using discretization methods.

There is a large number of free and open-source or commercial mesh generators, such as Gmsh, enGrid, Netgen, Discretizer, snappyHexMesh, ICEM-CFSD, CUBIT,etc. In this work, the authors used Gmsh [2], which is a free 3D finite element grid generator with a built-in Computer-aided design engine and post-processor. The major advantages of Gmsh are that it is fast, user-friendly and can generate tetrahedral and hexahedral meshes.

A huge variety of discretization methods can be applied to the eigenvalue problem of the neutron diffusion equation, such as finite difference, finite element, finite volume or nodal methods, as discussed by several authors [1],[3]. The nodal methods [4] are the most popular numerical techniques used to solve the neutron diffusion equation, which give accurate results in structured meshes. However, the application of these methods in unstructured meshes dealing with complex geometries is not straightforward and may cause problems of stability

and convergence of the solution [5]. In this work, the authors used a finite volume method, because it can be easily applied to unstructured meshes and is typically used in the transport equations due to the conservation of the transported quantity within the volume. In fact, the application of the finite volume method to the neutron diffusion equation is feasible, as demonstrated in [6].

Moreover, the neutron diffusion theory applied to discretized geometries requires additional equations at the interfaces of two cells: neutron flux and neutron current continuity, which imply an excess of equations. The neutron current is calculated by means of Fick's Law, which is proportional to the neutron flux gradient, and the proportionality constant depends on the cell material [1]. In this work, the calculation of the gradient is perfomed by using a polynomial expansion of the neutron flux in each cell of the discretized geometry [7, 8]. By means of this method, one obtains the same number of equations as unknowns and the gradient is calculated analytically. This method is explained in Section 2. In this polynomial expansion, the polynomial terms for each cell were assigned previously and the coefficients of the expansion are determined by solving the eigenvalue problem. The authors performed a sensitivity analysis for determining the best set of polynomial terms.

For the solution of the eigenvalue problem, a very effective class of methods are those based on projection onto a low-dimensional subspace, such as Krylov subspaces. There are several softwares and libraries containing the algorithm of these methods, which have been widely used. Currently, the state of the art for calculating eigenvalue problems is the SLEPc library [9, 10]. SLEPc, the Scalable Library for Eigenvalue Problem Computations, is a software library for the solution of large, sparse eigenproblems on parallel computers. It provides projection methods or other methods with similar properties, such as Krylov-Schur or Jacobi-Davidson. SLEPc is built on top of PETSc (Portable, Extensible Toolkit for Scientific Computation) [11] and extends it with all the functionality necessary for the solution of eigenvalue problems, which includes matrix operations and solution of linear systems. In this work, the authors used the Krylov-Schur algorithm of SLEPc for solving the eigenvalue problem

and different methods for solving the linear systems. Further details on these methods are given in Section 2.

Finally, the authors applied the method to a VVER reactor, which has hexagonal geometry. The authors already tested the method in other type of reactors in previous works [7],[8], which have Cartesian geometry. The polynomial expansion in these reactors with Cartesian geometry is composed of the following terms: 1, x, y, z, x^2, y^2, z^2. However, in VVER reactors, the polynomial expansion is composed of other terms, as discussed in Section 2. The polynomial terms used in VVER reactors give matrices which have excellent condition numbers, as shown in Section 3.4.

The novelty of this work includes two issues. First, a complete sensitivity analysis of the method applied to VVER reactors, involving the polynomial expansion, mesh and linear systems. Second, the parallelization of the method, showing excellent results of speedup, as shown in Section 3.

The outline of the paper is as follows. Section 2 explains the numerical methods, which is divided into two subsections. Subsection 2.1 shows the finite volume discretization of the neutron diffusion equation. Subsection 2.2 describes the matrices used in the eigenvalue problem and how it is solved. Section 3 defines the reactor used for the validation of the method and displays the results. Section 4 summarizes the major conclusions of this work.

2. Method

2.1. Neutron diffusion equation discretized with the Finite Volume Method

There are several approaches of the neutron diffusion equation depending on the energy discretization. The most commonly used in commercial nuclear reactors is the two energy group discretization. Eqs.(1)-(2) show the steady-state, two energy group discretization, neutron diffusion equation, without upscattering, for each energy group respectively. One can extend this method to more energy groups.

$$
\begin{aligned}
0 &= -\nabla \vec{J}_1(\vec{r}) - \left(\Sigma_{a,1}(\vec{r}) + \Sigma_{s,1\to 2}(\vec{r})\right)\phi_1(\vec{r}) + \\
&\quad + \frac{1}{\mathbf{k}}\left(\nu\Sigma_{f,1}(\vec{r})\phi_1(\vec{r}) + \nu\Sigma_{f,2}(\vec{r})\phi_2(\vec{r})\right) \qquad (1)
\end{aligned}
$$

$$
0 = -\nabla \vec{J}_2(\vec{r}) - \Sigma_{a,2}(\vec{r})\phi_2(\vec{r}) + \Sigma_{s,1\to 2}(\vec{r})\phi_1(\vec{r}) \qquad (2)
$$

In Eqs.(1)-(2), $\Sigma_{a,g}$ is the absorption macroscopic cross section of the neutrons of the g energy group, $\Sigma_{s,1\to 2}$ is the scattering macroscopic cross section of the neutrons from the first energy group to the second one, $\nu\Sigma_{f,g}$ is the nu-fission macroscopic cross section of the neutrons of the g energy group, ϕ_g is the neutron flux of the g energy group, $\vec{J}_g$ is the neutron current of the g energy group. Although ϕ_g is called neutron flux, it is a scalar variable. The macroscopic cross sections are coefficients depending on the material, the energy group and the nuclear reactions. They are related to the probability of a particular nuclear reaction of neutrons with target nuclei and their units are cm^{-1}. Absorption reactions eliminate neutrons, while nu-fission reactions produce neutrons by fission. Scattering reactions from the first energy group to the second one eliminate neutrons of the first energy group and produce them in the second energy group. The unit of the neutron flux is $cm^{-2}s^{-1}$. These equations represent an eigenvalue problem, in which $\mathbf{k}$ is the eigenvalue and ϕ_g is the eigenvector. The neutron current is, according to Fick's law, proportional to the gradient of the neutron flux, as shown in Eq.(3). In this equation, D_g is called the diffusion coefficient, its unit is cm and depends on the material and energy group.

$$
\vec{J}_g(\vec{r}) = -D_g(\vec{r})\vec{\nabla}\phi_g(\vec{r}) \qquad (3)
$$

Numerical methods should be applied to Eqs.(1)-(2) to obtain algebraic terms. Firstly, the geometry is discretized by using a mesh generator, such as Gmsh [2]. The discretized geometry will be composed of cells such as tetrahedra or hexahedra, containing only one homogenized material. Secondly, the

finite volume method and divergence theorem are applied to Eqs.(1)-(2), producing Eqs.(4)-(5) for each cell i. In these equations, n_f is the number of faces for each cell, S_j is the area of the face j of cell i, $J_{g,i,j}$ is the face averaged value of $\vec{J}_g$ at face j of cell i, V_i is the volume of cell i, $\Sigma^i_{s,1\to2}$ is the value of $\Sigma_{s,1\to2}$ in cell i, $\nu\Sigma^i_{f,g}$ is the value of $\nu\Sigma_{f,g}$ in cell i and $\phi_{g,i}$ is the cell averaged value of ϕ_g in cell i.

$$\sum_{j=1}^{n_f} S_j J_{1,i,j} + V_i \left(\Sigma^i_{a,1} + \Sigma^i_{s,1\to2}\right)\phi_{1,i} = \frac{1}{\mathbf{k}} V_i \left(\nu\Sigma^i_{f,1}\phi_{1,i} + \nu\Sigma^i_{f,2}\phi_{2,i}\right) \quad (4)$$

$$\sum_{j=1}^{n_f} S_j J_{2,i,j} + V_i\Sigma^i_{a,2}\phi_{2,i} - V_i\Sigma^i_{s,1\to2}\phi_{1,i} = 0 \quad (5)$$

Equations (4)-(5) do not contain derivatives, but $J_{g,i,j}$ is not known. These values are obtained by applying the finite volume method to Eq.(3) producing Eq.(6). In this equation, D^i_g is the value of D_g in cell i and $\vec{\nabla}\phi_{g,i,j}$ is the face averaged value of the gradient of the neutron flux at face j for cell i. Eq.(6) does not contain derivatives, because $\vec{\nabla}\phi_{g,i,j}$ is a face averaged value.

$$J_{g,i,j} = -D^i_g \vec{\nabla}\phi_{g,i,j} \quad (6)$$

Since the geometry is discretized and is not homogeneous, additional interfaces equations are required, as discussed in Section 1. These equations are the neutron flux continuity and the current continuity for each energy group, which are expressed in Eqs.(7) and (8), for face j, which is adjacent to cells i and l.

$$\phi_{g,i,j} = \phi_{g,l,j} \quad (7)$$

$$J_{g,i,j} = -J_{g,l,j} \quad (8)$$

To determine the total number of equations, the authors consider a simple discretized geometry, like the one illustrated in Figure 1. In this figure, one can see cell equations, which are the diffusion equations, and the face equations:

boundary conditions, neutron flux continuity and neutron current continuity. From this figure, one can conclude that the number of equations for each energy group and cell is $n_f + 1$.

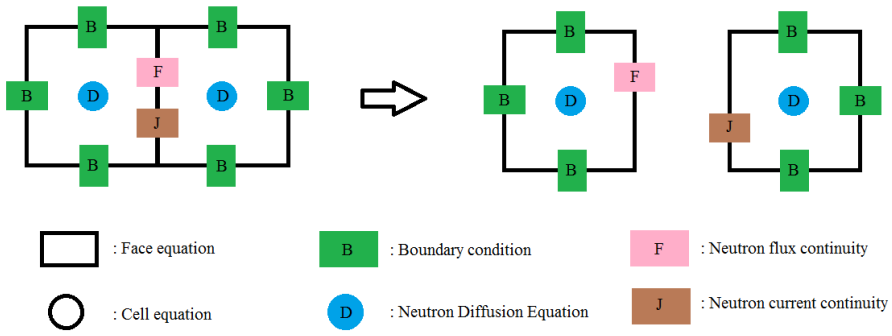

Figure 1: Equations applied to a discretized geometry.

If one considers that $\phi_{g,i,j}$ and $J_{g,i,j}$ can be calculated from $\phi_{g,i}$, there is only one unknown value for each energy group and cell: $\phi_{g,i}$. Thus, the authors proposed to expand the neutron flux in each cell to increase the number of unknowns, so one may obtain the same number of unknowns and equations, as expressed in Eq.(9) and discussed in [7]. In this expansion, the monomials $p_t(x, y, z) = x^{\alpha_t} y^{\beta_t} z^{\gamma_t}$ are fixed and the coefficients $a_{g,i,t}$ will be the unknowns, which will be determined by solving the eigenvalue problem. As $x^{\alpha_t} y^{\beta_t} z^{\gamma_t}$ are known, one can obtain $\phi_{g,i}$, $\phi_{g,i,j}$ and $\vec{\nabla}\phi_{g,i,j}$, as shown in Eqs.(10)-(12). In Eq.(12), $u_{i,j,x}$, $u_{i,j,y}$ and $u_{i,j,z}$ are the direction cosines of the normal to face j.

$$\phi_{g,i}(x, y, z) = \sum_{t=1}^{n_f+1} a_{g,i,t} p_t(x, y, z) = \sum_{t=1}^{n_f+1} a_{g,i,t} x^{\alpha_t} y^{\beta_t} z^{\gamma_t} \tag{9}$$

$$\phi_{g,i} = \frac{1}{V_i} \int_{V_i} \phi_{g,i}(x, y, z) dV = \sum_{t=1}^{n_f+1} a_{g,i,t} \frac{1}{V_i} \int_{V_i} p_t(x, y, z) dV = \sum_{t=1}^{n_f+1} a_{g,i,t} \overline{p}_t^{V_i} \tag{10}$$

$$\phi_{g,i,j} = \frac{1}{S_j}\int_{S_j}\phi_{g,i}(x,y,z)dS = \sum_{t=1}^{n_f+1} a_{g,i,t}\frac{1}{S_j}\int_{S_j} p_t(x,y,z)dS = \sum_{t=1}^{n_f+1} a_{g,i,t}\overline{p}_t^{S_{i,j}} \tag{11}$$

$$\begin{aligned}\vec{\nabla}\phi_{g,i,j} &= \frac{1}{S_j}\int_{S_j}\vec{\nabla}\phi_{g,i}(x,y,z)dS \\ &= \sum_{t=1}^{n_f+1} a_{g,i,t}\left(u_{i,j,x}\frac{1}{S_j}\int_{S_j}\frac{dp_t(x,y,z)}{dx}dS + u_{i,j,y}\frac{1}{S_j}\int_{S_j}\frac{dp_t(x,y,z)}{dy}dS\right. \\ &\quad \left. + u_{i,j,z}\frac{1}{S_j}\int_{S_j}\frac{dp_t(x,y,z)}{dz}dS\right) = \sum_{t=1}^{n_f+1} a_{g,i,t}\overline{\vec{\nabla}p_t}^{S_{i,j}}\end{aligned} \tag{12}$$

For fine discretization grid containing a large number of cells and interfaces, the number of equations will be extremely large. To reduce this number, one can define implicitly the boundary conditions and the current continuity, as proposed in [8]. Consequently, the number of equations for each energy group will be reduced to $N_c + N_f$, where N_c is the number of cells and N_f the number of interfaces, that is, the number of diffusion equations and flux continuity equations. As with the unknown values ($a_{g,i,t}$), the number of unknowns should be also reduced. For this purpose, coefficients $a_{g,i,t}$ will be calculated as a weighted sum of $\phi_{g,i}$ and $J_{g,i,j}$ and only one unknown current per each interface j will be considered: $J_{g,j} = J_{g,i,j} = -J_{g,l,j}$. For each interface j, whose adjacent cells are i and l, the direction of $J_{g,j}$ will be from cell i to cell l. So, the current continuity will be defined implicitly and the number of unknowns is reduced.

To calculate $a_{g,i,t}$ in terms of $\phi_{g,i}$ and $J_{g,j}$, one can formulate Eq.(13) for each cell i and energy group g. This equation contains the terms $f_{i,j,t}$ and $F_{g,i,j}$ defined in Eqs.(14) and (15) respectively. In Eqs.(14) and (15), $u_{i,j}$ is equal to 1 if $J_{g,i,j} = J_{g,j}$ and is equal to -1 if $J_{g,i,j} = -J_{g,j}$. Finally, one can obtain $a_{g,i,t}$

as in Eq.(16), where I_i^{-1} is the inverse of the matrix in Eq.13.

$$\begin{bmatrix} \overline{p}_1^{V_i} & \cdots & \overline{p}_{n_f+1}^{V_i} \\ f_{i,1,1} & \cdots & f_{i,1,n_f+1} \\ \vdots & & \vdots \\ f_{i,n_f,1} & \cdots & f_{i,n_f,n_f+1} \end{bmatrix} \begin{bmatrix} a_{g,i,1} \\ \vdots \\ a_{g,i,n_f+1} \end{bmatrix} = \begin{bmatrix} \phi_{g,i} \\ F_{g,i,1} \\ \vdots \\ F_{g,i,n_f} \end{bmatrix} \tag{13}$$

$$f_{i,j,t} = \begin{cases} \overline{p}_t^{S_{i,j}} & \text{if face } j \text{ is a boundary face of zero flux condition} \\ -u_{i,j} \overline{\vec{\nabla} p_t}^{S_{i,j}} & \text{the rest of cases} \end{cases} \tag{14}$$

$$F_{g,i,j} = \begin{cases} \phi_{g,i,j} & \text{if face } j \text{ is a boundary face of zero flux condition} \\ -u_{i,j} \vec{\nabla} \phi_{g,i,j} & \text{the rest of cases} \end{cases} \tag{15}$$

$$\begin{bmatrix} a_{g,i,1} \\ \vdots \\ a_{g,i,n_f+1} \end{bmatrix} = I_i^{-1} \begin{bmatrix} \phi_{g,i} \\ F_{g,i,1} \\ \vdots \\ F_{g,i,n_f} \end{bmatrix} \tag{16}$$

If one combines Eqs.(11) and (16), one obtains Eq.(17), which calculates $\phi_{g,i,j}$ in terms of $\phi_{g,i}$ and $F_{g,i,t}$. The coefficients $X_{i,j,k}$ are calculated as in Eq.(18). It is more convenient to multiply Eq.(17) by D_g^i, obtaining Eq.(19), because $D_g^i F_{g,i,j}$ can be expressed in terms of $J_{g,j}$ as in Eq.(20).

$$\phi_{g,i,j} = X_{i,j,1} \phi_{g,i} + \sum_{t=1}^{n_f} X_{i,j,t+1} F_{g,i,t} \tag{17}$$

$$X_{i,j,k} = \sum_{t=1}^{n_f+1} \overline{p}_t^{S_{i,j}} I_i^{-1}(t,k) \tag{18}$$

$$D_g^i \phi_{g,i,j} = X_{i,j,1} D_g^i \phi_{g,i} + \sum_{t=1}^{n_f} X_{i,j,t+1} D_g^i F_{g,i,t} \tag{19}$$

$$D_g^i F_{g,i,j} = \begin{cases} 0 & \text{if face } j \text{ is a boundary face} \\ u_{i,j} J_{g,j} & \text{if face } j \text{ is an inner face} \end{cases} \tag{20}$$

If boundary condition on face j is zero flux ($\phi_{g,i,j} = 0$), one should calculate $J_{g,i,j}$ on this face j. For doing this, one can combine Eqs.(6), (12) and (16), which gives Eq.(21). In this equation, coefficients $R_{i,j,k}$ are calculated as in Eq.(22).

$$J_{g,i,j} = R_{i,j,1} D_g^i \phi_{g,i} + \sum_{t=1}^{n_f} R_{i,j,t+1} D_g^i F_{g,i,t} \tag{21}$$

$$R_{i,j,k} = - \sum_{t=1}^{n_f+1} \overline{\vec{\nabla} p_t}^{S_{i,j}} I_i^{-1}(t,k) \tag{22}$$

If one substitutes Eq.(19) into Eq.(7), which is the flux continuity, one obtains Eq.(23). Finally, if one uses the implicit definition of the current ($J_{g,i,j} = u_{i,j} J_{g,j}$) in Eqs.(4)-(5), one obtains Eqs.(24)-(25). In conclusion, Eqs.(24)-(25) are the cell equations and Eq.(23) is the face equation of the final eigenvalue problem.

$$\begin{aligned} 0 &= \frac{D_g^i}{D_g^i} \phi_{g,i,j} - \frac{D_g^l}{D_g^l} \phi_{g,l,j} \\ &= \left(X_{i,j,1} \phi_{g,i} + \frac{1}{D_g^i} \sum_{t=1}^{n_f} X_{i,j,t+1} D_g^i F_{g,i,t} \right) \\ &\quad - \left(X_{l,j,1} \phi_{g,l} + \frac{1}{D_g^l} \sum_{t=1}^{n_f} X_{l,j,t+1} D_g^l F_{g,l,t} \right) \end{aligned} \tag{23}$$

$$\sum_{j=1}^{n_f} S_j u_{i,j} J_{1,j} + V_i \left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i \right) \phi_{1,i} = \frac{1}{\mathbf{k}} V_i \left(\nu \Sigma_{f,1}^i \phi_{1,i} + \nu \Sigma_{f,2}^i \phi_{2,i} \right) \tag{24}$$

$$\sum_{j=1}^{n_f} S_j u_{i,j} J_{2,j} + V_i \Sigma_{a,2}^i \phi_{2,i} - V_i \Sigma_{s,1\to 2}^i \phi_{1,i} = 0 \tag{25}$$

2.2. Eigenvalue problem of the neutron diffusion equation

Equations (23)-(25) can be arranged in matrix form as expressed in Eq.(26) for a cell i and face j, where $h_g = \frac{1}{D_g^i}\sum_{t=1}^{n_f} X_{i,j,t+1}\frac{D_g^i F_{g,i,t}}{J_{g,j}} - \frac{1}{D_g^l}\sum_{t=1}^{n_f} X_{l,j,t+1}\frac{D_g^l F_{g,l,t}}{J_{g,j}}$, for $g = 1, 2$. If one considers the whole geometry, the eigenvalue problem will be that of Eq.(27), whose eigenvector is defined in Eq.(28).

$$\begin{pmatrix} V_i\left(\Sigma_{a,1}^i + \Sigma_{s,1\to 2}^i\right) & \sum_{j=1}^{n_f} S_j u_{i,j} & 0 & 0 \\ X_{i,j,1} - X_{l,j,1} & h_1 & 0 & 0 \\ -V_i\Sigma_{s,1\to 2}^i & 0 & V_i\Sigma_{a,2}^i & \sum_{j=1}^{n_f} S_j u_{i,j} \\ 0 & 0 & X_{i,j,1} - X_{l,j,1} & h_2 \end{pmatrix} \begin{pmatrix} \phi_{1,i} \\ J_{1,j} \\ \phi_{2,i} \\ J_{2,j} \end{pmatrix} = = \frac{1}{\mathbf{k}} \begin{pmatrix} V_i\nu\Sigma_{f,1}^i & 0 & V_i\nu\Sigma_{f,2}^i & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \phi_{1,i} \\ J_{1,j} \\ \phi_{2,i} \\ J_{2,j} \end{pmatrix} \quad (26)$$

$$\begin{pmatrix} L_{1,1} & 0 \\ L_{2,1} & L_{2,2} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix} = \frac{1}{\mathbf{k}} \begin{pmatrix} M_{1,1} & M_{1,2} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix} \quad (27)$$

$$\Phi_g = \begin{pmatrix} \phi_{g,1} \\ \vdots \\ \phi_{g,N_c} \\ J_{g,1} \\ \vdots \\ J_{g,N_f} \end{pmatrix} \quad (28)$$

From Eq.(27), one can obtain Eqs.(29) and (30). If Eqs.(29)-(30) are combined, one obtains Eq.(31), which defines a matrix depending only on Φ_1. Eq.(30) is the eigenvalue problem that will be solved. The dimension of this eigenvalue problem is half of that of Eq.(27), Φ_1 is the eigenvector, $\mathbf{k}$ is the eigenvalue and the problem is solved by using an iterative process. It is important to point out that the inverse of $L_{g,g}$ is not calculated in Eq.(31), but a

linear system is solved: $y = L_{g,g}^{-1} z \rightarrow L_{g,g} y = z$.

$$L_{2,2}\Phi_2 = -L_{2,1}\Phi_1 \tag{29}$$

$$\mathbf{k}\Phi_1 = L_{1,1}^{-1}\left(M_{1,1}\Phi_1 + M_{1,2}\Phi_2\right) \tag{30}$$

$$\mathbf{k}\Phi_1 = \mathcal{A}\Phi_1 = L_{1,1}^{-1}\left(M_{1,1} - M_{1,2}L_{2,2}^{-1}L_{2,1}\right)\Phi_1 \tag{31}$$

As the solution of the linear systems, the authors used the iterative solvers of PETSc [11]. These solvers are based on a combination of a Krylov subspace method and a preconditioner. PETSc includes a great variety of solvers and preconditioners. Examples of solvers are the following methods: Conjugate Gradient , BiConjugate Gradient, Generalized Minimal Residual(GMRES), Generalized Conjugate Residual, BiCGSTAB and Conjugate Gradient Squared. Examples of preconditioner are the following methods: Jacobi, SOR, Incomplete LU and Additive Schwarz. The authors performed a sensitivity analysis in Section 3.4 and found out that the fastest solver was GMRES [12] with the Additive Schwarz preconditioner.

For the calculation of the eigenvalue problem, the authors have applied the Krylov-Schur algorithm implemented in SLEPc [9, 10]. The Krylov-Schur method is an Arnoldi method which uses an implicit restart based on a Krylov-Schur decomposition [13].

The method of Arnoldi is a Krylov-based projection method that computes an orthonormal basis of the Krylov subspace of order m associated with matrix $\mathcal{A}$ and initial vector x_0. This Krylov subspace is given in Eq. (32). Projection methods for eigenvalue problems are intended for computing a partial eigensolution, that is, given a square matrix $\mathcal{A}$ of order N, the objective is to compute a small number of eigenpairs, λ_i , x_i , $i = 1, \cdots, m$, with $m \ll N$. The Arnoldi method computes not only this orthonormal basis (V_m), but also the projected

matrix H at the same time in an efficient and numerically stable way.

$$\mathcal{K}_m(\mathcal{A}, x_0) = span\left\{x_0, \mathcal{A}x_0, \mathcal{A}^2x_0, \cdots, \mathcal{A}^{m-1}x_0\right\} \tag{32}$$

This projection method calculates the eigenvalue problem $Hy_i = \theta_i y_i$, of order m, instead of $\mathcal{A}x_i = \lambda_i x_i$, of order N. Taken into account that ($H = V_m^T \mathcal{A} V_m$) and ($V_m^T V_m = I_m$), one concludes that the pair (λ_i,$V_m y_i$) can be taken as an approximation of the eigenpair (λ_i,x_i) of matrix $\mathcal{A}$. This method will converge very fast, if the initial vector x_0 is rich in the direction of the wanted eigenvectors, which is usually not the case. So, many iterations may be required, which implies a growth in storage requirements and computational time. A solution for this problem is to stop after some iterations and restart the method, by using a new initial vector computed from the recently obtained spectral approximations.

Different approaches can be used for the restart: explicit and implicit. Explicit algorithms calculate the initial vector as a linear combination of the current eigenvector approximations, but it is difficult to choose the appropriate parameters. Implicit algorithms combine the Arnoldi process with the implicitly shifted QR algorithm, in which an m-step Arnoldi factorization is compacted into an $(m - d)$-step Arnoldi factorization, which retains the relevant eigeninformation of the large factorization. The implementation of the implicit restart in a numerically stable way is difficult, but it is solved by using a Krylov-Schur decomposition. More information about this decomposition can be found in [13].

3. Results

The outline of this section is as follows. Subsection 3.1 describes the reactor and models used to evaluate the method. Subsection 3.2 analyzes the polynomial expansion of the neutron flux for a specific mesh and linear system solver. Subsection 3.3 performs a sensitivity analysis of different meshes for a specific polynomial expansion of the neutron flux and linear system solver. Subsection

3.4 compares the computational time of different linear system solvers of PETSc. Subsection 3.5 shows the capability of the method in parallel computation.

As with the numerical results, the first five largest eigenvalues and their eigenvectors were calculated in each simulation. All the CPU time values reported in this work have been obtained on an AMD Opteron(TM) Processor 6272 with the CentOS 6.8 operating system.

The Power Errors (PE) and Eigenvalue Errors (EE) are used to evaluate the results and are defined in Eqs.(33) and (34). The power for each cell (P_i) is defined in Eq.(35). The *constant* is a normalization factor to obtain Mean Power (MP) equals 1.0, which is defined in Eq.(36). In this work, all the eigenvalue errors are exhibited, but only the power errors corresponding to the first eigenvector are shown, due to the extent of the results.

$$PE_i(\%) = \frac{|P_i - P_{i,ref}|}{P_{i,ref}} \cdot 100 \tag{33}$$

$$EE(pcm) = \frac{\mathbf{k} - \mathbf{k}_{ref}}{\mathbf{k}_{ref}} \cdot 10^5 \tag{34}$$

$$P_i = constant \cdot \left(\Sigma^i_{f,1}\phi_{g,1} + \Sigma^i_{f,2}\phi_{g,2}\right) \tag{35}$$

$$MP = \frac{\sum_i |P_i| V_i}{\sum_i V_i} \tag{36}$$

3.1. Reactor VV1K3D

VV1K3D is a Water-Water Energetic Reactor (VVER) mockup. It is composed of 1690 hexagonal prisms, distributed in 10 axial levels of 20 cm in length. All the hexagonal prisms are regular and their flat-to-flat distance is 23.6 cm. A cross section of the reactor is displayed in Figure 2, in which each number represents an assembly type. Assemblies from 1 to 5 are composed of materials from 1 to 5, respectively. Composition of assembly 6 varies with the axial level: in the first five axial levels it is composed of material 4 and in the last ones it is composed of material 3. The cross sections of the 5 materials and two energy

groups are shown in Table 1 [14]. Boundary conditions are zero flux for all boundaries.

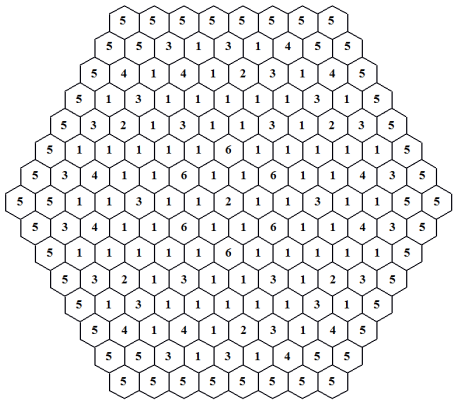

Figure 2: Assembly distribution in VV1K3D reactor.

Material	Group	$D_g\ (cm)$	$\Sigma_{a,g}\ (cm^{-1})$	$\nu\Sigma_{f,g}\ (cm^{-1})$	$\Sigma_{s,g\to g+1}\ (cm^{-1})$
1	1	1.38320	$8.3859{\cdot}10^{-3}$	$4.81619{\cdot}10^{-3}$	$1.64977{\cdot}10^{-2}$
	2	$3.86277{\cdot}10^{-1}$	$6.73049{\cdot}10^{-2}$	$8.46154{\cdot}10^{-2}$	
2	1	1.38299	$1.15550{\cdot}10^{-2}$	$4.66953{\cdot}10^{-3}$	$1.47315{\cdot}10^{-2}$
	2	$3.89403{\cdot}10^{-1}$	$8.10328{\cdot}10^{-2}$	$8.52264{\cdot}10^{-2}$	
3	1	1.39522	$8.9443{\cdot}10^{-3}$	$6.04889{\cdot}10^{-3}$	$1.56219{\cdot}10^{-2}$
	2	$3.86225{\cdot}10^{-1}$	$8.44801{\cdot}10^{-2}$	$1.19428{\cdot}10^{-1}$	
4	1	1.39446	$1.19932{\cdot}10^{-2}$	$5.91507{\cdot}10^{-3}$	$1.40185{\cdot}10^{-2}$
	2	$3.87723{\cdot}10^{-1}$	$9.89671{\cdot}10^{-2}$	$1.20497{\cdot}10^{-1}$	
5	1	1.39506	$9.1160{\cdot}10^{-3}$	$6.40256{\cdot}10^{-3}$	$1.54981{\cdot}10^{-2}$
	2	$3.84492{\cdot}10^{-1}$	$8.93878{\cdot}10^{-2}$	$1.29281{\cdot}10^{-1}$	

Table 1: Cross section data for VV1K3D

The reactor was modeled and meshed by means of Gmsh [2]. Three meshes were used. Mesh 1 is shown in Figure 3, which divides each hexagonal prism in 3 hexahedra as displayed in Figure 4 (left). Mesh 2 divides each hexahedron of Mesh 1 in 2x2x2 hexahedra and Mesh 3 divides each hexahedron of Mesh 1 in

3x3x3 hexahedra as exhibited in Figure 4.

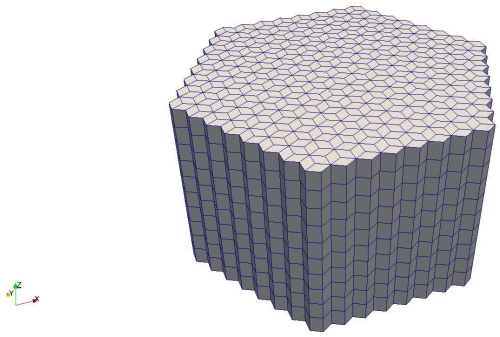

Figure 3: Mesh 1 of VV1K3D reactor.

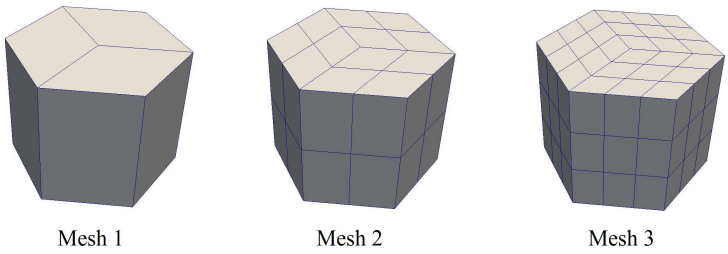

Figure 4: Subdivisions of hexagonal prism in Mesh 1, 2 and 3.

3.2. Analysis of the polynomial set

In this subsection, different polynomial sets are tested in Mesh 3. As the cells of this mesh are hexahedra, the number of faces of each cell (n_f) is 6. Thus, the polynomial expansion is limited to 7 ($7 = n_f + 1$) as discussed in Subsection 2.1. There are infinitely many possible polynomial sets, so the authors restricted the sets to monomials $x^{\alpha_t} y^{\beta_t} z^{\gamma_t}$ of order 2, that is, $\alpha_t + \beta_t + \gamma_t \leq 2$. There are ten 3D monomials of order 2: 1, x, y, z, x^2, y^2, z^2, xy, xz and yz. Thus, there are 120 possible 7-combinations of the set composed of these ten monomials. The authors tested these 120 combinations, and only 2 of them gave valid results. The first one is :1, x, y, z, x^2, z^2 and xy. The second one is :1, x, y, z, y^2, z^2 and xy.

The linear system solver used is GMRES with Additive Schwarz preconditioner. This preconditioner uses Incomplete LU preconditioner as local preconditioner by default. The computational time for each polynomial combination was: 4 minutes and 43 seconds for the first one; 4 minutes and 54 seconds for the second one. The results are evaluated considering that Combination 1 is the reference. In this work, the authors did not use other methods as reference solutions because of two reasons. First, the authors tested the method in other geometries and meshes in previous works, showing accurate results [8]. Second, the goal of this work is the sensitivity analysis of the method with the main variables affecting the numerical method: polynomial expansion, mesh and linear system solvers. The eigenvalue results are displayed in Table 2, which are accurate because $EE < 100$ pcm. Table 3 exhibits the axial power errors for the different axial levels, which are good since they are lower than 1 %. It can be concluded that for fine meshes, the results are almost insensitive to the polynomial sets.

Eigenvalue	Combination 1	Combination 2	$EE(pcm)$
1	1.005460	1.005503	4.23
2	0.987339	0.987434	9.62
3	0.987319	0.987403	8.57
4	0.968399	0.968575	18.14
5	0.964224	0.964336	11.64

Table 2: Eigenvalue results for Mesh 3

Axial level	10	9	8	7	6	5	4	3	2	1
$PE(\%)$	0.09	0.08	0.08	0.05	0.02	0.04	0.08	0.11	0.16	0.12

Table 3: Axial power results for Mesh 3

3.3. Analysis of the mesh

In this subsection, a sensitivity analysis of the mesh is performed, but the polynomial set is fixed to Combination 1: 1, x, y, z, x^2, z^2 and xy. Meshes 1,

2 and 3 defined in Subsection 3.1 were used. The linear system solver used is GMRES with Additive Schwarz preconditioner.

The number of rows of matrices $L_{g,g}$ for each mesh is: 19323 for Mesh 1; 158412 for Mesh 2; 538947 for Mesh 3. The computational time for each mesh was: 5 seconds for Mesh 1; 58 seconds for Mesh 2; 4 minutes and 43 seconds for Mesh 3. The results are evaluated considering that Mesh 3 is the reference. Table 4 shows the eigenvalue results. The axial power results are exhibited in Table 5. One can conclude that Mesh 2 is more accurate than Mesh 1; but results of Mesh 1 are good enough, because the maximum eigenvalue error is about 100 pcm and the maximum axial power error is about 1 %.

	Eigenvalue			$EE(pcm)$	
Eigenvalue	Mesh 1	Mesh 2	Mesh 3	Mesh 1	Mesh 2
1	1.005193	1.005389	1.005460	26.61	7.12
2	0.987292	0.987313	0.987339	4.78	2.64
3	0.986931	0.987234	0.987319	39.22	8.59
4	0.968356	0.968382	0.968399	4.46	1.78
5	0.962930	0.964002	0.964224	134.26	22.99

Table 4: Eigenvalue results for Combination 1

	Axial level	10	9	8	7	6	5	4	3	2	1
$PE(\%)$	Mesh 1	0.97	0.88	0.72	0.44	0.02	0.44	0.72	0.93	1.08	1.10
	Mesh 2	0.21	0.19	0.15	0.10	0.01	0.09	0.16	0.20	0.24	0.24

Table 5: Axial power results for Combination 1

3.4. Analysis of the linear system solver

In this subsection, the mesh and the polynomial set are fixed: Mesh 3 and Combination 1 (1, x, y, z, x^2, z^2 and xy). The authors tested the following linear system solvers of PETSc: BiConjugate Gradient (bicg), GMRES, Generalized Conjugate Residual (gcr), BiCGSTAB (bcgs) and Conjugate Gradient Squared

(cgs). The authors used these solvers because they can be applied to non-symmetric matrices. These solvers were used with the following preconditioners of PETSc: Jacobi, SOR and Additive Schwarz (asm), which is the same as Incomplete LU for one processor. The authors used the default tolerances of PETSc.

Among all these combinations of solvers and preconditioners, only one of them is forbidden in PETSc (for non-symmetric matrices): BiConjugate Gradient with SOR. For the rest, the results are the same, but there are differences in the computational time as shown in Figure 5. From this figure, one concludes that GMRES is the fastest method in combination with the Additive Schwarz preconditioner.

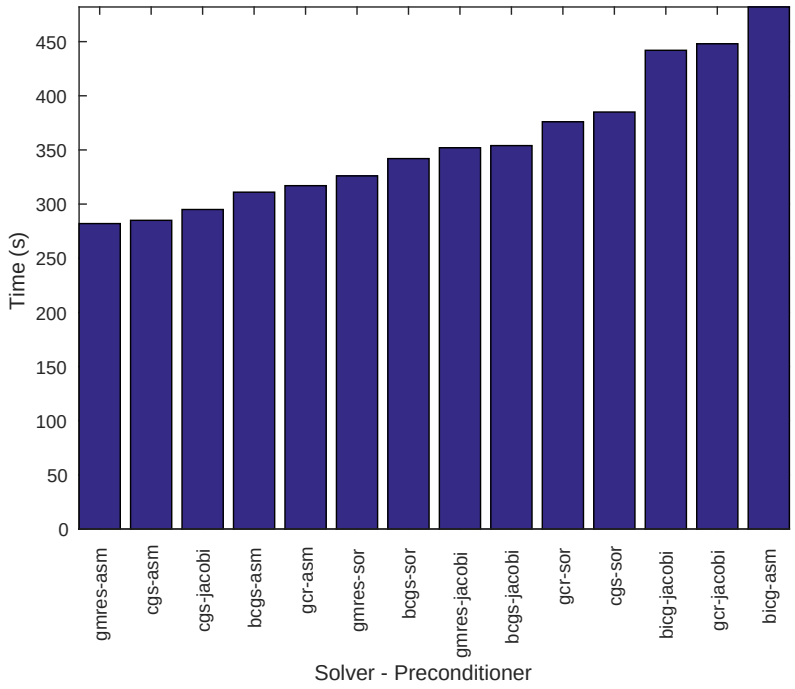

Figure 5: Time results for the linear system solvers.

An important issue concerning linear solvers is the condition number of the matrices, because the bigger the condition number, the slower the convergence

of the iterative linear solvers. The condition number of a matrix can be determined as the ratio of the maximum singular value to the minimum one. The authors calculated the condition number of matrices $L_{1,1}$ and $L_{2,2}$ by using the singular value decomposition solver of SLEPc. Table 6 shows the calculated condition number for each mesh and polynomial combinations. In this table, C.1 corresponds to Combination 1 and C.2 corresponds to Combination 2. One draws two conclusions from this table. First, the matrices are well-conditioned, since their condition number is very low. Second, the condition number of this discretization, applied to this reactor, is almost insensitive to the mesh and the polynomial terms.

$L_{1,1}$						$L_{2,2}$					
Mesh 1		Mesh 2		Mesh 3		Mesh 1		Mesh 2		Mesh 3	
C.1	C.2	C.1	C.2	C.1	C.2	C.1	C.2	C.1	C.2	C.1	C.2
56.1	56.1	56.3	56.3	62.9	56.3	74.4	74.6	74.6	74.6	74.6	74.6

Table 6: Condition number of $L_{1,1}$ and $L_{2,2}$

3.5. Parallelization

In this subsection, the authors assessed the capability of the parallel computation of the method. The parallelization includes: geometry pre-processing, equations discretization, eigenvalue and linear system solvers, linear algebra operations and post-processing. The parallel implementation uses the Message Passing Interface (MPI) standard for all message-passing communication. For testing the parallel computation, the authors run the simulation with Mesh 3, Combination 1, GMRES solver and Additive Schwarz preconditioner. The size of the matrices $L_{1,1}$ and $L_{2,2}$ for this case is 538947. To evaluate the parallelization, the speedup for N processors is defined as the ratio of the computational time with one processor to the computational time with N processors. Figure 6 shows the speedup of the parallelization. This figure shows the total simulation time. Two conclusions can be drawn from this figure. Firstly, the performance is close to ideal till 5 processors. Secondly, a reasonably good performance gain

is seen up to 14 processors in the strong scaling sense. It should be highlighted that the efficacy of the Additive Schwarz preconditioner decreases with the number of processors, so this behavior is normal. Finally, the parallel computation runs the case of Mesh 3 in 35 seconds, with 16 processors.

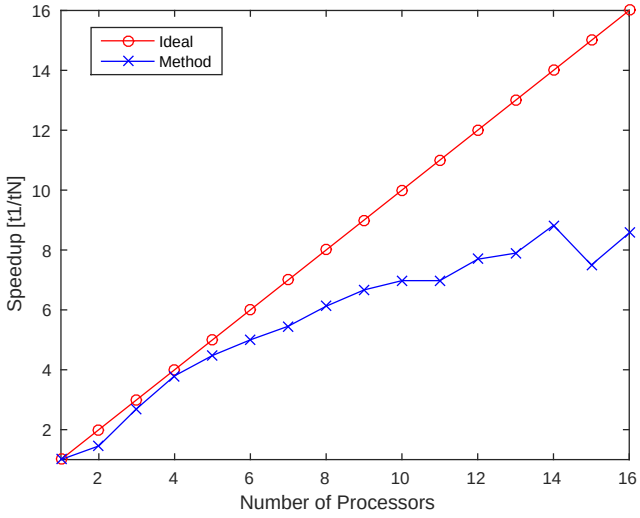

Figure 6: Speedup of the parallelization.

4. Conclusions

The spatial distribution of the neutron flux can be estimated by solving the eigenvalue problem of the steady state, 2 energy group, neutron diffusion equation.

For solving this eigenvalue problem in 3D geometries, one should discretize the geometry and apply numerical methods. In this work, the authors used the mesh generator called Gmsh and the finite volume method.

The discretized geometries in the neutron diffusion equation requires additional interfaces equations: neutron flux and current continuity. These equations imply an excess of equations, but this is solved in this work by using a polynomial expansion of the neutron flux. In addition, the authors applied an implicit

definition of these equations to reduce the size of the system matrices of the eigenvalue problem.

The final eigenvalue problem was reduced to half the dimension of the original one. Moreover, the problem was solved by using an iterative process based on the Krylov-Schur algorithm of the SLEPc library. Among the different linear algebra operations in this process, SLEPc solves linear systems by using the methods of the PETSc library. These libraries are the state of the art and can run the algorithm in parallel.

The method developed in this work was evaluated in a VVER reactor. Several sensitivity analyses were performed: polynomial expansion, mesh and linear solvers. The authors tested several polynomial sets up to order 2 and found out that only two combinations gave valid results for the mesh used: the first one is 1, x, y, z, x^2, z^2 and xy; the second one is 1, x, y, z, y^2, z^2 and xy. As regards the mesh, excellent results are obtained even for coarse meshes. With respect to the linear solvers, the fastest one is GMRES using the Addditive Schwarz preconditioner. The authors also studied the condition number of the system matrices and they drew two conclusions. First, the matrices are well-conditioned. Second, the condition number of this discretization, applied to this reactor, is almost insensitive to the mesh and the polynomial terms.

Finally, the authors evaluated the capability of parallelization of the method. Excellent speedup is obtained till 5 processors. A reasonably good performance gain is seen up to 14 processors in the strong scaling sense.

As regards the future work, the authors will perform the parallelization of the multigroup neutron diffusion equation. This multigroup formulation will include any number of energy groups, upscattering terms and fission production in any energy group.

Acknowledgments

This work has been partially supported by the Spanish Ministerio de Eduación Cultura y Deporte under the grant FPU13/01009, the Spanish Minis-

terio de Ciencia e Innovación under the project ENE2014-59442-P, the Spanish Ministerio de Economía y Competitividad and the European Fondo Europeo de Desarrollo Regional (FEDER) under the project ENE2015-68353-P (MINECO/FEDER), the Generalitat Valenciana under the project PROMETEOII/2014/008, and the Spanish Ministerio de Economía y Competitividad and the European Fondo Europeo de Desarrollo Regional (FEDER) under the project TIN2016-75985-P.

References

References

[1] W. Stacey, Nuclear Reactor Physics, John Wiley & Sons, 2001.

[2] C. Geuzaine, J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (2009) 1309–1331. doi:10.1002/nme.2579.

[3] A. Hébert, Applied Reactor Physics, Second Edition, Presses internationales Polytechnique, 2016.

[4] K. Smith, An analytic nodal method for solving the two-group, multidimensional static and transient neutron diffusion equations, Ph.D. thesis, Department of Nuclear Engineering, MIT (1979).

[5] K. Hoffmann, S. T. Chiang, Computational fluid dynamics, Vol. II, Engineering Education System, 2000.

[6] A. Bernal, R. Miró, D. Ginestar, G. Verdú, Resolution of the generalized eigenvalue problem in the neutron diffusion equation discretized by the finite volume method, Abstract and Applied Analysis 2014 (2014) 1–15. doi:10.1155/2014/913043.

[7] A. Bernal, J. Roman, R. Miró, D. Ginestar, G. Verdú, Development of a finite volume inter-cell polynomial expansion method for the neutron diffusion equation, Journal of Nuclear Science and Technology 53 (2016) 1212–1223. doi:10.1080/00223131.2015.1102661.

[8] A. Bernal, J. Roman, R. Miró, G. Verdú, Assembly discontinuity factors for the neutron diffusion equation discretized with the finite volume method. application to bwr, Annals of Nuclear Energy 97 (2016) 76–85. doi:10.1016/j.anucene.2016.06.023.

[9] V. Hernandez, J. E. Roman, V. Vidal, Slepc: A scalable and flexible toolkit for the solution of eigenvalue problems, ACM Transactions on Mathematical Software 31 (2005) 351–362. doi:10.1145/1089014.1089019.

[10] V. Hernandez, J. E. Roman, V. Vidal, Slepc: Scalable library for eigenvalue problem computations, Lecture Notes in Computer Science 2565 (2003) 377–391.

[11] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory (2016).
URL http://www.mcs.anl.gov/petsc

[12] Y. Saad, M. H. Schultz, Gmres - a generalized minimal residual algorithm for solving nonsymmetric linear-systems, SIAM Journal on Scientific and Statistical Computing 7 (1986) 856–869. doi:10.1137/0907058.

[13] G. Stewart, A krylov-schur algorithm for large eigenproblems, SIAM Journal on Matrix Analysis and Applications 23 (2002) 601–614. doi:10.1137/S0895479800371529.

[14] Y. Chao, Y. Shatilla, Conformal mapping and hexagonal nodal methods-ii:

Implementation in the anc-h code, Nuclear Science and Engineering 121 (1995) 210–225.