Schwarz type preconditioners for the neutron diffusion equation

A. Vidal-Ferràndiz[a], S. González-Pintor[b], D. Ginestar[d,*], G. Verdú[a], C. Demazière[c]

[a] *Instituto de Seguridad Industrial: Radiofísica y Medioambiental, Universitat Politècnica de València, Camino de Vera s/n, 46022, València, Spain*
[b] *Department of Mathematical Sciences Chalmers University of Technology, Maskingränd 2, 412 58 Göteborg, Sweden*
[c] *Division of Nuclear Engineering, Department of Applied Physics Chalmers University of Technology, Maskingränd 2, 412 58 Göteborg, Sweden*
[d] *Instituto Universitario de Matemática Multidisciplinar, Universitat Politècnica de València, Camino de Vera s/n, 46022, València, Spain*

Abstract

Domain decomposition is a mature methodology that has been used to accelerate the convergence of partial differential equations. Even if it was devised as a solver by itself, it is usually employed together with Krylov iterative methods improving its rate of convergence, and providing scalability with respect to the size of the problem.

In this work, a high order finite element discretization of the neutron diffusion equation is considered. In this problem the preconditioning of large and sparse linear systems arising from a source driven formulation becomes necessary due to the complexity of the problem. On the other hand, preconditioners based on an incomplete factorization are very expensive from the point of view of memory requirements. The acceleration of the neutron diffusion equation is thus studied here by using alternative preconditioners based on domain decomposition techniques inside Schur complement methodology. The study considers substructuring preconditioners, which do not involve overlapping, and additive Schwarz preconditioners, where some overlapping between the subdomains is taken into account.

The performance of the different approaches is studied numerically using two-dimensional and three-dimensional problems. It is shown that some of the proposed methodologies outperform incomplete LU factorization for preconditioning as long as the linear system to be solved is large enough, as it occurs for three-dimensional problems. They also outperform classical diagonal Jacobi preconditioners, as long as the number of systems to be solved is large enough in such a way that the overhead of building the preconditioner is less than the improvement in the convergence rate.

Keywords: Neutron Diffusion, Finite Element Method, Substructuring, Schwarz Preconditioner

*Corresponding author
Email addresses: anvifer2@upv.es (A. Vidal-Ferràndiz), sebastian.gonzalez-pintor@chalmers.se (S.

1. Introduction

Domain decomposition methods were first proposed by Schwarz [1] as an analytical tool. A renewed interest in this kind of methods was noticed with the appearance of parallel computers [2]. These methods were first proposed to solve partial differential equations on complex domains and later these techniques were extended to solve linear equations (see [3] and references therein). The algebraic Schwarz methods are based on partitioning the vector of unknowns into subsets, which correspond to a partition of the coefficients matrix, typically associated with different subdomains in the continuous problem. The solution of the whole system is achieved by solving the systems associated with the different blocks of the partitioned matrix, which are simpler problems than the original one. Schwarz methods to solve linear systems are not as competitive as other alternative methods such as multi-grid solvers, but they can be used as efficient preconditioners of Krylov methods, at the cost of a few more iterations.

Domain decomposition methods in nuclear engineering have been receiving increasing attention for the last years due to their potential to solve large problems by using a *divide-and-conquer* strategy. For example, a Schur complement is used to accelerate the core solver in [4]. In [5, 6] a method based on the Schwarz iterative algorithm is studied to solve the mixed neutron diffusion and the simplified spherical harmonics neutron equation. Finally, the response matrix method that implements a two-level model, a global and a local level, was analysed in [7, 8]. The local level is defined on a mesh fine enough to provide accurate results while the global level is defined on a coarse mesh which accelerates the convergence of the method. The methodology for linking the local and global solutions is the key aspect of the response matrix method.

The neutron diffusion equation is an approximation of the neutron transport equation [9]. This equation describes a balance between generation and loss of neutrons by a generalized differential eigenvalue problem. The dominant eigenvalue and its corresponding eigenfunction describe the steady state neutron distribution, thus, these quantities should be determined for most of the reactor analyses. The problem is discretized by a high order Galerkin Finite Element Method (FEM), thus transforming it into an algebraic eigenvalue problem. Different methods can be used to solve this eigenvalue problem, while the common bottle-neck of all of them is the solution of a large number of linear systems for a few different coefficient matrices.

Because of the discretization with a FEM, the matrices of the systems are large and sparse their related linear systems are symmetric and positive definite. Thus, these systems are well suited to be solved with an iterative Krylov subspace method, where a classical approach for preconditioning these linear systems is by using an incomplete factorization of the coefficient matrices [10]. Nevertheless, the computation of such preconditioner requires to store the coefficients matrices in the computer memory, in addition to the preconditioner itself, which results in large requirements of memory resources. These memory requirements can be lowered by different fill-in or threshold criteria for the preconditioner, although the the minimum memory requirement remains large if a fast preconditioner is used. In this work, we study alternative preconditioning techniques for these systems based on the domain decomposition methodology, aiming at lowering the memory requirements when high order Finite Element Methods are used for the spatial discretization.

The rest of the paper is organized as follows. In Section 2, the high order finite element method (FEM) that discretizes the problem using Lagrange polynomials is briefly reviewed.

González-Pintor), dginesta@mat.upv.es (D. Ginestar), gverdu@iqn.upv.es (G. Verdú), demaz@chalmers.se (C. Demazière)

These polynomials provide a partition of the shape functions set into vertices, edges, faces and interior functions. Using this natural partition, the linear systems of equations associated with each energy group can be solved with a Schur Complement method that algebraically decouples the interior degrees of freedom from the other ones. This method, also called static condensation method, is presented in Section 3. This method is advantageous when a high polynomial degree, p, is used in the FEM discretization. To precondition the resulting Schur complement system two different strategies are described in this work. First a substructuring block Jacobi preconditioner is studied in Section 4, where the coupling between the different elements is neglected. Also, a domain decomposition algorithm with overlapping between subdomains, like the additive Schwarz method, is considered in Section 5. Several benchmarks are studied in Section 6 to test numerically the performance of the different approaches proposed. Finally, the main conclusions of the paper are summarized in Section 7.

2. Neutron diffusion equation and its high order FEM discretization

The neutron diffusion equation is an approximation of the neutron transport equation relying on the assumption that the neutron current is proportional to the gradient of the neutron flux by means of a diffusion coefficient. This approximation is analogous to Fick's law in species diffusion and to Fourier's law in heat transfer [9]. For a given configuration of a nuclear reactor core, it is always possible to force its criticality dividing the neutron production rate by a positive number, λ, obtaining a neutron balance equation. This equation is known as the *Lambda modes problem*, and is of the form

$$\mathcal{L}\,\Phi = \frac{1}{\lambda}\mathcal{M}\,\Phi\,, \tag{1}$$

where $\mathcal{L}$ is the neutron loss differential operator and $\mathcal{M}$ is the neutron production operator. This problem is a generalized eigenvalue problem, and its fundamental eigenvalue (the largest one) is called the multiplication factor of the reactor core, k_{eff}. This eigenvalue and its corresponding eigenfunction describe the steady state neutron distribution in the core, thus, these quantities should be determined for most of the reactor analyses.

This equation has a well-defined block structure due to the neutron energy discretization of the problem, where the equation for the energy group $g \in \{1, \ldots, G\}$, has the following form

$$\mathcal{L}_{gg}\Phi_g = \sum_{h\neq g}^{G} \mathcal{L}_{gh}\Phi_h + \frac{1}{\lambda}\sum_{h=1}^{G} \mathcal{M}_{gh}\Phi_h, \tag{2}$$

where

$$\mathcal{L}_{gg}\Phi_g := -\vec{\nabla} D_g \vec{\nabla}\Phi_g + \Sigma_{r,g}\Phi_g, \tag{3}$$

$$\mathcal{L}_{gh}\Phi_h := \Sigma_{s,h\to g}\Phi_h, \tag{4}$$

$$\mathcal{M}_{gh}\Phi_h := \chi_g \nu\Sigma_{f,h}\Phi_h, \tag{5}$$

Here Φ_g is the neutron flux for the g-th energy group, D_g is the diffusion coefficient, $\Sigma_{r,g}$ is the macroscopic removal cross section (absorption plus out-scattering), $\Sigma_{s,h\to g}$ is the macroscopic scattering cross section from group g to h, $\nu\Sigma_{f,g}$ is the fission cross section, χ_g is the neutron energy spectrum of fission and λ is the he multiplication factor defined before. The cross sections and the diffusion coefficient are space dependent functions, usually defined as piecewise constants because of a previous homogenization procedure [11].

2.1. The Eigenvalue Problem

In the most general case, this eigenvalue problem is traditionally solved using a source iteration method with a combination of inner and outer iterations [12]. The inner iterations update the neutron flux, from fast to thermal energy groups, using a fixed source from a previous outer iteration that considers the fissions and the out-scattering from other groups, as follows

$$\mathcal{L}_{gg}\Phi_g^{(i)} = Q_g^{(i-1)}, \tag{6}$$

where the source for group g, Q_g, is generated using the updated neutron flux for the up-scattering terms, and the previous neutron flux for the down-scattering and for the fission terms

$$Q_g^{(i-1)} = \sum_{h=1}^{g-1} \mathcal{L}_{gh}\Phi_h^{(i)} + \sum_{h=g+1}^{G} \mathcal{L}_{gh}\Phi_h^{(i-1)} + \frac{1}{\lambda^{(i-1)}} \sum_{h=1}^{G} \mathcal{M}_{gh}\Phi_h^{(i-1)}. \tag{7}$$

Once the inner-iteration has finished, by performing either one single loop through the energy groups or iterating until convergence, the outer iteration updates the eigenvalue by using the actual flux and the previous flux with the previous eigenvalue, as follow

$$\lambda^{(i)} = \lambda^{(i-1)} \frac{\|\mathcal{M}\Phi^{(i)}\|}{\|\mathcal{M}\Phi^{(i-1)}\|}, \tag{8}$$

where $\|\cdot\|$ is a norm in $(L^2(\Omega))^G := \overbrace{L^2(\Omega) \times \ldots \times L^2(\Omega)}^{G}$. For this general scenario, the solution of equation (6) is the most demanding step from the computational point of view. Thus, the preconditioning of a finite element discretization of equation (6) is essential in order to accelerate the whole algorithm.

Nevertheless, another situation might be accounted where the Lambda modes problem is approximated using only two energy groups, assuming that the neutrons are born in the fast group and there is no up-scattering, thus obtaining a block lower triangular form of the operator that can be used to solve the system more efficiently. This scenario is widely considered because it provides a good approximation, at low cost, for simulating standard LWR type reactor cores. In this particular situation, equation (1) can be expressed in the more compact form (see [9]) by equation (9)

$$\begin{pmatrix} \mathcal{L}_{11} & 0 \\ -\mathcal{L}_{21} & \mathcal{L}_{22} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix} = \frac{1}{\lambda} \begin{pmatrix} \mathcal{M}_{11} & \mathcal{M}_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix}, \tag{9}$$

which can be formally expressed, due to the block lower triangular form of the system, as an ordinary eigenvalue problem of the form

$$\mathcal{L}_{11}^{-1} \left(\mathcal{M}_{11} + \mathcal{M}_{12}\mathcal{L}_{22}^{-1}\mathcal{L}_{21} \right) \Phi_1 = \lambda \Phi_1 \, . \tag{10}$$

This eigenvalue problem can be solved, after proper spatial discretization of the differential operators, by the source iteration method as before, or in a more efficient manner by, for example, a Krylov-Schur method using the library SLEPc [13]. This method for solving the eigenvalue problem, as well as the source iteration method, requires the solution of linear systems associated to a discretized form of the following equation

$$\mathcal{L}_{gg}\, \Phi_g = Q_g. \tag{11}$$

It is worth pointing out that, when using the Krylov-Schur method, the term on the right hand side, Q_g, is not the same as when using the source iteration, because the former is generated within the Krylov subspace and does not have a physical meaning.

2.2. Finite Element discretization.

In what follows we focus on a generic equation similar to equation (11) with zero flux boundary conditions in order to simplify the derivations, even if the results obtained can be easily extended to more realistic boundary conditions, as albedo boundary conditions. Using standard notation for the finite elements discretization, this problem reads as follows

$$-\vec{\nabla}(D\vec{\nabla})u + \Sigma_r u = f \quad \text{on } \Omega, \tag{12a}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{12b}$$

where Ω is the spatial domain and $\partial\Omega$ is the boundary of the domain and u is the solution for the neutron flux. Problem (12) can be expressed in variational form as follows. Find $u \in H_0^1(\Omega)$ such that

$$a(u, v) = b(v) \quad \forall v \in H_0^1(\Omega), \tag{13}$$

where $H_0^1(\Omega)$ is the Sobolev space of admissible functions vanishing at the boundary, and the bilinear and linear forms $a(\cdot, \cdot)$ and $b(\cdot)$ are defined by

$$a(u, v) = (D\nabla u, \nabla v)_\Omega + (\Sigma_r u, v)_\Omega \,, \quad \text{and} \quad b(v) = (f, v)_\Omega \,. \tag{14}$$

The scalar product is defined as the volume integral over the whole domain Ω. Now a conforming triangulation is chosen, $\mathcal{T}_h$, splitting the original domain Ω into subdomains $T \in \mathcal{T}_h$. Then, after integrating by parts, and using the boundary conditions (12b), we obtain

$$(D\nabla u, \nabla v)_{\mathcal{T}_h} + (\Sigma_r u, v)_{\mathcal{T}_h} = (f, v)_{\mathcal{T}_h}, \quad v \in V_{h,p}, \tag{15}$$

where the inner product over the triangulation is defined as the sum of the inner products over each element $T \in \mathcal{T}_h$ as follows

$$(f, g)_{\mathcal{T}_h} = \sum_{T \in \mathcal{T}_h} (f, g)_T \tag{16}$$

and $V_{h,p}$ is the discrete space of polynomials up to degree p over each element T, being continuous across the interfaces.

The shape functions used here to span the space of polynomials up to degree p are Lagrange polynomials defined on a set of Gauss-Lobatto Legendre quadrature points [14]. The finite element method has been implemented using the open source finite elements library *deal.II* [16]. With the help of this library, the code proposed is dimension independent and can manage different cell sizes and different types of finite elements. More details on the spatial discretization used can be found in [15].

3. Schur Complement and Preconditioning

Domain decomposition methods were first proposed by Schwarz [1] as an analytical tool, and one noticed an increased interest in this kind of methods in the last decades with the appearance of parallel computers [2]. For the neutron diffusion equation, the continuous method is based

on solving the problem in different domains, while connecting the different solutions through appropriate interface conditions, i.e.,

$$-\vec{\nabla}(D_k\vec{\nabla})u_k + \Sigma_{rk}u_k = f_k, \quad \text{in} \quad \Omega_i \quad \forall k \tag{17a}$$

$$u_k = 0, \quad \text{on} \quad \Omega_i \cap \partial\Omega \quad \forall k \tag{17b}$$

$$u_k = u_j, \quad \text{on} \quad \Gamma_{kj} \quad \forall k, j \tag{17c}$$

$$D_k\nabla u_k = D_j\nabla u_j, \quad \text{on} \quad \Gamma_{kj} \quad \forall k, j \tag{17d}$$

where u_k is defined as the restriction of the function u to the subdomain Ω_k, i.e., $u_k := u|_{\Omega_k}$, and the interfaces are defined as the intersection of the subdomains without considering the Dirichlet boundary, i.e., $\Gamma_{kj} := \partial\Omega_k \cap \partial\Omega_j$.

As it has been already mentioned, the neutron diffusion equation is assumed to have piecewise constant coefficients on different subdomains, due to a previous homogenization procedure in order to reduce the problem to this form. Thus, here we consider the mesh to be defined as the equation having constant coefficients over each element. Therefore, the accuracy of the method is increased by raising the degree of the polynomial expansion inside such a cell

We also make use of the fact that the Lagrange polynomials used in the high order finite element method provide natural partition of the set of degrees of freedom (DoFs) into vertices, edges, faces and interior shape functions as can be seen in Figure 1. Figure 1 shows a representation of the substructuring decomposition for a two-dimensional domain using polynomials of degree 3. Thus, after the spatial discretization, the local problem above can be expressed in matrix form as

$$A_k x_k := \begin{pmatrix} A_{k,ii} & A_{k,iv} & A_{k,ie} & A_{k,if} \\ A_{k,iv}^T & A_{k,vv} & A_{k,ve} & A_{k,vf} \\ A_{k,ie}^T & A_{k,ve}^T & A_{k,ee} & A_{k,ef} \\ A_{k,if}^T & A_{k,vf}^T & A_{k,ef}^T & A_{k,ff} \end{pmatrix} \begin{pmatrix} x_{k,i} \\ x_{k,v} \\ x_{k,e} \\ x_{k,f} \end{pmatrix} = \begin{pmatrix} f_{k,i} \\ f_{k,v} \\ f_{k,e} \\ f_{k,f} \end{pmatrix} =: f_k \,, \tag{18}$$

where the subscript i refers to the degrees of freedom related to the interior of the cells, and the subscripts v, e and f refer to the degrees of freedom related with the vertices, edges and faces, respectively.

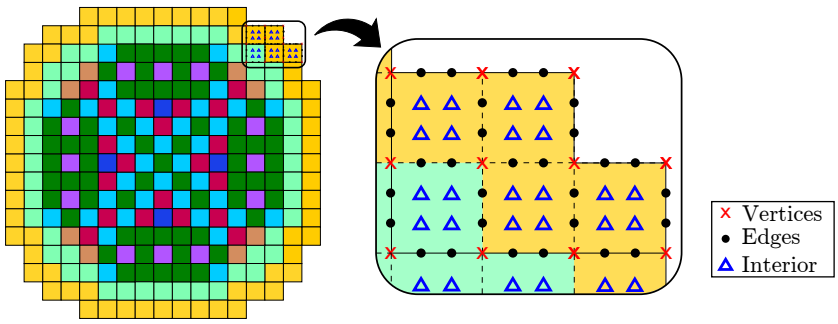


Figure 1: A model of a reactor and the representation of the different degrees of freedom using cubic polynomials in a two-dimensional problem.

3.1. Schur Complement Method

We notice here that the interior DoFs of the previous local system (18) are decoupled from the interior DoFs of the other local system. It allows us to apply the Schur complement method. After assembling the global matrix for the problem by putting all the local problems (18) together, we obtain a linear system as follows

$$Ax := \begin{pmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{pmatrix} \begin{pmatrix} x_I \\ x_B \end{pmatrix} = \begin{pmatrix} f_I \\ f_B \end{pmatrix} =: f \ , \tag{19}$$

where the block A_{II} is a block diagonal matrix, every block being the block matrix of the local interior nodes, i.e,

$$A_{II} = \text{diag}\{A_{1,ii}, \ldots, A_{k,ii}, \ldots, A_{K,ii}\}, \tag{20}$$

and the matrix A_{BB} is composed of the matrices for the interfaces, connecting the local matrices. In other words, the matrix contains the terms related with the vertices, edges and faces. Then we solve the system for the interior DoFs, which is block diagonal and thus easy to invert using a complete Cholesky factorization, and obtain a system for the boundary degrees of freedom, which is written as

$$S\, x_B = f_S \ , \tag{21}$$

where

$$S = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB} \ , \quad f_S = f_B - A_{IB} A_{II}^{-1} f_I \ .$$

Once the Schur complement system (21) is solved, the interior unknowns can be obtained by simple matrix multiplication,

$$x_I = A_{II}^{-1}(f_I - A_{BI}^T x_B) \ .$$

Inside a finite element partitioning the Schur complement matrix can be built locally as a sum of the contribution in each cell, S_k, as follows,

$$S = \sum_{k=1}^{K} S_k = \sum_{k=1}^{K} \left(A_{k,bb} - A_{k,ib}^T A_{k,ii}^{-1} A_{k,ib} \right), \tag{22}$$

where $A_{k,\_}$ is the corresponding local block matrix associated with the cell k. Then, the system matrix A does not need to be built explicitly to construct its Schur complement matrix.

It is worth to notice here that since the Schur complement is the stiffness matrix associated with a subspace of the space generated by the original basis, its condition number is bounded by the condition number of the complete matrix A and is typically far better [17].

A disadvantage of this approach is the additional expense of constructing the Schur complement matrix. For a single system this expense can outweigh the advantage of solving a better conditioned system for low order in the polynomial expansions [18]. However, for the resolution of the *Lambda modes problem* each matrix has to be solved several times with different right hand sides. Thus, the computational cost of constructing the Schur complement matrix is overcome by the number of linear systems to be solved.

4. Substructuring preconditioners

In the same way as the original matrix, the Schur complement matrix has a structure that can be algebraically separated into vertices, edges and faces degrees of freedom. The methods based on this kind of partition of the high order finite element shape functions are called substructuring methods. We use this partition of the DoFs to provide the Schur complement system (21) with the following structure

$$\begin{pmatrix} S_{vv} & S_{ve} & S_{vf} \\ S_{ve}^T & S_{ee} & S_{ef} \\ S_{vf}^T & S_{ef}^T & S_{ff} \end{pmatrix} \begin{pmatrix} x_{B,v} \\ x_{B,e} \\ x_{B,f} \end{pmatrix} = \begin{pmatrix} f_{S,v} \\ f_{S,e} \\ f_{S,f} \end{pmatrix}, \tag{23}$$

Thus, a substructuring preconditioner for this system can be defined as,

$$P = \begin{pmatrix} B_{vv} & 0 & 0 \\ 0 & B_{ee} & 0 \\ 0 & 0 & B_{ff} \end{pmatrix}^{-1}. \tag{24}$$

where different substructuring preconditioners for the Schur complement matrix are defined in the next sections by different choices for B_{vv}, B_{ee}, B_{ff}.

4.1. Preconditioner P_d

The simplest preconditioner for the Schur complement matrix is to use only use the diagonal of the local matrices for preconditioning, i.e.,

$$\begin{aligned} B_{vv} &= \text{diag}\left(S_{vv}\right), \\ B_{ee} &= \text{diag}\left(S_{ee}\right), \\ B_{ff} &= \text{diag}\left(S_{ff}\right), \end{aligned}$$

implementing a Diagonal Jacobi preconditioner.

4.2. Preconditioner P_v

As with the original matrix, A, the preconditioner can be improved if the vertices sub-matrix is solved in the preconditioner. This matrix is also explicitly assembled and Cholesky factorized. In this case, the preconditioner sub-matrices are defined as,

$$\begin{aligned} B_{vv} &= S_{vv}, \\ B_{ee} &= \text{diag}\left(S_{ee}\right), \\ B_{ff} &= \text{diag}\left(S_{ff}\right). \end{aligned}$$

4.3. Preconditioner P_{vef}

Similarly to the original matrix, the sub-matrices S_{ee} and S_{ff} represent the whole set of edges and faces degrees of freedom respectively. In the Schur complement preconditioner each edge and each face is considered independent. Thus, B_{ee} and B_{ff} also have a block diagonal structure. The preconditioner blocks are defined by,

$$\begin{aligned} B_{vv} &= S_{vv}, \\ B_{ee} &= \text{block-diag}\left(S_{ee}\right), \\ B_{ff} &= \text{block-diag}\left(S_{ff}\right). \end{aligned}$$

5. Restricted Additive Schwarz preconditioner

Another possibility of preconditioning consists of introducing overlapping between the subdomains by including the degrees of freedom related to the vertices and edges in the different blocks, referred as Additive Schwarz preconditioner. This preconditioner is an extension of the classical alternating Schwarz method at the continuous level [19], formulated by Schwarz in 1870. This method sequentially solves the Laplace equation in two continuous subdomains. The projection interpretation of the alternating Schwarz method led to the Additive Schwarz preconditioner defined at the matrix level. It should be noted that the Additive Schwarz will not converge in general, but nevertheless can be efficiently used to accelerate a Krylov subspace method [10].

In this work, a Restricted Additive Schwarz (RAS) [20] is used to accelerate the Schur complement system of equation (21). In [21] it is studied from a theoretical point of view why this preconditioner usually has better convergence properties than the unrestricted Additive Schwarz. The RAS is defined as,

$$P_{RAS} = \sum_{k=1}^{K} R_k^T X_k \left(R_k S R_k^T \right)^{-1} R_k, \tag{25}$$

where the subdomain k is the complete union of two finite element cells, R_k denotes the restriction operator from the global domain to the subdomain k, R_k^T in the corresponding interpolation operator and X_k is a partition of unity matrix that scales the contribution of each degrees of freedom depending on the number of subdomains where it is present. The partition of the unity matrix is the key aspect which distinguishes the restricted version from the unrestricted version of the additive Schwarz preconditioner [20]. It is important to note that the RAS preconditioner yields to non-symmetric iterations even for symmetric matrices. Thus, iterative solvers capable of dealing with non-symmetric system must be used, as the GMRES method.

The main difference between substructuring and the RAS preconditioners is the presence of overlapping between subdomains in the RAS, building a preconditioner more complicated than a simple block Jacobi preconditioner. Figure 2 shows the subdomain decomposition of the block Jacobi preconditioner and the Additive Schwarz preconditioner for the Schur complement matrix. This type of partitioning is similar to the one used in [22].

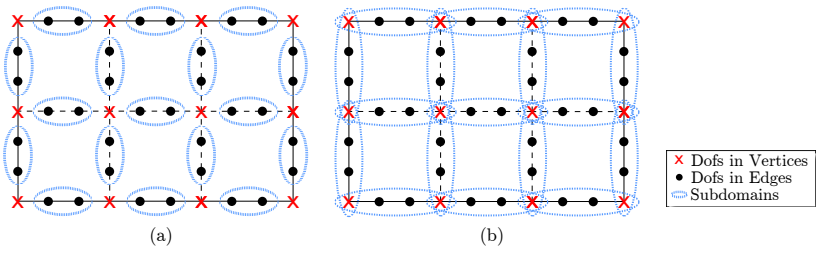


Figure 2: Subdomains decomposition of (a) substructuring preconditioner and (b) the Additive Schwarz preconditioner.

For comparative reasons a classical $ILU(0)$ decomposition, P_{ILU}, is also implemented.

6. Numerical Results

A two-dimensional and a three-dimensional problems in the approximation of two energy groups have been considered to study the performance of the proposed preconditioners to accelerate the solution of the linear systems associated with the blocks $\mathcal{L}_{11}$ and $\mathcal{L}_{22}$. For this purpose, the main quantities of interest are the largest eigenvalue λ, and the neutronic power density at element i, i.e.,

$$P_i = \frac{1}{V_i} \int_{V_i} \left(\Sigma_{f1} \Phi_1 + \Sigma_{f2} \Phi_2 \right) \mathrm{d}V.$$

where P_i is the obtained neutronic power density in the i-th cell (cell averages), and V_i is the volume of the cell. The neutronic flux is normalized in such a way that the average power in the reactor is 1. The accuracy of the schemes will be studied by observing the error in the eigenvalue and in the power density. For the eigenvalue, λ, the error will be measured in pcm units (10^{-5}), i.e., $\Delta\lambda = 10^5 \, |\lambda - \lambda^*|$, where λ^* is the reference value. Instead, the power density, the mean relative error $\bar{\varepsilon}$, and the maximum absolute error $\varepsilon_{\max}$, for the fission density with respect to the reference value are considered,

$$\bar{\varepsilon} = \frac{1}{V_t} \sum_{i=1}^{K} \frac{|P_i - P_i^*|}{|P_i|} \, V_i \, , \quad \text{and} \quad \varepsilon_{\max} = \max_i |P_i - P_i^*| \, ,$$

where P_i^* denotes the reference power in the i-th cell (cell averages), and V_t is the total volume of the reactor.

To compare the performance of the different preconditioners the average number of iterations needed to solve the systems associated with the linear systems $\mathcal{L}_{11}$ and $\mathcal{L}_{22}$ in the eigenvalue calculation is used. Also the memory used by the matrix objects and the preconditioners is shown.

We recall here that the code implementing the solution of the Lambda modes problem has been written in C++ using the open source finite element library *deal.II* [16]. The eigenvalue problem has been solved using the library SLEPc [13]. All the calculations have been performed in a computer with an Intel® i3-3220 CPU @ 3.30GHz processor with 8 GB of RAM running Ubuntu GNU/Linux 14.04. A relative tolerance of 10^{-7} has been set for the Krylov-Schur eigenvalue solver in all the examples. Also a relative tolerance of 10^{-9} has been set for the linear systems.

6.1. Two-dimensional problem: BIBLIS Reactor

First, a two-dimensional reactor example is considered, the BIBLIS 2D benchmark [23]. This is a classical two-group neutron diffusion problem taken as a benchmark for different numerical codes. It has 257 different assemblies including 64 cells modelling the reflector. The definition of the 8 different materials and their cross sections are given in Figure 3 and Table 1. Reference solution is extracted from [23].

Table 2 shows the size of the problem and the accuracy of the solutions obtained with different degrees of the polynomials, p, used in the finite element method. It is observed that it is necessary to use large degree polynomials in the finite element method to obtain solutions of the problem with high accuracy, thus motivating the use of a preconditioner.

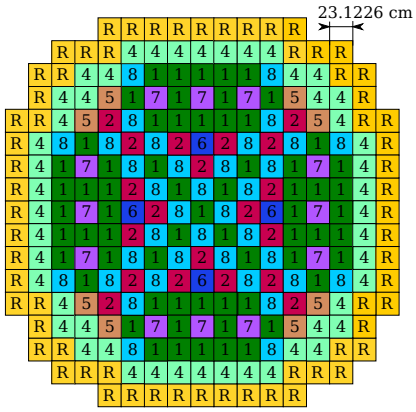


Figure 3: BIBLIS reactor materials definition.

Table 1: Macroscopic cross sections of the 2d problem: BIBLIS reactor.

Material	$\mathbf{D_1}$ (cm)	$\mathbf{D_2}$ (cm)	$\boldsymbol{\Sigma_{a1}}$ (1/cm)	$\boldsymbol{\Sigma_{a2}}$ (1/cm)	$\boldsymbol{\Sigma_{12}}$ (1/cm)	$\boldsymbol{\nu\Sigma_{f1}}$ (1/cm)	$\boldsymbol{\nu\Sigma_{f2}}$ (1/cm)
1	1.4360	0.3635	0.0095042	0.075058	0.017754	0.0058708	0.096067
2	1.4366	0.3636	0.0096785	0.078436	0.017621	0.0061908	0.103580
R	1.3200	0.2772	0.0026562	0.071596	0.023106	0.0	0.0
4	1.4389	0.3638	0.0103630	0.091408	0.017101	0.0074527	0.132360
5	1.4381	0.3665	0.0100030	0.084828	0.017290	0.0061908	0.103580
6	1.4385	0.3665	0.0101320	0.087314	0.017192	0.0064285	0.109110
7	1.4389	0.3679	0.0101650	0.088024	0.017125	0.0061908	0.103580
8	1.4393	0.3680	0.0102940	0.090510	0.017027	0.0064285	0.109110

Table 2: Summary of size and accuracy results for the BIBLIS reactor for different polynomial degrees in the finite element method.

Degree p	**Number of cells**	**Number of DoF**	**Eigenvalue iterations**	**Eigenvalue** λ_1	**Eigenvalue** $\Delta\lambda$ **(pcm)**	**Neutronic Power** $\bar{\varepsilon}$ **(%)**	**Neutronic Power** $\varepsilon_{\max}$
1	255	290	48	1.021764	360	1.09e+1	6.43e-1
2	255	1089	48	1.025709	22	3.45e+0	1.92e-1
3	255	2398	48	1.025601	12	2.10e+0	9.15e-2
4	255	4217	48	1.025533	4.9	4.47e-1	2.31e-2
5	255	6546	48	1.02549	0.7	1.27e-1	5.57e-3
6	255	9385	48	1.025485	0.2	1.90e-2	9.42e-4
7	255	12734	48	1.025483	0.0	4.73e-3	1.87e-4
Reference				1.025483			

Tables 3 and 4 display the performance of the different preconditioners for the Schur complement matrix in terms of the average number of iterations, memory used by the matrix related elements, and CPU time used to compute the solution for $p = 3$ and $p = 5$ degrees in the finite element method. These numerical results show a decrease of the number of iterations as the preconditioners become more complete. However, this improvement in the number of iterations does not represent an improvement in the computational time. The reason is that this benchmark is not big enough to show a time reduction in the Schwarz methodology, compensating the extra overhead needed to assemble and decompose the different sub-matrices. The $ILU(0)$ preconditioner is not the best neither, due to the fact that the diagonal preconditioner P_d is very fast to compute, so that the small reduction of the iteration counts is enough when considering the reduced cost of applying and computing the preconditioner. Moreover, it can be observed that the reduction in the number of iterations for the matrix $\mathcal{L}_{11}$ is different from the reduction obtained for $\mathcal{L}_{22}$, i.e., the preconditioners used here are sensitive to the coefficients of the equation.

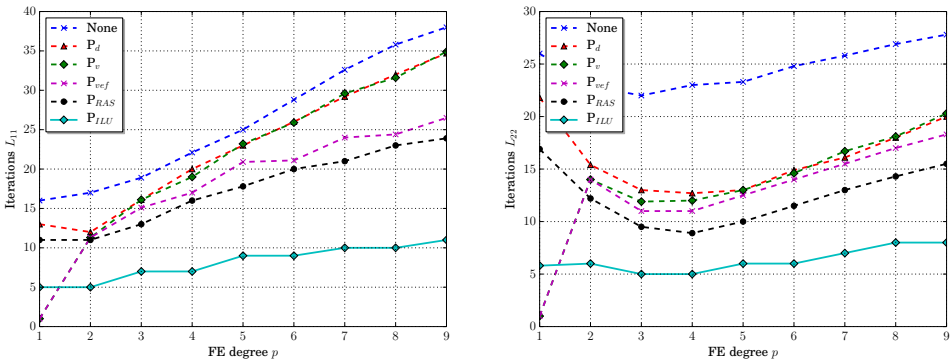
Figures 4a and 4b show the average number of iterations for solving the system related to the matrix $\mathcal{L}_{11}$ and $\mathcal{L}_{22}$, respectively using finite elements from degree 1 to degree 9. In these Figures can be seen that the number of iterations grows as the finite element degree increases in an almost linear way. This can be explained because when p is increased the number of non-zeros inside the matrix is enlarged and thus its condition number. However the number of iterations show a particular behaviour for low degree finite elements because of the relative importance of the degrees of freedom related to the vertices in these matrices.

Table 3: Preconditioners performance for BIBLIS reactor with $p = 3$.

Preconditioner	iterations $\mathcal{L}_{11}$	iterations $\mathcal{L}_{22}$	Memory (MB)	Time (s)
None	18.9	22.0	0.86	0.27
P_d	16.1	13.0	0.86	0.20
P_v	16.1	11.9	0.98	0.29
P_{vef}	15.1	11.0	1.1	0.51
P_{RAS}	13.0	9.5	1.1	0.44
P_{ILU}	7.0	5.0	1.6	0.22

Table 4: Preconditioners performance for BIBLIS reactor with $p = 5$.

Preconditioner	iterations $\mathcal{L}_{11}$	iterations $\mathcal{L}_{22}$	Memory (MB)	Time (s)
None	25.0	23.3	3.9	0.80
P_d	23.0	13.0	3.9	0.69
P_v	23.2	13.0	4.1	0.74
P_{vef}	20.9	12.5	4.3	1.10
P_{RAS}	17.8	10.0	4.4	0.92
P_{ILU}	9.0	6.0	6.0	0.75



(a) Iterations for the $\mathcal{L}_{11}$.

(b) Iterations for the $\mathcal{L}_{22}$.

Figure 4: Averaged number of iterations depending on the finite element degree used for BIBLIS reactor.

6.2. *Three-dimensional problem: IAEA Reactor*

As a second problem, a standard three-dimensional problem is considered. It is the IAEA PWR 3D benchmark [24]. The core is composed by 241 rod assemblies including 64 assemblies modelling the reflector. The definition of this reactor is given in Figure 5 and the cross sections of the different materials are shown in Table 5. Albedo boundary conditions are used with a extrapolation distance of $2.13 \times D_g$.

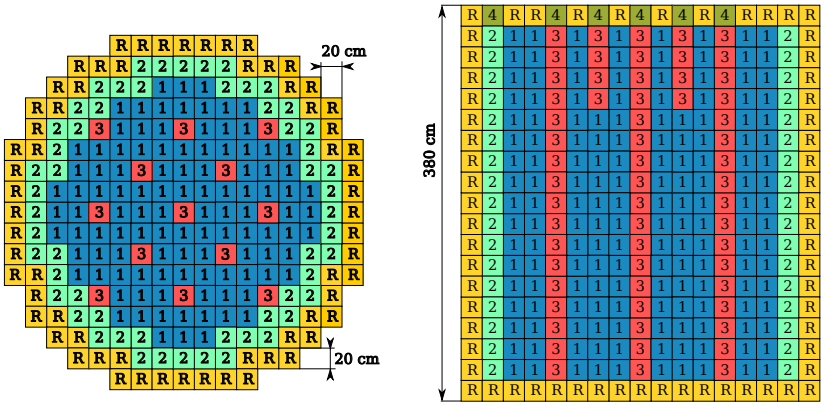


Figure 5: Geometry and material definition of the IAEA 3D reactor [24].

Table 6 shows a summary of the size of the problem and the accuracy of the solutions for different degrees of the polynomials, p, used in the finite element method. The reference values are extracted from [24]. As in the case of the previous benchmark, to obtain accurate solutions it is necessary to use high degree polynomials in the finite element method.

Table 5: Macroscopic cross sections of the IAEA 3D reactor [24].

Material	$\mathbf{D_1}$ (cm)	$\mathbf{D_2}$ (cm)	$\boldsymbol{\Sigma_{a1}}$ (1/cm)	$\boldsymbol{\Sigma_{a2}}$ (1/cm)	$\boldsymbol{\Sigma_{12}}$ (1/cm)	$\boldsymbol{\nu\Sigma_{f1}}$ (1/cm)	$\boldsymbol{\nu\Sigma_{f2}}$ (1/cm)
1	1.500	0.400	0.010	0.085	0.020	0.000	0.135
2	1.500	0.400	0.010	0.130	0.020	0.000	0.135
3	1.500	0.400	0.010	0.080	0.020	0.000	0.135
4	2.000	0.300	0.000	0.055	0.040	0.000	0.000
R	2.000	0.300	0.000	0.010	0.040	0.000	0.000

Table 6: Summary of size and accuracy results for the IAEA 3D reactor for different polynomial degrees in the finite element method.

Degree p	**Number of cells**	**Number of DoF**	**Eigenvalue iterations**	**Eigenvalue** λ_1	**Eigenvalue** $\Delta\lambda$ **(pcm)**	**Neutronic Power** $\bar{\varepsilon}$ **(%)**	**Neutronic Power** $\varepsilon_{\max}$
1	4579	5520	48	1.044942	1603	6.4e+1	1.84e+0
2	4579	40287	48	1.030340	123	7.6e+0	2.17e-1
3	4579	131776	48	1.029139	6.9	1.4e+0	4.10e-2
4	4579	307461	48	1.029085	1.0	1.6e-1	4.64e-3
5	4579	594816	48	1.029070	0.5	3.7e-2	1.12e-3
6	4579	1021315	48	1.029077	0.2	2.9e-3	8.60e-5
7	4579	1614432	48	1.029074	0.1	7.9e-4	2.30e-5
Reference				1.029075			

Tables 7 and 8 display the performance of the different preconditioners in terms of the average number of iterations, memory used by the matrix related elements and the CPU time needed to solve the eigenvalue problem when the degrees $p = 3$ and $p = 5$ are used in the finite element method. These Tables show a decrease of the number of iterations as the preconditioners become more complete. However, this improvement in the number of iterations does not always represent a decrease in the computational time because of the extra overhead needed to build and apply the preconditioner. Also, it can be seen that the partial preconditioner including the vertices, P_v, do not present significant improvement with respect to the previous preconditioners P_d, which is much faster to build and apply. However when the preconditioner includes the terms related with the vertices edges and face, P_{vef}, a significant improvement is achieved.

Nevertheless, for this benchmark, the fastest preconditioner is the proposed P_{RAS}, where the gain in CPU time is larger for the highest polynomial degree used. Here, although the number of iterations needed by the preconditioner based on the $ILU(0)$ decomposition is smaller, the time used to construct this preconditioner and to apply it, is larger than for the preconditioner based on the Schwarz method. It means that the saving in the memory usage is also improving the calculation time, because a smaller preconditioner is not only much faster to build, but also faster to apply.

Figures 6a and 6b show the average number of iteration for solving the system related to the matrix $\mathcal{L}_{11}$ and $\mathcal{L}_{22}$, respectively using finite elements from degree 1 to degree 6. In these Figures can be seen that the number of iterations grows as the finite element degree increases in an almost constant way. This can be explained because increasing p enlarges the number of non-

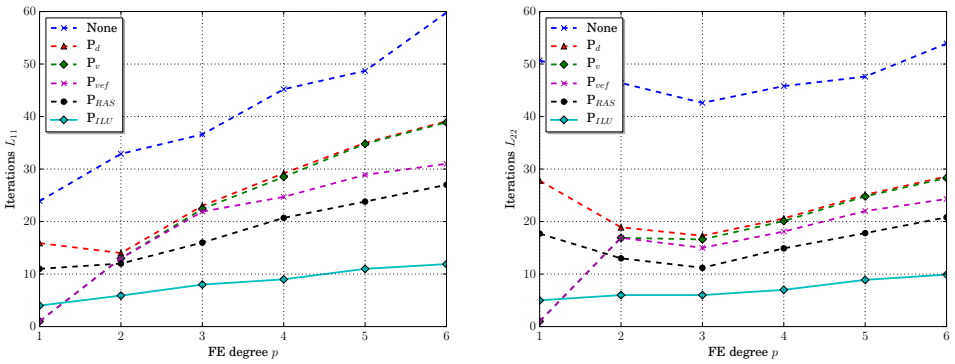
zeros inside the matrix and thus its condition number. However, the number of iterations show a particular behaviour for low degree finite elements, $p = 1$ and $p = 2$, because these matrices are almost composed of degrees of freedom related to the vertices.

Table 7: Preconditioners performance for IAEA 3D reactor using $p = 3$.

Preconditioner	iterations $\mathcal{L}_{11}$	iterations $\mathcal{L}_{22}$	Memory (MB)	Time (s)
None	36.6	42.6	220	130
P_d	23.0	17.3	220	60
P_v	22.4	16.6	230	63
P_{vef}	21.9	15.0	240	63
P_{RAS}	16.0	11.2	290	53
P_{ILU}	8.0	6.0	500	72

Table 8: Preconditioners performance for IAEA 3D reactor with $p = 5$.

Preconditioner	its $\mathcal{L}_{11}$	its $\mathcal{L}_{22}$	Memory (MB)	Time (s)
None	48.7	47.6	2500	1500
P_d	35.1	25.1	2500	1200
P_v	34.8	24.8	2500	1200
P_{vef}	29.0	22.1	2500	890
P_{RAS}	23.9	17.9	2800	800
P_{ILU}	11.0	8.9	4600	1280



(a) Iterations for the $\mathcal{L}_{11}$.

(b) Iterations for the $\mathcal{L}_{22}$.

Figure 6: Averaged number of iterations depending on the finite element degree used for the IAEA 3D reactor.

7. Conclusions

Different preconditioning strategies for the system matrices arising in a high order finite element discretization of the neutron diffusion equation are studied for two-dimensional and three-dimensional problems. These preconditioners are based on domain decomposition techniques, making use of a partition of the degrees of freedom on vertices, edges, faces and interiors, obtained through the identifications of the different shape functions with collocation points over each element. No improvement is observed in two-dimensional problems over traditional Jacobi preconditioners, due to the fact that the overhead cost of building and applying the preconditioner is not compensated by a significant enough improvement of the convergence ratio. Nevertheless, for large enough linear systems, as is the case for three dimensional problems, the preconditioners show an improvement of the convergence time for the studied linear systems, as well as for the memory usage, outperforming classical and widely used preconditioners as the Incomplete LU decomposition.

An additional feature studied here is that the linear systems have been preprocessed with the Schur complement method, in order to reduce the memory usage and the CPU time for convergence, and these Schur matrices have then been preconditioned. We recall that this preprocessing technique, also known as static condensation, already represents an improvement in convergence rates over the classical methodology used for this problem. It is due to the fact that many linear systems have to be solved with the same coefficient matrices and different right hand sides during the application of the eigenvalue solver, thus easily compensating initial cost of setting the Schur complement system.

As possible extensions, the application of these preconditioners for matrix-free algorithms can be studied. A matrix-free application should consider the extra cost of applying the Schur complement system at each iteration and the problem of building the preconditioner without explicitly building the matrix. This method can be used to efficiently accelerate the procedures of the neutron transport equation. Moreover, the construction of a preconditioner that performs equally well for different configurations of the same problem is of interest. The improvement for the convergence could be estimated a priory for the particular system in function of its size, thus choosing the optimal preconditioner for the particular situation.

Acknowledgements

The work has been partially supported by the spanish Ministerio de Economía y Competitividad under projects ENE 2014-59442-P and MTM2014-58159-P, the Generalitat Valenciana under the project PROMETEO II/2014/008 and the Universitat Politècnica de València under the project FPI-2013. The work has also been supported partially by the Swedish Research Council (VR-Vetenskapsrådet) within a framework grant called DREAM4SAFER, research contract C0467701.

References

[1] H. A. Schwarz, Über einen grenzübergang durch alternierendes verfahren, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich 15 (1870) 272–286.

[2] B. Smith, P. Bjorstad, W. Gropp, Domain decomposition: parallel multilevel methods for elliptic partial differential equations, Cambridge university press, 2004.

[3] M. J. Gander, Schwarz methods over the course of time, Electronic Transactions on Numerical Analysis 31 (2008) 228–255.

[4] M. Barrault, B. Lathuiliere, P. Ramet, J. Roman, Efficient parallel resolution of the simplified transport equations in mixed-dual formulation, Journal of Computational Physics 230 (5) (2011) 2004–2020.

[5] E. Jamelot, P. C. Jr, Fast non-overlapping schwarz domain decomposition methods for solving the neutron diffusion equation, Journal of Computational Physics 241 (2013) 445 – 463.

[6] A.-M. B. Erell Jamelot, J.-J. Lautard, Domain decomposition for the spn solver minos, Transport Theory and Statistical Physics 41 (7) (2012) 495–512. doi:10.1080/00411450.2012.694827.

[7] G. Cossa, V. Giusti, B. Montagnini, A boundary element-response matrix method for criticality diffusion problems in xyz geometry, Annals of Nuclear Energy 37 (7) (2010) 953 – 973. doi:http://dx.doi.org/10.1016/j.anucene.2010.03.012.

[8] J. A. Rathkopf, W. R. Martin, The finite element response matrix method for the solution of the neutron transport equation, Progress in Nuclear Energy 18 (1) (1986) 237–250.

[9] W. M. Stacey, Nuclear Reactor Physics, John Wiley & Sons, 2007.

[10] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[11] K. Smith, Assembly homogenization techniques for light water reactor analysis, Progress in Nuclear Energy 17 (3) (1986) 303 – 335.

[12] D. Ferguson, K. Derstine, Optimized iteration strategies and data management considerations for fast reactor finite difference diffusion theory codes, Nuclear Science and Engineering 64 (2) (1977) 593–604.

[13] V. Hernandez, J. E. Roman, V. Vidal, SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems, ACM Trans. Math. Software 31 (3) (2005) 351–362.

[14] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The finite element method: its basis and fundamentals, Butterworth-Heinemann, 2005.

[15] A. Vidal-Ferrandiz, R. Fayez, D. Ginestar, 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 (2014) 338–349. doi:http://dx.doi.org/10.1016/j.anucene.2014.05.026.

[16] W. Bangerth, R. Hartmann, G. Kanschat, deal.II – a general purpose object oriented finite element library, ACM Trans. Math. Softw. 33 (4) (2007) 24/1–24/27.

[17] S. J. Sherwin, M. Casarin, Low-energy basis preconditioning for elliptic substructured solvers based on unstructured spectral/hp element discretization, J. Comput. Phys. 171 (1) (2001) 394–417. doi:10.1006/jcph.2001.6805.

[18] D. Pardo, J. lvarez Aramberri, M. Paszynski, L. Dalcin, V. Calo, Impact of element-level static condensation on iterative solver performance, Computers & Mathematics with Applications 70 (10) (2015) 2331 – 2341. doi:http://dx.doi.org/10.1016/j.camwa.2015.09.005.

[19] H. A. Schwarz, Ueber einen Grenzübergang durch alternirendes Verfahren, Zürcher u. Furrer, Zrich, 1870.

[20] X.-C. Cai, M. Sarkis, A restricted additive schwarz preconditioner for general sparse linear systems, SIAM Journal on Scientific Computing 21 (2) (1999) 792–797.

[21] E. Efstathiou, M. Gander, Why restricted additive schwarz converges faster than additive schwarz, BIT Numerical Mathematics 43 (5) (2003) 945–959. doi:10.1023/B:BITN.0000014563.33622.1d.

[22] L. M. Carvalho, L. Giraud, P. Le Tallec, Algebraic two-level preconditioners for the schur complement method, SIAM Journal on Scientific Computing 22 (6) (2001) 1987–2005.

[23] E. Müller, Z. Weiss, Benchmarking with the multigroup diffusion high-order response matrix method, Annals of Nuclear Energy 18 (9) (1991) 535 – 544.
[24] Computational Benchmark Problems Committee, Benchmark Problem Book, American Nuclear Society, Argonne National Laboratory, 1977. doi:10.2172/5037820.