Nuclear Mathematical and Computational Sciences: A Century in Review, A Century Anew
Gatlinburg, Tennessee, April 6-11, 2003, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2003)

KRYLOV SUBSPACE ITERATIONS FOR THE CALCULATION OF k-EIGENVALUES WITH S$_N$ TRANSPORT CODES

James S. Warsa, Todd A. Wareing, Jim E. Morel, John M. McGhee
Transport Methods Group
Los Alamos National Laboratory
Los Alamos, NM 87545-0001
warsa@lanl.gov, wareing@lanl.gov, jim@lanl.gov, mcghee@lanl.gov

Richard B. Lehoucq
Computational Mathematics and Algorithms Department
Sandia National Laboratories
Albuquerque, NM 87185-1110
rblehou@sandia.gov

ABSTRACT

We apply the Implicitly Restarted Arnoldi Method (IRAM), a Krylov subspace iterative method, to the calculation of k-eigenvalues for criticality problems. We show that the method can be implemented with only modest changes to existing power iteration schemes in an S$_N$ transport code. Numerical results on three dimensional unstructured tetrahedral meshes are shown. Although we only compare the IRAM to unaccelerated power iteration, the results indicate that the IRAM is a potentially efficient and powerful technique, especially for problems with dominance ratios approaching unity.

Key Words: criticality eigenvalues, Implicitly Restarted Arnoldi Method (IRAM), deterministic transport methods

1 INTRODUCTION

The dominant k-eigenvalue(s) and corresponding eigenfunction(s) for criticality problems are often computed with classical iterative methods such as the power iteration method. Convergence of the power iteration is determined by the dominance ratio, or the ratio of the second largest eigenvalue to the maximum eigenvalue [1]. Problems of practical interest often have dominance ratios large enough to make calculations of the maximum k-eigenvalue difficult or intractable. Acceleration techniques may improve convergence, but they are often are not effective enough for the most difficult problems. This is because they rely on parameters which can not always be robustly estimated. Our purpose is to illustrate the use of an efficient Krylov subspace iterative method whose convergence is not limited by high dominance ratios for the calculation of the dominant eigenvalues for deterministic S$_N$ transport methods. This method is the Implicitly Restarted Arnoldi Method (IRAM) [2]. The IRAM is designed to be a robust and stable algorithm, with only the dimension of the Krylov subspace (the number of working vectors) as a free parameter.

A paper describing the application of the IRAM to k-eigenvalue calculations based on two-group diffusion theory has appeared recently [3]. Other alternatives to simple power iteration, most often for diffusion models, have been investigated in the past. For example, a good deal of work appears in the Russian literature indicating that approaches such as inverse and shifted inverse iteration have been applied to eigenvalue computations [4]. Subspace iteration (which is similar in spirit to Krylov subspace methods) and variational acceleration of that method, has been recently investigated in the context of a diffusion model [5]. Another example is found in a recent paper describing inverse iteration for k-eigenvalue calculations in one dimensional transport problems [6].

We describe how the IRAM can be easily implemented in an existing deterministic transport code using the freely available ARPACK software [7]. Because the IRAM solver is "wrapped around" the power iteration, the existing implementation does not need to be significantly altered. The eigenvectors corresponding to several of the largest eigenvalues can also be calculated efficiently with no further modification and only a little extra computation.

Our implementation is in the AttilaV2 transport code [8], a linear discontinuous finite element spatial discretization of the S$_N$ equations on unstructured meshes, described in Sec. 2.1. This is followed by a discussion of the solution methods we consider for the k-eigenvalue problem. This includes power iteration, discussed in Sec. 2.2, followed by qualitative introduction to the IRAM in Sec. 2.3. Numerical experiments are presented in Sec. 3 that illustrate significant improvements in efficiency due to the high rate of convergence of the IRAM. The first set of numerical results are presented in Sec. 3.1. Unaccelerated power iteration is compared to the IRAM for problems specifically constructed to have increasingly high dominance ratios. The next set of results, shown in Sec. 3.2, are intended to demonstrate the performance of the IRAM on a more realistic reactor problem. The first part in Sec. 3.2.1 the energy upscatter portion of scattering matrix set to zero. Then in Sec.3.2.2 we examine the decreased efficiency that arises from applying the IRAM – as currently implemented – to problems with upscatter (note that upscatter was not considered in [3] or [5]).

2 SOLUTION OF THE DISCRETE ORDINATES k-EIGENVALUE PROBLEM

2.1 Discretized S$_N$ Eigenvalue Problem on Tetrahedral Meshes

We use standard notation [1]. Given an angular quadrature set with N specified nodes and weights $\{\hat{\Omega}_m, w_m\}$ and anisotropic scattering of Legendre order L, the steady-state S$_N$ transport equation for energy group $g = 1, \ldots, G$ in a three-dimensional domain $\boldsymbol{r} \in V$ with boundary $\boldsymbol{r}_b \in \partial V$, is

$$\left(\hat{\Omega}_m \cdot \nabla + \sigma_{t,g}(\boldsymbol{r})\right) \psi_{g,m}(\boldsymbol{r}) = \sum_{g'=1}^{G} \sum_{l=0}^{L} \sigma_{l,g' \to g}(\boldsymbol{r}) \sum_{n=-l}^{l} Y_{ln}(\hat{\Omega}_m)\, \phi^n_{l,g'}(\boldsymbol{r}) + \frac{1}{k}\, \chi_g(\boldsymbol{r}) \sum_{g'=1}^{G} \nu\sigma_{f,g'}(\boldsymbol{r})\, \phi^0_{0,g'}(\boldsymbol{r}), \tag{1a}$$

for $m = 1, \ldots, N_A$. Here, $Y_{ln}(\hat{\Omega})$ are the normalized spherical harmonics functions and the scalar flux moments are given by

$$\phi^n_{l,g}(\boldsymbol{r}) = \sum_{m=1}^{N_A} w_m Y_{ln}(\hat{\Omega}_m) \psi_{g,m}(\boldsymbol{r}). \tag{1b}$$

This is an eigenproblem for the eigenvalue k.

The linear DFEM discretization on tetrahedra is derived with a Galerkin variational formulation. For a given energy group g, the angular flux is expanded in a set of four linear basis functions L_j on a tetrahedron T_s (cell index s):

$$\psi_{g,m,s}(\boldsymbol{r}) = \sum_{j=1}^{4} \psi_{g,m,j,s} L_j(\boldsymbol{r}). \tag{2}$$

The weak form of the transport equation is then constructed for each of the functions $L_i, i = 1, \ldots, 4$ on cell T_s for angle m:

$$\begin{aligned}\int_{\partial T_s} \left(\hat{\Omega}_m \cdot \hat{n}\right) \psi^b_{g,m}\, L_i\, dS - \int_{T_s} \psi_{g,m,s}(\boldsymbol{r}) \left(\hat{\Omega}_m \cdot \nabla L_i\right) dV + \sigma_{t,g,s} \int_{T_s} \psi_{g,m,s}(\boldsymbol{r})\, L_i\, dV \\ = \sum_{g'=1}^{G} \sum_{l=0}^{L} \sigma_{l,g' \to g,s} \sum_{n=-l}^{l} Y_{ln}(\hat{\Omega}_m) \int_{T_s} \phi^n_{l,g'}(\boldsymbol{r})\, L_i\, dV + \frac{1}{k}\, \chi_{g,s} \sum_{g'=1}^{G} \nu\sigma_{f,g',s} \int_{T_s} \phi^0_{0,g'}(\boldsymbol{r})\, L_i\, dV.\end{aligned} \tag{3a}$$

This gives four equations for the four unknowns $\psi_{m,j,s}$ on every cell s for every angle $\hat{\Omega}_m$. Cross sections and other parameters are constant in a cell. The fluxes on the cell interfaces, $\psi^b_{g,m}$, are defined to make the approximation discontinuous as follows. For a face j on ∂T_s, whose outward normal is $\hat{n}_j$,

$$\left(\hat{\Omega}_m \cdot \hat{n}_j\right) \psi^b_{g,m} = \begin{cases} \left(\hat{\Omega}_m \cdot \hat{n}_j\right) \psi_{g,m,h(j),s}, & \hat{\Omega}_m \cdot \hat{n}_j > 0, \quad \hat{n}_j \text{ in } V \\ \left(\hat{\Omega}_m \cdot \hat{n}_j\right) \psi_{g,m,h(j),p}, & \hat{\Omega}_m \cdot \hat{n}_j < 0, \quad \hat{n}_j \text{ in } V \backslash \partial V \\ \left(\hat{\Omega}_m \cdot \hat{n}_j\right) \Gamma(\hat{\Omega}_m), & \hat{\Omega}_m \cdot \hat{n}_j < 0, \quad \hat{n}_j \text{ on } \partial V \end{cases} \tag{3b}$$

where p is the cell that shares face j with cell s. The subscript $h(j)$ denotes three vertices i on a face j of a given cell. We will consider either vacuum boundary conditions, $\Gamma(\hat{\Omega}_m) = 0$, or specular reflection boundary conditions, where the reflected image of $\hat{\Omega}_m$, $\hat{\Omega}_{m'}$, is is defined by $\hat{\Omega}_{m'} = \hat{\Omega}_m - 2\,\hat{n}(\hat{\Omega}_m \cdot \hat{n})$. This determines an m' for $\hat{\Omega}_m$ and $\hat{n} = \hat{n}_j$ such that we can set $\Gamma(\hat{\Omega}_m) = \psi_{g,m',i(j),s}$. Reflective boundary faces are aligned parallel to the x, y or z coordinate axes such that the quadrature sets we use contain reflected pairs $\hat{\Omega}_m$ and $\hat{\Omega}_{m'}$ that satisfy the definition of specular reflection.

The integrals in (3) are evaluated, either analytically or by quadrature approximation, for every cell in the mesh. Note that we use a fully lumped version of (3). Describing it goes beyond the scope of this work, but suffice it to say that this lumping preserves the diffusion limit in thick, diffusive regimes (see [9]).

2.2 Power Iteration

We write the discretized S$_N$ equations (3) in operator notation as

$$\boldsymbol{L}\psi = \boldsymbol{MSD}\psi + \frac{1}{k}\boldsymbol{FD}\psi, \tag{4}$$

where detailed meanings of the operators can be deduced by comparing (4) with (3). Briefly, $\boldsymbol{L}$ is the streaming-plus-removal operator and $\boldsymbol{F}$ is the fission operator. The operators $\boldsymbol{M}$ and $\boldsymbol{D}$ represent the "moment-to-discrete" and "discrete-to-moment" operators, respectively, in the nomenclature of [10]. They convert a vector of scalar flux moments to angular fluxes and vice versa, respectively. The scattering operator $\boldsymbol{S}$ is (block) lower triangular if no upscatter is present. It is in general nonsymmetric.

Rearranging (4), applying $\boldsymbol{D}$ to both sides, and introducing iteration index ℓ, we have

$$\phi^{\ell+1} = \frac{1}{k_\ell}\boldsymbol{A}\phi^\ell, \tag{5}$$

where $\boldsymbol{A}$ is defined in operator notation as $\boldsymbol{A} = \boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{F}$, with $\boldsymbol{H} = \boldsymbol{L} - \boldsymbol{M}\boldsymbol{S}\boldsymbol{D}$. The vector ϕ represents all the scalar flux moments needed to construct the scattering source in (1).

The largest eigenvalue k is computed by iterating (5) until the expression

$$k_{\ell+1} = k_\ell \frac{\|\phi^{\ell+1}\|}{\|\phi^\ell\|}, \tag{6}$$

converges to within some tolerance, where $\|\phi\|$ represents some discrete norm of ϕ over all cells in the problem. Often, the norm used is just the fission rate in the problem. Instead of (6), we use the Rayleigh quotient, given by

$$\begin{aligned} k_{\ell+1} &= \frac{(\boldsymbol{A}\phi^\ell, \phi^\ell)}{(\phi^\ell, \phi^\ell)} && (7) \\ &= k_\ell \frac{(\phi^{\ell+1}, \phi^\ell)}{(\phi^\ell, \phi^\ell)}, && (8) \end{aligned}$$

to update the eigenvalue estimate. Here, $(\cdot,\cdot)$ is a discrete inner product over all cells. We have observed that using the Rayleigh quotient for the eigenvalue estimate can usually improve the efficiency of the power iteration method by providing a better estimate of the eigenvalue earlier in the iterative process [11].

Power iteration converges to the largest eigenvalue in magnitude, k_1 (the dominant eigenvalue), where the convergence rate is determined by the dominance ratio $\delta = |k_2|/|k_1|$ where $|k_2| \leq |k_1|$ is the next largest eigenvalue in magnitude [11]. The closer this ratio is to 1.0 the slower the convergence. Power iteration converges except when the eigenvalue with largest magnitude is complex and the (nonsymmetric) operator and initial guess ϕ_0^0 are real. For monoenergetic problems, the k-eigenvalue is real in the S$_N$ approximation, even for anisotropic scattering [12] We do not know that this is also true in the multigroup approximation. We therefore can not state definitively that power iteration will always converge, although experience indicates that the maximum eigenvalue is in fact real, in which case power iteration will converge. Nonetheless, a rigorous justification that the dominant eigenvalue is real for the multigroup S$_N$ equations would be interesting.

Power iteration works with a single vector. Convergence can be improved by using more than one vector in order to make use of information from previous iterations. Classical iterative acceleration techniques, such as Chebyshev iteration or successive over-relaxation (SOR), are sometimes used to accelerate k-eigenvalue power iterations. Such methods use a linear combination of a small number of previous iterates to update the scalar flux [13, 14]. Under certain circumstances it is possible to use knowledge of the spectrum of the operator to find the optimal linear combination that maximizes convergence. In practice, however, the spectrum is not known in advance and estimates of the spectrum are made to compute the necessary parameters. This results in a less-than-optimal method. Nonsymmetry complicates matters because the spectrum could extend into the complex plane, making robust estimation of the acceleration parameters significantly more difficult [15–17]. This aspect is often ignored in implementations and acceleration methods are typically based on the assumption that the operator is symmetric and positive definite. This creates the potential for degrading or destroying the effectiveness of the acceleration. The heuristics, approximations, and code logic needed to make adaptive estimates of the required parameters can also make these acceleration algorithms less than robust and make it difficult to predict if or when a given acceleration method will be successful. While a particular approach may work well in some problems, it may work poorly in others and may even cause the iteration to diverge in others [14].

2.3 The Implicitly Restarted Arnoldi Method

We now briefly describe the Implicitly Restarted Arnoldi method, or IRAM. Examination of (5) shows that power iteration computes the maximum eigenvalue in magnitude of the standard eigenvalue problem

$$\boldsymbol{A}\phi = k\phi. \tag{9}$$

Our approach is to compute the largest few eigenvalues of this standard eigenvalue problem using the IRAM implementation found in the ARPACK software package [7]. The ARPACK implementation is best suited for applications whose matrices are either sparse or not explicitly available. Only the "action" of a matrix on a vector, supplied by the solver, is calculated at every IRAM iteration. A reverse communication mechanism frees the user from any particular data structure formats. This simplified our implementation of the method because the IRAM solver can simply be "wrapped around" the existing power iteration in our transport code. The desired number of eigenvalues and corresponding eigenvectors, the dimension of the Krylov subspace, and the iterative convergence criterion are supplied to the ARPACK software.

The IRAM is a Krylov subspace iterative method. The Krylov subspace of dimension m $\mathcal{K}_m(\boldsymbol{A}, \phi_0) = \text{span}\{\phi_0, \boldsymbol{A}\phi_0, \ldots \boldsymbol{A}^{m-1}\phi_0\}$ is constructed from the same vector sequence generated by power iteration applied to an initial starting vector ϕ_0. An approximate eigenvector-eigenvalue pair, or eigenpair, is found by projecting a vector $x \in \mathcal{K}_m(\boldsymbol{A}, \phi_0)$ onto the Krylov subspace using the Galerkin orthogonality condition $(w, \boldsymbol{A}x - \lambda x) = 0$ for all $w \in \mathcal{K}_m(\boldsymbol{A}, \phi_0)$. The approximate eigenpair (x, λ) is termed a "Ritz pair". If the component of the actual eigenvector that is orthogonal to the subspace is small, then the Ritz pair will be a good approximation to the eigenpair of a nearby problem.

Ritz pairs that satisfy the Galerkin orthogonality condition are computed efficiently using the Arnoldi method, which computes an orthogonal basis—the Arnoldi basis—for $\mathcal{K}_m(\boldsymbol{A}, \phi_0)$. At the same time, an (upper Hessenberg) $(m \times m)$ projection of $\boldsymbol{A}$ on the Krylov subspace is computed. The Ritz pairs are then easily found by the QR algorithm [18] of this matrix. An estimate of the (backward) error in the approximation, or Ritz estimate, is also available through the Arnoldi method. Typically, the quality of the approximation improves as the dimension m of the subspace increases. In exact arithmetic, they are equal after n iterations, where n is the order of $\boldsymbol{A}$.

The cost of maintaining orthogonality of the Arnoldi basis increases with the dimension of the Krylov subspace. The computations could become intractable if the size of the subspace needed for a good approximation to the eigenpair is too large. One way to address this is to fix the computational and storage requirements by restarting the Arnoldi method.

Suppose that we are able to compute m steps of the Arnoldi method where m is chosen so that the cost of maintaining the orthogonality among the Arnoldi vectors (to machine precision) is small. Because we are interested in the s dominant eigenvalues of $\boldsymbol{A}$, consider constructing another set of Arnoldi vectors for $\mathcal{K}_m(\boldsymbol{A}, \hat{\phi}_0)$ such that the first s Arnoldi vectors of this new space span the same space as the Ritz vectors that correspond to the s dominant Ritz values of the original space $\mathcal{K}_m(\boldsymbol{A}, \phi_0)$. This is how the Arnoldi method is *restarted*, continuing until the s dominant eigenvalues emerge to the specified tolerance. Implicit restarting [2, 19] is an efficient and numerically stable way to restart the Arnoldi method or, equivalently, to compute $\hat{\phi}_0$, the new initial vector. The restart is called implicit because of the connection with the implicitly shifted QR algorithm. See [7] for further details on an efficient implementation of implicit restarting. In particular, implicitly restarting an Arnoldi method is equivalent to an accelerated subspace iteration [20] (see [3] for a performance comparison of subspace iteration and the IRAM on a two-group diffusion problem).

In contrast to power iteration, which converges to the dominant eigenvalue k_1 with convergence rate $|k_2|/|k_1|$, the IRAM converges at the rate

$$\frac{\max_{j=r+1,\ldots,n} |P(k_j)|}{\min_{j=1,\ldots,r} |P(k_j)|} \tag{10}$$

after every restart, where n again is the order of $\boldsymbol{A}$, r is the number of Arnoldi vectors remaining after an implicit restart and $P(x) = (x - x_1) \cdots (x - x_r)$. The zeros of the polynomial $P(x)$, $x_i, i = 1, \ldots, r$, and r, are selected so that (10) is as small as possible. Optimal choices do not, in general, exist and would require knowledge of the eigenvalues that we are attempting to approximate. If $r = 1$ and $x_1 = 0$, the power iteration convergence rate is recovered. If $r > 1$, and $x_i = 0$ for $i = 1 \ldots, r$, the convergence rate of subspace iteration is recovered. For both of these cases, r is chosen slightly larger than s (the desired number of eigenvalues), which is a well-known strategy in subspace iteration. The default choice in ARPACK is to choose the roots of $P(x)$ as the *unwanted* eigenvalues of the upper Hessenberg matrix computed during the Arnoldi method. These are the $m - r$ eigenvalues smallest in magnitude. This makes an excellent choice in practice and produces a convergence rate smaller than $|k_2|/|k_1|$. Moreover, this choice of $P(x)$ is mathematically equivalent to computing a new Krylov space with a starting vector that is a linear combination of the r dominant Ritz vectors. The value of r is chosen slightly larger than s in practice, which tends to decrease the ratio (10). See [7, 20] for more details on the IRAM.

Finally, note that power iteration can actually be viewed as a Krylov method with a subspace of dimension one. Similarly, classical techniques used to accelerate power iteration can be viewed as a Krylov method of a small dimension that is equal to the number of working vectors [21]. The parameters for computing the appropriate linear combination of these vectors have to be estimated or approximated in practice, meaning that the methods are non-optimal and possibly limiting their effectiveness. In contrast, the IRAM automatically and efficiently selects the Ritz pair(s) without any a-priori information or parameter estimation.

2.4 Special Considerations

Consider the operator $\boldsymbol{H} = \boldsymbol{L} - \boldsymbol{MSD}$, ignoring any spatial dependence and focusing only on the energy dependence of this operator for the moment. The operators $\boldsymbol{L}$, $\boldsymbol{M}$ and $\boldsymbol{D}$ are block diagonal where each block corresponds to the transport equation for a particular energy group. If the scattering in a problem consists of downscatter only, then the operator $\boldsymbol{S}$ is block lower triangular and the entire operator can be inverted by block forward substitution from high energy to low energy in sequence, with each diagonal block approximately inverted with an inner iteration. If upscatter is present, then the operator $\boldsymbol{S}$ is no longer block lower triangular. In that case, the operator $\boldsymbol{H}$ that is not lower triangular must be inverted iteratively. With power iteration, this inversion is actually “embedded” in the power iterations, because the power iteration method works with a single vector of scalar flux moments. Another important aspect of power iteration that further improves its efficiency is that the scalar flux moments from the most recent outer iteration can be used as an initial guess for the next set of inner iteration(s). This is why the eigenvalue problem is implemented as shown in (5). Scaling the eigenvalue problem with the most recent eigenvalue estimate k_ℓ not only prevents overflow or underflow but it also enables us to use the solution from the previous iteration as an initial guess for computing $\boldsymbol{H}^{-1}$. This significantly reduces the overall computational expense.

However, if there is upscatter in the problem, then the action of $\boldsymbol{H}^{-1}$ has to be computed iteratively at every IRAM iteration, because ARPACK supplies the Arnoldi basis vectors and not the Krylov basis vectors. This obviously has a strong impact on the overall efficiency of the IRAM calculations. We can compute this either with a Krylov method or with the block Gauss-Seidel iteration that is part of the power iteration coding. One possibility to improve efficiency would be to precondition a Krylov iterative method or accelerate the block Gauss-Seidel iteration with the upscatter acceleration method of [22]. While good initial guesses are not available for subsequent inner iterations in the IRAM iterations, we can instead relax the inner iteration convergence tolerance during the course of the IRAM iterations to improve efficiency. The authors of [23, 24] and [25] first observed that this is possible for Krylov subspace iterative methods for the solution of linear systems as well as eigenproblems. The theoretical justification underlying their observations was recently analyzed [26]. In our implementation we set the convergence tolerance for the next level of iteration (within-group inner iterations, or upscatter iteration) to be inversely proportional to the ARPACK residual. Inner (within-group) iterations are accelerated, in the case of source iteration, or preconditioned, in the case of a inner Krylov iteration, with a diffusion synthetic acceleration method [27].

3 NUMERICAL RESULTS

In this section we compare the IRAM to the power iteration method already implemented in the three-dimensional, tetrahedral mesh, transport code AttilaV2. We will present actual measurements of the computational cost for representative problems. We use CGS units and a triangular S_4 Chebyshev-Legendre quadrature for all the results reported here. Calculations are started with random initial vectors. In cases where power iteration is used to initialize the IRAM, the power iteration method is initialized with a random vector and the starting vector for the IRAM is initialized with the result of a few power iterations. Because of the unpredictable nature of the classical acceleration methods for general problems, we only consider unaccelerated power iteration for comparison.

ARPACK uses the backward error $\|\boldsymbol{A}x - \theta x\| \leq |\theta|\epsilon$ to determine convergence to a specified tolerance ϵ, where x and θ are the current Ritz pair. However, rather than calculate the residual explicitly, ARPACK uses the Ritz estimate, an indirect measure of the error in the associated eigenpair, that is readily available through the Arnoldi method [7]. So that the comparison of the computational expense between power iteration and the IRAM is as fair as possible, we use $\|\phi^{\ell+1} - \phi^{\ell}\| \leq \epsilon$ to determine power iteration convergence. All norms are measured in the 2-norm, $(\cdot,\cdot)^{1/2}$, again taking the scalar product over the entire vector of all scalar flux moments on the mesh.

The power iteration results are computed using DSA-accelerated source iteration, and solutions from the most recent outer iteration are used for the initial guess, for the inner within-group iterations. The ARPACK results are computed using a preconditioned Krylov method for the inner iterations, using the source vector (the uncollided flux) as the initial guess. The partially consistent simplified WLA DSA method [28, 29] for acceleration/preconditioning is used for both methods. The convergence criteria for the inners, outers, upscatter iterations (if necessary) and the DSA conjugate gradient (CG) iterations, for the two methods are listed in Table I. The quantities $\|r_{outer}\|$, $\|r_{inner}\|$ and $\|r_{upscat}\|$ are the norms of the residuals in the outer, inner, and upscatter iterations, for the most recent iteration at whatever their respective levels. For the IRAM, the quantity $\|r_o\|$ denotes $\|r_{upscat}\|$ in the case of upscatter and $\|r_{outer}\|$

otherwise. The power iteration initialization for ARPACK uses 10^{-5} for the inner iterations and $10^{-1}\|r_{inner}\|$ for the DSA iterations.

Table I. Convergence tolerance for the different levels of iteration.

	Power Iteration/Source Iteration	IRAM/Krylov Iteration
Outer	10^{-4}	10^{-4}
Inner	$\min(10^{-2}, 10\|r_{outer}\|)$	$\max(10^{-1}\|r_o\|, 10^{-1}\min(10^{-1}, 10^{-1}\|r_o\|/\min(1, \|r_o\|)))$
Upscatter	n/a	$\max(10^{-1}\|r_{outer}\|, 10^{-1}\min(10^{-1}, 10^{-1}\|r_{outer}\|/\min(1, \|r_{outer}\|)))$
DSA	$10^{-1}\|r_{inner}\|)$	$\max(10^{-1}\|r_{inner}\|, 10^{-1}\min(10^{-1}, 10^{-1}\|r_{inner}\|/\min(1, \|r_{inner}\|)))$

3.1 Cylindrical Mesh Problem

The first set of results is for a series of problems with dominance ratios approaching one. The problems are constructed by altering the symmetry of a cylindrical mesh, illustrated in Fig. 1. The cylinder is 3.5 cm in

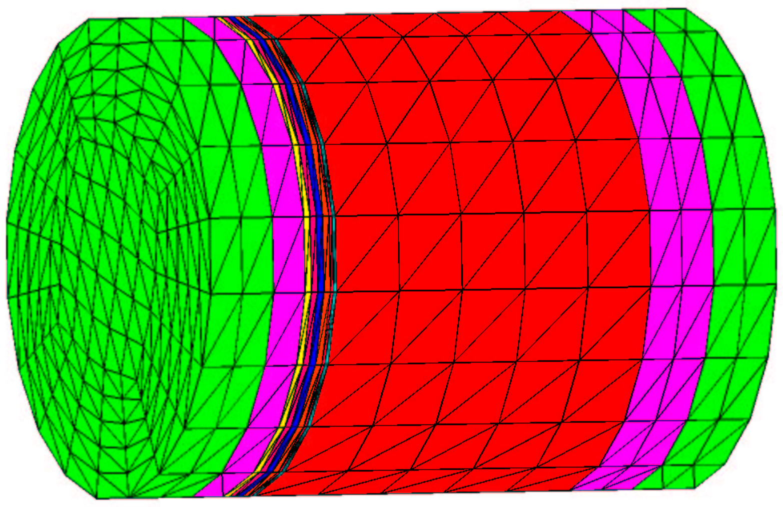

Figure 1. The cylindrical mesh problem. Note the thin disks in the left half of the mesh that are used to alter the symmetry of the problem.

radius and 9 cm long. It consists of a 5 cm layer of B^{10} absorber sandwiched between 1 cm thick water layers and 1 cm layers of highly enriched uranium (HEU). Vacuum boundary conditions are used on all the faces and there 13,500 cells in the mesh. Five 0.1 cm thick regions at the left of the central absorbing region can be seen in Fig. 1. The configuration is symmetric if all five regions are filled with water. The dominance ratio of this symmetric configuration is very close to one, a situation in which power iteration can be expected to converge very slowly. By successively substituting boron for water in the thin regions starting with the region nearest the absorber region we destroy the symmetry and reduce the dominance ratio in the problem. While the actual value of the eigenvalue does not change significantly with the departure from symmetry, the dominance ratio and fundamental eigenvector do.

Five energy group cross sections and fission data were collapsed from the Hansen and Roach sixteen group cross section data sets [30, 31]. The energy groups are collapsed from library groups 1-2, 3-4, 5-8, 9-12, and 13-16. The material composition data are shown in Table II.

Table II. Cross section data for the cylindrical mesh problem. ID is the isotope identifier in the Hansen and Roach library. Density is in g/cm^3.

Material	Density	ID	Mass Fraction
HEU	19.0	u252e1 u282e1	0.95 0.05
Water	1.0	hDE o	0.667 0.333
Boron	10.0	b10	1.0

Results for this sequence of problems are shown in Fig 2, and in Tables III and IV. The inner, within-group iterations for these problems were computed using restarted, flexible-GMRES, FGMRES(10). The computational expense of the methods is measured in floating point operation (FLOP) counts computed on a single dedicated 250 MHz SGI Origin 2000 CPU. Figure 2 illustrates the one to two orders of magnitude savings in computational cost with the IRAM relative to unaccelerated power iteration. The different curves in the figure are for several combinations of the number of eigenvalues sought, $\texttt{nev} = 0$ or 1, the maximum dimension of the Krylov subspace, $\texttt{ncv} = 5$ or 10, and the number of power iterations taken in initializing the IRAM iterations, $\texttt{init} = 0$ or 5. These values were chosen for illustration only. The figure shows that the resulting computational expense is not very sensitive to either `nev` or `ncv` while the initialization with a few power iterations is certainly worth the cost. For efficiency, the initialization steps involved a single inner iteration for each group for each power iteration and this was enough, apparently, to achieve a good initial guess.

In Tables III and IV we list the number of outer and total number of inner iterations, respectively, for the IRAM and the the power iteration. We found that the majority of the computations are devoted to the within-group inner iterations, taking around 95-99% of the total effort. This makes it is possible to accurately gauge the relative performance of the two methods by simply comparing inner iteration counts. The inner iteration counts demonstrate that computational expense depends more on the size of the Krylov subspace than on dominance ratio. The outer iteration counts show that the convergence of the IRAM is only mildly sensitive to the dominance ratio and that asking for a second eigenvector does not significantly increase the number of outer or inner iterations. Finally, the reduction in the number of outer iterations achieved by initializing the IRAM with power iteration results in the lowest overall amount of computational work, measured by the number of inner iterations.

Keep in mind that it is the eigenvector and not the eigenvalue that we used to determine convergence of

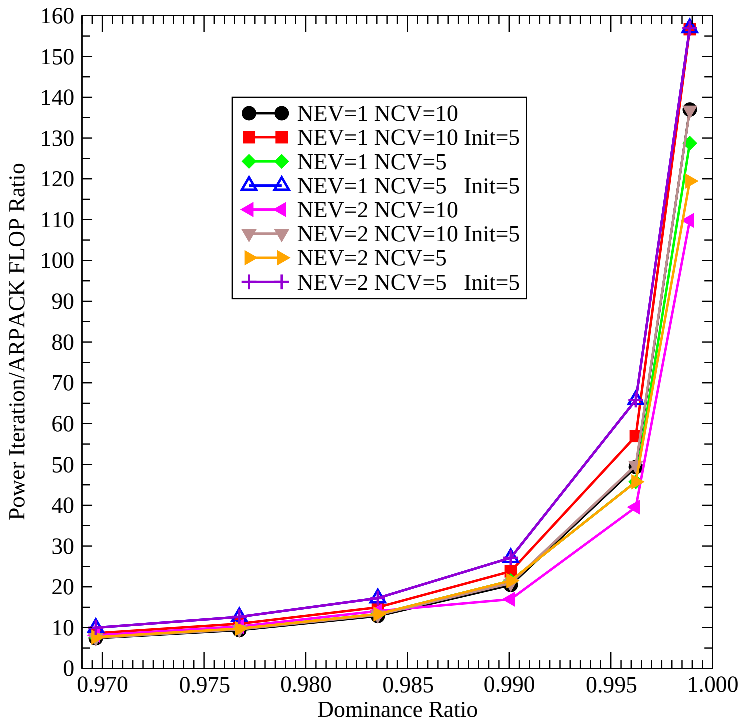

Figure 2. Relative computational expense of power iteration compared to ARPACK.

both the power iteration algorithm and the IRAM. The last line of entries in Table III shows the number of outer power iterations needed to converge the eigenvalue to an absolute *error* of 10^{-5}. This shows that the eigenvalue converges much more quickly than does the eigenvector. In the case of a symmetric operator, for instance, it is well known that convergence is quadratic in the dominance ratio for the eigenvalue and linear for the eigenvector [32].

3.2 MOX Fuel Assembly Problem

The second numerical example is for a three dimensional MOX fuel assembly benchmark problem (C5G7 MOX) developed by the Expert Group on 3-D Radiation Transport Benchmarks. The interested reader can consult Appendix C in [33] for detailed specifications of this problem. Briefly, however, the problem consists of four 17×17 pin (with cladding) fuel assemblies containing $U0_2$ and 4.3% MOX, 7.0% MOX and 8.7% MOX fuels in various configurations, surrounded by moderator (water). Guide tubes and fission chambers are also present in the assemblies. The overall dimensions are $64.26\text{cm} \times 64.26\text{cm} \times 214.20\text{cm}$ in the x, y, and z dimensions. Reflective boundary conditions are specified on the $x = 0$, $y = 0$, and $z = 0$ faces. Vacuum boundary conditions are specified on the other three faces. The tetrahedral grid constructed for this problem consists of 954,527 cells with 169,745 vertices. Seven group, transport-corrected, isotropic scattering cross section data are given. The problem consists of nearly 28 million degrees of freedom, a substantial size for a serial implementation. This problem is intended to show the relative merits of the two methods for large, realistic transport problems. The inner, within-group iterations were computed using BiCGStab.

Table III. Number of outer iterations for the cylindrical mesh problem.

Method			Dominance Ratio, δ					
`nev`	`ncv`	`init`	0.970	0.977	0.984	0.990	0.996	0.999
1	10	0	20	20	20	20	20	20
1	10	5	15	15	15	15	15	15
1	5	0	20	20	20	20	23	23
1	5	5	14	14	14	14	14	17
2	10	0	18	18	18	25	26	26
2	10	5	18	18	18	18	18	18
2	5	0	20	20	20	20	23	25
2	5	5	14	14	14	14	14	17
Power Iteration			468	602	833	1,329	3,170	8,824
k only			229	287	388	586	1,206	1,739

Table IV. Total inner iterations for the cylindrical mesh problem.

Method			Dominance Ratio, δ					
`nev`	`ncv`	`init`	0.970	0.977	0.984	0.990	0.996	0.999
1	10	0	460	460	458	457	447	447
1	10	5	392	390	389	389	385	390
1	5	0	440	436	438	428	472	463
1	5	5	327	327	327	332	322	372
2	10	0	424	424	422	544	550	549
2	10	5	446	441	440	438	436	443
2	5	0	440	436	438	428	472	497
2	5	5	327	327	327	332	322	373
Power Iteration			2,503	3,124	4,218	6,691	15,892	44,156
k only			1,308	1,549	1,993	2,976	6,077	8,747

3.2.1 MOX Fuel Assembly Problem without Upscatter

In the first set of results, the upscatter portion of the scattering matrix is set to zero. The dominance ratio for this problem was approximately 0.985 so that the power iteration method converged slowly, taking 6,144 inner iterations over the course of 860 outer iterations. We saw in the first set of results that the total number of inner iterations gives a very good idea of the relative performance of the methods. Thus the total number of outer and inner iterations for the two methods are shown in Table V. The power iteration calculation took about 150 hours of wall clock time on a dedicated 1000 MHz 64-bit Compaq Alpha EV6.8CB (21264C) CPU with an internal Alpha FPU and 8 GB RAM. For comparison, the ARPACK calculations ran from about 23 to 34 hours of wall clock time.

Table V. Iteration counts for the three dimensional MOX fuel assembly benchmark problem without upscatter.

Method			Iterations	
`nev`	`ncv`	`init`	Outer	Inner
1	10	0	25	1,109
1	10	5	25	1,050
1	5	0	38	1,404
1	5	5	32	1,132
2	10	0	34	1,396
2	10	5	33	1,303
2	5	0	44	1,578
2	5	5	38	1,299
Power Iteration			860	6,144
k only			296	2,196

As in the previous set of results, we see that initializing the IRAM with power iteration (one inner iteration per power iteration initialization step for each group) can make a difference. We found that taking more than five initialization iterations for the IRAM did not change the number of outer iterations and had only a small affect on the number of inner iterations for this problem. Overall, the IRAM is roughly five times faster than power iteration for this problem. Accelerating the power iteration method with a Chebyshev, SOR or some other acceleration technique, could conceivably affect this conclusion. Note that we tried simple SOR acceleration on this problem but could not find a robust sequence of relaxation parameters that would accelerate convergence without causing the iteration to diverge. Once again, convergence of the eigenvalue was much faster than the eigenvector for power iteration.

3.2.2 MOX Fuel Assembly Problem without Upscatter

The next set of results gives some indication as to how the IRAM performance compares to power iteration for a large problem with upscatter, shown in Table VI. The dominance ratio for this problem is very close to the same problem without upscatter, about 0.987. The inner iterations for ARPACK were computed using the BiCGStab Krylov subspace iterative method preconditioned with DSA. The iterative convergence tolerance for the upscatter iterations is inversely proportional to the IRAM residual and the inner iteration tolerance for the inners at each outer is proportional to the upscatter iteration residual for the Gauss-Seidel iteration or inversely proportional for GMRES iterations. For comparison with the previous results, the

Table VI. Iteration counts for the three dimensional MOX fuel assembly benchmark problem with upscatter.

Method			Iterations		
`nev`	`ncv`	`init`	Outer	Inner	Upscatter
1	5	5	32	6,011	504 [a]
2	10	5	33	6,030	483 [a]
1	5	5	32	6,991	309 [b]
Power Iteration			1,144	8,144	
k only			456	3,321	

[a] Gauss-Seidel Upscatter
[b] GMRES Upscatter

wall clock times for the ARPACK iterations ranged from about 143 to about 157 hours while the power iteration required about 203 hours. Again, this is on a dedicated CPU.

We can make some general observations of a qualitative nature regarding the choice of `nev`, `ncv` and `init`. Convergence of the IRAM is more sensitive to number of eigenvalues requested, `nev`, than it is to the maximum Krylov subspace dimension, `ncv`. We also find that overall computational cost is more sensitive to a good initial starting vector than is the convergence of the outer iteration because of the reduction in the outer iteration count.

4 CONCLUSIONS

We found the Implicitly Restarted Arnoldi Method, as available in the software package ARPACK, to be easy to implement using the inner and outer iteration coding and other available computational machinery in our existing discrete ordinates transport code. Our numerical experiments, although certainly not exhaustive, show that the method is robust and efficient for several difficult representative problems, when compared to the power iteration method that had already been implemented. We did not encounter any difficulties with convergence of the method.

The ARPACK implementation of IRAM has been extended to parallel platforms [34] so the method can be just as easily implemented in existing parallel transport codes as it can in serial codes. The results and performance reported here, and hence our conclusions, could be different on parallel platforms, although we do not have any reason at this time to expect that the performance will be any better or worse in parallel.

The IRAM is of course not limited to the discretized transport equation used in the numerical results we presented. We believe that the IRAM could perform just as well, relative to power iteration, for other types of angular and spatial discretizations. Improvements in efficiency could be obtained if the IRAM were implemented directly into the transport source code in an "in-line" fashion and optimized to use the existing data structures and storage that has already been allocated. Perhaps existing acceleration methods for multigroup fission problems, like Chebyshev or coarse-mesh rebalance, could be used to further improve the overall efficiency of the IRAM-based k-eigenvalue calculations. Other types of iterative eigenvalue methods, such as shifted-inverse iteration, might also be combined with the IRAM to improve overall efficiency. We have shown one example already, having used power iteration to initialize the IRAM and speed up convergence.

A significant drawback of the method as it is currently implemented is the inability to efficiently treat problems with energy upscatter. We hope to address this in the future.

Acknowledgments

The authors would like to thank Michele Benzi of Emory University for his very helpful discussions on this and related topics. The work of the first four authors was performed under the auspices of the United States Department of Energy and the National Nuclear Security Agency at the Los Alamos National Laboratory. The work of the fourth author was performed at the Sandia National Laboratories, a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

REFERENCES

[1] E. E. Lewis and W. F. Miller, **Computational Methods of Neutron Transport**. Wiley & Sons: New York (1984).

[2] D. C. Sorensen, "Implicit Application of Polynomial Filters in a k–step Arnoldi Method," *SIAM J. Matrix Anal. Appl.*, **13**, n. 1, pp. 357–385 (1992).

[3] G. Verdú, R. Miró, D. Ginestar, and V. Vidal, "The Implicit Restarted Arnoldi Method, An Efficient Alternative to Solve the Neutron Diffusion Equation," *Annals of Nuclear Energy*, **26**, pp. 579–593 (2002).

[4] G. I. Marchuk and V. I. Lebedev, **Numerical Method in the Theory of Neutron Transport**. Harwood Academic Publishers: Chur, Switzerland, Revised Second Edition (1986). Translated and edited by O. Germogenova.

[5] V. Vidal, G. Verdú, D. Ginestar, and J. L. Munõz-Cobo, "Variational Acceleration for Subspace Iteration Method. Application to Nuclear Power Reactors," *International Journal for Numerical Methods in Engineering*, **41**, pp. 391–407 (1998).

[6] E. J. Allen and R. M. Berry, “The Inverse Power Method for Calculation of Multiplication Factors,” *Annals of Nuclear Energy*, **29**, pp. 929–935 (2002).

[7] R. B. Lehoucq, D. C. Sorensen, and C. Yang, **ARPACK User’s Guide**. SIAM: Philadelphia (1998).

[8] T. A. Wareing, J. M. McGhee, and J. E. Morel, “ATTILA: A Three–Dimensional, Unstructured Tetrahedral Mesh Discrete Ordinates Transport Code,” *Transactions of the American Nuclear Society*, **75**, pp. 146–147 (1996).

[9] M. L. Adams, “Discontinuous Finite Element Methods in Thick Diffusive Problems,” *Nucl. Sci. and Engr.*, **137**, pp. 298–333 (2001).

[10] J. E. Morel, “A Hybrid Collocation–Galerkin–S_n Method for Solving the Boltzmann Transport Equation,” *Nuclear Science and Engineering*, **101**, 72–87 (1989).

[11] G. W. Stewart, **Matrix Algorithms, Volume II: Eigensytems**. SIAM: Philadelphia (2001).

[12] R. S. Modak, D. C. Sahni, and S. D. Paranjape, “Evaluation of Higher K–Eigenvalues of the Neutron Transport Equation by S_n–Method,” *Annals of Nuclear Energy*, **22**, pp. 359–366 (1995).

[13] R. S. Varga, **Matrix Iterative Analysis**. Prentice–Hall: Englewood Cliffs, New Jersey (1966).

[14] L. A. Hageman and D. M. Young, **Applied Iterative Methods**. Academic Press: New York (1981).

[15] T. A. Manteuffel, “The Tchebyshev Iteration for Nonsymmetric Linear Systems,” *Numerische Mathematik*, **28**, pp. 307–327 (1977).

[16] B. Fischer and R. Freund, “Chebyshev Polynomials Are Not Always Optimal,” *J. Approximation Theory*, **65**, pp. 261–272 (1991).

[17] O. Axelsson, **Iterative Solution Methods**. Cambridge University Press: Cambridge, England (1996).

[18] G. H. Golub and C. F. Van Loan, **Matrix Computations**. Johns Hopkins University Press, Second Edition (1989).

[19] D. C. Sorensen, “Numerical Methods for Large Eigenvalue Problems,” *Acta Numerica*, **11**, n. 00, pp. 519–584 (2002).

[20] R. B. Lehoucq, “Implicitly Restarted Arnoldi Methods and Subspace Iteration,” *SIAM J. Matrix Anal. Appl.*, **23**, n. 2, pp. 551–562 (2001).

[21] M. H. Gutknecht and S. Röllin, “The Chebyshev Iteration Revisited,” *Parallel Computing*, **28**, pp. 263–283 (2002).

[22] B. T. Adams and J. E. Morel, “A Two–Grid Acceleration Scheme for the Multigroup S_n Equations with Neutron Upscattering,” *Nuclear Science and Engineering*, **115**, pp. 253–264 (1993).

[23] A. Bouras and V. Frayssé, “A Relaxation Strategy for Inexact Matrix–Vector Products for Krylov Methods,” CERFACS TR/PA/00/15, European Centre for Research and Advanced Training in Scientific Computation, Toulouse, France (2000). Submitted to *SIAM J. Matrix Anal. Appl.*.

[24] A. Bouras, V. Frayssé, and L. Giraud, “A Relaxation Strategy for Inner–Outer Linear Solvers in Domain Decomposition Methods,” CERFACS TR/PA/00/17, European Centre for Research and Advanced Training in Scientific Computation (2000).

[25] A. Bouras and V. Frayssé, “A Relaxation Strategy for the Arnoldi Method in Eigenproblems,” CERFACS TR/PA/00/16, European Centre for Research and Advanced Training in Scientific Computation, Toulouse, France (2000).

[26] V. Simoncini and D. Szyld, “Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing,” Research Report 02–4–12, Temple University, Apr. (2002).

[27] T. A. Wareing, “New Diffusion–Synthetic Accelerations Methods for the S_N Equations with Corner Balance Spatial Differencing,” in **Joint International Conference on Mathematical Methods and Supercomputing in Nuclear Applications**, 19–23 April, Vol. 2, Karlsruhe, Germany, p. 500 (1993).

[28] T. A. Wareing, E. W. Larsen, and M. L. Adams, “Diffusion Accelerated Discontinuous Finite Element Schemes for the Sn Equations in Slab and X–Y Geometries,” in **Proc. International Topical Meeting on Advances in Mathematics, Computations, Reactor Physics**, 28 April – 2 May, Vol. 3, Pittsburgh, Pennsylvania, pp. 11.1 2–1 (1991).

[29] J. S. Warsa, T. A. Wareing, and J. E. Morel, “Fully Consistent Diffusion Synthetic Acceleration of Linear Discontinuous Transport Discretizations on Three–Dimensional Unstructured Meshes,” *Nuclear Science and Engineering*, **141**, pp. 236–251 (2002).

[30] G. E. Hansen and W. H. Roach, “Six and Sixteen Group Cross Sections for Fast and Intermediate Critical Assemblies,” LAMS–2543, Los Alamos Scientific Laboratory, Dec. (1961).

[31] G. I. Bell, J. J. Devaney, G. E. Hansen, C. B. Mills, and W. H. Roach, “Los Alamos Group–Averaged Cross Sections,” LAMS–2941, Los Alamos Scientific Laboratory, Sept. (1963).

[32] L. N. Trefethen and I. D. Bau, **Numerical Linear Algebra**. SIAM: Philadelphia (1997).

[33] E. E. Lewis (Chairman), “Expert Group on 3–D Radiation Transport Benchmarks: Summary of Meeting,” Report NEA/NSC/DOC(2001)17, OECD/NEA, Sept. (2001).

[34] K. J. Maschhoff and D. C. Sorensen, “P_ARPACK: An Efficient Portable Large Scale Eigenvalue Package for Distributed Memory Parallel Architectures,” in **Applied Parallel Computing in Industrial Problems and Optimization** (J. Wasniewski, J. Dongarra, K. Madsen, and D. Olesen, Eds.), Vol. 1184 of *Lecture Notes in Computer Science*, Berlin (1996), Springer–Verlag.