KRYLOV SUBSPACE ITERATIONS FOR DETERMINISTIC k–EIGENVALUE CALCULATIONS

James S. Warsa, Todd A. Wareing, Jim E. Morel, John M. McGhee
Transport Methods Group
Los Alamos National Laboratory
Los Alamos, NM 87545–0001

and

Richard B. Lehoucq
Computational Mathematics and Algorithms Department
Sandia National Laboratories
Albuquerque, NM 87185–1110

Send proofs to:

James S. Warsa
Los Alamos National Laboratory
CCS–4, MS D409
Los Alamos, NM 87545–0001

E-mail: *warsa@lanl.gov*
FAX: 505-665-5538

Number of pages: 21
Number of figures: 4
Number of tables: 4

KRYLOV SUBSPACE ITERATIONS FOR DETERMINISTIC k–EIGENVALUE CALCULATIONS

James S. Warsa, Todd A. Wareing, Jim E. Morel, John M. McGhee
Transport Methods Group
Los Alamos National Laboratory
Los Alamos, NM 87545–0001

and

Richard B. Lehoucq
Computational Mathematics and Algorithms Department
Sandia National Laboratories
Albuquerque, NM 87185–1110

Abstract

In this paper we present an efficient alternative to the power iteration method that is typically used to calculate k–eigenvalues, and corresponding eigenvectors, with deterministic transport codes. The alternative approach is based on a Krylov subspace iterative projection method called the Implicitly Restarted Arnoldi Method (IRAM). Only modest changes are needed to incorporate the IRAM into existing power iteration coding and the IRAM can be used to calculate several additional higher order eigenvectors with little extra computational cost. Numerical results for three dimensional S_N transport on unstructured tetrahedral meshes show that the k–eigenvalue calculations can be computed efficiently with the IRAM. These numerical results are compared to results computed with traditional power iteration calculations.

1 INTRODUCTION

Many deterministic transport codes use the power iteration method to calculate the dominant k–eigenvalue(s) and corresponding eigenfunction(s) for criticality problems. The convergence rate of power iteration is determined by the dominance ratio which is defined as the ratio of the second largest eigenvalue to the maximum eigenvalue.[1] Power iteration converges very slowly for problems with large dominance ratios and in certain cases power iteration may not converge at all.[2] Problems of practical interest can often have dominance ratios large enough to make calculations of the maximum k–eigenvalue difficult or even intractable with the power iteration method.

Alternatives to power iteration 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.[3] Another example is found in a paper that recently appeared on inverse iteration for k–eigenvalue calculations in one dimensional slab geometries.[4]

Classical iterative methods based on power iteration remain the method of choice even in the modern deterministic codes that are currently in use. We suggest that for some situations – problems with high dominance ratios, for example – modern Krylov subspace iterative methods could offer an improvement in performance over both the classical power iteration based approaches and other alternatives. This concept is a logical extension of recent work using DSA–preconditioned Krylov iterative methods to replace traditional DSA–accelerated source iteration.[5,6] The implementation of a Krylov subspace method for k–eigenvalue calculations is very similar to the implementation of the Krylov methods described in that work.

Our purpose in this paper is to describe the application of a Krylov subspace projection method called the Implicitly Restarted Arnoldi Method (IRAM)[7] to the problem of k–eigenvalue calculations with S_N transport codes. Using the freely available ARPACK software,[8] we easily incorporated the IRAM into a deterministic transport code by "wrapping" the IRAM solver around the existing power iteration coding. The method can be applied equally well to both symmetric and nonsymmetric problems, so that the implementation does not need to be altered in order to ensure a symmetric transport operator. We have found that a Krylov subspace dimension of order five or ten seems to be optimum for our applications. We have seen that the extra computational work and storage associated with the IRAM implementation is acceptable given the improvement in performance. We have found that with little additional computational expense, and no further code modification, the eigenvectors corresponding to several of the largest eigenvalues can be calculated efficiently.

To our knowledge the work reported in this paper represents the first application of a Krylov subspace method for k–eigenvalue calculations for large scale, three dimensional transport applications. Note, however, that a paper recently appeared describing the application of the IRAM to k–eigenvalue calculations based on two–group diffusion theory.[9] That work arrived at conclusions similar to our own.

The remainder of this paper is organized as follows. Section 2 presents some notation and a review of power iteration, followed by a discussion of the IRAM iterative eigenvalue calculation and how it is implemented in a deterministic S_N code. Section 3 presents the results of numerical calculations on three dimensional unstructured tetrahedral meshes. These results compare the performance of the IRAM and the power iteration method. Section 4 ends the paper with some concluding remarks.

2 k–EIGENVALUE CALCULATIONS WITH DETERMINISTIC S_N TRANSPORT

In this section we will start by describing power iteration for the multigroup in energy and linear discontinuous finite element method (DFEM) spatial discretization of the S_N equations on tetrahedral meshes that we consider in this paper. This is followed by a discussion of the power iteration method and then a brief outline of the Krylov subspace iterative eigenvalue computational method is given. The section finishes with a discussion of special considerations and issues associated with implementing the Krylov iterative method in the S_N transport code AttilaV2.[10]

2.1 Discrete S_N Equations on Tetrahedral Meshes

We use standard notation[1] and we will assume CGS units. 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

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

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

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

Boundary conditions are specified at the surface $\boldsymbol{r}_b$ with outward unit normal $\hat{n}$ by

$$\psi_m(\boldsymbol{r}_b) = \Gamma(\hat{\Omega}_m) \quad \text{for} \quad (\hat{\Omega}_m \cdot \hat{n}) < 0. \tag{1c}$$

We will consider either vacuum boundary conditions, where $\Gamma(\hat{\Omega}_m) = 0$, or specular reflection boundary conditions. In the latter case, the reflected image of $\hat{\Omega}_m$, denoted by $\hat{\Omega}_{m'}$, is defined by

$$\hat{\Omega}_{m'} = \hat{\Omega}_m - 2\,\hat{n}\left(\hat{\Omega}_m \cdot \hat{n}\right). \tag{1d}$$

In the case of reflective boundary conditions, we set $\Gamma(\hat{\Omega}_m) = \psi_{m'}(\boldsymbol{r}_b)$, where Eq. 1d determines m'.

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

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

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

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

This gives four equations for the four unknowns $\psi_{m,j,k}$ on every cell s for every angle $\hat{\Omega}_m$. We have assumed that the cross sections and other parameters are constant in a cell. Before the integrations in Eq. 3a can be carried out, however, we have to define the fluxes on the cell interfaces in order to make the discontinuous approximation. In particular, for a cell k with face j whose outward normal is $\hat{n}_j$, the "boundary fluxes" appearing in the first term on the left side of Eq. 3a are defined as

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

where l is the cell that shares face j with cell s. The subscript $i(j)$ denotes three vertices i on a face j of a given cell. The discrete boundary conditions are vacuum, $\Gamma(\hat{\Omega}_m) = 0$, or $\Gamma(\hat{\Omega}_m) = \psi_{g,m',i(j),s}$ for reflective boundary conditions, where m' satisfies Eq. 1d for $\hat{\Omega}_m$ and $\hat{n} = \hat{n}_j$. Reflection is implemented only for boundary faces aligned parallel to the x, y or z coordinate axes such that the standard quadrature sets we use contain the reflected angles that satisfy Eq. 1d.

2.2 Power Iteration

We now write the discretized S_N equations, Eqs. 3, in operator notation

$$\boldsymbol{L}\psi = \boldsymbol{MSD}\psi + \frac{1}{k}\boldsymbol{MFD}\psi. \tag{4}$$

Detailed meanings of the operators can be deduced by comparing Eq. 4 with Eqs. 3. Briefly, $\boldsymbol{L}$ is the transport operator, $\boldsymbol{S}$ is the scattering 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 Ref. 11. They convert a vector of scalar flux moments to angular fluxes and vice versa. It is not necessary to assume that they are inverses of one another. We will ignore boundary conditions for now.

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

$$\phi^{\ell+1} = \frac{1}{k_\ell} \boldsymbol{D} \left(\boldsymbol{L} - \boldsymbol{MSD} \right)^{-1} \boldsymbol{F} \phi^\ell, \quad \text{where} \quad \phi^\ell = \boldsymbol{D} \psi^\ell. \tag{5}$$

The vector ϕ represents all the scalar flux moments contained in Eqs. 1.

Power iteration computes the largest eigenvalue k by iterating Eq. 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. In many implementations the norm takes into account only the isotropic scalar flux moment component of ϕ, ϕ_0^0, although there is no mathematical justification for this. Furthermore, the eigenvalue estimate is often updated based on the total fission rate in the problem.[1] That is, the norm in Eq. 6 is taken to be

$$\|\phi\| \equiv \|\phi\|_F = \sum_{g=1}^{G} \sum_s \nu\sigma_{f,g,s} \phi_{0,g,s}^0, \tag{7}$$

where the sum over s is for all cells in the problem.

An alternative to Eq. 6 is to use the Rayleigh quotient to update the eigenvalue estimate as follows:

$$k_{\ell+1} = \frac{(\boldsymbol{A}\phi^\ell, \phi^\ell)}{(\phi^\ell, \phi^\ell)} \tag{8}$$

$$= k_\ell \frac{(\phi^{\ell+1}, \phi^\ell)}{(\phi^\ell, \phi^\ell)}, \tag{9}$$

where $(\cdot, \cdot)$ is a discrete inner product over all cells, and $\boldsymbol{A} = \boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{F}$ with $\boldsymbol{H} = \boldsymbol{L} - \boldsymbol{MSD}$. We have observed that using the Rayleigh quotient rather than Eq. 6 for the eigenvalue estimate can sometimes improve the efficiency of the power iteration method by providing a better estimate of the eigenvalue earlier in the iterative process.[12] It appears that the Rayleigh quotient estimate can, in some cases, lead to faster convergence of the inner iterations and better overall efficiency.

It can be shown that power iteration will converge to the largest eigenvalue in magnitude k_1 (the dominant eigenvalue) and that the convergence of this iteration is determined by the dominance ratio $\delta = k_2/k_1$ where $k_2 \leq k_1$ is the next largest eigenvalue in magnitude.[12] The closer this ratio is to one the more slowly power

iteration converges. It is possible that power iteration will not converge. This can happen, for instance, if the eigenvalue with largest magnitude is complex and the (nonsymmetric) operator and initial guess ϕ_0^0 are real. It has been shown in the monoenergetic case that the k–eigenvalues of the S_N approximation to the transport operator are are always real, even for anisotropic scattering.[13] However, we do not know if this is also true for the multigroup approximation so we cannot state whether or not there exist multigroup problems for which the power iteration method might not converge.

2.3 Acceleration of Power Iteration

Power iteration works with two vectors, or iterates, $\phi^{\ell+1}$ and ϕ^{ℓ}. This corresponds to a Krylov subspace dimension of one. Simple power iteration can be improved by keeping additional vectors and making use of information from previous iterations. Chebyshev iteration, successive over–relaxation (SOR), and other classical iterative techniques are commonly used to accelerate k–eigenvalue power iterations in this way.[14,15] Such methods update the iteration using a linear combination of several of the previous iterates. This can accelerate convergence at the cost of additional storage. Under certain circumstances it is possible to use knowledge of the spectrum of the operator to find the optimal linear combination that minimizes the convergence rate. However, in practice the spectrum is not known in advance and has to be estimated. The result is a less than optimal method. Furthermore, when these acceleration methods are applied to more general transport problems such as those that include anisotropic scattering, the situation becomes more complicated because the operators become nonsymmetric. A general implementation must take into account the fact that the spectrum could extend into the complex plane and methods for computing optimal estimates of the required acceleration parameters becomes significantly more difficult.[16–18] In practice, this possibility is therefore often ignored and acceleration methods are implemented 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. The uncertain nature of such algorithms make it difficult to predict when or if 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.[15]

2.4 Implicitly Restarted Arnoldi Method

Equation 5 represents power iteration for the standard eigenvalue problem

$$\boldsymbol{D}\boldsymbol{H}^{-1}\boldsymbol{F}\phi = k\phi, \tag{10}$$

with $\boldsymbol{H} = \boldsymbol{L} - \boldsymbol{MSD}$ as in Section 2.2. Our approach is to compute the largest few eigenvalues of this standard eigenvalue problem using the IRAM implementation found in the ARPACK software package.[8] The IRAM is a recently developed method for computing large scale eigenvalue problems. The ARPACK implementation is best suited for applications whose matrices are either sparse or not explicitly available. This is because only the "action" of a matrix on a vector, supplied by the solver, is calculated at every IRAM iteration. ARPACK uses a reverse communication mechanism that frees the user from any particular data structure formats. This greatly simplifies implementation of the method in an existing transport code that uses the power iteration method because the IRAM solver can simply be "wrapped" around the power iteration coding.

We will now briefly describe the IRAM. First, consider the power iteration method, which is a simple way to compute the dominant eigenvalue and corresponding eigenvector – the fundamental mode – of $\boldsymbol{A}x = \lambda x$. (in our case $\boldsymbol{A} = \boldsymbol{DH}^{-1}\boldsymbol{F}$ and $\lambda = k$). Only two vectors of storage are necessary and only the action of the operator $\boldsymbol{A}$ on a vector is required at every iteration. This can viewed as a Krylov subspace iteration with a subspace dimension of one. Now, imagine applying power iteration to several vectors, say p of them, simultaneously. The result would be that the p vectors would all converge to the dominant eigenvector. However, if the vectors were orthogonalized and normalized at every iteration they would instead converge to the p eigenvectors that correspond to the p eigenvalues largest in magnitude. This approach is called subspace iteration.

A natural question to ask is whether it is possible to store and combine several vectors in an attempt to take greater advantage of work that has already been performed. Power iteration operates on only the most recently computed vector, destroying any additional eigenvector information that might have been extracted from the previous iterations. The acceleration techniques mentioned previously in Section 2.3, do use information from the previous iterates, but often in a less than optimal way. However, suppose a starting vector for power iteration was expanded in terms of the eigenvectors of the operator. Then the expansion coefficients of the vector would evolve in a specific and identifiable way through the repeated application of the operator. A linear combination of the vector sequence generated during the course of the iterations, together with knowledge of this pattern of evolution, might then be constructed in such a way as to extract additional useful eigenvector information from the vector sequence that is optimal in a certain sense.

The Arnoldi method, which forms the basis of the IRAM, combines these two ideas. The Krylov subspace of the operator $\boldsymbol{A}$ is constructed from the vector sequence generated by power iteration. Approximate eigenvectors are computed in the subspace by satisfying a Galerkin orthogonality condition. The approximate eigenvector–eigenvalue pair (eigenpair) is termed a "Ritz pair". The Ritz vector is a projection of the eigenvector onto the smaller Krylov subspace. If the component orthogonal to the subspace is small, then the Ritz pair will be a good approximation to the eigenpair of a nearby problem.

Ritz pairs satisfying the Galerkin orthogonality condition can be found easily through the Arnoldi factorization. The Arnoldi factorization efficiently computes the orthogonal projections of $\boldsymbol{A}$ onto the Krylov subspace and follows from the Galerkin orthogonality condition. The factorization leads to a relatively small projected (upper Hessenberg) matrix from which the approximate Ritz pairs can be calculated easily. The desired eigenvalues are more closely approximated as the size of the subspace increases. In exact arithmetic they will equal the exact eigenvalues after a sufficient number of iterations. An additional useful feature is that an estimate of the (backward) error in the approximation, called the Ritz estimate, is available as a consequence of the Arnoldi factorization.

The cost of computing the Arnoldi factorization increases with the size of the Krylov subspace, a consequence of maintaining the orthogonality of the Arnoldi vectors. These 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. The IRAM gets its name from an implicit restarting technique that is based on the implicitly shifted QR factorization method.[7,19] The implicit restart approach maintains full orthogonality of the Krylov subspace, making the algorithm numerically stable and robust. Restarting in this way also allows the Ritz pairs to be "filtered" so as to produce eigenvectors that correspond to eigenvalues with the desired properties, such as those with the largest magnitude or largest real part, for example. Theoretical details of the IRAM can be found in Ref. 7, 20, 21.

The desired number of extremal eigenvalues that are largest or smallest in magnitude, or those with the largest or smallest real or imaginary parts, is supplied to the ARPACK software. The other information that is needed is the maximum dimension of the Krylov subspace and the convergence criteria. Some overhead, in terms of both memory requirements and computation per iteration, depends on the size of the Krylov subspace and the number of requested eigenvalues. Currently we are treating the IRAM as a black box eigenvalue solver and so the existing data structures in the transport code are repeated and further additional memory overhead is needed by the ARPACK implementation. However, the additional memory requirements are compensated by the improved convergence rates. We also found that the computational overhead of the ARPACK routines is modest and is comparable to the overhead of the power iteration method.

The ARPACK implementation is completely general in that it is not restricted to self–adjoint problems although the package does provide facilities that take advantage of symmetry leading to greater efficiency. The spatially discretized discrete ordinates transport equation is self–adjoint only under certain circumstances,[22,23] so we chose to use the nonsymmetric ARPACK routines because they can be applied to any transport problem without having to modify the transport code to ensure the operator is symmetric. This enables us to make a fair comparison with power iteration, which also can be applied to any problem –

symmetric or nonsymmetric.

2.5 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 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. Each group, or block, in the sequence is inverted approximately with an inner iteration. These solutions can be computed either with traditional source iteration or with a Krylov iterative method as described in Ref. 5. In any case, if upscatter is present the operator $\boldsymbol{S}$ is no longer block lower triangular and the operator $\boldsymbol{H}$ has to be inverted iteratively. In our transport code implementation, this inversion is actually "embedded" in the power iterations. This can be made clear if we suppose that a fixed source transport problem with upscatter was being solved, rather than a k–eigenvalue problem. In that case we would compute the approximate inverse of $\boldsymbol{H}$ with some iterative method given a specified energy dependent source. In our implementation we use a block Gauss–Seidel iteration over energy groups. Imagine that we replaced that fixed source with the updated fission source. Then we could, in a sense, view power iteration as block Gauss–Seidel iteration with the addition of some extra computations for the updated eigenvalue and fission source at every iteration.

The inversion of $\boldsymbol{H}$ is therefore computed implicitly during the course of the power iterations. At this time we are still investigating how we might do something similar with the IRAM. If we want to compute the eigenvalues for problems involving upscatter with ARPACK, we have to compute $\boldsymbol{H}^{-1}$ iteratively at every IRAM iteration. We can do that either with a Krylov method or with the block Gauss–Seidel iteration that is part of the power iteration coding. This approach would probably be less inefficient compared to the existing power iteration method, unless a very efficient acceleration or preconditioning technique could be developed. One possibility would be to precondition a Krylov iterative method or accelerate the block Gauss–Seidel iteration with the upscatter acceleration method of Ref. 24. In this paper, we will not consider problems with upscatter although we plan to address this issue in the future.

Another important aspect of power iteration that has a significant impact on its efficiency is that the current 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 Eq. 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 aspect of the power iteration algorithm significantly reduces the overall computational expense.

While similar good initial guesses are not available for subsequent inner iterations in the IRAM itera-

tions, there is an effective approach for relaxing the inner iteration convergence tolerance during the course of the IRAM iterations to improve efficiency. We called this approach the BFG (Bouras, Frayssé and Giraud) strategy when we originally used it for Krylov iterations preconditioned by an inner DSA conjugate gradient iteration.[25] The authors of Refs. 26, 27, and 28 observed that that Krylov subspaces are built in sequence, with every subsequent vector depending on the previous vectors. Then, loosely speaking, if the action of the operator is computed iteratively, which can alternatively be looked upon as an approximate or "inexact" matrix–vector product, it is important to compute the vector operations (inner iterations) to a strict convergence tolerance in the early outer iterations. The convergence criteria can then be relaxed as the outer iteration proceeds. By choosing the inner tolerance to be inversely proportional to the outer residual the computational work in the inner iterations is reduced without affecting either the outer convergence rate or the solution. If we wish to increase the inner convergence tolerance more slowly we can take it to be proportional to some inverse fractional power of the outer residual as well, although we have not found that to be necessary. However, some very recent work has uncovered a theoretical justification for such an approach, for both linear systems solution and eigenproblems.[29] In that work, a constant of proportionality for the inverse relationship of the inner iteration tolerance and the outer residual is derived. A proportionality constant based on theoretical arguments is compared to an ad hoc approach of simply setting it to unity and in many cases, but not all, there was only a little difference between the two. We selected the constant of proportionality to be 0.1. In the problems we have experimented with to date, including solutions to linear systems as well as the eigenvalue problems discussed in this paper, we have not encountered any difficulties with this strategy.

In any case, the inner iterations are usually accelerated or preconditioned with diffusion synthetic acceleration (DSA), although in some cases it may be more efficient to compute the inner iterations without DSA, particularly if inner Krylov iterations are used. Boundary conditions are implemented naturally in terms of the angular fluxes during the course of the inner iterations. We will find that there are some problems where, because of the efficiency of the power iteration implementation, we can use a few power iterations to initialize the IRAM and improve its convergence.

Finally, note that the IRAM is not the only existing method that can be used to improve upon power iteration. Traditional techniques used to accelerate power iteration improve convergence by working with more than one vector but they require knowledge of the spectrum of the operator. As mentioned previously, in practice this knowledge is not available a priori and has to estimated during the iterative process. The resulting linear combination of previous iterates is therefore going to be less than optimal. The potential for nonsymmetry, as is the case for anisotropic scattering, further complicates matters by bringing the eigenvalue distribution into the complex plane, and must be accounted for in a general implementation. Indeed, a Chebyshev or successive over-relaxation iteration can be viewed as a Krylov subspace iteration of

a (small) dimension that is equal to the number of working vectors.[30] In contrast, the IRAM is a projection method that selects the best linear combination from the Krylov subspace automatically without any a priori information.

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 discussed in Section 2.3, we only consider unaccelerated power iteration for comparison.

The question of how to measure convergence is a significant issue for iterative solution methods. The ARPACK solvers use the backward error,

$$\|\boldsymbol{A}x - \theta x\|_2 \leq |\theta|\epsilon \tag{11}$$

to determine convergence, where x and θ are the current approximate eigenpair, and the tolerance ϵ is specified. 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 factorization.[8] The norm of the residual at step ℓ is

$$\begin{aligned} \|r^\ell\|_2 &= \|\boldsymbol{A}\phi^\ell - k_\ell\phi^\ell\|_2 \\ &= \|k_\ell\phi^{\ell+1} - k_\ell\phi^\ell\|_2 \\ &= |k_\ell|\|\phi^{\ell+1} - \phi^\ell\|_2, \end{aligned} \tag{12}$$

so we can use $\|\phi^{\ell+1} - \phi^\ell\|_2 \leq \epsilon$ as the convergence criteria for the power iteration method. This is in rough agreement with the convergence criteria in ARPACK, Eq. 11. We feel that this makes a comparison of the computational expense as fair as can be expected.

The outer iteration convergence criterion for all the results is $\epsilon = 10^{-4}$. The power iteration results are computed using traditional source iteration accelerated with the partially consistent simplified WLA DSA method[25,31] using the previous outer iteration solutions for the initial guess. The convergence criteria was taken to be $10^{-1}\epsilon$ for the inner iterations and $10^{-2}\epsilon$ for the innermost DSA conjugate gradient (CG) iterations. The inner iteration convergence criterion is proportional to the outer iteration residual $\|r^\ell\|$,

that is, we set $\epsilon = \min(10^{-2}, 10\|r^{\ell}\|)$ at outer iteration ℓ. This is a commonly used approach in Newton–type methods (like power iteration) that reduces the total amount of work, although it may sometimes delay the outer iteration convergence, without affecting the accuracy of the solution. Having a good initial guess available for each within–group inner iteration, together with an effective DSA scheme, minimizes any deleterious effects on convergence.

3.1 Cylindrical Mesh Problem

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

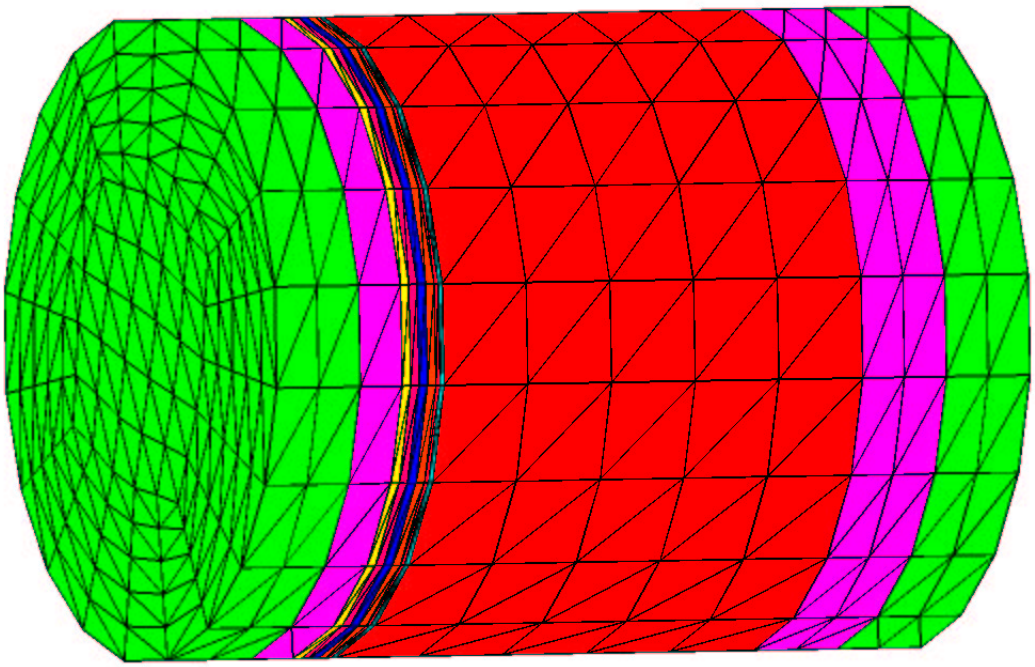

Figure 1: The cylindrical mesh problem. Note the thin disks in the left half of the mesh that are used to alter the symmetry of the problem.

is 3.5 cm in 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 collapsed from Hansen and Roach sixteen group cross section data sets[32,33] were used. The material composition data with material IDs from the cross section library are shown in Table 1. The energy groups are collapsed as follows:

Group	Library Groups
1	1 – 2
2	3 – 4
3	5 – 8
4	9 – 12
5	13 – 16

Table 1: Cross section data for the cylindrical mesh problem. 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

The results for this sequence of problems are shown in Figs. 2, 3, and 4, and in Tables 2 and 3.

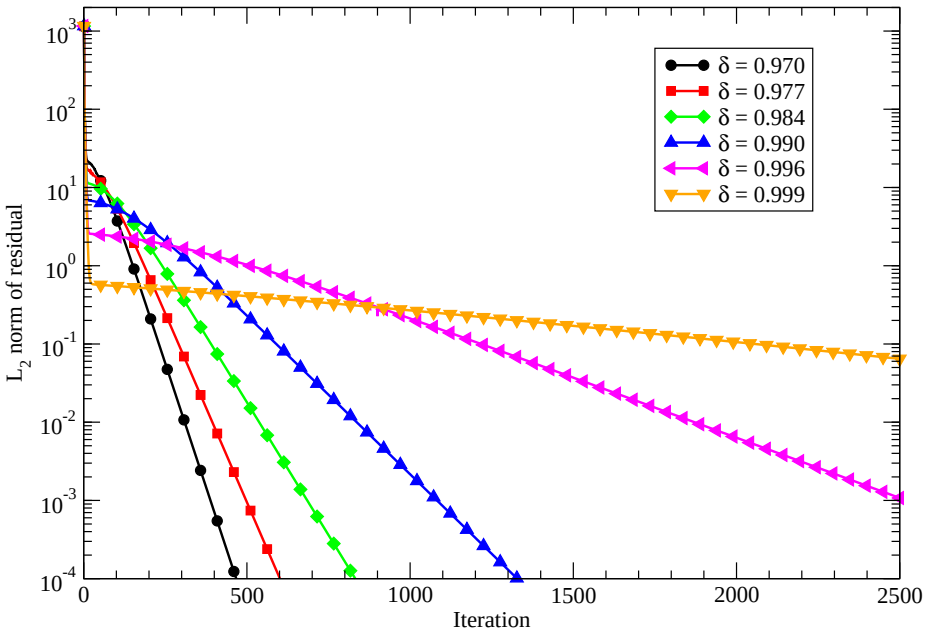

Figure 2: Convergence curves for power iteration illustrating the decreasing convergence rates for increasing dominance ratio, δ.

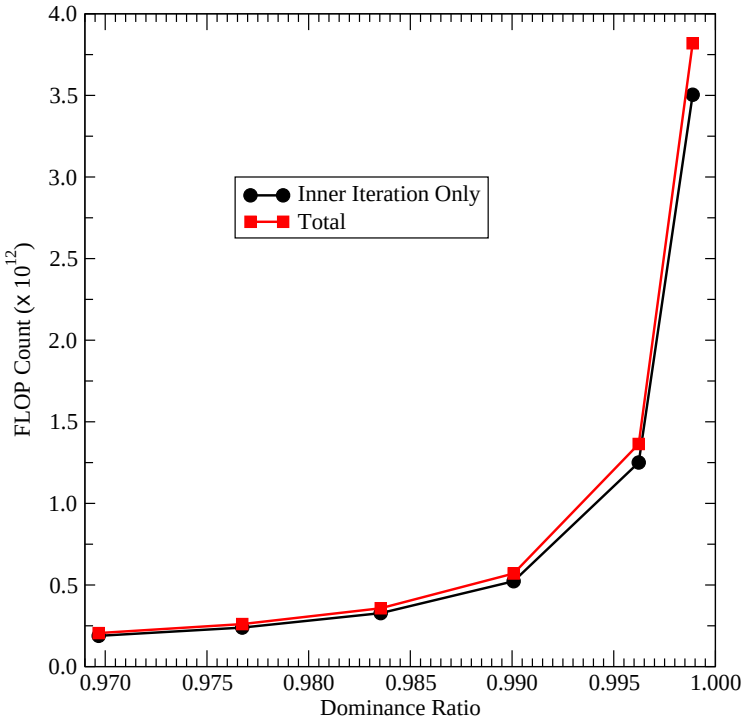

Figure 3: FLOP (floating point operation) counts for power iteration.

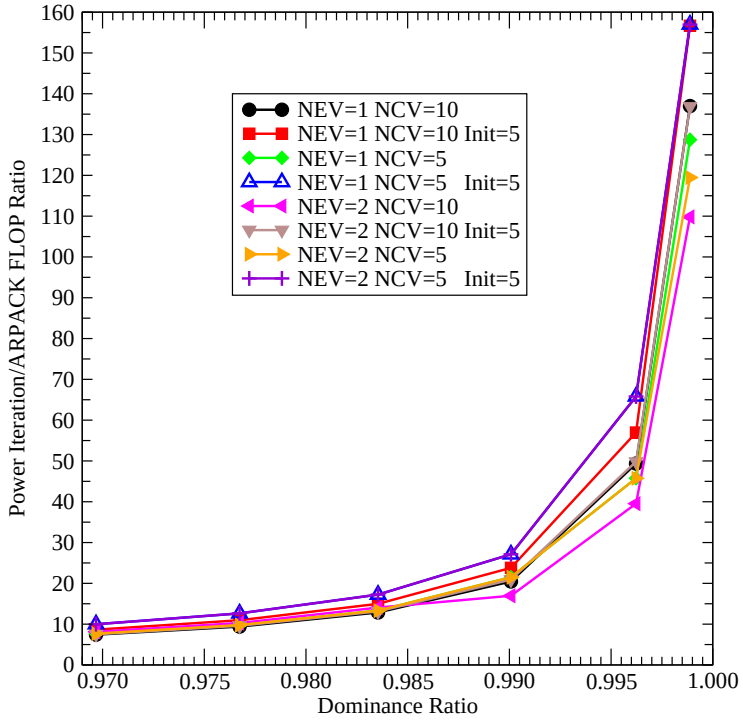

Figure 4: Relative computational expense of ARPACK compared to power iteration.

Figure 2 simply verifies the dependence of power iteration convergence on dominance ratio. Note in the figure that the two problems with largest dominance ratios did not converge to 10^{-4} within 2500 iterations. Figure 3 shows the how the computational expense (floating point operation (FLOP) counts measured on a single dedicated 250 MHz SGI Origin 2000 CPU) corresponds to the slower convergence and shows that most of the computation is devoted to the within–group inner iterations. Thus it is possible to gauge the relative performance of the two methods by simply comparing inner iteration counts. Figure 4 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, `nev` = 0 or 1, the maximum dimension of the Krylov subspace, `ncv` = 5 or 10, and the number of power iterations taken in initializing the IRAM iterations, `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. We did not plot the IRAM convergence curves in Fig. 2 because the convergence rate of the IRAM was so much faster than the power iteration method and would not be distinguishable on the scale of that the figure. However, for completeness, in Tables 2 and 3 we list the number of outer and total number of inner iterations, respectively. The most interesting observation is that convergence of the IRAM is largely insensitive to the dominance ratio. The computational expense depends more on the size of the Krylov subspace than on dominance ratio, as seen in the inner iteration data. We also see 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.

Keep in mind that it is the eigenvector and not the eigenvalue that we used to determine convergence of both the power iteration algorithm and the IRAM. The last line of entries in Table 2 shows the number of outer power iterations needed to converge the eigenvalue to an absolute error of 10^{-5}. This illustrates that the eigenvalue typically converges much more quickly than does the eigenvector for the power iteration method, a property the IRAM does not share. 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.[2]

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 Ref. 34 for detailed specifications of this problem. Briefly, however, the problem consists of four 17×17 pin (with cladding) fuel assemblies containing UO_2 and 4.3% MOX, 7.0% MOX

Table 2: Number of outer iterations for the IRAM on the cylindrical mesh problem. Power iteration results shown for comparison. The "k only" entry shows the number iterations for the power iteration method to converge the eigenvalue to an absolute error of 10^{-5}.

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 3: Total inner iterations for the IRAM on the cylindrical mesh problem. Power iteration results shown for comparison.

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

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 64.26cm $\times$ 64.26cm $\times$ 214.20cm 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. However, the upscatter portion of the scattering matrix is set to zero in order to compare power iteration calculations to the IRAM calculations.

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 4. 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. As in the previous set of results, we

Table 4: Iteration counts for the three dimensional MOX fuel assembly benchmark problem. The "k only" entry shows the number iterations for the power iteration method to converge the eigenvalue to an absolute error of 10^{-5}.

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

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 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 classical 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.

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. Finally, our experience suggests that the IRAM is insensitive to dominance ratio.

4 CONCLUSIONS

The Implicitly Restarted Arnoldi Method as available in the software package ARPACK was tested with the three dimensional, unstructured tetrahedral mesh, linear discontinuous discrete ordinates transport code AttilaV2. We were able to easily implement the method using the existing outer and inner iteration coding and other available computational machinery.

Our numerical experiments, although certainly not exhaustive, show that the method is robust and extremely efficient for several difficult representative problems, when compared to the power iteration method already implemented in AttilaV2. We did not encounter any difficulties with convergence of the method. We found that any increase in work per iteration associated with the eigenvalue solver is vastly outweighed by the improvement in convergence rate. The small additional memory requirement is tolerable, especially in light of the improved performance.

The ARPACK implementation of IRAM has been extended to parallel platforms[35] 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 IRAM will perform any better or worse in parallel.

The IRAM is not limited to the discretized transport equation used in the numerical results presented here. We believe that the IRAM could perform just as well, relative to power iteration, for other types of angular and spatial discretizations. Further 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 plan to address this in the future. Nonetheless, the potential of the IRAM for use in k–eigenvalue calculations is clear. This is especially true if it is applied to criticality searches, where multiple calculations are made over a range of parameters that are adjusted to achieve a desired eigenvalue.

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

3. 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.

4. 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).

5. J. S. Warsa, T. A. Wareing, and J. E. Morel, "Diffusion Synthetic Acceleration – Part I: Deficiencies in Multi–Dimensional Heterogeneous Problems," *Nuclear Science and Engineering* (2002), submitted.

6. J. S. Warsa, T. A. Wareing, and J. E. Morel, "Diffusion Synthetic Acceleration – Part II: Krylov Methods for Multi–Dimensional Heterogeneous Problems," *Nuclear Science and Engineering* (2002), submitted.

7. 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).

8. R. B. Lehoucq, D. C. Sorensen, and C. Yang, **ARPACK User's Guide**. SIAM: Philadelphia (1998).

9. 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).

10. 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).

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

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

13. 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).

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

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

16. T. A. Manteuffel, "The Tchebyshev Iteration for Nonsymmetric Linear Systems," *Numerische Mathematik*, **28**, pp. 307–327 (1977).

17. B. Fischer and R. Freund, "Chebyshev Polynomials Are Not Always Optimal," *J. Approximation Theory*, **65**, pp. 261–272 (1991).

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

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

20. D. C. Sorensen, "Numerical Methods for Large Eigenvalue Problems," *Acta Numerica*, **11**, n. 00, pp. 519–584 (2002).

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

22. G. L. Ramone, M. L. Adams, and P. F. Nowak, "A Transport Synthetic Acceleration Method for Transport Iterations," *Nucl. Sci. and Engr.*, **125**, pp. 257–283 (1997).

23. R. Sanchez and S. Santandrea, "Symmetrization of the Transport Operator and Lanzcos' Iterations," in **Proceedings of the 2001 International Meeting on Mathematical Methods for Nuclear Applications**, 9–13 Sep., Salt Lake City, Utah (2001).

24. 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).

25. 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. 1–16 (2002).

26. 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.*.

27. 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).

28. 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).

29. 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).

30. M. H. Gutknecht and S. Röllin, "The Chebyshev Iteration Revisited," *Parallel Computing*, **28**, pp. 263–283 (2002).

31. 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).

32. 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).

33. 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).

34. 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).

35. 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.