Efficient Solutions to the NDA-NCA Low-Order Eigenvalue Problem

Jeffrey A. Willert
North Carolina State University
Raleigh, North Carolina, 27606
jawiller@ncsu.edu

C. T. Kelley
Department of Mathematics
North Carolina State University
Raleigh, North Carolina, 27606
tim_kelley@ncsu.edu

ABSTRACT

Recent algorithmic advances combine moment-based acceleration and Jacobian-Free Newton-Krylov (JFNK) methods to accelerate the computation of the dominant eigenvalue in a k-eigenvalue calculation. In particular, NDA-NCA [1], builds a sequence of low-order (LO) diffusion-based eigenvalue problems in which the solution converges to the true eigenvalue solution. Within NDA-NCA, the solution to the LO k-eigenvalue problem is computed by solving a system of nonlinear equation using some variant of Newton's method. We show that we can speed up the solution to the LO problem dramatically by abandoning the JFNK method and exploiting the structure of the Jacobian matrix.

Key Words: Eigenvalue Calculation, Nonlinear Acceleration, Newton's Method

1. INTRODUCTION

We are interested in computing the dominant eigenvalue/eigenvector pair for neutron transport criticality problems. Traditional methods for computing this eigenvalue/eigenvector pair are generally iterative, and are often modeled around variants of the power method.

We'll write the general eigenvalue problem as

$$\mathcal{A}x = \lambda \mathcal{F}x \tag{1}$$

in which $\mathcal{A}$ is a linear operator. The eigenvalues of $\mathcal{A}$ are denoted by the set $\{\lambda_i\}_{i=1}^N$ in which N is the dimension of the problem. We assume that these eigenvalues are ordered such that $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_N|$. Along with each of these eigenvalues is the associated eigenvector which is a member of the set $\{x_i\}_{i=1}^N$.

This can be re-written in a more familiar form as

$$\mathcal{A}^{-1}\mathcal{F}x = \frac{1}{\lambda}x. \tag{2}$$

We seek the the smallest λ for which Equation 2 holds for nonzero x.

The power iteration (PI) method yields the dominant eigenpair $(\frac{1}{\lambda_1}, x_1)$ through successive applications of the linear operator $\mathcal{A}$ to some approximation of the eigenvector. The convergence rate is based upon the ratio λ_2/λ_1, hereafter referred to as the *dominance ratio* and denoted by ρ. When $\rho \approx 1$, convergence may become unacceptably slow and acceleration may be required [1,3].

Over the last several years, many authors have considered Newton's method based algorithms for computing the dominant eigenvalue in both neutron transport and diffusion criticality problems [1,3–5]. Most of these approaches are very similar in the way in which they build the nonlinear system of equations. The approach taken in [1] uses moment-based acceleration to move much of the work to a low-order system, not dissimilar to Quasidiffusion for k-Eigenvalue problems [6], but goes one step further and includes the use of Newton's method.

In [1], the authors present two novel algorithms for accelerating the convergence of criticality problems. We'll focus primarily on the second of these two algorithms, nonlinear criticality acceleration applied to the drift-diffusion equation from nonlinear diffusion acceleration. This algorithm, hereafter referred to as NDA-NCA, attempts to accelerate convergence of the eigenvalue by solving a sequence of low-order (LO) eigenvalue problems which are the same dimension as the angularly integrated Boltzmann transport equation [1]. This is done by applying nonlinear diffusion acceleration (NDA) [2] to the original transport problem, which removes the angular dependence in the neutron balance equation. Upon convergence, the consistency property of NDA guarantees that we will obtain the same eigenvalue as one would using power iterations.

We will solve the LO eigenvalue problems by building a nonlinear system of equation, $\tilde{F}(\phi, k)$, which vanishes everywhere at the eigenvector/eigenvalue associated with dominant eigenvalue. We can use nonlinear elimination to remove the constraint equation for k and write a new function $F(\phi)$ which has the same solution. By moving the majority of the work to the LO problem, we can reduce the cost of computing the dominant eigenvalue considerably.

2. NONLINEAR DIFFUSION ACCELERATED NONLINEAR CRITICALITY ACCELERATION

2.1 The Transport Criticality Problem

As in [1,5], we'll consider solving the multigroup formulation of the neutron transport eigenvalue problem with isotropic scattering,

$$\hat{\Omega} \cdot \nabla \psi_g(\hat{\Omega}, \vec{r}) + \Sigma_{t,g} \psi_g(\hat{\Omega}, \vec{r}) = \frac{1}{4\pi} \left[\sum_{g'=1}^{G} \Sigma_s^{g' \to g} \phi_{g'}(\vec{r}) + \frac{\chi_g}{k_{eff}} \sum_{g'=1}^{G} \nu \Sigma_{f,g'} \phi_{g'}(\vec{r}) \right], \tag{3}$$

in which ψ_g is the group angular flux and $\phi_g = \int_{4\pi} \psi_g d\Omega$ is the group scalar flux for groups $g = 1, 2, \cdots, G$. Furthermore, $\Sigma_{t,g}$, $\Sigma_s^{g' \to g}$, and $\Sigma_{f,g}$ are the total, scattering and fission cross-sections for group g. χ_g denotes the fission spectrum, ν is the mean number of neutrons emitted per fission event and k_{eff} is the dominant eigenvalue.

In the following sections, it will be helpful for us to represent Equation 3 in operator notation, so we write this as

$$\mathcal{L}\Psi = \mathcal{M}\left[\mathcal{S} + \frac{1}{k_{eff}}\mathcal{F}\right]\Phi. \tag{4}$$

Here, Ψ and Φ are vectors comprised of the individual group angular and scalar fluxes, respectively:

$$\begin{aligned}\Psi &= [\psi_1, \psi_2, \cdots, \psi_G] \\ \Phi &= [\phi_1, \phi_2, \cdots, \phi_G].\end{aligned}$$

In Sections 2.2 and 2.3 we introduce the notation necessary for Section 2.4 in which we give a detailed description of the analytic Jacobian formulation for NDA-NCA.

2.2 Acceleration via NDA

As in [1,2], we compute the zeroth angular moment of Equation 3 and arrive at the neutron balance equation,

$$\nabla \cdot \vec{J}_g(\vec{r}) + (\Sigma_{t,g} - \Sigma_s^{g\to g})\phi_g(\vec{r}) = \sum_{g'\neq g} \Sigma_s^{g'\to g}\phi_{g'}(\vec{r}) + \frac{\chi_g}{k_{eff}}\sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{g'}(\vec{r}) \tag{5}$$

in which the current, $\vec{J}$, is defined by

$$\vec{J}_g = \int_{4\pi} d\Omega\, \hat{\Omega}\psi_g(\hat{\Omega}, \vec{r}). \tag{6}$$

We use the following closure relationship for the current [1,2]:

$$\vec{J}_g = -\frac{1}{3\Sigma_{t,g}}\nabla\phi_g + \hat{D}_g\phi_g, \tag{7}$$

in which $\hat{D}$ is referred to as a consistency term. $\hat{D}$ enforces discrete consistency between the transport equation and Equation 8 upon convergence. In this sense, $\hat{D}$ is responsible for both compensating for the error in the Fick's law approximation [2] and matching the discretization error in the high-order and low-order problem.

The LO system is obtained by substituting Equation 7 into Equation 5 in place of the current:

$$\nabla \cdot \left[-\frac{1}{3\Sigma_{t,g}}\nabla\phi_g + \hat{D}_g\phi_g\right] + (\Sigma_{t,g} - \Sigma_s^{g\to g})\phi_g = \sum_{g'\neq g} \Sigma_s^{g'\to g}\phi_{g'} + \frac{\chi_g}{k_{eff}}\sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{g'}. \tag{8}$$

We compute $\hat{D}_g$ via a discretization of Equation 7 using high-order (HO) quantities ϕ_g^{HO} and $\vec{J}_g^{HO}$:

$$\hat{D}_g = \frac{\vec{J}_g^{HO} + \frac{1}{3\Sigma_{t,g}}\nabla\phi_g^{HO}}{\phi_g^{HO}}. \tag{9}$$

High-order quantities are those which are computed directly via integration of the high-order angular flux, ψ_g computed by the inversion of $\mathcal{L}$ in Equation 4 for some approximation to the scalar flux. ϕ^{HO} is computed at cell centers as cell-average values and $\vec{J}^{HO}$ is evaluated at cell faces.

We will express Equation 8 in operator notation,

$$\mathcal{D}\Phi = (\mathcal{S}_U + \mathcal{S}_L)\Phi + \frac{1}{k_{eff}}\chi\mathcal{F}\Phi, \tag{10}$$

in which $\mathcal{D}$ is the drift-diffusion operator and $\mathcal{S}_U$ and $\mathcal{S}_L$ are the upper and lower scattering operators, respectively.

2.3 NDA-NCA

With NDA-NCA, we use the HO equation, Equation 4, to compute an approximation, $\Psi^{(n)}$, to the angular flux given some approximation, $\Phi^{(n)}$, to the scalar flux. Given $\Psi^{(n)}$ and its first two angular moments, we can compute an approximation to the drift-term, $\hat{D}^{(n)}$. With this $\hat{D}^{(n)}$, we can build a low-order eigenvalue problem which we will solve (at least approximately) for a new scalar flux, $\Phi^{(n+1)}$. Upon convergence, NDA-NCA yields the dominant eigenpair. The authors of [1] describe the algorithm as we do in Algorithm 2.3.

Algorithm 2.3: NDA-NCA

Choose initial iterate $\Phi^{(0)}$, approximate eigenvalue k^0, outer iteration counter $m = 0$
while eigenvalue not converged **do**
Update counter, $m = m + 1$.
Execute transport sweep and compute consistency term:

$$\begin{aligned}
\Psi^{(m)} &= \mathcal{L}^{-1}\left(\mathcal{S}\Phi^{(m-1)} + \frac{1}{k^{(m-1)}}\chi\mathcal{F}\Phi^{(m-1)}\right) \\
\Phi^{HO} &= \int \Psi^{(m)} d\hat{\Omega}, \quad \vec{J}^{HO} = \int \hat{\Omega}\Psi^{(m)} d\hat{\Omega} \\
\hat{D}^{(m)} &= \frac{\vec{J}^{HO} + \frac{1}{3\Sigma_t}\nabla\Phi^{HO}}{\Phi^{HO}}
\end{aligned}$$

Solve $F(\phi) = 0$ using a Jacobian-Free Newton-Krylov method [7], where

$$\begin{aligned}
F(\phi) &= \mathcal{D}\phi - (\mathcal{S}_U + \mathcal{S}_L)\phi - \frac{1}{k_{eff}}\chi\mathcal{F}\phi \text{ in which} \\
k_{eff} &= \int \mathcal{F}\phi \, dV
\end{aligned}$$

Update flux and eigenvalue: $\Phi^{(m)} = \phi$, $k^{(m)} = \int \nu\Sigma_f\phi \, dV$.
end while

Inside the outer loop of NDA-NCA we are required to solve $F(\phi) = 0$. In practice, we solve this nonlinear equation only approximately by taking a single Newton step. To compute this Newton step, w, we must

solve the linear equation

$$F'(\phi)w = -F(\phi) \tag{11}$$

in which $F'(\phi)$ is the Jacobian of F at ϕ. The authors of [1] propose computing this step using Krylov methods [7]. As always with Krylov methods, the key to quickly solving the linear system of equations is preconditioning. The authors of [3] discuss two preconditioning routines, a power iteration and a power iteration employing the Wielandt shift. Each of these methods is effective in reducing the number of Krylov iterations per Newton step. In Table II in Section 3, we'll see that without preconditiong we may require a large amount of Krylov's per Newton iteration.

2.4 Analytic Jacobian Inversion in Computing the Newton Step

In many applications, JFNK is a preferred nonlinear solver due to its robustness and efficiency with which it solves $F(\phi) = 0$. A second reason why JFNK is often used is it does not require the user to compute or store a copy of the Jacobian matrix. Krylov methods, in general, only require a matrix-vector product routine which applies the Jacobian matrix to a vector. Generally, this matrix-vector product is computed using a finite-difference directional derivative, given by

$$F'(u)w = Jw \approx \frac{F(u + \epsilon w) - F(u)}{\epsilon}, \tag{12}$$

in which J is the Jacobian matrix at the point u. This is usually a perfectly adequate approximation to the Jacobian-vector product.

Furthermore, most implementations of JFNK utilize an inexact Newton iteration. That is, instead of solving

$$F'(u)w = -F(u) \tag{13}$$

exactly, the algorithm attempts to compute a vector w such that

$$\|F'(u)w + F(u)\|_2 < \gamma\|F(u)\|. \tag{14}$$

This algorithm will not yield quadratic convergence [1,7], however it effectively limits the amount of work the linear solver must do for each Newton iteration.

However, in certain cases, one can apply direct methods to exploit the structure of the Jacobian matrix when this structure can be computed. For this problem, we will show that we can represent the Jacobian as

$$J = M + yz^T \tag{15}$$

where M is a sparse, banded matrix and yz^T is a rank-one perturbation. In this special case, we can invert J using exactly two inversions of M using the Sherman-Morrison formula. This can yield major savings, as the inversion of M is equivalent to the application of the preconditioner used in [1]. Given the data in Table V of [1], this may yield a 50 - 75% reduction in cost when computing the Newton step.

Theorem 1. *The Jacobian, $F'(\phi)$, for the function, F, described by Equation 11, can be decomposed into the sum of a sparse matrix and a rank-one update.*

Proof. We can compute the action of the Jacobian on a vector using the formula

$$F'(\phi)w = \lim_{\epsilon \to 0} \frac{F(\phi + \epsilon w) - F(\phi)}{\epsilon}. \tag{16}$$

Before applying this formula, we should note that we can break down the function, F, into its linear and nonlinear components

$$\begin{aligned} F(\phi) &= \mathcal{A}\phi - \mathcal{N}(\phi) \text{ where} & (17)\\ \mathcal{A}\phi &= \mathcal{D}\phi - (\mathcal{S}_U + \mathcal{S}_L)\phi & (18)\\ \mathcal{N}(\phi) &= \frac{1}{k_{eff}(\phi)}\chi\mathcal{F}\phi & (19) \end{aligned}$$

In Equation 19 we write $k_{eff}(\phi)$ to note the explicit dependence on k_{eff} of ϕ. It is important to note that k_{eff} is a linear function of ϕ.

Now, we begin by noticing that we can write

$$F'(\phi)w = \mathcal{A}w + \mathcal{N}'(\phi)w. \tag{20}$$

At this point, the only thing left to do is compute $\mathcal{N}'(\phi)w$. For this, we turn back to Equation 16 and write

$$\begin{aligned} \mathcal{N}'(\phi)w &= \lim_{\epsilon \to 0} \frac{\mathcal{N}(\phi + \epsilon w) - \mathcal{N}(\phi)}{\epsilon}\\ &= \lim_{\epsilon \to 0} \frac{\frac{1}{k_{eff}(\phi+\epsilon w)}\chi\mathcal{F}(\phi + \epsilon w) - \frac{1}{k_{eff}(\phi)}\chi\mathcal{F}\phi}{\epsilon}\\ &= \lim_{\epsilon \to 0} \frac{\frac{\chi\mathcal{F}\phi+\epsilon\chi\mathcal{F}w}{k_{eff}(\phi)+\epsilon k_{eff}(w)} - \frac{\chi\mathcal{F}\phi}{k_{eff}(\phi)}}{\epsilon}\\ &= \lim_{\epsilon \to 0} \frac{\frac{\epsilon\chi\mathcal{F}w}{k_{eff}(\phi)+\epsilon k_{eff}(w)}}{\epsilon} + \lim_{\epsilon \to 0} \frac{\frac{\chi\mathcal{F}\phi}{k_{eff}(\phi)+\epsilon k_{eff}(w)} - \frac{\chi\mathcal{F}\phi}{k_{eff}(\phi)}}{\epsilon}\\ &= \frac{1}{k_{eff}(\phi)}\chi\mathcal{F}w + \lim_{\epsilon \to 0} \frac{\frac{\chi\mathcal{F}\phi\cdot k_{eff}(\phi) - \chi\mathcal{F}\phi\cdot\left[k_{eff}(\phi)+\epsilon k_{eff}(w)\right]}{\left[k_{eff}(\phi)+\epsilon k_{eff}(w)\right]\cdot k_{eff}(\phi)}}{\epsilon}\\ &= \frac{1}{k_{eff}(\phi)}\chi\mathcal{F}w + \lim_{\epsilon \to 0} \frac{-\chi\mathcal{F}\phi \cdot k_{eff}(w)}{[k_{eff}(\phi) + \epsilon k_{eff}(w)] \cdot k_{eff}(\phi)}\\ &= \frac{1}{k_{eff}(\phi)}\chi\mathcal{F}w - \frac{k_{eff}(w)}{k_{eff}(\phi)^2}\chi\mathcal{F}\phi \end{aligned} \tag{21}$$

Lastly, let's show that we can write the discrete form of the final term of Equation 21 as a rank-one matrix times the vector w. In order to see this, we should first note that we can write

$$k_{eff}(\vec{\phi}) = \vec{W}^T F \vec{\phi} \tag{22}$$

in which $\vec{W}$ is a vector of weights used to compute the numerical integral over the spatial domain.

Now, we have

$$\begin{aligned} -\frac{k_{eff}(\vec{w})}{k_{eff}(\vec{\phi})^2}\chi F\vec{\phi} &= -\frac{\vec{W}^T F\vec{w}}{(\vec{W}^T F\vec{\phi})^2}\chi F\vec{\phi} \\ &= -\frac{\chi F\vec{\phi}(\vec{W}^T F\vec{w})}{c^2} \\ &= -\frac{(\chi F\vec{\phi}\vec{W}^T F)\vec{w}}{c^2} \\ &= (\vec{y}\vec{z}^T)\vec{w} \end{aligned}$$

where

$$\begin{aligned} \vec{y} &= -\frac{\chi F\vec{\phi}}{c} \\ \vec{z}^T &= \frac{\vec{W}^T F}{c} \\ c &= \vec{W}^T F\vec{\phi}. \end{aligned}$$

Combining these results, we have

$$\begin{aligned} J\vec{w} &= A\vec{w} + \frac{1}{c}F\vec{w} + \vec{y}\vec{z}^T\vec{w} \\ &= M\vec{w} + \vec{y}\vec{z}^T\vec{w} \\ &= (M + \vec{y}\vec{z}^T)\vec{w} \end{aligned} \tag{23}$$

in which A is the matrix associated with the discretization of $\mathcal{A}$. □

We'll remind the reader that the Sherman-Morrison formula [7] states that

$$(M + yz^T)^{-1} = M^{-1} - \frac{M^{-1}yz^TM^{-1}}{1 + z^TM^{-1}y} \tag{24}$$

as long as $1 + z^TM^{-1}y \neq 0$. This means that the inverse of $M + yz^T$ can be applied to a vector w by simply computing the application of M^{-1} on w and M^{-1} on y. When M is a sparse matrix for which we have efficient methods for applying its inverse, we can significantly reduce computation time.

2.5 Inverting M

For simple problems, M may easily inverted using direct methods for solving a sparse, banded system of equations. However, as the problem becomes larger and there is significant group to group scattering, direct inversion may not be the best choice. In this case, using multigrid methods or preconditioned GMRES may be an acceptable alternative. In either case, M will be inverted only approximately and some error will exist in the computation of the Newton step. Theory exists for inexact Newton's method [7] which describes the behavior of the Newton iteration when the Jacobian is inverted using iterative methods.

3 1-D, 1-GROUP TEST PROBLEMS AND RESULTS

As a proof of concept, we will demonstrate results for four 1-D, 1-Group k-eigenvalue test problems. The parameters used for each of these test problems is displayed in Table I. Each of these domains includes a single fissile region surrounded by a 5 cm reflector on each side. The table displays the total system length, τ, in cm. Therefore, the width of the fissile region for each test is $\tau - 10$ cm. For each test we use an S_{20} discretization in angle and a standard finite difference discretization in space. N_x denotes the number of spatial grid points used. These are the same one-dimensional tests that were considered in [3].

Table I: One-Dimensional, One-Group, Slab Geometry k-Eigenvalue Test Problems

Test	1	2	3	4
Σ_s	.856	.856	.856	.856
Σ_t	1	1	1	1
$\nu\Sigma_f$	.144	.144	.144	.144
τ	210	110	60	35
N_x	2100	1100	600	350

Table I. 1D Tests for k-Eigenvalue Problem

For each of these algorithms, we set the convergence tolerance, η, equal to 10^{-6} for the eigenvalue, where

$$\frac{k_{eff}^{(n)} - k_{eff}^{(n-1)}}{k_{eff}^{(n-1)}} < \eta \tag{25}$$

is used as the stopping criterion. In Table II we present the required number of transport sweeps and the total number of inversions of M required to reach convergence (LO solves in parenthesis). For each of these test problems, we give the number of iterations required for power iteration to converge as a baseline comparison and report statistics for NDA-NCA in which a Krylov method is used to solve for the Newton step (unpreconditioned and preconditioned with power iteration) and NDA-NCA in which a direct inversion of the Jacobian is used.

Table II demonstrates the inefficiency of the power iteration algorithm when computing the eigenvalue for high dominance ratio problems. NDA-NCA(K-U) is giant improvement over PI; we can see that number of transport sweeps is reduced by a factor of roughly 60 - 100. There is a cost incurred in the LO function evaluations/solves, but these are much cheaper that HO transport sweeps. For Test 1, a LO function evaluation is 25 times less expensive that a transport sweep. In NDA-NCA(K-U), unpreconditioned GMRES is used to solve for the Newton step. The number in parenthesis refers to the number of nonlinear function evaluations used within all Krylov solves. NDA-NCA(K-U) often requires significantly more transport sweeps as we limit the number of Newton steps per LO solve to one and the number of Krylov vectors per Newton step to 20. This results in a lower accuracy update at each iteration. NDA-NCA(K-PI) utilizes GMRES preconditioned by a power iteration to solve for the Newton step. The number in parenthesis refers to the number of preconditioned Krylov iterations. Each iteration involves a nonlinear

Table II: One-Dimensional, One-Group, Slab Geometry k-Eigenvalue Test Problems Convergence Results

Method	Test 1	Test 2	Test 3	Test 4
Power Iteration	5417	3064	1486	651
NDA-NCA(K-U)	55(2254)	31(1271)	19(779)	11(410)
NDA-NCA(K-PI)	6(103)	5(51)	5(36)	5(27)
NDA-NCA(D)	7(14)	5(10)	5(10)	4(8)

Table II. 1-D Test Problem Convergence Results for k-Eigenvalue Problem

function evaluation and a preconditioner application. Finally, NDA-NCA(D) uses direct inversion of the Jacobian using the Sherman-Morrison formula to compute the Newton step. The number in parenthesis refers to the number of sparse matrix inversions - each of which is equivalent in cost to a preconditioner application in NDA-NCA(K-PI).

Furthermore, we can see that preconditioning the linear solve for the Newton step can cut the number of LO function evaluations by a factor of 15 - 20. Beyond this, NDA-NCA(D) eliminates much of the work that was incurred in NDA-NCA(K-PI). Each outer iteration of NDA-NCA(D) requires a single nonlinear function evaluation and two sparse matrix inversions, the same matrix inversions that take place as part of the preconditioning process in NDA-NCA(K-PI).

4 2-D, 2-GROUP TEST PROBLEMS AND RESULTS

We will now present results using NDA-NCA to compute the dominant eigenvalue for the LRA-BWR as in [1,8]. This test was run using a 165 $\times$ 165 mesh and an S_{16} angular quadrature. For this test we only compare NDA-NCA(K-PI) and NDA-NCA(D). For the test, we use the direct inverse of M as the preconditioner when using JFNK to solve for the low-order update. In this sense, a single low-order update for NDA-NCA(D) is the same cost as two preconditioner applications in NDA-NCA(K-PI). We present iteration statistics in Table III. This data was obtained via an implementation of NDA-NCA in MATLAB. The eigenvalue was converged to a relative tolerance of 10^{-12}.

As we can see in Table III, using the direct inverse for this problem can be highly beneficial. Not only does it produce a more accurate Newton step, which in this case reduces the number of total transport sweeps required, but it also lowers the cost of each LO solve by a factor of over 4. This can potentially yield significant savings for large problems.

5. CONCLUSIONS

The authors of [1] demonstrated the impressive efficiency of NDA-NCA in computing the dominant eigenvalue/eigenvector pair for the neutron transport criticality problem. We have shown that we can

Table III: LRA-BWR Iteration Statistics

Method	NDA-NCA(D)	NDA-NCA(K-PI)
Transport Sweeps	13	14
Time per Transport Sweep (s)	4.884	4.854
Direct Inverses of M per Iteration	2	8.93
Function Evaluations per Iteration	1	9.93
Average LO Time per Iteration (s)	1.206	5.213

Table III. Iteration counts and timings for NDA-NCA high-order and low-order solves.

improve this efficiency by computing the Newton step in the LO nonlinear solve using the Sherman-Morrison formula. This can reduce the work considerably over using a preconditioned Krylov solve for the step. We have demonstrated this effectiveness for a set of 1-D, 1-Group, multi-material test problems, as well as the well-known LRA-BWR benchmark problem.

In the future, we look to incorporate this analytic Jacobian into a Trilinos implementation of the NDA-NCA algorithm. In this setting, we will be able to test other variants of the algorithm. We plan to consider inverting M using multigrid methods and preconditioned GMRES and compare to using direct methods.

ACKNOWLEDGEMENTS

The work of these authors has been partially supported by the CoCoMANS LDRD DR Project at Los Alamos National Laboratory and the Consortium for Advanced Simulation of Light Water Reactors (www.casl.gov), an Energy Innovation Hub (http://www.energy.gov/hubs) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725, National Science Foundation Grant CDI-0941253 and Army Research Office Grants W911NF-11-1-0367, and W911NF-07-1-0112.

REFERENCES

[1] H. Park, D. A. Knoll, and C. K. Newman, "Nonlinear Acceleration of Transport Criticality Problems," *Nuclear Science and Engineering*, **171**, pp. 1-14 (2012).

[2] D. Knoll, H. Park and K. Smith, "Application of the Jacobian-Free Newton-Krylov Method to Nonlinear Accleration of Transport Source Iteration in Slab Geometry," *Nuclear Science and Engineering*, **167**, pp. 122 (2011).

[3] D. A. Knoll, H. Park, and C. Newman, "Acceleration of k-Eigenvalue/Criticality Calculations Using the Jacobian-Free Newton-Krylov Method," *Nuclear Science and Engineering*, **167**, pp. 133-140 (2011).

[4] Daniel F. Gill and Yousry Y. Azmy, "Newton's Method for Solving k-Eigenvalue Problems in Neutron Diffusion Theory," *Nuclear Science and Engineering*, **167**, pp. 141-153 (2011).

[5] Daniel F. Gill, Yousry Y. Azmy, J.S. Warsa, and J.D. Densmore, "Newton's Method for the Computation of k-Eigenvalues in S_N Transport Applications," *Nuclear Science and Engineering*, **168**, pp. 37-58 (2011).

[6] Marvin L. Adams and Edward W. Larsen, "Fast Iterative Methods for Discrete-Ordinates Particle Transport Calculations," *Progress in Nuclear Energy*, **40**, pp. 3 - 159 (2002).

[7] C. T. Kelley, "Iterative Methods for Linear and Nonlinear Equations," *no. 16 in Frontiers in Applied Mathematics*, SIAM, Philadelphia, 1995.

[8] "Argonne Code Center: Benchmark Problem Book," ANL-7416, suppl. 2, Argonne National Laboratory (1977).