THE LOBPCG ITERATIVE METHOD APPLIED ON A 2D REACTOR PHYSICS K EIGENVALUE PROBLEM

J.S. TCHAKERIAN

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
77 MASSACHUSETTS AVE, CAMBRIDGE, MA 02139

STCHAKER@MIT.EDU

Abstract. This paper demonstrates the locally optimal block preconditioned conjugate gradient as a potential iterative method for solving k-eigenvalue problems in reactor physics. Nuclear reactor systems are complex and the deterministic codes which exist to solve these systems require powerful iterative solvers. A 3 x 3, simple pressurized water reactor lattice was generated in the OpenMC Monte Carlo code to serve as a foundational problem to compare the implementation of the proposed iterative method. In addition, a 2D finite di erence di usion solver was created to discretize the neutron transport partial di erential equations. The resulting matrix equation was solved for the k-eigenvalue using both the more commonly documented inverse power iteration method, as well as the proposed locally optimal block preconditioned conjugate gradient method. The results of this analysis indicate that the locally optimal block preconditioned conjugate gradient method converges in fewer iterations and shows a higher degree of similarity to the monte carlo solution when compared to the traditionally used inverse power iteration method. Based on these findings, reactor physics deterministic solvers should attempt to integrate the locally optimal block preconditioned conjugate gradient method into their solution suites as an available iterative technique for k-eigenvalue problems with mono-energetic approximations.

Key words. Iterative Solvers, LOBPCG, Preconditioning, Eigenvalue Problems, Monte Carlo, Reactor Physics

1. Introduction. The solution of k-eigenvalue problems in reactor physics is of the highest importance in the validation and certification of designed nuclear fission systems. A solved k-eigenvalue problem results in the calculation of the fundamental mode eigenvalue, known in the reactor physics community as the K_{eff} value [1]. This K_{eff} value determines the generational growth or decay of neutrons in the nuclear system, and the subsequent fundamental mode eigenvector, the neutron flux, describes the path-length rate density of the neutron population. Both of these quantities are essential pieces of information for engineers to determine if the designed nuclear system is functioning as intended.

Nuclear reactor systems are modeled using a steady-state Boltzmann equation to govern the neutron transport across the domain. Modern deterministic neutron transport solvers are three-dimensional, have thousands of mesh points, and use dozens of energy groups to discretize the problem and nuclear data. They must also discretize the angular component of the problem through some sort of approximation [2]. Regardless of the angular approximation that is used, the problems generated by this analysis generate large, sparse matrices which lead to a generalized matrix eigenvalue problem [3]. In reactor physics analysis, these iterative problems are traditionally solved using inverse power iteration (IPI) [3]. While this method yields accurate convergence for the k-eigenvalue problem, it can also become computationally expensive when run on dense, multidimensional problems. This is because IPI must not only converge over a solution to a system of linear equations, but also on the eigenvalue and eigenvector. Due to this fact, development on the use of conjugate gradient (CG) style methods to accelerate the convergence of k-eigenvalue problems has been a heavily invested research topic in the reactor physics community [3].

The goal of this paper will be to implement a CG style iterative solver known

as the locally optimal block preconditioned conjugate gradient (LOBPCG) into a 2D k-eigenvalue solver with a diffusion approximation [4]. A direct comparison of this implementation will be made against the IPI method, and both methods will be compared, from an accuracy standpoint, to the neutron transport Monte Carlo (MC) code OpenMC [5]. The MC method is considered the gold standard for neutron transport analysis. Therefore, success of the LOBPCG implementation will depend on a comparison of both convergence properties, and similarity with the OpenMC k-eigenvalue. The following sections describe in greater detail the problem that will be solved, as well as the purpose and strengths of the LOBPCG solver when applied to k-eigenvalue problems.

2. Methodology. This section will discuss the necessary background information and methodology which governs the problem at hand. Necessary information will be provided in the equations and approximations for the neutron transport aspect of this paper. From this basis, the k-eigenvalue problem will be explicitly derived in matrix form. In addition, the IPI method applied to the k-eigenvalue problem is shown. Lastly, the LOBPCG iterative solver is discussed, and the primary reasons for its implementation in reactor physics problems is given.

2.1. Reactor Physics Review. A brief reactor physics review will be included to provide the necessary information for the problem at hand. The governing neutron transport equation can be written in a variety of ways depending on the approximations made to the specific system. For the analysis of the reactor system in this review, the neutron transport equation will have a spatial and angular component, but will be mono-energetic (meaning all neutrons will be assumed to have the same speed) and steady state. These two approximations will remove the energy and time discretizations of the problem respectively, and allow the neutron transport equation to be written in the form of equation 2.1. The mono-energetic approximation is necessary to implement the LOBPCG algorithm, which will be discussed in more detail later.

$$\vec{\Omega} \cdot \nabla \psi(\vec{r}, \vec{\Omega}) + \Sigma_t(\vec{r}) \psi(\vec{r}, \vec{\Omega}) = Q(\vec{r}, \vec{\Omega}) \tag{2.1}$$

Where $\psi(\vec{r}, \vec{\Omega})$ represents the angular neutron flux, $\Sigma_t(\vec{r})$ is the total neutron macroscopic cross section, and $Q(\vec{r}, \vec{\Omega})$ is the neutron source term. To further simplify the problem, and to obtain the k-eigenvalue, an approximation is needed on the angular component of equation 2.1. Many approximations exist to discretize the angular component of this problem, however, the most simplistic of these can be described by diffusion theory. Diffusion theory averages the angular components of neutron trajectories into a global average direction dictated through the use of Fick's Law, which can be seen in equation 2.2 [6].

$$J(\vec{r}) = \quad D(\vec{r}) \nabla \phi(\vec{r}) \tag{2.2}$$

Where $J(\vec{r})$ is the neutron current density which describes the angular component, $D(\vec{r})$ is the diffusion coefficient of the medium, and $\phi(\vec{r})$ is the "scalar" neutron flux (note that the use of scalar here simply means that the flux is no longer dependent on an angular component). As a result of the approximation made by Fick's Law in equation 2.2, the neutron transport equation has only a single remaining dimension, space. This particular form of equation 2.1 is known as the neutron diffusion equation, and its explicit form can be seen in equation 2.3.

$$\nabla \cdot D(\vec{r}) \nabla \phi(\vec{r}) + \Sigma_t(\vec{r}) \phi(\vec{r}) \quad \Sigma_s(\vec{r}) \phi(\vec{r}) = \frac{1}{k} \nu \Sigma_f(\vec{r}) \phi(\vec{r}) \tag{2.3}$$

Where $\Sigma_s(\vec{r})$ is the neutron scattering cross section, k is the k-eigenvalue, ν is the neutron reproduction term, and $\Sigma_f(\vec{r})$ is the neutron fission cross section. Furthermore, equation 2.3 can be re-arranged to yield the form seen in equation 2.4, provided the diffusion coefficient can be assumed to be constant in the spatial domain.

$$(\quad D\nabla^2 + \Sigma_t \quad \Sigma_s)\phi = \frac{1}{k}(\nu\Sigma_f)\phi \tag{2.4}$$

Equation 2.4 can be analogously thought of as a generalized eigenvalue problem of the form seen in equation 2.5

$$Ax = \lambda Bx \tag{2.5}$$

Where $(\quad D\nabla^2 + \Sigma_t \quad \Sigma_s)$, $(\nu\Sigma_f)$, $\frac{1}{k}$, and ϕ from equation 2.4 constitute A, B, λ, and x from equation 2.5. For future analysis, the $(\quad D\nabla^2 + \Sigma_t \quad \Sigma_s)$ term will be referred to as the L (loss) matrix, and the $(\nu\Sigma_f)$ term as the F (fission) matrix. Once a partial differential equation discretization scheme, such as finite difference, is applied across the spatial domain $\vec{r}$, the generalized eigenvalue problem seen in equation 2.4 can be iterativley solved for the k-eigenvalue [3].

2.2. Inverse Power Iteration in Reactor Physics. The eigenvalue matrix problem established through the use of equation 2.4 must be solved using numerical iterative methods that converge on the eigenvalue and eigenvector of the system. Nuclear reactor analysis is primarily concerned with the dominant eigenvalue, or K_{eff} value. One of the oldest methods used to find the K_{eff} value, and the one that will be used as a comparative basis to the LOBPCG iterative solver, is IPI. If it can be assumed that the matrix A from equation 2.5 is an n x n diagonalizable matrix, then it will have some ordered eigenvalues according to equation 2.6.

$$|\lambda_1| \geq |\lambda_2| \geq ..... \geq |\lambda_n| \tag{2.6}$$

From this assumption, we can also factorize the matrix A into a diagonal matrix $(\quad)$ of eigenvalues of the form seen in equation 2.7.

$$A = V \quad V \quad ^1 \tag{2.7}$$

The factorization seen in equation 2.7 can be expressed in the form of a matrix exponential, k, by equation 2.8.

$$A^k = (V \quad V \quad ^1)^k = V \quad ^k V \quad ^1 = A^k V = V \quad ^k \tag{2.8}$$

For a random guess on the initial eigenvector $x = V\tilde{x}$, we can further simplify the form of the eigenvalue problem seen in equation 2.5 to the form of equation 2.9 [7].

$$A^k x = A^k V\tilde{x} = \sum_{i=1}^{n} v_i \lambda_i^k \tilde{x}_i \tag{2.9}$$

The main reason for transforming the general eigenvalue problem seen in equation 2.5 with the form seen in equation 2.9 is to make it easy to determine the convergence properties of the dominant eigenvalue. From the inequality relationship in equation 2.6, this dominant eigenvalue can be described by λ_1. It is possible to factor out this value to get a special form of the equation which describes the convergence properties of this problem. This special form is displayed in equation 2.10 [7].

$$A^k x = \lambda_1 (\sum_{i=1}^{n} v_i (\frac{\lambda_i}{\lambda_1})^k \tilde{x}_i) \tag{2.10}$$

Equation 2.10 showcases many important properties of the problem at hand. For one, as the value of k is increased (analogous to iterations with the IPI method), all higher modes zero out except for the dominant eigenvalue. Additionally, the ratio of $\frac{-2}{1}$ is the slowest mode to zero out and is referred to as the dominance ratio in the reactor physics community [7]. The dominance ratio affects the convergence speed of nuclear reactor systems being solved using the IPI method. An example IPI algorithm to solve a nuclear system like the one described in equation 2.4, can be seen in algorithm 2.1.

Algorithm 2.1 Inverse Power Iteration as Applied to Reactor Physics.

Initialize $\phi = \phi^{\,0)}$
Normalize $\phi^{\,0)} = \frac{\phi^{\,0)}}{||\phi^{\,0)}||}$
Initialize $k = k^{\,0)}$
Initialize $b^{\,0)} = \frac{1}{k^{\,0)}} F\phi^{\,0)}$
Initialize $\epsilon = 1$
while $\epsilon \geq Tol$ **do**
    Solve $L\phi^{\,1)} = b^{\,0)}$
    Compute $k^{\,1)} = \frac{\langle F\phi^{\,1)} \rangle}{\langle F\phi^{\,0)} \rangle}$
    Normalize $\phi^{\,1)} = \frac{\phi^{\,1)}}{||\phi^{\,1)}||}$
    Compute ϵ
    Initialize $\phi^{\,0)} = \phi^{\,1)}$
    Initialize $k^{\,0)} = k^{\,1)}$
    Compute $b^{\,0)} = \frac{1}{k^{\,0)}} F\phi^{\,0)}$
end while
return k, ϕ

Algorithm 2.1 uses initial guesses for the normalized neutron flux and k-eigenvalue to solve a linear matrix equation for a new flux eigenvector. The new eigenvector is used to calculate a new eigenvalue through the use of a ratio weighted by the fission source matrix. This modification to the traditional IPI method is done to converge the problem on the fission source as well. Convergence on the fission source is important to ensure that the neutron flux solution is being weighed by the important regions of the problem. The ϵ value in algorithm 2.1, can be calculated by equation 2.11. This value is needed in order to determine the convergence between iterations of eigenvalue and eigenvector up to some user-specified tolerance, Tol.

$$\epsilon = ||\frac{x^i \quad x^{i\ \ 1}}{x^i}|| \tag{2.11}$$

Where x^i is the iterative quantity of interest at the current iteration and $x^{i\ \ 1}$ is the quantity at the previous iteration. Convergence is only met when the ϵ quantity calculated by equation 2.11 obtains a value that is less than Tol.

The IPI method as described in this section and by algorithm 2.1 is available for use in the vast majority of modern deterministic neutron transport solvers [7]. However, the method possesses some limitations which make the exploration of other iterative solvers important. For one, the IPI method converges only as fast as the dominance ratio. Certain reactor systems will contain dominance ratios close to unity. For this case, IPI does not work quickly, and can take an exponentially larger number of iterations to converge when compared to a system with a low dominance ratio.

Additionally, IPI is susceptible to matrices with large condition numbers, and a reactor system discretized by equation 2.4 may be ill-conditioned depending on the nuclear data and geometric configuration. For these reasons, the LOBPCG iterative method will be explored and implemented to determine if its algorithmic composition and ability to easily precondition the system improve convergence.

2.3. The LOBPCG Iterative Method. The IPI method has worked well for k-eigenvalue problems throughout the history of nuclear reactor physics, however, recent developments in computational power means that it is now realistic to solver large scale, 3D problems [2]. In order to efficiently solve these problems, more nuanced methods are developed which take advantage of problem structure and preconditioning. The following sections will discuss the motivational work behind the LOBPCG method, and will delve into the implementation of the algorithm itself.

2.3.1. Development of the LOBPCG Method. Highly specialized numerical linear algebra iterative solvers are developed in order to solve generalized eigenvalue problems as quickly as possible. In 1948, Kantorovich developed part of the fundamental basis for the LOBPCG method through the creation of the steepest descent method [8]. This method calculates the steepest descent direction using a scaled Rayleigh quotient, where the step size of the steepest descent is calculated by minimizing the Rayleigh quotient in a locally optimal form [8]. The Rayleigh quotient for a given real symmetric matrix A and nonzero vector x is given by equation 2.12, and the direction of steepest descent can be given by equation 2.13.

$$R(A, x) = \frac{x^t A x}{x^t x} \tag{2.12}$$

$$r = Ax \quad R(A, x) \tag{2.13}$$

More development was done to further enhance the foundation behind the steepest descent method, and in recent decades, the analysis of preconditioners for iterative methods has become a primary focus for numerical analysts [9]. The use of a good preconditioner matrix can dramatically reduce the number of iterations and total computation time of a generalized eigenvalue problem [10]. Applying a preconditioner T to the direction of steepest descent seen in equation 2.13 yields the preconditioned direction w seen in equation 2.14 [8].

$$w = Tr \tag{2.14}$$

With the information in equations 2.12 through 2.14 serving as a basis, LOBPCG was developed as an algorithm for computing extremal eigenvalues and eigenvectors of a symmetric/Hermitian matrix A and a symmetric/Hermitian positive definite (SPD) matrix B. The goal of this algorithm is a local minimization of the Rayleigh quotient on a unique subspace. This subspace is spanned by the current residual, as well as the previous eigenvector approximation and its block version [8]. When preconditioning is applied to the local minimization technique, the full LOBPCG algorithm was developed [4].

Several features of the LOBPCG algorithm make it worth implementing into generalized eigenvalue solve suites. For one, the cost per iteration and memory consumption of the algorithm is of similar magnitude to other conventional methods, such as the Lanczos method [8]. Additionally, LOBPCG can take advantage of direct

preconditioning that is built into the algorithm. In this way, utilizing a preconditioned matrix is done with much less hassle in LOBPCG. LOBPCG is also generally more numerically stable than the Lanczos method, despite both using the same Krylov Subspace [8]. That is not to say that LOBPCG suffers from no numerical instability. The algorithm can suffer from some instability when the basis vectors that form the subspace from which LOBPCG is solved on become linearly dependent [10]. This problem is exacerbated when the number of eigenpairs that the algorithm is solving for becomes relatively large. Importantly, for the k-eigenvalue problem discussed in this paper, only a single extremal eigenvalue is of interest so this issue should not be a problem. It is important to note that if one adds an energy dependence to the diffusion equation seen in 2.4, the matrices L and F will become multi-group. Multi-group is a term used in nuclear engineering to refer to the discretization of a neutron transport problem into multiple energy bins [7]. Matrices which take into account multi-group equations loose their symmetry, and so alternative algorithms, such as the ORTHOMIN(1) algorithm [1], have been developed to solve these non-symmetric matrix problems.

The most recent work on the LOBPCG algorithm has seen a modification of the original algorithm, called LOBPCG II, developed by the original creator Andrew Knyazev [4]. LOBPCG II was created to address speed issues for large block sizes by reducing the necessary dot products stemming from the Rayleigh-Ritz procedure [4]. This modification will not be necessary for the problem in this paper, but is is effective in increasing the parallelizability of the LOBPCG algorithm. Other applications for this algorithm include data mining, image segmentation via spectral clustering, and denoising [8]. Additionally, implementation into CUDA via the ABINIT code demonstrate one way in which the LOBPCG algorithm is being developed for high-performance GPU computing in material science codes. Of special interest to this paper, LOBPCG was also used to accelerate nuclear configuration interaction calculations by replacing an existing implementation of the Lanczos method [11]. It is clear that implementation of the LOBPCG algorithm presents exciting opportunities to potentially improve the solution of k-eigenvalue problems. The following section will discuss the LOBPCG algorithm as it will be implemented into the test problem.

2.3.2. LOBPCG Implementation. The LOBPCG method discussed in section 2.3.1 was implemented to solve the discretized k-eigenvalue problem based on the form of equation 2.4. Consideration was made for the initial preconditioning option for the L matrix, but by default the preconditioned matrix is set to an identity matrix of the same size as L. This is done to ensure that the preconditioner can be set after some analysis of the L matrix structure. The following algorithm is based off of Andrew Knyazev's original implementation of LOBPCG [4], but has been modified to be specified for use with equation 2.4. Algorithm 2.2 describes the standard LOBPCG algorithm as it has been implemented to solve k-eigenvalue problems.

Algorithm 2.2 LOBPCG Iterative Solver for k-eigenvalue Problems.

Initialize $\phi = \phi^{0)}$
Normalize $\phi^{0)} = \frac{\phi^{0)}}{||\phi^{0)}||}$
Initialize $p^{0)} = 0$
Initialize ϵ
for $i = 0$ until $(\epsilon \leq Tol)$ **do**
  Compute $k^{i)} = \frac{\langle \phi^{\ )}, F\phi^{\ )} \rangle}{\langle \phi^{\ )}, L\phi^{\ )} \rangle}$
  Compute $r = F\phi^{i)} \quad k^{i)} L\phi^{i)}$
  Compute $w^{i)} = Tr$
  Compute Rayleigh-Ritz values on trial subspace Span $\{w^{i)}, \phi^{i)}, p^{i)}\}$
  Compute $\phi^{i+1)} = w^{i)} + \tau^{i)} \phi^{i)} + \quad^{i)} p^{i)}$
  Normalize $\phi^{i+1)} = \frac{\phi^{+1)}}{||\phi^{+1)}||}$
  Compute $p^{i+1)} = w^{i)} + \quad^{i)} p^{i)}$
  Check for convergence on ϕ
end for
return ϕ
Compute K_eff from returned ϕ vector

The LOBPCG implementation seen in algorithm 2.2 is accomplished in a different way to the IPI implementation from algorithm 2.1. To start, an initial guess of the k-eigenvalue is handled within the algorithm, instead of being required by it. A for loop is then generated which will iterate until some convergence value has been achieved for our eigenvector. Algorithm 2.2 is more traditionally written in a way to allow for convergence across multiple eigenvalues and eigenvectors, however, the nature of the k-eigenvalue problem dictates that we are only interested in the dominant K_eff mode. The preconditioned residual $w^{i)}$, previous guess $\phi^{i)}$, and block directional $p^{0)}$, comprise a subspace to calculate the Ritz vector corresponding to the maximal Ritz value. [10]. The three terms in this subspace comprise what is referred to as a three-term-recurrence [4]. This linear combination results in a Ritz-vector $\phi^{i+1)}$, which represents the next iterate of our eigenvector search. In order to determine potential benefits of using algorithm 2.2 over algorithm 2.1, the following section discusses the k-eigenvalue problem setup.

3. Problem Setup. A few considerations need to be made in order to successfully generate a test problem for this analysis. Since the LOBPCG method requires a symmetric matrix eigenvalue problem to function, the problem setup must ensure the creation of such a matrix. Due to this fact, the finite-difference diffusion problem must be made energy independent. Adding energy dependency to equation 2.4 requires the splitting of neutron data into multi-group formats, which can be done easily, but undo the symmetry in the L and F matrices. The problem must also be realistic and sufficient in its complexity to allow for a potential benefit to be seen by using algorithm 2.2. For this analysis, a 3 x 3 assembly for a pressurized water reactor (PWR) was chosen. PWRs are the primary nuclear reactor type used within the United States, and data for these reactors is readily available. The open source MC code OpenMC [5] can easily create this assembly and calculate its k-eigenvalue. A visual representation of this test, in the form of a square 3 x 3 assembly, can be seen in figure 1.

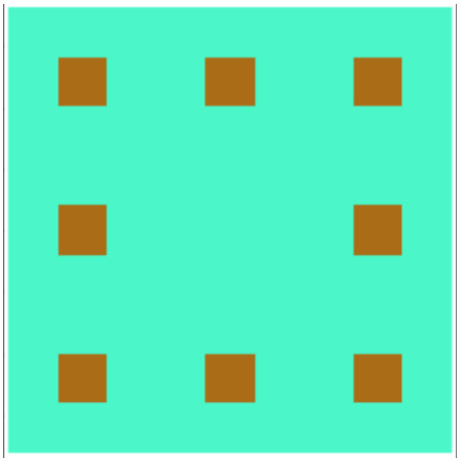

FIG 1. *3x3 PWR Assembly generated in OpenMC.*

Within figure 1, the light blue represents 400 ppm borated water. The water in this system serves both to cool the heat generated by the nuclear reactor, but also to moderate the neutrons via elastic scatter collisions to lower energies. This phenomenon is pursued in order to induce more fission events, which for Uranium-235, occur more often when the neutrons are at lower energies. Some amount of Boron is typically added to water in PWRs to keep the neutron population under control. The smaller, brown squares centered around the assembly contain a typical mixture of nuclear fuel, Uranium Dioxide, at 3.2% enrichment. Enrichment percentage simply refers to the weight percent of Uranium-235 used in the nuclear fuel. Each side of this square assembly is approximately 3.6 cm in length, with the density of the fuel cells being around $10.341 \frac{g}{cm^3}$. Realistically, a complete nuclear reactor core would be comprised of more than a single assembly, however, for the goal of simple LOBPCG implementation, this problem design will suffice.

OpenMC [7] will run the designed problem, not only to generate a reference k-eigenvalue and eigenvector, but also to generate the cross sections that will be needed for the diffusion solver. Ensuring consistency in the cross sections used by OpenMC and the diffusion solver is necessary in order to make the solutions free from any cross section biasing. To discretize the problem within the 2D diffusion code, individual cells will be created along a square 3.6 x 3.6 cm mesh. Cells will be assigned an identity tag to confirm whether they are fuel, or moderator cells. This will allow for the L and F matrices to be constructed correctly. Lastly, to re-iterate, only a single neutron energy group will be used for this problem, as any additional energy groups will disrupt the symmetry in the L and F matrices. For a full list of nuclear data generated by the problem, refer to table 1.

TABLE 1
Nuclear Data from OpenMC.

Property	**Value**	**Std. Dev.**
Fuel Total Cross Section cm	0.557	0.006
Fuel Fission-Reproduction Cross Section cm	0.289	0.005
Fuel Scatter Cross Section cm	0.390	0.005
Fuel Absorption Cross Section cm	0.170	0.003
Fuel Di usion Coefficient cm	0.599	0.007
Moderator Total Cross Section cm	1.72	0.013
Moderator Absorption Cross Section cm	0.013	0.0001
Moderator Scatter Cross Section cm	1.71	0.014
Moderator Di usion Coefficient cm	0.194	0.004

To conclude the discussion regarding the problem setup, the calculated K_{eff} value found in OpenMC was approximately 1.007. This value will be used to gauge the accuracy of each iterative solver on the convergence of the k-eigenvalue. The following section will showcase the results of each iterative method, and will look to determine if the LOBPCG algorithm shows the expected reduction in iterations, and improved convergence properties.

4. Results and Analysis. Once the MC reference solution was generated in OpenMC, an analysis of the data structures of the L and F matrices was done to verify the ability of the LOBPCG method to be used. In addition, the structure of matrix L will be used to later determine a good preconditioning matrix T. Figures 2 and 3 display spy plots of the structures of matrices L and F respectively, after the finite difference diffusion scheme has been applied to the problem.

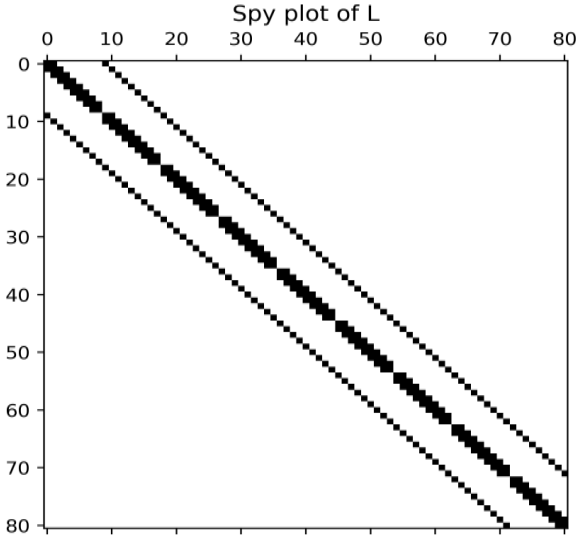

FIG 2. *Spy plot of matrix L.*

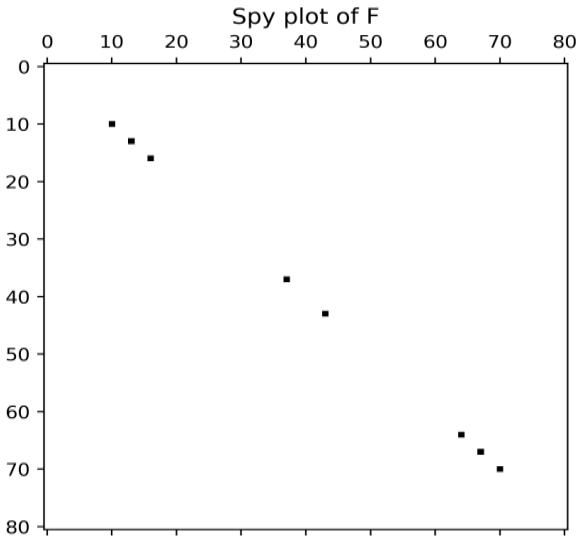

FIG 3. *Spy plot of matrix .*

The structures shown in figures 2 and 3 describe a significant amount of the problem. The tri-diagonal structure is easily seen for the diffusion and collision matrix, L. In addition, two other bands are seen in matrix L for the second spatial dimension of the finite difference scheme. On the other hand, the fission source matrix, F, is only filled with non-zeros when the central cell of the difference scheme is within designated nuclear fuel. Both matrices have a symmetric structure, which will be necessary to utilize the LOBPCG algorithm.

4.1. Results Without Preconditioning. The initial results of the LOBPCG implementation were conducted without a preconditioning matrix applied to the system. An identity matrix was used in the function as a stand-in so that the results of the iterative method would produce a result in its most bare form. Two metrics are of primary concern, the number of iterations to converge to the K_{eff} eigenvalue, as well as the overall accuracy of that value when compared to the MC reference solution generated by OpenMC. Table 2 summarizes the results of the initial tests when run with a mesh size of 0.4 x 0.4 cm. The initial guess of the k-eigenvalue for IPI was set to 1.02, which is a common guess for close-to-critical reactor systems.

TABLE 2
Initial Iterative Method Results for the k eigenvalue Problem.

Method	**K-Eigenvalue**	**Iterations**	**Eigenvalue Error ( )**
OpenMC	1.007	N/A	N/A
IPI	0.98812	7	1.87
LOBPCG	0.99352	3	1.33

The initial results shown in table 2 display a few important things. While the IPI method, as shown in algorithm 2.1, is highly optimized for nuclear engineering k-eigenvalue problems, it still takes around 7 iterations for it to converge. This is in contrast to the LOBPCG method, which takes only 3 iterations to converge. In

addition, a more accurate k-eigenvalue approximation is made when the LOBPCG method is used as opposed to the IPI method. To further this analysis, I decided to conduct a mesh sensitivity analysis to determine if smaller mesh sizes, and resulting denser matrices, have an effect on the accuracy of the k-eigenvalue. Figure 4 displays the results of this sensitivity analysis.

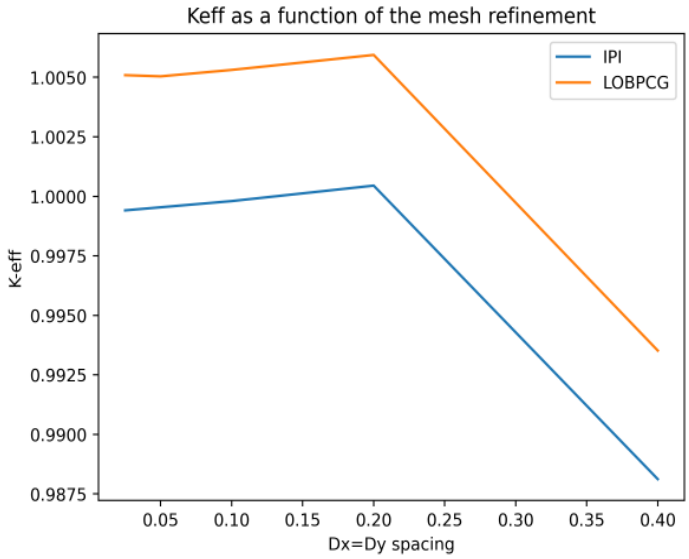

FIG 4. *Mesh Sensitivity Analysis of k eigenvalue Calculations.*

Based on the results of figure 4, it is clear that the LOBPCG method converges spatially to a k-eigenvalue that is closer to the reference solution (1.007) then the IPI scheme. In addition, it is capable of doing this in fewer iterations than the IPI method. However, it is important to note the specificity of this problem. Most diffusion codes that are still being used in nuclear analysis use multi-group energy approximations. Such approximations are not usable in the implementation of LOBPCG seen in this paper. The ORTHOMIN(1) algorithm [1] uses a similar procedure as LOBPCG, but has been adapted for multi-group use. In addition, ORTHOMIN(1) contains a multi-group in energy preconditioner which is capable of improving convergence properties of the algorithm [1]. The following section determines if a possible reduction in the number of iterations is possible by using a preconditioner matrix.

4.2. Preconditioned Results. The main goal of a preconditioner matrix, in broad terms, is to lower the condition number of the matrix used in the general eigenvalue problem [9]. As it is applied in algorithm 2.2, the preconditioner matrix is applied to the residual to accelerate convergence of the algorithm, as convergence is directly tied to the residual vector's changes over iterations. Efficient preconditioners for symmetric positive-definite matrices are well documented in engineering problems [4], and a simple inversion of the L matrix, L^{-1}, was used as the preconditioning matrix for this test problem. This matrix can easily be stored using a single inversion, and will not need to be re-computed during the iterative process. Unfortunately, the number of iterations needed to converge remained the same at 3. The most likely

cause of this phenomenon, is that the test is too simple for a preconditioner to make a sufficient reduction in iterations. Therefore, the results seen from the test with the preconditioned matrix are the exact same as those seen in Table 2.

5. Conclusions. For mono energetic neutron diffusion problems, the LOBPCG algorithm shows great potential for use as an iterative solver. When compared to the IPI method, the LOBPCG algorithm requires fewer iterations, and spatially converges to a closer k-eigenvalue then IPI when mesh size is reduced. Limitations exist in that the method requires a particular structure for k-eigenvalue problems, which is not possible to maintain when a common multi-group approximation is made to the problem. Algorithms which take into account multi-group structure and preconditioning [1] have been developed, and future work in this sphere should look to apply some of the block features of LOBPCG into algorithms like ORTHOMIN(1). Other work utilizing LOBPCG for nuclear applications [11] shows that broader analysis in nuclear science can be improved by the use of algorithms like LOBPCG. Possible future work that may be possible from this project, would include an expansion of the problem to be more realistic. A 17x17 assembly, like those actually used in PWRs, would be complex enough to potentially see benefits from preconditioning. For example, a mono energetic 17x17 problem could be run with a k-eigenvalue calculation. Despite the limitations of the test problem, it would be interesting to see if the preconditioning results of this more realistic problem could accelerate the convergence enough to make the mono energetic approximation worth using. To summarize, for the simple test problem in this analysis, the LOBPCG algorithm shows improved accuracy and convergence when compared to the IPI method. Future work should focus on larger problems, to determine if mono-energetic approximations needed for the symmetric matrices are sufficiently improved by LOBPCG.

REFERENCES

[1] A. Gupta, and R.S. Modak. "Krylov Sub-Space Methods for K-Eigenvalue Problem in 3-D Neutron Transport." Annals of Nuclear Energy 31, no. 18 (December 2004).

[2] R.N. Slaybaugh, M. Ramirez-Zweiger, T. Pandya, S. Hamilton, and T.M. Evans. "Eigenvalue Solvers for Modeling Nuclear Reactors on Leadership Class Machines." Nuclear Science and Engineering 190, no. 1 (April 3, 2018).

[3] E. Suetomi, and H. Sekimoto. "Conjugate Gradient like Methods and Their Application to Eigenvalue Problems for Neutron Di usion Equation." Annals of Nuclear Energy 18, no. 4 (January 1991).

[4] A.V. Knyazev, "Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method." SIAM Journal on Scientific Computing 23, no. 2 (January 2001).

[5] P.K. Romano, B.R. Herman, N.E. Horelik, B. Forget, K. Smith, and A.R. Siegel, "Progress and Status of the OpenMC Monte Carlo Code," Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Sun Valley, Idaho, (May 2013).

[6] M. Sapagovas, and V. Vileiniškis. "The Solution of Two-Dimensional Neutron Di usion Equation with Delayed Neutrons," Informatica 12, no. 2 (January 2001).

[7] B.Forget. "22.05 Lecture 18: Finite Di erence for the Di usion Equation", Nuclear Reactor Physics I. Massachusetts Institute of Technology, Cambridge. (November 2021).

[8] A.V. Knyazev. "Recent Implementations, Applications, and Extensions of the Locally Optimal Block Preconditioned Conjugate Gradient Method (LOBPCG)." arXiv, (August 2017).

[9] A.V. Knyazev. "PRECONDITIONED EIGENSOLVERS—AN OXYMORON?," Electronic Transactions on Numerical Analysis. no.7 (1998).

[10] J.A. Duersch, M. Shao, C. Yang, and M. Gu. "A Robust and Efficient Implementation of LOBPCG." SIAM Journal on Scientific Computing 40, no. 5 (January 2018).

[11] M. Shao, H.M. Aktulga, C. Yang, E.G. Ng, P. Maris, and J.P. Vary. “Accelerating Nuclear Configuration Interaction Calculations through a Preconditioned Block Iterative Eigensolver.” Computer Physics Communications. no. 222 (January 2018).