6-25-2010

Nonlinear acceleration methods for even-parity neutron transport

William Martin

William Joseph Martin

Candidate

Nuclear Engineering

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Cassiano R.E. de Oliveira, Chairperson

Anil K. Prinja

Evangelos A. Coutsias

NONLINEAR ACCELERATION METHODS FOR EVEN-PARITY NEUTRON TRANSPORT

BY

WILLIAM JOSEPH MARTIN

B.S., NUCLEAR ENGINEERING, UNIVERSITY OF NEW MEXICO, 2008

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

Nuclear Engineering

The University of New Mexico
Albuquerque, New Mexico

May, 2010

DEDICATION

To my wife and best friend, Stephanie, for always being there and keeping me positive through it all–a simple thanks will never be enough. Also, I owe everything to my mom and dad for making me who I am today.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Cassiano de Oliveira, for his guidance and assistance with this work over the last two years. You have without question made me a better engineer.

I owe my gratitude to Dr. Ryosuke Park for all his help with understanding the big picture in this research. Your help was invaluable and without it this work would not have been possible.

I would also like to show my appreciation to Dr. Anil Prinja and Dr. Evangelos Coutsias for not only serving on my committee but also for teaching me many of the subjects that went into this work.

NONLINEAR ACCELERATION METHODS FOR EVEN-PARITY NEUTRON TRANSPORT

BY

WILLIAM MARTIN

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

Nuclear Engineering

The University of New Mexico
Albuquerque, New Mexico

May, 2010

NONLINEAR ACCELERATION METHODS FOR EVEN-PARITY NEUTRON TRANSPORT

by

William Joseph Martin

B.S., Nuclear Engineering, University of New Mexico, 2008

M.S., Nuclear Engineering, University of New Mexico, 2010

ABSTRACT

Convergence acceleration methods for even-parity transport are being developed that have the potential to speed up transport calculations and provide a natural avenue for an implicitly coupled multiphysics code. An investigation was performed into the acceleration properties of the introduction of a nonlinear quasi-diffusion-like tensor in linear and nonlinear solution schemes. Although poor numerical properties prohibit the direct matrix solution of this reduced system, using it as a preconditioner for the conjugate gradients method proves highly efficient and effective and using it directly in a nonlinear solution scheme proves practical but costly. The results for the linear and nonlinear cases serve as the basis for further research into the application in a full three-dimensional spherical-harmonics even-parity transport code. Once moved into the nonlinear solution scheme, the implicit coupling of the convergence accelerated transport method into codes for other physics can be done seamlessly, providing an efficient, fully implicitly coupled multiphysics code with high order transport.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

Even-parity, or second order, neutron transport has been used in a limited capacity historically due to advantages and popularity of other deterministic methods. Although it has not been widely used in the United States, the numerical advantages for even-parity transport are plenty, especially when using higher-level unstructured discretization methods like finite elements. The ability for a nuclear system to be easily and quickly modeled using unstructured deterministic methods (e.g. finite elements) can bring nuclear modeling and design to new levels. By exploiting the ease of modeling complicated geometries as with Monte Carlo methods with the speed and precision of structured deterministic models, unstructured deterministic radiation transport has the potential to closely model real-world problems with lower computational costs than other methods.

The numerical advantages of the even-parity neutron transport method show due to its structure and use of off-the-shelf numerical solvers—specifically preconditioned conjugate gradients. Since these solution methods are general, improvements specifically for neutron transport problems can be made. This has led to the current investigation into acceleration of the method to improve its numerical performance. One such acceleration scheme being introduced includes the use of a generalized quasi-diffusion-like tensor form of the matrix, which in the application, has use in both linear and nonlinear solution environments. These methods are an extension of previous quasi-diffusion work by Gol'din (1964), Anistratov (2006), and others where the method has been extended for higher order moments of the spherical harmonics even-parity transport method. An advantage to this extension into even-parity transport as opposed to other transport methods is the fact that no accuracy is lost in the derivation.

This research is a feasibility study for the acceleration method and therefore is based on a simple form of the even-parity transport equation: fixed source, one-dimensional planar geometry, isotropic scattering, and one energy group. This is done without loss of generality from the full three-dimensional spherical harmonics second order transport method to allow it to be extended. The problem can be cast in two ways, explicit linear with updating or implicit nonlinear. When the problem is cast as linear, Krylov methods like generalized minimal residuals (GMRES) and preconditioned conjugate gradients (PCG) methods are used. This extended quasi-diffusion-like tensor is then used either for solution as the primary matrix (in GMRES) or as a preconditioner (in PCG). At each iteration the acceleration tensor is updated with the current flux to ensure proper solution. Since the equation is nonlinear in its nature, the problem can also be solved using different Newton methods or other nonlinear solvers. These methods take advantage of the fact that both sides of the equation depend on the same variable and in general can converge in only a few outer nonlinear iterations. Additionally, in being solved this way, the problem can be seamlessly integrated into multiphyiscs codes. This would create an implicit multiphysics code that couples other physics to high order neutron transport.

The next chapter will give a brief background on the even-parity method for one energy group and one dimension. Additionally, this chapter will give the current solution schemes for this equation. The third chapter will describe the accelerated methods and their theoretical background. The numerical results and discussion of these methods on several example problems will immediately follow the theory. Finally, conclusions and future work will be discussed.

CHAPTER 2. EVEN-PARITY NEUTRON TRANSPORT THEORY

Background

Before delving into the quasi-diffusion-like tensor and its application, it is worthwhile to look at the derivation and details of the second order, or even-parity, neutron transport equation.

To begin the derivation, consider a point in a six-dimensional phase space given by the following: its spatial position relative to a reference point, $\vec{r} = (x, y, z) = (r, z, \theta) = (r, \varphi, \theta)$; the unit direction its moving with respect to that reference, $\vec{\Omega} = (\theta, \varphi) = \left(\Omega_x, \Omega_y, \Omega_z\right)$; the energy at the point , E ; and all at a given time, t . These six dimensions uniquely describe any given point in the so called "phase space" of the problem for a given time. Next, consider a small phase space volume around the point which spans from $\vec{r}$ to $\vec{r} + d\vec{r}$ in space, from $\vec{\Omega}$ to $\vec{\Omega} + d\vec{\Omega}$ in directional angle, and from E to $E + dE$ in energy all at time t. In order to determine the distribution of neutrons in the entire phase space, it is the straightforward task of determining the number of neutrons in each phase space volume within the entire phase space and then summing over all volumes. This distribution in the whole space, given as the angular flux $\vec{\psi}\left(\vec{r}, \vec{\Omega}, E, t\right)$, is the ultimate goal of the nuclear modeling efforts as it is the basis for subsequent calculations like dose rates, shielding requirements, and nuclear burn-up.

To begin, the general form of the neutron transport equation (Lewis and Miller 1993) is

$$\begin{gathered}\frac{1}{v}\frac{\partial\psi\left(\vec{r},\vec{\Omega},E,t\right)}{\partial t}+\vec{\Omega}\cdot\vec{\nabla}\psi\left(\vec{r},\vec{\Omega},E,t\right)+\Sigma_t\left(\vec{r},E\right)\psi\left(\vec{r},\vec{\Omega},E,t\right)= \\ \int\int_{4\pi}\Sigma_s\left(\vec{r},\vec{\Omega}'\cdot\vec{\Omega},E'\to E\right)\psi\left(\vec{r},\vec{\Omega}',E',t\right)dE'd\vec{\Omega}'+S\left(\vec{r},\vec{\Omega},E,t\right)\end{gathered}, \qquad (2\text{-}1)$$

where $\Sigma_t\left(\vec{r},E\right)$ is the total neutron macroscopic cross section, v is the neutron speed, $\Sigma_s\left(\vec{r},\vec{\Omega}'\cdot\vec{\Omega},E'\to E\right)$ is the double differential macroscopic scattering cross section from any other phase space volume into the current one (with rotational symmetry), $S\left(\vec{r},\vec{\Omega},E,t\right)$ is a fixed source, and $\psi\left(\vec{r},\vec{\Omega},E,t\right)$ is the angular flux.

The terms on the left hand side of the equation are the time change and loss terms, either by particles streaming out of or by removal from the phase space volume. The terms on the right hand side of the equation are the production terms: the scattering into and the external source in the phase space volume. It should be noted that when rearranged, the time rate of change of the neutron population is equal to the production minus the losses as is logical.

Although this is a general form of the neutron transport equation, it must be reduced to a simpler form in order to be solved on a computer. Perhaps the most popular of these methods is the S_n or discrete ordinates method. This method reduces the equation by using a given set of discrete angular ordinates to reduce the scattering integral on the right hand side of the equation to an easier to compute summation. This method then allows for a simple solution of the equations by a “marching” scheme through the spatial grid. Although very popular and powerful, the S_n method can require a large number of iterations to converge to a solution in certain cases if not accelerated. Another method, the P_n method, assumes the independence of the angular variable and expands its solution in spherical harmonics (or Legendre polynomials for one dimensional models where it is then equivalent to the S_n method). The P_n method is again powerful but can prove to be cumbersome in its solution due to the size and structure of the matrices. This has caused the method to be historically unattractive. A third method, the integral equation method, reduces the equation by taking the integral of the whole equation, thus getting rid of the divergence term on the left hand side of the equation. The limitation with this method is that the solution is expensive as all points in the solution are linked to all others leading to a full matrix. This can thus be slow and computationally expensive. The problem also reduces the angular flux to the scalar flux, which could be a desirable quality since that is the most desired quantity for nuclear design and modeling.

The fourth deterministic method, and the one that is discussed in this thesis, is the even-parity or second order method. To begin the derivation of this method, the angular flux is split into two portions, the even-parity and the odd-parity fluxes (Lewis and Miller 1993).

$$\psi\left(\vec{r},\vec{\Omega},E,t\right)=\psi^{+}\left(\vec{r},\vec{\Omega},E,t\right)+\psi^{-}\left(\vec{r},\vec{\Omega},E,t\right) \tag{2-2}$$

These fluxes are defined by the following:

$$\psi^{+}\left(\vec{r},\vec{\Omega},E,t\right)=\frac{1}{2}\left[\psi\left(\vec{r},\vec{\Omega},E,t\right)+\psi\left(\vec{r},-\vec{\Omega},E,t\right)\right], \tag{2-3}$$

$$\psi^{-}\left(\vec{r},\vec{\Omega},E,t\right)=\frac{1}{2}\left[\psi\left(\vec{r},\vec{\Omega},E,t\right)-\psi\left(\vec{r},-\vec{\Omega},E,t\right)\right]. \tag{2-4}$$

It is useful to note additionally that, as their names imply, the even- and odd-parity fluxes have the property:

$$\psi^{+}\left(\vec{r},\vec{\Omega},E,t\right)=\psi^{+}\left(\vec{r},-\vec{\Omega},E,t\right), \tag{2-5}$$

$$\psi^-\left(\vec{r},\vec{\Omega},E,t\right) = -\psi^-\left(\vec{r},-\vec{\Omega},E,t\right). \tag{2-6}$$

By these definitions, the scalar flux and the current are given simply by only one of the components of the flux.

$$\varphi\left(\vec{r},E,t\right) = \int_{4\pi}\psi^+\left(\vec{r},\vec{\Omega},E,t\right)d\vec{\Omega} \tag{2-7}$$

$$\vec{J}\left(\vec{r},E,t\right) = \int_{4\pi}\vec{\Omega}\psi^-\left(\vec{r},\vec{\Omega},E,t\right)d\vec{\Omega} \tag{2-8}$$

In order to now transform the general transport equation, Equation (2-1), into even-parity form, Equation (2-1) is evaluated for $-\vec{\Omega}$ (omitting the energy and time dependencies for brevity and assuming steady state conditions).

$$-\vec{\Omega}\cdot\vec{\nabla}\psi\left(\vec{r},-\vec{\Omega}\right)+\Sigma_t\left(\vec{r}\right)\psi\left(\vec{r},-\vec{\Omega}\right) = \Sigma_s\left(\vec{r}\right)\varphi\left(\vec{r}\right)+S\left(\vec{r},-\vec{\Omega}\right) \tag{2-9}$$

At this point, the isotropic scattering assumption has been taken and the scalar flux results. This simplification is made for the purposes of this investigation, but as shown by de Oliveira (1987) can be reintroduced into to this method in a straightforward fashion. Additionally the source term is assumed to be isotropic but could be easily re-extended after this study. By adding this equation to the equivalently simplified Equation (2-1) for $+\vec{\Omega}$, the following is produced.

$$\vec{\Omega}\cdot\vec{\nabla}\psi^-\left(\vec{r},\vec{\Omega}\right)+\Sigma_t\left(\vec{r}\right)\psi^+\left(\vec{r},\vec{\Omega}\right) = \Sigma_s\left(\vec{r}\right)\varphi\left(\vec{r}\right)+S\left(\vec{r}\right) \tag{2-10}$$

When the same two equations are subtracted, the result is

$$\vec{\Omega}\cdot\vec{\nabla}\psi^+\left(\vec{r},\vec{\Omega}\right)+\Sigma_t\left(\vec{r}\right)\psi^-\left(\vec{r},\vec{\Omega}\right) = 0. \tag{2-11}$$

By now solving Equation (2-11) for the odd-parity flux and substituting it into Equation (2-10), the general form of the second order transport equation with isotropic scattering (Lewis and Miller 1993) is formed.

$$
\begin{aligned}
-\vec{\Omega}\cdot\vec{\nabla}\frac{1}{\Sigma_t\left(\vec{r},E\right)}\vec{\Omega}\cdot\vec{\nabla}\psi^+\left(\vec{r},\vec{\Omega},E,t\right)+\Sigma_t\left(\vec{r},E\right)\psi^+\left(\vec{r},\vec{\Omega},E,t\right)= \\
\Sigma_s\left(\vec{r},E\right)\varphi\left(\vec{r},E,t\right)+S\left(\vec{r},E,t\right)
\end{aligned}
\qquad (2\text{-}12)
$$

There are several things about the even-parity transport equation that should be detailed. The first of which is that in the transformation, a first-order differential initial value problem has been transformed to a second-order boundary value problem. This means that now instead of needing to know the initial state of the system, it’s only necessary to know the boundary conditions (which will be discussed later in the derivation). Also, this equation now has the total cross-section in the denominator of the streaming term. It is therefore not capable of handling true void problems and can become ill-conditioned when the cross-section is nearly zero since that term tends to infinity while others (except the source) tend to zero. It is still of course an integro-differential equation and requires simplification in order to be solved easily.

As mentioned briefly, for the current application, assumptions are made to simplify the formulation into a form that permits a less complicated solution scheme. These are done without loss of generality from the full three dimensional spherical harmonics even-parity transport equation. The reduced problem has the following assumptions: one-dimensional, steady-state, mono-energetic, isotropic scattering, isotropic source, and non-multiplying media. These assumptions are such that they allow the method to be seamlessly extrapolated back to the full case.

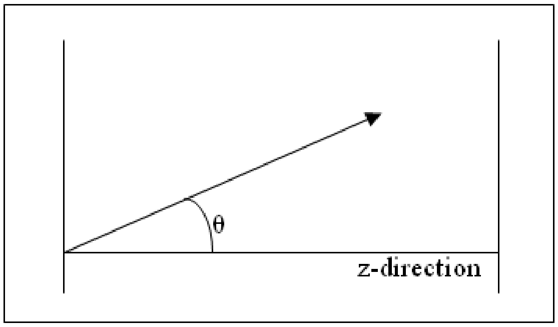


Figure 1: One-Dimensional Coordinates

Taking into consideration these simplifications listed above, we have reduced the even-parity transport equation in one-dimensional planar geometry:

$$\mu^2 \frac{\partial}{\partial z}\left(-\frac{1}{\Sigma_t}\frac{\partial \psi^+(z,\mu)}{\partial z}\right)+\Sigma_t\psi^+(z,\mu)=\Sigma_s\varphi(z)+S(z) \tag{2-13}$$

where μ is the cosine of the angle θ as given in **Figure 1**.

In order to solve this equation, the two independent variables must be transformed into something that is amenable to numerical solution, so an approximation is made in both space and angle. It is best to show this derivation by using the functional representation of the 1D even-parity transport equation, Equation (2-13). (Lewis and Miller, 1993)

$$\begin{aligned} F\left[\psi^+(z,\mu)\right] = \int_{z_L}^{z_R}\left\{\int_{-1}^{1}\left[\frac{\mu^2}{\Sigma_T(z)}\left(\frac{\partial\psi^+}{\partial z}\right)^2+\Sigma_t(z)\left(\psi^+\right)^2\right]\frac{d\mu}{2}-\Sigma_s(z)\varphi^2-2\varphi S\right\}dz \\ +\int_{-1}^{1}|\mu|\left(\psi^+\right)^2\frac{d\mu}{2}\Big|_{z=z_R} \end{aligned} \tag{2-14}$$

It is important to note a few key items in this equation. First, a scaling factor of one-half is used for the angular flux integral, e.g.

$$\varphi(z)=\int_{-1}^{1}\psi^+(z,\mu)\frac{d\mu}{2}. \tag{2-15}$$

It is important to recall that the even-parity flux is given with the splitting as shown in Equations (2-2) through (2-4). Additionally, for the derivation, a reflective boundary condition is taken at the left edge and a vacuum boundary condition is taken at the right edge. As will be shown with the final derived product, these boundary conditions can be interchanged simply by including or not including the boundary condition for the vacuum since reflection is natural.

If we require the functional to be stationary with respect to variations of the form

$$\psi^{+}\left(z,\mu\right)=\psi_{0}^{+}\left(z,\mu\right)+\delta\psi^{+}\left(z,\mu\right), \tag{2-16}$$

then we arrive at the Euler-Lagrange equations.

$$-\mu^{2}\frac{\partial}{\partial z}\frac{1}{\Sigma_{t}\left(z\right)}\frac{\partial}{\partial z}\psi_{0}^{+}\left(z,\mu\right)+\Sigma_{t}\left(z\right)\psi_{0}^{+}\left(z,\mu\right)=\Sigma_{s}\left(z\right)\varphi_{0}\left(z\right)+S\left(z\right) \tag{2-17}$$

$$\mu\frac{\partial}{\partial z}\psi_{0}^{+}\left(z,\mu\right)\Big|_{z=z_{L}}=0 \tag{2-18}$$

$$\psi^{+}\left(z_{R},\mu\right)\pm\frac{\mu}{\Sigma_{t}\left(z_{R}\right)}\frac{\partial}{\partial z}\psi^{+}\left(z,\mu\right)\Big|_{z=z_{R}}=0 \qquad \mu\neq 0 \tag{2-19}$$

Equation (2-18) is the natural, reflective boundary condition for the 1D even-parity derivation and Equation (2-19) is the vacuum boundary condition.

The spatial discretization is done by finite elements, where we define the expansion using a spatial trial function as

$$\psi^{+}\left(z,\mu\right)\approx\sum_{j=1}^{J}h_{j}\left(z\right)\psi_{j}^{+}\left(\mu\right) \tag{2-20}$$

where the trial function is given by

$$h_j\left(z\right)=\begin{cases}0, & z\le z_{j-1}\\ \dfrac{z-z_{j-1}}{z_j-z_{j-1}}, & z_{j-1}\le z\le z_j\\ \dfrac{z_{j+1}-z}{z_{j+1}-z_j}, & z_j\le z\le z_{j+1}\\ 0, & z_{j+1}\le z\end{cases},\qquad j=1,2,...,J \tag{2-21}$$

By substituting this expansion into Equation (2-14), we arrive at the reduced function in matrix/vector notation. A single bar denotes a vector while a double bar denotes a matrix.

$$F\left[\underline{\psi}^{+}\left(\mu\right)\right]=\frac{1}{2}\int_{-1}^{1}\underline{\psi}^{+}\left(\mu\right)^{T}\ \underline{\underline{A}}\left(\mu\right)\underline{\psi}^{+}\left(\mu\right)d\mu-\underline{\varphi}^{T}\underline{\underline{B}}\underline{\varphi}-2\underline{\varphi}^{T}\underline{s} \tag{2-22}$$

The variables in the reduced functional are given by the following definitions. Note the half-range simplification for the even integral in the scalar flux.

$$\underline{\varphi}=\frac{1}{2}\int_{-1}^{1}\underline{\psi}^{+}\left(\mu\right)d\mu=\int_{0}^{1}\underline{\psi}^{+}\left(\mu\right)d\mu \tag{2-23}$$

$$\underline{\underline{A}}\left(\mu\right)=\int_{z_L}^{z_R}\left[\frac{\mu^2}{\Sigma_t\left(z\right)}\left(\frac{d}{dz}\underline{h}\right)\left(\frac{d}{dz}\underline{h}^{T}\right)+\Sigma_t\left(z\right)\underline{h}\underline{h}^{T}\right]dz+\left|\mu\right|\underline{h}\left(z_R\right)\underline{h}\left(z_R\right)^{T} \tag{2-24}$$

$$\underline{\underline{B}}=\int_{z_L}^{z_R}\Sigma_s\left(z\right)\underline{h}\underline{h}^{T}dz \tag{2-25}$$

$$\underline{s}=\int_{z_L}^{z_R}\underline{h}Sdz \tag{2-26}$$

By requiring the functional to be stationary as given above, the previous equations yield the following matrix integral equation.

$$\underline{\underline{A}}\left(\mu\right)\underline{\psi}^{+}\left(\mu\right)=\underline{\underline{B}}\underline{\varphi}+\underline{s} \tag{2-27}$$

Now it is possible to move onto the angular discretization. This is done by using angular trial functions, similar to the spatial discretization. An even Legendre expansion, comprising half the angular space, is used to define the angular distribution.

$$\psi^{+}\left(z,\mu\right)\approx\sum_{n}\left(2n+1\right)\psi_{n}^{+}\left(z\right)P_{n}\left(\mu\right),\quad n=0,2,4,...,N-1 \tag{2-28}$$

The odd-parity flux would be indexed by the odd numbers omitted in the even-parity expansion to ensure Equation (2-2) is satisfied.

Substituting this approximation into the functional, Equation (2-14), we arrive at the reduced functional for the angular discretization.

$$F\left[\underline{\psi}^{+}\right]=\int_{z_L}^{z_R}\left[\begin{array}{l}\dfrac{1}{2\Sigma_t\left(z\right)}\left(\dfrac{d}{dz}\underline{\psi}^{+}\right)^{T}\displaystyle\int_{-1}^{1}\mu^{2}\left(2n+1\right)\left(2n'+1\right)\underline{P}_{n}\left(\mu\right)\underline{P}_{n'}\left(\mu\right)^{T}d\mu\left(\dfrac{d}{dz}\underline{\psi}^{+}\right)\\ \qquad+\dfrac{1}{2}\Sigma_t\left(z\right)\underline{\psi}^{+T}\displaystyle\int_{-1}^{1}\left(2n+1\right)\left(2n'+1\right)\underline{P}_{n}\left(\mu\right)\underline{P}_{n'}\left(\mu\right)^{T}d\mu\,\underline{\psi}^{+}\\ \qquad-\Sigma_s\left(z\right)\underline{\psi}^{+T}\displaystyle\int_{0}^{1}\left(2n+1\right)\underline{P}_{n}\left(\mu\right)d\mu\int_{0}^{1}\left(2n'+1\right)\underline{P}_{n}\left(\mu'\right)^{T}d\mu'\,\underline{\psi}^{+}\\ \qquad-\underline{\psi}^{+T}S\left(z\right)\displaystyle\int_{-1}^{1}\left(2n+1\right)\underline{P}_{n}\left(\mu\right)d\mu\end{array}\right]dz \tag{2-29}$$
$$+\underline{\psi}^{+}\left(z_R\right)^{T}\int_{0}^{1}\left(2n+1\right)\left(2n'+1\right)\mu P_{n}\left(\mu\right)\underline{P}_{n'}\left(\mu\right)^{T}d\mu\,\underline{\psi}^{+}\left(z_R\right)$$

This expansion allows for each moment of the flux to build upon the last, starting with the first moment of the angular flux being equal to the scalar flux. By combining the spatial finite elements and the Legendre polynomials, we arrive at the final form of the even-parity transport equation, in matrix solution form given by Lewis and Miller (1993).

$$\left[G_t\int_{-z_L}^{z_R}\frac{1}{\Sigma_t}\left(\frac{dh_j}{dz}\right)\left(\frac{dh_{j'}}{dz}\right)dz+G_c\int_{-z_L}^{z_R}\Sigma_t h_j h_{j'}dz-G_s\int_{-z_L}^{z_R}\Sigma_s h_j h_{j'}dz-G_v\delta_{jJ}\right]\psi_{n,j}^{+} \tag{2-30}$$
$$=\int_{-z_L}^{z_R}h_j S_n\left(z\right)dz$$

$$G_t = \int_0^1 \mu^2 P_n(\mu) P_{n'}(\mu)(2n+1)(2n'+1)\,d\mu, \qquad n,n' = 0,2,...,N-1 \qquad (2\text{-}31)$$

$$G_s = \int_0^1 P_n(\mu)(2n+1)\,d\mu \int_0^1 P_{n'}(\mu)(2n'+1)\,d\mu, \qquad n,n' = 0,2,...,N-1 \qquad (2\text{-}32)$$

$$G_c = \int_0^1 P_n(\mu) P_{n'}(\mu)(2n+1)(2n'+1)\,d\mu, \qquad n,n' = 0,2,...,N-1 \qquad (2\text{-}33)$$

$$G_v = \int_0^1 \mu P_n(\mu) P_{n'}(\mu)(2n+1)(2n'+1)\,d\mu, \qquad n,n' = 0,2,...,N-1 \qquad (2\text{-}34)$$

These coupled equations provide for a straightforward solution scheme, one that is done traditionally using preconditioned conjugate gradients. The key advantages to this solution is that the primary matrix is block tridiagonal, with the submatricies being tridiagonal as well, due to both the Legendre expansion and finite element method.

For the angular matrices, it is helpful to determine the contribution of each term, from Equation (2-31) to (2-34). This is done by performing the integration for each equation using the well known properties of Legendre polynomials (Lewis and Miller 1993).

$$\begin{aligned} G_t = {} & \frac{(2n+1)(n'+1)(n'+2)}{(2n'+5)(2n'+3)}\delta_{n,n'+2} + \left(\frac{(2n+1)(n'+1)^2}{(2n'+1)(2n'+3)} + \frac{(2n+1)n'^2}{(2n'+1)(2n'-1)} \right)\delta_{n,n'} \\ & + \frac{(2n+1)(n'-1)n'}{(2n'-3)(2n'-1)}\delta_{n,n'-2}, \qquad n,n' = 0,2,...,N-1 \end{aligned} \qquad (2\text{-}35)$$

$$G_s = \delta_{0,0}, \qquad n,n' = 0,2,...,N-1 \qquad (2\text{-}36)$$

$$G_c = (2n+1)\delta_{n,n'}, \qquad n,n' = 0,2,...,N-1 \qquad (2\text{-}37)$$

$$G_v = \left(2n+1\right)\left(n'+1\right)\left\{ \frac{\left(-1\right)^{\left[n+\left(n'+1\right)+1\right]/2} n!\left(n'+1\right)!}{2^{\left[n+\left(n'+1\right)-1\right]}\left[n-\left(n'+1\right)\right]\left[n+\left(n'+1\right)+1\right]\left[\left(\frac{n}{2}\right)!\right]^2\left[\left(\frac{n'+1-1}{2}\right)!\right]^2} \right\} \delta_{n,n'}^{even}$$

$$+\left(2n+1\right)n'\left\{ \frac{\left(-1\right)^{\left[n+\left(n'-1\right)+1\right]/2} n!\left(n'-1\right)!}{2^{\left[n+\left(n'-1\right)-1\right]}\left[n-\left(n'-1\right)\right]\left[n+\left(n'-1\right)+1\right]\left[\left(\frac{n}{2}\right)!\right]^2\left[\left(\frac{n'-1-1}{2}\right)!\right]^2} \right\} \delta_{n,n'}^{even},$$

$$n,n' = 0,2,\ldots,N-1 \qquad \text{(2-38)}$$

Equations (2-35) through (2-38) give these results. Note that only G_t , the streaming expansion, and G_v, the vacuum boundary expansion, contribute to off-diagonal terms. G_s , the scattering expansion, and G_c, the capture expansion, are strictly diagonal. Here, $\delta_{n,n'}$ is the Kronecker delta which is defined as

$$\delta_{n,n'} = \begin{cases} 1, & n = n' \\ 0, & n \neq n' \end{cases}. \qquad \text{(2-39)}$$

Therefore, the angular matrix when formed with these results is tridiagonal with the exception of the vacuum boundary term where the matrix is full.

Traditional Numerical Solution

One of the advantages of even-parity transport methods is the straightforward solution of the system of linear equations. As mentioned above, the matrix is block tridiagonal. Additionally, the matrix can be shown to be symmetric. Because of this mundane nature, the system of equations lends itself to off-the-shelf symmetric linear solvers, notably the preconditioned conjugate gradients (PCG) method. As the case has been simplified to one-dimension where the S$_n$ method is a lower triangular system, it is not a contention of this thesis that the even-parity method is faster—as the matrix inversion of a lower triangular system is much simpler than a tridiagonal system. However, if expanded to more than one dimension, that is not the case and it could be seen which method is faster.

It will aid in the description if the matrix-vector product is shown in matrix form.

$$A\psi = S \tag{2-40}$$

Note that the matrix has been named A, the even-parity flux has been transformed for brevity from ψ^+ to ψ, and the source has been named S.

The space-angle matrix, *A*, can be more explicitly written as the following $\left(\left(N-1\right)/2+1\right)\times\left(\left(N-1\right)/2+1\right)$ block angular sparse matrix form,

$$\begin{pmatrix} A^{0,0} & A^{0,2} & 0 & \cdots & 0 \\ A^{2,0} & A^{2,2} & A^{2,4} & \cdots & 0 \\ 0 & A^{4,2} & A^{4,4} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & A^{N-3,N-1} \\ 0 & 0 & 0 & A^{N-3,N-1} & A^{N-1,N-1} \end{pmatrix} \begin{pmatrix} \psi^0 \\ \psi^2 \\ \psi^4 \\ \vdots \\ \psi^{N-1} \end{pmatrix} = \begin{pmatrix} S^0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2-41}$$

where N is the order of the angular expansion. Each $J\times J$ finite element sub-matrix and sub-vector has a form of,

$$A^{n,n'} = \begin{pmatrix} A^{n,n'}_{1,1} & A^{n,n'}_{1,2} & 0 & \cdots & 0 \\ A^{n,n'}_{2,1} & A^{n,n'}_{2,2} & A^{n,n'}_{2,3} & \cdots & 0 \\ 0 & A^{n,n'}_{3,2} & A^{n,n'}_{3,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & A^{n,n'}_{J-1,J} \\ 0 & 0 & 0 & A^{n,n'}_{J,J-1} & A^{n,n'}_{J,J} \end{pmatrix}, \quad \psi^n = \begin{pmatrix} \psi^n_1 \\ \psi^n_2 \\ \psi^n_3 \\ \vdots \\ \psi^n_J \end{pmatrix}, \quad S^n = \begin{pmatrix} S^n_1 \\ S^n_2 \\ S^n_3 \\ \vdots \\ S^n_J \end{pmatrix}, \tag{2-42}$$

where J is the number of spatial nodes. The superscript indices denote the angular matrix position and the subscript indices denote the spatial grid position.

Since the source is assumed to be isotropic, only the first angular moment of the source is present. This system of equations can be solved through many methods, including by direct and iterative methods. Although direct methods might be useful for small problems, they take $O\left[\left(\left((N-1)/2+1\right)\cdot J\right)^3\right]$ (Kelley 1987) operations and become prohibitively expensive for most practical problems. Since iterative methods generally take fewer operations to converge for large problems, these methods become appropriate for these problems.

The popular numerical method for this problem is the preconditioned conjugate gradients method. This method takes $O\left[2\left(\left((N-1)/2+1\right)\cdot J\right)^2+10\left((N-1)/2+1\right)\cdot J\right]$ per outer conjugate gradient iteration and is guaranteed to converge within $\left(\left((N-1)/2+1\right)\cdot J\right)$ iterations given exact arithmetic (Kelley 1987). Although theoretically convergence is guaranteed, due to numerical inaccuracies, there is no such guarantee when solving on a computer. This method will, in general, converge in fewer iterations than direct methods given the fact that generally only ill-conditioned problems will take the full or greater number of iterations. The preconditioner is generally based on the needs of the application and aids in this convergence.

A common preconditioner for these methods is the symmetric Gauss-Seidel preconditioner. This is an approximation to the full matrix and is given by

$$M=\left(L+D\right)D^{-1}\left(L+D\right)^T, \tag{2-43}$$

where L is the lower triangular portion and D is the diagonal of the full A matrix. This preconditioner can be easily introduced in parts as an incomplete LU solver, using a forward and backward sweep to get the solution vector.

Although the preconditioner adds up to $O\left[2\left(\left((N-1)/2+1\right)\cdot J\right)^2+2\left((N-1)/2+1\right)\cdot J\right]$ operations per iteration, the iteration count for convergence is greatly improved. These iterative methods allow for a moment by moment solution and are widely employed for the second order transport method. The moment by moment (MBM) preconditioning stage allows for only the angular matrices to be stored, saving on the cost of storage for this stage (Park and de Oliveira 2005). MBM also enables storage to be separate for the angular and spatial integrals, saving again on storage. Therefore the drawback of this method is the effort required to perform the forward/backward substitution for the preconditioning step in PCG. In total, when considering the sparse nature of the matrix, the current methods take $O\left[48\left((N-1)/2+1\right)\cdot J\right]$ operations per iteration. More details of the conjugate gradients method will be given below in Chapter 3.

CHAPTER 3. NUMERICAL ACCELERATION METHODS

Acceleration Method Description

The general methods for solution of the even-parity transport equation are moderately efficient, but methods are sought to accelerate the convergence of the problem. By accelerating these methods, modeling of complex systems that typically take many iterations in the even-parity method to converge can potentially be performed much quicker without loss of accuracy. Some of these problems include problems with low total interaction probabilities (low total cross-section) and those with highly directionally peaked angular fluxes as in shielding models. The method being proposed to accelerate uses a modified quasi-diffusion-like tensor approximation for the *A* matrix.

Given the one-dimensional even-parity transport equation above, Equation (2-13), renaming and combining terms produces the following matrix version of this equation.

$$-\mu^2 \frac{\partial}{\partial \mathrm{z}} G \frac{\partial \underline{\psi}^+}{\partial \mathrm{z}} + C\underline{\psi}^+ = \underline{S} \tag{3-1}$$

In this form of the equation

$$G = {}^{1}\!/\!{}_{\Sigma_t} \tag{3-2}$$

is the streaming operator,

$$C = \Sigma_t - \Sigma_s \tag{3-3}$$

is the collision operator for isotropic scattering, and S is the fixed source.

By utilizing orthogonality, and assuming the cross-sections to be constant in each (finite) element, this equation is once again transformed into the form

$$-\frac{\partial^2}{\partial \mathrm{z}^2}\left[\mu^2 G\underline{\psi}^+\right] + C\underline{\psi}^+ = \underline{S}\ . \tag{3-4}$$

Recall that introducing the Legendre expansions into the equation we arrive at the following set of $(N-1)/2+1$ linear equations.

$$-\frac{\partial^2}{\partial z^2}\left[\mu^2 G(z) P_n(\mu)\psi_n^+(z)\right] + C(z) P_n(\mu)\psi_n^+(z) = S_n(\mu,z), \qquad n = 0,2,...,N-1 \text{ (3-5)}$$

Computationally, this method is already an improvement over other transport methods since the matrix is block tridiagonal where the blocks are themselves tridiagonal. However, when large expansions are necessary for problems with peculiarities (like those previously mentioned), it would be advantageous to lower the computational effort required. To do so, a quasi-diffusion-like tensor is used to modify the streaming term of the even-parity equation. Note as shown in Equation (2-35), only the streaming term contributes the off-diagonal terms to the bulk of the matrix, so in modifying this term, computation gains can be made. By defining a tensor as,

$$E_{n,n'}(z) = \frac{\int_{-1}^{1} P_n(\mu)\mu^2 G(z) P_{n'}(\mu)\psi_{n'}^+(z)\,d\mu}{\int_{-1}^{1} P_n(\mu) P_{n'}(\mu)\psi_{n'}^+(z)\,d\mu}, \quad n,n' = 0,2,...,N-1, \qquad \text{(3-6)}$$

a new version of the one dimensional second order transport equation is produced. Although in one dimension the tensor is a scalar, the terminology is kept for consistency.

$$-\frac{\partial^2}{\partial z^2}\left[E_{n,n'} P_n(\mu)\psi_n^+(z)\right] + C_n(\mu) P_n(\mu)\psi_n^+(z) = S_n(\mu,z), \quad n,n' = 0,2,...,N-1 \text{ (3-7)}$$

This can be shown to be equivalent to Equation (3-5) by taking the difference of the two equations.

$$\begin{aligned}&-\frac{\partial^2}{\partial z^2}\left[E_{n,n'}\left(z\right)P_n\left(\mu\right)\psi_n^+\left(z\right)\right]+\frac{\partial^2}{\partial z^2}\left[\mu^2G\left(z\right)P_n\left(\mu\right)\psi_n^+\left(z\right)\right]+\\&\qquad\left[P_n\left(\mu\right)C\left(z\right)\psi_n^+\left(\mu,z\right)-P_n\left(\mu\right)C\left(z\right)\psi_n^+\left(z\right)\right]-\\&\qquad\left[S^+\left(\mu,z\right)-S^+\left(\mu,z\right)\right]\overset{?}{=}0,\qquad n,n'=0,2,...,N-1\end{aligned}\tag{3-8}$$

By cancelling like terms, the following identity is reached.

$$-\frac{\partial^2}{\partial z^2}\left[E_{n,n'}\left(z\right)P_n\left(\mu\right)\psi_n^+\left(z\right)\right]\overset{?}{=}-\frac{\partial^2}{\partial z^2}\left[\mu^2G\left(z\right)P_n\left(\mu\right)\psi_n^+\left(z\right)\right],\ n,n'=0,2,...,N-1\tag{3-9}$$

Therefore, the two statements that are being differentiated must be equal with the exception of constants for this statement to be true. By expanding the tensor to its full representation and cancelling the Legendre polynomial and flux terms, the following must be true.

$$\frac{\int_{-1}^{1}P_n\left(\mu'\right)\mu^2G\left(z\right)\psi^+\left(z,\mu'\right)d\mu'}{\int_{-1}^{1}P_n\left(\mu'\right)\psi^+\left(z,\mu'\right)d\mu'}\overset{?}{=}\mu^2G\left(z\right),\qquad n,n'=0,2,...,N-1\tag{3-10}$$

Multiplying the right hand side by the denominator of the left hand side results in similar statements on either side of the equation.

$$\int_{-1}^{1}P_n\left(\mu'\right)\mu'^2G\left(z\right)\psi_{n'}^+\left(z,\mu'\right)d\mu'\overset{?}{=}\mu^2G\left(z\right)\int_{-1}^{1}P_n\left(\mu'\right)\psi^+\left(z,\mu'\right)d\mu',n,n'=0,2,...,N-1\tag{3-11}$$

Now by multiplying each side by $P_{n'}\left(\mu\right)$ and integrating from -1 to 1 over μ, we arrive at the equality.

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1}P_{n'}\left(\mu\right)P_n\left(\mu'\right)\mu'^2G\left(z\right)\psi^+\left(z,\mu'\right)d\mu'd\mu=\\&\qquad\int_{-1}^{1}\int_{-1}^{1}P_{n'}\left(\mu\right)P_n\left(\mu'\right)\mu^2G\left(z\right)\psi^+\left(z,\mu'\right)d\mu'd\mu,\quad n,n'=0,2,...,N-1\end{aligned}\tag{3-12}$$

Through this equality, it is important to note that no assumptions are made and no accuracy is lost in the overall system. The advantage to the introduction of this term is the diagonalization of the streaming term in the angular matrix. Since the angular sub-matrix was tridiagonal and is now strictly diagonal, the number of operations per manipulation with this matrix is reduced by approximately one third. For cases with vacuum boundary conditions, the boundary angular matrices will still be full rank as defined by the term in Equation (2-38). No reduction is made on these terms for this current acceleration method. The angular sub-matrices are reduced from the typical form to the following, where the off-diagonal parts are essentially summed into the diagonal.

$$\tilde{A}_{l,l'} = \begin{pmatrix} A_{l,l'}^{0,0} + A_{l,l'}^{0,2}\dfrac{\psi_l^2}{\psi_l^0} & 0 & \cdots & 0 \\ 0 & A_{l,l'}^{2,0}\dfrac{\psi_l^0}{\psi_l^2} + A_{l,l'}^{2,2} + A_{l,l'}^{2,4}\dfrac{\psi_l^4}{\psi_l^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & A_{l,l'}^{N-1,N-3}\dfrac{\psi_l^{N-3}}{\psi_l^{N-1}} + A_{l,l'}^{N-1,N-1} \end{pmatrix} \tag{3-13}$$

$\tilde{A}$ in Equation (3-13) above represents the tensor modified A matrix where again for brevity $\psi^+ = \psi$.

To ensure that the problem is still equal to the original system, a simple exercise in matrix-vector multiplication is used as in the system solution. Given $A\psi = S$, $\tilde{A}\psi = S$ should hold as well, which does as shown below for a representative spatial matrix element.

$$\tilde{A}_{l,l'}\psi_l = \begin{pmatrix} \left(A_{l,l'}^{0,0} + A_{l,l'}^{0,2}\dfrac{\psi_l^2}{\psi_l^0}\right)\psi_l^0 \\ \left(A_{l,l'}^{2,0}\dfrac{\psi_l^0}{\psi_l^2} + A_{l,l'}^{2,2} + A_{l,l'}^{0,4}\dfrac{\psi_l^4}{\psi_l^2}\right)\psi_l^2 \\ \vdots \\ \left(A_{l,l'}^{N-1,N-3}\dfrac{\psi_l^{N-3}}{\psi_l^{N-1}} + A_{l,l'}^{N-1,N-1}\right)\psi_l^{N-1} \end{pmatrix} = \begin{pmatrix} A_{l,l'}^{0,0}\psi_l^0 + A_{l,l'}^{0,2}\psi_l^2 \\ A_{l,l'}^{2,0}\psi_l^0 + A_{l,l'}^{2,2}\psi_l^2 + A_{l,l'}^{2,4}\psi_l^4 \\ \vdots \\ A_{l,l'}^{N-1,N-3}\psi_l^{N-3} + A_{l,l'}^{N-1,N-1}\psi_l^{N-1} \end{pmatrix} = A_{l,l'}\psi_l \quad (3\text{-}14)$$

Even though the action on a vector by the reduced matrix has the same result, the tensor modified matrix is no longer symmetric in general. This can be seen by looking at a portion of the reduced matrix. For this snapshot, an angular expansion of $N = 3$ is used for simplicity without loss of generality.

$$\begin{pmatrix} \ddots & \vdots & \vdots & 0 \\ \cdots & \begin{pmatrix} A_{l,l'}^{0,0} + A_{l,l'}^{0,2}\dfrac{\psi_l^2}{\psi_l^0} & 0 \\ 0 & A_{l,l'}^{2,0}\dfrac{\psi_l^0}{\psi_l^2} + A_{l,l'}^{2,2} \end{pmatrix} & \begin{pmatrix} A_{l,l'+1}^{0,0} + A_{l,l'+1}^{0,2}\dfrac{\psi_{l+1}^2}{\psi_{l+1}^0} & 0 \\ 0 & A_{l,l'+1}^{2,0}\dfrac{\psi_{l+1}^0}{\psi_{l+1}^2} + A_{l,l'+1}^{2,2} \end{pmatrix} & \cdots \\ \cdots & \begin{pmatrix} A_{l,l'+1}^{0,0} + A_{l,l'+1}^{0,2}\dfrac{\psi_l^2}{\psi_l^0} & 0 \\ 0 & A_{l,l'+1}^{2,0}\dfrac{\psi_l^0}{\psi_l^2} + A_{l,l'+1}^{2,2} \end{pmatrix} & \begin{pmatrix} A_{l+1,l'+1}^{0,0} + A_{l+1,l'+1}^{0,2}\dfrac{\psi_{l+1}^2}{\psi_{l+1}^0} & 0 \\ 0 & A_{l+1,l'+1}^{2,0}\dfrac{\psi_{l+1}^0}{\psi_{l+1}^2} + A_{l+1,l'+1}^{2,2} \end{pmatrix} & \cdots \\ 0 & \vdots & \vdots & \ddots \end{pmatrix}$$

(3-15)

In order for the matrix-vector multiplication $\tilde{A}\psi = S$ to hold, the off diagonal terms are forced to be asymmetric when the ratio of fluxes is not exactly 1. The implications of this will be discussed below with the solution method schemes used.

Acceleration Using Linear Solution Schemes

There are several ways that are being proposed in order to take advantage of this reduced matrix, each with its own advantages and disadvantages. Of the linear solution methods, one solves the reduced system by itself using GMRES, while the other uses it as a preconditioner in preconditioned conjugate gradients (PCG).

The first method is the direct use of this reduced matrix as $\tilde{A}\psi = S$. This method treats the system as linear, which must have a way to treat the nonlinear nature of the reduced matrix – this is done as an explicit update after each iteration. Since a linear method is desired, the generalized minimum residual method, GMRES, can be used to solve the system. By reducing the matrix in the way listed in the previous section, the overall modified matrix is no longer symmetric, excluding PCG as a solver. For this method, there must be an outer update loop and an inner solution loop. The outer loop updates the modified matrix with the new angular flux weights, starting with an initial guess at the flux (the standard zero initial flux is assumed). This matrix is then used in the inner solution loop. This procedure is repeated, lagging the matrix at each step, until the flux has converged as shown in the following algorithm, **Figure 2** (Kelley 1995). All variables in the algorithm, with the exception of $\tilde{A}$, ψ, and, S are internal to the solution algorithm and will not be discussed. However, the convergence criteria (the tolerance ε and maximum number of iterations $k_{\max}$) are specified in the input for each solution.

```
GMRES Algorithm
$r = S - \tilde{A}\psi,\ v_1 = {r}/{\|r\|_2},\ \rho = \|r\|_2,\ \beta = \rho,\ k = 0,\ \tilde{A}_0 = \tilde{A}(\underline{0})$
While $\rho > \varepsilon \|b\|_2$ and $k < k_{\max}$ do
  $k = k + 1$
  for $j = 1, \ldots, k$
    $h_{jk} = \left(\tilde{A}_{k-1} v_k\right)^T v_j$
  $v_{k+1} = \tilde{A} v_k - \sum_{j=1}^{k} h_{jk} v_j$
  $h_{k+1,k} = \|v_{k+1}\|_2$
  $v_{k+1} = v_{k+1} / h_{k+1,k}$
  $e_1 = (1, 0, \ldots, 0)^T \in R^{k+1}$
  Minimize $\left\|\beta e_1 - H_k y^k\right\|_{R^{k+1}}$ over $R^k$ to obtain $y^k$
  $\rho = \left\|\beta e_1 - H_k y^k\right\|_{R^{k+1}}$
  $\psi_k = \psi_o + V_k y^k$
  Update $\tilde{A}_k = \tilde{A}(\psi_k)$
```

Figure 2: GMRES Algorithm

Although this direct problem appears to be the most straightforward method, there is one large numerical problem – the matrix can quickly become very ill-conditioned causing the solution scheme to fail. This can be helped slightly by using the Steffensen method (Kelley 1987), which uses an Aitken's procedure to update the solution after every third iteration. Although this helps contain the procedure and reduce quick divergence, convergence is extremely slow due to the ill-conditioned matrices. Condition numbers for these matrices rise by over three decades in the examples given in Chapter 4. For the purpose of this investigation, this method was not used due to its poor performance.

The second method is to use this reduced matrix as a preconditioner for PCG. Preconditioning is in general used to accelerate the convergence of iterative methods by utilizing an approximate inverse of the original problem. The key to develop successful preconditioning strategy is to employ a simple matrix which still captures the dominant physics of the problem. By introducing this matrix at the preconditioning step and using an iterative solver to find the solution, the algorithm's convergence can be sped up greatly. Note that the full matrix is still used in the rest of the routine. **Figure 3** outlines the preconditioned conjugate gradients algorithm (Kelley 1995).

```
PCG Algorithm
$r = S - A\psi,\ \rho_0 = \|r\|_2^2,\ k = 1,\ \tilde{A}_0 = \tilde{A}(\underline{0})$
While $\sqrt{\rho_{k-1}} > \varepsilon \|b\|_2$ and $k < k_{\max}$ do
  $\tilde{A}_0 z = r$
  $\tau_{k-1} = z^T r$
  if $k = 1$ then $\beta = 0,\ p = z$
  else $\beta = \tau_{k-1} / \tau_{k-2}$, $p = z + \beta p$
  $w = Ap$
  $\alpha = \tau_{k-1} / p^T w$
  $\psi = \psi + \alpha p$
  $r = r - \alpha w$
  $\rho_k = r^T r$
  $k = k + 1$
```

Figure 3: PCG Algorithm

This convergence acceleration by the preconditioner does come at a cost, though. This matrix is introduced at the search direction correction step at the start of each iteration, solving the linear problem $\tilde{A}z = r$, where r is the functional residual, z is the correction for the search direction, and $\tilde{A}$ is the preconditioner matrix—in our case the tensor modified matrix. It should be noted here that the modified matrix would be weighted by the z vector in place of the angular flux in order for the tensor reduction to hold. However, since z at each step is unknown but at the true solution when converged should be the zero vector, the modified matrix is always weighted by zero. Therefore this matrix does not need updating. Typically, the preconditioner stage will introduce an extra $O\left[20\left(\left(N-1\right)/2+1\right)\cdot J\right]$ operations considering the bandwidth of the preconditioner, where N and J are again the Legendre expansion and finite element discretization sizes, respectively. For the acceleration method, the preconditioning stage when again considering the structure of the matrix will take $O\left[16\left(\left(N-1\right)/2+1\right)\cdot J\right]$operations. This method can employ the same SGS incomplete LU method for the matrix solution as in previous cases. Since the $\tilde{A}$ matrix is symmetric, another option is to use a preconditioned conjugate gradient inner loop for this solution. Since the fluxes are all weighted at zero, the numerical instabilities in the solution as found in the direct solution are no longer as big of an issue. However, for this method to actually accelerate, the number of iterations needed for this stage must negate enough iterations in the outer PCG algorithm to make the operation count below that of other PCG algorithms. Since the inner PCG algorithm would take $O\left[20\left(\left(N-1\right)/2+1\right)\cdot J\right]$ operations per inner iteration due to structure, it is no competition to the direct incomplete LU solution.

During this investigation, the incomplete LU solution method is used, so the method therefore costs $O\left[26\left(\left(N-1\right)/2+1\right)\cdot J\right]$ operations per iteration. This is in comparison to the total cost of the typical methods described in the previous chapter, Even-Parity Neutron Transport Theory, which is $O\left[48\left(\left(N-1\right)/2+1\right)\cdot J\right]$. Therefore, so long as the method can converge in a similar number of iterations, it will accelerate the problem solution.

Acceleration Using Nonlinear Solution Scheme

The third method, and perhaps the most promising method for application, is the fully implicit inexact-Newton's method. This method solves the nonlinear system of equations at once, alleviating the need for lagging the flux or using the full matrix during the solution. The drawback to this method is that it requires the Jacobian of the matrix, or an approximation to the Jacobian, which can be extremely expensive. The system is then solved as

$$J\left(F\left(\psi\right)\right)\delta\psi=-F\left(\psi\right)=S-\tilde{A}\psi\,, \tag{3-16}$$

where the flux dependence was emphasized to show that all terms are computed simultaneously, $F\left(\psi\right)$ is defined as the nonlinear functional residual, and as in previous chapters the even-parity flux notation has been removed for brevity, $\psi^{+}=\psi$.

One of the largest drivers for this method is its ability to seamlessly couple with other physics in a multiphysics code. This means that temperature, heat, and other multiphysics interdependencies in a code can be coupled in a straightforward fashion to higher order neutron transport. These interdependencies are then not lagged as in typical methods; they are all fully implicit in the solution. As an example, consider a small metal pulsed reactor. As the reactor becomes critical, the temperature of the system rises, heat is transferred quickly through the system, and through thermal expansion the density of the system decreases causing the reactor to go subcritical. For these fast neutron metal systems, high angular orders are necessary to properly model the neutronics of the system. Therefore quasi-diffusion-like tensor accelerated neutron transport using the even-parity derivation has promise for these multiphysics applications.

In order to use these methods the Jacobian must be formed. The Jacobian is defined as

$$J\left(F\left(\psi\right)\right)=\frac{\partial F\left(\psi_n\right)}{\partial\psi_m},\qquad n,m=0,2,\ldots,N-1. \tag{3-17}$$

Therefore, when the expansion for the functional nonlinear residual is plugged into the previous equation, the Jacobian is given as follows.

$$J\left(F\left(\psi\right)\right)=\frac{\partial}{\partial\psi_m}\left[-\frac{d^2}{dz^2}\left(E_{n,n'}\left(z\right)P_n\left(\mu\right)\psi_n\left(z\right)\right)+C_n\left(z\right)P_n\left(\mu\right)\psi_n\left(z\right)-S_n\left(z,\mu\right)\right], \tag{3-18}$$
$$n,n',m=0,2,\ldots,N-1$$

$$J\left(F\left(\psi\right)\right)=-\frac{d^2}{dz^2}\left[\underbrace{\frac{\partial}{\partial\psi_m}\left(E_{n,n'}\left(z\right)P_n\left(\mu\right)\psi_n\left(z\right)\right)}_{H}\right]+C_n\left(z\right)P_n\left(\mu\right)\delta_{n,m}-\cancel{\frac{\partial S_n}{\partial\psi_m}}, \tag{3-19}$$
$$n,n',m=0,2,\ldots,N-1$$

By taking the partial derivatives, Equation (3-19) is produced. Since most terms are easily differentiated, the derivation continues only with the streaming term, H .

$$H\left(z,\mu\right)=\frac{\partial}{\partial\psi_m}\left(\frac{\int_{-1}^{1}P_n\left(\mu'\right)\mu^2G\left(z\right)P_{n'}\left(\mu'\right)\psi_{n'}\left(z\right)d\mu'}{\int_{-1}^{1}P_n\left(\mu'\right)P_{n'}\left(\mu'\right)\psi_{n'}\left(z\right)d\mu'}P_n\left(\mu\right)\psi_n\left(z\right)\right),\quad n,n',m=0,2,...,N-1 \tag{3-20}$$

As shown with Equations (3-8) through (3-12), this is equivalent to

$$H\left(z,\mu\right)=\frac{\partial}{\partial\psi_m}\left(\mu^2G\left(z\right)P_n\left(\mu\right)\psi_n\left(z\right)\right),\quad n,m=0,2,...,N-1 \tag{3-21}$$

which can easily be differentiated to produce

$$H\left(z,\mu\right)=\mu^2G\left(z\right)P_n\left(\mu\right)\delta_{n,m},\quad n,m=0,2,...,N-1. \tag{3-22}$$

From this final result, it can be seen that all terms are diagonal in the angular matrix, producing the same structure as with the tensor reduced matrix, $\tilde{A}$. With this structure, matrix-vector multiplications will take $O\left[6\left(\left(N-1\right)/2+1\right)\cdot J\right]$ operations.

The Jacobian now has the form as given by Equation (3-23).

$$J\left(F\left(\psi\right)\right)=-\frac{d^2}{dz^2}\left[G\left(z\right)\mu^2P_n\left(\mu\right)\right]\delta_{n,m}+C\left(z\right)P_n\left(\mu\right)\delta_{n,m},\quad n,m=0,2,...,N-1 \tag{3-23}$$

This equation can be discretized as done for the standard matrix case above in Equation (2-30). One advantage to the Jacobian of the reduced matrix is that it again attains the spatially symmetric nature that the standard full solution matrix has.

Now that the Jacobian has been stated, the nonlinear solution methods can be discussed. For the purpose of this investigation two different Newton methods for nonlinear equations are used. Although these two methods have a large difference, the basic algorithm is the same and is given in **Figure 4** (Kelley 1995). Additionally, Newton methods for nonlinear equations prove to converge quickly to the solution, needing few nonlinear Newton iterations.

```
Newton Algorithm
$r_0 = \left\| S - \tilde{A}\psi \right\|$
While $\left\| S - \tilde{A}\psi \right\| > \varepsilon r_0$ do
  Compute $J\left(F\left(\psi\right)\right)$
  Solve $J\,\delta\psi = -F\left(\psi\right)$
  $\psi = \psi + \delta\psi$
  Evaluate $F\left(\psi\right)$
```

Figure 4: Newton Algorithm

While both methods use this same core algorithm, they differ in the way that they solve the system $J\,\delta\psi = -F\left(\psi\right)$. The first method, exact Newton, solves this by an LU solution, which costs $O\left[\left(\left(\left(N-1\right)/2+1\right)\cdot J\right)^3\right]$ operations. This method, although expensive, is direct and does not suffer from the ill-conditioning that causes a high number of iterations as with iterative methods. Those iterative methods lead into the second Newton method, Newton-PCG method, which solves the system using a PCG method. This method, as with those described in the previous sections, can greatly reduce the number of operations required for this solution. For the structure of the Jacobian, the same as the quasi-diffusion-like tensor reduced matrix, the system solution will require $O\left[26\left(\left(N-1\right)/2+1\right)\cdot J\right]$ operations for that step.

It is advantageous that this derivation of the Jacobian does not require an update – it is independent of the flux. Because of this fact, it only needs to be calculated once before the main loop. Therefore the total operation count for the method, excluding the solution of $J\,\delta\psi = -F(\psi)$ is $O\left[7\left(\left(N-1\right)/2+1\right)\cdot J\right]$.

In total, the exact Newton method takes $O\left[\left(\left(\left(N-1\right)/2+1\right)\cdot J\right)^{3}+7\left(\left(N-1\right)/2+1\right)\cdot J\right]$ operations per iteration. This method, although expensive, is an extremely useful solver for nonlinear systems. As previously stated, the advantage to the direct method is the fact that it does not suffer from the slow tendencies of iterative solvers and produces the numerically accurate solution in one forward-backward operation. In contrast, the Newton-PCG method takes only $O\left[26\left(\left(N-1\right)/2+1\right)\cdot J\right]$operations per inner iteration and an additional $O\left[7\left(\left(N-1\right)/2+1\right)\cdot J\right]$ operations per outer iteration. Therefore, the iterative method is much cheaper than the exact method depending on the number of iterations. A downside to the Newton-PCG method, however, is the convergence criteria on the iterative solver. If the solution is not found to a good enough tolerance, the Newton method can quickly diverge and no solution will be found. Also, the inner iteration count might be high if the matrices are not well conditioned.

To reiterate, with whichever Newton solver is used, the ability remains for the transport equation to be connected to other physics in a multiphysics program. This would allow for the capability to solve other physics that depend on the neutron flux profile and allow for the effect of other physics on the neutron flux through variables like cross-sections to be solved implicitly. Currently methods for solving implicit multiphysics problems with diffusion theory are available (Gaston, et. al. 2009), but the inclusion of even-parity transport would allow for the solution a wider range of situations. Because of this potential for being included in multiphysics programs, the computational performance, although important for this study, might not be the only benefit for this method.

CHAPTER 4. NUMERICAL RESULTS

To test the capabilities of this acceleration method, four different examples that stretch the capacity of the method are presented. The first example is a one-dimensional problem first presented by Reed (1971) that has been often used to test different transport schemes (e.g. discrete ordinates and even-parity transport). The second example is a typical block problem with a high absorbing material with a distributed source surrounded by a high scattering material – typical of slab nuclear fuel surrounded by hydrogenous material. The third example stretches one of the weaknesses of the even-parity method with a near void region. Finally, the last example is a shielding model that has a forward peaked angular flux which needs higher flux moments for the true solution. All methods were tested using the small one-dimensional even-parity transport program developed for this research, named ANT (nonlinearly Accelerated even-parity Neutron Transport). This code was tested with the direct LU and PCG-SGS methods to check the improvement by introducing the quasi-diffusion-like tensor in both the linear and nonlinear solvers. The flux results are compared to the even-parity neutron transport code EVENT (de Oliveira 1987) to ensure the proper results were produced.

EVENT is a benchmarked three-dimensional finite element, spherical harmonics, even-parity neutral particle transport code. It is capable of not only anisotropic scattering and upscatter but also time dependence and fission. It has a preprocessor for the front end data processing that can take many different multigroup cross-section formats (matxs, fido, etc.) and put them in a standard format for EVENT while also meshing the up to three dimensional problem, among other things.

In order to compare the methods, the total operation count, or cost, to get to a given convergence criteria is used. The approximate operation counts, or total computational effort, used in this comparison take into account the bandwidth and sparsity of each matrix to give a closer representation of the cost of each method. The operation counts are given in each section above, but will be given again in the tables below for quick reference.

Convergence in all examples was attained when the ratio of the 2-norm of the residual to the 2-norm of the source vector dropped below a tolerance of 5.0E-5. The maximum number of iterations for PCG was set at 5000 iterations to allow this to not be the limiting factor. Each case was run at the expansion levels P_5, P_7, P_{11} and P_{27}. This was done to ensure that each problem's flux had converged (difference between P_{11} and P_{27}) while also proving the acceleration at all different expansion levels.

For details on the program, Appendix A: Program Description contains the description of its operations.

Reed Example

Often new transport schemes use this example to test the method because of the complexity of some numerical solutions of the problem. When care is not taken, methods such as the common S_n method with diamond differencing can produce oscillations in the total scalar flux approximation (Martin et.al. 1981). **Figure 5** shows the setup of this one-dimensional problem.

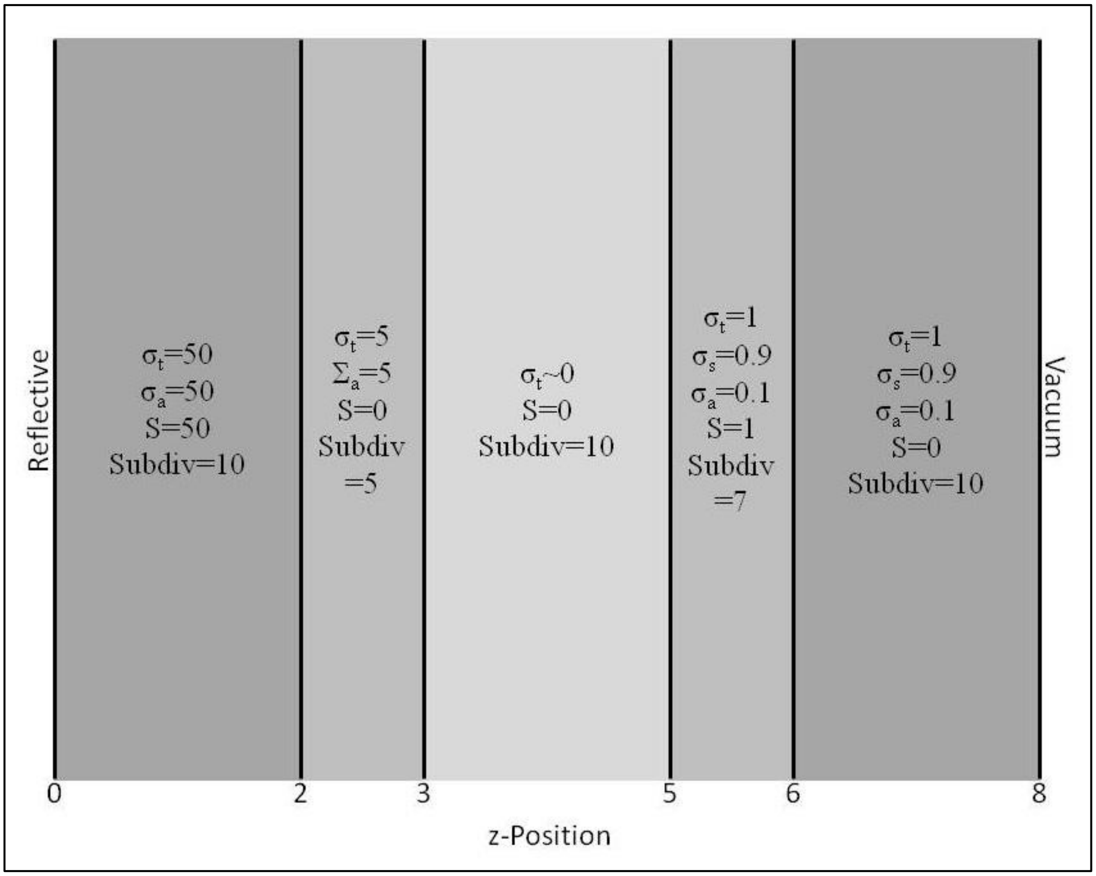


Figure 5: Reed Example Configuration

To test the quasi-diffusion-like acceleration method on this problem, five methods are used in ANT: direct method by Gaussian elimination, PCG with symmetric Gauss-Seidel (PCG-SGS), PCG with the quasi-diffusion-like tensor (PCG-QD), exact Newton method with the quasi-diffusion-like tensor (Newton/LU-QD) and Newton-PCG method with the quasi-diffusion like tensor (Newton/PCG-QD). The operation count results for each method are given in **Table 1**.

Table 1. Reed Example Operation Count Comparison

Method	Flops/Iteration	Iterations - P7	Flops - P7	Iterations - P5	Flops - P5
Direct LU	*(((N-1)/2+1)*J)^3*	1	5.09E+06	1	2.15E+06
PCG-SGS	*48(((N-1)/2+1)*J)*	231	1.91E+06	155	9.60E+05
PCG-QD	*26(((N-1)/2+1)*J)*	20	8.94E+04	15	5.03E+04
Newton/LU-QD	*(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)*	149	7.58E+08	434	9.32E+08
Newton/PCG-QD	*7(((N-1)/2+1)*J)*	40	2.24E+07	31	2.13E+07
	*26(((N-1)/2+1)*J)*	125		205	
Method	**Flops/Iteration**	**Iterations - P27**	**Flops - P27**	**Iterations - P11**	**Flops - P11**
Direct LU	*48(((N-1)/2+1)*J)*	1	2.18E+08	1	1.72E+07
PCG-SGS	*26(((N-1)/2+1)*J)*	747	2.16E+07	384	4.76E+06
PCG-QD	*(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)*	87	1.36E+06	29	1.95E+05

As compared to all other cases, the PCG-QD results prove to be a very large numerical savings. For the P_5 through P_{27} expansions, PCG-QD proves to be about a 93% operation savings over PCG-SGS, the current common solution method. These improvements are significant when considering problems that could have thousands of mesh points and large Legendre expansions – matrix sizes in the hundreds of thousands or millions. For the Newton/PCG-QD case, acceleration was not realized. In fact, the method was as much as a factor of 22 higher in computational cost than the current method in P_5 and P_7. It was therefore not run for the larger angular expansion cases.

Since the nonlinear method did not accelerate, it is necessary to investigate the cause of this to see if improvements can be made. The reason the method had such a high cost was due to the inner iterations of the Newton iteration – which is solved by PCG-SGS but with the block tridiagonal structure. Since this inner loop took as many iterations as the standard PCG-SGS, doing it with each of the 40 outer iterations negated the fact that the work required for the inner loop is less per iteration. As mentioned above, even though the Newton iteration might take more iterations, casting the problem in this nonlinear fashion allows it to be introduced into a multiphysics code easily for a fully implicit solution. Therefore as long as the solution doesn’t take an extraordinary amount of work it can still be extremely useful.

To ensure the results of this method match with those produced by EVENT the scalar flux results are compared. The flux results from ANT are taken from the PCG-QD and Newton/PCG-QD methods with P_7 Legendre expansion. **Figure 6** shows this comparison. Two EVENT flux curves are reported, a P_7 expansion scalar flux to compare directly with the results from ANT and a P_{27} expansion scalar flux to capture more flux moments in the solution and show a more precise flux profile prediction (again compared with the ANT results).

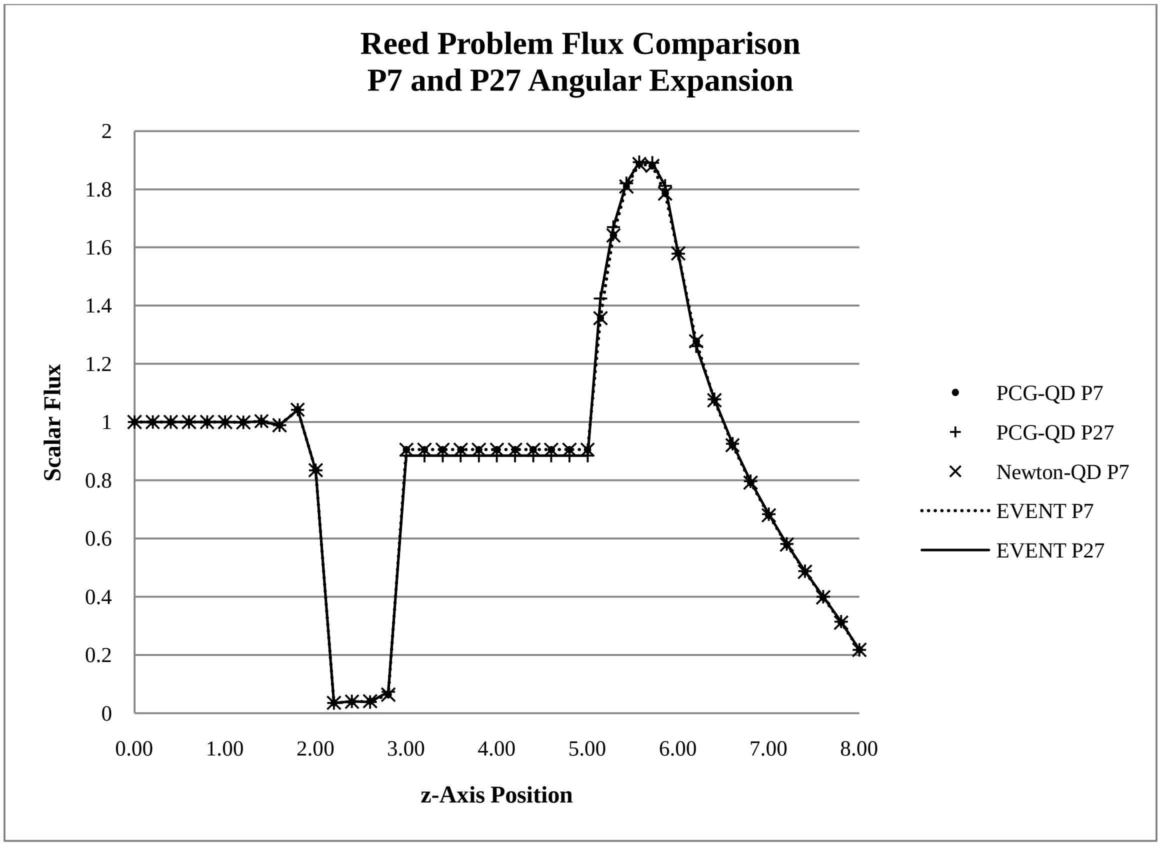


Figure 6: Reed Example Flux Comparison, ANT vs. EVENT

Figure 6 shows that the profiles match well between ANT and EVENT, even using the different solution methods. Over all P_7 results, there are only numerical uncertainty differences between the points of the different methods – a relative difference of less than 0.2%. The methods are taking identical information and processing them through different methods to arrive at the same result, an indication that the acceleration methods solve the system correctly. Also, it can be seen that these P_7 methods model the Reed system well as compared to a higher order Legendre expansion of the flux, P_{27}. The region of highest discrepancy is the approximate void area from $z = 3$ to $z = 5$. In this portion of the problem the P_{27} is able to model the rapid flux changes from region to region better than the P_7 expansion methods. A successful check was done for this problem, as well as all others, to ensure the flux was converging in a trend from P_7 to P_{11} to P_{27}.

Overall performance on the Reed problem using the PCG-QD method is considered good due to both the flux output and computational acceleration. For the Newton/PCG-QD method, the flux output was also good; however the computational performance was behind that of current methods.

Simple Fuel/Moderator Example

The second example is a simplified version of nuclear fuel surrounded by a high scattering material, like water. This example is commonly encountered in nuclear modeling and the acceleration on a real-world problem is important. Not only is this model used when fully modeling a nuclear reactor system, but it is used also in cross-section collapsing for homogenization (Duderstadt and Hamilton 1976). The acceleration achieved on this example could have an impact on the ease and speed of performing this common modeling.

The following figure, **Figure 7**, depicts the system being modeled.

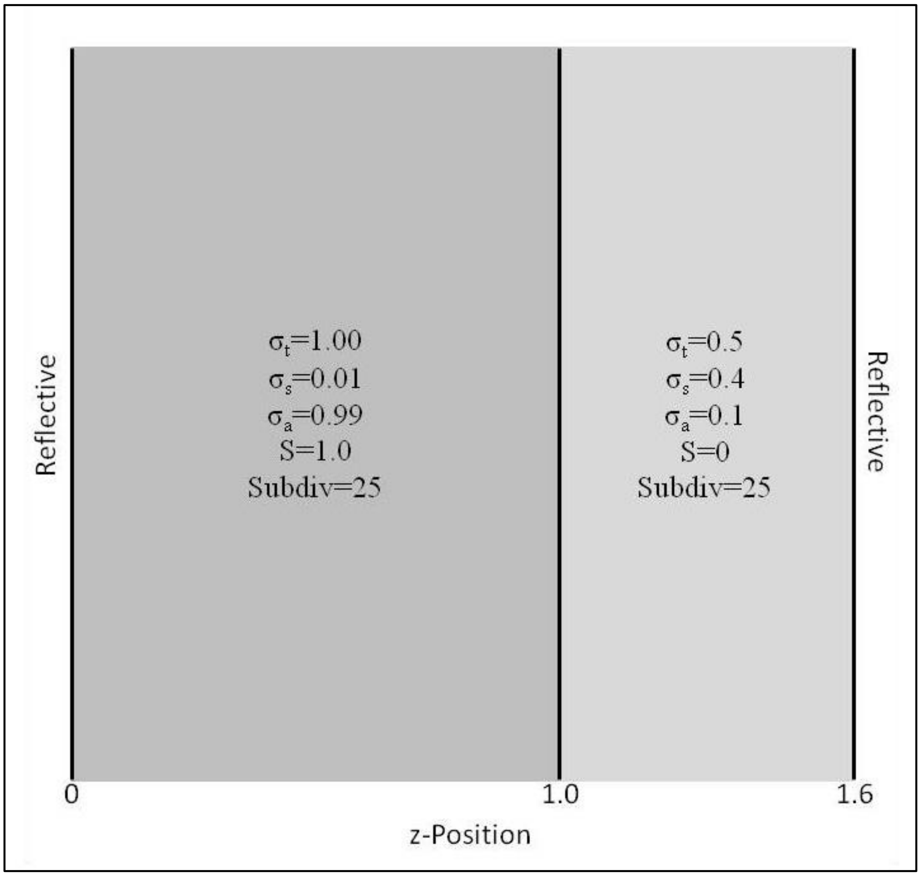
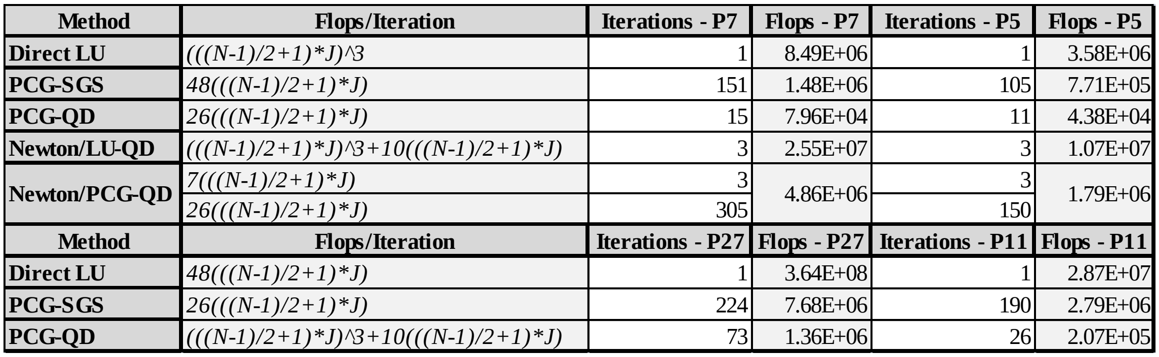


Figure 7: Fuel/Moderator Example Configuration

The same comparisons are performed on this model as with the Reed model in the previous section, Reed Example. Again, the goal is to minimize the number of total operations to get to the desired level of accuracy. **Table 2** shows the performance of several different methods on this case, using P_5, P_7, P_{11}, and P_{27} Legendre expansions for the angle. Again, the methods that were tested and reported are the direct method, PCG-SGS, PCG-QD, Newton/LU-QD, and Newton/PCG-QD.

Table 2. Fuel/Moderator Example Operation Count Comparison



Method	Flops/Iteration	Iterations - P7	Flops - P7	Iterations - P5	Flops - P5
Direct LU	*(((N-1)/2+1)*J)^3*	1	8.49E+06	1	3.58E+06
PCG-SGS	*48(((N-1)/2+1)*J)*	151	1.48E+06	105	7.71E+05
PCG-QD	*26(((N-1)/2+1)*J)*	15	7.96E+04	11	4.38E+04
Newton/LU-QD	*(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)*	3	2.55E+07	3	1.07E+07
Newton/PCG-QD	*7(((N-1)/2+1)*J)*	3	4.86E+06	3	1.79E+06
	*26(((N-1)/2+1)*J)*	305		150	
Method	**Flops/Iteration**	**Iterations - P27**	**Flops - P27**	**Iterations - P11**	**Flops - P11**
Direct LU	*48(((N-1)/2+1)*J)*	1	3.64E+08	1	2.87E+07
PCG-SGS	*26(((N-1)/2+1)*J)*	224	7.68E+06	190	2.79E+06
PCG-QD	*(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)*	73	1.36E+06	26	2.07E+05

The acceleration achieved for this problem for PCG-QD is again approximately 82-94% for up to P_{27} versus the current solution method, PCG-SGS. Again, for the larger problems where the expansions are needed to be very large, cutting the total number of floating point operations to only a few percent of the usual amount is an enormous savings. For the applications listed above, that could be a very large savings as this calculation might just be one of many to be run for a given system's homogenization. The Newton/PCG-QD method fared much better on this example versus the Reed problem, but still failed to accelerate the solution as it took about double the number of operations. Again, this is contributed to the need for a large amount of inner iterations, but thankfully the method quickly converges in the outer iterations so the impact is not as large.

In order to ensure that the solution is the correct even-parity transport method solution, it is compared with the identical case as well as the P_{27} case from EVENT and ANT. The plot of the scalar flux is given in **Figure 8**.

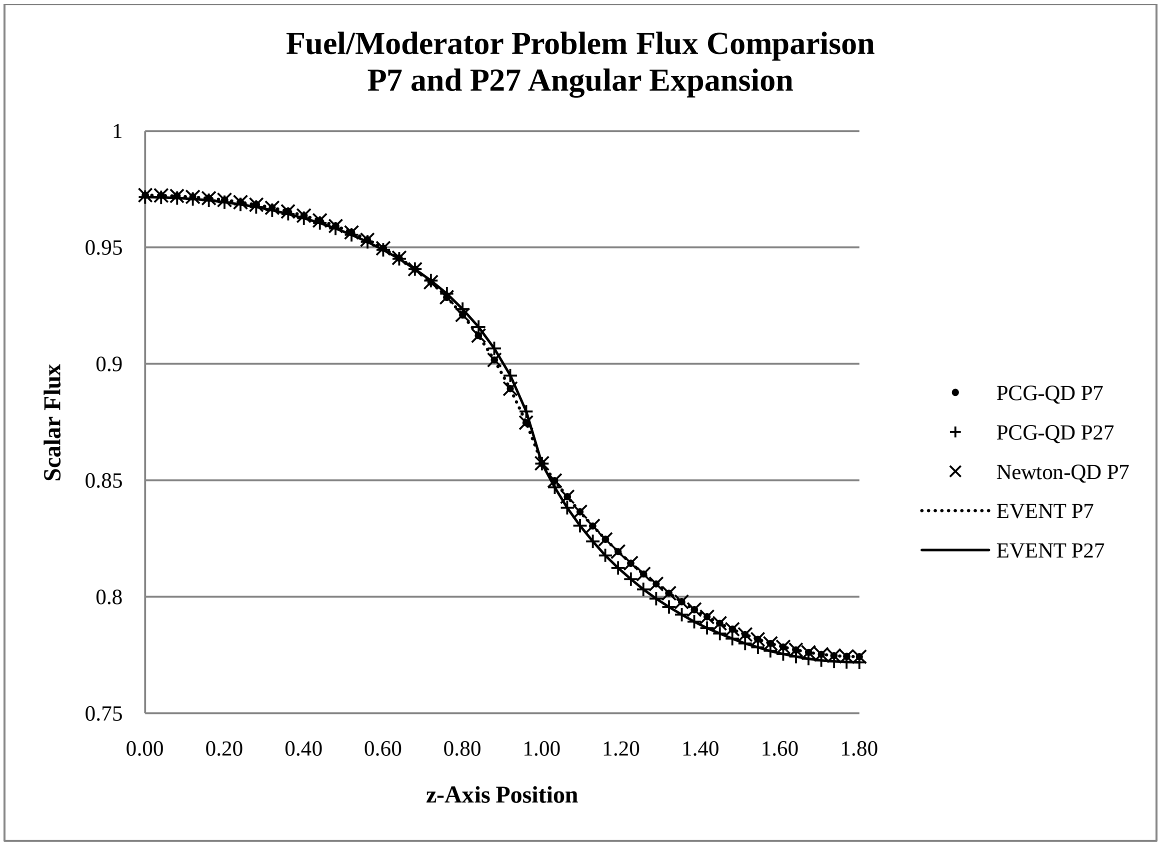


Figure 8: Fuel/Moderator Example Flux Comparison, ANT vs. EVENT

The above figure shows that the different solution methods again are able to still produce identical results, with a maximum relative difference between P_7 methods of about 0.00025%. This is much smaller than with the Reed problem due to the small flux changes in that problem and the variability of the methods in converging to a solution about the true even-parity solution. Again, the P_7 models closely match the P_{27} higher order model and represent the system dynamics fairly well. The most noticeable difference between the two levels of expansion is at the interface where the larger expansion is capable of capturing the more realistic smooth flux transition.

This example shows that even for a mundane problem encountered in nuclear modeling, there are gains to be had by using this acceleration method. Not only was the PCG-QD method able to capture the flux profile without loss of accuracy, it was able to do it in a small fraction of the computations. For the Newton-QD method, although it was again not able to accelerate the solution convergence, it was able to capture the flux profile also without losing any accuracy.

Shielding Example

The third example problem to test the developed acceleration methods is another commonly encountered example. Accurate shielding calculations are important and in order to compute them using the even-parity method high angular orders are necessary. This is due to the forward peaked flux as the neutrons travel through the shielding material and get absorbed. This absorption and low scattering makes the angular distribution of the streaming particles become more and more directed only in one direction. Since the higher the expansion order of the flux the more it can capture highly anisotropic behavior, shielding problems require large expansions.

The shielding problem, given in **Figure 9**, has a source region followed by a high scattering region to make the flux closer to isotropic followed by a shielding/high absorbing region.

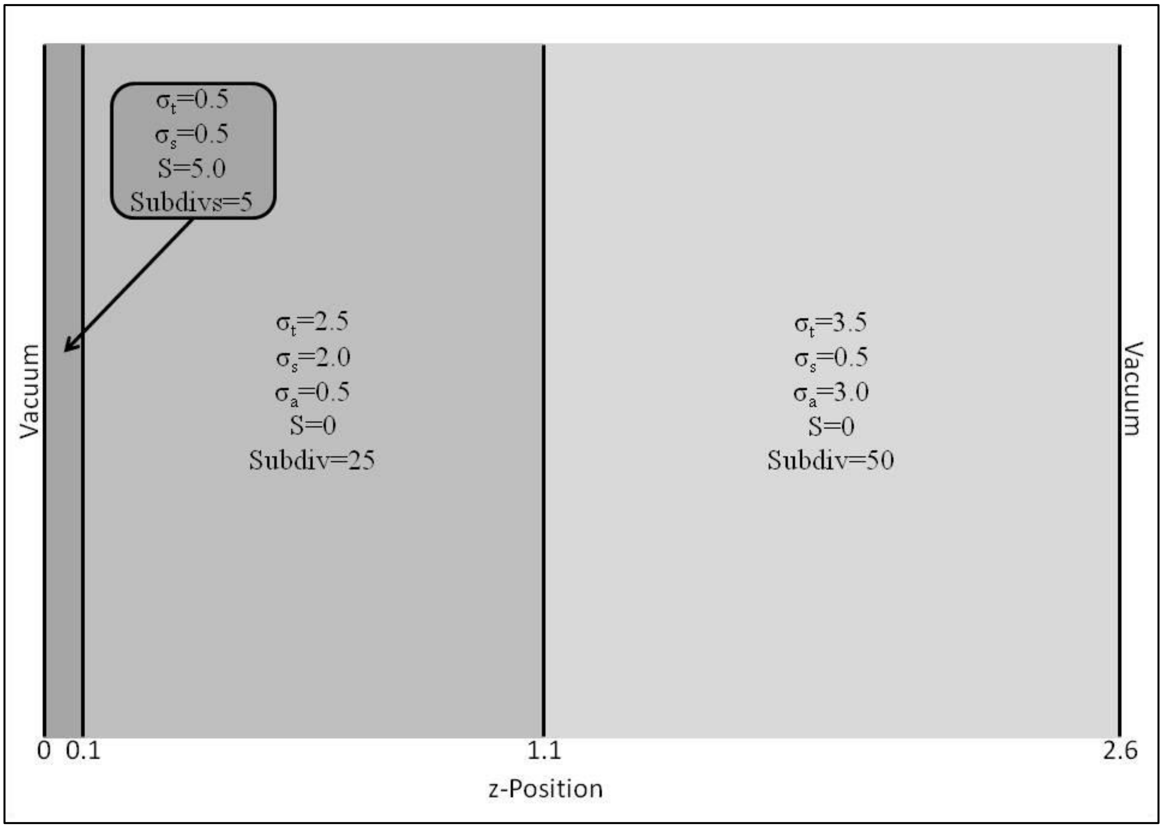


Figure 9: Shielding Example Configuration

As with the previous examples, the direct, PCG-SGS, PCG-QD, Newton/LU-QD, and Newton/PCG-QD methods were all run through ANT. These were compared to the results of EVENT, using up to a P_{27} expansion of the flux. The scalar flux profiles for this problem are given in **Figure 10**.

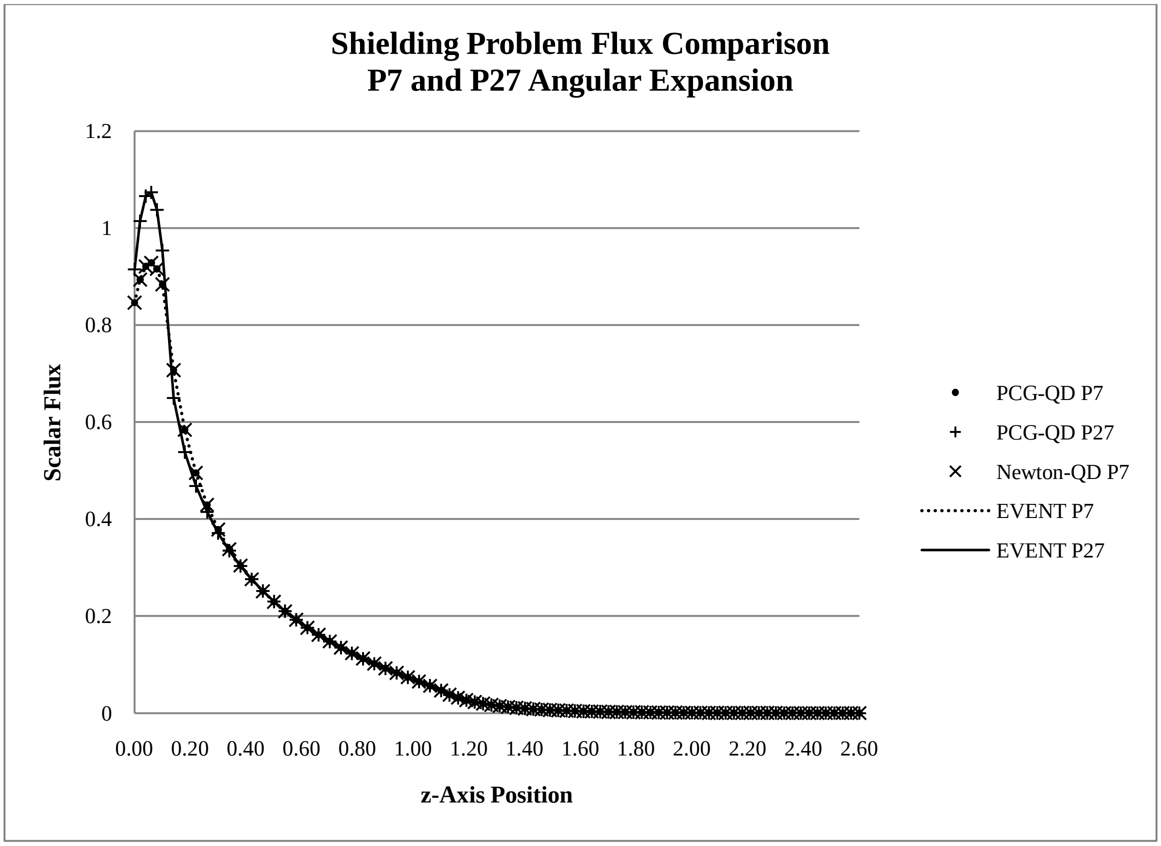


Figure 10: Shielding Example Flux Comparison, ANT vs. EVENT

The flux profiles shown in this figure exemplify the need for a high order expansion on shielding problems. The P_7 results are much lower in the source region than that of the P_{27} result which is able to better approximate the region's behavior. The angular expansion trend check showed the P_{27} was adequate to model the converged system. Overall, however, the results follow the same general shape. In comparing the different P_7 methods used, there is again little discrepancy in the converged results, which again says that the acceleration methods are able to model the system without loss of accuracy. The maximum difference in the scalar fluxes of the P_7 method is found at the very low value right end of the problem. At this point, the maximum relative difference is

1.5%, which in comparison to the last examples seems extremely high. However, it should be noted that the 0th flux moment (scalar flux) in this region is of the same order as the 2nd and 4th flux moments. Since the bulk of the flux values are at this level when compared to the high source region flux are approximately five orders of magnitude less, the numerical methods have some variability in the converged solution. The different methods are not considered to be in error in this region then as it is an artifact of the convergence criteria.

Since the accelerated method flux profiles again match the standard even-parity flux, the computational performance is needed to compare the methods. The operation counts for this example are given in **Table 3**.

Table 3. Shielding Example Operation Count Comparison

Method	Flops/Iteration	Iterations - P7	Flops - P7	Iterations - P5	Flops - P5
Direct LU	(((N-1)/2+1)*J)^3	1	3.40E+07	1	1.43E+07
PCG-SGS	48(((N-1)/2+1)*J)	70	1.09E+06	53	6.18E+05
PCG-QD	26(((N-1)/2+1)*J)	29	2.44E+05	20	1.26E+05
Newton/LU-QD	(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)	28	9.52E+08	24	3.44E+08
Newton/PCG-QD	7(((N-1)/2+1)*J)	28	3.78E+07	24	2.28E+07
	26(((N-1)/2+1)*J)	160		150	
Method	**Flops/Iteration**	**Iterations - P27**	**Flops - P27**	**Iterations - P11**	**Flops - P11**
Direct LU	48(((N-1)/2+1)*J)	1	1.46E+09	1	1.15E+08
PCG-SGS	26(((N-1)/2+1)*J)	95	5.17E+06	76	1.77E+06
PCG-QD	(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)	104	3.07E+06	44	5.56E+05

These operation results again show that for the cases greater than P_5 the PCG-QD method can reduce the computational costs by over 40% (and up to nearly 80%) versus the PCG-SGS standard method. As discussed previously, this is done without loss of accuracy versus this current solution scheme. However, again the Newton/PCG-QD method fails to reduce the number of operations required for the solution. For this case, the solution required nearly 37 times the number of operations. This is due to the large number of inner iterations required per Newton iteration as with previous examples. But again, this nonlinear casting of the equation can be used to exploit other functionality and operation count might not be the driving factor in these situations.

Near Void Example

The fourth and final example problem to test the developed acceleration methods is based on the common known limitation of even-parity methods, a near void region. Since the streaming term of the second order transport equation contains the inverse of the total cross section, $1/\Sigma_t$, when the cross-section approaches zero as with near void regions, the term tends toward infinity. This can cause many problems in the solution of the system; numerically it can cause instabilities in the solution and physically it causes the streaming term to dominate the transport equation (as it should). Care must then be taken when dealing with this type of region. Additionally since the streaming term dominates the equation in that region, other physics must still be able to be modeled in the solution matrix and not be washed out, especially those in other regions. These precautions have been taken with this example and in the program to ensure the proper solution without errors.

The near void problem, given in **Figure 11**, has a reflected boundary on the left of a source region followed by a region of equal scattering and absorption which is then followed by the near void region and a high absorber. This problem was designed to ensure the void region had adequate particle streaming in a single direction to also get anisotropy in the flux of the region.

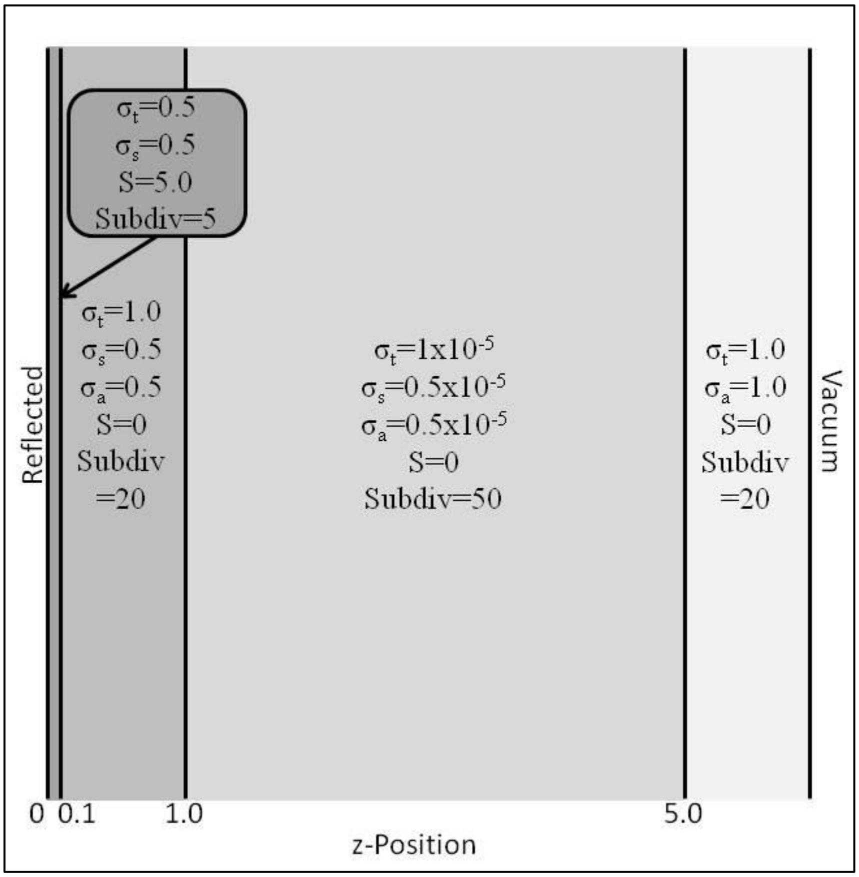


Figure 11: Near Void Example Configuration

As with the all other problems, five solution methods were used: direct, PCG-SGS, PCG-QD, Newton/LU-QD, and Newton/PCG-QD. These ANT methods were compared to the results of EVENT using again up to P_{27} expansions. The flux profiles for this problem are given in **Figure 12**.

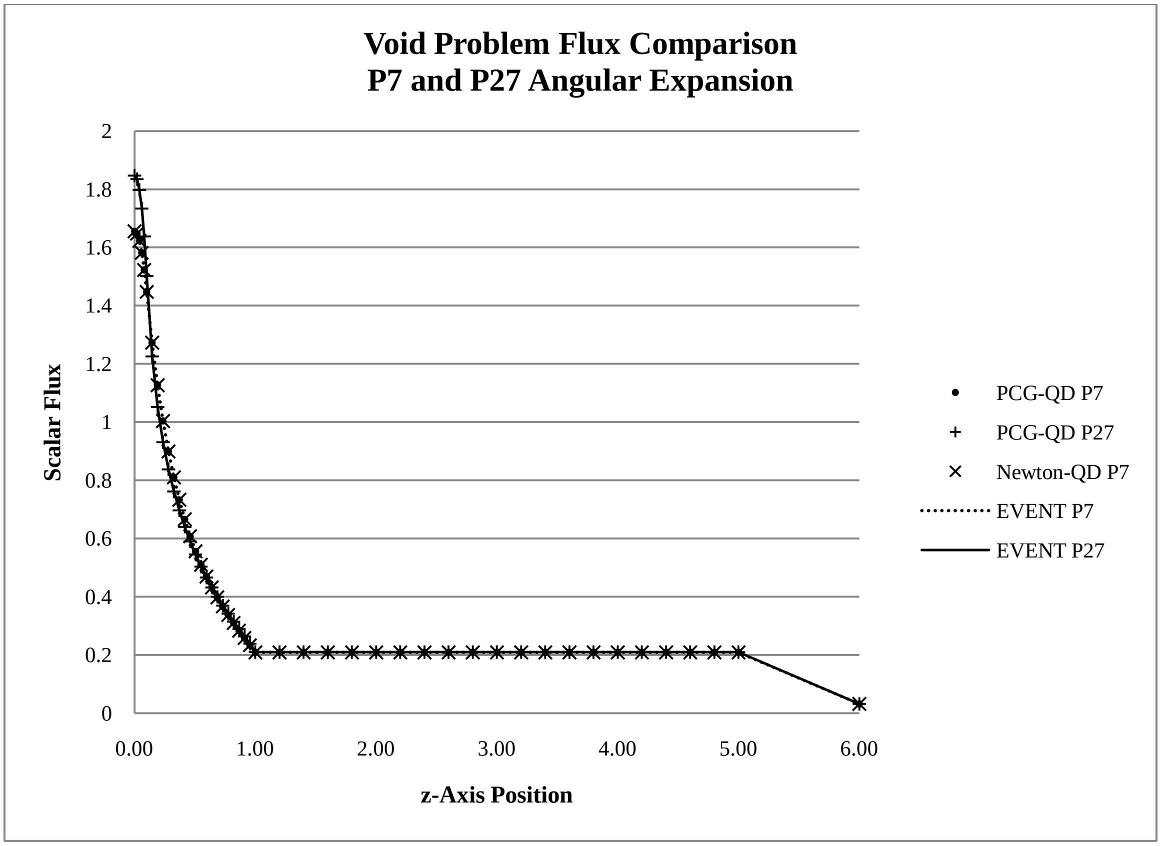


Figure 12: Near Void Example Flux Comparison, ANT vs. EVENT

Overall, the ANT results match extremely closely to those of EVENT. The maximum relative difference between the P_7 methods for the near void example is 0.0023% again showing that the accelerated methods produce the flux results without loss of accuracy. This example also shows the effect of high order expansions on regions of high neutron current, in this case in the source region. In general though, the lower order expansion's scalar flux matches well with that of the higher order expansion – in the void region only a 0.3% relative difference.

Since the accelerated method flux profiles again match the standard even-parity flux extremely close, the computational performance is needed to compare the methods. The operation counts for this example are given in **Table 4**.

Table 4. Near Void Example Operation Count Comparison

Method	Flops/Iteration	Iterations - P7	Flops - P7	Iterations - P5	Flops - P5
Direct LU	(((N-1)/2+1)*J)^3	1	5.66E+07	1	2.39E+07
PCG-SGS	48(((N-1)/2+1)*J)	295	5.44E+06	190	2.63E+06
PCG-QD	26(((N-1)/2+1)*J)	20	2.00E+05	13	9.73E+04
Newton/LU-QD	(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)	16	9.06E+08	14	3.34E+08
Newton/PCG-QD	7(((N-1)/2+1)*J)	16	1.20E+08	14	5.14E+07
	26(((N-1)/2+1)*J)	750		490	
Method	**Flops/Iteration**	**Iterations - P27**	**Flops - P27**	**Iterations - P11**	**Flops - P11**
Direct LU	48(((N-1)/2+1)*J)	1	2.43E+09	1	1.91E+08
PCG-SGS	26(((N-1)/2+1)*J)	1220	7.87E+07	516	1.43E+07
PCG-QD	(((N-1)/2+1)*J)^3+10(((N-1)/2+1)*J)	85	2.97E+06	34	5.09E+05

Yet again for this final example, the PCG-QD method reduces the operation requirement by over 96% as compared to PCG-SGS, the current method. Once more this acceleration is accomplished without loss of accuracy versus this current solution scheme. For the fourth time, though, the Newton/PCG-QD method failed to reduce the cost of the computation. The solution required about 22 times the number of operations. Even though it again took more operations, the resulting flux converged to the standard EVENT solution showing the method is possible and does not lose any accuracy.

Discussion

After the four examples whose results are given above, the acceleration method can be fully examined for its effectiveness and appropriateness for certain problems. The

Reed problem tested the quasi-diffusion-like tensor accelerated methods versus a frequently used reference case, while the fuel-moderator problem tested the method against a frequently considered problem for deterministic transport methods. The final two examples, the shielding and near void cases, tested some of the limitations of the method's implementation and theory.

In all cases the PCG-QD acceleration method, where the nonlinear, quasi-diffusion-like tensor reduced matrix is introduced as the preconditioner, performed extremely well. Over all four example problems, the operation count was reduced by between 40% and 96% versus the so-called standard method of this investigation, PCG-SGS. In the case of the lower end of that range, it is the PCG-SGS method that is accelerating nearly as well as the PCG-QD method. It is not that the PCG-QD method does not perform well, which can be seen in a direct comparison with the other examples. This could be checked by doing an investigation into the spectrum of each preconditioner matrix. These operation counts take into consideration the structure of the matrices in order to get as accurate an estimate for the total cost of the method. Additionally, the PCG-QD method matched in scalar flux the comparison standard even-parity program, EVENT, extremely closely. The largest difference in the results was observed in the shielding problem where the scalar flux range was approximately 5 orders of magnitude, leading to numerical fluctuations in the results. These fluctuations could be eliminated if the convergence criteria in the problem are tightened. With that in mind, the performance of the method as an accelerator for even-parity neutron transport is extremely promising.

On the other hand, the cases of the Newton-QD acceleration scheme, where the tensor reduced system is solved implicitly using different Newton methods did not perform as well in terms of acceleration. Two different methods were used, an exact Newton method and a Newton-PCG method. For each, the tensor reduced matrix was used throughout to create the Jacobian of the functional residual. Unfortunately, the cost of both methods proved to hinder these methods when compared to their linear counterparts. The lowest computational requirements were needed for the fuel-moderator example, where the Newton/PCG-QD method was only double the cost of the PCG-SGS existing method. These methods, though, proved to get the correct solution without a problem. Therefore, the usefulness of these nonlinear solution methods lies in a different class of applications – coupling with other physics in a multiphysics program. Although this requires a slightly higher amount of work than a typical linear solution method for even-parity neutron transport, the nonlinear solution method allows for implicit solution of fully nonlinear problems.

CHAPTER 5. CONCLUSIONS AND FURTHER WORK

The extended quasi-diffusion tensor method for the acceleration of even-parity neutron transport provides a way to greatly reduce the effort for a calculation and opens up the possibility of a multitude of other applications. The introduction of this modified matrix can be done in both a linear and nonlinear model.

The linear solution method was presented first. The solution, using matrix updating after each iteration proved to be extremely powerful. However, due to numerical instabilities, a direct solution of this reduced system itself is unpractical. The preconditioned conjugate gradients method using quasi-diffusion-like tensor reduced matrix as a preconditioner (PCG-QD) proved to reduce the total operations required for all sample cases by between 40% and 96% while producing the same accuracy. These results were compared against the typical solution method for the second order transport equation, PCG with the symmetric Guass-Seidel preconditioner (PCG-SGS). The results using the accelerated method were checked against the spherical harmonics even-parity neutral particle transport code EVENT and were shown to match to the precision of the problem.

The results show that there is much promise in using this acceleration method for second order unstructured transport methods. The savings in the linear methods for everyday problems is significant, and for large problems could be massive.

The other solution schemes, the nonlinear methods, were presented second. Since the tensor reduced problem is inherently nonlinear (the flux is needed in order to solve for the flux), these methods are the most natural and possibly the most powerful. The downside as shown is that these methods are more expensive computationally than existing solution methods. The exact Newton method (Newton/LU-QD) was by far the most expensive method due to the fact that there is a direct matrix solve during each Newton iteration. This is contrasted by the second method that uses PCG to solve the inner matrix-vector problem. Newton/PCG-QD performed much better than the exact method but suffered from the same problem due to the fact that many PCG iterations were required in order to converge the inner solution during each outer iteration. However, these methods produced the correct answer as compared to the EVENT output without problems. As the method works, applications for this form can be sought. By moving to this type of a solution scheme, high order transport methods can find application by being seamlessly integrated into implicit multiphysics codes to attain higher accuracy. This higher accuracy is compared to the use of diffusion methods currently in these programs.

These methods were investigated without loss of generality to the full three-dimensional spherical harmonics even-parity transport method. Their extension to this regime is a future step that will potentially allow large savings in effort for systems that require high angular moment modeling. Further investigations into different methods can be done to further this research. Looking into different solution schemes altogether, like the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, have the potential to converge to the solution in fewer operations than even the accelerated method outlined in this thesis. It would also be extremely beneficial to see the spectrum of the new acceleration method versus the standard preconditioner. This could give insight into the observed performance trends. Additionally, as a logical next step, this extension could include both linear and nonlinear solution methods to be later coupled with other physics for a fully implicit, high accuracy multiphysics solution. Additionally, the Newton method could be used with an incomplete LU solver in the inner loop to attempt to reduce the total cost of the method or with a preconditioner to speed up the convergence.

APPENDIX A: PROGRAM DESCRIPTION

This appendix gives a brief summary of the processes in the Accelerated even-parity Neutron Transport code, ANT. This discussion is not meant to give the inner details of the specific process but rather give a general idea of how ANT works.

ANT is an even-parity 1-D slab transport code written in C++ capable of solving a variety of problems. The capabilities include one-dimensional finite element based planar neutron transport with isotropic scattering, reflected/vacuum/albedo boundary conditions, unlimited regions, unlimited materials, and unlimited sources. For mesh creation and material input, a preprocessor is used. The preprocessor, GEM, is the identical preprocessor for EVENT (de Oliveira 1987). This program takes a general input and creates the meshing, associates materials and sources with regions, and puts all information in a repeatable ASCII file format. These files are the only input for ANT.

The program is not written for efficiency in terms of memory usage as its goal is to be able to traceably compare the different solution methods employed. With that in mind, matrices are explicitly formed instead of programming their action on a vector. Although this can lead to some numerical inaccuracies due to excess operations and error cancellation, these errors are assumed to be inherent in the program and constant over all methods. Therefore comparing methods all solved within ANT is assumed to be valid.

The following subsections give a description of each source file within ANT and the functions they perform. The sequence and information path is highlighted throughout as to show how files are processed. The program has been tested and compiled using both g++ and the Intel ICC compilers on a Fedora 10 Linux 32-bit machine. ANT is invoked by the command `ant` *`filename solver_number`* from the terminal.

ant.cpp

This file is the driver for the entire program. This *main()* function of the program is located in this file, which accepts up to two different values – the filename and the solver number. If either is omitted, prompts will ask for the values. The main function also ensures the input files exist and that the solver requested is valid – if it isn't the default of PCG-SGS is used. The main function then calls the routine to get all the input information, returning populated pointer arrays and variables with the information from the GEM created files. At this point, the solution routines can be called.

The next step in the main function is to call the appropriate solution file. ANT has the capability of solving both the even-parity and S_n transport equations, the latter of which will not be discussed. For the purpose of this investigation, the main function calls the *solve()* function which will solve the system for the angular fluxes. Once returned, the function output the results and returns an error code, if applicable.

The other functions in this file are the *errorExit()* and *errCodeToMsg()* functions. The first function is invoked when an error or exception is thrown somewhere in the program. This function will take then invoke the second function to output the appropriate message before exiting. That second function takes the error code integer value from all over the program and translates it to an easy to read, descriptive error message.

getInputs.cpp

The main driver function for this file is the *getInputs()* function which is invoked by *main()*. The functions called by this routine, in order, are *readControl()*, *readMesh()*, and *readCxs()*. So long as these functions do not return errors, *getInputs()* returns the populated variables to the main program.

The first function called, *readControl()*, reads the control file that is created by the preprocessor, GEM. This function reads in all of the control information for the current run, including the maximum number of iterations and the tolerances, both inner and outer, the number of materials, the angular order, and number of materials, cross-sections, etc. Dynamic allocation of some arrays is performed when these values are input.

The second function, *readCxs()*, reads the GEM created cross-section file and inputs the data into dynamically allocated arrays. Although the program can only process isotropic scattering, this function will read in all scattering moments. Additionally, although the program cannot process fission, the fission cross-section for each material is read in, if present. Finally the source strengths are read into an array. These values pointer arrays are then returned to the *getInputs()* function.

The third and final reading routine, *readMesh()*, reads in the mesh data created by in the preprocessor. The node locations, element-node associations, material/source-element associations, and boundary information are all read into the program during this routine. The dynamic allocation of the remaining variables is performed at this step as well. All of this information is returned to the *getInputs()* routine, as well as any errors that were encountered if the cross-reference value checking, like the number of elements, nodes, etc. actually present in the file.

solve.cpp

The main routine in this file is the *solve()* function, which handle the calling of appropriate routines based on the desired solver. The first routines invoked independent of the solver are the angular matrix creation routine *createG()* and the *createMatrixSlab()* which populates the appropriate matrices and arrays with data for the transport problem solution. This is then followed by a call to the direct LU, PCG, CG-QD, or Newton-QD solver. Finally, the function returns to the main program for output of the results.

The *createMatrixSlab()* function populates both the A matrix and the source vector S . The function will add only the appropriate non-streaming terms to the reduced matrix.

The *solveLU()* function solves the matrix problem given an input matrix and vector using the direct LU solution method.

The *solvePCG()* routine solves the matrix problem given an input matrix and vector using the PCG method. This routine updates the reduced matrix if necessary at each iteration and convergence is based on the 2-norm of the functional residual multiplied by the tolerance dropping below the 2-norm of the source vector as in a typical PCG method.

The *solveEddingtonCG()* function solves the reduced matrix problem with the main matrix being the tensor reduced matrix. Each iteration updates the reduced matrix with the current flux and convergence is based on the same as the PCG method. This is a standard conjugate gradients method with matrix updating.

The *solveEddingtonNewton()* routine solves the reduced matrix problem using either exact Newton or Newton-PCG methods. The function calls all appropriate inner solvers as well as the Jacobian building function. Convergence is based on the same as that of the PCG and CG methods above. The outer Newton iterations are performed in this function.

The next function, *createG()*, is the beginning of the non-solver portion of solve.cpp. This function creates the angular scattering matrices for streaming, scattering, capture, and vacuum boundaries.

Preconditioner matrices are created in the *createM()* function. This function can create the preconditioner matrix for PCG-Identity (in essence standard CG), PCG-Jacobi, and PCG-SGS methods. These are used in the *solvePCG()* routine.

createEddingtonSlab() is the function used to add the flux-weighted reduced A matrix streaming terms for use in the acceleration methods. This is coupled with the streaming-less A matrix created by *createMatrixSlab()*.

The final two functions, *createJ()* and *JStream()* couple together to create the Jacobian of the reduced matrix for solution in the Newton methods. This function also creates the source vector to ensure it has not been modified in other solution processes.

REFERENCES

1. D.Y. Anistratov, "Nonlinear Quasidiffusion Acceleration Methods with Independent Discretization," *Transactions of the ANS,* **95**, pp. 553-555 (2006).
2. J.J. Duderstadt and L.J. Hamilton, *Nuclear Reactor Analysis*, John Wiley & Sons Inc. (1976).
3. D. Gaston, et. al., "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design,* **239**, pp. 1768-1778 (2009).
4. V.Ya. Gol'din, "A Quasi-Diffusion Method for solving the Kinetic Equation," *USSR Comp. Math. and Math. Phys.*, **4**, pp. 136 (1964).
5. C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia (1987).
6. E.E. Lewis and W.F. Miller, *Computational methods of neutron transport,* American Nuclear Society, La Grange Park (1993).
7. W.R. Martin, C.E. Yehnert, L. Lorence, J.J. Duderstadt, *Annals of Nuclear Energy*, **8**, pp. 633 (1981).
8. C.R.E. de Oliveira, "Finite element techniques for multigroup neutron transport calculations with anisotropic scattering," *Nuclear Engineering,* Queen Mary College, University of London, London (1987).
9. C.R.E de Oliveira, C.C. Pain, and M. Eaton, "Hierarchical Angular Preconditioning for the Finite Element-Spherical Harmonics Radiation Transport Method," *Proceedings of PHYSOR 2000*, Pittsburgh (2000).

10. H. Park and C.R.E. de Oliveira, “A Space-Angle Algebraic Multigrid Preconditioner for the Finite Element-Spherical Harmonics Method,” *Proceedings of ANS M&C 2005*, Avignon, France (2005).

11. H. Park and C.R.E. de Oliveira, “Adaptive Angular Resolution for the Finite Element-Spherical Harmonics Method,” *Transactions of the ANS*, **93**, pp. 517-519 (2005).

12. W.H. Reed, “New difference schemes for the neutron transport equation,” *Nuclear Science and Engineering*,**46**, pp. 31 (1971).

13. L. Roberts and D.Y. Anistratov, “Nonlinear Weighted Flux Methods for Particle Transport Problems,” *Proceedings of ANS M&C 2005¸* Avignon, France, pp. 10 (2005)

14. L.N. Trefethen, D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia (1997).