Accuracy of linear and quadratic finite elements when solving the Boltzmann transport equation

Brian Hesselmann
Studentnumber: 1234412

Bachelor Thesis

Supervisor: Dr. ir. Danny Lathouwers

Delft University of Technology
Physics of Nuclear Reactors

10 September 2010

Abstract

The goal of this paper is to determine quadratic finite elements are more accurate compared to linear finite elements when used for solving the Boltzmann transport equation. The problem of calculating the neutron flux in several simple reactor geometries by use of the finite element method is treated and in particular we investigate the use of second-order finite elements and compare their accuracy, based on the root mean square error, to first-order elements.
The elements are tested on several simple reactor geometries, using different element types and varying scattering –and absorbing cross-sections. A simple non-neutron transport case was first considered with Matlab and the main neutron transport problems were solved using an existing solver called Phantom, which was already in use and will now be expanded to use second-order elements.
Results show that second-order finite elements are much more accurate than first-order elements, showing a factor of magnitude increase in the order of 10^2 all the way up to 10^6.
We conclude that second-order elements are much more accurate and should be used to solve complex systems that would be otherwise harder or more time consuming to solve using only first-order elements. Future research should point out if and when it is more efficient and less time consuming to use second-order elements.

1 Introduction

In the field of neutron transport there is a need for efficient and accurate computational methods. One of these methods is the finite element method or FEM. Using the FEM it is possible to solve complex partial differential equations that appear in the this field. The goal of this report is to improve the accuracy of an already existing program, Phantom, that solves the first-order Boltzmann equation. Phantom currently uses first-order finite elements to solve the spatial part of the Boltzmann equation and in this report we hope to achieve an increase of accuracy and a reduction of required computation time by implementing second-order finite elements into Phantom. This will be calculated by using the root mean square error as a measure of the accuracy.

A large part of this project was spent becoming familiar with the theory behind the finite element method; as such we include a section explaining the basics of FEM. In addition a program was created in Matlab to test the finite elements in a simple test case before moving on to more complex cases that solve for the neutron flux. The Boltzmann equation that describes the transport of neutrons will also be briefly explained and we will show how to transform this into a suitable minimization problem for use with FEM using a least squares method.

2 The Neutron Transport Equation

The equation describing neutron transport in a single energy group is the first-order Boltzmann equation:

$$\mathbf{\Omega}\cdot\nabla\psi(\boldsymbol{r},\Omega)+\sigma\psi(\boldsymbol{r},\Omega)=\int\sigma_s(\Omega\cdot\Omega')\psi(\boldsymbol{r},\Omega')d\Omega'+S(\boldsymbol{r},\Omega), \tag{2.1}$$

where $\mathbf{\Omega}$ is the direction, ψ the neutron flux, σ the cross-section, $\boldsymbol{r}$ the space coordinate and S the source(We will assume the reader is familiar with the terms used here, for a background on nuclear reactor physics we refer to Duderstadt et al.(1976)[1]).
With the following boundary conditions:

$$\psi(\boldsymbol{r},\Omega)=0, \qquad \boldsymbol{r}\in\delta V,\mathbf{\Omega}\cdot\hat{\boldsymbol{n}}<0.$$

Equation (2.1) is a balance equation, where the left-hand side represents the outflow and the right-hand side the inflow. The first term on the LHS represents the net outflow, while the second term takes into account a decrease by interaction. On the RHS we have an increase by interaction in the first term and the second term represents a source. Equation (2.1) can be written simply as:

$$\mathbf{L}\psi=\mathbf{S}, \tag{2.2}$$

where $\boldsymbol{L}=\mathbf{\Omega}\cdot\nabla+\sigma-\sigma_s$. Rewriting this further into a more preferred form we reach:

$$\boldsymbol{L}\psi=\mathbf{\Omega}\cdot\nabla\psi+\mathrm{M}\psi \tag{2.3}$$

Where M includes total removal and scattering, and is written as:

$$M=\sigma(I-\kappa)\psi+\sigma_a\kappa\psi.$$

The scattering cross-section is then expanded in Legendre polynomials

$$\sigma_s(\mathbf{\Omega}\cdot\mathbf{\Omega}')=\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\sigma_{sl}P_l(\mu_0), \tag{2.4}$$

and equation(2.3) is rewritten as

$$\boldsymbol{L}\psi=\mathbf{\Omega}\cdot\nabla\psi+\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\sigma_l\int d\mathbf{\Omega}'P_l(\mu_0)\psi(\boldsymbol{r},\mathbf{\Omega}'). \tag{2.5}$$

Where $\sigma_l=\sigma-\sigma_{sl}$ and $P_l(\mu_0)$ are the Legendre polynomials defined as

$$\begin{aligned}P_0(\mu)&=1\\ P_n(\mu)&=\frac{1}{2^n n!}\frac{d^n}{d\mu^n}(\mu^2-1)^n.\end{aligned} \tag{2.6}$$

We will have control over the order of Legendre polynomials used when solving for the neutron flux, this will determine how much of the angular dependency will be included in the final result.

Next we want to solve equation (2.1), to do this we rewrite this equation into a minimization problem. From D. Lathouwers(2007)[2] we find that this can be done by using a least squares approach, resulting in the following functional:

$$\begin{aligned}F[\psi]=&\left\langle T^{-1}\mathbf{\Omega}\cdot\nabla\psi,\mathbf{\Omega}\cdot\nabla\psi\right\rangle+2\left\langle\mathbf{\Omega}\cdot\nabla\psi,T^{-1}M\psi\right\rangle+\left\langle T^{-1}M\psi,M\psi\right\rangle\\ &-2\left\langle\mathbf{\Omega}\cdot\nabla\psi,T^{-1}S\right\rangle-2\left\langle T^{-1}M\psi,S\right\rangle+2\iint_{\delta V\,\mathbf{\Omega}\cdot\hat{\boldsymbol{n}}}d\delta Vd\mathbf{\Omega}\left|\mathbf{\Omega}\cdot\hat{\boldsymbol{n}}\right|\psi^2,\end{aligned} \tag{2.7}$$

where T is the scaling operator. This functional is unwieldy and we will rework it in a reduced form. We will only give a brief description of the steps involve and will explain the result, for a more in depth view of this derivation we again refer to D. Lathouwers(2007)[2].
To start reworking the functional we separate the flux into its angular and spatial parts. Starting with expanding the angular part into a series of real spherical harmonic polynomials we have:

$$\psi(\boldsymbol{r},\boldsymbol{\Omega}) = \sum_{i=1}^{M} Q_i(\boldsymbol{\Omega})\psi_i(\boldsymbol{r}) = \boldsymbol{Q}^T(\boldsymbol{\Omega})\boldsymbol{\psi}(\boldsymbol{r}), \tag{2.8}$$

where the vector **Q** contains the sets of all spherical harmonic polynomials. This includes the Legendre polynomials used to describe the angular dependency of the cross-section in equation(2.5).

The spatial part is solved by using finite elements, subdividing the volume V into N_e elements such that:

$$V = \sum_{e=1}^{N_e} V_e, \qquad \delta V = \sum_{e=1}^{N_e} \delta V_e, \tag{2.9}$$

allowing us to rewrite $\boldsymbol{\psi}(\boldsymbol{r})$ as:

$$\boldsymbol{\psi}(\boldsymbol{r}) = \sum_{n=1}^{N} \psi_n \varphi_n . \tag{2.10}$$

Here ψ_n are the nodal moment vectors and φ_n are the basis functions, in the next section we will describe the finite element method in more detail.
We can now rewrite the functional into a reduced form as following:

$$F[\psi\ ] = \sum_{n=1}^{N}\sum_{n'=1}^{N} \psi_n^T \boldsymbol{L}_{nn'}^T \psi_{n'} - 2\sum_{n=1}^{N} \psi_n^T \boldsymbol{F}_n , \tag{2.11}$$

where $\boldsymbol{L}_{nn'}^T$ is:

$$\begin{aligned} \boldsymbol{L}_{nn'}^T = {} & \sum_{ii'} \int_V dV \boldsymbol{A}_{ii'}^{gg} R_i \varphi_n(\boldsymbol{r}) R_{i'} \varphi_{n'}(\boldsymbol{r}) + 2\sum_{\mathrm{i}} \int_V dV \boldsymbol{G}_i \boldsymbol{H}^g R_i \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) \\ & + \int_V dV \boldsymbol{H}^g \beta^{gg} \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) + 2\int_{\delta V} d\delta V \boldsymbol{E} \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) \end{aligned} \tag{2.12}$$

Taking equation (2.11) and requiring the functional to be stationary gives us the following set of linear equations:

$$\boldsymbol{A}_{\boldsymbol{nn'}}^{\boldsymbol{g}} \psi = \boldsymbol{F}_{\boldsymbol{n}}, \tag{2.13}$$

where the matrix $\boldsymbol{A}_{\boldsymbol{nn'}}^{\boldsymbol{g}}$ is represented by the equation

$$\begin{aligned} \boldsymbol{A}_{nn'} = {} & \sum_{ii'} \int_V dV \boldsymbol{A}_{ii'}^{gg} R_i \varphi_n(\boldsymbol{r}) R_{i'} \varphi_{n'}(\boldsymbol{r}) \\ & + \int_V dV \boldsymbol{G}_i \boldsymbol{H}^g R_i \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) \\ & + \int_V dV \boldsymbol{H}^g \boldsymbol{G}_{\boldsymbol{i}}^T R_i \varphi_{n'}(\boldsymbol{r}) \varphi_n(\boldsymbol{r}) \\ & + \int_V dV \boldsymbol{H}^g \beta^{gg} \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) \\ & + 2\int_{\delta V} d\delta V \boldsymbol{E} \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) \end{aligned} \tag{2.14}$$

and $\boldsymbol{F_n}$ as

$$F_n = \sum_{n'} \int_V dV \sum_i \boldsymbol{G}_i \boldsymbol{\tau}^{g-1} \boldsymbol{S}_n^{'} R_i \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) + \sum \int_V dV \sum_i \boldsymbol{H}^g \boldsymbol{S}_{n'} \varphi_n(\boldsymbol{r}) \varphi_{n'}(\boldsymbol{r}) . \qquad (2.15)$$

These equations still correspond to our original least squares functional from equation(2.7). Equation (2.14) and (2.15) now contain the angular parts of the Boltzmann equation in the abbreviations $\boldsymbol{A}_{ii'}^{gg}$,$\boldsymbol{G}_i$, $\boldsymbol{H}^g$ and $\boldsymbol{E}$. The β^{gg} in equation(2.14) tells us which cross-section we should use and is defined as

$$\beta^{gg} = \begin{cases} \sigma_l^g, & \text{if } g'=g \\ \sigma_{sl}^{g'g}, & \text{if } g' \neq g \end{cases} . \qquad (2.16)$$

The spatial part of is solved using the test functions φ_n as mentioned before and in the next section we will describe what these basis functions look like and how we use them to solve an equation of the form as equation(2.13).

3 The Finite Element Method

To solve the spatial part of the first-order Boltzmann equation we use finite elements. This section will briefly explain the theory behind the finite element method, both for first-order and second-order finite elements.
To introduce the finite element method we will use Figure 3-1 and Figure 3-2. In Figure 3-1 we see a certain function in blue being approximated by a red function consisting of several lines, what we want to do with the finite element method is solve this approximate solution for a difficult partial differential equation numerically. In Figure 3-1 we also notice that the X-axis has been divided in multiple sections, labeled x_0 through x_5. These sections are what we call elements and the values x_0 through x_5 are called nodes. For example, in Figure 3-1 the first element would reach from x_0 to x_1 and contains 2 nodes.
In Figure 3-2 we see again the approximate red solution and this time we also see a number of blue functions, known as basis functions or "tent" functions. These "tent" functions are known and will be used to construct to approximate solution.

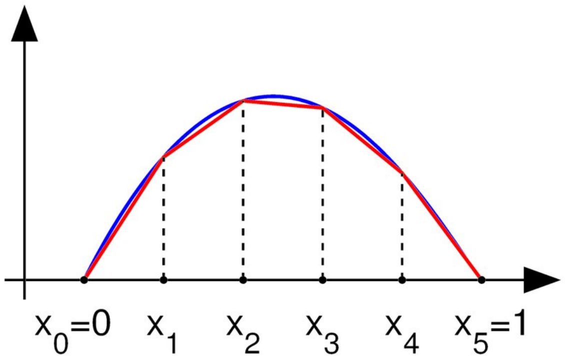


Figure 3-1A function(blue) with it's approximate solution(red) on a interval between x=0 and x=5 divided in 5 elements using 6 nodes.

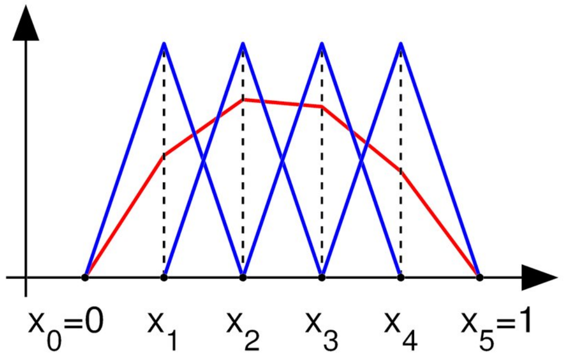


Figure 3-2The same approximate function(red) as in Figure 3-1 this time showing the "tent" functions(blue) used to construct this solutions on the some interval.[3]

To apply the finite element method to a partial differential equation we first need to rewrite it in the weak form, then find the appropriate basis functions to use and finally solve it numerically. In the following section we will use the Poisson equation $\nabla^2 u = f$ as an example to clarify the theory of finite elements.

3.1 *Solving the 1D Poisson equation with the finite element method*

We will start with the 1D Poisson equation with homogeneous boundary conditions,

$$\nabla^2 u = f(x) \text{ with } u = 0 \text{ on the boundary,} \tag{3.1}$$

where f(x) is given.
To rewrite the Poisson equation in the weak form we start by introducing $U(x)$ as an approximation for u as:

$$u(x) \approx U(x) = \sum_{j=1}^{n} U_j \varphi_j(x) \tag{3.2}$$

Where n is the number of nodes, U_j are unknown coefficients and $\varphi_j(x)$ are the "tent" functions defined by the following rules:

1. $\varphi_j(x)$ is linear in each element.
2. $\varphi_j(x_i) = \delta_{ij}$, meaning zero at all nodes except their own.

To give a little more insight in the basis functions, one has been sketched in Figure 3-3. The function $\varphi_j(x)$ has been plotted over two elements and three nodes. It is easy to see that $\varphi_j(x)$ is linear in both elements, and $\varphi_j(x)$ is equal to one when above the nodal point x_j and zero for the two other nodes.

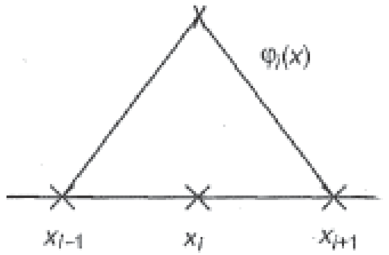


Figure 3-3"Tent" function consisting of 2 basis functions over two elements containing three nodes.[1]

The approximation in (3.2) will not be able to satisfy the partial differential equation in (3.1). Instead we determine the n different coefficients U_j and insist that the approximation satisfies the n conditions known as the weak form of the partial differential equation. To do this we insist that the left –and right-hand sides of the partial differential equation when multiplied by the basis function $\varphi_i(x)$ are equal:

$$\int \nabla^2 u\varphi_i dx = \int f\varphi_i dx \qquad i=1...n . \tag{3.3}$$

We can simplify this equation by integrating by parts using $\nabla^2 u\varphi_i = \nabla\cdot(\varphi_i \nabla u) - \nabla\varphi_i \cdot \nabla u$ to arrive at:

$$\int \left[\nabla\cdot(\varphi_i\nabla u) - \nabla\varphi_i\cdot\nabla u\right]dx = \int f\varphi_i dx . \tag{3.4}$$

Simplifying this further by applying the divergence theorem($\int \nabla\cdot \boldsymbol{P}(x)dx = P(b) - P(a)$) and knowing that the function is zero at the boundaries we find:

$$\int \nabla\varphi_i\cdot\nabla u dx = -\int f\varphi_i dx . \tag{3.5}$$

Equation (3.5) is called the weak form(Galerkin) of the partial differential equation. All solutions u(x) satisfy the weak form equation (3.5) and our approximation in equation (3.2) should also satisfy the weak form:

$$\sum_{j=1}^{n} U_j \int \nabla\varphi_j\cdot\nabla\varphi_i dx = -\int f\varphi_i dx . \tag{3.6}$$

Now we introduce a new matrix S_{ij}:

$$S_{ij} = \int \nabla\varphi_i\cdot\nabla\varphi_j dx , \tag{3.7}$$

where $S_{ij}=S_{ji}$. S_{ij} is a square n x n matrix which is mostly empty because of the properties of the basis functions. Next we introduce a vector F_i:

$$F_i = -\int f\varphi_i dx , \tag{3.8}$$

which allows us to further rewrite equation (3.6) to:

$$\sum_{j=1}^{n} S_{ij}U_j = F_i \,. \tag{3.9}$$

In equation (3.9) it is clear that we have n equations with n unknowns(the U_j). This equation can also be written in matrix form:

$$\boldsymbol{S}U = F \,. \tag{3.10}$$

And finally our solution is:

$$U = \boldsymbol{S}^{-1}F \,. \tag{3.11}$$

3.1.1 The 2D case

Solving the two dimensional Poisson equation using finite elements is much the same as the one dimensional case. We slightly adjust our partial differential equation (3.1) as follows:

$$\nabla^2 u = f(x,y) \text{ with } u(x,y) = 0 \text{ on the boundary}. \tag{3.12}$$

The approximation in equation (3.2) holds except for a change in dimension:

$$u(x,y) \approx U(x,y) = \sum_{j=1}^{n} U_j \varphi_j(x,y) \tag{3.13}$$

only this time our elements no longer consist of just lines, but are usually triangles or squares. Again we use the weak form by integrating and multiplying both sides of equation (3.12) by the basis function to arrive at:

$$\int\int \nabla^2 u \varphi_j dA = \int\int f \varphi_j dA \,. \tag{3.14}$$

Again using integration by parts, the divergence theorem($\iint \nabla \cdot \boldsymbol{P} dA = \oint \boldsymbol{P} \cdot \hat{\mathbf{n}} ds$) and the knowledge that the boundary conditions are zero to reduce equation (3.14) to:

$$\sum_{j=1}^{n} U_j \int\int \nabla \varphi_i \cdot \nabla \varphi_j dA = -\int\int f \varphi_i dA \tag{3.15}$$

Defining our S_{ij} matrix and vector F_i again allows us to arrive again at equation (3.10) in matrix form, with solution equal to equation(3.11).
Although the resulting equations are again the same as for the one dimensional case, the solutions of these two are of course nothing alike.

3.1.2 Implementing Boundary Conditions

In the previous sections we have assumed homogeneous boundary conditions equal to zero, this is not usually the case however and we would like to include other boundary conditions. Because we will solve equation (3.10) numerically there is an easy way to include the boundary conditions. As an example we will use a 1-D region of length 10 with the following Dirichlet boundary conditions:

$$u(0) = a$$

$$u(10) = b$$

For a simple system containing only 3 elements the S_{ij} matrix will be 4x4. For this example we will choose a simple matrix and use it to rewrite equation (3.10) as following:

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} F1 \\ F2 \\ F3 \\ F4 \end{bmatrix} = \begin{bmatrix} U1 \\ U2 \\ U3 \\ U4 \end{bmatrix}$$

From this we can insert our boundary conditions. We know that U1 in this equation gives the solution at u=0, so if we replace the first row by:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} a \\ F2 \\ F3 \\ F4 \end{bmatrix} = \begin{bmatrix} U1 \\ U2 \\ U3 \\ U4 \end{bmatrix}$$

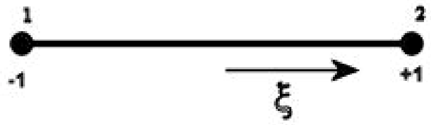
And doing the matrix multiplication we find U1=a, solving the first boundary condition. The second boundary condition can be solved in much the same way. This time replacing the bottom row of the system matrix by zeros except for the last column and replacing the last value of the vector F by b.

3.2 First-Order Basis Functions

The driving force behind the finite element method is the use of basis functions; these functions allow us to solve many partial differential equations in the same way. We would like to show some of these basis functions for both one and two dimensional problems. The functions given in this section are all first-order functions, meaning that they are linear in both the x –and y-direction.
The simplest basis function is the one for the 1-D line element, shown below with its basis functions:[2]

1-D Line Element

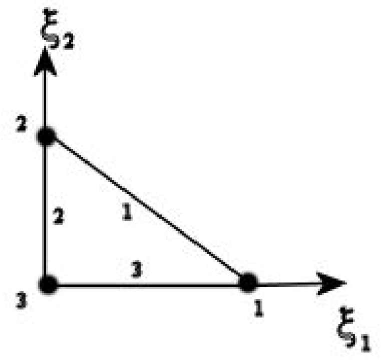
$$\varphi^1(\xi) = 0.5(1-\xi)$$
$$\varphi^2(\xi) = 0.5(1+\xi)$$



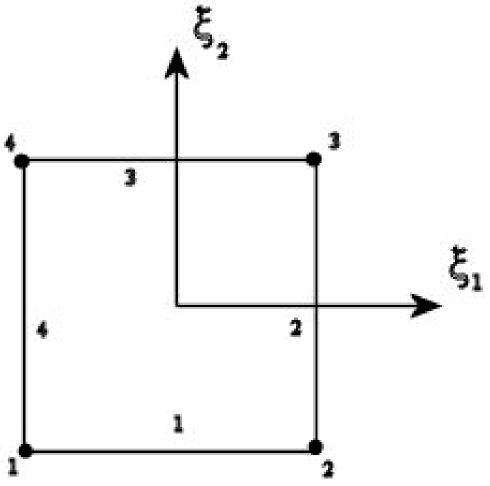
There are a few things to note about this element. First of all we see that a new coordinate has been introduced: ξ. From now on we will use ξ as the local coordinate in each element, this is not the actual location of the element in the region of interest but a special coordinate system that will be mapped to the actual coordinate. All elements will range from -1 to +1 in the ξ coordinate; this is convenient for numerical integration which we will discuss more in depth later on in section 3.4. Finally we note the 2 basis functions φ^1 and φ^2, one for each node inside the element. Checking the rules given in the previous section, we see that these functions are indeed linear and zero at all nodes except for their own.
In 2-D there are other elements to consider, we will give the basis functions for the triangular and quadrilateral elements. We'll also give the node and face numbering as used in this report. The faces are numbered on the inside of the element and are used to determine which side is facing which direction.

2-D Triangular Element

$$\varphi^1 = \xi_1$$
$$\varphi^2 = \xi_2$$
$$\varphi^3 = 1 - \xi_1 - \xi_2$$



2-D Quadrilateral Element



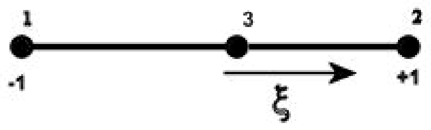
$$\varphi^1 = 0.25(1-\xi_1)(1-\xi_2)$$
$$\varphi^2 = 0.25(1+\xi_1)(1-\xi_2)$$
$$\varphi^3 = 0.25(1+\xi_1)(1+\xi_2)$$
$$\varphi^4 = 0.25(1-\xi_1)(1+\xi_2)$$

3.3 Second-order Basis Functions

In the previous section we showed an example of linear basis functions, it's also possible to use quadratic elements that include second-order polynomials in their basis function. These elements will be referred to as second-order elements. The second-order elements used in this report are the 3-node line element, the 6-node triangular element, the 8-node quadrilateral element and the 9-node quadrilateral element. These elements are given below with their basis functions.

3-Node Line Element

$$\varphi^1 = -0.50\xi(1-\xi)$$
$$\varphi^2 = 0.5\xi(1+\xi)$$
$$\varphi^3 = (1-\xi)(1+\xi)$$



6-Node Triangular Element

$\varphi^1 = (2\xi_1 - 1)\xi_1$

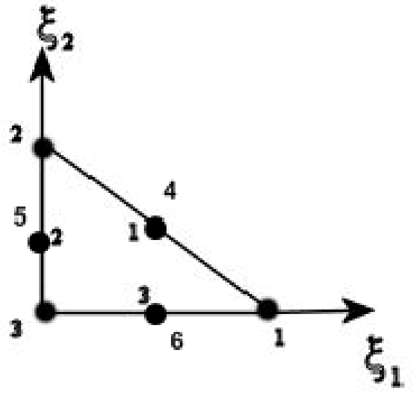
$\varphi^2 = (2\xi_2 - 1)\xi_2$

$\varphi^3 = (2(1-\xi_1-\xi_2)-1)(1-\xi_1-\xi_2)$

$\varphi^4 = 4\xi_1\xi_2$

$\varphi^5 = 4\xi_2(1-\xi_1-\xi_2)$

$\varphi^6 = 4\xi_1(1-\xi_1-\xi_2)$



8-Node Quadrilateral Element

$\varphi^1 = -0.25(1-\xi_1)(1-\xi_2)(1+\xi_1+\xi_2)$

$\varphi^2 = 0.25(1+\xi_1)(1-\xi_2)(\xi_1-\xi_2-1)$

$\varphi^3 = 0.25(1+\xi_1)(1+\xi_2)(\xi_1+\xi_2-1)$

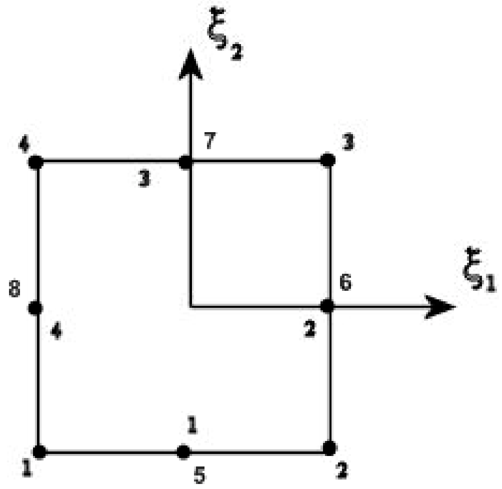
$\varphi^4 = 0.25(1-\xi_1)(1+\xi_2)(-\xi_1+\xi_2-1)$

$\varphi^5 = 0.50(1-\xi_1^{\,2})(1-\xi_2)$

$\varphi^6 = 0.50(1+\xi_1)(1-\xi_2^{\,2})$

$\varphi^7 = 0.50(1-\xi_1^{\,2})(1+\xi_2)$

$\varphi^8 = 0.50(1-\xi_1)(1-\xi_2^{\,2})$



9-Node Quadrilateral Element

$\varphi^1 = 0.25 * \xi_1 * \xi_2 (1-\xi_1)(1-\xi_2)$

$\varphi^2 = -0.25 * \xi_1 * \xi_2 (1+\xi_1)(1-\xi_2)$

$\varphi^3 = 0.25 * \xi_1 * \xi_2 (1+\xi_1)(1+\xi_2)$

$\varphi^4 = -0.25 * \xi_1 * \xi_2 (1-\xi_1)(1+\xi_2)$

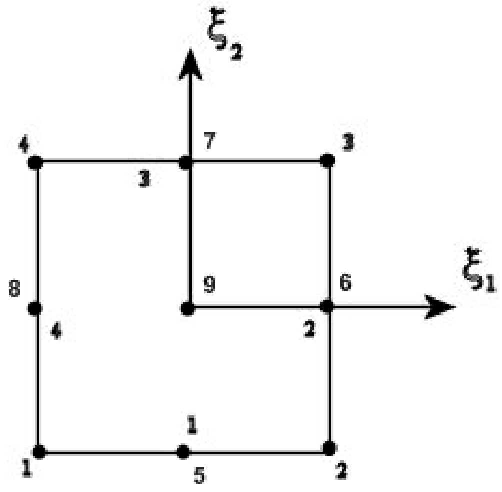
$\varphi^5 = -0.50 * \xi_2 (1-\xi_2)(1-\xi_1^{\,2})$

$\varphi^6 = 0.50 * \xi_1 (1+\xi_1)(1-\xi_2^{\,2})$

$\varphi^7 = 0.50 * \xi_2 (1+\xi_2)(1-\xi_1^{\,2})$

$\varphi^8 = -0.50 * \xi_1 (1-\xi_1)(1-\xi_2^{\,2})$

$\varphi^9 = (1-\xi_1^{\,2})(1-\xi_2^{\,2})$



3.4 Numerical integration

We now have all the tools required to solve our partial differential equation (3.1), however we still need a method to solve this system numerically. To create our matrix S_{ij} and vector F_i we need to evaluate their respective integrals. To this end we apply a numerical approximation of the integrals using Gaussian quadrature.
When using Gaussian quadrature we use the following approximation to evaluate an integral numerically:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i), \tag{3.16}$$

where f(x) is some function, $f(x_i)$ is the function f(x) evaluated at some specific sampling point x_i and w_i is the weight ascribed to that point. These points x_i and their weights w_i are unknown, but have been tabulated in various books and those used in this paper have been presented in *use only for quadrilateral elements
Table 3-1.

We will not give a complete overview of how to find these sampling points and weights, but will give an example of how to use them and refer to Griffiths et al.(2006)[4] for those who want to learn more about numerical integration methods.
The accuracy of the numerical integration increases as we use more sampling points, however the computation time required also increases when we add more sampling points. As a rule the Gaussian quadrature is accurate for polynomials of order 2n-1 where n is the number of sampling points used. Numerical integration using Gaussian quadrature can also be used in 2D. *use only for quadrilateral elements
Table 3-1 already shows the appropriate sampling points in that case.
Important to note is that the weights and sampling points in *use only for quadrilateral elements
Table 3-1 can only be used for integrals with boundaries from -1 to 1. Obviously this is not usually the case for the integrals that we would like to solve. Therefore we will introduce a way to transform the limits of any integral to the ones needed. If we would like to transform the following integral:

$$\int_{a}^{b} F(t)dt \equiv \int_{-1}^{1} f(x)dx \tag{3.17}$$

We would use

$$t = \frac{(b-a)x+(b+a)}{2} \tag{3.18}$$

and

$$dt = \frac{(b-a)}{2}dx\,. \tag{3.19}$$

This allows us to use the Gaussian quadrature on any integral after transforming it to the correct boundaries.

As an example we consider the following integral:

$$\int_{0}^{2} t^2 dt\,,$$

first of all we'll need to transform this integral to have the correct limits. Using equations (3.17) and (3.18) we arrive at:

$$\int_0^2 t^2 dt = \int_{-1}^{1} \left(x^2 + 2x + 1\right) dx,$$

now we can use equation (3.16). Because we are integrating a quadratic function we shall use 2 sampling points:

$$\int_{-1}^{1} \left(x^2 + 2x + 1\right) dx \approx w_1 f\left(x_1\right) + w_2 f\left(x_2\right).$$

If we fill in the correct sampling points and weights we can calculate the following:

$$w_1 f\left(x_1\right) + w_2 f\left(x_2\right) = 2.666666666666667 \approx \frac{8}{3},$$

in this case $\frac{8}{3}$ is also the exact solution of the integral. Apparently for this simple example integral the numerical integration is exact.

n	x_i	y_i	w_i
2	-0,577350269189626	-	1,000000000000000
	0,577350269189626	-	1,000000000000000
3	-0,774596669241484	-	0,555555555555556
	0,000000000000000	-	0,888888888888889
	0,774596669241484	-	0,555555555555555
4	-0,891136311594053	-	0,347854845137454
	-0,339981043584856	-	0,652145154862546
	0,339981043584856	-	0,652145154862546
	0,891136311594053	-	0,347854845137454
4*	-0,577350269189625	-0,577350269189625	1,000000000000000
	0,577350269189625	-0,577350269189625	1,000000000000000
	-0,577350269189625	0,577350269189625	1,000000000000000
	0,577350269189625	0,577350269189625	1,000000000000000
6	0,445948490915965	0,445948490915965	0,111690794839005
	0,445948490915965	0,108103018168070	0,111690794839005
	0,108103018168070	0,445948490915965	0,111690794839005
	0,091576213509771	0,091576213509771	0,054975871827661
	0,091576213509771	0,816847572980458	0,054975871827661
	0,816847572980458	0,091576213509771	0,054975871827661
7	0,101286507323456	0,101286507323456	0,062969590272414
	0,797426985353087	0,101286507323456	0,062969590272414
	0,101286507323456	0,797426985353087	0,062969590272414
	0,470142064105115	0,059715871789770	0,066197076394253
	0,470142064105115	0,470142064105115	0,066197076394253
	0,059715871789770	0,470142064105115	0,066197076394253
	0,333333333333333	0,333333333333333	0,112500000000000
9*	-0,774596669241483	-0,774596669241483	0,296296296296296
	-0,774596669241483	0,774596669241483	0,296296296296296
	0,774596669241483	-0,774596669241483	0,296296296296296
	0,774596669241483	0,774596669241483	0,296296296296296
	0,000000000000000	-0,774596669241483	0,493827160493827
	-0,774596669241483	0,000000000000000	0,493827160493827
	0,000000000000000	0,774596669241483	0,493827160493827
	0,774596669241483	0,000000000000000	0,493827160493827
	0,000000000000000	0,000000000000000	0,790123456790123

*use only for quadrilateral elements

Table 3-1Sampling points and weights for normalized Gaussian quadrature. Data from: n=2,3,4: Griffiths et al(2006), Page260[5];n=6, 7: Guermond et al(2004), page360[6];4*,9*:already present in Phantom code.

4 Testing the finite element method in Matlab

To get a better grip on the finite element method a Matlab program was written to solve differential equations using finite elements. This program solves equations of the form:

$$D\frac{d^2u}{dx^2}+\Sigma_a u = f\left(x\right), \tag{4.1}$$

in one dimension given a function f(x) and using Neumann boundary conditions. This equation resembles a neutron transport equation where Σ_a can be interpreted as a cross-section and D as a diffusion constant.

The program includes both first –and second order finite elements and automatically calculates the root mean square error between the (known) analytical solution and the numerical solution of the chosen differential equation.

4.1. Error calculation for determining the precision of the finite element method

One of the main reasons to develop this program was to test the accuracy of second order elements compared to first order elements. To determine the accuracy of a given method the root mean square error(RMSE) was used, given by formula(4.2):

$$RMSE=\sqrt{\frac{1}{n}*\sum_{all\ \text{n}}\left\{\left[u_{num}\left(x_n\right)-u_{an}\left(x_n\right)\right]^2\right\}} \tag{4.2}$$

Where n is the total number of nodes used in calculating the numerical solution, $u_{num}(x_n)$ and $u_{an}(x_n)$ are the numerical and analytical solution at node n respectively.

What formula(4.2) calculates is amount by which the analytical and numerical solutions differ to give a measure of the precision of the numerical method. This error calculation is repeated multiple times, each time using more elements, and thus more nodes, to create a finer mesh. We expect to see a more accurate numerical solution and a lower RMSE as the mesh is increasingly finer. The amount of elements in this test case ranges from 25 to 400, starting from a coarse 25 element mesh and doubling until a much finer 400 element mesh is reached. In total 10 measurements of the RMSE are found, 5 for both the first –and second order elements.

As the number of elements increases the size of each individual element decreases. From J. van Kan et al.[5] we expect a relation between the size of the elements used and the order of element used. Using first-order elements we expect the error to decrease with ch^2 where h is the element size and c is some constant, while using second-order elements we expect the error to decrease with ch^3. To confirm our expectations all error data will be plotted in a log scale to easily determine the slope of the curve, in this case we expect a slope of ~2 for first order elements and a slope of ~3 for second order elements. The slope will be approximated from the numerical data by using $slope = \frac{\Delta y}{\Delta x}$, which we will calculate using our last two data points.

4.2. Results for solving the Poisson equation

We modify equation (4.1) by choosing D=1 and $\Sigma_a = 0$, thus reducing it to the Poisson equation. The Poisson equation is relatively simple to solve and was solved for many different functions f(x), below the results for two of these functions will be given. Namely:

$$\begin{gathered} f(x) = x^3 \\ f(x) = x^2 \cos(x) \end{gathered},$$

with the following boundary conditions:

$$u(0) = 0;\ u(10) = 0.$$

For $f(x) = x^3$ the analytical and numerical solution are plotted below in Figure 4-1 for a solution with 400 second-order elements. The analytical solution for $\frac{d^2u}{dx^2} = x^3$ is:

$$u_{analytical} = -\frac{1}{20}\mathrm{x}^5 + 501\mathrm{x}.$$

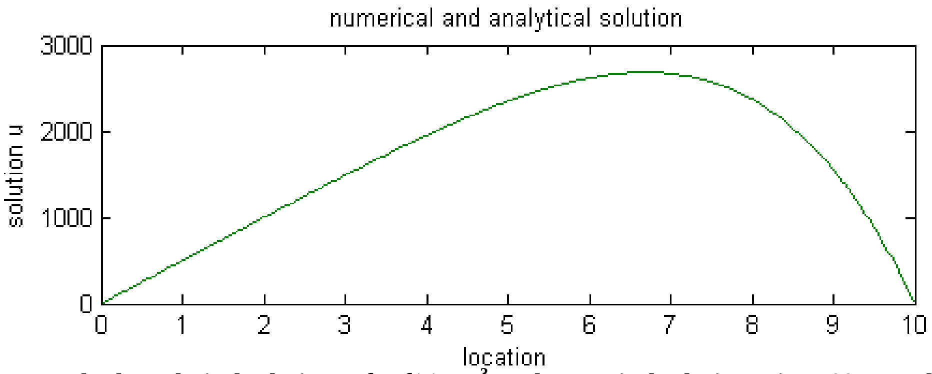


Figure 4-1 both analytical solution u, for f(x) = x^3, and numerical solution using 400 second-order elements

From Figure 4-1 it is clear that the numerical and analytical solutions are very close together, they are indiscernible in the graph. What we do learn from this is that the error between the two solutions should be very small when using 400 elements.

The second result that will be discussed is the solution of $\frac{d^2u}{dx^2} = x^2 \cos(x)$. The analytical solution in this case is:

$$\mathrm{u}_{\mathrm{analytical}} = -4\mathrm{x}*\sin(\mathrm{x}) + (\mathrm{x}^2-6)*\cos(\mathrm{x}) + \frac{4-94\cos(10)+40\sin(10)}{10}\mathrm{x} + 6.$$

Figure 4-2 shows the solution for second-order finite elements. As before we are unable to distinguish the analytical and numerical solution, the error between the solutions is too small to see.

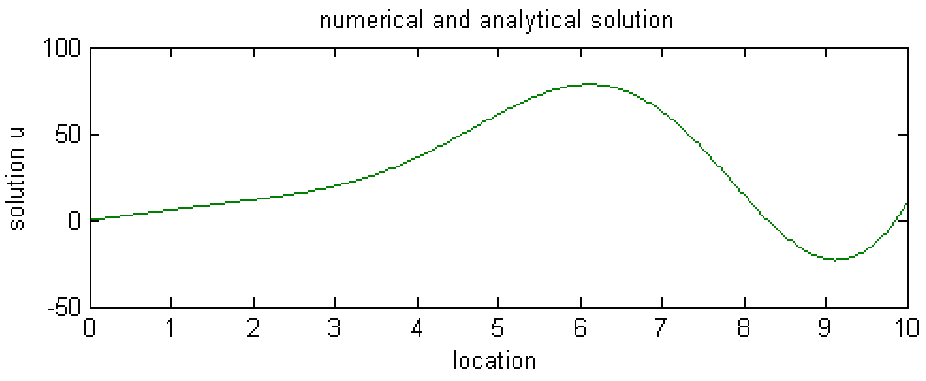


Figure 4-2 both analytical solution u, for f(x) = x^2cos(x), and numerical solution using 400 second-order elements

The RMSE was calculated for both these cases and was in the range of 10^{-12}-10^{-13}, indicating that the finite element method is, for all intents and purposes, exact in these cases and the error here is almost entirely represented by rounding errors.

4.3. Solving a neutron transport like differential equation

Again we take equation(4.1), but this time we use:

$$f(x) = \left(\Sigma_a + \left(\frac{\pi}{10}\right)^2\right)\sin\left(\frac{\pi x}{10}\right)$$
$$D = 1 \qquad .$$
$$\Sigma_a = 1$$

This introduces a new u dependant term into our equation, which now looks like this:

$$\frac{d^2u}{dx^2} + u = \left(1 + \left(\frac{\pi}{10}\right)^2\right)\sin\left(\frac{\pi x}{10}\right), \tag{4.3}$$

And has the following boundary conditions:

$$u(0) = u(10) = 0 .$$

The analytical solution for this equation is:

$$u(x) = \sin\left(\frac{x\pi}{10}\right).$$

To apply the finite element method to this equation we again use the procedure described in section3. Doing this we arrive at the following expression for the system matrix S:

$$S_{ij} = \int\left[\nabla\varphi_i \cdot \nabla\varphi_j + \varphi_j \cdot \varphi_i\right]dx . \tag{4.4}$$

The other expressions remain the same and we will still solve equation(3.11) as before only substituting the new matrix from equation(4.4).

In Figure 4-3 we see both the analytical and the numerical solution of equation(4.3). Again the difference is impossible to see in the figure, however the error analysis will show that the error is much larger in this slightly more complex case compared to the Poisson case.

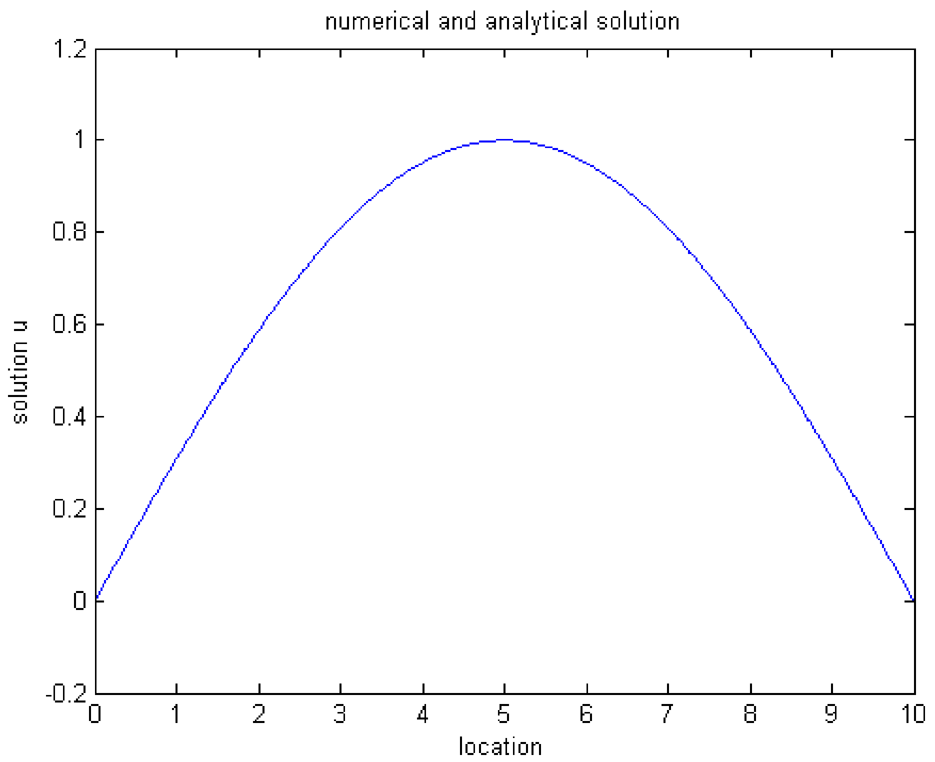


Figure 4-3both analytical solution u and numerical solution using 400 second-order elements

In Figure 4-4 we see the RMSE for both first –and second-order elements plotted against an increasing number of elements. The results show a slope of ~2 for the first-order elements and a slope that is almost 4 for the second-order elements. From this we learn that the second-order elements perform even better than initially expected. The overall increase in accuracy for the second-order elements over the first-order elements for the same number of elements used is also significant. Second-order elements increase the accuracy by as much as a factor of 10^6.

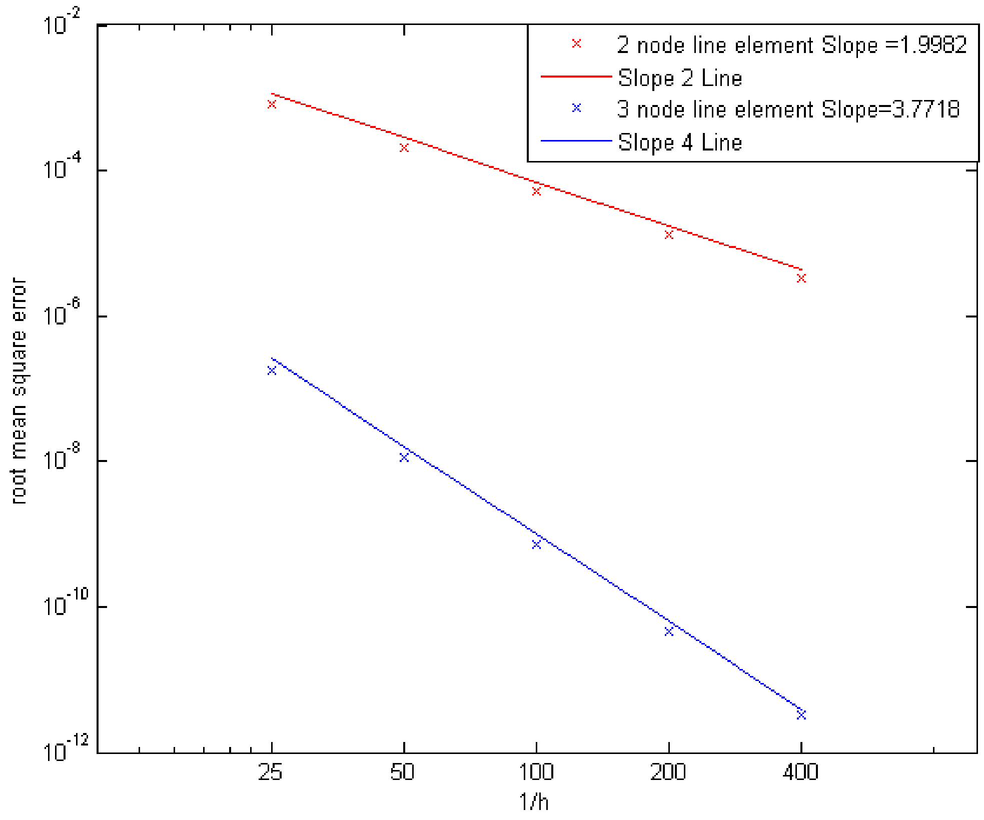


Figure 4-4Root mean square for linear and quadratic line element. Including two additional lines to illustrate the slope of the RMSE.

4.4. Conclusions from Matlab case

By solving a simplified equation in one dimension we were able to test the method of finite elements. From the results obtained in this case it is clear that an increase in the number of elements leads to a much more accurate result.
The most important conclusion from this case is that the second-order elements are far superior in accuracy to the first-order elements. Comparing the slopes of the curves in these cases shows that the accuracy of second order elements exceeds expectations.

5 The one dimensional Reed problem

In this section we will solve the Boltzmann transport equation for a very specific geometry known as the Reed geometry. According to A.G. Buchan et al.[8] the Reed problem is known as "a demanding test case for transport codes" and "has regions in which the angular flux distribution takes isotropic, highly peaked and intermediate forms". Because of these factors the Reed problem is an excellent benchmark for testing the accuracy of the solver.

In this case the Phantom code mentioned previously will be used to solve the Boltzmann equation in the Reed geometry. Points of interest are again the accuracy of second order elements compared to, the known to be correct, first order elements.

5.1 Description of the Reed geometry

The Reed geometry consists of 5 regions of varying materials and cross-sections. The regions lie within a reactor lattice cell at the edge of a bare core. In a diagram of the Reed region has been given. Every region occurs twice in the figure, as the Reed geometry is symmetric.

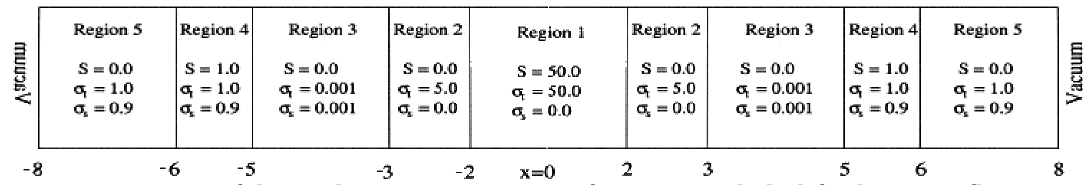


Figure 5-1 Diagram of the Reed geometry, consisting of 5 regions with the left edge on a reflective boundary and the right edge on a vacuum boundary. Cross-sections and sources are given in the diagram.

The first region lies in the middle of the geometry and contains an optically thick material. It also contains an isotropic neutron source. The second region is a 1cm fuel can, region 3 is a 2cm void region. Regions 4 and 5 are moderators of 1cm and 2cm respectively, while region 5 is without a source and lies next to a vacuum boundary.

The geometry has a total length of 16cm and contains vacuum boundaries on both edges. The geometry used in the solver is scaled to a unit length. Because the entire cell has a length of 16cm, all the sources and cross-sections have been scaled with a factor of 16 in the solver and the region now starts at x=-0.5 and ends at x=0.5.

5.2 Results and error calculation for the Reed problem

The Boltzmann transport equation was solved for the Reed geometry using the Phantom code. The scalar part was solved using both first –and second order finite elements to compare the accuracy between the two, additionally the equation was solved using first, fifth and ninth order Legendre polynomials as described in equation(2.6).
The number of elements starts from 128 and doubles until 4096 equidistant elements are reached. The choice for the final amount of elements was made by looking at the convergence

of the solutions. It was computed that the difference between the solutions for 2048 and 4096 was smaller than 10^{-7}; further increase in the number of elements hardly decreased this difference while sharply increasing the computation time.

First we'll look at the solutions found using first order elements, these solutions are known to be correct from previous tests with the Phantom code. In Figure 5-2 the scalar neutron flux is plotted, we can clearly see the 5 regions present in the Reed geometry. Starting from the middle at x=0.000cm where region 1 is, we notice that the neutron flux is almost constant only dropping at the edges of region 1. The cause for this is region 1 being optically thick, with the mean free path being much smaller than the diameter of the region and the balance between absorption and source resulting in a flat profile. Moving to the right we move into region 2, here we see a steep decline in the scalar flux because of the absorbing cross-section present in this region. Region 3 again shows a constant flux as it is a vacuum region. In region 4 we see an increase in the flux due to the source present and finally a sharp decrease in the flux as we enter region 5 as the source term is gone and we come closer to the vacuum boundary.

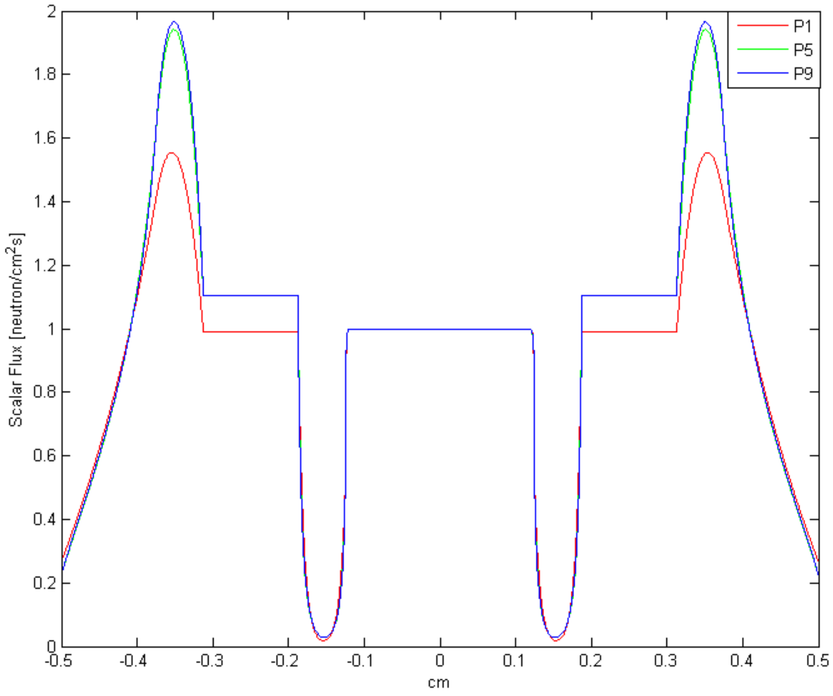


Figure 5-2Neutron flux in the Reed region, results for 4096 first order elements. Solutions plotted for 1(P1), 5(P5) and 9(P9) polynomials.

Looking at the difference in the solutions with respect to the order of angular polynomials used in solving them, we can see from that the solution for the first-order polynomial(P1) is lower than the solutions for the higher order polynomials P5 and P9. The cause for this is that some regions of the Reed geometry require a higher angular resolution to be able to correctly determine the neutron flux. In the constant flux regions the different solutions are much closer together than in the regions were the flux is rapidly increasing or decreasing. From this we note that for problems were the flux is expected to have a large angular dependency we should use an appropriate order of polynomials to ensure an accurate solution.

Next we'll look at the solutions found for second order elements. The results for this case are indistinguishable from the results with first-order elements, and we refer to Figure 5-2 for the results. What we can conclude is that the results for the second-order case are correct, the shape and value of the scalar flux are incredibly close to those of the first-order elements which were already known to be correct.

Finally we want to determine the accuracy of the two methods, for this we use the root mean square error as described in section 4.1 with the solution for 4096 second-order elements as the “true” solution. This solution will be treated as if it were the exact analytical solution.
In Figure 5-3 the RMSE is plotted for first order elements and 3 different orders of angular polynomials. The slope of the three curves has been approximated by using the last two points of the RMSE. By comparing the results with the slope 2 line plotted in blue it is clear that the slope of the RMSE is increasing for more elements. The difference in slope between the different polynomials is small, noting that the lowest order polynomial has the largest increase in accuracy when the number of elements increases.
Before drawing conclusions we will first consider Figure 5-4 where the RMSE for second-order elements has been plotted for the same angular polynomials and elements as in Figure 5-3. Again we note the slope of the 3 curves, which is considerably higher than in the first-order case. The RMSE when compared to the first-order elements is also much lower, dropping as low as 10^{-6} whereas the first-order elements don’t go below 10^{-4}. This is especially significant, considering that the amount of first-order elements used to obtain the most accurate value for the RMSE was twice as big as the number of second-order elements used to obtain a better accuracy.

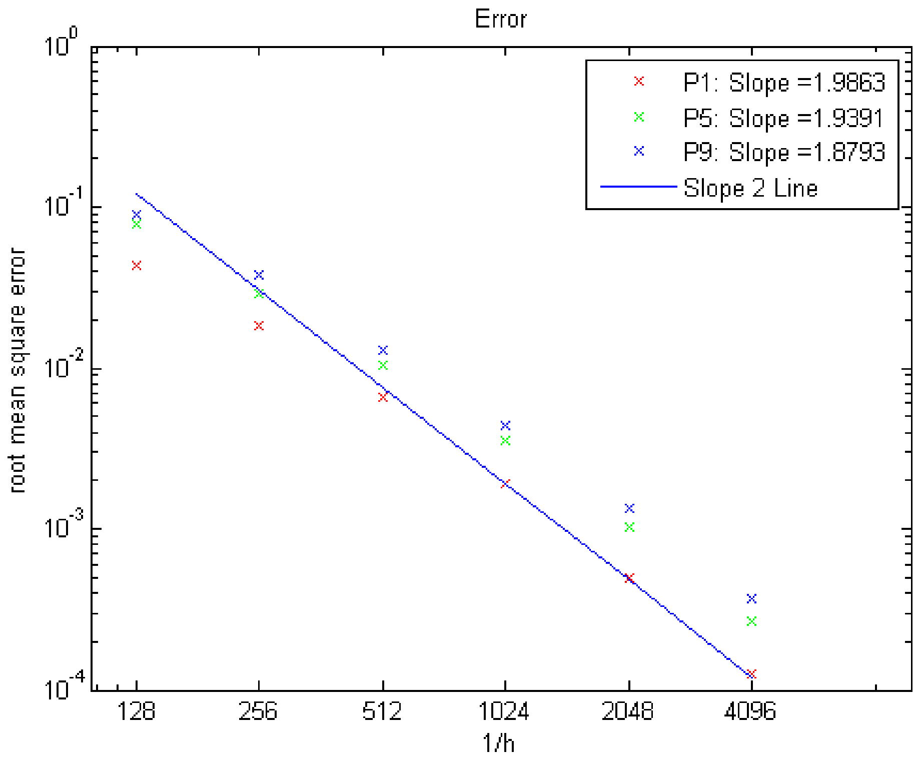


Figure 5-3 RMSE for first, fifth and ninth order polynomials and increasing numbers of first order elements.

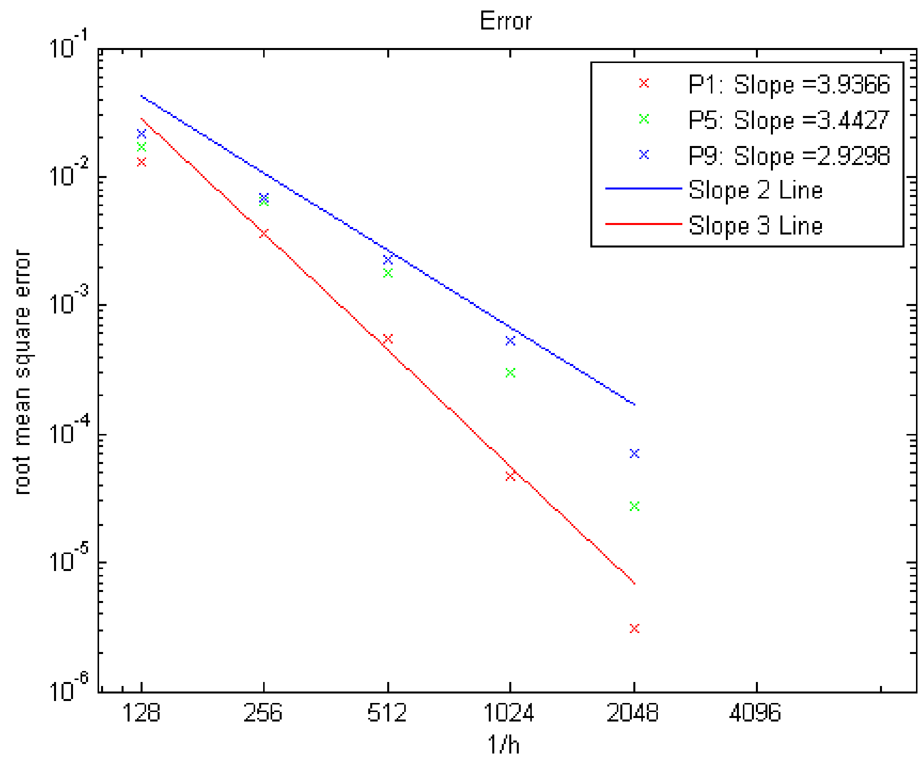


Figure 5-4 RMSE for first, fifth and ninth order polynomials and increasing numbers of second order elements.

5.3 Conclusions from Reed problem

The Reed problem gave us a benchmark to test our implemented second-order finite element method in one dimension. From the results it first became clear that the second-order elements were indeed correctly implemented when compared to the, known to be correct, solutions found with first-order elements. From the scalar flux in Figure 5-2 it is clear that the higher order angular polynomials contain larger discontinuities, which make it harder to accurately calculate the scalar flux in those regions. Because of the flux being less smooth in these regions, the expected accuracy is not reached when determining the slope of the RMSE.
The most important conclusion found from testing in the Reed problem was that the improvement in accuracy was indeed significant for second-order elements. The accuracy for only half as many elements was found to be much higher for second-order elements than for first-order elements in the Reed problem. Thus making the implementation of second-order elements useful in calculations that require higher accuracy.

6 Testing on absorbing and scattering cross-section

To further test the implemented second order elements we determine the neutron flux on a simple geometry with given homogeneous source and known cross-sections. This was done with both first -and second-order elements in order to compare results and determine the increase of accuracy, if any, when using second-order elements. We hope to conclude that the second-order elements are much more accurate and therefore help reduce computation time by using fewer elements to receive the same result. The flux in these cases was only calculated for first-order polynomials in the angle(P1).
First we'll discuss the results found in one dimension for a pure absorbing cross-section and then move on to the results in 2D for both absorbing and scattering cross-sections. In 1D we use both the linear 2-node line element and the quadratic 3-node line element, while in 2D we use 3 –and 6-node triangles and 4, 8 –and 9-node quadrilaterals.

6.1 1D Calculations

In the 1D case we consider a one dimensional slab of 1 cm and a slab of 10 cm both with the same homogeneous source and cross-section.
The mesh for this geometry was generated automatically in Phantom using same-sized equidistant elements.
Figure 6-1 shows a diagram of the geometry for the 1cm slab.

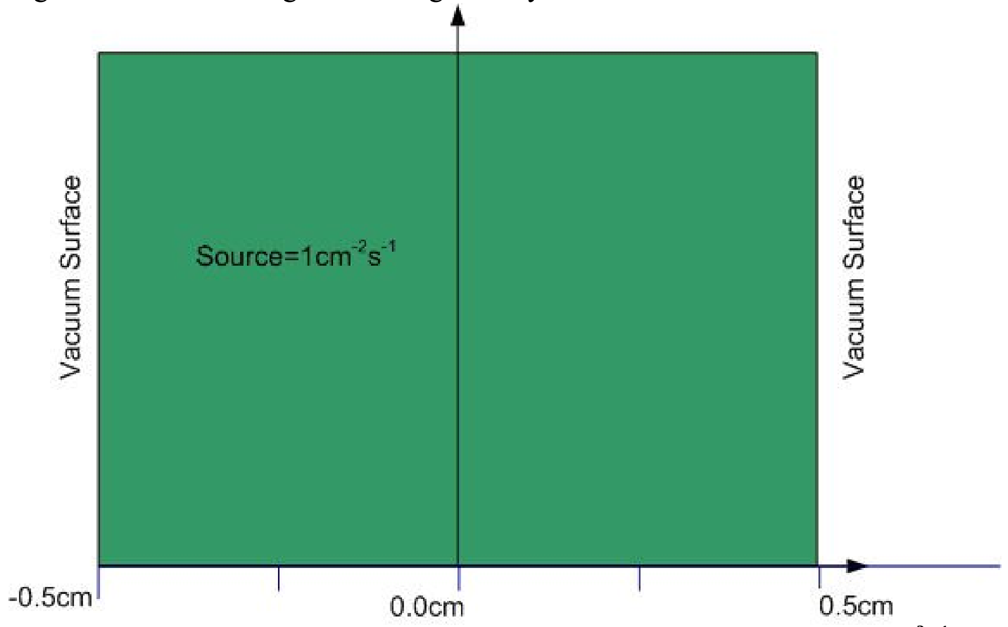


Figure 6-1Diagram of the one dimensional region with constant isotropic source 1cm^{-2}s^{-1} and vacuum boundaries on both sides.

The geometry of the 10cm slab is exactly the same, except for the boundaries being at -5cm and +5cm. The neutron flux was calculated several times for these cases, starting at the coarsest mesh using only 8 line elements and doubling the amount of elements until a more accurate result was reached with 4096 elements.

6.1.1 Pure absorbing cross-section on 1D slab geometry

The neutron flux will be given for a mesh of 4096 elements using the following cross-sections:

$$\sigma_s = 0cm^{-1}$$
$$\sigma_t = 1cm^{-1},$$

where σ_s is the scattering cross-section and σ_t is the total cross-section.
The flux for the 1cm slab geometry is shown below in Figure 6-2 and the flux for the 10cm slab geometry in Figure 6-3, both for second-order elements. We expect the neutron flux to be at its peak in the middle of the region, while slowly dropping to its lowest point on the boundaries to create a parabolic curve.

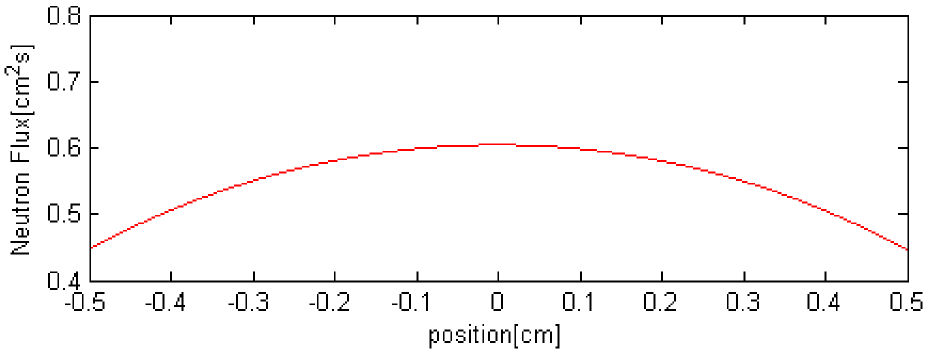


Figure 6-2Neutron Flux found with the second-order FEM using 4096 elements on the 1cm slab geometry.

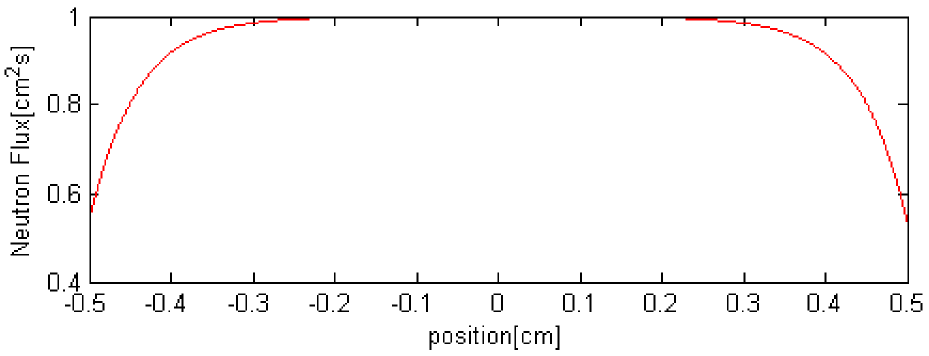


Figure 6-3 Neutron Flux found with the second-order FEM using 4096 elements on the 10cm slab geometry.

The results from Figure 6-2 confirm our expectations and show a parabolic decay from the centre. The result for first-order elements will not be explicitly plotted since both first-order and second-order finite elements reach nearly the same result for a mesh of 4096 elements.

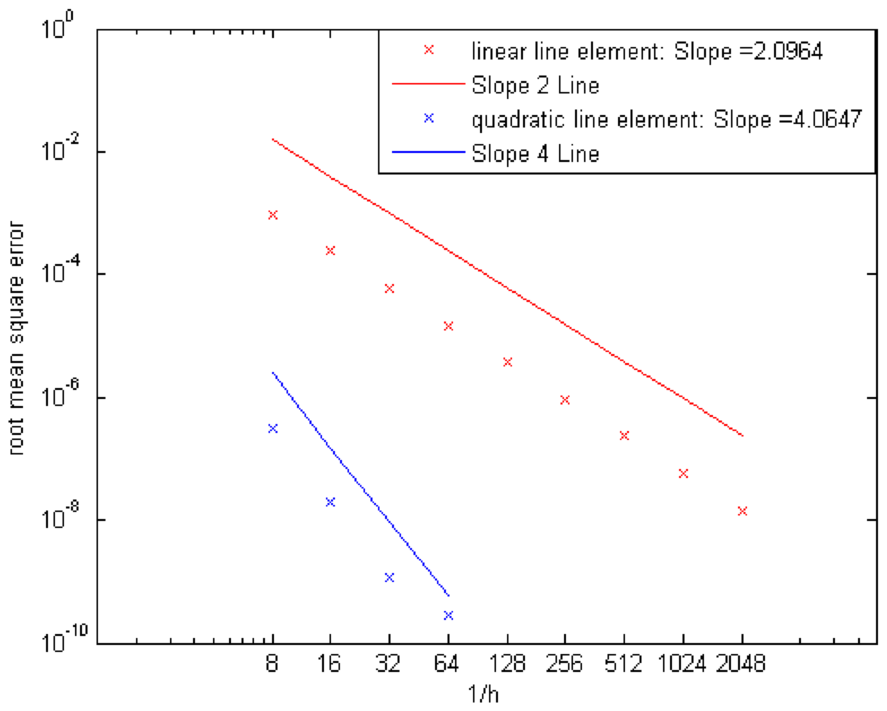


Figure 6-4Root mean square error on 1cm slab geometry for first(red) –and second-order(blue) elements of decreasing size h. Including two additional lines to illustrate the slope of the RMSE.

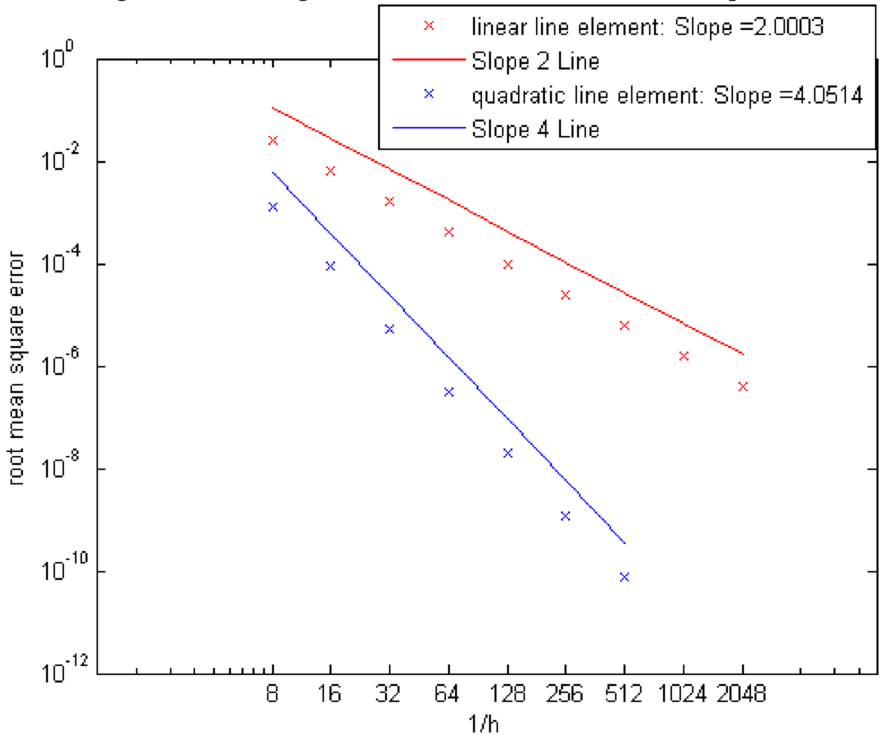


Figure 6-5Root mean square error on 10cm slab geometry for first(red) -and second-order(blue) elements of decreasing size h. Including two additional lines to illustrate the slope of the RMSE.

To calculate the accuracy of the solutions we will use the second order result with 4096 elements as the exact solution and calculate the RMSE using equation(4.2). In Figure 6-4 and Figure 6-5 the RMSE on the 1cm and 10cm slab has been plotted respectively. In both figures the first-order RMSE has 9 data points, while the second-order RMSE has been limited to a lowest error of 10^{-10}. This was done because we were unable to accurately determine the RMSE below this point.

The results from Figure 6-4 and Figure 6-5 show that the second-order elements are much more accurate than the first-order elements. In the 1cm geometry the second-order elements reach an error of $\sim 10^{-10}$ at 64 elements while the first-order elements never reach an error below $\sim 10^{-8}$ even when using 2048 elements.

6.2 2D Calculations

Testing the code for second-order elements in 1D show promising results. We now want to test the code for second-order elements in two dimensions to determine if these are also correctly implemented and of course to find out if they are more accurate than the first-order elements. The geometry in the 2D case is very similar to the 1D geometry. The geometry shown in Figure 6-1 is expanded in two dimensions, creating a 1cm by 1cm square with vacuum boundary conditions on each side and a constant source of $1\text{cm}^{-2}\text{s}^{-1}$. The same is done to create a 10cm by 10cm square region. The number of elements is again doubled from a coarse 2x2 grid until a much finer 64x64 element grid is reached. The elements used in 2D are the quadrilateral and triangular elements, as described in sections 2.2 and 2.3. The 3-node triangle and 4-node quadrilateral elements will be referred to as the first-order elements, while the 6-node triangle and the 8 –and 9-node quads are the second-order elements.

6.2.1 Pure absorbing cross-section on 2D square geometry

The neutron flux will be shown for a mesh of 64x64 elements using the same cross-sections used in section 5.1.1 for the 1D case. In Figure 6-6 we see the neutron flux for the 1cm^2 square geometry found using 9-node quadrilateral elements. The flux is highest in the center and declines towards the edges of the square. This is as expected since the mean free path and diameter are the same order of magnitude, which causes the boundaries of the region to have much more influence on the neutron flux. Figure 6-7 shows the neutron flux for the 100cm^2 square region using 6-node triangular elements. In this case the flux remains nearly constant along the whole geometry. This was also expected since in this case the mean free path is much smaller than the diameter of the geometry, which means that the boundaries have little influence on the neutron flux and thus maintaining a much more constant result.

Our other results using quadrilateral elements instead of triangular or vice-versa will not be shown since these are very similar to the other cases. The same is true for the 4 –and 8-node quadrilaterals and 3-node triangles.

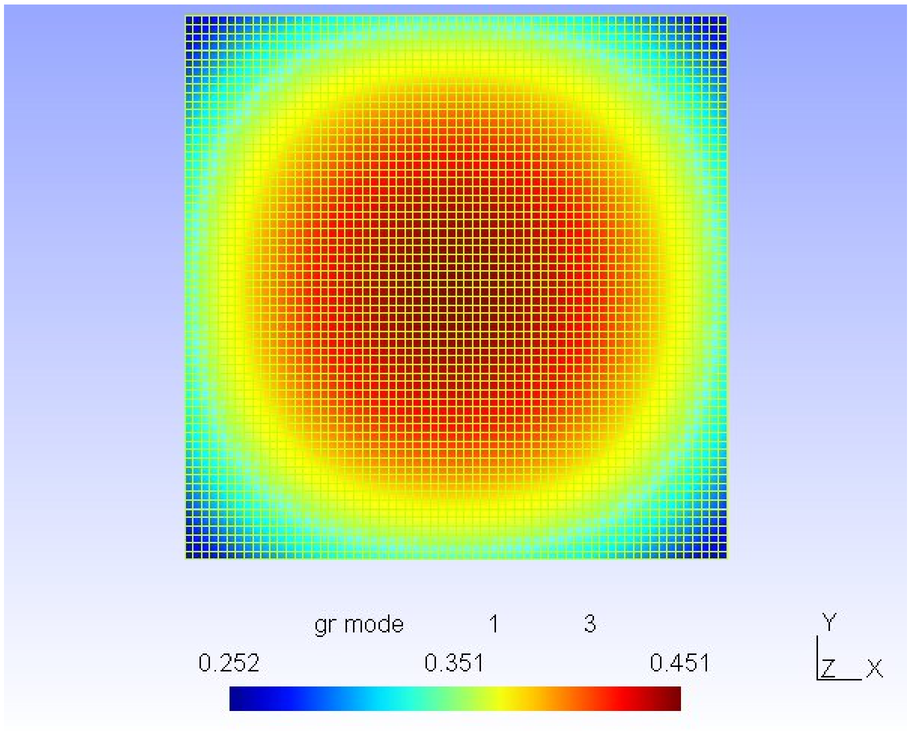


Figure 6-6 Neutron Flux using 4096 9-node quadrilateral elements on the 1cm^2 square geometry.

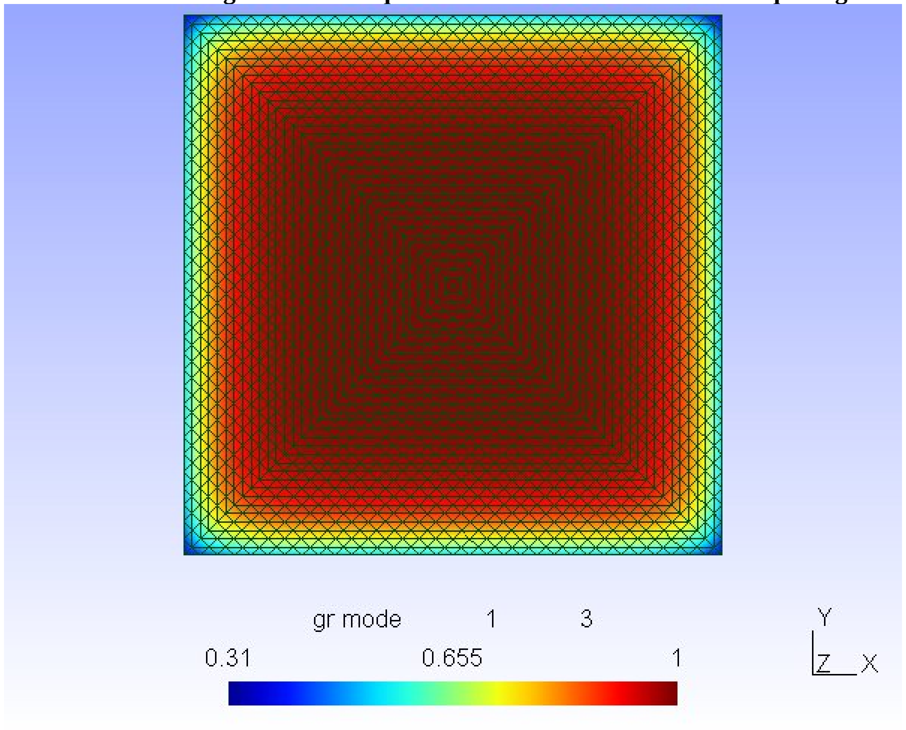


Figure 6-7 Neutron Flux using 4096 6-node triangular elements on the 100cm^2 square geometry.

Next we will move on to the error calculation in the 2D absorption case, where we again compare our most accurate solution of 4096 elements with the other solutions. We give both the RMSE for the triangular and the quadrilateral elements, for both the 1cm^2 and 100cm^2 square geometries. In Figure 6-8 we see the RMSE for the 4, 8 –and 9-nodes quadrilateral elements and for the 3 –and 6-node triangular element in Figure 6-9, both for the 1cm^2 square geometry. For the 100cm^2 square geometry we see the results in Figure 6-10 and Figure 6-11 for quadrilateral and triangular elements respectively.

Interesting to note is that the 8 –and 9-node quadrilateral elements are much more accurate than the 6-node triangular element. On the other hand the 4-node quad reaches a comparable accuracy to the 3-node triangle, even being a bit less accurate. The slope of the RMSE for the different elements is as expected however, reaching values of ~2 for the first-order elements and ~4 for the second-order elements. Oddly the second-order triangle elements are less accurate for the 100cm^2 case than for the 1cm^2 case, while the quads are nearly the same. Before adding more conclusions we will look at another 2D case which includes a scattering cross-section.

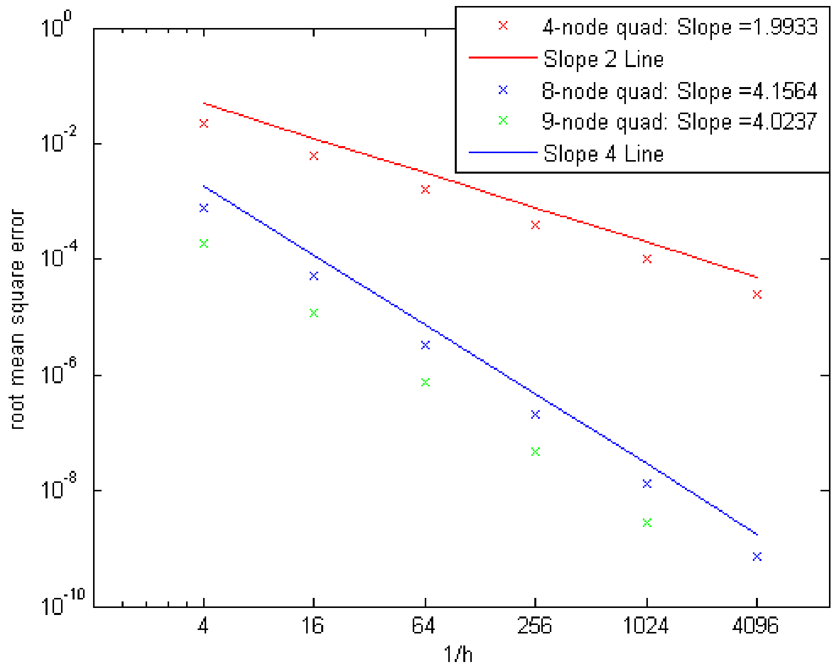


Figure 6-8Root mean square error on 1cm^2 square geometry for three types of quadrilateral elements and decreasing element size h. Including two additional lines to illustrate the slope of the RMSE.

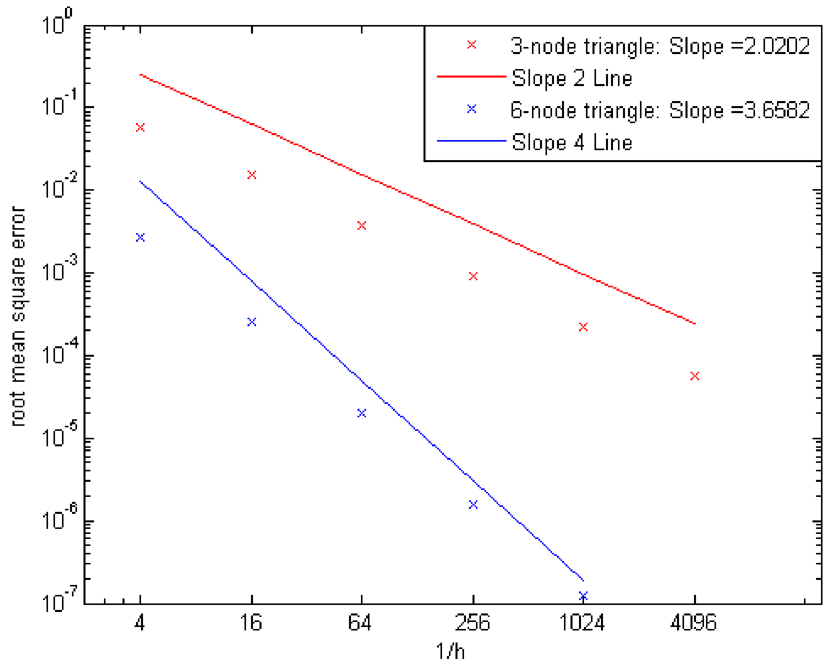


Figure 6-9Root mean square error on 1cm^2 square geometry for two types of triangular elements and decreasing element size h. Including two additional lines to illustrate the slope of the RMSE.

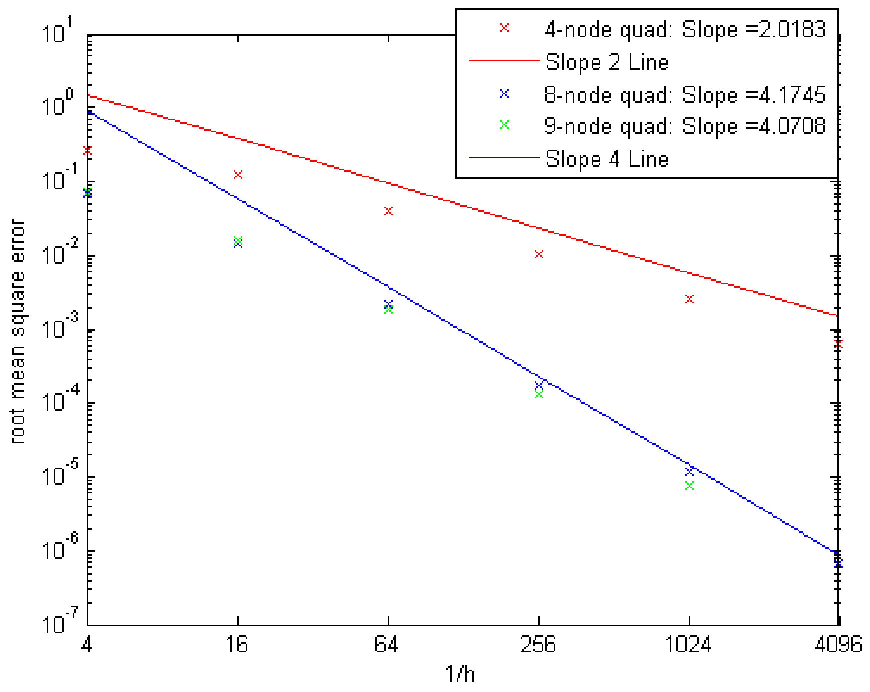


Figure 6-10Root mean square error on 100cm^2 square geometry for three types of quadrilateral elements and decreasing element size h. Including two additional lines to illustrate the slope of the RMSE.

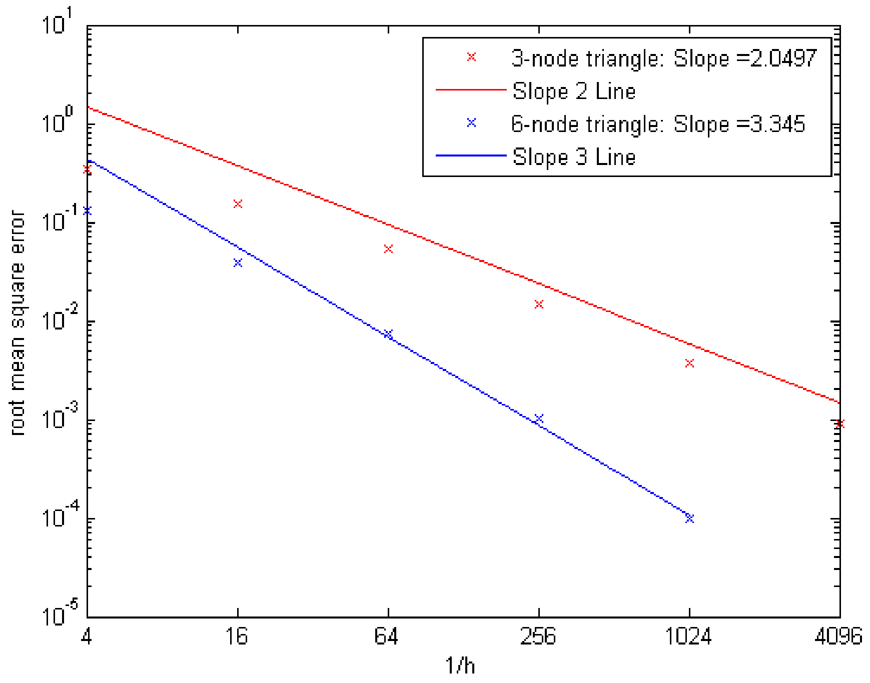


Figure 6-11Root mean square error on 100cm^2 square geometry for two types of triangular elements and decreasing element size h. Including two additional lines to illustrate the slope of the RMSE.

6.2.2 Scattering cross-section on 2D square geometry

We want to test our implemented second-order elements on different types of cross-section, so for this case we will look at a geometry containing a scattering cross-section. We use the same constant source and square 1cm^2 geometry as in section 6.2.1 but with the following cross-section:

$$\begin{aligned} \sigma_s &= 9.99cm^{-1} \\ \sigma_t &= 1cm^{-1} \end{aligned},$$

making this an almost purely scattering case with only a very small absorbing cross-section. We'll only treat the 1cm^2 case this time, because testing our elements on a scattering cross-section is the main goal.
The neutron flux has been plotted below in Figure 6-12 using 4096 9-node quadrilateral elements. Comparing this to Figure 6-6 we note that shape of the flux is less smooth at the edges of each colored region. The flux still drops of from its highest value in the middle to the lowest value at the vacuum boundaries as expected from the mean free path being in the same order of magnitude as the diameter of the geometry.
The neutron flux was also calculated using triangular elements, but again is very similar and will be omitted.

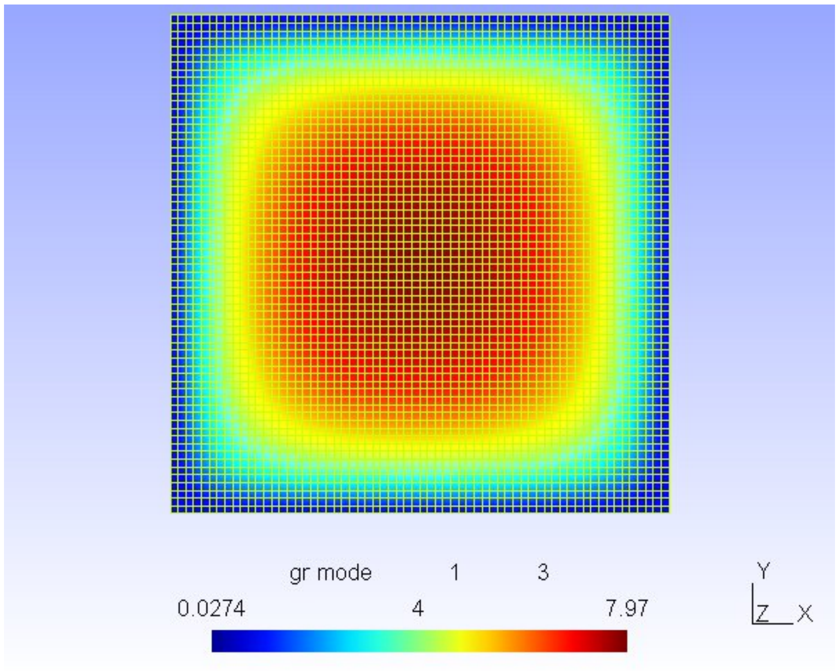


Figure 6-12 Neutron Flux using 4096 9-node quadrilateral elements on the 1cm^2 square geometry with scattering cross-section.

In Figure 6-15 and Figure 6-16 the RMSE error has been plotted. Immediately we notice that the slope of the second-order elements is much lower than in the previous case, this time being around 3 instead of 4. The scattering cross-section is harder to accurately calculate, resulting in lower slopes than in the absorbing case. The main reason for this is the smoothness of the neutron flux in Figure 6-13, the flux in the area between two differently colored regions is less smooth than in Figure 6-14. This means that there is a big difference in the flux between two, spatially close, points.

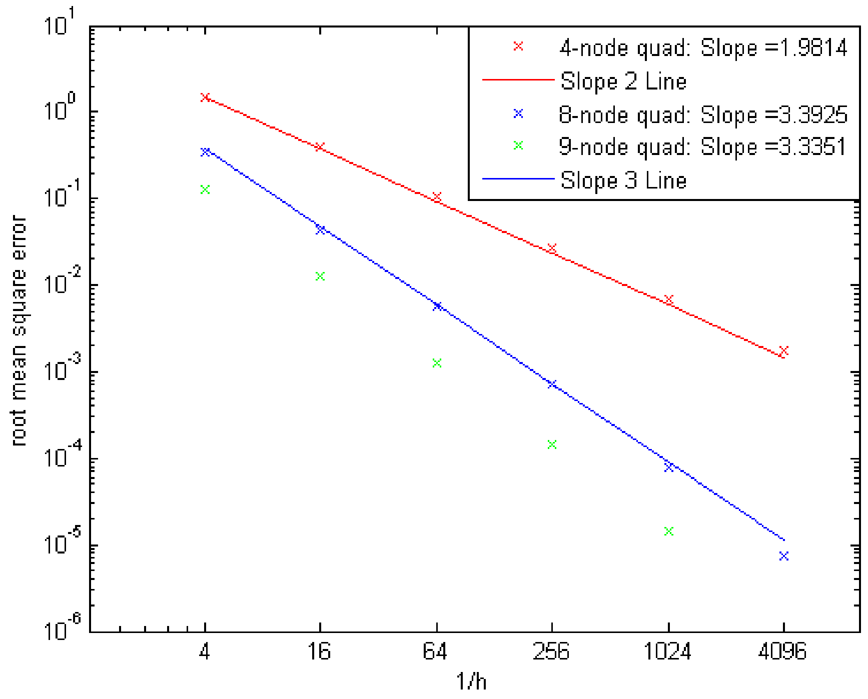


Figure 6-15Root mean square error on 1cm^2 square geometry for three types of quadrilateral elements and decreasing element size h. Including two additional lines to illustrate the slope of the RMSE.

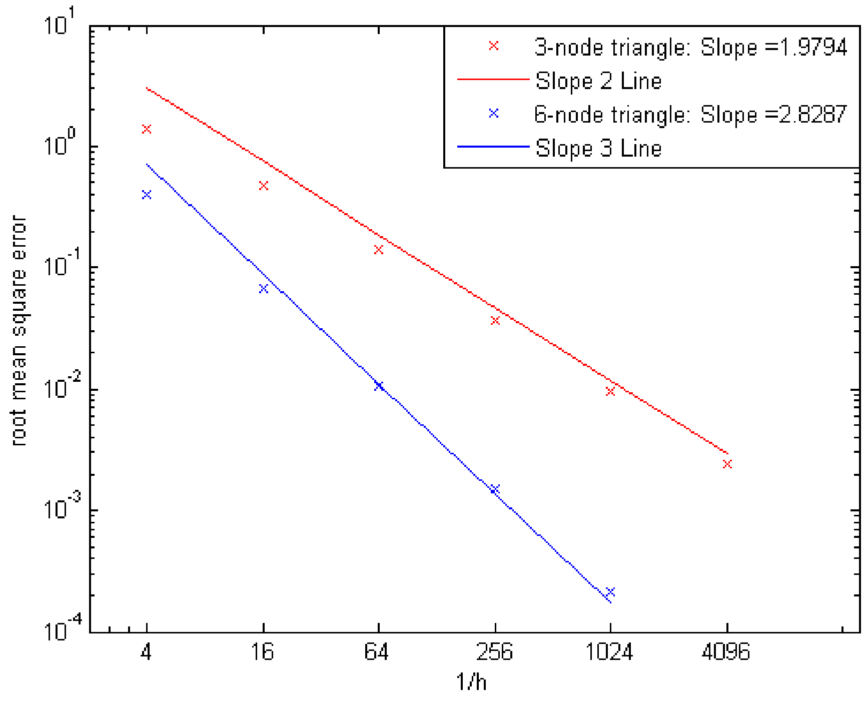


Figure 6-16Root mean square error on 1cm^2 square geometry for two types of triangular elements and decreasing element size h. Including two additional lines to illustrate the slope of the RMSE.

6.3 Conclusions from testing on different cross-sections

First of all the second-order elements have been shown to be correctly implemented and reach the same results for the neutron flux as their first-order counterparts. Knowing this, the main interest is to determine the increase in accuracy when using second-order elements. From our results it is clear that the second-order elements are far superior in terms of accuracy compared to the first-order elements. Both in the 1D and 2D case the second-order elements reach the same accuracy at only a fraction of the elements necessary when using first-order elements. The different types of elements used in 2D have shown that the 9-node quad is the most accurate second-order element, providing better results than the triangular or 8-node quadrilateral element in each of our cases.

7 Conclusion

In this report we tested the accuracy of second-order elements when solving the Boltzmann transport equation. We hoped to establish that the quadratic elements were more accurate, and by using several test cases we were able to determine that this was indeed the case.

Our first test cases using Matlab showed that the quadratic finite element was accurate, but since these cases were very simple compared to the actual Boltzmann equation we could not conclude if this was true for actual neutron transport problems.
The Reed problem showed that the quadratic elements were indeed much more accurate than second order, increasing the decrease in order from order 2 to almost order 4. Some interesting results were seen when using different orders of polynomials to solve for the angular part of the Boltzmann equation. It became clear that a change in angular distribution and the way this is handled by the solver can have a large effect on the results.
While the Matlab and Reed cases were all done in one dimension, we also tested several square geometries. Here we tested both several different types of elements and different cross-sections. From these two dimensional cases we conclude that again the quadratic elements are much more accurate, this being especially true for the quadrilateral 9-node element which was shown to be the most accurate.

With these results we see a definitive increase in accuracy when using the quadratic elements. However there are still more improvements to be made. As a recommendation we would like to explore the possibility to decrease the computation time needed. One way to do this is by starting with a very rough mesh and determining from there what areas are most difficult to calculate. It would then be possible to increase the number of elements only in the areas that are harder to compute and thus save time and money on wasting computation time on less interesting regions.

References

[1] James J. Duderstadt, Louis J. Hamilton, “Nuclear Reactor Analysis”, 1976, Wiley.
[2]D. Lathouwers, “Neutron Transport Discretization using a Least Squares Approach”, 2007, Delft University of Technology.
[3] Images from: http://en.wikipedia.org/wiki/Finite_element_method
[4]D.V. Griffiths and I.M. Smith, “Numerical Methods for Engineering” Second Edition, 2006, Chapman & Hall/CRC.
[5]Basis functions and elements from http://solidmechanics.org/text/Chapter8_1/Chapter8_1.htm#Sect8_1_1, Theory and Implementation of the Finite Element Method
[6] Theory and Practice of Finite Elements : Alexandre Ern, Jean-Luc Guermond, 2004, Springer, Page360.
[7]J. van Kan, A. Segal and F. Vermolen,”Numerical Methods in Scientific Computing”, 2005, VSSD.
[8] A.G. Buchan, C.C. Pain, M.D. Eaton, R.P. Smedley-Stevenson, A.J.H. Goddard, “Linear and quadratic octahedral wavelets on the sphere for angular discretisations of the Boltzmann transport equation”, Computational Physics and Geophysics Group, Department of Earth Science and Engineering, Imperial College of Science, Technology and Medicine, University of London, 29 April 2005, page1242.