Including general scattering in high-fidelity deterministic particle transport calculations for proton therapy applications

L.F. de Niet
Guided by Dr. D. Lathouwers

February 2, 2016

Abstract

Normal proton scatter simulations rely on the Fokker-Planck approximation. In this BEP, we developed a new way to work around this restriction. In order to do so, the formula are adapted to a new form and implemented in the program.

The results of this BEP are mixed. When the imposed solution and scatter function are not very peaked, the simulations produce the predicted results. The order of convergence of these results is 0.89 with function set 1 and around 1.8 with function sets 2 and 3. In order to reduce the runtime of the simulations, some approximations are used. The constant σ_s approximation shows the most potential of those approximations.

When the scatter function and the imposed solutions become too peaked, some strange behavior occurs. The result of the simulation does not look like the imposed solution in those cases. The reason behind this behavior is yet unknown and can be the subject of further research.

Contents

1 Background/Introduction **2**

2 Forward peaked proton scattering **3**
2.1 The Boltzmann equation 3
2.2 Numerical discretization by the discontinuous Galerkin method . 4
2.3 Scatter in deterministic particle transport 5
2.3.1 σ_s is zero 5
2.3.2 σ_s is a non-zero constant 6
2.3.3 General scatter functions 6
2.4 Numerical quadrature 7
2.5 Different scatter kernels 7
2.6 Testing with manufactured solutions 8

3 Simulation results **10**
3.1 A smooth solution 10
3.2 A more peaked solution 10

4 Reducing the run time of the simulation **13**
4.1 Reduce the number of coupling patches 13
4.2 Use the average σ_s of each patch 13
4.3 Local mesh refinement 14

5 Conclusions **21**

6 Further research **22**
6.1 Smarter choice of the direction of the imposed solution 22
6.2 Expanding the scatter function in Legendre polynomials 22

Chapter 1

Background/Introduction

Each year, millions of people are diagnosed with cancer. Four out of ten receive radiotherapy as primary treatment. The accurate prediction of the dose distribution throughout the body is vital for obtaining an optimal treatment plan where the dose to the tumor is maximized while sparing healthy surrounding tissue as much as possible. The accuracy that can be achieved is limited on the one hand by the dose calculation methods used in the clinic and by uncertainties in patient anatomy, patient positioning and nuclear data on the other hand.

Traditional those calculations are based on approximate analytical methods which are highly efficient but suffer from inaccuracies in heterogeneous situations (e.g. lung). In such cases one may revert to Monte Carlo methods that are as accurate as the nuclear data but these have found relatively little clinical use due to computational restrictions. There is therefore a clear clinical need for an alternative approach that preserves this accuracy. This project aims at developing a dose engine that can deliver Monte Carlo accuracy at a fraction of the cost.

By basing a dose engine on the so-called deterministic transport approach this goal is now within reach. As the underlying description is the same as in Monte Carlo, the accuracy that can be achieved is similar but with the added values of no statistical error, high execution speed, allowing for systematic sensitivity analysis.

In previous work the differential cross section is expanded into Legendre polynomials. But the differential cross section is a very peaked function, nearly singular. So it would take an enormous number of polynomials to accurately describe the differential cross section. To overcome this issue, the Fokker-Planck approximation is used to transform the linear Boltzmann equation into the Fokker-Planck equation, with the Fokker-Planck operators describing the small angle Coulomb scatter interactions with the atomic nuclei and the small energy transfers to the atomic electrons.

The goal of this work is to remove this physics restriction and use the Henyey-Greenstein function to model the scattering of the protons instead of the Fokker-Planck approximation.

Chapter 2

Forward peaked proton scattering

2.1 The Boltzmann equation

In particle physics, the Boltzmann equation describes the distribution of particles in a volume in terms of energy, direction, time and space. In its most general form the Boltzmann equation reads.

$$\begin{aligned} &\frac{\partial n}{\partial t} + v\hat{\Omega}\cdot\nabla n + v\sigma_t n(r,E,\hat{\Omega},t) \\ -\int_0^\infty dE' \int_{4\pi} d\hat{\Omega}'\sigma_s(E'\to E,\hat{\Omega}'\to\hat{\Omega})n(r,E',\hat{\Omega}',t) &- S(r,E,\hat{\Omega},t) = 0 \end{aligned} \tag{2.1}$$

In this equation n is the number of particles in a location with a certain energy and direction at a given moment. v is the speed of the particles, σ_t is the macroscopic total cross section, σ_s is the scattering cross section which characterizes the scattering of a particle from E' and Ω' to E and Ω and S is the source term. Also, $v\hat{\Omega}\cdot\nabla n$ is the drifting term and $v\sigma_t n$ is the loss through collisions.

In this paper a simplified form of the equation is used, because we only focus on the description of angular scattering. This work further limits itself to a steady-state solution neglecting spatial and energy dependance. This work focuses on the flux, which is the angular density multiplied by the speed ($\varphi = vn$). When all this is taken into account, equation 2.1 becomes more compact. This may be done, because this work only focuses on the anglular flux and nothing more.

This means the main equation of this work is the following:

$$\sigma_t\varphi(\Omega) = \int_{4\pi} \sigma_s(\mu_0)\varphi(\Omega')d\Omega' + S(\Omega) \tag{2.2}$$

2.2 Numerical discretization by the discontinuous Galerkin method

Since this work only focuses on one spatial element and neglects energy dependance, the solution can plotted on a single sphere. Therefor, the program uses a mesh of a unit sphere. In its basic configuration this sphere is refined in eight triangular patches. Hereto, each patch is an octant of the unit sphere. To increase the accuracy of the solution, the mesh can be refined. Each patch can be divided into four new patches. This refining can be local or global. Global means that every patch on the sphere gets divided and with local refinement the user can decide witch patches should be divided. An example of a unit sphere with one locally refined patch can be seen in figure 2.1.

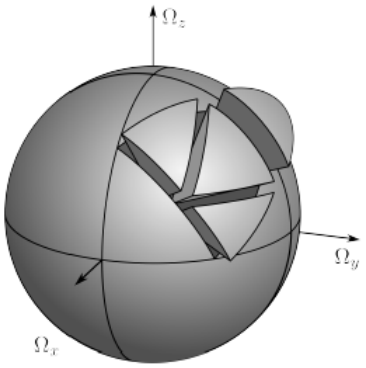

Figure 2.1: A graphical representation of the angular mesh and the refinement of one octant.

In order to solve the problem, the solution is approached with the discontinuous Galerkin method. We assume the approximation of the solution of the equation is a linear combination of all those basis functions.

$$\varphi(\Omega) = \sum_p \sum_i \phi_{p,i} h_p^i(\Omega) \tag{2.3}$$

Where $h_p^i(\Omega)$ is the i^{th} basis function on patch p and $\phi_{p,i}$ is the coefficient of that basis function.

In order to determine the coefficients for each basis function on each patch, the model equation is multiplied with the test function and integrated over the domain. This results in a system of linear equations.

In this report, three sets of basis functions are used. Function set 1 is a constant function, function set 2 a set of three linear functions on each patch and function set 3 a set of four functions linear in Ω.

To express the accuracy of the approximation, h is introduced as a measure for the size of the mesh. In this project h is expressed as the reciprocal of the

root of the product of the number of patches and the number of test functions. We expect that the approximation gets better with decreasing h.

$$h = \frac{1}{\sqrt{N_{patches} N_{testfunctions}}} \tag{2.4}$$

2.3 Scatter in deterministic particle transport

In this section of the report, the formula for the code are derived step by step. From a simple version of the main equation to the final implemented version of the formula.

2.3.1 σ_s is zero

First we will look at the simple case where σ_s is zero. This reduces equation 2.2 to:

$$\sigma_t \varphi(\Omega) = S(\Omega) \tag{2.5}$$

Now the Galerkin method is applied. To do so, both sides of the equation are multiplied with $h_{p'}^{i'}(\Omega)$ and integrated over the sphere. Next the angular flux is substituted with the model equation.

$$\sigma_t \int_{p'} \varphi(\Omega) h_{p'}^{i'}(\Omega) d\Omega = \int_{p'} S(\Omega) h_{p'}^{i'}(\Omega) d\Omega$$

$$\sigma_t \int_{p'} \sum_{p} \sum_{i} \phi_{p,i} h_{p}^{i}(\Omega) h_{p'}^{i'}(\Omega) d\Omega = \int_{p'} S(\Omega) h_{p'}^{i'}(\Omega) d\Omega$$

$$\sigma_t \sum_{i} \phi_{p',i} \int_{p'} h_{p'}^{i}(\Omega) h_{p'}^{i'}(\Omega) d\Omega = \int_{p'} S(\Omega) h_{p'}^{i'}(\Omega) d\Omega$$

In the last step we use the fact that the basis functions of different patches do not overlap. This last equation is actually a system of equations, because they hold for all i' and p'. If we take P for the number of patches and I for the number of functions, the matrix can be defined. Since S is known, the right side of the equations can be evaluated. The resulting system is:

$$\sigma_t \begin{pmatrix} \int_{p_1} h_1^1 h_1^1 & \cdots & \int_{p_1} h_P^I h_1^1 \\ \vdots & \ddots & \vdots \\ \int_{p_P} h_1^1 h_P^I & \cdots & \int_{p_P} h_P^I h_P^I \end{pmatrix} \begin{pmatrix} \phi_1^1 \\ \vdots \\ \phi_P^I \end{pmatrix} = \begin{pmatrix} \int_{p_1} S h_1^1 \\ \vdots \\ \int_{p_P} S h_P^I \end{pmatrix} \tag{2.6}$$

2.3.2 σ_s is a non-zero constant

When we choose σ_s as a non zero constant the main equation preserves an extra term.

$$\sigma_t\varphi(\Omega) = \frac{\sigma_s}{4\pi}\int\limits_{4\pi}\varphi(\Omega')d\Omega' + S(\Omega)$$

The removal and source terms stay the same, so we only look at the scatter term.

$$\sigma_s\int\limits_{p'}\int\limits_{4\pi}\varphi(\Omega')d\Omega' h^{i'}_{p'}d\Omega$$

$$\sigma_s\sum_p\sum_i\varphi_{p,i}\int\limits_{4\pi}h^i_p(\Omega')d\Omega'\int\limits_{p'}h^{i'}_{p'}d\Omega$$

$$\sigma_s\begin{pmatrix} \int\limits_{p_1}h^1_1\int\limits_{p_1}h^1_1 & \int\limits_{p_1}h^2_1\int\limits_{p_1}h^1_1 & \cdots & \int\limits_{p_P}h^I_P\int\limits_{p_1}h^1_1 \\ \int\limits_{p_1}h^1_1\int\limits_{p_1}h^2_1 & \ddots & & \int\limits_{p_P}h^I_P\int\limits_{p_1}h^2_1 \\ \vdots & & \ddots & \vdots \\ \int\limits_{p_1}h^1_1\int\limits_{p_P}h^I_P & \int\limits_{p_1}h^2_1\int\limits_{p_P}h^I_P & \cdots & \int\limits_{p_P}h^I_P\int\limits_{p_P}h^I_P \end{pmatrix}\begin{pmatrix}\varphi^1_1\\ \varphi^2_1\\ \vdots\\ \phi^I_P\end{pmatrix}$$

This matrix expression does not contain any zeros. Combining all the terms results in the following expression:

$$\begin{aligned}\sigma_t\begin{pmatrix} \int\limits_{p_1}h^1_1h^1_1 & \cdots & \int\limits_{p_1}h^I_Ph^1_1 \\ \vdots & \ddots & \vdots \\ \int\limits_{p_P}h^1_1h^I_P & \cdots & \int\limits_{p_P}h^I_Ph^I_P \end{pmatrix}\begin{pmatrix}\phi^1_1\\ \vdots\\ \phi^I_P\end{pmatrix} = \\ \sigma_s\begin{pmatrix} \int\limits_{p_1}h^1_1\int\limits_{p_1}h^1_1 & \cdots & \int\limits_{p_P}h^I_P\int\limits_{p_1}h^1_1 \\ \vdots & \ddots & \vdots \\ \int\limits_{p_1}h^1_1\int\limits_{p_P}h^I_P & \cdots & \int\limits_{p_P}h^I_P\int\limits_{p_P}h^I_P \end{pmatrix}\begin{pmatrix}\phi^1_1\\ \vdots\\ \phi^I_P\end{pmatrix} + \begin{pmatrix}\int\limits_{p_1}Sh^1_1\\ \vdots\\ \int\limits_{p_P}Sh^I_P\end{pmatrix}\end{aligned} \tag{2.7}$$

2.3.3 General scatter functions

Now we look at the realistic case where σ_s is μ_0 dependent. μ_0 is the dot product of Ω and Ω'. This is because of the symmetry between the two Ω's. And since both Ω and Ω' are unit vectors, the dot product of those two reduce to the cosine of the angle between both vectors. In this case the integral cant be evaluated that easily. If we look at the scatter term, we see the following:

$$\int_{4\pi} \sigma_s(\Omega \cdot \Omega')\varphi(\Omega')d\Omega'$$

$$\int_{4\pi} \sigma_s(\Omega \cdot \Omega') \sum_p \sum_i \varphi_{p,i} h_p^i(\Omega')d\Omega'$$

Next, the Galerkin method is applied.

$$\int_{p'} \int_{4\pi} \sigma_s(\Omega \cdot \Omega') \sum_p \sum_i \varphi_{p,i} h_p^i(\Omega') h_{p'}^i(\Omega)d\Omega' d\Omega$$

$$\sum_p \sum_i \varphi_{p,i} \int_{p'} \int_p \sigma_s(\Omega \cdot \Omega') h_p^i(\Omega') h_{p'}^i(\Omega)d\Omega' d\Omega$$

This is a very expensive function to evaluate. The integral can't be solved analytical and to do so numerical, a product quadrature is needed.

2.4 Numerical quadrature

Some integrals can't be evaluated analytical. Those integrals are solved with numerical quadrature. This is done by taking a number of points on patch p and p' and evaluate the value of the integral at those points. The value of this integral becomes the sum of these values, but with certain weights. In formula form:

$$\int f(x)dx = \sum_n w_n f(x_n) \tag{2.8}$$

All these points and weights have been tabulated for a single patch. But in this work, some of the the integrals have a domain of two patches. To evaluate these integrals over two patches, product quadrature is used.

$$\int \int f(x,y)dxdy = \sum_n \sum_m w_n w_m f(x_n, y_m)$$

In the implementation of the code, a specially developed Gaussian quadrature is used[6].

2.5 Different scatter kernels

In this project we use two functions to determine the value of $\sigma_s(\mu_0)$. The fist one is a constant. The next formula is used in that case.

$$\sigma_s(\mu_0) = \frac{\sigma_s}{4\pi} \tag{2.9}$$

Here μ_0 is between -1 and 1. The factor of 4π makes sure the integral over the sphere of the scatter kernel equals σ_s.

This is easy for calculations, but not physically accurate. Protons don't scatter isotropic but tend to divert very little from their path. The scattering of a proton in matter is dominated by the elastic scatter with atomic nuclei. An approximation of this behavior is the Henyey-Greenstein function.

$$\sigma_s(\mu_0, g) = \frac{\sigma_s}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\mu_0)^{\frac{3}{2}}} \tag{2.10}$$

This formula has some interesting properties. The integral over the sphere is independent of α and the center of mass of the graph between -1 and 1 is equal to g. If g equals zero, 2.10 reduces to 2.9. To give some insight is the Henyey-Greenstein formula, the formula is plotted in figure 2.2 with different values of g. In this graph the value of σ_s is 1. A typical g for proton scattering is 0.9999.

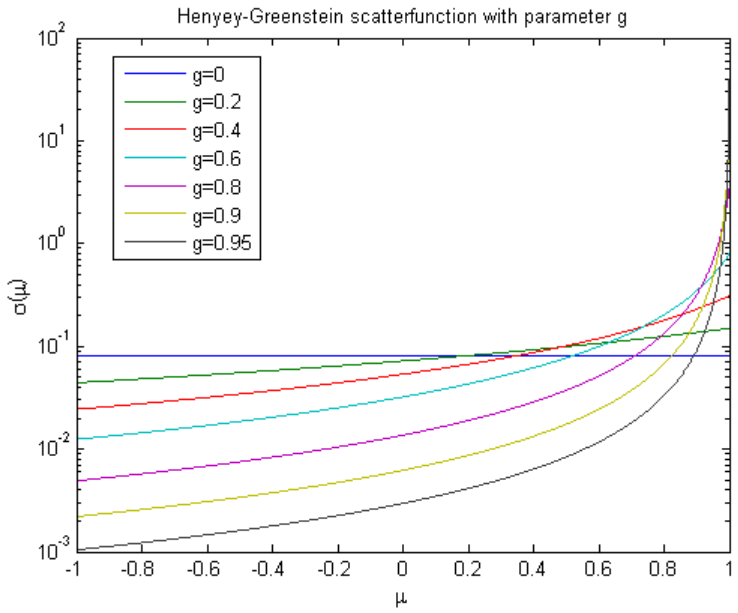

Figure 2.2: The Henyey-Greenstein scatter function for various g parameters as a function of the scatter cosine μ_0

2.6 Testing with manufactured solutions

To check if the code works, it is necessary to check if the results are as one would expect. To do so, the source is calculated from the desired result. This

is done with the following formula:

$$S(\Omega) = \sigma_t \varphi(\Omega) - \int_{4\pi} \sigma_s(\mu_0)\varphi(\Omega')d\Omega' \tag{2.11}$$

On the right hand side of the equation a quadrature is used for each point on the sphere. This means this is a very costly way to calculate the source. In normal research this isn't necessary, because the source is given or imposed. Applying this quadrature leads to the following formula, where the sum is over all the quadrature points on the sphere.

$$S(\Omega) = \sigma_t \varphi(\Omega) - \sum_p w_p \sigma_s(\Omega \cdot \Omega_p)\varphi(\Omega_p)$$

In this paper a few different solutions are used; a constant solution, a smooth solution and a more peaked solution.

$$\varphi(\Omega) = 1 \tag{2.12}$$

$$\varphi(\Omega) = e^{-(\Omega_z - 1)^2} \tag{2.13}$$

$$\varphi(\Omega) = e^{-4(\Omega_z - 1)^2} \tag{2.14}$$

The constant solution is used to check if the code works in the most general case. The cases we are interested in are the peaked solutions. In proton therapy a highly forward peaked solution is desired to minimize collateral damage in the human body. Equation 2.14 is not a highly forward peaked solution, but a more peaked solution than 2.13. For real proton therapy a pencilbeam solution is desired, but this is not covered in this work.

Equation 2.11 is also used to check the conservation of particles. To do so, the integral of both sides of the equation is taken over the whole sphere. This results in a simple equation, where on the left side the calculated source is integrated over the whole sphere and on the right side the integral over the imposed solution is multiplied with the absorption cross section.

$$\int_{4\pi} S(\Omega)d\Omega = (\sigma_t - \sigma_s) \int_{4\pi} \varphi(\Omega)d\Omega \tag{2.15}$$

In order to check the approximation, the error of the program is calculated. Hereto, the imposed solution is compared with the calculated solution. In this work we take the root of the squared error.

$$\epsilon = \sqrt{\int_{4\pi} (\varphi_{calculated}(\Omega) - \varphi_{imposed}(\Omega))^2} \tag{2.16}$$

Chapter 3

Simulation results

Using the equations from the previous chapter, we run the simulation to test the results. In all cases the mesh is uniformly refined and the scatter function is the Henyey-Greenstein function with parameter 0.7.

When a constant solution is used to generate the source for the program, the result is always within machine precision. Therefor a plot is considered redundant.

To quantify the convergence of the solution, the error is plotted versus h on a log-log scale. The order of convergence is the exponent of the fit with model equation 3.1.

$$\epsilon = ah^b \tag{3.1}$$

3.1 A smooth solution

This following graph shows the squared error versus the h in the case of the smooth solution 2.13. As can be seen, the error goes to zero when h decreases. Also, the various sets of basis functions show the behavior we expect. The first function set is the least accurate and the third function set are the most accurate. The order of convergence of the first function set is $0,89$ and the order of convergence of the second and third function set are both close to 1.8.

3.2 A more peaked solution

After this smooth solution, the more peaked solution 2.14 was also put through the program. As can be seen, the graph shows the same trend as with the smooth solution. The slopes of the fits through the data points are 0.94 for function set 1 and around 1.7 for the second and third function set.

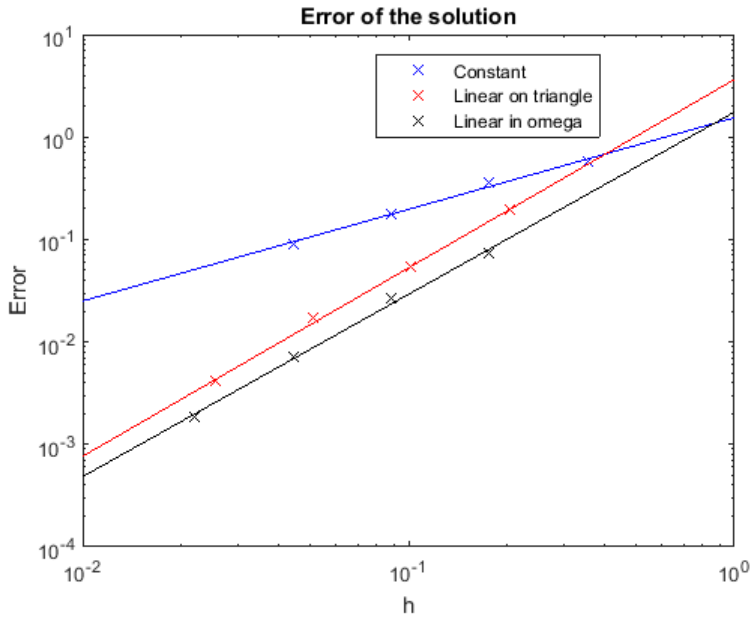

Figure 3.1: The error, calculated as the root of the squared difference between the imposed solution and the calculated solutions, of the program with a smooth imposed solution.

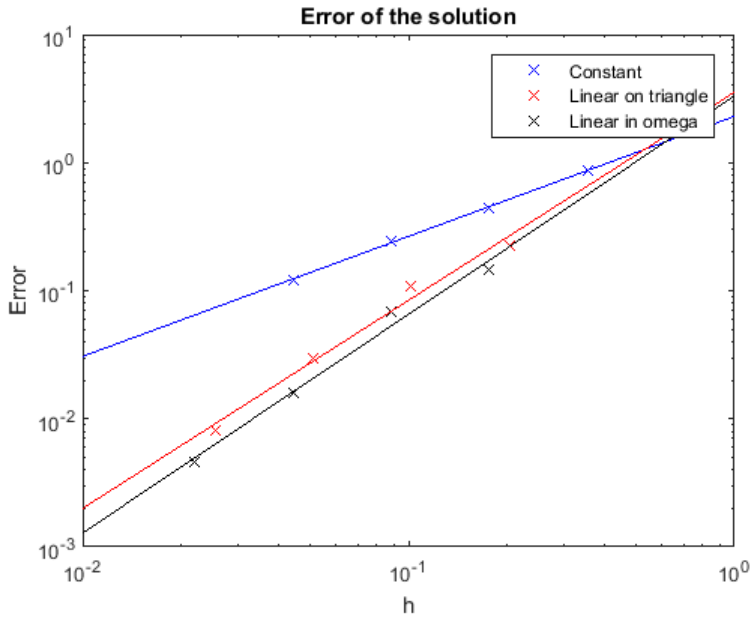

Figure 3.2: The error, calculated as the root of the squared difference between the imposed solution and the calculated solutions, of the program with a peaked imposed solution.

Chapter 4

Reducing the run time of the simulation

This part of the project focuses on the reduction of the run time of this program. The number of matrix elements that have to be determined is the number of patches and basis functions squared. So if the mesh is one step further globally refined, there are four times as much patches, so the matrix is sixteen times bigger. The scatter term is the most expensive part of the code, so most approximations will focus on that term.

4.1 Reduce the number of coupling patches

The Henyey-Greenstein function is very forward peaked. So the coupling between patches that lie far away from each other is very small. So small, the coupling can be neglected. If the mean μ of two patches is small, that patch pair will not be included in the scattering, unless the patches are neighbors. This may only be done if the scatter function is very peaked. This approximation introduces an error that can't be removed with further refining of the mesh.

In this approximation we take a threshold of 0.7. As can be seen in figure 4.1, the error doesn't follow the linear convergence seen in the previous chapter. The error flattens with decreasing h, because not all patches are included in the calculation. Of course, when the solution and the scattering are both very peaked, this error will be much lower.

4.2 Use the average σ_s of each patch

Another approximation is taking σ_s constant between two patches. The value of this σ_s is determined by the dot product of the mean omega's of the two patches. In that case the σ_s can be taken out of the integral and the integral

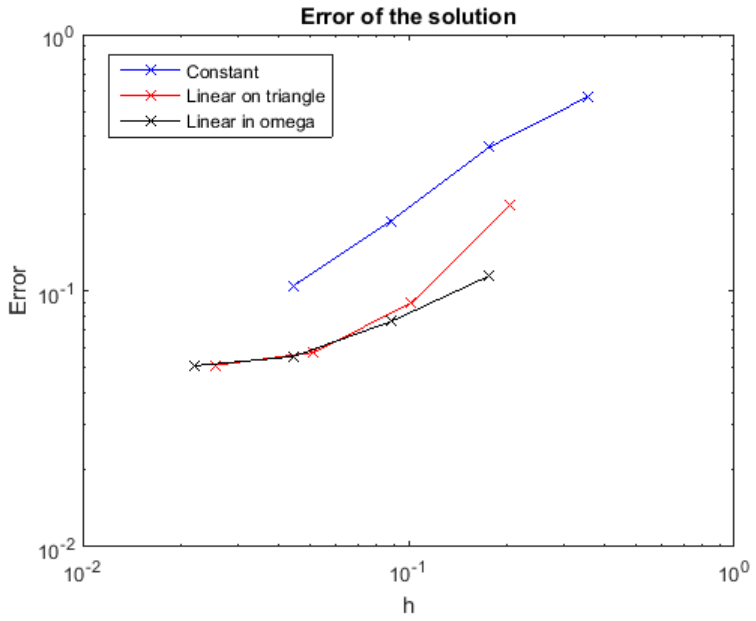

Figure 4.1: A graph showing the error, calculated as the root of the squared difference between the imposed solution and the calculated solutions, on the smooth solution with a μ_0 threshold of 0.7. The scatter is the Henyey-Greenstein function with parameter 0.7.

becomes a product of two easy to determine integrals as can be seen in section 2.3.2.

The results of this approximation is shown in figure 4.2. The figure shows that the slope of the fit is similar, but the offset is bigger. This approximation reduces the run time of the simulation with a factor 10000. So with a slightly decreased h, this approximation produces the same error in much less time. But decreasing h leads to an increase in required memory, so there is a tradeoff between speed and accuracy.

The constant σ_s approximation can also be combined with the μ_0 threshold. The results of those simulations can be seen in figure 4.3. For readability of the plot, only function set 1 has a fit. Again the error flattens at $5,1 * 10^{-3}$ because of the threshold.

4.3 Local mesh refinement

When the solution is very forward peaked and the mesh is globally refined, a lot of patches contribute marginally to the peaked solution. It is preferable to refine the mesh local and only create a fine mesh in the peak of the solution. Therefor the code was adopted to make local mesh refinement possible. To do so, the

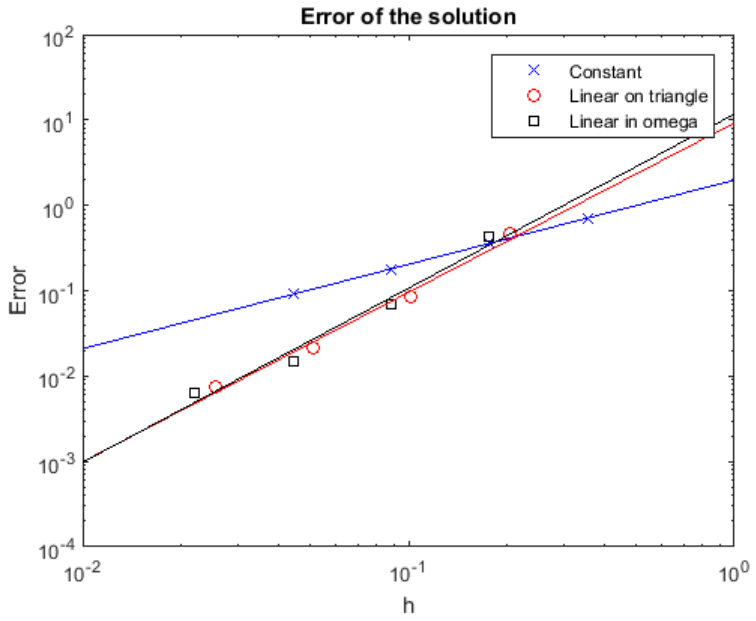

Figure 4.2: A graph showing the error, calculated as the root of the squared difference between the imposed solution and the calculated solutions, with the constant σ_s approximation on the smooth solution with a Henyey-Greenstein scatter function with parameter 0.7.

local h is compared with the desired h for that Ω. The local h is defined here as the shortest side of a patch. The comparison function used in this project is shown in figure 4.4. This function leaves the bottom patches unrefined, hence the 1.67 where $_z$ is -1. And stops when a patch is refined five times maximum, hence the 0.1 where Ω_z is 1.

Figure 4.5 shows a refined mesh using the function in figure 4.4. As can be seen, the mesh refines further with increasing Ω_z.

Both the smooth solution and the peaked solution were tested with the local refinement and for the three sets of test functions without any of the mentioned approximations. In each data point the maximum number of refinement levels is incremented. As can be seen in 4.6, local mesh refinement doesn't reduce the error as good as with global mesh refinement. This comes from the fact that the smooth solution is not constant at the unrefined patches. The solution varies over the whole sphere, so in this case local mesh refinement is not a good way to reduce the run time of the simulation. But in 4.7 it can be seen that local mesh refinement is a good way to reduce the run time with the peaked solution. The error reduces until the last refinement step, because the refined patches is in the last step lie on top of the mesh, where the solution becomes linear again. So for the best result the mesh should be refined in places where the first derivative of

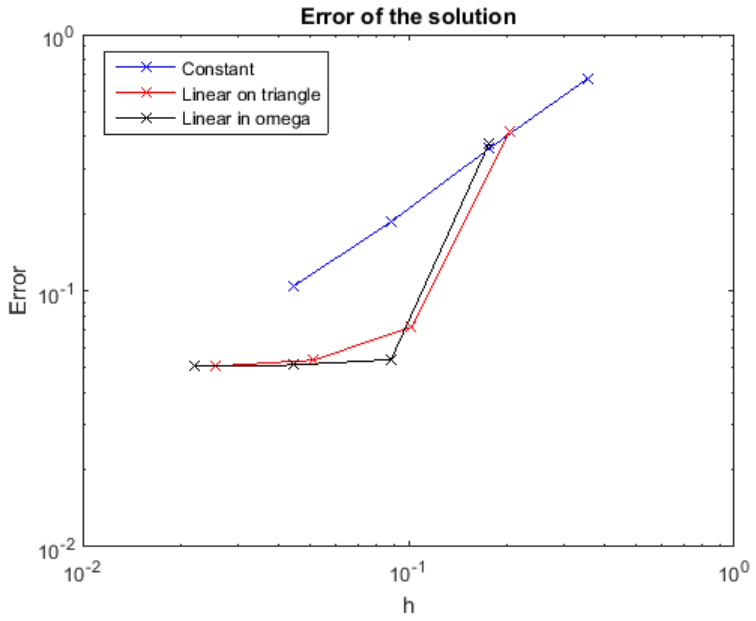

Figure 4.3: A graph showing the error, calculated as the root of the squared difference between the imposed solution and the calculated solutions, with the constant σ_s approximation and a μ_0 threshold of 0.7 on the smooth solution with a Henyey-Greenstein scatter function with parameter 0.7.

the solution has a relative high absolute value and not in places where the first derivative of the solution is close to zero.

As can be seen in 4.8, function set 2 shows a good approximation of the imposed solution. But when the function set 3 is used, as can be seen in 4.9, the approximation doesn't look like the imposed solution even though the error is smaller. The reason of this behavior is unknown.

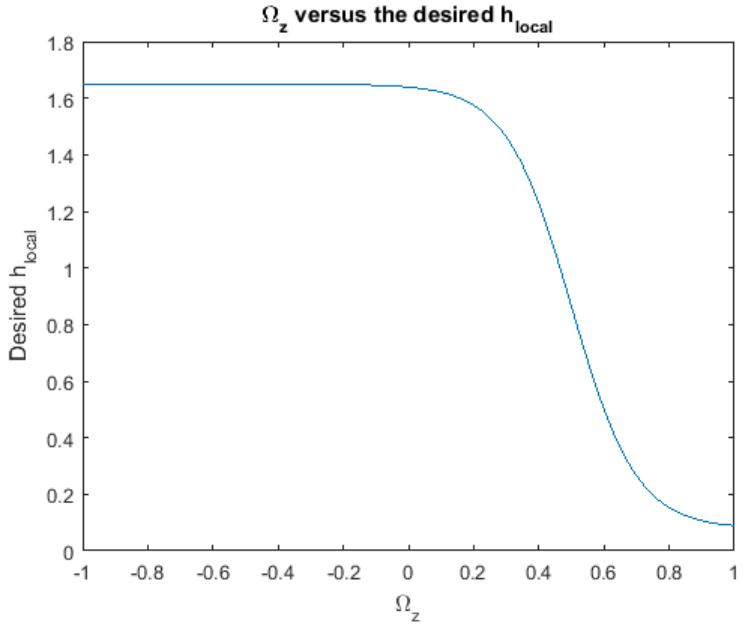

Figure 4.4: A graph showing the desired h_{local} versus Ω_z. This function is used in the local refine algorithm to check if a patch has to be refined.

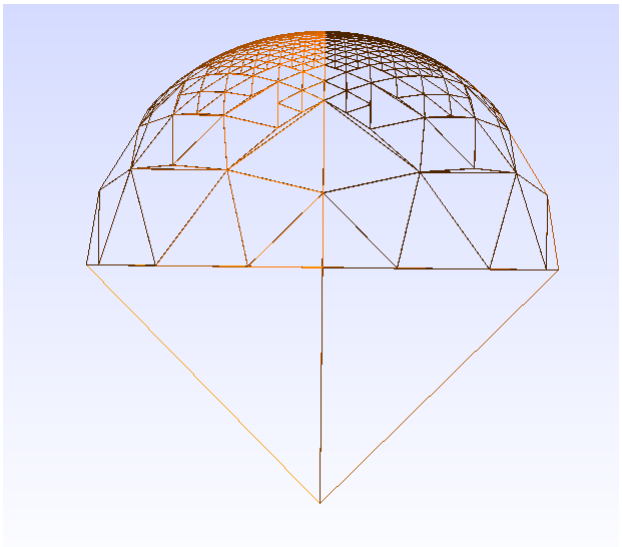

Figure 4.5: The locally refined mesh as seen for the side.

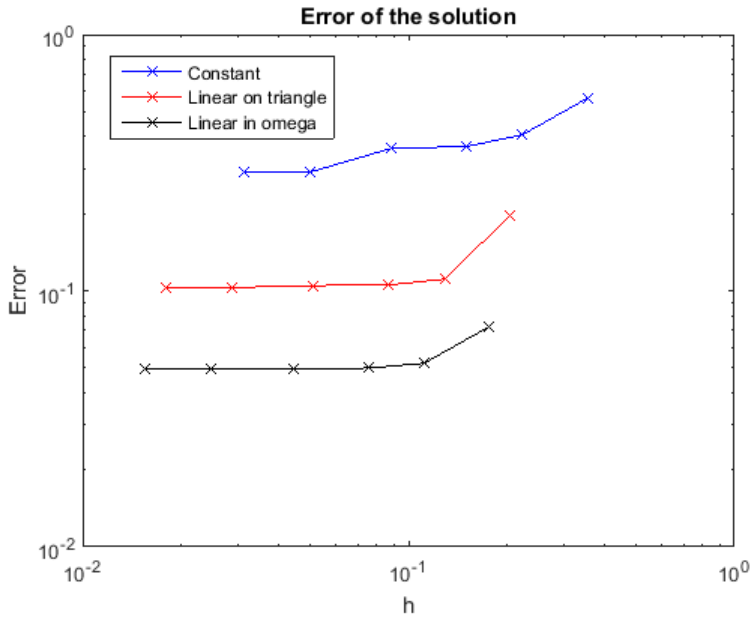

Figure 4.6: A graph showing the error, calculated as the root of the squared difference between the imposed solution and the calculated solutions, on the smooth imposed solutions with an increasing maximum level of local refinement with a Henyey-Greenstein scatter function with parameter 0.7.

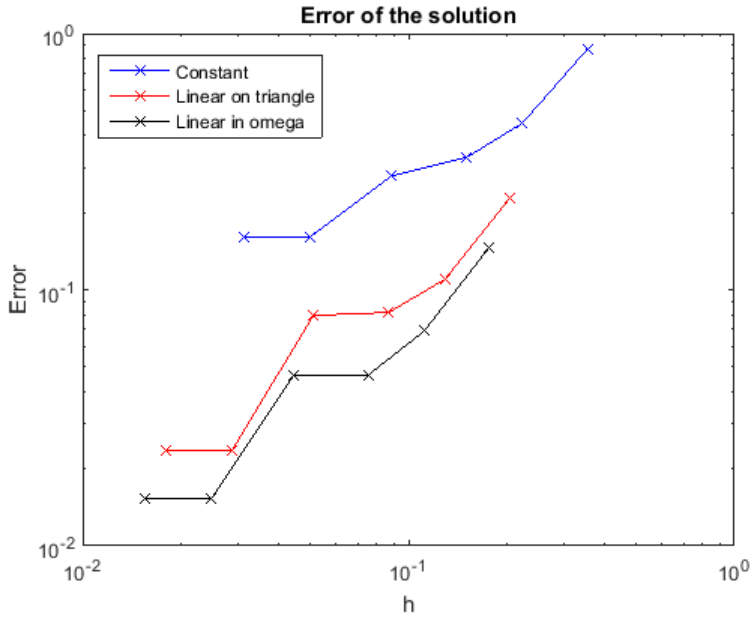

Figure 4.7: A graph showing the error, calculated as the root of the squared difference between the imposed solution and the calculated solutions, on the peaked imposed solutions with an increasing maximum level of local refinement with a Henyey-Greenstein scatter function with parameter 0.7.

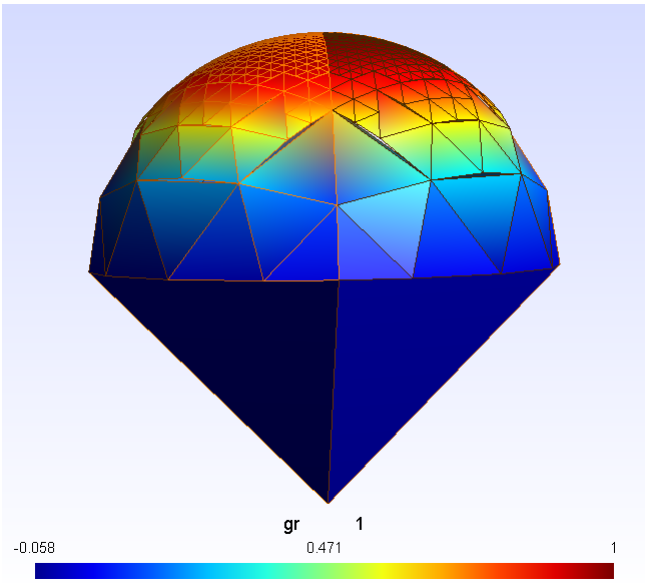

Figure 4.8: The peaked solution with function set 2 on a locally refined mesh.

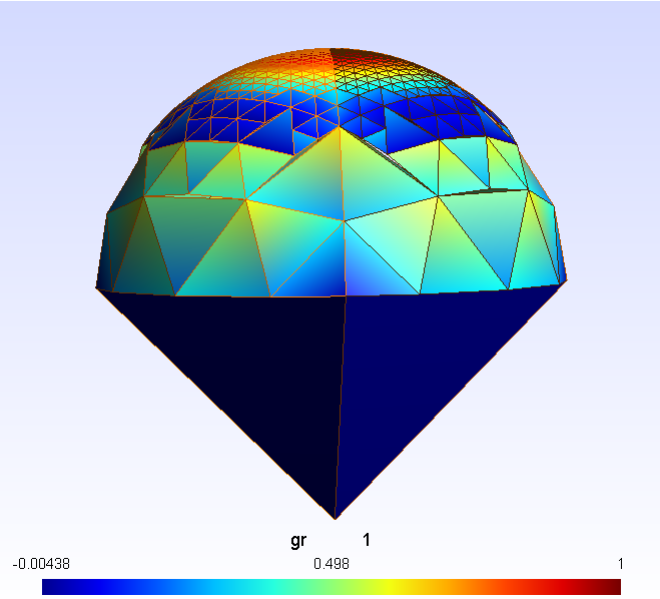

Figure 4.9: The peaked solution with function set 3 on a locally refined mesh.

Chapter 5

Conclusions

This work set out to find out if general scatter in deterministic particle transport can be modeled with the Henyey-Greenstein function in stead of using the Fokker-Planck operator.

The results of this work are mixed. On one hand the simulation gives good results when the scatter function and the imposed solution are both smooth. Also, the run time can be cut down by a factor 10000, when the constant σ_s approximation is used. But on the other hand, when the imposed solution and the scatter function becomes more peaked, the program shows unexpected behavior. And that is the region where the focus of this work lies. Proton therapy needs a very peaked (a pencilbeam) solution to minimalize collateral damage in the human body and proton scatter is also very forward peaked and not smooth.

Further research can focus on the origin of this unexpected behavior. A few examples of future research are given in the next chapter.

Chapter 6

Further research

This chapter lists two ideas for future research. First an idea for the reduction of patches with local refinement and second a way to possibly reduce the error in the quadrature of the determination of the source.

6.1 Smarter choice of the direction of the imposed solution

In this work the source and the increasing local mesh refinement were pointed at the positive y-axis. And since the mesh is first divided in the eight octants, the local mesh refinement is symmetrical at the y-axis. But when the source and the increasing local mesh refinement is pointed at the center of one of the octants, there will be less patches after the local refining. This will decrease the size of the scatter matrix and reduce the run time of simulation.

6.2 Expanding the scatter function in Legendre polynomials

In further projects the scatter function can be expanded in L Legendre polynomials.

$$\sigma_s(\mu_0) \approx \sum_{l=0}^{L} \sigma_{sl} \frac{2l+1}{4\pi} P_l(\mu_0) \tag{6.1}$$

In the case of a Henyey-Greenstein scatter function with parameter g, the coefficient σ_{sl} is equal to g^l.

This has the advantage that the source associated with the solution can be determined much more accurate. After substituting 6.1 in 2.11 the addition

theorem is applied.

$$S(\Omega) = \sigma_t \varphi(\Omega) - \int_{4\pi} \sigma_s(\mu_0)\varphi(\Omega')d\Omega' \tag{6.2}$$

$$\int_{p'} S(\Omega)h_{p'}^{i'}(\Omega)d\Omega = \sigma_t \int_{p'} \varphi(\Omega)h_{p'}^{i'}(\Omega)d\Omega - \int_{p'}\int_{4\pi} \sum_{l=0}^{L} \frac{2l+1}{4\pi} P_l(\mu_0)\varphi(\Omega')d\Omega' h_{p'}^{i'}(\Omega)d\Omega$$

$$\int_{p'} S(\Omega)h_{p'}^{i'}(\Omega)d\Omega = \sigma_t \int_{p'} \varphi(\Omega)h_{p'}^{i'}(\Omega)d\Omega - \sum_{l=0}^{L} \sigma_{sl} \sum_{m=-l}^{m=l} \phi_{lm} \int_{p'} Y_l^m(\Omega)h_{p'}^{i'}(\Omega)d\Omega \tag{6.3}$$

With the flux moments given by

$$\phi_{lm} = \int_{4\pi} \varphi(\Omega)Y_l^m(\Omega)d\Omega \tag{6.4}$$

This reduces the error in the calculation of the source and guaranties conservation of particles. And maybe it resolves the strange behavior seen in figure 4.9.

Bibliography

[1] J.J. Duderstadt, L.J. Hamilton Nuclear Reactor Analysis, 1976, Hoboken: Wiley

[2] D. Lathouwers, Essentials of Discontinious Galerkin Finite Element discretization for neutron transport, 2014, Note TUD

[3] S.B. Uilkema, Proton Therapy Planning using the S_n Method with the Fokker-Planck Approximation, 2012, MSc Thesis TUD

[4] R. Sanchez, N.J. McCormick, Discrete Ordinates Solutions for Highly Forward Peaked Scattering, 2004, Nuclear Science and Engeneering, Volume 147, pages 249-274

[5] C.D. Ahrens, Lagrange Discrete Ordinates: A New Angular Discretization for the Three-Dimensional Linear Boltzmann Equation, 2014, Nuclear Science and Engeneering, Volume 180, pages 273-285

[6] A. Hennink, A Discontinuous Galerkin Method for Charged article Transport in the Fokker-Planck Limit, 2015, MSc Thesis TUD

[7] J. Kópházi, D. Lathouwers, A Space-Angle DGFEM Approach for the Boltzmann Radiation Transport Equation with Local Angular Refinement, 2015, Journal of Computational Physics, Volume 297, pages 637-668