The Lattice Boltzmann Method applied to linear particle transport

Bernard Erasmus
22023712

Dissertation submitted in partial fulfilment of the requirements for the degree of *Master of Science in Nuclear Engineering* at the Potchefstroom campus of the North-West University

Supervisor: Dr F.A. van Heerden
South African Nuclear Energy Corporation,
Pelindaba, Pretoria, South Africa.

Co-supervisor: Prof. E. Mulder
North-West University,
Department of Mechanical and Nuclear Engineering,
Potchefstroom, South Africa.

November 2012

Abstract

In this study, the applicability of the Lattice Boltzmann Method to neutron transport is investigated. The transport model used, is derived from the Boltzmann equation for neutral particles by inverting the streaming operator and casting the integral transport equation into an operator form. From the operator equation, an iterative solution to the transport problem is presented, with the first collision source as the starting point for the iteration scheme. One of the main features of the method is the simultaneous discretization of the phase space of the problem, whereby particles are restricted to move on a lattice.

A full description of the discretization scheme is given along with the iterative procedure and quadrature set used for the angular discretization. To mitigate lattice ray effects, an angular refinement scheme is introduced to increase the angular coverage of the problem phase space.

The method is then applied to a model problem to investigate its applicability to neutron transport. Three cases are considered where constant, linear and exponential interpolants are used to account for the accumulation of flux due to the streaming of particles between nodes. The results obtained are compared to a reference solution, that was calculated by using the MCNP code and to the values calculated using a nodal S_N method. Finally, areas of improvement are identified and possible extensions to the algorithm are provided.

Keywords: Lattice Boltzmann, neutron transport.

Acknowledgements

I would like to thank my supervisor, Dr Francois van Heerden, for his suggestion of and guidance throughout this project. The insights I gained from the discussions we had, has enriched my understanding of transport theory. I am very grateful for the MCNP and nodal S_N results provided by Dr Oscar Zamonsky, and the discussions on various transport methods and their differences. I would also like to thank the South African Nuclear Energy Corporation (Necsa) for its support and the use of their facilities to conduct this research.

I would like to give a special word of thanks to Estian Behrens for performing the language editing of this thesis and to Dr Pavel Bokov for helping me with the formatting of the final version. Lastly, I would like to thank my family for their support throughout my career, in particular my brother and my fiancée.

Contents

List of Figures

Chapter 1

Introduction

Neutron transport and radiative transfer have been the subjects of study for many years (see for instance [1]–[8]). Modelling transport phenomena plays an important role in the study of radiation and its interaction with matter.

Neutral particle transport, such as neutron and gamma transport, have many applications. Neutron transport is of particular interest in the modelling of nuclear reactors and the neutron distribution in the reactor core. Transport theory is also used in radiation protection, shielding design and analysis.

Neutrons interact with matter through collisions with the atomic nuclei within a material. Collisions between neutrons and nuclei are governed by nuclear processes, in which different types of collisions can occur. The type of collision between a neutron and nuclide depends on the neutron energy and the characteristics of the target nuclei.

Between collisions, neutrons travel in straight lines. This is due to the fact that neutrons are neutral particles and their trajectories are not influenced by the charge of the atomic nuclei in the material.

When a neutron does collide with a nucleus, the neutron is either scattered or absorbed by the target nuclide. Different types of scattering and absorption can occur, such as elastic scattering, inelastic scattering, radiative capture and fission. Each of these reactions leaves the neutron–nuclide pair in different states.

For example, elastic scattering conserves both the momentum and kinetic energy of the neutron–nuclide system, while, during inelastic scattering the neutron loses some of its kinetic energy and the target nuclide is left in an excited state. The nuclide can then lose the excitation energy (decay) through different modes, normally associated with secondary particle emission.

In the case where the target is a heavy nuclide, the neutron can penetrate the nucleus forming a compound nucleus – making it highly unstable. All the kinetic energy of the incident neutron is transferred to the nucleus together with the binding energy associated with the extra neutron in the nucleus. The compound nucleus can then decay via fission, through which the compound nucleus breaks into lighter nuclides and emits a number of secondary neutrons.

Each type of reaction listed above is characterized by a nuclear cross section, which is a function of incident neutron energy. Collisions are governed by laws that determine the dynamics of the interaction, while the cross sections give a measure of the probability for a specific interaction to

occur. Reaction cross sections may strongly depend on the energy of the incident neutron. Some reactions, such as fission in ^{238}U, have a minimum energy cut-off below which the probability for the reaction to occur is very small.

When considering a beam of neutrons incident on a target, the reaction rate for a specific reaction x per unit area, is proportional to the intensity of the beam (neutrons $\cdot$ cm^{-2} $\cdot$ s^{-1}), and the number density of target nuclei in an infinitesimal width ds along the trajectories of the neutrons. The reaction rate is then defined as the number of interactions between neutrons and nuclei per second per unit area. The microscopic cross section for reaction x is then defined as the proportionality constant, and the relation is given by

$$dR_x = \sigma_x N I(s) ds, \tag{1.1}$$

where R_x is the reaction rate, σ_x is the microscopic cross section, N is the number density of the target nuclei, and $I(s)$ is the beam intensity as a function of s – the penetration depth of the neutrons along their trajectories [9].

Number density of the target nuclei refers to the number of nuclei contained per cubic centimetre in the material, and is calculated according to

$$N = \frac{\rho N_A}{M}. \tag{1.2}$$

In equation (1.2) ρ is the material density in g/cm^3, M is the molar mass of the material in g/mol, and N_A is Avogadro's constant in atoms/mol.

For equation (1.1) to be dimensionally consistent, the microscopic cross section must have units of area. Usually microscopic cross sections are expressed in terms of barns (b), where 1b $= 10^{-24}$ cm^2.

A quantity more frequently used is the macroscopic cross section, defined as

$$\Sigma_x = \sigma_x N \tag{1.3}$$

which is usually given in units of cm^{-1}. The total macroscopic cross section of a material, is the sum of the macroscopic cross sections for every possible reaction.

By using equation (1.1) and equating the total reaction rate to the number of neutrons removed from the beam per second per unit area, the equation can be written as

$$-dI(s) = \Sigma_t I(s) ds, \tag{1.4}$$

where Σ_t is the total macroscopic cross section. Integrating equation (1.4) over a thickness s gives

$$I(s) = I_0 e^{-\Sigma_t s}, \tag{1.5}$$

which is the intensity of the beam after the neutrons travelled a distance s in the target. The probability for a neutron to travel a distance s is $I(s)/I_0$, and the probability for a neutron to interact with a nucleus in a distance ds is given by

$$P(s)ds = \Sigma_t e^{-\Sigma_t s} ds. \tag{1.6}$$

From this definition of the probability of a neutron to interact with a target nuclide, the average distance or mean free path of the neutron in the target can be determined. This is done by multiplying the probability for the neutron to interact with s, and integrating over the width of the target. In an infinite target the mean free path is given by

$$\lambda = \int_0^\infty ds\, P(s)\, s = \frac{1}{\Sigma_t}. \tag{1.7}$$

From equation (1.7) it can be seen that the macroscopic cross section gives a measure of the distance a neutron will travel through a material before suffering a collision.

In general the reaction rate is defined in terms of reactions per unit *volume* per second, and not per unit area as in equation (1.1). Volumetric reaction rate is defined in terms of the macroscopic cross section and the neutron flux as

$$R_x = \Sigma_x \phi. \tag{1.8}$$

The neutron flux ϕ, also called the scalar flux, is a representation of the distance travelled by all the neutrons contained in a cubic centimetre in one second. It has units of neutrons per square centimetre per second ($\mathrm{n} \cdot \mathrm{cm}^{-2} \cdot \mathrm{s}^{-1}$). This definition of reaction rate is related to other physical quantities such as volumetric power density in nuclear reactors.

The scalar flux is the global quantity usually calculated when modelling the neutron distribution. It is related to the neutron density through

$$\phi(\mathbf{r}, E, t) = vn(\mathbf{r}, E, t), \tag{1.9}$$

with $n(\mathbf{r}, E, t)$ the neutron density at position $\mathbf{r}$, energy E and time t. In equation (1.9) v is the neutron speed related to the neutron energy E and neutron mass m_n by

$$v = \sqrt{\frac{2E}{m_n}}. \tag{1.10}$$

The neutron flux is used to calculate different reaction rates relating to other quantities, such as heat generation in reactors, and radiation damage in shields or reactor vessels.

Mathematically, neutron transport is modelled with the neutron transport equation, which is derived from the Boltzmann equation for gases. It is a statistical model for the behaviour of the neutron distribution when interacting with matter.

When modelling neutron transport, the neutrons are treated as a rarefied gas, meaning neutron–neutron interactions are neglected. Instead, only interactions between neutrons and the nuclei of the material through which they move are taken into account. Because the neutron density is low compared to the atomic density of other materials, the probability of neutrons interacting with each other is small compared to their probability to interact with the nuclei of the material. This simplifies the model by making the collision term linear.

The transport model for neutral particles can be derived directly from the Boltzmann equation by using the assumptions above, or from conservation of neutrons in a differential volume element dV. For a derivation of the neutron transport equation directly from the Boltzmann equation, see Chapter 2 of [5], and for a derivation from conservation considerations see Chapter 3 in [9].

The equation used as the model for neutron transport is:

$$\frac{1}{v}\frac{\partial\varphi(\mathbf{r},E,\mathbf{\Omega},t)}{\partial t}+(\mathbf{\Omega}\cdot\nabla+\Sigma_t(\mathbf{r},E))\varphi(\mathbf{r},E,\mathbf{\Omega},t)=Q(\mathbf{r},E,\mathbf{\Omega},t). \tag{1.11}$$

In equation (1.11) each of the symbols are:

- v – the neutron speed
- t – time
- $\mathbf{r}$ – the position vector
- E – the neutron energy
- $\mathbf{\Omega}$ – the unit direction vector
- $\varphi(\mathbf{r},E,\mathbf{\Omega},t)$ – the angular neutron flux
- $\mathbf{\Omega}\cdot\nabla$ – the streaming operator, with ∇ the differential operator
- $\Sigma_t(\mathbf{r},E)$ – the total macroscopic cross section, and
- $Q(\mathbf{r},E,\mathbf{\Omega},t)$ – the total source including scattering, fission and external sources.

The scattering source is defined as

$$q_{\text{scat}}(\mathbf{r},E,\mathbf{\Omega},t)=\int_0^\infty dE'\int_{S^2}d\mathbf{\Omega}'\Sigma_s(\mathbf{r},E\leftarrow E',\mathbf{\Omega}\leftarrow\mathbf{\Omega}')\varphi(\mathbf{r},E',\mathbf{\Omega}',t), \tag{1.12}$$

where $\Sigma_s(\mathbf{r},E\leftarrow E',\mathbf{\Omega}\leftarrow\mathbf{\Omega}')$ is the scattering kernel that changes the state of a neutron with energy E' and direction $\mathbf{\Omega}'$, to energy E and direction $\mathbf{\Omega}$ after a scattering event. The operator $\int_{S^2}d\mathbf{\Omega}$ indicates that the integration is carried out over the entire surface of the sphere, where S^2 represents the surface of the unit sphere.

When fission occurs, it is assumed that the secondary neutrons emitted are done so isotropically with respect to their angular distribution. That means that any neutron ejected from a fission event has an equal probability to have any flight direction. With this assumption, the fission source is defined as

$$q_{\text{fiss}}(\varphi,\mathbf{r},E,\mathbf{\Omega},t)=\frac{1}{4\pi}\sum_i\chi_i(E)\int_{S^2}d\mathbf{\Omega}'\int_0^\infty dE'\nu_i(E')\Sigma_{i,f}(\mathbf{r},E')\varphi(\mathbf{r},E',\mathbf{\Omega}',t). \tag{1.13}$$

Neutrons emitted during a fission event have energies distributed according to the fission spectrum $\chi_i(E)$ of the specific nuclide undergoing fission. The fission spectrum is the probability that a neutron emitted during fission will have an energy E. In equation (1.13), the index i runs over all the fissionable isotopes in the system.

Each fission event releases a number of neutrons, depending on the energy of the incident neutron. This varies from element to element and isotope to isotope and is given by $\nu_i(E)$. Together with the fission cross section $\Sigma_{i,f}$, this gives the probability for fission to occur and the average number of neutrons emitted during fission.

Finally, external sources $q(\mathbf{r}, E, \mathbf{\Omega}, t)$, are defined as sources that introduce additional neutrons into the system not related to fission or scattering. In reactors, external sources are used to introduce neutrons into the reactor core to start the fission chain reaction.

In this study only fixed source problems are considered in non-multiplicative media, and thus fission will be excluded from the source for the remainder of the work. Fixed sources introduce neutrons into the system independent of the flux in the system, and non-multiplicative media are materials in which neutron production does not occur.

The angular flux in equation (1.11) is related to the scalar flux by

$$\phi(\mathbf{r}, E, t) = \int_{S^2} d\mathbf{\Omega} \varphi(\mathbf{r}, E, \mathbf{\Omega}, t), \tag{1.14}$$

and gives the maximum information about the neutron distribution. By solving equation (1.11) for the angular flux all information regarding the neutron distribution can be recovered.

Each term in equation (1.11) is related to a physical process. Streaming of the neutrons is described by the term $\mathbf{\Omega} \cdot \nabla \varphi(\mathbf{r}, E, \mathbf{\Omega}, t)$. It describes the physical transport of neutrons along their flight paths between collisions, and gives the variation in neutron position along the direction $\mathbf{\Omega}$.

$\Sigma_t(\mathbf{r}, E)\varphi(\mathbf{r}, E, \mathbf{\Omega}, t)$ is the removal term that describes the number of neutrons that are removed through collisions. All neutron capture collisions are taken into account when considering removal, such as radiative capture, secondary particle emission (excluding neutrons), transmutation, etc.

The source term represents all the neutrons introduced into the system and includes a scattering source. Collisions act as sources for the energy E and direction $\mathbf{\Omega}$ by changing the neutron energy and direction from E' and $\mathbf{\Omega}'$ to E and $\mathbf{\Omega}$ through the scattering kernel $\Sigma_s(\mathbf{r}, E \leftarrow E', \mathbf{\Omega} \leftarrow \mathbf{\Omega}')$.

Each of the physical quantities in equation (1.11) depends on a number of independent variables, seven in total. Dependence of the neutron flux and the cross sections on each of the independent parameters is typically approximated in order to simplify equation (1.11).

Time dependence can be removed when considering steady-state neutron distributions. This means that, in the system under consideration, enough time has elapsed for the neutrons to have an equilibrium distribution. Equation (1.11) is then reduced to the steady-state transport equation and the time dependence can be removed. In the scope of this study, only steady-state solutions will be considered and thus the time dependent term is neglected.

The energy variable in reactor analysis spans a large range from electron Volt (eV) to Mega electron Volt (MeV). Apart from some Monte Carlo codes, most transport codes treat the energy dependence by dividing the energy spectrum into broad energy groups. Multi-group cross section libraries are generated from thermilization calculations of neutrons slowing down in an infinite homogeneous medium. From the thermilization calculations cross sections per energy group are generated, where the cross section is averaged over an energy range [9].

In this study, a one-group or mono-energetic model is used and therefore the energy dependence of the flux and cross sections are averaged over the entire energy spectrum. These simplifications reduces equation (1.11) to the one-group, steady-state neutron transport equation

$$\mathbf{\Omega} \cdot \nabla \varphi(\mathbf{r}, \mathbf{\Omega}) + \Sigma_t(\mathbf{r})\varphi(\mathbf{r}, \mathbf{\Omega}) = \int_{S^2} d\mathbf{\Omega}' \Sigma_s(\mathbf{r}, \mathbf{\Omega} \leftarrow \mathbf{\Omega}')\varphi(\mathbf{r}, \mathbf{\Omega}') + q(\mathbf{r}, \mathbf{\Omega}). \tag{1.15}$$

The heterogeneity of the problem, makes equation (1.15) difficult to solve analytically. Besides the energy dependence, there is the spatial dependence of the cross sections, which means that the cross sections can vary greatly between different material zones. Usually the problem is discretized into a number of regions, or volumes, each containing a single material, and equation (1.15) is solved in each of these regions. Finally, the solution in each of the volumes is coupled to the solution in the neighbouring regions and used as sources during an iteration procedure.

Lastly, the dependence of the angular flux on $\mathbf{\Omega}$ is treated by dividing the angular domain into a finite set of directions, along which equation (1.15) is solved. The set of directions is chosen so that with spatial discretization the entire phase space of the problem is approximately covered.

There is a variety of transport solution methods currently used to calculate the neutron distribution in reactor analysis. Two classes of method exist, which can be categorized as stochastic (Monte Carlo) or deterministic.

Monte Carlo methods use Markov processes to simulate physical phenomena. Sampling of variables is done by generating random number sequences to sample probability distributions related to the system being simulated [10]. The main advantage of Monte Carlo methods is the level of accuracy with which the neutron population can be simulated. Although the method is precise, it converges slowly, and its main drawback is the large amount of simulation time needed for the method to fully converge. This study, however, focuses on the deterministic solution methods of the neutron transport equation and readers interested in Monte Carlo methods are referred to works by Spanier and Gelbard [10] and Lapeyre, Pardoux and Sentis [11].

Deterministic space-angle methods used in transport calculations can be divided into classes according to how the streaming operator $\mathbf{\Omega} \cdot \nabla$ is treated, such as the integrodifferential-, integral- and surface-integral methods. For the purpose of this classification the treatment of the energy variable is neglected.

Although deterministic methods have the disadvantage of reduced accuracy when compared to Monte Carlo methods, their calculational times are significantly shorter when solving complex problems. Deterministic methods are used in shielding analysis as well as depletion calculations, where it is required to calculate the number density of fissionable isotopes as they are depleted during reactor operation.

Transport methods most commonly used in reactor calculations include the Spherical Harmonics Method [12, 13], the Discrete Ordinates Method [14, 15, 16], Collision Probabilities Method [17, 18, 19] and the Method of Characteristics [20]–[23].

A review of the transport methods that are currently used in reactor analysis is given in the following sections, followed by a review of the Lattice Boltzmann Method (LBM). In the review the major benefits and drawbacks of each method are given with their most common uses. Of particular interest are the multi-processor implementations of these transport methods. This chapter concludes with a comparison between the methods and with an outline of the project.

1.1 A brief review of neutron transport calculational methods

This section gives a description of some of the methods commonly used in neutron transport calculations. In particular, an overview is given of methods which exhibit similar behaviour and possess

similar qualities to the LBM. A wide variety of neutron transport methods exists and interested readers are referred to the review given by Sanchez and McCormick [24].

1.1.1 The Discrete Ordinates Method

The Discrete Ordinates Method is used to solve the differential form, equation (1.15), of the neutron transport equation. This method, also called the S_N method, was first suggested for neutron transport by Carlson in 1958 [14]. A quadrature set $S_N = \{\mathbf{\Omega}_n, w_n\}$ with N even, of discrete angles $\mathbf{\Omega}_n$ with corresponding weights w_n, are chosen along which the transport equation is solved. The number of discrete directions is given by N in 1D, $\frac{1}{2}N(N+2)$ in 2D and $N(N+1)$ in 3D [15].

The quadrature set is chosen so that it integrates the spherical harmonics exactly, even when low values for N are used. Quadrature sets are usually chosen so that the base points in each of the octants in 3D are invariant with respect to $\pi/2$ rotations [9]. These sets are called level symmetric quadrature sets. When calculating integral quantities, such as the scalar flux, the angular integral is approximated by

$$\phi(\mathbf{r}) = \int_{S^2} d\mathbf{\Omega}\varphi(\mathbf{r}, \mathbf{\Omega}) \approx \sum_n w_n \varphi_n(\mathbf{r}). \tag{1.16}$$

In equation (1.16), the subscript n indicates that the angular flux is evaluated at one of the discrete directions in the set S_N, where $\varphi_n(\mathbf{r}) = \varphi(\mathbf{r}, \mathbf{\Omega}_n)$. The angular flux is evaluated at the chosen quadrature points, with corresponding weights, to produce an approximate value of the integral. The scattering source in equation (1.15) is obtained by expanding the scattering cross section Σ_s in terms of Legendre polynomials.

Different conditions for choosing the weights and directions can lead to different types of quadrature sets. Generally, the choice depends on the symmetry of the problem that is being solved. By choosing the directions optimally the accuracy of the method can be increased. For example, using Gauss-Legendre and Gauss-Chebyshev quadrature sets instead of a level symmetric quadrature set in 1D cylindrical geometry [9].

When discretizing the angular dependence of the flux into a finite set of directions, equation (1.15) becomes

$$\mu_n \frac{d\varphi_n(x)}{dx} + \Sigma_t(x)\varphi_n(x) = \sum_j w_j \Sigma_s(x, \mu_n \leftarrow \mu_j)\varphi_j(x) + q_n(x), \tag{1.17}$$

where μ_n is the direction cosine of the discrete streaming directions. In order to obtain the 1D form of the transport equation, Equation (1.17), it is assumed that the angular flux has no y or z dependence.

Finite difference relations are typically used to discretize the differential operator. Subdivision of the problem into regions with constant material properties, allows for the set of coupled equations to be solved numerically along the discrete angular directions. Regions are then coupled to each other through the angular flux on their boundaries.

By integrating equation (1.17) over a region, a balance relation is obtained between the incoming, outgoing and region-centred flux values along a specific direction i.e.

$$\mu_n(\varphi_{n,i+1/2} - \varphi_{n,i-1/2}) + \Delta x_i \Sigma_{t,i} \varphi_{n,i} = \Delta x_i q_{n,i}, \tag{1.18}$$

where Δx_i is the size of the region centred on x_i, $\varphi_{n,i\pm1/2}$ are the angular flux values evaluated at the boundaries of the region and $\varphi_{n,i}$ is the average angular flux in the region. $\Sigma_{t,i}$ is the total cross section in region i, and $q_{n,i}$ is the average source in the region calculated as

$$q_{n,i} = \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} dx q_n(x). \tag{1.19}$$

To close the set of coupled equations, an additional equation is required relating the incoming, average and outgoing flux along the same direction. Closure relations can be in the form of weighted differencing schemes relating the three quantities via

$$\varphi_{n,i} = a\varphi_{n,i+1/2} + (1-a)\varphi_{n,i-1/2}, \tag{1.20}$$

where a is the weighting parameter. An approximation such as the diamond difference approximation ($a = \frac{1}{2}$) is an example of such a scheme. The diamond difference approximation relates the incoming and outgoing flux of a region to the average flux in that region. Alternatively, finite element methods can also be used to perform the spatial discretization.

Subsequent application of the finite difference relations may in some cases lead to negative angular fluxes. This places a restriction on the mesh size given (in 1D) by

$$\Sigma_t \Delta x < \frac{|\mu_n|}{1-a}, \tag{1.21}$$

and plays a role in the anomalous flux behaviour related to ray effects.

In the expression above, Σ_t is the total cross section, Δx is the size of the spatial mesh, μ_n is the smallest direction cosine and a is the weighting factor of the differencing scheme being used. The occurrence of negative fluxes is most pronounced when the diamond differencing scheme is used. In the case where step differencing is used ($a = 1$), Equation (1.21) shows there is no restriction on the spatial mesh size and negative fluxes cannot occur [15].

Once the discretization has been performed, the region-centred fluxes can be calculated by using equations (1.18) and (1.20) in a sweeping algorithm starting from the boundary of the problem. The boundary conditions are applied to determine the incoming flux to the regions at the boundary. From the angular flux in each region, different quantities relating to the original problem can be calculated.

The main drawback of the S_N method is the so called ray effect. Interaction of the spatial differencing scheme and the angular quadrature can cause unphysical flux solutions, because neutrons are only transported along the set of discrete directions. For instance, if an isotropic source is present in a problem, some parts of the domain far from the source might not "see" flux because there are no angles directly connecting this region with the source. These regions are linked to the source only through the spatial differencing scheme which causes numerical diffusion by linking regions to each other through their boundaries. Numerical diffusion may become more pronounced depending on the value of a – the weighting parameter of the differencing scheme. Ray effects also become more pronounced in regions of low or no scattering.

By increasing the number of angles in the set S_N, ray effects can be reduced but cannot be completely eliminated, as it is an effect of the discretization itself. Improvements on the Discrete Ordinates

Method are given in [16]. However, the increased streaming directions lead to more computer storage requirements and longer calculational times.

In a related method, the spherical harmonics are used to expand the angular flux to a chosen order. This is called the Spherical Harmonics Method, and is one of the oldest methods applied to solve neutron transport problems (see [12, 13]). The flux expansion is substituted into the differential form of the transport equation which produces a system of coupled differential equations. For each order of flux expansion, a new equation is added to the coupled system.

Each generated equation has terms that contain the next order of spherical harmonics. The expansion is truncated at some chosen order, and an approximation is made for the higher order moment in terms of the current or lower order moments. By truncating the flux expansion, and by making the approximation for the higher order moment, the system of equations is closed. As with the S_N method, finite difference schemes can be used to approximate the differential operator.

One major advantage of the S_N method is that it is more suited to the implementation of a general solution routine; in contrast with the Spherical Harmonics Method. In the Spherical Harmonics Method, each order of approximation has its own set of typical equations, making it difficult to implement a general routine for solving transport problems. For the S_N method a general program can be implemented, taking higher order approximations into account without changing the algorithm [15].

Sweeping of the flux along the different directions of the quadrature set, can be calculated in parallel. Once all the flux solutions from different directions have been calculated, they can be summed to give the outgoing/incoming flux on the region boundaries.

1.1.2 The Collision Probability Method

In the previous section, the S_N method was described as a solution method which treats the streaming operator directly in its differential form. To transform equation (1.15) to an integral equation, the streaming operator $\mathbf{\Omega} \cdot \nabla$ needs to be inverted. This can be done by writing the operator as a total derivative by making a suitable variable change. After the variable change is made, equation (1.15) reduces to an ordinary differential equation, and can be integrated to find the integral form of the neutron transport equation

$$\varphi(\mathbf{r}, \mathbf{\Omega}) = e^{-\tau(l)}\varphi(\mathbf{r} - l\mathbf{\Omega}, \mathbf{\Omega}) + \int_0^l ds e^{-\tau(s)} \int_{S^2} d\mathbf{\Omega}' \Sigma_s(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega} \leftarrow \mathbf{\Omega}')\varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}') + \int_0^l ds e^{-\tau(s)} q(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}), \quad (1.22)$$

with

$$\tau(s) = \int_0^s ds' \Sigma_t(\mathbf{r} - s'\mathbf{\Omega}), \quad (1.23)$$

the optical path through the medium. A more detailed derivation of the integral transport equation is given in Chapter 2. The main difference between methods, such as the Discrete Ordinates Method and integral methods, is the way in which the streaming operator is treated – either explicitly or inverted.

Collision Probability (CP) methods are used to solve the integral neutron transport equation [3, 4,

17]. The calculational domain is divided into regions where it is approximated that both the flux and sources are spatially constant in each region. If the domain is divided into N regions containing homogeneous mixtures, the CP method produces an $N \times N$ matrix which relates each region to all other regions in the domain.

The matrix entries are the probability that a neutron born in region i will have its first collision in region j. In the CP method, it is typically assumed that the neutrons are emitted isotropically with a uniform distribution throughout the region of origin. Once the matrix entries have been calculated they can be used to determine the average flux in every region, which are the principle unknowns in the CP method. Matrix elements in the collision probability matrix are scalar quantities, which are independent of both the neutron distribution and the sources.

A tracking process is required that spans the geometry with enough neutron trajectories to accurately calculate the integrals contained in the matrix entries. The tracking information is used in conjunction with a numerical integration scheme to evaluate the collision probabilities in each region of the geometry.

CP methods are mainly used for calculations with few, optically thin regions, such as in reactor lattice calculations. This is because calculational domains with many regions produce very large collision probability matrices that require large amounts of computer memory.

One of the main advantages of the CP method is that the matrix entries only need to be calculated once. To calculate the integrated flux, the CP matrix can be applied repeatedly throughout the scattering iterations.

A drawback of the method is the poor spatial representation of large flux gradients, when a coarse spatial mesh is used with a spatially constant flux approximation. If the spatial mesh spans large regions where the flux decreases rapidly, such as in absorbers, the flat flux approximation is not adequate to represent the flux behaviour. This forces the spatial mesh to be reduced in size and increases the number of matrix entries to be calculated. This is, however, not unique to the CP method and applies to other methods when the same constant flux approximation is used.

As an alternative to the strong coupling between regions used in the normal CP method, the Interface Current (IC) method can be used. When using the IC method the calculational domain is divided into cells. Within a cell, there are then a number of regions for which the collision probabilities have to be calculated.

Each cell is coupled to its neighbouring cells by using the currents on the interfaces between them. The currents on a cell's interfaces with its neighbouring cells are used to calculate the neutronic response of that cell to the incoming flux. Once the response of each cell has been calculated, the detailed flux can be reconstructed.

Although the IC method eases some of the calculational burden, the incoming flux to each cell is assumed to be isotropic and the addition of anisotropy will incur calculational expense.

Anisotropy can also be included in the CP method [18, 19], but at a high computational cost. With each order of anisotropy added, a full collision probability matrix calculation is required. For this reason, CP methods are preferred for problems where the flux and scattering have isotropic distributions.

Parallel implementation of the CP method is possible as each of the matrix entries can be calculated independently, but the flux calculation in each region requires that the whole CP matrix be available.

The full matrix produced by the CP method can be reduced by using the IC method which couples cells. However, using it only adds to the efficiency if there are many similar cells in the system.

1.1.3 The Method of Characteristics

Another integral method used in reactor analysis to solve the neutron transport equation, is the Method of Characteristics (MOC). Application of the MOC to neutron transport originated in 1972 [20, 21].

The MOC is a method used to solve partial differential equations by integrating along the characteristics of the differential operator [23]. Arrays of parallel lines (characteristics) are constructed and are used to track through the geometry of a specified problem. Along the characteristics, the differential operator becomes a total derivative. The neutron transport equation is solved along the neutron paths as they cross the calculational domain.

For every line segment l_{ij}, intersecting a region in the geometry, the average angular flux and average source along the segment are calculated. Each of the characteristics represents a tube of cross sectional area A_{ij}, meaning the flux in the tube is approximated by the flux calculated along the single line it contains. The subscripts i and j refer to direction $\mathbf{\Omega}_i$ and ray j (see Figure 1.1).

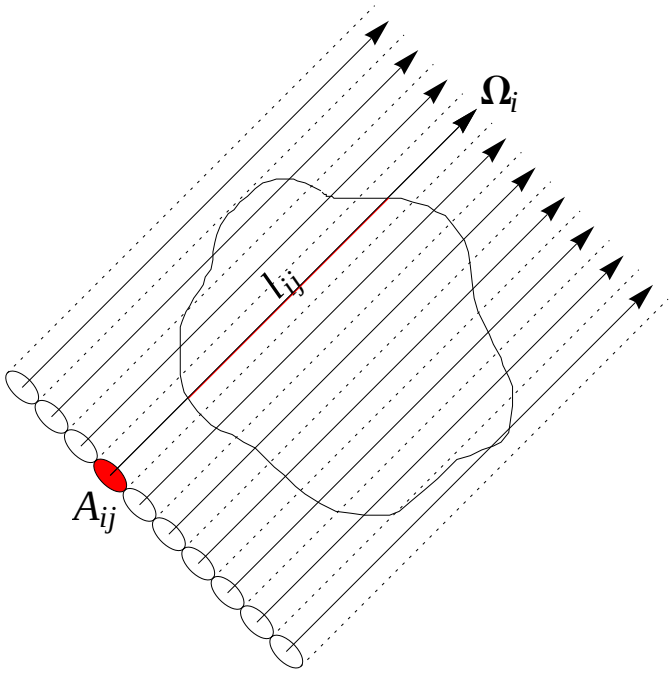


Figure 1.1: Set of parallel rays (characteristics) used in the MOC.

Arrays of characteristics are created for a number of different directions to cover both the angular and spatial domain of the problem. Once the average flux is calculated for each of the lines created, they are summed to calculate the average flux in each region of the geometry.

Scattering contributions to the flux in each region are incorporated in the source term, and because the source term depends on the flux, the calculation of both the source and flux is placed in an iterative scheme. In each step of the iteration, the absorption along a characteristic is taken into account to calculate the flux. From the current iteration, the calculated flux is used in the following

iteration to calculate the source and corresponding flux. The iteration is then continued until convergence is reached.

As an alternative, the MOC overcomes some of the problems encountered with CP methods. Where the CP method produces full matrices, the MOC is a matrix free method. This is, however, only an advantage if the track data is not stored. The different regions in the calculational domain are connected directly by the characteristics, and the neutron distribution in each region is calculated by summing the contribution of the characteristics going through that region. Sources and absorption are treated explicitly along the characteristics.

For calculations that contain a large number of regions, it is preferred to use the MOC instead of the CP method, because the computer resources needed to calculate the neutron distribution are less. There is no need to store a matrix in order to calculate the neutron distribution. For problems with very small regions, however, large characteristic (track) densities are required to give adequate spatial coverage. The choice of angles at which the integration lines are tracked through the geometry is also important. To insure proper angular coverage, a large number of angles are needed if there are many small regions in the geometry.

The same tracking data used in the CP method can also be used in the MOC. This makes it easy to implement in environments where CP methods are already used, while the MOC can also be extended to include anisotropic effects [22].

There is potential for parallel implementation of the MOC. Not only can the tracking procedure be parallelized, but after the entire domain has been tracked, the flux in each of the regions can be calculated independently.

1.2 A review of the Lattice Boltzmann Method

1.2.1 Historical development

The Lattice Boltzmann Method (LBM) historically originated from the lattice gas automata or lattice gas cellular automata [25, 26, 27]. The Lattice Gas Automaton (LGA) is an idealisation of physical systems where the physical quantities, such as time and space, only take a discrete set of values. For example, the occupation numbers on the lattice sites can either be 0 or 1, with the exclusion principle that no more than one particle with a specific velocity can occupy a lattice site. This exclusion principle naturally leads to a Fermi-Dirac local equilibrium distribution [28].

Lattice based methods, such as LGA, restrict particles to move from node to node on a lattice with discrete velocities. The velocities are defined such that after each time step a particle has moved from its current lattice site to a neighbouring site along a lattice direction.

Collisions between particles are also restricted to lattice sites and are controlled by deterministic collision rules such as particle and momentum conservation. This makes the LGA a fully discrete molecular dynamic model, based on (fictitious) particles on a lattice [26].

The LGA does, however, suffer from statistical noise, and obtaining the correct macroscopic averages requires long run times to accumulate proper statistics. Statistical noise is introduced because a single collision between particles can have multiple outcomes which are governed by probability

distributions. To determine the outcome of a collision, the probability distribution needs to be sampled.

As an alternative numerical simulation method to the LGA, the Lattice Boltzmann Method originated. The same macroscopic quantities can be determined by simply shifting from the LGA to the LBM. In the LBM, the occupation numbers (0 or 1) are replaced by a smoothly varying single particle distribution function, while still using the discrete velocities as in the LGA. Boolean operations that govern the collisions in LGA on the lattice sites are replaced with the appropriate arithmetic operations corresponding to the collision rules [29]. This makes the LBM the floating-point counterpart of the LGA, without the statistical noise.

If the particle density at a lattice site $\mathbf{r}$, at time t, in direction i is written as $f_{\mathrm{i}}(\mathbf{r}, t)$, then the lattice Boltzmann equation is written as:

$$f_i(\mathbf{r}, t+1) = f_i(\mathbf{r} - \mathbf{v}_i, t) + \mathrm{C}_i(f_i(\mathbf{r} - \mathbf{v}_i, t)), \quad i = 0, 1, ..., M \tag{1.24}$$

where i is a lattice direction, $\mathbf{v}_i$ is the discrete particle velocity along direction i, and C_i is the collision operator which characterizes the collisions at lattice sites. In Equation (1.24) $f_i(\mathbf{r} - \mathbf{v}_i, t)$ is the particle density at a neighbouring lattice site as time t, streaming to the current node along direction i. Equation (1.24) is written so that the particles stream to a neighbouring lattice site first before undergoing a collision.

Despite the historical development of the LBM, it can directly be derived from the continuous Boltzmann equation [30, 31]. Starting from the (Bhatnagar, Gross, Krook) BGK form of the Boltzmann equation, the lattice BGK equation is recovered by simultaneously discretizing the coordinate and momentum space. As before, the momentum space is discretized by restricting the particle velocity directions to lattice directions. Velocities are restricted so that particles move to the nearest neighbouring lattice site along the particular lattice direction at every time step.

The LBM has been used mainly for fluid dynamics simulations [32, 33, 28], where complex interactions need to be modelled. It has subsequently been used to simulate turbulent flow, multiphase flow and Rayleigh-Taylor instability [34], flow around a cylinder [35] and flow through porous media [36].

Because of the difficulty of treating binary mixtures with the Navier-Stokes equation, the LBM presents an alternative for simulating suspension flows. For example, this method has been used to simulate snow transport by wind [25].

The kinetic nature of the method, moreover, makes it possible to model a variety of physical phenomena. LBMs are based on the physics of the underlying microscopic system, and can simulate systems that have no current macroscopic models. For applications of this method to fluids and model approximations relating to the Navier-Stokes equation, see [37]–[42].

Therefore, the simplicity and ease of parallel implementation of the LBM, makes it an attractive method to perform neutron transport calculations. In contrast with fluid simulations, the algorithms are simplified further by the fact that only linear interactions need to be considered for neutron transport.

1.2.2 The Lattice Boltzmann Method applied to neutron transport

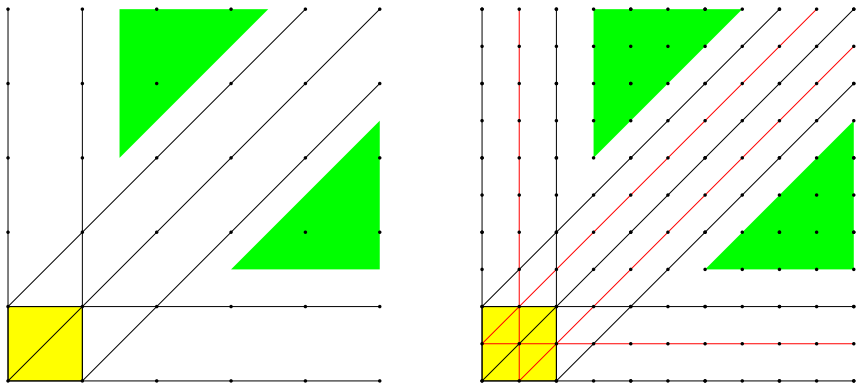
Material cross sections are used to characterize the neutron–nuclei interactions at lattice sites. The fact that the collisions are restricted to the lattice sites, divides the calculational algorithm into streaming and scattering parts. This separation in the algorithm allows each of the nodes to be updated independently of its neighbours. At each streaming step the absorption and sources are treated explicitly as the neutrons move from one node to another.

By treating absorption explicitly between nodes, the lattice spacing is not restricted by the total neutron mean free path – which may be considerably shorter than the scattering mean free path. However, the lattice size is restricted by the scattering mean free path because the scattering is restricted to the lattice nodes. This means for large problems with high scattering regions, a large lattice will be generated. Moreover, if the calculational domain contains regions smaller than the scattering mean free path, there is no guarantee that every region will contain a scattering node.

Lattice ray effects are another problem that the LBM faces. Because the streaming directions are restricted by the lattice, blind spots in the angular coverage of the domain occur. In a fluid flow simulation there is no need to take this effect into account, as the fluid particles constantly scatter from boundaries and each other, and no localized sources are present. In order to solve this problem, a mesh and angular refinement strategy needs to be implemented.

Ray effects, as referred to in the LBM case, are similar to the ray effects encountered in the S_N method. Where ray effects in the LBM are also related to the fact that only a finite set of directions are used to cover the angular space of the problem, there is no differencing scheme in the LBM that causes numerical dispersion of the flux away from the streaming directions. Because of the absence of a differencing scheme, no unphysical flux solutions are obtained at nodes away from streaming directions.

This difference between the S_N method and the LBM is better illustrated when considering a problem without scattering with a localized source, as in Figure 1.2. When no scattering is present, only the nodes connected by the black lines in Figure 1.2(a) will have flux contributions from the source when the LBM is used.



(a) Lattice with blind spots caused by lattice directions.

(b) Spatially refined lattice with blind spots still present.

Figure 1.2: Lattice ray effects, in two dimensions, encountered in the LBM as a result of the limited number of lattice directions.

In contrast to this, when the S_N method, is used the numerical dispersion caused by the differencing

scheme will result in non-zero flux values at the points in the green triangles. These non-zero flux values are purely caused by the numerical methods used, and not physical phenomena such as scattering or streaming.

Some nodes on the lattice in the LBM are, however, only connected to the source through the scattering and subsequent streaming mechanisms. Connecting nodes to the source in this secondary way, leads to the underestimation of the flux on nodes located in blind spots on the lattice.

Figure 1.2(a) illustrates how and where blind spots on a lattice occur (in 2D) as a result of the limited number of lattice directions. The neutron source is situated at the bottom left corner (yellow), while the blind spots are indicated with green triangles. In Figure 1.2(b) it is illustrated that blind spots caused by the limited number of lattice directions cannot be eliminated by simply refining the lattice.

A possible solution to the lattice ray effect encountered in the LBM, is the implementation of the first collision source method [43]. When using the first collision source method, the contribution to the neutron flux as a result of the source is calculated at each lattice node. This eliminates the initial blind spots on the lattice caused by the limited number of the lattice directions.

When calculating the first collision source, only the absorption and out-scattering is taken into account. Out-scattering here refers to neutrons that change direction as a result of a collision and does not contribute to the angular flux at the node for a specific direction.

To calculate the first collision source contribution at a lattice node, the integral transport equation (1.22) is solved by integrating over the source region(s). This can be accomplished either numerically or analytically depending on the complexity of the source(s). Once the first collision source is calculated, the source has effectively been distributed.

Although implementing a first collision source method will eliminate the initial blind spots on the lattice, the angular discretization will still need refinement to correctly account for the flux contributions from off-lattice directions. This is especially true for nodes located far from sources where flux contributions from lattice directions are overestimated because of the low angular quadrature.

1.3 Comparison between the features of the different methods

In comparison with other transport methods, the LBM is also a matrix free method as in the case of the Method of Characteristics. Like the Method of Short Characteristics (an extension of the method of long characteristics) the absorption and sources present in the domain are treated explicitly along the characteristics.

The angular variable is discretized, as mentioned before, by a finite number of directions or tracks along which the neutrons travel. This is the same as in the Discrete Ordinates Method and the Method of Characteristics, but the tracks are restricted to the lattice directions. By making this restriction, there is no freedom to choose a quadrature set.

Quadrature weights, used for the angular integration, can be chosen so that they depend on the basis functions used for the expansion of the angular flux.The basis functions chosen may also have local support to more accurately capture the streaming phenomenon, without excessive dispersion as in the case of the Spherical Harmonics Method.

Although the lattice is restrictive, no attempt is made to ensure that the tracks fall within homogeneous regions. This means that the tracks between nodes may cross several regions with different material properties, retaining the most attractive feature of traditional Method of Characteristic schemes. For purely absorbing media, the LBM is equivalent to the Method of Characteristics, with transport only along the lattice directions.

1.4 Outline

In this study, the applicability of the LBM to linear particle transport is investigated, specifically neutron transport. A simplified, steady-state, mono-energetic transport model will be used for this purpose. The work presented here aims to introduce the LBM as an alternative way to calculate the neutron distribution in linear transport problems.

A derivation of the steady-state, mono-energetic transport model is presented in the next chapter, which includes the derivation of a model scattering kernel. From the integral transport equation, the scattering and transport operators are redefined and used to cast the transport equation into an operator form. An iterative solution to the operator equation is then presented, and the first collision source method is introduced. The discretized equations are derived directly from the continuous Boltzmann equation.

Chapter 2 also includes a detailed description of the calculational lattice and the interpolation rules, both in space and angle, used to determine the neutron density at off-lattice points. It is shown how the quadrature weights that are used to approximate the angular integrals are related to the basis functions chosen to represent the angular flux. Strategies for both mesh and angular refinement are also discussed. The chapter concludes with a description of how the first collision source is calculated, and how this calculation differs from the rest of the algorithm.

Thereafter, the complete calculational algorithm is described in Chapter 3, together with all the relevant data structures that are used and storage requirements that are needed. A description of how the first collision source is implemented is also given in this chapter. In the description of the algorithm, the implementation of the scattering model is given, with an implementation of the corresponding iteration scheme.

The Lattice Boltzmann Method is then applied to a simple 3D radiation transport problem that demonstrates the main features of the method, and it's capability to model transport phenomena. The problem includes a localised source in a scattering medium surrounded by a vacuum. Reference results are calculated using the Monte Carlo Neutral Particle transport code (MCNP), and comparisons of the LBM results to both the reference solution and a nodal S_N code are presented.

Chapter 5 concludes with comments on the results of the simulations and the applicability of the Lattice Boltzmann Method to linear transport problems. Although the model used in the study presented here is very simplified, it is aimed at introducing the Lattice Boltzmann Method as a possible fast alternative to methods currently used.

Chapter 2

The Lattice Boltzmann Method as applied to neutron transport

This chapter contains the full theoretical description of the Lattice Boltzmann Method (LBM) as applied to neutron transport. Here, a derivation of the integral form of the neutron transport equation is given. The first collision source method is described, and the transport equation is cast into operator form, for which an iterative solution is derived.

The theoretical description given here, is the starting point for the calculation of the neutron distribution by using the LBM. Discretization of the equations will be presented once the basic theoretical background is established. A description of how the first collision source is calculated at any point in space will be given at the end of this chapter. Distribution of the first collision source to the lattice nodes, will be treated in the Chapter 3 with a complete description of the algorithm.

2.1 The transport equation in characteristic form

The characteristic curves (or characteristics) of a partial differential equation (PDE), are the curves along which the partial derivatives become total derivatives [23]. Transformation of the equation to its characteristic form can be accomplished by making a suitable variable change. Once the equation has been reduced, it can be solved for the solution of the original PDE along its characteristics.

Starting from equation (1.15), the characteristic equation is derived by making the substitution $\mathbf{r} \rightarrow \mathbf{r} - s\mathbf{\Omega}$. The transport equation can then be written as

$$-\frac{d}{ds}\varphi(\mathbf{r}-s\mathbf{\Omega},\mathbf{\Omega}) + \Sigma_t(\mathbf{r}-s\mathbf{\Omega})\varphi(\mathbf{r}-s\mathbf{\Omega},\mathbf{\Omega}) = \int_{S^2} d\mathbf{\Omega}'\Sigma_s(\mathbf{r}-s\mathbf{\Omega},\mathbf{\Omega}\leftarrow\mathbf{\Omega}')\varphi(\mathbf{r}-s\mathbf{\Omega},\mathbf{\Omega}') + q(\mathbf{r}-s\mathbf{\Omega},\mathbf{\Omega}). \quad (2.1)$$

As mentioned before, once the transport equation is written in its characteristic form, the partial differential operator $\mathbf{\Omega}\cdot\nabla$ becomes the total derivative $-\frac{d}{ds}$.

From the characteristic form of the transport equation, a solution to equation (2.1) can be found by integrating the equation by use of an integrating factor. The optical path through a medium is

defined as

$$\tau(s) = \int_0^s ds' \Sigma_t(\mathbf{r} - s'\mathbf{\Omega}), \tag{2.2}$$

where Σ_t is the total cross section of the medium and the integral is along the neutron path. The left hand side of equation (2.1) can therefore be written as a single term using

$$-\frac{d}{ds}\varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega})e^{-\tau(s)} = -e^{-\tau(s)}\frac{d}{ds}\varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}) + \varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega})e^{-\tau(s)}\frac{d}{ds}\tau(s). \tag{2.3}$$

By using the integrating factor $e^{-\tau(s)}$ to contract the left hand side of equation (2.1), the equation can be integrated to yield

$$-\int_0^\infty ds\frac{d}{ds}\varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega})e^{-\tau(s)} = \int_0^\infty dse^{-\tau(s)}\int_{S^2} d\mathbf{\Omega}'\Sigma_s(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega} \leftarrow \mathbf{\Omega}')\varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}') + \int_0^\infty dse^{-\tau(s)}q(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}). \tag{2.4}$$

When using the vacuum boundary condition, it is required that both the inward flux and the source tend to zero when the integration domain is extended to infinity. With this condition, equation (2.4) is reduced to

$$\varphi(\mathbf{r}, \mathbf{\Omega}) = \int_0^\infty dse^{-\tau(s)}\int_{S^2} d\mathbf{\Omega}'\Sigma_s(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega} \leftarrow \mathbf{\Omega}')\varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}') + \int_0^\infty dse^{-\tau(s)}q(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}). \tag{2.5}$$

Equation (2.5) is the integral form of the neutron transport equation with the vacuum boundary condition applied. This equation is used as the theoretical basis of this chapter. From equation (2.5), the operator form of the transport equation will then be derived along with its iterative solution.

2.2 The integral transport equation in operator form

In order to combat the lattice ray effects which is caused by restricting the neutron movement to lattice directions, the first collision source method is used. An initial neutron distribution is calculated throughout the system, using the first collision source method, to give a more accurate starting point for the rest of the simulation.

Using equation (2.5), the neutron flux at a given point $\mathbf{r}$, can be written as an expansion of the first collision source and the scattering and transport operators [43]. Equation (2.5) can be written in operator form by defining the scattering and transport operators as

$$S\varphi(\mathbf{r}, \mathbf{\Omega}) := \int_{S^2} d\mathbf{\Omega}'\Sigma_s(\mathbf{r}, \mathbf{\Omega} \leftarrow \mathbf{\Omega}')\varphi(\mathbf{r}, \mathbf{\Omega}') \tag{2.6}$$

and

$$T\varphi(\mathbf{r}, \mathbf{\Omega}) := \int_0^\infty dse^{-\tau(s)}\varphi(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}). \tag{2.7}$$

With the definitions above, equation (2.5) becomes

$$\varphi = TS\varphi + Tq, \tag{2.8}$$

which is solved iteratively by

$$\varphi^{[i+1]} = TS\varphi^{[i]} + Tq \; for \; i \in \mathbb{N}, \tag{2.9}$$

with iteration index i and the starting point for the iteration

$$\varphi^{[0]} = Tq. \tag{2.10}$$

The first collision source is the neutron flux at a point $\mathbf{r}$ as a result of neutrons emitted by the source q. In equation (2.9), it is represented by Tq – the transport operator applied to the source. In the calculation of the first collision source the out-scattering is explicitly taken into account along the integration lines. The term *first collision source* refers to the fact that it will serve as the scattering source for the first iteration step after the scattering operator has been applied.

2.3 The scattering model

Before deriving the discretized equations from the continuous transport equation, the scattering model used in the rest of this work will be presented here. Throughout the theoretical description it is assumed that both scattering and all neutron sources are isotropic. This means that the scattering cross section is independent of both the incoming and outgoing angle. The angular dependence of $\Sigma_s(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega} \leftarrow \mathbf{\Omega}')$ is removed and the scattering operator S becomes

$$S\varphi(\mathbf{r}, \mathbf{\Omega}) = \frac{\Sigma_s(\mathbf{r})}{4\pi} \int_{S^2} d\mathbf{\Omega}' \varphi(\mathbf{r}, \mathbf{\Omega}') = \frac{\Sigma_s(\mathbf{r})}{4\pi} \phi(\mathbf{r}). \tag{2.11}$$

In equation (2.11) $\phi(\mathbf{r})$ is the scalar flux at $\mathbf{r}$, defined as the integral of the angular flux over the surface of the sphere S^2. By assuming isotropic scattering, it is assumed that after a collision the incoming flux is equally distributed to all directions, thus, the resulting angular flux is only dependent on the scattering rate.

For the sources present in the domain, the same model is used, but as before, anisotropic sources can be included in the LBM. With the restriction to isotropic sources the angular dependence of the source $q(\mathbf{r}, \mathbf{\Omega})$ is also removed. By using an isotropic model for both the scattering and source, the terms in equation (2.9) become

$$\begin{aligned} TS\varphi^{[i]} = \int_0^\infty ds e^{-\tau(s)} \frac{\Sigma_s(\mathbf{r} - s\mathbf{\Omega})}{4\pi} \int_{S^2} d\mathbf{\Omega}' \varphi^{[i]}(\mathbf{r} - s\mathbf{\Omega}, \mathbf{\Omega}') \\ = \int_0^\infty ds e^{-\tau(s)} \frac{\Sigma_s(\mathbf{r} - s\mathbf{\Omega})}{4\pi} \phi^{[i]}(\mathbf{r} - s\mathbf{\Omega}) \end{aligned} \tag{2.12}$$

and

$$Tq = \int_0^\infty ds e^{-\tau(s)} \frac{q(\mathbf{r} - s\mathbf{\Omega})}{4\pi}, \tag{2.13}$$

respectively.

2.4 The discretized equations

One of the most important properties of the LBM is the simultaneous discretization of the full phase space of the problem. This means that the angular discretization is determined by the spatial discretization, restricting neutrons to move along lattice directions from node to node. If the position of a lattice node is given by $\mathbf{r}_\nu$, then the neighbouring node position along a given direction is given by

$$\mathbf{r}_{\nu'} = \mathbf{r}_\nu + s_d \mathbf{\Omega}_d, \tag{2.14}$$

where $\mathbf{\Omega}_d$ is a lattice direction and s_d is the spacing between nodes for a given direction. Thus far there is no restriction placed on the type of lattice (cubic, hexagonal, etc.), but for the rest of the development only a cubic lattice will be considered.

Each lattice node corresponds to a point where the neutron flux will be calculated. The size of the lattice determines the number of unknowns, and as the lattice size increases so does the number of unknowns. All interior nodes of the lattice are surrounded by 26 neighbouring nodes, meaning, without angular refinement, there are 26 lattice directions connecting each node with its nearest neighbours. These directions correspond to the number of angular degrees of freedom.

The distance between nodes is directionally dependent. For instance, if the lattice spacing along one of the Cartesian axes is s_d, then the distance between neighbouring nodes along the direction $\mathbf{\Omega}_d = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$ would be $\sqrt{3}s_d$. (See Figure 2.1 for an illustration of a unit cell of a cubic lattice. The figure shows a node and its 26 nearest neighbours, defining the base lattice directions.)

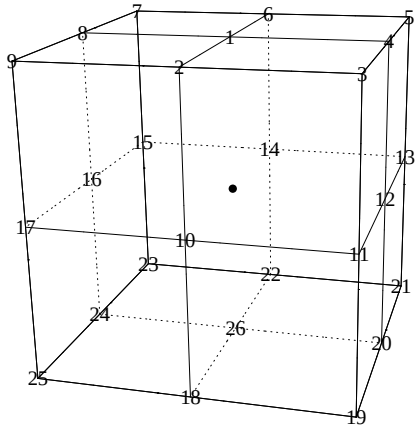


Figure 2.1: A cubic lattice unit cell showing the three different types of lattice directions.

2.4.1 The discretized TS operator

Using the cubic lattice described above, the TS operator in equation (2.9) can be discretized. From this point forward ν will be used to index the node position $\mathbf{r}_\nu$, and d will index the direction $\mathbf{\Omega}_d$. Between the nodes an interpolating function is used to calculate the neutron flux and account for the fact that flux values are only stored on lattice points. The TS operator then becomes

$$TS\varphi^{[i]}_{\nu,d} = \int_0^\infty ds e^{-\tau(s)} \mathcal{I}(S\varphi_{\nu,d}, S\varphi_{\nu',d}, s), \tag{2.15}$$

where $\varphi_{\nu,d}^{[i]}$ is the angular flux (in direction $\mathbf{\Omega}_d$) at node ν and at iteration step i, and $\mathcal{I}(S\varphi_{\nu,d}, S\varphi_{\nu',d}, s)$ is the interpolating function used between the nodes. The flux value $\varphi_{\nu',d}$, is the angular flux at position $\mathbf{r}_{\nu'} = \mathbf{r}_\nu - s\mathbf{\Omega}_d$ in direction $\mathbf{\Omega}_d$. The integral over the s variable can be broken up into a sum of integrals between the nodes, then equation (2.15) is changed to

$$TS\varphi_{\nu,d}^{[i]} = \sum_j \int_{js_d}^{(j+1)s_d} ds e^{-\tau_j(s)} \mathcal{I}(S\varphi_{\nu_j,d}, S\varphi_{\nu_{j+1},d}, s). \tag{2.16}$$

In the equations above $\mathcal{I}(S\varphi_{\nu_j,d}, S\varphi_{\nu_{j+i},d}, s)$ represents any interpolating function between the lattice nodes at position $\mathbf{r}_{\nu_j} = \mathbf{r}_\nu - js_d\mathbf{\Omega}_d$ and $\mathbf{r}_{\nu_{j+1}} = \mathbf{r}_\nu - (j+1)s_d\mathbf{\Omega}_d$. The interpolation between nodes is done by using the angular flux at the nodes after the scattering operator has been applied. This ensures that if the scattering model is changed, the streaming part of the algorithm will remain unaffected.

The optical path $\tau_j(s)$ in equation (2.16) is the integral of the total cross section along direction $\mathbf{\Omega}_d$ from node ν to s. Breaking the integral contained in $\tau_j(s)$ into a sum of integrals of the total cross section over the segments between successive nodes, gives

$$\tau_j(s) = \sum_{k=0}^{j-1} \int_{ks_d}^{(k+1)s_d} ds' \Sigma_t(\mathbf{r}_\nu - s'\mathbf{\Omega}_d) + \int_{js_d}^{s} ds' \Sigma_t(\mathbf{r}_\nu - s'\mathbf{\Omega}_d). \tag{2.17}$$

The first term is simply the exponent used for attenuation between $\mathbf{r}_\nu$ and $\mathbf{r}_\nu - js_d\mathbf{\Omega}_d$, while the second term takes attenuation between nodes $\mathbf{r}_\nu - js_d\mathbf{\Omega}_d$ and $\mathbf{r}_\nu - (j+1)s_d\mathbf{\Omega}_d$ into account.

2.4.2 Mesh refinement

Mesh or lattice refinement is done by halving the lattice spacing in each of the coordinate directions, and adding the intermediate points to the lattice. Flux values are calculated explicitly at the added node positions instead of using interpolation. However, this causes a cubic growth rate for the size of the lattice (in 3D), which drastically increases computer storage requirements.

Refining the mesh leads to a better spatial approximation of the flux, even though the interpolating function used between nodes may be of a low order. Successive mesh refinements will eventually lead to a very dense lattice where the interpolation will not play an important role. In the limit a spatially continuous equation is recovered for each angle. To recover the original equation, it is required to take the angular limit as well by refining the angular discretization.

2.4.3 Angular discretization

When using a cubic lattice, the directions at each node are defined on the surface of a unit hexahedron H, that surrounds the node. To be able to calculate the scalar flux it is required to integrate the angular flux, which is defined on the surface of the sphere S^2. In order to use the angular flux values along the directions defined on H a mapping $M : H \to S^2$ must be used [45]. The mapping from H to S^2 is defined by

$$M(\mathbf{h}) = \mathbf{h}/||\mathbf{h}|| = \mathbf{s} \text{ with } \mathbf{h} \in H \text{ and } \mathbf{s} \in S^2. \tag{2.18}$$

The mapping $M : H \to S^2$ is thus defined as the normalisation of any vector $\mathbf{h}$ defined on H so that the vector $\mathbf{s}$ points in the same direction as $\mathbf{h}$ but is defined on the unit sphere. Any function F defined on H is equivalent to the function f defined on S^2 through $f(\mathbf{s}) = F(M^{-1}(\mathbf{s})) = F(\mathbf{h})$ [45]. Through this mapping the angular flux φ defined on S^2 is related to the angular flux defined on H. Figure 2.2 shows a graphical representation of the mapping M.

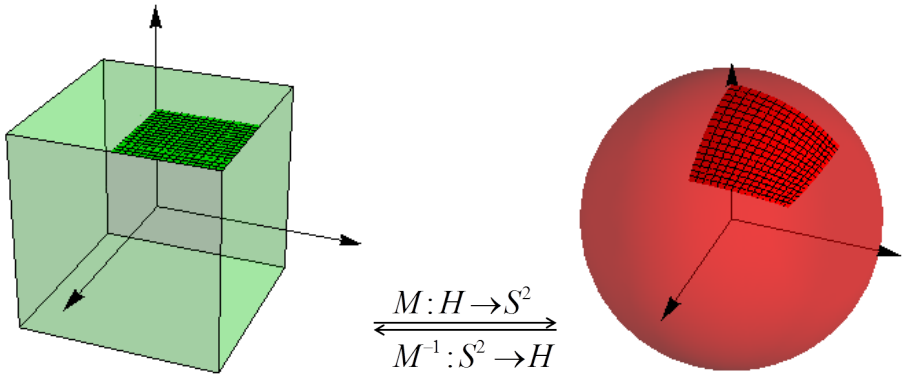


Figure 2.2: Graphical representation of the mapping between H and S^2.

Calculating the scalar flux means integrating the angular flux at a node, and because the angular flux values are known only at the points corresponding to the lattice directions, this must be done with a quadrature formula.

The angular integral for the scalar flux is approximated by a sum over the lattice angles as follows

$$\phi_\nu = \int_{S^2} d\mathbf{\Omega} \varphi_\nu(\mathbf{\Omega}) \approx \sum_d w_d \varphi_{\nu,d}, \tag{2.19}$$

where $\varphi_{\nu,d}$ are the angular flux values at the quadrature points, and w_d are the corresponding quadrature weights. As a first step to determine the quadrature weights, the angular flux at a node ν is expanded in terms of basis functions as

$$\varphi_\nu(\mathbf{\Omega}) = \sum_d \varphi_{\nu,d} \mathcal{B}_d(\mathbf{\Omega}), \tag{2.20}$$

where the coefficients $\varphi_{\nu,d}$ are defined as

$$\varphi_{\nu,d} = \int_{S^2} d\mathbf{\Omega} \mathcal{B}_d^*(\mathbf{\Omega}) \varphi_\nu(\mathbf{\Omega}). \tag{2.21}$$

Here, $\mathcal{B}_d(\mathbf{\Omega})$ are basis functions defined on S^2 with $\mathcal{B}_d^*(\mathbf{\Omega})$ dual to $\mathcal{B}_d(\mathbf{\Omega})$ and it is required that $\int_{S^2} d\mathbf{\Omega} \mathcal{B}_d^*(\mathbf{\Omega}) \mathcal{B}_d(\mathbf{\Omega}) = \delta_{d,d'}$, where $\delta_{d,d'}$ is the Kronecker delta. In the expansion of the angular flux in terms of these basis functions, d does not denote higher moments of the basis functions as is the case when using the spherical harmonics. It rather refers to basis functions with local support defined as a subset of S^2. The subscript d then denotes the direction on which the support of a specific basis function is centred. Choosing basis functions with local support on S^2 instead of globally supported basis functions, limits the amount by which the angular flux is dispersed over the sphere.

Because the basis functions $\mathcal{B}_d(\mathbf{\Omega})$ are chosen to have local support centred on $\mathbf{\Omega}_d$, the integration domain (S^2) is broken up into the supports of the basis functions. Each quadrature weight

corresponding to a direction is equal to the integral of the basis function, with support centred on that specific direction. This subdivision gives the relation between the quadrature weight and basis function associated with a single direction

$$w_d = \int_{\mathcal{A}_{\mathcal{B}_d}} d\mathbf{\Omega}\mathcal{B}_d(\mathbf{\Omega}), \tag{2.22}$$

where the integration range $\mathcal{A}_{\mathcal{B}_d} \subset S^2$ is the support of basis function $\mathcal{B}_d(\mathbf{\Omega})$ on the sphere.

On H there are a number of equivalent points/directions, such as the corners, face- and edge centres. The integrals of basis functions corresponding to equivalent directions are equal, and thus the weights corresponding to equivalent directions are equal as well. By using this symmetry in the case where no angular refinement is used, only three quadrature weights need to be calculated. It is required that the weights preserve averages and so the requirement on both the weights and basis functions is

$$\sum_d w_d = \sum_d \int_{\mathcal{A}_{\mathcal{B}_d}} d\mathbf{\Omega}\mathcal{B}_d(\mathbf{\Omega}) = 4\pi. \tag{2.23}$$

Basis functions that are used to represent the angular flux can be chosen arbitrarily, as long as they satisfy equation (2.23). In this study, the linear splines are chosen as the basis functions to represent the angular flux with corresponding dual $\mathcal{B}_d^*(\mathbf{\Omega}) := \delta^2(\mathbf{\Omega} - \mathbf{\Omega}_d)$. The linear splines satisfy the requirement of local support as well as the requirement to preserve averages given in equation (2.23).

In the definition of the dual $\mathcal{B}_d^*(\mathbf{\Omega})$, $\delta^2(\mathbf{\Omega} - \mathbf{\Omega}_d)$ represents the Dirac delta function defined on the surface of the sphere S^2. As with the regular Dirac delta function, the definition of $\delta^2(\mathbf{\Omega} - \mathbf{\Omega}_d)$ can be given in terms of its integral over the surface of the sphere as

$$\int_{S^2} d\mathbf{\Omega}\delta^2(\mathbf{\Omega} - \mathbf{\Omega}')f(\mathbf{\Omega}) = f(\mathbf{\Omega}'). \tag{2.24}$$

If s and t are used to parameterize the top face of H, then the linear spline centred on the positive z-axis on a unit cube is defined as

$$\mathcal{L}(s,t) = \begin{cases} (1-2s)(1-2t), & 0 \leq s \leq \frac{1}{2}, \quad 0 \leq t \leq \frac{1}{2}; \\ 2s(1-2t), & -\frac{1}{2} \leq s < 0, \quad 0 < t \leq \frac{1}{2}; \\ (1-2s)2t, & 0 < s \leq \frac{1}{2}, \quad -\frac{1}{2} \leq t < 0; \\ 4st, & -\frac{1}{2} \leq s < 0, \quad -\frac{1}{2} \leq t < 0; \\ 0, & \text{otherwise.} \end{cases} \tag{2.25}$$

Figure 2.3 shows a graphical representation of $\mathcal{L}(s,t)$ for $0 \leq s \leq 0.5$ and $0 \leq t \leq 0.5$ listed above (coloured orange), were the height above the surface is regarded as the function value for illustration purposes. The green square is a quarter segment of the support of the spline on the surface of H.

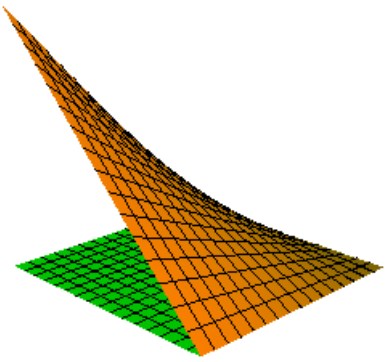


Figure 2.3: Linear spline (quarter segment) centred around the positive z-axis on H.

To calculate the angular integrals over S^2, the splines need to be projected to the sphere using the mapping M. The mapping of the splines and their supports from H to S^2 is shown in Figure 2.4. Green is used to indicate the support of the spline on H and red indicates the support of the spline on S^2.

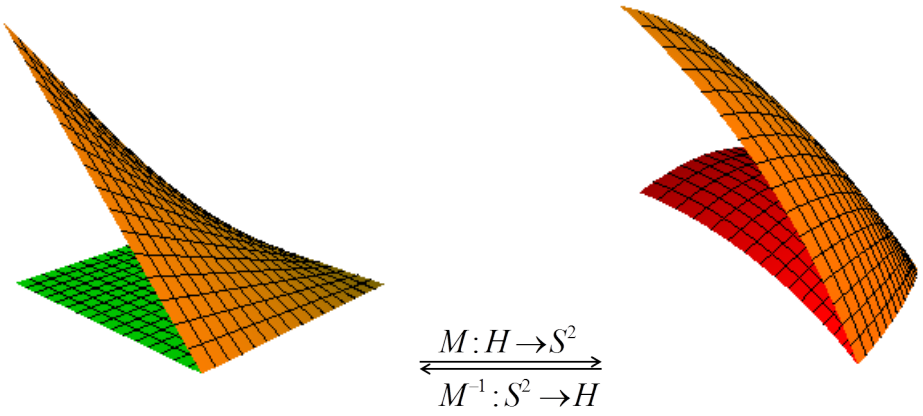


Figure 2.4: Mapping of the splines and their supports from H to S^2.

Evaluating the angular integrals over the S^2 are more cumbersome than calculating their values over H. This is because the splines have relatively simple parameterizations in Cartesian coordinates and the integration domains are squares. Calculating the angular integrals in Cartesian coordinates can be done using the Jacobian of the inverse transformation M^{-1}.

Using the (s,t) parameterization of the surface of H and parameterizing the polar and azimuth angles in terms of s and t, the Jacobian of the inverse transformation is calculated as

$$J(s,t) = \frac{2}{\sqrt{\frac{s^2+t^2}{1+4s^2+4t^2}}(1+4s^2+4t^2)^{\frac{3}{2}}}. \tag{2.26}$$

Using the inverse mapping with its Jacobian $J(s,t)$, transforms equation (2.19) to

$$\phi_\nu = \int_{S^2} d\mathbf{\Omega}\varphi_\nu(\mathbf{\Omega}) \approx \sum_d \varphi_{\nu,d} \iint dsdtJ(s,t)\mathcal{B}_d(s,t). \tag{2.27}$$

2.4.4 Angular refinement

In the case where only the nearest neighbours on the lattice are used to determine the angular discretization, the number of directions may not be enough to account for the angular dependence of the angular flux. This low level of angular discretization leads to incorrect scalar flux values being calculated when summing the contributions of the individual angular flux values.

The quadrature weights, corresponding to the different lattice directions, effectively represent an area on the sphere. If the angular flux has great variations over that area, a linear approximation is not enough to represent the angular dependence of the flux, and more directions are needed. Alternatively, higher order approximations, such as quadratic or cubic splines, may be used, but this poses the risk of negative angular flux values being calculated at nodes.

Using the linear splines ensures an inherently positive scheme, although more streaming directions are needed when better angular resolution is required. Angular refinement is required, for instance, in cases where nodes are located far from localized sources and the contribution from the directions connecting the node and the sources are over estimated.

The solid angle associated with each lattice direction spans a large area on the surface of the sphere when a low angular quadrature is used. When a node is located far from a source, the solid angle spanning the source is much smaller than the solid angle associated with any given direction. This difference in solid angle causes the contribution to the scalar flux value from that direction to be overestimated.

Overestimation of the flux is, however, not restricted only to lattice directions connecting a node and a source. In the case of low angular quadrature, all directions share the property of being associated with a solid angle spanning a large area on the sphere. The effect is more pronounced in the case where a lattice direction connects a node to a source, because the angular flux from that direction is much larger than the flux values of other directions.

An example (in 2D) of the difference between the solid angles spanning a lattice direction and the source is illustrated in Figure 2.5. The red cone represents the angular region spanning the source, while the green cone represents the region associated with the lattice direction drawn in black.

For nodes situated further away from the source, the effect of over estimation becomes more pronounced. The green cone Figure 2.5 shows the extent to which the angular flux contribution is over estimated. Angular refinement is required to mitigate this effect.

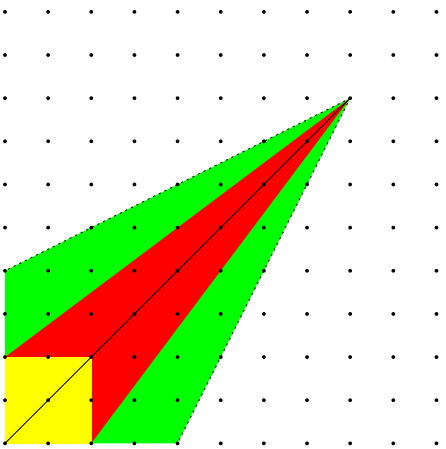


Figure 2.5: Overestimation of the angular flux value at a node (in 2D) as a result of a low angular quadrature.

Angular refinement is accomplished by considering directions to next-nearest neighbours, as additional angles along which to transport neutrons. Figure 2.6 shows how angular refinement is done on a 2D lattice. The first level of angular refinement adds the directions coloured in red.

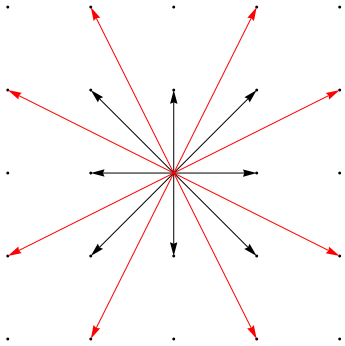


Figure 2.6: Angular refinement on a 2D lattice.

Figure 2.7 shows how the overestimation of the angular flux at a node is decreased as a result of angular refinement. Once again, the green cone represents the solid angle associated with the lattice direction, linking the node and the source.

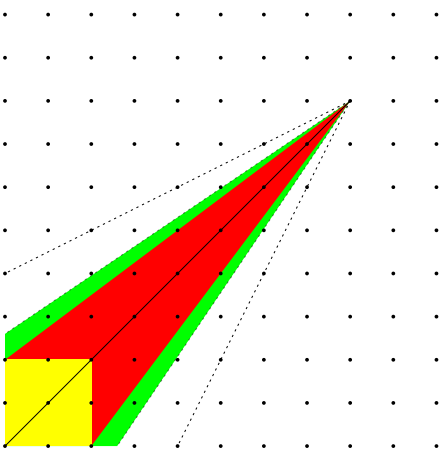


Figure 2.7: Decreased overestimation of the angular flux as a result of angular refinement.

Each lattice direction now represents a smaller area on the sphere (or cube) than before, and the weight for each direction needs to be recalculated. This can be seen when looking at how the support of the basis function, corresponding to a specific direction, decreases when higher levels of angular refinement are used. In Figure 2.8 it is shown how the support of the basis function associated with the positive z-axis decreases when the level of angular refinement is increased. The support of the basis function is shown in green on the surface of the cube.

Once again this subdivision on H generates a number of equivalents points/directions and only those weights need to be calculated. In 3D, the first level of angular refinement adds 72 new directions to the existing 26 making a total of 98 directions, and the second level of refinement has a total of 386 directions.

In general, the total number of directions for a given level of angular refinement is (see Appendix A for proof)

$$N_L = 8 + 12(2^{L+1} - 1) + 6(2^{L+1} - 1)^2 \text{ with } L = 0, 1, 2, ... \tag{2.28}$$

In equation (2.28), L is the level of angular refinement. Counting the number of directions is done by dividing the different directions into classes: those associated with corners, edges and faces. Each level of refinement has $2^{L+1} - 1$ directions associated with each of the 12 edges and $(2^{L+1} - 1)^2$ directions associated with each of the 6 faces. The corners of the cube are counted separately to avoid double counting.

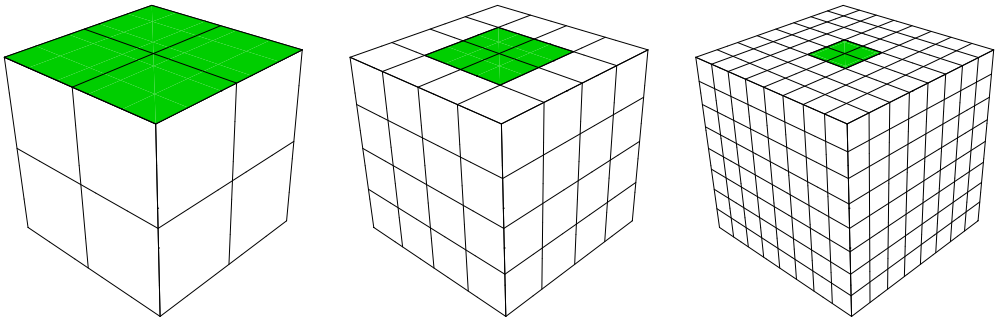


Figure 2.8: Basis function support and how it changes with different levels of angular refinement.

2.5 Calculating the first collision source distribution

Thus far, the focus was on discretizing the TS operator in equation (2.9) and calculating the scalar flux at each node after a scattering and streaming step in the iteration scheme. Before any of the iterations can be performed, the first collision source needs to be calculated, (see equation (2.9) and (2.10)). In this section, it will be illustrated how to calculate the first collision source at an arbitrary point as a result of a localized isotropic volumetric source.

Calculating the first collision source at a lattice node differs from the rest of the algorithm, because the integration in the transport operator is not performed along lattice directions only. A fine discretization of the angular variable at each node is used, not only to get an accurate approximation of the scalar flux as a result of the source, but also to eliminate blind spots caused by the lattice.

By using the first collision source method the initial blind spots on the lattice can be eliminated. Subsequent iterations will still suffer from the fact that only lattice directions are used to perform the transport of angular fluxes.

The same isotropic model used for the scattering will be used to treat the source. Let the set $\mathcal{A}$ be the support of the source and assume the source is defined as

$$q(\mathbf{r}, \mathbf{\Omega}) = \frac{q_s}{4\pi}\chi_{\mathcal{A}}(\mathbf{r}), \tag{2.29}$$

where q_s is the source strength and $\chi_{\mathcal{A}}$ is the indicator function defined as

$$\chi_{\mathcal{A}}(\mathbf{r}) = \begin{cases} 1, & \mathbf{r} \in \mathcal{A} \subset \mathbb{R}^3; \\ 0, & \text{otherwise.} \end{cases} \tag{2.30}$$

The angular flux Tq in equation (2.9) for a specific direction $\mathbf{\Omega}$ can be written as

$$\varphi(\mathbf{r}, \mathbf{\Omega}) = \int_0^\infty ds e^{-\tau(s)} \frac{q_s}{4\pi}\chi_{\mathcal{A}}(\mathbf{r} - s\mathbf{\Omega}), \tag{2.31}$$

with

$$\tau(s) = \int_0^s ds' \Sigma_t(\mathbf{r} - s'\mathbf{\Omega}) = \int_0^{l_r(\mathbf{\Omega})} ds' \Sigma_t(\mathbf{r} - s'\mathbf{\Omega}) + \int_{l_r(\mathbf{\Omega})}^s ds' \Sigma_t(\mathbf{r} - s'\mathbf{\Omega}). \tag{2.32}$$

As before, the first term is simply the attenuation between the source and the point $\mathbf{r}$, and the second term accounts for the attenuation inside the source region. Here, the integration along the $\mathbf{\Omega}$ is split into the region outside $\mathcal{A}$ (up to l_r) and inside $\mathcal{A}$ (from l_r to l_s). In the case where the cross sections in each region are constant, the first collision source at any point outside $\mathcal{A}$ in a specific direction $\mathbf{\Omega}$ is given by

$$\varphi(\mathbf{r}, \mathbf{\Omega}) = \frac{q_s}{4\pi} \frac{e^{-\Sigma_{t,l} l_r(\mathbf{\Omega})}}{\Sigma_{t,\mathcal{A}}} (1 - e^{-\Sigma_{t,\mathcal{A}}(l_s(\mathbf{\Omega}) - l_r(\mathbf{\Omega}))}), \tag{2.33}$$

and for any point inside $\mathcal{A}$ the angular flux is given by

$$\varphi(\mathbf{r}, \mathbf{\Omega}) = \frac{q_s}{4\pi} \frac{1}{\Sigma_{t,\mathcal{A}}} (1 - e^{-\Sigma_{t,\mathcal{A}} l_s(\mathbf{\Omega})}), \tag{2.34}$$

where $\Sigma_{t,\mathcal{A}}$ is the total cross section in $\mathcal{A}$, and $\Sigma_{t,l}$ is the total cross section outside $\mathcal{A}$. The distance inside the source region is given by $l_s - l_r$ and the distance outside the source region to point $\mathbf{r}$ is given by l_r. In the calculation of the first collision source, the accumulation of the source and attenuation are treated explicitly along the path of integration which is consistent with the rest of the LBM.

Finally, to calculate the scalar flux at a given point it only remains to perform the angular integration. The angular dependence of the first collision source is contained in the track lengths l_r and l_s, and the total cross section of each region through which the tracks pass. Once again the angular integral is approximated using the quadrature formula

$$\phi(\mathbf{r}) = \int_{S^2} d\mathbf{\Omega} \varphi(\mathbf{r}, \mathbf{\Omega}) \approx \sum_n w_n \varphi_n(\mathbf{r}), \tag{2.35}$$

where $\varphi_n(\mathbf{r})$ denotes the angular flux at point $\mathbf{r}$ evaluated at direction $\mathbf{\Omega}_n$.

Adding anisotropic sources will affect the way in which the angular integral for the scalar flux is calculated. Instead of the angular dependence residing only in the track lengths, the source strength

q_s will depend on $\mathbf{\Omega}$ and needs to be taken into account when evaluating the integral.

2.6 Spatial and angular interpolation

Flux values are only calculated at lattice nodes, and a scheme for approximating the flux values at off-lattice points is required to calculate integral quantities, such as the scalar flux in a region. To accomplish this, a spatial interpolant must be used to estimate the flux values from the given node values.

A cubic lattice consists of cells with nodes located at each of their corners. Volume integrals are calculated by using eight point interpolating functions in each cell of the lattice surrounded by eight nodes. The nodes used as corners for a cell differ depending on the direction being considered. That is, at a fixed point in space, many different nodes are used to determine each of the angular flux values.

To illustrate how the interpolation is done, consider the following example in 2D. The angular flux at an arbitrary point in direction $\mathbf{\Omega}_d$ needs to be calculated. When $\mathbf{\Omega}_d$ is a lattice direction, only spatial interpolation is required. Labelling of the lattice directions is done clockwise starting at the $\mathbf{\Omega}_1 = (0, 1)$, as illustrated in Figure 2.9.

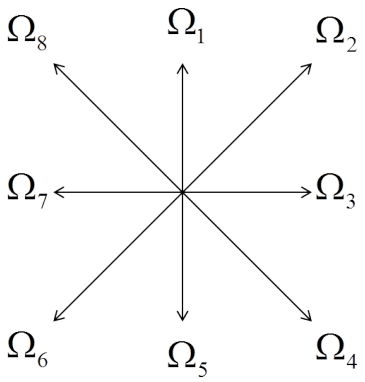
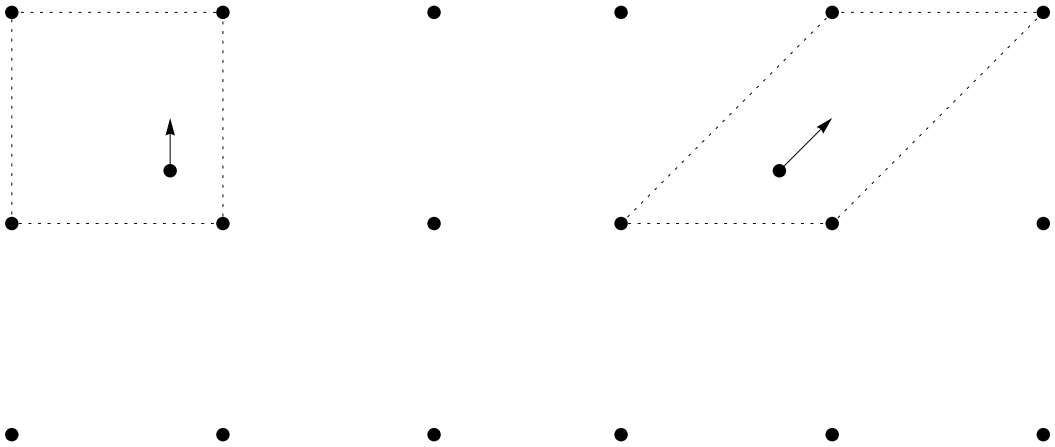


Figure 2.9: Numbering of lattice directions in 2D.

For an angle along any of the coordinate axes, the interpolation is done using the angular flux values (along that direction) at the nodes of the square surrounding the point. For any of the diagonal directions the nodes at the corners of the surrounding parallelogram are used. Figure 2.10(a) shows the cell used for interpolating at a point for direction $\mathbf{\Omega}_1$, and Figure 2.10(b) shows the cell used for interpolating at the same point for direction $\mathbf{\Omega}_2$.

In the case where $\mathbf{\Omega}_d$ is not a lattice direction, angular interpolation is needed as well. If the basis functions used for the angular flux expansion are interpolating functions, they can be used to determine the angular flux value. Otherwise, a suitable interpolant must be used. In this case, where the linear splines are used for the flux expansion, the basis functions can be used directly.



(a) Spatial interpolation along direction $\mathbf{\Omega}_1$. (b) Spatial interpolation along direction $\mathbf{\Omega}_2$.

Figure 2.10: Nodes used to determine the flux value at an arbitrary point for a given direction.

Consider the case depicted in Figure 2.11, where the angular flux needs to be determined along $\mathbf{\Omega}_d$ between $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$. At the point where the angular flux value is calculated, the values along $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ are determined according to the procedure described above. From these calculated flux values, the angular flux value along the off-lattice direction $\mathbf{\Omega}_d$, is determined by interpolating between the values.

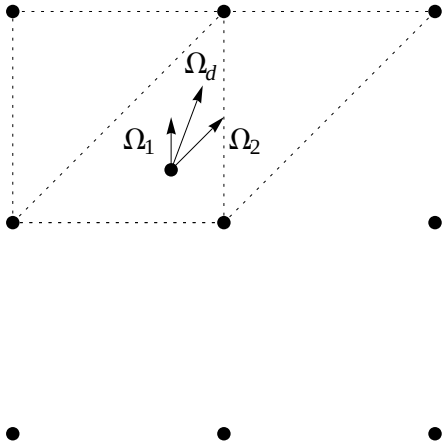


Figure 2.11: Interpolation of the angular flux value along an off-lattice direction $\mathbf{\Omega}_d$.

Higher order interpolants may be used to calculate the volume integrals and can extend beyond a single unit cell of the lattice. Although, for a consistent approach throughout the algorithm, the same order interpolants, as used during the streaming of fluxes between nodes should be used when calculating the volume integrals.

2.7 Concluding remarks

The basic theory of the LBM, on which the algorithm for calculating the neutron flux is based, was presented in this chapter. Discretizing the transport equation in terms of a lattice is the main feature of the LBM. Confining the scattering to lattice nodes is one of the main differences between the LBM and other methods currently used for neutron transport calculations.

However, some similarities do exist between the LBM and other methods, such as the MOC and the S_N method, in that a finite number of angles are used along which neutron transport is performed. Although the lattice used for the discretization may seem restrictive, it allows for detailed scattering models to be used at lattice nodes.

Drawbacks of using a lattice, such as the limited number of directions, are overcome by using properties of the lattice itself. More angles are added by using the lattice directions connecting next-nearest neighbours. Spatial refinement on a lattice is then accomplished by halving the lattice spacing and adding the intermediate nodes.

The particle based nature of the LBM makes the method intuitive and easy to understand. Separation of the algorithm into streaming and scattering parts makes it flexible in terms of adding to, or extending the order of the various approximations. Extensions to the LBM, such as anisotropic scattering, higher order interpolants between nodes, and higher order basis functions to represent the angular flux, can be added.

Chapter 3

The calculational algorithm

Now that the theory of the Lattice Boltzmann method (LBM) for neutron transport has been established, it can subsequently be used to describe how the algorithm is implemented on a computer. The implementation of all the different components that are needed for the algorithm, come directly from the derived equations in Chapter 2. In this chapter, a full description of the calculational algorithm, along with the required data structures, are given.

This chapter starts by showing how the first collision source, as described at the end of the previous chapter, is distributed to the lattice. A simple parallel implementation for calculating the first collision source is used to lower the calculational time. Calculating the first collision source is therefore the first step of the iteration scheme in calculating the neutron distribution on the lattice.

3.1 The first collision source calculation

In Chapter 2, equation (2.35) gives the scalar flux at any point in space as a result of the source(s) present in the problem that is being solved. The quadrature points used for the numerical integration of the angular integral, correspond to directions on the surface of the unit sphere surrounding that point. By restricting the calculation of the flux values to a lattice, the general position vector in equation (2.35) is replaced by the position vector of each of the lattice nodes, transforming equation (2.35) to

$$\phi_\nu = \int_{S^2} d\mathbf{\Omega}\varphi(\mathbf{r}_\nu, \mathbf{\Omega}) \approx \sum_n w_n \varphi_{\nu,n}. \tag{3.1}$$

In equation (3.1), $\phi_\nu = \phi(\mathbf{r}_\nu)$ is the scalar flux at the lattice node ν and $\varphi_{\nu,n}$ is the angular flux at lattice node ν in direction n given by

$$\varphi_{\nu,n} = \varphi(\mathbf{r}_\nu, \mathbf{\Omega}_n) = \int_0^\infty ds e^{-\tau(s)} \frac{q_s}{4\pi} \chi_{\mathcal{A}}(\mathbf{r} - s\mathbf{\Omega}_n), \tag{3.2}$$

from equation (2.31). Using this quadrature formula, the first collision source is calculated at each of the lattice nodes ν.

The calculated angular flux values are not stored, but they are multiplied by the quadrature weight and added to the scalar flux directly. Only the scalar flux value is required when isotropic scattering is considered, as is the case here.

When calculating the first collision source at a node as a result of localized sources, it is not necessary to evaluate the angular integral over the entire surface of the sphere. Many of the integration lines, along the directions used as quadrature points in equation (3.1), may not intersect the source(s). Lines along directions not intersecting the source results in an angular flux value of zero, which will not contribute to the angular integral.

At a node outside a source region, the integration lines that intersect the source are limited to the solid angle spanning the source, from the point of view of that node. Only the angular flux from the directions contained in that solid angle contribute to the angular integral. For nodes inside a source region, the entire sphere must be discretized to accurately calculate the first collision source, as all directions will contribute to the sum for the scalar flux.

However, instead of discretizing the entire sphere when considering a node outside the source regions, a bounding box for each source can be used to break up the integration domain into subsets of S^2. These subsets are the solid angles that contain directions along which integration lines intersect the source(s). Using bounding boxes greatly reduces the number of directions required to accurately approximate the first collision source at a node.

The bounding box for a source is the rectangle enclosing it when the source is projected onto a plane between the source and the node. Each bounding box corresponds to a solid angle on the unit sphere around the node in question. Once the solid angle has been determined, it is discretized by creating a grid of points, each corresponding to a quadrature point for evaluating the angular integral in equation (3.1). The bounding box that encloses the source differs from node to node, meaning that the quadrature set used for the angular integration is determined per node.

Each of the quadrature points on the sphere is used to create an integration line from the lattice node to the source. The integration lines are used to calculate the track lengths within the different regions which they intersect. Track lengths are used to calculate the accumulated flux inside the source and the attenuation along integration lines between the node and source. Along each of the integration lines, the accumulation of the source and the absorption are treated explicitly as in equation (3.2).

The plane onto which the source is projected, is chosen so that the normal of the plane is parallel to the vector connecting the lattice node and the centre of the source. Choosing the projection plane in an optimal way minimizes the solid angle on the sphere needed to cover the source. This limits the number of quadrature points needed to accurately calculate the first collision source. For the purpose of calculating the first collision source in this study, the projection plane is chosen so that it contains the corner(s) closest to the lattice node in question. After the source has been projected onto the plane, the bounding box for the source is then determined.

Using the bounding box, the maximum and minimum values of θ and ϕ that span the source are calculated . Here, θ and ϕ are the polar and azimuth angles with respect to the node where the first collision source is calculated. These maximum and minimum values define the integration ranges for the angular integral, or solid angle in which the grid of quadrature points are generated to, perform the numerical integration.

A midpoint integration rule is used in the current implementation to evaluate the angular integral. The integration ranges $[\theta_{\min}, \theta_{\max}]$ and $[\phi_{\min}, \phi_{\max}]$ are divided into N pieces, respectively, creating

a grid of squares on the surface of the sphere each with an area $\Delta\theta\Delta\phi$ where

$$\Delta\theta = \frac{\theta_{\max} - \theta_{\min}}{N} \text{ and } \Delta\phi = \frac{\phi_{\max} - \phi_{\min}}{N}. \tag{3.3}$$

The midpoint of each square is used as the quadrature point where the angular flux value $\varphi_{\nu,n}$ is calculated according to equation (3.2). Using the definitions of $\Delta\theta$ and $\Delta\phi$ above, $\varphi_{\nu,n}$ is defined as

$$\varphi_{\nu,n} = \varphi\left(\mathbf{r}, \theta_{\min} + \frac{(2n+1)\Delta\theta}{2}, \phi_{\min} + \frac{(2n+1)\Delta\phi}{2}\right) \text{ with } n = 0, 1, ..., N-1. \tag{3.4}$$

In equation (3.1) the quadrature weights are all equal to $\Delta\theta\Delta\phi$ – the area of each square.

As illustration, consider a square source centred at the origin with side lengths a in 2D, and a lattice node outside the source region at $(1.5a, 1.5a)$. In 2D, the source is projected to a line instead of a plane. The equation for the line onto which the source is projected is given by

$$\mathbf{n} \cdot \mathbf{x} + d = -\frac{x}{\sqrt{2}} - \frac{y}{\sqrt{2}} + d = 0, \tag{3.5}$$

where d is determined by substituting the coordinates of the corner(s) closest to the node into the equation. The normal of the line is given by the normalized vector $\mathbf{n} = (-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$, and the position vector of a point on the line is $\mathbf{x} = (x, y)$. In this case, the coordinates of the closest corner to the node are $(\frac{a}{2}, \frac{a}{2})$, and gives the equation

$$-\frac{x}{\sqrt{2}} - \frac{y}{\sqrt{2}} + \sqrt{2}a = 0, \tag{3.6}$$

or

$$y = 2a - x. \tag{3.7}$$

Once the equation for the line has been determined, each of the corners of the source is projected onto the line, after which the bounding box for the source is determined. Figure 3.1 shows the 2D example that is described above. In the figure the source is the red square centred on the origin, and the solid angle spanning the source is shown on the circle around the node.

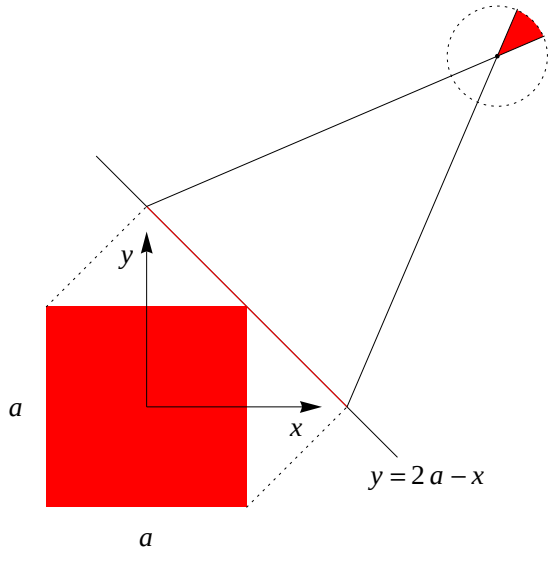


Figure 3.1: Projection of a source onto a line (in 2D) and the solid angle used to calculate the first collision source.

Note that the solid angle used for the integration is on the opposite side of the node, with respect to the source. This is because the angular flux values in those directions need to be calculated and added to the scalar flux. By defining the transport operator as in equation (2.7), the integration in equation (3.2) is performed along the lines $\mathbf{r}_\nu - s\mathbf{\Omega}_n$ with $n = 1, 2, ..., N$. For each direction $\mathbf{\Omega}_n$, the integration line is the line extending from $\mathbf{r}_\nu$ in the direction opposite to $\mathbf{\Omega}_n$.

Calculating the first collision source at each node can be done independently. Moreover, each angular flux value used to perform the numerical integration in equation (3.1) can be calculated independently, as only the attenuation (absorption and out-scattering) is taken into account when calculating the first collision source.

In the current implementation of the algorithm, the first collision source calculation is parallelized using the Compute Unified Device Architecture (CUDA) parallel computing platform and programming model for graphics processing units developed by NVIDIA [46]. For each node in the lattice, the first collision source is calculated in parallel, but the calculation of the angular flux values at quadrature points is still done in serial. Parallelizing the first collision source calculation greatly reduces the overall calculational time needed to solve the transport equation.

Apart from the need to minimize the calculational time, it is also necessary to accurately calculate the first collision source on all of the lattice nodes as it is used repeatedly throughout the algorithm. An error made while performing the first collision source calculation will consequently transfer directly to the final solution of the scalar flux at each node.

3.2 The lattice structure

Conceptually, the lattice that is used for the spatial discretization of the transport equation, is a container of nodes. It is implemented as a multi-dimensional array with each element therein corresponding to a calculational node. Apart from the nodes, the lattice contains data itself, such as the number of nodes in each dimension and the lattice spacing along the coordinate axes. The number of nodes in each dimension is used to do range checking, while the spacing provides the lengths needed for the attenuation and interpolation between nodes.

For a given lattice spacing, the number of nodes in each dimension is calculated by using a bounding box for the problem being solved. In each dimension, the extent of the bounding box is divided by the given spacing, whereby the number of nodes is equal to the integer larger or equal to this number plus one. The extra node is added because the node indices in each dimension start from 0, and one extra node is required to ensure the lattice covers the entire geometry of the problem.

In the computer code, this is accomplished by using the mathematical ceiling function as follows:

$$N_d = \text{ceiling}\left(\frac{\text{extent in direction } d}{\text{spacing}}\right) + 1, \tag{3.8}$$

where d represents the coordinate, e.g. x, y or z, when using Cartesian geometry in 3D. The ceiling function returns the integer larger or equal to the expression in brackets. A multi-dimensional array consisting of $N_x \times N_y \times N_z$ nodes (in 3D) is then created.

Each node in the lattice is a data container that is used to store information needed throughout the solution algorithm. At each node in the lattice the scattering cross section, scalar flux and first

collision source values are stored. Node coordinates are linked to their indices via the lattice spacing to avoid additional storage. The absorption cross sections per direction is not stored at each node, as is discussed later in the extensions to the algorithm.

Two scalar flux values are stored at each node, corresponding to the current and previous iteration. This is done to be able to compare the scalar flux values of successive iterations to determine if convergence has been reached.

3.3 The iteration scheme

The iteration scheme follows three steps: scattering, streaming and convergence testing. During each iteration, the scattering operator is applied to the scalar flux of the previous iteration, in order to calculate the angular flux that is used during the streaming step. The newly calculated angular flux values are then streamed through the lattice, and the scalar flux values for the current iteration are calculated at each node.

For convergence testing, the scalar flux value of the current iteration is compared to the scalar flux from the previous iteration, and the iteration process is stopped once the difference between the scalar flux values of successive iterations is less than the set tolerance. The following three sections describe how each of the steps is performed during every iteration.

3.3.1 Scattering

The starting point for the iteration scheme is the first collision source at each node in the lattice, as given by equation (2.10). From the first collision source, the angular flux at every node in the lattice is calculated by applying the scattering operator. Angular flux values are the required quantities that are used during the streaming step of the algorithm. In the case where isotropic scattering is used, only the scalar flux is needed to calculate the angular flux value, as can be seen from equation (2.11):

$$S\varphi(\mathbf{r},\mathbf{\Omega}) = \frac{\Sigma_s(\mathbf{r})}{4\pi}\int_{S^2} d\mathbf{\Omega}'\varphi(\mathbf{r},\mathbf{\Omega}') = \frac{\Sigma_s(\mathbf{r})}{4\pi}\phi(\mathbf{r}). \tag{3.9}$$

This means that the angular flux values at a node are equal in all directions after scattering, if an isotropic scattering model is used.

A problem encountered with the LBM, however, is how to treat material interfaces and the boundary of the problem. Unlike other methods where the problem domain is discretized using volumes and the flux values calculated in the interior of the volumes, the LBM uses points and lines. This poses a problem if nodes and lattice directions are on the boundary or material interfaces, where discontinuities in the cross sections exist between the different material zones. These discontinuities need to be treated to determine the correct flux behaviour during the streaming and scattering steps.

Consider an infinitesimal circular area surrounding a node on a material interface or boundary in 2D, as illustrated in Figure 3.2. This concept is extended to 3D by considering an infinitesimal volume instead of an area. The average reaction rate in the area is equal to

$$R = \frac{1}{A}\int_A dA\frac{\Sigma_s}{4\pi}\phi(x,y). \tag{3.10}$$

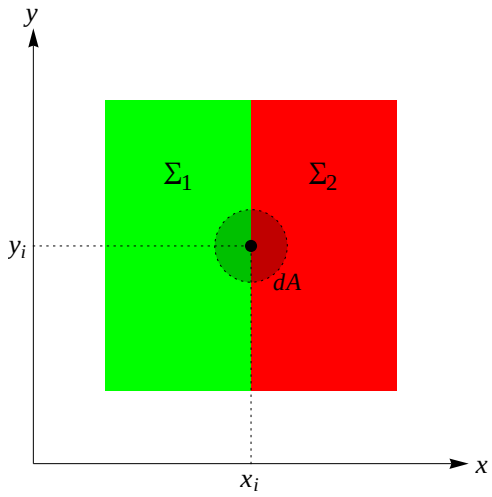


Figure 3.2: A node located on a material interface with an infinitesimal area dA surrounding it.

Assuming that the chosen area is small enough that the flux can be approximated with a constant value, equal to the flux at (x_i, y_i), the reaction rate becomes

$$R = \frac{1}{2A}\frac{\Sigma_{1,s}}{4\pi}\phi(x_i, y_i)\int dA + \frac{1}{2A}\frac{\Sigma_{2,s}}{4\pi}\phi(x_i, y_i)\int dA, \tag{3.11}$$

where the integral is split into two terms accounting for the different cross sections. In order to predict the correct flux behaviour at the lattice point, the limit $r \to 0$ must be evaluated, where r is the radius of the circular area. Evaluating the integrals in equation (3.11) and taking the limit yields

$$R = \frac{\Sigma_{1,s}}{4\pi}\frac{\phi(x_i, y_i)}{2} + \frac{\Sigma_{2,s}}{4\pi}\frac{\phi(x_i, y_i)}{2}. \tag{3.12}$$

This means that half the flux at (x_i, y_i) is used to calculate the scattered flux in each of the regions, which is then summed to find the total scattered/angular flux. Similarly, when streaming along a lattice direction on the interface, half the flux is attenuated by using the cross section to the left and the other half is attenuated by using the cross section to the right of the interface. The resulting flux at the next node is then the sum of the two separately attenuated flux values.

When anisotropic scattering is used, higher order angular flux moments have to be stored at nodes. This will increase storage requirements, depending on the order of anisotropy or number of flux moments stored. Scattering cross sections are usually tabulated in terms of Legendre moments, making the spherical harmonics the natural choice for the angular flux expansion when anisotropic scattering is considered.

Expanding the flux in terms of the spherical harmonics leads to better data compression and limits the amount by which the data storage will increase. Although an increase in the required storage is limited by using the spherical harmonics, increased angular dispersion of the flux is introduced. Storing higher order moments can therefore be used to recover some of the angular resolution.

Calculating the expansion coefficients are subsequently done by using the discrete lattice directions as quadrature points, as before. The quadrature weights need to be recalculated when using the spherical harmonics, as they are related to the basis functions used for the flux expansion. Weights are determined by requiring that the quadrature formula integrates the spherical harmonics exactly up to a given order.

As an alternative to the spherical harmonics, wavelets can be used to represent the angular flux at nodes. Interested readers are referred to [45] for a description of wavelets and their application to neutron transport.

3.3.2 Streaming

Once the angular flux has been calculated at each node, it can be streamed through the lattice according to equation (2.16). Equation (2.16) gives the discrete form of the transport operator that is implemented in the streaming step of the algorithm.

When streaming between nodes, flux is accumulated and attenuated along each of the lattice directions. Flux accumulated between a pair of nodes is attenuated along the streaming direction between the next pair of nodes. This means that the flux contributions to different nodes along a streaming direction is just the accumulated flux attenuated over the streaming distance. For each neighbouring node in the streaming direction, the accumulated flux is attenuated over the distance between the nodes, and need not be recalculated but can simply be multiplied by an exponential attenuation factor.

To avoid recalculating the flux contributions from nodes in a specific direction, equation (2.16) is written as a recursive relation. A running total of the accumulated and attenuated flux is kept, which prevents the unnecessary recalculation of the streamed flux.

For example, consider nodes connected by a lattice direction as depicted in Figure 3.3. Between each pair of nodes there are different material cross sections, each assumed to be constant in this example. In general, the line connecting two nodes may pass through different material regions, and the material properties between two nodes will not be constant. In that case, the intersecting lengths of the line with each region need to be calculated. The streaming direction $\mathbf{\Omega}_d$ is from left to right in Figure 3.3, and the lattice spacing between the nodes is s_d.

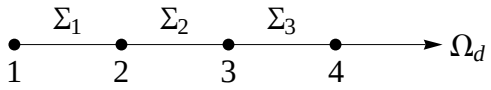


Figure 3.3: Nodes connected by a lattice direction, with different material properties between each pair of nodes.

By making the substitution

$$s' = s - js_d, \tag{3.13}$$

in equation (2.16) for iteration i, the equation becomes

$$TS\varphi^{[i]}_{\nu,d} = \sum_j \int_0^{s_d} ds' e^{-\tau_j(s')} \mathcal{I}(S\varphi_{\nu_j}, S\varphi_{\nu_{j+1}}, s'). \tag{3.14}$$

Factoring out the attenuation, and using the fact that the total cross sections are constant between

nodes in this example, the angular flux at node 4 is given by equation (2.9) as

$$\varphi_{4,d}^{[i+1]} = \int_0^{s_d} ds' e^{-\Sigma_3 s'} \mathcal{I}(S\varphi_3, S\varphi_4, s') + e^{-\Sigma_3 s_d} \int_0^{s_d} ds' e^{-\Sigma_2 s'} \mathcal{I}(S\varphi_2, S\varphi_3, s') + e^{-\Sigma_3 s_d} e^{-\Sigma_2 s_d} \int_0^{s_d} ds' e^{-\Sigma_1 s'} \mathcal{I}(S\varphi_1, S\varphi_2, s'). \quad (3.15)$$

Note that it is not required for the material cross sections to be constant between nodes, it is assumed here for simplicity. The recursive relation between the angular flux at a node, and the flux values at the other nodes along a specific direction, is given by

$$\varphi_{\nu,d}^{[i+1]} = \varphi_{\nu-1,d}^{[i]} e^{-\Sigma_{\nu-1} s_d} + \int_0^{sd} ds' e^{-\Sigma_{\nu-1} s'} \mathcal{I}(S\varphi_{\nu-1,d}, S\varphi_{\nu,d}, s'), \quad (3.16)$$

with the condition that

$$\varphi_{2,d}^{[i]} = \int_0^{s_d} ds' e^{-\Sigma_1 s'} \mathcal{I}(S\varphi_{1,d}, S\varphi_{2,d}, s'). \quad (3.17)$$

Note that the condition is at node $\nu = 2$, and not at node $\nu = 1$. This is because the interpolating function $\mathcal{I}(S\varphi_{1,d}, S\varphi_{2,d}, s')$ takes the flux at $\nu = 1$ (the boundary) into account. The running total is accumulated in $\varphi_{\nu-1,d}^{[i]}$ and added to the flux at the nodes as the accumulated flux is propagated through the lattice.

Once the angular flux in a given direction has been calculated, it is then multiplied with the quadrature weight corresponding to that direction and added to the scalar flux. Each of the angular flux values at a node can be calculated, and added to the scalar flux independently.

When angular refinement is used, some lattice directions do not intersect the problem boundary at a node. In such cases where the neighbouring node would lie outside of the problem domain, the intersection between the problem boundary and the line along the lattice direction needs to be calculated.

At the calculated point of intersection, the flux value on the boundary is determined by using the flux values of the surrounding nodes. Any off-lattice point on the boundary is confined to a square, with four nodes surrounding it. Barycentric coordinates are used as interpolants to determine the angular flux value at the intersection point.

Barycentric coordinates were originally introduced by August Ferdinand Möbius in 1827 [47] (original work reprinted in 1976), and are a form of homogeneous coordinates. (For a description of barycentric coordinates defined on irregular convex polygons see [48].)

For a convex polygon with unit area in 2D, generalized barycentric coordinates in the interior of the polygon are defined by the following three properties:

$$\lambda_i(x, y) \geq 0, \quad (3.18)$$

$$\sum_i \lambda_i(x, y) = 1 \quad (3.19)$$

and

$$\sum_i \lambda_i(x_p, y_p) \mathbf{r}_i = \mathbf{p}. \quad (3.20)$$

In the equations above, $\mathbf{p} = (x_p, y_p)$ is an interior point, the $\lambda_i(x_p, y_p)$ are the barycentric coordinates at $\mathbf{p} = (x_p, y_p)$ and $\mathbf{r}_i = (x_i, y_i)$ are the vertices of the polygon. Any function fulfilling the three requirements set out above can be used as barycentric coordinates. In this study the barycentric coordinates defined on the unit square are linear functions chosen as

$$\begin{aligned} \lambda_1(x, y) &= (1 - x)(1 - y), \\ \lambda_2(x, y) &= x(1 - y), \\ \lambda_3(x, y) &= xy, \\ \lambda_4(x, y) &= (1 - x)y, \end{aligned} \tag{3.21}$$

where the indices correspond to the vertices of the square, numbered in an anti-clockwise direction, starting at the bottom left corner.

Graphically, each barycentric coordinate can be associated with an area on the square as depicted in Figure 3.4. As the point is moved towards any node, the corresponding area increases and the areas associated with the other nodes decrease.

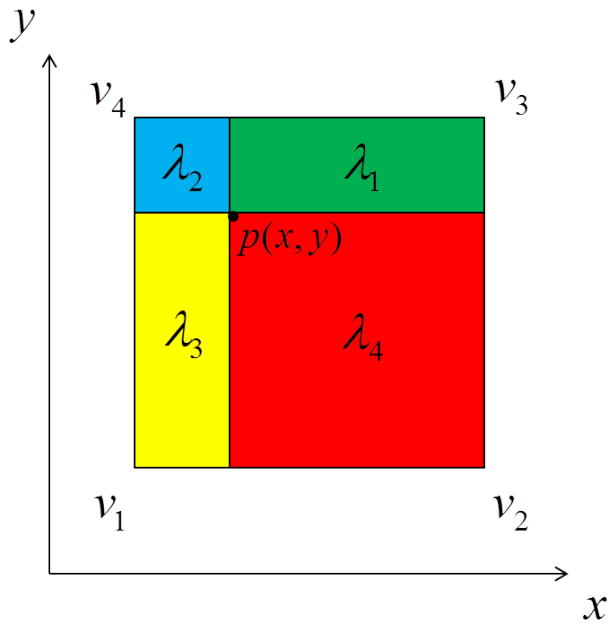


Figure 3.4: Graphical representation of barycentric coordinates on a square.

To determine the angular flux value at a given point $\mathbf{p}$ on the square, the barycentric coordinates are used as weights in the sum

$$\varphi_d(x_p, y_p) = \sum_{\nu=1}^{4} \lambda_\nu(x_p, y_p)\varphi_{\nu,d}, \tag{3.22}$$

where $\varphi_d(x_p, y_p)$ is the angular flux at point $\mathbf{p}$ in direction $\mathbf{\Omega}_d$, $\lambda_\nu(x_p, y_p)$ are the barycentric coordinates at $\mathbf{p}$, and $\varphi_{\nu,d}$ are the angular flux values at the nodes in direction $\mathbf{\Omega}_d$. The weights $\lambda_\nu(x_p, y_p)$ are recovered from equation (3.20).

When high levels of angular refinement are used, without spatial refinement of the lattice, the distances between nodes in the added directions become bigger. Over large distances, low order interpolants cannot accurately approximate the flux shape between nodes and spatial refinement is required.

Even though additional calculations are required to account for the flux at off-lattice points, the calculation of the scalar flux at nodes remains the same. The sum over the lattice directions is extended to include the flux contributions from the added lattice directions. After the flux values at a node have been calculated, the stored first collision source is then added and the total scalar flux is calculated.

3.3.3 Convergence

After each streaming step, the flux values from successive iteration steps are compared to determine if the flux solution is converged. The difference between flux values of successive iterations at a node is defined as

$$\Delta_\nu = \left| \frac{\phi_\nu^{[i+1]} - \phi_\nu^{[i]}}{\phi_\nu^{[i]}} \right|. \tag{3.23}$$

In equation (3.23), Δ_ν is the relative difference between the scalar flux values of successive iterations, and $\phi_\nu^{[i]}$ is the scalar flux at node ν at iteration i. Convergence is determined by comparing the difference at every node to a set tolerance ε. If the condition

$$\Delta_\nu \leq \varepsilon \; \forall \; \nu \tag{3.24}$$

is satisfied, the iteration process is stopped. If the convergence criterion is not met, another iteration is performed, and the iteration process is continued until convergence is reached.

Different nodes in the lattice converge at different rates, depending on their position in the geometry. Nodes separated by heavy absorbers converge at very different rates, and nodes located in very low scattering regions converge much quicker than the rest of the lattice. The entire iteration scheme is shown graphically in Figure 3.5.

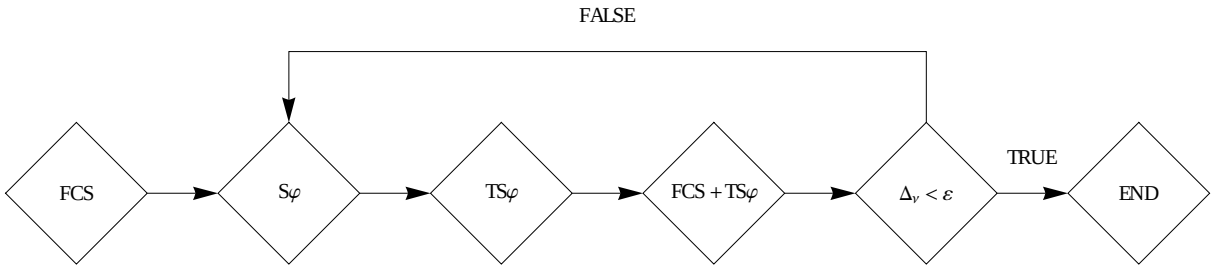


Figure 3.5: Flow chart showing the execution sequence of the iteration scheme.

3.4 Extensions to the algorithm

As the distance between a lattice node and the source increases, the solid angle which covers the source subsequently decreases. This means that nodes further away from the source require less quadrature points to accurately calculate the first collision source. By using a coarser discretization further away from the source, the calculational time of the first collision source can be reduced. In conjunction with a coarser discretization of the integration domain, higher order integration schemes, such as Gaussian integration on the sphere [49], can be used to improve accuracy.

In the streaming section above, the material cross sections between nodes are assumed to be constant. When the cross sections are not constant along a lattice direction, the optical path through each material must be calculated. This can be done by calculating the intersection of the line along a lattice direction, with the various material regions it passes through. Each of the intersection lengths are then used for attenuation during the streaming step.

The optical paths along lattice directions can either be stored, or calculated while performing the streaming step. Calculating the optical paths in a preprocessing step, and storing the values will increase the required computer storage. This will, however, decrease the number of calculations done per iteration.

3.5 Concluding remarks

The entire algorithm, as described in Chapter 2 and 3, was implemented in a C++ program. Initially, the first collision source was implemented in serial code. Even with the optimization, suggested above, of limiting the integration domain, the first collision source is very expensive to calculate (in terms of calculational time). Futhermore, calculating the first collision source for large lattices in serial is not feasible. For this reason, the first collision source was implemented in parallel, which resulted in a great improvement in calculational time.

Streaming on the lattice is done by sweeping across the lattice in each direction, instead of looping over the streaming directions at each node. The sweeping algorithm, along with the recursive definition of the streaming operator, increase the efficiency of the streaming part of the algorithm.

Chapter 4

Simulation results

In order to investigate the applicability of the Lattice Boltzmann Method (LBM) to neutron transport, a model problem was chosen that would show the main features of the method. A number of control points were chosen where the calculated results are compared to a reference solution, calculated using the Monte Carlo Neutral Particle (MCNP) transport code. MCNP is a transport code based on stochastic methods. It is widely used as the industry standard to calculate reference solutions for particle transport problems. The relative error between the LBM and MCNP is calculated using

$$\text{Err} = \frac{\text{LBM} - \text{MCNP}}{\text{MCNP}} \times 100. \tag{4.1}$$

Apart for the reference solution, the LBM results are also compared to that of a nodal S_N code. This is done to illustrate the performance of the LBM, as compared to one of the currently used transport methods.

Three different interpolants, namely constant, linear and exponential, were used to account for the angular flux accumulated between nodes during the streaming step. The interpolants essentially approximate the angular flux between nodes, along a specific direction by assuming the flux has the same "shape" as the chosen function. In the streaming step, the analytical expressions for the integrals in equation (2.16) are used to calculate the accumulated flux.

4.1 The model problem

The chosen problem consists of a localized source that is embedded in a scattering and absorbing material. Material cross sections are constant throughout the entire problem including the source region. This problem was chosen as a prototype for a set of benchmark problems proposed by Kobayashi, Sugimura and Nagaya [50]. These problems were proposed to test the accuracy of various transport methods, where localized sources and void regions are contained in the problem geometry.

A number of points were selected throughout the geometry, where the calculated flux values are compared to the reference solution. These points correspond to a subset of the points that is used in the full set of benchmark problems mentioned above.

Figure 4.1 shows the problem geometry which consists of a source region shown in red, and a

scattering medium shown in blue. The source region is a cube with side length 10 cm embedded in the scattering material – a cube of side length 25 cm.

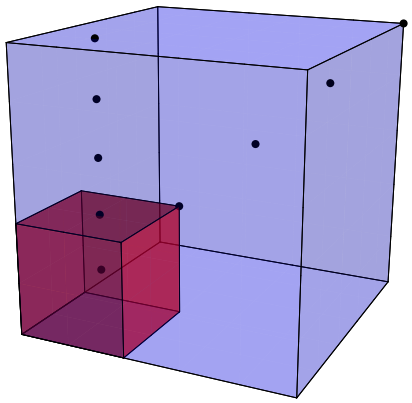


Figure 4.1: Localized source embedded in a material with control points shown.

Points along the diagonal and along one of the coordinate axes are used as control points where the results are compared. Along the diagonal, the problem exhibits symmetry and points along the other coordinate axes are equivalent, meaning that the scalar flux values at the equivalent points along the other axes are equal.

Choosing points along the diagonal and the coordinate axis, indicates the effect of over estimation caused by the low angular quadrature. These points are directly linked to the source through lattice directions. As the distance from the source is increased, the effect becomes more pronounced.

The chosen points also indicate the greatest effect that angular refinement has on the flux solution. Points along the coordinate axis are used to investigate how the interpolation of the flux on the boundary influences the flux solution in the interior of the problem.

The total cross section for the entire problem was chosen to be $\Sigma_t = 0.1$, corresponding to the cross section used in the 3D benchmarks mentioned above. A scattering ratio of $\Sigma_s/\Sigma_t = 0.5$ is used, meaning the scattering cross section is $\Sigma_s = 0.05$ throughout the problem.

Although the results are only compared to the reference solution at the control points, the flux solution in the LBM is still calculated at each of the lattice nodes. For a lattice with no spatial refinement, the lattice spacing along any coordinate axis is restricted to a quarter of the scattering mean free path. The scattering mean free path is given by $\Sigma_s^{-1} = 20\,\text{cm}$, and so the lattice spacing is $s_d = 5\,\text{cm}$ along each coordinate axis.

Setting the lattice spacing to a quarter of the scattering mean free path, means the effect of scattering is treated more accurately along lattice directions. This starting point was chosen arbitrarily, but a coarser lattice can also be used as a starting point. Starting with a coarser mesh however, requires more levels of spatial refinement to produce accurate results. With each level of spatial refinement, the number of nodes along a lattice direction increase, and the effects of scattering are captured more accurately.

To determine the number of lattice nodes along any of the coordinate directions, the lattice spacing and the problem size in that direction are used. For the problem considered here, a cube with side

length $L = 25\,\text{cm}$, the number of nodes along any coordinate direction (without spatial refinement) is given by

$$N_x = \text{ceiling}\left(\frac{L}{s_d}\right) + 1 = 6. \tag{4.2}$$

In equation (4.2), ceiling(x) is the mathematical ceiling function which gives the smallest integer larger or equal to the fraction L/s_d. For this particular case L/s_d is an integer, but this may not always be the case. One node is added in each direction to ensure that the lattice covers the entire problem geometry. Once the number of nodes per coordinate direction is determined, the nodes are evenly distributed along each of the coordinate axes.

The number of nodes in each coordinate direction need not be equal. That is, the problem geometries that can be considered are not restricted to cubic geometries. Although there is no restriction of the problem geometries that can be handled by the LBM, it is required that the lattice spacing be equal in all of the coordinate directions, i.e. a cubic lattice.

When the outer boundary of the problem consists of curved shapes, such as spheres and cylinders, the bounding box of the problem is used to determine the number of nodes along each coordinate axis in the lattice. By using the bounding box, it insures that the lattice covers the entire problem geometry.

In each of the following sections the results are presented for points along the diagonal and along one of the coordinate axes. In the case where points along the diagonal are used, the x, y and z coordinates of each point are equal. Points along the diagonal are chosen so that

$$(x, y, z) = (5i, 5i, 5i) \text{ with } i = 1, 2, 3... \tag{4.3}$$

and the coordinates of the control points along the coordinates axis is given by

$$(x, y, z) = (5, 5, 5i) \text{ with } i = 1, 2, 3... \tag{4.4}$$

In order to calculate the reference solution, the problem described above was perturbed with regards to the boundaries, so that the point detectors used in MCNP are contained within the geometry. This is done to ensure that the solution at the points on the boundary can reach convergence. Because the boundaries were extended, the tracks and lattice points on the boundaries in the LBM are considered to be contained within the problem geomtery as well. For this reason, the total flux is considered during both the streaming and the scattering steps and need not be divided as described in Chapter 3.

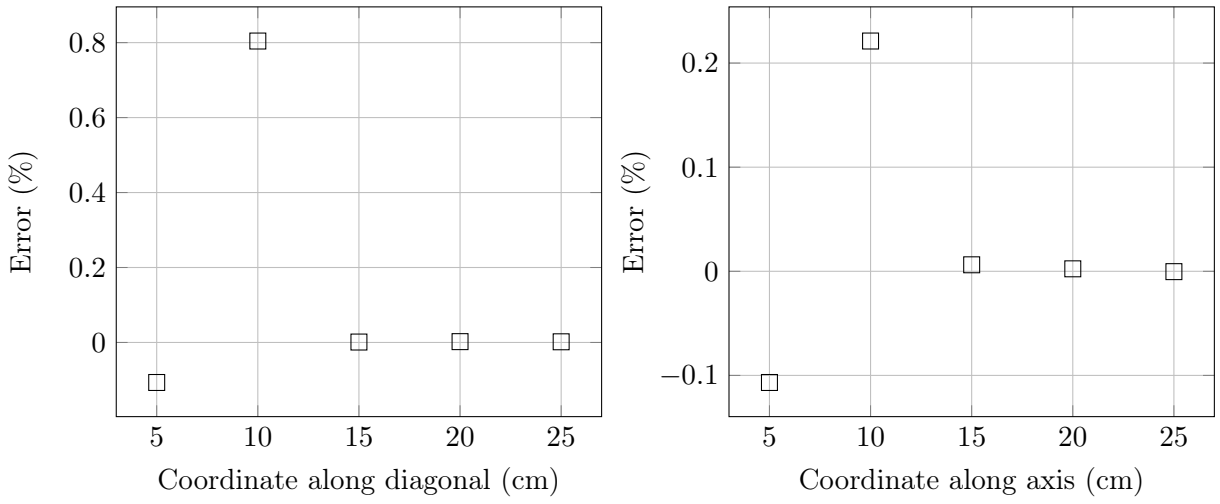
4.2 First collision source results

Using the algorithm described in Chapter 3, the first collision source is calculated. Once again, the first collision source is calculated at each node in the lattice but only compared to the reference solution at the selected points. With each level of spatial refinement, the first collision source for the newly added nodes must be calculated. This makes the first collision source a very expensive calculation for large lattices, in terms of calculational time.

The same angular discretization for calculating the integral in equation (2.35) was used throughout the lattice. At each node, the bounding box for the source was divided into 500×500 points giving a total of 250 000 integration points. Although this might be unnecessary for points far from the source, a very fine angular discretization is required for points inside the source region. Within the source region integration is performed over the entire sphere.

Higher order integration schemes on the sphere can be used to decrease the number of quadrature points that are needed to perform the angular integration of the first collision source. An example of Gaussian quadrature on the surface of the sphere is given in [49].

Figures 4.2(a) and 4.2(b) show that the largest errors, on the first collision source values at the control points, occur on the boundary of the source region. This is due to the fact that the angular integral is evaluated over the entire sphere, while the number of quadrature points used is the same as for nodes outside the source region. Outside the source region, where the integration is limited by using the approach described in Chapter 3, the error is much less than inside the source region. Although errors are made when calculating the first collision source, the results show good agreement with the reference solution.



(a) Errors along the diagonal.

(b) Errors along the coordinate axis.

Figure 4.2: Relative errors on the first collision source values as compared to MCNP.

4.3 Interpolants used during streaming

A number of different interpolants were used to account for the accumulated flux between nodes. In this section, the results for constant, linear and exponential interpolants between nodes are given. Each of the cases presented shows how different approximations affect the flux solution.

For each interpolant, results are given for two cases: (a) spatial refinement without angular refinement; and (b) spatial refinement with one level of angular refinement. The first case shows how spatial refinement alone improves the final flux solution, and where angular refinement is needed to improve the solution further. One level of angular refinement is added to show how the flux solution is improved.

The interpolants that are used are directionally dependent, because the streaming length and values

used for interpolation change with the direction. Each interpolant takes as input the streaming length and the angular flux values stored at the nodes. In the sections below, the expression for the interpolant and analytical expression for the evaluated integrals in equation (2.16) are given. In each case, the expressions are given for constant cross sections between the nodes. This is, however, only true for the particular problem being considered here and not in general.

A section on interpolant performance is also included to compare the best results of the different interpolants with each other. There it is shown how the order of the interpolant that is used during streaming influences the accuracy of the final solution.

4.3.1 Constant interpolant

When using a constant interpolant between the lattice nodes, only the angular flux value of the node at the current position is required while sweeping through the lattice. It is assumed that the flux values between the two nodes in question are constant, and therefore equal to the value at the current node. The expression for the constant interpolant is given by

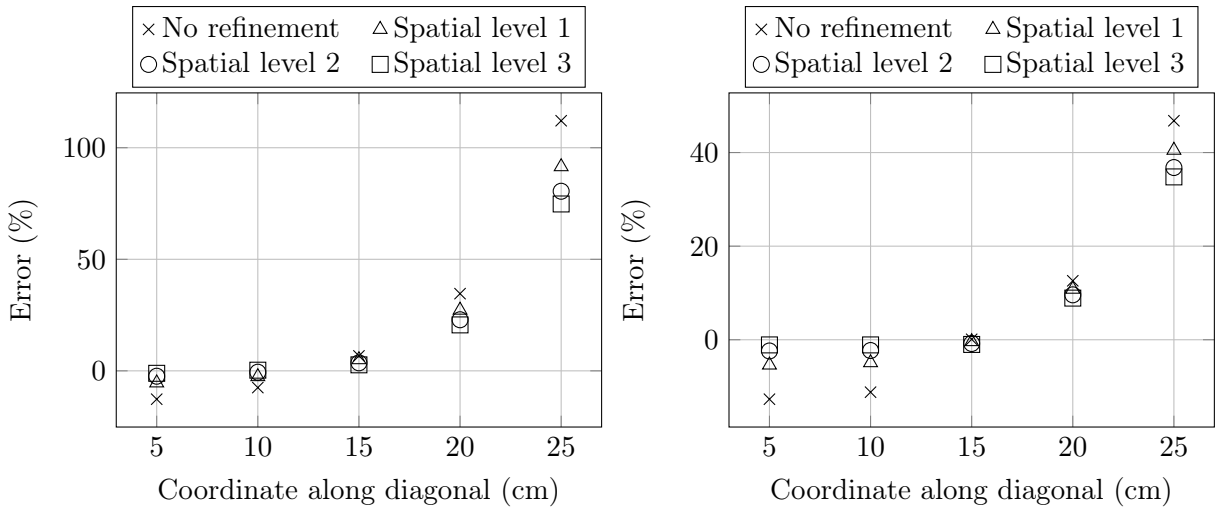
$$\mathcal{I}(S\varphi_\nu, S\varphi_{\nu'}, s) = \varphi_{\nu,d}, \tag{4.5}$$

and the expression for the evaluated integral is given by

$$\int_0^{s_d} ds e^{-\tau(s)} \mathcal{I}(S\varphi_\nu, S\varphi_{\nu'}, s) = \frac{\varphi_{\nu,d}}{\Sigma_t}(1 - e^{-\Sigma_t s_d}), \tag{4.6}$$

where $\varphi_{\nu,d}$ is the flux value at the current node, Σ_t is the total cross section between the nodes and s_d is the distance between the nodes.

In Figures 4.3(a) and 4.3(b) it can be seen that without angular refinement the results in and around the source region agrees well with the reference solution. As the node positions move further away from the source, larger errors are made due to the low angular quadrature.

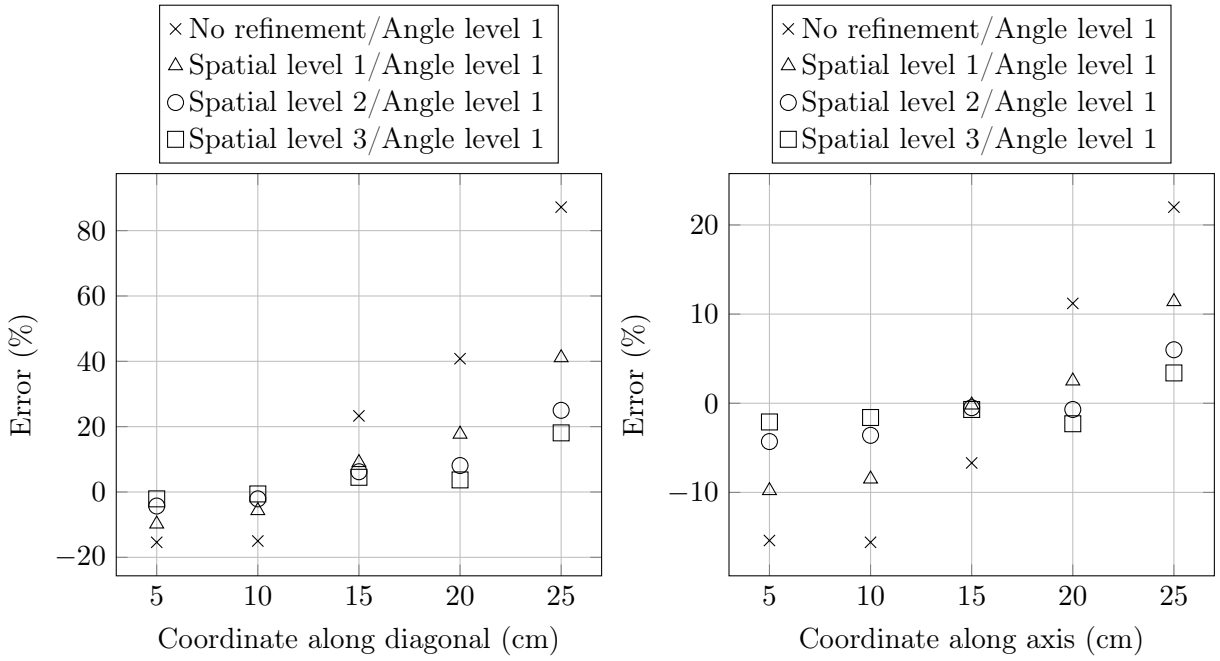


(a) Errors along the diagonal.

(b) Errors along the coordinate axis.

Figure 4.3: Relative errors on the flux values when using constant interpolants without angular refinement, as compared to MCNP.

To reduce the errors made at nodes that are far from the source, angular refinement is required. Figure 4.4 shows a significant reduction in errors that occur when using one level of angular refinement in conjunction with multiple levels of spatial refinement. The maximum error along the diagonal with three levels of spatial refinement is reduced from 74.8 % to 18.1 % (Figure 4.4(a)), and along the coordinate axis the error is reduced from 34.8 % to 3.4 % (Figure 4.4(b)). In both cases the maximum error occurs at the node that is furthest away from the source along the specific direction.



(a) Errors along the diagonal.

(b) Errors along the coordinate axis.

Figure 4.4: Relative errors when using constant interpolants with one level of angular refinement, as compared to MCNP.

4.3.2 Linear interpolant

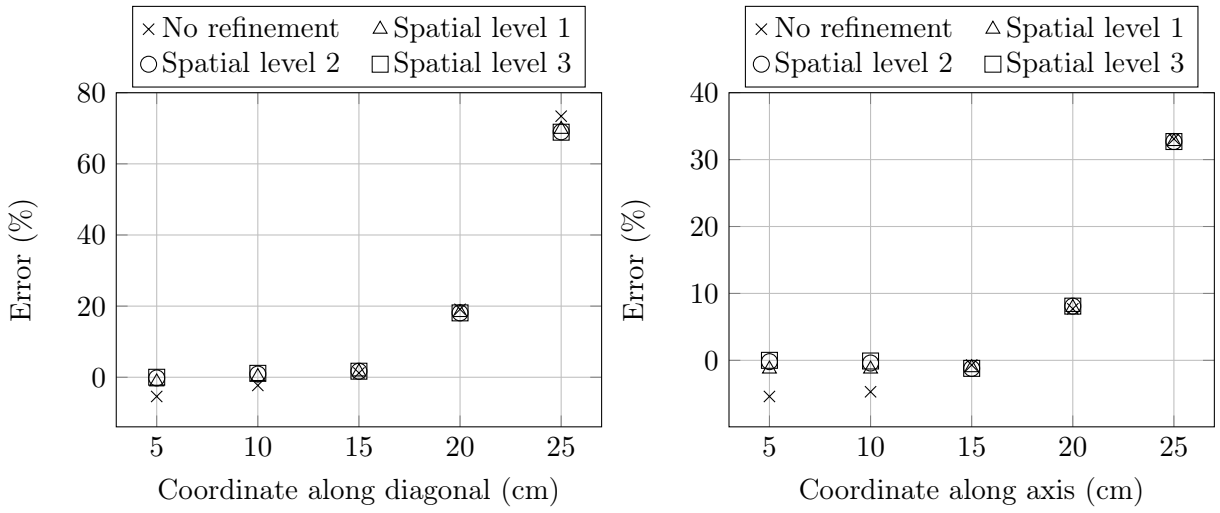
For linear interpolation, both the current node flux value and the neighbouring node flux value along a particular direction is required. The expression for the linear interpolant between nodes is given by

$$\mathcal{I}(S\varphi_\nu, S\varphi_{\nu'}, s) = \varphi_{\nu,d} - \frac{(\varphi_{\nu,d} - \varphi_{\nu',d})}{s_d} s. \tag{4.7}$$

When using a linear interpolant, it is assumed that the flux values between the nodes have a linear dependence on the flux values at the nodes. Substituting this expression for the interpolant into equation (2.16) and performing the integration, yields

$$\int_0^{s_d} ds e^{-\tau(s)} \mathcal{I}(S\varphi_\nu, S\varphi_{\nu'}, s) = e^{-s_d\Sigma_t} \frac{(\varphi_{\nu,d} - \varphi_{\nu',d} - s_d\Sigma_t\varphi_{\nu',d} + e^{s_d\Sigma_t})(\varphi_{\nu',d} + \varphi_{\nu,d}(s_d\Sigma_t - 1))}{s_d\Sigma_t^2}. \tag{4.8}$$

Using linear interpolation, instead of a constant value when streaming, decreases the errors that are made on the flux values (as can be seen in Figure 4.5). In contrast with the constant interpolant, spatial refinement only decreases errors slightly outside the source region.



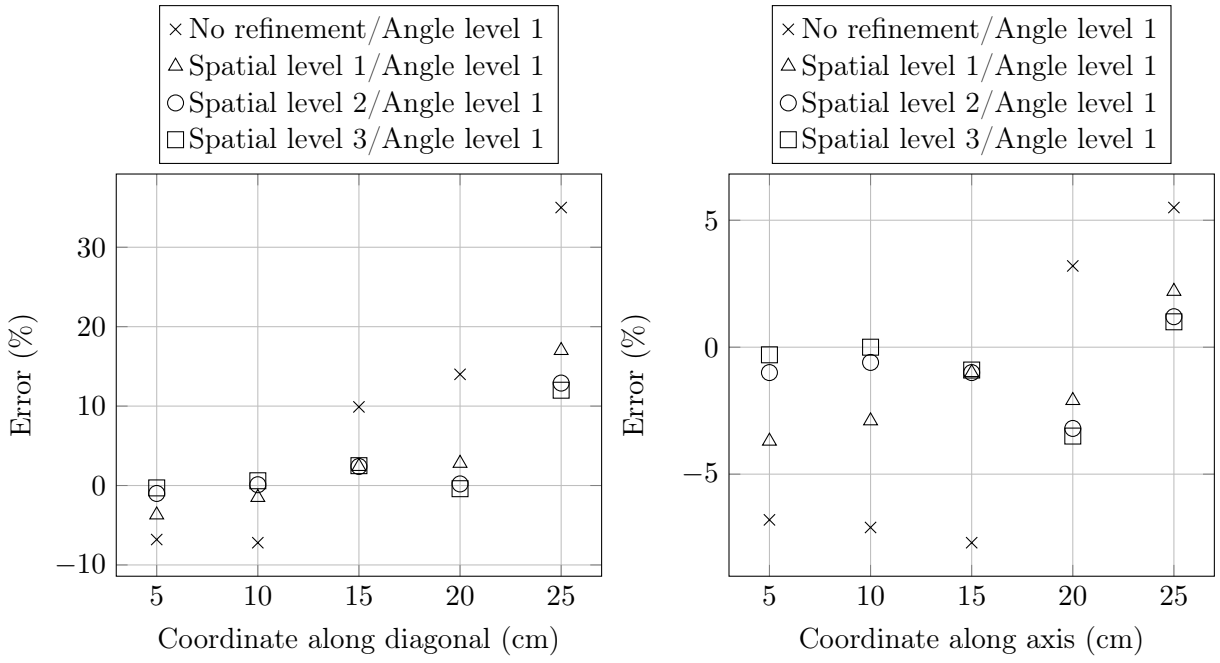
(a) Errors along the diagonal.

(b) Errors along the coordinate axis.

Figure 4.5: Relative errors when using linear interpolants without angular refinement, as compared to MCNP.

For nodes within the source region, spatial refinement decreases errors more significantly. This is due to the fact that the order of the interpolant is not high enough to accurately account for the variation in the flux during streaming. With spatial refinement, the linear interpolant becomes a better approximation of the flux shape within the source region.

When angular refinement is used with the linear interpolant, once again a significant reduction in error is observed. Figure 4.6 shows the errors along the diagonal and coordinate axis when one level of angular refinement is used with spatial refinement. In this case, spatial refinement gives a greater improvement in the error both inside and outside the source region.



(a) Errors along the diagonal.

(b) Errors along the coordinate axis.

Figure 4.6: Relative errors when using linear interpolants with one level of angular refinement, as compared to MCNP.

Along the diagonal, the maximum error is reduced from 68.8 % to 12 % (Figure 4.6(a)), and along the coordinate axis the error is reduced from 32.7 % to 1 % (Figure 4.6(b)). The reduction in error is less dramatic than in the case where a constant interpolant is used. That is because the linear interpolant already approximates the flux shape between nodes more accurately, and thus the flux solution without any refinement is more accurate. Although the overall reduction in error is less in this case, the overall accuracy of the linear interpolant yields better results than the constant interpolant.

4.3.3 Exponential interpolant

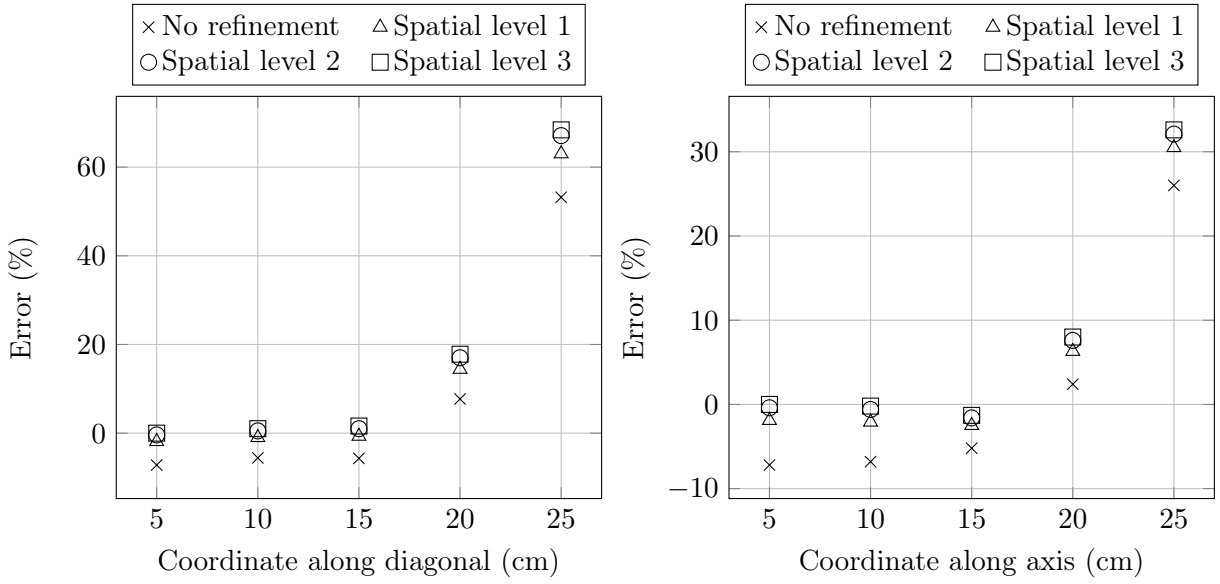
Lastly, an exponential interpolant is considered to approximate the flux between nodes. The expression for the exponential interpolant is given by

$$\mathcal{I}(S\varphi_\nu, S\varphi_{\nu'}, s) = \varphi_{\nu,d} e^{\ln\left(\frac{\varphi_{\nu',d}}{\varphi_{\nu,d}}\right)\frac{s}{s_d}}, \tag{4.9}$$

with the evaluated integral in equation (2.16) given by

$$\int_0^{s_d} ds e^{-\tau(s)} \mathcal{I}(S\varphi_\nu, S\varphi_{\nu'}, s) = \frac{(\varphi_{\nu,d} - e^{-s_d\Sigma_t}\varphi_{\nu',d})s_d}{s_d\Sigma_t - \ln\left(\frac{\varphi_{\nu',d}}{\varphi_{\nu,d}}\right)}. \tag{4.10}$$

Figure 4.7 shows that spatial refinement alone increases the errors further away from the source region, while in and around the source region errors are decreased with spatial refinement. As the distances between nodes are decreased, the exponential interpolant yields similar results to the linear interpolant.



(a) Errors along the diagonal.

(b) Errors along the coordinate axis.

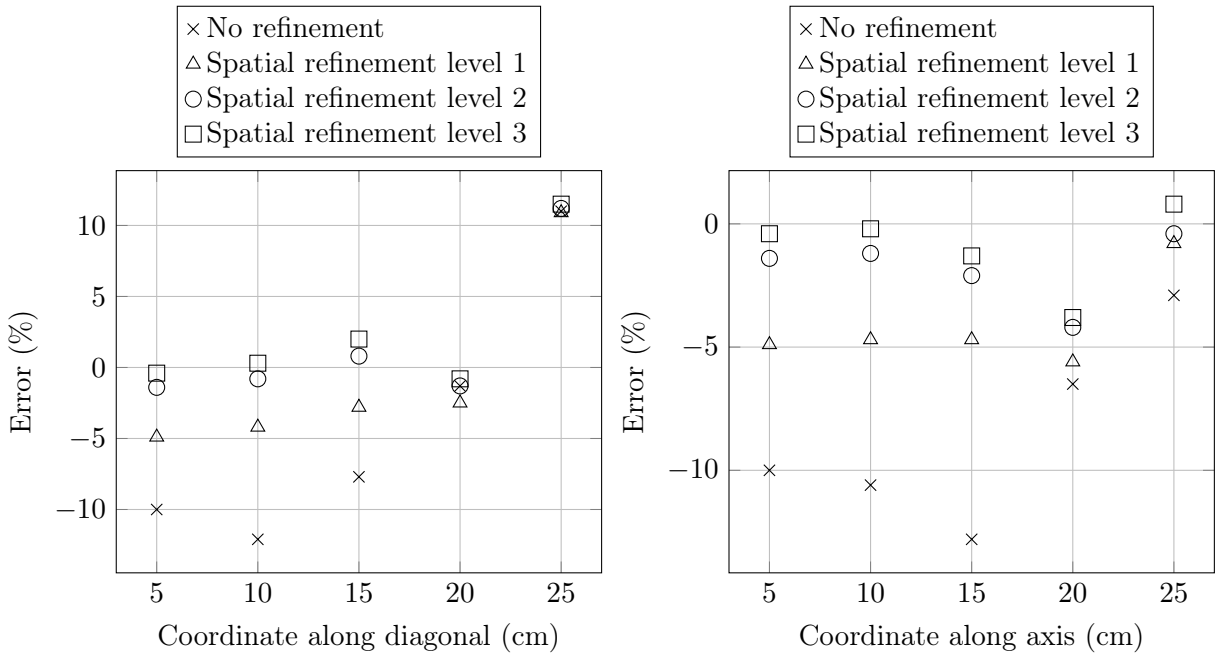
Figure 4.7: Relative errors when using exponential interpolants without angular refinement, as compared to MCNP.

In and around the source region errors are decreased with spatial refinement. As the distances between nodes are decreased, the exponential interpolant approximates the flux more accurately.

Angular refinement without spatial refinement slightly increases errors inside and around the source region. This is because the interpolant is used over a greater distance in the newly added directions. With the increased distance, more of the higher order behaviour of the exponential function is taken into account which is a weaker flux approximation within the source region.

Spatial refinement improves this error, although further away from the source the improvement in error is less. Overall, the accuracy of the exponential interpolant is better than the constant interpolant, while exhibiting similar results as that of the linear interpolant.

Along the diagonal with both spatial and angular refinement, the maximum error is reduced from 68.4 % to 11.5 % (Figure 4.8(a)). For control points along the coordinate axis, the maximum error is reduced from 32.6 % to 0.8 % (Figure 4.8(b)).



(a) Errors along the diagonal.

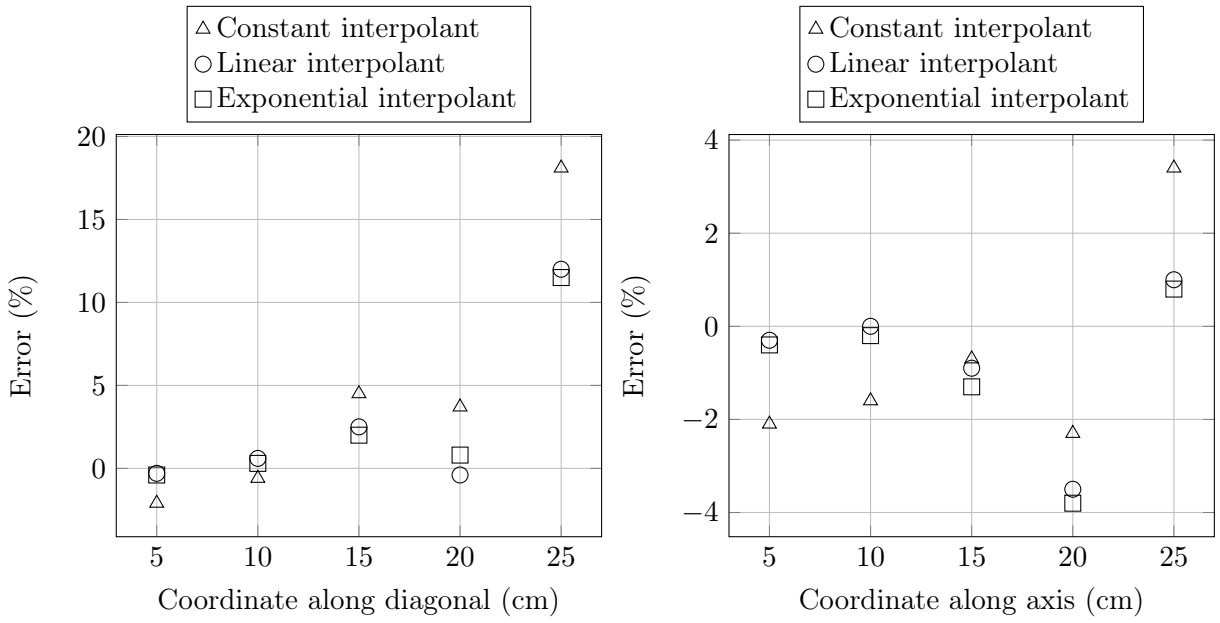
(b) Errors along the coordinate axis.

Figure 4.8: Relative errors when using exponential interpolants with one level of angular refinement, as compared to MCNP.

4.3.4 Comparison between different interpolants

When the results from different interpolants are compared, it is clear to see that the higher order interpolants (linear and exponential) outperforms the constant interpolant in terms of accuracy.

Figures 4.9(a) and 4.9(b) show the errors that are made on the flux value when using different interpolants. In each case, one level of angular refinement and three levels of spatial refinement were used. In the figure it can be seen that the linear and exponential interpolants give more accurate results than the constant interpolant. As the lattice spacing is reduced, the linear behaviour of the exponential interpolant dominates and yield results similar to the linear interpolant.



(a) Errors along the diagonal.

(b) Errors along the coordinate axis.

Figure 4.9: Different interpolant results compared to one another.

4.4 LBM compared to nodal S_N

Nodal Discrete Ordinates Methods, or nodal S_N methods, differ from finite difference discrete ordinates methods in that the flux, both at the boundary and interior of the node, is expanded in terms of polynomials. This allows larger node sizes as compared to the cells used in finite difference S_N, but with similar accuracy [51].

In each node, the polynomials that are used for the flux expansion, can also be used as interpolating functions to determine the flux values at different points. Although nodal S_N methods are more suited for volume averaged flux calculations, interpolation was used to determine the scalar flux values at the control points. For the comparison between the LBM and nodal S_N, an S_{16} quadrature set was used with polynomial expansion order 6.

Figure 4.10 shows the relative errors along the diagonal for the LBM with constant, linear and exponential interpolants and nodal S_{16}, all compared to the MCNP reference results. In the figure the green markers indicate the relative errors of the nodal S_N results as compared to the MCNP reference.

In comparison to the nodal S_N method, the LBM performs well in terms of flux values at the selected control points. In most cases, with the exception of the point $(15, 15, 15)$, the LBM gives more accurate results – even when a low order interpolant is used between nodes.

4.5 Concluding remarks

As is evident from the results, angular refinement needs to be accompanied with spatial refinement. When spatial refinement is used, the larger streaming lengths of the newly added directions are consequently decreased. This leads to a better spatial approximation, even when lower order interpolants are used.

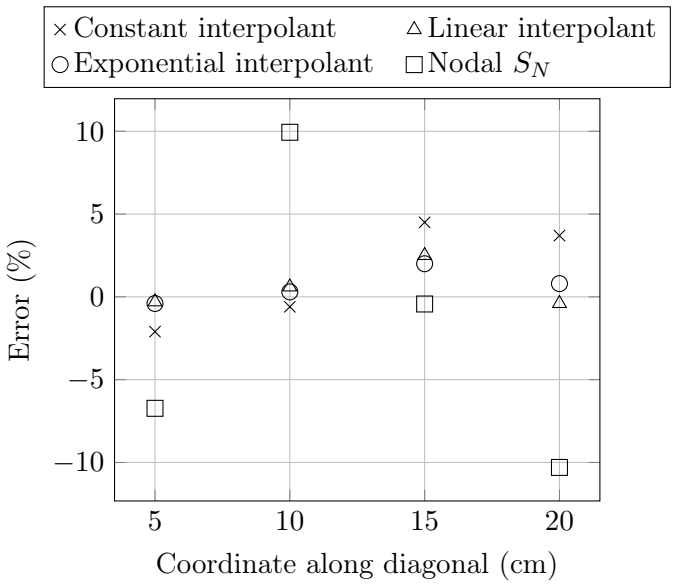


Figure 4.10: LBM results for constant, linear and exponential interpolants as compared to nodal S_N results.

The LBM is more suited to calculate flux values at points, such as detector responses, than volume averaged values. The reason for this is, that the flux values are only calculated and stored at lattice nodes, and interpolation (both spatial and angular) is required when calculating volume integrated quantities.

Within isotropic source regions or when many distributed isotropic sources are present, the LBM yields more accurate flux values without requiring high levels of spatial and angular refinement. As the distance between a node and a localized source increases, the effect of overestimation due to a low angular quadrature becomes more pronounced. Further away from localized sources, higher levels of angular and spatial refinement are required to accurately approximate the flux values.

Because the first collision source can be calculated with very high accuracy, the LBM yields more accurate results when the scattering ratio is low. That is, when the scattering cross section is small compared to the total cross section.

Chapter 5

Conclusions and future work

The results that are presented in Chapter 4 indicate that the Lattice Boltzmann Method (LBM) is capable of solving neutron transport problems, although some problems arise at nodes that are located far from localized sources. These difficulties are addressed with the implementation of an angular refinement scheme to increase the angular coverage of the problem phase space.

From the results presented, it can be seen that the LBM is more suited to solve transport problems in and around regions containing sources, when low levels of angular refinement are used. To obtain more accurate results in regions far from sources, higher levels of both spatial and angular refinement are required.

Because of the rapid increase in the number of directions with the level of angular refinement and the cubic growth rate of the lattice, solving problems which require high levels of refinement will be calculationally expensive.

For problems that contain very high scattering media, a large lattice is required to accurately calculate the flux solution. With the increase in lattice size, the required first collision source calculation will incur a high calculational expense. To be able to solve such problems, a more sophisticated parallel implementation of the first collision source is required.

To minimize the calculational time further, the rest of the calculational algorithm should be implemented in parallel as well. Because of the nature of the LBM, the algorithm lends itself to massive parallelization such that larger problems can be solved more efficiently. Further investigations are also required to determine if the effect of angular refinement is sufficient to yield results with the required accuracy far from localized sources, and what effect this will have on the efficiency of the method.

Apart from large problems with localized sources, a possible application of the LBM is the calculation of the flux in an infinite lattice environment of repeated structures, such as reactor fuel elements, to generate homogenized cross sections for reactor components. This will, however, require the implementation of boundary conditions to simulate the infinite environment, and also the inclusion of fission in the source model.

A drawback of the LBM is that obtaining accurate scalar flux solutions in volumes, will require a spatially dense lattice when low order interpolants are used when performing the spatial integration. This is because the LBM is more suited to calculate the flux at lattice nodes instead of volumes. However, for cross section homogenization, flux solutions integrated over volumes are required.

This problem can be overcome by using higher order quadrature rules when performing the spatial integration, with the flux values at the nodes used as quadrature points.

As an alternative, the LBM can be used as a solution method to calculate the adjoint problem, yielding importance solutions that can be used to generate weight windows for codes such as MCNP. Using regions of importance or weight windows is a method that is used to accelerate stochastic calculations.

Appendix A

Combinatorial proof of equation (2.28)

This appendix gives a combinatorial proof of equation (2.28),

$$N_L = 8 + 12(2^{L+1} - 1) + 6(2^{L+1} - 1)^2 \text{ with } L = 0, 1, 2, ..., \tag{A.1}$$

the total number of directions on the cube for a given level of angular refinement L. Each of the terms in equation (A.1) corresponds to the number of directions associated with the corners (8), edges ($12(2^{L+1} - 1)$) and faces ($6(2^{L+1} - 1)^2$), respectively.

On the surface of the cube, each edge corresponds to a line and each direction corresponds to a point on the line. The corners are counted separately and are excluded when counting the number of points on a specific edge to avoid double counting. With each level of angular refinement, directions to next-nearest neighbours on the lattice are added. This corresponds to dividing each segment on an edge into two equal parts and adding that point.

Consider a line divided into two equal segments by a point. At each next level of refinement, every segment is divided into another two equal segments by inserting a point at the centre of each segment. Let L be the level of refinement, then for $L = 0$ there are $2^{0+1} = 2$ segments and $2^{0+1} - 1 = 1$ point, and for $L = 1$ there are 4 segments and 3 points. Figure A.1 illustrates the cases for $L = 0$ and $L = 1$.

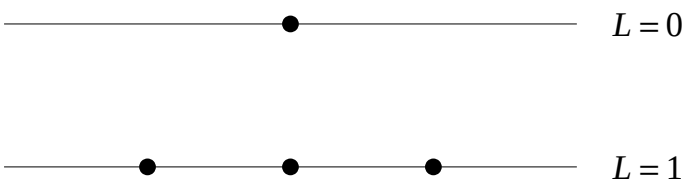


Figure A.1: Illustration of the number of points and segments for refinement level $L = 0$ and $L = 1$.

For the next level of refinement, the number of points that need to be inserted is equal to the number of segments in the current level. That is because each segment requires one point to divide that segment into two parts. Dividing each segment results in doubling the number of segments from one level of refinement to the next, i.e. for $L = k$ there are 2^{k+1} segments and for $L = k + 1$ there are $2 \cdot 2^{k+1} = 2^{k+2}$ segments.

The inductive hypothesis is that at $L = k$ there are $2^{k+1} - 1$ points, and it is required to prove that at level $L = k + 1$ there are $2^{k+2} - 1$ points on the line. For $L = k + 1$ the number of points that

need to be inserted is equal to the number of segments at $L = k$. That means that 2^{k+1} points need to be added in order to divide each segment at level k into two equal segments. The number of points at $L = k + 1$ is then

$$2^{k+1} - 1 + 2^{k+1} = 2 \cdot 2^{k+1} - 1 = 2^{k+2} - 1, \tag{A.2}$$

where the inductive hypothesis was used that level k contains $2^{k+1} - 1$ points and the fact that there are 2^{k+1} segments that must be divided. Thus, by mathematical induction, the number of points on the line at level L is $2^{L+1} - 1$.

From the proof above, the number of directions per edge for a level of angular refinement L, is equal to $2^{L+1} - 1$. There are 12 edges on the cube and the total number of directions, corresponding to the edges, for a given level of angular refinement is $12(2^{L+1} - 1)$.

Similarly, the number of points/directions in the interior of a face on the cube is equal to the product of the number of points on two perpendicular edges as shown in Figure A.2.

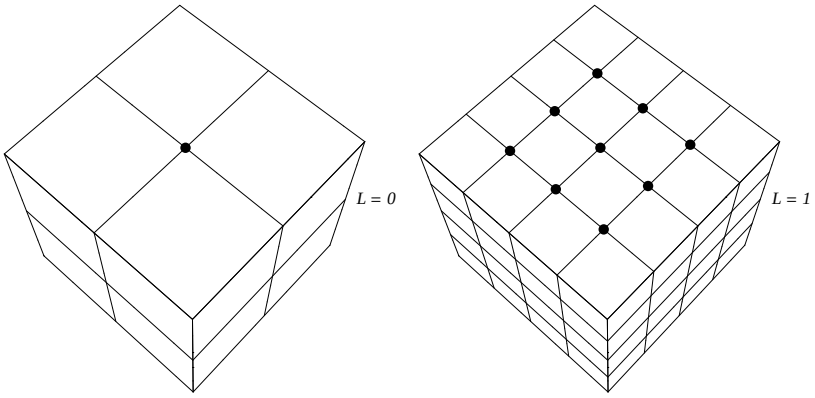


Figure A.2: Number of points contained in the interior of a face on the cube for levels $L = 0$ and $L = 1$.

That means that the number of points or directions per face at angular refinement level L is equal to $(2^{L+1} - 1)^2$. The total number of directions associated with the faces of the cube is then $6(2^{L+1} - 1)^2$ for a given level of refinement .

All that remains is to add the 8 corners which were excluded from both the edges and faces, then sum the totals of each component to find the total number of directions on the cube at refinement level L.

Bibliography

[1] Case, K.M. & Zweifel, P.F. (1967). *Linear transport theory.* Reading, MA: Addison-Wesley Pub. Co.

[2] Carlvik, I. (1966). *Integral transport theory in one-dimensional geometries.* Stockholm: Aktiebolaget Atomenergi.

[3] Carlvik, I. (1967). *Studies of integral neutron transport methods for nuclear reactor calculations.* Göteborg: Elanders Boktryckeri Aktiebolag.

[4] Carlvik, I. (1967). *Calculations of neutron flux distributions by means of integral transport methods.* Stockholm, Aktiebolaget Atomenergi.

[5] Williams, M.M.R. (1971). *Mathematical methods in particle transport theory.* London: Butterworths.

[6] Clark, M. & Hansen, K.F. (1964). *Numerical methods of reactor analysis.* New York: Academic Press.

[7] Davison, B. (1957). *Neutron transport theory.* Oxford: Clarendon Press.

[8] Chandrasekhar, S. (1950). *Radiative transfer.* London: Oxford University Press.

[9] Hébert, A. (2009). *Applied reactor physics.* Montréal: Presses Internationales Polytechnique.

[10] Spanier, J. & Gelbard, E. M. (1969). *Monte Carlo principles and neutron transport problems.* Reading, MA: Addison-Wesley.

[11] Lapeyre, B., Pardoux, E. & Sentis, R. (2003). *Introduction to Monte Carlo methods for transport and diffusion equations.* Oxford: Oxford University Press.

[12] Kofink, W. (1957). *Studies of the spherical harmonics method in neutron transport theory. Part I: The relation between PL- and gauss quadrature solutions of the Milne's problem.* Oak Ridge, TN: Oak Ridge National Laboratory.

[13] Kofink, W. (1957). *Studies of the spherical harmonics method in neutron transport theory. Part II: Behavior of the solution of the Milne problem with anisotropic scattering for L-∞.* Oak Ridge, TN: Oak Ridge National Laboratory.

[14] Carlson, B.G. & Bell, G.I. (1958). Solution of the Transport Equation by the S_N method. In *Proc. 2nd U.N. Intl. Conf. Peaceful uses of Atomic Energy*, Geneva, pages 2386–2411.

[15] Stamm'ler, R.J.J. & Abbate, M.J. (1983). *Methods of steady-state reactor physics in nuclear design.* London: Academic Press.

[16] Askew, J.R. & Brissenden, R. J. (1963). Some improvements in the discrete ordinate method of B.G. Carlson for solving the neutron transport equation. Report AEEW-R 161, Dorchester, Dorset, United Kingdom Atomic Energy Establishment, Winfrith, Fast Reactor Physics Division.

[17] Carlvik, I. (1967). *Collision probabilities for finite cylinders and cuboids.* Stockholm: Aktiebolaget Atomenergi.

[18] Roy, R. (1990). Anisotropic scattering for integral transport codes. Part 1. Slab geometries. *Annals of Nuclear Energy*, **17**(7):379–388.

[19] Roy, R. (1991). Anisotropic scattering for integral transport codes. Part 2. Cyclic Tracking and its application to XY lattices. *Annals of Nuclear Energy*, **18**(9):511–524.

[20] Askew, J.R. (1972). A Characteristics Formulation of the Neutron Transport Equation in Complicated Geometries. Report AEEW-M 1108, United Kingdom Atomic Energy Establishment, Winfrith.

[21] Halsall, M.J. (1980). CACTUS: A Characteristics Solution to the Neutron Transport Equations in Complicated Geometries. Report AEEW-R 1291, United Kingdom Atomic Energy Establishment, Winfrith.

[22] Roy, R. (1999). The Cyclic Characteristics Method with Anisotropic Scattering. *M&C'99 Conference, Mathematics and Computation,* 1225–1234. Madrid, Spain, September 27–30.

[23] McOwen, R.C. (1996). *Partial Differential Equations: Methods and Applications.* Upper Saddle River, New Jersey: Prentice Hall.

[24] Sanchez, R. & McCormic, N.J. (1982). A Review of Neutron Approximations. *Nuclear Science and Engineering*, **80**:481–535.

[25] Chopard, B. & Masselot, A. (2000). Cellular automata and lattice Boltzmann methods: a new approach to computational fluid dynamics and particle transport. *Future Generations Computer Systems*, **16**:249–257.

[26] Chopard, B., Dupuis, A., Masselot, A. & Luthi, P. (2002). Cellular Automata and Lattice Boltzmann Techniques. *Advances in Complex Systems*, **5**:2–3.

[27] Succi, S. (2001). *The lattice Boltzmann equation for fluid dynamics and beyond.* Oxford: Clarendon Press.

[28] Chen, S., Doolen, G. D. (1998). Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, **30**:329–364.

[29] McNamara, G.R. & Zanetti, G. (1988). Use of the Boltzmann Equation to Simulate Lattice-Gas Automata. *Phys. Rev. Lett.*, **61**(20):2332–2335.

[30] He, X. & Luo, L.-S. (1997). A priori derivation of the lattice Boltzmann equation. *Phys. Rev. E*, **55**:6333–6336.

[31] He, X. & Luo, L.-S. (1997). Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Phys. Rev. E*, **56**:6811–6817.

[32] Qian, Y.H., Succi, S. & Orszag, S.A. (2000). Recent advances in lattice Boltzmann computing. *Annual Reviews of Computational Physics*, **3**:195–242.

[33] Chen, S., Wang, Z., Shan, X.W. & Doolen, G. D. (1992). Lattice Boltzmann computational fluid dynamics in three dimensions. *J. Stat. Phys.*, **68**:379–400.

[34] He, X., Chen, S. & Zhang, R. (1999). A Lattice Boltzmann Scheme for Incompressible Multiphase Flow and Its Application in Simulation of Rayleigh-Taylor Instability. *Journal of Computational Physics*, **152**:642–663.

[35] Mazzocco, F., Arrighetti, C., Bella, G., Spagnoli, L. & Succi, S. (2000). Multiscale Lattice Boltzmann Schemes: a Preliminary Application to Axial Turbomachine Flow Simulations. *International Journal of Modern Physics C*, **11**:233-246.

[36] Van Der Sman, R.G.M. (1997). Lattice-Boltzmann Scheme for Natural Convection in Porous Media. *International Journal of Modern Physics. C*, **8**:879–888.

[37] d'Humières, D., Lallemand, P. & Frisch, U. (1986). Lattice Gas Models for 3D Hydrodynamics. *Europhys. Lett.*, **2**:291–297.

[38] Balasubramanian, K., Hayot, F. & Saam, W.F. (1987). Darcy's law from lattice-gas hydrodynamics. *Phys. Rev. A*, **36**:2248–2253.

[39] Bhatnagar, P.L., Gross, E.P. & Krook, M. (1954). A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Phys. Rev.*, **94**(3):511–525.

[40] Higuera, F.J. & Jiménez, J. (1989). Boltzmann Approach to Lattice Gas Simulations. *Europhys. Lett.*, **9**:663–668.

[41] Qian, Y.H., d'Humières, D. & Lallemand, P. (1992). Lattice BGK Models for Navier-Stokes Equation. *Europhys. Lett.*, **17**:479–484.

[42] Jin, S., Pareschi, L. & Slemrod, M. (2002). A Relaxation Scheme for Solving the Boltzmann Equation Based on the Chapman-Enskog Expansion. *Acta Mathematicae Applicatae Sinica*, **18**:37–62.

[43] Alcouffe, R.E. (1985). *First collision source method for coupling monte carlo and discrete ordinates for localized source problems.* Los Alamos, NM: Los Alamos National Laboratory.

[44] Lewis, E.E. & Miller, W.F. (1984). *Computational methods of neutron transport.* New York: Wiley.

[45] Buchan, A., Pain, C., Eaton, M., Goddard, A. & Smedley-Stevenson, R. (2008). Linear and Quadratic Hexahedral Wavelets on the Sphere for Angular Discretizations of the Boltzmann Transport Equation. *Nuclear Science and Engineering*, **159**:127–152.

[46] NVIDIA. (2007). *CUDA Programming Guide.* Santa Clara, CA: NVIDIA corp.

[47] Möbius, A.F. (1976). *Der barycentrische Calcul ein neues Hülfsmittel zur analytischen Behandlung der Geometrie.* Hildesheim: Olms.

[48] Meyer, M., Barr, A., Lee, H. & Desbrun, M. (2002). Generalized Barycentric Coordinates on Irregular Polygons. *Journal of Graphics Tools*, **7**:13–22.

[49] Atkinson, K. (1982). Numerical integration on the sphere. *J. Austral. Math. Soc. B*, **23**:332–342.

[50] Kobayashi, K., Sugimura, N. & Nagaya, Y. (2000). *3-D radiation transport benchmark problems and results for simple geometries with void regions.* Paris: Nuclear Energy Agency.

[51] Walters, W.F., O'Dell, R.D. (1981). *Nodal methods for discrete-ordinates transport problems in (x,y) geometries.* Los Alamos, NM: Los Alamos National Labratory.