Master of Science Thesis

Analysis of the Fission Matrix Based Monte Carlo Method for Criticality Simulations

Gustaf Holst

Nuclear Reactor Technology, Department of Physics
Royal Institute of Technology, SE-106 91 Stockholm, Sweden

Stockholm, Sweden 2016
Scientific thesis for the degree of Master of Science (MSc) in the subject area of Nuclear Engineering.

TRITA-FYS-2016:80
ISSN 0280-316X
ISRN KTH/FYS/--16:80--SE

© Gustaf Holst, December 2016
Abstract

Reactor physics Monte Carlo criticality calculations depend on good initial conditions to produce accurate results. One of these initial conditions is the starting positions of simulated neutrons, called the neutron source. To produce accurate results the neutron source should correspond to the neutron density of the given system, however the neutron density is usually unknown prior to the calculation. Therefore the neutron source is estimated and iteratively improved before results are calculated. The purpose of the work presented in this thesis is to reduce the dependence on a correct neutron source estimate. Reducing the dependence allows the iterative process to be skipped hence computational resources to be saved.

Previous studies have shown that Fission Matrix Based Monte Carlo (FMBMC) can produce results that are independent of errors in the neutron source estimate. In FMBMC a mesh is superimposed over the space of the system. The mesh is used to track neutron migration which together with fission yield is stored in a matrix called the Fission Matrix. Because neutron migration is tracked the errors in the neutron source can be compensated for hence result accuracy becomes independent of the neutron source. However sufficient compensation can only be done when the neutron tracking is precise i.e. when the mesh is fine enough. In this thesis simulations are performed to investigate how fine the mesh needs to be for independence to be achieved.

The simulations confirm that, for the tested system, the results become independent of the neutron source. The required mesh fineness is presented and an analysis of matrix properties is used to give recommendations for future simulations. Albeit the results are constrained to the tested system, the potential of FMBMC is demonstrated. It might be possible to extrapolate the results to other systems as well but further work will have to confirm that. In the end recommendations are made for further investigations on the method.

In Chapter 1 a brief historical review and introduction to the Monte Carlo method is presented followed by a summary of recent publications on the topic. Next in Chapter 2 the theory of general Monte Carlo criticality calculations is presented followed by Chapter 3 that establishes the theory of FMBMC and how the fission matrix is calculated and used in practice. In Chapter 4 a methodology is laid out for simulations to be performed. The results of the performed simulations are then presented and discussed and finally in Chapter 5 the thesis is concluded. In the appendices some of the obstacles that were encountered along the way are summarised followed by a programming guide for the supercomputer that was used to carry out the simulations.
Sammanfattning

Monte Carlo-kriticitetsberäkningar i reaktorfysik är beroende av bra begynnelsevillkor för att ge korrekta resultat. Ett av dessa begynnelsevillkor är startpositionerna för de simulerade neutronerna, kallad neutronkällan. För att ge korrekta resultat ska neutronkällan motsvara det givna systemets neutrondensitet, dock är neutrondensiteten vanligtvis okänd före beräkningen. Därför uppskattas neutronkällan först och sedan förbättras den iterativt innan resultaten beräknas. Syftet med arbetet som presenteras i denna avhandling är att minska beroendet av en korrekt uppskattning av neutronkällan. Ett minskat beroendet tillåter den iterativa processen att förbigås och därmed kan beräkningsresurser besparas.

Tidigare studier har visat att Fission Matrix Based Monte Carlo (FMBMC) kan ge resultat som är oberoende av fel i uppskattningen av neutronkälla. I FMBMC läggs ett rutnät över systemet. Rutnätet används för att kartlägga neutronmigration som tillsammans med ”fission yield” lagras i en matris som kallas Fission Matrix. Eftersom neutronmigration kartläggs kan felet i uppskattningen av neutronkällan kompenseras för vilket resulterar i att noggrannheten i resultaten blir oberoende av neutronkällan. Dock kan endast tillräcklig kompensation göras när kartläggningen av neutronerna är exakt dvs när rutnätet är tätt nog. I denna avhandling utförs simuleringar för att undersöka hur tätt rutnätet måste vara för att oberoende ska uppnås.


I kapitel 1 görs en kort historisk tillbakablick och en introduktion till Monte Carlo-metoden presenteras, följt av en sammanfattning av de senaste publikationer i området. Därefter i kapitel 2 läggs teorin för allmänna Monte Carlokriticitetsberäkningar fram följt av kapitel 3 som lägger fram teorin om FMBMC och hur Fission Matix beräknas och används i praktiken. I kapitel 4 läggs en metodik fram för hur simuleringarna som skall utföras. Resultaten av de utförda simuleringar presenteras och diskuteras sedan och slutligen i kapitel 5 summeras avhandlingen. I bilagorna sammanfattas några av de hinder som uppstod längs vägen följt av en programmeringsguide för superdatorn som användes för att genomföra simuleringarna.
I would like to thank my supervisor Jan Dufek for providing me with an interesting topic as well as valuable advice and guidance. My thanks also go to Börge Olsen for our collaborations and for all the valuable discussions. I am also grateful to the rest of the Reactor Technology Division for the guidance and interesting lunch conversations.

Last but not least I would like to thank my fiancee Alex and the rest of my family for enduring this period. It would have been impossible to carry through without your support.
# Contents

Abstract .......................................................................................... I  
Sammanfattning .......................................................................... II  

Acknowledgments .................................................................. III  

Contents ....................................................................................... V  

1 Introduction ........................................................................ 1  
   1.1 Background ....................................................................... 1  
   1.2 Introductory Monte Carlo Methods ................................. 2  
   1.3 Purpose of the Thesis ..................................................... 2  
   1.4 Recent Publications ...................................................... 3  

2 Theory of Monte Carlo Criticality Calculations .................. 5  
   2.1 Quantities of Reactor Physics ......................................... 5  
   2.2 The Neutron Transport Equation ................................. 6  
   2.3 The Eigenvalue Equation .............................................. 8  
   2.4 The Power Iteration ..................................................... 9  
   2.5 Neutron Tracking ...................................................... 10  
   2.6 Error Sources .......................................................... 11  

3 Theory of Fission Matrix Based Calculations ...................... 13  
   3.1 Introduction ............................................................ 13  
   3.2 Definition of the Fission Matrix ................................... 13  
   3.3 Generating the Fission Matrix in practice ..................... 14  
   3.4 Eigenvalue Solvers .................................................... 15  
   3.5 Source Independence ................................................ 16  

4 Analysis of Fission Matrix Based Calculations .................. 17  
   4.1 Introduction ............................................................ 17  
   4.2 Methodology ........................................................... 18  
   4.3 Simulations ............................................................. 19  
   4.4 Results and Discussion ............................................. 21
Chapter 1

Introduction

1.1 Background

The Monte Carlo method is as old as the modern computer itself. Some might even credit Buffon’s needle, dating back as early as the 18th century, as the first Monte Carlo problem [1]. But it would take another two centuries before the name was introduced and the potential of the method realised. The breakthrough of the electrical digital computer enabled simple arithmetic to be carried out faster and on a larger scale than ever before. This meant that for the first time it became viable to simulate single neutrons in numbers sufficient to calculate macroscopic quantities like $k_{\text{eff}}$. One of the earliest computers of its kind, the ENIAC, ran the first Monte Carlo simulation in a Los Alamos project in 1947 [2]. The name Monte Carlo, which is an allusion to the random nature of the method, was introduced by Nicholas Metropolis who was working at Los Alamos at the time [3].

Before the introduction of the Monte Carlo method, deterministic methods were the only available tools to solve reactor physics problems. A great difficulty when utilising these methods is that the mathematics involved rapidly become very complex. To overcome the complexity, the mathematical model is tailored to the specific problem and substantial simplifications are made that in some way or another misrepresent reality. In contrast, the Monte Carlo method can be used on any geometry without the need for any extensive modifications. Neutrons are simply “let loose” in the system and results are calculated as an average of the observed behaviour. To reduce the variance in the average however, a lot of neutrons have to be simulated hence the Monte Carlo method is very computationally demanding. Therefore efforts have been directed at improving the efficiency of the method ever since it was invented.
1.2 Introductory Monte Carlo Methods

Neutrons in Monte Carlo criticality calculations are simulated in iterations, or cycles. The batch of neutrons simulated in one cycle determine the initial positions i.e. the neutron source, or just the source, for the neutrons in the next cycle. Over a number of cycles the spatial distribution of the source will reach a steady-state. Any results e.g. neutron flux and $k_{\text{eff}}$, gathered before the steady-state is reached will contain an error [4], onwards referred to as source error. Therefore results are only gathered during the cycles after the steady-state distribution has been reached, called the active cycles. Cycles before, when the distribution is still converging, are called inactive cycles. It is only during the active cycles that the variance in the results is reduced.

Conventional Monte Carlo methods spend a big share of the allocated resources on improving the initial conditions, in the inactive cycles, before calculation of results can even begin. Furthermore there is no certain way to determine how many inactive cycles are needed. For some systems the convergence of the source is very slow. If the method was less sensitive to errors in the source then more resources could be spent on calculating results.

1.3 Purpose of the Thesis

It has been shown that by using Fission Matrix Based Monte Carlo (FMBMC) [5] useful results can be gathered even during the inactive cycles when the source is still converging. This has three promising implications:

- Results are insensitive to errors in the source so any source can be used.
- All cycles contribute to the results so a higher statistical significance is obtained from the same amount of allocated resources.
- Results from different simulations can be freely combined so the scalability when parallelising simulations increases dramatically [6, 7].

The fission matrix is generated from a mesh superimposed over the geometry and while it can make the results insensitive to errors in the source, that will only happen when its mesh is fine enough [8]. This becomes an optimisation problem because the computational cost of using the fission matrix method increases with finer meshes. The purpose of this thesis is to estimate how fine the mesh needs to be able to remove inactive cycles and still have a negligible source error.
1.4 Recent Publications

The inspiration for this thesis is the paper by J. Dufek and W. Gudowski [5] in which FMBMC is suggested to be used to entirely cancel, instead of just reducing the number of, inactive cycles. Since its publication in 2009 several papers and reports have been published on the topic.

In [7] M. Wenner suggests a method to improve the confidence interval estimation over the plus and minus approach used in [5], the parallelisation potential is also confirmed. In [9] Wenner also investigates the impact of using different initial sources on a given mesh. Wenner shows that complete cancelation of inactive cycles by FMBMC actually give different results depending on the chosen source, suggesting that some source error is still present for some sources. For these cases a few inactive cycles are still needed. The flat source shows the best potential as default guess.

Undersampling, i.e. biases due to an insufficient number of neutrons per batch (batch size), is investigated in [10]. Undersampling is normally detected by performing several simulations with varying batch sizes, but [10] shows that the undersampling can be detected during a single simulation with the aid of the fission matrix. However, FMBMC, i.e. complete cancelation of inactive cycles, is not investigated.

In [11] theoretical and empirical verification that finer meshes indeed reduce source error is presented. A method is also developed for decreasing the storage need by making the fission matrix sparser. The method shows great promise because it enables even finer meshes, i.e. larger fission matrices, to be used. However FMBMC is not investigated.

To conclude many authors have touched upon the different aspects of the fission matrix but few investigate the full potential of FMBMC.
Chapter 2

Theory of Monte Carlo
Criticality Calculations

2.1 Quantities of Reactor Physics

The angular neutron density, defined as $n(r, E, \Omega, t)dV dE d\Omega$, is the expected number of neutrons in volume $dV$ around $r$, within energy interval $[E, E+dE]$, travelling in an angle $d\Omega$ around $\Omega$, at time $t$, see Figure 2.1.

![Figure 2.1. Angular neutron density.](Image)
When the angular neutron density is integrated over all angles the standard
neutron density, or just the neutron density, is obtained as

\[ N(r, E, t) = \int_{4\pi} n(r, E, \Omega, t) d\Omega. \quad (2.1) \]

The angular neutron flux density can be formed

\[ \varphi(r, E, \Omega, t) = vn(r, E, \Omega, t) \quad (2.2) \]

where \( v \) is the neutron speed. Similarly the angular neutron current density is
formed

\[ j(r, E, \Omega, t) = v n(r, E, \Omega, t) = \Omega \varphi(r, E, \Omega, t) \quad (2.3) \]

where \( v \) is the neutron velocity.

To form the neutron transport equation in the next section Gauss’s theorem
is used. It is defined as follows

\[ \int_A j(r, E, \Omega, t) \cdot dA = \int_V \nabla \cdot j(r, E, \Omega, t) dV \quad (2.4) \]

and relates the current, \( j \), through a surface area, \( A \), to the divergence, \( \nabla \), of \( j \) in
the volume, \( V \), contained in \( A \) [12].

### 2.2 The Neutron Transport Equation

The neutron transport equation is now formed by setting up a balance of the rate
of change of neutrons in a volume \( V \)[13]:

\[ \frac{\partial}{\partial t} \left[ \int_V n(r, E, \Omega, t) dV \right] dEd\Omega = G_f + G_i - L_l - L_i + \int_V S dV \quad (2.5) \]

where the left hand side is the rate of change of the integrated neutron density over
the whole volume \( V \) and \( G_f \) is the gain of neutrons due to fission, \( G_i \) is the gain
due to interactions, \( L_l \) is the loss due to leakage, \( L_i \) is the loss due to interaction
and \( S \) is an external source. All of these terms will be handled separately below.
But first we note that if it is assumed that the volume does not change over time,
the time derivative can be moved inside the integral. The left hand side can also
be rewritten in terms of \( \varphi \) by using the relations described in Eq. 2.2:

\[ \frac{\partial}{\partial t} \left[ \int_V n(r, E, \Omega, t) dV \right] dEd\Omega = \int_V \frac{\partial n}{\partial t} dV dEd\Omega = \left[ \int_V \frac{1}{v} \frac{\partial \varphi}{\partial t} \right] dEd\Omega \quad (2.6) \]

Below, the terms of the right hand side of Eq. 2.5 are discussed. To begin with
the loss due to interaction is described as

\[ L_i = \int_V \Sigma_i(r, E, t) \varphi(r, E, \Omega, t) dV \quad (2.7) \]
2.2. The Neutron Transport Equation

where $\Sigma_t$ is the total macroscopic cross section including both absorption and scattering out of the energy $E$ and direction $\Omega$. Next the leakage out of the volume is described

$$L_l = \int_A j(r, E, \Omega, t) \cdot dA \quad \text{Eq. 2.4}$$

Using the relations described in Eq. 2.3, $L_l$ can be further evaluated to

$$\int_V \nabla \cdot j(r, E, \Omega, t) dV = \int_V \nabla \cdot \left[ \frac{\Omega}{\Omega} \varphi(r, E, \Omega, t) \right] dV = \int_V \nabla \varphi(r, E, \Omega, t) dV \quad \text{(2.9)}$$

Now that all the losses of neutrons over time in volume $V$ are considered, let’s continue to the terms that add to the number of neutrons. The gain of neutrons into the energy and direction under consideration, $E$ and $\Omega$, due to fissions caused by neutrons of any energy $E'$ and direction $\Omega'$ is described

$$G_f = \int_V \frac{\chi(E)}{4\pi} \int_0^\infty \int_{4\pi} \nu(E') \Sigma_f(r, E', t) \varphi(r, E', \Omega', t) d\Omega' dE' dV \quad \text{(2.10)}$$

where $\chi$ is the energy distribution of the fission neutrons (which angular distributions are assumed to be isotropic), $\Sigma_f$ is the macroscopic cross section for fission and $\nu$ is the number of neutrons released from each fission event. Neutrons can also be gained into the volume by interaction of neutrons of other energies $E'$ or directions $\Omega'$ than those under consideration ($E$ and $\Omega$), which is described as

$$G_i = \int_V \int_0^\infty \int_{4\pi} \Sigma_s(r, E' \rightarrow E, \Omega' \rightarrow \Omega, t) \varphi(r, E', \Omega', t) d\Omega' dE' dV \quad \text{(2.11)}$$

where $\Sigma_s$ is the macroscopic cross section for scattering.

The complete neutron transport equation can now be formed by putting all terms together. Since the equation should be valid for any arbitrary volume $V$ the volume integrals are left out

$$\frac{1}{v} \frac{\partial \varphi}{\partial t} = \frac{\chi(E)}{4\pi} \int_0^\infty \int_{4\pi} \nu' \Sigma'_f \varphi' d\Omega' dE'$$

$$+ \int_0^\infty \int_{4\pi} \Sigma'_s \varphi' d\Omega' dE' - \Omega \cdot \nabla \varphi - \Sigma_t \varphi + S \quad \text{(2.12)}$$

where $\nu' = \nu(E')$, $\Sigma'_f = \Sigma_f(r, E', t)$, $\varphi' = (r, E', \Omega', t)$, $\Sigma'_s = \Sigma_s(r, E' \rightarrow E, \Omega' \rightarrow \Omega, t)$, $\varphi = (r, E, \Omega, t)$ and $\Sigma_t = \Sigma_t(r, E, t)$. Accompanied with the right initial and boundary conditions the state of the reactor can now be described by Eq. 2.12. In its current form it is however extremely hard to solve due to the intricate geometry of a reactor core and complex shape of cross section energy dependencies, and it does not even consider delayed neutrons, a paramount parameter for any dynamics analysis.
Chapter 2. Theory of Monte Carlo Criticality Calculations

2.3 The Eigenvalue Equation

When steady state ($\frac{\partial \phi}{\partial t} = 0$) and no external source ($S = 0$) is assumed, Eq. 2.12 can be rewritten

$$\Omega \cdot \nabla \varphi + \Sigma_t \varphi = \frac{\chi(E)}{4\pi} \int_0^\infty \int_{4\pi} \nu' \Sigma'_f \varphi' d\Omega' dE' + \int_0^\infty \int_{4\pi} \Sigma'_s \varphi' d\Omega' dE'. \quad (2.13)$$

The operator $L$ is introduced for the net loss of neutrons

$$L\varphi = \Omega \cdot \nabla \varphi + \Sigma_t \varphi - \int_0^\infty \int_{4\pi} \Sigma'_s \varphi' d\Omega' dE'. \quad (2.14)$$

and operator $G$ for the net gain of neutrons

$$G\varphi = \frac{\chi(E)}{4\pi} \int_0^\infty \int_{4\pi} \nu' \Sigma'_f \varphi' d\Omega' dE'.$$

Equation 2.12 is then reformulated in terms of $L$ and $G$ accordingly [14]:

$$L\varphi = \frac{1}{k} G\varphi \quad (2.15)$$

where $1/k$ has been introduced in order for the equation to be able to represent sub- and supercritical systems. Further let $H\varphi = G\varphi L^{-1} \varphi$, then Eq. 2.15 reduces to[15]:

$$k \varphi = H \varphi \quad (2.16)$$

It is now obvious that what we have in front of us is an eigenvalue problem. While the above work did not make it easier to solve analytically or numerically, it set a foundation for Monte Carlo methods to be used as solvers.

It is important to realise how $k$ and $H$ are related. Assume that $\varphi$ is the solution of Eq. 2.16 and at steady state. Since $k$ is defined as the growth of the neutron population between generations, $k\varphi$ must be the next generation of $\varphi$. Hence, $H\varphi$ must also be the next generation of $\varphi$, according to Eq. 2.16. In other words operating $H$ on $\varphi$ represents applying the properties of the system onto $\varphi$, which includes everything from geometry to materials, to obtain the next generation of $\varphi$. 


2.4 The Power Iteration

In the previous section the eigenvalue equation was formed, but because the neutron density \( s(r, E) \) is a more readily available quantity in Monte Carlo simulations and it relates closely to the neutron flux \( \varphi \), Eq. 2.16 is rewritten into:

\[
ks(r, E) = Hs(r, E)
\]  

(2.17)

where \( k \) is the multiplication factor, \( s(r, E) \) is the steady state neutron density at \( r \) and \( E \) and

\[
Hs(r, E) = \int_0^\infty dE' \int_V d^3r' f(r', E' \rightarrow r, E)s(r', E')
\]  

(2.18)

where \( f(r', E' \rightarrow r, E)dEd^3r \) is the number of first generation fission neutrons produced in the volume element \( d^3r \) at \( r \), in the energy element \( dE \) at \( E \) resulting from fission neutrons born at \( r' \) with energy \( E' \), and \( V \) denotes the space of the system [5]. The equation is solved by the power iteration with the neutron source \( s^{(n)} \) iterated over \( n \) iterations (cycles):

\[
s^{(n)} \leftarrow \frac{Hs^{(n-1)}}{k_{\text{eff}}^{(n-1)}}, \quad n = 1, 2, ...
\]  

(2.19)

where \( Hs^{(n-1)} \) is randomly sampled, and \( k_{\text{eff}}^{(n-1)} \) is an estimate of the fundamental mode eigenvalue \( k_{\text{eff}} \) obtained in cycle \( n \) as:

\[
k_{\text{eff}}^{(n)} = \frac{\langle Hs^{(n-1)} \rangle}{\langle s^{(n-1)} \rangle}.
\]  

(2.20)

In Eq. 2.20, \( \langle Hs^{(n-1)} \rangle \) and \( \langle s^{(n-1)} \rangle \) are the total statistical weights of \( Hs^{(n-1)} \) and \( s^{(n-1)} \) respectively calculated as [5]:

\[
\langle s(r, E) \rangle = \int_0^\infty dE \int_V d^3r s(r, E).
\]  

(2.21)
2.5 Neutron Tracking

To sample $H_s^{(n-1)}$ in Eq. 2.19, neutrons or neutron histories are simulated in the given system. A neutron history is the whole lifetime of a neutron from its birth to its death.

2.5.1 Analog Simulations

In analog simulations the notion of analog refers to how neutrons in the simulated system are somewhat analogous to real neutrons [16]. The following algorithm is used to simulate the neutron histories:

1. The neutron is born.
2. The neutron speed and direction are sampled.
3. The neutron moves a distance $S$ to the next collision.
4. The neutron collides.
   a. It either scatters, go back to 2
   b. or fissions/is absorbed/leaks out, it dies.

In step 1 the initial position of the neutron is fetched from the neutron source ($s^{(n-1)}$ in Eq. 2.19). A pseudorandom number generator is utilised in step 2 to sample the speed and direction, in step 3 to sample $S$ and in step 4 to sample type of interaction. After step 4b the position of the neutron is saved in the neutron source for the next cycle $s^{(n)}$.

The probability of a neutron to interact with particles in the medium it travels through is independent of the distance travelled, which makes the distance between two collisions, $S$, exponentially distributed. With the uniformly distributed random variable $\xi$, obtained from a pseudorandom number generator, the distance is calculated as:

$$S = -\frac{1}{\Sigma_T(E)} \log \xi$$  \hspace{1cm} (2.22)

where $\Sigma_T(E)$ is the total macroscopic neutron cross section for that medium [17].

If the next collision is calculated to be in a medium with a different $\Sigma_T(E)$ than the current medium Eq. 2.22 is no longer valid. The neutron is then moved back, along its path, to the boundary between the two mediums. The path is then resampled from there but with $\Sigma_T(E)$ of the new medium.
2.6. Error Sources

2.5.2 Non-Analog Simulations

In non-analog simulations, in contrast to analog simulations, some part of the simulation has been altered in such a way that it no longer represents real neutrons in the same fashion [16]. For Monte Carlo neutron transport the alteration lies in the interactions of neutrons with matter.

In steps 2 to 4 the outcomes of the samplings affect a weight associated with the neutron instead of altering the whole neutron history. In that way it is possible to follow neutron histories well beyond the point at which it would normally be terminated, by lowering the weight instead. At each step the neutron source for the next generation is built up instead of exclusively at termination.

At some point the weight will fall below a predetermined value. A game of Russian roulette is then played to determine the fate of the neutron. Depending on the outcome, it is either terminated or the weight is increased so that the neutron history can continue. The virtue of the method is that regions of the reactor with very low neutron densities are still simulated.

2.6 Error Sources

2.6.1 Statistical Error

In neutron transport Monte Carlo a finite number, $h$, of neutron histories are sampled in order to calculate key results about the system, e.g. $k_{\text{eff}}$ and the neutron distribution. Like with all sampling of random variables a statistical error is always present in the results that depend on the number of samples taken. The size of the statistical error is dictated by the central limit theorem [16] and is on the order of $1/\sqrt{h}$.

2.6.2 Systematic errors

Undersampling

Unlike photon transport Monte Carlo, neutrons do not only scatter and absorb but also multiply i.e. fission. This has the effect that in a super critical system the neutron population grows exponentially, and respectively in a sub critical system the neutron population decays exponentially. The problem with a decaying neutron population is that eventually there will be no more neutrons to simulate but also the opposite is undesirable. If the exponential growth is not accommodated for in some way the computational time for each subsequent generation will grow exponentially.

The remedy for both problems is to periodically normalise the neutron population by sampling a predetermined value of $M$ new neutrons, the batch size, out of from the previous cycle. In this way the population stays constant across cycles. The normalisation does however introduce a bias of its own, called undersampling, that is related to the batch size. The bias from undersampling, which is on the
order of $1/M$ [18, 19], does not decay over subsequent cycles. It is a direct con-
sequence of the rules of which the normalisation process chooses the next cycle of 
neutrons, which are often theoretically unjustified [20].

**Source Error**

The initial guess $s^{(0)}$ for Eq. 2.19 has to be done by the user and because the steady 
state neutron source, $s(r, E)$, is not known in advance, it has to be estimated. A 
common procedure is to distribute the neutrons evenly in the fissionable material 
[21]. Between each cycle, $s^{(n)}$ is adjusted according to the fission sites of the 
previous cycle which leads to a concentration of neutrons in more reactive areas of 
the system. This process proceeds over many cycles until $s^{(n)}$ has reached steady 
state and therefore is representative for $s(r, E)$.

If any results are tallied before steady state has been reached those results will 
have an error, referred to as *source error*. The error comes from the mismatch 
between $s(r, E)$ and $s^{(n)}$. When e.g. $k_{\text{eff}}$ is estimated through Eq. 2.20 an average 
is made over all neutrons, without regard to their birth place. However because $s^{(n)}$ 
is not converged the concentration of neutrons in reactive areas is too small and 
vice versa for less reactive areas. The consequence is that the average neutron in 
$s^{(n)}$ is born in a less reactive area than the average neutron in $s(r, E)$ and Eq. 2.20 
therefore underestimates $k_{\text{eff}}$. The same is true for any tallied result like detector 
response etc.

In integral estimates that make averages over all cycles the source error will 
diminish as $n$ grows. However because the error is largest in the first cycles, these 
cycles are usually left out of the estimate and are therefore referred to as inactive 
cycles. Cycles used in tallies are referred to as active cycles.

In [4] it is suggested that neutrons should be weighted depending on where they 
are born. This enables a method of counteracting the source error and it is the basis 
for the fission matrix which will be described in the next chapter. By recording and 
saving their birthplace, all neutrons can contribute to the estimate of the results as 
long as they are properly weighted in the estimate.
Chapter 3

Theory of Fission Matrix Based Calculations

3.1 Introduction

In FMBMC [5, 8] a more direct approach at solving the eigenvalue equation (Eq. 2.16) is taken. The operator $H$ is actually determined which opens the possibility to solve for $k$ and $s(r, E)$ as the fundamental mode eigenvalue and eigenvector of $H$ instead of using Eq. 2.20. The advantage of the method is that the source error, originating from insufficient mapping of neutron migration, can be minimised.

To set up the eigenvalue problem, $H$ is discretised into the finite $N$ by $N$ Fission Matrix, $H$, and the space of the system is divided into $N$ zones. The fission matrix $H$ is then built up by recording fission yield from neutron migration inside and between the zones.

3.2 Definition of the Fission Matrix

The $(i,j)^{th}$ cell in the fission matrix $H$ represents the average number of neutrons born from fissions in zone $Z_i$, induced by neutrons born in zone $Z_j$, per neutron in $Z_j$. It is cell-wise defined in cycle $n$ as

$$H^{(n)}[i,j] = \frac{\int_0^\infty dE dE' \int_{Z_i} d^3r \int_{Z_j} d^3r' f(r', E' \rightarrow r, E) s^{(n-1)}(r', E')}{\int_0^\infty dE' \int_{Z_j} d^3r' s^{(n-1)}(r', E')} , \forall i,j \quad (3.1)$$

where $f(r', E' \rightarrow r, E)dEdr$ is already defined in Eq. 2.18 and $s^{(n-1)}(r, E)$ is the fission source at cycle $(n - 1)$ [15].

13
3.3 Generating the Fission Matrix in practice

When the fission matrix is computed in a simulation, the fission matrix is formed by two entities; the cumulative source vector, \( \mathbf{v} \), and the cumulative matrix, \( \mathbf{C} \). Let \( \mathbf{v}^{(n)} \) be the statistical weight of all neutron histories at birth, summed over all cycles up until and including cycle \( n \), then

\[
\mathbf{v}^{(n)}[j] = \sum_{l \in s_j} w_{l,0}
\]

(3.2)

where \( w_{l,0} \) is the statistical weight of neutron history \( l \) at birth and \( s_j \) is a set of all neutron histories in the source originating in zone \( j \). Next let \( \mathbf{C}^{(n)} \) be the collected weight of all neutron histories over all cycles up until and including \( n \), then

\[
\mathbf{C}^{(n)}[i,j] = \sum_{l \in s_{i,j}} w_{l,c}
\]

(3.3)

where \( w_{l,c} \) is the weight of neutron history \( l \) after collision \( c \) and \( s_{i,j} \) is a set of all neutron histories causing fissions in zone \( i \), that were born in zone \( j \). To form the fission matrix, \( \mathbf{C} \) is normalised by \( \mathbf{v} \) accordingly:

\[
\mathbf{H}^{(n)}[i,j] = \frac{\mathbf{C}^{(n)}[i,j]}{\mathbf{v}^{(n)}[j]}.
\]

(3.4)

The weight \( w_{l,0} \) is sampled from the neutron source and is assigned to neutron history \( l \) at birth, i.e. at \( c = 0 \). \( w_{l,c} \) is accumulated over the course of the simulation of \( l \), described in Section 2.5, as

\[
w_{l,c} = f_{l,c} w_{l,c-1}, \forall c > 0
\]

(3.5)

where \( f_{l,c} \) is the average number of fission neutrons produced in collision \( c \) calculated as

\[
f_{l,c} = \frac{\sum_{q \in s_{l,c}} a_q \bar{\nu}_q \sigma_{f_q}}{\sum_{q \in s_{l,c}} a_q \sigma_{\text{tot}_q}}
\]

(3.6)

where \( q \) is a nuclide in the set \( s_{l,c} \) of all nuclides at the point of collision \( c \) for neutron history \( l \) and \( a_q \) is the atomic fraction of \( q \). At the incident neutron energy of \( l \) at \( c \), \( \bar{\nu}_q \) is the average number of neutrons produced per fission in \( q \), \( \sigma_{f_q} \) is the microscopic cross section for fission in \( q \) and \( \sigma_{\text{tot}_q} \) is the total microscopic cross section in \( q \) [15].
3.4 Eigenvalue Solvers

To extract important quantities e.g. $k_{\text{eff}}$ and the power distribution from the fission matrix, eigenvalue algorithms are used. Fission matrices, especially of larger dimensions, are sparse and therefore two algorithms are of special interest, namely the Power Method and Arnoli’s Method.

**Power Method**

The most straightforward approach is to use the power method where an arbitrary nonzero vector, $v_0$, is iteratively multiplied with the matrix of interest, $A$, and normalised. The vector converges to the eigenvector associated with the eigenvalue of largest modulus, $\lambda_1$, and $h_j$ converges to $\lambda_1$. The algorithm is computed as follows:

*Start:* Choose a nonzero initial vector $v_0$.

*Iterate:* for $j = 1, 2, \ldots$ until convergence, compute:

$$ v_j = Av_{j-1}/h_j $$

where $h_j = \|Av_{j-1}\|$. While simple and powerful the algorithm can be unstable and more specifically will not converge when $v_0$ has no component in the invariant subspace associated with $\lambda_1$ [22]. Furthermore the method is slow for large sparse matrices.

**Arnoldi’s Method**

The instability problem is solved by ensuring that the consecutive vectors $v_0, v_1, \ldots, v_j$ form an orthonormal basis. The orthonormalisation is done with the Gram-Schmidt process. The method is also faster for large sparse matrices than the power method. The algorithm is computed as follows:

*Start:* Choose an initial vector $v_1$ of norm 1.

*Iterate:* for $j = 1, 2, \ldots, m$ compute:

$$ h_{ij} = (Av_j, v_i), i = 1, 2, \ldots, j, $$

$$ w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i, $$

$$ h_{j+1,j} = \|w_j\|_2 $$

$$ v_{j+1} = w_j/h_{j+1,j}. $$
The iteration breaks down if \( h_{j+1,j} = 0 \) in Eq. 3.10. However in this case the eigenvalue and eigenvector approximations, \( h_{j+1,j} \) and \( v_{j+1} \), exactly represent the eigenvalue and associated eigenvector [22].

### 3.5 Source Independence

The fundamental mode eigenvalue and eigenvector of the fission matrix gives approximations of \( k_{\text{eff}} \) and the power distribution. It is however dependent on the fundamental mode fission source \( s_0 \) (see Eq. 3.1) in contrast to \( H \) (see Eq. 2.18), i.e. it is affected by source error. The dependence comes from the space discretisation. When the number of zones approaches infinity (i.e. \( Z_j \to 0, \forall j \)) the fission matrix becomes independent of the fission source as showed in [5, 11]. Assuming that the energy distribution of the fission source is correct, when the number of zones goes to infinity the fission matrix can be correctly calculated independently of the fission source. This makes the convergence of the source unnecessary.

In practice the number of zones can never be infinite but at some point the error introduced by an incorrect source will be shadowed by the statistical noise, which is always present in Monte Carlo calculations. There is therefore a finite number of zones that is sufficient for source independence. The practical limitation is rather that the fission matrix grows as the square of the number of zones. When the fission matrix gets too big, calculating its eigenvalue becomes more expensive than the inactive cycles, made unnecessary by it, were in the first place.
Chapter 4

Analysis of Fission Matrix Based Calculations

4.1 Introduction

It was shown in [5, 11] that source independence is theoretically achievable when the fission matrix mesh is fine. It is however unknown how fine it has to be to attain this feature in practice. In this chapter simulations are presented that try to answer that question.

There are several different aspects of Monte Carlo simulations that have to be taken into account. What source independence promises to cancel out is the source error (Section 2.6.2), but there are other errors that might be present as well e.g. statistical errors and undersampling, that also affect the results. An unambiguous way of measuring error has to be developed so that the simulations can be compared to a given reference in a quantitative manner. It is expected that finer fission matrix meshes will give a smaller source error. In the next section a methodology is presented that tries a take all of these considerations into account.
4.2 Methodology

Several different cases with varying mesh coarseness are simulated, hereafter called standard cases. In all cases the first generation neutrons are distributed evenly in the fissionable material i.e. a flat source is chosen as the initial guess. This is a common procedure when the steady state neutron density is unknown [21], which it most often is. It also ensures that an incorrect source is present that induces source error. This enables measurement of the source error dependency on the mesh because different convergence profiles should be observed for the different cases.

To accurately capture the source error, only the cycles affected by the error are simulated. The source error is dominantly present in the first tens up to thousands of cycles depending on the system [19]. Test runs of the given system have to be simulated until convergence is observed to determine how many cycles are affected. Only these cycles are simulated to avoid smearing out the results.

In order to avoid undersampling the batch size has to be sufficient. It is recommended to have a batch size of a thousand neutrons or more [19], depending on the system, to avoid undersampling. The batch size should therefore be well over 10 000 to be on the safe side.

Statistical error is always present in Monte Carlo simulations. To isolate the source error from statistical error an ideal case is run for each standard case. The ideal case has the same statistics as the standard case but it has a correct initial source, i.e. the initial source is proportional the steady state neutron density. In this way a case that is truly free from both undersampling and source error is ensured. It is used as a measurement of the statistical error present with the given statistics in the given system and denotes a minimum error that will be present independently of biases.

The correct initial source for the ideal case is sampled from the steady state neutron density of a reference case that is run on the same system but with better statistics. The reference case enables quantitative measurement of the error in the different cases as it can be viewed as the correct answer that the other cases can be compared against. It is run with inactive cycles as a standard Monte Carlo simulation to ensure that it is not contaminated by source bias.

A quantity has to be selected to study the performance of the method. Two obvious choices are the fundamental mode eigenvalue and its corresponding eigenvector. In [19] and [23], Brown pointed out that the convergence rates of the fundamental mode eigenvalue and eigenvector differ. To get a complete picture of the convergence of the system both should be monitored. While measurement of error in vectors of equal dimension is possible, difficulties arise when vectors of different dimension i.e. the different standard cases, are compared to one another, see Appendix B. Normalisations of vectors of different dimensions introduce biases that distort the results. Therefore only the eigenvalue is monitored as it is one-dimensional in all cases and therefore more easily comparable.
4.3 Simulations

4.3.1 Setup

Simulations were run on the Tegnér supercomputer\(^1\) (1800 core, 100 teraflops). The simulations were performed using a proprietary non-analogue continuous energy Monte Carlo criticality code. The code Serpent\(^2\) was first used but the implementation of the fission matrix appeared incorrect in it, see Appendix A.

The multi-core nature of the computer enabled the standard case, ideal case and reference case to be run a large number of times per case which minimised statistical errors. It did however also mean that data from different runs for the same case would have to be combined without distorting the results. For more details on the implementation, see Appendix C.

For each case, except the reference case, the fission matrix, \(H^{(n)}_r\), from cycle \(n\) and run \(r\) is combined from \(N\) runs into

\[
H^{(n)} = \frac{1}{N} \sum_{r=1}^{N} H^{(n)}_r
\]

(4.1)

which results in the combined fission matrix, \(H^{(n)}\), for cycle \(n\) for the specific case. The fundamental mode eigenvalue, \(\lambda_{H^{(n)}}\), of the combined fission matrix is then calculated with *Arnoldi's method* for meshes with 100 zones or more and the *power method* for coarser meshes, see Section 3.4. The reference case fundamental mode eigenvalue, \(\lambda_R\), is calculated accordingly

\[
\lambda_R = \frac{1}{N} \sum_{r=1}^{N} \lambda_{R_r}
\]

(4.2)

where \(\lambda_{R_r}\) is the eigenvalue of the fission matrix, \(R_r\), from the last cycle of run \(r\) of the reference case. \(\lambda_{R_r}\) of \(R_r\) is calculated with *Arnoldi’s method*.

The error in the eigenvalue, \(\delta^{(n)}\), at cycle \(n\) is then calculated with

\[
\delta^{(n)} = \frac{|\lambda_{H^{(n)}} - \lambda_R|}{\lambda_R}
\]

(4.3)

for each case. The formula is used both for the standard cases and the ideal cases.

For the ease of discussion a simplified error analysis equation is proposed for \(\delta\):

\[
\delta = \delta_{\text{statistical}} + \delta_{\text{undersampling}} + \delta_{\text{source}}
\]

(4.4)

where \(\delta_{\text{statistical}}\) is the contribution to the total error from the statistical error, \(\delta_{\text{undersampling}}\) is from undersampling and \(\delta_{\text{source}}\) is from the source bias. While it is impossible to explicitly separate or eliminate any particular error the equation facilitates the discussion in Section 4.4.

---

\(^1\)https://www.pdc.kth.se/resources/computers/tegner

\(^2\)http://montecarlo.vtt.fi
Chapter 4. Analysis of Fission Matrix Based Calculations

4.3.2 System Description

The parameters of the system are those of a typical PWR [24]. The simulated system is a single 3 meter fuel rod with Zn cladding, surrounded by light water. For more details on the geometry see Table 4.1. Reflective boundary conditions are set radially to emulate an infinite lattice. Black boundary conditions are set axially to enable neutrons to leak out. This induces a cosine-like neutron distribution. Because a flat source is chosen as the initial guess for the neutron distribution, an error in the source that needs to be converged is ensured.

<table>
<thead>
<tr>
<th>Table 4.1. Description of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rod length</td>
</tr>
<tr>
<td>Fuel pellet diameter</td>
</tr>
<tr>
<td>Cladding thickness</td>
</tr>
<tr>
<td>Pitch</td>
</tr>
<tr>
<td>Moderator</td>
</tr>
<tr>
<td>Water density</td>
</tr>
<tr>
<td>Fuel</td>
</tr>
<tr>
<td>Enrichment</td>
</tr>
<tr>
<td>Cladding material</td>
</tr>
<tr>
<td>Initial neutron distribution</td>
</tr>
</tbody>
</table>

4.3.3 Standard Cases

The aim of the simulation is to measure the source error. Therefore no inactive cycles are simulated for the standard cases. Instead results are tallied over all cycles, knowing a source bias will be present and hopefully measurable. Eight different cases are investigated with meshes of 1, 3, 5, 10, 25, 50, 100 and 500 zones respectively. Each case is simulated over 1008 runs of 500 cycles with 50 000 neutrons in each cycle granting a total of $2.5 \times 10^{10}$ neutron histories. Throughout the simulation $H_r^{(n)}$ is recorded, see Section 4.3.1, to enable monitoring of the cycle to cycle progression of $\lambda_H$.

4.3.4 Reference Case

The statistical error of Monte Carlo simulations die off as $1/\sqrt{H}$, see Section 2.6.1. In order to decrease the error by a factor of 10, a reference with 100 times more neutron histories than the standard case is needed. The initial source distribution of the reference case is converged over 11300 inactive cycles. After the inactive cycles the source distribution is reflected to reduce the statistical error, which is
possible because the geometry is symmetric. The results are tallied over 120 runs of $2 \times 10^4$ active cycles with $2 \times 10^6$ neutrons in each cycle, granting a total of $4.8 \times 10^{12}$ neutron histories.

To calculate the fundamental mode eigenvalue ($\lambda_R$) and corresponding eigenvector a 1000 zone mesh is used. In the end of each of the 120 runs $\lambda_{Rr}$ is calculated and combined into $\lambda_R$, see Section 4.3.1.

4.3.5 Ideal Cases

To separate the source error in the eigenvalue from the statistical errors an ideal case with a converged source is simulated, for each standard case. The ideal case is simulated over 1008 runs of 500 cycles with 50 000 neutrons in each cycle, for every standard case. The fundamental mode eigenvalue is calculated in the same way as for the standard cases, see Section 4.3.1.

The source is sampled from the reference case, which is a good approximation of the correct source. For each ideal case each of the 1008 runs have a batch size of 50 000 resulting in $5.04 \times 10^7$ neutrons. The birth places of these neutrons are sampled from the end positions of 120 runs of $2 \times 10^6$ neutrons from the reference case runs resulting in $2.4 \times 10^8$ birth places. This means that no birth place has to be used twice within an ideal case.

4.4 Results and Discussion

The progression of the fundamental mode eigenvalues of the standard cases and ideal case are presented in Figure 4.1 together with the final result of the reference case eigenvalue ($\lambda_R$). Note that for the reference case only the final value is presented, not the whole progression. Some standard cases have been left out of the figures for increased readability. The ideal cases all gave the same curve and have therefore been summarised into one curve.

As mentioned in Section 2.6.2 the source error underestimates $k_{\text{eff}}$ in standard Monte Carlo when a flat source is used as the initial guess. This is clearly illustrated in Figure 4.1 by the difference between the standard case with 1 zone and the ideal cases. The only difference between these cases is how converged the source is. The shapes are the same but the error is much smaller in the ideal case. Expressed in Eq. 4.4 this can be summarised by saying that $\delta_{\text{statistical}}$ and $\delta_{\text{undersampling}}$ is the same but $\delta_{\text{source}}$ is different.

Between the standard cases (1, 10 and 100 zone cases) it is clear that there is a correlation between number of zones and $\delta$. All of these cases use the flat source and have the same statistics (number of cycles, neutrons per cycle and runs) but the error is smaller for meshes with more zones. The contribution from $\delta_{\text{statistical}}$ and $\delta_{\text{undersampling}}$ is the same so the remaining part of $\delta$ must come from $\delta_{\text{source}}$. It is unexpected that the 100 zone case is even closer to $\lambda_R$ than the ideal cases, a detailed view of Figure 4.1 around $\lambda_R$ is presented in Figure 4.2.
Chapter 4. Analysis of Fission Matrix Based Calculations

Figure 4.1. Comparison of eigenvalue convergence. The cases represented by black lines start with a flat source and have meshes with 1, 10 and 100 zones. The Ideal cases is a representative curve for 8 cases, with meshes of 1 to 500 zones, that start with a converged source. The ideal cases all showed the same convergence pattern.

Figure 4.2 is a zoom of Figure 4.1 in the vicinity $\lambda_R$. Standard cases 25, 50 and 500 are now shown. In the light of the additional standard cases a pattern is starting to emerge. The source bias is indeed decreasing for finer meshes but somewhere between 25 and 50 zones, $\lambda$ at cycle one goes from being underestimated to overestimated. With meshes of 50 zones or more $\lambda$ is, at cycle one, increasingly overestimated.

Because the ideal cases only contain statistical errors it should be impossible to get a smaller error. The hypothesis, to explain how any standard case can have a smaller error, is that fine fission matrices not only reduce the source error but also overestimate $\lambda$. The hypothesis is supported by the fact that meshes with 50 zones or more actually do overestimate $\lambda$. In other words the reason meshes with 10 zones and more can reduce the error, even below the limit for statistical error, is a combination of a reduction of the source error and a tendency to overestimate instead of underestimate $\lambda$. 
4.4. Results and Discussion

Figure 4.2. Zoom of Figure 4.1 in the vicinity of the reference case. Cases with meshes of 25, 50 and 500 zones are now shown. To avoid cluttering the plot the 100 zone case is hidden. Red curves are cases with a converged source, black curves are cases with a flat source.

In Figure 4.3 the convergence rate of $\lambda$ is presented. The cycle by cycle progression of $\lambda$ has been evaluated with Eq. 4.3 which results in $\delta$. Again some cases have been left out to retain the readability. A reference line is drawn with the convergence rate $1/n$, where $n$ is the number of simulated cycles, that starts at $\delta$ for the ideal cases at cycle one. It is observed that the convergence rates of all cases tend towards this trend, it is therefore presented in Figure 4.3 as a guideline. Note however that no suggestion towards $1/n$ as a significant convergence rate has been found in the literature.

There are several considerations to take into account. To begin with $\delta$ for standard cases 50 and 100 is initially smaller than $\delta$ for the ideal case. The reason for this is the overestimation of $\lambda$. Eventually (at cycle 18 or so) the estimation of the 50 zone standard case is worse than that of the ideal cases. The 100 zone standard case however stays constantly below the ideal cases (as well as the 500 zone case which is not shown in the figure). Next the convergence rate is tending
Chapter 4. Analysis of Fission Matrix Based Calculations

Figure 4.3. Comparison of the progression of the relative error ($\delta$) relative to the number of simulated cycles ($n$). The 1, 50 and 100 zone cases (black curves) have flat sources. The red dashed curve has a converged source. The red solid curve is a reference line for the convergence rate $1/n$.

Towards $1/n$, which is very obvious for the ideal cases along which the reference line has been aligned. But it is the general trend for the other cases as well. Interestingly the 1 zone standard case initially has a worse than $1/n$ convergence rate but then speeds up and perfectly align to the reference line. The same is true for all cases but it is not as easily detectable for the cases where $\lambda$ is initially overestimated. Lastly the sharp valleys for the 50 and 100 zone cases at cycle 2 and 5 are when the $\lambda$ estimate crosses $\lambda_R$. At these crossings (in Figure 4.2) $\delta$ is momentarily 0, which gives misleading behaviour in Figure 4.3 where it gives an impression of a much lower than actual error.

To conclude there is a strong correlation between finer meshes and a decreased $\delta$, most clearly visible in Figure 4.1. This indicates that the source error gets reflected less into the fission matrix for finer meshes. However, there is also a tendency for finer meshes make an overestimation of $\lambda$. Albeit, this tendency can not be derived solely to the finer meshes. Since it did not show up in the ideal cases, even for
the 500 zone case, it has to be related to the flat source in combination with finer meshes.

Nonetheless, in this system an overestimation was made by finer meshes and, underestimation by coarser meshes. This is the key to reducing the error even beyond what should be possible, the ideal cases. At some number of zones the estimation will go from an under- to overestimation which will give an impression of a very small error. The standard case with the smallest $\delta$ at cycle one is the 50 zone case, it is however not consistently the best estimate of $\lambda$. It just happens to be in a sweet spot for the system at hand.

![Figure 4.4](image.png)

**Figure 4.4.** Comparison of largest element in the fission matrix for different number of zones. The value of the most optimal, 100 zone, case is marked out.

That said the overestimation of the finer meshes is magnitudes smaller than the underestimation of the coarser meshes, see Figure 4.1. The overall recommendation is therefore to have a finer rather than a coarser mesh. Suggestions have been made [6] that the value of the largest element in the fission matrix might be a good indicator of which mesh should be used. The value of the largest element not only depends on the spacial dimensions of the zones but also on the diffusion length
of the neutrons in the given system. In Figure 4.4 the largest value of the fission matrix is presented for each standard case.

The coarsest mesh that still consistently gives a better estimate than the ideal cases is the 100 zone mesh. The largest fission matrix element of this case is 0.192. Until more systems have been simulated the significance of the largest element of the fission matrix is unknown.
Chapter 5

Summary and Conclusions

The fission matrix method shows great potential as an alternative to conventional Monte Carlo. Several verifications have been presented, in this thesis and in previous works, that the source error indeed is reduced by the fission matrix. Further conclusions of this however have to be drawn cautiously.

The ambition to use any source is shown in [9] to be too optimistic because the fission matrix needs to be developed throughout the whole system for it to be useful. This does however not limit the source independence potential as long as a flat source is used as the initial guess. Possibly the same default initial guess (the flat source) could be used for all problems without inducing source error, this has to be investigated further.

The simulations presented in this thesis does clearly show that source error is reduced with finer meshes. Unexpectedly though, the total error was reduced even beyond what should be possible (the ideal cases) which is an indication that the finer meshes overestimate $\lambda$. This overestimation is believed to be a consequence of the reduced particle density (number of neutrons per zone), which will decrease when the number of zones is increased for a fixed batch size. For future investigations it is therefore recommended to keep the particle density constant for all meshes. Because this means that the batch size will increase for finer meshes, fewer cycles should be simulated to compensate and keep the total number of neutron histories constant. The reason that several cycles are simulated in the first place is to remove source error, which is already accounted for.

Furthermore more systems need to simulated to investigate the validity of the results produced in this thesis. Especially the largest element of the fission matrix should be studied. It is an interesting measure because it has the potential to give general recommendations on mesh properties, if it can be proven to give predictable values for different systems.
Chapter 6

Bibliography


Appendix A

Fission Matrix implementation in Serpent

A.1 Introduction

Initially the code Serpent was intended to be used to perform the simulations because it is an established, and open source, code. It was discovered however that the fission matrix implementation was incorrect. In this appendix the problems that were encountered in the work with Serpent are presented. This includes a memory conservation procedure that is used in Serpent to reduce fission matrix storage needs and a normalisation problem that gives an error in the neutron source.

A.2 Memory Conservation Procedure

It was discovered that the number of non-zero elements in the fission matrix did not strictly increase throughout the simulation, see Figure A.1. This should not be possible because fission matrix elements can only accumulate data. Initially it was believed that there was a bug in the code. It was discovered however that the behaviour originated from a memory conservation procedure that only saved elements with less than 10 % relative error.

Because storage optimisation is a concern in FMBMC the procedure was investigated. If storage can be saved, at no or little cost to the accuracy of the results, it should be considered.
Figure A.1. As the simulation progresses the number of non-zero elements in the fission matrix should strictly increase. As shown in the figure however there are places where the number decreases. This is due to a memory conservation procedure.

Figure A.2. Comparison of eigenvalues of fission matrices from simulations in Serpent. The black curves are simulations with the memory conservation procedure and the red one is without. Larger batch sizes gave a faster convergence per cycle but consequently each cycle also took longer to simulate.
After several runs both with and without this procedure it was found that the eigenvalue was always underestimated with the procedure and slowly converged upwards from there, see Figure A.2. Albeit larger batch sizes gave faster convergence per cycle, there was no overall performance gain per neutron history because each cycle took longer to simulate. Without the procedure however the eigenvalues did not have any bias in this fashion. The non-reduced matrices was found to be 150-225 % the size of the reduced matrices.

In the end the procedure was skipped but nevertheless it illustrated the philosophy of averaging all results. Even neutron histories with individually high relative errors are worth counting because evenly distributed statistical variations cancel out when big averages are made.

A.3 Normalisation Problem

After further investigation of the fission matrix implementation in Serpent it was discovered that a normalisation problem must be present somewhere in the code. Smaller meshes, see Figure A.3 and A.4 did not accurately score the fission matrix.

![Figure A.3](image)

**Figure A.3.** The test case with 4 zones had 40k neutrons per cycle and was simulated for 1000 cycles. The reference had 2M neutrons per cycle, was simulated for 2000 cycles and had a mesh of 1024 zones. The reference eigenvector was averaged over 4 zones (discretised reference) so comparison could be made to the test case.
Appendix A. Fission Matrix implementation in Serpent

The fundamental mode eigenvector was too flat, even after 1000 cycles. The shape should be the same as the discretised reference, which is collapsed by Eq. B.1 from the reference. It was observed however that an underestimation is made in the middle of the eigenvector and an overestimation is made at the periphery. After this discovery the Serpent code was abandoned for the in-house code, see Section 4.3.1.

![Graph showing relative neutron density vs zone number](image)

**Figure A.4.** The test case with 16 zones had 16k neutrons per cycle and was simulated for 1000 cycles. The reference had 2M neutrons per cycle, was simulated for 2000 cycles and had a mesh of 1024 zones. The reference eigenvector was averaged over 16 zones (discretised reference) so comparison could be made to the test case.
Appendix B

Measurement of eigenvector error

B.1 Equations for Eigenvector Evaluation

To compare vectors of different dimensions some kind of normalisation has to be performed to collapse them to the same dimension. This normalisation does however introduce a bias.

In the following context, let the vector operator \( \sim \cdot \) be defined as

\[
\sim x = \frac{x}{\|x\|_1},
\]

which normalises an arbitrary vector, \( x \). Furthermore let the vector operator \( \rangle \langle d \)

which collapses an arbitrary vector, \( x \), of dimension, \( D \), into a smaller dimension, \( d \), be defined for element \( i \), as

\[
\langle x \rangle_{d}[i] = \sum_{j=i \cdot w}^{(i+1) \cdot w-1} x_j
\]

for \( i = 0, ..., d - 1 \) where \( w = D/d \).

The error, \( \delta_x \), in vector, \( x \), is calculated by comparing it to the reference vector, \( R \), accordingly

\[
\delta_x = \| \sim x - \langle x \rangle_{d_x} \|_1
\]

where \( d_x \) is the dimension of \( x \).

From Eq. B.2 it is now evident that \( \delta_x \to 0 \) as \( d_x \to 1 \). In other words vectors of smaller dimensions will inherently have a smaller error, down to vectors of dimension 1 that will have an error of 0, independently of the actual discrepancy between the two vectors.
Appendix B. Measurement of eigenvector error

B.2 Evaluated Simulations

This bias became a big problem when the eigenvectors from fission matrices of different dimensions were compared. In Figure B.1 the progression of the error in eigenvectors, evaluated with Eq. B.2, is presented. In this simulation four different fission matrix meshes (5, 25, 100 and 500 zones) were used. The materials and geometry used is presented in Table 4.1. A varying number of inactive cycles were simulated followed by 1000 active cycles. In each cycle 50 000 neutrons were simulated.

![Graph showing the progression of error in eigenvectors.](image)

**Figure B.1.** Comparison of eigenvector error convergence. The error in the eigenvector is compared to the number of inactive cycles preceding the 1000 active cycles.

The 5 zone case shows the expected behaviour. Initially the error in the eigenvector is big and then steadily decreases to a steady value i.e. the source error diminishes. With the 25 zone case however the bias is starting to emerge. As expected the initial error is smaller than that of the 5 zone case but it converges to a larger value. Because of the bias in Eq. B.2 the final error will inherently be bigger for larger matrices. This is very evident when comparing the 100 and 500 zone cases. There is no progression of error in any of these cases i.e. the source error is negligible. However the error is larger in the 500 zone case due to the bias in Eq. B.2.

Because two separate phenomena are associated to measuring the eigenvector error, it was decided to compare eigenvalues instead. The eigenvalue is one dimensional and will therefore be more easily comparable between different meshes.
Appendix C

Running simulations on the supercomputer

C.1 Introduction

The code, written in C++, is a proprietary non-analogue continuous energy Monte Carlo criticality code based on algorithms from the COHORT-II code [25]. To take full advantage of the Tegnér supercomputer processing power the code needs to be parallelised. This means that:

- The main.cpp file needs to be modified.
- The seed algorithm needs to be modified.
- A bash script needs to be created that can initiate the simulations.

The required modules are:

- **gcc** Standard C++ compiler.
- **gsl** Scientific libraries for the random number generator.
- **i-compilers** Intel compilers with libraries required by **intelmpi**.
- **intelmpi** Multithreading compiler and executable used to run the code.

C.2 Modifying main.cpp

main.cpp is the only file in the code that needs modification. In the main file a few lines are added to denote which part of the code is supposed to be split into multiple instances. Three lines, see Figure C.1, are added right at the beginning of the file to initialise of the parallelisation.
int main( int argc, char *argv[] )
{
    MPI_Init (&argc, &argv);
    MPI_Comm_rank (MPI_COMM_WORLD, &rank);
    MPI_Comm_size (MPI_COMM_WORLD, &size);
    :

    MPI_Finalize();
    return 0;
}

Figure C.1. Initialisation of the parallelisation highlighted in violet.

The variable rank holds the current instance number (0 indexed) and size holds the total number of instances. One line, see Figure C.2, is added at the end of the file to finalise the parallelisation.

:
    MPI_Finalize();
    return 0;
}

Figure C.2. Finalisation of the parallelisation highlighted in violet.

C.3 Modifying the seed algorithm

The default procedure to choose seed in the code is to use the number of seconds since 1970, readily available in C++ from the function call time(NULL);. Unpredictable behaviour may however arise when multiple parallel simulations are started simultaneously, sometimes even at the same second. The seed is therefore instead based on the rank variable, which will be different for all instances.

C.4 Bash script initialiser

Tegnér is made up of 65 nodes where every node has at least 24 cores\(^1\). A separate sbatch call, which adds a job for one node into the queue, is made 42 times by a for-loop, see Figure C.3. It is decided to use 42 nodes in order to get over 1000 runs in total (24 cores \(\times\) 42 nodes = 1008 simulations).

```
#!/bin/bash
for i in {1..42}
do
    sbatch [#options] ./job.sh ${i}
done
```

Figure C.3. The bash script file execute_script.sh that initialises the 1008 runs

\(^1\)https://www.pdc.kth.se/resources/computers/tegner/hardware
Options can either be submitted directly into the `sbatch` call at `[#options]` or added into the `job.sh` file. For options added directly into the `sbatch` call, see Figure C.4. These options use the node number `$i` as a part of the argument in order to make the argument unique for that call.

```
-o slurm/output_file$i.o
-e slurm/error_file$i.e
-J m25$_i
```

**Figure C.4.** Dynamic options for `execute_script.sh`.

The options in order are `-o` that sets the name of the output file, `-e` that sets name of the error file and `-J` that sets the name for the call in the queueing system.

When a call reaches the top of the queue the `job.sh` file, see Figure C.5, is run. In it is the actual call to `mpirun`, which executes the code. The number of cores to use is set by the `-n` option for `mpirun`. In the `job.sh` file the remaining options `--mail`, `-t`, and `-A` are also set. These options are static and can therefore be called from within the file. They set mail notification options, time limit for the simulation and account (project) number. All modules are also loaded with `module load`.

```
#/bin/tcsh

#SBATCH --mail-type=ALL
#SBATCH -t 01:00:00
#SBATCH -A 2015-51

module load gcc/5.1
module load gsl/1.16
module load i-compilers/15.0.2
module load intelmpi/5.0.3

mpirun -n 24 ./a.out -sn $i > out/log$_i.dat
```

**Figure C.5.** Queue system submission file `job.sh`.

Console output from the C++ code from each call is written to `log$_i.dat`, in contrast to the output from the queueing system which is written to `slurm/output_file$i.o`. To which catalogue results from the simulations are written is specified inside the code. Together with the node number `$i`, loaded into the code as an argument, the `rank` variable in the code can be used to make unique file names for result files.