Efficiency improvement of local power estimation in the general purpose Monte Carlo code MCNP5

Derk van Veen

Delft University of Technology
Faculty of Applied Sciences
Department of Radiation, Radionuclides
and Reactors

Physics of Nuclear Reactors
Mekelweg 15
2629 JB Delft

Delft, 2009

Professor: T.H.J.J. van der Hagen
Supervisor: J.E. Hoogenboom

Abstract

The general purpose Monte Carlo code MCNP can be used for calculating the effective multiplication factor in a nuclear reactor with all geometrical details. However, when more detailed information is required, such as the local power density for a large number of small volumes, it takes a long time to calculate the scores for all these volumes. The topic of this research project is to optimize the local power calculations in the general purpose Monte Carlo code MCNP5.
The total power production depends on the neutron flux and the properties of all nuclides in the fissile material. The relative contribution to the local power production of each nuclide in the fissile material is independent of the neutron flux.
The first efficiency improvement is to sum and store these relative contributions as function of energy at the start of the calculation. When the local power production is now requested during the simulation of neutron histories the combined contribution of all the nuclides is immediately known as function of the energy of the incoming neutron.
Saving these combined contribution as function of energy takes about 8Mb memory, but the energy grid contains much more points than necessary for an accurate interpolation. Removing all points which are unnecessary for an accurate interpolation is called grid thinning. This process does not significantly reduce the overall calculation time, but the required memory reduces with a factor four.
To search as efficient as possible in the energy grid a second, smaller grid is used. This second grid does not influence the original grid, but contains references to this grid. Since this second grid is much smaller it takes less time to find a certain value. But because of the complexity this second grid no time reduction could be obtained.
Using these combined contributions to calculate the local power production decreases the overall calculation time with 1%, although the calculation time for a single event is four times as low. This is because some information that would have been obtained during the calculation per nuclide is requested during the simulation and these calculations have to be done anyway.
The second efficiency improvement is to create a direct relation between the volume in the reactor and the position in memory where scores for this volume have to be stored. In the current version of MCNP moment it takes a relatively long time to find this location. Applying a logical numbering to all volumes where the local power production has to be calculated and creating a direct relation of this number where the score has to be stored results in a overall time reduction from 971 minutes per calculation to 622 minutes, which is an improvement of 36%. Within this time the statistics are not only calculated for the smallest volumes requested, but also for the fuel pins, the fuel assemblies and the entire reactor core.

Contents

CHAPTER 1

Introduction

The general purpose Monte Carlo code MCNP [4] can be used for calculating the effective multiplication factor in a nuclear reactor with all geometrical details. However, when more detailed information is required, such as the local power density for a large number of small volumes, it takes a long calculation time to calculate the scores for all these volumes.
Although computers become faster over time and the number of processors per computer will probably increase, there are other possibilities to decrease the calculation time for calculating the detailed power distribution in a nuclear reactor.
With the latest version of MCNP, MCNP5, is has become possible to calculate the detailed power distribution for a full core nuclear reactor within a reasonable time and with an acceptable statistical error (<1%), but two optimizations can be included the improve the efficiency of the calculation. First the way the local power production is calculated can be done in a more efficient way and secondly a direct relation between the position of a volume in the reactor and the location in the memory where the score has to be saved is made.
The efficiency improvement of the power density calculation is presented in Ch. 3.1 and the relation between the position of a volume and the assignment of storage location in memory where the score is saved is presented in Ch. 3.2. The results of both efficiency improvements are presented in Ch. 4.

CHAPTER 2

Theory

The power distribution in a nuclear reactor depends especially on the design of the reactor and the composition of the fuel. As a model for the reactor the model proposed for a benchmark test for Monte Carlo codes [2] is used in this research. In general a nuclear reactor consists of a number of fuel assemblies, which consists of a number of fuel pins. In this report the reactor is constructed out of 241 fuel assemblies, each containing 17x17 fuel pins with a length of 366 cm. The core is surrounded by a water filled vessel. A schematic view of the reactor and a fuel assembly is shown in Fig. 2.1. In a nuclear reactor the neutrons cause fission in the fuel pins and as a result of this

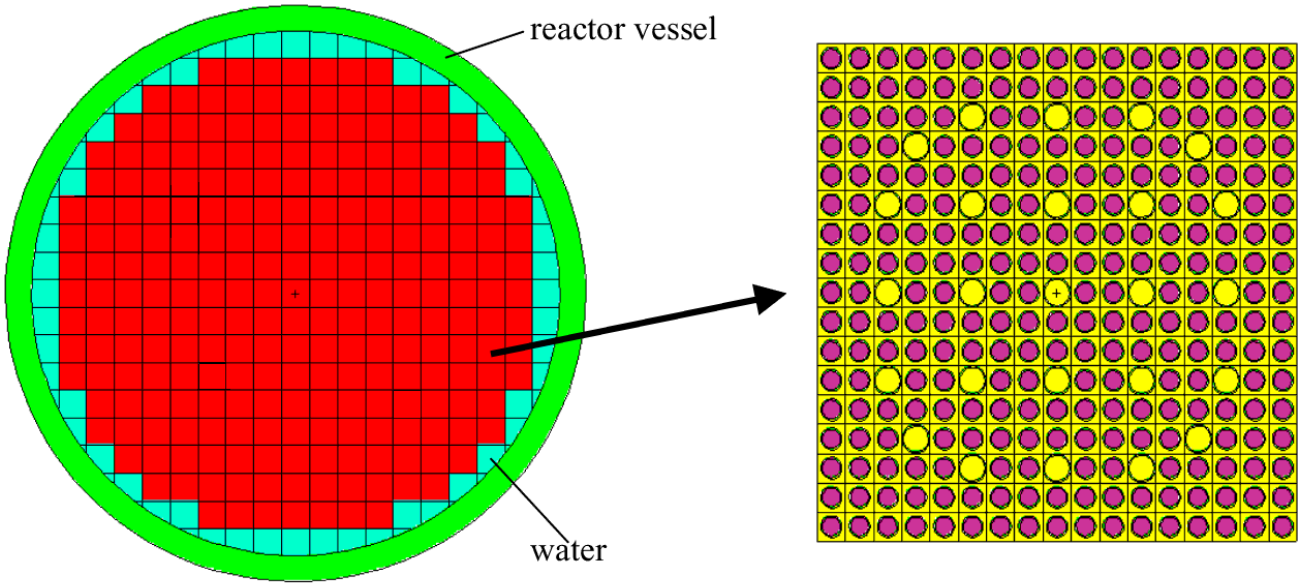

Figure 2.1: Reactor and fuel assembly

fission reaction energy is released, together with some new neutrons. These fission neutrons can cause fission at other locations in the reactor. The process where the fission neutrons cause new fission is called a chain reaction, and the fission energy released at every fission reaction is the energy production in a nuclear reactor.
The neutron transport and how this results in energy production is presented in the following section, as well as the simulation of these processes in MCNP.

2.1 The local power production

The power production in a nuclear reactor is the result of the fission energy released when a large nucleus is split into multiple smaller nuclei. The more often fission takes place, the more energy will be released. The number of fission reactions taking place depends on the amount of available neutrons and fissile material and the probability for a neutrons to cause fission. A mathematical expression for the local power production is given in Eq. (2.1).

$$P = \int_V \int_0^\infty \sum_i Q_i N_i \sigma_{f,i}(E) \phi(\mathbf{r}, E) dE dV \tag{2.1}$$

In which Q_i is the fission energy released by nuclide i in a fission, N_i the neutron density of nuclide i, σ_f is the microscopic cross section and ϕ is the neutron flux, the amount of neutrons moving through the reactor per cm^2 per second. The total power is a summation of this product over all nuclides in the material integrated over all neutron energies and the volume.

Microscopic cross section

The microscopic cross section can be seen as the probability of an interaction between a neutron and a nucleus. When a beam of neutrons with intensity I cm^{-2} s^{-1} strikes a one atom thick layer of atoms with N_a atoms cm^{-2}, the number of interactions per cm^2 will be proportional to the intensity and the atom density. The proportionality factor is the microscopic cross section σ. This cross section is usually expressed in barns (1 b = 10^{-24} cm^2).

$$\sigma = \frac{\text{Number of reactions /nucleus/sec}}{\text{Number of incident neutrons /cm}^2\text{/sec}} \tag{2.2}$$

Sometimes this microscopic cross section is called the total microscopic cross section σ_t, where t refers to the total amount of interactions, whatever this reaction might be. The subscript is used to indicate the different kind of cross sections. The probability of fission is expressed by σ_f, a c is used for capture and the s for scattering.

The cross sections are not constant but depend on the energy of the incoming neutron. For most nuclides the cross section is larger for thermal neutrons and very small for very fast neutrons. The fission cross sections for ^{235}U as a function of the energy is shown in Fig. 2.2.

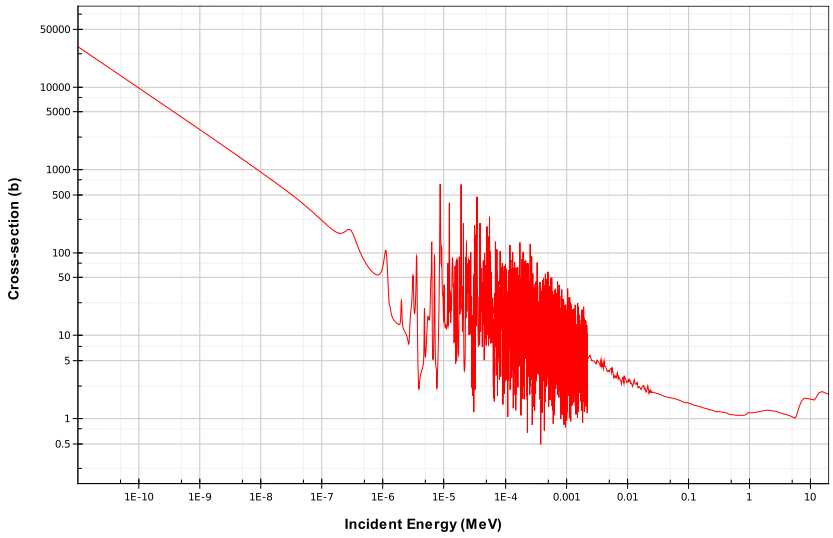

Figure 2.2: The fission cross section of ^{235}U

Macroscopic cross section

But the microscopic cross section only describes the probability of interaction between a neutron and a thin layer of nuclides, actually it describes only the interaction between a neutron and a single nucleus, while the neutrons moves through a three dimensional material. To describe the interactions in a three dimensional material the macroscopic cross section is used. This macroscopic cross section is the product of the nuclide density of a material and the microscopic cross section, and represented by the capital sigma.

$$\Sigma_t = N\sigma_t \tag{2.3}$$

The probability a neutrons travels a distance x within a material without any kind of interaction is equal to $e^{-\Sigma_t x}$, the probability a neutrons has some interaction between a point x and $x + dx$ is $\Sigma_t dx$. From this it follows that the mean free path (λ), or the expectation value for the path traveled without interaction of a neutron is given by

$$\lambda = \int_0^\infty x e^{-\Sigma_t x} \Sigma_t dx = \frac{1}{\Sigma_t} \tag{2.4}$$

In a material which consists of multiple nuclides, the cross section is the sum over all cross sections in the material:

$$\Sigma_t = \sum_i N_i \sigma_{t,i} \tag{2.5}$$

Neutronics

The total number of interactions between neutrons and nuclei is described by the product of the macroscopic cross section and the flux, which can be read as the probability of interaction for a single neutron times the number of neutrons passing by.

$$F(\mathbf{r}, E) = \Sigma_t(E)\phi(\mathbf{r}, E) \tag{2.6}$$

When a neutron interacts with a nucleus two kind of reactions can occur. The first reaction is a scattering reaction, where the neutron changes its direction and energy, but will continue its flight after the collision and it can still cause a fission reaction at some other position. In the second case the neutron will be absorbed in the nucleus. After the absorption two things can happen. The nucleus can decay to a stable energy level by emitting a γ photon, or the nucleus can split. In the case of fission energy will be released together with new neutrons.
These new neutrons are vital to produce energy in a nuclear reactor, because these new neutrons can cause fission in another nucleus. To operate a nuclear reactor at a stable state the amount of neutrons has to be stable over the time. This means that for every neutron absorbed one new neutron has to be produced in a fission reaction.
The amount of neutrons produced with respect to the previous generation is the multiplication factor or the k_{eff} value.

$$k_{eff} = \frac{\#\text{ neutrons in current cycle}}{\#\text{ neutrons in the preceding cycle}} \tag{2.7}$$

When $k_{eff} > 1$ the amount of neutrons in the next cycle is larger than in the current cycle. This means the neutron population grows in time. In the case where $k_{eff} < 1$ the amount of neutrons decreases in time, and finally the chain reaction will stop. When k_{eff} is equal to one the neutron population does not change in time.

2.2 Monte Carlo

The most common method to simulate deterministic problems is a by using a deterministic method, which solves the (average) behavior of particles described by a transport equation. By contrast, Monte Carlo methods simulate individual particles and records their average behavior. From this average behavior an estimation is made, using the central limit theorem, about behavior of particles in the real physical world. Because an estimation is made the answer has some uncertainty.

The Monte Carlo method uses statistics to describe the deterministic behavior of particles. The advantage is no transport equation is necessary or has to be solved, but the price to pay is an answer with some uncertainty.
The statistics are used to describe stochastic processes in neutron transport: the neutron interactions and the numbers of fission neutrons produced in a fission process. In order to do this random numbers are used.

2.2.1 Random numbers

In Monte Carlo an enormous amount of random numbers is required to simulate the particle behavior. A random number is used to decide a collision or a fission event will take place, how far a neutrons travels before its next interaction, the direction after a scattering event, etc. These random numbers have to be simulated by a computer, where randomness does not really exists. Random means there is no correlation between 2 values and a set of random numbers does not repeat itself. In a Monte Carlo simulation a billion requested random numbers is not uncommon. This mean a random number generator has to be able to produce this amount of random numbers without repeating itself. In MCNP, one of the common Monte Carlo programs in reactor physics, a random number generator is used which produces 2^{46} random numbers before it starts repeating itself. All random numbers, ρ, produced by this simulator are between zero and one.

2.2.2 Simulation of the neutron path length

The first process to be simulated is the path length a neutron travels before it interacts with a nucleus. This path length is given by Eq. (2.8):

$$f(s) = \Sigma_t e^{-\Sigma_t s} \tag{2.8}$$

From this equation it can be seen that the distance traveled by a neutron varies between zero and infinity, but has to be described by a random number, ρ, between zero and one. How this is done in shown in the following equations

$$\begin{aligned} \rho &= \Sigma_t \int_0^{s_1} e^{-\Sigma_t s} ds \\ &= 1 - e^{-\Sigma_t s_1} \end{aligned} \tag{2.9}$$

Which results in a path length s_1 of

$$s_1 = -\frac{\ln(1-\rho)}{\Sigma_t} \tag{2.10}$$

From Eq. (2.10) can be seen that the path traveled is indeed between zero and infinity.

2.2.3 Sampling of an event

Since cross section are related to the probability an event will happen it makes sense to use these cross section to sample events. The total cross section for interaction must be the sum of the cross sections for scattering and absorption, each interaction is or a scatter- or an absorption reaction. The absorption cross section itself is the sum of the cross sections for capture and fission.

$$\sigma_t = \sigma_s + \sigma_a = \sigma_s + \underbrace{\sigma_f + \sigma_c}_{\sigma_a} \tag{2.11}$$

The probability one of the events at a collision will happen is one, and the cross sections are a measure for how large the probability of one event is with respect to an other event. For this reason it is chosen to normalize Eq. (2.11) through dividing by σ_t.

$$1 = \frac{\sigma_s}{\sigma_t} + \frac{\sigma_f}{\sigma_t} + \frac{\sigma_c}{\sigma_t} \tag{2.12}$$

The random numbers generated by the random number generator are now used to to make decisions about neutron behavior. In the case of interaction between a neutron and a nucleus the probability of a scattering reaction is P($s_{cattering}$) and the probability of an absorption reaction is P($a_{bsorption}$). To decide which of the two events will happen a random number, ρ, is taken. When $\rho \leq$P(s) a scattering reaction will take place and an absorption reaction when $\rho >$ P(s). Whether fission or capture will take place in an absorption reaction depends on the relative difference of the cross sections σ_f and σ_c and a new random number.

2.2.4 Scoring of a neutron

A Monte Carlo code only supplies information which is specifically requested by the user. Information could be the flux at a position, the power produced in a certain volume, or the number of neutrons within a detector.
Let's assume the amount of neutrons having a scattering reaction in a detector is requested. To calculate this N neutrons are released and every time a neutron scatters within the detector a value of one is scored. But a single neutron can have multiple scores within the detector. In the case a neutron scatters three times within the detector the score for this neutron is three, and the square of the score for this neutron is nine. To calculate the average score for N neutrons the following formula is used:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{2.13}$$

in which x_i is the number of scattering reactions for neutron i. In general x_i can be any score requested by the user. In this report the overbar is used to indicate the average, or the expectation value of a score. Depending on the number of scattering reactions in the detector the average will vary over a large or smaller interval. The size off this variation is expressed by the variance s^2.

$$s^2 = \overline{(x_i - \bar{x})^2} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 \tag{2.14}$$

When the variance is zero, it means that the estimation will be exactly the same every time the experiment is done. When the variance is very large the different outcomes can differ more than 100%.

The average value of an estimator should not be confused with the true mean. The true mean would be the result of a scoring process with an infinite number of scores. Sometimes the true mean is described as the mean of the population. The population is an assemble containing all possible scores. The average calculated with a limited number of samples is an estimation of this true mean.

$$\mu = E[x_i] \tag{2.15}$$

The variance of the population is desribed by

$$\sigma^2 = E\left[(x_i - \mu)^2\right] = E[x_i^2] - \mu^2 \tag{2.16}$$

which is the variance of an infinite number of samples. When the expectation value of the standard deviation with a limited number of samples is calculated, it can be seen that this result differs from the variance of an infinite number of samples.

$$E\left[\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2\right] = \frac{N-1}{N}\sigma^2 \tag{2.17}$$

To make an unbiased estimation for N approaching infinity the variance for a limited number of samples is calculated in the following way:

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{1}{N-1}\left[\sum_{i=1}^{N}x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N}x_i\right)^2\right] \tag{2.18}$$

The expectation value of this estimator is equal to the one of an infinite number of samples:

$$E\left[\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2\right] = \sigma^2 \tag{2.19}$$

For a very large N this is a very small detail, because the difference between $\frac{1}{N}$ and $\frac{1}{N-1}$ becomes smaller with increasing N.
Finally the variance of the mean can be calculated. This is the statistical spread in the mean value.

$$s_{\bar{x}}^2 = \frac{1}{N}s^2 = \frac{1}{N-1}\left[\frac{1}{N}\sum_{i=1}^{N}x_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N}x_i\right)^2\right] \tag{2.20}$$

In most practical situations the variance is expressed as percentage of the average value:

$$\text{Relative error} = \frac{s_{\bar{x}}}{\bar{x}} \times 100\% \tag{2.21}$$

2.3 Variance reduction

One of the most important issues in Monte Carlo calculations is the reduction of the variance in the estimation. There are multiple methods to achieve this, but the basis of most methods is to assign a weight to each neutron. In general the weight of a starting neutron is one. In analog simulations this weight will always remain equal to one. In the case of implicit capture the weight of a neutron is changed after each event.
Assume the probability of absorption is δ, then the weight of the neutron is diminished with the relative size of this δ with respect to the current weight at every collision. The weight corrects for the probability that a neutron could have been absorbed in an analog case. When scoring finally takes place, the neutron does no longer count for one neutron, but only for the weight the neutron still has.
In the case where the number of scattering reactions within the detector was requested each scattering reaction in the detector contributed one to the total score. In the case of implicit capture only the weight w would be added to the score for that neutron. For a second scattering reaction within the detector directly after the previous scattering would give an extra contribution of w(1 - δ).
The advantage of this method is that a neutron 'lives' much longer and travels much further through the geometry. The disadvantage is that a lot more simulation has to be done, which takes more time. When the weight of a neutron has become very small one can image the time it takes to simulate the behavior of this neutron does no longer outweigh the contribution the neutron gives to a score. The following methods are generally used to make optimal use of the weight of the particle.

Implicit capture Implicit capture is already described in the previous paragraph, but a little more formal definition will be given over here. Implicit capture means the neutron survives a collision in all cases, but the weight of the neutron is reduced to take the probability into account the neutron could have been absorbed in one of the previous collisions. The weight of the neutron after a collision becomes the probability of survival in a collision times the previous weight: $w = w'(1-\frac{\Sigma_a}{\Sigma_t})$. Implicit capture decreases the variance, but increases the calculation time.

Russian roulette Russian roulette is played when a neutron has a very small weight, so small that the simulation time has become to large in comparison to the contribution the neutrons gives to the final score. The limit for this weight is w_{RR}. Below this weight the russian roulette is played. It is possible to terminate a neutron with probability $\frac{1}{2}$ and double its weight when the neutron survives. But this doubled weight can still be very small. For this reason it is most common to set a survival weight w_{sur}. Because the total weight of the neutrons should not change, the probability of survival has to be changed as well. The probability a neutron survives is now equal to $\frac{w}{w_{sur}}$ and after survival the neutron a weight w_{sur} is assigned to the neutron. In most calculations the survival weight is twice the russian roulette threshold. This means a neutron with a weight exactly equal to the threshold, $w = w_{RR}$, has $\frac{1}{2}$ chance of survival. Russian roulette decreases calculation time, but increases the variance.

Particle splitting When the weight of a neutron exceeds a certain limit, w_{sp}, the neutron is splitted into N neutrons with weight $\frac{w}{N}$. The scores of these N neutrons all give contribution to the original neutron score. Particle splitting decreases variance and increases the calculation time.

The final two variance reduction techniques discussed above can be summarized in a weight windows figure. An example of such figure is shown in Fig. 2.3.

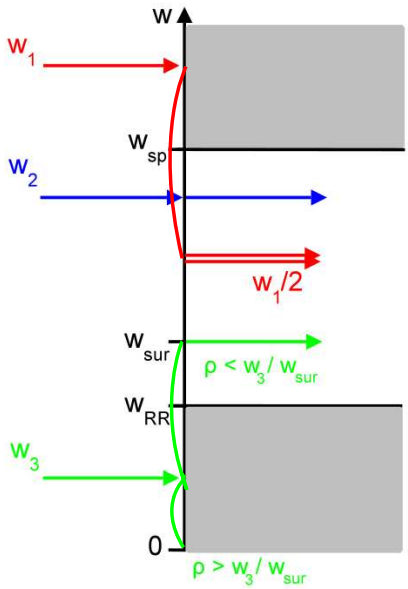

Figure 2.3: Visualisation of weight windows

In the vertical direction the weight of the incoming particle is visualized by a red, a blue and a green arrow. In the case the weight of a neutron is very high (the red arrow), the particle is split into two particles with half the weight of the incoming neutron. When the weight of the particle is somewhere in the middle nothing happens, and when the weight of the neutron is small, pointed out by green arrow, the neutron can survive or be terminated. When the neutron survives a new weight, w_{sur}, is assigned to it.

Figure of Merit

The balance between calculation time and the variance is expressed in the Figure of Merit. The Figure of Merit is a measurement of the variance with respect to the calculation time.

$$FOM = \frac{1}{\sigma^2 T} \tag{2.22}$$

In the analog situation the FOM is constant, besides some statistical fluctuations, but with the variance reduction techniques one tries to maximize the FOM. The FOM applies only to the same calculation on the same type of computer. On a different computer the processor speed can be different which could results in a longer calculation time, and so a lower Figure of Merit.

2.4 MCNP5

MCNP5 is a general purpose Monte Carlo transport code and can be used for the transport simulation of neutrons, photons and electrons. The basis for the transport calculations is the k_{eff} calculation.
The original source of MCNP5 is much older than most of the current users. The very first k_{eff} estimators for nuclear reactors with Monte Carlo techniques are written in the 1960s. In the 1970s the first k_{eff} calculations where done for single fuel assemblies within reactors.
In 1977 three different codes are combined into one more general code for Monte Carlo calculations for particle transport in nuclear physics. The three original codes where designed for photon transport, neutron transport and k_{eff} calculations. This combined code is the first code with the name MCNP and it can be used for all three kind of calculations.
From this point in time MCNP and its libraries have been developed by the Monte Carlo team in X-Division of Los Alamos National Laboratory for over 25 years which resulted in 2003 in the latest version of MCNP: MCNP5. This version consist of 340 separate subroutines which contain in total over 88,000 lines of fortran code.
In this version much more calculations can be done in more complex geometries and a new tally is introduced to handle large geometries.

In this report the main focus is on the neutron transport within a nuclear reactor and everything related to this, like the geometry of a reactor, the calculation and processing of tallies and neutron transport itself.

The basis of all calculations is the movement of neutrons through the reactor core. Basically the neutron transport can be described in the following way: The neutron starts at a certain position and moves some distance before it has some interaction with a nucleus. This interaction can be a scattering reaction, where the energy and direction changes, or an absorption reaction. In an absorption reaction the neutron history ends, but in the case of fission new neutrons are produced which have to be saved for the next cycle. This process is visualized in the flowchart of Fig. 2.4.

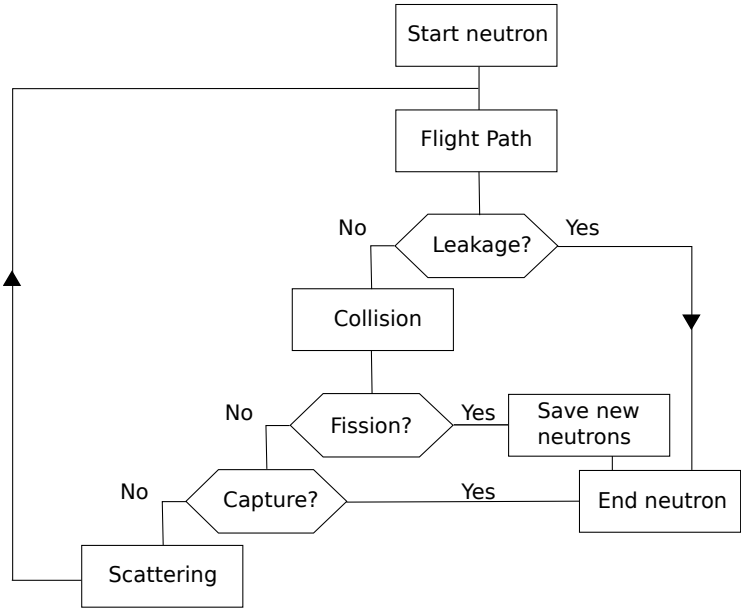

Figure 2.4: Flow chart of neutron transport.

All neutrons within a single cycle are processed in this way. All the fission neutrons produced in this cycle are saved; they are the start neutrons for the next cycle.
Although the basics are very simple a lot of calculations have to be done. At first the starting position of the neutron has to be sampled from some neutron source. From this starting point the direction and energy have to be sampled and the flight distance has to be calculated. Along this flight path a lot information has to be collected. For example the composition of the material, the cross sections for all nuclides in this material and whether the neutron crosses a boundary between two different kind of materials during its flight.
When the neutron is in some region where a tally has to be scored this score must be stored at some position in memory where it can be retrieved. At the end of the program the scores of a tally together with the square of scores are used to calculate the (relative) error.

2.4.1 Cycles

MCNP works with cycles of neutrons. In a single cycle all neutrons are processed along the lines in the flowchart of Fig 2.4 till they are terminated. When a neutron causes fission along its path, these new fission neutrons and their positions are saved for the next cycle. In the following cycle all fission neutrons from the previous cycle are processed.

At the start of the first cycle, the distribution of neutrons is not known, at best a good estimate can be made. Before any serious calculation can be started, the neutron population must be constant in time. The best way to check this is by monitoring the k_{eff} values over the successive cycles. After a number of cycles k_{eff} will stabilize around a value little larger then one. This process of convergence is shown in Fig. 2.5.

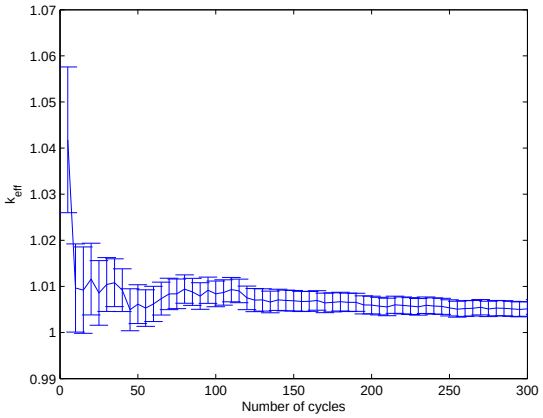

Figure 2.5: Source convergence after number of cycles

The cycles before convergence are the inactive cycles, the cycles after convergence are the active cycles. In the active cycles the neutron population is a good estimator of the real physical behavior of neutrons and can therefore be used for calculations.

With a small number of neutrons per cycle the source will converge fast, but some parts in the geometry might be undersampled. With a large number of neutrons this problem does not arise, but it can take a long time before the neutron source converges. A balance has to be found between the number of neutrons per cycle and the number of cycles before convergence is reached. The number of required neutrons per cycle depends on the size and composition of the geometry. In Fig. 2.6 three very simple, successive cycles are plotted.

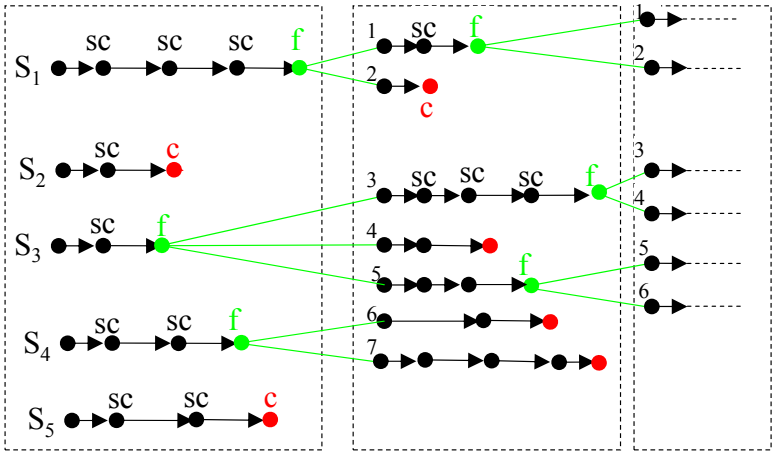

Figure 2.6: Illustration of successive cycles

After the fist cycle the $k_{eff} = \frac{7}{5} = 1.40$ and after the second cycle $k_{eff} = \frac{6}{7} = 0.86$. The estimation of k_{eff} during one cycle can be considered as estimating a detector response, so normal statistics apply to it. With Eq. (2.13) $\bar{k}$ can be calculated for this simple case: $\bar{k} = \frac{1}{2}(1.40 + 0.86) = 1.13$ with a variance $s^2_{\bar{k}} = \frac{1}{2-1}(\bar{k^2} - \bar{k}^2) = 1.35 - 1.28 = 0.07$ which is calculated with Eq. (2.20). The relative error of this example is 6.1%.

2.4.2 Estimators

In the active cycles the actual calculations can be done. This calculations can consist of any tally possible in the current geometry, like a current tally, a flux tally or a reaction rate tally. It is even possible to solve dosimetry problems (the dose received in tissue according to radiation and neutrons). All these tallies can be calculated with estimators. A flux tally can for example be calculated with a collision estimator, a track length estimator or a surface crossing estimator.
In the case a flux tally is requested for a volume V, the flux in this volume can be estimated by looking at the number of collisions within this volume. When the number of collisions is known, it is possible to calculate the flux in the volume, the flux tally for this volume.
A reaction rate tally is a flux tally times the cross section of a material. The fission rate for example is the flux times the fission cross section. Two of the estimators are discussed below in more detail, because they are used in the rest of the report.

Track length estimators In the case of a track length estimator the flux tally is calculated by adding all distances from neutrons traveling through volume V in a single cycle. The distances are weighted distances in the case the neutron does not have a weight of one, as in variance reduction techniques.
To calculate the flux, the total distance traveled through the volume is divide by the volume to get a flux. In general the contribution per neutron is the parameter of interest i.e. the average contribution for each neutron released in the geometry. The final expressing for the flux tally with a track-length estimator is

$$\phi = \frac{1}{NV} \sum_i d_i w_i \tag{2.23}$$

in which d_i is the distance traveled by neutron number i in the volume V and w_i is the weight of neutron i. A graphical interpretation for the track length estimator can be found in Fig. 2.7.

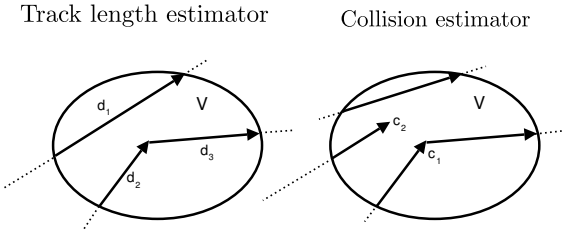

Figure 2.7: Tracklength and collision estimator

In this figure the flux is estimated by adding all distances traveled in this volume: $d_1 + d_2 + d_3$. In the case these neutrons would have had a weight the distance traveled would be $d_1w_1+d_2w_2+d_3w_3$.

Collision estimators A collision estimator uses the number of collisions within a volume to estimate the flux in that volume. The number of interactions in a volume is equal to the flux in the volume times the probability of interaction according to Eq. (2.6). When the number of interactions and the probability of interaction are known, the flux can be estimated with the following formula:

$$\phi = \frac{1}{NV} \sum_i \left(\frac{1}{\Sigma_t} \right)_i w_i \tag{2.24}$$

which gives the average contribution to the flux per neutron in this volume. Of course the contribution for a single collision is corrected for the weight of the colliding neutron. In an analog calculation all the w_i's would be equal to one.

CHAPTER 3

Efficiency improvement of the local power density calculation

As presented in Ch. 2 the power produced in a nuclear reactor and its spatial distribution are usually the parameters of interest. The power produced in a nuclear reactor can be calculated with Eq. (3.1).

$$P = \int_V \int_0^\infty \sum_i Q_i N_i \sigma_{f,i}(E) \phi(\mathbf{r}, E) dE dV \tag{3.1}$$

Calculating the power in this way is straightforward, but in the general purpose Monte Carlo code MCNP5 not done in the most efficient way. This regards the summation over nuclides in Eq. (3.1) and the storage of scores for the local power estimates in case of a large number of volumes for which the power has to be estimated. A more efficient way of this calculation is presented in the next two sections.

3.1 Calculation of the local power in MCNP5

In Eq. (3.1) the nuclide density and the energy produced per fission are constants for a nuclide in the fuel pin, but the microscopic cross section depends on energy. Besides these three variables the flux has to be known to calculate the local power production. The flux is calculated in every cycle with a collision estimator, called a f7 tally in MCNP5, or a track-length estimator, called a f4 estimator in MCNP5, as described in Sect. 2.4.2. In both cases the summation over the nuclides is calculated at every collision. The summation over the nuclides is given in Eq. (3.2)

$$R(E) = \sum_i Q_i N_i \sigma_{f,i}(E) \tag{3.2}$$

As this summation is independent of the flux estimator it is more efficient to calculate this sum in advance of the simulation and store it in memory as function of energy for each fissile medium.

Moreover, looking up the function value at a certain energy can be done more efficient than in MCNP5.

In MCNP5 the microscopic cross sections are assumed to be known and are stored in databases. In the case of cross sections for nuclides these databases are called cross section tables. These cross section tables are discussed in the next section.

3.1.1 Cross section tables

In MCNP5 the properties of nuclides are assumed to be known and stored in tables. These tables do not only contain the cross sections, but also other properties of the nuclides like the fission energy per fission and the density. Multiple cross section tables are available for different kind of calculations, for example calculations at different temperature. The tables used for the simulation are called the libraries of the simulation.
One commonly known datafile is the Evaluated Nuclear Data File, or simply the ENDF file. A small part of the data table of ^{235}U is given in Table 3.1.

Table 3.1: Selection from ENDF/B-VII.0 library [1] for ^{235}U

E(eV)	σ_t	σ_f	σ_c
2.00179896	70.3123676	21.2179377	36.85937145675421
2.0165961	88.2300478	24.600149687558776	51.3003946
2.02044576	92.0560278	25.4800742	54.221878955529235
2.02429543	95.5539545	26.031937849102636	57.1433709
2.0286263	98.3404387	26.6527832	59.281367285650944
2.0291075	98.604508	26.676229883347297	59.5189186
2.03555569	100.167288	26.9904208	60.72635111734377
2.03570006	100.188173	26.983350807750178	60.7533846

The steps in energy are so small that linear interpolation can be used to calculate the cross section for any energy of the incoming neutron.
From this table with cross sections it is clear that the cross sections are not given for each energy. The cross section for an incoming neutron with energy 2.02429543 MeV is known, but when the energy is equal to 2.02200 MeV linear interpolation has to be used to find the cross section at that specific energy. For linear interpolation Eq. (3.3) can be used.

$$\sigma(E) = \frac{E - E_{j-1}}{E_j - E_{j-1}}(\sigma_j - \sigma_{j-1}) + \sigma_{j-1} \tag{3.3}$$

In this equation E_{j-1} is the highest energy in the table lower than the requested energy, and E_j is the lowest energy higher than the requested energy, 2.02044576 and 2.02429543 MeV respectively. In this case the library is very small and it is immediately clear between which value the interpolation has to take place, but the entire table for ^{235}U consists of almost 20,000 energy entries. Because these tables are this large it is important to use an efficient search algorithm to find a requested value as fast as possible. The interpolation algorithm is presented in the next paragraph.

Interpolation algorithms When the data is ordered from the smallest to the largest value a linear search is a way to find a value in an array of length N. In a linear search each index from the array, starting at the fist index, is read until the required value or the first larger value has been found. The efficiency of this search is $\frac{N}{2}$ on the average and is proportional with N; when N

becomes twice as large, it takes twice at much time to find the correct value.

A **binary search** is comparable to how most people find a person in the telephone book. The book is just opened at a certain page. When the name is before this page they glance a few pages back. This is only possible when the names are alphabetically ordered.
The same kind of search can be performed on data in an array, but in a more systematical way. In this algorithm the value in the middle of an array is examined. When the value aimed at is larger than that value the search continues in the upper half of the table and otherwise in the lower half of the table. After one step the problem is halved. These steps are continued until the aimed value has been found.
Assume the value of 1 is the value aimed at in the next table.

Table 3.2: Interpolation example.

Index	0	1	2	3	4	5	6	7	8
Value	0	1	4	9	16	25	36	49	64

First the value at index 4 is examined. The value at this index, 16, is larger than 1, so the search continues in the lower half of the array: the elements at index 0, 1, 2 and 3. From this 4 elements the value in the middle is examined. In this case there is no middle index, but by convention the lower one of the two middle values is examined. This value is exactly the value aimed at, so the search ends here.
At every step the region to be searched is halved, which means that it takes one extra step to find the same value in an array with double size. The efficiency of this binary search is $^{2}\log N$. In comparison with the linear search this is much more efficient to find a certain value in an ordered array. When the array to be searched is of length 1024 it takes at most 1024 steps to find a value with a linear search and 10 steps with a binary search.

3.1.2 Cross sections in the local power calculation

In the previous section cross section tables are discussed. These tables contain the microscopic cross sections for all used nuclides as function of the energy of the incoming neutron. In Fig. 3.1 the cross sections for three nuclides are plotted as function of the energy between 2 and 3.1 eV.

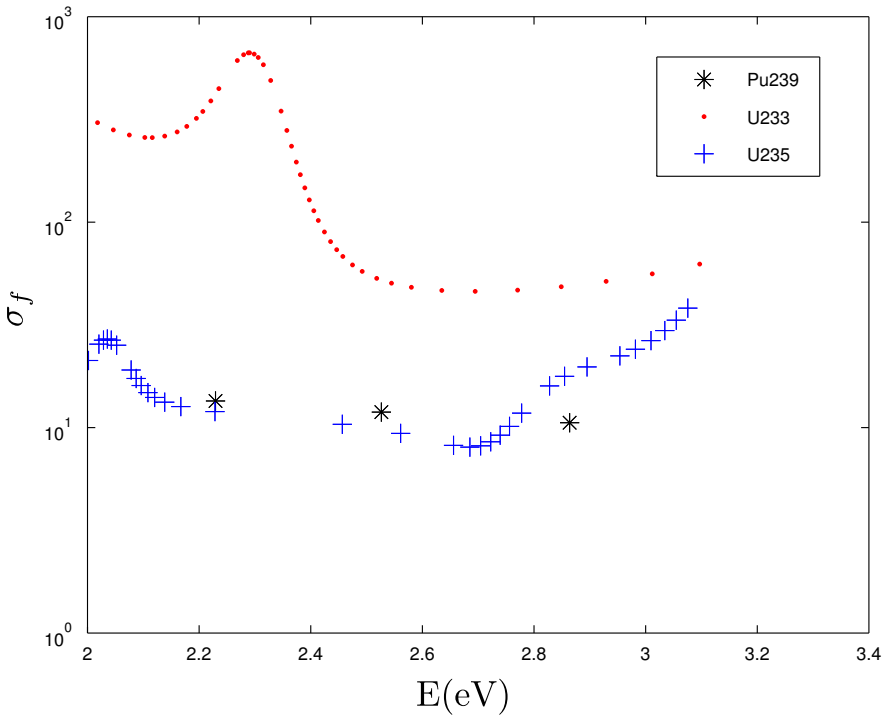

Figure 3.1: Energy grid for each nuclide

Each point in the graph represents one point from a data table. When the cross sections do not change much as function of the energy very few points are needed, as is the case for ^{239}Pu. On the other hand a lot of points are necessary when the cross section varies much as a function of energy. This can be seen from the blue line in Fig. 3.1, which is described by many more points than ^{239}Pu in the same graph. Especially where the function value changes fast, a lot of points are given to make linear interpolation possible between these points.

In Fig. 3.1 the separate cross section are plotted, but in Eq. (3.2) the sum over all these cross sections has to be calculated, weighted with the nuclide density and the energy released per fission reaction. These weights are constant per nuclide and independent of the energy of the incoming neutron.

The cross sections in Eq. (3.2) for each fissile nuclide have to be found through interpolation and the contributions have to be added to the total. When a material consist of a large number of nuclides, usually 30 or more for fuel pins in a nuclear reactor, this process takes a relatively long time. For this reason the summation is carried out at the start of the calculation.

3.1.3 Construction of a unified energy grid

After the summation over all nuclides, see Eq. (3.2) has been done, there is no longer an energy grid for each distinct nuclide but a single energy grid which contains all the energy points of all fissile nuclides in the material. The result of the summation of the 3 nuclides from Fig. 3.1 is shown in Fig. 3.2.

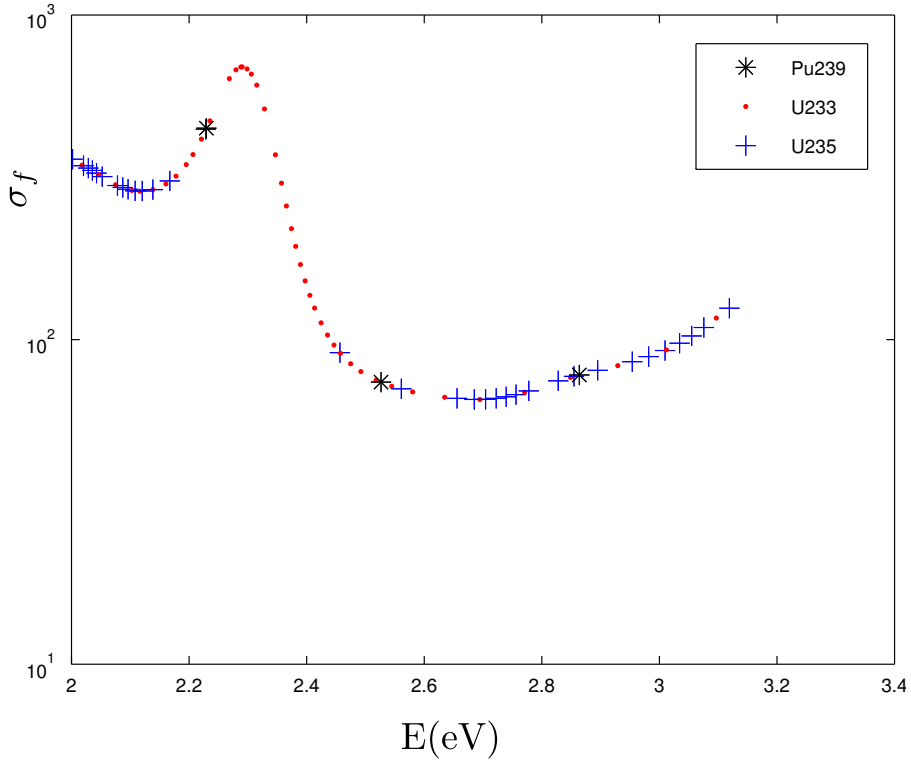

Figure 3.2: Unified energy grid, containing the sum over all cross sections for all three nuclides

The total number of points in the graph is the same as in Fig. 3.1, but the function value as a function of energy is the summation over the cross sections for the three nuclides.
When the local power density is requested during the simulation of neutron histories, the interpolation has to be performed only once, and no summation has to be done.

3.1.4 grid thinning

The disadvantage of this method is that the new energy grid can become very large and has to be stored somewhere in memory, together with the result of the summation.
Because the new grid contains all energy points from all nuclides in the material, it might contain more points than necessary to perform accurate interpolation. The process of removing all the points from the grid which are not necessary for accurate interpolation is called grid thinning.

Not all of the points can be removed from the grid, even when they are very close to some other point. Points which should not be removed are points at:

- Local cross section minima and maxima.
- Minimum energies of threshold reactions

as these points characterize the cross sections. How many other points one wants to use depends on the accepted error τ. Points are not included in the grid when the relative difference between two succeeding energy points is larger than the accepted error, as expressed in Eq. (3.4)

$$\frac{E_j - E_{j-1}}{E_{j-1}} < \tau \tag{3.4}$$

The further away points are from each other, the larger the error in the result will be, but less memory has to be used for storage. Besides the amount of required memory there is a second reason to perform grid thinning: in a smaller grid interpolation takes less time. The effect of grid thinning on the memory usage as function of τ is presented in Table 3.3

Table 3.3: Number of energy grid points and memory usage for different levels of grid thinning in the test case. [3]

τ	Grid points	Memory usage (Mb)
0	796,804	8.4
10^{-6}	737,333	7.83
10^{-5}	449,637	5.14
10^{-4}	165,118	2.27
10^{-3}	83,824	1.24
10^{-2}	70,859	1.06
10^{-1}	69,035	1.03

3.1.5 Double indexing

After grid thinning the grid possible still consists of over 100,000 points, which means that interpolation still needs a maximum of 17 steps to find the correct value. Double indexing is another method to reduce the time it takes to find a certain value in a logically ordered array by making a good first estimate of the index.
Double indexing does not incorporate changes in the energy grid, but makes use of a second, smaller grid. This second grid contains references to the indexes of the unified grid. To find a value in the unified grid first table lookup in the second grid is used to reduce the time needed for table lookup in the unified grid.
In this research even more logic is applied to the second grid; to make interpolation no longer necessary the energy steps in the second grid are of equal size in energy. In Fig. 3.3 the second grid is displayed on top of the larger unified grid.

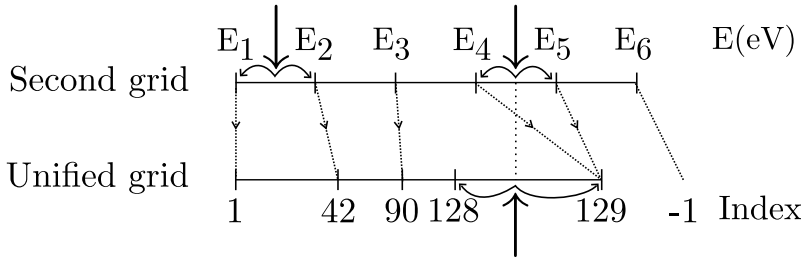

Figure 3.3: Double indexing

In the second grid the steps in energy are equal to 1 eV while in the unified grid there is no constant step size between the points. In Fig. 3.3 both grids start at 1 eV. From energy E_1 is a reference to index number 1 in the unified grid, which means that at index number 1 of the unified grid is the first value equal to or larger than E_1. When the unified grid does not contain a value equal or larger than the value in the second grid, the reference points to -1 to indicate the absence this value.

To find a certain energy in the unified grid, the second grid is used to make a first approximation of the index of that energy. To find a value in the unified grid around 1.5 eV, indicated by the first arrow in Fig. 3.3, first the second grid is used. 1.5 eV is between E_1 and E_2 in the second grid. E_1 refers to index 1 of the unified grid and E_2 in the second grid refers to, say, index 42 in the unified grid. In the unified grid table lookup is used to find the correct value. But this interpolation has only to be done between index 11 and 42, and not in the entire grid.
When 2 points in the unified grid are further away than the spacing in the second grid, as is the case between E_4 and E_5, two references in the second grid will refer to the same index.
When a value is requested in such region, between E_4 and E_5, both the references point to the same index. This means the requested value must be in between this index and the previous index and no table lookup has to be done. In this case the value lies in between the indices 128 and 129.

Because the energy density varies over specific regions it is chosen to split the second grid into three different grids.

- linear for $E < 1$ eV
- logarithmic for 1 eV $< E < 10^4$ eV
- linear for $E > 10^4$ eV

In the grid below 1 eV the energy density is more or less constant so a linear grid is used for the second grid. The same is true for energy above 10^4 eV, but the region with all the resonance frequencies is in between these two energies, as can be seen in Fig. 2.2. Because the amount of points in this region is so large a logarithmic scale is used with base 10. The energy represented by point n is not $E_n = n \times \Delta E$, but $E_n = \ln(E_0) - n \times \Delta u$ where u is the lethargy.

3.2 Relation between the geometry and the tallying process

To understand how the tallying in MCNP works it is necessary to understand how the geometry of a nuclear reactor is constructed. The construction of a geometry in MCNP5 starts with the smallest element which occurs multiple times in the geometry.
The basis element within a nuclear reactor is a fuel pin. This pin is a cylinder with a height close to 4 meter and has a diameter around 8 mm. The center of this cylinder contains the fissile material, surrounded by cladding, which separates the fuel and the coolant. This unit cell is shown in Fig. 3.4.

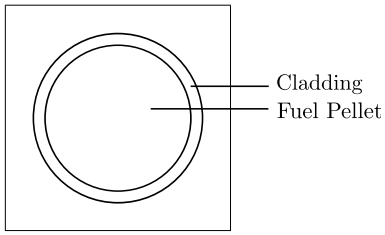

Figure 3.4: Fuel Pin

For designing purposes the cylinder is surrounded by a box filled with coolant, but because the cylinder is surrounded by other cylinders there is no physical boundary at the edge of the box.
These pins are used to make a larger construction part for the reactor: the fuel assembly. In this research the fuel assemblies consists of 264 fuel pins and 25 control rods ordered in a square of 17 x 17. A fuel assembly is shown in Fig. 3.5.

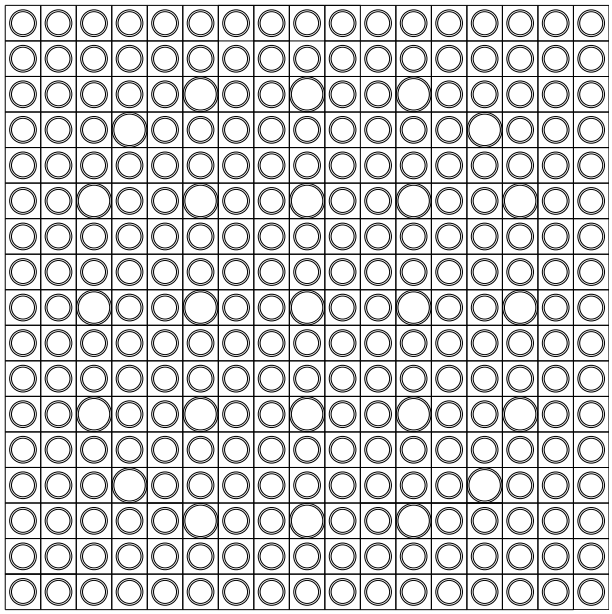

Figure 3.5: Fuel assembly constructed with 289 fuel pins.

In the reactor core these fuel assemblies are placed in a more or less circular pattern. Fig. 3.6 shows a realistic configuration of a core with 241 fuel assemblies.

		-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8		
8							1	2	3	4	5	6	7							
7					8	9	10	11	12	13	14	15	16	17	18					
6				19	20	21	22	23	24	25	26	27	28	29	30	31				
5			32	33	34	35	36	37	38	39	40	41	42	43	44	45	46			
4			47	48	49	50	51	52	53	54	55	56	57	58	59	60	61			
3		62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78		
2		79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95		
1		96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112		
0		113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129		
-1		130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146		
-2		147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163		
-3		164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180		
-4			181	182	183	184	185	186	187	188	189	190	191	192	193	194	195			
-5			196	197	198	199	200	201	202	203	204	205	206	207	208	209	210			
-6				211	212	213	214	215	216	217	218	219	220	221	222	223				
-7					224	225	226	227	228	229	230	231	232	233	234					
-8							235	236	237	238	239	240	241							

Figure 3.6: Realistic configuration of a reactor core with 241 fuel assemblies. The assemblies are numbered from 1 to 241.

3.2.1 Current tallies in MCNP5

Within MCNP an input file is used to specify what information has to be calculated, and what kind of estimators should be used to calculate this.
It is possible, for example, to calculate the power produced in the reactor, in a fuel assembly, in only one pin in an assembly or even in a small part of a fuel pin. This can be calculated with both a collision estimator and a track length estimator.

Tallying with a collision estimator

In MCNP it is possible to use a collision estimator, Sect. 2.4.2, to calculate the local power production in a volume. To do this the part of the geometry has to be specified where this tallying has to be done.
When tallying takes place, MCNP works in the opposite direction as the geometry is constructed. It starts with the entire reactor, than it searches through all fuel assemblies and finally it locates in which fuel pin fission has taken place.
Internally MCNP works with a list which contains all the places where a tally has to be scored. It checks every item in this list versus the position of the fuel assembly is which fission has taken place. When the local power production in each fuel assembly is requested, it checks for every fission event in the reactor each single fuel assembly.
The same process is used for the tallying in the fuel pins.

When a reactor consists of 241 fuel assemblies and every fuel assembly contains 289 fuel pins, it takes 241 + 289 steps to find a single fuel assembly and a specific pin within this assembly. This process takes a very long time when the amount of requested tallies increases. The time it takes as function of the amount of tallies is shown in Fig. 3.7

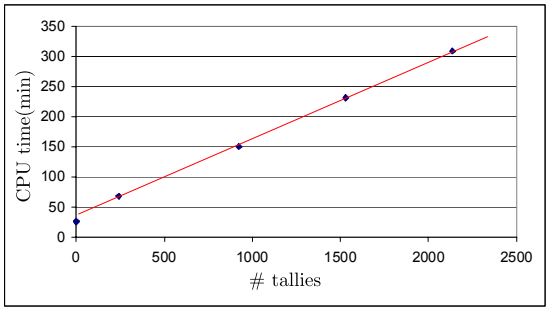

Figure 3.7: Calculation time versus the number of tallies

From this figure it is possible to make an estimation how much time it would take to calculate the power production in the reactor shown in Fig. 3.6 given that each fuel pin is split up in 100 separate axial parts. This would bring the total amount of volumes around 7 million tallies.

Tallying with a mesh tally

As shown in the previous paragraph it takes a long time to calculate a large amount of tallies when a collision estimator is used. Besides this every single tally has to be specified in a input file, what would result in a file of 800,000 lines for this geometry, which is very unpractical. To calculate the power production for a large amount of tallies in a geometry a mesh tally is used. A mesh tally superimposes a mesh over the geometry in a rectangular (or cylindrical) form. This mesh is divided in K, L and M parts for respectively the x, y and z direction. To score for all regions in the entire reactor shown in Fig. 3.6 a rectangular mesh is used divided in 289 parts in the x and y-direction and in 100 parts in the z-direction. The number of axial parts in the z-direction is related to the precision of the calculation and not to physical boundaries. The separation in the horizontal directions is because of the 17 fuel assemblies times 17 fuel pins per fuel assembly on each row and column of the reactor. A horizontal area of 1 by 1 in the mesh grid is now equal to one fuel pin.
To calculate the score in each part of the mesh a track length tally, discussed in Sect. 2.4.2, is used. The track length tally is used to estimate the flux in this part of the volume. When the flux is multiplied with the nuclide density, the cross section and the fission energy per fission the contribution to the local power is obtained. The only way for a neutron to give a contribution to this local power production is to travel through a fuel pin, which contains fissile material. Along the rest of the track of the neutron the fission cross section of the material is zero, and no fission energy is produced.
The path of a neutron through a unit cell is shown in Fig. 3.8.

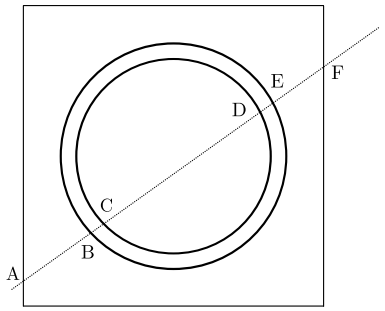

Figure 3.8: Path of a neutron through a unit cell

To calculate the total contribution of a neutron the contribution along each part of the track is calculated. The first contribution for this neutron is between A, where the neutron enters this unit cell, and B, where the material changes. Because all fission cross sections for the materials between the fuel pins is zero, no contribution is made in this part of the trajectory. The only non-zero contribution comes from the distance the neutron travels through the fuel pin, between C and D. Along the path between D and E the contribution is zero again, because there is no fissile material in the cladding, but all the calculations are done to come to this answer.
And that is the main disadvantage of this estimator. When the neutron travels through non-fissile material the local power production is calculated for each nuclide in this material. This means all the cross section are collected as function of the energy as well as the fission energy released per fission and the nuclide density, while is it possible to know the only contributions can come from the fuel pins. It is only not possible to do this with a mesh tally. An other disadvantage of a mesh tally is that it can not calculate the correct statistics for separate fuel assemblies or for the entire reactor.

Translation from position to index

The mean difference between the two tallies described in the previous sections is that the tally with the collision estimator, the f7 tally, is directly related to layers in the geometry, while the fmesh tally is not related to this layers. The advantage of the f7 tally is that it is possible to specify a small region for the power calculation, while the fmesh tally is more suitable for a large number of tallies.

The mean problem with the f7 tally is the time it takes to find a specific volume in the reactor. The specific location is required, because the exact location is directly related to the position in memory where the score for the volume is stored.
To find the exact position of a volume in the reactor is the most time consuming process in the f7 tally. For this reason a new relation is made between the position of volumes in the reactor and the position in the memory where the score is stored.

As described in Sect. 3.2 each fuel assemblies consist of 289 fuel pins arranged in 17 columns with 17 fuel pins as shown in Fig. 3.5. To save the scores for each fuel pin in this assembly an array of length 289 is required.
To save the scores in an efficient way a number, which corresponds to the index of the array where the score has to be saved, is assigned to each pin. A simple numbering is applied to all pins in the assembly: the first row of fuel pins is numbered from 1 to 17, the second row form 18 to 34 and the final row from 273 to 289. When fission has taken place in one of these pins the score can be

immediately saved.

The numbering of the fuel assembly is a little harder, because the fuel assemblies are not ordered in a square, but in a kind of a circle. Because of this somewhat unordered structure it is not possible to save the contributions of the fuel assemblies in the same efficient way as with the fuel pins.

To save the contributions for the fuel assemblies a number is assigned to each fuel assembly. This numbering is shown in Fig. 3.6. The assembly at position (-3, 8) has number 1 and the one at position (3, 0) has number 124. These references are saved in an array of 1 by 289. This array has the size to contain all the number for fuel assemblies if they would have been in a square of 17 by 17. This is the size of the minimum square to surround all the assemblies. At the positions where no fuel assembly is, the reference points to the value 0 to indicate the absence of an assembly at this position.
The first 17 entries of this array is shown in Table 3.4.

Table 3.4: First row of array with translation from position to index

0	0	0	0	0	1	2	3	4	5	6	7	0	0	0	0	0

This table says that the contribution for power production in the assembly at position (-3, 8) should be saved at the first index of an array which contains the contributions for all fuel assemblies. The contributions for the assembly at position (3, 8) are saved at the 7^{th} index, and no contributions are saved for (4, 8) because at that position is no fuel assembly.
With this translation step it is possible to store data for the fuel assemblies in an efficient way as well as for the fuel pins in the assemblies.

CHAPTER 4

Results

In MCNP5 the scoring of the tallies takes most of the time of the entire calculation. The simulation of neutron histories in a full size reactor core takes about 7 minutes, while a run with 29,000 tallies takes approximately 800 minutes. For this reason it is important to score these tallies as efficient as possible to minimize the overall calculation time.
To calculate an accurate power distribution in the reactor core each fuel pin is divided in 100 smaller axial volumes, equally distributed in the axial direction. These smaller volumes indicate the precision of the calculation and are not related to physical boundaries in axial direction.
For this research a reactor is used with 241 fuel assemblies each containing 289 fuel pins [2]. When each pin is divided into 100 volumes in axial direction the total amount of volumes becomes 6,964,900 for this reactor.

A detailed description about how the different tallies are used in the current version of MCNP is given in section 2 and 3 of this chapter. In the fourth section a new tally is introduced.

4.1 Validation of the code

Because of only the way the score is calculated has changed, and not the calculation itself it is possible to compare the values calculated with the different kind of tallies. In Table 4.1 the results and their errors for both the old f7 tally end the new tally which uses the unified grid are presented.

Table 4.1: Results of the calculations with different tallies after 600 cycles with 1000 neutrons per cycle

	Old f7 tally		Tally with the unified grid	
Geometry	Score	Relative error	Score	Relative error
Reactor	4.80274	0.0013	4.80274	0.0013
Fuel assembly (-3, -8)	3.51125×10^{-3}	0.0623	3.51125×10^{-3}	0.0623
Fuel assembly (0, 0)	3.81536×10^{-3}	0.0188	3.81536×10^{-3}	0.0187
Fuel pin (0, 1)	2.10280×10^{-4}	0.2189	2.10280×10^{-4}	0.2189
Fuel pin (-8, -8)	1.72691×10^{-4}	0.2077	1.72691×10^{-4}	0.2077
27^{th} axial part	8.26727×10^{-8}	1.0	8.26727×10^{-8}	2.7020
57^{th} axial part	1.53643×10^{-6}	1.0	1.53643×10^{-6}	2.1018
Single collision	33.0621701443773	-	33.0621701438435	-

For each individual neutron it is possible to compare the score calculated with the old f7 tally and the result of the calculation with the unified grid. These values for the score are the same within the limits of the computers accuracy to calculate a single number, as is shown in the last line of the previous table.

Because the new tally stores its scores at a different way in the memory a second test is done. In this test half the fissile material in a single fuel pin has been replaced by coolant. If the results are correctly stored in the memory the power distribution of this fuel pin should show power production in the lower half of the pin and no power production in the upper half of the fuel pin. The power distribution in this fuel pin is shown in Fig. 4.1.

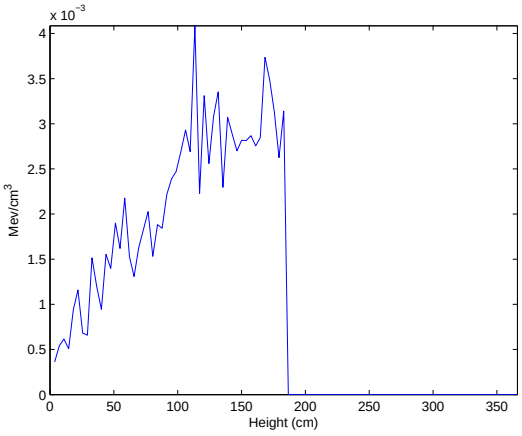

Figure 4.1: Half of the fissile material has been replace by coolant

From this figure can be seen the program works correctly. No fission energy is produced in the part of the fuel pin where the fuel has been replaced by coolant.

4.2 Power estimations with a collision estimator

As explained in Sect. 2.4.2 one of the current methods to tally a score is a collision estimator. The collision estimator counts and scores the number of collisions within a certain volume, which has to be specified by the user.
To specify an axial volume in a fuel pin, first the location of the fuel assembly has to be specified. Then the position of the fuel pin within this assembly and finally the location of the axial volume. This is also the way how MCNP5 searches for the location of a volume and why it takes so much time to tally axial volumes of a fuel pin in a reactor core.
Although the collision estimator could be the most efficient estimator of all, it only scores events which contribute to the final result, it is the slowest estimator when a large number of tallies is requested because of the long time needed to find a specific volume.
To be sure all tallies requested by the user are performed, MCNP5 saves all requested tallies in a list. When an event has taken place at a certain location MCNP5 compares this location with all entries of the internal list off locations. In the case of a few requested locations this is no problem at all, but as the number of requested tallies increases the extra calculation time increases as well. The time it takes to perform a number of tallies is given in Table 4.2

Table 4.2: CPU time for different number of tallies. All calculation are performed over 600 cycles with each 1000 neutrons.

Tally	# Tallies	CPU time (min)
No tallies	0	7
single fuel assembly	2	8
10 fuel assemblies	11	9
100 fuel assemblies	101	11.5
241 fuel assemblies	242	17
241 fuel assemblies + 100 axial parts of a single fuel pin	342	18
241 fuel assemblies + 100 axial parts of 5 fuel pins	742	25
100 axial parts of 289 fuel pins in a single fuel assembly	28901	770

From the CPU times per calculation given in Table 4.2 follows that the time it takes to score a number of tallies does not linearly depend on the number of tallies. It matters as well in which layer of the geometry the scoring has to take place. As described in Sect. 3.2 the geometry consists of a collection smaller unit cells. Each layer consists of one type of smaller unit cells. The reactor core itself is the top layer in the geometry. The layer below consists of the fuel assemblies, which are collections of fuel pins. The fuel pins are the smallest unit cells in the geometry of a reactor.
MCNP does not use a single list to save all the tallies which have to be performed, but uses a list for each layer of the construction. For each item in the list a location in memory is created to save the score for this item in the list. The list of the reactor, the top layer, contains only one index; there is only one reactor core in the geometry. For this reason it never takes much time to score the total power production in the reactor.
When this score is combined with the score of a single fuel assembly MCNP uses 2 lists, each containing one item. This means it is still easy to find whether fission has taken place in the fuel assembly in the list or not.
When 10 separate fuel assemblies have to be scored together with the reactor core MCNP still uses 2 lists, but the second list now contains 10 items. When a fission event has taken place all these 10 items are checked. For this specific calculation on a single 2 GHz processor it takes 2.2 seconds for every requested fuel assembly.

The dependence of the computation time on the different layers of the geometry is visualized in Fig. 4.2, where the number of tallies versus the calculation time per calculation from Table 4.2 are plotted.

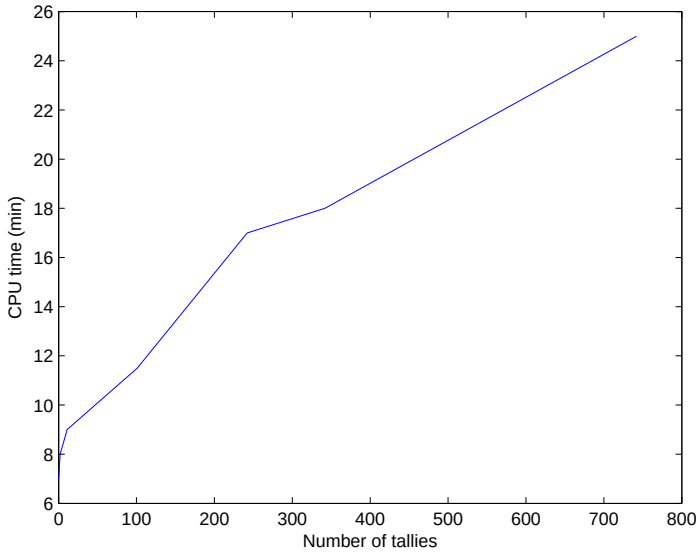

Figure 4.2: CPU time for calculations with a different number of tallies in different layers of the geometry

Scoring individual pins within a fuel assembly takes more time per requested tally, because the pin is in a layer below the fuel assemblies in the geometry. First the right fuel assembly has to be found. When the correct assembly is found, the fuel pin in the assembly has to be found. How much time this takes depends on the location of the assembly in the reactor and the location of the pin in the assembly. For this reason there is no fixed time for calculating the score for a single fuel pin.
The calculation of the score for the fuel pin is independent of the scoring process for the fuel assembly, and also the other way around. When the score for the fuel pin is known, MCNP starts a new calculation to score the contribution to the fuel assembly. Even the value of the score is recalculated.

The smallest part of the entire geometry is a subsection of the fuel pin. These parts are even smaller than the fuel pin and for this reason it takes therefore more time to find these volumes in the geometry of a reactor core and the calculation time increases faster per tally.
A calculation in which each fuel pin is split into 100 smaller parts would take over 180,000 minutes or more than 4 months.

4.3 Power estimations with a track-length estimator

As described in the previous section it is not practical to calculate a detailed power distribution of a nuclear reactor or another large object with a lot of smaller volumes when using a collision estimator in MCNP. It takes to much time in those cases to find the exact location of the tally,

which is directly related to the location in memory where the score has to be saved.
To solve this problem a fmesh tally is used. Because the mesh of the fmesh tally is not directly correlated to the geometry it does not take more or less time to calculate the score after an event in different layers of the geometry. This is presented in Table 4.3

Table 4.3: CPU time for different number of mesh tallies. All calculation are performed with 4 parallel processors over 600 cycles with each 10^5 neutrons.

Tally	CPU time (min)
2 Fuel assemblies	300
13 Fuel assemblies	302
100 axial parts for each pin in 13 assemblies	331

The number of neutrons used in this calculations is 100 times larger than the population size for the calculations with the collision estimator.
In comparison with the results from the collision estimator, represented in Table 4.2, it takes much less time to calculate scores in the lowest layer of the geometry, as the fmesh does not use the geometry of the construction to score an event. It only checks whether the fission occurred within the region of the fmesh or not. When the fission event had taken place into a region of the fmesh, it is immediately known where to save the score in memory.

The disadvantage of the fmesh tally is that it uses a track-length estimator to calculate the score for a single neutron. As explained in Sect. 2.4.2 the track-length estimator calculates the score for a neutron along the path it travels. This process is less efficient than scoring a collision, because the neutron travels through all kind of materials where fission can not occur, and therefore gives no contribution to the power production.
The problem of double scoring has not been solved either with this fmesh tally.

The most efficient tally would be a collision estimator but without the problem of the geometrical complexity as described in Sect. 4.2. With the fmesh tally the problem of the geometrical complexity is solved by uncoupling the tally and the geometry, but a track-length estimator has to be used for the power density calculation.
In the next section a new tally is introduced where a collision estimator is used without the geometrical complexity of the f7 tally. The problem of double scoring is solved in this tally, because the geometrical structure of the reactor is used.

4.4 Power estimation tally for a detailed local power calculation

At this moment the tallying in MCNP, the fmesh tally excepted, goes from the largest geometrical part, the reactor, to the smallest part, a (axial volume of the) fuel pin. The construction of the reactor starts with the smallest part of the reactor and creates the larger part with this small part. A collection of pins forms a fuel assembly, and a number of fuel assemblies forms the reactor.
When a score has to be calculated MCNP works in opposite direction. It starts with the reactor, then it finds the fuel assembly and finally the fuel pin in which fission has taken place. This is the only way in which a single fuel pin can be distinguished from a fuel pin at the same position in another fuel assembly. And only when the exact location of the fuel pin is known it is possible to

store the score for this pin at the right part of the memory.

4.4.1 Relation between the geometry and the position where the score has to be saved

Although tallying in MCNP goes from the reactor core to a fuel assembly and then to the pin to find the location of an event, this is not how the fission energy is produced in the reactor. Fission takes place in a fuel pin, where the fission energy is released. Because this pin is part of a fuel assembly, the energy is produced in the fuel assembly as well. The fuel assembly is a part of the reactor, which means energy is also released in the reactor core. But the fission events only take place in the fuel pins, while the score must be saved for multiple parts in the geometry.

To use this information about the fission events one of the most elementary structures of MCNP has to be changed. At this moment MCNP works for every geometry one can design, but to process fission events in a nuclear reactor in a more efficient way this generality can not be saved.

For each neutron MCNP keeps track of its exact location within the geometry. This means the position of the fuel assembly is known, as well as the fuel pin where the neutron caused a fission reaction. But this information is not coupled to the requested tallies and not used when the scoring calculations are performed.
In the tally introduced in this section the structure of the reactor is used to store scores in a more efficient way. Information about the exact position of the neutron is used to save the score for that neutron directly for the fuel pin, the assembly and the reactor.
Scoring the power production for the entire reactor is easy, while there is only one reactor, and the value can be saved as a single number. Saving the score for the fuel assemblies is a little harder, but this is solved with a connection between the physical position of the fuel assembly and the position in memory where the score for the assemblies has to be saved, as described in Sect. 3.2.1.
Scoring the contribution for the fuel pin is again straightforward because the fuel pins are placed in an ordered square within the assembly.

Because the collision estimator is a very efficient way of tallying the local power production, this estimator is used in the new tally. When the power production is known, this has to be stored in memory at a number of places. First the score has to be saved for the axial volume of the fuel pin in which the fission has taken place. Next the score has to be saved for the fuel pin, the fuel assembly and finally for the reactor. The value of the score is identical in all cases, because they are caused by the same fission event.
The scoring for the axial volumes of the fuel pins is The scoring has to be saved for all these parts of the geometry. Although the total power produced in a fuel assembly is the sum of the power produced in all the fuel pins within this assembly, the square of the scores is required to calculate the variance and the relative error for the fuel assembly. For this reason the score has to be saved for both the fuel assemblies and the reactor as well as for the individual fuel pins during a neutron history.

Saving scores for the neutrons in this way decreases the calculation time for the last tally described in Table 4.2 from *971 minutes to 629 minutes* under the same conditions, which is a 35% decrease in calculation time. This time is independent of the total number of requested tallies and the

statistics are known for all parts of the reactor and not only for the smallest requested parts. This means no other simulations have to be done for calculating the statistics for the fuel pins, the fuel assemblies and for the reactor core.
To compare this results with the f7 tally, only a comparison with a simulation where 100 axial volumes per pin for all fuel pins in a single fuel assembly are calculated can be made. Other calculations would take too much time with this tally. The calculation for these 28,900 tallies is done 86 times as fast with the new estimator, and this number would only increase with the number of requested tallies.

4.4.2 Tallying with the advanced power scoring function

To decrease the calculation time as much as possible the unified energy grid, as described in Sect. 3.1.3, is used to calculate the score for each neutron. This method decreases the time to calculate the power produced at a fission reaction with a factor four. This is because a single table lookup is done instead of a table lookup for each nuclide in the material and no summation over these scores has to be done. But the information that would be found in the table lookup process is required at other parts in the neutron simulation, and for this reason a lot of the table lookups has to be done anyway.

Grid thinning As described in Sect. 3.3 grid thinning can be used to decrease the amount of points saved in the unified energy grid. When half of the original points is removed from the reduced energy grid, the iteration process will take on average one step less to find a certain value. With an accepted error less than 10^{-4} the unified energy grid becomes 4 times as small, which saves 2 steps in the iteration process and saves 3 quarters of the memory required to save all the data.
It appeared that the use of grid thinning does not reduce the calculation time significantly, but one can save 3 quarter of the memory while the accuracy changes less than 0.01%

Double indexing Double indexing is the second method to reduce the iteration time in the unified energy grid. As described in Sect. 3.1.5 the reduced energy grid is split in three regions; two linear parts and one logarithmic part.
In the two linear parts a good estimation can be made in a short time, but to find a certain value in the logarithmic part takes so much time all the benefits from double indexing are canceled out. Describing the entire spectrum with a linear grid does not save much time either and for this reason double indexing is not implemented in the final version of the program.

4.4.3 Time reduction of the new tally

Using a reduced energy grid reduces the calculation time from 629 to 622 minutes and uses 2 to 8.5 Mb of memory depending on the accepted error τ. The results of all tallies are summarized in Table 4.4.

The calculation time for the collision estimator for the entire reactor is an estimated value, and not the result of an actual calculation. The results of grid thinning are not presented in this table, because the reduction in calculation time is in the order of half a minute and the effect on memory usage is given in Table 3.3.

Table 4.4: Calculation time for 6,964,900 volumes with different kind of tallies.

Tally	CPU time (min)
Collision estimator	2×10^7 (estimated)
Fmesh tally	971
New tally with the connection between geometry and memory index	629
New tally with both geometry connection and reduced energy grid	622

4.4.4 Limitations

This new tally uses the fact that a nuclear reactor is build in an organized structure. In the reactor used as model for this research the reactor consists of 241 fuel assemblies each containing 289 fuel pins. Although not all reactors are build this way, most reactors are constructed with fuel assemblies containing smaller elements.
As long as the reactor is designed along these lines it is possible to use this tally to do detailed power calculations. At this moment most reactors have fuel assemblies in a logical ordered structure. Some of them contain 15x15 fuel pins per assembly, or the assemblies are build out of plates instead of cylinders but this does not effect the applicability of the calculation method.

4.5 Results of the power distribution in the reactor core

The total amount of neutrons and number of cycles used in a calculation is related to the accuracy of the result. The larger the total amount of neutrons, the number of neutrons per cycle times the number of cycles, the more accurate the final result will be. According to Eq. (2.14) the variance of the calculation decreases with one over the square root of N.

The results presented in this section are the result of a simulation with 600 cycles with 10^5 neutrons per cycle. In Fig. 4.3 the power distribution of the reactor core is shown.

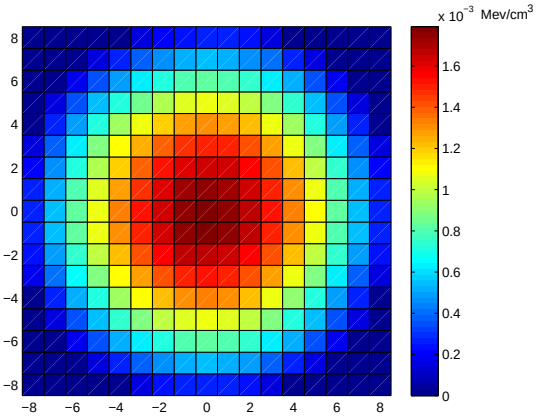

Figure 4.3: Power distribution in the reactor core

The colors indicate the energy production per unit volume. From this figure it can be seen that

the reactor consists of 241 fuel assemblies, each colored block represents a single fuel assembly. The dark blue squares in the figure consists of water or other non fissile material and the power production in these squares is zero.
The power production has a maximum at the center of the reactor core and decreases as the fuel assemblies are further away.
With Eq. 2.21 the relative error for the power produced in the entire reactor can be calculated. For this calculation the relative error is 0.018%.

It is also possible to zoom in into a single fuel assembly within the core. The central assembly from the reactor core is shown in Fig. 4.4

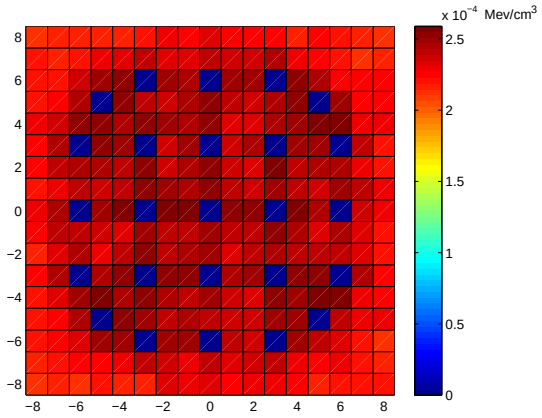

Figure 4.4: Fuel assembly at the center of the reactor core.

This fuel assembly has the highest power production of all the assemblies, because there is no leakage at the center of the reactor core. At the positions of the blue squares there are no fuel pins, but only borated coolant and guide tubes for the control rods. At these positions no fission takes place and the power production is equal to zero. The relative error for the power production of this fuel assembly is 0.18% and 0.97% on average for the fuel pins in this assembly.

The fuel assemblies at the boundary of the core show a different power distribution along the assembly. More power is produced at the places closest to the core, while less power is produced at the outsize of the core. The power distribution of the fuel assembly at position (-5, -7) in Fig. 4.3 is shown in Fig. 4.5.

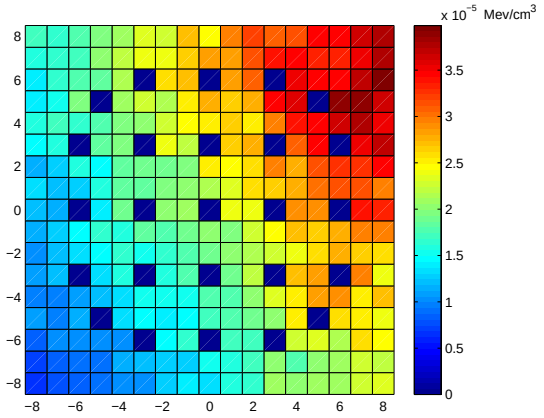

Figure 4.5: Fuel assembly at position (-5, -7) in the reactor.

In this fuel assembly much less power is produced. This fuel assembly is at the side of the core where most leakage takes place. Most leakage takes place at the lower left side of the assembly, colored blue in the figure. The upper right part is closer to the center of the core where less leakage occur and more power is generated. Because less energy is produced in this assembly the relative error is larger: 0.60% for this fuel assembly.

It is also possible to zoom in to a single fuel pin within this assembly. Most power is produced at the center of the pin and the lowest energy production is at the top and bottom of the pin as a result of leakage. This is shown in Fig. 4.6.

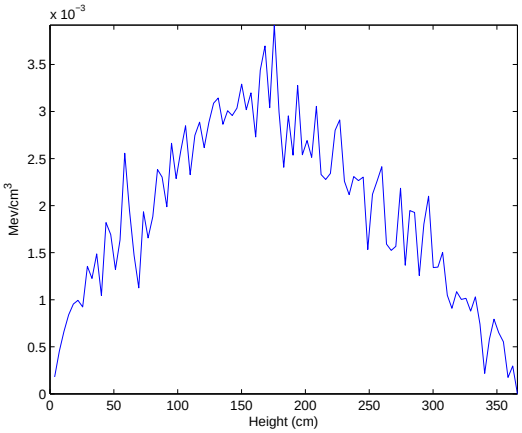

Figure 4.6: Power production within a single fuel pin.

When an infinite amount of neutrons is used in the calculation this line would be a cosine. Because the amount of neutrons in the calculations was limited the error in the power distribution is in the order of 10 %. The same figure is shown in Fig. 4.7, but in this case the error bars indicate the uncertainty in the calculation ($\pm 1\sigma$).

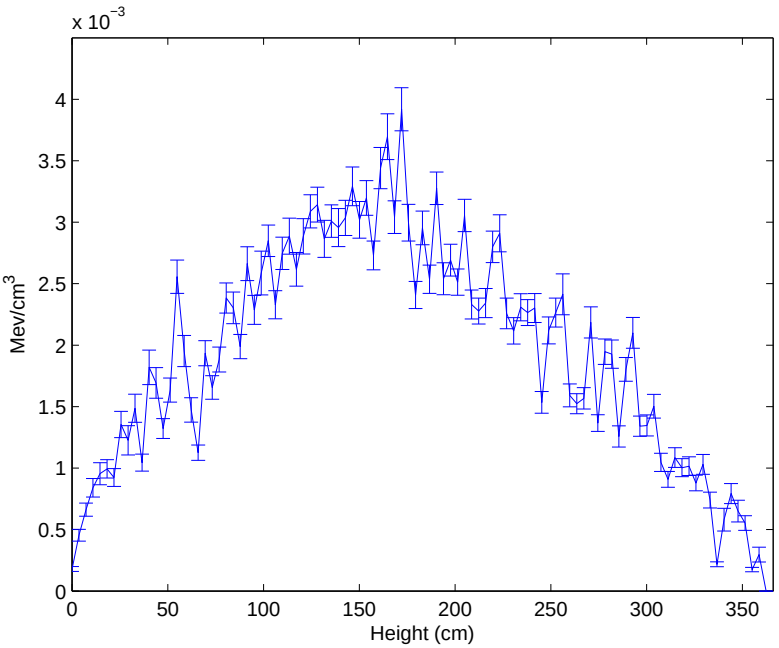

Figure 4.7: Power production within a single fuel pin with errorbars.

The relative error in this figure is in the order of 10%. To decrease this error to 1% 100 times more neutron are required. This means that the calculation takes 100 times longer, 1,5 months with a single processor, to receive a certainty of 1% within a one hundredth part of a fuel pin.

When the power distribution of the fuel pin is compared with a fuel pin at the outside of the reactor it becomes clear that not enough neutron have cause fission in this part of the reactor to calculate a reliable flux profile at this point of the reactor. The power distribution of the fuel pin at the lower left corner of the fuel assembly at position (-5, -7) is shown in Fig. 4.8

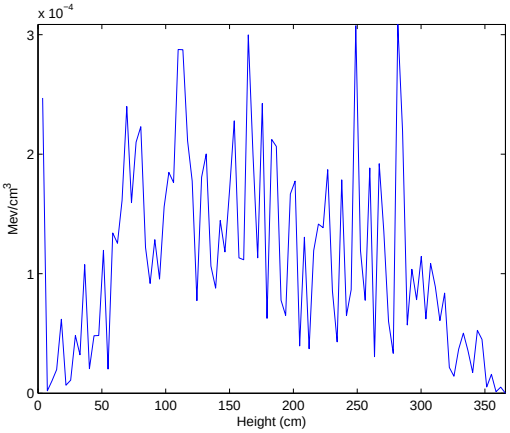

Figure 4.8: Power production from a pin at the outside of the reactor.

From this figure it becomes immediately clear that the relative error for this fuel pin is very large.

CHAPTER 5

Conclusions

The goal of this research was to decrease the calculation time for a detailed power distribution calculation in the general purpose Monte Carlo code MCNP5. With this version of MCNP it is already possible to calculate a detailed power distribution for a nuclear reactor, but this calculation is not done in the most efficient way.
For this calculations a model of a nuclear reactor is used constructed with 241 fuel assemblies each with 17x17 fuel pins, and each pin is divided in 100 axial parts to obtain a detailed power distribution of the core. For this geometry it takes 971 minutes to do a simulation with 600 cycles and 10^5 neutrons per cycle. In this report two methods are discussed to decrease this calculation time; in the first method the contributions to the power distribution are calculated in a more efficient way and in the second method the position of a volume in the geometry is directly related to the location in memory where the score has to saved. Because of this direct relation between the position in the geometry and the location in the memory it takes less time to save a score at the right location in memory.

Efficiency improvement of the local power density calculation

With a unified grid the calculation time for the local power production for a single fission reaction takes four times as less time. However, the information that would have been obtained during this calculation is required for other calculations in the simulation of neutron behavior as well. In these cases the calculation has to be performed anyway.
The final decrease in total CPU time as result of using the unified grid is in the order of 1%.

Grid thinning

Because the result of the summation discussed in the previous section is much larger than required for accurate interpolation grid thinning is used. Is this process points which are not necessary for accurate interpolation are removed from the grid. Grid thinning does not significantly decrease the calculation time of the simulation, but the amount of memory to store the unified energy grid can be reduced by a factor 4 without significant loss of accuracy.

Double indexing

After grid thinning the unified grid can still contains over 70,000 energy points. To reduce the table lookup time to find a certain point in this grid a second grid is created. This second grid does not influence the unified grid, but contains references to this unified grid. This second grid is much smaller than the unified grid to reduce the table lookup time. Unfortunately double indexing takes so much time for internal calculation that the time benefits are canceled out and double indexing is not implemented in the final code.

Relation between the geometry and the tallying process

The calculation of the local power production takes relatively short time. The mean problem is to find the exact position of the volume in the reactor geometry. This position is directly related to the location in the memory where the scored has to be stored.
In this research all volumes in the reactor are numbered in a logical way, and the number is directly related to some location in the memory where scores for this volume have to be stored. In this way the score can be directly stored in memory. By applying this numbering the overall calculation time for a simulation with 7 million tallies can be decreased by 36% off the total CPU time.

Further work

At this moment the geometrical structure is implemented by hand, but it could be done by some calculations. Al the volumes in which scoring has be done have to be numbered from 1 to N. A number has to be assigned to a volume when the score of the volume is requested by the user and the volume contains fissile material.
Another extension of this work would be the implementation of the improvements for parallel calculations

List of Figures

List of Tables

References

ENDF/B-VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technologly . *Nuclear Data Sheets*, 107:2931–3060, December 2006.

J. Eduard Hoogenboom, William R. Martin, Bojan Petrovic. Monte carlo performance benchmark for detailed power density calculations in a full size reactor core. `http://www.nea.fr/html/dbprog/MonteCarloPerformanceBenchmark.htm`, November 2009.

Jaakko Leppänen. Two practical methods for unionized energy grid construction in continuous-energy monte carlo neutron transport calculation. *Annals of Nuclear Energy*, 36:878–885, May 2009.

X-5 Monte Carlo Team. MCNP-A General Monte Carlo N-Particle Transport Code, Version 5. Los Alamos National Laborary, 2008.