Riku Tuominen

Coupling Serpent and OpenFOAM for neutronics – CFD multi-physics calculations

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 30.7.2015

Thesis supervisor:

Professor Filip Tuomisto

Thesis advisor:

M.Sc. (Tech) Ville Valtavirta

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Riku Tuominen

Title: Coupling Serpent and OpenFOAM for neutronics – CFD multi-physics calculations

Date: 30.7.2015 Language: English Number of pages: 7+64

Department of Applied Physics

Professorship: Tfy-56

Supervisor: Professor Filip Tuomisto

Advisor: M.Sc. (Tech) Ville Valtavirta

The main goal of this work was to couple the Monte Carlo neutronics code Serpent 2 with a CFD solver from the OpenFOAM toolbox. The coupling was implemented with the already available multi-physics interface of Serpent. The interface allows the passing of high fidelity density and temperature distributions from an external solver to Serpent and also the passing of fission power distribution from Serpent to the external solver. The coupled CFD-neutronics problem was solved by iteration. At each iteration Serpent solves a new power distribution based on the current temperature and density distribution. The power distribution is passed to the CFD solver to solve the corresponding temperature and density distributions which are passed back to Serpent to start a new iteration.

The coupling was tested by modelling a mock-up 5x5 fuel assembly cooled with water in a steady state condition at full power. The effects of boiling were neglected and the flow was modelled as single phase. The main results of the coupled calculation were high fidelity temperature and density distributions. The effect of distribution fidelity on neutronics was studied by running Serpent simulations with varying level of axial refinement. Most obvious differences were observed in the axial power density and in the collision density in the coolant, with minor differences in e.g. the effective multiplication factor and radial capture density in the fuel.

In addition performance tests related to the multiphysics interface were run and the convergence of the coupled calculation is briefly discussed. The work of this Master's thesis can be expanded in the future for example by including the effects of boiling in the CFD solution.

Keywords: Monte Carlo neutronics, CFD, multiphysics

AALTO-YLIOPISTO
PERUSTIETEIDEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Riku Tuominen

Työn nimi: Serpentin kytkentä OpenFOAM ratkaisijaan CFD multifysiikkalaskentaa varten

Päivämäärä: 30.7.2015 Kieli: Englanti Sivumäärä: 7+64

Teknillisen fysiikan laitos

Professuuri: Tfy-56

Työn valvoja: Professori Filip Tuomisto

Työn ohjaaja: Diplomi-insinööri Ville Valtavirta

Tämän työn päätavoite oli toteuttaa kytkentä Serpent Monte Carlo-koodin ja OpenFOAM paketin virtausratkaisijan välille. Kytkennässä hyödynnettiin jo olemassa olevaa Serpentin multifysiikkarajapintaa, joka mahdollistaa korkearesoluutioisten tiheys- ja lämpötilakenttien välittämisen ulkoiselta ratkaisijalta Serpentille ja toisaalta tehojakauman välittämisen Serpentiltä ulkoiselle ratkaisijalle. Kytketty ongelma ratkaistiin iteroimalla. Jokaisella iteraatiolla Serpent laski uuden tehojakauman senhetkisten lämpötila- ja tiheysjakaumien pohjalta. Tehojakauma välitettiin virtausratkaisijalle, joka laski vastaavat tiheys- ja lämpötilajakaumat, jotka välitettiin takaisin Serpentille uutta iteraatiota varten.

Kytkentää testattiin mallintamalla kuvitteellista vedellä jäädytettyä 5x5 polttoainenippua tasapainotilassa. Yksinkertaisuuden vuoksi virtausratkaisussa ei huomioitu kiehumista. Kytketyn laskun päätuloksena saatiin korkearesoluutioiset tiheys- ja lämpötilajakaumat. Näiden jakaumien tarkkuuden vaikutusta neutroniikkaan tutkittiin ajamalla Serpent simulaatioita aksiaalisuunnassa eri tarkkuuksisilla jakaumilla. Selvimmät erot havaittiin aksiaalisessa tehojakaumassa sekä törmäystiheydessä jäähdytteessä. Pienempiä eroja oli esimerkiksi efektiivisessä kasvukertoimessa sekä radiaalisessa kaappaustiheydessä polttoaineessa.

Lisäksi ajettiin suorituskykytestejä multifysiikka rajapintaan liittyen sekä pohdittiin lyhyesti kytketyn laskun konvergenssiä. Tässä diplomityössä tehtyä tutkimusta voidaan jatkaa tulevaisuudessa esimerkiksi ottamalla kiehuminen mukaan virtausmalliin.

Avainsanat: Monte Carlo neutroniikka, laskennallinen virtausdynamiikka, multifysiikka

Preface

This Master's thesis was carried out in the CFD group at VTT. I'm not much of a storyteller so I would just like to thank my great instructors Ville Valtavirta, Juho Peltola and Timo Niemi. Without your help this work would not have been possible. I am also grateful to Jaakko Leppänen for providing the topic of this Master's thesis, to Filip Tuomisto for acting as the supervisor of this thesis and to Lars Kjäldman for allowing me to do this work in his group.

Espoo, 30.7.2015

Riku Tuominen

Contents

Symbols

$\vec{U}$	velocity field
ρ	density or reactivity
τ	viscous stress tensor
$\mathbf{S}$	source term
μ	dynamic viscosity
λ	bulk viscosity or thermal conductivity
μ_t	turbulent viscosity
E	specific energy
κ	thermal conductivity
e	specific internal energy
h	specific enthalpy
T	temperature
p	pressure
Γ	diffusion coefficient
q	source term or heat flux
u	x-component of velocity
v	y-component of velocity
w	z-component of velocity
k	turbulent kinetic energy or multiplication factor
δ_{ij}	Kronecker delta
Σ	macroscopic cross section
α	thermal diffusivity or additional under-relaxation factor
c_m, c_p	specific heat capacity
P	power density distribution

1 Introduction

Nuclear reactor design and testing has relied on computational methods for decades. Computer simulations offer a cheaper and safer alternative to actual tests performed with real reactors. Unfortunately, accurate modelling of a nuclear reactor is a cumbersome task. It is not enough to only consider the neutrons of the core as several important feedback mechanism exist. Probably two of the most important ones are related to fuel burnup and thermal-hydraulics.

The main focus in thermal-hydraulics is the calculation of the density and temperature distributions which have a major effect on the neutron flux distribution. An increase in the moderator temperature results in a decrease in moderator density and moderating effectiveness. This in turn hardens the neutron spectrum which gives rise to negative reactivity addition in a thermal reactor. In the fuel the neutron flux is mainly affected by the Doppler broadening of the effective resonance cross sections. As the fuel temperature increases so does the resonance absorption of neutrons.

The availability of relatively cheap, high speed parallel computers has triggered a growing interest in the development of high-fidelity reactor simulation tools. These are intended to replace the less accurate methods used currently. One of the major goals is to increase the spatial resolution of the solution.

Different solutions to the coupled problem with neutronics and thermal-hydraulics are developed worldwide. The neutronics are solved either using multi-group deterministic or continuous energy Monte-Carlo methods. Thermal-hydraulic codes can be divided into subchannel solvers and computational fluid dynamics (CFD) codes. In this thesis the focus is on a coupling between a Monte Carlo code and a CFD solver. Publications on this topic are scarce probably because both methods demand a lot of computational power and only recently it has become possible to solve interesting test cases in reasonable time.

More research has been done with the coupling of subchannel solvers and Monte Carlo codes: In [1] the Monte Carlo N-Particle (MCNP) code was coupled with the subchannel code STAFAS (Subchannel Thermal-hydraulics Analysis of a Fuel Assembly under Supercritical conditions) to model a High-Performance Light-Water Reactor (HPLWR) fuel assembly. In [2] a coupling between MCNP5 and the subchannel code COBRA-TF was used to fine tune results obtained from a coupled nodal expansion method (NEM)/COBRA-TF code. The code system was tested by modelling a 3-D 2x2 pin array extracted from a Boiling Water Reactor (BWR) assembly with reflective radial boundary conditions. The research on this coupled hybrid Monte Carlo system was continued in References [3] and [4]. Reference [5] presents an internal coupling for MCNP5 and the subchannel code SUBCHANFLOW. A scheme to modify the density and temperature distributions on-the-fly in MCNP is explained. In addition, a stochastic approximation based relaxation scheme was implemented along with a collision density flux estimator. As a test

case, a 17x17 Pressurized Water Reactor (PWR) fuel assembly consisting of 16,380 thermal–hydraulic feedback cells was modelled.

In pursue of the goal to increase the spatial resolution of the solution, Serpent 2, a state-of-the-art Monte Carlo neutronics code developed at VTT, offers a multi-physics interface to pass high fidelity temperature and density distributions from an external solver to Serpent. The use of arbitrarily defined distributions is made possible by a special on-the-fly temperature treatment for microscopic cross sections referred as TMS.

The goal of this work was to develop an external coupling between Serpent and a CFD solver from the open source OpenFOAM toolbox. Previously Serpent has been coupled externally with the subchannel code SUBCHANFLOW [6]. For simplicity, only a steady state case with single phase flow was considered in this work. The coupled problem was solved by iteration. First Serpent calculates an initial power distribution which is passed on to the CFD solver. Next temperature and density distributions are solved based on the power distribution. New temperature and density distributions are passed back to Serpent and the iteration is continued until convergence.

The coupling was tested with a mock-up 5x5 assembly. The results of the coupled calculation were high fidelity temperature and density distributions. A separate Monte Carlo simulation was run with these distributions and the results were compared to the results of simulations with less accurate distributions. The comparison was done in order to find out how the additional fidelity affects neutronics. In addition Serpent performance and the convergence of the coupled calculation were studied.

2 Theory

2.1 Computational fluid dynamics (CFD)

In computational fluid dynamics (CFD) problems involving fluid flow are solved with computers using numerical methods and solution algorithms. CFD is a complex topic and this section gives only a very brief overview of the CFD theory. A more thorough overview can be found in many introductory CFD textbooks including An Introduction to Computational Fluid Dynamics: The Finite Volume Method by H. K. Versteeg and W. Malalasekera [7].

2.1.1 General equations

In fluid dynamics fluid flow is characterized by a set of equations. Basic principles of conservation of mass, momentum and energy can each be expressed as an equation for the fluid flow. Continuity equation describes the conservation of mass. The Navier-Stokes equation deals with conservation of momentum (Newton's second law) and the third important equation deals with energy. In addition, equations of state are used to provide linkage between these equations when necessary. A common practise in fluid dynamics is to use substantial or material derivative, which for physical quantity ϕ is defined as

$$\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \vec{U} \cdot \nabla\phi, \tag{1}$$

where $\vec{U}$ is the velocity field of the fluid. The substantial derivative describes the rate of change of quantity ϕ as experienced by an observer that is moving along with the fluid flow.

Continuity equation

In vector notation the continuity equation is given by

$$\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho\vec{U}) = 0, \tag{2}$$

where ρ is density, t time and $\vec{U}$ velocity field. The continuity equation states that the rate of change of the density in a fluid element (first term) is equal to the net flow of mass across its boundaries (second term).

With substantial derivative and continuity equation introduced the concept of incompressible flow is defined. If the density of a fluid element does not change during

its motion, the fluid flow is said to be incompressible, the opposite being compressible flow. By using substantial derivative for density ρ the condition for incompressible flow can be expressed as

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \vec{U} \cdot \nabla \rho = 0 \tag{3}$$

and from the continuity equation follows

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \vec{U} = 0 \tag{4}$$

or

$$\nabla \cdot \vec{U} = 0. \tag{5}$$

Navier-Stokes equations

The Navier-Stokes equations can be derived by applying Newton's second law to a fluid particle. General form of the equations is given by

$$\rho \frac{D\vec{U}}{Dt} = -\nabla p + \nabla \cdot \tau + \mathbf{S}, \tag{6}$$

where p is pressure, τ the viscous stress tensor and $\mathbf{S}$ a source term. The left hand side of the vector equation describes the rate of change of momentum per unit volume of the fluid particle. On the right hand side is the sum of forces acting on the particle. It is common practice to divide the forces to surface and body forces, the latter of which are included as a source term $\mathbf{S}$ in Equation 6. Some common examples of body forces are gravity and centrifugal force.

The first term on right hand side is the surface force due to spatial changes in pressure. Pressure can be considered simply as a normal stress. The second term describes the effect of viscous stresses which include both shear and normal stresses. The viscous stresses must be modelled somehow and a common simplification is to assume the fluid to be Newtonian, meaning that the viscous stresses are linearly proportional to the rates of deformation. For a Newtonian fluid the viscous stress tensor is given by

$$\tau_{ij} = \mu \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) + \delta_{ij} \lambda (\nabla \cdot \vec{U}). \tag{7}$$

In this notation the subscripts i and j are used to indicate that the stress component τ_{ij} acts in the j-direction on a surface normal to the i-direction. The first coefficient called dynamic viscosity, μ, relates stresses to linear deformations and the second coefficient, λ, named bulk viscosity takes into account volumetric deformation.

Energy equation

The energy equation can be written for several different quantities such as temperature or specific enthalpy. The specific energy equation is given by

$$\rho \frac{DE}{Dt} = -\nabla \cdot (p\vec{U}) + \frac{\partial \tau_{ij} U_i}{\partial x_j} + \nabla \cdot (\kappa \nabla T) + \mathbf{S}, \tag{8}$$

where the source term $\mathbf{S}$ includes heat sources and the work done by body forces. The left hand side is the rate of change of energy per unit volume. The first term on the right hand side describes the work done on the fluid particle by pressure and the second one the work done by viscous stresses. The third term is the rate of heat transfer by diffusion.

Equations of state

Among the unknowns of the equations describing fluid flow are four thermodynamic variables: pressure p, density ρ, specific internal energy e or specific enthalpy h and temperature T. If thermodynamic equilibrium is assumed, these four can be related to each other. The assumption is usually justified, as a fluid can thermodynamically adjust itself to new conditions so quickly that the changes can be considered as instantaneous.

A state of a fluid in thermodynamic equilibrium can be described using just two state variables. Equations of state relate the other variables to the state variables. If p and T are used as state variables, equations of state are written for ρ and e or h:

$$p = p(\rho, T) \quad \text{and} \quad e = e(\rho, T) \text{ or } h = h(\rho, T). \tag{9}$$

In compressible flows the equations of state provide linkage between energy, continuity and Navier-Stokes equations, as pressure and temperature variations cause density variations. In incompressible flows density variations on the other hand are not linked to pressure and there is no equation of state for pressure. Pressure equation can be derived by combining the continuity equation with the Navier-Stokes equations.

Initial and boundary conditions

Whether the equations concerning fluid flow are solved analytically or numerically, proper initial and boundary conditions must be set. First of all, for unsteady flow the spatial distribution of each variable must be known at an initial time $t = t_0$. When using numerical methods the initial conditions must also be set for steady flows in order to start the solution process. Thereafter, at each point of time at each boundary of the computational domain, some information about the variables must be known. Some examples of the most common types of boundaries are a solid, impermeable wall, an inlet and an outlet which are now briefly discussed.

To start with, in most cases there is no slippage on a solid wall and the velocity of the fluid at the wall equals the velocity of the wall. Several different boundary conditions are used for the temperature at the wall. For example a zero gradient boundary condition can be used if there is no heat conduction at the wall. Secondly, at each outlet or inlet either the velocity or pressure has to be set. Typically transported quantities such as temperature are specified at the inlet. By solving the equations concerning fluid flow, the values of other variables at inlet and outlet are discovered.

2.1.2 Finite volume method

In order to solve numerically the partial differential equations concerning fluid flow, the equations must be converted into a set of algebraic equations. The most widely used method is the finite volume method (FVM), in which the computational domain is divided into cells of finite volume and the values of different variables are usually stored at the cell centres. The set of algebraic equations is formed by demanding that in each cell the differential equations must be satisfied in integral form. Integration introduces surface terms for which the values of certain variables at the cell surfaces must be calculated. This is done by interpolation using the cell center values.

As an introduction to finite volume method let's consider a general conservation equation for a quantity ϕ

$$\nabla \cdot (\rho \phi \vec{U}) = \nabla \cdot (\Gamma \nabla \phi) + q, \tag{10}$$

where Γ is the diffusion coefficient and q a source term for quantity ϕ. By integrating this equation over a control volume CV and by using the divergence theorem the following integral form of the conservation equation is derived

$$\int_S \rho \phi \vec{U} \cdot \vec{n} \mathrm{d}S = \int_S \Gamma \nabla \phi \cdot \vec{n} dS + \int_\Omega q \mathrm{d}\Omega. \tag{11}$$

The next step is to approximate the surface and volume integrals using only the variable values at CV centres. The simplest approximation for the volume integral is to assume that the mean value of the integrand is the same as the integrand value at cell center and replace the integral with the product of this value and the CV volume

$$\int_\Omega q \mathrm{d}\Omega \approx q_p \Delta\Omega, \tag{12}$$

where q_p is the value of q at the CV center.

Surface integrals can be expressed as a sum of surface integrals over the CV faces:

$$\int_S f \mathrm{d}S = \sum_k \int_{S_k} \mathrm{d}S = \sum_k F_k, \tag{13}$$

where f is the convection $\rho \phi \vec{U} \cdot \vec{n}$ or the diffusion term $\Gamma \nabla \phi \cdot \vec{n}$. Each of the integrals F_k must be approximated. Easiest solution is to use the midpoint rule in which the

integral is approximated as a product of the integrand value at the cell-face center f_k and the cell-face area S_k:

$$F_k = \int_{S_k} f \, \mathrm{d}S \approx f_k S_k. \tag{14}$$

Value of f is not known on cell faces so it has to calculated by interpolation. One of the most straightforward approximations is linear interpolation between the two nearest CV cell centres. In one dimensional case CV-face center value can be calculated as

$$f_k \approx \lambda f_1 + (1-\lambda) f_2, \tag{15}$$

where f_1 and f_2 are the two nearest CV cell center values of f and the interpolation coefficient λ is defined as

$$\lambda = \frac{x_k - x_1}{x_2 - x_1}, \tag{16}$$

where x_k is the x-coordinate of the face center and the nearest cell center are located at x_1 and x_2. By assuming linear profile between the CV centres the simplest approximation of the gradient needed for the diffusion terms is also found

$$\left(\frac{\partial \phi}{\partial x}\right)_{x=x_k} \approx \frac{\phi_2 - \phi_1}{x_2 - x_1}. \tag{17}$$

In reality more advanced interpolation and differentiation methods are used to achieve better accuracy and to avoid certain problems like oscillatory solutions.

2.1.3 Turbulence modelling

Almost all flows encountered in engineering practice are turbulent and so is the one modelled in this thesis. Turbulent flows are characterized by unsteady behaviour. Velocity and other flow properties vary rapidly in time. Important quality of turbulence is the fact that it increases heat, mass and momentum exchange. Turbulent flows are also always three dimensional and they fluctuate on a broad range of length and time scales. The latter property makes direct simulation of turbulence a difficult task. Turbulent flows occur usually at high Reynolds numbers. Reynolds number is a dimensionless quantity that describes the ratio of inertial forces to viscous forces. In a turbulent flow the inertial forces dominate.

Several approaches exist to predict turbulent flows and few of the most important ones are introduced in the following. The most accurate approach is called Direct Numerical Simulation (DNS) in which the Navier-Stokes equations are solved for all of the motions in a turbulent flow without approximations other than discretization. The use of this method is limited by available computational power to relatively simple flows with low Reynolds number. This is due to the already mentioned fact that turbulence has a wide range of length and time scales resulting in a requirement for a large number of cells and very small time step in order to accurately capture all details of turbulence. In between direct numerical simulation and other

less accurate models is the Large Eddy Simulation (LES) in which the largest scale motions of the flow are solved directly and small scale motions are modelled.

For most applications is it unnecessary to resolve the finest details of turbulent flow and it is enough to know the time-averaged properties of the flow such as mean velocities and pressures. The most widely used approach to calculate turbulence flows is to use Reynold-averaged Navier-Stokes (RANS) equations which are a time averaged form of the regular Navier-Stokes equations. The averaging is done by using Reynolds decomposition in which the flow variables in Navier-Stokes equations are replaced by the sum of a mean and fluctuating component. For the x-component of the velocity the decomposition is

$$u = \overline{u} + u', \tag{18}$$

where $\bar{u}$ is the mean and u' the fluctuating component. The time-averaging results in six additional unknowns, the Reynold stresses

$$-\rho\overline{u'^2}, -\rho\overline{v'^2}, -\rho\overline{w'^2}, -\rho\overline{u'v'}, -\rho\overline{u'w'}, -\rho\overline{v'w'},$$

which must be modelled. The simplest approach is to use the so called Boussinesq hypothesis which states that the Reynold stresses are proportional to the mean rates of deformation

$$-\rho\overline{u_i'u_j'} = \mu_t\left(\frac{\partial\overline{u_i}}{\partial x_j} + \frac{\partial\overline{u_j}}{\partial x_i} - \frac{2}{3}\frac{\partial\overline{u_k}}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k\delta_{ij}, \tag{19}$$

where μ_t is the turbulent or eddy viscosity and $k = \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2})$ is the turbulent kinetic energy. These two must be calculated somehow to close the system of equations. In the most used standard $k-\epsilon$-turbulence model two additional transport equations are solved, one for the turbulent kinetic energy k and one for the rate of dissipation of turbulent kinetic energy ϵ. The eddy viscosity is derived using the solutions of these equations.

k-ω SST turbulence model

In this thesis Menter SST k-ω turbulence model is used. This model blends the $k-\epsilon$ model to $k-\omega$ model which is another two equation turbulence model with turbulence frequency $\omega = \epsilon/k$ being the second variable. In the Menter SST k-ω model the $k-\omega$ model is used in the near-wall region and the $k-\epsilon$ model in the fully turbulent region far from the wall. This is done to avoid unsatisfactory near wall performance of the standard $k-\epsilon$-model and the well-known $k-\omega$ problem with over-sensitivity to inlet free-stream turbulence properties.

The form of k-ω SST implemented in OpenFOAM is the one described in Reference [8] with updated coefficients from Reference [9] with an additional parameter for rough walls from Reference [10]. Only a very short description is given here so the

reader is advised to check the references for more details. First of all, the transport equations solved for k and ω are

$$\frac{\partial(\rho k)}{\partial t} + \nabla\cdot(\rho\vec{U}k) - \nabla\cdot(D_k\nabla k) = P_k - \frac{2}{3}\rho\nabla\cdot\vec{U}k - \rho\beta^*\omega k \tag{20}$$

$$\frac{\partial(\rho\omega)}{\partial t} + \nabla\cdot(\rho\vec{U}\omega) - \nabla\cdot(D_\omega\nabla\omega) = P_\omega - \frac{2}{3}\rho\Gamma\nabla\cdot(\vec{U})\omega - \rho\beta\omega^2 - \rho(F_1-1)CD_{k\omega} \tag{21}$$

In the equations D_k and D_ω are effective diffusivities for k and ω. P_k and P_ω are production terms in the respective equations, and $CD_{k\omega}$ is a limiter function. β^* is simply a constant model parameter. F_1 is a blending function which is used to achieve smooth transition between the two models. β and Γ are parameters which are calculated as a weighted averages of $k-\omega$ and $k-\epsilon$ parameters using F_1 as a weight function. Finally the turbulent viscosity is calculated from

$$\mu_t = \frac{a_1\rho k}{\max(a_1\omega, b_1F_{23}\sqrt{2}S)}, \tag{22}$$

where a_1 and b_1 are constant model parameters, F_{23} is a bounding function and S is the magnitude of the rate-of-strain tensor.

Near-wall treatment

Several layers with varying importance of viscous and turbulent effects can be identified in turbulent flow near a solid wall. Closest to the wall is the viscous sublayer where the behaviour of the fluid is dominated by viscous shear. At high Reynolds numbers, this layer is so thin, that resolving it requires a large amount of computational cells. This requirement can be avoided by approximating the near wall behaviour using wall functions. The simplest wall functions rely on the existence of a logaritmic sublayer where the velocity profile is given by the log-law

$$u^+ = \frac{\overline{u_t}}{u_\tau} = \frac{1}{\kappa}\ln y^+ + B, \tag{23}$$

where $\overline{u_t}$ is the mean velocity parallel to the wall, κ the von Karman constant ($\kappa \approx 0.41$) and B an empirical constant related to thickness of the viscous sublayer ($B \approx 5.5$ in a boundary layer over a smooth flat plane). u_τ is called the shear velocity and it is given by

$$u_\tau = \sqrt{\frac{|\tau_w|}{\rho}} \tag{24}$$

where τ_w is the shear stress at the wall and ρ density. y^+ is the dimensionless distance from the wall defined as

$$y^+ = \frac{\rho u_\tau y}{\mu}, \tag{25}$$

where μ is the dynamic viscosity. The log-law is valid in the region $30 < y^+ < 500$ and the first cell centres should be in this region when using wall functions based on the log-law.

2.1.4 Conjugated heat transfer (CHT)

Conjugated hear tranfer (CHT) refers to problems with two or more thermally connected subdomains in which heat transfer governed by differential equations is solved. The differential equations in different subdomains are connected by conjugate conditions on the subdomain interfaces. The usual conjugate condition is to demand continuity of temperature and heat flux on the interface. In solid type subdomains the only equation to solve is the heat equation but in fluid type subdomains the Navier-Stokes equations along with the energy equation must be solved due to convection. [11]

Two principal numerical approaches exist for CHT problems. The first less common one is to solve simultaneously the differential equations of all subdomains. The second approach is to use iteration and solve the differential equations in each subdomain separately. The subdomains are connected only by boundary conditions. In the iteration loop each subdomain is solved in turn and the boundary conditions updated after each subdomain solution. The iteration is continued until the solution of each subdomain converges. In this thesis the conjugated heat transfer problem is solved by iteration.

2.1.5 Semi-Implicit Method for Pressure Linked Equations (SIMPLE)

Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm is a widely used iterative approach to solve the Navier-Stokes equations. It is essentially a guess-and-correct method for the calculation of pressure. A brief description of the algorithm follows and more details can be found in most CFD textbooks such as Reference [12].

The SIMPLE algorithm is started by guessing a pressure field p^*. This pressure field is modified during the iteration with pressure corrections p' and the updated pressure field p is obtained from

$$p = p^* + p' \tag{26}$$

Similar corrections are applied to velocities

$$u = u^* + u' \quad v = v^* + v' \quad w = w^* + w'. \tag{27}$$

The velocity corrections are calculated from the pressure corrections using a velocity correction formula which is obtained from discretized momentum equation by omission of certain terms. This omission results in the words semi-implicit in the name SIMPLE.

The first step in the SIMPLE iteration loop is to solve the momentum equations for new ”starred” velocities. Unless the pressure field is correct, the obtained velocity field will not satisfy the continuity equation. The aim in the SIMPLE algorithm is to correct the pressure field in each iteration in a way that the starred velocity field

gets closer to satisfying the continuity equation. In order to do so, the pressure corrections must be solved somehow. This is achieved by solving a pressure-correction equation which is obtained from the continuity equation with some manipulation. The second step in the SIMPLE algorithm is to solve this equation for pressure corrections and calculate new pressure field from equation 26. After this the velocity fields are updated using the velocity correction formulas. The final step in the iteration loop is to solve differential equations of other quantities (such as temperature and turbulence quantities) which influence the flow field. A new iteration is started by treating the corrected pressure p as new guessed pressure. The iteration process in continued until no pressure correction is needed and convergence is achieved.

2.2 Monte Carlo method in neutron transport calculations

The Monte Carlo method in neutronics is used to calculate statistical estimates for integral reaction rates and other derived parameters by tracking a large number of neutrons (in the order of $10^6 - 10^9$). In neutron tracking the movement of a single neutron through the problem geometry is simulated one interaction at a time. By collecting information about different events during the tracking process of each neutron, statistical estimates can be calculated for parameters such as the multiplication factor of the system. This section describes briefly the most important physics and concepts needed to understand the basics of Monte Carlo method in neutron transport. A more thorough overview of the topic can be found for example in Reference [13].

2.2.1 Neutron interactions

The physics of fission reactors is determined by the transport of neutrons and their interactions with matter. Neutrons are subatomic particles with no electric charge and they interact easily with nuclei but not generally with each other. This insignificance of neutron-neutron reactions results in the linearity of neutron transport theory. The three most important neutron interactions in a nuclear reactor are neutron induced fission, radiative capture and elastic scattering. The following subsections briefly describes each of them.

Neutron induced fission

In neutron induced fission a neutron is absorbed into a heavy nucleus to form a compound nucleus which splits into two or more nuclei releasing energy and typically several neutrons. The total released energy is in the order of 200 MeV and most of it is in the form of the kinetic energy of the fission fragments.

Radiative capture

When a compound nucleus is formed in the absorption of a neutron, fission is only one of the possible reactions. The most important reaction competing with neutron induced fission is radiative capture in which the incident neutron is bound in the target nucleus and the resulting excited state decays by photon emission. At certain energies of the incident neutron referred as resonance energies the probability of radiative capture is greatly enhanced. Each of these energies correspond to a difference between the current energy and a higher energy level of the target nucleus. The resonances are of great importance for the thermal feedback in the fuel and will be further discussed in Section 2.3.

Scattering

Scattering reactions can be divided into elastic and inelastic scattering. In elastic scattering the kinetic energy of the reacting particles is conserved. The simplest form of elastic scattering is referred as potential scattering in which the neutron does not penetrate the nucleus. Another possibility is a formation of compound nucleus followed by an emission of a neutron that returns the compound nucleus to the ground state of the original nucleus. In thermal reactors fast fission neutrons are slowed down mainly by elastic scattering in the moderator.

2.2.2 Neutron chain reaction

A fission reaction releases a number of new neutrons, which in turn can initiate fission and release new neutrons, if they do not get absorbed or leak out of the system first. This gives rise to a chain reaction, where generations of neutrons follow each other. The most important parameter characterizing this chain reaction is the multiplication factor k defined as:

$$k = \frac{\text{Number of neutrons in generation } n+1}{\text{Number of neutrons in generation } n}. \tag{28}$$

The system is referred as critical, if k equals unity. The size of the neutron population is constant from generation to generation and the chain reaction is stable. A multiplication factor less than unity corresponds to a decreasing neutron population and the system is sub-critical. The last possible state of the system is supercriticality in which the multiplication is greater than unity and the chain reaction is growing. Other important parameters related to the neutron chain reaction and especially to the kinetics of the reactor are neutron lifetime l, mean generation time Λ and delayed neutron fraction β. Neutron lifetime is the average time elapsing between the birth of a neutron and its loss from the system by absorption or leakage. Mean generation time is the average time from a neutron emission to a fission. As the name suggests, delayed neutron fraction is the fraction of fission neutrons which are delayed.

2.2.3 Overview of a neutron tracking routine

In the core of the Monte Carlo method is the tracking of neutrons as they move through the the problem geometry. The following paragraphs give a general step-by-step overview of the neutron tracking routine.

Sampling a new neutron

A new neutron is sampled by determining its initial location $\vec{r}$, energy E and direction of movement $\vec{\Omega}$. These are determined by user-given external neutron source or in reactor physics calculations by fission source (see Section 2.2.4).

Sampling the distance to the next interaction

After the birth of the neutron the next step is to sample the distance the neutron travels in straight line before interaction. The probability of interaction for a neutron in a medium with macroscopic total cross section Σ_t while moving distance ds is

$$\mathrm{d}P = \Sigma_t \mathrm{d}s. \tag{29}$$

A neutron can only change its energy in interactions so the energy-dependence of Σ_t can be omitted. Using the previous equation the probability $p(s)$ that a neutron travels a distance s without interaction and then interacts within the interval ds can be derived as follows:

$$\begin{aligned} \mathrm{d}p &= (1-p(s))\Sigma_t \mathrm{d}s \\ \int_0^p \frac{1}{1-p(s)} \mathrm{d}p &= \int_0^s \Sigma_t \mathrm{d}s \\ p(s) &= 1 - \exp(-\Sigma_t s). \end{aligned} \tag{30}$$

The distance following this distribution can be sampled by generating a pseudo-random number and using inverse transform sampling. As the macroscopic cross section is assumed to be constant, the neutron has to be stopped on each boundary in which the macroscopic cross section changes. In this case the sampling process is referred to as surface tracking. In a more advanced method, referred to as Delta-tracking this limitation is removed and the neutron can cross for example material surfaces without stopping at the boundary.

Sampling the interaction type

Next step in the tracking routine is to determine what type of a reaction occurs and with what type of a nuclide. The probability for a reaction of type x with a nuclide type i is simply

$$p_{i,x} = \frac{\Sigma_{i,x}}{\Sigma_t}, \tag{31}$$

where $\Sigma_{i,x}$ is the macroscopic cross-section of reaction x for nuclide i. Using these discrete probabilities the interaction type can be easily sampled.

Modelling the interaction

Depending on the sampled reaction the neutron track is either terminated or the energy and direction of movement of the neutron altered. In a criticality source method the former occurs in the case of capture, fission etc. and the latter in the case of scattering reaction etc. If the neutron track was not terminated the tracking process continues from step 2. For each neutron the tracking is continued until the neutron track is terminated.

2.2.4 Methods for simulating the neutron chain reaction

Calculation methods in Monte Carlo neutronics can be divided into external and criticality source methods. In external source method all neutron histories are started from a user-defined source distribution. Each new neutron produced in fission is tracked independently and the neutrons are tracked until they are captured or leak out of the system.

The alternative method, used also in this thesis, is the criticality source method in which the simulation proceeds in cycles. The neutron source distribution on each cycle is determined by the fission reaction distribution of the previous cycle. Neutron histories are terminated by fission and all histories in a cycle are completed before beginning the next one. The number of new source neutrons generated by fission is generally not equal to the number of neutron histories started at the beginning of the cycle. The multiplication factor in cycle n is defined as:

$$k_n = \frac{\text{number of source neutrons in cycle } n+1}{\text{number of simulated histories in cycle } n}. \tag{32}$$

The value usually differs from unity which implies a constantly changing source size. This causes problems in the simulation and the solution is to artificially add or remove neutrons from the population to maintain constant source size at each cycle.

The criticality source method requires an initial guess for the source distribution to start the simulation. This initial guess may differ drastically from the saturated source distribution. Therefore a number of cycles are run at the beginning of the simulation before starting to collect results in order to achieve fission source convergence.

Several criticality source methods exist. In this thesis all Monte Carlo simulations are run with the k-eigenvalue method which simulates a stationary self-sustaining chain reaction. As the name suggests, this method is equivalent to solving the eigenvalue form of the neutron transport equation. At the beginning of each cycle

the number of source neutrons is adjusted in order to maintain constant source size. One should note that, when the system is far from criticality, this modification leads to over- or under-estimation of the fission source importance. In those cases other methods such as the α-eigenvalue method produce more reliable results.

2.3 Motivation for coupled calculations

A large amount of fission heat is produced in a nuclear reactor during operation. A change in the neutron flux distribution results in changes in fission heating which produces changes in temperature, which in turn will produce changes in neutron flux distribution. The modelling of this feedback mechanism is vital in reactor dynamics calculations. Inclusion of the feedback mechanism means that the problem is no longer linear (as was the case with only neutron transport). A brief introduction to the most important feedback effects in thermal reactors follows.

As the neutron population increases locally, so does the fission heating. The heat is deposited to the fuel element, which immediately increases the fuel temperature. This results in the broadening of the effective resonance absorption (and fission) cross section, which in turn generally increases neutron absorption and causes a negative reactivity addition. The broadening of effective resonance cross sections due to increased thermal motion of the nuclei is referred to as Doppler broadening. The fuel also expands as the temperature rises changing the fuel moderator geometry. In addition, the increase in the fuel volume decreases fuel density resulting in a reduction of resonance absorption and an increase in reactivity.

The increased fission heating also increases moderator and coolant temperatures with small delay as the heat is transported from the fuel to the moderator and coolant. For example, in a pressurized water reactor water acts as both coolant and moderator. An increase in the moderator temperature causes a decrease in the moderator density which reduces moderating effectiveness and hardens the neutron spectrum. This results in a negative reactivity addition in a thermal reactor as the fission cross section is much higher for thermal than fast neutrons. One specific change due to higher moderator temperature in the neutron spectrum is the fact that the thermal neutrons have on average a higher energy. The limiting energy for up-scattering is higher in moderator with higher temperature. The reduced moderator density also increases leakage out of the reactor core as the diffusion of neutrons increases.

3 Models and methods

3.1 OpenFOAM

3.1.1 Overview

OpenFOAM (Open Field Operation And Manipulation) [14] is a free, open source C++ toolbox for continuum mechanics problems, including CFD distributed by the OpenFOAM Foundation. It can be used to solve extensive range of problems from complex fluid flows involving chemical reactions, turbulence and heat transfer, to electrodynamics and solid dynamics. OpenFOAM version 2.3.1 was used in this work.

The base of OpenFOAM is a large library which offers the core functionality of the code such as tensor and field operations, discretization of partial differential equations, mesh, solution to linear equations, turbulence models etc. By using this library the user can develop applications such as solvers for partial differential equations or extensions such as new boundary conditions. OpenFOAM also includes over 80 ready made solvers, each for a specific problem, and tools for meshing and pre- and post-processing. Almost everything in OpenFOAM runs in parallel.

3.1.2 Usage

OpenFOAM is designed for Linux operating systems and doesn't feature a graphical user interface. Therefore, each included application is designed to be executed from a terminal command line. To run an application an OpenFOAM case has to be created. A case is simply a directory containing the files required to run the particular application.

The basic structure of a case is shown in Figure 1. Each of the folders have a specific purpose. *constant* directory contains the mesh in *polyMesh* subdirectory and configuration files specifying physical properties e.g. *thermophysicalProperties*. *system* directory has all the information regarding the solution process itself and contains at least the following files: *controlDict* specifies the run control parameters including solver, start/end time and time step. *fvSchemes* is used to select discretization schemes. The linear equation solvers, tolerances and solution algorithms are controlled by *fvSolution*. The time directories contain individual data file for each field such as pressure p or velocity $\vec{U}$. The data can be either user specified initial or boundary conditions or results written by the application. The name of each folder indicates the simulated time at which the data is written. Consequently, initial conditions are usually defined in *0* time directory.

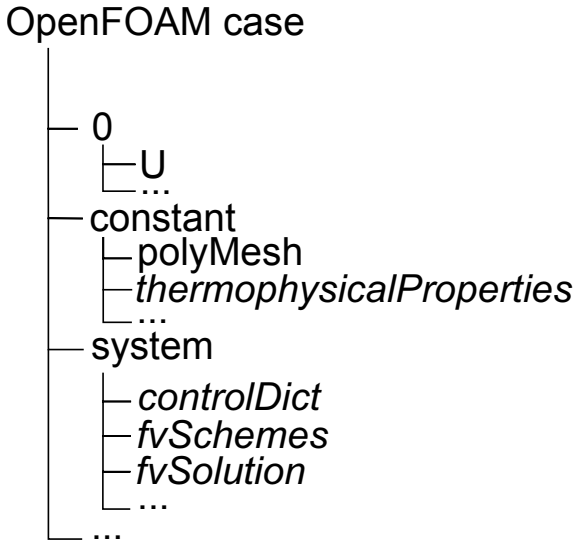


Figure 1: Basic OpenFOAM case structure.

3.1.3 chtMultiRegionSimpleFoam-Solver

The chtMultiRegionSimpleFoam is a steady-state solver for CHT problems with buoyant, turbulent flows of compressible fluids including radiation. The "Multi-Region" in the name of the solver refers to the fact that each subdomain or region has its own mesh. The Navier-Stokes equations are solved with SIMPLE algorithm as suggested by the name of the solver. A brief description of the solver loop follows:

```
Read fluid fields;
Read solid fields;
while t < tMax do
    for all fluid regions do
        Solve the momentum equation;
        Solve the energy equation and update thermophysical properties;
        Solve the pressure equation and correct velocities;
        Solve the turbulence equations;
    end for
    for all solid regions do
        Solve the energy equation and update thermophysical properties;
    end for
    t++;
end while
```

First the initial fields are read for all regions. Then at each time step the differential equations are first solved for all fluid regions and then for all solid regions. For each fluid region the solution process starts by gathering previous fields and solving the momentum equation for velocities. Then the energy equation written

for enthalpy is solved and thermophysical properties updated according to the new temperature field and pressure field of the previous time step. Next step is to solve the pressure equation and correct velocities to satisfy the continuity equation using the new pressure field. Final step for a fluid region is to solve the turbulence equations and update the turbulence viscosity. For a solid region only energy equation for enthalpy is solved.

The different regions are connected by boundary conditions. Each time an equation is solved the corresponding boundary conditions of the region are updated. This means that the region interfaces are constantly updated and adjacent regions are accounted for properly.

3.2 Serpent

3.2.1 Overview

Serpent is a state-of-the-art Monte Carlo reactor physics code developed at VTT Technical Research Centre of Finland since 2004. The code was originally written for spatial homogenization but has gained many additional features over the years of development. These include built-in burnup calculation capability and a multi-physics interface which allows coupling to external CFD, thermal hydraulics and fuel performance codes [15]. The latter feature is discussed in more detail later because of its importance to this thesis. Serpent uses several special methods for efficient simulation of the neutrons. These include Woodcock delta tracking for neutron transport which simplifies the geometry routine and reduces calculation times, and on-the-fly temperature treatment routine based on the Target Motion Sampling (TMS) technique which reduces significantly the required cross section data to store in RAM memory.

The newest completely rewritten version of Serpent, Serpent 2 is in a beta-testing phase and available to registered users by request. Currently some of the main development topics are advanced methods for spatial homogenization and coupled multi-physics applications.

3.2.2 Usage

Serpent doesn't feature a graphical user interface and is run from terminal on a Linux machine. The simulation problem is defined in one or several input files. Basic serpent input consists of three parts: geometry, materials and options. First of all, the geometry defines boundaries of the computational domain and regions of space filled with each material. Secondly, properties such as density and nuclide composition are given for each material. Finally several options are specified. These include for example the number of simulated neutrons, path of the cross section data library and whether to include detectors.

Results of the Serpent simulations are written into output files. Each simulation produces at least the main output file containing a variety of data from used run parameters to actual results such as critical eigenvalues or few group cross sections.

3.2.3 Target Motion Sampling (TMS) temperature treatment technique

In Monte Carlo calculations the effect of temperature on cross sections has been taken into account traditionally by prebroadening cross sections for each nuclide and temperature combination in the problem geometry. Each of these cross sections must be stored in RAM memory, size of which becomes a major limitation in burnup calculations and systems with detailed temperature fields. To overcome this problem, on-the-fly temperature treatment techniques have been developed. In these methods the effect of thermal motion on the cross sections is taken into account during the Monte Carlo transport cycle and therefore the effective cross sections do not need to be stored in memory or at least not for all of the temperatures in the problem geometry.

Serpent uses on-the-fly temperature treatment referred as target motion sampling (TMS) technique [16]. In this method target velocities are sampled separately at each collision site and collisions are dealt with in the target-at-rest frame using precalculated cross sections below or equal to the actual target temperature. Sampling the velocity at each site results in distributed cross sections which is accounted for in the rejection sampling of the neutron paths. Without the TMS method the Monte Carlo simulations with high resolution temperature distributions performed in this thesis would not have been possible.

3.2.4 Multi-physics interface

The multi-physics interface is used for coupling Serpent with external solvers. The interface allows modelling of materials with arbitrarily refined temperature and density distributions supplied by the external solver. The coupling is implemented by separating the state point information (temperature and density) from the Monte Carlo geometry routine. This is made possible by using rejection sampling for neutron path lengths and on-the-fly TMS temperature treatment for microscopic cross sections.

The multi-physics interface has several different formats to pass the state point information from the external solver to Serpent. These include piece-wise constant distribution on a regular mesh, special interface for fuel performance codes and unstructured mesh based interface for CFD codes [17]. The last one is utilized in this thesis and it's currently based on the OpenFOAM mesh format. In this interface the density and temperature distributions are comprised of separate values in each cell of the mesh. The interface features a separate adaptive search mesh to speed up

the cell search for finding correct state point information at each interaction point. The unstructured mesh based interface allows the passing of density and temperature distributions directly without modification from the CFD code into Serpent but in addition to pass a power distribution back from Serpent to the CFD code. It is also possible to define the problem geometry based on the OpenFOAM mesh which enables modelling of complex irregular geometries [18].

The following describes the definition and the functionality of the adaptive search mesh in more detail: The effect of the search mesh size on memory usage and calculation times is investigated in Section 5.2. As stated above, at each interaction point Serpent must find the corresponding cell of the OpenFOAM mesh in order to determine the temperature and density at the interaction point. The most simple solution to this problem is to loop over all OpenFOAM cells to find the cell which includes the interaction point. As the OpenFOAM mesh can have millions of cells, this is not a practical approach. The search mesh is used to speed up the cell search process by limiting the number of possible OpenFOAM cells based on the coordinates of the interaction point.

The search mesh is a cartesian mesh placed on top of the OpenFOAM mesh. Each search mesh cell includes the OpenFOAM cells inside it. Since the search mesh is cartesian the search mesh cell corresponding to the interaction point can be determined from the interaction coordinates with simple arithmetic operations which is a speedy process. To determine the correct OpenFOAM cell only the OpenFOAM cells, whose bounding boxes fall into the search mesh cell, have to be checked.

Memory consumption of the search mesh is reduced by adaptivity. The search mesh is gradually refined in the regions where the density of OpenFOAM cells is large. The adaptive search mesh is defined with card:

```
<msh_split> <msh_dim> <s0> <sz1> ... <sz_dim>
```

`<msh_split>` is the limiting number of cells, after which the search mesh cell is split into a new mesh. `<msh_dim>` defines the maximum number of times a search mesh cell can be split. This value is followed by an equal number of parameters giving the size of the search mesh at each level. For example

```
100 2 5 3
```

card defines a search mesh consisting of two levels. The first level is a 5x5x5 mesh. If the number of OpenFOAM cells in any of these cells exceeds 100, the corresponding cell is further divided into a 3x3x3 mesh.

3.3 Coupling

3.3.1 Coupling program

A simple coupling program was used in order to automatically run Serpent and OpenFOAM in turns. This program, written in C-language, was modified from SCSS Serpent coupling program provided by Ville Valtavirta. The structure of the coupling program resembles that of Serpent 2 and some of the functions are copies from Serpent 2 source code with slight modification. The program is run from terminal with command

```
scss -omp n input
```

n is the number of threads to use in Serpent OpenMP parallel calculations and input is the name of the input file which specifies Serpent executable, Serpent input file, OpenFoam solver, number of processors to use in OpenFOAM MPI parallel calculations and the maximum number of time steps calculated in OpenFOAM on each iteration.

Before beginning the actual iteration cycle the coupling program does some initialization, parses the command line and reads the input file. In the beginning of the iteration cycle Serpent is started as a background process and Serpent calculates the first heat source. After Serpent is done with the calculation, it signals the coupling program with POSIX-signal to continue. Next step is to perform the OpenFOAM calculations to solve temperature and density distributions using the new heat source. Contrary to Serpent, the OpenFOAM solver is run separately on each solution cycle. Once the OpenFOAM calculations are done, the Coupling program signals Serpent to continue the Monte Carlo simulation with updated density and temperature distributions. The iteration cycle is continued until either maximum number of iterations, defined in Serpent input file or convergence based on checks performed in the OpenFOAM solver is achieved.

3.3.2 Convergence of the CFD solution

At each iteration the convergence of the individual OpenFOAM calculation is determined by monitoring energy balance in the fluid. The balance is calculated by integrating the convective and diffusive fluxes of enthalpy over the fluid region boundaries

$$B = \int_S \rho h \vec{U} \cdot \vec{n} \mathrm{d}S + \int_S \alpha \nabla h \cdot \vec{n} \mathrm{d}S. \tag{33}$$

In a steady flow the balance should be equal to zero as the enthalpy (or temperature) field is constant in time and there are no heat sources in the fluid region. The actual variable monitored by the OpenFOAM solver describes relative energy balance and it is calculated by dividing the absolute value of the energy balance with enthalpy

flux from the regular fuel rods

$$B_{rel} = \frac{|B|}{\int_{S_f} \alpha \nabla h \cdot \vec{n} \mathrm{d}S}. \tag{34}$$

The iteration is stopped when the relative energy balance has been lower than an user specified limit for an user specified number of time steps in a row. The latter criterion is used because during the iteration the energy balance can temporarily go close to zero even though the energy balance has not yet reached its final level.

The energy balance criteria was easy to implement and that's the first reason why it was used for automatically stopping the individual OpenFOAM iterations. However, an energy balance close to zero in the fluid region is not enough to ensure that the solution has converged. Numerous other checks must be performed manually. First of all, energy balance should be satisfied in all other regions as well. Secondly, the residuals of solved equations should be monitored. Thirdly, in the fluid region overall mass balance and continuity equation should be satisfied.

The second reason to use the energy balance based convergence checking was the fact that it performed well in the coupled calculations of this thesis. The other criteria described above were adequately satisfied when the energy balance criteria in the fluid region was met.

3.3.3 New boundary condition

OpenFOAM has a *turbulentTemperatureCoupledBaffleMixed* boundary condition to couple solid regions or a solid and a fluid region thermally. In this boundary condition both sides of the boundary agree on temperature which is set to a value that balances the heat flux on the boundary

$$\kappa_1 \nabla_1 T = \kappa_2 \nabla_2 T. \tag{35}$$

By using linear approximation for the gradient, the heat flux on the boundary can be evaluated as

$$q \approx \kappa(T_b) \frac{T_b - T}{\Delta}, \tag{36}$$

where $\kappa(T_b)$ is the heat conductivity and T_b the temperature on the boundary. T is the first cell temperature and Δ the normal distance from the boundary to the first cell centre. With this approximation Equation 35 can be solved for the temperature on the boundary:

$$T_b = \frac{\frac{\kappa_1}{\Delta_1} T_1 + \frac{\kappa_2}{\Delta_2} T_2}{\frac{\kappa_1}{\Delta_1} + \frac{\kappa_2}{\Delta_2}}, \tag{37}$$

where κ_1, Δ_1 and T_1 are the effective thermal conductivity on the boundary, the normal distance from the boundary to the first cell centre and the temperature of the first cell in the current region. The corresponding variables for the neighbouring

region are κ_2, Δ_2 and T_2. The effective conductivity is obtained either from the thermodynamic and turbulence models or from a separate κ-field.

Heat flux on the boundary can be calculated using enthalpy as

$$q = \frac{\kappa}{c_p}\nabla h = \alpha \nabla h, \tag{38}$$

where c_p is the specific heat capacity, h specific enthalpy and α effective thermal diffusivity as it is defined in OpenFOAM. The heat equation in solid regions and the energy equation in fluid regions are solved for enthalpy and this term appears in these equations so the heat flux on the boundary evaluated using this form should be in balance. By assuming linear profile between the boundary and the first cell centre Equation 38 can be approximated by

$$q \approx \frac{\kappa(T_b)}{c_p(T_b)}\frac{h_b - h}{\Delta}, \tag{39}$$

where the specific heat capacity and the effective thermal conductivity are evaluated at the boundary. As the enthalpy is calculated in the code by integrating the specific heat capacity, the heat flux evaluated with this equation differs from the one evaluated with Equation 36 unless the specific heat capacity is constant or Δ approaches zero. This fact was noticed during initial testing of the coupled problem using the *turbulentTemperatureCoupledBaffleMixed* boundary condition when the heat fluxes calculated from enthalpy didn't agree on the different sides of the boundary when temperature dependent specific heat capacities were used. The original boundary condition had to be modified in order to use non-constant specific heat capacities.

The new boundary condition simply balances the heat fluxes calculated from enthalpy

$$\alpha_1 \nabla_1 h = \alpha_2 \nabla_2 h, \tag{40}$$

which can be approximated by

$$\frac{\kappa_1(T_1)}{c_{p,1}(T_1)\Delta_1}[h_{b,1}(T_1) - h_1] = \frac{\kappa_2(T_2)}{c_{p,2}(T_2)\Delta_2}[h_{b,2}(T_2) - h_2], \tag{41}$$

where T_1, κ_1, $c_{p,1}$ and $h_{b,1}$ are the temperature, heat conductivity, specific heat capacity and specific enthalpy on the boundary of the current region. The corresponding variables for the neighbouring region are T_2, κ_2, $c_{p,2}$ and $h_{b,2}$. Δ_1 is the normal distance from the boundary to the first cell centre and h_1 the first cell value of enthalpy for the current region. The corresponding values for the neighbouring region are Δ_2 and h_2. The boundary values of temperature, T_1 and T_2, agree if no thermal resistance exist. Otherwise the neighbouring region temperature is solved from

$$T_2 - T_1 = qR = \frac{\kappa_1(T_1)}{c_{p,1}(T_1)\Delta_1}[h_{b,1}(T_1) - h_1]\frac{\Delta_R}{\kappa_R}, \tag{42}$$

where R is the thermal resistance, Δ_R effective thickness and κ_R effective thermal conductivity of the artificial layer inducing the thermal resistance.

In the new boundary condition the equation 41 is solved for the temperature on the boundary of the current region T_1 with Newton's method. The iteration is started from the current value on the boundary. At each step of the iteration the required values for c_p and h are retrieved from thermodynamic and turbulence models. Value of κ is assumed constant during the iteration with Newton's method due to difficulties in retrieving it's value for a given temperature. If convergence have been achieved, equation 41 is fulfilled by the initial values and no iteration is required so this simplification doesn't effect the final results.

3.3.4 Thermophysical models

Thermophysical properties in OpenFOAM are calculated with thermophysical models each of which is constructed as a pressure-temperature system from which other properties are computed. A thermophysical model has several layers or submodels, each with a different purpose. For example one submodel defines the equation of state, another transport properties such as viscosity and thermal conductivity and yet another is concerned with the evaluation of basic thermophysical properties such as specific heat capacity and enthalphy. OpenFOAM has a collection of ready-made submodels which enable basic calculations. Three new submodels had to be created for the calculations in this thesis in order to use commonly used correlations for the thermophysical properties of the fuel and the cladding. The following describes how the thermopysical properties in different materials were modelled.

Fluid

Thermophysical properties of water were evaluated using libFluid which is a thermophysical library written by Joona Kurki (2014, VTT). libFluid was coupled with OpenFOAM with a custom thermophysical class. libFluid implements IAPWS-IF97 steam tables for the calculations but also dynamically tabulates the calculated values as needed and uses them to interpolate the requested properties. libFluid code was modified slightly in order to use constant properties when the temperature of water is higher than the saturation temperature for the given pressure. This extrapolation method results in properties which are not physically correct but it allows the heating of water above saturation temperature without modelling the boiling process.

Fuel

For the fuel two new submodels were created. The first one evaluates specific heat capacity and enthalpy which is calculated by integrating specific heat capacity. The second one is a transport model and it calculates thermal conductivity for a given temperature.

The specific heat of the fuel is given by FRAPTRAN correlation [19]

$$c_m = K_1 \frac{\Theta^2 \exp(\Theta/T)}{T^2[\exp(\Theta/T)-1]^2} + K_2 T + K_3 \frac{Y E_d}{2RT^2} \exp(-E_d/RT), \tag{43}$$

where T is the temperature in K, R is the ideal gas constant ($\approx$8.314 J/(mol·K)) and Y is the oxygen-to-metal ratio. For UO_2 fuel, the other constants are K_1 = 296.7 J/(kg·K), K_2 = 0.0243 J/(kg·K^2), K_3 = 8.745×10^7 J/kg, Θ = 535.285 K and E_d = 1.577×10^5 J/mol.

The thermal conductivity of the fuel is calculated from the Fraptran correlation [19]

$$\lambda = 1.789 \frac{d}{1+0.5(1-d)} \lambda_{95}, \tag{44}$$

where λ_{95} is the thermal conductivity for UO_2 at 95 % of the theoretical density, and d is the as-fabricated density of the fuel pellet as a fraction from the theoretical value. The value of λ_{95} is given by

$$\lambda_{95} = [A + a \cdot gad + BT + f(Bu) + (1 - 0.9\exp(-0.04Bu))g(Bu)h(T)]^{-1} + \frac{E}{T^2}\exp(-F/T), \tag{45}$$

where gad is the gadolinia weight factor, Bu the burnup in GWd/MTU and T the temperature in K. The constants are A = 0.0452 (m·K)/W, a = 1.1599, B = 2.46×10^{-4} m/W, E = 3.5×10^9 (W·K)/m) and F = 16361 K. The functions are defined as

$$f(Bu) = 0.00187Bu \tag{46}$$

$$g(Bu) = 0.038Bu^{0.28} \tag{47}$$

$$h(T) = [1 + 396\exp(-Q/T)]^{-1}, \quad Q = 6380 \text{ K}. \tag{48}$$

Cladding

For the cladding it was necessary to create only one new submodel which evaluates specific heat capacity and enthalpy. The thermal conductivity is evaluated with ready made *polynomialTransport* submodel which calculates the thermal conductivity from a polynomial with user specified constants.

The specific heat capacity of the cladding is calculated by linear interpolation from the values of Table 1. The data is adopted from Reference [19]. For temperatures lower than 300 K and higher than 1248 K the specific heat is assumed constant (Do not exist in the test case).

Table 1: The specific heat capacity of the cladding [19].

Temperature (K)	Specific heat (J/(kg·K))
300	281
400	302
640	331
1090	375
1093	502
1113	590
1133	615
1153	719
1173	816
1193	770
1213	619
1233	469
1248	356

The thermal conductivity of the Zircaloy cladding is calculated from FRAPTRAN correlation

$$\lambda = 7.51 + 2.09 \cdot 10^{-2} T - 1.45 \cdot 10^{-5} T^2 + 7.67 \cdot 10^{-9} T^3, \tag{49}$$

where the conductivity λ is given in W/(m·K) and the temperature in K [19].The correlation is valid for temperatures below 2098 K.

3.4 Test case

The Serpent-OpenFOAM coupling was tested with a case based on pressurized water reactor TMI-1 assembly presented in OECD benchmarks [20] and [21]. The test case models a mock-up 5x5 fuel assembly cooled with water in a steady state condition at full power. Reflective boundary conditions are used horizontally so the test case is horizontally infinite. Axially the problem is finite. The following sections describe the test case geometry, materials and other parameters, and also how the case was modelled with Serpent and OpenFOAM.

3.4.1 Geometry and materials

3D-render of the modelled 5x5 assembly is shown in figure 2 and the design parameters are presented in Table 2. The assembly contains both regular UO_2 fuel pins and gadolinia fuel pins with integral Gd burnable poison. The cladding in all of the fuel rods is Zircaloy-4 with the following composition presented in weight percent: O 0.125 %, Cr 0.100 %, Fe 0.210 %, Zr 98.115 % and Sn 1.450 %. Between the cladding and the fuel pellet there is a small gap which is modelled differently in OpenFOAM and Serpent.

Table 2: Fuel assembly design parameters for the test case

Parameter	Value
Fuel assembly dimensions	5x5
Fuel rod length (mm)	3600
Unit cell pitch (mm)	14.427
Fuel pellet diameter (mm)	9.391
Regular fuel material	UO_2
Regular fuel density (g/cm^3)	10.283
Regular fuel enrichment (w/o)	4.85
Poison fuel material	UO_2+Gd_2O_3
Poison fuel density (g/cm^3)	10.144
Poison fuel enrichment (w/o)	4.12
Gd_2O_3 concentration (wt. %)	2.0
Cladding material	Zircaloy-4
Cladding density (g/cm^3)	6.55
Cladding outside diameter (mm)	10.928
Cladding thickness (mm)	0.673

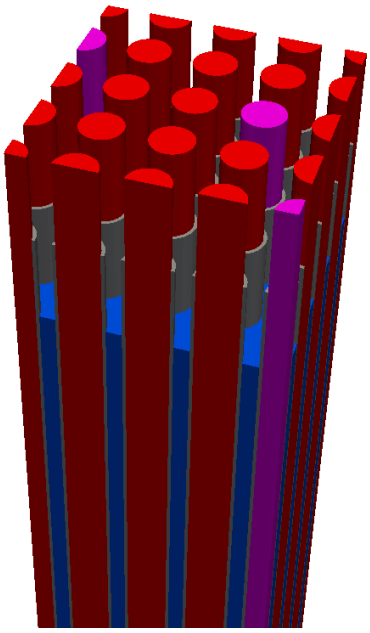


Figure 2: 3D-render of the test case geometry with the following colours: regular fuel in red, gadolinia fuel in purple, cladding in grey and coolant in blue.

3.4.2 OpenFOAM model

The CFD calculations were performed with a slightly modified version of the chtMultiRegionSimpleFoam solver (see Section 3.1.3). In the OpenFOAM model the test case geometry is divided into four regions, three of which are solid regions and one is a fluid region. The first region contains the water flowing past the fuel rods, second the cladding of all fuel rods, third the regular fuel and fourth the fuel with gadolinia. The fuel must be divided into two separate region because the two fuel types have different thermophysical properties. The gap between a fuel pellet and cladding is not included in the geometry and it is replaced with cladding. The effect of the gap is taken into account by introducing a thermal resistance between the fuel and the cladding with a boundary condition. Each region has a separate mesh and the meshes were generated with *snappyHexMesh* application included in the OpenFOAM distribution based on the design parameters presented in Table 2. The mesh generation process with *snappyHexMesh* is highly automated and fast, but requires some fiddling around with the parameters determining how the mesh is actually generated. The total number of cells is 2228224 with the following distribution: fluid 1175040, cladding 589824, regular fuel 412672 and gadolinia fuel 50688. The OpenFOAM solver was run in parallel with MPI. The mesh and the fields in each region were decomposed using method referred as *simple* in which each region was divided into pieces in the axial direction.

The fluid flow was modelled as compressible and the thermophysical properties of water were evaluated with libFluid. Turbulence was modelled with k-ω SST turbulence model and near wall treatment was implemented with wall functions. For the velocity Spalding continuous law-of-the-wall wall function [22] was used and for heat tranfer the Jaytilleke [23] correction were applied. In the k-ω SST model blended near-wall ω formulation by Menter [8] was applied.

Boundary conditions for the fluid region and the solid regions are presented in tables 3 and 4. $symmetry$ boundary condition was used horizontally at the boundaries of the computational domain to model a horizontally infinite problem. At the inlet of the fluid region *mapped* boundary condition was used for the velocity to create a periodic flow. The velocity profile was sampled at a distance of 8 cm from the inlet and scaled to an average value of 3.6 m/s. Temperature at the inlet was fixed to a value of 561 K. Pressure at the outlet was set to a fixed average value of 15.51 MPa. The modified version of *turbulentTemperatureCoupledBaffleMixed* boundary condition presented in Section 3.3.3 was used for the temperature on the fluid-cladding interface.

The densities of the both fuel types and the cladding were constant in the calculations, and they are presented in Table 2. Thermophysical properties of the both fuel types and the cladding were evaluated using the submodels presented in Section 3.3.4. The regular fuel, gadolinia fuel and cladding regions were connected to each other with the modified version of *turbulentTemperatureCoupledBaffleMixed* boundary condition which was also used to add thermal resistance to the fuel-cladding

interface. The values specifying the thermal resistance were $\kappa = 0.2731\,\mathrm{W/(m \cdot K)}$ and $\Delta = 3.55 \times 10^{-5}\,\mathrm{m}$.

At each iteration the convergence of the individual OpenFOAM calculation was determined by monitoring energy balance in the fluid. The OpenFOAM iteration was stopped when the relative energy balance defined in Equation 34 had been less than 1×10^{-4} for 200 time steps in a row. On the first coupled iteration the limit was set to 0.2 to speed up the coupled calculation.

Table 3: Boundary conditions for the fluid region.

	inlet	outlet	fluid to cladding interface
U	mapped setAverage true average 3.6 m/s	inletOutlet	fixedValue (0 0 0)
p	calculated	calculated	calculated
p_rgh	fixedFluxPressure	fixedMean 15.51 MPa	fixedFluxPressure
T	fixedValue 561 K	zeroGradient	turbulentTemperature CoupledBaffleMixedMod
k	mapped setAverage false	zeroGradient	kqRWallFunction
omega	mapped setAverage false	inletOutlet	omegaWallFunction
mut	calculated	calculated	mutUSpaldingWallFunction
rho	fixedValue 751 kg/m^3	zeroGradient	calculated
alphat	calculated	calculated	alphatJayatillekeWallFunction

Table 4: Boundary conditions for the solid regions.

	inlet	outlet	solid-solid interfaces and cladding to fluid interface
T	zeroGradient	zeroGradient	turbulentTemperatureCoupledBaffleMixedMod

3.4.3 Serpent model

In the coupled calculations the problem geometry in Serpent was defined with Serpent's own geometry model. It's also possible to generate the geometry based on the unstructured OpenFOAM mesh or meshes but this slows down the calculations. In the Serpent input file the geometry and material definitions were set according to the design parameters presented in Table 2. Accurate material compositions for different materials are shown in Table 5. The gas gap between the cladding and

the fuel pellet was modelled as void. In the x- and y-directions reflective boundary condition and in the z-direction black boundary condition was used to model a horizontally infinite problem.

The k-eigenvalue criticality source method was used in the Serpent calculations. At each cycle of the coupling program 40×10^6 active neutron histories (100 cycles, 4×10^5 neutrons per cycle) were simulated. In addition on the first iteration 50 inactive cycles were run before starting to collect the results in order to allow the initial fission source distribution to converge. On the following iterations the number of inactive cycles was reduced to 20. The power distribution was relaxed on each iteration with a relaxation scheme presented in Section 5.3.

In order to couple Serpent with OpenFOAM separate interface files were created for each of the materials (cladding, fluid, gadolinia fuel, regular fuel) in the problem geometry. Each material region in the Serpent model corresponded to one region with separate mesh in the OpenFOAM model. With the interface temperature distribution in all of the regions and density distribution in the fluid region were passed from OpenFOAM to Serpent and the power distribution in the fuel regions was passed back to OpenFOAM. The adaptive search mesh in all regions was defined as:

```
10 1 275
```

The effect of search mesh size on performance is investigated in Section 5.2. The iteration process was started by running a Serpent calculation with uniform temperature and density distributions which produced a cosine shaped power distribution peaking at the center of the assembly. To accelerate convergence this initial power distribution was disregarded in the calculation of the relaxed power distribution on the sequential iterations. Total heating power of the assembly was set to 1.0368 MW.

Table 5: Material compositions for the Serpent model expressed in mass fractions except for the composition of coolant which is expressed in atomic fractions.

	Regular fuel	Poison fuel	Cladding	Coolant (in atomic fractions)
^{235}U	0.04275144	0.03559074	-	-
^{238}U	0.83872152	0.82826221	-	-
^{152}Gd	-	0.00003353	-	-
^{154}Gd	-	0.00037027	-	-
^{155}Gd	-	0.00253012	-	-
^{156}Gd	-	0.00352201	-	-
^{157}Gd	-	0.00271000	-	-
^{158}Gd	-	0.00432878	-	-
^{160}Gd	-	0.00385778	-	-
^{16}O	0.11852704	-	0.00125	0.333333
^{1}H	-	-	-	0.666667
Cr	-	-	0.00100	-
Fe	-	-	0.00210	-
Zr	-	-	0.98115	-
Sn	-	-	0.01450	-

4 Results

4.1 Coupled results

This section gives a brief overlook on the results of a single coupled calculation performed with the OpenFOAM and Serpent models presented in Section 3.4. The main results of this calculation are the high fidelity temperature and density distributions which can be used in an independent Monte Carlo calculation. Section 4.2 investigates how the fidelity of the temperature and density distributions affects the results of a Monte Carlo neutronics problem. The convergence of the coupled calculation is studied in Section 5.3.

The coupled calculation was run for 60+1 iterations. To accelerate convergence the power distribution of the first iteration was disregarded in the calculation of the relaxed power distribution on the sequential iterations. The total calculation time was approximately 72 h on a computer node consisting of two Eight-Core Intel Xeon E5-2680 2.7 GHz processors with 128 GB RAM memory. The OpenFOAM solver utilized 16 processors with OpenMPI library and Serpent was run with 16 OpenMP threads.

The power distribution is presented in Figure 3. The axial distribution is presented on a plane with a normal parallel with y-axis and origin at $y \approx -0.028$ m. The plane intercepts approximately the center of each fuel rod in the bottommost row. The location of the plane is visualized in Figure 4. The radial distribution is presented on plane with a normal parallel with z-axis and origin at $z \approx -0.58$ m. The power density has its maximum approximately at this height. Figures 5 and 6 present the axial and radial temperature distributions for the coolant, the fuel and the cladding. The fuel and the cladding temperature distributions are presented on the same planes as the power distribution. Radial temperature distribution of the coolant is presented on the same plane as other distributions. The axial temperature distribution of the coolant is presented on a plane with a normal parallel with the y-axis and origin at $y \approx -0.022$ m. The plane is located at the middle of the coolant region between the two bottommost fuel rod rows. The location of the plane is visualized in Figure 4.

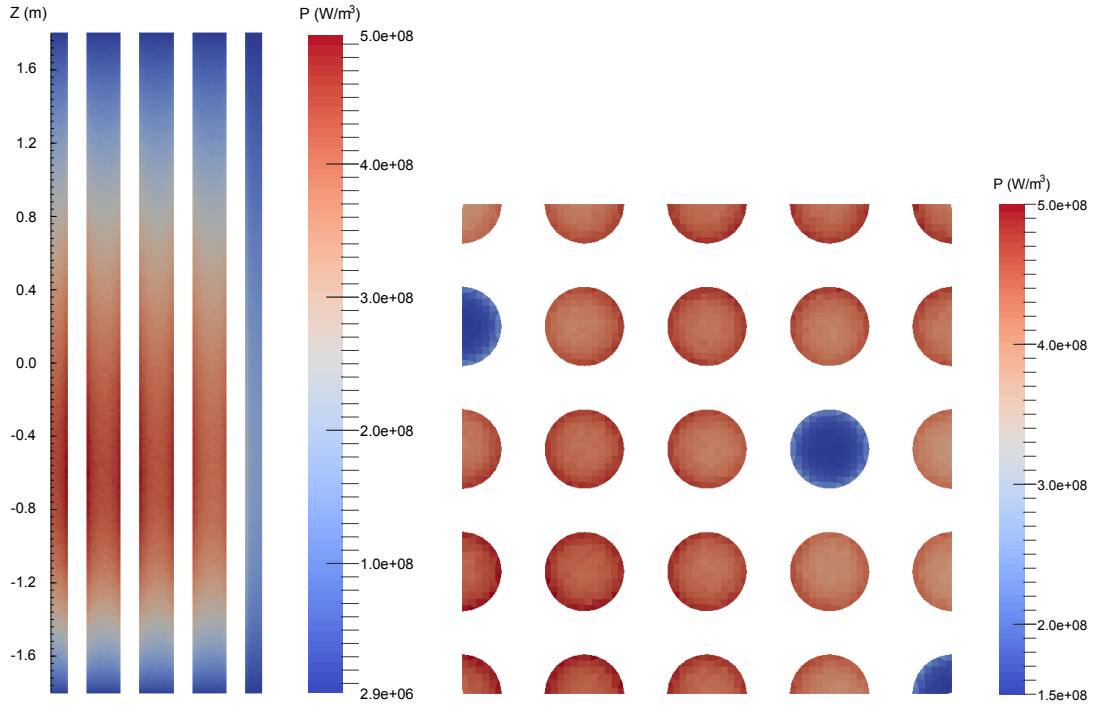
The power density peaks at the lower half of the assembly with a maximum approximately at $z \approx -0.58$ m. As the heat is deposited into the fuel, the axial temperature distribution of the fuel closely resembles that of the power density. The peak fuel temperature occurs at the lower half due to thermal feedback. The coolant enters the assembly from below and heats up as it flows through the assembly. The average temperature at inlet is $T_{\text{inlet}} \approx 561$ K and at outlet $T_{\text{outlet}} \approx 598$ K. Higher temperature results in lower density and less effective moderation. As the moderation is reduced, so is the thermal neutron flux and the power.

If the radial power density in a single rod is examined, it is observed that the power density is the highest on the surface of the rod and decreases towards the center of

the rod. This is due to the fact that the neutron flux is lower in the center of the rod than in the region near the surface of the rod, as some of the neutrons get absorbed before they reach the center of the rod.

Approximately 95 % of the total power is produced in the rods with regular fuel. This is expected as the gadolinia in the poisoned fuel absorbs effectively neutrons and suppresses the neutron flux. Due to uneven power distribution in the regular and poisoned fuel, average temperature in the regular fuel is approximately 920 K and in the poisoned fuel 700 K. The effect of uneven power distribution is also observed in the coolant temperature. Coolant temperature near the fuel rods with gadolinia is lower than in the other parts of the assembly.

In general coolant temperature near the surfaces of the fuel rods is higher than in the middle of the flow channels. Fuel temperature is the highest at the rod centres and there is a temperature jump in the fuel cladding interface due to the added thermal resistance in the OpenFOAM model.



(a) Axial (b) Radial

Figure 3: Axial power density distribution at y≈ – 0.028 m and radial power density distribution at z≈ – 0.58 m for the fuel.

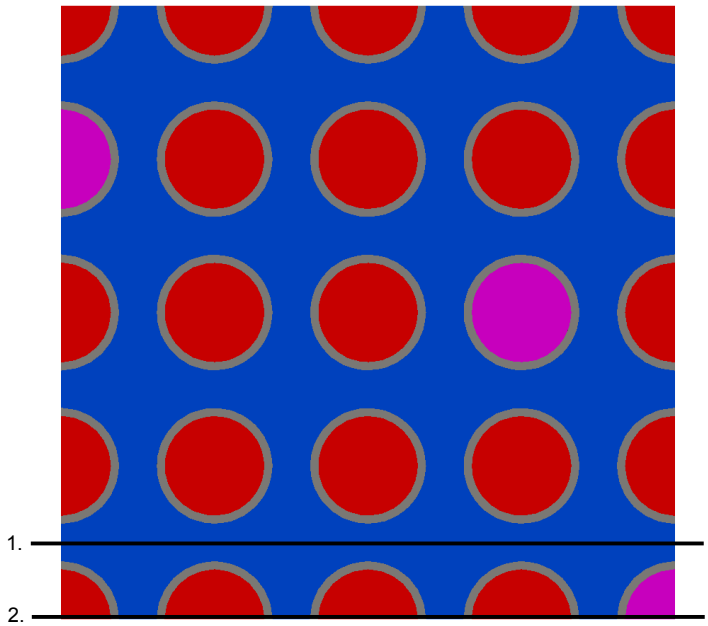
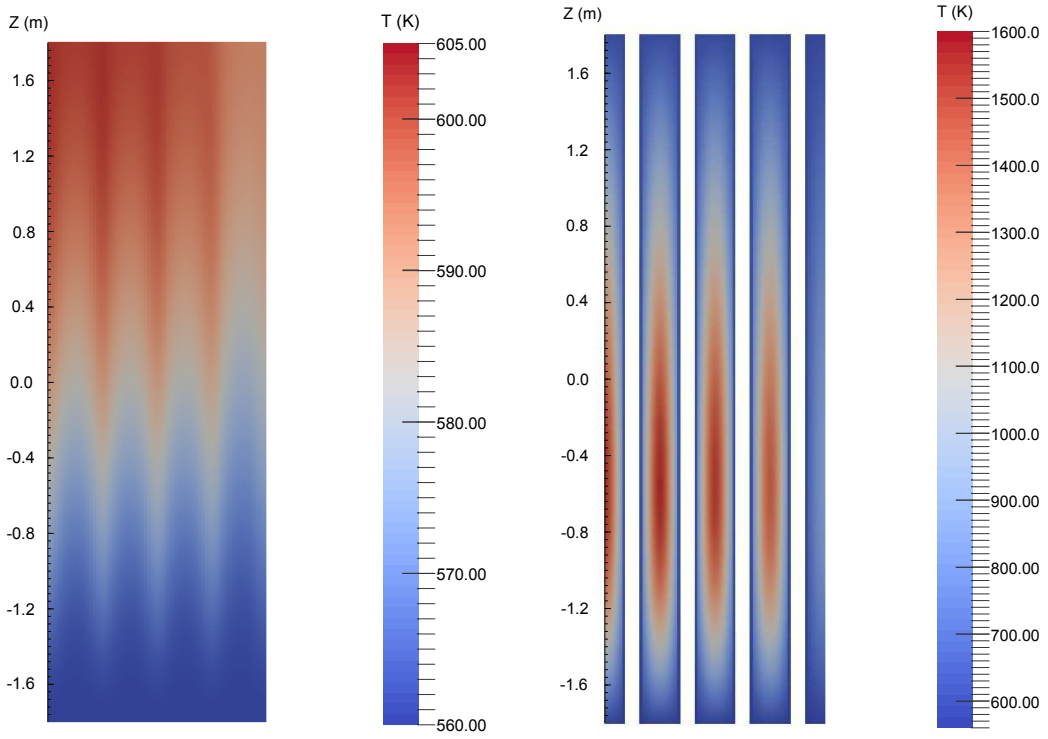
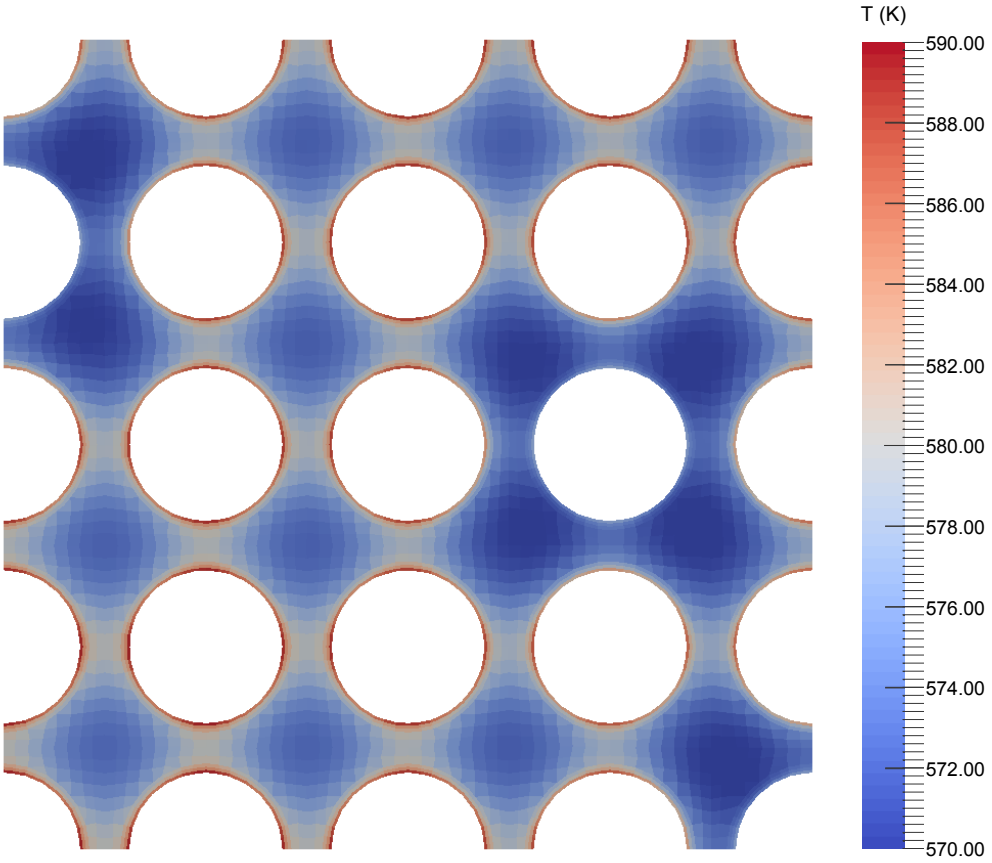


Figure 4: Axial distributions for the coolant are presented on plane number 1 and axial distributions for the fuel and the cladding on plane number 2.

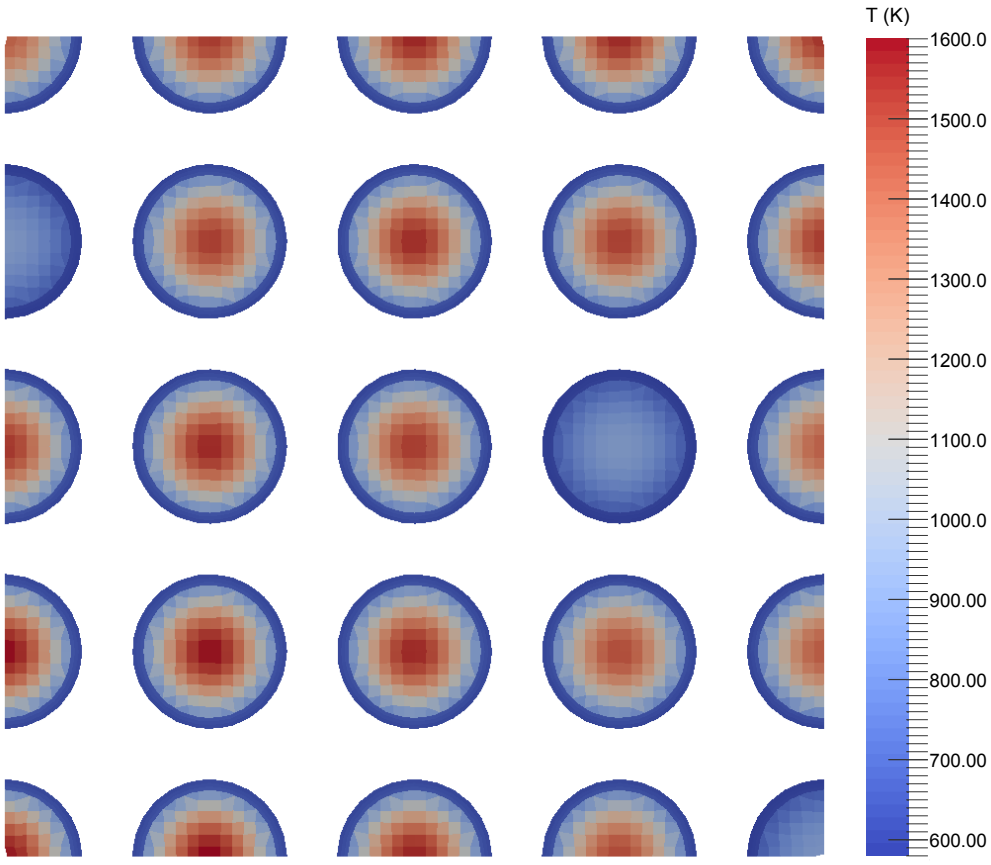


(a) Coolant (b) Fuel and cladding

Figure 5: Axial temperature distributions at $y \approx -0.022\,\mathrm{m}$ for the coolant and at $y \approx -0.028\,\mathrm{m}$ for the fuel and the cladding.



(a) Coolant



(b) Fuel and cladding

Figure 6: Radial temperature distributions at z≈ – 0.58 m.

4.2 Effect of spatial fidelity on the results

The effect of temperature and density field fidelity on the Monte Carlo neutronics was tested by running an identical Serpent calculation multiple times, each time with different level of axial temperature and density field refinement. The assembly was divided axially into 1, 4, 8, 16 or 256 layers with uniform temperature and density distribution. 256 corresponds to the number of axial cell layers in the OpenFOAM mesh. 16 is close to the number of layers in typical nodal core simulators. In addition one run was performed with the reference temperature and density distributions obtained from the coupled calculation. Each layer-wise temperature value was calculated as cell volume weighted average of the reference cell temperatures in the specific layer. A separate value was calculated for each OpenFOAM region or material so that the regular fuel, the gadolinia fuel, the cladding and the coolant had separate temperature distributions. The fuel and the cladding had already uniform density in the model so no averaging was necessary. Layer wise density distribution in the coolant was calculated with LibFluid from the layer-wise temperature distribution and average pressures. At each Serpent calculation 1×10^9 active neutron histories (5000 cycles, 2×10^5 neutrons per cycle) and 50 inactive cycles were simulated.

Table 6 presents the effective multiplication factor, mean neutron generation time, mean neutron lifetime and effective fraction of delayed neutrons for each level of axial refinement. No notable differences are observed in the mean neutron lifetime and in the delayed neutron fraction. The mean neutron generation time shows a difference between the calculation with constant material-wise distributions and the more refined cases. The effective multiplication factor increases as the level of axial refinement decreases. The biggest jump occurs from 4 axial layers to constant material-wise properties. It seems that a peaked temperature distribution and the corresponding density distributions has a larger negative reactivity effect than uniform distributions.

Table 6: Effective multiplication factor k_{eff}, mean neutron generation time Λ, mean neutron lifetime l and effective fraction of delayed neutrons β_{eff} for each level of axial refinement.

	k_{eff}	Λ (s)	l (s)	β_{eff}
Reference	$1.16605 \pm 2 \times 10^{-5}$	$1.1995 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.3986 \times 10^{-5} \pm 2 \times 10^{-9}$	$7.07 \times 10^{-3} \pm 1 \times 10^{-5}$
256	$1.16613 \pm 2 \times 10^{-5}$	$1.1993 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.3984 \times 10^{-5} \pm 2 \times 10^{-9}$	$7.07 \times 10^{-3} \pm 1 \times 10^{-5}$
16	$1.16615 \pm 2 \times 10^{-5}$	$1.1996 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.3989 \times 10^{-5} \pm 2 \times 10^{-9}$	$7.08 \times 10^{-3} \pm 1 \times 10^{-5}$
8	$1.16630 \pm 2 \times 10^{-5}$	$1.1993 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.3988 \times 10^{-5} \pm 2 \times 10^{-9}$	$7.06 \times 10^{-3} \pm 1 \times 10^{-5}$
4	$1.16686 \pm 2 \times 10^{-5}$	$1.1992 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.3992 \times 10^{-5} \pm 2 \times 10^{-9}$	$7.06 \times 10^{-3} \pm 1 \times 10^{-5}$
Constant	$1.16836 \pm 2 \times 10^{-5}$	$1.1970 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.3985 \times 10^{-5} \pm 2 \times 10^{-9}$	$7.07 \times 10^{-3} \pm 1 \times 10^{-5}$

4.2.1 Axial power distribution

The normalized axial power distribution for each level of refinement is presented in Figure 7. Each distribution was formed by tallying the total fission energy produc-

tion in 256 axial bins. The power distribution calculated with constant material-wise density and temperature distributions has a cosine like shape with a maximum at the middle of the assembly as expected. The other power distributions are closer to the reference. What is interesting, is the fact that there is a observable difference between the reference and the power distribution obtained with 256 axial layers. These distributions are presented in Figure 8. Both distributions peak at the same height but the one with 256 layers is shifted towards the upper half of the assembly. This indicates that the horizontal refinement which is lost in the averaging process has a large enough effect to be observable even in the axial power profile. Important variations in the horizontal temperature field result from the fact that certain fuel rods heat up more than others and these differences also affect the density distribution of the surrounding coolant. Same coolant flows through all axial layers.

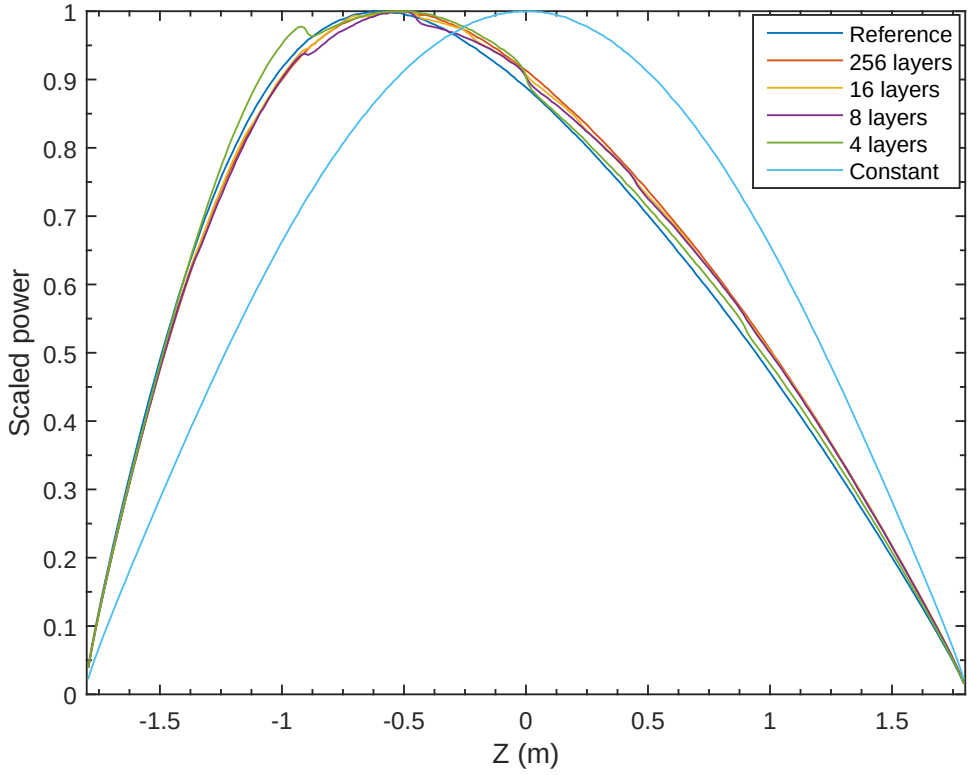


Figure 7: Normalized power distribution for each level of axial refinement.

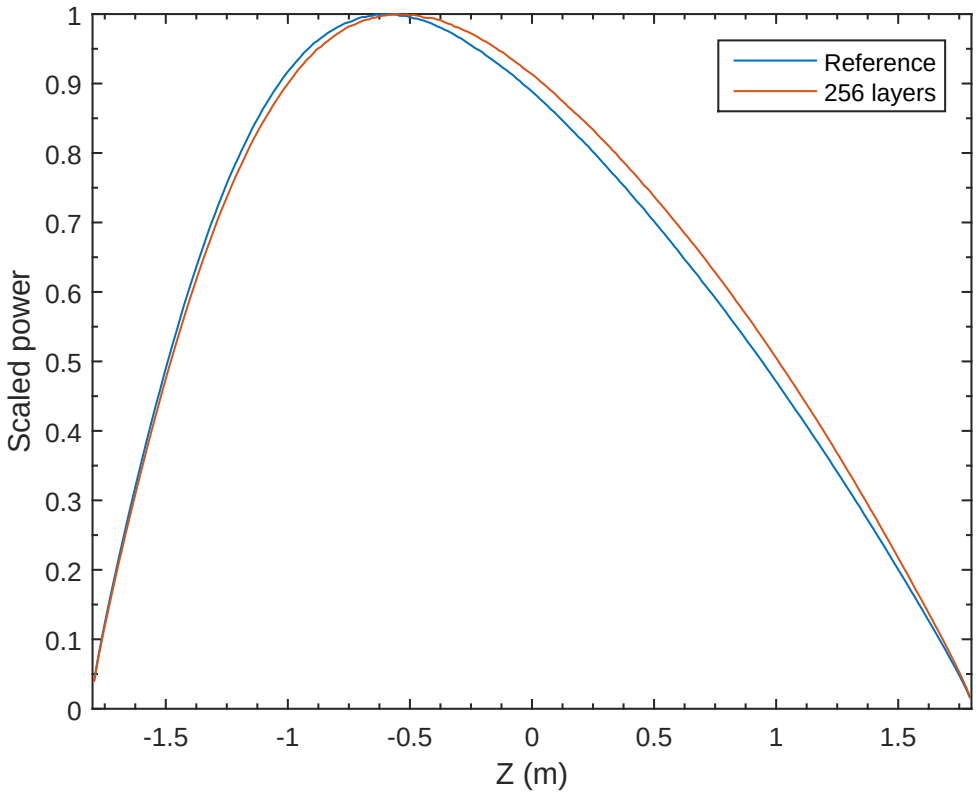


Figure 8: Normalized power distribution for reference case and for the distributions with 256 axial layers.

4.2.2 Neutron flux spectrum

Neutron flux spectrum at 8 different heights for the reference temperature and density distributions is shown in Figure 9. The maximum value of each spectrum is scaled to one. Several important features can be observed. First of all, the increase in the neutron flux around $E \approx 0.1\,\text{eV}$ corresponds to the thermal neutron flux. Secondly, the increased flux at $E \approx 1\,\text{MeV}$ is related to the fast neutrons emitted by the fission process. Finally, the dips in the range between fast and thermal neutron flux are due to resonances. Spectra in the layer with midpoint at $z = -0.675\,\text{m}$ for the reference and for the run with 256 axial layers are presented in Figure 10. The spectra are almost identical. Figure 11 presents the thermal neutron peak and Figure 12 the resonance at $E \approx 6.7\,\text{eV}$ in the same layer for a few different levels of axial refinement. No notable differences are observed in the resonance. The scaled height of the thermal neutron peak varies with the level of axial refinement. This is mostly due to the different axial moderator density distributions which affect the size of thermal neutron population compared to the fast neutron population at the fixed height.

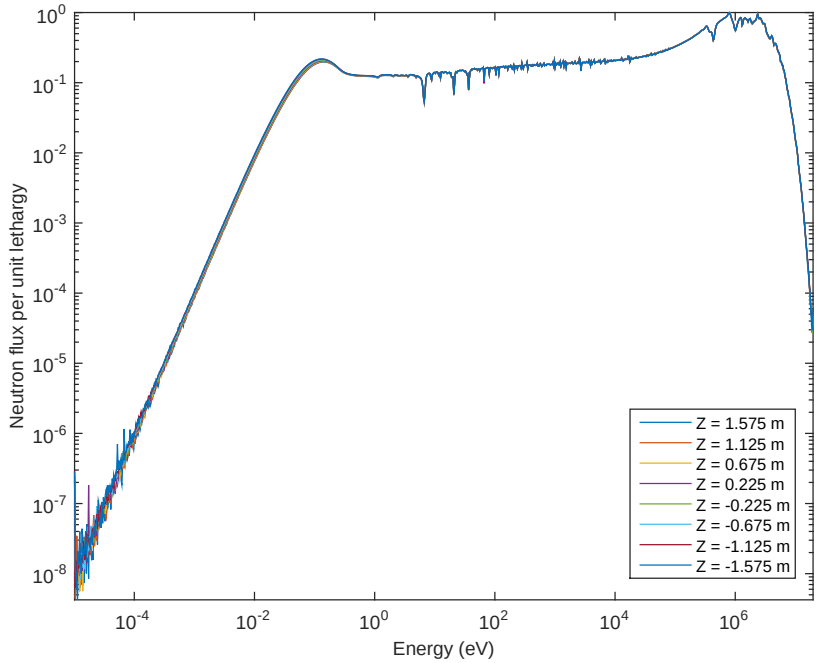


Figure 9: Neutron flux spectrum at different heights for the reference temperature and density distributions. The z-coordinates are the midpoints of the axial layers.

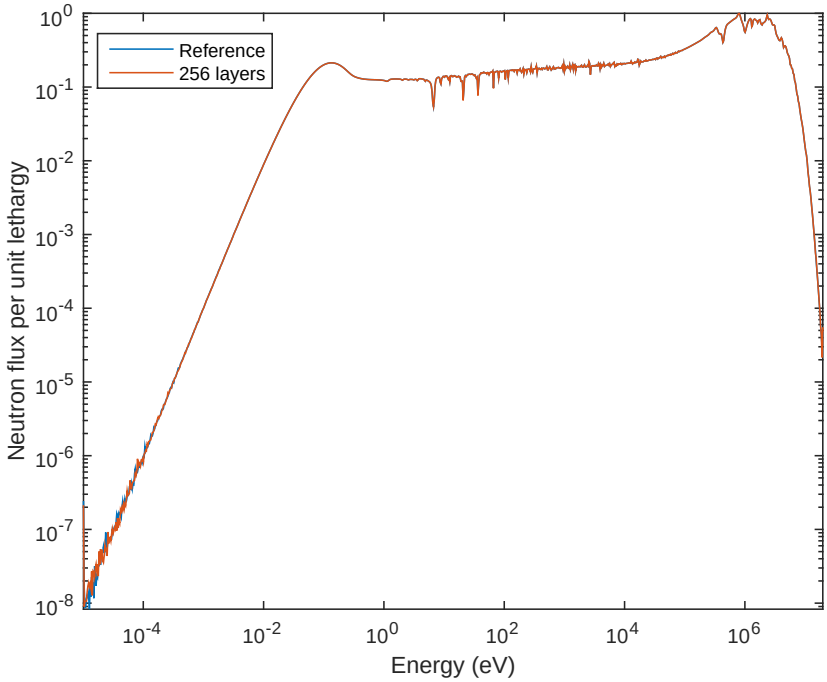


Figure 10: Neutron flux spectra in the layer with midpoint at $z = -0.675$m for the reference and for the run with 256 axial layers.

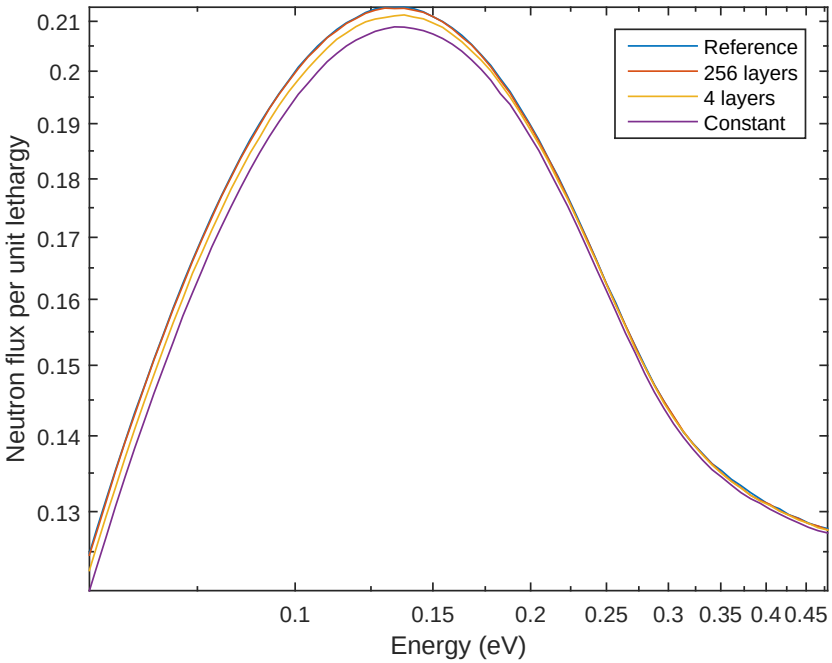


Figure 11: Thermal neutron peak in the layer with midpoint at $z = -0.675\,\mathrm{m}$ for a few different levels of axial refinement.

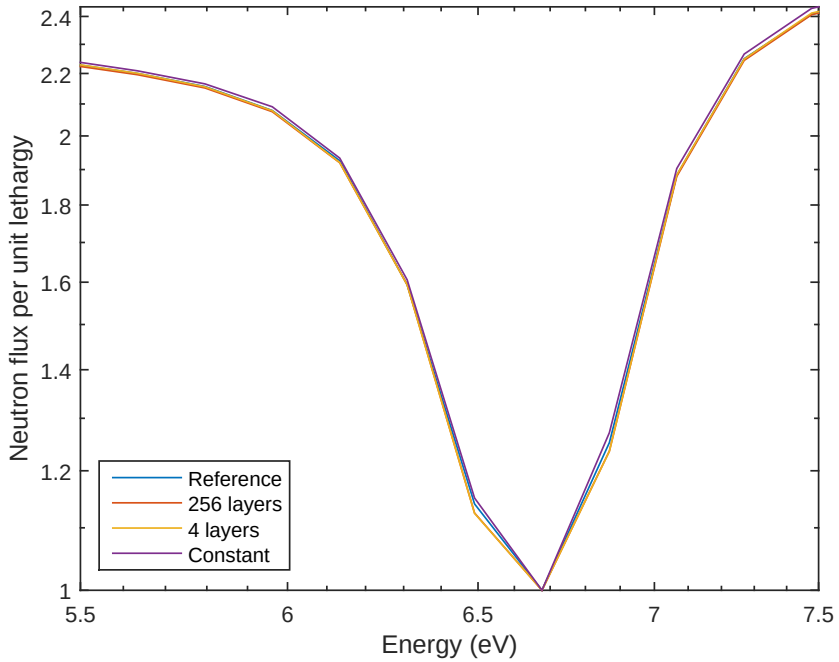


Figure 12: Resonance at $E \approx 6.7\,\mathrm{eV}$ in the layer with midpoint at $z = -0.675\,\mathrm{m}$ for a few different levels of axial refinement.

Figure 13 presents the energy region of the neutron flux spectrum with the thermal neutron peak in more detail. The peaks exhibit two properties which vary with height. To start with, the peaks are shifted to higher energies as the average height of the axial bin increases. This due to the moderator temperature which increases as the water flows through the assembly from bottom to top. In the thermal region the neutrons are in a thermal equilibrium with the atoms of the moderator. Higher moderator temperature results in the increase of the average thermal neutron energy. Secondly, the maximum value of the thermal peak decreases with increasing height. Higher water temperature leads to lower density which in turn results in less effective moderation and the thermal neutron population decreases compared to the fast neutron population.

The dip in the neutron flux due to resonance at $E \approx 6.7\,\text{eV}$ is shown in Figure 14. For illustration purposes the minimum of neutron flux in the visible energy range is scaled to one. The dip becomes wider as the fuel temperature increases. This is due to the Doppler effect. The width reaches its maximum value in the axial bin with center at $z = -0.675\,\text{m}$. The maximum power and the maximum fuel temperature are also located in this axial layer.

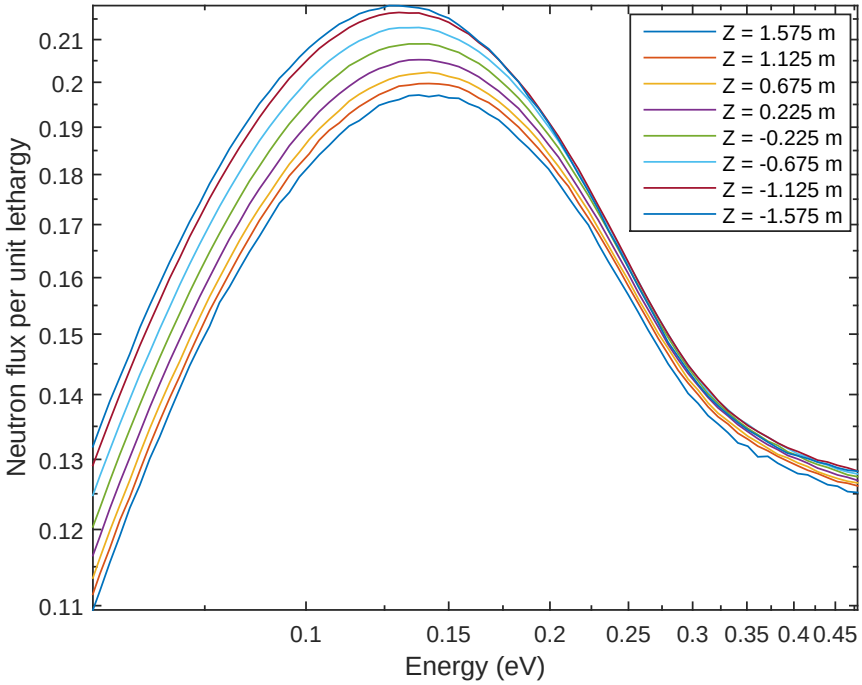


Figure 13: Thermal neutron peak at different heights for the reference temperature and density distributions.

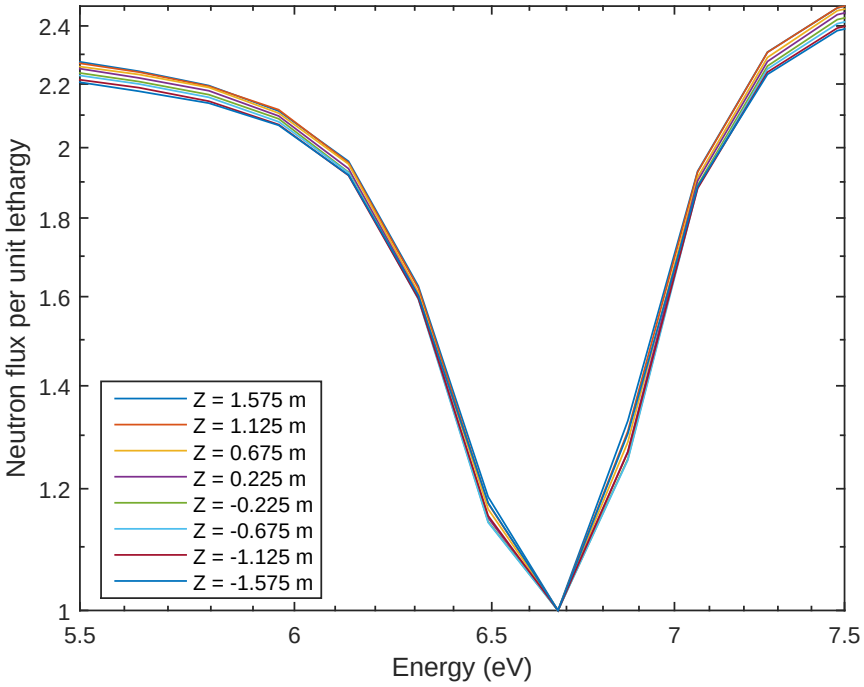


Figure 14: Resonance at $E \approx 6.7\,\text{eV}$ at different heights for the reference temperature and density distributions.

4.2.3 Collision density in the moderator

Collision density in the moderator for the reference and for the run with 256 axial layers is presented in Figures 15 and 16. The collisions were tallied in 2500 bins with equal volumes with 50 bins in the x-direction ($x_{\text{min}} = -0.028854\,\text{m}$, $x_{\text{max}} = -0.014427\,\text{m}$), and 50 bins in the y-direction ($y_{\text{min}} = 0.0\,\text{m}$, $y_{\text{max}} = 0.014427\,\text{m}$). In the z-direction the bin limits were $z_{\text{min}} = -0.6\,\text{m}$ and $z_{\text{max}} = -0.5\,\text{m}$.

In both figures the collision density decreases near the rod in the upper left corner. This rod contains gadolinium which effectively absorbs neutrons resulting in a decrease in the neutron flux and collision density. In the reference case the decrease is slightly compensated by higher moderator density near the gadolinia rod compared to regular rods as less power is produced in the gadolinia fuel and therefore the surrounding water is in lower temperature. As the macroscopic total cross-section is directly proportional to density, higher moderator density leads to higher collision density. The non-uniform horizontal density distribution is also the reason why the collision density is elevated in the middle of the flow channel, where the water temperature is lower and density higher. These observations are limited to the reference case, as in the other cases the density distribution is horizontally uniform. The moderator density for the reference case at $z = -0.55\,\text{m}$ is presented in Figure 17.

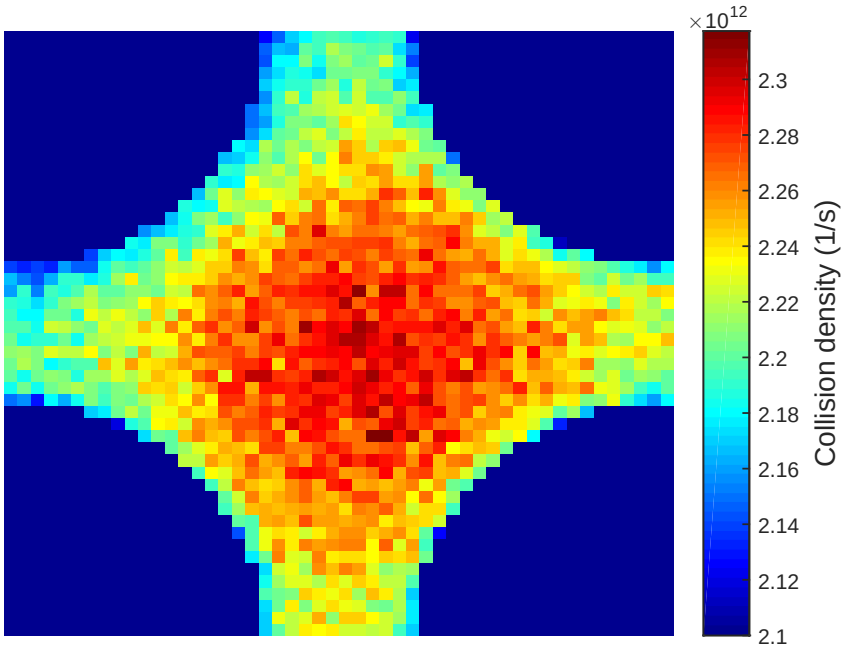


Figure 15: Collision density in the moderator for the reference distributions.

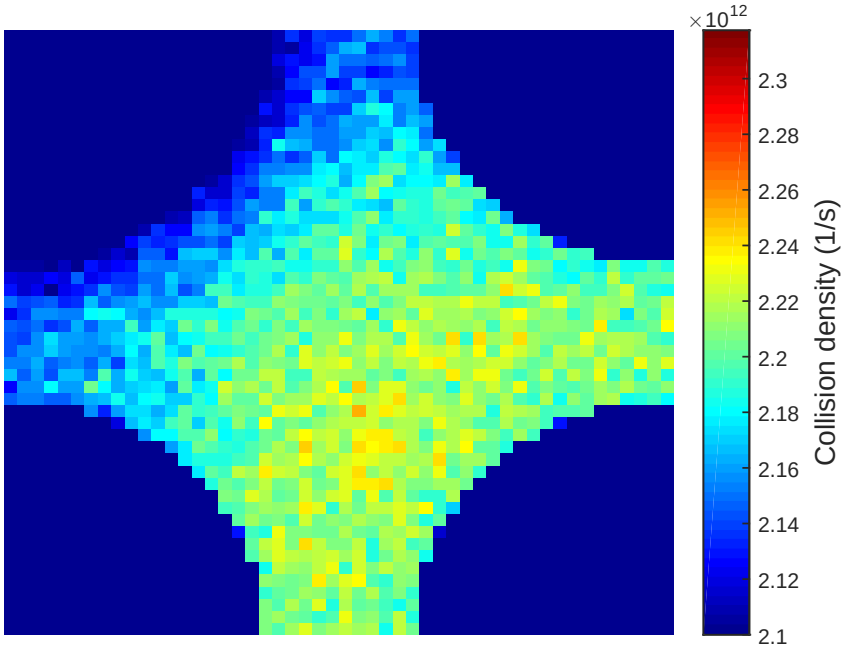


Figure 16: Collision density in the moderator for the run with 256 axial layers.

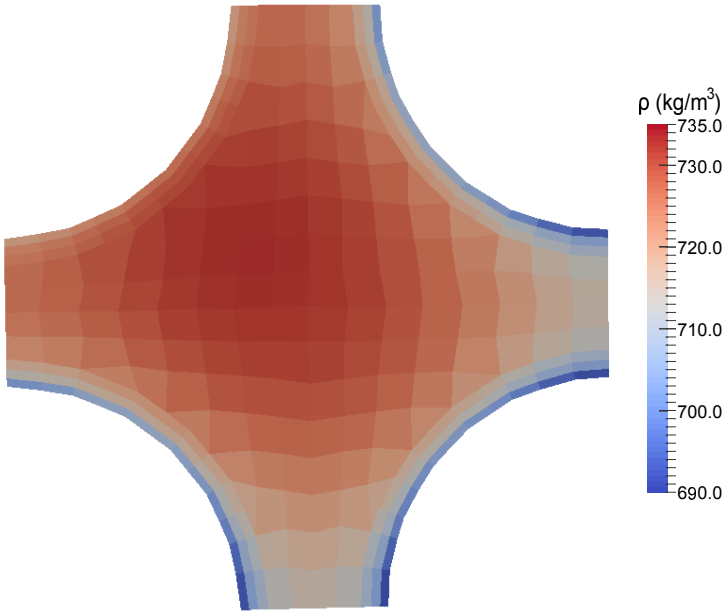


Figure 17: Moderator density for the reference distributions at $z = -0.55\,\mathrm{m}$.

4.2.4 Radial capture density

Radial capture density in the regular fuel at 8 different heights is presented in Figure 18 for reference distributions and in Figure 19 for distributions with 256 axial layers. The maximum capture density in each layer is scaled to 1.0. In the case with 256 axial layers the value of the scaled capture density at fixed radius decreases as the fuel temperature increases. Higher fuel temperature results in the increase of resonance absorption as the resonances are broadened (Doppler broadening). As the resonance absorption increases, so does the depression of fast neutron flux towards the center of the rod since the probability that a fast neutron is absorbed already in the outer layers increases. The total capture density is constant in the radial direction and therefore the increased fast flux depression explains the observation made about the axial dependency of the capture density.

Figure 18 demonstrates that the clear axial dependency observed in the radial capture density with 256 axial layers is no longer visible in the simulation with the reference distributions. This due to smaller differences in the fuel surface temperature in the 8 presented layers compared to the case with 256 axial layers. In the horizontally averaged temperature fields the fuel temperature and therefore the

fuel surface temperature ranges approximately from 580 K to 1130 K. With reference case the fuel surface temperature ranges approximately only from 580 K to 900 K. As the differences in the surface temperature are much smaller so are the differences in the resonance absorption in the fuel surface.

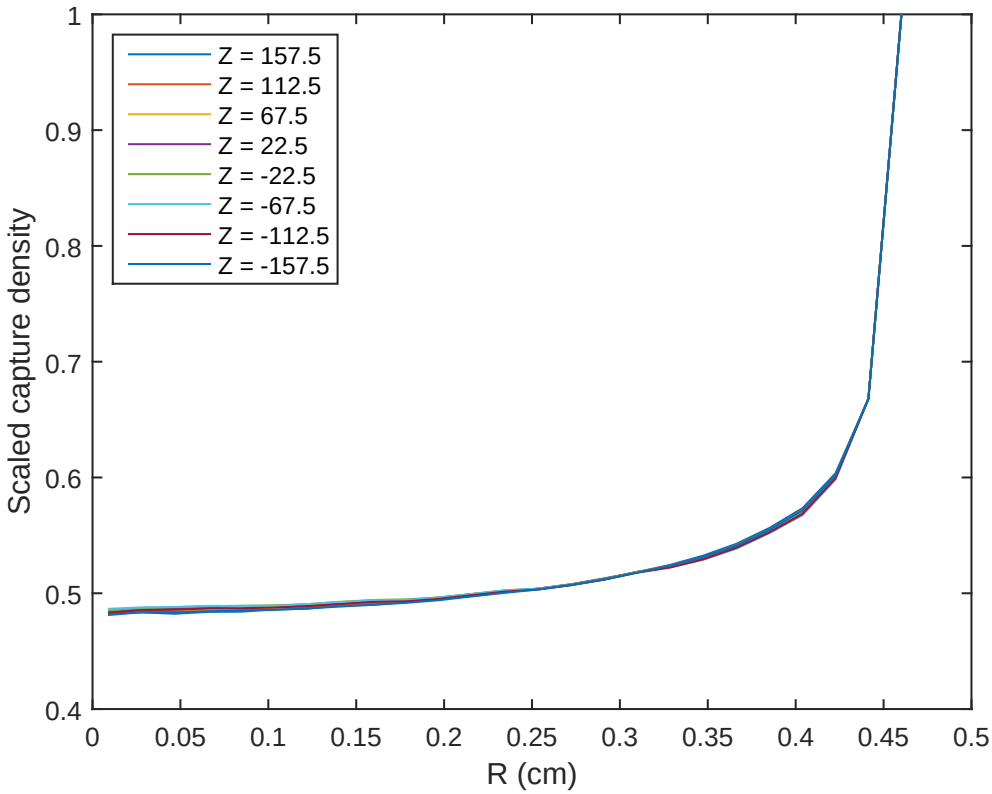


Figure 18: Radial capture density in the regular fuel at 8 different heights for the reference distributions.

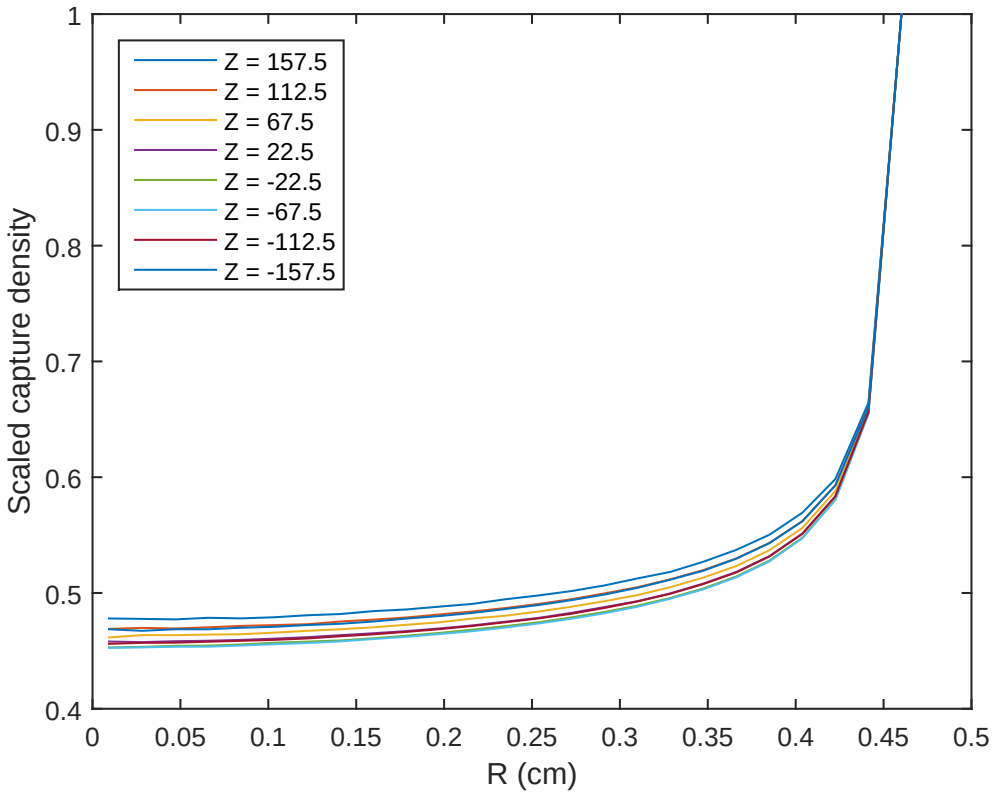


Figure 19: Radial capture density in the regular fuel at 8 different heights for the distributions with 256 axial layers.

4.2.5 Radial fission density

Radial fission density in the regular fuel at 8 different heights is presented in Figure 20 for the reference distributions and in Figure 21 for the distributions with 256 axial layers. The maximum fission density is scaled to 1.0. In both figures the scaled fission density at a fixed radius increases with increasing z-coordinate and decreasing moderator density. As shown in Figure 13 higher moderator density results in thermal neutrons with higher average energy. In the thermal range the fission and absorption cross-section decrease with increasing energy. Therefore thermal neutrons with higher energy are more likely to travel longer distance in the fuel without fission or absorption compared to those with lower energy.

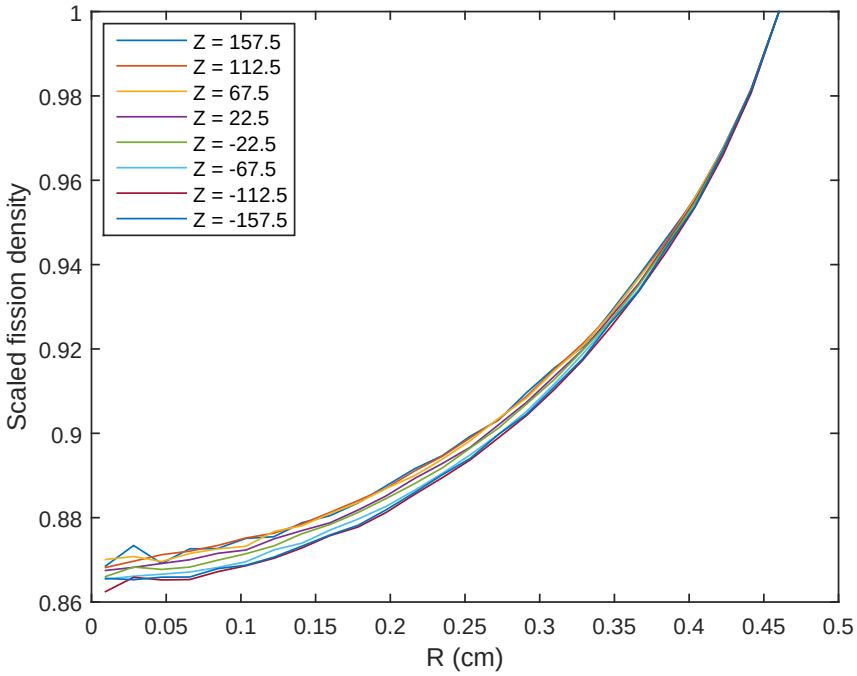


Figure 20: Radial fission density in the regular fuel at 8 different heights for the reference distributions.

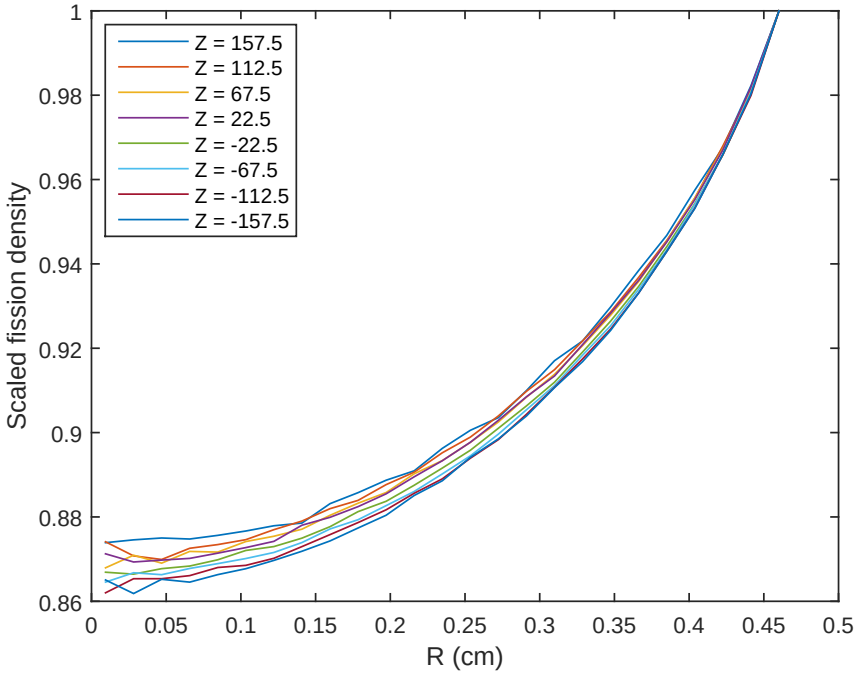


Figure 21: Radial fission density in the regular fuel at 8 different heights for the distributions with 256 axial layers.

All in all, the biggest differences in the results are observed when going from 4 axial layers to uniform distributions. Firstly, the effective multiplication factor is significantly increased and the power distribution peaks at the center of the assembly. Secondly, the height depended differences introduced above in the neutron spectrum, radial capture density and radial fission density are lost.

4.3 Feedback coefficients

To compare the relative importance of different feedback effects several feedback coefficients were evaluated for both the reference and the constant temperature and density distributions. For each coefficient a separate Monte Carlo simulation was run with modified distributions to get a new effective multiplication factor k_1. The resulting reactivity change was evaluated according to

$$\Delta\rho = \rho_1 - \rho_0 = \frac{k_1 - k_0}{k_1 k_0}, \tag{50}$$

where k_0 is the effective multiplication factor from the simulation with unmodified distributions. The feedback coefficient is then calculated as

$$f = \frac{\Delta\rho}{\Delta\theta}, \tag{51}$$

where $\Delta\theta$ is either temperature difference in K, coolant density (void fraction) difference as percentage or difference in the power expressed also as percentage. An error estimate based on total differential is evaluated as

$$\epsilon_f = \frac{1}{\Delta\theta}\left(\frac{\epsilon_{k_0}}{k_0^2} + \frac{\epsilon_{k_1}}{k_1^2}\right), \tag{52}$$

where ϵ_{k_0} is the error in the original effective multiplication factor and ϵ_{k_1} is the error in the effective multiplication factor from the simulation with modified distributions.

In each Serpent run 100×10^6 active neutron histories (500 cycles, 2×10^5 neutrons per cycle) and 50 inactive cycles were simulated. The following modifications were implemented to calculate the feedback coefficients. To calculate the coefficient for the fuel temperature each fuel cell temperature was increased by 50 K. For the coolant three different simulations and modifications were made. For the first simulation each cell temperature in the coolant was increased by 50 K and density distributions was left untouched. For the second simulation the increase in each temperature was 15 K and corresponding densities were calculated with LibFluid. The increase was limited to 15 K, as higher increase would lead to temperature higher that the saturation temperature in a large amount of coolant cells. For the third simulation the coolant void fraction was increased by decreasing the coolant density by 10 %. The four feedback coefficients presented so far were evaluated with both the reference temperature and density distributions, and the constant distributions. In addition, a power feedback coefficient was calculated for the reference

distributions. First, a coupled calculation was run with 5 % higher power to generate new temperature and density fields. Secondly, a Monte Carlo simulation was run with these fields to get a multiplication factor k_1.

The feedback coefficients are presented in Table 7. The coefficients are slightly more negative for the constant distributions but the differences are small. Especially, the coolant void fraction feedback coefficients agree within the error bounds. The reduction in the coolant density has by far the largest effect on reactivity. An increase in the fuel or coolant temperature has a very minor effect. As a conclusion, the most important feedback effect is caused by the reduced moderation as the coolant density is decreased.

Table 7: Feedback coefficients for the reference , and the constant temperature and density distributions.

	Reference	Constant
Fuel temperature	-1.8 ± 0.2 pcm/K	-2.3 ± 0.2 pcm/K
Coolant temperature	-0.6 ± 0.2 pcm/K	-1.2 ± 0.2 pcm/K
Coolant temperature and density	-37.2 ± 0.8 pcm/K	-39.3 ± 0.7 pcm/K
Coolant void fraction	-109 ± 1 pcm/%	-111 ± 1 pcm/%
Power	-27 ± 2 pcm/%	

5 Further considerations

5.1 Mesh based geometry

As Serpent is able to define the problem geometry based on the unstructured OpenFOAM mesh, in this section the results of simulations using Serpent's geometry model are compared to simulations with OpenFOAM mesh based geometry. As long as the geometry is identical, the results should agree. However, it proved to be difficult to produce an OpenFOAM mesh that has exactly the given dimensions, as the mesh generation process is partly automated. The volume of each material region in Serpent and OpenFOAM unstructured mesh based geometry is presented in Table 8. The biggest difference is in the volume of the cladding as the gas gap is replaced with cladding in the OpenFOAM mesh. Otherwise the volumes agree pretty well. Especially the differences in fuel volumes are very small which is vital as the fissile mass has a strong effect on the effective multiplication factor and neutronics in general.

Table 8: The volume of each material region in Serpent and OpenFOAM unstructured mesh based geometry.

Region	Serpent volume (cm^3)	OpenFOAM volume (cm^3)	Difference (%)
Coolant	6587	6658	+1.1
Cladding	1177	1400	+16.0
Regular fuel	3500	3500	-0.006
Gd fuel	430	430	-0.08
Total	11694	11988	+2.5

First of all, a coupled calculation with OpenFOAM mesh based geometry was run with the same parameters as the reference one presented in Section 4.1. A comparison on the final temperature fields of the coupled calculation with Serpents own geometry model and OpenFOAM mesh based geometry is shown in Table 9. All in all, the differences are so small that they have an insignificant effect on the neutronics. The most important feedback mechanism is related to the coolant temperature and the maximum absolute difference is less than 1 K and on average the differences are even smaller based on the L2-norm of the relative differences. The biggest differences are in the regular fuel in which the temperature variations in general are the largest. The observations are explained by the small differences in the geometries and by the stochastic nature of the Monte Carlo simulation.

Table 9: A comparison on the final temperature fields of the coupled calculation using Serpents own geometry model vs. OpenFOAM mesh based geometry.

Region	Max. abs. diff. (K)	Max. rel. diff.	L2-norm
Coolant	0.90	1.5×10^{-3}	1.2×10^{-4}
Cladding	1.47	2.2×10^{-3}	4.3×10^{-4}
Regular fuel	18.25	1.4×10^{-2}	3.0×10^{-3}
Gd fuel	5.32	6.3×10^{-3}	1.7×10^{-3}

Secondly, the effect of mesh based geometry on an individual Serpent calculation was studied by running an Monte Carlo simulation with identical temperature and density fields with Serpent's own geometry model and OpenFOAM mesh based geometry. Run parameters were identical to the ones presented in Section 4.2. The results are shown in Table 10. To start with, the transport cycle time is approximately 27 % higher with mesh based geometry. If possible the use of mesh based geometry should be avoided to save computational time. The advantage of mesh based geometry is in the modelling of highly irregular geometries which can not be defined in practise with Serpent's own geometry model.

The effective multiplication factor is slightly higher with mesh based geometry. The mean neutron generation time Λ and the mean neutron lifetime show both a statistically significant increase compared to the Serpent geometry. The observed differences might results from the slightly different geometry or there may a bug in the Serpent code. Serpent simulations with simpler exactly identical geometries did not show any statistically significant differences in the results.

Table 10: The results of Monte Carlo simulations with Serpents own geometry model and OpenFOAM mesh based geometry.

Mesh type	k_{eff}	Λ (s)	l (s)	Transport cycle time (min)
Serpent	$1.166\,05 \pm 2 \times 10^{-5}$	$1.1995 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.3986 \times 10^{-5} \pm 2 \times 10^{-9}$	610
OpenFOAM	$1.166\,20 \pm 4 \times 10^{-5}$	$1.2212 \times 10^{-5} \pm 2 \times 10^{-9}$	$1.4242 \times 10^{-5} \pm 2 \times 10^{-9}$	774

5.2 Search mesh optimization

The effect of search mesh optimization to memory usage and computational time was tested by running the same Serpent calculation with varying search mesh sizes. The problem geometry and materials were identical to the ones used in the coupled calculations and the geometry was generated based on the OpenFOAM meshes. Density and temperature distributions were obtained from the final iteration of the coupled calculation. Delta-tracking mode was set to full delta tracking (threshold value 1.0) and optimization mode to mode 1. At each Serpent run 20×10^6 active neutron histories (500 cycles, 4×10^4 neutrons per cycle) were simulated. In addition one inactive cycle was run as that is the minimum number of inactive cycles in

Serpent. For the sake of simplicity, the testing was carried out with a search mesh consisting of only one level. The simulations were run with 16 OpenMP threads on a computer node consisting of two Eight-Core Intel Xeon E5-2680 2.7 GHz processors with 128 GB RAM memory. Table 11 presents the results of the testing procedure. For each search mesh size the initialization time, the transport cycle time and the memory (RAM) usage are given. The initialization time includes the time required for the generation of the search mesh. As the name suggests, transport cycle time is the computational time required for neutron transport. Memory usages and transport cycle times as a function of search mesh size are presented in Figures 22 and 23.

Table 11: Results of the search mesh optimization.

Size (NxNxN)	Initialization time (min)	Transport cycle time (min)	Memory usage (GB)
10	0.92	1469.43	14.0
25	0.93	75.73	15.1
50	0.84	26.79	17.8
75	1.22	18.76	21.3
100	1.28	15.71	25.9
125	1.36	14.44	31.5
150	1.50	13.57	38.5
175	1.43	13.12	46.9
200	1.59	12.74	56.8
225	1.79	12.48	68.0
250	1.63	12.24	81.5
275	1.79	12.22	96.5
285	1.86	12.15	103.1
285 (Serpent geometry)	1.82	8.57	103.1

The results show that the initialization time and the memory usage increase with increasing search mesh size. This is expected, as Serpent has to determine for each search mesh cell which OpenFOAM cells fall into this particular cell and storing larger search mesh requires more memory. On the other hand, the transport cycle time decreases with increasing search mesh size. By making the search mesh denser, the average number of OpenFOAM cells falling into a search mesh cell decreases. This speeds up the transport cycle by decreasing the average number of OpenFOAM cells which are checked at each interaction point before finding the correct OpenFOAM cell. The smallest transport cycle time would be achieved with a search mesh in which the maximum number of OpenFOAM cells falling into any search mesh cell is one. In this ideal case each search mesh cell also directly determines the OpenFOAM cell and increasing the search mesh size further does not speed up the simulation. The transport cycle times for mesh sizes 250 and up are very close to each other. This might indicate that the calculation time-wise opti-

mal search mesh size has been reached. In this test case the use of OpenFOAM mesh based geometry increased the transport cycle time approximately by 40 % compared to Serpents own geometry model. Using the default delta-tracking threshold value instead of 1.0 increased the transport cycle time to approximately 20 minutes. Completely disabling delta-tracking further increased the required time to approximately 4 hours.

Based on the results of the testing procedure, the search mesh size has a significant impact on both the required computational time and the memory usage. Optimal parameters defining the search mesh are case dependent and the search mesh must be optimized separately for each OpenFOAM mesh. In general it can be stated that the finer the better.

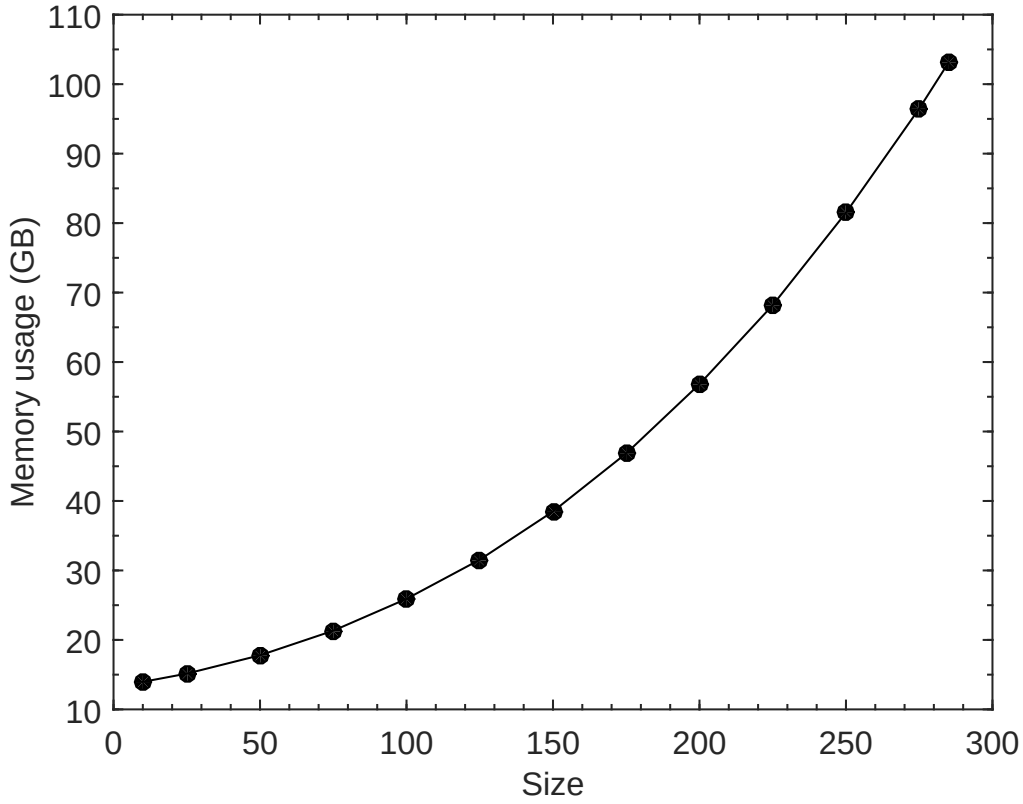


Figure 22: Memory usage as a function of search mesh size.

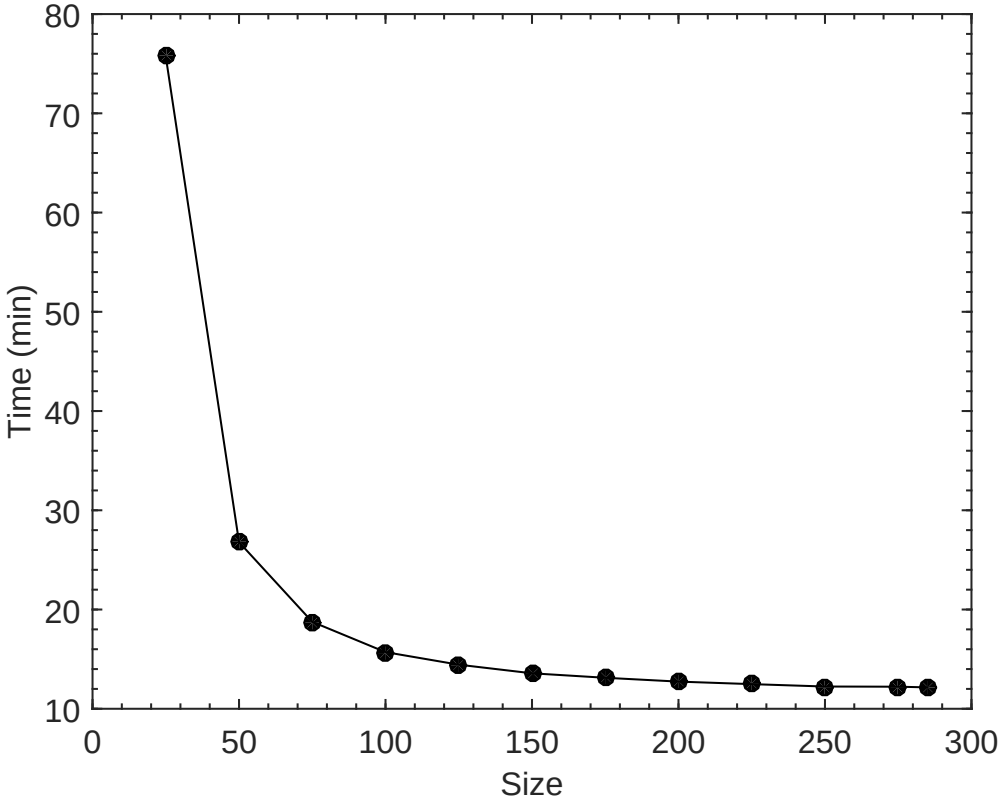


Figure 23: Transport cycle time as a function of search mesh size. Transport cycle time for the 10x10x10 search mesh is omitted for better visualization.

5.3 Relaxation and convergence

The stochastic nature of the Monte Carlo neutronics poses a significant convergence challenge in the coupled calculations. There is always some statistical uncertainty in the fission power distribution and therefore two separate Monte Carlo calculations with identical temperature and density distributions do not produce exactly identical power distributions. As described in [24], the goal in the coupled calculation is to find a solution to the following fixed point problem

$$P = \hat{G}(T(P), D(P)), \tag{53}$$

where $T(P)$ and $D(P)$ are the temperature and density distributions, and $\hat{G}(T(P), D(P))$ is the estimate of the power distribution given by the Monte Carlo method. This estimate contains an additive noise term $\epsilon(P)$

$$\hat{G}(T(P), D(P)) = G(T(P), D(P)) + \epsilon(P). \tag{54}$$

One could try to solve equation 53 by simple iteration as

$$P^{(n+1)} = \hat{G}(T(P^{(n)}), D(P^{(n)})) \tag{55}$$

and updating the temperature and density distributions on each iteration. Clearly, the convergence is limited by the magnitude of $\epsilon(P)$ and one must simulate a huge amount of neutron histories to reduce the noise to an acceptable level. Also, this method is unstable in many problems and it would diverge almost surely if the effect of xenon was modelled. To overcome these problems Serpent uses a relaxation scheme based on the stochastic approximation scheme presented in Reference [24]. The fission power distribution is relaxed according to

$$P_{\text{rel}}^{(n+1)} = P_{\text{rel}}^{(n)} - \frac{s_{n+1}}{\sum_{i=1}^{n+1} s_i} \alpha (P_{\text{rel}}^{(n)} - P^{(n+1)}), \tag{56}$$

where $P_{\text{rel}}^{(n+1)}$ and $P_{\text{rel}}^{(n)}$ are the relaxed power distributions on iterations $n+1$ and n. $P^{(n+1)}$ is the iteration wise power distribution on iteration $n+1$, s_i the number of simulated neutrons on iteration i and α is an additional under-relaxation factor. It can be shown easily with mathematical induction that this relaxation scheme combines the power distributions of all of the iterations with weights $s_i / \sum_i s_i$. If the same number of neutron histories is simulated on each iteration, the weight simplifies to $1/n$, where n is the number of iterations. With each iteration the statistical uncertainty in the power distribution is reduced and by running enough iterations any convergence level can be reached.

In this thesis the convergence of the coupled calculation was evaluated retrospectively by comparing the regular fuel temperature distributions on two consecutive iterations. The variations in the regular fuel temperature from iteration to iteration were larger than in other regions as most of the fission heat was deposited in this fuel type. The convergence was measured with three quantities. First of all, the maximum absolute and relative differences are given by

$$E_{\text{abs}} = \max_i |T_i^{(n+1)} - T_i^n| \tag{57}$$

and

$$E_{\text{rel}} = \max_i \left| \frac{T_i^{(n+1)} - T_i^n}{T_i^{(n+1)}} \right|. \tag{58}$$

The third convergence parameter is based on volume weighted L2-norm of the relative difference defined as

$$E_{\text{L2}} = \sqrt{\sum_i \frac{V_i}{V} \left(\frac{T_i^{(n+1)} - T_i^n}{T_i^{(n+1)}} \right)^2}, \tag{59}$$

where V is the total volume.

The convergence parameters as a function of iteration for the coupled calculation presented in Section 4.1 are shown in Figure 24. To accelerate convergence the initial power distribution produced by the first Serpent simulation was disregarded in the calculation of the relaxed power distribution on the sequential iterations.

This is also taken into account in Figure 24 in which the first presented iteration is in fact the second. In total the coupled calculation had 60+1 iterations.

As the number of simulated neutrons is the same on each iteration, the statistical uncertainty in the power distribution should decrease as $1/n$ after the system has reached a physically converged state in which the temperature and density distributions only change due to random statistical fluctuations in the power distribution. To test this prediction a function of the form $y = A \cdot x^B$ was fitted to the convergence parameter values with iteration number higher than 10. Based on the theory the value of B should equal -1. The first 10 iterations were not taken into account in the fit to assure that physical convergence had been achieved. After 10 iterations the maximum absolute difference in the coolant and cladding temperature distributions are ~0.3 K and ~0.5 K on consecutive iterations. As these changes are very small, it can be argued that the physically converged state has been most likely achieved. Figure 25 shows the fit determined with the method of least squares and the original data points for the maximum absolute difference. The least square estimate for B is $B_{\text{abs}} \approx -1.04$. In the case of maximum relative difference and L2-norm the estimators are $B_{\text{rel}} \approx -1.05$ are $B_{\text{rel}} \approx -1.15$. The results of a simple fit with no consideration on the uncertainty are not enough to make any reliable conclusions but the convergence parameters seem to follow approximately the $1/n$-behaviour predicted by theory.

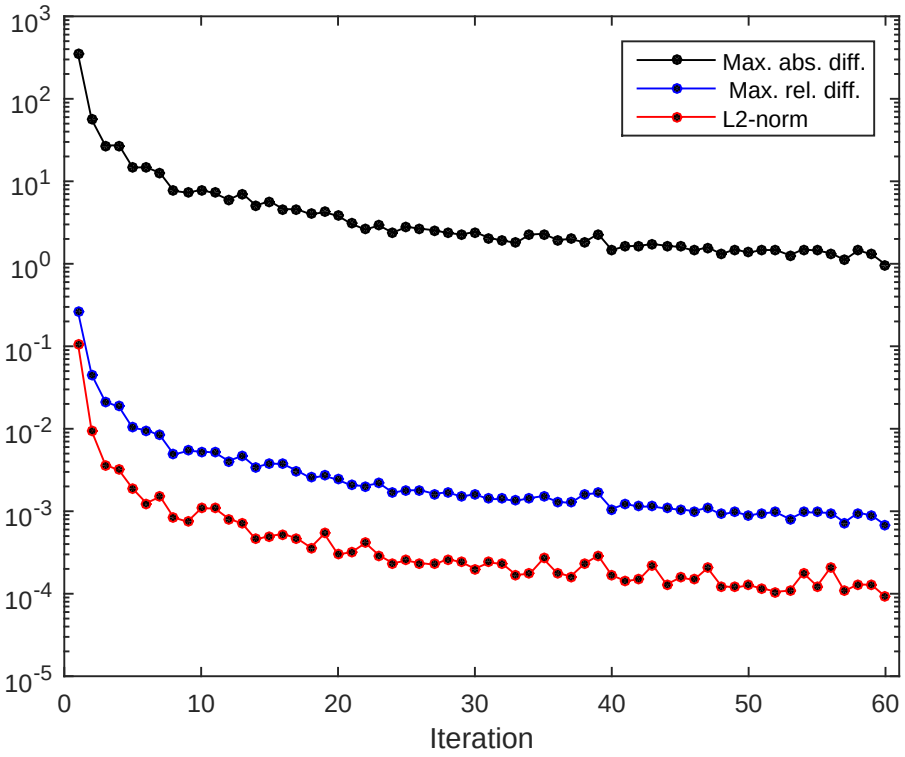


Figure 24: Convergence parameters as a function of coupled iteration.

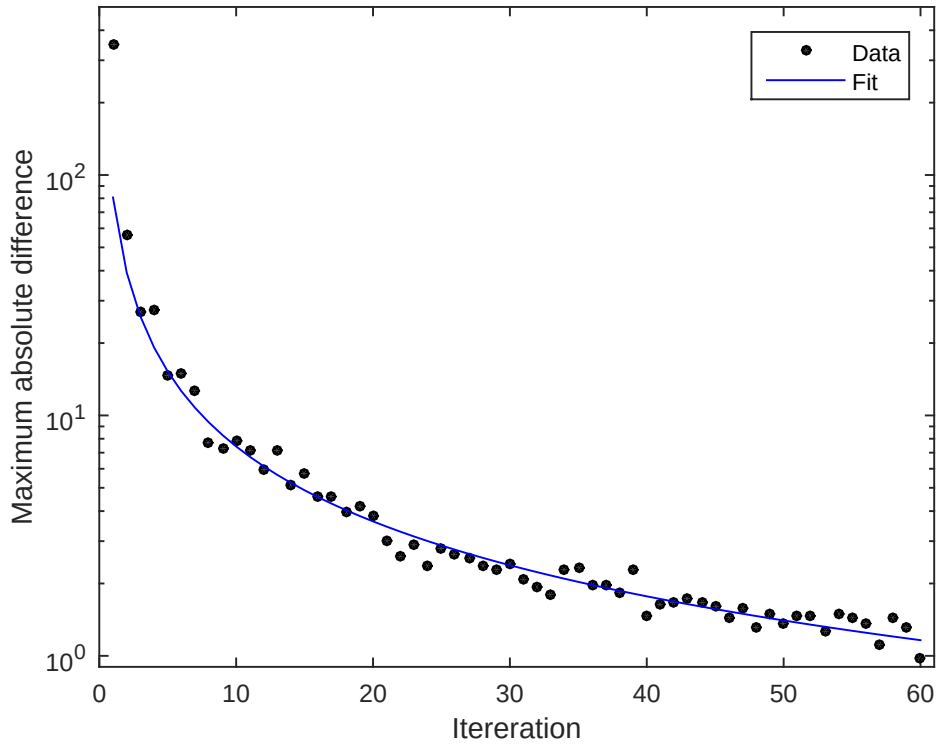


Figure 25: Maximum absolute difference and $y = A \cdot x^B$ fit as a function of coupled iteration.

To demonstrate the effectiveness of the stochastic approximation based relaxation scheme an additional coupled calculation was run with a fixed under-relaxation factor. The power distribution was relaxed on each iteration according to

$$P_{\text{rel}}^{(n+1)} = (1-\alpha)P_{\text{rel}}^{(n)} + \alpha P^{(n+1)}, \tag{60}$$

where the under-relaxation factor α was set to $\alpha = 0.1$. Otherwise the calculation was performed with the same parameters as the reference. The convergence parameters as a function of iteration with fixed under-relaxation factor are presented in Figure 26. All of the convergence parameters reach their respectful minimum values only after a few iterations. This is due to the constant under-relaxation factor as each time the power distribution is relaxed, an fixed amount of statistical noise is added to the distribution. In this case better convergence can not be achieved by increasing the number of coupled iterations. To achieve better convergence either the under-relaxation factor must be lowered or the statistics of the Monter Carlo calculation must be improved by increasing the number of simulated neutrons. Both of these increase the required computational time.

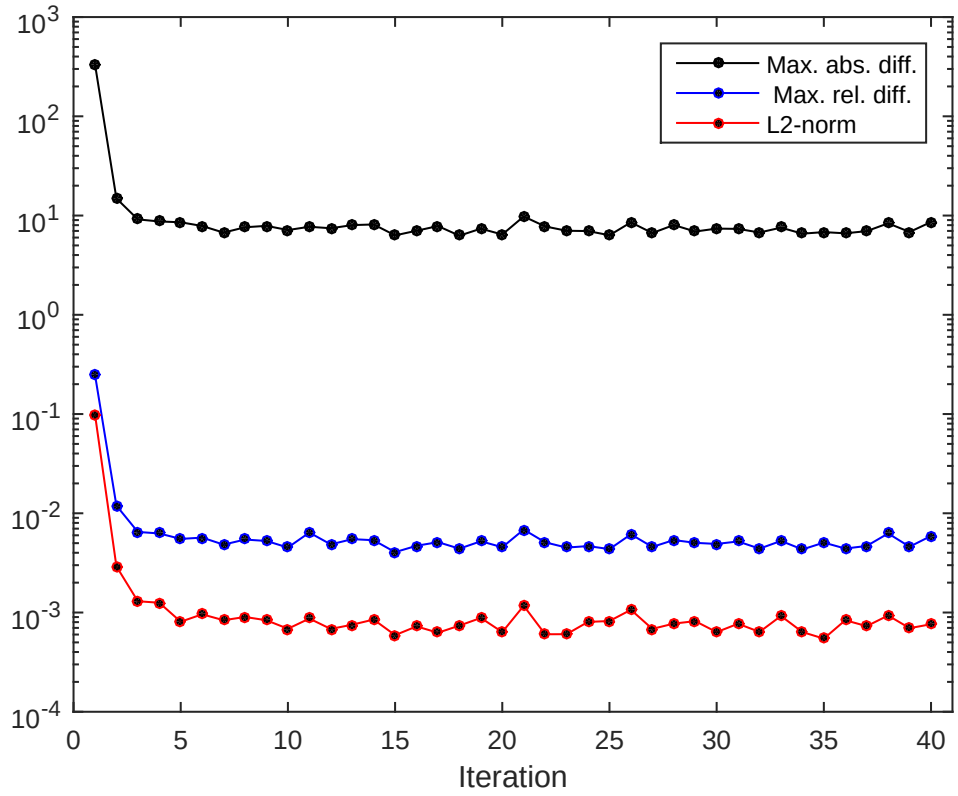


Figure 26: Convergence parameters as a function of coupled iteration with fixed under-relaxation factor.

Maximum absolute differences for both tested relaxation schemes with two different number of simulated neutrons per iteration are presented in Figure 27. For this comparison two additional coupled calculations were run with 10×10^6 neutrons per iteration and 160 iteration cycles. The reduction in the number of neutrons per iteration has an expected effect on the calculation with fixed relaxation factor and the minimum value of the maximum absolute difference is increased as the amount of statistical noise added to the power distribution on each iteration is increased. With the relaxation based on the stochastic approximation the maximum absolute difference decreases more rapidly as a function of the total number of simulated neutrons with lower number of simulated neutrons per iteration. This makes sense, as even though the statistical noise added to the power distribution due to decreased number of simulated neutrons per iteration in increased, at the same time the number of iteration needed to simulate the same total number of neutrons is increased. When the number of iterations is increased, the under-relaxation factor for a fixed total number of simulated neutrons decreases and has a more prominent effect. However, it is important to realize that simulating a certain total amount of neutrons is much faster with a higher number of neutrons per iteration as on each iteration the CFD calculation requires a constant amount of time.

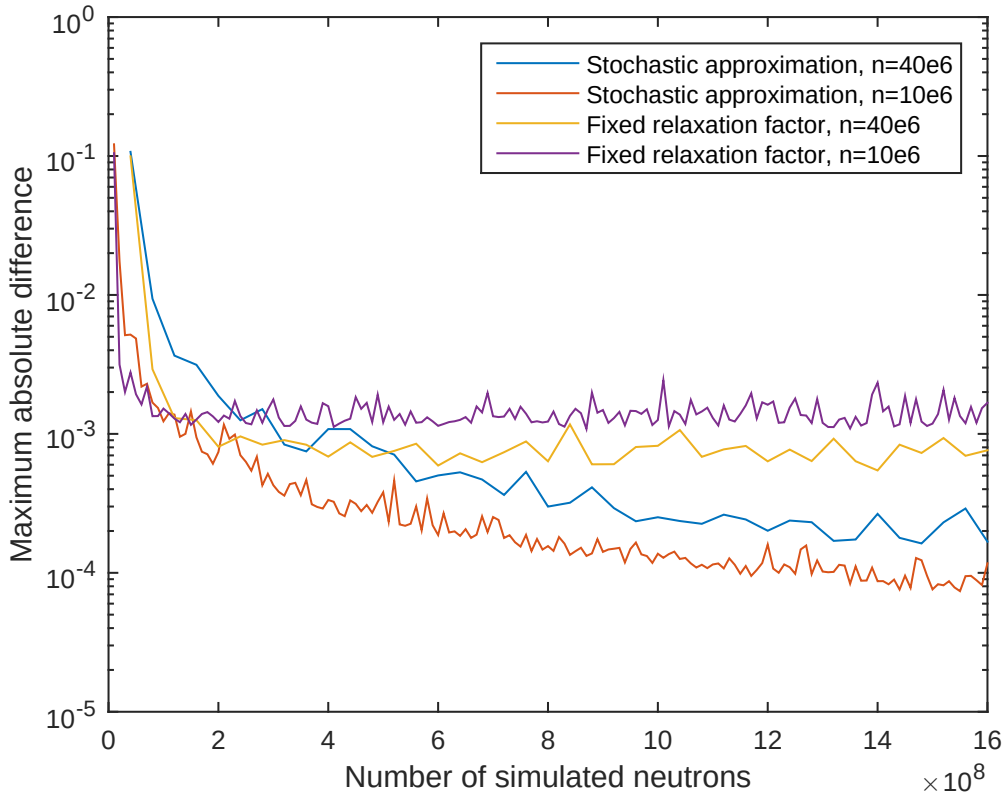


Figure 27: Maximum absolute differences for both tested relaxation schemes with two different number of simulated neutrons per iteration.

In theory there is no guarantee that an iteration process using the stochastic approximation based relaxation scheme will reach the solution of the coupled problem in a finite number of iterations. If the relaxation factor is lowered too quickly, the problem can end up in a situation where the power distribution is far from the final solution and approaches it on a constantly slowing rate due to the relaxation factor which becomes smaller and smaller on each iteration. The simple coupled calculations of this thesis did not show this kind of behaviour.

6 Future work

The coupled calculations of this work rely on many limiting approximations. Probably the most important one is the fact that possible boiling effects are neglected. At least two possible options exist to include boiling and also condensation in future work. The first option is a single-phase simulation with material properties of the water-steam mixture provided by libFluid. Heat transfer on the fuel rod surface is modelled with effective wall heat transfer coefficient evaluated from boiling correlations. This approach assumes local thermal equilibrium and zero slip velocity. The second more advanced and also difficult option is to perform a two-phase simulation were the boiling is modelled explicitly. The two-phase simulation is an on-going work topic at VTT in co-operation with OpenFOAM Foundation.

The assembly modelled in this work is also heavily simplified. Probably the first step to increase the complexity of the geometry would be to add spacer grids. This would increase mixing in the water and make the heat transfer from the cladding to the water more effective. Then of course instead of modelling a small mock-up assembly, a real larger assembly with more fuel pins could be modelled.

Another important aspect is the accurate modelling of the fuel. In the current model only the temperature of the fuel is considered. Several separate fuel behaviour solvers exist so in the future OpenFOAM should only be used to solve the fluid flow to get more reliable results. This complicates the coupling as now three separate codes have to be coupled instead of only two. Also, a new boundary condition would have to be created for OpenFOAM to couple it with the fuel behaviour solver.

When all of the physical phenomena have been taken into account the next step is to do comparisons with experimental data. This includes separate comparisons for the Serpent and OpenFOAM calculations, but also for the coupled calculations. One possibility would be to model a TRIGA reactor which is relatively small and simple. In the future the high fidelity methods used to solve coupled problems are probably used mainly to validate less accurate faster methods until the available computational resources have increased vastly.

7 Conclusions

The main goal of this work was to implement an external coupling between Serpent and OpenFOAM CFD solver. The goal was achieved successfully with a simple coupling program and small modifications to the CFD solver. In addition it was necessary to write a few new thermophysical submodels to model temperature dependent specific heat capacity and thermal conductivity in the fuel and the cladding. Inter-region coupling in the CHT calculation was also improved to allow correct behaviour with temperature dependent material properties. The Serpent-OpenFOAM coupling was tested by modelling a mock-up 5x5 assembly in a steady state with one phase flow. The main results of the coupled calculation were high fidelity temperature and density fields for separate Monte Carlo neutronics simulations.

The effect of temperature and density distribution fidelity on the Monte Carlo results was tested by running a Serpent simulation multiple times with varying level of axial refinement. The temperature distribution was averaged horizontally to either 1, 4, 8, 16 or 256 axial layers. Most obvious differences were observed in the axial power distribution with a notable difference even between the reference and the case with 256 axial layers resulting from horizontal homogenization. The effective multiplication factor was also affected by the axial refinement and its value increased as the number of axial layers decreased. In addition, the shapes of radial capture and fission densities in the fuel had height dependent differences which were only visible with horizontally averaged distributions. Collision density in the coolant was highly dependent on the fidelity of the density distribution. All in all, the use of high fidelity temperature and density distributions produces slightly different results than the less refined alternatives. Here the low fidelity fields were calculated based on the high fidelity fields so no direct comparison to less accurate methods can be made. Still as the required computational time with less accurate methods is considerably lower, the use of high fidelity distributions is not a practical approach to model thermal feedback in all reactor modelling applications.

A few secondary topics related to Serpents multi-physics interface were studied in addition to the analysis of the coupled results. The problem geometry in Serpent can be defined based on OpenFOAM mesh in addition to Serpent's own geometry model. The effect of mesh based geometry was studied by running a coupled calculation with mesh based geometry and a Serpent calculation with identical temperature and density distribution but with different geometry models. Small differences were observed in the final temperature fields of the coupled calculation. These are explained by small differences in the geometries and the stochastic nature of Monte Carlo simulation. A comparison on the results of the Serpent calculation revealed that in the test case the use of mesh based resulted in an increase in mean generation time and neutron lifetime. Transport cycle time was approximately 27 % higher with mesh based geometry.

The effect of search mesh size to memory usage and computational time was tested by running an identical Serpent simulation with varying search mesh size. Based

on the results both of these quantities are highly dependent on the search mesh size. In general, the computational time decreases and memory usage increases with increasing search mesh size. The search mesh size should be optimized separately for each calculation as the definition of the search mesh requires a lot of input from the user as no automatic optimization process exist.

The final real topic was the convergence of the coupled calculation which was briefly discussed. The evaluation of convergence is non-trivial due to the stochastic nature of the Monte Carlo neutronics as there is always some statistical uncertainty in the fission power distribution. In Serpent this problem is tackled with the use of relaxation based on the stochastic approximation scheme. In the coupled calculations of this Master's thesis the performance of the relaxation scheme was excellent. For the sake of comparison, an additional coupled calculation was run with a more traditional fixed under-relaxation factor. In this case the convergence of the solution was limited by the uncertainty of the Monte Carlo simulation.

Finally, the limitations of this work and possible ideas for future development were considered. These include for example the modelling of boiling and solving fuel behaviour with a separate code.

References

[1] C. Watta, T. Schulenburg, and X. Cheng. Coupling of MCNP With a Sub-Channel Code For Analysis of a HPLWR Fuel Assembly. FZKA7233, FZK, Karlsruhe Germany.

[2] C. Tippayakul, M. Avramova, F. Puente Espel, and K. Ivanov. Investigations on Monte Carlo based coupled core calculations. In: Proceedings of ICAPP 2007, Nice, France, May 13–18, 2007, Paper 7134.

[3] F. Puente-Espel, M. Avramova, K. Ivanov, and S. Misu. High accuracy modeling for advanced nuclear reactor core designs using Monte Carlo based coupled calculations. Invited paper for M & C 2009, Saratoga Springs, New York, May 3–7, 2009.

[4] F. Puente-Espel, M. Avramova, and K. Ivanov. New developments of the MCNP/CTF/NEM/NJOY Code system-Monte Carlo based coupled code for high accuracy modeling. In: Proceedings of Advances in Reactor Physics to Power the Nuclear Renaissance (PHYSOR), Pittsburgh, Pennsylvania, USA, May 9–14, 2010.

[5] A. Ivanov, V. Sanchez, R. Stieglitz, and K. Ivanov. Internal multi-scale multi-physics coupled system for high fidelity simulation of light water reactors. *Annals of Nuclear Energy*, 66:104–112, 2014.

[6] M. Daeubler, J. Jimenez, and V. Sanchez. Development of a high-fidelity Monte Carlo thermal-hydraulics coupled code system Serpent/SUBCHANFLOW–first results. In *Proceedings of PHYSOR 2014 Conference*, 2014.

[7] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, second edition, 2007.

[8] F. R. Menter and T. Esch. Elements of Industrial Heat Transfer Prediction. 16th Brazilian Congress of Mechanical Engineering (COBEM), 2001.

[9] F. R. Menter, M. Kuntz, and R. Langtry. Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer*, 4(1):625–632, 2003.

[10] A. Hellsten. Some Improvements in Menter's k-omega-SST turbulence model. 29th AIAA Fluid Dynamics Conference, AIAA-98-2554, 1998.

[11] A. Dorfman and Z. Renner. Conjugate problems in convective heat transfer: Review. *Mathematical Problems in Engineering*, 2009.

[12] J. H. Ferziger and M. Perić. *Computational methods for fluid dynamics*, volume 3. Springer Berlin, 2002.

[13] J. Leppänen. Development of a new Monte Carlo reactor physics code. D.Sc. Thesis, Helsinki University of Technology, 2007.

[14] OpenFOAM Foundation. OpenFOAM. `http://www.openfoam.org`. (16.3.2015).

[15] J. Leppänen, T. Viitanen, and V. V. Multi-physics coupling scheme in the Serpent 2 Monte Carlo code. *Trans. Am. Nucl. Soc.*, 107:1165–1168, 2012.

[16] T. Viitanen and J. Leppänen. Target Motion Sampling Temperature Treatment Technique with Elevated Basis Cross-Section Temperatures. *Nuclear Science and Engineering*, 177(1):77–89, 2014.

[17] J. Leppänen, T. Viitanen, V. V., and M. Aufiero. Unstructured Mesh Based Multi-Physics Interface for CFD Code Coupling in the Serpent 2 Monte Carlo Code. In proc. PHYSOR 2014, Kyoto, Japan, Sep. 28 - Oct. 3, 2014.

[18] J. Leppänen and M. Aufiero. Development of an Unstructured Mesh Based Geometry Model in the Serpent 2 Monte Carlo Code. In proc. PHYSOR 2014, Kyoto, Japan, Sep. 28 - Oct. 3, 2014.

[19] W. G. Luscher and K. J. Geelhood. Material property correlations: Comparisons between FRAPCON-3.4, FRAPTRAN 1.4, and MATPRO. Technical Report NUREG-CR-7024, Pacific Northwest National Laboratory, 2011.

[20] K. Ivanov, M. Avramova, S. Kamerow, I. Kodeli, E. Sartori, E. Ivanov, and O. Cabellos. Benchmarks for Uncertainty Analysis in Modelling (UAM) for the Design, Operation and Safety Analysis of LWRs-Volume I: Specification and Support Data for Neutronics Cases (Phase I). Technical report, Organisation for Economic Co-Operation and Development, Nuclear Energy Agency-OECD/NEA, Le Seine Saint-Germain, 12 boulevard des Iles, F-92130 Issy-les-Moulineaux (France), 2013.

[21] T. Blyth, M. Avramova, K. Ivanov, E. Royer, E. Sartoni, O. Cabellos, H. Feroukhi, and E. Ivanov. Benchmarks for Uncertainty Analysis in Modelling (UAM) for the Design, Operation and Safety Analysis of LWRs-Volume II: Specification and Support Data for the Core Cases (Phase II). Technical report, Organisation for Economic Co-Operation and Development, Nuclear Energy Agency-OECD/NEA, Le Seine Saint-Germain, 12 boulevard des Iles, F-92130 Issy-les-Moulineaux (France), 2013.

[22] D. B. Spalding. A single formula for the "law of the wall". *Journal of Applied Mechanics*, 28(3):455–458, 1961.

[23] C. Jayatilleke. The influence of Prandtl number and surface roughness on the resistance of the laminar sublayer to momentum and heat transfer. *Prog. Heat Mass Transfer*, 1:193–321, 1969.

[24] J. Dufek and W. Gudowski. Stochastic approximation for Monte Carlo calculation of steady-state conditions in thermal reactors. *Nuclear science and engineering*, 152(3):274–283, 2006.