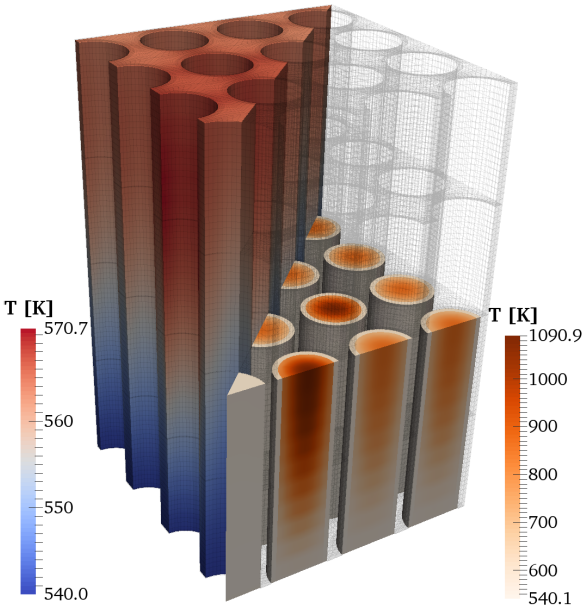
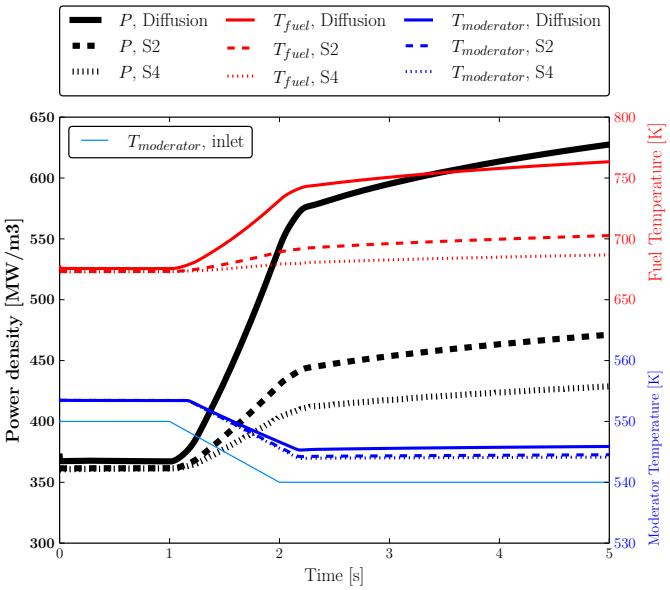
CHALMERS
UNIVERSITY OF TECHNOLOGY





Development of a Transient Multiphysics Solver for Nuclear Fuel Assemblies within a CFD Framework

Master's thesis in Nuclear Engineering

RASMUS ANDERSSON

Department of Applied Physics
Division of Nuclear Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Development of a Transient Multiphysics Solver for Nuclear Fuel Assemblies within a CFD Framework

Master's thesis in Nuclear Engineering

Department of Applied Physics
Division of Nuclear Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

CTH-NT-313

Author:
Rasmus Andersson

Supervisor:
Klas Jareteg
Examiner:
Christophe Demazière

Development of a Transient Multiphysics Solver for Nuclear Fuel Assemblies within a CFD Framework

Author: Rasmus Andersson
Thesis for the degree of Master of Science in Engineering

CTH-NT-313
ISSN 1653-4662
Division of Nuclear Engineering
Department of Applied Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Gothenburg, Sweden 2015

Cover: *Left*: Temperature profiles in the moderator and the fuel in a 7x7 fuel pin diffusion simulation. *Right*: Comparison of the transient behaviors of the implemented diffusion, S_2 and S_4 solvers caused by lowering the moderator temperature at the inlet.

Abstract

The aim of this thesis is to develop and implement a code for solving the coupled multiphysics of PWR fuel assemblies at transient conditions using a computational fluid dynamics (CFD) methodology.

The coupled neutronic and thermal-hydraulic problem of nuclear reactors is typically not resolved at all scales in current reactor codes. Due to the complexity of the problem only a coarse coupling between the separate fields of physics is typically resolved. With advances in affordable computer power it has become increasingly feasible to solve the coupled problem with full resolution on the fuel pin cell scale. While not yet practical for full core simulations, such calculations can be applied to fuel assemblies in high performance computing (HPC) environments.

This work extends the capabilities of an existing code framework, developed within the project FIRE at Chalmers University of Technology. The FIRE code is based on the open source C++ library OpenFOAM, a continuum mechanics code built on finite volume discretization. While the focus of FIRE has previously been on steady-state solvers, this thesis is focused on solution of transients.

The main application couples a neutronic module with a thermal-hydraulic module, solving the coupled problem by iterating between the modules and advancing the time at joint convergence. The length of time steps is controlled throughout the simulation by an adaptive algorithm.

Two neutronic solvers have been implemented; one based on the diffusion approximation and the other solving the neutron transport equation using the discrete ordinates (S_N) methodology.

The thermal-hydraulic module is based on a Reynolds averaged Navier-Stokes (RANS) methodology solving the single-phase conservation equations for mass, momentum and enthalpy. Turbulence is modeled using the standard k-ε model. The pressure-velocity coupling is based on the PISO algorithm. In addition to the equations for fluid transport the conjugate heat transfer from the fuel to the moderator is solved.

The coupled solver has been applied to a set of inlet temperature transients to demonstrate its multiphysics capabilities. The code is shown to capture both fields of physics as well as their interdependence.

Keywords: *Multiphysics, neutronics, thermal-hydraulics, CFD, deterministic reactor modeling, conjugate heat transfer, diffusion, discrete ordinates method, OpenFOAM*

Table of Contents

Acknowledgements

I would like to sincerely thank my supervisor, Klas Jareteg, for providing an inordinate amount of technical support during this project. Without his expertise and guidance in navigating the complex world of computational multiphysics this would probably have been a very thin thesis.

Thanks also go to my examiner, Christophe Demazière, who along with Klas has been a great source of inspiration and knowledge during the course of my master's education, helping me find my passion for the modeling and simulation of nuclear reactors.

Thanks to all the friends I have made during my years at Chalmers, you have enriched my world tremendously. I hope I will manage to keep most of you in my life also after this chapter of our lives is over.

I want to thank my parents for always believing in me, even when I have doubted myself.

Last, but certainly not least, I would like to thank Emily for her love and moral support, and not least for graciously doing all the cooking, cleaning and dishwashing in the past week-and-a-half while I sat night and day writing this thesis. I promise I will make it up to you.

Abbreviations

CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
CUT	Chalmers University of Technology
FEM	Finite Element Method
FVM	Finite Volume Method
HPC	Hight Performance Computing
JFNK	Jacobian-Free Newton Krylov
LWR	Light Water Reactor
NPP	Nuclear Power Plant
PDE	Partial Differential Equation
PISO	Presssure coupled Implicitly with Splitting of Operators
PWR	Pressurized Water Reactor
RANS	Reynolds Averaged Navier-Stokes
S_N	discrete ordinates method

Nomenclature

β	Fraction of delayed neutrons
β_i	Fraction of delayed neutrons produced from precursor group i
β_m	Moderator thermal expansion coefficient
χ^d	Delayed neutron spectrum
χ^p	Prompt neutron spectrum
ε	Dissipation of turbulent kinetic energy
μ	Moderator viscosity
μ_t	Turbulent viscosity
ν	Average number of neutrons created in a fission reaction
$\mathbf{\Omega}$	Neutron traveling direction
ω	Solid angle
Φ	Scalar neutron flux
Ψ	Angular neutron flux
ρ	Density
ϱ	Reactivity
Σ_f	Fission cross-section
Σ_s	Scattering cross-section
Σ_T	Total cross-section
τ_{ij}	Moderator deviatoric stress tensor
C_i	Concentration of neutron precursor group i
c_P	Specific moderator heat capacity at constant pressure
D	Neutron diffusion coefficient
e	Specific energy in moderator
E	Neutron energy
f (f_i)	Generic scalar (vector) quantity
g_i	Gravitational acceleration
$\boldsymbol{J}$	Neutron current density
K_f	Fuel heat transfer coefficient

K_m	Moderator heat transfer coefficient
k	Turbulent kinetic energy
k_{eff}	Effective neutron multiplication factor
P	Moderator mean flow pressure (slowly changing)
p	Moderator instantaneous pressure
$\tilde{p}$	Moderator pressure fluctuation
q''_j	Heat flux vector
q'''_f	Fuel volumetric heat source
q'''_m	Moderator volumetric heat source
$\boldsymbol{r}$	Position vector
T	Temperature
t	Time
U_i ($\boldsymbol{U}$)	Moderator mean flow velocity (slowly changing)
u_i	Moderator instantaneous fluid velocity
$\tilde{u}_i$	Moderator velocity fluctuation
v	Neutron speed

Subscripts

g	Energy groups
i	Neutron precursor groups
m	Discrete ordinates
n	Time steps

Chapter 1

Introduction

1.1 General Background

Modeling and simulation of the physics in nuclear reactors is of great importance to ensure safe and economical operation of NPPs. It is also a very challenging task, due to the strong couplings between multiple scales and multiple branches of physics [1].

The difficulty involved in modeling the physical behavior of nuclear reactors arises because of their multiscale and multiphysics nature. Fields of physics relevant to the reactor core include nuclear reaction kinetics, radioactive decay, neutron transport, fluid dynamics, heat transfer by conduction, radiation and convection, thermodynamic state relations and structural mechanics [1]. However, for some applications, e.g. transient modeling for safety calculations, only time scales from a fraction of a microsecond up to minutes are relevant. On these time scales the nuclear physics can be treated using statistical physics while structural mechanics and some variations of isotopic composition are largely irrelevant. In such circumstances the fields of physics that remain to be solved can be divided into neutron transport (known as neutronics or neutron kinetics), fluid dynamics and heat transport (the two latter are known as thermal-hydraulics in nuclear engineering) [2]. Due to the statistical physics treatment the geometry can then be considered homogeneous on scales smaller than millimeters so that only the spatial scales from fuel pellets to the global core scale need to be considered. For a more detailed treatment of the multiphysics and multiscale aspects of nuclear reactors, see [2].

Depending on the problem to be addressed, reactor modeling has typically been divided into different subproblems which have traditionally been solved using different codes. Notable examples of such problems and codes are `FRAPCON` [3] for fuel performance, `SIMULATE-5` [4] for core neutronics, `CASMO-5` [5] for lattice neutronics and `TRACE` [6] for thermal-hydraulics. For an overview of current practices w.r.t. the coupling between neutronics and thermal-hydraulics, see [2].

1.2 Literature Study

In recent years there has been a trend towards integrating the physics more closely and modeling smaller scales in order to approach a high-fidelity modeling based to a greater extent on first principles [7]. A number of such projects are currently underway:

Within the Consortium for Advanced Simulation of LWRs (CASL) [8], an innovation hub of the US DOE, a finely coupled neutronic/thermal-hydraulic solver has been implemented using the commercial CFD code `STAR-CCM+` [9] externally coupled to a fine-mesh neutronic code implemented within CASL [10], known as `DeCART`. `DeCART` builds on a 2D method of characteristics (MOC) model

in the radial direction coupled to a 1D coarse mesh finite difference (CMFD) model in the axial direction. While this approach allows an unusually high resolution for full core calculations it is limited in the sense that it is not a fully 3D approach.

Ivanov et. al, in a collaboration between the Karlsruhe Institute of Technology and the Pennsylvania State University, have developed and applied a method of dynamically coupling the Monte Carlo code MCNP [11] with the thermal-hydraulic code SUBCHANFLOW [12] in order to achieve higher fidelity calculations. While this is high-fidelity compared to current industry practices it still has a lower resolution than sought in the present work.

In Europe, the NURESAFE project has implemented a platform for integrating existing codes in order to perform multiphysics and multiscale calculations [13].

A commonality between the approaches outlined above is that they build on the external coupling of different codes. This limits the manner in which the modules for different physics can be coupled, due to inefficient data transfer between the codes.

The Multiphysics Object Oriented Simulation Environment (MOOSE) code developed at Idaho National Laboratory [14] is a Finite Element Method (FEM) based unified framework for treating the mathematical solution of different types of closely coupled physics problems, using a Jacobian Free Newton Krylov (JFNK) approach to solve all relevant equations simultaneously. The framework is designed to support efficient parallelization of problems. This approach avoids the limitations of those previously described. Presented work using the MOOSE framework has included neutronics and material modeling among other fields of application [15].

At Chalmers University of Technology (CUT) there is a project in progress, known as FIRE, with the purpose of implementing a unified fine-mesh multiphysics framework to deal with the fully coupled problem of neutronics and thermal-hydraulics efficiently. For larger problems using high performance computing (HPC), problems are parallellized w.r.t. spatial domain decomposition rather than according to the physics, as is the case when separate codes are used. In contrast to MOOSE, the code developed at CUT is based on the finite volume method (FVM) and the nonlinear problem is solved iteratively by operator splitting in place of the JFNK method.

Within FIRE, a number of different solver modules have so far been implemented dealing with the steady-state coupled multiphysics problem [16, 17, 18, 19]. These modules include diffusion and discrete ordinates method (S_N) solvers for the neutronics and different variants of one- and two-phase thermal-hydraulic solvers.

1.3 Purpose

The purpose of this master's thesis project is to extend the FIRE framework to dealing with transient scenarios as well as steady-state ones.

A transient solver is needed to address questions concerning the dynamics of the physical systems. While the steady-state solvers within FIRE have been able to resolve the equilibrium effects of the finely coupled physics in fuel assemblies they cannot recover any information on how the system responds over time to perturbations.

While many of the existing neutronic and thermal-hydraulic solvers can be coupled to solve transient problems, e.g. SIMULATE-5 [4] and TRACE [6], they do not resolve the fully coupled physics with the required resolution to answer questions about how the spatial distributions of variables change over time within fuel pins or within sub-channels.

This work aims to implement and test a transient multiphysics solver to be applied to systems of this resolution to address questions of how the fine spatial profiles behave over time scales on the order of seconds.

1.4 Outline of the Methodology

The main tool used in this thesis project, as well as in the FIRE project as a whole, is the open source C++ library foam-extend-3.1, which has its root in the OpenFOAM code [20]. OpenFOAM is an object oriented framework for implementing and solving continuous field equations using an FVM methodology. It was primarily written for the purpose of CFD calculations.

However, the code is very generally formulated using a template based implementation where differential equations are implemented in a syntax familiar from tensor formalism. It is therefore possible to extend the capabilities of OpenFOAM with relative ease, e.g. to treat additional fields of physics.

The meshes used for the transient calculations are generated by Python meshing scripts developed within FIRE.

Macroscopic few-group cross-sections are pre-generated by the continuous energy Monte Carlo code Serpent [21]. Serpent uses input data from evaluated nuclear libraries and gives macroscopic cross-sections and other material parameters as output.

Initial guesses for all fields at the start of a transient simulation are provided from steady-state simulations, using the previously implemented solvers within FIRE.

The transient problem is then solved by iterating between the neutronic and thermal-hydraulic modules. Each module converges individually before leaving control of the execution to the other. The iteration continues until the coupled problem has converged before moving on to the next time step.

The time step length is decided in an adaptive manner, where both the neutronic and the thermal-hydraulic solvers propose time steps whereafter the shorter of the two is selected. The thermal-hydraulic time step is selected to follow the CFL criterion [22], which dictates that no parsel of fluid is to pass through more than the length of a cell in the main flow direction per time step. The neutronic time step is proposed on the basis that the thermal power produced in any cell should not change more than a given relative amount between consecutive time steps.

The general purpose visualization program ParaView [23] has been used to dynamically visualize the results of calculations during the development of the code.

A number of Python scripts have been implemented as needed for automation of simulations, data analysis and visualization of results.

1.5 Outline of the Thesis

Chapter 2 introduces all equations that are solved in the code. Starting from first principles, each section then moves on to introduce necessary approximations and closure relations.

Next, Chapter 3 describes how the equations are discretized in time and on computational grids and how the differential operators are discretized using these grids. It also treats the general formalism for solving systems of equations using linear algebra and introduces the difference between *explicit* and *implicit* terms.

Chapter 4 goes into greater details on the implementation of different parts of the code, presenting the algorithms used to solve the neutronics and thermal-hydraulics and then details the coupled solution scheme. This chapter also presents the different tools used within the thesis and the roles they play in the project.

The results of simulations, along with discretization studies, comparison between the different implemented solvers and an assessment of the convergence behavior of the solvers are presented in Chapter 5.

Finally, in Chapter 6 some general conclusions are reached and further work is proposed.

Chapter 2

Governing Equations

This chapter details all equations that govern the behavior of the coupled neutronic and thermal-hydraulic system. Starting from first principles, assumptions and approximations are successively introduced in order to build models that can be implemented in the code and solved within the existing limitations of computing power.

2.1 Neutronics

The task of the neutronic module is to solve the problem of neutron transport in the core. The exact equation governing this transport is known as the *Boltzmann transport equation*. When the time dependent Boltzmann equation is solved it also needs to be coupled to an equation for the balance of *delayed neutron precursors*, i.e. radioactive fission fragments with decay chains that release neutrons, known as *delayed neutrons*.

Though the Boltzmann equation is exact it is very difficult to solve it and therefore two approximations are introduced: The *discrete ordinates (S_N) method* which solves the equation for a finite number of directions, and the *diffusion equation* which does not resolve the different neutron directions.

2.1.1 Boltzmann Transport Equation

Equation (2.1) is the integro-differential Boltzmann transport equation for the angular neutron flux in its most general form. It is not completely self-contained but needs to be coupled to Equation (2.2), which keeps track of the concentrations of precursors for delayed neutrons [24].

$$\begin{aligned}
&\frac{1}{v(E)}\frac{\partial\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega})}{\partial t} + \boldsymbol{\Omega}\cdot\nabla\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) \;= \\
&\quad -\Sigma_T(\boldsymbol{r},t,E)\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) + \oint_{4\pi}\int_0^\infty d\omega' dE' \Sigma_s(\boldsymbol{r},t,E'\to E,\boldsymbol{\Omega}'\to\boldsymbol{\Omega})\Psi(\boldsymbol{r},t,E',\boldsymbol{\Omega}') \\
&\quad + (1-\beta)\frac{\chi^p(E)}{4\pi}\int_0^\infty dE'\nu(\boldsymbol{r},t,E')\Sigma_f(\boldsymbol{r},t,E')\Phi(\boldsymbol{r},t,E') + \frac{\chi^d(E)}{4\pi}\sum_i \lambda_i C_i(\boldsymbol{r},t)
\end{aligned} \tag{2.1}$$

$$\frac{dC_i(\boldsymbol{r},t)}{dt} = \beta_i \int_0^\infty dE' \nu(\boldsymbol{r},t,E')\Sigma_f(\boldsymbol{r},t,E')\Phi(\boldsymbol{r},t,E') - \lambda_i C_i(\boldsymbol{r},t) \tag{2.2}$$

The physical meaning of each term in Equation (2.1) is as follows:

$$\frac{1}{v(E)}\frac{\partial\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega})}{\partial t}$$

This is the quantity for which the balance equation is formulated. v is the speed of neutrons with energy E, Ψ is the angular flux of neutrons of energy E in the direction $\boldsymbol{\Omega}$, defined as $\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) = n(\boldsymbol{r},t,E,\boldsymbol{\Omega})v(E)$, where n is the concentration of neutrons. The term can be read out as the rate of change in the number of neutrons per unit volume with kinetic energy E, traveling in the direction given by the directional unit vector $\boldsymbol{\Omega}$ at point $\boldsymbol{r}$ in space at time t.

$$\boldsymbol{\Omega}\cdot\nabla\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega})$$

This is a streaming term, that describes transport of neutrons of energy E in direction $\boldsymbol{\Omega}$. The constant of proportionality is simply unity because this term is defined as streaming in the absence of nuclear reactions and the velocity is already included in the definition of the flux.

$$-\Sigma_T(\boldsymbol{r},t,E)\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega})$$

The above term represents the sum of all sinks of neutrons with energy E in direction $\boldsymbol{\Omega}$. The cross-section Σ_T is sometimes split up into three parts, $\Sigma_T = \Sigma_c + \Sigma_f + \Sigma_s$, where the indices denote capture, fission and scattering, respectively. Note that this term does not describe spatial leakage, but only the disappearance of neutrons due to nuclear reactions (leakage through the system boundaries is accounted for by the streaming term and the boundary conditions).

The terms treated so far have described the rate of change of the neutron concentration, the spatial streaming of neutrons and the sum of all neutron sinks. The remaining terms all describe sources of neutrons with energy E flowing in the direction of $\boldsymbol{\Omega}$. For mainly physical reasons, the sources are typically split into terms due to scattering, fission and radioactive decay.

$$\oint_{4\pi}\int_0^\infty d\omega' dE' \Sigma_s(\boldsymbol{r},t,E'\to E,\boldsymbol{\Omega}'\to\boldsymbol{\Omega})\Psi(\boldsymbol{r},t,E',\boldsymbol{\Omega}')$$

This term is the source term from scattering, integrated over all source energies E' and source directions $\boldsymbol{\Omega}'$. $\Sigma_s(\boldsymbol{r},t,E'\to E,\boldsymbol{\Omega}'\to\boldsymbol{\Omega})$ is called the *differential scattering cross-section* for neutrons with energy E' and direction $\boldsymbol{\Omega}'$ to be scattered to energy E and direction $\boldsymbol{\Omega}$. The term can be read out as the contribution to the concentration of neutrons at energy E streaming in the direction $\boldsymbol{\Omega}$ from all other energies and directions due to scattering against atomic nuclei. The neutrons within this term are thus not new to the neutron population but merely migrate from one velocity vector to another. Note that the neutrons from this source are also accounted for within the sink term in their original energy and direction.

$$(1-\beta)\frac{\chi^p(E)}{4\pi}\int_0^\infty dE' \nu(\boldsymbol{r},t,E')\Sigma_f(\boldsymbol{r},t,E')\Phi(\boldsymbol{r},t,E')$$

This term describes the rate of prompt neutron production through fission. Since not all of the neutrons in the population are spawned directly through fission, but a fraction of them in a

delayed manner through decay of various fission products, it is necessary to distinguish between these contributions. β is the fraction of delayed neutrons so that $1-\beta$ is the fraction of prompt neutrons. χ^p is the energy spectrum of promptly produced fission neutrons. It is normalized to unity so that the value at energy E gives the probability density per neutron of being spawned at that particular energy. ν is the mean number of neutrons produced in each fission event and depends on the isotopic composition of the fuel. Σ_f is the macroscopic fission cross-section and Φ is the scalar neutron flux, defined in terms of the angular flux as:

$$\Phi(\boldsymbol{r},t,E) = \oint_{4\pi} d\omega \Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) \tag{2.3}$$

The scalar flux is used in this case instead of the angular flux because neutron emission by fission is an isotropic process in the laboratory reference system. This is also the origin of the $1/4\pi$ factor, since the neutrons produced by fission are spread out angularly over the whole sphere.

$$\frac{\chi^d(E)}{4\pi}\sum_i \lambda_i C_i(\boldsymbol{r},t)$$

The final term in Equation (2.1) represents the delayed neutron source, which has a different spectrum χ^d, compared to the prompt neutrons. Like fission, release of neutrons from radioactive decay is an isotropic process in the laboratory reference system, so this term also has to be divided with 4π. The sum goes in principle over all species of delayed neutron precursors. However, in practice there are too many to track and they are instead usually grouped together based on their decay times. The decay of these with the resulting release of neutrons occurs with intensities proportional to their respective decay constants λ_i and their concentrations, C_i.

The coupling from Equation (2.2) to Equation (2.1) has its origin in this last term. Equation (2.2) is a system of conservation equations for the concentrations C_i. It is manifest in Equation (2.2) that it is in turn coupled to Equation (2.1) through the fission source of precursors, while the sink of precursors is simply due to the kinetics of radioactive decay. It is worth mentioning that compared to the Boltzmann equation, the precursor equation is relatively easy to solve since the different precursor groups are independent of one another.

2.1.2 Diffusion Approximation

To reduce Equations (2.1)-(2.2) to the diffusion approximation, all dependencies on direction are integrated over the full sphere to instead obtain a conservation equation for the scalar neutron flux [24].

The diffusion equation for neutron kinetics is given below, Equation (2.4). The auxiliary precursor equation is Equation (2.5), here written for I different precursor groups.

$$\begin{aligned}
&\frac{1}{v(E)}\frac{\partial\Phi(\boldsymbol{r},t,E)}{\partial t} - \nabla\cdot D(\boldsymbol{r},t,E)\nabla\Phi(\boldsymbol{r},t,E) \;= \\
&-\Sigma_T(\boldsymbol{r},t,E)\Phi(\boldsymbol{r},t,E) + \int_0^\infty dE'\Sigma_{s0}(\boldsymbol{r},t,E'\to E)\Phi(\boldsymbol{r},t,E') \\
&+(1-\beta)\chi^p(E)\int_0^\infty dE'\nu(\boldsymbol{r},t,E')\Sigma_f(\boldsymbol{r},t,E')\Phi(\boldsymbol{r},E') + \chi^d(E)\sum_{i=1}^{I}\lambda_i C_i(\boldsymbol{r},t)
\end{aligned} \tag{2.4}$$

$$\frac{dC_i(\boldsymbol{r},t)}{dt} = \beta_i \int_0^\infty dE' \nu(\boldsymbol{r},t,E')\Sigma_f(\boldsymbol{r},t,E')\Phi(\boldsymbol{r},t,E') - \lambda_i C_i(\boldsymbol{r},t), \qquad i = 1,...,I \quad (2.5)$$

There is a subtle difference in how the scattering term is treated in the diffusion equation compared to the transport equation. Note that the differential scattering cross-section $\Sigma_s(\boldsymbol{r},t,E' \rightarrow E, \boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega})$ has been replaced by $\Sigma_{s0}(\boldsymbol{r},t,E' \rightarrow E)$, which is still differential w.r.t. energy, but from which the angular dependence is no longer resolved. The derivation of this isotropic form of the scattering cross-section is reviewed in Section 2.1.4.

The most notable difference between Equations (2.1)-(2.2) and (2.4)-(2.5) is contained in the second term on the left-hand-side of Equation (2.4). The streaming term in Eq. (2.1) can be recast as:

$$\boldsymbol{\Omega} \cdot \nabla\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) = \nabla \cdot \boldsymbol{\Omega}\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) \quad (2.6)$$

since $\nabla \cdot \boldsymbol{\Omega}\Psi = \boldsymbol{\Omega} \cdot \nabla\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) + \Psi\nabla \cdot \boldsymbol{\Omega}$ where the second term vanishes because constant vectors are divergence free. Integrating the right-hand-side of Eq. (2.6) over all solid angles gives [25]:

$$\oint_{4\pi} d\omega \nabla \cdot \boldsymbol{\Omega}\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) = \nabla \cdot \oint_{4\pi} d\omega \boldsymbol{\Omega}\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) = \nabla \cdot \boldsymbol{J}(\boldsymbol{r},t,E) \quad (2.7)$$

where $\boldsymbol{J}$ is the neutron current density. In the diffusion formulation the current density needs to be approximated since neutron directions are not resolved. To this end, Fick's law, which assumes that net currents always stream in the direction of the negative flux gradient, Eq. (2.8), is used [25]:

$$\boldsymbol{J}(\boldsymbol{r},t,E) = -D(\boldsymbol{r},t,E)\nabla\Phi(\boldsymbol{r},t,E) \quad (2.8)$$

where the diffusion coefficient, D is given by [26]:

$$D(\boldsymbol{r},t,E) = \frac{1}{3\Sigma_{T,tr}} \quad (2.9)$$

where $\Sigma_{T,tr}$ is the transport corrected isotropic total cross-section [26]. The form of the streaming term in the diffusion equation consequently becomes:

$$\oint_{4\pi} d\omega \boldsymbol{\Omega} \cdot \nabla\Psi(\boldsymbol{r},t,E,\boldsymbol{\Omega}) \approx -\nabla \cdot D(\boldsymbol{r},t,E)\nabla\Phi(\boldsymbol{r},t,E) \quad (2.10)$$

which conforms to Eq. (2.4).

2.1.3 Energy Discretization - Multi Group Formalism

Up to this point the energy dependence of the neutron kinetic equations has been treated continuously. However, since the energy dependence of nuclear cross-sections is very complex this form of the equation is not analytically solvable. Regardless of whether one is solving a transport equation

or a diffusion equation it is necessary to discretize the energy dependence. Instead of treating a continuum of energies a finite number of energy bins are considered [27].

Rewriting the governing equations in the multi group formalism gives for the Boltzmann equation:

$$\begin{aligned}
&\frac{1}{v_g}\frac{\partial \Psi_g(\boldsymbol{r},t,\boldsymbol{\Omega})}{\partial t} + \boldsymbol{\Omega}\cdot\nabla\Psi_g(\boldsymbol{r},t,\boldsymbol{\Omega}) \;= \\
&\quad -\Sigma_{T,g}(\boldsymbol{r},t,\boldsymbol{\Omega})\Psi_g(\boldsymbol{r},t,\boldsymbol{\Omega}) + \oint_{4\pi} d\omega' \sum_{g'=1}^{G} \Sigma_{s,g'\to g}(\boldsymbol{r},t,\boldsymbol{\Omega}'\to\boldsymbol{\Omega})\Psi_{g'}(\boldsymbol{r},t,\boldsymbol{\Omega}') \\
&\quad + (1-\beta)\frac{\chi_g^p}{4\pi}\sum_{g'=1}^{G}\nu_{g'}(\boldsymbol{r},t)\Sigma_{f,g'}(\boldsymbol{r},t)\Phi_{g'}(\boldsymbol{r},t) + \frac{\chi_g^d}{4\pi}\sum_i \lambda_i C_i(\boldsymbol{r},t), \quad g = 1,..,G
\end{aligned} \tag{2.11}$$

and for the diffusion equation:

$$\begin{aligned}
\frac{1}{v_g}\frac{\partial \Phi_g(\boldsymbol{r},t)}{\partial t} - \nabla\cdot D_g(\boldsymbol{r},t)\nabla\Phi_g(\boldsymbol{r},t) = &-\Sigma_{T,g}(\boldsymbol{r},t)\Phi_g(\boldsymbol{r},t) + \sum_{g'=1}^{G}\Sigma_{s0,g'\to g}(\boldsymbol{r},t)\Phi_{g'}(\boldsymbol{r},t) \\
&+ (1-\beta)\chi_g\sum_{g'=1}^{G}\nu_{g'}(\boldsymbol{r},t)\Sigma_{f,g'}(\boldsymbol{r},t)\Phi_{g'}(\boldsymbol{r},t) \\
&+ \sum_{i=1}^{I}\lambda_i C_i(\boldsymbol{r},t), \qquad\qquad g = 1,...,G
\end{aligned} \tag{2.12}$$

In both cases, the form of the precursor balance equation becomes:

$$\frac{dC_i(\boldsymbol{r},t)}{dt} = \beta_i\sum_{g'=1}^{G}\nu_{g'}(\boldsymbol{r},t)\Sigma_{f,g'}(\boldsymbol{r},t)\Phi_{g'}(\boldsymbol{r},t) - \lambda_i C_i(\boldsymbol{r},t), \qquad i = 1,...,I \tag{2.13}$$

2.1.4 Angular Discretization - Discrete Ordinates Method

In order for an FVM computer code to be able to solve the transport equation, Eq. (2.1), the continuous angular dependence needs to discretized as well. There are many methods for treating the angular variable, but in this work, the discrete ordinates (S_N) method is used, and therefore only this method is detailed here.

The idea of the S_N method is to divide the unit sphere into a finite number of directions (ordinates). Each direction is assigned a weight to be used in the summing of contributions from different directions. The choice of directions and weights can be made in many different ways. The method used within this work, known as the *level symmetric quadrature set* [27], is summarized below:

The unit sphere is first divided into octants (two hemispheres, each divided into four azimuthal sectors). Because of this the number of resolved directions is always a factor of eight. The N in S_N denotes the number of polar zones that the sphere is divided into, so that each hemisphere is split into half that number. In the first polar zone (that nearest the pole of each hemisphere) one ordinate is selected. The second polar zone is split in two azimuthal sections for each octant. The third is split in three and so on down to the equator. This means that the number of directions M as a function of the order N will be

$$M = 8 \times \sum_{q=1}^{N/2} q \tag{2.14}$$

Eq. (2.14) gives rise to the number of directions presented in Table 2.1 depending on the order of the S_N method applied.

Table 2.1: Number of discrete ordinates for the S_N method, as a function of N.

N	2	4	6	8	10	12	14	16
M	8	24	48	80	120	168	224	288

The S_N methodology converts Eq. (2.1) into a number of coupled equations, one for each discrete ordinate. The equations are coupled through the scattering term. However, this term is complex as it couples each individual combination of direction and energy to all others. Therefore, before presenting the S_N equation, this term will be simplified.

Treatment of the Scattering Term

The scattering term in Equation (2.1) is proportional to the differential scattering cross-section, $\Sigma_s(\boldsymbol{r}, t, E' \rightarrow E, \boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega})$, i.e. the macroscopic cross-section for scattering from one specific combination of energy and direction to another. The energy dependence is naturally treated with the multigroup formalism so that the differential cross-section can be written as $\Sigma_{s,g' \rightarrow g}(\boldsymbol{r}, t, \boldsymbol{\Omega}' \rightarrow \boldsymbol{\Omega})$.

Still, keeping track of the differential scattering cross-section in angular space demands data for transitions between each of the resolved directions to all of the others, which amounts to an $M \times M$ tensor for each energy group transition. However, it is usually assumed that the materials in the core are isotropic, in which case only the angle between initial and final directions determine the cross-section [27].

Under the isotropy assumption the scattering cross-sections can be expanded using Legendre polynomials of the cosine of the angle between incoming and outgoing directions, i.e. $P_l = P_l(\mu) = P_l(\cos\theta)$. The differential scattering cross-section can now be written [27]:

$$\Sigma_{s,g' \rightarrow g}(\boldsymbol{r}, t, \mu) = \sum_{0}^{\infty} \frac{2l+1}{4\pi} \Sigma_{sl,g' \rightarrow g}(\boldsymbol{r}, t) P_l(\mu)$$

where $\Sigma_{sl,g' \rightarrow g}$ are the expansion coefficients. The Legendre polynomials are defined as:

$$P_l(\mu) = \frac{1}{2^l l!} \frac{d^l}{d\mu^l} (\mu^2 - 1)^l$$

which gives for the first two orders:

$$P_0(\mu) = 1, \quad P_1(\mu) = \mu$$

In the present work the expansion is truncated after the first two terms so that we get for Σ_s:

$$\Sigma_{s,g' \rightarrow g}(\boldsymbol{r}, t, \mu) \approx \frac{1}{4\pi} \Sigma_{s0,g' \rightarrow g}(\boldsymbol{r}, t) + \frac{3\mu}{4\pi} \Sigma_{s1,g' \rightarrow g}(\boldsymbol{r}, t)$$

S_N Equation

In the S_N equation, the integration over solid angles is transformed into a weighted sum of discrete ordinates. Equation (2.3) details how the integration of the continuous angular flux over 4π gives the scalar flux. We seek now a corresponding equation for the discrete case.

The weights w_m are chosen to sum up to 1, i.e. $\sum w_m = 1$. Since the integral $\int d\omega$ over all solid angles is 4π we need to multiply the weighted sum with this factor.

$$\Phi_g(\boldsymbol{r},t) \;= \oint_{4\pi} d\omega \Psi_g(\boldsymbol{r},t,\boldsymbol{\Omega}) \rightarrow \Phi_{g,m}(\boldsymbol{r}) = 4\pi \sum_{m=1}^{M} w_m \Psi_{g,m}(\boldsymbol{r},t) \tag{2.15}$$

Rewriting Eq. (2.11) with a finite number of directions, each with weight w_m, and using (2.15) gives a system of M equations. Below the mth equation is written, using a Legendre expansion to order L:

$$\begin{aligned}
&\frac{1}{v_g}\frac{\partial \Psi_{g,m}(\boldsymbol{r},t)}{\partial t} + \boldsymbol{\Omega}_m \cdot \nabla \Psi_{g,m}(\boldsymbol{r},t) \\
&\quad = -\Sigma_{T,g}(\boldsymbol{r},t)\Psi_{g,m}(\boldsymbol{r},t) + w_m \sum_{g'=1}^{G}\sum_{m'=1}^{M}\sum_{l=0}^{L}(2l+1)P_l(\boldsymbol{\Omega}_{m'}\cdot\boldsymbol{\Omega}_m)\Sigma_{s,l,g'\rightarrow g}(\boldsymbol{r},t)\Psi_{g',m'}(\boldsymbol{r},t) \\
&\qquad + w_m(1-\beta)\chi_g^p \sum_{g'=1}^{G}\nu_{g'}(\boldsymbol{r},t)\Sigma_{f,g'}(\boldsymbol{r},t)\Phi_{g'}(\boldsymbol{r},t) + \frac{w_m\chi_g^d}{4\pi}\sum_{i=1}^{I}\lambda_i(\boldsymbol{r},t)C_i(\boldsymbol{r},t)
\end{aligned} \tag{2.16}$$

The precursor equation system remains unchanged since both production and disappearance of precursors are isotropic processes.

2.2 Thermal-hydraulics

The thermal-hydraulic module deals with fluid flow, heat transfer through conduction and convection and thermodynamic relations between the state variables (the gas gap is approximated as a solid so that no radiative heat transfer is treated in the code). The fluid transport equations are based on the conservation of mass, momentum and energy. Because of the high prevalence of turbulence in nuclear reactor cores, turbulent quantities must be modeled. Doing so at a reasonable computational cost for systems of fuel assembly size requires several approximations. In the final section of this chapter, the equation for pressure solved within the code is derived by coupling the mass and momentum conservation equations in an approximate manner.

2.2.1 Conservation Equations

Note: The equations in this section are written using index notation with Einstein's summation convention, instead of the vector notation used in previous sections. This conforms to standard practice within fluid dynamics.

The basis for thermal-hydraulic theory lies in the fluid transport of mass, momentum and enthalpy, as well as the transfer of heat from the solid fuel, cladding and gas gap to the fluid moderator.

A general fluid transport equation can be written for a generic transported quantity f (scalar) or f_i (vector) [22]:

$$\frac{\partial \rho f}{\partial t} + \frac{\partial (\rho f u_j)}{\partial x_j} = \theta + \phi_i \tag{2.17}$$

$$\frac{\partial \rho f_i}{\partial t} + \frac{\partial (\rho f_i u_j)}{\partial x_j} = \theta_i + \phi_{ij} \tag{2.18}$$

where ρ is the density of the fluid, u_j is the fluid velocity, θ or θ_i is the total of all volumetric sources of f or f_i and ϕ_i or ϕ_{ij} is the sum of all surface sources. Depending on the nature of f and its transport these terms take on different forms.

In the mass conservation equation f is simply 1, and there are no sources or sinks. This gives the equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_j)}{\partial x_j} = 0 \tag{2.19}$$

Eq. (2.19) is commonly known as the continuity equation [22].

The equation for conservation of momentum is a vector equation, and it also has more terms. f_i is in this case u_i, i.e. the velocity of the fluid. This gives rise to an outer product in the second term in Eq. (2.18) which couples the different flow directions to one another. The sources in this case consist of the forces from gravity, g_i, pressure p and deviatoric stresses, τ_{ij}. Gravity is a volumetric source while stresses are surface sources. This gives the following equation:

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = \frac{\partial \tau_{ji}}{\partial x_j} - \frac{\partial p}{\partial x_i} + \rho g_i \tag{2.20}$$

Water is a Newtonian fluid, i.e. its deviatoric stress tensor τ_{ij} is proportional to the rate of deformation of the fluid with viscosity, μ, as the coefficient of proportionality [22]:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \delta_{ij} \frac{\partial u_i}{\partial x_i} \tag{2.21}$$

The second term in Eq. (2.21) is present in order to suppress normal stresses (which are accounted for in the action of the pressure). Contracting the deviatoric stress term in Eq. (2.20) gives:

$$\begin{aligned} \frac{\partial \tau_{ii}}{\partial x_i} &= \frac{\partial}{\partial x_i} \left[\mu \left(\frac{\partial u_i}{\partial x_i} + \frac{\partial u_i}{\partial x_i} \right) \right] - \frac{\partial}{\partial x_i} \left[\frac{2}{3} \mu \delta_{ii} \frac{\partial u_i}{\partial x_i} \right] \\ &= 2 \frac{\partial}{\partial x_i} \left[\mu \frac{\partial u_i}{\partial x_i} \right] - 2 \frac{\partial}{\partial x_i} \left[\mu \frac{\partial u_i}{\partial x_i} \right] \\ &= 0 \end{aligned} \tag{2.22}$$

since the Kronecker δ_{ij} tensor is summed up to 3 on contraction. Writing Eq. (2.20) using the Newtonian form for the stress tensor gives:

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \delta_{ij} \frac{\partial u_i}{\partial x_i} \right] - \frac{\partial p}{\partial x_i} + \rho g_i \tag{2.23}$$

Equation (2.23) is known as the Navier-Stokes equation for compressible fluids. It should be noted that while the flow treated in this project is clearly subsonic, so that pressure waves need not be considered, the density is still variable in time and space due to the nuclear heating. The governing equations therefore need to be written in their compressible forms, although the solution methodology used is based on pressure-velocity coupling rather than a density formulation.

Because of the heating of the fuel and the moderator there is also a need to write a conservation equation for energy. The balance of *specific energy*, i.e. internal plus kinetic energy per unit mass, depends on the addition of heat through volumetric and surface sources, as well as the work done on the fluid by the various forces acting on it. The specific energy conservation equation is written [28]:

$$\frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho e u_j)}{\partial x_j} = q'''_m - \frac{\partial q''_{m,j}}{\partial x_j} + \frac{\partial(\tau_{ij} u_i)}{\partial x_j} - \frac{\partial(p u_j)}{\partial x_j} + \rho g_j u_j \tag{2.24}$$

where the volumetric heat source (mainly from gamma decay) appears as a simple source term. The second term on the right-hand-side represents the flow of heat across the boundary between fuel and moderator. The negative sign represents the convention of positive flow out of a system (in practice q'' will be negative, so that heat enters the moderator from the fuel). The remaining terms represent work done by deviatoric stresses, pressure and gravity, respectively.

However, because the heat transfer problem is solved not only for convection in the fluid but also for heat transfer in the solid fuel, it is more convenient to formulate the heat transfer problem in terms of temperature. Eq. (2.24) is then restated as [29]:

$$\rho c_p \frac{\partial T}{\partial t} + \rho c_p u_j \frac{\partial T}{\partial x_j} = q'''_m + \frac{\partial}{\partial x_j}\left(K_m \frac{\partial T}{\partial x_j}\right) + \beta_m T \left(\frac{\partial p}{\partial t} + u_j \frac{\partial p}{\partial x_j}\right) \tag{2.25}$$

Eq. (2.25) has a slightly different form compared to the other fluid transport equations due to this reformulation. Here c_p is the specific heat capacity at constant pressure, K_m is the moderator heat conductivity which has been used in Fourier's law [25] to approximate the heat flux $q''_{m,j}$ and β_m is the thermal expansion coefficient of the moderator that moderates the work done by pressure. Work done by viscous stresses and gravity is neglected in Eq. (2.25) (and in the code). At the present state of FIRE, the gamma heat term, q'''_m is also neglected.

Finally, the heat conduction equation in the fuel is given in Eq. (2.26) [30]:

$$\rho c_p \frac{\partial T}{\partial t} = q'''_f + \frac{\partial}{\partial x_j}\left(K_f \frac{\partial T}{\partial x_j}\right) \tag{2.26}$$

where K_f is the heat conductivity of the fuel and q'''_f is the fission heat source, given by [27]:

$$q'''_f = \kappa_f \sum_g \Sigma_{f,g} \Phi_g \tag{2.27}$$

Here κ_f is the mean energy released per fission event, and the sum adds up to the total number of fission events per unit volume and time.

2.2.2 Reynolds Averaged Equations

In the Reynolds Averaged Navier-Stokes (RANS) approach to fluid dynamics, rapid fluctuations in time are filtered out from the equations. To formulate Eqs. (2.19) and (2.23) in their RANS versions, each rapidly fluctuating quantity in the equations is decomposed into a lower frequency part and a higher frequency part [22]. The low frequency, or *mean flow* component is still time dependent, but all frequencies above some threshold are included in the high frequency term. The time scale of the filter is chosen such that all the fluctuating quantities are averaged to zero. Pressure, velocity and enthalpy are treated with Reynolds decomposition, while e.g. density varies negligibly and need not be decomposed [28]. Equations (2.19) and (2.23) can then be written (with capital letters for steady parts and tildes over fluctuating parts):

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left[\rho(U_j + \tilde{u}_j)\right] = 0 \tag{2.28}$$

$$\begin{aligned} &\frac{\partial}{\partial t}[\rho(U_i + \tilde{u}_i)] + \frac{\partial}{\partial x_j}[\rho(U_i + \tilde{u}_i)(U_j + \tilde{u}_j)] \\ &= \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\mu\left(\frac{\partial U_i}{\partial x_i} + \frac{\partial \tilde{u}_i}{\partial x_i}\right)\right] - \frac{\partial}{\partial x_i}[P + \tilde{p}] + \rho g_i \end{aligned} \tag{2.29}$$

If a moving time average is performed on Eqs. (2.28)-(2.29) with the window chosen so that fluctuating parts vanish upon averaging, the resulting RANS equations are:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho U_j)}{\partial x_j} = 0 \tag{2.30}$$

$$\frac{\partial(\rho U_i)}{\partial t} + \frac{\partial}{\partial x_j}[\rho U_i U_j + \rho\overline{\tilde{u}_i\tilde{u}_j}] = \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial U_i}{\partial x_i}\right] - \frac{\partial P}{\partial x_i} + \rho g_i \tag{2.31}$$

It is seen that all fluctuating quantities vanish upon time averaging, except for the second term in the convective term on the left hand side of Eq. (2.31). This means that even though the equations have been averaged in time to filter out high frequency quantities, turbulence effects still need to be modeled.

It is customary to subtract the turbulence term in Eq. (2.31) from both sides so that it appears instead on the right-hand-side of the equation:

$$\frac{\partial(\rho U_i)}{\partial t} + \frac{\partial}{\partial x_j}[\rho(U_i U_j)] = \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial U_i}{\partial x_i}\right) - \rho(\overline{\tilde{u}_i\tilde{u}_j})\right] - \frac{\partial P}{\partial x_i} + \rho g_i \tag{2.32}$$

In Eq. (2.32) the turbulent term, $\rho(\overline{\tilde{u}_i\tilde{u}_j})$, plays the role of an additional stress term. This quantity is therefore known as the *Reynolds stress tensor* [22].

2.2.3 Turbulence Modeling

The Boussinesq Assumption

The purpose of the turbulence model used is to approximate the Reynolds stress tensor so that the fluid transport equations can be solved. In this thesis the standard k-ε turbulence model is used. In this model an additional assumption, the *Boussinesq assumption*, is used to relate the Reynolds stress to the velocity gradients.

The Boussinesq assumption postulates that the total stress tensor can be written in terms of the velocity gradients times an effective viscosity due to both viscous forces and turbulence [31]:

$$\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}-\frac{2}{3}\delta_{ij}\frac{\partial U_i}{\partial x_i}\right)-\rho(\overline{\tilde{u}_i\tilde{u}_j})\right]=\frac{\partial}{\partial x_j}\left[(\mu+\mu_t)\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}-\frac{2}{3}\delta_{ij}\frac{\partial U_i}{\partial x_i}\right)\right] \tag{2.33}$$

This gives for the Reynolds averaged Navier-Stokes equation, Eq. (2.32):

$$\frac{\partial(\rho U_i)}{\partial t}+\frac{\partial}{\partial x_j}[\rho(U_iU_j)]=\frac{\partial}{\partial x_j}\left[(\mu+\mu_t)\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}-\frac{2}{3}\delta_{ij}\frac{\partial U_i}{\partial x_i}\right)\right]-\frac{\partial P}{\partial x_i}+\rho g_i \tag{2.34}$$

If the energy equation is treated in a similar manner as the Navier-Stokes equation (derivation is skipped for brevity) the result is:

$$\rho c_p\frac{\partial T}{\partial t}+\rho c_pU_j\frac{\partial T}{\partial x_j}=q'''_m+\frac{\partial}{\partial x_j}\left(K_{\text{eff}}\frac{\partial T}{\partial x_j}\right)+\beta_mT\left(\frac{\partial p}{\partial t}+U_j\frac{\partial p}{\partial x_j}\right) \tag{2.35}$$

where

$$K_{\text{eff}}=K_m+\frac{\mu_t\rho c_p}{Pr} \tag{2.36}$$

In Eq. (2.36) Pr is the *Prandtl number* which describes the ratio of momentum diffusivity to thermal diffusivity [28]. The addition of a turbulent contribution to the heat conductivity is analogous to the Boussinesq approximation which adds a contribution to the viscosity due to turbulent flow.

The k-ε Model

The Boussinesq assumption introduces a new quantity, the *turbulent viscosity*, μ_t. The task of the turbulence model is to find a profile for μ_t. In the standard k-ε model this is done by expressing μ_t in terms of two transported scalars: the turbulent kinetic energy k and its dissipation, ε, and solving approximated transport equations for these scalars [28].

Dimensional analysis suggests defining the turbulent viscosity on the form:

$$\mu_t=\rho C_\mu\frac{k^2}{\varepsilon} \tag{2.37}$$

The coefficient C_μ in Eq. (2.37) is a dimensionless constant that has to be measured from experiments or finer scale simulations and might vary depending on flow conditions. The standard value

(used in this work) can be found in Table 2.2 along with other empirical parameters used in the standard k-ε model.

The derivations of the transport equations for k and ε are here omitted. Instead of using the exact forms of these equations several approximations are made, which are motivated by physical reasoning. The standard form for the equations are [28]:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho k U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right] + 2\mu_t \frac{\partial U_i}{\partial x_j}\frac{\partial U_i}{\partial x_j} - \rho\varepsilon \tag{2.38}$$

$$\frac{\partial \rho \varepsilon}{\partial x_i} + \frac{\partial \rho \varepsilon U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\frac{\mu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}2\mu_t \frac{\partial U_i}{\partial x_j}\frac{\partial U_i}{\partial x_j} - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \tag{2.39}$$

In Equations (2.38) and (2.39) σ_k, σ_ε, $C_{1\varepsilon}$ and $C_{2\varepsilon}$ are parameters that are not predicted by the underlying theory. They are instead provided by correlations based on a large number of experiments [28]. Their standard values are found in Table 2.2.

Table 2.2: Empirical parameters in the standard k-ε model [28].

σ_k	σ_ε	$C_{1\varepsilon}$	$C_{2\varepsilon}$	C_μ
1 .0	1.30	1.44	1.92	0.09

2.2.4 Thermodynamic State Relations

The number of unknowns in the thermal-hydraulic problem is nine: three velocity components, temperature, pressure, density, turbulent viscosity, turbulent kinetic energy and turbulent dissipation. The number of equations presented above is only eight: Continuity equation (2.30), three Navier-Stokes equations (2.34), temperature equation (2.35)/(2.26), two equations in the k-ε model (2.38), (2.39) and the relation between k, ε and μ_t (2.37). In order to fully close the thermal-hydraulic problem another relation is needed. This final relation is given by the thermodynamic state. Knowing the pressure and temperature allows determining the density of the moderator. In this work the thermodynamic relation is given by a water table using the temperature profile from the solution of Eq. (2.35) and assuming constant pressure.

2.2.5 Derivation of the Pressure Equation

The main thermal-hydraulic variables of interest are pressure, velocity, temperature and density. Density is directly acquired through water table interpolation if the temperature is known. The temperature is given by the conjugate heat transfer equations Eqs. (2.26) and (2.35). The velocity profiles are obtained by solving the momentum equation, Eq. (2.34) and its auxiliary turbulent transport equations, Eqs (2.38) and (2.39). What is left in the equation system is the continuity equation, Eq. (2.30), but it does not include pressure. There is therefore the question of how the pressure profile should be updated.

In the PISO algorithm for incompressible fluids the continuity and Navier-Stokes equations are coupled and preconditioned in order to yield an approximate equation for the pressure [28]. The thermal-hydraulic solver in this work is based on the same principle but adds a correction for the buoyancy caused by inhomogeneous temperature distributions.

In the derivation of the pressure equation, the time derivative of density is neglected in Eq. (2.30) while the spatial derivatives in Eqs. (2.30) and (2.34) and the explicit buoyancy term in Eq. (2.34), ρg_i, is kept. This yields the following equations:

$$\frac{\partial \rho U_i}{\partial x_i} = 0 \tag{2.40}$$

$$\frac{\partial \rho U_i}{\partial t} + \frac{\partial \rho U_i U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right] - \frac{\partial P}{\partial x_i} + \rho g_i \tag{2.41}$$

Eqs. (2.40) and (2.41) are written in a block matrix equation with the velocity and pressure distributions as dependent variables:

$$\begin{pmatrix} A & \frac{\partial}{\partial x_i} \\ \frac{\partial}{\partial x_i}\rho & 0 \end{pmatrix} \begin{pmatrix} U_i \\ P \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\frac{\partial U_j^0}{\partial x_i}\right] + \rho g_i \\ 0 \end{pmatrix} \tag{2.42}$$

where the operator A is given by:

$$A = \frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x_j}\rho U_j^0 - \frac{\partial}{\partial x_j}(\mu + \mu_t)\frac{\partial}{\partial x_j} \tag{2.43}$$

The superscript 0 above U_j denotes that this quantity needs to be treated explicitly in order to linearize the problem. If U_i is solved for in the first row block in Eq. (2.42) it can thereafter be eliminated in the second, giving an equation for the pressure. The first row reads:

$$AU_i + \frac{\partial P}{\partial x_i} = \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\frac{\partial U_j^0}{\partial x_i}\right] + \rho g_i \tag{2.44}$$

It is clear in Eq. (2.44) that U_i can be solved by left-multiplying all terms with A^{-1}. However, while A is a sparse matrix because the spatial derivatives in Eq. (2.42) couple at most a few neighboring cells, its inverse is probably not.

In order to avoid a full matrix inversion the off-diagonal elements of A are instead moved to the right-hand-side of the equation. Eq. (2.44) then gets the form:

$$DU_i + \frac{\partial P}{\partial x_i} = \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\frac{\partial U_j^0}{\partial x_i}\right] + \rho g_i - BU_i^o \tag{2.45}$$

where $A = D + B$ and D is the diagonal of A. U_i^o is latest guess for U_i. Unlike A, D is trivially inverted. Solving for U_i gives:

$$U_i = D^{-1}\frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\frac{\partial U_j}{\partial x_i}\right] + D^{-1}\rho g_i - D^{-1}BU_i^o - D^{-1}\frac{\partial P}{\partial x_i} \tag{2.46}$$

Entering the expression for U_i into the second line of Eq. (2.42) gives:

$$\frac{\partial}{\partial x_i}\left(\rho D^{-1}\frac{\partial}{\partial x_j}\left[(\mu+\mu_t)\frac{\partial U_j^0}{\partial x_i}\right]+D^{-1}\rho g_i - D^{-1}BU_i^o - D^{-1}\frac{\partial P}{\partial x_i}\right)=0 \tag{2.47}$$

Rearranging and sorting terms finally gives:

$$\frac{\partial}{\partial x_i}\left[D^{-1}\frac{\partial P}{\partial x_i}\right]=\frac{\partial}{\partial x_i}\left[\rho D^{-1}\frac{\partial}{\partial x_j}\left((\mu+\mu_t)\frac{\partial U_j^0}{\partial x_j}\right)\right]+\frac{\partial}{\partial x_i}\left[D^{-1}\rho g_i\right]-D^{-1}BU_i^o \tag{2.48}$$

Eq. (2.48) is the pressure equation that is used in the thermalhydraulic solver.

Chapter 3

Discretization of the Problem

To solve the multiphysics problem, all equations need to be discretized. This discretization has several components. To begin with all physical fields need to be represented at a discrete number of points. This is generally achieved by integration of the fields over finite volumes. Furthermore, all differential operators need to be discretized. The result of the discretization is to transform each continuous partial differential equation into a finite (but large) system of coupled algebraic equations.

3.1 Numerical Solution of Discretized Equations

All discretized systems of equations to be solved can be schematically written in the form of the generic matrix equation [22]:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{3.1}$$

where $\boldsymbol{x}$ is a vector of the dependent variables for which the equation is solved, $\boldsymbol{A}$ is a square matrix consisting of coefficients of the dependent variables in the PDE and $\boldsymbol{b}$ is a vector of independent source terms. $\boldsymbol{b}$ and $\boldsymbol{x}$ are vectors, not in the geometrical sense, but in a mere algebraic sense, meaning that though they do not fulfill the requirements of tensors in normal geometric space, matrix algebra is the most efficient way to represent them schematically. In practice, they are typically arrays of values of the dependent variables at the cell centers. Note that A is generally sparse due to differential operators operating only on nearby cells [22].

The solution of Eq. (3.1) is schematically given by inverting the matrix $\boldsymbol{A}$. In practice this is done using various different matrix methods, depending on the form of the equation system.

When solving discretized PDEs on the form (3.1) it will be possible to write some terms as linear combinations of elements of the vector of dependent variables. Such terms can be discretized in *implicit* form, by including their coefficients in the coefficient matrix ($\boldsymbol{A}$). Other terms cannot be treated implicitly, or for some reason it is not suitable to do so. Those terms are included on the right-hand-side, in the source vector $\boldsymbol{b}$, and are known as *explicit* terms.

It is also common for terms to be split into an implicit and an explicit part. In the next few sections the discretization of differential operators will be treated. It will be made clear in each case to what extent the terms are implicit contra explicit.

Special consideration needs to be taken for nonlinear terms which cannot directly be solved by linear algebra. An example is the velocity convection term in the Navier-Stokes equation (2.34):

$\frac{\partial}{\partial x_j}(\rho U_i U_j)$. In this case the solution is to treat U_i implicitly and use the latest available velocity profile to discretize U_j.

When several discretized PDEs are coupled and solved together there are two alternatives for how to approach the solution of the whole system. Either all equations are included in the same matrix system (fully implicit solution) or each PDE is discretized into its own system and the solution of the coupled system is arrived at iteratively. The latter approach is called *operator splitting* and is the route taken in this work.

The advantage of fully implicit methods is that they may converge in fewer iterations, since all information about the linearized problem is included in the same matrix system while split operator matrices contain only part of the available information. The main advantage of the operator splitting method is its simplicity, and the fact that nonlinearities may to some extent be treated within outer iterations of the coupled problem, since the fields solved for are updated before starting the solution of the next matrix equation.

3.2 Spatial Discretization

This section deals with the discretization of the volumes and boundaries of the system on a computational grid. It also treats the discretization of spatial differential operators.

3.2.1 Representation of the Fields

In discretized computational methods, problems are always solved on a spatial mesh, consisting of a discrete number of contiguous (i.e. space-filling but not overlapping) computational cells. In this work all cells are also convex. In the finite volume framework, fields are represented mainly by their cell center values. These can be schematically thought of as the values obtained by volume integrating quantities over each cell. However, in some cases the discretization of differential operators demands that values on the cell faces be used instead. In such cases interpolation between neighboring cell centers is used to provide the face values [22]. In addition to cell center values, each field has specified boundary conditions that are needed to solve the PDE problem. OpenFOAM uses a number of data types to represent discretized fields, depending on their rank. Common for these are that they are represented in memory by a list of cell center values and boundary values [32].

Not only the resolution of the solved fields, but also their accuracy, is limited by the resolution of the grid. This is because of the discretization of the differential operators, to be detailed in the following sections.

3.2.2 Divergence

Divergence operators are present in the convective terms in both the mass, momentum and temperature conservation equations, Eqs. (2.30), (2.34) and (2.35).

The discretization procedure for divergence is exemplified using the simplest of these terms, from the mass conservation equation. Integrating the convective term over the volume of a given cell, using Gauss' theorem and discretizing gives:

$$\begin{aligned} \int_V dV(\nabla \cdot \rho \boldsymbol{U}) &= \int_S d\boldsymbol{S} \cdot \rho \boldsymbol{U} \\ &\rightarrow \sum_f \boldsymbol{S}_f \cdot \boldsymbol{U}_f \rho_f \end{aligned} \tag{3.2}$$

where f indexes the faces of the cell. The face vectors $\boldsymbol{S}_f$ are obtained by multiplying the face normal vector with its area. Both of these quantities are determined from the mesh. The face values of fields, ρ and $\boldsymbol{U}$ in this case, are obtained using some interpolation scheme. For all quantities in this thesis upwind differencing is used to assess the values at the faces.

Upwind differencing means that the face value is taken to be the value at the previous cell w.r.t. the direction of the differentiated vector. These schemes are primarily applicable to hyperbolic flows, where information exclusively travels in the direction of the flow. Upwind differencing is stable if the Courant-Friedrichs-Lewy (CFL) condition is fulfilled [22] (see section 4.6.2).

3.2.3 Gradients

Most of the gradients evaluated in this project are face gradients, i.e. the gradients are evaluated at faces of cells in order to determine the flow of some quantity between cells. These are discretized as follows (taking the temperature gradient as example):

$$\nabla T \rightarrow \frac{T^p - T^n}{d_{n\rightarrow p}} + C_{non-orthogonal} \tag{3.3}$$

In Eq. (3.3) p and n denote the present cell and its neighbor, respectively. The form above assumes that the flow direction is from the neighbor to the present cell, otherwise, the sign is reversed. $C_{non-orthogonal}$ is a correction term that is used if the mesh is not orthogonal. The correction term is always explicit while the finite difference term may be implicit. Cell center gradients are always explicit in OpenFOAM, with their own set of discretization schemes.

Note that there is a gradient term in Eq. (2.16); $\boldsymbol{\Omega}_m \cdot \nabla \Psi_{g,m}(\boldsymbol{r}, t)$. However, since it would generally be advantageous to keep as many terms as possible implicit unless they negatively affect numerical stability [22] this term is rewritten as a divergence term, noting that

$$\begin{aligned} \nabla \cdot (\Psi_{g,m} \boldsymbol{\Omega}_m) &= \boldsymbol{\Omega}_m \cdot \nabla \Psi_{g,m} + \Psi_{g,m} \nabla \cdot \boldsymbol{\Omega}_m \\ &= \boldsymbol{\Omega}_m \cdot \nabla \Psi_{g,m} \end{aligned} \tag{3.4}$$

since the divergence of a spatially constant vector field will always vanish. Because of this the streaming term in the S_N equation can be discretized as an implicit divergence term instead of as an explicit gradient term.

3.2.4 Laplacians

Both the neutron diffusion equation, and the momentum conservation equation contain Laplacian terms, of the form $\nabla \cdot D \nabla \Phi$ (where D may be position and time dependent). These are discretized as follows:

$$\begin{aligned} \int_V dV (\nabla \cdot D \nabla \Phi) &= \int_S \boldsymbol{dS} \cdot D \nabla \Phi \\ &\rightarrow \sum_f D_f \boldsymbol{S}_f \cdot (\nabla \Phi)_f \end{aligned} \tag{3.5}$$

The degree to which Laplacian terms are implicit or explicit depends on the interpolation scheme for face center values. With upwind differencing Laplacians can be fully implicit using Gauss-Seidel type solution of the matrix system along the flow direction [22].

3.2.5 Source Terms

Source terms that can be written as coefficients of the dependent variable may be treated either implicity or explicitly, depending on whether they increase or decrease diagonal dominance of the solution matrix. The stronger the diagonal dominance the more rapid the convergence of the problem. If the matrix is not diagonally dominant the problem may be unstable.

In the case that a source term has no dependence on the dependent variable, it can only be explicit.

The choice of whether a particular source term should be explicitly or implicitly discretized is taken adaptively at runtime and may therefore change during the course of a simulation.

3.3 Temporal Discretization

The temporal discretization simply consists of a finite number of consecutive time steps. Each physical field has a separate set of cell centre values and boundary values for each time step.

3.3.1 Time Derivatives

Time derivatives are discretized as finite differences according to:

$$\frac{\partial \Phi}{\partial t} \rightarrow \frac{\Phi^n - \Phi^{n-1}}{\Delta t^n} \tag{3.6}$$

where n indexes time steps. If n is the latest time step for which the equation has previously been solved the time derivative is fully explicit, i.e. the whole term ends up on the right-hand-side in Eq. (3.1). If, on the other hand, n in Eq. (3.6) is the time step for which the equation currently is to be solved Φ_n will be unknown and will be discretized implicitly, i.e. in the matrix A on the left-hand-side of Eq. (3.1). The first case is known as *explicit Euler* or *Euler forward* discretization while the latter is called *implicit Euler* or *Euler backward* [22].

In the present work a third method, the *Crank-Nicolson scheme*, is used, belonging to a family of semi-implicit schemes where a weighted sum of explicit and implicit contributions is used:

$$\frac{\partial \Phi}{\partial t} \rightarrow \gamma \left(\frac{\Phi^{n+1} - \Phi^n}{\Delta t^n} \right) + (1 - \gamma) \left(\frac{\Phi^n - \Phi^{n-1}}{\Delta t^{n-1}} \right) \tag{3.7}$$

where $n+1$ is the time step currently solved for and γ is the weight of the implicit term. If $\gamma = 0.5$ so that both contributions have equal weight, the Crank-Nicolson scheme is obtained. It is second order accurate in contrast to the fully implicit and explicit schemes which are first order accurate [22].

Chapter 4

Implementation

4.1 Cross-section Generation

The macroscopic cross-sections needed in solving the multiphysics problem are generated using the Monte Carlo code Serpent [21]. Serpent simulates neutrons at continuous energies, which is of great advantage when used for macroscopic cross-section generation. It uses evaluated nuclear data files as input data and generates multigroup cross-sections within a given spatial discretization.

The Serpent calculations are performed on a 2D lattice with support for arbitrary meshes. A typical Serpent mesh for a case consisting of a quarter of a 7x7 fuel pin system is shown in Fig. 4.1.

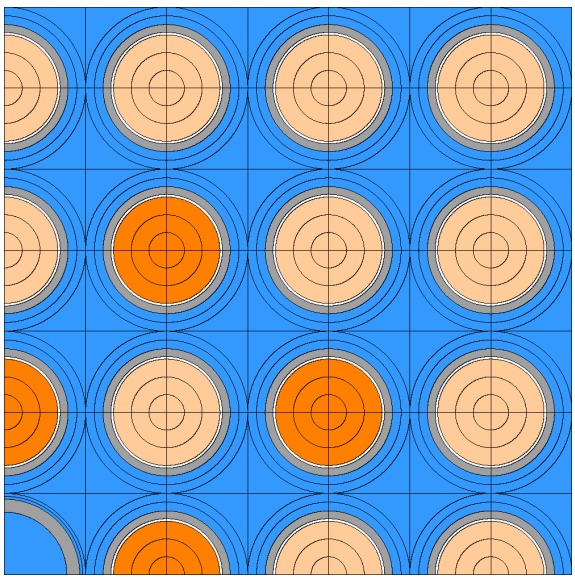


Figure 4.1: Resolved regions in Serpent cross-section generation case for a quarter of a 7x7 fuel assembly system.

Since the data are to be used in simulations where temperatures vary over time and since temperatures affect cross-sections, Serpent is run for a set of different temperatures. During the course of the multiphysics simulations, macroscopic cross-sections are obtained by interpolating the Serpent data based on the temperature profile in the simulation.

4.2 OpenFOAM

OpenFOAM [20] is a C++ library developed to handle primarily CFD problems using finite volume methods. The code, which is open source, is highly modular and readily extensible to handle other fields of physics, e.g. neutron transport. The code has a readable syntax for implementing physics solvers where the mathematical language of partial differential equations is closely mimicked. The physics solvers are well separated from the matrix solvers so that modifications in the solution procedure can be implemented in a straight-forward manner [33]. An example of how PDEs are implemented can be found in Fig. 4.2.

```
fvScalarMatrix PhiEqn
(
    fvm::ddt((1/XK_.v(e)),Phi[e])
    ==
    fvm::laplacian(X_.D(e),Phi[e])
    + fvm::SuSp(X_.Ss(e,e),Phi[e])
    - fvm::SuSp(X_.ST(e),Phi[e])
    + D
    + U
    + (1-beta)*F*XK_.chip(e)
    + XK_.chid(e)*d
);
```

Figure 4.2: Example OpenFOAM code defining the neutron diffusion equation.

The main data types of OpenFOAM are recognized as the standard entities encountered in tensor algebra, i.e. scalar, vector and tensor fields. These are known respectively as, `volScalarField`, `volVectorField` and `volTensorField`.

Depending on whether a quantity is scalar or vectorial it is implemented as a `volScalarField` or a `volVectorField`. In addition to this, many neutronic quantities have group and angular dependence. When that is the case the field is instead implemented as a `list` of such fields. In the case of the angular flux, Ψ, the list could be thought of as two-dimensional, since it must run over both energy groups and ordinates. However, the implementation is as a one-dimensional `list`, with indices organized as $M \times (g-1) + m$ (recall that g indexes energy groups up to G and m neutron directions up to M) in order to achieve more efficient memory management since the neutrons with different directions within one energy group are more closely coupled than those in different energy groups. Table 4.1 details the implementations of many of the main physical fields dealt with by the solver.

Among the quantities listed in Table 4.1 most are discretized on only one of the meshes and used only in the corresponding module of the code (i.e. neutronics or thermal-hydraulics). However, due to the coupling between the physics some fields need to be included in both modules and their values interpolated from one mesh to the other. The power profile is obtained from the neutron flux profiles solved for within the neutronics. In the thermal-hydraulic module it plays the part of a heat source and therefore needs to be interpolated onto the thermal-hydraulic mesh. On the other hand, the thermal-hydraulic module resolves the temperature profile throughout the system, which needs to be interpolated onto the neutronic mesh in order to update the cross-section profiles. The interpolation scheme is designed to be fully conservative w.r.t. transport of mass, momentum and enthalpy. Because enthalpy rather than temperature is the conserved quantity for heat transfer processes, the thermal-hydraulic module first calculates the enthalpy profile from the known temperature profile and the interpolation is carried out in terms of the enthalpy. The neutronic module then uses the interpolated enthalpy to compute the temperature profile on the neutronic mesh.

Table 4.2 presents the discretization schemes used for the different physical fields. The naming of

the discretization schemes conforms to that used in OpenFoam. See its User Guide [33] for more information.

Table 4.1: Physical fields and their implementation in OpenFOAM. NK denotes neutron kinetics, TH denotes thermal-hydraulics. Parentheses after lists specifies the length of the lists.

Field	Module	Data type
Scalar neutron flux (Φ)	NK	`list` (G) of `volScalarField`s
Angular neutron flux (Ψ)	NK	`list` ($G \times M$) of `volScalarField`s
Delayed neutron precursors (C)	NK	`list` (I) of `volScalarField`s
Thermal power (P)	NK/TH	`volScalarField`
Temperature (T), enthalpy (h)	TH/NK	`volScalarField`
Density (ρ)	TH	`volScalarField`
Velocity (U_i)	TH	`volVectorField`
Cross-sections (Σ_T,Σ_f)	NK	`list` (G) of `volScalarField`s
Differential scattering cross-section (Σ_s)	NK	`list` (G) of `volScalarField`s

Table 4.2: Discretization schemes for differential operators and interpolation.

Operator	Scheme
Time derivatives	Crank Nicholson (0.5)
Face gradients	Gauss linear
Divergence	Gauss upwind
Laplacians	Gauss linear corrected
Interpolation (default)	Linear
Interpolation (K,h)	Harmonic

The PDEs are solved by linearizing their discretized versions into matrix equations and solving these using different matrix solvers depending on the nature of the equation to be solved. The matrix solvers used in the solution of each field are listed in Table 4.3.

Table 4.3: Matrix solvers used for the fields.

Field	Matrix solver	Preconditioner
Scalar neutron flux, Φ	PCG	DIC
Angular neutron flux, Ψ	PBiCG	DILU
Delayed neutron precursors, C	PCG	DIC
Temperature, T	GMRES	Cholesky
Mass flux, ρU_i	PCG	DIC
Pressure, p	AAMG	DIC
Velocity, U	PBiCG	DILU
Turbulent kinetic energy, k	PBiCG	DILU
Dissipation of turbulent energy, ε	PBiCG	DILU

4.3 Neutronic Solvers

4.3.1 Diffusion Solver

The overall algorithm for solving the discretized group coupled diffusion equation, Eq. (2.12), and its precursor equation, Eq. (2.13) is outlined in Algorithm 4.1. The solver consists of an outer loop that iterates between the diffusion equation and the precursor equation.

Within this loop there is also a loop structure for solving the diffusion equations for all groups g, since they are all mutually coupled through the fission and scattering source terms. The inner loop goes through all energy groups in order from fastest to slowest, thereafter checking for inter-group convergence.

The outer loop then moves on to solve the precursor equation for each group of neutron precursors. There is no need for an inner iteration here, since all precursor balance equations are independent from one another.

Algorithm 4.1: Solution algorithm for the diffusion solver.

```
1  repeat
2      repeat
3          foreach energy group g do
4              Update scattering source term;
5              Update fission source term;
6              Update delayed neutron source term;
7              Solve diffusion equation (2.12) for group g;
8          end
9      until convergence between groups;
10     foreach precursor group i do
11         Solve precursor equation (2.13) for group i;
12     end
13 until outer convergence;
```

4.3.2 S_N Solver

The S_N solver solves the S_N equation, Eq. (2.16), and the multi-group precursor equation, (2.13).

The outer looping structure is similar to that of the diffusion solver, but in addition to an inner loop over energy groups, there is an additional inner layer looping over directions. For each energy group, the S_N equation is solved for each direction until convergence between all directions for that energy group. The loop over energy groups then continues until convergence between all directions and all energy groups is achieved. Thereafter the precursor equation is solved and the outer loop is continued until the whole neutronic problem has converged.

Algorithm 4.2: Solution algorithm for the S_N solver.

```
repeat
    repeat
        foreach energy group g do
            repeat
                foreach direction n do
                    Update scattering source term;
                    Update fission source term;
                    Update delayed neutron source term;
                    Solve S_N equation (2.16) for group g, direction n;
                end
            until convergence between directions;
        end
    until convergence between groups;
    foreach precursor group i do
        Solve precursor equation (2.13) for group i;
    end
until outer convergence;
```

4.4 Thermal-hydraulic Solver

The thermal-hydraulic solver is summarized in Algorithm 4.3. The first few steps solve equations concerning only the fluid moderator.

First the momentum equation is solved using an initial guess for the pressure profile taken from the previous iteration. The solution is used to update the velocity profile. After solving the momentum equation the turbulence model equations are solved and the turbulent viscosity is updated for the next iteration. Before and after solving for the temperature profiles the thermodynamic state is updated. The thermodynamic model includes density, but also other temperature dependent quantities used in the coupled solver, e.g. thermal conductivity and neutron diffusion coefficient.

The temperature equations are solved simultaneously by coupling the heat conduction equation with the heat convection equation. After solving the conjugate heat transfer problem the temperature profile and thermodynamic state is updated and the solver moves on to solving the pressure equation in the fluid.

In PISO coupling it is conventional to solve the pressure equation two times per iteration in order to stabilize the solver [22]. In this work experience has suggested that the best convergence is reached at as many as four pressure correction steps. After each step the mass fluxes through cell faces are updated to ensure conserved mass.

Algorithm 4.3: Solution algorithm for the thermal-hydraulic solver.

```
repeat
    Solve momentum prediction equation (2.34);
    Update velocity profile;
    Solve k equation (2.38);
    Solve ε equation (2.39);
    Update turbulence;
    Update thermodynamic state;
    Solve conjugate heat transfer, Eqs. (2.26) ∪ (2.35);
    Update thermodynamic state;
    for i=1:4 do
        Solve pressure correction equation (2.48);
        Update face fluxes;
    end
    Correct velocities;
until joint convergence of pressure, velocity and temperature;
```

4.5 Coupled Solution Procedure

The main application controls the overall flow of the simulation. Its iterative structure is to iterate between neutronics and thermal-hydraulics until the coupled problem has converged for the current time step. Thereafter the solver moves forward one time step (see next section for how the time step length is determined).

In each iteration the resulting power profile is transferred from the neutronics to the thermal-hydraulics and the temperature profiles of the fuel and the moderator is transferred from thermal-hydraulics to neutronics. The other variables relevant for multiphysics are updated using the thermodynamic model.

The coupled solution procedure is illustrated in Fig. 4.3. In the figure the preparatory steps are also included. Those are performed before the simulation in order to create material data and initial guesses for all fields. The material data are subsequently retrieved by temperature interpolation during runtime.

4.6 Time Step Control

4.6.1 Neutronics

The main idea behind the time control algorithm for the neutronics module is that the relative change of thermal power should not be too large between consecutive computational time points for any mesh cell. The implementation is based on an Ansatz for the time evolution of power when it is subject to constant forcing. Due to the locally determined and statistical nature of the driving forces for power changes the most natural approach is to assume exponential evolution of the power in any given cell:

$$P_i(t_0 + \Delta t) = P_i(t_0)e^{\lambda_i(t)\Delta t} \tag{4.1}$$

where i indexes cells. The task of the time control algorithm is to choose Δt so as to ensure that this projected change is not too large compared to the absolute value of P. The first step then, is to determine the present value of λ for the cell in which the power is changing the fastest.

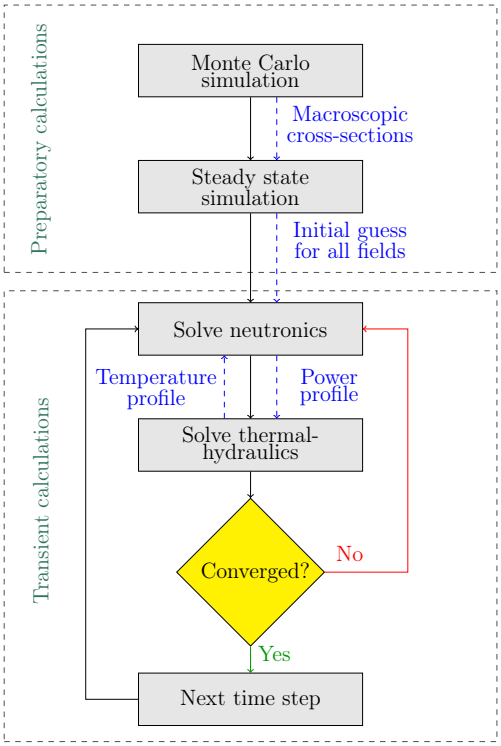


Figure 4.3: Coupling scheme for the solver

It should be noted that λ is neither constant over time nor between cells. It is to be seen as a characterization of the change rate in each cell at each computational time. In the continuation it will be assumed that the equations concern the cell with the highest relative rate of change, and the index i will be dropped.

Writing Eq. (4.1) for consecutive previous times (indexed by n) gives:

$$P_n = P_{n-1} e^{\lambda_{n-1} \Delta t_{n-1}} \tag{4.2}$$

The finesse of Eq. (4.2) is that all quantities except λ_{n-1} are known in advance. Solving for λ gives:

$$\lambda_{n-1} = \frac{1}{\Delta t_{n-1}} \log \left(\frac{P_n}{P_{n-1}} \right) \tag{4.3}$$

Although it should rather be λ_n instead of λ_{n-1} that is used to choose the next time step this quantity cannot be known at this stage, since the time step not yet selected would then need to be used in Eq. (4.3).

Δt_n should now be chosen so that

$$P_{n+1} = P_n e^{\lambda_n \Delta t_n} = (1 \pm \alpha) P_n \tag{4.4}$$

where α is the maximum tolerated relative change in power between consecutive points in time. The sign of this term is the same as the sign of λ. Solving for Δt_n gives:

$$\begin{aligned} \Delta t_n &= \frac{1}{\lambda_n} \log(1 \pm \alpha) \\ &\approx \frac{1}{\lambda_{n-1}} \log(1 \pm \alpha) \end{aligned} \tag{4.5}$$

4.6.2 Thermal-hydraulics

The thermal-hydraulic time step is chosen on the basis of the *Courant-Friedrichs-Lewy* (CFL) criterion [22], which dictates that the local *Courant number*, defined in Eq. (4.6), does not exceed 1 anywhere in the system.

$$Co = \max_i \left(\frac{U_i \Delta t}{\Delta x_i} \right) \tag{4.6}$$

The physical significance of the CFL criterion is that it does not allow any parcel of fluid to traverse more than one mesh cell per time step. Not fulfilling this criterion may lead to numerical difficulties [22].

4.7 Boundary conditions

All fields have specular reflective boundary conditions in the radial direction, i.e. the systems simulated in this project can be seen as cuboids in an infinitely repeated lattice. On the contrary, the axial boundary conditions are based on physical reasoning and CFD praxis. Table 4.4 details all axial boundary conditions used in the project.

Table 4.4: Boundary conditions in the axial direction.

Field	Inlet BC	Outlet BC
Scalar neutron flux, Φ	0	0
Angular neutron flux, Ψ	Free	Free
Delayed neutron precursors, C	0	0
Temperature, T	550 K	Zero gradient
Temperature, T (after inlet transient)	540 K	Zero gradient
Pressure, p	Zero gradient	15.5 MPa
Velocity, U_i	(0,0,4) m/s	Zero gradient
Turbulent kinetic energy, k	0.02 m^2/s^2	Zero gradient
Dissipation of turbulent energy, ε	0.02 m^2/s^2	Zero gradient

The boundary conditions for the angular neutron flux need a more detailed explanation. The boundary conditions treat directions differently depending on whether they point out from the system or into it through the boundary in question. Outward pointing neutron directions have a zero gradient boundary condition. The inward facing angular flux is set to a zero boundary condition, i.e. no neutrons enter or reenter the system. These boundary conditions approximate what would be the case if the reactor was situated in a vacuum, hence the name.

Chapter 5

Results and Analysis

In this chapter, a number of simulations performed by the implemented solver are presented and analyzed. Some of the simulations were done using the diffusion solver and others using the S_N solver for the neutronics. Which of them was used for each simulation is specified in the main text. All simulations presented in this thesis were done using four energy groups and six groups of delayed neutron precursors. The material data pre-generated from Serpent were prepared using three different temperature points.

5.1 Demonstration of Solver Capabilities

The coupled solver (using diffusion neutronics) was tested for a single fuel pin geometry, illustrated in Fig. 5.1. The geometry data for this case is presented in Table 5.1 and the mesh data is presented in Table 5.2. One quarter of a fuel pin cell was simulated. Five simulations were run where the system was allowed to approach its steady state undisturbed with five different concentrations of aqueous boron. The result is shown in Fig. 5.2, where the power density in the middle of the pin is plotted over time. The difference between the curves in Fig. 5.2 demonstrates the existence of neutronic feedback on power through absorption cross-sections.

Table 5.1: Geometry of the fuel pins in different meshes.

Dimension	1x1/7x7 diffusion	2D
Height	3.5 m	0.5 m
Fuel pitch	1.25 cm	1.6 cm
Fuel pin radius	0.41 cm	0.2 cm
Cladding thickness	0.06 cm	-
Gap thickness	0.02 cm	-

All power curves, except for the one with 700 ppm boron, which is very close to criticality, initially diverge rapidly depending on their differences in reactivity. After some tenths of a second the divergence is slowed down. This demonstrates that the solver captures the dynamics of prompt and delayed neutrons.

It can also be seen that the curves with 400 and 500 ppm boron overshoot before they settle in. This oscillatory behavior demonstrates the presence of different time scales. In the absence of thermal-hydraulics there is no overshoot; the only feedback present in neutronics is Doppler broadening

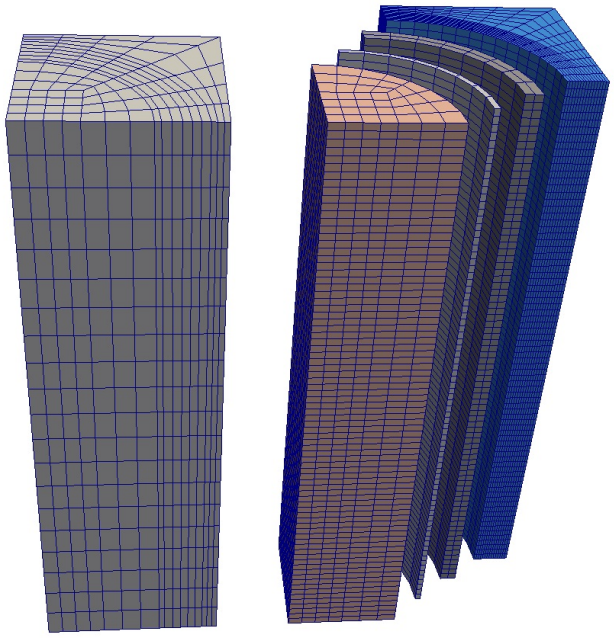


Figure 5.1: Mesh and geometry for single fuel pin system [34]. Axial direction compressed to 1% of real length for visualization purposes. Neutronic mesh to the left and thermal-hydrualic mesh on the right, decomposed into fuel, gap, cladding and moderator regions.

Table 5.2: Number of cells in different regions in the different meshes.

	1x1	7x7	2D
Fuel (TH)	3200	153600	1000
Gap (TH)	640	30720	-
Cladding (TH)	1280	15860	-
Moderator (TH)	19200	846080	1000
Full System (NK)	2240	114702	1400

which can be considered instantaneous. The overshoot is due to the time it takes for the increased heat produced in the fuel during the power excursion to reach the moderator, decreasing its density and thus its moderator quality, leading to an eventual decrease in power.

In addition to power transients, the reactivity over time for the case of 700 ppm boron is also plotted in Fig. 5.2. Reactivity is defined as [25]:

$$\varrho = \frac{k_{\text{eff}} - 1}{k_{\text{eff}}} \approx k_{\text{eff}} - 1 \quad \text{for } k \approx 1 \tag{5.1}$$

and is often measured in *percent-mille* (pcm), i.e. in units of 10^{-5}.

The reactivity was determined at one second intervals using a steady-state diffusion solver with frozen thermal-hydraulic conditions in order to take snap-shots of the neutronic state. As can be seen the reactivity is constant at 2 pcm. This is consistent with the constancy of the power density. Indeed, at such small deviations from criticality one should not expect to observe any changes in

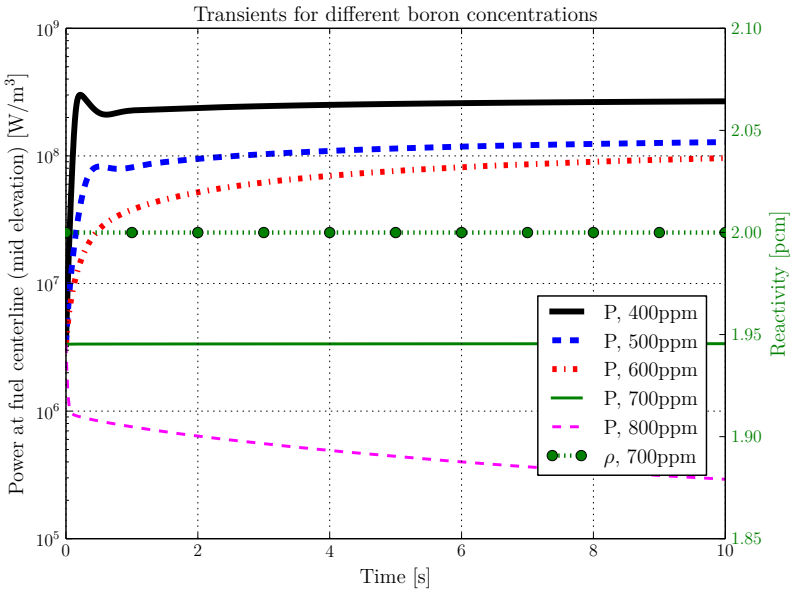


Figure 5.2: Diffusion simulation on single fuel pin cell system. Left axis: Power density over time in an undisturbed simulation starting from different boron concentrations. Right axis: Reactivity over time for the case with 700 ppm boron [34].

power during 10 seconds. A back-of-the-envelope calculation using an approximated form of the Inhour equation [25]:

$$T \approx \frac{1}{\rho} \sum_{i=1}^{6} \frac{\beta_i}{\lambda_i} \tag{5.2}$$

yielded a reactor period on the order of 3600 seconds. This predicts that the power at the end of the simulation would be a factor $e^{10/3600} \approx 1.003$, i.e. 0.3%, greater than the initial power, clearly not visible in the graph. This demonstrates good agreement between the transient and steady-state solvers.

Figure 5.3 shows the same dial-in process during the first two seconds. At 2 seconds into the simulation the moderator inlet temperature is lowered with 10 K over the course of one second. As expected, the lowering of the inlet temperature leads to an increase in power due to better moderation. It can be observed that the increase in power is relatively greater for the less supercritical cases. This is also expected because of the initially higher moderation for these cases.

Once again the reactivity development is plotted for the case closest to criticality. It is seen that the lowering of 10 K caused a reactivity insertion peaking at approximately 500 pcm and then slowly tapering off. If the simulation was allowed to continue, the reactivity would eventually return to zero as the system would settle into a new steady-state. Due to the sluggishness of the slower delayed neutron groups it would have taken minutes to fully return to a critical state. Nevertheless, it can still be seen in the power plot that the system is about to settle on a new higher power level.

The same inlet temperature transient was also applied to a larger case, consisting of 7x7 fuel pins. The geometry is shown from above in Fig. 5.4. It is shown in the figure that the fuel pins in the system have different enrichments of ^{235}U. The resulting temperature profiles in the moderator and the fuel are shown in Fig. 5.5, just before the temperature is lowered. The mesh is outlined

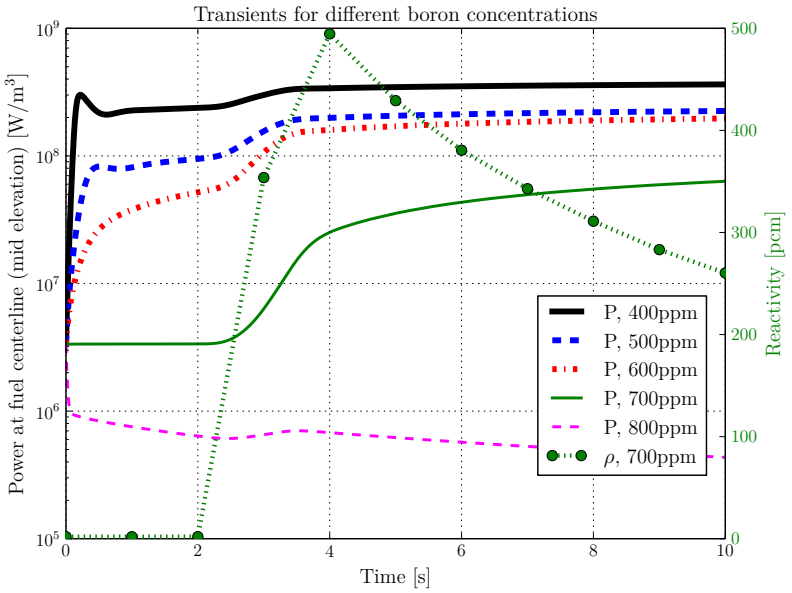


Figure 5.3: The first two seconds of this simulation are identical to Fig. 5.2. Thereafter, the inlet temperature in the moderator is lowered with 10 K [34].

in the figure. Geometry data is once again presented in Table 5.1 and mesh data is presented in Table 5.2.

The temporal behavior of this simulation is presented in Fig. 5.6, with data for the points at Probe A and Probe B, as shown in Fig. 5.4. Since this is a supercritical case, the same type of overshoot of power is seen in this case, as in Figures 5.2 and 5.3. However, in Fig. 5.6 it is seen that the fuel temperature exhibits the same behavior. This is expected, since to the first order the fuel temperature is proportional to the power produced in the fuel.

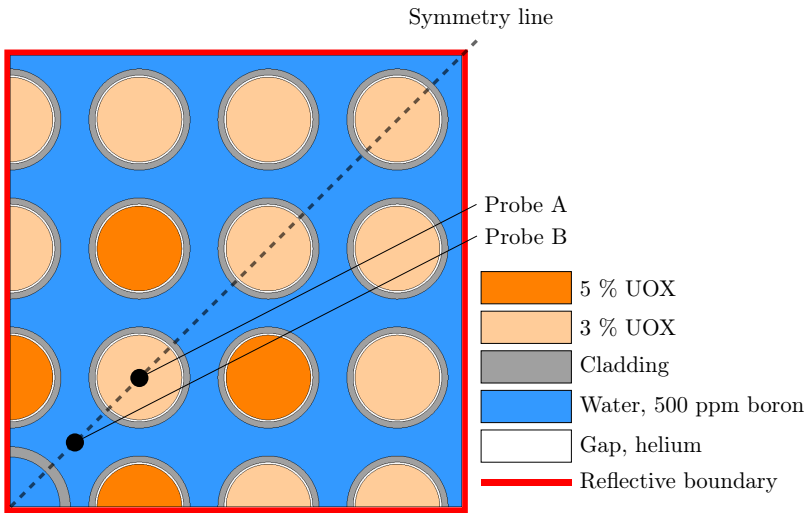


Figure 5.4: Horizontal geometry for a quarter of a 7×7 system, including horizontal probe locations used for post-processing [34].

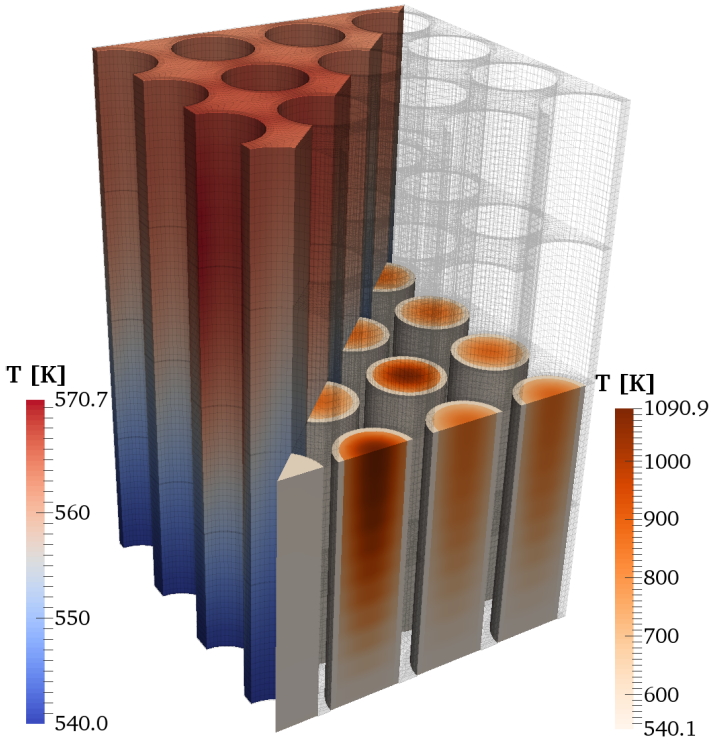


Figure 5.5: Temperature profiles in fuel and moderator after two seconds of unperturbed simulation using diffusion neutronics [34].

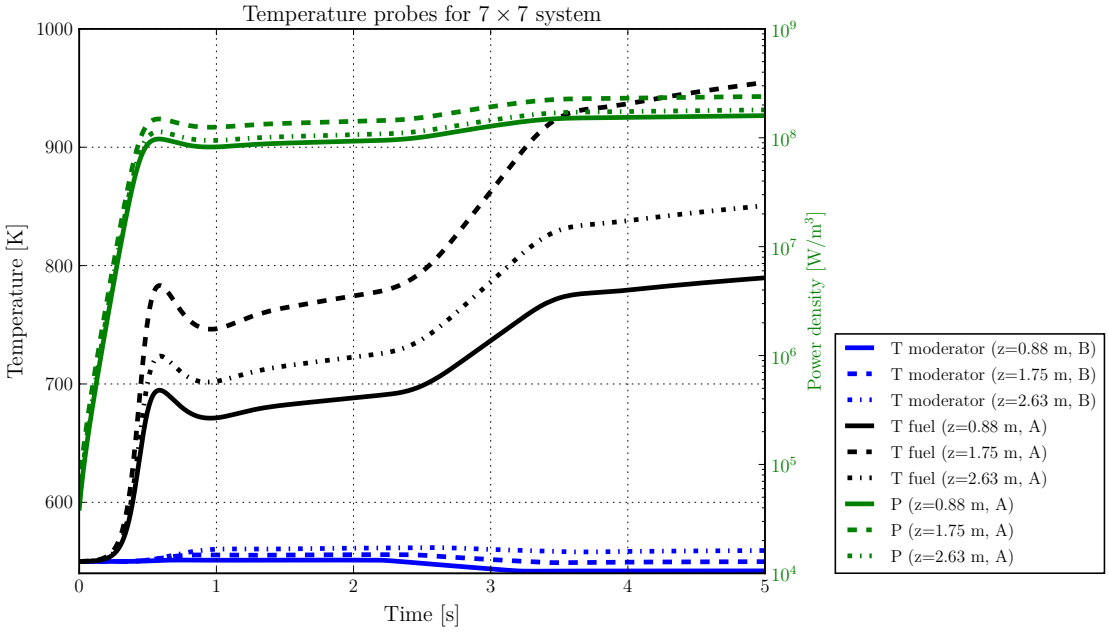


Figure 5.6: Transient behavior of power and temperature [34] in 7x7 fuel pin simulation.

5.2 Comparison between Diffusion and S_N Solvers

In order to compare the behavior of the diffusion solver with that of the S_N solvers the geometric system presented in Fig. 5.7 was used to simulate inlet temperature transients on the same mesh with diffusion, S_2 and S_4 solvers. Due to the computational cost of the simulation a 2D system was used for the comparison, consisting of a cuboid of fuel with surrounding moderator in the x-direction. The 2-dimensionality comes from only resolving one layer of cells in the y-direction so that even though OpenFOAM solves the equations in 3-dimensional space, the y-direction is not resolved. The geometry data are provided in Table 5.1 and the mesh properties of the 2D system are specified in Table 5.2.

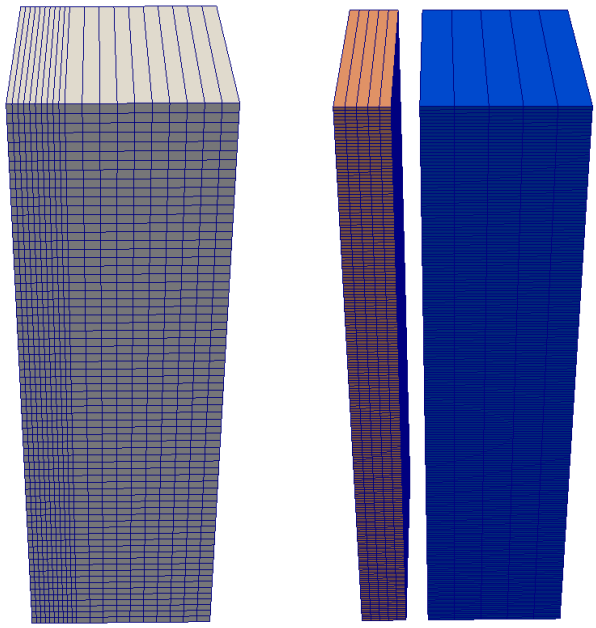


Figure 5.7: Mesh used in the 2D case. Axial direction compressed to 5% of real length for visualization purposes. Neutronic mesh to the left, and thermal-hydrualic mesh to the right, decomposed into fuel and moderator. In the 2D case, gas gap and fuel cladding are not modeled.

Since the effective multiplication factor, k_{eff}, is the most important global parameter controlling the behavior of the reactor, a criticality search was performed for each of the solvers. Because of their differences, slightly different concentrations of boron gave rise to critical conditions. The boron concentrations used are presented in Table 5.3 and the results of the simulation are shown in Fig. 5.8.

Table 5.3: Critical boron concentrations for different neutronic solvers.

Solver	Concentration (ppm)
Diffusion	525
S_2	663
S_4	673

In Fig. 5.8 the inlet temperature is lowered 10 K between 1 and 2 seconds. This leads to increases in power for all solvers. Simultaneously the temperature in the fuel rises. In this case the net effect

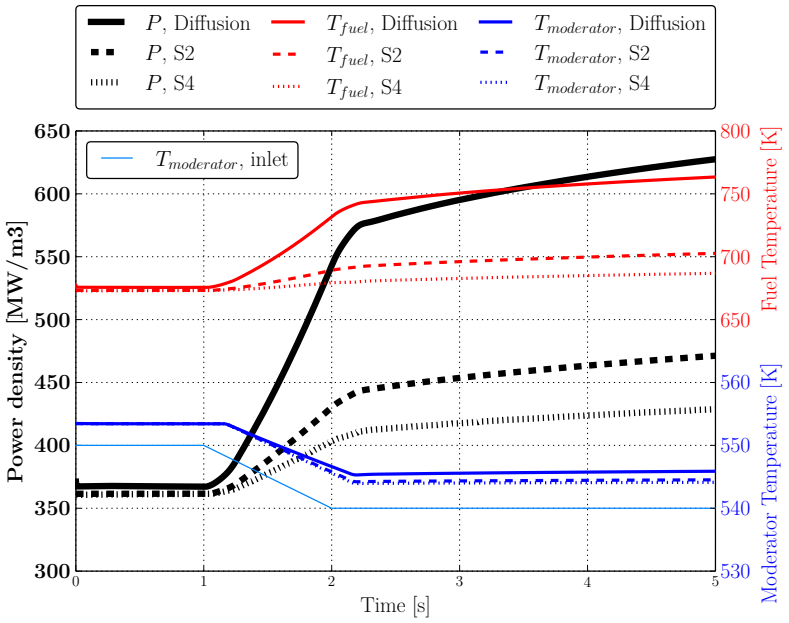


Figure 5.8: Response of power density, fuel temperature and moderator temperature to a lowering of the inlet temperature by 10 K.

on the moderator temperature is a lowering, although its temperature in the middle of the core is not lowered by as much as the inlet temperature due to the increase in nuclear heating. Note that the transient behavior of the three solvers is qualitatively similar, although there are clear quantitative differences.

To further compare the behavior of the solvers the spatial profiles acquired for the scalar neutron fluxes were compared. The profiles were examined at 0.0, 2.0 and 5.0 seconds into the simulations. In all cases the profiles were normalized by their maxima to facilitate comparison of their shapes. It was found that the spatial profiles did not change significantly during the course of the transient. Figure 5.9 shows the radial profile of fast and thermal scalar fluxes and Fig. 5.10 shows the axial profiles, in both cases at 2.0 seconds.

In Fig. 5.9 it can be clearly seen that the S_N solvers resolve heterogeneities that the diffusion solver smooths out. It is also clear that the heterogeneity at the interface between fuel and moderator is greater for S_4 than S_2, implying that a higher order solver would probably resolve an even greater heterogeneity.

In Fig. 5.10 it is seen that the diffusion solver gives a narrower axial profile than the S_N solvers, both for fast and thermal fluxes. A likely explanation for this is the difference in boundary conditions at the inlet and outlet of the core between the diffusion and the S_N solvers. While in the diffusion solver the flux is simply forced to zero at the boundaries, the situation is a bit more subtle for the S_N solver due to the fact that it solves for the angular neutron flux rather than the scalar flux. Instead it is supposed that no neutrons enter the system through the boundaries and that the flux of neutrons leaving the system has a gradient equal to zero. This explains why the S_N profiles are wider, since only half the scalar flux is forced to zero at the axial boundaries; the other half is not limited in magnitude, only their gradients are. If different boundary conditions were used for the diffusion solver, such as those proposed by Mark or Marshak [27] (which are both more akin to the free boundary conditions), the difference in width of the axial profiles may have been smaller.

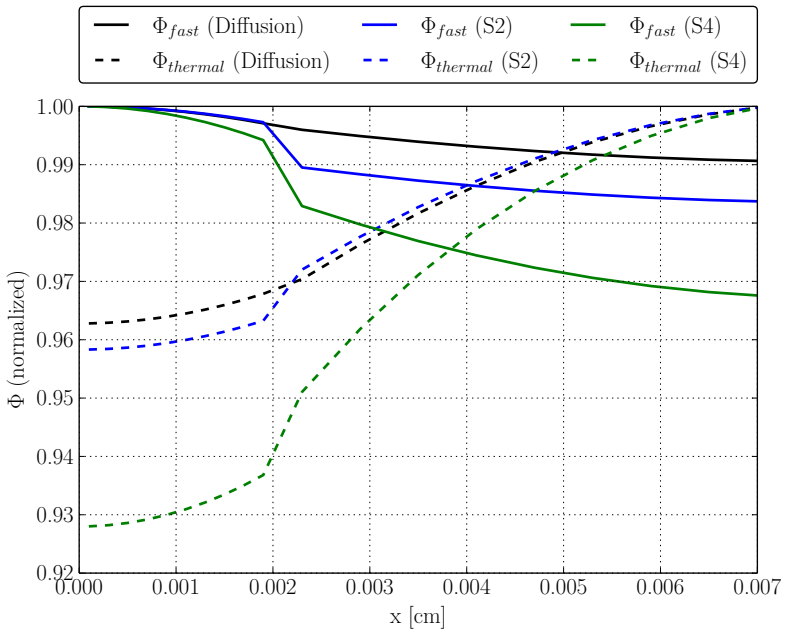


Figure 5.9: Radial profiles for the scalar neutron flux in mid-core at 2.0 s, with the contributions of the individual energy groups summed up into fast and thermal flux. All profiles are normalized by their respective maxima.

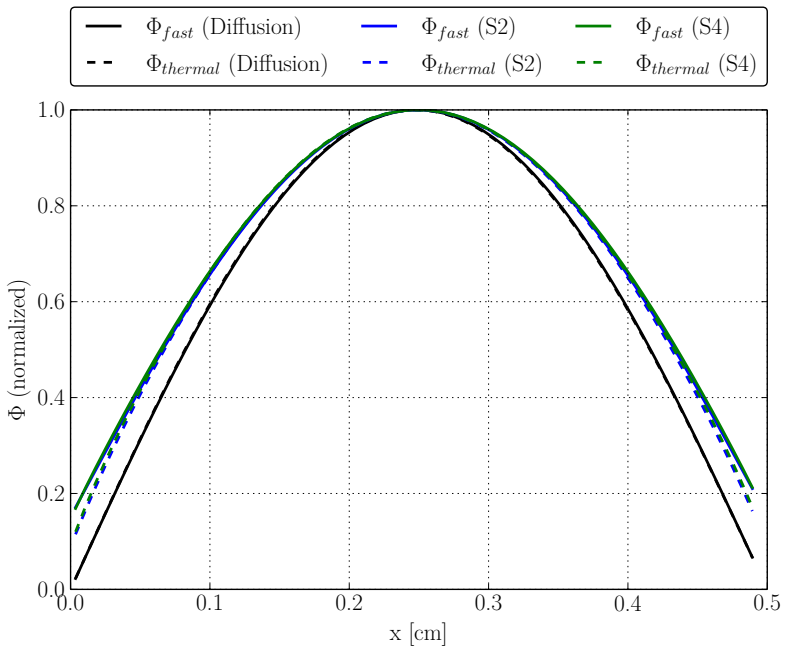
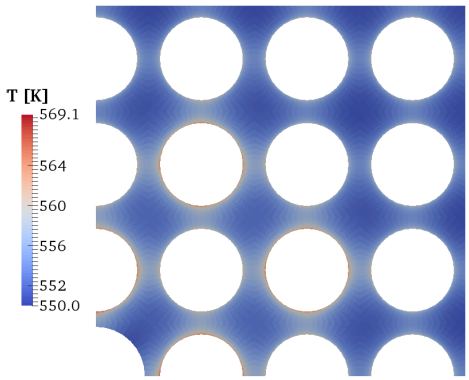
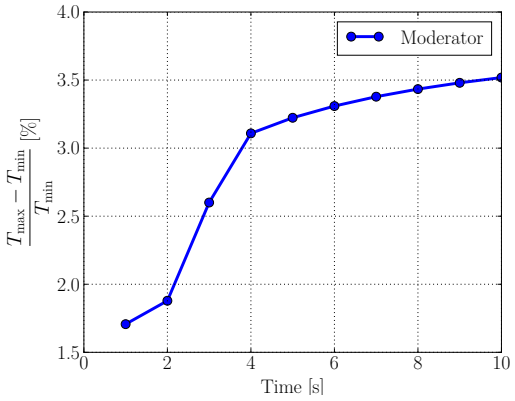


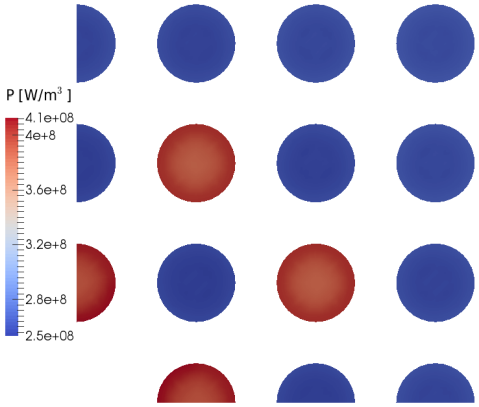
Figure 5.10: Axial profiles for the scalar neutron flux in the fuel centerline at 2.0 s, with the contributions of the individual energy groups summed up into fast and thermal flux. All profiles are normalized by their respective maxima.



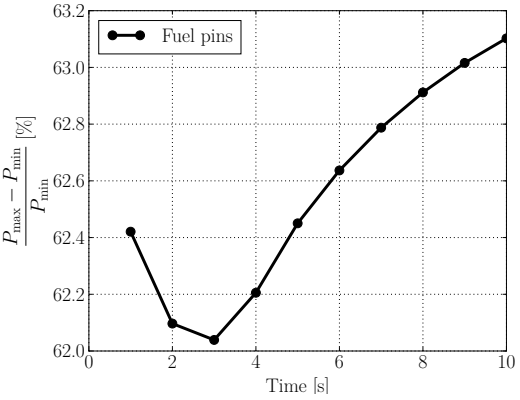
(a) Moderator temperature at mid-elevation ($t = 3$ s)



(b) Time-dependent heterogeneity of temperature in the moderator.



(c) Fuel power density at mid-elevation ($t = 3$ s)



(d) Time-dependent difference between max and min power at mid-elevation.

Figure 5.11: Axial slices of moderator temperature and power density with corresponding time development of max and min values at mid-elevation [34].

5.3 Evolution of Spatial Heterogeneity during Transients

An interesting question concerning the finely coupled multiphysics that cannot be answered using steady-state solvers is what effect transient conditions have on spatial heterogeneity.

To investigate this a horizontal plane was cut out at the middle of the core height in the 7x7 fuel pin simulation presented in Figs. 5.5 and 5.6. In this plane a measure of the heterogeneity of a variable was defined as the difference between the maximum and minimum value of that variable within the plane. This measure was then normalized by the minimum and plotted over time. The results for temperature and power density are plotted in Fig. 5.11.

It is seen that the horizontal heterogeneity of the moderator temperature approximately doubles during the course of the transient. This is interesting because the lowering of the inlet temperature is uniform. Still, the dynamics of the system causes a significant transient increase in heterogeneity, implying that the full temporal behavior is important for the finely coupled multiphysics.

The heterogeneity of the power distribution in the fuel does not show a clear trend and changes only minutely in relative terms during the transient. This is not wholly unexpected, since this particular transient has its origins in the moderator.

5.4 Discretization Studies

5.4.1 Time Step Length Study

To assess how short time steps are necessary to yield converged results two time discretization studies were performed; one for the diffusion solver and one for S_4, both using the standard 2D system described in Tables 5.1 and 5.2 and Fig. 5.7. The results are shown for S_4 in Fig. 5.12 and for diffusion in Fig. 5.14.

The study was performed by using fixed time steps and varying them between some short reference step length to subsequently longer time steps. A measure of the deviation in a variable from the reference at some given point in time can be calculated as:

$$\varepsilon = \frac{x_{\Delta t} - x_{ref}}{x_{ref}}$$

In this analysis the total system power was used to quantify the accuracy of simulations using different time steps. A time step of 10^{-4} s was used as the reference. The result for S_4 simulations is presented in Fig. 5.12. One simulation was performed using adaptive time step selection with $\alpha = 0.001$, cf. Eq. (4.4).

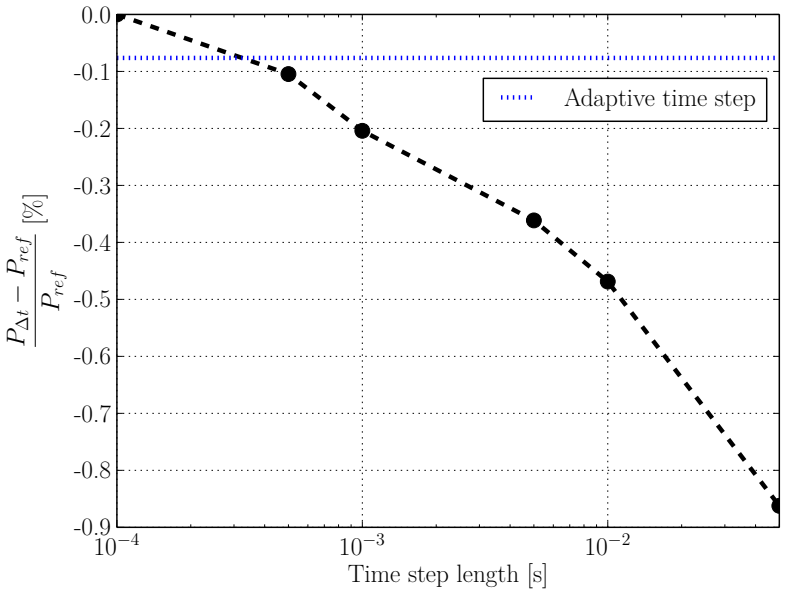


Figure 5.12: Time discretization study for an S_4 simulation. The measurements were made 2.0 seconds into the simulation, one second after starting an inlet temperature transient. Reference time step: 10^{-4} s.

The accuracy seems to be converging rather smoothly, or at least monotonously for shorter time steps. The blue horizontal dashed line marks the measured accuracy of the adaptive time step algorithm. It can be seen that in this specific case the adaptive step performed better than all

fixed time steps. The variation of the adaptive time step over time during this simulation is plotted in Fig. 5.13.

There is a subtle difference in how the data points are obtained between the adaptive case and the others. While the simulations with fixed time step always have a time step exactly at 2 seconds, this is not the case for the adaptive step. Therefore the value of the error for the adaptive step is interpolated between the two nearest time steps, using simple linear interpolation.

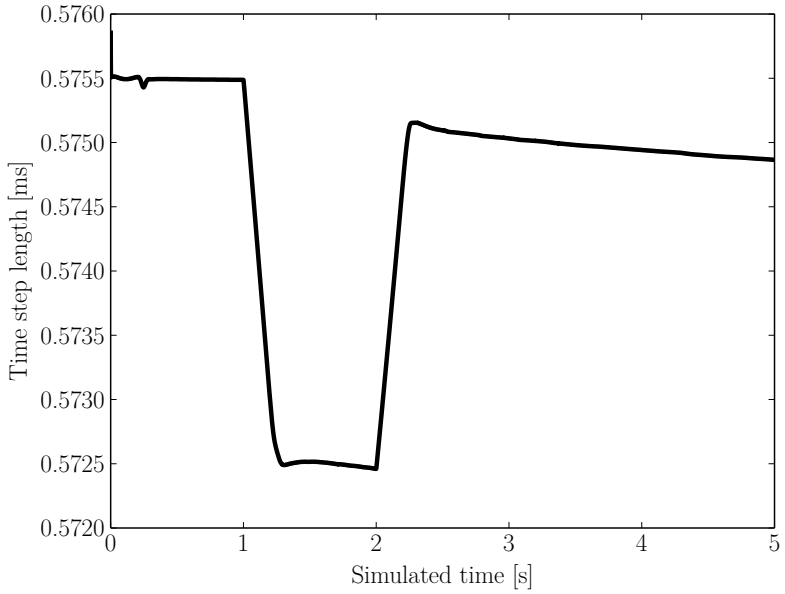


Figure 5.13: Adaptive time step length over time for the S4 case.

It is seen in Fig. 5.13 that the time step becomes shorter at the onset of the transient conditions. After 2 seconds of simulated time the time step increases again. The length of the adaptive time step varies around a level that is slightly longer than $5 \cdot 10^{-4}$ s. That it performed slightly better than that step should be considered as a coincidence; there is no reason why a time step that is longer at every point in time should outperform the shorter step. The reason for this result may be the measurement error introduced because of interpolation between neighboring time steps. It is worth noting that the errors in general are small, only on the scale of tenths of percents. The problems with even longer time steps are due to convergence problems rather than accuracy issues.

An equivalent experiment was performed with the diffusion solver. The result can be seen in Fig. 5.14. In this case the adaptive time step actually seems to be performing worse than the fixed steps in spite of having constantly a shorter time step than three of them, as can be seen in Fig. 5.15. As discussed above, the most probable explanation for this unexpected result is that the real error is smaller than that introduced by interpolation.

It should be noted that in both cases the adaptive time step was in practice solely determined by thermal-hydraulics. The neutronic module typically proposed time steps many orders of magnitude larger. Thermal-hydraulics seems to most often be the limiting factor with regards to time step length for inlet temperature transients.

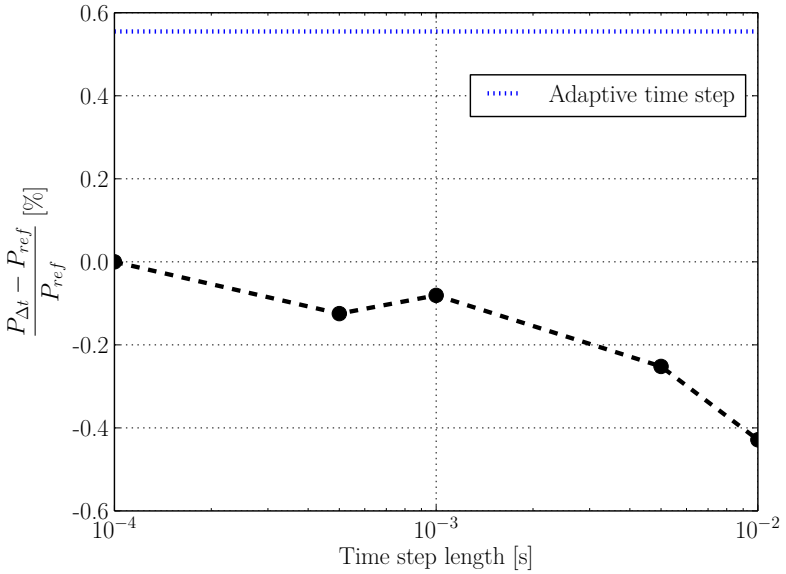


Figure 5.14: Time discretization study for a diffusion simulation. Measurement is made 2.0 seconds into the simulation, one second after starting an inlet temperature transient. Reference time step: 10^{-4} s.

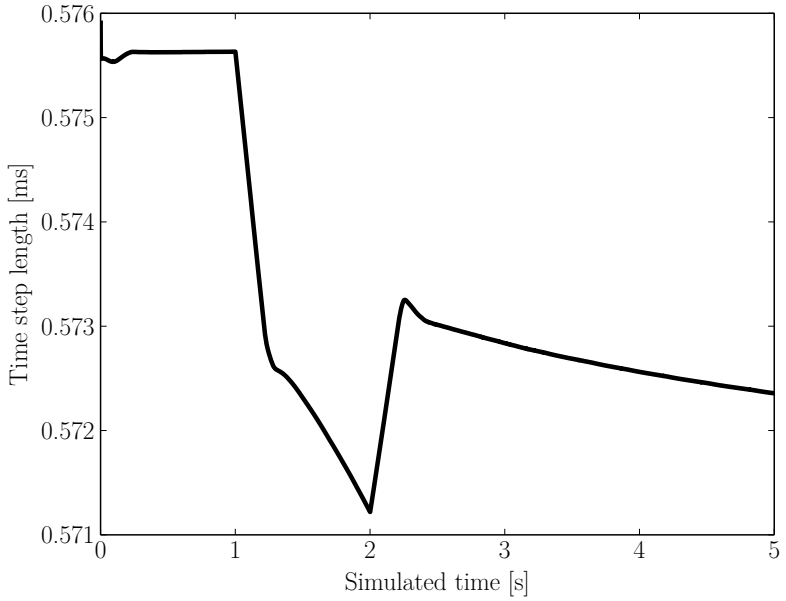


Figure 5.15: Adaptive time step length over time for the diffusion case.

The main conclusion from the time discretization studies is that although the implemented solvers do not always converge, the choice of time step length seems to have little effect on the accuracy of the solvers.

5.4.2 Mesh Refinement Study

Two studies were conducted on the effects of mesh resolution on accuracy and computational cost. Both were done using the S_4 solver. The reason for focusing on the S_N rather than the diffusion solver is that it is known from the results in section 5.2 that the diffusion solver does not sufficiently resolve the heterogeneities of the physics on reasonably fine meshes.

The studies were made on the 2D geometric system described in Table 5.2 and Fig. 5.7, but with variable number of cells in one of the grids at a time. Starting from the reference, the number of cells in each direction in the mesh to be varied was halved in a series of steps. The first study, presented in Fig. 5.16 and Table 5.4, varies the number of thermal-hydraulic cells, the second, presented in Fig. 5.17 and Table 5.5, does the same for the neutronic grid.

The measure of accuracy used in the mesh refinement study is the same as in the time discretization studies.

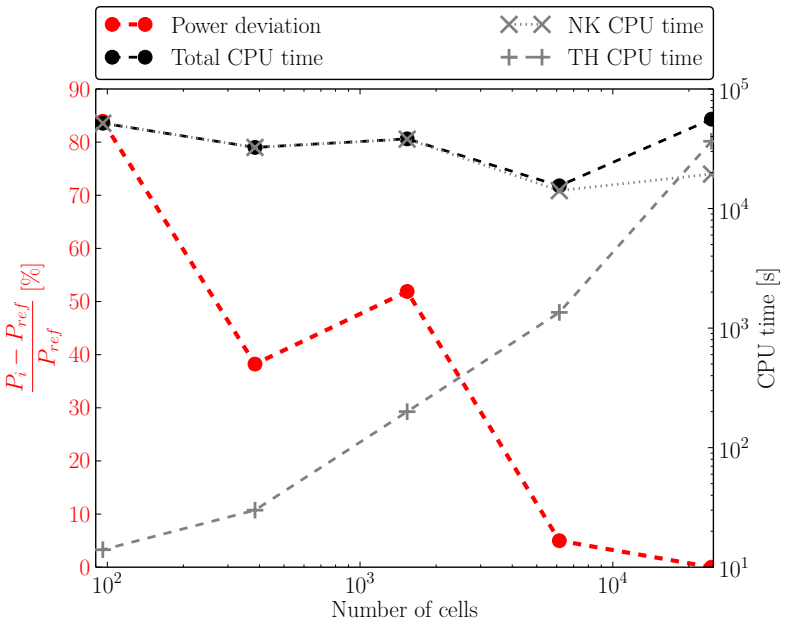


Figure 5.16: Red: Relative deviation in total system power from reference case as a function of the number of thermal-hydraulic mesh cells. All data points correspond to t=2.0 s with an inlet temperature transient applied at t=1.0 s. Black and gray: CPU time for neutronics (NK), thermal-hydraulics (TH) and total as a function of the number of cells.

There seems at first sight to be a very complex relationship between mesh resolution and the results. It can also be seen in Figs. 5.16 and 5.17 that the errors are quite large. However, note in Tables 5.4 and 5.5 that in both cases the initial reactivities of the system vary quite substantially between the different mesh resolutions. This initial difference will propagate in time so that the system responds to the transient from different starting states. This makes it difficult to draw far reaching conclusions based on the presented mesh studies. Figures 5.18 and 5.19 confirm that the power levels of the different cases are quite different at the onset of the inlet transient, which makes comparisons difficult.

An important conclusion from this analysis is that when comparing systems with different configurations it is important that all systems have similar initial reactivity. Preferably all systems should be initialized at a critical state so that no feedback effects in the multiphysics affect the cases differently. However, at present no automatic criticality search is supported within the framework, which makes it a very costly and laborious process.

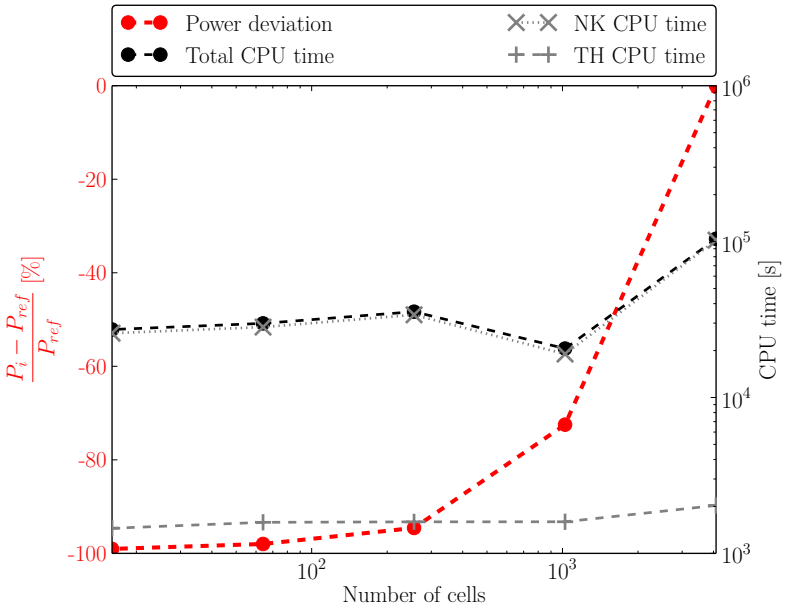


Figure 5.17: Red: Relative deviation in total system power from reference case as a function of the number of neutronic mesh cells. All data points correspond to t=2.0 s with an inlet temperature transient applied at t=1.0 s. Black and gray: CPU time for neutronics (NK), thermal-hydraulics (TH) and total as a function of the number of cells.

Table 5.4: Number of cells in the mesh refinement study for thermal-hydraulics, initial reactivity, and CPU time spent solving neutronics contra thermal-hydraulics.

Number of cells by region					
Fuel	8 192	2 048	512	128	32
Moderator	16 384	4 096	1 024	256	64
Total (TH)	24 576	6 144	1 536	384	96
Neutronics	1 400	1 400	1 400	1 400	1 400
Total (TH+NK)	25 976	7 544	2 936	1 784	1 496
Initial reactivity [pcm]	+3	-5	+151	+112	+210
CPU time (TH) [s]	36 530	1 353	200	30	14
CPU time (NK) [s]	19 420	14 134	38 064	32 436	51 726
CPU time (Total) [s]	55 950	15 487	38 264	32 466	51 740
CPU time/cell [s]	2.15	2.05	13.03	18.19	34.59

Table 5.5: Number of cells in the mesh refinement study for neutronics, initial reactivity and CPU time spent solving neutronics contra thermal-hydraulics.

Number of cells by region					
Fuel	2 048	2 048	2 048	2 048	2 048
Moderator	4 096	4 096	4 096	4 096	4 096
Total (TH)	6 144	6 144	6 144	6 144	6 144
Neutronics	4 096	1 024	256	64	16
Total (TH+NK)	10 240	7 168	6 400	6 208	6 160
Initial reactivity [pc]	+950	-202	-2362	-6173	-12158
CPU time (TH) [s]	2 030	1 595	1 593	1 582	1 449
CPU time (NK) [s]	102 207	19 002	33 912	28 296	25 846
CPU time (Total) [s]	104 237	20 597	35 505	29 878	27 295
CPU time/cell [s]	10.18	2.87	5.55	4.81	4.43

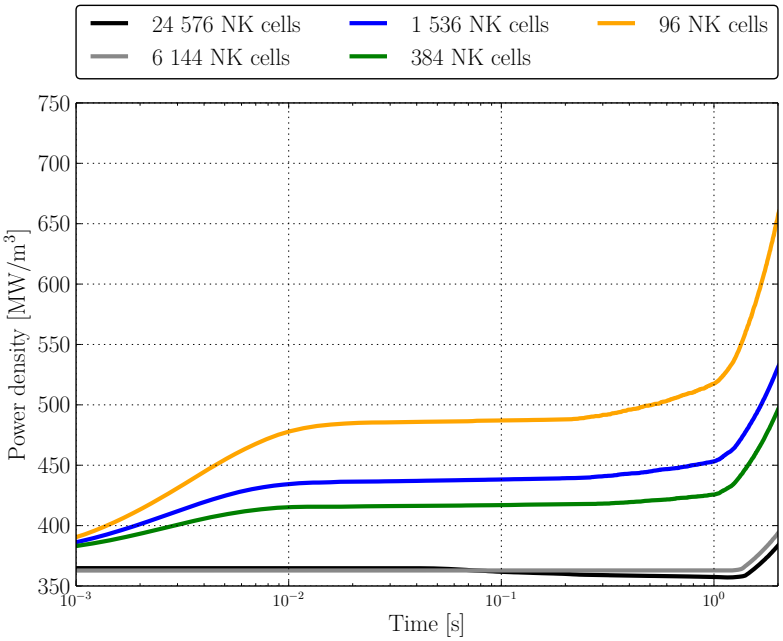


Figure 5.18: Transient power behavior for different thermal-hydraulic mesh resolutions. Note that the time axis is log scaled.

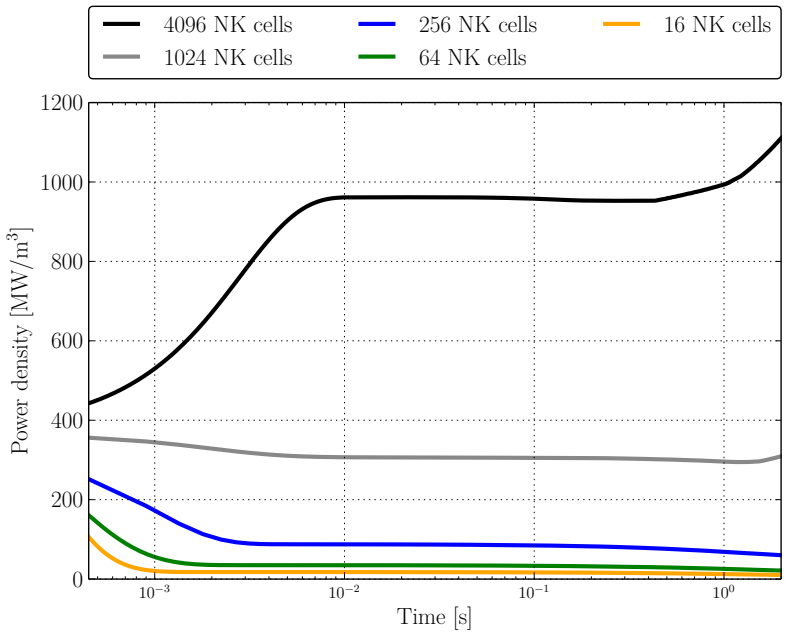


Figure 5.19: Transient power behavior for different neutronic mesh resolutions. Note that the time axis is log scaled.

5.5 Convergence Behavior and Computational Cost

Fig. 5.20 shows the computational burden over time, measured as CPU seconds per simulated second for the S_4 simulation in Fig. 5.8. It is seen that for the present case the neutronics is much more computationally heavy than the thermal-hydraulics. While not always true, depending on mesh resolutions and other factors, in all reasonable configurations this has been the case for the S_N solver. This means that accelerations of the individual solvers should be prioritized for the S_N solver rather than the thermal-hydraulic solver. It should be noted however, that the time step length is typically limited by the thermal-hydraulics, so that the computational cost of the whole solution process could potentially be lowered by using more closely coupled algorithms capable of using longer time steps. In Fig. 5.20 it can also be seen that the transient originates in the moderator, in that the thermal-hydraulic computer load goes up rapidly at the onset of the transient while the neutronics responds first when information of the disturbance has traveled from the fluid at the inlet to the fuel region.

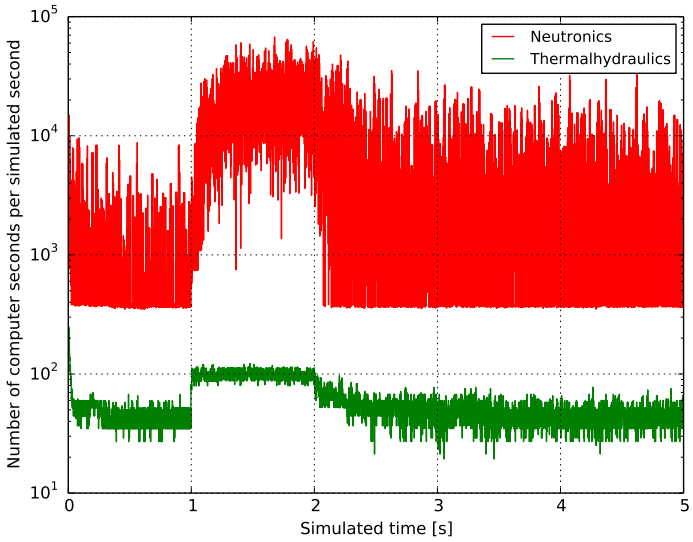


Figure 5.20: Computational load over time for the S4 transient presented in Fig. 5.8. A moving average was used to smooth the curves.

Fig. 5.21 shows (for the same simulation) the number of multiphysics iterations per time step over time. It is seen that the coupled nonlinear problem converges very well, in that the solver for this case (which is by this respect quite typical) most of the time only needs one multiphysics iteration to converge. During initialization and the early phase of the transient the solver needs two iterations per time step, but this must also be considered fast convergence.

The good performance of the iterative multiphysics solution procedure decreases the incentives for implementing a fully implicit method for the nonlinear problem and shows that the operator splitting approach works well in the conditions tested within this thesis.

Fig. 5.22 is a histogram over the number of group iterations, i.e. the number of times lines 2-13 in Algorithm 4.2, are run in order for the full neutronic problem to converge. Figure 5.23 shows an analogous histogram for the number of inner iterations, lines 4-11 in Algorithm 4.2, performed until convergence between all directions within a specific group.

The computational cost of the neutronic solver may be decreased by optimizing the sweeping order of the directions. This has earlier been done for the steady-state solvers within FIRE [16].

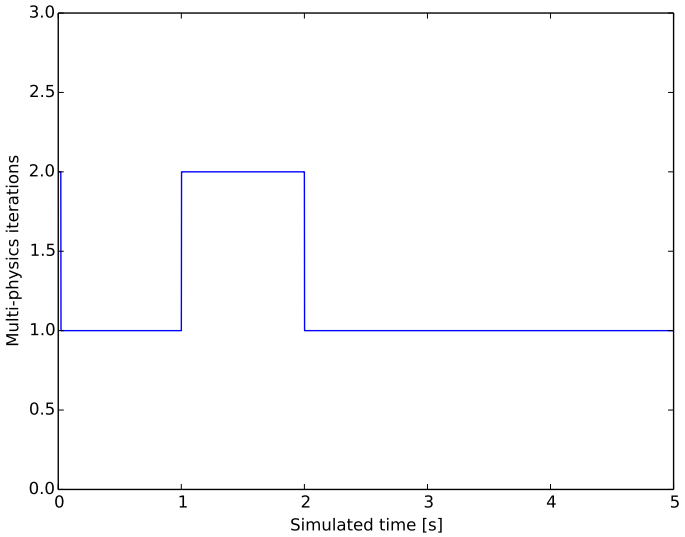


Figure 5.21: Number of multiphysics (i.e. outer) iterations over time.

A different possibility is to couple the directions within each group into a single matrix system. This looks attractive from a theoretical point of view since the S_N equation is completely linear. However, the treatment of such large matrix systems may cause problems in practice.

If successful, any of the above mentioned acceleration schemes would potentially lower the number of inner iterations needed to solve the neutronic problem, which could lower the computational cost of the neutronics.

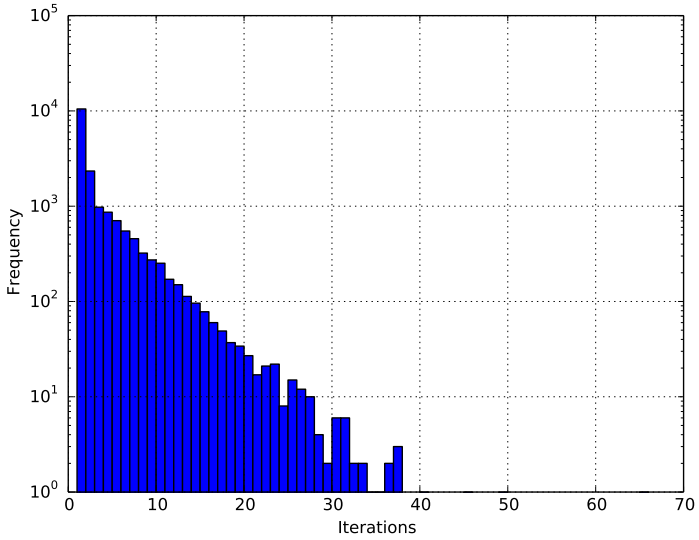


Figure 5.22: Histogram over the number of iterations between neutron groups needed to solve the neutronic problem, i.e. the number of times lines 2-13 in Algorithm 4.2 are executed for each call to the neutronic module.

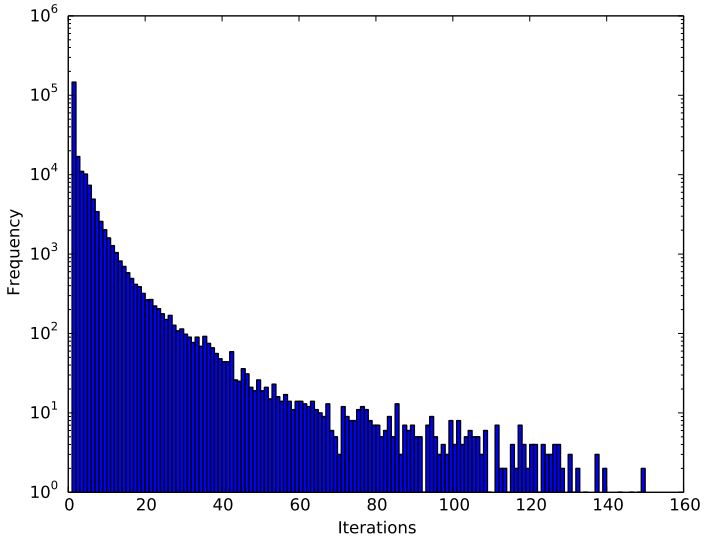


Figure 5.23: Histogram over the number of times lines 4-11 in Algorithm 4.2 are executed in order for each energy group to converge. The graph does not differentiate between energy groups.

Chapter 6

Conclusions and Outlook

In this thesis, a fine-mesh transient solver for the coupled neutronic/thermal-hydraulic problem in PWR fuel assemblies was presented.

Two alternative modules have been implemented to solve the neutronic problem; a diffusion solver and an S_N solver. Both are shown to exhibit the same qualitative transient behavior for inlet moderator temperature transients. Although qualitatively similar there exist clear differences between the two with respect to the heterogeneity they capture at material boundaries. This confirms that angularly resolved methods are needed to accurately predict the physics on the fuel pin scale.

The solver predicts that the heterogeneity of the moderator temperature during a uniform lowering of the inlet temperature increases significantly as a result of the transient conditions, even though the disturbance driving the transient is applied uniformly.

The solver accuracy is shown to be quite insensitive to time step choice, although convergence issues have been encountered under various circumstances, predominantly for large systems containing more than one fuel pin cell.

A greater sensitivity is found with regards to the mesh resolutions of the neutronics and thermal-hydraulics. It is seen that every doubling of the number of cells in each mesh causes the effective multiplication factor of the system to change significantly, i.e. the choice of mesh resolution has strong impacts on the criticality of the system. In order to perform a more quantitative mesh refinement study a boron criticality search should be done for each mesh resolution before comparisons can be made.

For most time steps in the presented simulations the multiphysics solver makes only one outer iteration with two iterations being the rare exception. This suggests that more complex nonlinear methods for the operator coupling might be unnecessary in order for the system of equations to behave well numerically, at least in the context of relatively homogeneous transients.

However, for some cases the S_N solver converges very slowly because it is presently not accelerated by any means. Judging by the substantial computational cost of running the coupled solver with the S_N module it should be possible to lower the computational cost by accelerating the solver in order to be able to run simulations on larger systems.

Within this project the underlying tools needed to apply control rod transients to the solver have been developed. It would be interesting to see how the transient fine-mesh physics would behave in more heterogeneous conditions, such as during control rod insertion or extraction. However, due to the mentioned problems with convergence of the S_N solvers for larger systems, such transients have at the present moment not been run. Acceleration of the S_N solver and application of control

rod driven or mediated transients would be examples of appropriate next steps in the development and test of this code.

The thermal-hydraulic solver is computationally relatively cheap compared to the S_N solver. Acceleration of the thermal-hydraulic solver is therefore not a primary concern at the present stage of development.

In testing the performance of the code, comparisons between different configurations are essential. Due to the strong effect of the effective multiplication factor on the system dynamics, systems should always be compared at equal reactivity levels. In order to eliminate differences in feedback effects between configurations they should also be started from the same thermal-hydraulic state and kept at a critical state in order to maintain the same state until starting the transient. A criticality search module would therefore be a highly desirable extension of the code capabilities.

In summary the implemented solver is demonstrated to capture the relevant multiphysics feedbacks and to give plausible results in the cases for which it has been tested. The multiphysics convergence works very well with the time steps required to converge the thermal-hydraulics within the presented methodology. There is, however, clear improvements to be made concerning the convergence rate of the neutronic module which would allow the solver to be applied to larger systems than is presently practical.

References

[1] Kord Smith and Benoit Forget. "Challenges in the development of high-fidelity LWR core neutronics tools". In: *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013)*. Sun Valley, Idaho, USA, 2013.

[2] Christophe Demazière. "Multi-physics modelling of nuclear reactors: current practices in a nutshell". In: *Int. J. Nuclear Energy Science and Technology* 7.4 (2013), pp. 288–318.

[3] FRAPCON/FRAPTRAN User Group. May 2014. URL: http://frapcon.labworks.org/ (visited on 06/10/2015).

[4] Studsvik AB. *SIMULATE5*. URL: http://www.studsvik.com/en/Business-Areas/Fuel-and-Materials-Technology/Nuclear-Fuel-Analysis-Software/In-Core-Fuel-Management/SIMULATE5/ (visited on 06/10/2015).

[5] Studsvik AB. *CASMO5*. URL: http://www.studsvik.com/en/Business-Areas/Fuel-and-Materials-Technology/Nuclear-Fuel-Analysis-Software/In-Core-Fuel-Management/CASMO5/ (visited on 06/10/2015).

[6] United States Nuclear Regulatory Commission. *TRACE*. URL: http://www.nrc.gov/about-nrc/regulatory/research/safetycodes.html\#th.

[7] Paul J. Turinsky. "Advances in multi-physics and high performance computing in support of nuclear reactor power systems modeling and simulation". In: *Nuclear Engineering and Technology* 44.2 (2012), pp. 103–122.

[8] *Consortium for Advanced Simulation of LWRs*. URL: www.casl.gov (visited on 06/10/2015).

[9] CD-adapco. *STAR-CCM+*. URL: http://www.cd-adapco.com/products/star-ccm%C2%AE.

[10] Han Gyu Joo et al. "Methods and Performance of a Three-Dimensional Whole-Core Transport Code". In: *PHYSOR 2004 -The Physics of Fuel Cycles and Advanced Nuclear Systems: Global Developments*. American Nuclear Society. Lagrange Park, Illinois, 2004.

[11] Los Alamos National Laboratory. *MCNP*. URL: https://mcnp.lanl.gov/ (visited on 06/10/2015).

[12] Karlsruher Institute of Technology. *Subchannel Code SUBCHANFLOW*. URL: http://www.inr.kit.edu/english/632.php.

[13] Bruno Chanaron et al. "Advanced multi-physics simulation for reactor safety in the framework of the NURESAFE project". In: *Annals of Nuclear Energy* In press (2015).

[14] Derek Gaston et al. "MOOSE: A parallel computational framework for coupled systems of nonlinear equations". In: *Nuclear Engineering and Design* 239.10 (2009), pp. 1768–1778.

[15] Derek Gaston et al. "Massive Hybrid Parallellism for Fully Implicit Multiphysics". In: *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013)*. American Nuclear Society. Sun Valley, Idaho, USA: LaGrange Park, IL, 2013.

[16] Klas Jareteg et al. "Coupled fine-mesh neutronics and thermal-hydraulics - Modeling and implementation for PWR fuel assemblies". In: *Annals of Nuclear Energy* 84 (2015), pp. 244–257.

[17] Klas Jareteg, Paolo Vinai, and Christophe Demazière. "Fine-mesh deterministic modelling of PWR fuel assemblies: proof-of-principle of coupled neutronic/thermal-hydraulic calculations". In: *Annals of Nuclear Energy* (2014).

[18] Klas Jareteg, Paolo Vinai, and Christophe Demazière. "Investigation of the possibility to use a fine-mesh solver for resolving coupled neutronics and thermal-hydraulics". In: *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013)*. Sun Valley, Idaho, USA, 2013.

[19] Klas Jareteg. "Development of an integrated deterministic neutronic/thermal-hydraulic model using a CFD solver". MA thesis. Chalmers University of Technology, 2012.

[20] OpenCFD Ltd (ESI Group). *OpenFOAM ®*. URL: `http://www.openfoam.com`.

[21] VTT Technical Research Centre of Finland. *Serpent: A continuous-energy Monte Carlo reactor physics burnup calculation code*. URL: `http://montecarlo.vtt.fi/`.

[22] Joel H. Ferziger and Milovan Peric. *Computational Fluid Dynamics*. 3rd ed. ISBN 3-540-42074-6. Springer-Verlag, 2002.

[23] *Paraview*. URL: `http://www.paraview.org/`.

[24] Anil K. Prinja and Edward W. Larsen. *General Principles of Neutron Transport*. Ed. by Dan Gabriel Cacuci. Handbook of Nuclear Engineering. Springer-Verlag, 2010.

[25] Christophe Demazière. *Physics of Nuclear Reactors (lecture notes)*. Chalmers University of Technology, 2012.

[26] Christophe Demazière. *Modelling of Nuclear Reactors (lecture notes)*. Chalmers University of Technology, 2014.

[27] Alain Hébert. *Multigroup Neutron Transport and Diffusion Computations*. Ed. by Dan Gabriel Cacuci. Handbook of Nuclear Engineering. Springer-Verlag, 2010.

[28] H.K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics*. 2nd ed. Pearson Education Limited, 2007.

[29] Ronald L. Panton. *Incompressible Flow*. John Wiley & Sons, 2013.

[30] Yunus Cengel, John M. Cimbala, and Robert H. Turner. *Fundamentals of Thermal-Fluid Sciences*. Fourth Edition. McGraw Hill, 2012.

[31] Lars Davidson. *An Introduction to Turbulence Models*. Publication 97/2. Chalmers University of Technology, 2015.

[32] Christopher J. Greenshields. *OpenFOAM Programmer's Guide*. 2015. URL: `http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf`.

[33] Christopher J. Greenshields. *OpenFOAM User's Guide*. 2015. URL: `http://sourceforge.net/docs/Guides-a4/UserGuide.pdf`.

[34] Klas Jareteg, Rasmus Andersson, and Christophe Demazière. "Development and test of a transient fine-mesh LWR multi-physics solver in a CFD framework". In: *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2015)*. Nashville, Tennessee, 2015.