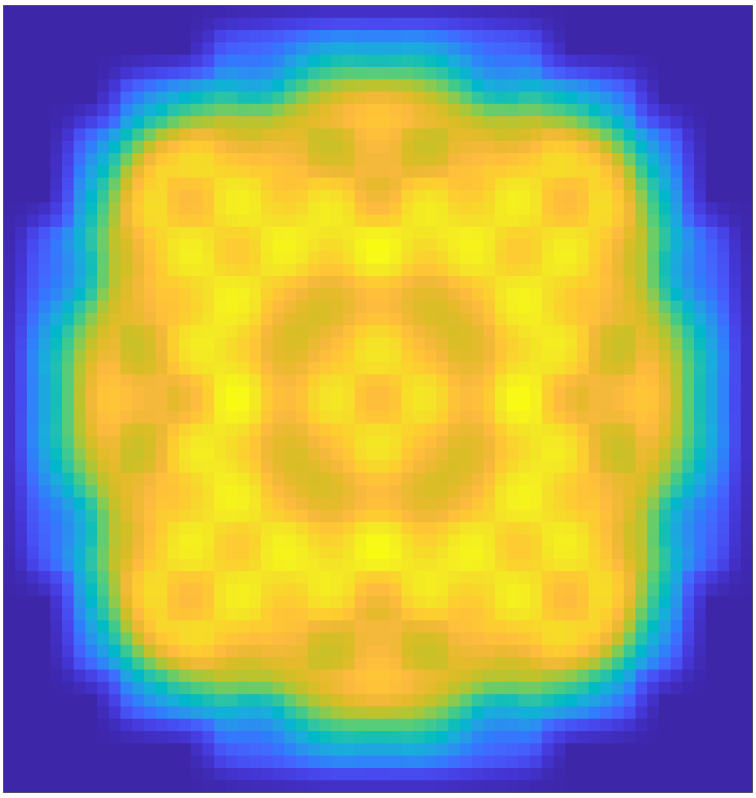
CHALMERS
UNIVERSITY OF TECHNOLOGY



Non-linear time-dependent modelling of heterogeneous nuclear reactors

Applications to Xenon oscillations in pressurized water reactors

Master's thesis in Physics

FREDRIK ÖHRLUND

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Non-linear time-dependent modelling of heterogeneous nuclear reactors

Applications to xenon oscillations in pressurized water reactors

FREDRIK ÖHRLUND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Division of Subatomic, High Energy and Plasma Physics
Chalmers University of Technology
Gothenburg, Sweden 2023

Non-linear time-dependent modelling of heterogeneous nuclear reactors
Applications to xenon oscillations in pressurized water reactors
FREDRIK ÖHRLUND

Supervisor & Examiner: Christophe Demazière

Master's Thesis 2023
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualization of the fast scalar neutron flux with a control rod perturbation engaged, constructed in Matlab using the graphics function `imagesc`.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Non-linear time-dependent modelling of heterogeneous nuclear reactors
Applications to xenon oscillations in pressurized water reactors
FREDRIK ÖHRLUND
Department of Physics
Chalmers University of Technology

Abstract

This thesis is about time-dependent core-level neutronic simulations of nuclear reactors. In such systems, the neutron flux may oscillate in space and time due to the production and consumption of given fission products, in particular Xenon-135. This phenomenon is known as Xenon oscillations. A numerical simulator was developed to model such a phenomenon. The simulator numerically solves large systems of non-linear equations, which in this case requires the use of implicit methods. The way this has been accomplished uses techniques that are different from standard practice. The tool capitalizes on a static solver earlier developed and uses the same approximations in terms of spatial discretization and boundary conditions. The simulator was demonstrated to produce physically-correct results when applied to a realistic model of a pressurized water reactor. The simulator allows for time-dependent perturbations defined in terms of insertions of control rod banks, and the realistic PWR model includes realistic control rod insertions. Xenon oscillations were predicted by the tool in some specific situations.

Keywords: Nuclear energy, Light water reactors, Xenon oscillations, Diffusion theory, Non-linear dynamics, Newton method.

Acknowledgements

I want to thank my supervisor and examiner Christophe Demazière for all the help and encouragement i have received throughout this Master's Thesis, and for the opportunity to do a project in nuclear reactor modelling. Doing this project has been a joyous and exciting endeavor from start to finish.

I want to thank Teknisk fysik, the Physics master's programme, the Department of Physics and Chalmers University of Technology for giving me a superb education.

These past few years of studying physics have been the best years of my life.

I dedicate this thesis to the love of my life, my wife Lisa Öhrlund.

Fredrik Öhrlund, Gothenburg, June 2023

Contents

List of Figures

List of Tables

1
Introduction

Nuclear power plants have been in large scale commercial use since the 1950's. The average age of all reactors in the world is approaching 40 years, and the great majority of all reactors in operation in the western world are Light Water Reactors, or LWRs [1], [2]. The behaviour of such reactors in normal operating and transient conditions is well mastered, through the extensive accumulated operating experience and thanks to the use of numerical simulations.

The physical phenomena underpinning the basic mechanism of energy production in a nuclear reactor, i.e. a self-sustaining nuclear chain reaction, are such that a nuclear reactor naturally prefers to be operated in steady-state conditions at all times [7, p.8]. In general therefore, nuclear reactors are designed for steady-state operation, where the control mechanisms, core geometry, fuel composition, etc, are all chosen with this goal in mind. Part of accomplishing this goal is of course to minimize, control or to otherwise avoid any transient behaviour of the reactor. There are however many parts of operating a reactor which unavoidably entail system transients.

Nuclear power plants in general, and the nuclear reactor cores contained within them in particular, are both systems with complex dynamics. The latter system has dynamics governed by the neutron transport equation. Numerically solving this equation is done using stochastic methods or deterministic methods. Deterministic methods, which are used in this thesis work, solve the governing balance equations by first discretizing them in the appropriate phase-space.

One important part of the dynamics inherent to the operation of most types of LWRs is the build up of Xenon-135, or ^{135}Xe, throughout the reactor core. ^{135}Xe is what is referred to as a neutron poison, having a very high absorption cross-section. It is produced in the core as part of the neutron chain reaction. Dealing with Xenon is an important albeit well understood part of standard operating procedure. However, there is a phenomena that can occur that is associated with changing the state of the reactor in unintended ways, called Xenon oscillations. Such oscillations are highly undesirable. The objective of this Master thesis project is to build software capable of numerically simulating the time-evolution of a typical LWR at the core level, and to design and implement perturbations of the system capable of causing Xenon oscillations.

The unintended ways of operating an LWR referenced above primarily involve changing the core power too frequently and with specific frequencies in time such that

oscillations of the ^{135}Xe concentration between different parts of the core become amplified. Xenon oscillations also require that there exists some kind of asymmetry in the Xenon atomic density distribution, perhaps caused by a partial insertion of control rods during power level adjustments. This is the reason why simulation of control rod insertion was chosen as the means of inducing Xenon oscillations in this project. One realistic scenario in which such conditions could arise is when reactors are operated in so-called load-following conditions i.e. where the core power is set to change according to the demands from the electrical grid. Lets briefly consider when load-following might be used in practice.

In many western countries today, a significant portion of all power produced comes from nuclear power plants. The country with the highest share of nuclear power generation in the world in France, where in 2021 about 68% all power came from nuclear. In Sweden, roughly 30% of all domestic energy production comes from nuclear, and together with hydroelectric power generation, it constitutes the base energy production of the country. Since domestic power consumption varies significantly on several different time scales, perhaps most prominently on the time scale of hours throughout the day and night cycle as well as on the time scale of weeks or months throughout the seasons of the year, any system of domestic power production must have the ability to vary its energy generation to match these variations in demand. To do this, France regularly operates reactors in load-following conditions. Moreover, the increasing share of energy production from intermittent sources, such as from wind power, further increase the variability of the demands put on the other sources of energy on the grid. The prospect of using load-following nuclear reactors has been considered in Sweden in order to meet this increasing demand of flexibility in energy production [3], [4], [5], [6].

1.1 Thesis Background & Objective

There are two topics that need to be discussed before the objective of this thesis can be stated, since they constrain the scope of the thesis in essential ways. The first topic is a software called `CORE SIM`, and the second topic is of the mathematical model used and its development.

1.1.1 About `CORE SIM`

In 2011, the task force on Deterministic REActor Modelling, the DREAM group, at Chalmers University of Technology released a neutronic tool called `CORE SIM` [9]. It is a `MatLab` based core-level neutronic simulator using two-group diffusion theory capable of handling a wide range of heterogeneous reactor cores in static and dynamic cases in the frequency domain (i.e. for stationary fluctuations). Both critical and subscritical systems with external neutron sources can be modelled. Furthermore, the eigenfunctions of the system can be estimated. `CORE SIM` uses finite differences for spatial discretization of the the reactor core. The reactor core is specified by the user through the input data, consisting of a set of arrays whose shape and values describe the geometry and materials properties of the core.

`CORE SIM` is not a tool for simulating the time-evolution of the state of a reactor core. It can however calculate the steady-state flux and effective multiplication factor $\phi_{eq}(\vec{\mathbf{r}}), k_{eff}$ which when paired with the input data suffices to initialize time-evolution. One of the goals of this thesis is to produce software capable of precisely such time-evolution simulations.

1.1.2 About the Mathematical Model

The DREAM group is currently working on developing analysis methods to support utilities in the determination of the occurrences of Xenon oscillations. PhD student Kristoffer Pedersen is currently working on physics based Reduced Order Modelling methods, or ROM methods, to help better understand Xenon oscillations as part of his PhD-thesis [8]. These ROM methods need to be verified by comparing their predictions with real data, taken from real nuclear reactors. Unsurprisingly, large quantities of such data is not readily available since nuclear reactors facilities have secrecy concerns to consider. In order to compare the ROM predictions to other methods, it was proposed that a time-dependent simulator for heterogeneous reactor cores could be used.

This idea led to this Master's Thesis. The mathematical model used in the simulator was therefore specifically chosen and designed such that the tool will complement Kristoffer Pedersen's work[1], while being general enough to warrant its own existence. Since Xenon oscillations is a phenomena occurring over time spans of tens of hours, and do not depend on many of the intricate details pertaining to the dynamics of nuclear reactor cores[2], the dynamical equations of such a system were simplified in important ways while still accurately describing the causes of such oscillations.

The mathematical model chosen is built from the standard diffusion equation using two energy groups for the neutron flux, without delayed neutrons as they are assumed to be in equilibrium at all times in the relevant time spans. The model includes the effect of ^{135}Xe on the reactor core through coupling to the thermal neutron flux macroscopic absorption cross-section. The mathematical model is also equipped with a feedback term that models some state-dependencies of the reactor. The mathematical model and all relevant details underpinning it are further discussed in detail in chapter 2. The model itself is presented in section 2.4.

1.1.3 Thesis Objective

The thesis objective is twofold:

- To develop a time-dependent three-dimensional heterogeneous solver in two-group diffusion theory for modelling xenon oscillations. This solver should

[1] In fact if time permits, the simulator built here will be in turn be used to build a purely data-based non-intrusive reduced order model to predict Xenon oscillations.

[2] Some examples of such disregarded details are the prompt neutron response and the thermo-hydraulics.

use the same input format, spatial discretization, spatial node enumeration and boundary conditions as the software `CORE SIM` [9] and should perform the time-integration based on the dynamical equations presented in section 2.4.
- To apply the solver to a realistic model of a Pressurized Water Reactor and generate time-signals with different time-dependent perturbations applied to the system.

To satisfy criteria in the first point, the algorithm for spatial discretization employed by `CORE SIM` has been used, and the node enumeration is done in an equivalent fashion to `CORE SIM` as well. To fulfill the criteria of applying the solver to a realistic model of a PWR and also to ensure compatibility with `CORE SIM` input data, one data-set fulfilling both of these points has been used throughout development of the time-dependent solver. The reactor model used throughout this work corresponds to the OECD/NEA and US NRC PWR MOX/UO2 core benchmark [16]. This data-set is from here on referred to as the development data-set[3].

1.2 Physics of Light Water Reactors

Many of the mechanisms involved in the energy production in LWRs are essentially similar to any other industrial application generating heat for electricity production. Some bulk mechanisms in an LWR can be simplistically described as follows. As the nuclear fuel in the reactor core produces heat, water flows past the fuel and heat transfers to the water, warming it up. This warmer water is subsequently used to drive a generator, thereby producing electricity. It is still arguably the case that nuclear power generation is importantly different from other technologies, essentially for one reason: the level of complexity involved in the phenomena causing the heat generation in the nuclear fuel.

Understanding the physics and dynamics of the heat producing nuclear chain reaction occurring in the fuel and all the intricate details pertaining to it is certainly not simple, at least not in the sense described above. In contrast to heating up a body of water, it involves pure quantum mechanical phenomena such a the fissioning of uranium-235 nuclei, and the build-up and subsequent consumption of hundreds of different fission fragments or fission products, which are the new materials that are produced as a result of said fission events. Depending on the material properties of these fission fragments and the current state of the reactor, i.e. the current fuel temperature, core flow, core pressure, etc, the behaviour of the entire reactor can be affected by the production of these new materials. One such fission product, the properties of which are of great importance in LWR operation in general and this thesis in particular, is the isotope ^{135}Xe. This so-called neutron poison is discussed in section 1.3.

There are two primary types of LWRs, Pressurized Water Reactors (PWRs) and

[3]Dr. Abdelhamid Dokhane from the Paul Scherrer Institute, Switzerland, is acknowledged for preparing the data of the OECD/NEA and US NRC PWR MOX/UO2 core benchmark used in this work.

Boiling Water Reactors (BWRs), both of which are so-called thermal reactors. This name refers to the fact that they rely on thermal neutrons, or slow neutrons, to induce new fission events and thereby sustain the chain reaction. Importantly, any such fission event itself produces fast neutrons, not thermal neutrons, and in an LWR fast neutrons alone cannot induce a sufficient number of new fission events. Therefore, these fast neutrons need to be slowed down somehow. In general, the process of slowing down fast neutrons to thermal speeds, so that they can cause new fission events, is called neutron moderation. This is done by the moderator, which in LWRs is the water that is flowing through the core. In essence the mechanism behind the neutron moderation, i.e. the mechanism that causes fast neutrons to slow down to thermal neutrons, is scattering. Water has a high scattering cross-section, meaning that as neutrons move through the water they repeatedly collide with water molecules, thereby losing energy.

While the thesis objective from section 1.1.3 only states the need to apply the solver to a realistic model of a PWR, there should be no issue applying it to inputs specifying a BWR as well. There are some properties that both are common to both types of LWR reactors that should be known about going forward:

- The geometry of an LWR reactor core is roughly cylindrical, where it makes sense to speak of axial layers of the reactor core. The axial and radial length scales are on the order of meters

- So-called control rods can be inserted into the reactor core from the top towards the bottom (for PWRs) or from the bottom towards the top (for BWRs). In either case the amount of insertion is called insertion level, and control rods are inserted in groups, called control rod banks.

1.3 Xenon Oscillations

Xenon oscillations are spatial oscillations of local power generation between regions in a reactor core. The oscillations themselves typically have a period between 15 and 30 hours and can therefore be difficult to detect. In the worst case scenario, the oscillations in local power could be of the sort that the total power level remains relatively stable, while the local power in either of the core regions becomes dangerously high at the peak of oscillation. Xenon oscillations are caused by several otherwise independent mechanisms which in certain circumstances can act together. To understand this, first the evolution chain of ^{135}Xe in a reactor is discussed together with its effect on the chain reaction. Second, using this information, an idealized situation in which oscillation would occur is presented.

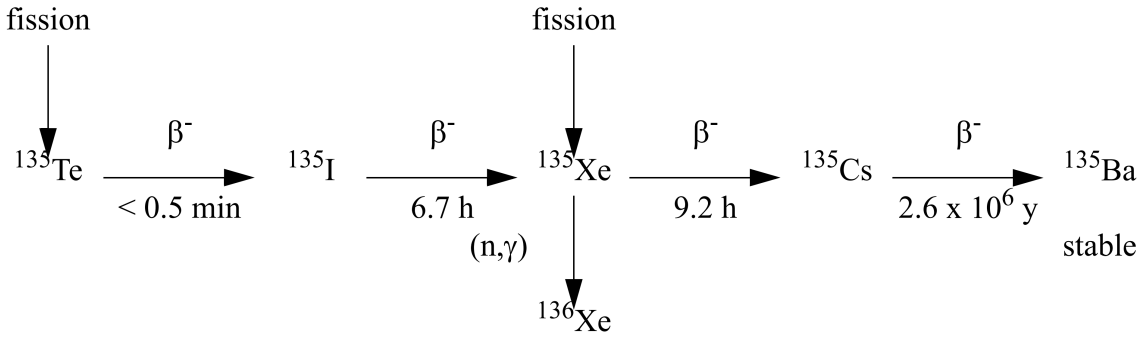


Figure 1.1: Evolution chain of ^{135}Xe. Image taken from [10, p.176] with expressed permission of the author.

The physical phenomena described in the following can all be found in the evolution chain shown in figure 1.1. The reason ^{135}Xe is called a neutron poison is because it has a very high microscopic capture cross-section of $\sigma_X = 2.7 \times 10^6$ barns at thermal energies. Such neutron captures by ^{135}Xe leads to ^{136}Xe, which itself does not matter for the neutron balance. This 'removal' of ^{135}Xe due to absorption is called burn-off. In other words, the higher the concentration of ^{135}Xe, the more the neutrons are absorbed by it instead of inducing new fission events, which itself also decreases the concentration of ^{135}Xe. The concentration of ^{135}Xe in the nuclear fuel is itself increased by fission, as both ^{135}Xe is created from fission, but also through β^- decay into ^{135}Xe of another fission fragment, Iodine-135, or ^{135}I[4]. In fact most ^{135}Xe from fission is due to decay of ^{135}I, since the fission yields of the isotopes are roughly $\gamma_X = 0.002$ and $\gamma_I = 0.062$ meaning only 0.2% of fissions result in ^{135}Xe while 6.2% lead to ^{135}I. The half-life of ^{135}I is roughly 6.7 hours, meaning that there is a delay between fission and the increase of the ^{135}Xe concentration it causes with a time scale of roughly 6.7 hours. Finally, ^{135}Xe undergo β^- decay into ^{135}Cs with a half-life of roughly 9.2 hours. While the Cesium is irrelevant here, this decay of course decreases the concentration of ^{135}Xe. These sources of decrease and increase in the ^{135}Xe concentration and the negative effect ^{135}Xe has on the nuclear chain reaction are relevant to Xenon oscillations, which we turn to now [10, p.173-176].

Xenon oscillations between different parts of the core would occur in the following situation. Consider a homogeneous LWR running steadily at full nominal power, when two local opposing perturbations that are equal in magnitude are applied simultaneously to the upper half, region I, and the bottom half, region II, respectively. The perturbations thus cause the amount of fission events taking place, governed by what is called the neutron flux (see section 2.1), in region I to increase instantaneously. The opposing perturbation similarly causes the flux in region II to decrease instantaneously. Such changes then instantaneously change the availability of neutrons which instantaneously increases the burn-off and thereby decreasing the ^{135}Xe concentration in region I, while the opposite happens in region II where the ^{135}Xe concentration starts increasing. Less ^{135}Xe in region I means less neutron poison

[4]Looking at figure 1.1 one notices that ^{135}I is actually obtained from the decay of Tellurium-135. The half-life of ^{135}Te is so small compared to ^{135}I that one can think of Iodine as being a fission fragment instead, thereby ignoring the Tellurium altogether.

that absorbs neutrons, which leads to increasing flux, and the opposite leads to decreasing flux in region II. Since the initial perturbation caused a change in the flux initially, and ^{135}Xe is produced due to fission with a delay of 6.7 hours through the decay of the initially produced ^{135}I, this causes an increase of the ^{135}Xe concentration with some delay, while the opposite happens in region II where the initial decrease of the flux leads to a decrease of the ^{135}Xe concentration with some delay. Since ^{135}Xe is a poison, less poison is thus made available in region I leading to a decrease in flux in region I, while more poison is made available in region II leading to an increase in flux. With some delay, the decrease in the flux in region I leads to an increase in the ^{135}Xe concentration in region I, while with some delay, the increase in neutron flux in region II leads to a decrease in the ^{135}Xe concentration in region II. This pattern continues, and is the phenomena called Xenon oscillations [10, p.178-180].

In less idealized situations, with more realistic perturbations such as control rod insertions, oscillation could occur with different phase shifts and amplitudes between different parts of the core, compared to what is described above. This is the case in the results shown in chapter 4.

2

Theory

In this chapter, the basic physical theory relevant to this project is presented, as well as the applicable approximations used. This entails defining some basic physical quantities and describing the neutron transport balance equation. The approximations described are the multi-group formalism, the diffusion approximation as well as the flux feedback term used to model the state-dependencies of the material properties of the reactor core. Furthermore, the coupled dynamics of the fission products ^{135}I and ^{135}Xe are described. Lastly, the dynamical equations obtained from applying the above approximations to the neutron transport equation as well as from coupling the flux, I-135 and Xe-135 equations, are presented.

2.1 Basic Physical Quantities

As mentioned in chapter 1, the dynamics of a reactor core are governed by the neutron transport equation [10, chapter 3.3]. The terms appearing in this equation are defined in terms of the neutron density, $n(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t)$. In the most general case it is time-, energy-, angle and position dependent. Therefore, the quantity $n(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t)\,\mathrm{d}^3\vec{\mathbf{r}}\,\mathrm{d}^2\vec{\mathbf{\Omega}}\,\mathrm{d}E$ gives the number of neutrons at time t, located within volume $\mathrm{d}^3\vec{\mathbf{r}}$ about $\vec{\mathbf{r}}$, with a direction within the solid-angle $\mathrm{d}^2\vec{\mathbf{\Omega}}$ about $\vec{\mathbf{\Omega}}$ and an energy within the bin $\mathrm{d}E$. The neutron density thus has units of number of neutrons per unit volume per unit solid angle per unit energy.

The neutron density is used to define the angular scalar neutron flux as

$$\psi(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t) \equiv v(E) n(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t) \tag{2.1}$$

where the neutron speed is calculated using $v(E) = \sqrt{\frac{2E}{m}}$. The angular scalar flux has units of neutrons per unit area per unit solid angle per unit energy per unit time. Just like the neutron density, this quantity has a seven dimensional phase space[1]. The angular scalar neutron flux is the primary unknown quantity of the neutron transport equation. However, before moving on to a discussion of the neutron transport equation itself, let us define two more quantities.

The scalar neutron flux is defined in terms of the angular scalar flux, averaged over the two-sphere S^2;

[1]Notice that the phase space is isomorphic to the space $(\vec{\mathbf{r}}, \vec{\mathbf{v}}, t)$, where the neutron velocity $\vec{\mathbf{v}}$ describes the same d.o.f information as the pair $(\vec{\mathbf{\Omega}}, E)$.

$$\phi(\vec{\mathbf{r}}, E, t) \equiv \int_{S^2} \psi(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t) \mathrm{d}^2\vec{\mathbf{\Omega}}, \tag{2.2}$$

It has units of number of neutrons per unit area per unit energy per unit time. The physical significance of this quantity on its own is somewhat unintuitive. One can think of it as the average flux of neutrons through all planes with all possible orientations through the point $\vec{\mathbf{r}}$.

The last quantity to be defined is the neutron current density vector. Much like the scalar neutron flux, it is also defined in terms of an integral over S^2 of the angular scalar neutron flux, but in this case the integrand is weighted by the vectorial direction $\vec{\mathbf{\Omega}}$;

$$\vec{\mathbf{J}}(\vec{\mathbf{r}}, E, t) \equiv \int_{S^2} \vec{\mathbf{\Omega}} \psi(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t) \mathrm{d}^2\vec{\mathbf{\Omega}} \tag{2.3}$$

This neutron current density vector has the same units as the scalar neutron flux, namely number of neutrons per unit area per unit energy per unit time.

2.2 Neutron transport

In words, the neutron transport equation is a balance equation involving phenomena that tend to increase and phenomena that tend to decrease the amount of neutrons at a point in space, at a moment in time, of neutrons with a specific energy and which are moving in a specific direction. Each term in the equation corresponds a specific phenomena that causes such an increase or decrease and which can be categorised as follows. Consider a small volume V bounded by surface $S \equiv \partial V$, and consider the balance of neutrons within V within a time interval $\mathrm{d}t$. Expressed in terms of the neutron density, this balance equation (per solid-angle and energy) reads

$$\begin{aligned} &\int_V \mathrm{d}V\, n(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t + \mathrm{d}t) - \int_V \mathrm{d}V\, n(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t) = \int_V \mathrm{d}V\, \mathrm{d}t\, \frac{\partial}{\partial t} n(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t) \\ &= + \text{production in V} - \text{disappearance in V} - \text{transfer through S} \end{aligned} \tag{2.4}$$

The above framework, color coding included, is illustrated in figure 2.1. The categories of production, disappearance and transfer/streaming will be referred to without further clarification from now on.

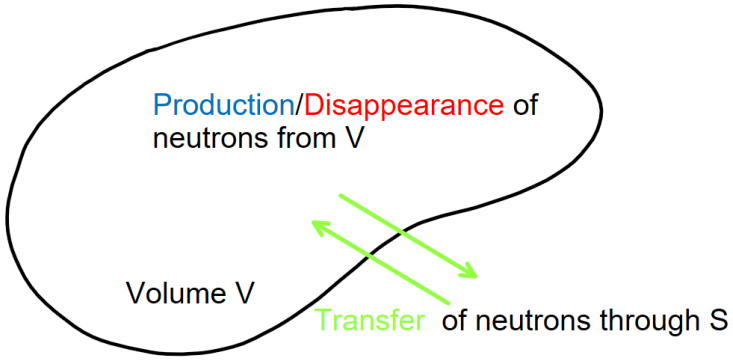


Figure 2.1: Illustration and color coding of the phenomenological categories of neutron balance. The distinction between production and disappearance might seem pointless at first, since one can be described as the negative of other. However, the distinction has some merit since certain phenomena, such as fission, can only ever contribute positively to the neutron balance, while other phenomena, such as scattering, can contribute positively or negatively.

The neutron transport equation is in no sense only applicable to nuclear reactors, but instead describes the motion and material interaction of neutrons in general. There are several equivalent formulations of it, including the integro-differential form [11, eq 2.29], the integral form [11, eq 2.42] and the characteristic form [11, eq 2.51]. The equation presented now is a version of the integro-differential formulation where all delayed neutron precursors are assumed to be in equilibrium at all times[2].

The neutron transport balance equation at phase space point $(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t)$, state[3]:

$$\begin{aligned} \left[\frac{1}{v(E)}\frac{\partial}{\partial t} + \vec{\mathbf{\Omega}} \cdot \vec{\boldsymbol{\nabla}} + \Sigma_t(\vec{\mathbf{r}}, E, t)\right]\psi(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t) = \\ = \frac{\chi(E)}{4\pi}\int_{\mathbb{R}^+} \nu_p(E')\Sigma_f(\vec{\mathbf{r}}, E', t)\phi(\vec{\mathbf{r}}, E', t)\,\mathrm{d}E' \\ + \int_{S^2}\int_{\mathbb{R}^+} \Sigma_s(\vec{\mathbf{r}}, E' \to E, \vec{\mathbf{\Omega}}' \to \vec{\mathbf{\Omega}}, t)\psi(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}', E', t)\,\mathrm{d}^2\vec{\mathbf{\Omega}}'\,\mathrm{d}E' \,. \end{aligned} \tag{2.5}$$

A proper interpretation of each term appearing in equation 2.5 will be given at the end of this chapter in section 2.5, alongside the presentation of the version of this equation that this thesis aims to solve. This final version of the equation is obtained from this fully general transport equation by the introduction of several approximation schemes, as well as by the explicit coupling of it to the dynamical equations describing the concentration of ^{135}Xe. For the remainder of this chapter therefore, the goal will be to present said approximation schemes and coupling, with the ultimate goals of arriving at, as well as motivating the validity of, the dynamical equations treated in this thesis.

[2]The referenced literature for the neutron transport equation, which is excellent, develops the theory of neutron transport together with the dynamics and coupling of the concentrations of precursors of delayed neutrons in an intimate fashion. When such precursors are in equilibrium, they are effectively absorbed into the fission term.

[3]An arbitrary source of neutrons $s(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t)$ could exist. This is not considered here.

2.3 Neutron transport approximation schemes

There are several aspects of the transport equation 2.5 that make numerical treatments of it difficult. Perhaps the most salient one being the seven-dimensional continuous phase space of the dependent variable. It is the purpose of both the multi-group formalism as well the diffusion approximation to mitigate this issue. The multi-group formalism replaces the continuum of E with a set of G energy groups, thereby increasing the number of equations to solve by the same factor. The diffusion approximation removes the dependence on $\vec{\mathbf{\Omega}}$ altogether by integrating any directional information over S^2, leaving only diffusive net neutron motion information behind. This reduces the phase space to the four dimensions of $(\vec{\mathbf{r}}, t)$, making numerical simulation much more feasible. Moreover, present in equation 2.5 are several factors describing material properties having phase space dependencies. These dependencies are here modeled by a flux feedback term as well as by the coupling to Xe-135. All of these topics are discussed in this section one-by-one followed by putting it all together at the end, where the dynamical equations that are to be solved are presented in detail.

2.3.1 Multi-group formalism

The multi-group formalism in itself is nothing but an energy-discretization, whereby the subspace $\mathbb{R}^+$ of the phase space is divided into a set of disjoint energy bins $B_g \equiv [E_g, E_{g-1}]$ such that they union to the subspace

$$\bigcup_{g=G}^{1} [E_g, E_{g-1}] = \mathbb{R}^+. \tag{2.6}$$

By convention the energy decreases with the index g and the bin indexed by g is referred to as energy group g. Notation-wise, the resulting energy-discretized variables follow the convention that $X(E \in [E_g, E_{g-1}]) \to X_g$ [11, chapter 3.1].

When the multi-group formalism is employed, the appropriate number of energy groups to use varies dramatically with the application. Primed by the discussion from section 1.2 on the mechanisms responsible for sustaining a nuclear chain reaction in LWRs, the prospect of using two energy groups, one for fast neutrons and one for thermal neutrons, hopefully seems intuitive and plausible. Such a two-group model is exactly what is used here. Thus, moving forward, the dependence on E of any previously seen variables, such as $\phi(\vec{\mathbf{r}}, E, t)$ and $\Sigma_a(\vec{\mathbf{r}}, E, t)$, can now be considered to have been exchanged for an index g, giving $\phi_g(\vec{\mathbf{r}}, t)$ and $\Sigma_{a,g}(\vec{\mathbf{r}}, t)$. The new quantities $\phi_1(\vec{\mathbf{r}}, t), \Sigma_{a,1}(\vec{\mathbf{r}}, t)$ will be referred to as the fast (scalar neutron) flux and the fast (macroscopic) absorption cross-section, respectively, while $\phi_2(\vec{\mathbf{r}}, t), \Sigma_{a,2}(\vec{\mathbf{r}}, t)$ will be referred to as the thermal flux and thermal absorption cross-section, respectively. Note that for two-group formalism the energies at the boundary of the two bins are $E_0 = \infty, E_1, E_0 = 0$, meaning that only the cut-off energy between fast and slow neutrons is free to be chosen. Intelligently choosing this cut-off allows one to assume that fission events only ever produce fast neutrons. This is also assumed to

be true.[4]

Transforming the transport equation from its continuous energy representation to a discrete energy representation using multi-group formalism should be done without changing the neutron balance. This can be done by preserving reaction rates, which are quantities of the quantities of the form

$$\Sigma_f(\vec{\mathbf{r}}, E, t)\phi(\vec{\mathbf{r}}, E, t) \quad \text{and} \quad \Sigma_s(\vec{\mathbf{r}}, E' \to E, \vec{\mathbf{\Omega}}' \to \vec{\mathbf{\Omega}}, t)\psi(\vec{\mathbf{r}}, \vec{\mathbf{\Omega}}, E, t), \tag{2.7}$$

as well as preserving neutron currents. Transforming to the multi-group transport equation thus entails, in principle at least, integrating the equation over each energy bin while adhering to the constraint that each macroscopic cross-section should be integrated with respect to the angular scalar flux on that bin. In practice however, due to several different drawbacks associated with using the angular flux as the weighing function, the scalar flux is used instead [11, p.74].

In general the group-averaged scalar flux ϕ_g, generic macroscopic cross-section $\Sigma_{\alpha,g}$ and inverse neutron speed $\frac{1}{v_g}$ can then be expressed in a multi-group formalism as[5]

$$\begin{aligned} \phi_g(\vec{\mathbf{r}}, t) &= \int_{E_g}^{E_{g-1}} \phi(\vec{\mathbf{r}}, E, t)\,\mathrm{d}E \\ \Sigma_{\alpha,g}(\vec{\mathbf{r}}, t) &= \frac{\int_{E_g}^{E_{g-1}} \Sigma_\alpha(\vec{\mathbf{r}}, E, t)\phi(\vec{\mathbf{r}}, E, t)\,\mathrm{d}E}{\phi_g(\vec{\mathbf{r}}, E, t)} \\ \frac{1}{v_g} &= \frac{\int_{E_g}^{E_{g-1}} \frac{1}{v(E)}\phi(\vec{\mathbf{r}}, E, t)\,\mathrm{d}E}{\phi_g(\vec{\mathbf{r}}, E, t)} \end{aligned} \tag{2.8}$$

In order to introduce some standard notation as well as to lay some ground work for the next section on diffusion, the expression for the group-averaged macroscopic scattering cross-section $\Sigma_s(\vec{\mathbf{r}}, E' \to E, \vec{\mathbf{\Omega}}' \to \vec{\mathbf{\Omega}}, t)$ is now given special attention. For isotropic media this cross-section only depends on $\mu \equiv \vec{\mathbf{\Omega}}' \cdot \vec{\mathbf{\Omega}}$. It can therefore be expanded in Legendre polynomials $P_l(\mu)$ as

$$\Sigma_s(\vec{\mathbf{r}}, E' \to E, \vec{\mathbf{\Omega}}' \to \vec{\mathbf{\Omega}}, t) = \Sigma_s(\vec{\mathbf{r}}, E' \to E, \mu, t) = \sum_{l \in \mathbb{N}} \frac{2l+1}{4\pi} \Sigma_{sl}(\vec{\mathbf{r}}, E' \to E, t) P_l(\mu) \tag{2.9}$$

where the expansion coefficients are

$$\Sigma_{sl}(\vec{\mathbf{r}}, E' \to E, t) = 2\pi \int_{-1}^{1} \Sigma_s(\vec{\mathbf{r}}, E' \to E, \mu, t) P_l(\mu)\,\mathrm{d}\mu \tag{2.10}$$

The first two coefficients, namely $\Sigma_{s0}(\vec{\mathbf{r}}, E' \to E, t)$ and $\Sigma_{s1}(\vec{\mathbf{r}}, E' \to E, t)$, are referred to as the isotropic scattering cross section and the anisotropic scattering

[4] This means that the neutron energy spectrum $\chi(E)$ that appear in the transport equation 2.5, which in two-group theory become χ_g, is assumed to satisfy $\chi_1 = 1, \chi_2 = 0$. Therefore, this quantity itself does not appear in the final equations. Only its effect of removing all production of thermal neutrons from fission remains.

[5] It should be mentioned that producing these "averaged" cross sections is quite involved [11, ch.3]

cross section, respectively. These two terms alone are often sufficiently accurate for reactor calculations, and can be used to approximate the total group-averaged cross section as

$$\Sigma_{s,g'\to g}(\vec{\mathbf{r}},\vec{\mathbf{\Omega}}'\to\vec{\mathbf{\Omega}},t)\approx\frac{1}{4\pi}[\Sigma_{s0,g'\to g}(\vec{\mathbf{r}},t)+3\mu\Sigma_{s1,g'\to g}(\vec{\mathbf{r}},t)] \tag{2.11}$$

where $\Sigma_{s0,g'\to g}(\vec{\mathbf{r}},t)$ and $\Sigma_{s1,g'\to g}(\vec{\mathbf{r}},t)$ are both evaluated using the given formula for group-averaged cross-sections. The meaning of the indices $g'\to g$ follows from the group-average formula for $\Sigma_s(\vec{\mathbf{r}},E'\to E,\vec{\mathbf{\Omega}}'\to\vec{\mathbf{\Omega}},t)$ [11, p.78-81].

2.3.2 The Diffusion approximation

The Diffusion approximation in reactor theory is the assumption that a certain relationship between ϕ and $\vec{\mathbf{J}}$, called Fick's law, holds true. The validity of this assumption, in turn, presupposes a long list of quite restrictive assumptions. However, derivations starting from these assumptions lead to results which, after further consideration, makes it clear that some of them can be relaxed, thereby extending the applicability of Fick's law and the diffusion equation derived with it [10, p.100-110], [14, p.125-132]. Fick's law states that[6]

$$\vec{\mathbf{J}}_g(\vec{\mathbf{r}},t)=-D_g(\vec{\mathbf{r}},t)\vec{\mathbf{\nabla}}\phi_g(\vec{\mathbf{r}},t), \tag{2.12}$$

where $D_g(\vec{\mathbf{r}},t)$ is called the diffusion coefficient of group g, having units of length. For the sake of completeness, at this point the definition of the group-averaged diffusion coefficient is given;

$$D_g(\vec{\mathbf{r}},t)=\frac{\int_{E_g}^{E_{g-1}}D(\vec{\mathbf{r}},E,t)\left\|\vec{\mathbf{\nabla}}\phi(\vec{\mathbf{r}},E,t)\right\|\mathrm{d}E}{\left\|\vec{\mathbf{\nabla}}\phi_g(\vec{\mathbf{r}},t)\right\|} \tag{2.13}$$

As was mentioned in the introduction to this section, applying the diffusion approximation to the transport equation entails integrating the equation over S^2, i.e. applying $\int_{S^2}\mathrm{d}^2\vec{\mathbf{\Omega}}$ to the equation. Looking at each term in equation 2.5, we see immediately that the fourth term just gets multiplied by 4π. Recalling the definition of the scalar neutron flux from equation 2.2, we see that the integration of any term whose sole dependence on $\vec{\mathbf{\Omega}}$ comes from $\psi_g(\vec{\mathbf{r}},\vec{\mathbf{\Omega}},t)$ has the effect of simply exchanging $\psi_g(\vec{\mathbf{r}},\vec{\mathbf{\Omega}},t)$ with $\phi_g(\vec{\mathbf{r}},t)$. This leaves only the second and fifth terms requiring further treatment.

To integrate the second term, note that $\vec{\mathbf{\Omega}}$ is independent of $\vec{\mathbf{r}}$ and recall the basic result from vector calculus $\vec{\mathbf{\nabla}}\cdot f(\vec{\mathbf{r}})\vec{\mathbf{A}}(\vec{\mathbf{r}})=\vec{\mathbf{A}}(\vec{\mathbf{r}})\cdot\vec{\mathbf{\nabla}}f(\vec{\mathbf{r}})+f(\vec{\mathbf{r}})\vec{\mathbf{\nabla}}\cdot\vec{\mathbf{A}}(\vec{\mathbf{r}})$, giving that $\vec{\mathbf{\Omega}}\cdot\vec{\mathbf{\nabla}}\psi_g(\vec{\mathbf{r}},\vec{\mathbf{\Omega}},t)=\vec{\mathbf{\nabla}}\cdot\vec{\mathbf{\Omega}}\psi_g(\vec{\mathbf{r}},\vec{\mathbf{\Omega}},t)$. We get that

$$\int_{S^2}\vec{\mathbf{\Omega}}\cdot\vec{\mathbf{\nabla}}\psi_g(\vec{\mathbf{r}},\vec{\mathbf{\Omega}},t)\,\mathrm{d}^2\vec{\mathbf{\Omega}}=\vec{\mathbf{\nabla}}\cdot\int_{S^2}\vec{\mathbf{\Omega}}\psi_g(\vec{\mathbf{r}},\vec{\mathbf{\Omega}},t)\,\mathrm{d}^2\vec{\mathbf{\Omega}}=\vec{\mathbf{\nabla}}\cdot\vec{\mathbf{J}}_g(\vec{\mathbf{r}},t), \tag{2.14}$$

[6] One of the initial assumptions that is eventually relaxed is that of steady-state conditions. Here the time-dependence is included from the start.

where equation 2.3 was used in the final step. Fick's law now gives

$$\vec{\boldsymbol{\nabla}} \cdot \vec{\mathbf{J}}_g(\vec{\mathbf{r}}, t) = -\vec{\boldsymbol{\nabla}} \cdot D_g(\vec{\mathbf{r}}, t)\vec{\boldsymbol{\nabla}}\phi_g(\vec{\mathbf{r}}, t) \tag{2.15}$$

This term is referred to as the streaming term in the diffusion equation. It physically corresponding to transfer type phenomena[7].

Integrating the sixth term can be done by assuming[8] both an isotropic medium as well as the further assumption of isotropic scattering in the lab frame [11, p.205]. Using the same Legendre expansion as was introduced in equation 2.9 this means $\Sigma_s(\vec{\mathbf{r}}, E' \to E, \vec{\boldsymbol{\Omega}}' \to \vec{\boldsymbol{\Omega}}, t) = \frac{1}{4\pi}\Sigma_{s0}(\vec{\mathbf{r}}, E' \to E, t)$. This of course makes the integration trivial;

$$\begin{aligned}
&\int_{S^2}\int_{S^2}\int_{\mathbb{R}^+} \Sigma_s(\vec{\mathbf{r}}, E' \to E, \vec{\boldsymbol{\Omega}}' \to \vec{\boldsymbol{\Omega}}, t)\psi(\vec{\mathbf{r}}, \vec{\boldsymbol{\Omega}}', E', t)\,\mathrm{d}^2\vec{\boldsymbol{\Omega}}\,\mathrm{d}^2\vec{\boldsymbol{\Omega}}'\,\mathrm{d}E' \\
&\quad = \int_{S^2}\int_{\mathbb{R}^+} \Sigma_{s0}(\vec{\mathbf{r}}, E' \to E, t)\psi(\vec{\mathbf{r}}, \vec{\boldsymbol{\Omega}}', E', t)\,\mathrm{d}^2\vec{\boldsymbol{\Omega}}'\,\mathrm{d}E' \\
&= \int_{\mathbb{R}^+} \Sigma_{s0}(\vec{\mathbf{r}}, E' \to E, t)\phi(\vec{\mathbf{r}}, E', t)\,\mathrm{d}E' = \sum_{g' \to g}^{G} \Sigma_{s0,g' \to g}(\vec{\mathbf{r}}, t)\phi_g(\vec{\mathbf{r}}, t)
\end{aligned} \tag{2.16}$$

At this point, every term in the transport equation 2.5 that is ultimately relevant to this thesis has been treated using the diffusion approximation scheme, giving us the diffusion equation. It is now is stated in its most general form, using general multi-group formalism, without precursors[9];

$$\begin{aligned}
&\left[\frac{1}{v_g}\frac{\partial}{\partial t} - \vec{\boldsymbol{\nabla}} \cdot D_g(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}} - \Sigma_{t,g}(\vec{\mathbf{r}}, t)\right]\phi_g(\vec{\mathbf{r}}, t) = \\
&\qquad \chi_g(\vec{\mathbf{r}}) \sum_{g'=1}^{G} \nu_{g'}(\vec{\mathbf{r}})\Sigma_{f,g'}(\vec{\mathbf{r}}, t)\phi_{g'}(\vec{\mathbf{r}}, t) + \\
&\qquad \sum_{g'=1}^{G} \Sigma_{s0,g' \to g}(\vec{\mathbf{r}}, t)\phi_g(\vec{\mathbf{r}}, t)
\end{aligned} \tag{2.17}$$

Further discussion of the terms in the equation is done in the final section of this chapter, when all details are in place.

It was mentioned at the beginning of this section that Fick's law is only valid under certain conditions, and in turn the diffusion equation as well. The assumptions made in the literature for the derivation of the diffusion equation are the following: and infinite and homogeneous medium without neutron sources, isotropic scattering in the lab frame, steady-state conditions and slowly varying flux as a function of position. These restrictions are however relaxable, such that Fick's law can be considered valid over distances of a few mean free paths of the neutrons, which is sufficient for the types of situations considered here [10, p.105-106], [11, ch.4.1].

[7] Looking back at figure 2.1 this term corresponds to all green type phenomena. This might seem intuitively clear, given its close resemblance to a standard Laplacian type term.

[8] It is mentioned that this way of introducing the diffusion approximation is equivalent to a more rigorous mathematical approach relying of first deriving the P1 approximation of the transport equation and making one further approximation on the anisotropic transfer of neutrons [11, ch 4.1]

[9] The source term that was mentioned in footnote 3 would now have the form $s_g(\vec{\mathbf{r}}, t)$.

2.3.3 Feedback term

Feedback coefficients, also called reactivity coefficients, can be used to model any variation of the material properties of the system, caused by variations of the state of the system. There are several standardized types of coefficients corresponding to cross-section changes due to changes in reactor core pressure, fuel temperature, gaseous voids in the coolant, and more [10, p.167].

The choice of feedback coefficients to approximate such material property dependencies is an important one, since it potentially allows for substantial reductions in computational cost and complexity without substantial loss in physicality, or because it is necessary to make a specific simulation possible when certain data is unavailable, etc. In essence, this is done by replacing the exact material property dependencies of the state within some region of the state space, with some superposition of so-called feedback terms which are linear or of low order in the feedback coefficient and the state variables.

The dynamical equations treated in this thesis employs a single feedback term with linear dependence on the flux $\phi_g(\vec{\mathbf{r}}, t)$. The feedback coefficient itself should be understood to have the general structure $\alpha_{g\to g'}(\vec{\mathbf{r}}) = -\frac{\partial\Sigma_{a,g'}(\vec{\mathbf{r}})}{\partial\phi_g} f_g(\vec{\mathbf{r}})$[10] where $f_g(\vec{\mathbf{r}})$ is an arbitrary shape function, here taken as the equilibrium state flux $f_g(\vec{\mathbf{r}}) \equiv \phi_{eq,g}(\vec{\mathbf{r}})$[11]. This incurs on the feedback term a functional dependence of the equilibrium state flux of the system. The value of $\frac{\partial\Sigma_{a,g'}(\vec{\mathbf{r}})}{\partial\phi_g}$ is assumed to be known and the feedback coefficient has units of a macroscopic cross section. Using all of the above, the flux feedback term can be written in full generality as

$$\alpha_{g\to g'}(\vec{\mathbf{r}})(\phi_g(\vec{\mathbf{r}}, t) - \phi_{eq,g}(\vec{\mathbf{r}})) = -\frac{\partial\Sigma_{a,g'}(\vec{\mathbf{r}})}{\partial\phi_g}\phi_{eq,g}(\vec{\mathbf{r}})(\phi_g(\vec{\mathbf{r}}, t) - \phi_{eq,g}(\vec{\mathbf{r}})), \tag{2.18}$$

The value of $\alpha_{g\to g'}(\vec{\mathbf{r}})$ describes the variation in reaction rate of energy group g' due to variation of the flux of energy group g. It should be noted that the second term in equation 2.18 is time-independent, contributing an affine term to the system of equations from the perspective of time-integration.

2.3.4 Xenon Dynamics & Coupling

This is the last component of the dynamical equations that needs to be explained. From a computational point of view it is arguably the most interesting component as well, since it provides the equations with a non-linear term. The coupling of the dynamics of the ^{135}Xe and ^{135}I concentrations to the scalar neutron flux comes into play as the time-dependence of the thermal macroscopic absorption cross-section

[10] Due to limitations in the development data-set, only a single number inputted as `dSIGa_dphi` to the solver have been available during development in place of the space and group dependent quantities $\frac{\partial\Sigma_{a,g'}(\vec{\mathbf{r}})}{\partial\phi_g}$. The software has been written with the general quantities in mind however, and in all following discussions of theory and numerical implementations the general quantity is considered.

[11] The steady-state or equilibrium state, $\phi_{eq,g}(\vec{\mathbf{r}})$, is further discussed in section 2.4.

$\Sigma_{a,2}(\vec{\mathbf{r}},t)$, modelling the effect of ^{135}Xe on the chain reaction as a neutron poison, an effect that was discussed in section 1.3. The macroscopic cross-section in energy-group g of reaction type α due to all present species labeled X with atomic concentrations $N_X(\vec{\mathbf{r}},t)$ and with associated α-reaction microscopic cross-section $\sigma_{\alpha X,g}$, can be generally expressed as [11, p.30]

$$\Sigma_{\alpha,g}(\vec{\mathbf{r}},t) = \sum_X N_X(\vec{\mathbf{r}},t)\sigma_{\alpha X,g}. \tag{2.19}$$

Since here only the effect of species ^{135}Xe on the thermal absorption macroscopic cross-section is considered, some simplifying notation is introduced. Using the notation

$$N_{^{135}\mathrm{Xe}}(\vec{\mathbf{r}},t) \equiv X(\vec{\mathbf{r}},t) \quad \text{and} \quad \sigma_{a^{135}\mathrm{Xe},2} \equiv \sigma_X, \tag{2.20}$$

and assuming that the thermal absorption cross-section of the fuel without any poisoning, denoted $\Sigma_{a,2,wox}(\vec{\mathbf{r}})$, is time-independent, the total time-dependent thermal absorption cross-section can be written

$$\Sigma_{a,2}(\vec{\mathbf{r}},t) = \Sigma_{a,2,wox}(\vec{\mathbf{r}}) + \sigma_X X(\vec{\mathbf{r}},t). \tag{2.21}$$

This coupling introduces the need to simulate the time-evolution of the concentration of ^{135}Xe. Given the discussion of the dynamics of ^{135}Xe from section 1.3 one realizes that those dynamics are themselves dynamically coupled to the concentration of ^{135}I. Using the quantities defined there, the dynamics of the Iodine concentration, denoted $I(\vec{\mathbf{r}},t)$, can be expressed as

$$\frac{\partial}{\partial t} I(\vec{\mathbf{r}},t) = \gamma_I \sum_g \Sigma_{f,g}(\vec{\mathbf{r}},t)\phi_g(\vec{\mathbf{r}},t) - \lambda_I I(\vec{\mathbf{r}},t), \tag{2.22}$$

where λ_I is the decay constant for ^{135}I to decay to ^{135}Xe, deducible from the known half-life of $T_{1/2} = 6.7$ hours as $\lambda_I = \frac{\ln 2}{T_{1/2}}$. Similarly, from the known half-life of ^{135}Xe being 9.2 hours, its decay constant can be deduced, denoted λ_X. Furthermore, with the discussion in section 1.3 still in mind, with special emphasis on the concept of the burn-off of the Xenon due to absorption which itself is described by the absorption reaction rate, the dynamical equation for the ^{135}Xe concentration is expressed as

$$\frac{\partial}{\partial t} X(\vec{\mathbf{r}},t) = \gamma_X \sum_g \Sigma_{f,g}(\vec{\mathbf{r}},t)\phi_g(\vec{\mathbf{r}},t) + \lambda_I I(\vec{\mathbf{r}},t) - \lambda_X X(\vec{\mathbf{r}},t) - \sigma_X X(\vec{\mathbf{r}},t)\phi_2(\vec{\mathbf{r}},t). \tag{2.23}$$

It should certainly be noted that this equation is non-linear if one considers $\phi_2(\vec{\mathbf{r}},t)$ to be an independent variable, due to the final term on the RHS. Equations 2.22 and 2.23 together constitute the complete dynamical equations of the ^{135}Xe concentration in the final model under consideration.

2.4 Dynamical Equations to be solved

The dynamical equations that this thesis pertains to are now presented. There is some new notation in the equations here that is clarified below. These equations

are obtained from the neutron transport equation 2.5 by introducing the two-group formalism[12], the diffusion approximation[13], the feedback term[14] and the Xenon coupling[15]. The equations state that:

$$\begin{aligned}
\frac{1}{v_1}\frac{\partial}{\partial t}\phi_1(\vec{\mathbf{r}},t) &= \Big[\vec{\boldsymbol{\nabla}}\cdot D_1(\vec{\mathbf{r}},t)\vec{\boldsymbol{\nabla}} + \nu\Sigma'_{f,1}(\vec{\mathbf{r}},t) - \Sigma_{a,1}(\vec{\mathbf{r}},t) - \Sigma_r(\vec{\mathbf{r}},t)\Big]\phi_1(\vec{\mathbf{r}},t) \\
&\quad + \nu\Sigma'_{f,2}(\vec{\mathbf{r}},t)\phi_2(\vec{\mathbf{r}},t) \\
&\quad + \alpha_{1\to1}(\vec{\mathbf{r}})(\phi_1(\vec{\mathbf{r}},t) - \phi_{eq,1}(\vec{\mathbf{r}})) + \alpha_{2\to1}(\vec{\mathbf{r}})(\phi_2(\vec{\mathbf{r}},t) - \phi_{eq,2}(\vec{\mathbf{r}})) \\
\frac{1}{v_2}\frac{\partial}{\partial t}\phi_2(\vec{\mathbf{r}},t) &= \Sigma_r(\vec{\mathbf{r}},t)\phi_1(\vec{\mathbf{r}},t) + \Big[\vec{\boldsymbol{\nabla}}\cdot D_2(\vec{\mathbf{r}},t)\vec{\boldsymbol{\nabla}} - \Sigma_{a,2,wox}(\vec{\mathbf{r}},t)\Big]\phi_2(\vec{\mathbf{r}},t) \\
&\quad + \alpha_{1\to2}(\vec{\mathbf{r}})(\phi_1(\vec{\mathbf{r}},t) - \phi_{eq,1}(\vec{\mathbf{r}})) + \alpha_{2\to2}(\vec{\mathbf{r}})(\phi_2(\vec{\mathbf{r}},t) - \phi_{eq,2}(\vec{\mathbf{r}})) \\
&\quad - \sigma_X X(\vec{\mathbf{r}},t)\phi_2(\vec{\mathbf{r}},t) \\
\frac{\partial}{\partial t}I(\vec{\mathbf{r}},t) &= \gamma_I\Sigma'_{f,1}(\vec{\mathbf{r}},t)\phi_1(\vec{\mathbf{r}},t) + \gamma_I\Sigma'_{f,2}(\vec{\mathbf{r}},t)\phi_2(\vec{\mathbf{r}},t) - \lambda_I I(\vec{\mathbf{r}},t) \\
\frac{\partial}{\partial t}X(\vec{\mathbf{r}},t) &= \gamma_X\Sigma'_{f,1}(\vec{\mathbf{r}},t)\phi_1(\vec{\mathbf{r}},t) + \gamma_X\Sigma'_{f,2}(\vec{\mathbf{r}},t)\phi_2(\vec{\mathbf{r}},t) + \lambda_I I(\vec{\mathbf{r}},t) - \lambda_X X(\vec{\mathbf{r}},t) \\
&\quad - \sigma_X X(\vec{\mathbf{r}},t)\phi_2(\vec{\mathbf{r}},t).
\end{aligned} \tag{2.24}$$

As per the thesis objective (see section 1.1.3), these equations are solved using Marshak boundary conditions;

$$\vec{\mathbf{J}}_g(\vec{\mathbf{r}}_b,t)\cdot\vec{\mathbf{n}}_b = \frac{1}{2}\phi_g(\vec{\mathbf{r}}_b,t), \quad \forall t, \tag{2.25}$$

where $\vec{\mathbf{r}}_b$ is a point on the system boundary and $\vec{\mathbf{n}}_b$ is the outwards normal at that point. This boundary condition is invoked as part of the spatial discretization, discussed in section 3.1. While solving for $\phi_g(\vec{\mathbf{r}},t)$, $I(\vec{\mathbf{r}},t)$ and $X(\vec{\mathbf{r}},t)$, all other quantities are known at all times[16]. It is mentioned at this point that any and all time-dependencies of the diffusion coefficients and cross-sections are strictly simultaneous step function changes due to insertion of withdrawal of control rod banks. In between such discrete changes to the system, they can all be thought of as time-independent with known values, and when denoted without time-dependence, such an intermediate fixed value is considered. The new notation appearing here is the effective multiplication factor -scaled fission cross-sections and the removal cross section, the latter defined in terms of the scattering cross-sections and the fluxes;

$$\Sigma'_{f,g}(\vec{\mathbf{r}},t) \equiv \frac{\Sigma_{f,g}(\vec{\mathbf{r}},t)}{k_{eff}} \tag{2.26}$$

$$\Sigma_r(\vec{\mathbf{r}},t) \equiv \Sigma_{s0,1\to2}(\vec{\mathbf{r}},t) - \frac{\Sigma_{s0,1\to2}(\vec{\mathbf{r}},t)\phi_2(\vec{\mathbf{r}},t)}{\phi_1(\vec{\mathbf{r}},t)} \tag{2.27}$$

[12] See section 2.3.1 and equations 2.8,2.11 and 2.13 in particular.
[13] See section 2.3.2 and equation 2.17 in particular.
[14] See section 2.3.3 and equation 2.18 in particular.
[15] See section 2.3.4 and equations 2.22 and 2.23 in particular.
[16] A caveat is that actually only $\Sigma_{a,2,eq}(\vec{\mathbf{r}}) \equiv \Sigma_{a,2,wox}(\vec{\mathbf{r}}) + \sigma_X X_{eq}(\vec{\mathbf{r}})$ is assumed known at all times. $\Sigma_{a,2,wox}(\vec{\mathbf{r}})$ has to be computed after solving for the steady-state and calculating $X_{eq}(\vec{\mathbf{r}})$.

To calculate the time-evolution according to equations 2.24, first the steady-state and effective multiplication factor $\phi_{eq,g}(\vec{\mathbf{r}})$ and k_{eff} are found by solving 2.24 with vanishing time derivatives and without any control rod perturbation. It is enough to solve for the equilibrium flux because the equilibrium concentrations $I_{eq}(\vec{\mathbf{r}})$ and $X_{eq}(\vec{\mathbf{r}})$ can then be found by direct calculation. Setting the time derivatives in 2.24 to zero, rearranging the equation and expressing the two flux equations in matrix form gives the equations for $\phi_{eq,g}(\vec{\mathbf{r}})$ and k_{eff} as the eigenvalue problem[17]:

$$\begin{aligned}
&\left(\begin{bmatrix}\vec{\boldsymbol{\nabla}}\cdot D_1(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}} & \\ & \vec{\boldsymbol{\nabla}}\cdot D_2(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}}\end{bmatrix} + \begin{bmatrix}-\Sigma_{a,1}(\vec{\mathbf{r}})-\Sigma_r(\vec{\mathbf{r}}) & \\ \Sigma_r(\vec{\mathbf{r}}) & -\Sigma_{a,2,eq}(\vec{\mathbf{r}})\end{bmatrix}\right)\begin{bmatrix}\phi_{eq,1}(\vec{\mathbf{r}})\\ \phi_{eq,2}(\vec{\mathbf{r}})\end{bmatrix} \\
&\qquad = -\frac{1}{k_{eff}}\begin{bmatrix}\nu\Sigma_{f,1}(\vec{\mathbf{r}}) & \nu\Sigma_{f,2}(\vec{\mathbf{r}})\\ & \end{bmatrix}\begin{bmatrix}\phi_{eq,1}(\vec{\mathbf{r}})\\ \phi_{eq,2}(\vec{\mathbf{r}})\end{bmatrix} \\
&\iff \left(\vec{\boldsymbol{\nabla}}\cdot\mathbf{D}(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}} + \boldsymbol{\Sigma}(\vec{\mathbf{r}})\right)\vec{\phi}_{eq}(\vec{\mathbf{r}}) = -\frac{1}{k_{eff}}\nu\mathbf{F}(\vec{\mathbf{r}})\vec{\phi}_{eq}(\vec{\mathbf{r}}) \\
&\iff \mathbf{M}(\vec{\mathbf{r}})\vec{\phi}_{eq}(\vec{\mathbf{r}}) = -\frac{1}{k_{eff}}\nu\mathbf{F}(\vec{\mathbf{r}})\vec{\phi}_{eq}(\vec{\mathbf{r}}) \\
&\implies -\mathbf{M}(\vec{\mathbf{r}})^{-1}\nu\mathbf{F}(\vec{\mathbf{r}})\vec{\phi}_{eq}(\vec{\mathbf{r}}) \equiv \mathbf{A}(\vec{\mathbf{r}})\vec{\phi}_{eq}(\vec{\mathbf{r}}) = k_{eff}\vec{\phi}_{eq}(\vec{\mathbf{r}})
\end{aligned} \tag{2.28}$$

In general, the operator $\mathbf{A}(\vec{\mathbf{r}}) = -\mathbf{M}(\vec{\mathbf{r}})^{-1}\nu\mathbf{F}(\vec{\mathbf{r}})$ has an infinite set of eigenvalue and eigenvector pairs $k_n, \vec{\phi}_n(\vec{\mathbf{r}})$. Solving this eigenvalue problem for the eigenvalue and eigenvector pair corresponding to the largest eigenvalue, say $k_0 > k_{n\geq 1}$, gives the fundamental mode[18] which is the steady-state flux and the effective multiplication factor [11, p.112]. With these quantities, the equilibrium concentrations of $I_{eq}(\vec{\mathbf{r}})$ and $X_{eq}(\vec{\mathbf{r}})$ are calculated from equation 2.24 with vanishing time derivatives, which after rearrangement gives the equilibrium concentrations as:

$$\begin{aligned}
I_{eq}(\vec{\mathbf{r}}) &= \frac{\gamma_I}{\lambda_I}\left[\frac{\Sigma_{f,1}(\vec{\mathbf{r}})}{k_{eff}}\phi_{eq,1}(\vec{\mathbf{r}}) + \frac{\Sigma_{f,2}(\vec{\mathbf{r}})}{k_{eff}}\phi_{eq,2}(\vec{\mathbf{r}})\right] \\
X_{eq}(\vec{\mathbf{r}}) &= \frac{\gamma_X}{\lambda_X + \sigma_X\phi_{eq,2}(\vec{\mathbf{r}})}\left[\frac{\Sigma_{f,1}(\vec{\mathbf{r}})}{k_{eff}}\phi_{eq,1}(\vec{\mathbf{r}}) + \frac{\Sigma_{f,2}(\vec{\mathbf{r}})}{k_{eff}}\phi_{eq,2}(\vec{\mathbf{r}})\right] + \frac{\lambda_I I_{eq}(\vec{\mathbf{r}})}{\lambda_X + \sigma_X\phi_{eq,2}(\vec{\mathbf{r}})}
\end{aligned} \tag{2.29}$$

At this point equation 2.24 can be solved for the time-evolution of $\phi_g(\vec{\mathbf{r}}, t), I(\vec{\mathbf{r}}, t)$ and $X(\vec{\mathbf{r}}, t)$. The numerical implementation of the steady-state solver and the time-dependent solver is the topic of the next chapter. For the sake of completeness, a brief discussion of the physical interpretation of the terms appearing in equations 2.24 and 2.28 is now given.

2.5 Interpretation & Discussion of terms

An interpretation of the terms appearing in equation 2.24 can be done by interpreting the effect of the matrix operators defined in the second step of equation

[17] The definition of the matrix quantities in the second step is given by matching terms between the first and second steps of the first equivalence. Furthermore, $\mathbf{M}(\vec{\mathbf{r}}) \equiv \left(\vec{\boldsymbol{\nabla}}\cdot\mathbf{D}(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}} + \boldsymbol{\Sigma}(\vec{\mathbf{r}})\right)$

[18] Only this solution is of interest here, motivating the use of k_{eff} and $\phi_{eq,g}(\vec{\mathbf{r}})$ in equation 2.28 instead of the general expression.

2.28 when acting on an arbitrary flux state $\vec{\phi}(\vec{\mathbf{r}},t) = [\phi_1(\vec{\mathbf{r}},t), \phi_2(\vec{\mathbf{r}},t)]^T$. The phenomenological categories of transfer, production and disappearance referred to here were introduced in equation 2.4 and figure 2.1.

The fission operator $\mathbf{F}(\vec{\mathbf{r}})$ gives the total local fission reaction rate $\mathbf{F}(\vec{\mathbf{r}})\vec{\phi}(\vec{\mathbf{r}},t)$. Multiplied by the average number of neutrons produced by fission ν, which is here assumed spatially invariant due to limitations in the development data-set, this operator gives the local production of neutrons due to fission of the state as $\nu\mathbf{F}(\vec{\mathbf{r}})\vec{\phi}(\vec{\mathbf{r}},t)$. When multiplied by the fission yield γ_I the fission operator gives the total local production of ^{135}I as $\gamma_I\mathbf{F}(\vec{\mathbf{r}})\vec{\phi}(\vec{\mathbf{r}},t)$. Similarly, $\gamma_X\mathbf{F}(\vec{\mathbf{r}})\vec{\phi}(\vec{\mathbf{r}},t)$ gives the total local production of ^{135}Xe. Regarding the scaled fission operator $\mathbf{F}'(\vec{\mathbf{r}}) = \frac{1}{k_{eff}}\mathbf{F}(\vec{\mathbf{r}})$, from a physical perspective this essentially changes the physical system such that it is exactly critical. Criticality is an important topic in reactor theory in general but it is not essential to this thesis, as all time-evolution simulations use the k_{eff} -scaled fission cross-sections meaning the system at the start of the simulations is always critical. For this reason, the topic of criticality is not discussed further in this report.

The streaming operator $\vec{\boldsymbol{\nabla}} \cdot \mathbf{D}(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}}$ gives the total local transfer of neutrons as $\vec{\boldsymbol{\nabla}} \cdot \mathbf{D}(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}}\vec{\phi}(\vec{\mathbf{r}},t)$, and is due to the motion of neutrons through the boundary of an infinitesimal volume about $\vec{\mathbf{r}}$. For computational considerations this term requires special attention due the the spatial derivatives. After spatial discretization using finite differences as in `CORE SIM` as discussed in section 3.1, the matrix that this operator corresponds to has a non-trivial and non-standard banding structure.

The operator[19] $\boldsymbol{\Sigma}(\vec{\mathbf{r}})$ gives the local production and disappearance of fast and thermal neutrons due to absorption and scattering as the two components of $\boldsymbol{\Sigma}(\vec{\mathbf{r}})\vec{\phi}(\vec{\mathbf{r}},t)$. From the discussion of the Legendre expansion coefficients (equation 2.10) and the definition of the removal cross section $\Sigma_r(\vec{\mathbf{r}})$ given in equation 2.27, it is clear that $\Sigma_r(\vec{\mathbf{r}})$ describe the net scattering of neutrons from the fast group to the thermal group.

The feedback term $\alpha_{g\to g'}(\vec{\mathbf{r}})(\phi_g(\vec{\mathbf{r}},t) - \phi_{eq,g}(\vec{\mathbf{r}}))$ linearly models the dependency of the macroscopic absorption cross-section $\Sigma_{a,g}(\vec{\mathbf{r}},t)$ on the current flux state relative to the equilibrium flux. The total core power $P = P(t)$ is a functional of the state variable $\vec{\phi}(\vec{\mathbf{r}},t)$ where $P[\vec{\phi}] = \int_{core} \mathrm{d}^3\vec{\mathbf{r}}\,\kappa(\vec{\mathbf{r}})\sum_{g=1}^{G}\Sigma_{f,g}(\vec{\mathbf{r}})\phi_g(\vec{\mathbf{r}},t)$. From this expression, it is easy to see that $P[\vec{\phi}] - P[\vec{\phi}_{eq}] = P[\vec{\phi} - \vec{\phi}_{eq}]$, and it follows that the current power level relative to equilibrium power at any one point in space at any one moment in time is directly proportional to $\phi_g(\vec{\mathbf{r}},t) - \phi_{eq,g}(\vec{\mathbf{r}})$. This gives some indication that this feedback will vary with the current power relative to steady-state power in such a way that it tends to draw the state towards the steady-state with a strength that increases with deviation from the steady-state.

[19] Unlike the fission operator and the streaming operator, this operator does not have a standardized name

3

Numerical Implementations

This chapter is divided into four main sections. The first section is primarily on spatial discretization, where `CORE SIM` is followed closely to ensure the requisite compatibility as per the thesis objective from section 1.1.3. The second section is on numerically solving the steady-state problem defined in equations 2.28 and 2.29. The third section is on numerically solving the time-dependent non-linear equations 2.24 in detail, taking up the bulk of this chapter. This section introduces some standardized mathematical formalism for solving large systems of non-linear equations. Time-discretization methods of the dynamical equations are discussed, with particular focus on the implicit Crank-Nicholson method and the implicit backward Euler method, both of which are implemented. The Newton iterations at the heart of the time-integration algorithm are discussed, with accompanying discussions of the residual vector $\vec{\mathbf{H}}(\vec{\mathbf{x}})$ of the time-discretized dynamical equations and the corresponding Jacobian $\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}})$, for any time-discretization method. The implementation of time-dependent perturbations in the form of control rod bank insertions are discussed. The inclusion of variable time-stepping is discussed. The fourth section is on the time-integration implementation in `MatLab`, and speaks in terms of code instead of mathematics. Pseudo-code for the time-integration algorithm is given, accompanied by a discussion of some of the details of algorithm.

3.1 Spatial Discretization Algorithm

The algorithm employed here is described in detail in the article published alongside the release of `CORE SIM` [9], and an even more detailed description is available in a document distributed together with `CORE SIM`. Therefore, only a basic outline of the algorithm that closely follows the article is given here.

The algorithm for spatial discretization presumes that any three-dimensional system consists of N adjacent volumes, or nodes, arranged in an array where node n has cartesian indexes (I, J, K), i.e. $n = n(I, J, K)$. The dynamical equations from section 2.4 are spatially-averaged on each of these nodes, resulting in systems of equations involving all nodes. As with the multi-group formalism from section 2.3.1, the definition of the node-averaged quantities ensure the preservation of reaction rates. Any dependence on $\vec{\mathbf{r}}$ is replaced by a dependence on node n where

$$\phi_{g,n}(t) \equiv \frac{1}{V_n} \int_{V_n} \phi_{g,n}(t)\, \mathrm{d}^3\vec{\mathbf{r}} \tag{3.1}$$

$$\Sigma_{g,n}(t) \equiv \frac{\frac{1}{V_n} \int_{V_n} \Sigma_{g,n}(t)\phi_{g,n}(t)\, \mathrm{d}^3\vec{\mathbf{r}}}{\phi_{g,n}(t)} \tag{3.2}$$

A valid[1] $3D$-system has $N_x \geq 3, N_y \geq 3$ and $N_z \geq 3$ nodes along each cartesian dimension and $N \leq N_x N_y N_z$ nodes in total. The node enumeration is generally done by iterating through the (I, J, K)-indices of the inputted arrays such that the IJ-planes are stored contiguously in memory and such that $K = 1, .., N_z$. This follows the standard that node enumeration in the z-direction goes from the inlet to the outlet of the reactor core.

The node-averaging transforms the steady-state equations 2.28 for the flux state variable $\vec{\phi}(\vec{\mathbf{r}}) = [\phi_1(\vec{\mathbf{r}}), \phi_2(\vec{\mathbf{r}})]^\top$ into a set of $2N$ coupled equations for the quantities $\{\phi_{1,n}\}_{n=1}^N$ and $\{\phi_{2,n}\}_{n=1}^N$. This motivates introducing notation for the total transformed flux state variable $\vec{\mathbf{x}} \in \mathbb{R}^{2N}$, defined in terms of the node enumeration as

$$\begin{aligned} \vec{\mathbf{x}} \equiv \begin{bmatrix} x_1 & \cdots & x_N & x_{N+1} & \cdots & x_{2N} \end{bmatrix}^\top &= \begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,N} & \phi_{2,1} & \cdots & \phi_{2,N} \end{bmatrix}^\top \\ &\iff x_{n+(g-1)N} = \phi_{g,n}, \begin{array}{c} 1 \leq n \leq N \\ g = 1, 2 \end{array} \end{aligned} \tag{3.3}$$

Later on when treating the time-dependent case, the node-averaged system state $\vec{\mathbf{x}}$ will be redefined to include the node-average transformed state variables $I(\vec{\mathbf{r}}, t)$ and $X(\vec{\mathbf{r}}, t)$.

The node-averaging induces a transformation of all terms in the equations found in section 2.4. Most terms in the dynamical equations 2.24 are identical in form to the ones found in the steady-state equations 2.28 and 2.29, allowing us to focus on the latter for now. The induced transformation of the matrix elements in equation 2.28 for any one node-enumeration trivially gives a diagonal matrix with the node-averaged macroscopic cross-section values along the diagonal. The only exception is the streaming operator, which is now given special attention.

By using a finite difference scheme, using the box-scheme[2], and by invoking Marshak boundary conditions (see equation 2.25) at the periphery of the system, the node-averaged streaming term can be approximated according to equation 3.4 using the so-called coupling coefficients $a_{g,n}^{\aleph}, b_{g,n}^{\aleph}, c_{g,n}^{\aleph}$ found in table 3.1;

$$\begin{aligned} -\frac{1}{V_n} \int_{V_n} \vec{\boldsymbol{\nabla}} \cdot D_g(\vec{\mathbf{r}}, t) \vec{\boldsymbol{\nabla}} \phi_g(\vec{\mathbf{r}}, t)\, \mathrm{d}^3\vec{\mathbf{r}} = \\ = \sum_{\aleph = x,y,z} \left[a_{g,n}^{\aleph}(t)\phi_{g,n}(t) + b_{g,n}^{\aleph}(t)\phi_{g,n+1}(t) + c_{g,n}^{\aleph}(t)\phi_{g,n-1}(t) \right] \end{aligned} \tag{3.4}$$

[1] $N_\aleph \geq 3$ is necessary for the spatial finite difference scheme.

[2] This is the assumption that the scalar neutron flux in the center of each node equals the node-averaged value

$b^{\aleph}_{g,n}$	$a^{\aleph}_{g,n}$	$c^{\aleph}_{g,n}$
If node $n-1$ does not exist		
$-\frac{2D_{g,n}}{(\Delta\aleph)^2}\frac{D_{g,n+1}}{D_{g,n}+D_{g,n+1}}$	$\frac{2D_{g,n}}{(\Delta\aleph)^2}\left(\frac{D_{g,n+1}}{D_{g,n}+D_{g,n+1}}+\frac{\Delta\aleph}{4D_{g,n}+\Delta\aleph}\right)$	0
If nodes $n-1$ and $n+1$ both exist		
$-\frac{2D_{g,n}}{(\Delta\aleph)^2}\frac{D_{g,n+1}}{D_{g,n}+D_{g,n+1}}$	$\frac{2D_{g,n}}{(\Delta\aleph)^2}\left(\frac{D_{g,n+1}}{D_{g,n}+D_{g,n+1}}+\frac{D_{g,n-1}}{D_{g,n}+D_{g,n-1}}\right)$	$-\frac{2D_{g,n}}{(\Delta\aleph)^2}\frac{D_{g,n-1}}{D_{g,n}+D_{g,n-1}}$
If node $n+1$ does not exist		
0	$\frac{2D_{g,n}}{(\Delta\aleph)^2}\left(\frac{D_{g,n-1}}{D_{g,n}+D_{g,n-1}}+\frac{\Delta\aleph}{4D_{g,n}+\Delta\aleph}\right)$	$-\frac{2D_{g,n}}{(\Delta\aleph)^2}\frac{D_{g,n-1}}{D_{g,n}+D_{g,n-1}}$

Table 3.1: Coupling coefficients in the $\aleph = (x, y, z)$-direction for node n. As seen in the RHS of equation 3.4, these coefficients are the elements of the matrix corresponding to the spatially discretized streaming operator $-\vec{\boldsymbol{\nabla}} \cdot \mathbf{D}(\vec{\mathbf{r}})\vec{\boldsymbol{\nabla}}$. For a general three-dimensional system, the resulting streaming matrix has non-trivial banding structure.

It should be mentioned that any and all matrices representing operators that act on the total system state $\vec{\mathbf{x}}$ have to be stored as sparse matrices at all times, including during initialization. This is by necessity due to RAM-constraints.

3.2 Steady-State Solver

Since all individual matrix elements appearing in the first equality in equation 2.28 except for the elements of streaming operator transform into diagonal matrices of size $N \times N$ and the elements of the transformed streaming operator can be found using table 3.1, it is clear that the same steps as in 2.28 give the transformed steady-state equations in the form of the eigenvalue problem $\mathbf{A}\vec{\mathbf{x}} = k\vec{\mathbf{x}}$, where the definition of terms is analogous to those introduced in the final equality of the equation. The matrix in this eigenvalue problem, $\mathbf{A} \in \mathbb{R}^{2N\times 2N}$, is thus $\mathbf{A} = (\mathbf{M}^{-1})(-\nu\mathbf{F})$.

As was mentioned in section 2.4, the steady-state solution $\vec{\mathbf{x}}_{eq}$ and effective multiplication factor k_{eff} corresponds to the pair of eigenvector and eigenvalue with the largest eigenvalue. To solve this system of equations one needs to be able to evaluate $\mathbf{M}^{-1}$, which is done by means of LU-decomposition of $\mathbf{M}$, such that

$$\mathbf{LU} = \mathbf{PMQ}, \tag{3.5}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are row and column permutation matrices, respectively. This preserves sparsity of the matrix $\mathbf{M}$ and allows its inverse to be evaluated through backward and forward substitutions as

$$\left(\mathbf{M}^{-1}\right)\vec{\mathbf{y}} = \mathbf{Q}\{\mathbf{U}\backslash[\mathbf{L}\backslash(\mathbf{P}\vec{\mathbf{y}})]\}. \tag{3.6}$$

Thus, the result of the matrix-vector operation $\mathbf{A}\vec{\mathbf{x}}$ can be evaluated as a vector function of any state $\vec{\mathbf{y}}$ as

$$\vec{\mathbf{A}}(\vec{\mathbf{y}}) \equiv \mathbf{Q}\{\mathbf{U}\backslash[\mathbf{L}\backslash(\mathbf{P}(-\nu\mathbf{F}\vec{\mathbf{y}}))]\} \tag{3.7}$$

As in `CORE SIM`, the LU-decomposition is done by calling `[L,U,P,Q] = lu(M)` in `MatLab` [18]. At this point the steady-state solver written here deviates from `CORE SIM`. The solution to the eigenproblem $\vec{\mathbf{x}}_{eq}, k_{eff}$ is obtained by calling the function `eigs` in `MatLab`, whereas `CORE SIM` uses a hand-coded Arnoldi-Krylov method to obtain an arbitrary number of higher eigenstates as well as the fundamental mode. To ensure compatibility with `CORE SIM`, a limited version of the Arnoldi-Krylov method for calculating only the fundamental mode has been implemented as well.

When the eigenproblem has been solved giving $\vec{\mathbf{x}}_{eq}, k_{eff}$, first the system is scaled to be critical, secondly the physical values of the steady-state fluxes are obtained by scaling them to the total reactor power through a functional relation $P[\vec{\mathbf{x}}_{eq}]$, and then thirdly the physical values for the node-averaged states corresponding to $I_{eq}(\vec{\mathbf{r}})$ and $X_{eq}(\vec{\mathbf{r}})$ are calculated according to equation 2.29. More specifically, the following procedure is applied:

The first step is to scale the fission cross-sections by the effective multiplication factor according to equation 2.26:

$$\Sigma'_{f,g,n} \equiv \frac{1}{k_{eff}} \Sigma_{f,g,n} \tag{3.8}$$

The second step is to enforce that the total power produced by the reactor in the steady state equals full nominal power `Pwr`. This gives the power-scaling coefficient `PwrScaling` as:

$$\begin{aligned} \texttt{PwrScaling} &= \frac{\texttt{Pwr}}{\sum_{n=1}^{N} \kappa_n V_n \sum_{g=1}^{2} \Sigma'_{f,g,n} x_{eq,n+(g-1)N}} \\ \implies \vec{\phi}_{eq} &\equiv \texttt{PwrScaling} \times \vec{\mathbf{x}}_{eq}, \end{aligned} \tag{3.9}$$

where $V_n = D_x D_y D_z$ and κ_n are the volume and steady-state power per fission event of node n, respectively. Here, all of `Pwr`,D_x, D_y, D_z and ${\kappa_n}^3$ are input-data.

The third and final step of the steady-state solver is to find the node-averaged values of the concentrations of ^{135}I and ^{135}Xe from equations 2.29, which at this point have been transformed into

$$\begin{aligned} I_{eq,n} &= \frac{\gamma_I}{\lambda_I} \left[\Sigma'_{f,1,n} \phi_{eq,1,n} + \Sigma'_{f,2,n} \phi_{eq,2,n} \right] \\ X_{eq,n} &= \frac{\gamma_X \left[\Sigma'_{f,1,n} \phi_{eq,1,n} + \Sigma'_{f,2,n} \phi_{eq,2,n} \right] + \lambda_I I_{eq,n}}{\lambda_X + \sigma_X \phi_{eq,2,n}}. \end{aligned} \tag{3.10}$$

3.3 Time-dependent Solver

The general total state $\vec{\mathbf{x}}$ of the reactor core, which was originally defined in equation 3.3, is now re-defined by straight-forwardly appending to it the node-averaged values

[3] For clarity, as $\nu\Sigma_{f,g,n}$ is provided as input, the expected data is actually $\frac{\kappa_n}{\nu}$.

of the concentrations of ^{135}I, denoted I_n and ^{135}Xe denoted X_n, in that order. The general dynamical state vector $\vec{\mathbf{x}}(t) \in \mathbb{R}^{4N}$ obtained is thus defined as

$$\begin{aligned} \vec{\mathbf{x}}(t) &\equiv \begin{bmatrix} x_1(t) & \cdots & x_{4N}(t) \end{bmatrix}^\top \\ &= \begin{bmatrix} \cdots & \phi_{1,n}(t) & \cdots & \phi_{2,n}(t) & \cdots & I_n(t) & \cdots & X_n(t) & \cdots \end{bmatrix}^\top \\ &\iff x_{n+(k-1)N}(t) = \begin{bmatrix} \phi_{1,n}(t), & k=1 \\ \phi_{2,n}(t), & k=2 \\ I_n(t), & k=3 \\ X_n(t), & k=4 \end{bmatrix}, \quad 1 \leq n \leq N. \end{aligned} \tag{3.11}$$

To simplify expressions when appropriate, the equivalent decomposition of the total dynamical state $\vec{\mathbf{x}}(t)$ into each of the four dynamical variables $\vec{\mathbf{x}}_i(t) \in \mathbb{R}^N, i = 1,2,3,4$ will be used, where

$$\vec{\mathbf{x}}(t) = \begin{bmatrix} \vec{\mathbf{x}}_1(t) & \vec{\mathbf{x}}_2(t) & \vec{\mathbf{x}}_3(t) & \vec{\mathbf{x}}_4(t) \end{bmatrix}^\top = \begin{bmatrix} \vec{\phi}_1(t) & \vec{\phi}_2(t) & \vec{\mathbf{I}}(t) & \vec{\mathbf{X}}(t) \end{bmatrix}^\top. \tag{3.12}$$

Following the notational standard for time-dependent non-linear systems established in [12, ch.3], the system dynamics that are to be time-integrated can in their most general form be written in terms of a relation between the time-derivative of the system state and a vector-valued function of the state as well as time;

$$\frac{\mathrm{d}\vec{\mathbf{x}}(t)}{\mathrm{d}t} = \vec{\mathbf{F}}(\vec{\mathbf{x}}(t), t). \tag{3.13}$$

Any eventual linear part of these dynamics, described by a linear time-dependent operator[4] $\mathbf{L}(t)$, can be separated from any non-linear part denoted $\vec{\mathbf{N}}(\vec{\mathbf{x}}(t), t)$, giving the general decomposition [12, p.19]

$$\vec{\mathbf{F}}(\vec{\mathbf{x}}(t), t) = \mathbf{L}(t)\vec{\mathbf{x}}(t) + \vec{\mathbf{N}}(\vec{\mathbf{x}}(t), t). \tag{3.14}$$

Recalling the discussions from sections 2.4, 3.1 and 3.2 and looking back at the general expressions for the dynamical equations 2.24 where the first and second equations are now multiplied by v_1 and v_2, respectively, it is clear that all terms appearing in the system of equations are linear except for the non-linear coupling terms[5] $-v_2\sigma_X\vec{\phi}_2(t) \odot \vec{\mathbf{X}}(t)$ and $-\sigma_X\vec{\phi}_2(t) \odot \vec{\mathbf{X}}(t)$ in the second and fourth equations and the affine feedback terms $\alpha_{g\to g'}\left(\vec{\phi}_g(t) - \vec{\phi}_{eq,g}(t)\right)$ in the first and second equations. Furthermore, all individual linear terms appearing in equations 2.24 except the streaming terms are transformed into diagonal matrices of size $N \times N$, due to the node-averaging and the streaming operator matrix elements can be found using table 3.1 just as before. These terms taken together give rise to $4N \times 4N$ matrices, which except for the streaming matrix is filled with blocks of diagonal $N \times N$-matrices. For now it is stated as a fact that the linear operator $\mathbf{L}(t) \in \mathbb{R}^{4N\times 4N}$ is explicitly known at all times from the input-data alone, where discussion of the

[4]It should perhaps be mentioned that the sparsity pattern of this operator is time-independent and is always highly sparse, having only 0.0043% nonzero elements for the development data-set.

[5]The Hadamard product $\vec{\mathbf{w}} = \vec{\mathbf{v}} \odot \vec{\mathbf{u}}$ is just element-wise multiplication, corresponding to `.*` in `MatLab`.

time-dependent perturbations simulating control rod insertions, which are causing all time-dependencies of $\mathbf{L}$, are postponed until section 3.3.3.

By choice, the affine feedback term is grouped together with the properly non-linear coupling terms in the total non-linear contribution to equation 3.14, which evidently lacks any explicit time-dependence, i.e. $\vec{\mathbf{N}}(\vec{\mathbf{x}}, t) = \vec{\mathbf{N}}(\vec{\mathbf{x}})$. This term is expressed as a matrix $\mathbf{N} \in \mathbb{R}^{4N\times 3N}$ multiplied by a vector $\vec{\mathbf{f}}(\vec{\mathbf{x}}(t)) \in \mathbb{R}^{3N}$, the latter containing essentially all non-linearity of the system dynamics and is defined as

$$\vec{\mathbf{f}}(\vec{\mathbf{x}}(t)) \equiv \begin{bmatrix}(\vec{\mathbf{x}}_1(t) - \vec{\mathbf{x}}_{eq,1}) & (\vec{\mathbf{x}}_2(t) - \vec{\mathbf{x}}_{eq,2}) & \vec{\mathbf{x}}_2(t) \odot \vec{\mathbf{x}}_4(t)\end{bmatrix}^\top. \tag{3.15}$$

All nonzero elements of the matrix $\mathbf{N}$ then consist of six different diagonal matrices $\mathbf{N}_{pq} \in \mathbb{R}^{N\times N}$, where $\mathbf{N}_{11}, \mathbf{N}_{12}, \mathbf{N}_{21}$ and $\mathbf{N}_{22}$ have the appropriate feedback coefficients along their diagonals while $\mathbf{N}_{23}$ and $\mathbf{N}_{43}$ are both simply the identity matrix scaled by the appropriate factors for the two coupling terms, i.e

$$\begin{aligned}\mathbf{N}_{gg'} &= \mathrm{diag}(\begin{bmatrix}\cdots & v_g\alpha_{g'\to g,n} & \cdots\end{bmatrix}), \quad g, g' = 1, 2\\ \mathbf{N}_{23} &= -v_2\sigma_X \mathbb{1}_{N\times N} \quad \text{and} \quad \mathbf{N}_{43} = -\sigma_X \mathbb{1}_{N\times N}.\end{aligned} \tag{3.16}$$

The resulting expression for the non-linear term is:

$$\vec{\mathbf{N}}(\vec{\mathbf{x}}, t) = \mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}(t)) = \begin{bmatrix}\mathbf{N}_{11} & \mathbf{N}_{12} & \\ \mathbf{N}_{21} & \mathbf{N}_{22} & \mathbf{N}_{23} \\ & & \\ & & \mathbf{N}_{43}\end{bmatrix}\begin{bmatrix}\vec{\mathbf{x}}_1(t) - \vec{\mathbf{x}}_{eq,1}\\ \vec{\mathbf{x}}_2(t) - \vec{\mathbf{x}}_{eq,2}\\ \vec{\mathbf{x}}_2(t) \odot \vec{\mathbf{x}}_4(t)\end{bmatrix} \tag{3.17}$$

With the time-dependence of $\vec{\mathbf{x}}$ suppressed, equation 3.14 expressed in these new quantities become

$$\vec{\mathbf{F}}(\vec{\mathbf{x}}, t) = \mathbf{L}(t)\vec{\mathbf{x}} + \mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}). \tag{3.18}$$

Now we turn to the issue of time-integration of equation 3.13, starting with the time-discretization.

3.3.1 Time-discretization methods

Discrete times are denoted t_m and the time-step at time m is here defined as $dt_m \equiv t_{m+1} - t_m$. The time-step denotes finite quantities. Any quantity evaluated at time t_m is denoted $y(t_m) = y^m$. A generic form of the time-discretized equation 3.13 can be written using a first-order expansion of the time-derivative and a weighted average of the RHS evaluated at t_{m+1} and t_m as

$$\frac{\vec{\mathbf{x}}^{m+1} - \vec{\mathbf{x}}^m}{dt_m} = \theta\vec{\mathbf{F}}^{m+1} + (1-\theta)\vec{\mathbf{F}}^m \tag{3.19}$$

where the weighting parameter $\theta \in [0, 1]$ specifies the time-discretization method itself [12, p.19].

Here, both the Crank-Nicholson method where $\theta = \frac{1}{2}$ as well as the backward Euler method where $\theta = 1$ have been implemented. Both are implicit methods and both

are handled with the same method of time-integration, namely Newton iterations, as in equation 3.25. The systems of non-linear equations that need to be solved in each time-step for each of the methods are presented in the following, together with a brief discussion of their respective properties. Except for some very specific tasks, the Crank-Nicholson scheme turns out to be generally preferable here and can be considered the primary method used for time-integration.

3.3.1.1 The Crank-Nicholson method

For $\theta = \frac{1}{2}$, equation 3.19 expanded using equation 3.18 gives

$$\left[\mathbb{1} - \frac{dt_m}{2}\mathbf{L}^{m+1}\right]\vec{\mathbf{x}}^{m+1} - \frac{dt_m}{2}\mathbf{N}\vec{\mathbf{f}}^{m+1} = \left[\mathbb{1} + \frac{dt_m}{2}\mathbf{L}^{m}\right]\vec{\mathbf{x}}^{m} + \frac{dt_m}{2}\mathbf{N}\vec{\mathbf{f}}^{m} \tag{3.20}$$

The Crank-Nicholson method is second order accurate in time [11, p.181]. This allows the use of large time-steps dt_m such that typical Xenon oscillation simulations have execution times of no more than a few minutes. This is why Crank-Nicholson is preferred here.

Crank-Nicholson is vulnerable to slowly damped numerical oscillations, particularly when a system undergoes a very sharp transient [15]. The control rod perturbations, discussed in section 3.3.3, strongly suffer from this problem. It is crucial for the sake of the simulation as a whole that this problem is alleviated somehow. The implemented solution to this problem comes in the form of variable time-steps and variable time-discretization method, and this is discussed in section 3.3.4.

3.3.1.2 The backward Euler method

For $\theta = 1$, equation 3.19 expanded using equation 3.18 gives

$$\left[\mathbb{1} - dt_m\mathbf{L}^{m+1}\right]\vec{\mathbf{x}}^{m+1} - dt_m\mathbf{N}\vec{\mathbf{f}}^{m+1} = \vec{\mathbf{x}}^{m} \tag{3.21}$$

The backward Euler method, also referred to as simply the Implicit method, is only first order accurate in time. For physical accuracy to be guaranteed, this method requires time-steps dt_m so small that the execution time of typical Xenon oscillation simulations is on the order of days. However, this method is immune to numerical oscillations, and it can be used to dampen such oscillations. This is further discussed in section 3.3.4.

3.3.2 Time-integration method

Time-integration of the system dynamics specified by equation 3.13 using any time-discretization method parameterized by θ, corresponds to repeatedly solving the system of equations 3.19 to obtain the future state $\vec{\mathbf{x}}^{m+1}$ given the present state $\vec{\mathbf{x}}^{m}$ and the time-step dt_m. The algorithm that has been produced for this purpose relies on expanding the residual $\vec{\mathbf{H}}(\vec{\mathbf{x}}^{m+1})$ of equation 3.19 to first order about the current state $\vec{\mathbf{x}}^{m}$. Using that $\vec{\mathbf{H}}(\vec{\mathbf{x}}^{m+1}) = 0$ and then iteratively solving the obtained linear

equation involving the Jacobian $\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}})$ for $\delta\vec{\mathbf{x}}$, the solution $\vec{\mathbf{x}}^{m+1} = \vec{\mathbf{x}}^m + \delta\vec{\mathbf{x}}$ is obtained. These iterations constitute the foundation of the produced time-integration algorithm and are therefore discussed in more detail now.

Equation 3.19 can be rearranged to $\vec{\mathbf{x}}^{m+1} - \theta dt_m \vec{\mathbf{F}}^{m+1} = \vec{\mathbf{x}}^m + (1-\theta)dt_m \vec{\mathbf{F}}^m$ for any time-discretization method parameterized by θ. The RHS, now denoted $\vec{\mathbf{b}} = \vec{\mathbf{b}}(\vec{\mathbf{x}}^m, dt_m, \theta)$, is independent of the future state and can thus be treated as a constant vector when solving this equation for $\vec{\mathbf{x}}^{m+1}$. Moving the RHS of this equation to the LHS and expanding terms using equation 3.18 gives the defining expression for the residual vector of this problem

$$\vec{\mathbf{H}}(\vec{\mathbf{x}}) \equiv \left[\mathbb{1} - \theta dt_m \mathbf{L}^{m+1}\right]\vec{\mathbf{x}} - \theta dt_m \mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}) - \vec{\mathbf{b}}. \tag{3.22}$$

Clearly $\vec{\mathbf{H}}(\vec{\mathbf{x}})\Big|_{\vec{\mathbf{x}}=\vec{\mathbf{x}}^{m+1}} = \vec{\mathbf{0}}$ since this is equivalent to equation 3.19 for any θ. Solving 3.19 for $\vec{\mathbf{x}}^{m+1}$ given $\vec{\mathbf{x}}^m$ and dt_m is thus equivalent to solving the equation $\vec{\mathbf{H}}(\vec{\mathbf{x}}) = \vec{\mathbf{0}}$. To solve the latter equation, the residual is expanded about the current time-step

$$\vec{\mathbf{H}}(\vec{\mathbf{x}}) = \vec{\mathbf{H}}(\vec{\mathbf{x}}^m) + \mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}^m)\delta\vec{\mathbf{x}} + \mathcal{O}\left(\delta\vec{\mathbf{x}}^2\right) \tag{3.23}$$

Evaluating this at $\vec{\mathbf{x}}^{m+1}$ where $\vec{\mathbf{H}}(\vec{\mathbf{x}}^{m+1}) = 0$ and truncating the expansion at first order gives

$$\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}^m)\delta\vec{\mathbf{x}} \simeq -\vec{\mathbf{H}}(\vec{\mathbf{x}}^m). \tag{3.24}$$

Iteratively solving this linear matrix equation for $\delta\vec{\mathbf{x}}$ to find $\vec{\mathbf{x}}^{m+1}$ is known as Newton iterations [12, p.26-27], which as iterations in k can be formulated in this case as

$$\begin{gathered}\vec{\mathbf{x}}_{k+1}^{m+1} = \vec{\mathbf{x}}_k^{m+1} + \delta\vec{\mathbf{x}}_k, \quad \text{where} \quad \vec{\mathbf{x}}_0^{m+1} = \vec{\mathbf{x}}^m \quad \text{and where} \quad \delta\vec{\mathbf{x}}_k \quad \text{solves} \\ \mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}_k^{m+1})\delta\vec{\mathbf{x}}_k = -\vec{\mathbf{H}}(\vec{\mathbf{x}}_k^{m+1})\end{gathered} \tag{3.25}$$

The core of the implemented time-integration algorithm are Newton iterations as in equation 3.25. They require evaluating the Jacobian and solving the linear equation involving the Jacobian at each iteration. The structure of the node-averaged time-dependent system of non-linear equations treated in this thesis allows the Jacobian to be found analytically. Furthermore, the structure of the Jacobian enables evaluating it in a computationally cheap way for any state $\vec{\mathbf{x}}$. Some effort has gone into taking maximal advantage of this, resulting in a somewhat delicate but highly efficient way to evaluate the Jacobian. For this reason, the Jacobian is now treated in detail, both analytically and regarding the details of the implemented evaluation.

It should be noted that this method of performing the time-integration is different from the standard practice method using predictor-corrector approach, as one can read about in [11, p.178-182]. The reason for this is the non-linear nature of the dynamical equations that can be handled using a non-linear solver instead of the classical approach.

3.3.2.1 About the Jacobian, $\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}})$

As is known from any course on elementary vector or multi-variable calculus, the elements $J_{ij}(\vec{\mathbf{x}})$ of the Jacobian $\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}) \in \mathbb{R}^{4N\times 4N}$ of the vector function $\vec{\mathbf{H}} : \mathbb{R}^{4N} \to$

$\mathbb{R}^{4N}$ are

$$J_{ij}(\vec{\mathbf{x}}) \equiv \frac{\partial H_i(\vec{\mathbf{x}})}{\partial x_j} \tag{3.26}$$

The first task here is to find the Jacobian of the residual in equation 3.22. The third term, $-\vec{\mathbf{b}}$, of the RHS vanishes, and the first term on the RHS has the form $\mathbf{A}\vec{\mathbf{x}}$ i.e. a constant square matrix $\mathbf{A}$ times the input vector $\vec{\mathbf{x}}$ of the residual. The Jacobian of such an expression is simply the matrix $\mathbf{A}$ itself, which can easily be shown[6]. We thus have that

$$\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}) = \left[\mathbb{1} - \theta dt_m \mathbf{L}^{m+1}\right] - \theta dt_m \mathbf{J}_{\mathbf{N}\vec{\mathbf{f}}}(\vec{\mathbf{x}}). \tag{3.27}$$

The Jacobian of the non-linear term requires special attention. Looking back at equations 3.15, 3.16 and 3.17 it is clear from inspection that for the purposes of calculating the Jacobian, the affine feedback term is more appropriately thought of as the sum of a constant vector, which vanishes, and a term which is just a constant square matrix times the input vector. The feedback term thus contribute as a constant matrix $-\theta dt_m \mathbf{N}_\alpha$ to the Jacobian, with nonzero elements specified by the previously defined diagonal matrices $\mathbf{N}_{11}, \mathbf{N}_{12}, \mathbf{N}_{21}$ and $\mathbf{N}_{22}$ occupying the $N \times N$-blocks of $\mathbf{N}_\alpha$ as indicated by their indices.

The non-linear coupling between the thermal flux and the Xenon concentration gives the only non-constant contribution $-\theta dt_m \mathbf{N}_\sigma$ to the Jacobian. The $4N$ nonzero elements[7] of $\mathbf{N}_\sigma = \mathbf{N}_\sigma(\vec{\mathbf{x}})$ lie along the four diagonals which are contained within the $N \times N$-sized blocks and which have the values as indicated by

$$\mathbf{N}_\sigma(\vec{\mathbf{x}}) = -\sigma_X \begin{bmatrix} \ddots & \ddots & \ddots & \\ \ddots & v_2\vec{\mathbf{x}}_4 & \ddots & v_2\vec{\mathbf{x}}_2 \\ \ddots & \ddots & \ddots & \ddots \\ & \vec{\mathbf{x}}_4 & \ddots & \vec{\mathbf{x}}_2 \end{bmatrix} \tag{3.28}$$

The Jacobian $\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}})$ can thus be expressed in terms of a matrix $\mathbf{J}_1$ and a state-dependent matrix $\mathbf{J}_2(\vec{\mathbf{x}})$, both of which are defined through the following analytic expression of the node-averaged Jacobian

$$\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}) = \mathbf{J}_1 + \mathbf{J}_2(\vec{\mathbf{x}}) \equiv \left[\mathbb{1} - \theta dt_m \Big(\mathbf{L}^{m+1} + \mathbf{N}_\alpha\Big)\right] - \theta dt_m \mathbf{N}_\sigma(\vec{\mathbf{x}}) \tag{3.29}$$

The matrix $\mathbf{J}_1$ is always constant throughout any one set of Newton iterations since dt_m, θ and $\mathbf{L}^{m+1}$ are all then fixed. Evaluating the full Jacobian $\mathbf{J}_{\vec{\mathbf{H}}}$ at any intermediate step can therefore be done by updating the small number of variable elements of $\mathbf{J}_{\vec{\mathbf{H}}}$ from the previous step. Maximally contiguous indices for the variable Jacobian elements can easily be found from the sparse matrix $\mathbf{N}_\sigma$ in `MatLab` and such an index-array can be used throughout the simulation. Moreover, as is made clear in

[6] For example, using Einstein notation; The result $\frac{\partial x_j}{\partial x_k} = \delta_{jk}$ immediately gives that $\frac{\partial A_{ik}x_k}{\partial x_j} = A_{ik}\frac{\partial x_k}{\partial x_j} = A_{ik}\delta_{kj} = A_{ij}$.

[7] A caveat to this is that not even all of these elements in the full Jacobian are variable; any element corresponding to a reflector node has vanishing Xenon-concentration always. This reduces the actual number of nonzero elements even further in a predictable way.

section 3.3.3, the matrix $\mathbf{J}_1$ remains constant also in-between Newton iterations in the vast majority of cases. To summarise; in this implementation the Jacobian can always be evaluated exactly and this is cheap enough to never significantly affect overall performance.

3.3.3 Time-dependent Perturbations - Control Rods

Collections of control rods called banks with individually specifiable insertion levels can be inserted into the reactor core. All banks are inserted simultaneously for any amount of time until they are all simultaneously withdrawn. Rods can be inserted either from the top or from the bottom. For an individual node a rod is considered either fully inserted or not inserted at all at any one time. The presence of a control rod in a node is modelled as a fractional change to the cross-sections and diffusion coefficients of that node. The fractions by which each material property should change are specified as an input to the solver, and a complete specification of the nodes that belong to each bank is also an input to the solver. For the development data-set, the fractions were obtain from [13].

The insertion of the control rods thus have the effect of transforming the system matrix without the perturbation, denoted as $\mathbf{L}$, according to

$$\mathbf{L} \mapsto \mathbf{L}' = \mathbf{L} + \delta\mathbf{L} \tag{3.30}$$

Therefore the total time-dependence of $\mathbf{L}(t)$ due a control rod perturbation, where insertion happens at $t = t_i$ and withdrawal happens at $t = t_w$, can be expressed using Heaviside step functions as

$$\mathbf{L}(t) = \mathbf{L} + \theta(t - t_i)\theta(t_w - t)\delta\mathbf{L} \tag{3.31}$$

If the corresponding discrete time-steps of insertion and withdrawal have indices m_i and m_w, the time-discretized version can be expressed using Kronecker delta functions as

$$\mathbf{L}^m = \mathbf{L} + \left(\delta^m_{m_i<m'<m_w}\right)\delta\mathbf{L}, \quad \text{where} \quad \delta^m_{m_i<m'<m_w} \equiv \sum_{m'=m_i+1}^{m_w-1} \delta^m_{m'} \tag{3.32}$$

The reason for the exact index bounds will be made clear shortly, in the next section. The induced transformation on the system dynamics, equation 3.13, is straightforward. The following general expression for the residual, equation 3.22, and for the Jacobian, equation 3.13, for any time-step dt_m and time-discretization method θ, are

$$\begin{aligned}
\vec{\mathbf{H}}^m(\vec{\mathbf{x}}) &= \left[\mathbb{1} - \theta dt_m\left(\mathbf{L} + \left(\delta^{m+1}_{m_i<m'<m_w}\right)\delta\mathbf{L}\right)\right]\vec{\mathbf{x}} - \theta dt_m \mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}) \\
&\quad - \left[\mathbb{1} + (1-\theta) dt_m\left(\mathbf{L} + \left(\delta^m_{m_i<m'<m_w}\right)\delta\mathbf{L}\right)\right]\vec{\mathbf{x}}^m - (1-\theta) dt_m \mathbf{N}\vec{\mathbf{f}}^m \\
\mathbf{J}^m_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}) &= \mathbf{J}_1 + \left(\delta^m_{m_i<m'<m_w}\right)\delta\mathbf{J}_1 + \mathbf{J}_2(\vec{\mathbf{x}}),
\end{aligned} \tag{3.33}$$

where the definition of the induced transformation term in the Jacobian is $\delta\mathbf{J}_1 \equiv -\theta dt_m \delta\mathbf{L}$. At this point, all quantities that are ever used throughout the simulation

have been introduced. It remains to discuss how the quantities $\delta\mathbf{L}$ are calculated.

The value of any material property of type α at node n when the node is occupied by a control rod is denoted $\Sigma^{in}_{\alpha,n}$ and otherwise it is denoted $\Sigma^{out}_{\alpha,n}$. The fractions that need to be inputted are the single numbers η_α that satisfy

$$\eta_\alpha = \frac{\Sigma^{in}_{\alpha,n} - \Sigma^{out}_{\alpha,n}}{\Sigma^{out}_{\alpha,n}} \implies \Sigma^{in}_{\alpha,n} = (1+\eta_\alpha)\Sigma^{out}_{\alpha,n}. \tag{3.34}$$

With the insertion levels of every bank also specified in the options of the solver, the matrix $\mathbf{L}' = \mathbf{L} + \delta\mathbf{L}$ is calculated from scratch, the same way that $\mathbf{L}$ is calculated. The matrix $\delta\mathbf{L}$ is then calculated simply as $\delta\mathbf{L} = \mathbf{L}' - \mathbf{L}$.

3.3.4 Variable time-step dt_m

It was mentioned in sections 3.3.1.1 and 3.3.1.2 that the control rod perturbations discussed above have a strong tendency to produce numerical oscillations when using Crank-Nicholson time-discretization, and that it is absolutely necessary to fix this problem for the simulations to work properly. This is a well known issue with the Crank-Nicholson method and several ways of damping the oscillations have been proposed [15]. Many of the proposed solutions involve running a relatively small number of extremely short time-steps immediately as the sharp system transient is engaged, using the Crank-Nicholson method. Another typical ingredient to the proposed solutions involve taking one or perhaps two initial steps using the backward Euler method, thereby strongly damping the oscillations. This latter method is allowable while preserving the global second order accuracy of the entire simulation since the local error of the backward Euler method is still second order [15, p.814].

What is a good solution to the problem of numerical oscillations varies with the perturbation. For this reason, the standard options of the solver allows two sets of fast time-steps, each with its own dt_m and θ, to be used immediately as a perturbation is applied to the system. If such a set or sets are chosen to be used, as soon as control rods are inserted or withdrawn, all of the time-steps of the first set are executed, immediately followed by the entire second set.

This explains the index bounds appearing in equation 3.32. Any perturbation is considered to happen at a specific moment in time with time-step index m_p, and then to be felt by the system in the next time-step with index $m_p + 1$ and all subsequent time-steps until withdrawal. This is necessary to ensure that the time-discretized dynamical equation 3.19 are solved in the correct way at the exact moment of insertion/withdrawal. The following equations illustrate the issue here by showing the correct residuals before, at, and after the moment of perturbation. The quantities θ,dt_{m_p-1}, dt_{m_p} and dt_{m_p+1} can all be different for each of the residuals, due to the

versatility of the solver:

$$\begin{aligned}
t_{m_p-1} \to t_{m_p} : \vec{\mathbf{H}}^{m_p-1}(\vec{\mathbf{x}}) &= \Big[\mathbb{1} - \theta dt_{m_p-1}\mathbf{L}\Big]\vec{\mathbf{x}} - \theta dt_{m_p-1}\mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}) \\
&\quad - \Big[\mathbb{1} + (1-\theta)dt_{m_p-1}\mathbf{L}\Big]\vec{\mathbf{x}}^{m_p-1} - (1-\theta)dt_{m_p-1}\mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}^{m_p-1}) \\
t_{m_p} \to t_{m_p+1} : \vec{\mathbf{H}}^{m_p}(\vec{\mathbf{x}}) &= \Big[\mathbb{1} - \theta dt_{m_p}(\mathbf{L}+\delta\mathbf{L})\Big]\vec{\mathbf{x}} - \theta dt_{m_p}\mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}) \\
&\quad - \Big[\mathbb{1} + (1-\theta)dt_{m_p}\mathbf{L}\Big]\vec{\mathbf{x}}^{m_p} - (1-\theta)dt_{m_p}\mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}^{m_p}) \\
t_{m_p+1} \to t_{m_p+2} : \vec{\mathbf{H}}^{m_p+1}(\vec{\mathbf{x}}) &= \Big[\mathbb{1} - \theta dt_{m_p+1}(\mathbf{L}+\delta\mathbf{L})\Big]\vec{\mathbf{x}} - \theta dt_{m_p+1}\mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}) \\
&\quad - \Big[\mathbb{1} + (1-\theta)dt_{m_p+1}(\mathbf{L}+\delta\mathbf{L})\Big]\vec{\mathbf{x}}^{m_p+1} - \theta dt_m\mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}^{m_p+1})
\end{aligned} \tag{3.35}$$

3.4 Time-integration in MatLab with pseudo-code

The heart of the solver is the algorithm for time-evolution, and in terms of computing time, the most important part of this algorithm is the code for solving the large sparse system of linear equations found in equation 3.25. MatLab is equipped with many different direct solvers and iterative solvers for large matrix equations, and has flow-charts in its documentation with proposals for which solver to use in what circumstance [17]. After extensive testing, the fastest reliable method that has been found is the combination of using an incomplete LU-decomposition to obtain preconditioner matrices L and U followed by solving the matrix equation using the iterative method called the generalized minimum residual method [19], [20]. The exact function calls are

$$\begin{aligned}
[L, U] &= \texttt{ilu}(\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}})) \\
[\delta\vec{\mathbf{x}}, \sim] &= \texttt{gmres}(\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}), -\vec{\mathbf{H}}(\vec{\mathbf{x}}), 10, \texttt{tolGMRES}, 200, L, U)
\end{aligned} \tag{3.36}$$

The pseudo-code for the time-integration algorithm is the following:

$$\begin{aligned}
&\textit{Update all quantities \& function handles associated with } \vec{\mathbf{H}}(\vec{\mathbf{x}}) \\
&\textit{Update} \quad \mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}) \\
&[L, U] = \texttt{ilu}(\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}})) \\
&[\delta\vec{\mathbf{x}}, \sim] = \texttt{gmres}(\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}), -\vec{\mathbf{H}}(\vec{\mathbf{x}}), 10, \texttt{tolGMRES}, 200, L, U) \\
&\vec{\mathbf{x}} = \vec{\mathbf{x}} + \delta\vec{\mathbf{x}} \\
&\textit{while} \quad \textit{Objective}(\vec{\mathbf{x}}) > \texttt{tolNEWTON} : \\
&\qquad \textit{Update} \quad \mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}) \\
&\qquad [L, U] = \texttt{ilu}(\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}})) \\
&\qquad [\delta\vec{\mathbf{x}}, \sim] = \texttt{gmres}(\mathbf{J}_{\vec{\mathbf{H}}}(\vec{\mathbf{x}}), -\vec{\mathbf{H}}(\vec{\mathbf{x}}), 10, \texttt{tolGMRES}, 200, L, U) \\
&\qquad \vec{\mathbf{x}} = \vec{\mathbf{x}} + \varepsilon(\vec{\mathbf{x}}, \delta\vec{\mathbf{x}})\delta\vec{\mathbf{x}} \\
&\textit{end}
\end{aligned} \tag{3.37}$$

The objective function is simply the norm of the current residual i.e:

$$Objective(\vec{\mathbf{x}}) = \left\| \vec{\mathbf{H}}(\vec{\mathbf{x}}) \right\|_2 \tag{3.38}$$

With this objective function, it is straight-forward to programmatically assign an acceptable tolerance to the Newton iterations as the error associated with the chosen time-discretization method θ and current time-step dt_m themselves if one were to take a step from the equilibrium state to the equilibrium state without perturbation:

$$\begin{aligned} \texttt{tolNEWTON} = \texttt{tolNEWTON\_scaling}\times \\ \times \left\| [\mathbb{1} - \theta dt_m \mathbf{L}]\vec{\mathbf{x}}_{eq} - \theta dt_m \mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}_{eq}) - [\mathbb{1} + (1-\theta) dt_m \mathbf{L}]\vec{\mathbf{x}}_{eq} - (1-\theta) dt_m \mathbf{N}\vec{\mathbf{f}}(\vec{\mathbf{x}}_{eq}) \right\|_2 \end{aligned} \tag{3.39}$$

The factor `tolNEWTON_scaling` allows the user to relax or tighten these constraints as desired.

In the while-loop, the quantity $\delta\vec{\mathbf{x}}$ is used to find the next iterate as $\vec{\mathbf{x}} = \vec{\mathbf{x}} + \varepsilon(\vec{\mathbf{x}}, \delta\vec{\mathbf{x}})\delta\vec{\mathbf{x}}$. The function $\varepsilon(\vec{\mathbf{x}}, \delta\vec{\mathbf{x}})$ finds the minimum of $f(\epsilon) = Objective(\vec{\mathbf{x}} + \epsilon\delta\vec{\mathbf{x}})$ on the interval $[0, 2]$ using the `MatLab` optimizer-function `fminbnd`.

4

Results

This chapter shows some results that have been obtained with the solver. First is the main result of this thesis: a demonstration of induced Xenon oscillations by means of control rod insertion. Secondly, a benchmark of the dependence of the solver on the size of the time-step dt is shown, where the same simulation as in the first section is repeated one additional time, using a dt that is two orders of magnitude smaller.

4.1 Simulation: Inducing Xenon oscillations

The simulation shown here uses a time-step of $dt = 900$ seconds and consists of three parts. First the system is initialized to the steady-state $\vec{\mathbf{x}}_0 = \vec{\mathbf{x}}_{eq}$ and 2.5 hours are simulated, where the system should and does remain stationary. Secondly, control rod banks A,B,C and D [16] are inserted to an insertion level of 50% for two hours. The simulation uses two sets of fast time-steps, the first set consists of 2 backward Euler steps with $dt = 0.1$ seconds, while the second set consists of 50 Crank-Nicholson steps with $dt = 0.0001$ seconds[1]. Thirdly, after the rods are withdrawn and both sets of fast time-steps have been executed once again, 48 hours of simulation are carried out. This entire simulation took 2 minutes and 18 seconds of wall time[2], which includes real-time updating of the plots every iteration. The final two graphs, shown in figures 4.3 and 4.4, extend the third part of the simulation to 498 hours of simulation time, not all of which is shown. This entire simulation took 19 minutes and 57 seconds of wall time.

Throughout the simulation, there are seven quantities that are calculated. The first four quantities are meant to visualize the time-dependence of the total state $\vec{\mathbf{x}}(t) = \left[\vec{\phi}_1(t) \quad \vec{\phi}_2(t) \quad \vec{\mathbf{I}}(t) \quad \vec{\mathbf{X}}(t)\right]^\top$, and are defined as the unitless core-averaged quantities relative to the corresponding core-averaged equilibrium quantities

$$\begin{aligned} \tilde{\phi}_1(t) &\equiv \frac{<\vec{\phi}_1(t)>_{core}}{<\vec{\phi}_{1,eq}>_{core}}, \quad \tilde{\phi}_2(t) \equiv \frac{<\vec{\phi}_2(t)>_{core}}{<\vec{\phi}_{2,eq}>_{core}}, \\ \tilde{I}(t) &\equiv \frac{<\vec{\mathbf{I}}(t)>_{core}}{<\vec{\mathbf{I}}_{eq}>_{core}}, \quad \tilde{X}(t) \equiv \frac{<\vec{\mathbf{X}}(t)>_{core}}{<\vec{\mathbf{X}}_{eq}>_{core}}, \end{aligned} \tag{4.1}$$

[1] The reason the longer time-step set comes first is to make the backward Euler method immediately damp the oscillations due to the perturbation.

[2] This was measured using a machine equipped with an i7-12700k CPU (at factory clock settings) and Corsair Vengeance RAM at 3200MHz and CL16.

These four quantities are shown in figure 4.1. In the graph, the three aforementioned parts of the simulation are clearly visible, first the system is in a steady-state for 2.5 hours, until time $t = 0$. At this point, the control rods are inserted, resulting in an immediate drop in flux. Throughout the insertion period therefore, the burn-off of Xenon is decreased, leading to an increase in the Xenon concentration. Furthermore, the lower flux leads to a decrease in Iodine production from fission, lowering the Iodine concentration. At $t = 2$ hours, the rods are withdrawn, and the system is subsequently free except for the feedback term which tends to draw the flux towards the steady-state. The dynamics of the system after withdrawal show signs of the type of phenomena described in detail the section 1.3 on Xenon oscillations, indicating that Xenon oscillations of only the fundamental mode have been induced by the control rod perturbation.

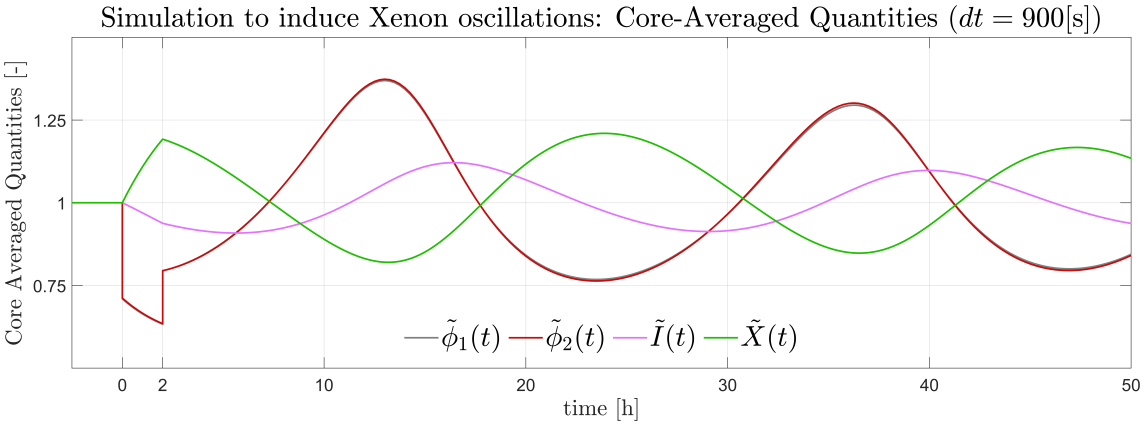


Figure 4.1: Plot of the core-averaged quantities for the simulation using $dt = 900$ seconds. The three aforementioned parts of the simulation are clearly visible. It should be noted that the variations about the equilibrium values of the fast and thermal core-averaged fluxes overlap almost perfectly throughout the simulation.

The fifth quantity is specifically meant to measure and visualize the time-dependence of spatial Xenon power oscillations and is called axial shape index, or ASI. It is defined as the difference between the power produced in the bottom and the top of the reactor relative to the total power produced in the reactor[3]:

$$ASI(t) \equiv \frac{P_{bottom}[\vec{\phi}(t)] - P_{top}[\vec{\phi}(t)]}{P[\vec{\phi}(t)]}. \tag{4.2}$$

This quantity is unitless, and it takes on values in the range $[-1, 1]$. A larger positive value indicates that a larger fraction of all power is produced in the bottom half of the reactor, and vice-versa. The $ASI(t)$ is shown in figure 4.2. The graph shows how the perturbation causes an axially asymmetric deviation of the steady-state of the reactor, and axial spatial oscillations are observed after withdrawal. It should be noted that the equilibrium state is already strongly shifted towards the bottom of the reactor, having an ASI of roughly $ASI_{eq} \approx 0.2025$.

[3]Power calculations as a functional of the flux state $\vec{\phi}(t)$ were discussed in section 2.5

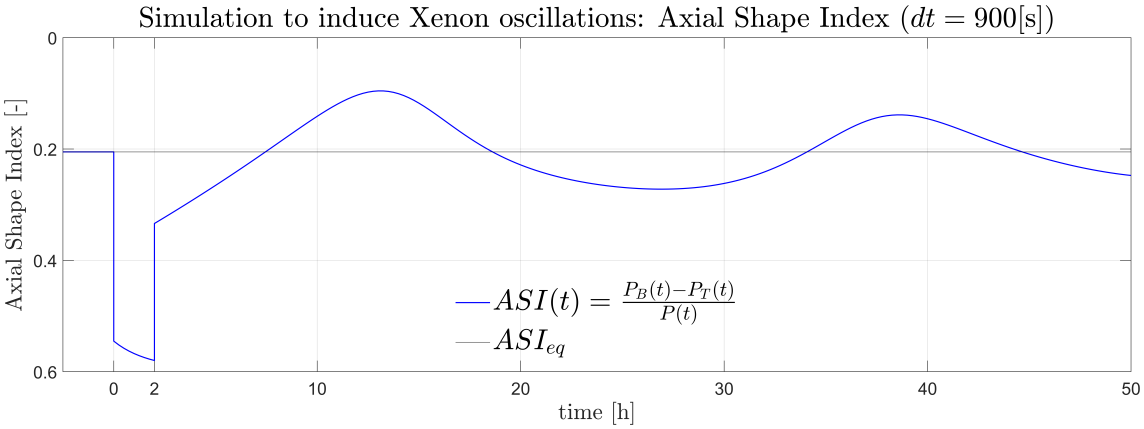


Figure 4.2: Plot of the axial shape index for the simulation using $dt = 900$ seconds. Notice that the range of the y-axis is limited and its direction reversed. The reversal of the axis allows the graph of the ASI to be interpreted in a more natural way, where points further down in the graph indicate that a larger fraction of all power is produced in the bottom of the reactor, and vice-versa.

The sixth and seventh quantities are also meant to measure and visualize Xenon oscillations. Throughout the simulation, the thermal flux is measured repeatedly at two points of the reactor which are so-called parity points. The graph of the parity quantities is shown in figure 4.3. Both these quantities are once again unitless as they are measured relative to their equilibrium state values. In order to understand what these quantities are, consider the cylindrical reactor of the development dataset to be centered about the origin. The thermal flux $\phi_2(\vec{\mathbf{r}}, t)$ would then have parity function $\phi_2(-\vec{\mathbf{r}}, t)$. Here, a point $\vec{\mathbf{r}}$ was chosen in the middle of the first octant, located in the top half of the reactor, meaning the parity point $-\vec{\mathbf{r}}$ is located in the bottom half of the reactor. In order to be able to appreciate the behaviour of the parity points, the third part of the simulation was extended significantly, running all the way to $t = 500$ hours. For comparison, the corresponding extended ASI is shown in figure 4.4.

Because of the clearly visible damping of the oscillations, the data past $t = 350$ hours is not shown as the oscillations become to small to be seen. Clearly, the oscillations are not opposite each other, as would have been the case for equal and opposite perturbations like those described in section 1.3. In the case shown here, the much more realistic perturbation of half-way control rod insertion is applied instead. We do see a clear time-independent phase shift in the oscillations between the two halves of the core. We also see a variation of the relative amplitude between the two halves.

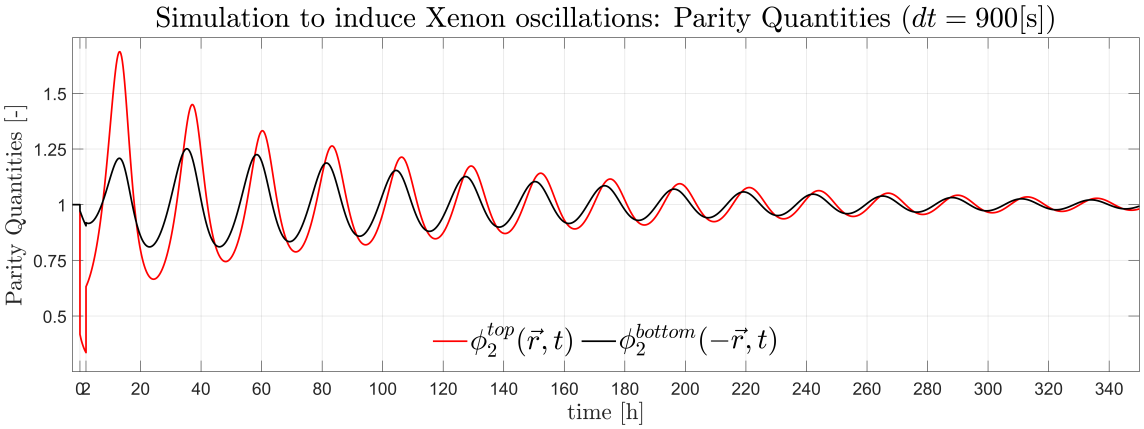


Figure 4.3: Plot of parity quantities for the simulation using $dt = 900$ seconds, extended to simulate until $t = 500$ hours. The final 150 hours of simulation is not shown due to the damping. Notice that the range of the y-axis is extended compared with the graph of the core-averaged quantities shown in figure 4.1.

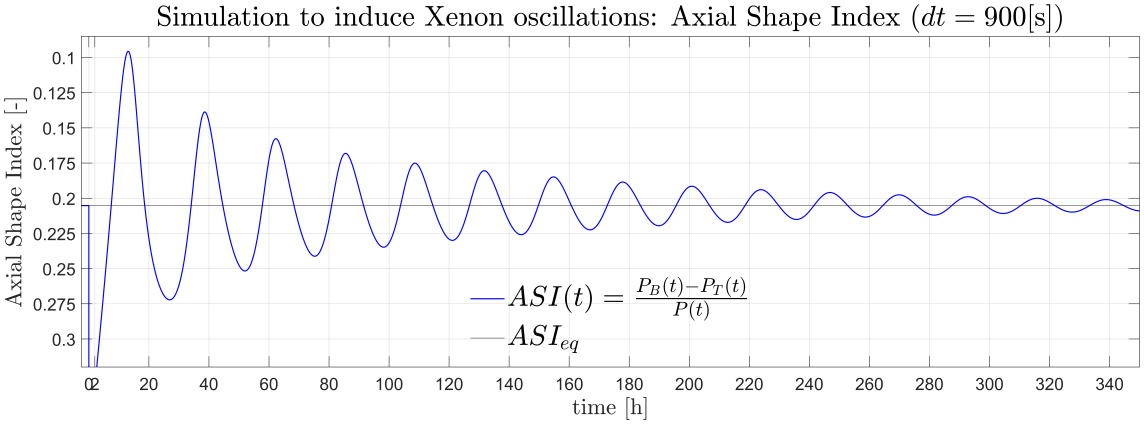


Figure 4.4: Plot of axial shape index for the simulation using $dt = 900$ seconds, extended to simulate until $t = 500$ hours. The final 150 hours of simulation is not shown due to the damping. Notice that the y-axis range is even more limited than before, compared to the graph in figure 4.2. The graph shown here is meant to be compared with the graph of the parity quantities.

Given the prior discussion on Xenon oscillations from section 1.3 and the graphs shown in figures 4.1, 4.2, 4.3 and 4.3, it is concluded that Xenon oscillations have been successfully induced.

4.2 Benchmark: Dependence of the solver on dt

In this benchmark, a simulation using a time-step $dt = 9$ seconds and which is otherwise identical to the one from section 4.1 is shown. To make comparison fair, only every 100th data-points were stored, meaning that the two graphs contain the same number of points, taken at the same times. The graphs corresponding to the ones

found in figures 4.1 and 4.2 is shown in figures 4.5 and 4.6, respectively.

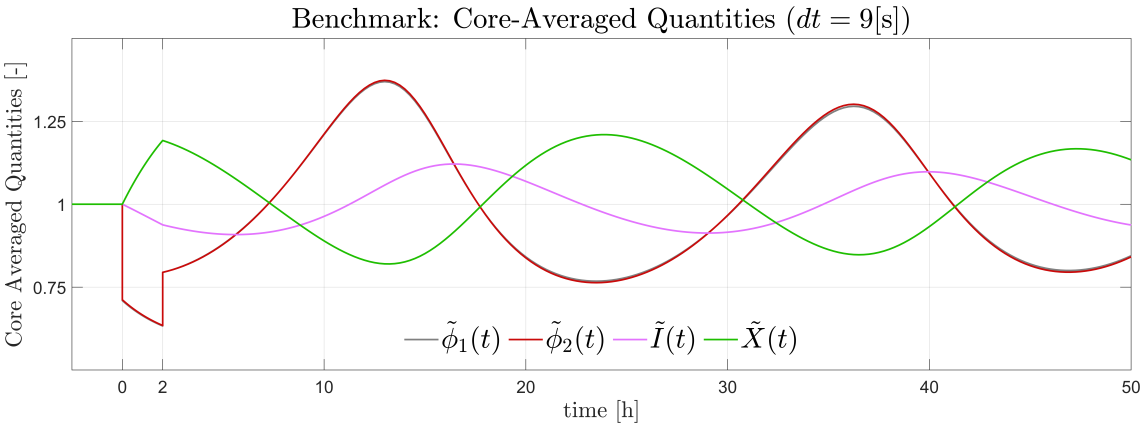


Figure 4.5: Plot of the core-averaged quantities using $dt = 9$ seconds, where for the final 48 hours of the simulation, only every 100th data-point is shown. This graph is meant to be compared to the graph in figure 4.1.

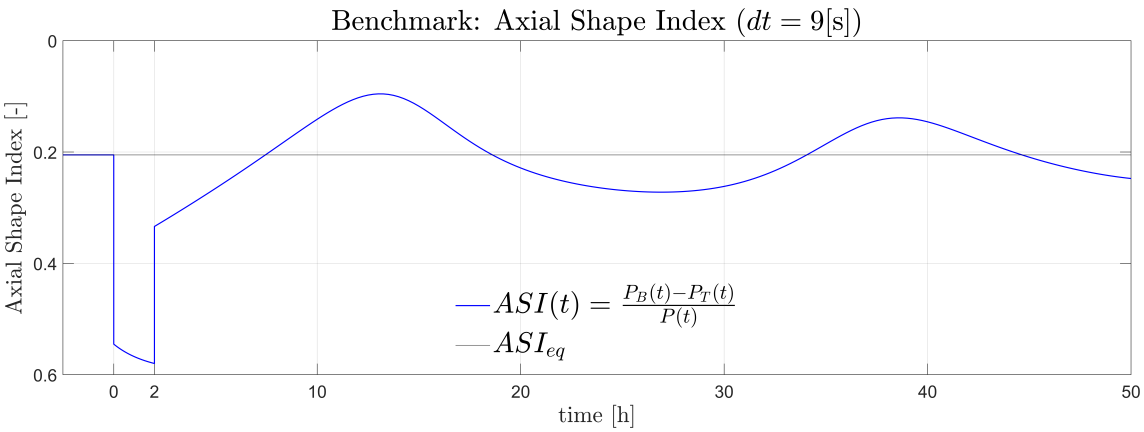


Figure 4.6: Plot of the axial shape index using $dt = 9$ seconds, where for the final 48 hours of the simulation, only every 100th data-point is shown. This graph is meant to be compared to the graph in figure 4.2.

As was mentioned in section 4.1, the simulation shown there took 2 minutes and 18 seconds. In contrast, the simulation shown here took 2 hours and 52 minutes. The longer time-step simulation clearly passes the eye-test; there is no visible difference when compared to the simulation using a much finer time-step, as also demonstrated quantitatively hereafter.

Treating the time-series of the core-averaged quantities obtained from the $dt = 9$[s] simulation as the reference data, the final 48 hours of both simulations can be directly compared to each other. Calling the root mean squared error function in

`MatLab`, called `rmse` [21], for all pairs of core-averaged quantities gives the values

$$\begin{aligned} \texttt{rmse}\left(\tilde{\phi}_1^{9s}, \tilde{\phi}_1^{900s}\right) \approx 5.9 \times 10^{-4}, \quad \texttt{rmse}\left(\tilde{\phi}_2^{9s}, \tilde{\phi}_2^{900s}\right) \approx 6.0 \times 10^{-4} \\ \texttt{rmse}\left(\tilde{I}^{9s}, \tilde{I}^{900s}\right) \approx 2.3 \times 10^{-4}, \quad \texttt{rmse}\left(\tilde{X}^{9s}, \tilde{X}^{900s}\right) \approx 3.9 \times 10^{-4} \end{aligned} \tag{4.3}$$

The fact that these values are all similar and small further strengthens the conclusion one is tempted to draw from the eye-test, namely that running the much more computationally demanding simulation gave no substantial benefit in terms of physical accuracy. As a final demonstration that this conclusion is in fact valid, the total state $\vec{\mathbf{x}}$ at the final time-step from both simulations, which corresponds to the states obtained after 50 hours of simulation, are compared using the same method as above. The result is

$$\texttt{rmse}\left(\vec{\mathbf{x}}_{t=50[\mathrm{h}]}^{9s}, \vec{\mathbf{x}}_{t=50[\mathrm{h}]}^{900s}\right) \approx 5.3 \times 10^{-4} \tag{4.4}$$

This benchmark thus clearly shows that the solver is capable of performing typical Xenon oscillation simulations, which require on the order of 50 hours of simulation time, in no more than a few minutes.

5

Conclusion & Outlook

The thesis objective from section 1.1.3 has been successfully achieved. The solver that has been built meets all the requirements originally specified in terms of spatial discretization, node enumeration, boundary conditions and input format. The solver has been applied to a realistic PWR model as described by the development data-set and Xenon oscillations have been induced using realistic time-dependent perturbations. Furthermore, a large set of realistic time-dependent perturbations can be applied and are easily chosen by the user through specifying duration and insertion levels of the control rod banks in the standard options of the solver. These perturbations can be made asymmetric, random or otherwise different than what has been implemented here by changing the input-data that specifies what nodes belong to what control rod banks. Top-to-bottom and bottom-to-top insertion can be chosen in the options in order to ensure that both PWR type simulations and BWR type simulations, respectively, can be performed. As was discussed in the very first section of chapter 1, one real-world reason to study Xenon oscillations is be able to predict them in order to avoid them in the case of running nuclear reactors in load-following conditions. This tool should be able to help with such predictive work for real-world applications.

The solver performs significantly better in terms of execution time than what was originally expected. There are several reasons for this. For example, the very high sparsity of the system matrices used for time-evolution and their simple sparsity patterns. The simplicity of the time-independent sparsity patterns makes the incomplete LU-decomposition very cheap. One of the main things allowing for fast execution is the fact that the exact Jacobian is essentially free to evaluate at any time and for any state $\vec{\mathbf{x}}$, as was discussed at length in section 3.3.2.1. The reason the Jacobian can be evaluated so cheaply is, as the analytic expression for it indicates, that a universal and maximally contiguous index array can be found which allows all variable Jacobian elements to be updated according to a very simple formula using the system state $\vec{\mathbf{x}}$ and the parameters σ_X and v_2. This leads to a possible source of improvement of the time-integration algorithm: the Hessian tensor.

Recall the discussion from section 3.3.2 leading to the formulation of Newton iterations as in equation 3.25. There, the Taylor expansion of the residual is stopped at first order. The subsequent discussion on the evaluation of the Jacobian clearly indicates that a second order term, which would necessarily involve two derivatives with respect to the state space variables constituting $\vec{\mathbf{x}}$, could only ever have one surviving term. This term would correspond to a derivative of the expression in

equation 3.28 of the variable Jacobian elements. Since every individual matrix element appearing in this term is linear in the state space variables x_i, applying any set of derivatives $\frac{\partial}{\partial x_j}$ to this term would necessarily result in some constant mathematical object. In the Taylor series, this object would be a rank 3 tensor usually called the Hessian tensor, which for a vector function such as the residual $\vec{\mathbf{H}}(\vec{\mathbf{x}})$ has elements defined according to the following expression:

$$Hessian(\vec{\mathbf{H}})_{ijk} \equiv \frac{\partial^2 H_i}{\partial x_j \partial x_k} \tag{5.1}$$

This Hessian would be extremely sparse, have very simple and predictable sparsity pattern and most importantly be constant for the residual treated here. It should be possible to use it very efficiently if the Taylor series were to be truncated at second order. Since the Hessian tensor would be constant, this second order Taylor series would then be exact. This has not been pursued here, mainly because the time-integration algorithm is already so successful. If this was not the case however, it would be the first line of inquiry to determine if the Hessian could be used to improve the integration.

The final topic discussed in this thesis is on further developing the feedback term. Primed by the discussions of the feedback term in sections 2.3.3 and 2.5, the feedback term as it is currently implemented makes certain simulations essentially impossible. Perhaps the most obvious simulation that it is not really possible to perform is a full shutdown of the reactor, called a SCRAM. The feedback term is designed to be a good approximation of the state-dependencies of the material properties of the reactor *close to the steady-state*, and becomes worse and worse as the state deviates further from the steady-state. When attempting to SCRAM the reactor by fully inserting all control rod banks, the flux never decreases enough for a proper shutdown to be simulated. The feedback term draws the state towards the steady-state too much, as the flux gets close to zero. In order to simulate a shutdown, a relatively straight-forward fix could be attempted, where the feedback coefficient is equipped with a time-dependency such that it is switched off as the reactor undergoes the SCRAM. In terms of the feedback coefficient $\alpha_{g\to g'}(\vec{\mathbf{r}})$, the insertion time t_i, the withdrawal time t_w and heaviside functions as in equations 2.18 and 3.31, this transformation would look something like:

$$\alpha_{g\to g'}(\vec{\mathbf{r}}) \mapsto \theta(t - t_i)\theta(t_w - t)\alpha_{g\to g'}(\vec{\mathbf{r}}) \tag{5.2}$$

Such a transformation would have ramifications for many terms throughout the simulation. It would however not require any new sets of fast time-steps to be used, as the feedback term would only change when the state matrices are changed due to the control rod perturbations. This has not been implemented mainly due to time-constraints. It should certainly be noted that introducing this time-dependence *in a way that properly retains the physicality of the model* would require some serious thought and care. Specifically, the effects of delayed neutrons would need to be accounted for by a modification of the dynamics, described by equation 2.24.

Bibliography

[1] IAEA, "Age Distribution," 2023. [Online]. Available: https://pris.iaea.org/pris/worldstatistics/operationalbyage.aspx (accessed on: 2023-05-06).

[2] IAEA, "In Operation & Suspended Operation Reactors," 2023. [Online]. Available: https://pris.iaea.org/PRIS/WorldStatistics/OperationalReactorsByType.aspx (accessed on: 2023-05-06).

[3] EIA, "Nuclear power plants generated 68% of France's electricity in 2021," [Online]. Available: https://www.eia.gov/todayinenergy/detail.php?id=55259 (accessed on: 2023-05-19).

[4] A. Lokhov, "Load-following with nuclear power plants," [Online]. Available: https://www.oecd-nea.org/nea-news/2011/29-2/nea-news-29-2-load-following-e.pdf (accesed on: 2023-05-19).

[5] Vattenfall, "Kärnkraft - Stöd för koldioxidsnåla lösningar för energiomställningen är på gång," [Online]. Available: https://group.vattenfall.com/se/var-verksamhet/vara-energislag/karnkraft (accessed on: 2023-05-19).

[6] K. Andgren *et al*, "Lastföljning i kärnkraftverk," Elforsk, 12:08, Dec. 2011, [Online]. Available: https://energiforskmedia.blob.core.windows.net/media/19886/lastfoljning-i-karnkraftverk-elforskrapport-2012-08.pdf (accessed on: 2023-15-19)

[7] H. Anglart, *Nuclear Reactor Dynamics and Stability,*. Textbook, Royal Institute of Technology, Stockholm, Sweden, 2011.

[8] Pedersen K.T., Demazière C., and Vinai P., Understanding xenon oscillations through physics-based Reduced Order Modelling. Proc. 2022 American Nuclear Society Annual Meeting - The New Outlook, Anaheim, CA, USA, June 12-16, 2022 (2022).

[9] C. Demazière, (2011) CORE SIM: A multi-purpose neutronic tool for research and education. Annals of Nuclear Energy, Volume 38, Issue 12,2011,Pages 2698-2718

[10] C. Demazière, *Physics of Nuclear Reactors,*. Textbook, Chalmers University of Technology, Gothenburg, Sweden, 2017.

[11] C. Demazière, *Modelling of Nuclear Reactor Multi-physics: From Local Balance Equations to Macroscopic Models in Neutronics and Thermal-Hydraulics,* ISBN-978-0-12-815069-6, Academic Press/Elsevier, 2020.

[12] C. Demazière, "INTRODUCTION TO SOLVING LARGE SYSTEMS OF LINEAR AND NON-LINEAR EQUATIONS". Textbook, Chalmers University of Technology, Gothenburg, Sweden, 2022.

[13] C. Demazère, Private communication, 2023.

[14] J. Lamarsh, *Introduction to Nuclear Reactor Theory,* Addison-Wesley Publishing Company, 1966.

[15] D. Britz, O. Østerby and J. Strutwolf, "Damping of Crank-Nicholson error oscillations," Computational biology and chemistry, vol. 27, no. 3, pp. 253-263, Jul. 2003, doi:10.1016/S0097-8485(02)00075-X.

[16] T. Kozlowski, T. J. Downar, "PWR MOX/UO2 Core Transient Benchmark Final Report," NEA, Jan. 2007, [Online]. Available: `https://www.oecd-nea.org/science/wprs/MOX-UOX-transients/benchmark_documents/final_report/mox-uo2-bench.pdf`, (Accessed on: 2023-05-06).

[17] MathWorks, "Iterative Methods for Linear Systems", [Online]. Available: `https://se.mathworks.com/help/matlab/math/iterative-methods-for-linear-systems.html#f6-14578` (Accessed on 2023-05-11).

[18] MathWorks, "lu," [Online]. Available: `https://se.mathworks.com/help/matlab/ref/lu.html` (Accessed on 2023-05-12).

[19] MathWorks, "ilu," [Online]. Available: `https://se.mathworks.com/help/matlab/ref/ilu.html` (Accessed on 2023-05-11).

[20] MathWorks, "gmres," [Online]. Available: `https://se.mathworks.com/help/matlab/ref/gmres.html` (Accessed on 2023-05-11).

[21] MathWorks, "rmse," [Online]. Available: `https://se.mathworks.com/help/matlab/ref/rmse.html` (Accessed on 2023-05-16).

A

Files and Inputs

The software is made of the following files and directories, that should all be located under the same root directory:

- `main.m`: a `MatLab` M-file containing the main code. All settings used by the solver are specified inside this file, as the fields of the struct called `opts`. The options are thoroughly described in the form of comments in the file.

- `init_main.m`: a `MatLab` M-file which performs all spatial node enumeration and which initializes all system matrices necessary to run the steady-state solver.

- `SteadyStateSolver.m`: a `MatLab` M-file which solves for the steady state.

- `init2_main.m`: a `MatLab` M-file which initializes most system matrices and other necessary variables to run the time-dependent solver.

- `init2_perturbation.m`: a `MatLab` M-file which initializes the perturbed system matrices in order to run the time-dependent solver.

- `input_files/[INPUT_ID]data/`: a `directory` which contains all necessary input files. All input files must be pre-fixed by a chosen unique identifier `INPUT_ID`. The total data-set must then be chosen in the options as the field of the struct `opts` called `INPUT_ID`, which is a string. For example, the development data-set has the `INPUT_ID` string `"03_"`, meaning all development data is located in the directory `input_files/03_data/`

Inside the directory `[INPUT_ID]data/` there can exist the following `.mat`-files:

- **(compulsory)** `[INPUT_ID]XS_data.mat`: a file containing the equally sized 3D-arrays `ABS1, ABS2, NUFIS1, NUFIS2, REM, D1, D2` and `KN`. These files should contain the spatial distribution of the node-averaged macroscopic cross-sections in the first five arrays, the diffusion coefficients in the next two arrays and the energy per fission event divided by the average number of neutrons per fission event, ν, in the last array. Node $n = n(I, J, K)$ should be located at index I, J, K. Notice that `NUFISg` should be given as $\nu\Sigma_{f,g,n}$, i.e. the fission cross-section multiplied by ν. The cross-sections should have units of inverse centimeters, the diffusion coefficients should have units of centimeters and the

final array should have units of Joules. This input file is identical (except for the presence of `KN`) to the input with the same name used by `CORE SIM` [9].

- **(compulsory)** `[INPUT_ID]GEOM_data.mat`: a file with the three variables `DX,DY` and `DZ`, specifying the side-lengths of each node in centimeters. This input file is identical to the input with the same name used by `CORE SIM` [9].

- **(compulsory)** `[INPUT_ID]PARAMETERS_data.mat`: a file containing twelve parameters necessary for several different calculations. The parameters are:
  `v1 & v2`: the fast and thermal neutron speeds in $\frac{[cm]}{[s]}$
  `Pwr`: the total equilibrium state power in [W]
  `Nu`: the average number of neutrons per fission event [-]
  `lambdaI, gammaI & I0_eq`: The decay constant $\frac{[1]}{[s]}$, fission yield [-] and the average concentration at equilibrium $\frac{[1]}{[cm^3]}$ of ^{135}I.
  `sigmaX,lambdaX, gammaX & X0_eq`: the microscopic thermal absorption cross-section $[cm^2]$, the decay constant $\frac{[1]}{[s]}$, fission yield [-] and the average concentration at equilibrium $\frac{[1]}{[cm^3]}$ of ^{135}Xe.
  `dSIGa_dphi`: a number with units $[cm][s]$ which will be used in place of the space and group dependent quantities $-\frac{\partial \Sigma_{a,g'}(\vec{\mathbf{r}})}{\partial \phi_g}$ appearing in the feedback term.

- **(optional)** `[INPUT_ID]STEADYSTATE_data.mat`: a file with the results obtained from the steady state solver. The user can choose to save this file when possible and choose to load this file if available in order to directly start the time-evolution part of the simulation.

- **(optional)** `[INPUT_ID]CRBANKS_data.mat`: a file containing a single 4D array called `CONTROLRODBANKS`. The first three dimensions must be of the same size as the arrays from `XS_data.mat`, and the fourth dimension denotes the different control rod banks. If the node $n = n(I, J, K)$ belongs to bank m, the element `CONTROLRODBANKS(I,J,K,m)` must equal 1. All other elements must equal 0.

- **(optional)** `[INPUT_ID]dXS_data.mat`: a file containing the single numbers `dABS1, dABS2, dNUFIS1, dNUFIS2, dREM, dD1, dD2`. These are the fractions used to determine the change in the material properties in a node due to the presence of a control rod in the said node. These unitless numbers are described at the end of section 3.3.3.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY