

# Simulations of Adjoint Neutronics and Transient Thermal Hydraulics using a GPU-Accelerated Filter-Matrix Lattice Boltzmann Tool for Multiphysics of a Molten Salt Fast Reactor Core

Master Thesis

Sweder Willem Schelto Max Blanken

# Simulations of Adjoint Neutronics and Transient Thermal Hydraulics using a GPU-Accelerated Filter-Matrix Lattice Boltzmann Tool for Multiphysics of a Molten Salt Fast Reactor Core

## Master Thesis

by

Sweder Willem Schelto Max Blanken

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Friday September 4, 2026 at 10:30 AM.

Student number: 5082128  
Project duration: November 12, 2025 – June 21, 2026  
Thesis committee: dr. ir. M. Rohde, TU Delft, supervisor

Cover: Section of a downward propagating Kármán vortex street created by pressure-driven flow around a square obstacle with Reynolds number 100.

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.



# Highlights

*The most important advancements made in this work are:*

- The implementation of the adjoint neutron flux (i.e. importance) and adjoint precursor concentration equations for circulating reactor fuel in a Filter-Matrix Lattice Boltzmann (FM-LBM) framework, derived from the forward neutron diffusion and precursor advection-diffusion equations.*
- The successful implementation of the Predictor-Corrector Quasi-Static method in the FM-LBM algorithm to model transient neutronics. Using the adjoint neutron flux as the appropriate weight function, the current multiphysics tool showed increased accuracy in modelling transient reactor core behaviour.*
- A GPU-friendly approach to handling complex geometries in the streaming step of the LBM through lookup tables (LUTs) for boundary nodes, eliminating the need for ghost nodes. This capability was validated against a channel-flow benchmark with a square obstacle. Together with further GPU optimisations, such as group and stack-allocated static arrays for the filter-matrix collisions, the LUT approach also drastically improves the computational performance of the algorithm compared to previous LBM studies.*

# Abstract

This research presents the development of a GPU-accelerated multiphysics simulation tool to model the complex thermal hydraulics and neutronics inside a Molten Salt Fast Reactor (MSFR) core. This is done by implementing a Filter-Matrix Lattice Boltzmann Method (FM-LBM) algorithm for the thermal hydraulics, neutron diffusion, and precursor advection-diffusion present in the molten salt. In this algorithm, the macroscopic quantities are represented by distribution functions, which evolve in time through repeated collision and streaming steps according to the equations and source terms that govern the overlying physics. By filtering out non-physical terms during the matrix multiplications in the collision step, the stability of the simulations is improved. Furthermore, as it mostly consists of local operations, the FM-LBM algorithm lends itself very well to parallel computing. Steady-state simulations of various single-physics and multiphysics problems demonstrate the correct implementation of the different physical fields in the FM-LBM framework, showing very strong agreement with the results reported by reference codes for different steps of the Tiberger benchmark, with average discrepancies of less than 1% for all observables along the domain centrelines.

To facilitate the modelling of complex geometries, a lookup table containing the pull-streaming information for every node in a stationary computational domain is constructed, which simplifies the task of the streaming and boundary kernels and eliminates the need for ghost nodes. This complex-geometry capability was validated against the benchmark by Breuer of channel flow around a square obstacle, reproducing the reference solutions accurately in both steady and unsteady flow regimes, including for boundaries shifted off the lattice through the interpolated bounce-back method. Additionally, the lookup table approach greatly improved the computational efficiency of the tool, achieving up to 1100% faster performance compared with previous LBM studies, and falling within the same range as studies focused on optimising GPU-accelerated LBM implementations.

Transient multiphysics simulations are performed using the Predictor-Corrector Quasi-Static Method (PC-QSM) for neutronics modelling. The correct implementation of this method uses the solution to the adjoint version of the neutron diffusion equation as a weight function, which was replaced with the forward neutron flux in previous work. Simplified simulations showed that the forward and adjoint neutron fluxes differ significantly in the case of multiple neutron energy groups, even for stationary fuel. Moreover, the choice of appropriate weight function has a large effect on the transient behaviour of neutronics when modelling a reactivity insertion. In circulating fuel the difference was found to be even greater, as the adjoint flux is coupled more strongly to the fuel salt convection due to the increased importance of circulating precursors. Using the correct adjoint flux as a weight function to model the transient case of the Tiberger benchmark, the current FM-LBM implementation showed closer agreement with the results reported by the reference codes than previous studies using LBM and PC-QSM for neutron diffusion. Having shown the importance of the adjoint neutron flux for transient modelling, this work offers a foundation for future research extending the multiphysics tool towards full-scale, three-dimensional reactor simulations using the lookup table approach for complex geometries, the modelling of neutron transport beyond the diffusion approximation, and the implementation of advanced physics effects such as turbulence modelling.

# Contents

|                                                                      |           |
|----------------------------------------------------------------------|-----------|
| <b>Highlights</b>                                                    | <b>i</b>  |
| <b>Abstract</b>                                                      | <b>ii</b> |
| <b>Nomenclature</b>                                                  | <b>xi</b> |
| <b>1 Introduction</b>                                                | <b>1</b>  |
| 1.1 Generation IV Reactors and the MSFR . . . . .                    | 1         |
| 1.2 Computational Modelling in MSFR Research . . . . .               | 3         |
| 1.2.1 Existing Literature . . . . .                                  | 3         |
| 1.3 Research Goals . . . . .                                         | 5         |
| 1.4 Thesis Outline . . . . .                                         | 5         |
| <b>2 Theory</b>                                                      | <b>6</b>  |
| 2.1 Thermal Hydraulics . . . . .                                     | 6         |
| 2.1.1 Fluid Dynamics . . . . .                                       | 6         |
| 2.1.2 Heat transfer . . . . .                                        | 7         |
| 2.1.3 Dimensionless numbers . . . . .                                | 8         |
| 2.1.4 Kinetic Theory . . . . .                                       | 9         |
| 2.2 Nuclear Reactor Physics . . . . .                                | 10        |
| 2.2.1 Neutron Transport . . . . .                                    | 10        |
| 2.2.2 Neutron Diffusion . . . . .                                    | 11        |
| 2.2.3 Reactor Dynamics . . . . .                                     | 14        |
| 2.2.4 Adjoint Neutron Diffusion . . . . .                            | 16        |
| 2.3 Multiphysics coupling . . . . .                                  | 19        |
| 2.3.1 Circulation of heat and precursors . . . . .                   | 19        |
| 2.3.2 Fission products and precursor decay . . . . .                 | 20        |
| 2.3.3 Fission heat and salt expansion feedback . . . . .             | 20        |
| 2.4 Parallel Programming on the GPU . . . . .                        | 21        |
| 2.4.1 General-purpose computation on GPUs . . . . .                  | 21        |
| 2.4.2 NVIDIA GPU Hardware Architecture & Memory Hierarchy . . . . .  | 22        |
| 2.4.3 The CUDA Programming Model . . . . .                           | 22        |
| 2.4.4 Performance Considerations . . . . .                           | 23        |
| <b>3 Numerical Method</b>                                            | <b>24</b> |
| 3.1 The Lattice Boltzmann Method . . . . .                           | 24        |
| 3.2 The Filter-Matrix Lattice Boltzmann Method . . . . .             | 26        |
| 3.2.1 Filter Matrix Implementations for the Coupled Fields . . . . . | 27        |
| 3.3 Boundary Conditions . . . . .                                    | 31        |
| 3.3.1 Periodic Boundaries . . . . .                                  | 31        |
| 3.3.2 Dirichlet Boundaries for Momentum . . . . .                    | 32        |
| 3.3.3 Boundary Conditions for Scalar Quantities . . . . .            | 33        |
| 3.4 Lattice Conversion . . . . .                                     | 34        |
| 3.5 FM-LBM for Coupled Steady-State Simulations . . . . .            | 35        |
| 3.6 FM-LBM for Adjoint Neutronics Simulations . . . . .              | 35        |
| 3.6.1 Filter Matrices for Adjoint Quantities . . . . .               | 35        |
| 3.6.2 FM-LBM solver for Adjoint Neutronics . . . . .                 | 36        |
| 3.7 FM-LBM and the Predictor-Corrector Quasi-Static Method . . . . . | 37        |
| 3.8 GPU Acceleration . . . . .                                       | 38        |
| 3.8.1 Julia-CUDA . . . . .                                           | 38        |
| 3.8.2 Race Conditions . . . . .                                      | 38        |

|          |                                                                    |           |
|----------|--------------------------------------------------------------------|-----------|
| 3.8.3    | Performance Gains                                                  | 39        |
| 3.9      | Handling Complex Geometries with Lookup Tables                     | 41        |
| 3.9.1    | Interpolated Bounce-Back Method                                    | 41        |
| 3.9.2    | Lookup Table Construction                                          | 41        |
| 3.9.3    | Pull-streaming and Sentinel Handling                               | 42        |
| <b>4</b> | <b>Validation of Steady-State Multiphysics Models</b>              | <b>46</b> |
| 4.1      | Description of the Tiberga Benchmark                               | 46        |
| 4.1.1    | Reference Code Packages                                            | 47        |
| 4.1.2    | Error Quantification                                               | 48        |
| 4.2      | Single-Physics Simulations                                         | 48        |
| 4.2.1    | Step 0.1: Momentum                                                 | 48        |
| 4.2.2    | Step 0.2: Neutronics                                               | 50        |
| 4.2.3    | Step 0.3: Temperature                                              | 52        |
| 4.2.4    | Performance                                                        | 54        |
| 4.3      | Coupled Simulations                                                | 57        |
| 4.3.1    | Step 1.1: Circulating Fuel                                         | 57        |
| 4.3.2    | Step 1.2: Power Coupling and Salt Expansion Feedback               | 58        |
| 4.3.3    | Step 1.3: Buoyancy                                                 | 58        |
| 4.3.4    | Step 1.4: Full Coupling                                            | 60        |
| 4.4      | Fluid Dynamics Simulations in Complex Geometries                   | 64        |
| 4.4.1    | Description of the Breuer Benchmark                                | 64        |
| 4.4.2    | Flow patterns                                                      | 65        |
| 4.4.3    | Recirculation in Steady Flow                                       | 65        |
| 4.4.4    | Velocity profiles in Unsteady Flow                                 | 66        |
| 4.4.5    | Validation of the IBB implementation                               | 70        |
| <b>5</b> | <b>Validation of Transient Multiphysics Models</b>                 | <b>72</b> |
| 5.1      | Transient Neutronics Simulations                                   | 72        |
| 5.1.1    | Problem Description                                                | 72        |
| 5.1.2    | Steady-State Simulations of a 1D slab                              | 73        |
| 5.1.3    | Adjoint Flux Simulations                                           | 74        |
| 5.1.4    | Transient Flux Simulations                                         | 75        |
| 5.1.5    | Stability of the PC-QSM Implementation                             | 77        |
| 5.2      | Transient Coupled Simulations                                      | 78        |
| 5.2.1    | Problem Description                                                | 78        |
| 5.2.2    | Adjoint Simulations                                                | 78        |
| 5.2.3    | Transient Multiphysics Simulations                                 | 80        |
| <b>6</b> | <b>Conclusion &amp; Recommendations</b>                            | <b>82</b> |
| 6.1      | Complex Geometries                                                 | 82        |
| 6.2      | Computational Performance                                          | 83        |
| 6.3      | Transient FM-LBM Model Development                                 | 83        |
| 6.4      | Recommendations for Future Research                                | 84        |
|          | <b>References</b>                                                  | <b>85</b> |
| <b>A</b> | <b>Neutronics Data from the Tiberga Benchmark</b>                  | <b>90</b> |
| <b>B</b> | <b>Heatmaps of Tiberga Benchmark Observables</b>                   | <b>92</b> |
| B.1      | Single-Physics Simulations                                         | 92        |
| B.2      | Coupled Simulations                                                | 93        |
| <b>C</b> | <b>Snapshots of Tiberga Benchmark Transient Simulation Results</b> | <b>98</b> |

# List of Figures

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |    |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1  | Approximate layout of the Molten Salt Fast Reactor fuel circuit. The heat exchangers are denoted with 'HX'[10]. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 2  |
| 2.1  | Schematic of the main physical fields in an MSFR core and the effects coupling the fields. The omitted effects of decay heat and Doppler broadening are indicated with dashed arrows. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                           | 21 |
| 2.2  | Hardware architecture of an NVIDIA GPU, including the hierarchy of global, constant, shared, and register memory. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 22 |
| 2.3  | Schematic of the software abstractions used in the CUDA programming model. The number of threads in a warp is fixed at 32, but CUDA allows for the user to specify both the total number of blocks and the number of threads per block at every kernel launch. .                                                                                                                                                                                                                                                                                                                                                        | 23 |
| 3.1  | Common lattice velocity sets for LBM modelling in 2 (left) and 3 (right) dimensions. Note the rest velocity (with index 1) in all sets. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 25 |
| 3.2  | Visual representation of the collision and streaming step of the LBM for the distribution function $f_i$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 26 |
| 3.3  | Two discretisations of the same domain with link-wise (left) and wet-node (right) boundary conditions. Fluid nodes are illustrated as open circles, boundary nodes as solid circles. This schematic is taken from [18]. . . . .                                                                                                                                                                                                                                                                                                                                                                                         | 32 |
| 3.4  | Realisation of the periodic boundary conditions. Outwardly flowing populations at $x_N$ (marked in red) re-enter the domain on the opposite side at $x_1$ . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 32 |
| 3.5  | Schematic of the linear interpolation to determine the scalar neutron flux at the wall for the vacuum boundary condition of the neutron diffusion equation. Figure is inspired by [52]. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                         | 34 |
| 3.6  | The relation between the three time steps used in the Predictor-Corrector Quasi-Static Method. Figure is inspired by [71]. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 38 |
| 3.7  | Overview of the GPU-accelerated FM-LBM algorithm. The same structure holds for steady-state, adjoint, and transient simulations. The diagram shows the division of tasks between the CPU and GPU, and how the processing units share data. . . . .                                                                                                                                                                                                                                                                                                                                                                      | 39 |
| 3.8  | Illustration of the array flattening for a 2D array. Storing distribution functions with equal velocity directions contiguously optimises GPU performance through memory coalescing. Figure is inspired by [28]. . . . .                                                                                                                                                                                                                                                                                                                                                                                                | 40 |
| 3.9  | Schematic of the interpolated bounce-back scheme. For $q < 1/2$ , the distance $d$ between the wall (vertical line) and the boundary node $r_b$ is smaller than half a lattice spacing. For $q \geq 1/2$ , $d$ is larger than half a lattice spacing, and the endpoint of the streaming population (small black circle) lies between the wall and $r_b$ . Figure is inspired by [18]. .                                                                                                                                                                                                                                 | 41 |
| 3.10 | Example of a computational domain with complex geometries. The 'RPNM' letters form solid obstacles in a plane channel, and the surrounding nodes are classified as boundary nodes. The axes are expressed in lattice units. The corresponding flow field is shown in Figure 4.32. . . . .                                                                                                                                                                                                                                                                                                                               | 42 |
| 3.11 | Schematic of a 3D lattice, projected on the $x - z$ plane for clarity. In this projected plane, the fluid node at position $r$ (red) is surrounded by 7 fluid nodes (blue) and one solid node $r_s$ (black), separated from $r$ in the velocity direction $c_{11} = [1, 0, 1]^T$ as defined by the D3Q19 lattice velocity set (see Figure 3.1b). Therefore $r$ is classified as a boundary node in this scheme. However, in the D3Q7 scheme $c_{11}$ is not defined, so there is no streaming of particle distributions between $r$ and $r_s$ . Therefore $r$ is classified as a fluid node in the D3Q7 scheme. . . . . | 43 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.12 | Left: Adjacent configuration of solid node $\mathbf{r}_s$ (black), boundary node $\mathbf{r}_b$ (red), and fluid node $\mathbf{r}_f$ (blue). Right: Schematic of the lookup table used for the pull-streaming mechanism. For each index, the LUT contains the node class (fluid, solid, boundary), the index of the source node from which the distribution should be pulled, and (for boundary nodes) the fractional distance $q$ to the boundary in lattice units. . . . . | 43 |
| 3.13 | Solving scheme of the Filter-Matrix Lattice Boltzmann Method for coupled steady-state simulations. Note that source terms and other macroscopic variables (such as the salt expansion feedback or 'sef') are also calculated in the penultimate step of the outer LBM iteration (rightmost blue rectangle). . . . .                                                                                                                                                          | 44 |
| 3.14 | Solving scheme of the FM-LBM with the Predictor-Corrector Quasi-Static Method for transient neutronics. Note that in this research, the $\Delta t_S$ loop is replaced by a matrix multiplication as explained in Section 3.7. . . . .                                                                                                                                                                                                                                        | 45 |
| 4.1  | The domain of the Tiberga benchmark. The cavity is insulated, surrounded by vacuum, and filled with molten salt at an initial temperature of 900 K. Observables are pointwise compared along the $AA'$ and $BB'$ centrelines. Figure is inspired by [60]. . . . .                                                                                                                                                                                                            | 47 |
| 4.2  | The horizontal velocity component for Step 0.1 of the Tiberga benchmark case along the horizontal $AA'$ (left) and vertical $BB'$ (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                                                                                                                 | 49 |
| 4.3  | The vertical velocity component for Step 0.1 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                                                                                                                               | 49 |
| 4.4  | Discrepancy of the horizontal and vertical velocity components for Step 0.1 of the Tiberga benchmark case along the horizontal and vertical centrelines. For increasing spatial resolution, Figure 4.4a shows the discrepancy without temporal scaling, while diffusive scaling is employed in Figure 4.4b. . . . .                                                                                                                                                          | 50 |
| 4.5  | The fission rate density for Step 0.2 of the Tiberga benchmark case along the horizontal $AA'$ centreline. As the domain is symmetric in this step, the results along the vertical centreline are identical. The simulation results are shown alongside data from reference codes. . . . .                                                                                                                                                                                   | 51 |
| 4.6  | The fission rate density for Step 0.2 of the Tiberga benchmark case along the horizontal centreline, for both choices of extrapolated boundary length $\tilde{x}$ , alongside data from the reference codes. . . . .                                                                                                                                                                                                                                                         | 52 |
| 4.7  | Discrepancy of the temperature for Step 0.3 of the benchmark case along the horizontal and vertical centrelines as a function of the dimensionless Prandtl number. . . . .                                                                                                                                                                                                                                                                                                   | 53 |
| 4.8  | Pointwise discrepancy of the temperature for Step 0.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. . . . .                                                                                                                                                                                                                                                                                                                    | 53 |
| 4.9  | The temperature for Step 0.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                                                                                                                                               | 54 |
| 4.10 | The computational performance of the GPU-accelerated FM-LBM algorithm for Steps 0.1 (left) and 0.2 (right) of the Tiberga benchmark case. The results are shown alongside the performance reported by Pijls[43]. Here, $N_X$ is the number of grid points along the $x$ -axis, since the simulations take place on a 2-dimensional grid $N_{Grid} = N_X^2$ . . . . .                                                                                                         | 56 |
| 4.11 | The computational performance of the GPU-accelerated FM-LBM algorithm for Step 0.3 of the Tiberga benchmark case. The results are shown alongside the performance reported by Pijls[43]. . . . .                                                                                                                                                                                                                                                                             | 56 |
| 4.12 | The computational performance of the GPU-accelerated FM-LBM algorithm for the single-physics steps of the Tiberga benchmark case, expressed in Million Direction Updates per Second (MDUPS). The results are shown alongside the performance reported by Pijls[43]. . . . .                                                                                                                                                                                                  | 56 |
| 4.13 | The delayed neutron source for Step 1.1 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                                                                                                                                    | 57 |
| 4.14 | The temperature for Step 1.2 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                                                                                                                                               | 59 |

|      |                                                                                                                                                                                                                                                                                                                                                                              |    |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.15 | The change of fission rate density with respect to the results of Step 0.2 for Step 1.2 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                    | 59 |
| 4.16 | The horizontal velocity component for Step 1.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                             | 60 |
| 4.17 | The vertical velocity component for Step 1.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                               | 60 |
| 4.18 | The temperature for Step 1.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                                               | 61 |
| 4.19 | The delayed neutron source for Step 1.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes. . . . .                                                                                                                                                    | 61 |
| 4.20 | Schematic of the Breuer benchmark, with the centre of the square defined as the domain origin. The figure is inspired by [79]. . . . .                                                                                                                                                                                                                                       | 64 |
| 4.21 | Velocity fields around and behind the square for (top-to-bottom) Reynolds numbers 40, 70, 100, 200. The axes are scaled to the side length $D$ of the square, while the colorbar is scaled to the maximum inlet velocity $u_{max}$ . For clarity, the x-axis has been reported only once at the bottom. . . . .                                                              | 65 |
| 4.22 | Computed recirculation lengths $L_r$ for various Reynolds numbers. The linear fit reported by Breuer et al.[79] is added in red. . . . .                                                                                                                                                                                                                                     | 66 |
| 4.23 | Pointwise discrepancy of the horizontal and vertical velocity components of the Breuer benchmark case along the horizontal centrelines between $N_X = 1900$ and $N_X = 2000$ . The discrepancy is highest near the zero-crossings of the velocities (compare with Figure 4.24). . . . .                                                                                      | 67 |
| 4.24 | Streamwise (above) and cross-stream (below) velocity profiles at $t_{eval}$ along the horizontal centreline obtained by Breuer et al.[79] (left) and the current FM-LBM implementation (right). The vertical axes are scaled to the maximum inlet velocity $u_{max}$ . . . . .                                                                                               | 68 |
| 4.25 | Streamwise (above) and cross-stream (below) velocity profiles at $t_{eval}$ along the vertical centrelines ( $x = 0$ , $x = 4$ , and $x = 8$ ) obtained by Breuer et al.[79] (left) and the current FM-LBM implementation (right). . . . .                                                                                                                                   | 68 |
| 4.26 | Discrepancy of the horizontal and vertical velocity components of the Breuer benchmark case along the horizontal centreline for different grid lengths $N_X$ , presented on a linear (left) and logarithmic scale (right). . . . .                                                                                                                                           | 69 |
| 4.27 | Discrepancy of the horizontal velocity components of the Breuer benchmark case along vertical lines for different grid lengths $N_X$ , presented on a linear (left) and logarithmic scale (right). . . . .                                                                                                                                                                   | 69 |
| 4.28 | Discrepancy of the vertical velocity components of the Breuer benchmark case along vertical lines for different grid lengths $N_X$ , presented on a linear (left) and logarithmic scale (right). . . . .                                                                                                                                                                     | 69 |
| 4.29 | Schematic of the standard square obstacle boundaries in black, as well as a square shifted horizontally in the lattice, indicated with dashed red lines. Note that the $\delta_x$ shift can be positive or negative, corresponding to a shift to the right or left respectively. . . .                                                                                       | 70 |
| 4.30 | Streamwise velocity profile along the horizontal centreline for three shifts $\delta_x$ of the square obstacle, in full (left) and zoomed near the square region (scatter plot, right). Note that the horizontal axis's origin is positioned at the centre of the square with $\delta_x = 0$ . The vertical axis is scaled to the maximum inlet velocity $u_{max}$ . . . . . | 71 |
| 4.31 | Cross-stream velocity profile along the horizontal centreline for three shifts $\delta_x$ of the square obstacle, in full (left) and zoomed in the far-wake (right). . . . .                                                                                                                                                                                                 | 71 |
| 4.32 | Velocity field around the 'RPNM' letters for Reynolds number 40. . . . .                                                                                                                                                                                                                                                                                                     | 71 |
| 5.1  | Steady state distributions of the scalar neutron flux (left) and delayed neutron precursor concentrations (right) for a 1D slab with 1 neutron energy group and 8 precursor families. The one-speed flux can be fitted with a cosine function. . . . .                                                                                                                       | 73 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 5.2  | Steady state distributions of the scalar neutron flux (left) and delayed neutron precursor concentrations (right) for a 1D slab with 6 neutron energy groups and 8 precursor families.                                                                                                                                                                                                                                                                                                                                                      | 74 |
| 5.3  | Steady state distributions of the adjoint neutron flux (left) and adjoint precursors concentrations (right) for a 1D slab with 1 neutron energy group and 8 precursor families. The adjoint values (dashed lines) are shown together with the solutions to the forward simulations (full lines).                                                                                                                                                                                                                                            | 75 |
| 5.4  | Steady state distributions of the adjoint neutron flux (left) and adjoint precursors concentrations (right) for a 1D slab with 6 neutron energy groups and 8 precursor families. The adjoint values (dashed lines) are shown together with the solutions to the forward simulations (full lines).                                                                                                                                                                                                                                           | 75 |
| 5.5  | Neutron amplitude response for a 1D slab with 6 neutron energy groups and 8 precursor families for a reactivity insertion of $\rho_n = 0.1\beta_{tot}$ at $t = 1$ s. The blue and red lines correspond to the use of the steady state forward flux $\phi_0$ or adjoint flux $\Phi_0^\dagger$ as a weight function $W$ respectively. The reference solution for the PK equations obtained by the forward Euler method is shown in orange (dashed line). Left: time evolution from 0.9 s to 1.1 s. Right: time evolution from 0.1 s to 100 s. | 76 |
| 5.6  | Stable (grey) and unstable (black) configurations for the PC-QSM in the 1D slab problem using different quantities of medium time steps per large time step ( $N_{TM}$ ) and number of small time steps per medium time step ( $N_{TS}$ ). The fit corresponding to $N_{TM} \times N_{TS} = 155$ is added in red.                                                                                                                                                                                                                           | 77 |
| 5.7  | The forward fission rate density (left) and delayed neutron source (right) for Step 1.4 of the Tiberger benchmark case. Please note that the colorbar scale for each graph is different.                                                                                                                                                                                                                                                                                                                                                    | 79 |
| 5.8  | The adjoint fission rate density (left) and delayed neutron source (right) for Step 1.4 of the Tiberger benchmark case. Please note that the colorbar scale for each graph is different.                                                                                                                                                                                                                                                                                                                                                    | 79 |
| 5.9  | Snapshots of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberger benchmark case. Left: $f = 0.025$ Hz. Right: $f = 0.2$ Hz. Note the different scaling of the $x$ -axis for each graph.                                                                                                                                                                                                                                                                      | 80 |
| 5.10 | Gain of the frequency response of the power production $P$ of the transient Tiberger benchmark as a function of an oscillating heat transfer coefficient $\gamma_{th}$ . The simulation results are shown alongside the data from the reference codes and previous LBM studies.                                                                                                                                                                                                                                                             | 81 |
| 5.11 | Phase shift of the frequency response of the power production $P$ of the transient Tiberger benchmark as a function of an oscillating heat transfer coefficient $\gamma_{th}$ . The simulation results are shown alongside the data from the reference codes and previous LBM studies.                                                                                                                                                                                                                                                      | 81 |
| B.1  | Velocity field for Step 0.1 of the Tiberger benchmark case. Streamlines added in the right figure denote the flow direction of the molten salt.                                                                                                                                                                                                                                                                                                                                                                                             | 92 |
| B.2  | Fission rate density for Step 0.2 of the Tiberger benchmark case. Isoline intervals of $2.0 \times 10^{18} \text{ m}^{-3} \text{ s}^{-1}$ are added.                                                                                                                                                                                                                                                                                                                                                                                        | 93 |
| B.3  | Temperature field for Step 0.3 of the Tiberger benchmark case. Isoline intervals of 50 K are added.                                                                                                                                                                                                                                                                                                                                                                                                                                         | 93 |
| B.4  | Delayed neutron source for Step 1.1 of the Tiberger benchmark case. Isoline intervals of $2.5 \times 10^{16} \text{ m}^{-3} \text{ s}^{-1}$ are added.                                                                                                                                                                                                                                                                                                                                                                                      | 93 |
| B.5  | Change of fission rate density (left) and temperature (right) for Step 1.2 of the Tiberger benchmark case. Isoline intervals of $2.0 \times 10^{17} \text{ m}^{-3} \text{ s}^{-1}$ and 50 K are added to the left and right figure respectively.                                                                                                                                                                                                                                                                                            | 94 |
| B.6  | Velocity field for Step 1.3 of the Tiberger benchmark case. Streamlines added in the right figure denote the flow direction of the molten salt.                                                                                                                                                                                                                                                                                                                                                                                             | 94 |
| B.7  | Delayed neutron source (left) and temperature field (right) for Step 1.3 of the Tiberger benchmark case. Isoline intervals of $2.5 \times 10^{16} \text{ m}^{-3} \text{ s}^{-1}$ and 50 K are added to the left and right figure respectively.                                                                                                                                                                                                                                                                                              | 94 |
| B.8  | Velocity field for Step 1.4 of the Tiberger benchmark case. For every reactor power, simulations were run with lid velocity $U_{lid} = [0.1, 0.3, 0.5] \text{ m s}^{-1}$ shown in the top, middle, and bottom row respectively. Streamlines denote the flow direction of the molten salt.                                                                                                                                                                                                                                                   | 95 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| B.9  | Temperature field for Step 1.4 of the Tiberger benchmark case. For every reactor power, simulations were run with lid velocity $U_{lid} = [0.1, 0.3, 0.5] \text{ m s}^{-1}$ shown in the top, middle, and bottom row respectively. Isoline intervals of 25 K, 33 K, and 50 K are added to the left, middle, and right column respectively. Note that the colorbar scale for each column is different. . . . .                                                                                                                                           | 96 |
| B.10 | Distribution of the first family of delayed neutron precursors for Step 1.4 of the Tiberger benchmark case. For every reactor power, simulations were run with lid velocity $U_{lid} = [0.1, 0.3, 0.5] \text{ m s}^{-1}$ shown in the top, middle, and bottom row respectively. Isoline intervals of $2.0 \times 10^{16} \text{ m}^{-3}$ , $3.3 \times 10^{16} \text{ m}^{-3}$ , and $5.0 \times 10^{16} \text{ m}^{-3}$ are added to the left, middle, and right column respectively. Note that the colorbar scale for each column is different. . . . | 97 |
| C.1  | Snapshot of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberger benchmark case, with $f = 0.0125 \text{ Hz}$ . . . . .                                                                                                                                                                                                                                                                                                                                                   | 98 |
| C.2  | Snapshots of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberger benchmark case. Left: $f = 0.05 \text{ Hz}$ . Right: $f = 0.1 \text{ Hz}$ . Note the different scaling of the $x$ -axis for each graph. . . . .                                                                                                                                                                                                                                                         | 99 |
| C.3  | Snapshots of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberger benchmark case. Left: $f = 0.4 \text{ Hz}$ . Right: $f = 0.8 \text{ Hz}$ . Note the different scaling of the $x$ -axis for each graph. . . . .                                                                                                                                                                                                                                                          | 99 |

# List of Tables

|      |                                                                                                                                                                                                                                       |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1  | Dimensionless numbers used in this research, along with their formulas and physical interpretations. . . . .                                                                                                                          | 8  |
| 2.2  | Overview of the relevant memory types of an NVIDIA GPU[61]. . . . .                                                                                                                                                                   | 22 |
| 4.1  | Fuel salt composition[60]. . . . .                                                                                                                                                                                                    | 47 |
| 4.2  | Salt thermodynamic and transport properties. . . . .                                                                                                                                                                                  | 47 |
| 4.3  | Tiberga benchmark steps. A white square ( $\square$ ) denotes a field that is actively solved in the step, while a black square ( $\blacksquare$ ) denotes a field that is prescribed or held fixed from a previous step. . . . .     | 48 |
| 4.4  | Details of the reference code packages.[74, 75, 15, 16] . . . . .                                                                                                                                                                     | 48 |
| 4.5  | Simulation parameters for Step 0.1 of the Tiberga benchmark. $N_X$ and $N_Y$ denote the number of lattice points. . . . .                                                                                                             | 49 |
| 4.6  | Simulation parameters for Step 0.2 of the Tiberga benchmark. . . . .                                                                                                                                                                  | 51 |
| 4.7  | Simulation results for the reactivity for step 0.2 of the Tiberga benchmark case, alongside results from earlier LBM studies and the reference codes. . . . .                                                                         | 51 |
| 4.8  | Simulation parameters for Step 0.3 of the Tiberga benchmark. . . . .                                                                                                                                                                  | 54 |
| 4.9  | Simulation parameters for Step 1.1 of the Tiberga benchmark. Note that $D_p$ denotes the mass diffusivity of the delayed neutron precursors. . . . .                                                                                  | 57 |
| 4.10 | Simulation results for the reactivity difference between Steps 1.1 and 0.2 of the Tiberga benchmark case, alongside results from earlier LBM studies and the reference codes. . . . .                                                 | 58 |
| 4.11 | Simulation results for the reactivity difference between Steps 1.2 and 1.1 of the Tiberga benchmark case, alongside results from earlier LBM studies and the reference codes. . . . .                                                 | 59 |
| 4.12 | Simulation results for the reactivity difference between Steps 1.3 and 0.2 of the Tiberga benchmark case, alongside results from earlier LBM studies and the reference codes. . . . .                                                 | 61 |
| 4.13 | Simulation results for the reactivity difference between Steps 1.4 and 0.2 of the Tiberga benchmark case for different lid velocities and reactor powers, alongside results from earlier LBM studies and the reference codes. . . . . | 63 |
| A.1  | Definition of the neutron energy groups, along with the total and fission cross sections. . . . .                                                                                                                                     | 90 |
| A.2  | $P_0$ scattering cross sections. . . . .                                                                                                                                                                                              | 90 |
| A.3  | $P_1$ -corrected diffusion coefficients and inverse neutron velocities. . . . .                                                                                                                                                       | 90 |
| A.4  | Average total number of neutrons emitted per fission event, prompt and delayed fission spectra, and average energy emitted per fission event. . . . .                                                                                 | 91 |
| A.5  | Decay constant and fraction for each family of delayed neutron precursors. . . . .                                                                                                                                                    | 91 |

# Nomenclature

## Abbreviations

| Abbreviation | Definition                              |
|--------------|-----------------------------------------|
| ABB          | Anti-Bounce-Back                        |
| ADE          | Advection Diffusion Equation            |
| CFD          | Computational Fluid Dynamics            |
| CPU          | Central Processing Unit                 |
| CUDA         | Compute Unified Device Architecture     |
| FM           | Filter-Matrix                           |
| GIF          | Generation IV International Forum       |
| GPU          | Graphics Processing Unit                |
| HBB          | Halfway Bounce-Back                     |
| IBB          | Interpolated Bounce-Back                |
| LBE          | Lattice Boltzmann Equation              |
| LBM          | Lattice Boltzmann Method                |
| LUT          | Lookup Table                            |
| FEM          | Finite Element Method                   |
| FVM          | Finite Volume Method                    |
| MLUPS        | Million Lattice Updates per Second      |
| MSFR         | Molten Salt Fast Reactor                |
| MSR          | Molten Salt Reactor                     |
| NDE          | Neutron Diffusion Equation              |
| NSE          | Navier-Stokes Equations                 |
| NTE          | Neutron Transport Equation              |
| PC-QSM       | Predictor-Corrector Quasi-Static Method |
| PK           | Point Kinetics                          |
| SEF          | Salt Expansion Feedback                 |
| SM           | Streaming Multiprocessor                |

## Symbols

| Symbol         | Definition                                  | Unit                             |
|----------------|---------------------------------------------|----------------------------------|
| $B$            | Ratio of bulk to kinematic viscosity        |                                  |
| $\mathbf{c}_i$ | Discrete lattice velocity in direction $i$  | $\text{m s}^{-1}$                |
| $c_s$          | Lattice speed of sound                      | $\text{m s}^{-1}$                |
| $C_d$          | Delayed neutron precursor concentration     | $\text{m}^{-3}$                  |
| $C_p$          | Specific heat capacity                      | $\text{J kg}^{-1} \text{K}^{-1}$ |
| $D_g$          | Neutron diffusion coefficient               | $\text{m}$                       |
| $D_p$          | Precursor diffusion coefficient             | $\text{m}^2 \text{s}^{-1}$       |
| $E$            | Energy                                      | $\text{J}$                       |
| $E_{ik}$       | Filter Matrix                               | —                                |
| $f$            | Frequency                                   | $\text{Hz}$                      |
| $\mathbf{f}$   | Specific body force                         | $\text{N kg}^{-1}$               |
| $f_i$          | Particle distribution function for momentum | —                                |
| $g$            | Gravitational acceleration of the Earth     | $\text{m s}^{-2}$                |
| $g_i$          | Particle distribution function for enthalpy | —                                |

| Symbol                | Definition                                                 | Unit                            |
|-----------------------|------------------------------------------------------------|---------------------------------|
| $h$                   | Enthalpy                                                   | $\text{J m}^{-3}$               |
| $k$                   | Multiplication factor                                      | —                               |
| $m_d$                 | Precursor amplitude function                               | —                               |
| $n$                   | Neutron amplitude function                                 | —                               |
| $n_{i,g}$             | Particle distribution function for scalar neutron flux     | —                               |
| $p$                   | Pressure                                                   | $\text{N m}^{-2}$               |
| $p_{i,d}$             | Particle distribution function for precursor concentration | —                               |
| $P$                   | Power                                                      | W                               |
| $Pr$                  | Prandtl number                                             | —                               |
| $q$                   | Heat source or sink                                        | $\text{W m}^{-3}$               |
| $\mathbf{r}$          | Space coordinate                                           | m                               |
| $Re$                  | Reynolds number                                            | —                               |
| $Sc$                  | Schmidt number                                             | —                               |
| $t$                   | Time                                                       | s                               |
| $T$                   | Temperature                                                | K                               |
| $\mathbf{u}$          | Fluid velocity                                             | $\text{m s}^{-1}$               |
| $v$                   | Neutron velocity                                           | $\text{m s}^{-1}$               |
| $W$                   | Weight function                                            | —                               |
| $\alpha$              | Thermal diffusivity                                        | $\text{m}^2 \text{s}^{-1}$      |
| $\alpha_k^\pm$        | Momentum solution vector                                   | —                               |
| $\beta_d$             | Delayed neutron fraction                                   | —                               |
| $\beta_k^\pm$         | Enthalpy solution vector                                   | —                               |
| $\beta_{th}$          | Thermal expansion coefficient                              | $\text{K}^{-1}$                 |
| $\gamma$              | Solution vector for criticality eigenvalue problem         | —                               |
| $\gamma_k^\pm$        | Neutron flux solution vector                               | —                               |
| $\gamma^{(l)}$        | Solution vector for power iteration                        | —                               |
| $\gamma_{th}$         | Volumetric heat transfer coefficient                       | $\text{W m}^{-3} \text{K}^{-1}$ |
| $\delta_{k,d}^\pm$    | Precursor concentration solution vector                    | —                               |
| $\varepsilon$         | Discrepancy                                                | —                               |
| $\epsilon$            | Point-wise discrepancy                                     | —                               |
| $\kappa$              | Thermal conductivity                                       | $\text{W m}^{-1} \text{K}^{-1}$ |
| $\lambda_d$           | Precursor decay constant                                   | $\text{s}^{-1}$                 |
| $\Lambda$             | Prompt neutron generation time                             | s                               |
| $\nu$                 | Kinematic viscosity                                        | $\text{m}^2 \text{s}^{-1}$      |
| $\nu_n$               | Neutron multiplicity                                       | —                               |
| $\rho$                | Density                                                    | $\text{kg m}^{-3}$              |
| $\rho_n$              | Nuclear reactivity                                         | —                               |
| $\boldsymbol{\sigma}$ | Stress tensor                                              | —                               |
| $\sigma$              | Microscopic nuclear cross section                          | $\text{m}^2$                    |
| $\Sigma$              | Macroscopic nuclear cross section                          | $\text{m}^{-1}$                 |
| $\phi$                | Scalar neutron flux                                        | $\text{m}^{-2} \text{s}^{-1}$   |
| $\tilde{\phi}$        | Neutron shape function                                     | $\text{m}^{-2} \text{s}^{-1}$   |
| $\Phi^\dagger$        | Adjoint neutron flux                                       | $\text{m}^{-2} \text{s}^{-1}$   |
| $\varphi$             | Angular neutron flux                                       | $\text{m}^{-2} \text{s}^{-1}$   |
| $\chi$                | Neutron spectrum                                           | —                               |
| $\omega_i$            | Numerical weight                                           | —                               |
| $\Omega$              | Collision operator                                         | —                               |
| $\Omega_f$            | Fluid domain                                               | —                               |
| $\Omega_s$            | Solid domain                                               | —                               |
| $\hat{\Omega}$        | Angular direction                                          | —                               |
| $\theta_{k,g}^\pm$    | Adjoint neutron flux solution vector                       | —                               |
| $\boldsymbol{\xi}$    | Particle velocity                                          | $\text{m s}^{-1}$               |
| $\zeta_{k,d}^\pm$     | Adjoint precursor concentration solution vector            | —                               |

| Subscript          | Definition                               |
|--------------------|------------------------------------------|
| $d$                | Delayed precursor family                 |
| $eff$              | Effective                                |
| $f$                | Fission                                  |
| $g$                | Neutron energy group                     |
| $g' \rightarrow g$ | Scattering from energy group $g'$ to $g$ |
| $i$                | Lattice direction                        |
| $N$                | Neutronics timescale                     |
| $ph$               | Quantity in physical units               |
| $ref$              | Reference value                          |
| $s$                | Scattering                               |
| $t$                | Total removal                            |
| $TH$               | Thermal hydraulics timescale             |
| $0$                | Initial / Steady-state value             |

| Superscript | Definition                                 |
|-------------|--------------------------------------------|
| $d$         | Delayed                                    |
| $eq$        | Equilibrium                                |
| $p$         | Prompt                                     |
| $\dagger$   | Adjoint                                    |
| $*$         | Post-collision / Quantity in lattice units |

# 1

## Introduction

Ever since the first self-sustaining fission chain reaction was demonstrated in Chicago Pile-1 in 1942[1], nuclear fission has been pursued as a source of power generation. The design and analysis of different types of nuclear reactors and power plants have been the topic of much research and development. It can be safely said that these efforts have borne fruit. As of today, nuclear energy accounts for roughly 9% of global electricity generation, while it makes up around 23% of Europe's power supply[2]. Recent years have marked a turning point in the global position towards nuclear energy. Accelerated climate policies and the increased pressure to decarbonise have reopened the nuclear debate after decades in which concerns about safety, hazardous waste, and proliferation kept it sidelined. More countries are turning to nuclear power to accelerate their carbon-footprint reduction[3]. One recent noteworthy example is the founding of the state-owned 'Nucleaire Energie Organisatie' (NEO NL) in February 2026, which is tasked with the construction of two new nuclear power plants in the Netherlands[4].

This renewed interest is partly driven by the fact that the power supply provided by renewable alternatives such as solar and wind is subject to weather fluctuations. Meanwhile, the rapid rise of energy-intensive industries like datacentres and artificial intelligence will only further increase the global electricity demand[5]. For these reasons, the constant baseload generation that nuclear power plants offer has become especially attractive.

One of the most common types of nuclear plants currently in commercial operation is the pressurised water reactor (PWR). These reactors have a long operational track record and a well-developed regulatory framework. In the reactor cores, purified water is used both as a neutron moderator and coolant fluid. To keep the fission-heated water in a liquid state, PWRs operate at pressures of around 150 bar[6], which requires very strong materials. This results in high construction costs and contributes to the risk of loss-of-coolant accidents. Other drawbacks of PWRs include the production of long-lived nuclear waste, particularly actinides, and fairly low operating temperatures ( $\approx 590$  K) to prevent the water from boiling, which limits thermal efficiency. These drawbacks have motivated the development of next-generation reactor concepts.

### 1.1. Generation IV Reactors and the MSFR

In 2001 the Generation IV International Forum (GIF) was founded with the aim of designing and developing the next generation of nuclear reactors. The selection criteria of the new designs included sustainability, safety & reliability, and proliferation resistance. A total of six reactor technologies have been selected for further research and development. One of these is the Molten Salt Reactor (MSR), which is distinguished by dissolving the fissile material directly in a molten fluoride or chloride salt[7]. The salt therefore serves both as the fuel and the primary coolant. This configuration eliminates the classical core meltdown scenario by construction, as the fuel is already in a liquid state. The boiling temperature of the molten salt at atmospheric pressure is much higher than the operating temperature of the MSR, around 1000 K. The reactor can therefore operate without the high system pressures required by light-water reactors to keep the coolant liquid, eliminating the need for costly pressure

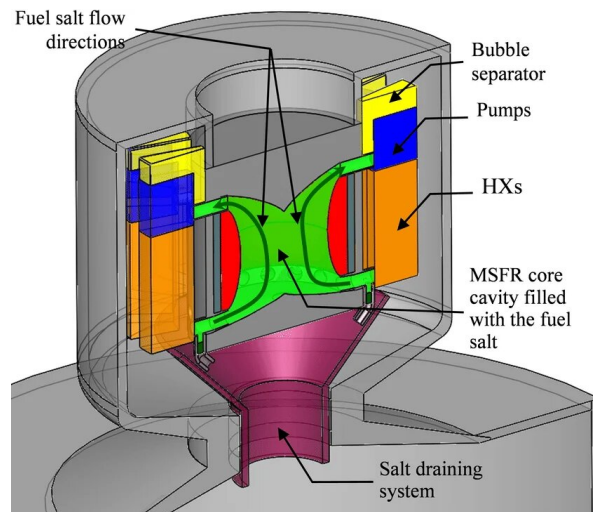


**Figure 1.1:** Approximate layout of the Molten Salt Fast Reactor fuel circuit. The heat exchangers are denoted with 'HX'[10].

vessels. The higher operating temperature of the MSR also yields higher thermal efficiencies than the PWR. Furthermore, the high-grade heat produced by MSRs can be processed for various industrial applications. Additionally, gaseous fission products that are not soluble in the molten salt can be captured through the injection of helium gas bubbles and removed from the system. This enables the continuous, online reprocessing of the fuel, eliminating the need for periodic refuelling shutdowns inherent to conventional, solid-fuelled reactors. The design of molten salt reactors also includes numerous passive safety features, such as the leakage of neutrons from the core due to expansion of the liquid salt in case of overheating. This reduces the fission rate and prevents criticality accidents. Moreover, some MSR designs are equipped with a 'freeze plug' at the bottom of the reactor core. This plug is a block of frozen salt and is actively cooled during normal reactor operation. If the core overheats, the plug melts which allows the molten salt to drain into emergency storage tanks below, terminating the fission chain reaction.

One variant of the molten salt reactor is the molten salt *fast* reactor (MSFR), developed primarily within the EU SAMOFAR / SAMOSAFER framework[8]. These reactors can operate within the fast neutron spectrum, which means that neutrons with average energies above 1 MeV help to sustain the fission chain reaction, in contrast to slower thermal neutrons ( $\approx 25$  meV) in other types of reactors. Combined with an initial presence of fertile isotopes in the fuel salt, this allows for a breeder reactor that produces more fissile material than it consumes. The fast neutron spectrum also makes the MSFR capable of burning long-lived transuranic actinides as fuel, recycling the waste produced by PWRs and other light-water reactors. The numerous compositions of the fuel salt make molten salt reactors adaptable to a variety of fuel cycles, such as that of thorium, which is more abundant in the Earth's crust than uranium[9]. Figure 1.1 shows a concept layout of the reactor core of a molten salt fast reactor[10]. Another key design feature that distinguishes the circulating fuel-configuration of the MSR from solid-fuel reactors is the convection of delayed neutron precursors. As these fission products are carried with the salt flow, a fraction of them decays outside the active core region, affecting the reactivity of the system. The distribution of delayed neutron sources is therefore intrinsically coupled to the flow field, which ties the neutronics and thermal hydraulics of an MSR together.

The research of molten salt use in nuclear reactors dates back to the mid-twentieth century, where energy-dense reactors were tested as possible propulsion techniques for supersonic aircraft[11]. The Molten-Salt Reactor Experiment at Oak Ridge National Laboratories in the 1960s demonstrated the design concepts and technical feasibility of a molten salt reactor[12], and its results still form much of the basis for current MSR research. With the renewed interest of MSR technologies through the GIF programme, in part because of the aforementioned advantages, research and development projects have expanded across multiple countries. One contemporary example is the TMSR-LF1 reactor in China, in which critical operation of the molten salt reactor was reported, as well as nuclear breeding using fertile thorium[13].

## 1.2. Computational Modelling in MSFR Research

The liquid state of the MSFR fuel makes the system well suited for modelling as a computational fluid dynamics (CFD) problem. However, simulating the full multiphysics, including thermal hydraulics, neutron transport, and precursor flow, has faced limitations with conventional codes. In particular, accounting for the feedback mechanisms between the coupled physical fields presents significant challenges for traditional approaches. These feedback mechanisms include the convection of neutron-emitting fission products and the effect of heated salt expansion on the neutron cross sections and diffusion coefficients. Conventional reactor transport codes are typically built for stationary, solid fuel modelling, while CFD codes are primarily used for non-reacting fluids. Several tools have therefore been developed to solve both the thermal hydraulics and the neutronics within a single numerical framework[14]. For example, the OpenFOAM-based GeN-Foam solver couples a mesh-based multi-group neutron diffusion sub-solver to a finite-volume thermal-hydraulics sub-solver within one platform[15]. At TU Delft, Tiberga et al.[16, 17] developed an in-house tool that couples the PHANTOM- $S_N$  transport code and the Navier-Stokes solver DGFLOWS, built on the Discontinuous Galerkin Finite Element discretisation. These approaches demonstrate that a unified single-framework treatment of MSFR multiphysics is well established for both FEM and FVM.

This research explores an alternative unified framework, using the Lattice Boltzmann Method (LBM). In contrast to conventional Navier-Stokes solvers like Finite Element Methods or particle-based solvers like molecular dynamics, which treat the fluid at the macroscopic and microscopic scale respectively, LBM operates on a mesoscopic level, modelling the evolution of particle distributions on a discrete level. This method offers several advantages in the modelling of MSFR multiphysics. First, it is well suited to simulate mass-conserving flows in complex reactor geometries. Second, the heaviest numerical computations are performed locally through a collision-streaming process, which scales well using parallel computing methods. Third, with a minimal framework one can quickly incorporate additional fields that are hard to describe macroscopically and study the coupled effects[18].

### 1.2.1. Existing Literature

This research aims to further develop a multiphysics code to simulate the coupled neutronics and thermal hydraulics that govern the behaviour of the MSFR cores. Numerous works have demonstrated LBM's ability to model multiphysics problems in many applications and settings. In the context of transport phenomena in molten salt reactors, the LBM has been actively investigated within TU Delft's 'Reactor Physics and Nuclear Materials' group. Below is an overview of noteworthy findings inside and outside the research group.

#### LBM for Thermal Hydraulics Simulations

Although the Lattice Boltzmann Method was created initially as a numerical solver for the Navier-Stokes Equations which describe fluid flow, adaptations to the method have been made to model advection-diffusion equations. One notable advancement by He et al.[19] introduced an internal energy density distribution function to simulate the temperature field. This double distribution function (DDF) demonstrated a more numerically stable simulation compared with multispeed thermal LBM models. In place of the internal energy distribution for temperature, Chatterjee[20] introduced the enthalpy density distribution function, which facilitated the incorporation of external energy sources when solving the energy equation.

As the standard single-relaxation time (SRT) LBM model, which uses the Bhatnagar-Gross-Krook (BGK) operator[18], suffers from numerical instability, more advanced collision schemes have been developed over the years, such as the multi-relaxation time (MRT) scheme developed by d'Humières[21]. Another LB scheme is the Filter-Matrix Lattice Boltzmann Method (FM-LBM) developed by Somers[22], in which a non-linear collision operator acts on the distribution functions through multiplications of matrices and solution vectors that filter out non-physical terms that arise due to the discretisation of the governing equations. Zhuo et al. showed the successful implementation of the FM-LBM scheme to model fluid and gas flow in various settings, such as incompressible thermal flow[23], natural convection in a square cavity[24], microchannel gas flow[25], and fully developed turbulent channel flow[26]. In the RPNM group, Bus implemented the Filter-Matrix LBM with the enthalpy distribution approach to model phase changes for convective melting and solidification, which is a common phenomenon in the heat exchangers of molten salt reactors[27]. On the other hand, Entes used an FM-LBM implementation of

the temperature distribution function to model heat flow in a molten salt reactor core[28]. Sundström investigated theoretically how the  $k - \epsilon$  equations can be solved within the FM-LBM framework as a closure scheme for turbulent flow[29]. Terpstra started an initial implementation of  $k - \epsilon$  turbulence modelling using a moment projected Boltzmann collision operator with non-Cartesian boundaries, but reported significant instabilities in the turbulent simulations[30].

### LBM for Neutron Transport and Diffusion Simulations

Apart from fluid flow and thermal transport, the Lattice Boltzmann Method has also been used in some studies to model nuclear reactor physics. Erasmus confirmed the applicability of the LBM for neutron transport, and modified the collision scheme for his application to overcome blind spots in the model domain[31]. Bindra & Patil used the LBM framework to obtain the solution for the one-speed linear radiative or neutron transport equation[32]. Agarwal et al. used the neutron transport LBM for criticality problems in 1D and 2D geometries with multiple materials and multiple neutron energy groups[33]. By creating distribution functions for each neutron group, Wang et al. developed a family of Lattice Boltzmann Methods that differ in the neutronics equation they solve and in the specific capability each one adds. For neutron diffusion, a high-order Chapman-Enskog expansion yielded an accuracy well above that of conventional first-order LBM at a comparable computational cost[34]. For the full neutron transport problem, they proposed a Lattice Boltzmann model of the transient Boltzmann transport equation, in which each discrete direction is treated as a conservation law equation and solved independently[35]. For the  $SP_3$  equations, which are solved through a double-distribution function, the LBM was cast in an unstructured finite-volume form so that the solution can be obtained on complex, unstructured geometries[36]. These neutronics models were then progressively unified with thermal hydraulics, starting with a shared LBM data structure and boundary treatment for the heat equation and the  $S_N$ ,  $SP_3$ , and  $P_1$  approximations[37]. This unified model included the temperature feedback on the neutron cross sections. Simulating fluid flow using the large-eddy-based LB model yielded a fully coupled neutronics-thermal-hydraulics (NTH) tool for reactor multiphysics[38].

A naive simulation of neutron transport in combination with thermal hydraulics is computationally expensive, as the neutronics evolve on a very small timescale. Therefore, in later work[39] Wang et al. modelled transient neutron diffusion using the Predictor-Corrector Quasi-Static Method (PC-QSM), developed by Dulla et al.[40]. This method factorises the scalar neutron flux into a slowly changing shape function, which is solved over (relatively) big time steps, and a quickly changing amplitude function, which is solved over small time steps using the point-kinetic equations[41].

In the RPNM group, Polderman developed a multiphysics tool using the Lattice Boltzmann Method with the BGK collision operator to solve the Navier-Stokes equation, heat equation, as well as the neutron diffusion and advection-diffusion of delayed neutron precursors[42]. Her implementation produced steady-state simulation results that agreed well with benchmark codes, but showed reduced accuracy for transient multiphysics simulations using the PC-QSM algorithm. Pijls built on the work by Entes and Polderman to develop a (GPU-accelerated) Filter-Matrix LBM tool for multiphysics simulations, again using the PC-QSM algorithm for transient neutron diffusion[43]. Although his results agreed better with benchmark studies, Pijls' implementation still showed significant inaccuracies for transient neutronics, even in simplified 1D problems.

### GPU-accelerated methods for LBM modelling

In conventional methods for computational fluid dynamics, much of the complexity lies in determining derivative approximations non-locally from adjacent nodes, such as diffusive and convective terms[18]. On the other hand, the heaviest, non-linear computations in the Lattice Boltzmann Method are the collision steps, where density distribution functions on a single node are redistributed. This makes the LBM a very suitable method for specialised high-performance computing on parallel architectures, such as GPUs. For example, Tiribocchi et al. introduced a GPU-accelerated version of the LBM for efficient simulation of soft materials[44]. This is done by reconstructing distribution functions from available hydrodynamic variables, rather than storing the full set of discrete populations. Recently, Jaber et al.[45] presented a GPU-native LBM algorithm for incorporating stationary geometries into adaptive octree grids. To store accurate bounce-back boundary conditions, a lookup table of cut-link distances between fluid and solid cells is constructed. In the context of LBM for neutronics modelling, Wang et al. presented single- and multi-GPU acceleration methods for steady-state and transient simulations

of neutron diffusion[39] and neutron transport[46]. In their heterogeneous implementation, sequential tasks such as the initialisation, convergence judgement, and post-processing were realised on the CPU, while the parallel collision, streaming, and boundary kernel functions were executed on the GPU.

In the RPNM group, both Entes' and Pijls' FM-LBM implementations used GPU optimisation techniques in a Julia-CUDA framework to accelerate their multiphysics calculations. Entes reported satisfactory computational performance for natural convection simulations with 390 Million Lattice Updates Per Second (MLUPS)[28]. On the other hand, Pijls' multiphysics tool could only obtain 112 MLUPS for single-physics simulations of heat transport, significantly lower than reported in other literature studies.

### 1.3. Research Goals

The main goal of this thesis is to build on the previous work of the research group in the development of a multiphysics tool that combines thermal hydraulics and neutronics simulations into a single code base that is able to perform accurate steady-state and transient simulations of molten salt reactors. This is achieved by building GPU-accelerated FM-LBM algorithms for the thermal-hydraulics as well as neutron diffusion modelling in the Julia-CUDA framework. During the research, the following questions will be answered.

1. **Complex geometries:** *How can the multiphysics tool incorporate complex geometries?*
  - *How can complex geometries be implemented in the GPU-accelerated FM-LBM algorithm?*
  - *To what extent does the multiphysics tool reproduce accurate results for complex geometry flow simulations?*
2. **Computational performance:** *What is the computational performance of the multiphysics tool?*
  - *How can additional GPU optimisation techniques be implemented to achieve significant improvements in computational performance of the multiphysics tool compared to previous work?*
  - *How does the GPU-accelerated simulation tool compare to other GPU-accelerated tools from the literature?*
3. **Transient Multiphysics:** *How does the FM-LBM algorithm need to be changed to accurately model transient behaviour of an MSFR core?*
  - *What modifications to the current implementation of the PC-QSM algorithm are needed to produce accurate and stable results for transient neutronics?*
  - *How can the adjoint version of the neutron diffusion equation be correctly implemented in the current FM-LBM algorithm?*
  - *What is the effect of using the adjoint neutron flux as a weight function on the results of transient single-physics and multiphysics simulations?*

### 1.4. Thesis Outline

The rest of the thesis is outlined as follows. In Chapter 2, the necessary theoretical background will be given to understand the various principles that govern the dynamics of a molten salt reactor core. Next, a comprehensive explanation of this work's implementation of the (Filter-Matrix) Lattice Boltzmann Method to model the different physical fields is included in Chapter 3. Chapter 4 describes the research results of steady-state single-physics and multiphysics simulations. This chapter also includes the validation of fluid dynamics simulations in complex geometries. The importance of modelling the adjoint neutron flux for transient neutronics is demonstrated in Chapter 5, after which the results of perturbed, transient reactor simulations are presented and discussed. Conclusions and recommendations for future work will follow in Chapter 6.

# 2

## Theory

This chapter presents the theoretical background necessary to understand the physical principles that govern the complex behaviour of a molten salt reactor core. In Section 2.1 the fundamental concepts of hydraulic flow in thermal fluids are introduced, including a brief description of kinetic theory. Next, a comprehensive explanation of general nuclear reactor physics is given in Section 2.2. The coupled interactions between physical fields are discussed in Section 2.3. Lastly, Section 2.4 provides a short technical background on parallel programming using CUDA and GPU hardware basics.

Throughout this chapter and the rest of this thesis, various symbols are used to represent physical concepts or quantities, not all of which are given a complete explanation. The reader is therefore advised to consult the list of symbols at the beginning of the thesis in case of confusion.

### 2.1. Thermal Hydraulics

Thermal hydraulics is the study of fluid flow and heat transfer, combining thermodynamics, fluid mechanics, and heat transfer principles. It focuses on how heat influences moving fluids and conversely, how moving fluid influences convective heat transfer. This coupled study is critical for designing and analysing complex systems like molten salt reactors.

#### 2.1.1. Fluid Dynamics

To study the behaviour of fluids in terms of mathematical formulas, it is assumed that they can be viewed as a continuous mass distribution. In other words, the discrete interactions of atoms or molecules in the fluid can be ignored. This is known as the continuum approximation[47], and allows the fluid to be described in terms of macroscopic quantities such as its density  $\rho$ , velocity  $\mathbf{u}$ , and pressure  $p$ . In continuum physics, one can derive laws that state that certain measurable quantities cannot change over time in isolated physical systems. These conservation laws can then be used to study the dynamical behaviour of the system.

One of these conservation laws states that in a control volume, no mass can be created or destroyed. This means that any change in mass can only be the result of mass entering or leaving the volume. This conservation law is expressed as the continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \quad (2.1)$$

Another conserved property is momentum. A change in momentum in a control volume can only be the result of momentum entering or leaving the system, stresses at the volume's boundary, or body forces acting on the boundary. This conservation law is expressed as the Cauchy momentum equation:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \frac{1}{\rho} \nabla \cdot \boldsymbol{\sigma} + \mathbf{f}, \quad (2.2)$$

Where  $\sigma$  denotes the Cauchy stress tensor and  $\mathbf{f}$  denotes the specific body force. If it is assumed that the fluid is Newtonian (i.e. the viscosity is independent of the applied shear stresses), and the flow is incompressible (i.e.  $\nabla \cdot \mathbf{u} = 0$ ), then the momentum equation simplifies considerably. More concretely, these assumptions lead to the well-known Navier-Stokes Equations (NSE) for incompressible Newtonian fluids:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \quad (2.3)$$

where  $\nu$  denotes the kinematic viscosity of the fluid. Together with the continuity equation, the NSE form a complete set of equations for describing the dynamics of incompressible Newtonian fluids[47].

### 2.1.2. Heat transfer

Similarly to the conservation of mass and momentum, the energy of a system is also conserved. In a control volume, a change in energy can only be the result of energy flowing in or out of the volume (through convection or diffusion), or the production or destruction of energy through sources or sinks. By assuming that the rate of change of internal energy is proportional to the rate of change of the fluid's temperature  $T$ , and assuming that heat transfer is proportional to the negative gradient of temperature (known as Fourier's law[48]), the heat equation can be derived:

$$\rho C_p \frac{\partial T}{\partial t} + \rho C_p \mathbf{u} \cdot \nabla T = \nabla \cdot \kappa \nabla T + q, \quad (2.4)$$

where  $C_p$  denotes the specific heat capacity,  $\kappa$  the thermal conductivity, and  $q$  the heat source or sink term. From left to right, the terms in the heat equation denote the change of temperature, the convective heat transport, the diffusive heat transport, and the production or destruction of heat. Equation 2.4 follows from the full energy balance by assuming single-phase, incompressible flow, and neglecting viscous dissipation and pressure work[27]. Moreover, for constant heat conductivity,  $\kappa$  can be taken out of the divergence to give  $\kappa \nabla^2 T$ .

In the context of (turbulent) freezing of fluids in molten salt reactors, it is more convenient to describe a system's energy through its total enthalpy  $H$  and the thermal diffusivity  $\alpha = \frac{\kappa}{\rho C_p}$ . For single-phase flow there is no latent heat from phase changes, in which case the total enthalpy simplifies to the *sensible* enthalpy (per unit volume)  $h = \rho C_p T$ [49]. Furthermore, if the conductivity, density and specific heat capacity are assumed constant, then  $\alpha \neq f(\mathbf{x}, t)$ , and the heat equation for single-phase, incompressible flow can be rewritten as:

$$\frac{\partial h}{\partial t} + \mathbf{u} \cdot \nabla h = \alpha \nabla^2 h + q. \quad (2.5)$$

To accommodate for the study of freezing effects in further research, the heat transfer is modelled using Equation 2.5. Note that in the case of phase changes, varying heat capacities and thermal diffusivities affect heat transfer, as does the latent heat released or absorbed during the phase transition. Therefore Equation 2.5 should be modified to incorporate these effects, as is done in [50] and [27].

The heat equation is coupled to the NSE through the velocity field  $\mathbf{u}$ . To couple the temperature back to momentum, the Boussinesq approximation for buoyancy-driven flow is used. This approximation states that the fluid's density depends on its temperature, and that these variations are so small that  $\rho(T)$  can be linearised about a reference density  $\rho_{ref}$  corresponding to a reference temperature  $T_{ref}$  to obtain[18]:

$$\rho(T) \approx \rho_{ref} [1 - \beta_{th} (T - T_{ref})], \quad (2.6)$$

where  $\beta_{th}$  denotes the thermal expansion coefficient. In the presence of a gravitational field with acceleration  $\mathbf{g}$ , this variation in density will induce a buoyancy force term in the NSE:

$$\mathbf{F}_b = (\rho(T) - \rho_{ref})\mathbf{g} = -\beta_{th}\rho_{ref}(T - T_{ref})\mathbf{g}. \quad (2.7)$$

Moreover, these density variations are only significant in the buoyancy force term, but negligible in the other equations, i.e.  $\frac{\Delta\rho}{\rho} \ll 1$ . Therefore the thermal diffusivity  $\alpha = \frac{\kappa}{\rho C_p}$  can still be considered constant, even in buoyancy-driven flow.

### 2.1.3. Dimensionless numbers

When evaluating the behaviour of a fluid and its flow, it is often useful to study the ratios of different physical quantities and characteristics, rather than their absolute values. These ratios are known as dimensionless numbers, and they can form a powerful tool in the study of fluid dynamics. For example, one can perform a non-dimensional analysis of the NSE by introducing non-dimensional counterparts of the used variables:

$$\tilde{x} = \frac{x}{L}, \tilde{t} = \frac{t}{\tau}, \tilde{\mathbf{u}} = \frac{\mathbf{u}}{U} = \frac{\mathbf{u}\tau}{L}, \tilde{\nu} = \frac{\nu}{\nu'}, \text{etc.} \quad (2.8)$$

Here,  $L$ ,  $\tau$ ,  $U$ , and  $\nu'$ , are the characteristic length, time, velocity, and viscosity scales of the system. By inserting these non-dimensional variables in the NSE, they can be rewritten as:

$$\frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{t}} + (\tilde{\mathbf{u}} \cdot \tilde{\nabla})\tilde{\mathbf{u}} = \frac{1}{\tilde{\rho}}\tilde{\nabla}\tilde{p} + \frac{1}{Re}\tilde{\nu}\tilde{\nabla}^2\tilde{\mathbf{u}} + \tilde{\mathbf{f}}, \quad (2.9)$$

In this dimensionless formulation of the NSE, the Reynolds number is introduced:

$$Re = \frac{UL}{\nu'}. \quad (2.10)$$

The size of this dimensionless number can be used to study the behaviour of the flow. If the Reynolds number is high, the non-linear convective term (i.e. the second term on the left hand side) dominates over the linear diffusive term (i.e the second term on the right hand side). This will result in chaotic and irregular flow patterns, known as turbulent flow. On the other hand, for low Reynolds numbers, the linear diffusive term dominates the equation, which will result in a smooth and stable flow, known as laminar flow.

This example demonstrates the utility of dimensionless numbers. They can be calculated relatively easily, yet they provide a lot of insights regarding the behaviour of the fluid flow. In addition to the Reynolds numbers for discerning laminar and turbulent flow, several other dimensionless numbers exist. The most important numbers that are used throughout this research are outlined in Table 2.1.

**Table 2.1:** Dimensionless numbers used in this research, along with their formulas and physical interpretations.

| Dimensionless number | Definition                | Physical Interpretation                                                                                                                      |
|----------------------|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Reynolds             | $Re = \frac{UL}{\nu}$     | Ratio between the inertial and viscous forces acting on the fluid. Determines if the flow is laminar or turbulent.                           |
| Prandtl              | $Pr = \frac{\nu}{\alpha}$ | Ratio between the viscosity and the thermal diffusivity of the fluid. Compares the rate of momentum diffusion to the rate of heat diffusion. |
| Schmidt              | $Sc = \frac{\nu}{D}$      | Ratio between the viscosity and the mass diffusivity of the fluid. Compares the rate of momentum diffusion to the rate of mass diffusion.    |

### 2.1.4. Kinetic Theory

Using the continuum approximation, fluids can be studied on a macroscopic scale, with quantities such as density and velocity. Fluids can also be studied on a microscopic scale by simulating the interactions between individual molecules or atoms. This approach is therefore known as molecular dynamics. However these simulations become very computationally intensive for studying the behaviour of large-scale systems, such as nuclear reactors. Another approach treats fluids on the mesoscopic scale, known as kinetic theory, which tracks the *distribution* of particles in a gas or fluid. These particle distributions are given by  $f(\mathbf{r}, \boldsymbol{\xi}, t)$ , denoting the density of particles at location  $\mathbf{r} = (x, y, z)$  moving with velocity  $\boldsymbol{\xi} = (\xi_x, \xi_y, \xi_z)$  at time  $t$ . These particle distributions can be related back to macroscopic quantities through special integrals known as moments[18]:

$$\begin{aligned}\rho(\mathbf{r}, t) &= \int f(\mathbf{r}, \boldsymbol{\xi}, t) d^3\xi, \\ \rho(\mathbf{r}, t) \mathbf{u}(\mathbf{r}, t) &= \int \boldsymbol{\xi} f(\mathbf{r}, \boldsymbol{\xi}, t) d^3\xi, \\ \rho(\mathbf{r}, t) E(\mathbf{r}, t) &= \frac{1}{2} \int |\boldsymbol{\xi}|^2 f(\mathbf{r}, \boldsymbol{\xi}, t) d^3\xi.\end{aligned}\tag{2.11}$$

To study the evolution of these distributions  $f$ , the total derivative with respect to time is obtained using the chain rule:

$$\frac{df}{dt} = \left(\frac{\partial f}{\partial t}\right) \frac{dt}{dt} + \left(\frac{\partial f}{\partial \mathbf{r}}\right) \frac{d\mathbf{r}}{dt} + \left(\frac{\partial f}{\partial \boldsymbol{\xi}}\right) \frac{d\boldsymbol{\xi}}{dt}.\tag{2.12}$$

By substituting  $\frac{df}{dt} = \Omega(f)$ ,  $\frac{dt}{dt} = 1$ ,  $\frac{d\mathbf{r}}{dt} = \boldsymbol{\xi}$ ,  $\frac{d\boldsymbol{\xi}}{dt} = \mathbf{f}$ , the last term being the body force, one arrives at the Boltzmann equation:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f + \nabla_{\boldsymbol{\xi}} f \cdot \mathbf{f} = \Omega(f).\tag{2.13}$$

From left to right, the terms in the Boltzmann equation denote the change of particle distribution  $f$  in time, the advection of  $f$  with particle velocity  $\boldsymbol{\xi}$ , the forces acting on the distribution functions, and on the right hand side the collision operator  $\Omega(f)$ . This operator acts as a source term, and describes how the collisions of particles redistribute their densities locally. This redistribution should conserve mass and momentum at the mesoscopic level. Additionally, kinetic theory dictates that through these collisions the distribution functions  $f$  should converge over time towards an equilibrium distribution  $f^{eq}(\mathbf{r}, \boldsymbol{\xi}, t)$ , after which  $\Omega(f^{eq}) = 0$ . This equilibrium distribution is isotropic in velocity space around the macroscopic velocity  $\mathbf{u}$ .

It can be shown that, through Chapman-Enskog analysis, the macroscopic Navier-Stokes equations can be recovered from the mesoscopic Boltzmann Equation. This analysis includes the decomposition of the particle distributions and collision operator into unperturbed and perturbed terms, and consequently equating terms of similar order between the two expansions. By using the relationships between the kinetic moments of Equation 2.11, the same conservation laws of Section 2.1.1 can be derived[51].

## 2.2. Nuclear Reactor Physics

The operation of a nuclear reactor rests on the controlled exploitation of nuclear fission. Fission is a reaction in which a heavy, unstable nucleus splits into smaller fragments, releasing energy as well as several neutrons. The released energy originates from the difference between the mass of the original nucleus and the total mass of the fission products. This mass difference is converted into energy using Einstein's famous equation  $E = mc^2$ . Fission can occur spontaneously through radioactive decay, but this process is slow. In a nuclear reactor, fission is induced in a nucleus by the absorption of an incident neutron, forming an unstable compound nucleus that subsequently decays by splitting[52]. As the neutron emitted by one fission event can in turn induce another fission event, a self-sustaining chain reaction can be established within the reactor core.

To ensure a safe operation of the reactor at a constant power level, the sustained chain reaction must be balanced. On average, exactly one of the neutrons produced in a given fission event should induce a later fission event, while the remaining neutrons should be either absorbed in non-fission reactions or removed from the core. This balance is expressed through the multiplication factor  $k$ , defined as the ratio of the number of neutrons in one generation to the number in the preceding generation. If  $k = 1$ , the reactor is deemed *critical*, and the population of neutrons is steady in time. If  $k > 1$ , the positive feedback loop of the fission events results in an exponential increase of neutrons, and the reactor is *supercritical*. If  $k < 1$ , the reactor is *subcritical*, and the neutron population will decay exponentially. Small deviations from the critical state are represented by the neutron reactivity  $\rho_n = (k - 1)/k$ , typically expressed in per cent mille, or pcm. Note that in this thesis the subscript  $n$  is added to distinguish between reactivity and fluid density. The safe design and control of a nuclear reactor aims to maintain the multiplication factor at unity and to predict the response of the reactor to deviations from it[52].

Predicting the behaviour of a reactor core requires a stochastic description of the neutron population that accounts for the various ways in which the neutrons interact with nuclei, including scattering, fission, and absorption. The likelihood that a given interaction  $x$  occurs over a unit path length is expressed by the macroscopic cross section  $\Sigma_x$ , which is the product of the microscopic cross section  $\sigma_x$  and the atomic number density  $N$  of the material. The microscopic cross sections depend strongly on the neutron energy, as the absorption probability is highest when the kinetic energy of the incident neutron matches an excited state of the compound nucleus. These resonance peaks in the absorption spectrum are themselves temperature-dependent, since thermal motion of the target nuclei broadens the peaks and lowers their height, reducing the probability of absorption. This is known as the (nuclear) Doppler effect. In a molten salt reactor, increasing temperature also results in thermal expansion of the fuel salt, reducing the atomic number density and subsequently the absorption probability. These mechanisms couple the neutron population to the temperature field and describe the negative temperature feedback in a reactor core.

Another feature of the neutron production that affects reactor behaviour is the timescale on which neutrons appear. Most neutrons are emitted within  $10^{-14}$  s of the fission event, and are therefore called *prompt* neutrons. A small fraction of neutrons, however, is emitted at a later time by the radioactive decay of certain fission products. These fission products are therefore referred to as delayed neutron precursors, and the neutrons they emit as *delayed* neutrons. The precursors are characterised by their decay constant  $\lambda$  and the fraction of delayed neutrons  $\beta$  they emit. Although the delayed neutrons only amount to a small fraction of the total neutron population, they lengthen the effective response time of the reactor by several orders of magnitude, making active control on practical timescales possible[52].

### 2.2.1. Neutron Transport

To determine the distributions of neutrons in a nuclear reactor, one must account for the motion of neutrons about the reactor core and their interactions with nuclei in the core. This is also known as neutron transport, and the governing balance equation is accordingly called the neutron transport equation (or NTE), which is given by[52]:

$$\frac{1}{v} \frac{\partial \varphi}{\partial t} + \hat{\Omega} \cdot \nabla \varphi + \Sigma_t(\mathbf{r}, E) \varphi(\mathbf{r}, E, \hat{\Omega}, t) = \int_{4\pi} \int_0^\infty \Sigma_s(E' \rightarrow E, \hat{\Omega}' \rightarrow \hat{\Omega}) \varphi(\mathbf{r}, E', \hat{\Omega}', t) dE' d\hat{\Omega}' + s_f(\mathbf{r}, E, \hat{\Omega}, t), \quad (2.14)$$

where  $\varphi$  represents the *angular* neutron flux, a measure of the number of neutrons passing through a unit surface per unit time. From left to right, the terms in the NTE denote the rate of change of the number of neutrons, the leakage of neutrons into or out of the volume element, the total removal of neutrons due to absorption or collisions, and on the right-hand side the gain due to neutrons scattering into  $dE$  about  $E$ ,  $d\hat{\Omega}$  about  $\hat{\Omega}$ , from other energies  $E'$  and directions  $\hat{\Omega}'$ , and the addition of neutrons due to fission. This source term of (prompt) neutrons due to fission can be expressed as:

$$s_f(\mathbf{r}, E, \hat{\Omega}, t) = \frac{\chi(E)}{4\pi} \int_{4\pi} \int_0^\infty \nu(E') \Sigma_f(\mathbf{r}, E') \varphi(\mathbf{r}, E', \hat{\Omega}', t) dE' d\hat{\Omega}', \quad (2.15)$$

where the fission spectra  $\chi$ , neutron multiplicity  $\nu$ , and (macroscopic) fission cross-section  $\Sigma_f$  are functions of the neutron energy, but not of the direction  $\hat{\Omega}$  of its propagation. This means that isotropic emission of the neutrons produced from fission is assumed, indicated by the  $1/4\pi$  prefactor.

In theory, the NTE must be solved for an infinite number of possible energies  $E$  traveling in an infinite number of possible directions  $\hat{\Omega}$ . For most situations, this is impossible to solve analytically, and even numerically it is a difficult and time-consuming endeavour due to the NTE's seven degrees of freedom. In order to simplify calculations, neutrons with roughly the same energy can be grouped into discrete energy bands. Neutrons in the same energy group  $E_g$  are then characterised by shared diffusion coefficients  $D_{n,g}$  and cross sections  $\Sigma_{x,g}$ .

A similar discretisation can be performed on the direction of propagation  $\hat{\Omega}$  by representing the angular variable by a discrete set of directions or rays  $\hat{\Omega}_n$ . This discrete ordinate treatment of angle turns the integral over  $\hat{\Omega}$  into a summation, and consequently the NTE into a set of equations known as the  $S_N$  equations, where  $N$  denotes the number of discretised angular directions. One can also use spherical harmonics to perform functional expansion on the angular directions. This approach is described in [52] and corresponds to an expansion in Legendre polynomials. The set of equations that replaces Equation 2.14 is known as the  $P_N$  equations.

For most reactor calculations, the angular flux  $\varphi$  has weak angular dependence, and so often it can be replaced with the *scalar* neutron flux  $\phi(\mathbf{r}, E, t) = \int_{4\pi} \varphi(\mathbf{r}, E, \hat{\Omega}, t) d\hat{\Omega}$  (which is occasionally shortened to just 'neutron flux' throughout the remainder of this thesis). By integrating the NTE over angle, and replacing the angle-dependent scattering cross section  $\Sigma_s$  and source term  $s_f$  with their integrated (and therefore angle-independent) counterparts, the neutron continuity equation is obtained:

$$\frac{1}{v} \frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{J}(\mathbf{r}, E, t) + \Sigma_t(\mathbf{r}, E) \phi(\mathbf{r}, E, t) = \int_0^\infty \Sigma_s(E' \rightarrow E) \phi(\mathbf{r}, E', t) dE' + S_f(\mathbf{r}, E, t). \quad (2.16)$$

Here the neutron current density is introduced:  $\mathbf{J}(\mathbf{r}, E, t) = \int_{4\pi} \hat{\Omega} \varphi(\mathbf{r}, E, \hat{\Omega}, t) d\hat{\Omega}$ .

### 2.2.2. Neutron Diffusion

The above equations can be simplified further with a few more assumptions or approximations. First, the  $P_1$  equations as angle discretisation are used. Next, it is assumed that the neutron current density  $\mathbf{J}$  changes much slower than the frequency of neutron collision ( $\approx 10^5$  Hz). This assumption leads to the following simplification[52]:

$$\mathbf{J}(\mathbf{r}, E, t) = -D(\mathbf{r}, E) \nabla \phi(\mathbf{r}, E, t). \quad (2.17)$$

This relation is more commonly known as Fick's law, and also arises in other areas of physics to describe diffusion of particles. In this expression the neutron diffusion coefficient  $D(\mathbf{r}, E) = \frac{1}{3 [\Sigma_t(\mathbf{r}, E) - \bar{\mu}_0 \Sigma_s(\mathbf{r}, E)]}$  is introduced, where  $\bar{\mu}_0$  is the average scattering cosine. Lastly, assuming that the prompt fission sources are isotropic, i.e.  $s_f(\mathbf{r}, \hat{\Omega}, E) = \frac{1}{4\pi} S_f(\mathbf{r}, E)$ , and considering discretised energy groups, the multi-group neutron diffusion equation (or NDE) for energy group  $g$  is obtained:

$$\frac{1}{v_g} \frac{\partial \phi_g}{\partial t} - \nabla \cdot (D_g \nabla \phi_g) = -\Sigma_{t,g} \phi_g + \chi_g \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \sum_{g'} \Sigma_{s,g' \rightarrow g} \phi_{g'}. \quad (2.18)$$

From left to right, the terms in the NDE denote the rate of change of scalar neutron flux, the diffusion of neutrons, and on the right hand side the total removal of neutrons due to absorption or collisions, the production of prompt neutrons due to fission, and the in-scattering of neutrons from energy group  $g'$  into group  $g$ .

As the neutron diffusion equation has derivatives in both space and time, it is crucial that suitable boundary and initial boundary conditions are applied to complete the description of any problem. A commonly used boundary condition for neutron transport is the vacuum boundary condition, which sets the incoming angular neutron flux (i.e.  $\varphi(\mathbf{r}, E, \hat{\Omega}, t) = 0$  for  $\hat{\Omega} \cdot \hat{\mathbf{e}}_s < 0$ ). However in diffusion theory, no angular information is available. As a workaround one can impose that the scalar flux, when linearly extrapolated beyond the vacuum boundary, becomes zero at some distance  $\tilde{x}_{boundary,g}$ . This extrapolated boundary length at which the scalar neutron flux is zero is proportional to the neutron diffusion coefficient[53]:

$$\tilde{x}_{boundary,g} = x_{boundary,g} + 2D_g, \quad (2.19)$$

although more advanced transport theory[52] sets the coefficient prefactor to 2.1312.

The above equations only account for the production and loss of prompt neutrons. However as stated earlier, some neutrons are not immediately generated through fission, but emitted at a later time as decay products from highly excited precursors. Analogously to the neutron energy groups, different sorts of precursors can be bundled together into families (denoted by index  $d$ ) that share an averaged decay constant  $\lambda_d$  and delayed neutron fraction  $\beta_d$ . The concentration of a precursor family in space is a scalar quantity, and in a circulating molten salt the fragments also experience convective effects of the fuel. Therefore, similarly to the heat equation, the advection-diffusion equation for the concentration  $C$  of precursor family  $d$  can be formulated as:

$$\frac{\partial C_d}{\partial t} + \nabla \cdot (\mathbf{u} C_d) - \nabla \cdot (D_p \nabla C_d) = -\lambda_d C_d + \beta_d \sum_g (\nu \Sigma_f)_g \phi_g. \quad (2.20)$$

From left to right, the terms in the precursor advection-diffusion equation denote the rate of change of precursor concentration, the convective transport, the diffusion of precursors, and on the right hand side the natural decay of precursors, and the production of precursors due to fission. As the decay of precursors also generates additional neutrons in the reactor core, this delayed neutron source term must be added to Equation 2.18. The complete neutron diffusion equation with prompt and delayed neutron sources is formulated as:

$$\frac{1}{v_g} \frac{\partial \phi_g}{\partial t} - \nabla \cdot (D_g \nabla \phi_g) = -\Sigma_{t,g} \phi_g + \chi_g^p (1 - \beta_{tot}) \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \chi_g^d \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g' \rightarrow g} \phi_{g'}. \quad (2.21)$$

### Steady-State Neutronics for Reactor Criticality

Under normal operation a nuclear reactor is supposed to be in critical condition. This means that the number of neutrons stays constant over time, so the time-dependent Equations 2.20 and 2.21 can be respectively reduced to:

$$\nabla \cdot (\mathbf{u} C_d) - \nabla \cdot (D_p \nabla C_d) = -\lambda_d C_d + \beta_d \sum_g (\nu \Sigma_f)_g \phi_g, \quad (2.22)$$

$$-\nabla \cdot (D_g \nabla \phi_g) = -\Sigma_{t,g} \phi_g + \chi_g^p (1 - \beta_{tot}) \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \chi_g^d \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g' \rightarrow g} \phi_{g'}. \quad (2.23)$$

This system of coupled equations only has a solution when the neutronics parameters and boundary conditions represent the composition and geometry of a critical reactor. However in many cases the real question is whether the chosen parameters and geometry will yield criticality. Therefore the problem can be studied as an eigenvalue problem:

$$\nabla \cdot (\mathbf{u}C_d) - \nabla \cdot (D_p \nabla C_d) = -\lambda_d C_d + \frac{\beta_d}{k_{eff}} \sum_g (\nu \Sigma_f)_g \phi_g, \quad (2.24)$$

$$-\nabla \cdot (D_g \nabla \phi_g) = -\Sigma_{t,g} \phi_g + \frac{\chi_g^p (1 - \beta_{tot})}{k_{eff}} \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \chi_g^d \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g' \rightarrow g} \phi_{g'}, \quad (2.25)$$

where the multiplication factor  $k_{eff}$  is added to the fission terms and acts as the eigenvalue of the problem. The neutron fluxes  $\phi_g$  and precursor densities  $C_d$  are the corresponding eigenvectors of this system of equations. Using this approach, one can assess whether the chosen composition and size will yield a subcritical ( $k_{eff} < 1$ ), critical ( $k_{eff} = 1$ ), or supercritical ( $k_{eff} > 1$ ) reactor.

To solve for this eigenvalue, Equations 2.24 and 2.25 can be grouped and rewritten into matrix form:

$$\mathbf{M}\gamma = \frac{1}{k_{eff}} \mathbf{F}\gamma, \quad (2.26)$$

where matrices  $\mathbf{M}$  and  $\mathbf{F}$  denote the destruction and the fission operator respectively, and  $\gamma$  denotes the eigenvector. For a reactor model with 2 neutron groups and 2 precursor families, the matrices and eigenvector are formulated as:

$$\mathbf{M} = \begin{bmatrix} \nabla \cdot \mathbf{u} - \nabla \cdot (D_p \nabla) + \lambda_1 & 0 & 0 & 0 \\ 0 & \nabla \cdot \mathbf{u} - \nabla \cdot (D_p \nabla) + \lambda_2 & 0 & 0 \\ -\chi_1^d \lambda_1 & -\chi_1^d \lambda_2 & -\nabla \cdot (D_1 \nabla) + \Sigma_{t,1} - \Sigma_{s,1 \rightarrow 1} & 0 \\ -\chi_2^d \lambda_1 & -\chi_2^d \lambda_2 & -\Sigma_{s,1 \rightarrow 2} & -\nabla \cdot (D_2 \nabla) + \Sigma_{t,2} - \Sigma_{s,2 \rightarrow 2} \end{bmatrix}, \quad (2.27)$$

$$\gamma = \begin{bmatrix} C_1 \\ C_2 \\ \phi_1 \\ \phi_2 \end{bmatrix}, \quad (2.28)$$

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & \beta_1 \nu_1 \Sigma_{f,1} & \beta_1 \nu_2 \Sigma_{f,2} \\ 0 & 0 & \beta_2 \nu_1 \Sigma_{f,1} & \beta_2 \nu_2 \Sigma_{f,2} \\ 0 & 0 & \chi_1^p (1 - \beta_{tot}) \nu_1 \Sigma_{f,1} & \chi_1^p (1 - \beta_{tot}) \nu_2 \Sigma_{f,2} \\ 0 & 0 & \chi_2^p (1 - \beta_{tot}) \nu_1 \Sigma_{f,1} & \chi_2^p (1 - \beta_{tot}) \nu_2 \Sigma_{f,2} \end{bmatrix}. \quad (2.29)$$

These matrices can be rewritten in block matrix form:

$$\begin{bmatrix} A_{CC}(\mathbf{u}) & 0 \\ A_{\phi C} & A_{\phi\phi} \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \phi \end{bmatrix} = \frac{1}{k_{eff}} \begin{bmatrix} 0 & A_{C\phi}^F \\ 0 & A_{\phi\phi}^F \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \phi \end{bmatrix}, \quad (2.30)$$

with  $A_{CC}$  an operator contributing and acting only on the precursor concentrations,  $A_{\phi\phi}$  contributing and acting only on the scalar neutron fluxes, and  $A_{\phi C}$  acting on the precursor concentrations but contributing to the neutron flux. For the fission block,  $A_{C\phi}^F$  is a fission operator acting on the neutron fluxes but contributing to the precursor concentrations, and  $A_{\phi\phi}^F$  an operator contributing to and acting on neutron flux. It can now be seen that  $\mathbf{M}$  is a (lower) triangular matrix, with distinct diagonal entries. Therefore it is diagonalisable, and thus one can apply the power iteration method to find the largest eigenvalue and accompanying criticality eigenvector.

Using the power method, initial guesses for the multiplication factor  $k_{eff}^0$  and fission source vector  $\mathbf{S}^0$  are randomly chosen at  $l = 0$  and used to solve the in-homogeneous matrix equation and calculate the next value for  $\gamma^{(l+1)}$ :

$$\mathbf{M}\gamma^{(l+1)} = \frac{1}{k_{eff}^{(l)}} \mathbf{S}^{(l)} \quad (2.31)$$

This new value for the eigenvector is then used to calculate the new fission source vector and eigenvalue with:

$$\mathbf{S}^{(l+1)} = \mathbf{F}\gamma^{(l+1)}, \quad (2.32)$$

$$k_{eff}^{(l+1)} = k_{eff}^{(l)} \frac{\int_V \mathbf{S}^{(l+1)} dV}{\int_V \mathbf{S}^{(l)} dV}. \quad (2.33)$$

This scheme is repeated until convergence, i.e. when  $(k_{eff}^{(l+1)} - k_{eff}^{(l)})$  falls below a certain imposed threshold. Note that in this thesis, similarly to the approach of [42], the traditional power method is split up in an "outer" iteration that calculates the fission vector and multiplication factor, and an "inner" iteration that calculates the scalar neutron flux using the Filter-Matrix Lattice Boltzmann method. More on this in Chapter 3.

### 2.2.3. Reactor Dynamics

#### The Point Reactor Kinetic Equations

To study the transient behaviour of a molten salt reactor, the time-dependent diffusion equations (Eqs. 2.20 & 2.21) have to be solved. Time-dependent situations can arise, for example, due to the insertion or ejection of control rods, fuel depletion or isotopic build-up. For these cases it is a common approach[52, 53, 54] to factorise the scalar neutron flux  $\phi$  into a slowly changing *shape* function  $\tilde{\phi}(\mathbf{r}, t)$  and a fast changing *amplitude* function  $n(t)$ :

$$\phi_g(\mathbf{r}, t) = \tilde{\phi}(\mathbf{r}, t)n(t), \quad (2.34)$$

Using this factorisation, the neutron flux space-time distribution can be simulated using different timescales, including a big time step for predicting the neutron flux shape and a small time step for the amplitude function. This is known as the quasi-static method, as it is assumed that the shape function is weakly time-dependent and therefore doesn't change for small timescales. For clarity the operators that act on the neutron fluxes will be shortened to:

$$M_g \phi_g = -\nabla \cdot (D_g \nabla \phi_g) + \Sigma_{t,g} \phi_g - \sum_{g'} \Sigma_{s,g' \rightarrow g} \phi_{g'}, \quad (2.35)$$

$$F_g^P \phi_g = \chi_g^p (1 - \beta_{tot}) \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'}, \quad (2.36)$$

$$F_{d,g}^D \phi_g = \chi_g^d \beta_d \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'}, \quad (2.37)$$

$$F_g^D \phi_g = \sum_d F_{d,g}^D \phi_g, \quad (2.38)$$

$$F_g \phi_g = (F_g^P + F_g^D) \phi_g, \quad (2.39)$$

$$S_g^D = \chi_g^d \sum_d \lambda_d C_d. \quad (2.40)$$

Using these operator expressions and the flux factorisation from Equation 2.34, and multiplying with a weighting function  $\Phi_g^\dagger$ , the time-dependent neutron diffusion equation is stated as:

$$\Phi_g^\dagger \frac{1}{v_g} \tilde{\phi}_g \frac{\partial n}{\partial t} + \Phi_g^\dagger \frac{1}{v_g} n \frac{\partial \tilde{\phi}_g}{\partial t} = \Phi_g^\dagger (F_g^P - M_g) \tilde{\phi}_g n + \Phi_g^\dagger S_g^D. \quad (2.41)$$

The weighting function  $\Phi_g^\dagger$  is known as the *conjugate* neutron flux or *adjoint* neutron flux. The equations that give its solution will be discussed later in this chapter. For now it suffices to know that this weighting function is used to ensure the uniqueness of the factorisation in Equation 2.34. More concretely, the neutron shape function is normalised by imposing:

$$\sum_g \left\langle \Phi_g^\dagger, \frac{1}{v_g} \tilde{\phi}_g \right\rangle = \sum_g \frac{1}{v_g} \int_V \Phi_g^\dagger(\mathbf{r}) \tilde{\phi}_g(\mathbf{r}, t) dV = 1 = \sum_g \left\langle \Phi_g^\dagger, \frac{1}{v_g} \phi_{g,0} \right\rangle, \quad (2.42)$$

where the solution to the steady-state problem  $\phi_{g,0}$  is used initially to normalise the adjoint neutron flux. Here the bracket notation  $\langle x, y \rangle$  denotes a volume integration over the entire domain of  $x$  and  $y$ . Applying a similar integration to Equation 2.41 yields:

$$\left\langle \Phi_g^\dagger, \frac{1}{v_g} \tilde{\phi}_g \right\rangle \frac{\partial n}{\partial t} = \left\langle \Phi_g^\dagger, (F_g^P - M_g) \tilde{\phi}_g \right\rangle n + \left\langle \Phi_g^\dagger, S_g^D \right\rangle, \quad (2.43)$$

where the second term on the left-hand side of the original equation is omitted, as it was assumed that the neutron shape function was constant over small timescales. By following the same approach for the time-dependent precursor advection-diffusion equation (Equation 2.20), yields

$$\frac{\partial \langle \Phi_g^\dagger, \chi_g^d C_d \rangle}{\partial t} = \left\langle \Phi_g^\dagger, F_{d,g}^D \tilde{\phi}_g \right\rangle - \lambda_d \langle \Phi_g^\dagger, \chi_g^d C_d \rangle. \quad (2.44)$$

Note the added factor of  $\chi_g^d$  to all terms. In the above equation the convective and diffusive term are omitted. This simplification is justified because the timescale of precursor convection and diffusion is much larger than the timescale for precursor concentration change[55]. If one defines the precursor amplitude  $m_d$  as:

$$m_d = \frac{\sum_g \langle \Phi_g^\dagger, \chi_g^d C_d \rangle}{\sum_g \left\langle \Phi_g^\dagger, \frac{1}{v_g} \tilde{\phi}_g \right\rangle}, \quad (2.45)$$

and divides Equations 2.43 and 2.44 by  $\left\langle \Phi_g^\dagger, \frac{1}{v_g} \tilde{\phi}_g \right\rangle$  and summing over all neutron energy groups, this yields:

$$\frac{\partial n}{\partial t} = \frac{\left\langle \Phi_g^\dagger, (F_g^P - M_g) \tilde{\phi}_g \right\rangle}{\left\langle \Phi_g^\dagger, \frac{1}{v_g} \tilde{\phi}_g \right\rangle} n + \sum_d \lambda_d m_d, \quad (2.46)$$

$$\frac{\partial m_d}{\partial t} = \frac{\left\langle \Phi_g^\dagger, F_{d,g}^D \tilde{\phi}_g \right\rangle}{\left\langle \Phi_g^\dagger, \frac{1}{v_g} \tilde{\phi}_g \right\rangle} n - \lambda_d m_d. \quad (2.47)$$

Under the assumption that the neutron shape is weakly time-dependent, the bracket fractions above can be assumed constant in time. Therefore the equations are reduced to ordinary differential equations in time for the amplitudes  $n(t)$  and  $m_d(t)$ . In other words, for the smallest timescale the shape of the neutron flux distribution is ignored, and the reactor is reduced to a point. This is why these types

of equations are also known as point kinetics (PK) equations[41, 52, 53, 56]. The equations can be rewritten as:

$$\frac{dn(t)}{dt} = \frac{\rho_n(t) - \beta_{tot}(t)}{\Lambda(t)} n(t) + \sum_d \lambda_d m_d(t), \quad (2.48)$$

$$\frac{dm_d(t)}{dt} = \frac{\beta_d(t)}{\Lambda(t)} n(t) - \lambda_d m_d(t), \quad (2.49)$$

where the PK parameters are defined as[39]:

$$\rho_n(t) = \frac{\sum_g \langle \Phi_g^\dagger, (F_g - M_g) \tilde{\phi}_g \rangle}{\sum_g \langle \Phi_g^\dagger, F_g \tilde{\phi}_g \rangle}, \quad (2.50)$$

$$\beta_d(t) = \frac{\sum_g \langle \Phi_g^\dagger, F_{d,g}^D \tilde{\phi}_g \rangle}{\sum_g \langle \Phi_g^\dagger, F_g \tilde{\phi}_g \rangle}, \quad (2.51)$$

$$\beta_{tot}(t) = \frac{\sum_g \langle \Phi_g^\dagger, F_g^D \tilde{\phi}_g \rangle}{\sum_g \langle \Phi_g^\dagger, F_g \tilde{\phi}_g \rangle} = \sum_d \frac{\sum_g \langle \Phi_g^\dagger, F_{d,g}^D \tilde{\phi}_g \rangle}{\sum_g \langle \Phi_g^\dagger, F_g \tilde{\phi}_g \rangle} = \sum_d \beta_d(t), \quad (2.52)$$

$$\Lambda(t) = \frac{\sum_g \langle \Phi_g^\dagger, \frac{1}{v_g} \tilde{\phi}_g \rangle}{\sum_g \langle \Phi_g^\dagger, F_g \tilde{\phi}_g \rangle}. \quad (2.53)$$

Note that, for normalised adjoint fluxes  $\Phi_g^\dagger$  and neutron shapes  $\tilde{\phi}_g$ , the numerator in the last equation simplifies to unity according to Equation 2.42.

#### 2.2.4. Adjoint Neutron Diffusion

The remainder of this section establishes the role of the adjoint neutron flux as the proper weight function in the point-kinetics formulation, and describes how it is computed. The argument starts from the point-kinetics reactivity  $\rho_n(t)$  given in Equation 2.50, rewritten with the weight function  $W$  left unspecified:

$$\rho_W(t) = \frac{\sum_g \langle W, (F_g - M_g) \tilde{\phi}_g \rangle}{\sum_g \langle W, F_g \tilde{\phi}_g \rangle}, \quad (2.54)$$

Since the shape function  $\tilde{\phi}$  is predicted in between time steps, this naturally leads to some error. The question then arises how much this error affects  $\rho_W(t)$ . The predicted shape can be rewritten as:

$$\tilde{\phi} = \tilde{\phi}_{exact} + \delta\tilde{\phi}, \quad (2.55)$$

where  $\delta\tilde{\phi}$  is the error introduced by holding the shape function constant across a big time step. Similarly, the time-dependent operator can be expressed as (ignoring the subscript  $g$  for clarity):

$$F(t) - M(t) = (F_0 - M_0) + (\delta F + \delta M) = (F_0 - M_0) + \delta A(t), \quad (2.56)$$

where  $\delta A(t)$  captures the operator's drift from its initial critical configuration  $(F_0 - M_0)$ , for example due to shifting cross sections. Expanding the numerator of Equation 2.54 yields:

$$\begin{aligned}\langle W, (F - M)\tilde{\phi} \rangle &= \langle W, (F_0 - M_0)\tilde{\phi}_{exact} \rangle + \langle W, (F_0 - M_0)\delta\tilde{\phi} \rangle + \langle W, \delta A\tilde{\phi}_{exact} \rangle + \langle W, \delta A\delta\tilde{\phi} \rangle \\ &= \langle W, (F - M)\tilde{\phi}_{exact} \rangle + \langle W, (F_0 - M_0)\delta\tilde{\phi} \rangle + \langle W, \delta A\delta\tilde{\phi} \rangle.\end{aligned}\quad (2.57)$$

Here the terms on the right hand side denote the (errorless) value for the numerator, a first-order error dependent on the shape of  $\delta\tilde{\phi}$  (so  $\propto \mathcal{O}(\epsilon_\phi)$ ), and a much smaller second-order term of order  $\mathcal{O}(\epsilon_A\epsilon_\phi)$ . Ideally a weight function  $W$  is chosen such that the first-order error term always yields zero. This can be done by using the inner product and defining the operator  $X^\dagger$  *adjoint* to the operator  $X$  as that operator for which

$$\langle X^\dagger f, g \rangle = \int_V X^\dagger f^*(\mathbf{r})g(\mathbf{r}) \, dV = \int_V f^*(\mathbf{r})Xg(\mathbf{r}) \, dV = \langle f, Xg \rangle \quad (2.58)$$

holds for every (complex) function  $f(\mathbf{r})$  and  $g(\mathbf{r})$  satisfying homogeneous boundary conditions. If the adjoint flux  $\Phi^\dagger$  is defined as the corresponding solution of the steady-state (critical) adjoint neutron diffusion equation:

$$\mathbf{M}_0^\dagger \Phi^\dagger = \mathbf{F}_0^\dagger \Phi^\dagger, \quad (2.59)$$

then by definition,

$$\langle \Phi^\dagger, (F_0 - M_0)\delta\tilde{\phi} \rangle = \langle (F_0^\dagger - M_0^\dagger)\Phi^\dagger, \delta\tilde{\phi} \rangle = \langle 0, \delta\tilde{\phi} \rangle = 0. \quad (2.60)$$

So in other words, choosing the adjoint flux as a weighting function substantially reduces the error that arises from solving the neutron shape function at larger timescales.

Using the results for the 'regular' neutron flux, the adjoint operators can be constructed by taking the Hermitian adjoint (or conjugate transpose) of Equations 2.27 and 2.29:

$$\mathbf{M}^\dagger = \begin{bmatrix} -\mathbf{u} \cdot \nabla - \nabla \cdot (D_p \nabla) + \lambda_1 & 0 & -\chi_1^d \lambda_1 & -\chi_2^d \lambda_1 \\ 0 & -\mathbf{u} \cdot \nabla - \nabla \cdot (D_p \nabla) + \lambda_2 & -\chi_1^d \lambda_2 & -\chi_2^d \lambda_2 \\ 0 & 0 & -\nabla \cdot (D_1 \nabla) + \Sigma_{t,1} - \Sigma_{s,1 \rightarrow 1} & -\Sigma_{s,1 \rightarrow 2} \\ 0 & 0 & 0 & -\nabla \cdot (D_2 \nabla) + \Sigma_{t,2} - \Sigma_{s,2 \rightarrow 2} \end{bmatrix}, \quad (2.61)$$

$$\gamma^\dagger = \begin{bmatrix} C_1^\dagger \\ C_2^\dagger \\ \Phi_1^\dagger \\ \Phi_2^\dagger \end{bmatrix}, \quad (2.62)$$

$$\mathbf{F}^\dagger = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \beta_1 \nu_1 \Sigma_{f,1} & \beta_2 \nu_1 \Sigma_{f,1} & \chi_1^p (1 - \beta_{tot}) \nu_1 \Sigma_{f,1} & \chi_2^p (1 - \beta_{tot}) \nu_1 \Sigma_{f,1} \\ \beta_1 \nu_2 \Sigma_{f,2} & \beta_2 \nu_2 \Sigma_{f,2} & \chi_1^p (1 - \beta_{tot}) \nu_2 \Sigma_{f,2} & \chi_2^p (1 - \beta_{tot}) \nu_2 \Sigma_{f,2} \end{bmatrix}. \quad (2.63)$$

There are three things to note about this new structure compared to the forward matrices. The first one is the sign change in the convection operator for  $C_d^\dagger$ . If one considers the 'forward' operator  $\nabla \cdot (\mathbf{u}C) = \mathbf{u} \cdot \nabla C + C(\nabla \cdot \mathbf{u})$  acting on the regular precursor field in the context of the inner product stated in Equation 2.58 and using integration by parts and Gauss' divergence theorem, this gives:

$$\begin{aligned}
\langle f, \nabla \cdot (\mathbf{u}g) \rangle &= \int_V f^* \nabla \cdot (\mathbf{u}g) dV = \int_V \nabla \cdot (f^* g \mathbf{u}) dV - \int_V g \mathbf{u} \cdot \nabla f^* dV \\
&= \oint_S f^* g (\mathbf{u} \cdot \mathbf{n}) dS - \int_V g \mathbf{u} \cdot \nabla f^* dV.
\end{aligned} \tag{2.64}$$

Under the assumption that the complex functions  $f$  and  $g$  satisfy homogeneous boundary conditions, the surface integral on the right hand side vanishes, which yields:

$$\langle f, \nabla \cdot (\mathbf{u}g) \rangle = \int_V f^* \nabla \cdot (\mathbf{u}g) dV = - \int_V g \mathbf{u} \cdot \nabla f^* dV = \langle -\mathbf{u} \cdot \nabla f, g \rangle. \tag{2.65}$$

So in other words, the adjoint of the convection operator ' $\nabla \cdot (\mathbf{u} \cdot)$ ' is ' $-\mathbf{u} \cdot (\nabla \cdot)$ '. Note that for incompressible flow,  $\nabla \cdot \mathbf{u} = 0$ , and so the forward operator simplifies to  $\mathbf{u} \cdot (\nabla \cdot)$ . In that case the adjoint is just a sign flip. For the diffusion operator ( $-\nabla \cdot (D_p \nabla \cdot)$ ) it can be found that the adjoint is the same operator (i.e. it's self-adjoint), so it doesn't warrant a sign flip in the adjoint destruction matrix  $\mathbf{M}^\dagger$ .

The other two important things to note about the adjoint fission and destruction matrices are the transposition of the off-diagonal entries, namely the fission coupling and the scattering cross sections. To better see this effect, the adjoint matrix equation is reformulated into a system of differential equations for an arbitrary number of (adjoint) neutron flux groups  $\Phi_g^\dagger$  and (adjoint) precursor concentration families  $C_d^\dagger$ .

$$-\mathbf{u} \cdot \nabla C_d^\dagger - \nabla \cdot (D_p \nabla C_d^\dagger) = -\lambda_d C_d^\dagger + \lambda_d \sum_g \chi_g^d \Phi_g^\dagger, \tag{2.66}$$

$$-\nabla \cdot (D_g \nabla \Phi_g^\dagger) = -\Sigma_{t,g} \Phi_g^\dagger + \frac{(\nu \Sigma_f)_g}{k_{eff}^\dagger} \left[ (1 - \beta_{tot}) \sum_{g'} \chi_{g'}^p \Phi_{g'}^\dagger + \sum_d \beta_d C_d^\dagger \right] + \sum_{g'} \Sigma_{s,g \rightarrow g'} \Phi_{g'}^\dagger. \tag{2.67}$$

Compared to the forward system (Equations 2.24 and 2.25), it appears that in the forward system the fission produces neutrons with spectrum  $\chi_g^p$  and precursors with yield  $\beta_d$ , weighted by  $(\nu \Sigma_f)_{g'}$  summed over source groups. In the adjoint system, the roles swap:  $(\nu \Sigma_f)_g$  sits outside the sum as a property of energy group  $g$ , and the summation is over the spectra  $\chi_{g'}^p$  weighted by adjoint fluxes. Similarly, the  $\beta_d C_d^\dagger$  sum appears in the adjoint neutron flux equation (rather than the  $\chi_g^d \lambda_d C_d$  sum), and the adjoint precursor concentration is sourced by  $\chi_g^d \Phi_g^\dagger$  (rather than  $(\nu \Sigma_f)_{g'} \phi_{g'}$ ).

The neutron scattering also reverses direction due to the transposition. In the forward equation, the groups are summed with  $\Sigma_{s,g' \rightarrow g} \phi_{g'}$ , i.e. neutrons scatter *from* group  $g'$  *into*  $g$ . In the adjoint system, the groups are summed with  $\Sigma_{s,g \rightarrow g'} \phi_{g'}$ , i.e. adjoint flux scatters *out of* group  $g$  into  $g'$ . This scattering reversal gives us a hint of the physical interpretation of the adjoint neutron flux. While  $\phi_g(\mathbf{r}, E, t)$  denotes the scalar flux of neutrons at a certain point in space and time, the adjoint flux  $\Phi_g^\dagger(\mathbf{r}, E, t)$  represents the expected contribution that a single neutron will make to the fission chain. Inserting one neutron of energy  $E$  at position  $\mathbf{r}$ , gives rise to some chain of scattering, absorptions, and fissions. The adjoint flux is a measure of how many descendants that initial neutron will eventually produce. Conversely, it is also a measure of how effective a neutron absorber inserted at that location is in changing the reactivity of the core. The transposed scattering source no longer represents what scatters *into* group  $g$ , but where a neutron *out of* group  $g$  scatters to. For the contribution of that neutron, it does not matter where it came from, but rather where it can go and what it can do in the future. This is why the adjoint neutron flux is also often named the neutron *importance* [57, 58, 59]. Following the same train of thought, a neutron's importance for the fission chain is increased by its ability to cause fission ( $\propto (\nu \Sigma_f)_g$ ), multiplied by the importance of the prompt neutrons ( $\propto \sum_{g'} \chi_{g'}^p \Phi_{g'}^\dagger$ ) and the importance of the delayed neutron precursors ( $\propto \sum_d \beta_d C_d^\dagger$ ) it will produce.

Similarly, the importance of a delayed precursor (or adjoint precursor concentration) is measured by importance of the delayed neutrons ( $\propto \sum_g \chi_g^d \Phi_g^\dagger$ ) it will produce. Furthermore, the forward equation

for precursors describes how the flow carries them downstream, away from where they are produced. The adjoint (or importance) equation solves the reverse question: a precursor's importance is not large because of its current position, but where the flow will have taken it by the time it decays. If the precursor is eventually carried by the flow to a region of high (neutron) importance (e.g. near the core centre) where it will decay and produce a neutron, then it will have a higher importance than a precursor which is generated near the core but decays elsewhere due to flow convection. Therefore when computing importance, it must propagate *upstream*, i.e. against the flow. This is encoded in the adjoint equation by the negative convection.

Two more things to note. First, the adjoint counterpart  $k_{eff}^\dagger$  is added to the adjoint fission term in Equation 2.67 to act as an eigenvalue again. Similarly to the forward case, one can study this problem as an eigenvalue problem through a power iteration method. Note that, in contrast to the forward case, there is no  $k_{eff}^\dagger$  in the adjoint precursor equation as there is also no direct fission term there; the effect that the adjoint precursor concentration has on the eigenvalue problem is represented by the (new)  $\sum_d \beta_d C_d^\dagger$  term in the adjoint neutron flux diffusion equation. Second, it can be shown that for a one-group (or one-speed) neutron diffusion model, the forward and the adjoint neutron problem are the same. It is trivial to see that if there is only one energy group  $g$ , then the summation of prompt fission sources/ products reduce to just one term, and there is also no cross-group neutron scattering. This means that, for a reactor model with one neutron energy group (and static fuel),

$$\phi_{0,g} = \Phi_{0,g}^\dagger. \quad (2.68)$$

In this simplified case, one can just use the solution to the steady-state (forward) equation (Eq. 2.25) as a weighting function for the point-kinetic parameters, as is done in earlier research by [42, 43]. However, if the precursors are circulating in the fuel or if there are non-zero off-diagonal entries due to multiple neutron energy groups, then the equivalence in Equation 2.68 does not hold, and one can not use this short-cut. In that case, the solutions to the steady-state forward system (Eqs. 2.24 & 2.25) and the solutions to the steady-state adjoint system (Eqs. 2.66 & 2.67) must be computed *separately*, before one can use these solutions as the initial conditions and weighting of the transient problem (Eqs. 2.48 & 2.49) respectively.

## 2.3. Multiphysics coupling

The above sections give a basic explanation of the physical fields that govern the thermal hydraulics of the core of a molten salt reactor. As explained earlier, although the fields represent different physical domains, they are not entirely independent. They act on each other through source & sink terms and convection. Therefore, to simulate the complete behaviour of a reactor, this 'coupling' of the fields must be included. Before the method of implementation is discussed, it can be worthwhile to briefly reiterate the coupling and its physical interpretation, so that the modelling can be set up correctly.

### 2.3.1. Circulation of heat and precursors

Temperature is a property of matter, and so if the molten salt moves through the reactor, then the movement of the fluid also brings about the transfer of heat through space. This is known as convection, and is the main driver behind heat transfer in liquids. Conversely, changes in the fluid's density through thermal expansion can induce buoyant effects, which affect the fluid's velocity field.

The principle of convective heat transfer also holds for any other scalar field in a fluid. In the case of molten salt, the flow of concentrations of delayed neutron precursors is also influenced by the velocity field. This convective influence of precursor transport is quantified through the mass diffusivity coefficient  $D_p$  in the precursor advection-diffusion equation. In contrast to convective heat transfer, this coupling between fluid velocity and precursor concentration is one-way, i.e. there is no 'buoyant' analogy induced by the precursors which acts on the velocity field directly.

It is important to state that, although the neutrons also move through the molten salt, their mass is so small and their speed so high that they are not directly affected by the 'slow' circulation of the molten salt. Therefore the neutron diffusion equation does not have a 'convection term', unlike the precursor equation.

### 2.3.2. Fission products and precursor decay

As described in Section 2.2, the precursor densities and neutron fluxes also act on each other as their respective source terms. When a nucleus of fissionable material undergoes fission, it splits into two smaller nuclei, along with a few prompt neutrons. The fission products are also known as fission fragments. Most of these will undergo a chain of beta decay events to form more stable compositions, but some neutron-rich fragments are excited enough that they can emit a neutron instead of a  $\gamma$ -ray. Although these 'delayed' neutrons are emitted orders of magnitude later than their prompt counterparts and only make up a very small amount of the total neutron yield, their presence has a large effect on the average neutron lifetime. Therefore the delayed neutrons play a big role in the controllability of a reactor.

As the neutron flux is directly dependent on the distribution of delayed neutron precursors, it is also indirectly influenced by everything on which the precursor distributions depend, like the convection of the salt. This second-order influence makes it so that the neutron flux is also affected by the movements on the salt, even though this movement happens on a much larger timescale than the fast neutronics.

### 2.3.3. Fission heat and salt expansion feedback

Apart from prompt neutrons and fission products, a fission event also releases heat in the form of (prompt)  $\gamma$ -rays. Although a single fission event only releases about 200 MeV, the large rate at which nuclear fission occurs in the reactor core makes nuclear power generation viable to produce electricity. Because of the increased temperature, the salt will expand, and the subsequent decrease in density affects the cross sections  $\Sigma_t$  and neutron diffusion coefficient  $D_n$  of the fuel salt linearly[60]:

$$\Sigma_t(T) = \Sigma_t(T_{ref}) \cdot \frac{\rho_{fuel}(T)}{\rho_{fuel}(T_{ref})}, \quad (2.69)$$

$$D_n(T) = D_n(T_{ref}) \cdot \frac{\rho_{fuel}(T_{ref})}{\rho_{fuel}(T)}. \quad (2.70)$$

Using the same Boussinesq approximation for the salt density defined earlier, Equations 2.69 and 2.70 simplify to:

$$\Sigma_r(T) = \Sigma_r(T_{ref}) \cdot [1 - \beta_{th}(T - T_{ref})] \quad (2.71)$$

$$D_n(T) = D_n(T_{ref}) \cdot \frac{1}{1 - \beta_{th}(T - T_{ref})}. \quad (2.72)$$

Equation 2.71 clearly shows the strong negative reactivity temperature feedback that the fuel salt expansion causes: with increasing temperature, the macroscopic cross sections (among which the fission cross section) of the salt decreases, which decreases the likelihood of a fission event. This in turn decreases the source term in the heat equation (Eq. 2.5), which causes the temperature to decrease again.

It must be noted that, analogously to the concept of prompt and delayed neutrons, the fission fragments can also emit energy when undergoing radioactive decay, orders of magnitude later than the initial fission event. This 'decay heat' also has a small contribution to the overall generation of energy, and subsequently to the salt expansion feedback on the macroscopic cross-sections and neutron diffusion that govern the neutronics. However, since the benchmark that is used in this thesis does not include the heat generated by precursor decay, it is also omitted from the reported simulations. Similarly, the nuclear Doppler effect (explained earlier) on macroscopic cross sections is assumed to be negligible and is omitted from the model for the same reason. Further research can include the direct effect of precursor decay heat and Doppler feedback on the thermal hydraulics of the system; see Chapter 6.

To summarise, the different physical fields that act in the core of a molten salt reactor are all coupled to each other through first-order (e.g. movement of salt  $\rightarrow$  convective heat transfer), second-order (e.g. expansion of heated salt  $\rightarrow$  reduced neutron cross sections  $\rightarrow$  reduced precursor production through fission) and even third-order effects (e.g. emission of delayed neutrons through precursor decay  $\rightarrow$

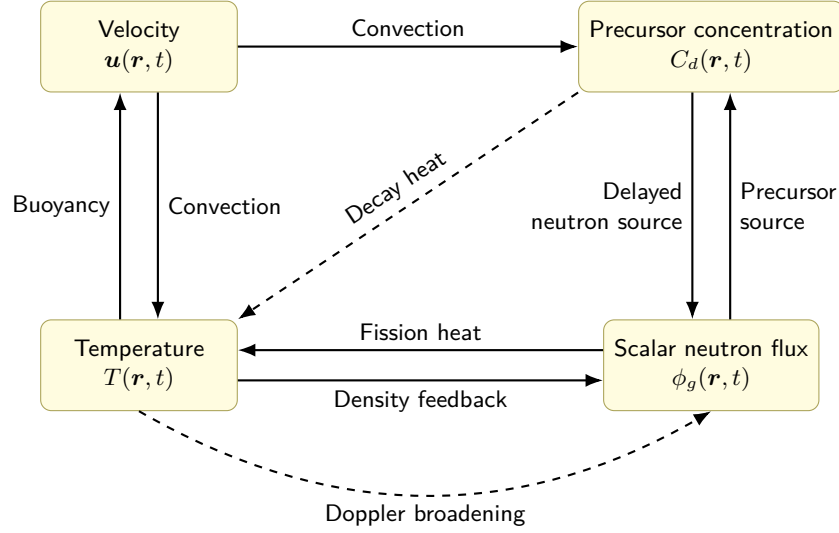


**Figure 2.1:** Schematic of the main physical fields in an MSFR core and the effects coupling the fields. The omitted effects of decay heat and Doppler broadening are indicated with dashed arrows.

increased heat through fission  $\rightarrow$  temperature differences driving the velocity field through buoyancy). Figure 2.1 shows the interactions between the four fields that are considered in the used molten salt model. The non-linear, higher-order way these fields affect one another make the governing equations considerably difficult to solve analytically, and therefore one must rely on numerical modelling to study the thermal hydraulics and neutronics of a molten salt reactor. In the next chapter the implemented method is introduced: The (Filter-Matrix) Lattice Boltzmann Method.

## 2.4. Parallel Programming on the GPU

As will become clear in the next chapter, one of the main advantages of the implemented Lattice Boltzmann Method in comparison with other CFD methods is the locality of its computations, in the sense that the new state at each computational node is determined entirely by the previous state at that node and at its immediate neighbours. The same set of instructions can therefore be independently applied to all nodes in parallel. This local property has motivated the use of parallel hardware for LBM computations, such as GPUs. The remainder of this section describes the hardware and software concepts on which this parallelisation rests.

### 2.4.1. General-purpose computation on GPUs

The graphics processing unit (GPU) was originally developed as a special-purpose computer processor for three-dimensional rendering, which requires the same numerical operation applied independently to large numbers of pixels or vertices. The hardware of a GPU was therefore designed around a large population of relatively simple computational cores operating in parallel, in contrast to the small number of more complex cores that characterise central processing units (CPUs). Successive generations of GPU hardware became progressively more programmable, and in 2006 NVIDIA introduced the Compute Unified Device Architecture (CUDA) programming model to enable any computational workload to use the throughput capability of GPUs independent of any graphics application[61]. Since then, CUDA and GPU-accelerated computing have been applied across a broad range of scientific and engineering domains, from fluid dynamics[62] to medical imaging[63], and, more recently, the development of generative artificial intelligence such as diffusion models[64] or large language models[65].

The use of GPU hardware for numerical workloads beyond graphics is known as general-purpose computing on GPUs, or GPGPU. This procedure makes use of the different design objectives of the two device classes. The CPU is optimised for the rapid execution of a *serial* sequence of operations (or threads), with hardware resources (i.e. the transistors) devoted primarily to data caching and control flow. On the other hand, the GPU is optimised for the *parallel* execution of many thousands of threads, with a larger share of its transistors dedicated to arithmetic units. Within a similar power envelope, a

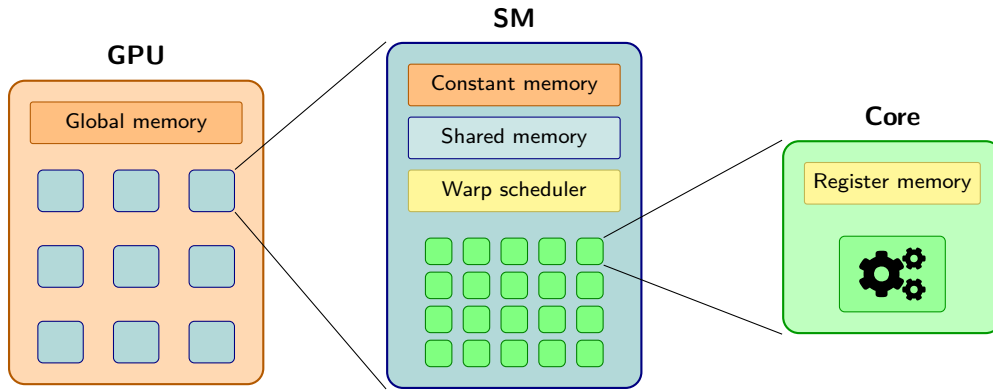


**Figure 2.2:** Hardware architecture of an NVIDIA GPU, including the hierarchy of global, constant, shared, and register memory.

GPU can provide a much higher instruction throughput and memory bandwidth than a CPU[61]. However, this high throughput is only accessible to workloads that can be decomposed into a large number of independent subtasks, such as the update step of the Lattice Boltzmann Method.

#### 2.4.2. NVIDIA GPU Hardware Architecture & Memory Hierarchy

In the CUDA programming model, the GPU is partitioned into a collection of streaming multiprocessors (SMs), which are independent processing units. Each SM contains a register file, a unified data cache, and a number of functional units that perform the arithmetic computations of the user's programme. These functional units are known as CUDA cores and form the elementary execution units to which work is dispatched.

The same partitioning is reflected in the GPU's memory, which is organised in a hierarchy of progressively faster but smaller storage areas closer to the core. Four types of memory are relevant to the present work, summarised in Table 2.2[61]. The trade-off between accessibility, capacity, and access speed that this memory hierarchy embodies is the central consideration in optimising GPU code. Variables used repeatedly within a single computation should reside as close to the cores as possible, while data that must be shared across the device or persist between computational stages must live in global memory. The layered architecture of an NVIDIA GPU alongside the memory hierarchy is displayed in Figure 2.2.

**Table 2.2:** Overview of the relevant memory types of an NVIDIA GPU[61].

| Memory Type     | Description                                                                                                                                                                                                                |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Global memory   | Main storage of the GPU, located on DRAM chips.<br>Accessible by all SMs and CUDA cores in the GPU through the L2 cache, but its physical distance from the cores make information transfer slow.                          |
| Constant memory | Small section of global memory that is read-only, stored on constant caches in the SM.<br>Used for values that are declared constant over the life of the programme.                                                       |
| Shared memory   | Smaller memory type private to a single streaming multiprocessor.<br>Accessible only by CUDA cores within the SM.<br>Stored on the on-chip unified data cache along with the L1 cache, and therefore substantially faster. |
| Register memory | Smallest and fastest memory type, private to a single CUDA core.<br>Stores thread-local variables on which the core's arithmetic operations act.                                                                           |

#### 2.4.3. The CUDA Programming Model

The hardware described above is addressed through the CUDA parallel computing platform, developed by NVIDIA. This model assumes a heterogeneous computing system in which a CPU and a GPU

coexist. The CPU and the memory directly connected to it are called the *host* and *host memory* respectively, while GPU and its directly-attached memory are referred to as the *device* and *device memory* respectively. Execution of a programme or application always starts on the host, which can transfer data to the device, allocate output buffers in the device global memory, launch computational work on the device, and retrieve results when the device work is completed.

The computational work on the device is expressed as kernel functions, which define the operations performed by a single thread, the smallest unit of instructions that can be executed in parallel. In the context of the Lattice Boltzmann Method, a single thread typically corresponds to the update of a single lattice node. Starting or launching a kernel can be thought of as starting many threads executing the same kernel code in parallel on the device[61].

CUDA handles the parallel execution of a kernel through a number of software abstractions, illustrated in Figure 2.3. Threads are organised into thread blocks of user-chosen size, and these blocks are organised into a grid that spans the entire parallel workload. At the time of the kernel launch, the user specifies both the number of blocks and the number of threads per block. Each block is mapped by the device scheduler onto a single streaming multiprocessor for execution. This means that all threads on a given block are executed in a single SM, and therefore have access to that SM's shared memory.

Underneath this structure, the device hardware groups threads into sets of 32, known as warps. In a given warp, all threads are handled by the SM's warp scheduler (also shown in Figure 2.2) to execute the same kernel code, each on its own data. This paradigm is known as Single-Instruction, Multiple-Threads (SIMT)[61]. The SIMT model affects how parallel code should be written. First, the number of threads in a block is most efficient when chosen as a multiple of 32, so that the allocated warp slots are all occupied. Second, the device hardware can only issue one instruction per warp at a time. Threads that follow different branches of a conditional statement therefore cannot execute in parallel. Instead, the warp serialises the divergent branches by masking off the threads on the inactive branch while the active branch executes, and vice versa. This is known as warp divergence, and can significantly reduce throughput in code paths whose control flow depends on thread-dependent data.

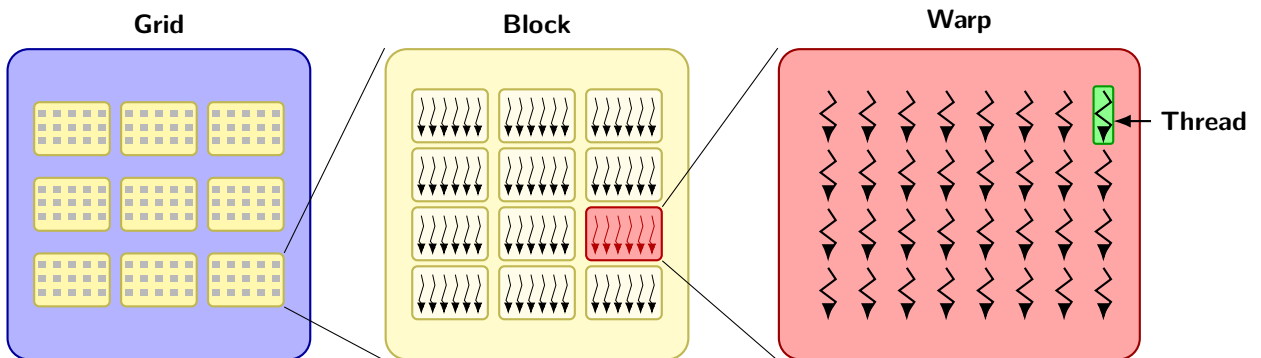


**Figure 2.3:** Schematic of the software abstractions used in the CUDA programming model. The number of threads in a warp is fixed at 32, but CUDA allows for the user to specify both the total number of blocks and the number of threads per block at every kernel launch.

#### 2.4.4. Performance Considerations

Several characteristics of this execution model directly affect the efficiency of a parallel implementation, and these form the basis of the design choices presented in Chapter 3. Memory coalescing refers to the GPU's ability to facilitate simultaneous accesses of consecutive addresses in (global) memory by consecutive threads, provided the data are laid out in memory to allow it. Warp divergence, as explained above, penalises control flow that depends on thread-dependent data. Kernel-launch overhead is the fixed cost incurred each time the host launches a kernel on the device. An implementation with a small number of larger kernels is therefore preferred over one with a large number of smaller kernels. Lastly, host-device transfer cost penalises frequent moving of data between the CPU and the GPU[66]. The techniques implemented in the present research to manage these factors are described in Section 3.8

# 3

## Numerical Method

In this chapter the numerical methods and algorithms of this research are presented. The physical fields introduced in Chapter 2 will be simulated using the Lattice Boltzmann Method (LBM), the outline of which will be given in Section 3.1. Section 3.2 covers the specific extension of the LBM that is implemented, namely the Filter-Matrix Boltzmann Method (FM-LBM). This section also describes the formulation of the present fields and governing equations into solution vectors and source terms that can be used in the FM-LBM framework, including the adjoint cases for neutron flux and precursor concentration. The way in which boundary methods are implemented is presented in Section 3.3, and Section 3.4 covers the non-dimensionalisation of physical parameters & quantities using lattice unit conversion. The structure of the overall numerical scheme to solve the coupled equations is detailed in Section 3.5. The new FM-LBM implementation to model adjoint neutronics is presented in Section 3.6. For transient neutronics simulations, the Predictor-Corrector Quasi-Static Method is used, which is described in Section 3.7. Section 3.8 discusses the way in which GPU acceleration is integrated in the FM-LBM algorithm to achieve parallel computing and accounts for non-structured or complex boundaries. Finally, the method used to handle simulations with complex geometries is explained in Section 3.9.

### 3.1. The Lattice Boltzmann Method

The use of kinetic theory and the Boltzmann equation introduced in Section 2.1.4 enables the study of fluid dynamics problems at a mesoscopic scale, and from its solution one can often find a solution to the NSE by calculating the moments of the distribution function  $f$ . However, as  $f$  is a function of seven parameters  $(x, y, z, \xi_x, \xi_y, \xi_z, t)$ , finding an analytical solution for it proves even more difficult than for the traditional NSE. Fortunately, the numerical scheme for the Boltzmann equation turns out to be relatively simple to solve and to parallelise on the GPU.

In the numerical scheme, the continuous distribution function  $f$  is replaced by the discrete-velocity distribution function  $f_i(\mathbf{x}, t)$ , representing the density of particles with (discrete) velocity  $\mathbf{c}_i = (c_{ix}, c_{iy}, c_{iz})$  at position  $\mathbf{x}$  and time  $t$ . All argument variables of the distribution function  $f$  are discretised. The spatial positions  $\mathbf{x}$  on which  $f_i$  can reside are located on a square lattice with spacing  $\Delta x$ . Time is also discretised in time steps  $\Delta t$ . This discretisation leads to a set of discrete velocities  $\mathbf{c}_i$  chosen such that, at each time step, the distribution functions can move from one lattice point to a neighbouring point.

Together with a corresponding set of weighting coefficients  $\omega_i$  these discrete velocities form lattice velocity sets. Depending on what physical properties are to be simulated, one can choose different lattice velocity sets. These sets are often notated by the abbreviation  $DdQq$ , with  $d$  indicating the number of spatial dimension and  $q$  the number of velocity directions. Figure 3.1 displays the most common velocity sets for 2 and 3 dimensions. Analogously to Equation 2.11, macroscopic variables can be retrieved by calculating the moments of these distribution functions, which in the discretised space reduces to summations:



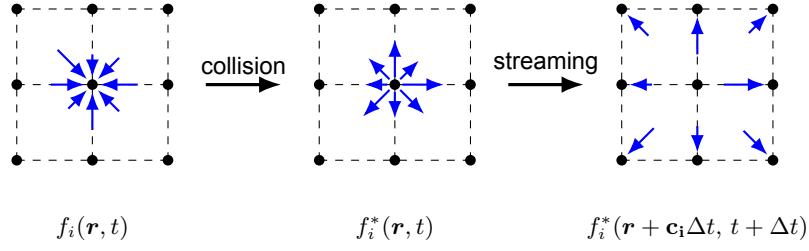


**Figure 3.2:** Visual representation of the collision and streaming step of the LBM for the distribution function  $f_i$ .

through Chapman-Enskog analysis that the force-free Navier-Stokes equations can be solved in the limit by the discretized approach of the Lattice Boltzmann Equation. Therefore a collision operator must be chosen that 'relaxes' the distribution functions towards the equilibrium distribution. In a sense, this collision operator should represent all the forces and effects that influence the behaviour of the physical system that is represented by the distribution function  $f_i$ . A very simple choice for  $\Omega$  is the Bhatnagar-Gross-Krook (or BGK) operator, which relaxes the distribution with a single relaxation parameter  $\tau$ :

$$\Omega_{BGK}(f) = -\frac{1}{\tau}(f_i - f_i^{eq}). \quad (3.6)$$

For this operator, the relaxation parameter  $\tau$  is inherently linked with the kinematic shear viscosity  $\nu$  of the macroscopic fluid; more explicitly,  $\nu = c_s^2(\tau - \frac{\Delta t}{2})$ . Although the BGK operator is very simple to implement due to its single parameter characteristic, its performance is often lacking in stability and accuracy. This is why more advanced collision operators have been developed over the years, including the Multi-Relaxation Times operator (MRT), the Moment-Projected Boltzmann method (MPB), and the Filter-Matrix Lattice Boltzmann Method (FM-LBM), the latter of which is used in this research.

### 3.2. The Filter-Matrix Lattice Boltzmann Method

The collision operator for the Filter-Matrix Lattice Boltzmann Method can be derived by applying a shift of half a grid spacing and half a time step. Using this staggered grid, the Lattice Boltzmann equation can be rewritten to:

$$f_i\left(\mathbf{r} + \frac{\Delta t}{2}\mathbf{c}_i, t + \frac{\Delta t}{2}\right) = f_i\left(\mathbf{r} - \frac{\Delta t}{2}\mathbf{c}_i, t - \frac{\Delta t}{2}\right) + \Delta t \Omega_i(\mathbf{r}, t). \quad (3.7)$$

Applying a Taylor expansion around  $f_i(\mathbf{r}, t)$  and filling it in the staggered LBE yields:

$$f_i\left(\mathbf{r} \pm \frac{\Delta t}{2}\mathbf{c}_i, t \pm \frac{\Delta t}{2}\right) = f_i(\mathbf{r}, t) \pm \frac{\Delta t}{2}\Omega_i(f) + \mathcal{O}(\Delta t^2). \quad (3.8)$$

Next, the Chapman-Enskog analysis can be applied again to obtain expressions for the discretised distribution functions  $f_i$  and collision operator  $\Omega_i$  in terms of macroscopic variables. These expressions ensure that, as explained in Section 2.1.4, the staggered LBE correctly simulates incompressible flows according to the NSE. The obtained formulations are:

$$f_i(\mathbf{r}, t) = \rho \omega_i \left[ 1 + \frac{\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r})}{c_s^2} + \frac{1}{2} \left( \frac{(\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r}))^2}{c_s^4} - \frac{\mathbf{u}(\mathbf{r}) \cdot \mathbf{u}(\mathbf{r})}{c_s^2} \right) - \nu \left( \frac{(\mathbf{c}_i \cdot \nabla)(\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r}))}{c_s^4} - \frac{\nabla \cdot \mathbf{u}(\mathbf{r})}{c_s^2} \right) \right], \quad (3.9)$$

$$\Omega_i(f) = \frac{\rho \omega_i}{c_s^2} \left[ (\mathbf{c}_i \cdot \nabla)(\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r})) - c_s^2 \nabla \cdot \mathbf{u}(\mathbf{r}) + \mathbf{c}_i \cdot \mathbf{f} \right], \quad (3.10)$$

where, again,  $c_s$ ,  $\nu$ , and  $\mathbf{f}$  denote the speed of sound in the fluid, the kinematic viscosity of the fluid, and the body force acting on the fluid (with Cartesian components  $f_x, f_y, f_z$ ). Substituting these expressions back into 3.7 (and omitting the  $\mathcal{O}(\Delta t^2)$  term) yields:

$$\begin{aligned} f_i\left(\mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2}\right) = \rho \omega_i \left[ 1 + \frac{\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r})}{c_s^2} + \frac{1}{2} \left( \frac{(\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r}))^2}{c_s^4} - \frac{\mathbf{u}(\mathbf{r}) \cdot \mathbf{u}(\mathbf{r})}{c_s^2} \right) \right. \\ \left. - \nu \left( \frac{(\mathbf{c}_i \cdot \nabla)(\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r}))}{c_s^4} - \frac{\nabla \cdot \mathbf{u}(\mathbf{r})}{c_s^2} \right) \right. \\ \left. \pm \frac{\Delta t}{2} \left( \frac{(\mathbf{c}_i \cdot \nabla)(\mathbf{c}_i \cdot \mathbf{u}(\mathbf{r}))}{c_s^2} - \nabla \cdot \mathbf{u}(\mathbf{r}) + \frac{\mathbf{c}_i \cdot \mathbf{f}}{c_s^2} \right) \right]. \end{aligned} \quad (3.11)$$

This equation can be written more compactly as a matrix multiplication by introducing the filter matrix  $E_{ik}$  and solution vector  $\alpha_k^\pm(\mathbf{r}, t)$ :

$$f_i\left(\mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i E_{ik} \alpha_k^\pm(\mathbf{r}, t). \quad (3.12)$$

In this formulation, the filter matrix contains terms depending on the discrete velocities  $\mathbf{c}_i$ , while the solution vector contains terms depending on the macroscopic variables  $\rho, \mathbf{u}, \nu$ , and  $\mathbf{f}$ . The explicit formulas for  $E_{ik}$  and  $\alpha_k^\pm(\mathbf{r}, t)$  also depend on what velocity set DdQq is chosen and what physical quantity is modelled. Before the matrices and vectors used in this research are given, it is worthwhile to note that Equation 3.12 can be inverted by defining the inverted matrix  $E_{ki} = (\omega_i E_{ik})^{-1}$ , which leads to:

$$\alpha_k^\pm(\mathbf{r}, t) = \sum_i E_{ki} f_i\left(\mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2}\right). \quad (3.13)$$

Using these relations the collision step of the Lattice Boltzmann Method can be reformulated as matrix multiplications. First, the (pre-collision) solution vector  $\alpha_k^-(\mathbf{r}, t)$  is constructed from  $f_i$  using Equation 3.13. Next,  $\alpha_k^-$  is updated to the post-collision solution vector  $\alpha_k^+$ . The way in which this update happens depends on the choice of velocity set and the equations that govern the behaviour of the physical quantity. Lastly, the vector  $\alpha_k^+$  is transformed back to the post-collision distribution function  $f_i^*$  using Equation 3.12.

### 3.2.1. Filter Matrix Implementations for the Coupled Fields

#### Chosen Velocity Sets and Filter Matrices

In this research, two distinct velocity sets are implemented, each with their own corresponding filter matrices. To model the complex hydrodynamics and momentum transport governed by the NSE, the D3Q19 scheme is used. Note that there are other schemes available for 3D fluid simulations, such as D3Q15 or D3Q27, that each have different trade-offs regarding computational intensity and numerical accuracy. The D3Q19 scheme is chosen a priori as a deliberate compromise between accuracy and speed, but further research can investigate the use of different schemes in the context of reactor multi-physics simulations, especially in the case of more complex turbulent flow (see Chapter 6). For scalar quantities, such as enthalpy, neutron flux or precursor concentration, whose advection-diffusion equations are less complex than the NSE, the D3Q7 scheme is deemed sufficient. The technical reason for this is that the NSE require velocity moments up to the second order, but the ADE only requires the zeroth and first moments[67].

For the D3Q7 velocity set, the weights and filter matrix are:

$$\omega_i^7 = \begin{cases} 1/4, & i = 1 \\ 1/8, & i = 2, 3, 4, 5, 6, 7 \end{cases} \quad (3.14)$$

$$E_{ki}^7 = \left[ 1, c_{ix}, c_{iy}, c_{iz}, 4c_{ix}^2 - 1, 4c_{iy}^2 - 1, 4c_{iz}^2 - 1 \right]^T, \quad (3.15)$$

Where index  $i = 1$  denotes the zero velocity (i.e.  $\mathbf{c}_1 = [0, 0, 0]^T$ ). For the D3Q19 velocity set, the weights and filter matrix are:

$$\omega_i^{19} = \begin{cases} 1/3, & i = 1 \\ 1/18, & i = 2, 3, 4, 5, 6, 7 \\ 1/36, & i = 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 \end{cases} \quad (3.16)$$

$$E_{ki}^{19} = \begin{bmatrix} 1, \\ c_{ix}, c_{iy}, c_{iz}, 3c_{ix}^2 - 1, 3c_{iy}^2 - 1, 3c_{iz}^2 - 1, \\ 3c_{ix}c_{iz}, 3c_{ix}c_{iy}, 3c_{iy}c_{iz}, 3c_{ix}(c_{iy}^2 - c_{iz}^2), 3c_{iy}(c_{iz}^2 - c_{ix}^2), 3c_{iz}(c_{ix}^2 - c_{iy}^2), \\ c_{ix}(3c_{iy}^2 + 3c_{iz}^2 - 2), c_{iy}(3c_{ix}^2 + 3c_{iz}^2 - 2), c_{iz}(3c_{ix}^2 + 3c_{iy}^2 - 2), \\ 3(2c_{ix}^2 - c_{iy}^2 - c_{iz}^2)(|\mathbf{c}_i|^2 - \frac{3}{2}), \\ 3(c_{iy}^2 - c_{iz}^2)(|\mathbf{c}_i|^2 - \frac{3}{2}), \\ 3|\mathbf{c}_i|^2(|\mathbf{c}_i|^2 - 2) \end{bmatrix}^T. \quad (3.17)$$

It should be noted that the speed of sound  $c_s$  is also expressed in lattice units and varies for different velocity sets. For D3Q7, the speed of sound  $c_s = 1/2$  ls lt<sup>-1</sup>, while for D3Q19  $c_s = 1/\sqrt{3}$  ls lt<sup>-1</sup>. Although these schemes are originally defined for 3D problem modelling, in this research they are mainly implemented to model 2D multiphysics flow. It can be shown that the D3Q19 and D3Q7 can be projected onto the  $x - y$  plane to become the D2Q9 and D2Q5 velocity sets respectively[18].

### Momentum

To model the momentum transport through a fluid using the Filter-Matrix Lattice Boltzmann Method with the D3Q19 velocity set, the solution vector  $\alpha_k^\pm$  accompanying the filter matrix  $E_{ki}^{19}$  can be found by combining Equations 3.12, 3.10 and 3.17[23]:

$$\alpha_k^\pm = \begin{bmatrix} \rho \\ \rho u_x \pm \Delta t f_x / 2 \\ \rho u_y \pm \Delta t f_y / 2 \\ \rho u_z \pm \Delta t f_z / 2 \\ 3\rho u_x^2 + \rho(-6\nu \pm \Delta t)\partial_x u_x + (2 - 3B)\rho\nabla \cdot \mathbf{u} \\ 3\rho u_y^2 + \rho(-6\nu \pm \Delta t)\partial_y u_y + (2 - 3B)\rho\nabla \cdot \mathbf{u} \\ 3\rho u_z^2 + \rho(-6\nu \pm \Delta t)\partial_z u_z + (2 - 3B)\rho\nabla \cdot \mathbf{u} \\ 3\rho u_y u_z + \rho(-3\nu \pm \Delta t/2)(\partial_y u_z + \partial_z u_y) \\ 3\rho u_x u_z + \rho(-3\nu \pm \Delta t/2)(\partial_x u_z + \partial_z u_x) \\ 3\rho u_x u_y + \rho(-3\nu \pm \Delta t/2)(\partial_x u_y + \partial_y u_x) \\ \gamma_{sim,1}, k = 11, \dots, 16 \\ \gamma_{sim,2}, k = 17, 18, 19 \end{bmatrix}. \quad (3.18)$$

Apart from the parameters defined earlier,  $B$  is the ratio of the bulk and kinematic viscosities, and  $\partial_{x,y,z}$  denote the partial derivatives in space. The terms  $\alpha_{11-16}^\pm$  and  $\alpha_{17-19}^\pm$  are six third-order terms and three fourth-order terms respectively. These terms emerge as artefacts because of the discretisation of the LBE. Retaining them adds no information to the simulated dynamics, but can act as a source of numerical instability. Therefore they can be filtered out by setting  $\gamma_{sim,1} = \gamma_{sim,2} = 0$  to improve the stability of the scheme, although nonzero values have been investigated in previous research as well[28]. The filtering of these non-physical entries is the property from which the Filter-Matrix LBM takes its name.

### Enthalpy

Aside from fluid dynamics, the LBM can also be adapted into a powerful method to solve advection-diffusion problems, like the heat equation given in Section 2.1.2. For a general advection-diffusion equation describing the dynamics of a scalar field  $C$ , the Lattice Boltzmann Equation for the corresponding discrete-velocity distribution function  $g_i$  can be stated as[18]:

$$g_i(\mathbf{r} + \mathbf{c}_i \Delta t, t + \Delta t) = g_i(\mathbf{r}, t) + \Omega_i(\mathbf{r}, t) + Q_i(\mathbf{r}, t), \quad (3.19)$$

With  $\Omega_i$  and  $Q_i$  appropriate collision operators and source terms, respectively. If  $g_i$  is defined as the discrete-velocity distribution function for the macroscopic enthalpy  $h$ , then:

$$h = \sum_i g_i, \quad (3.20)$$

similarly to  $f_i$  being the distribution function from which the macroscopic density  $\rho$  and velocity  $\mathbf{u}$  could be retrieved. The same staggered-grid approach can be followed as with the NSE and the momentum field to derive a suitable filter-matrix expression for the enthalpy field (using the D3Q7 velocity set):

$$g_i\left(\mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i^7 E_{ik}^7 \beta_k^\pm(\mathbf{r}, t). \quad (3.21)$$

Similarly to Equation 3.13, this relation between  $g_i$  and the new solution vector  $\beta_k^\pm$  can also be inverted to facilitate the different steps in the FM-LBM collision procedure. For the heat equation it can be derived that[23]:

$$\beta_k^\pm = \begin{bmatrix} h \pm \frac{\Delta t}{2} q \\ hu_x + \frac{-8\alpha \pm \Delta t}{8} \partial_x h \\ hu_y + \frac{-8\alpha \pm \Delta t}{8} \partial_y h \\ hu_z + \frac{-8\alpha \pm \Delta t}{8} \partial_z h \\ 0, k = 5, 6, 7 \end{bmatrix}, \quad (3.22)$$

where  $\alpha$  denotes the thermal diffusivity. Similarly to the momentum solution vector  $\alpha_k^\pm$ , the higher-order non-physical terms have been filtered to 0. Furthermore, the heat source  $q$  is defined by macroscopic variables as:

$$q = \gamma_{th} \left( \frac{h_{ref}}{\rho_{ref} C_p} - \frac{h}{\rho C_p} \right) + E_{fission} \sum_g \Sigma_{f,g} \phi_g. \quad (3.23)$$

Here  $\rho_{ref}$  and  $h_{ref}$  are reference values for the density and enthalpy corresponding to some preset reference temperature  $T_{ref}$  ( $= h_{ref}/C_p$ ). From left to right, the terms in the enthalpy source term denote a volumetric heat sink around the reference enthalpy  $h_{ref}$  with transfer coefficient  $\gamma_{th}$ , and the heat source term due to nuclear fission.

### Neutron Flux and Delayed Neutron Precursor Concentration

Just like with the advection-diffusion equation for the fluid's enthalpy, the FM-LBM algorithm can be used to model the diffusion equations for the scalar fields of delayed neutron precursor concentrations  $C_d$  and scalar neutron fluxes  $\phi_g$ . For every precursor family  $d$  or neutron energy group  $g$ , a discrete-velocity distribution function  $p_{i,d}$  and  $n_{i,g}$  can be defined, for which holds:

$$C_d = \sum_i p_{i,d}, \quad (3.24)$$

$$\phi_g = \sum_i n_{i,g}, \quad (3.25)$$

and, using the D3Q7 velocity set:

$$p_{i,d} \left( \mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2} \right) = \sum_k \omega_i^7 E_{ik}^7 \delta_{k,d}^\pm(\mathbf{r}, t), \quad (3.26)$$

$$n_{i,g} \left( \mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2} \right) = \sum_k \omega_i^7 E_{ik}^7 \gamma_{k,g}^\pm(\mathbf{r}, t). \quad (3.27)$$

Comparing Equations 2.5 and 2.20, it becomes clear that they have the same shape, and therefore the (family-dependent) solution vector  $\delta_{k,d}^\pm$  closely resembles that of the enthalpy field expressed in Equation 3.22:

$$\delta_{k,d}^\pm = \begin{bmatrix} C_d \pm \Delta t S_d/2 \\ C_d u_x + \frac{-8D_p \pm \Delta t}{8} \partial_x C_d \\ C_d u_y + \frac{-8D_p \pm \Delta t}{8} \partial_y C_d \\ C_d u_z + \frac{-8D_p \pm \Delta t}{8} \partial_z C_d \\ 0, k = 5, 6, 7 \end{bmatrix}, \quad (3.28)$$

with the substitutions of  $h$  and  $\alpha$  by the macroscopic precursor Concentration  $C_d$  and mass diffusivity  $D_p$ . The new precursor source term  $S_d$  stems from Equation 2.24:

$$S_d = -\lambda_d C_d + \frac{\beta_d}{k_{eff}} \sum_g (\nu \Sigma_f)_g \phi_g, \quad (3.29)$$

where  $\beta_d$  is the delayed neutron fraction of precursor family  $d$ . From left to right, the terms in the precursor source term denote the natural decay of precursors, and the production of precursors due to fission.

The solution vector for the scalar neutron flux also has a similar structure. However, as the scalar neutron flux does not experience the effects of convective transport, these terms are omitted from the solution vector:

$$\gamma_{k,g}^\pm = \begin{bmatrix} \phi_g \pm \Delta t S_g/2 \\ \frac{-8D_g \pm \Delta t}{8} \partial_x \phi_g \\ \frac{-8D_g \pm \Delta t}{8} \partial_y \phi_g \\ \frac{-8D_g \pm \Delta t}{8} \partial_z \phi_g \\ 0, k = 5, 6, 7 \end{bmatrix}, \quad (3.30)$$

with  $D_g$  being the (group-dependent) diffusion coefficient. The neutron source term is obtained from Equation 2.25:

$$S_g = -\Sigma_{t,g} \phi_g + \frac{\chi_g^p (1 - \beta_{tot})}{k_{eff}} \sum_{g'} (\nu \Sigma_f)_{g'} \phi_{g'} + \chi_g^d \sum_d \lambda_d C_d + \sum_{g'} \Sigma_{s,g' \rightarrow g} \phi_{g'}. \quad (3.31)$$

From left to right, the terms in the neutron source term denote the total removal of neutrons due to absorption or collisions, the production of prompt neutrons due to fission, the production of delayed

neutrons due to the natural decay of precursors, and the in-scattering of neutrons from energy group  $g'$  into group  $g$ .

The coupling of the different physical fields is shown in the terms of the solution vector, most notably in the convective ( $\mathbf{u}$ ) terms and the source terms ( $\phi_g, C_d$ ). The enthalpy field, or rather the temperature field also has a more subtle effect on the modelling of  $\phi_g$  and  $C_d$ . As explained in Chapter 2, the expansion of the molten salt due to an increase in temperature affects the macroscopic cross sections  $\Sigma_{x,g}$  and neutron diffusion coefficients  $D_g$ , which translates into altered entries in the solution vectors  $\delta_{k,d}^\pm$  and  $\gamma_{k,g}^\pm$ . These coupled effects show the strength of the Filter-Matrix approach. By extending the algorithm from just velocity fields to also incorporate the scalar fields for enthalpy and even neutronics, it becomes possible to solve for all fields simultaneously, which is paramount for accurate multiphysics simulations.

### 3.3. Boundary Conditions

Although the partial differential equations stated in the previous chapter give an accurate description of the thermal hydraulics in the bulk of a fluid, the problem is not yet complete for all regions of the domain. In order to make the problem well-posed and retrieve unique solutions for different physical systems, one must provide additional information in the form of boundary conditions (and in the transient case, initial conditions as well). As the name indicates, these conditions describe how the different fields are constrained near the boundaries of the domain, e.g. solid walls, inlets, or outlets. For example, they can impose that the value of a quantity  $\varphi$  or its flux  $\frac{\partial \varphi}{\partial n}$  remains fixed to a certain value at a specific boundary. These conditions are known as Dirichlet and Neumann boundary conditions respectively. In the context of the Lattice Boltzmann Method and discrete-velocity distribution functions  $f_i$ , the importance of proper boundary conditions also becomes clear if one considers nodes along the border of the computational domain. Without defining proper ways to handle outwardly streaming populations, the conservation laws for mass, momentum, and energy would no longer be true. Adequate values for incoming populations, i.e. those streaming from solid boundaries into the fluid region, can also be prescribed by LB boundary conditions.

There are a very large number of methods available to apply different types of boundary methods, but most of them fall into two categories. The boundary methods used in this research are all *link-wise*, which means the physical boundaries lie on the links between the nodes where the distribution functions are computed. This approach can also be interpreted as having the lattice points placed at the centres of the computational cells. In the case of straight boundaries aligned with the lattice directions, this brings about half a grid spacing between the computational boundary nodes and the "physical" boundaries. This way the borders of the physical domain coincide with the surfaces of the computational cells. To correctly implement boundary methods in this framework, previous research has found it convenient to add so-called 'ghost nodes' to their computations. These nodes sit just outside the physical domain, with the physical boundary located halfway between the boundary nodes and ghost nodes. However, the introduction of these padding nodes introduces some unwanted numerical overhead, which can affect computational performance for large-scale simulations. Therefore in Section 3.9 a GPU-friendly approach to handling boundary nodes is introduced that eliminates the need for ghost nodes in a link-wise scheme. The link-wise approach is in contrast to *wet-node* boundary schemes, where the boundaries are located on the nodes themselves. Here the lattice points are placed at the vertices of computational cells. In this fashion the borders of the physical domain coincide with vertices of the computational cells. Figure 3.3 displays the differences between link-wise and wet-node boundary schemes[18].

#### 3.3.1. Periodic Boundaries

Periodic boundary conditions apply to situations where the flow solution is periodic, and they state that the fluid leaving the domain on one side will immediately reappear at the opposite side. Therefore mass and momentum are conserved at all times. This condition is straightforward to implement in the LBM by considering the edges of the flow domain as if they were attached together. In the streaming step, unknown populations which enter the domain on one side are given by the post-collision populations that leave the domain on the opposite side:

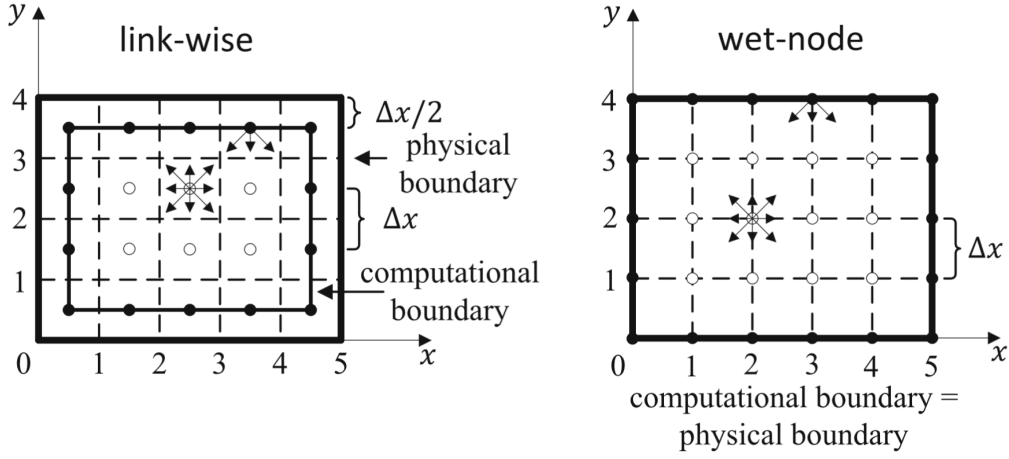


**Figure 3.3:** Two discretisations of the same domain with link-wise (left) and wet-node (right) boundary conditions. Fluid nodes are illustrated as open circles, boundary nodes as solid circles. This schematic is taken from [18].

$$f_i(\mathbf{r}_b, t + \Delta t) = f_i^*(\mathbf{r}_b + \mathbf{L} - \mathbf{c}_i \Delta t, t), \quad (3.32)$$

where  $\mathbf{r}_b$  denotes a boundary lattice point and  $\mathbf{L}$  the periodicity direction and length of the flow pattern. Figure 3.4 demonstrates this implementation of periodic boundaries. This boundary treatment allows for the modelling of 2D flow problems with the developed 3D code. Consider a cubic lattice with  $N_x \times N_y \times N_z$  nodes along the Cartesian axes, upon which 3D fluid flow can be simulated using LBM. By setting the number of nodes along the  $z$ -direction  $N_z = 1$ , and applying periodic boundary conditions on the two  $x - y$  planes, the problem essentially reduces to 2D flow.

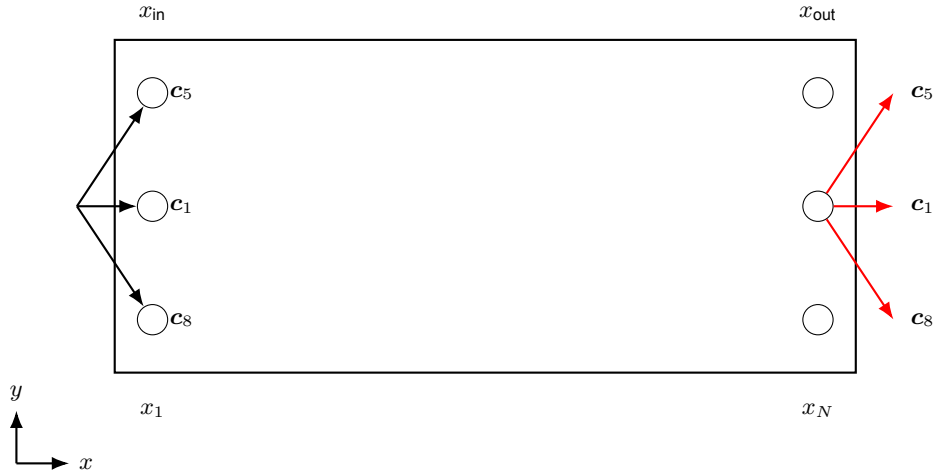


**Figure 3.4:** Realisation of the periodic boundary conditions. Outwardly flowing populations at  $x_N$  (marked in red) re-enter the domain on the opposite side at  $x_1$ .

### 3.3.2. Dirichlet Boundaries for Momentum

A very commonly used Dirichlet condition for fluid-solid boundary interfaces is the no-slip boundary condition. This constraint states that at a solid boundary, a fluid attains zero bulk velocity[47], i.e.  $\mathbf{u}_{wall} = 0$ . This condition can be enforced by reflecting populations that hit a rigid wall back to the lattice point where they originally came from. As explained above, in the most simple link-wise case it is assumed that the physical boundaries are located approximately halfway between the solid and boundary nodes, and so the distributions also travel half a link distance before their directions are inverted. Therefore this method is known as the Halfway Bounce-Back Method or HBB. For resting walls, this can be achieved by replacing the standard streaming step by:

$$f_i(\mathbf{r}_b, t + \Delta t) = f_j^*(\mathbf{r}_b, t), \quad (3.33)$$

where  $j$  is the opposite direction of  $i$  such that, in a particular velocity set,  $\mathbf{c}_j = -\mathbf{c}_i$ . For non-zero wall velocities, the bounced-back particles have to gain or lose a given amount of momentum after hitting the wall. To account for this, the HBB equation is modified to:

$$f_i(\mathbf{r}_b, t + \Delta t) = f_j^*(\mathbf{r}_b, t) + 2\omega_i \rho_{wall} \frac{\mathbf{c}_i \cdot \mathbf{u}_{wall}}{c_s^2}, \quad (3.34)$$

where  $\rho_{wall}$  is the density at the wall. In this research, this local density is approximated by  $\rho_{wall} \approx \rho(\mathbf{r}_b)$ .

### 3.3.3. Boundary Conditions for Scalar Quantities

The application of Dirichlet boundary conditions for the relevant scalar quantities (i.e. enthalpy, neutron flux, precursor concentration) governed by advection-diffusion equations follows a similar approach as that for the fluid dynamics. However, instead of the HBB method, the Anti-Bounce-Back method (or ABB) is implemented for walls located half-way between a boundary node  $\mathbf{r}_b$  and a solid node:

$$g_j(\mathbf{r}_b, t + \Delta t) = -g_i^*(\mathbf{r}_b, t) + 2\omega_i C_{wall}, \quad (3.35)$$

where, again,  $j$  is the direction opposite to  $i$ , and  $C_{wall}$  is the imposed concentration of the scalar quantity at the wall. Note the minus sign. In contrast to the HBB algorithm, which recovers a zero-velocity condition, the anti-bounce-back algorithm recovers a zero-gradient condition for the concentration. In this research, the enthalpy and precursor concentrations in the reactor core are better described by (homogeneous) Neumann boundary conditions, which specify a zero normal gradient, i.e.  $\frac{\partial C_d}{\partial n} = \frac{\partial h}{\partial n} = 0$  and therefore zero normal flux  $\mathbf{j} \cdot \mathbf{n}$  at the wall. This boundary condition can be enforced by setting the physical wall quantity  $C_{wall}$  equal to the macroscopic value at the boundary node  $C(\mathbf{r}_b)$ , which yields[18]:

$$-\mathbf{j} \cdot \mathbf{n} = D \frac{\partial C}{\partial n} \approx D \frac{C(\mathbf{r}_b) - C_{wall}}{|\mathbf{r}_b - \mathbf{r}_{wall}|} = 0. \quad (3.36)$$

Similarly as with half-way-bounce-back, the ABB scheme has to be modified for non-zero wall velocities. This is because the enthalpy and precursor concentrations are also affected by fluid convection. The modified ABB equation reads[68]:

$$g_j(\mathbf{r}_b, t + \Delta t) = -g_i^*(\mathbf{r}_b, t) + 2\omega_i C_{wall} \left[ 1 + \frac{(\mathbf{c}_i \cdot \mathbf{u}_{wall})^2}{2c_s^4} - \frac{|\mathbf{u}_{wall}|^2}{2c_s^2} \right]. \quad (3.37)$$

As explained in Section 2.2.2, the neutron diffusion equation is completed by the vacuum boundary condition, which prescribes that the scalar neutron flux is zero at a linearly extrapolated distance  $\tilde{x}_{boundary}$  from the wall. This extrapolated distance is a function of the neutron diffusion coefficient and formulated in Equation 2.19. Using this, the neutron flux at the wall  $\phi_{wall}$  can be obtained by linearly interpolating between the neutron flux at the boundary ( $= \phi(x_b)$ ) and the (zero) flux at the extrapolated boundary  $\tilde{x}_b$ :

$$\phi_{wall} = \frac{\phi(\tilde{x}_b) - \phi(x_b)}{\tilde{x}_b - x_b} (x_{wall} - x_b) + \phi(x_b) = \frac{\tilde{x}_b}{1/2 + \tilde{x}_b} \phi(x_b). \quad (3.38)$$

Figure 3.5 shows this linear interpolation. As opposed to the enthalpy and precursor concentrations, the neutron flux does not experience convective effects, since its diffusion occurs at a much smaller timescale than that of the thermal hydraulics. Therefore the effect of moving walls can be neglected for the neutron flux, and so the unmodified ABB method (Eq. 3.35) is sufficient.

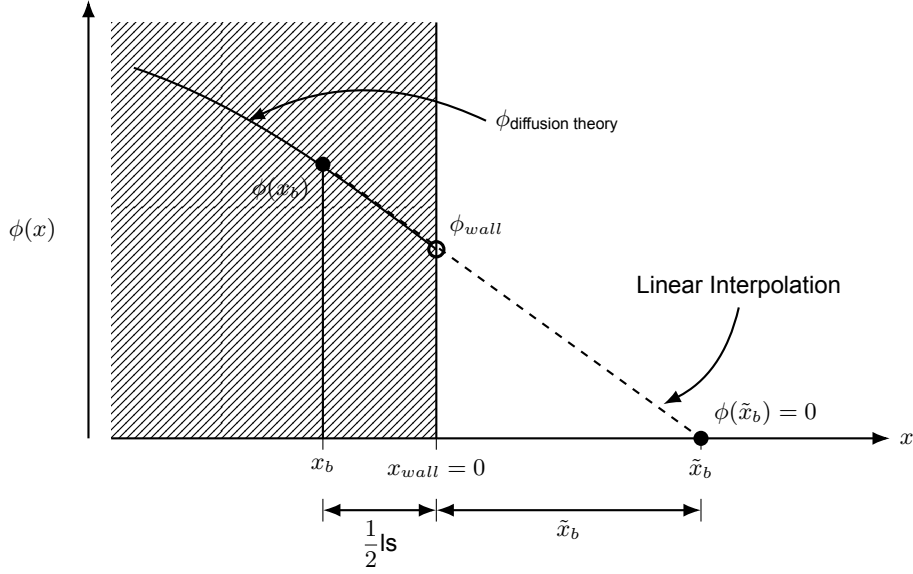


**Figure 3.5:** Schematic of the linear interpolation to determine the scalar neutron flux at the wall for the vacuum boundary condition of the neutron diffusion equation. Figure is inspired by [52].

### 3.4. Lattice Conversion

In Section 3.1 a discretisation of the physical space into a lattice with spacing  $\Delta x$  was introduced, along with a discretisation of time in time steps  $\Delta t$ . In the context of numerical modelling it is often more convenient to perform calculations and discuss intermediate results in terms of these "lattice units", rather than in terms of the physical values they represent. Apart from the "lattice space" unit ( $1 \text{ ls} = \Delta x$ ) and the "lattice time" unit ( $1 \text{ lt} = \Delta t$ ), other physical quantities can also be expressed in their corresponding lattice units. For example, the lattice unit of velocity is given by  $\text{ls lt}^{-1}$ . In fact, if just one more lattice unit for the fluid's density is defined, e.g. the initial density ( $1 \text{ ld} = \rho_0$ ), then all mechanical quantities can be uniquely expressed in their corresponding lattice units as a combination of these three fundamental dimensions[69]. In this research, whenever deemed necessary, an asterisk superscript is added to a macroscopic variable to indicate it is expressed in (non-dimensional) lattice units (e.g.  $A^*$ ). On the other hand, when it is deemed necessary to accentuate that the quantity is expressed in physical units, the subscript "ph" is added. (e.g.  $A_{ph}$ ).

One constraint on the choice of lattice units is the law of similarity. This states that the behaviour of a system or the outcome of a simulation should not depend on whether the dimensional or dimensionless quantities are used. If the dimensionless Reynolds number is taken as an example, then:

$$Re = \frac{U_{ph} L_{ph}}{\nu_{ph}} = \frac{U^* L^*}{\nu^*}. \quad (3.39)$$

The conversion of quantities from physical values to lattice units and back is known as non-dimensionalisation. This is achieved by dividing the dimensional quantity by a chosen reference quantity of the same dimension. This reference quantity is known as the conversion factor. For example, the conversion factor for distance is given by:

$$C_x = \frac{L_{ph}}{L^*} = \frac{L_{ph}}{N} = \Delta x, \quad (3.40)$$

where  $L_{ph}$  is the length of a domain axis expressed in physical values (for example, in meters), and  $N$  is the number of lattice points along that axis. Analogously, one can set  $C_t = \Delta t$  and  $C_\rho = \rho_0$ . Using Equation 3.39 and others like it, the conversion factors of other quantities can be found, e.g. for viscosity:

$$C_\nu = \frac{C_x^2}{C_t}. \quad (3.41)$$

### 3.5. FM-LBM for Coupled Steady-State Simulations

To simulate the steady-state multiphysics of a molten salt reactor, the Filter-Matrix Lattice Boltzmann Method is used as a solver for the momentum field, the enthalpy field, as well as for the delayed neutron precursor concentration and the scalar neutron flux. As mentioned before, the dynamics of this last field occur at a much smaller timescale than the other fields, and therefore it is more convenient to run two instances of the FM-LBM: one for the neutronics and one for the other fields, hereafter called the "thermal hydraulics". If the power iteration method to solve for the multiplication factor  $k_{eff}$  is implemented as well, which was introduced in Section 2.2.2, then the entire scheme is composed of three iterations. After initialisation, the neutron diffusion equation is solved for initial guesses for  $k_{eff}^0$  and precursor concentrations  $C_{d,0}$  using the described FM-LBM algorithm until convergence of the scalar neutron flux  $\phi_g$  of the  $L^2$ -norm, i.e.

$$\max_i \left\{ \sqrt{\frac{\sum_g |\phi_g(\mathbf{r}_i, t) - \phi_g(\mathbf{r}_i, t - \Delta t)|^2}{\sum_g |\phi_g(\mathbf{r}_i, t)|^2}} \right\} < 10^{-6}. \quad (3.42)$$

This will be known as the "inner" iteration. Next, the resulting scalar flux is used to calculate a new fission source vector  $\mathbf{S}$  and corresponding multiplication factor  $k_{eff}$ , until that too has converged. This will be known as the "intermediate" iteration. Note that, for every  $k_{eff}$  power iteration, the inner FM-LBM algorithm is executed again until convergence to get a new value for the fission source. However in practice it has been found that after sufficient time the inner neutronics solver already converges in just one iteration. Lastly, the converged scalar flux and multiplication factor are used to simulate the "slow" thermal hydraulics for one large time step, after which the inner and intermediate iterations recommence. This is repeated until converged solutions have been obtained for the velocity field  $\mathbf{u}$ , enthalpy  $h$ , and precursor concentrations  $C_d$ :

$$\max_i \left\{ \sqrt{\frac{\|\mathbf{u}(\mathbf{r}_i, t) - \mathbf{u}(\mathbf{r}_i, t - 500\Delta t)\|^2}{\|\mathbf{u}(\mathbf{r}_i, t)\|^2}}, \sqrt{\frac{|h(\mathbf{r}_i, t) - h(\mathbf{r}_i, t - 500\Delta t)|^2}{|h(\mathbf{r}_i, t)|^2}}, \sqrt{\frac{\sum_d |C_d(\mathbf{r}_i, t) - C_d(\mathbf{r}_i, t - 500\Delta t)|^2}{\sum_d |C_d(\mathbf{r}_i, t)|^2}} \right\} < 10^{-6}. \quad (3.43)$$

This will be known as the "outer" iteration. Note that for the thermal hydraulics variables, the convergence is checked every 500 time steps to ensure sufficient stability of the simulation results. The entire solving scheme is shown in Figure 3.13.

### 3.6. FM-LBM for Adjoint Neutronics Simulations

#### 3.6.1. Filter Matrices for Adjoint Quantities

Using the previously stated filter-matrix expressions for the advection-diffusion equations for the forward scalar neutron flux and forward precursor concentration, it becomes straightforward to construct similar expressions for the system of equations for the adjoint neutron and precursor concentration, denoted by  $\Phi_g^\dagger$  and  $C_d^\dagger$ . For every precursor family  $d$  or neutron energy group  $g$ , a discrete-velocity distribution function  $p_{i,d}^\dagger$  and  $n_{i,g}^\dagger$  are defined for which holds:

$$C_d^\dagger = \sum_i p_{i,d}^\dagger, \quad (3.44)$$

$$\Phi_g^\dagger = \sum_i n_{i,g}^\dagger, \quad (3.45)$$

and

$$p_{i,d}^\dagger\left(\mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i^7 E_{ik}^7 \zeta_{k,d}^\pm(\mathbf{r}, t), \quad (3.46)$$

$$n_{i,g}^\dagger\left(\mathbf{r} \pm \frac{\Delta t}{2} \mathbf{c}_i, t \pm \frac{\Delta t}{2}\right) = \sum_k \omega_i^7 E_{ik}^7 \theta_{k,g}^\pm(\mathbf{r}, t). \quad (3.47)$$

The solution vectors are also nearly identical to their forward counterparts. However note that, as explained in Section 2.2.4, precursor importance flows *upstream*, and so the signs of the convective terms are flipped:

$$\zeta_{k,d}^\pm = \begin{bmatrix} C_d^\dagger \pm \Delta t S_d^\dagger / 2 \\ -C_d^\dagger u_x + \frac{-8D_p \pm \Delta t}{8} \partial_x C_d^\dagger \\ -C_d^\dagger u_y + \frac{-8D_p \pm \Delta t}{8} \partial_y C_d^\dagger \\ -C_d^\dagger u_z + \frac{-8D_p \pm \Delta t}{8} \partial_z C_d^\dagger \\ 0, k = 5, 6, 7 \end{bmatrix}, \quad (3.48)$$

$$\theta_{k,g}^\pm = \begin{bmatrix} \Phi_g^\dagger \pm \Delta t S_g^\dagger / 2 \\ \frac{-8D_g \pm \Delta t}{8} \partial_x \Phi_g^\dagger \\ \frac{-8D_g \pm \Delta t}{8} \partial_y \Phi_g^\dagger \\ \frac{-8D_g \pm \Delta t}{8} \partial_z \Phi_g^\dagger \\ 0, k = 5, 6, 7 \end{bmatrix}, \quad (3.49)$$

with adjoint source terms derived from Equations 2.66 and 2.67:

$$S_d^\dagger = -\lambda_d C_d^\dagger + \lambda_d \sum_g \chi_g^d \Phi_g^\dagger, \quad (3.50)$$

$$S_g^\dagger = -\Sigma_{t,g} \Phi_g^\dagger + \frac{(\nu \Sigma_f)_g}{k_{eff}^\dagger} \left[ (1 - \beta_{tot}) \sum_{g'} \chi_{g'}^g \Phi_{g'}^\dagger + \sum_d \beta_d C_d^\dagger \right] + \sum_{g'} \Sigma_{s,g \rightarrow g'} \Phi_{g'}^\dagger. \quad (3.51)$$

### 3.6.2. FM-LBM solver for Adjoint Neutronics

Unlike the forward neutron flux and precursor concentrations, these adjoint quantities don't have a direct influence on the multiphysics coupling or the general steady-state behaviour of the fluid. Instead, as explained in Section 2.2.3, the solution for the steady-state adjoint neutron flux  $\Phi^\dagger$  is used as a weight function for the point-kinetic parameters, in order to model the transient behaviour of neutronics after a change in the domain, e.g. a reactivity insertion. The details of this transient scheme are presented in the next section.

The role of the adjoint flux dictates the order in which the multiphysics simulations must be performed. The forward steady-state problem is solved first, yielding the solutions for the velocity  $\mathbf{u}$ , the enthalpy  $h$ , the (forward) neutron flux  $\phi$ , and the (forward) precursor concentration  $C$ . These converged results are then used to solve the steady-state adjoint problem, yielding the solutions for the adjoint neutron flux  $\Phi^\dagger$  and adjoint precursor concentration  $C^\dagger$ . These intermediate results are then used in the calculation of the PK parameters, with which the transient multiphysics simulations can be performed.

The structure of the adjoint solver itself mirrors the forward solver of Figure 3.13, including an inner LBM iteration for the convergence of the adjoint flux  $\Phi_g^\dagger$ , an intermediate power iteration, and an outer LBM iteration for the convergence of the adjoint precursor concentration  $C_d^\dagger$ . Note that in contrast to the forward case, it is not necessary to recalculate solutions for the 'adjoint' versions of the velocity field or the enthalpy. Instead, their steady-state solutions obtained by the forward solvers are used as fixed values in the adjoint solver.

### 3.7. FM-LBM and the Predictor-Corrector Quasi-Static Method

To model the behaviour of the physical systems under transient conditions, the Predictor-Corrector Quasi-Static Method (PC-QSM) is used, the theory of which is described in Section 2.2.3. The approach in this research follows that of Wang et al. ([39]), and uses the steady-state solutions of the different physical fields as initial conditions. The scalar neutron flux  $\phi_g$  is factorised in a slowly changing shape function  $\tilde{\phi}_g(\mathbf{r}, t)$  and a fast changing amplitude function  $n(t)$ . Here the adjoint neutron flux  $\Phi^\dagger$  is used as a weight function for the shape to ensure uniqueness in the factorisation.

In the PC-QSM, the neutron flux  $\phi_g$  is first predicted using the steady-state FM-LBM scheme over a big time step  $\Delta t_B$ , from which the neutron shape function  $\tilde{\phi}_g$  is calculated. Next, the point-kinetic parameters  $\rho_n(t)$ ,  $\beta_d(t)$  and  $\Lambda(t)$  are calculated using Equations 2.50 - 2.53 and the linearly interpolated shape function at intermediate time-steps  $\Delta t_M$ . On the shortest timescale  $\Delta t_S$ , the PK parameters are held constant, and the neutron and precursor amplitudes ( $n$  and  $m_d$ ) are updated according to Equations 2.48 and 2.49. The relations between  $\Delta t_B$ ,  $\Delta t_M$ , and  $\Delta t_S$  are displayed in Figure 3.6. In this research,  $\Delta t_B \propto 10^{-3} - 10^{-4}$ , similar to the thermal-hydraulics time step for steady-state simulations, and the ratios  $\Delta t_B/\Delta t_M$  and  $\Delta t_M/\Delta t_S$  can be varied to study its effects on numerical stability & accuracy.

In the numerical scheme, the point-kinetic equations can be reformulated as a matrix multiplication for an arbitrary number  $N_{TS}$  of small time steps  $\Delta t_S$ :

$$\begin{bmatrix} n(t + N_{TS}\Delta t_S) \\ m_1(t + N_{TS}\Delta t_S) \\ m_2(t + N_{TS}\Delta t_S) \\ \vdots \\ m_d(t + N_{TS}\Delta t_S) \end{bmatrix} = \begin{bmatrix} 1 + \frac{\rho_n - \beta_{tot}}{\Lambda}\Delta t_S & \lambda_1\Delta t_S & \lambda_2\Delta t_S & \cdots & \lambda_d\Delta t_S \\ \frac{\beta_1}{\Lambda}\Delta t_S & 1 - \lambda_1\Delta t_S & 0 & \cdots & 0 \\ \frac{\beta_2}{\Lambda}\Delta t_S & 0 & 1 - \lambda_2\Delta t_S & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\beta_d}{\Lambda}\Delta t_S & 0 & 0 & \cdots & 1 - \lambda_d\Delta t_S \end{bmatrix}^{N_{TS}} \begin{bmatrix} n(t) \\ m_1(t) \\ m_2(t) \\ \vdots \\ m_d(t) \end{bmatrix}. \quad (3.52)$$

Note that in this expression, only the terms in the first column vary in time for different intermediate time steps  $\Delta t_M$ ; the other entries can be precomputed once at the start of the simulations. The initial values for the amplitudes are given by[70]:

$$n(0) = 1, \quad (3.53)$$

$$m_d(0) = \frac{\beta_d}{\lambda_d\Lambda}. \quad (3.54)$$

When the big time step  $\Delta t_B$  is reached by repeatedly calculating the PK parameters and updating the amplitudes, the predicted neutron flux is corrected by multiplying the predicted shape function with the adjusted neutron amplitude. With the corrected neutron flux, one cycle of the FM-LBM is performed for the thermal hydraulics fields. This procedure is then repeated until the total time of the simulation is reached. The modified multiphysics scheme, including the PC-QSM for transient neutronics, is displayed in Figure 3.14.

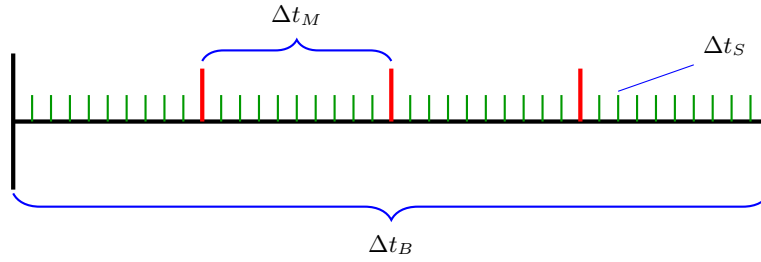


**Figure 3.6:** The relation between the three time steps used in the Predictor-Corrector Quasi-Static Method. Figure is inspired by [71].

## 3.8. GPU Acceleration

### 3.8.1. Julia-CUDA

As the FM-LBM algorithm is composed of many smaller collision and streaming steps, which are independent and locally calculated for different lattice points, parallel computing can be used to increase speed and efficiency. Moreover, as the complex computations are reduced to matrix multiplications and other relatively easy arithmetic operations, they can be executed on a GPU.

As in previous studies, the codebase of this research is developed in Julia. Although Python is the main programming language taught in Applied Physics undergraduate and graduate courses, the Julia programming language offers several advantages over Python in high-performance computing. While still a dynamic, high-level language and therefore easy to read, Julia achieves just-in-time (JIT) compilation using LLVM techniques to deliver execution speeds similar to static, low-level languages such as C and C++. Furthermore, Julia contains the CUDA.jl library, which enables memory allocation and code execution on the GPU[72].

To take advantage of this GPU framework, the different steps of the FM-LBM algorithm are written into small units of code, called CUDA kernels, that are called once during simulation but are executed in parallel by a number of threads on the GPU core. In this research, kernel functions are written for the initialisation, collision, and (bulk) streaming of the distribution functions, as well as for the handling of boundary conditions, the calculation of macroscopic quantities, and the convergence calculations. These kernels are parallelised across the different computational nodes representing the discretised lattice points in space. For multi-group variables, namely the precursor concentrations and the neutron fluxes, the collision, streaming, and boundary kernels are also parallelised across the different groups. It must be stated that in this research the streaming kernels were also tested for parallelisation across the velocity directions. However, this did not result in faster simulations, and therefore the velocity parallelisation was not implemented in the end. To minimise host-device transfer costs, the data stays stored on the GPU during runtime, and moves back to the CPU only when the simulation is completed. During steady-state simulations, a single boolean variable is transferred to the host every time the convergence kernel is launched to notify the CPU whether to continue the solver. Figure 3.7 shows the division of code execution between the CPU and the GPU, including the parameters, the CUDA kernels, and the saving of results.

### 3.8.2. Race Conditions

In the context of parallel computing, one must be aware of race conditions. This phenomenon can occur when two or more threads access and modify the same piece of data in the computer's memory. If the threads take a different amount of time than expected, the order of data access (and possible modification) is also executed in a different order than expected, which can cause bugs or other unwanted behaviour. In the context of LBM, this is especially relevant for the streaming step, where for each node a thread reads data (i.e. the distribution function) of the "upstream" neighbouring node and writes it to the memory address of the new node. However, if the neighbouring node's data has already been overwritten by a different streaming thread running in parallel, then this will cause an erroneous streaming calculation. The same logic holds for the modification of memory after a collision step. To prevent this type of race condition, two copies of arrays holding the distribution functions' data are stored in memory. For a distribution function  $f$ , the collision kernel reads from the  $f\_dist$  array, performs the

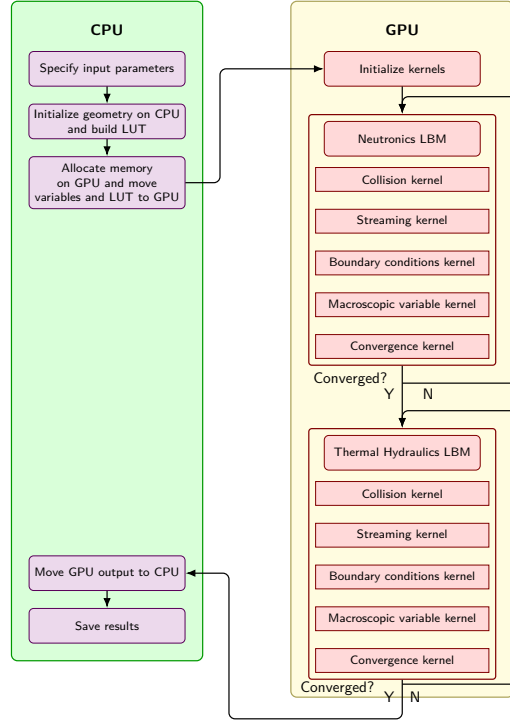


**Figure 3.7:** Overview of the GPU-accelerated FM-LBM algorithm. The same structure holds for steady-state, adjoint, and transient simulations. The diagram shows the division of tasks between the CPU and GPU, and how the processing units share data.

filter-matrix calculations, and then writes to the  $f\_star$  array. In the streaming & boundary conditions kernel, the data is read from  $f\_star$  and, after the correct propagation, written back to  $f\_dist$ . The arrays holding the macroscopic variables don't need a copy, their entries are updated in the macroscopic variable kernel by reading from  $f\_dist$ , and they are read from in the convergence kernel. This approach ensures that no race conditions can arise inside individual kernels launches that modify the data of the distribution functions.

In the LBM, another type of race condition can occur between kernel launches. For example, if the streaming kernel is launched before all threads have finished their execution of the collision kernel, then faulty, unsynchronised data from  $f\_star$  may be read. To prevent this type of race condition, the `CUDA.synchronize()` function is used after each kernel launch to ensure that all operations are fully completed before the programme moves forward.

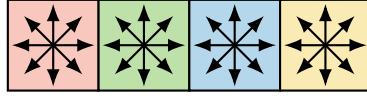
### 3.8.3. Performance Gains

To make optimal use of the capabilities of the GPU, a number of additional techniques have been adopted to improve the computational efficiency of the FM-LBM solver. First, the multidimensional arrays that store the distribution functions and macroscopic variables are flattened to 1D arrays on the GPU. This is because storing multidimensional arrays on the GPU would incur additional overhead when indexing over the different dimensions. Instead, different indexing conventions are used to access data defined by 3 (e.g. the macroscopic density  $\rho(x, y, z)$ ), 4 (e.g. the distribution function for enthalpy  $g_i(x, y, z)$ ) or even 5 parameters (e.g. the distribution function for neutron flux  $n_{i,g}(x, y, z)$ ):

$$\begin{aligned}
 idx_{3D \rightarrow 1D} &= x + yN_x + zN_xN_y, \\
 idx_{4D \rightarrow 1D} &= x + yN_x + zN_xN_y + iN_xN_yN_z, \\
 idx_{5D \rightarrow 1D} &= x + yN_x + zN_xN_y + iN_xN_yN_z + gN_xN_yN_zN_i.
 \end{aligned} \tag{3.55}$$

Here,  $(x, y, z)$  denote the spatial coordinates,  $i$  the lattice velocity direction, and  $g$  the group that together determine the requested element's location in the flattened array. Note that in the index functions,

Grid layout



Memory address in 1D array

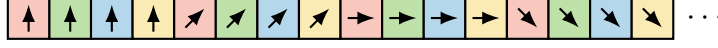


**Figure 3.8:** Illustration of the array flattening for a 2D array. Storing distribution functions with equal velocity directions contiguously optimises GPU performance through memory coalescing. Figure is inspired by [28].

the spatial parameters are placed before any direction or group index. This is done to facilitate the process known as memory coalescing. When consecutive threads of a GPU warp access consecutive addresses, the hardware services those addresses with a single coalesced transaction, rather than many separate ones. Because each thread is mapped to one lattice node, the flattened layout must therefore be ordered so that threads handling spatially adjacent nodes access adjacent addresses. Placing the spatial indices as the fastest-varying index achieves this: for a given velocity direction (and group), the distribution values of all lattice nodes are stored contiguously, so a warp processing a contiguous block of nodes reads contiguous memory. The implementation of this multidimensional array flattening is shown in Figure 3.8.

Furthermore, the collision kernels of the different physical fields operate on small, fixed-sized data structures. For example, for the FM momentum collision scheme in a single lattice point  $(x, y, z)$ , calculations are performed on the 19-element distribution vector  $f_i(x, y, z)$ , the 19-element solution vectors  $\alpha_k^\pm$ , the  $19 \times 19$  filter matrix  $E_{ki}^{19}$  (as well as its inverse), and the discrete velocity set D3Q19 with accompanying weights. Regular Julia Arrays are heap-allocated, carry their size as runtime information, and therefore require dynamic memory management. This is not allowed for code to run on CUDA devices, and therefore the Array type is unsuitable for GPU kernels.

The `StaticArrays.jl` package addresses this issue by encoding the array dimensions in the type itself. Because their size is fixed at compile time, statically-sized arrays are of the `isbits` type and thus stack-allocated. Therefore arrays of this type can be used directly within `CUDA.jl` kernels. The compile-time knowledge of vector and matrix sizes allows the compiler to construct specialised code for the size of the arrays, e.g by unrolling vector operations and loops. It also allows the compiler to keep the data in registers or local memory rather than global memory, and to eliminate unnecessary bounds-checking and dynamic-allocation overhead. In this work, the discrete velocities  $c_i$ , the corresponding weights  $\omega_i$ , the filter matrices  $E_{ki}^Q$  are all converted to `StaticArrays`.

Statically-sized arrays are used in this research in two more distinct roles. First, they are used to represent small fixed-sized quantities attached to each node in the collision kernel, i.e. the per-node distribution function  $f_i$  and solution vectors  $\alpha_k^\pm$ . Holding these as `SVector` type structures allows each thread to construct and manipulate its own node-local data entirely in registers, without accessing global memory beyond the initial load and final store. Second, it must be noted that the collision step of the FM-LBM scheme is, at its core, a dense linear-algebra collision: the pre-collision solution vector is obtained from the distribution functions through a matrix multiplication with the filter matrix, the solution vector is transformed, and the post-collision distributions are recovered through a second matrix multiplication. In prior work, as no standard package within the CUDA framework was present, this operation was implemented as hand-written nested loops within a custom kernel, with the loop ordering tuned manually to minimise costly global-memory accesses[28]. In the present implementation the  $q$ -element distribution vector is instead held as an `SVector` and the filter matrix as an `SMatrix`, and the two matrix-vector products are written as ordinary multiplications. As explained above, the compiler resolves these at compile time, fully unrolling the underlying loops into scalar arithmetic on register-level data.

### 3.9. Handling Complex Geometries with Lookup Tables

On a regular  $N_X \times N_Y \times N_Z$  Cartesian lattice with axis-aligned domain walls, the treatment of boundaries is straightforward. Nodes at the edges of the computational domain (i.e.  $x \in \{1, N_X\} \vee y \in \{1, N_Y\} \vee z \in \{1, N_Z\}$ ) are considered boundary nodes, as opposed to the "bulk" fluid nodes at the interior of the domain. The streaming step in the bulk reduces to a uniform shift of the discrete-velocity distribution functions, and the boundary methods described in Section 3.3 can be applied at the domain edges. This regularity is what makes the standard LBM streaming so efficient.

The regularity is lost once the simulated domain contains internal obstacles or curved walls that do not align with the lattice axes. The fluid subdomain  $\Omega_f$  and the solid subdomain  $\Omega_s$  are then no longer determined by the grid coordinates alone, and the physical boundaries no longer coincide with the surfaces of the computational cells. An accurate treatment of the boundary requires, for every lattice direction  $i$  that crosses a wall, the fractional distance  $q$  from the fluid node to the wall,

$$q(\mathbf{r}, i) = \frac{|\mathbf{r} - \mathbf{r}_w|}{|\mathbf{c}_i| \Delta t}, \quad (3.56)$$

where  $\mathbf{r}_w$  denotes the point where the link from  $\mathbf{r}$  in the direction of  $\mathbf{c}_i$  intersects the wall.

#### 3.9.1. Interpolated Bounce-Back Method

Standard halfway bounce-back (HBB) assumes that boundaries lie exactly midway between a fluid node and its solid neighbour, i.e.  $q = 1/2$  for every boundary-crossing link. For shifted or curved walls this assumption introduces a wall-position error of order  $\Delta x$ , which is incompatible with the second-order accuracy that LBM otherwise delivers in the bulk[18]. The interpolated bounce-back (IBB) scheme of Bouzidi et al.[73] addresses this by using the actual fractional distance  $q$  to reconstruct the post-streaming distribution at the boundary node from the post-collision distributions of the surrounding fluid nodes. In its linear form, the IBB scheme states:

$$f_i(\mathbf{r}_b, t + \Delta t) = \begin{cases} 2q f_j^*(\mathbf{r}_b, t) + (1 - 2q) f_j^*(\mathbf{r}_f, t) & q < \frac{1}{2}, \\ \frac{1}{2q} f_j^*(\mathbf{r}_b, t) + \frac{2q - 1}{2q} f_i^*(\mathbf{r}_b, t) & q \geq \frac{1}{2}, \end{cases} \quad (3.57)$$

where, again,  $j$  denotes the opposite direction of  $i$ , and  $\mathbf{r}_f$  denotes the location of the adjacent fluid node with which the interpolation is performed. A schematic of the IBB method is shown in Figure 3.9. Note that both cases reduce to simple halfway bounce-back for  $q = 1/2$ .

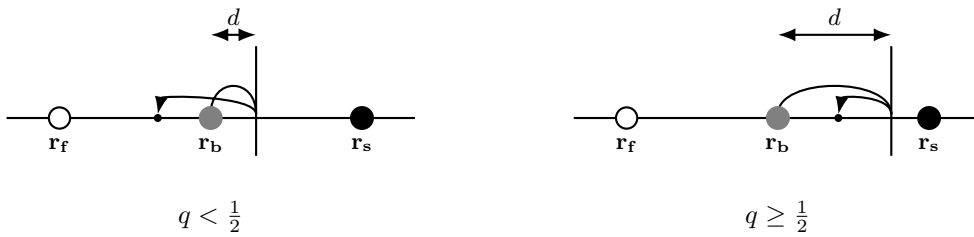


**Figure 3.9:** Schematic of the interpolated bounce-back scheme. For  $q < 1/2$ , the distance  $d$  between the wall (vertical line) and the boundary node  $\mathbf{r}_b$  is smaller than half a lattice spacing. For  $q \geq 1/2$ ,  $d$  is larger than half a lattice spacing, and the endpoint of the streaming population (small black circle) lies between the wall and  $\mathbf{r}_b$ . Figure is inspired by [18].

#### 3.9.2. Lookup Table Construction

Recomputing  $q(\mathbf{r}, i)$  and the inside/outside test at every time step by every thread inside the streaming or boundary kernel introduces thread-dependent control flow and the associated warp divergence, significantly reducing instruction throughput as explained in Section 2.4. For most cases, however, the geometries are fixed for the duration of the simulation. In other words, boundary nodes stay boundary nodes, and their distance to the boundary stays constant over time. These quantities can therefore

be precomputed once before the time iterations begin, and stored. This is the motivation for a lookup table (LUT) approach, where the geometric analysis is performed once on the CPU and consulted by the GPU kernels through simple (and fast) array reads. The approach used in this research is an adaptation of that introduced by Jaber et al.[45] for GPU-native embedding of complex geometries.

The geometry is defined modularly on the Cartesian lattice. Physical boundaries (e.g. walls, cylinders, blocks) can be specified in physical space and mapped onto the lattice, and each lattice node is then assigned a class:

$$\text{class}(\mathbf{r}) = \begin{cases} 1 & \mathbf{r} \in \Omega_s \quad (\text{solid}), \\ 2 & \mathbf{r} \in \Omega_f \wedge \exists i : \mathbf{r} + \mathbf{c}_i \in \Omega_s \quad (\text{boundary}), \\ 0 & \text{otherwise} \quad (\text{fluid}). \end{cases} \quad (3.58)$$

For domains containing different types of boundaries, such as inlets or outlets, extra classes can be defined as desired. An example domain with complex internal obstacles is shown in Figure 3.10.

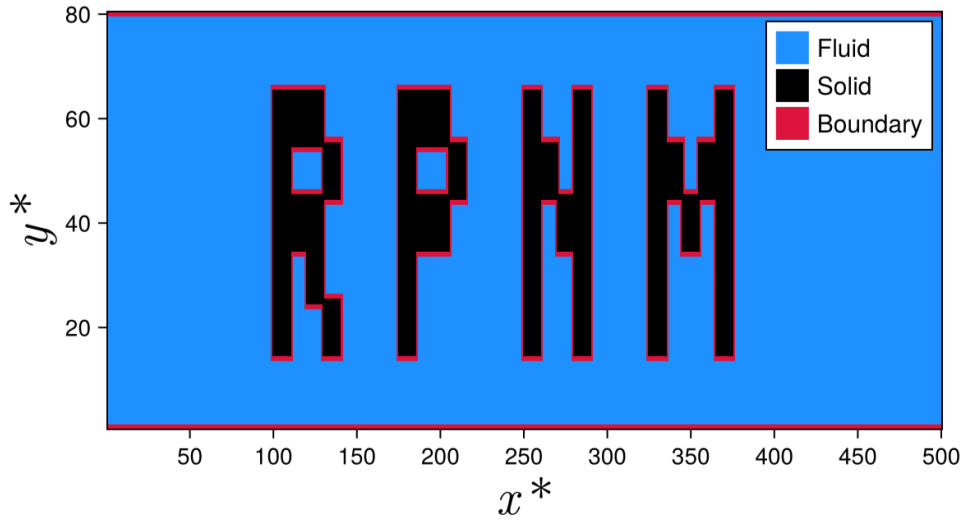


**Figure 3.10:** Example of a computational domain with complex geometries. The 'RPNM' letters form solid obstacles in a plane channel, and the surrounding nodes are classified as boundary nodes. The axes are expressed in lattice units. The corresponding flow field is shown in Figure 4.32.

From this classified grid the LUT is constructed on the CPU. For every node–direction pair  $(\mathbf{r}, i)$ , the LUT stores a tuple:

$$\text{LUT}(\mathbf{r}, i) = (\text{class}(\mathbf{r}), \text{src}(\mathbf{r}, i), q(\mathbf{r}, i)), \quad (3.59)$$

where  $\text{src}(\mathbf{r}, i)$  is the flattened index of the upstream node from which the post-collision distribution must be pulled. For solid and boundary nodes, the reserved value 0 acts as a sentinel for boundary-crossing or solid links. Note that Julia uses 1-based indexing, so the value 0 is unused by valid array indices and free to repurpose as a sentinel. It should also be pointed out that the classification of a node depends on the chosen velocity set: a node that is a boundary node under the D3Q19 scheme may be a fluid node under D3Q7, as illustrated in Figure 3.11. A separate LUT should therefore be constructed for each velocity set.

### 3.9.3. Pull-streaming and Sentinel Handling

Streaming is implemented as a pull scheme,

$$f_i(\mathbf{r}, t + \Delta t) = \begin{cases} f_i^*(\text{src}(\mathbf{r}, i), t) & \text{if } \text{src}(\mathbf{r}, i) \neq 0, \\ \text{handled by boundary kernel} & \text{if } \text{src}(\mathbf{r}, i) = 0, \end{cases} \quad (3.60)$$

where each non-solid node reads the post-collision distributions of its upstream neighbours, rather than pushing its own outward. Boundary-crossing links are skipped by the streaming kernel via the sentinel

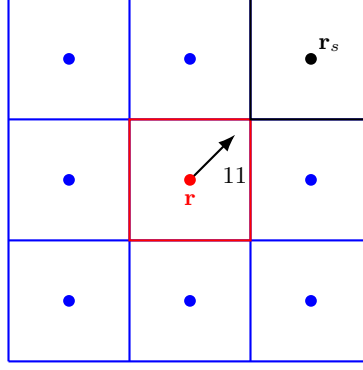


**Figure 3.11:** Schematic of a 3D lattice, projected on the  $x - z$  plane for clarity. In this projected plane, the fluid node at position  $\mathbf{r}$  (red) is surrounded by 7 fluid nodes (blue) and one solid node  $\mathbf{r}_s$  (black), separated from  $\mathbf{r}$  in the velocity direction  $\mathbf{c}_{11} = [1, 0, 1]^T$  as defined by the D3Q19 lattice velocity set (see Figure 3.1b). Therefore  $\mathbf{r}$  is classified as a boundary node in this scheme. However, in the D3Q7 scheme  $\mathbf{c}_{11}$  is not defined, so there is no streaming of particle distributions between  $\mathbf{r}$  and  $\mathbf{r}_s$ . Therefore  $\mathbf{r}$  is classified as a fluid node in the D3Q7 scheme.

value check and handled by the dedicated boundary kernel using Equation 3.57 with the stored  $q(\mathbf{r}, i)$ . In this way the geometric analysis is removed entirely from the time loop.

Similarly to all other data structures, the LUTs are laid out to facilitate efficient GPU access. Its arrays are stored in the same flattened form as the distribution functions, with the spatial indices as the fastest-varying index. This is done so that consecutive threads, which are mapped to consecutive lattice nodes, read consecutive LUT entries. The LUTs therefore benefit from the same memory coalescing effect described in Section 3.8.3. For geometry-static problems, the tables are built only once on the CPU and transferred to the GPU as `CuVector` type arrays before the time loop begin. Hence the construction cost is negligible relative to the cost of the simulation itself. Figure 3.12 shows a schematic of the LUT structure and the pull-streaming/sentinel mechanism.

A further consequence of the LUT approach is the removal of the ghost node padding usually employed in LBM implementations, as explained in Section 3.3. Since every boundary-crossing link is explicitly marked by the sentinel value, the streaming step never attempts to access a node outside the physical domain. Every stored node is therefore a genuine computational node, and the total node count is reduced from the padded value  $(N_X + 2)(N_Y + 2)(N_Z + 2)$  to  $N_X N_Y N_Z$ , with a corresponding reduction in both the distribution-array memory footprint and the number of threads launched per kernel. The relative additional overhead  $(N_{\text{padded}} - N_{\text{LUT}})/N_{\text{LUT}}$  shrinks as the grid is refined, so the proportional benefit of ghost-node elimination is largest for smaller domains.

| index               | class    | src                            | $q$      |
|---------------------|----------|--------------------------------|----------|
| $\vdots$            | $\vdots$ | $\vdots$                       | $\vdots$ |
| $(\mathbf{r}_f, i)$ | 0        | $(\mathbf{r}_b, i)$            | 0        |
| $(\mathbf{r}_b, i)$ | 2        | 0                              | 1/2      |
| $(\mathbf{r}_s, i)$ | 1        | 0                              | 0        |
| $\vdots$            | $\vdots$ | $\vdots$                       | $\vdots$ |
| $(\mathbf{r}_f, j)$ | 0        | $(\mathbf{r}_f + \Delta x, j)$ | 0        |
| $(\mathbf{r}_b, j)$ | 2        | $(\mathbf{r}_f, j)$            | 0        |
| $(\mathbf{r}_s, j)$ | 1        | 0                              | 0        |
| $\vdots$            | $\vdots$ | $\vdots$                       | $\vdots$ |

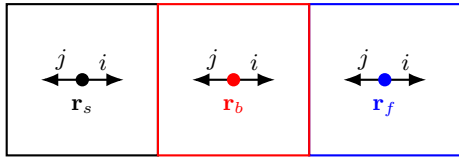


**Figure 3.12:** Left: Adjacent configuration of solid node  $\mathbf{r}_s$  (black), boundary node  $\mathbf{r}_b$  (red), and fluid node  $\mathbf{r}_f$  (blue). Right: Schematic of the lookup table used for the pull-streaming mechanism. For each index, the LUT contains the node class (fluid, solid, boundary), the index of the source node from which the distribution should be pulled, and (for boundary nodes) the fractional distance  $q$  to the boundary in lattice units.

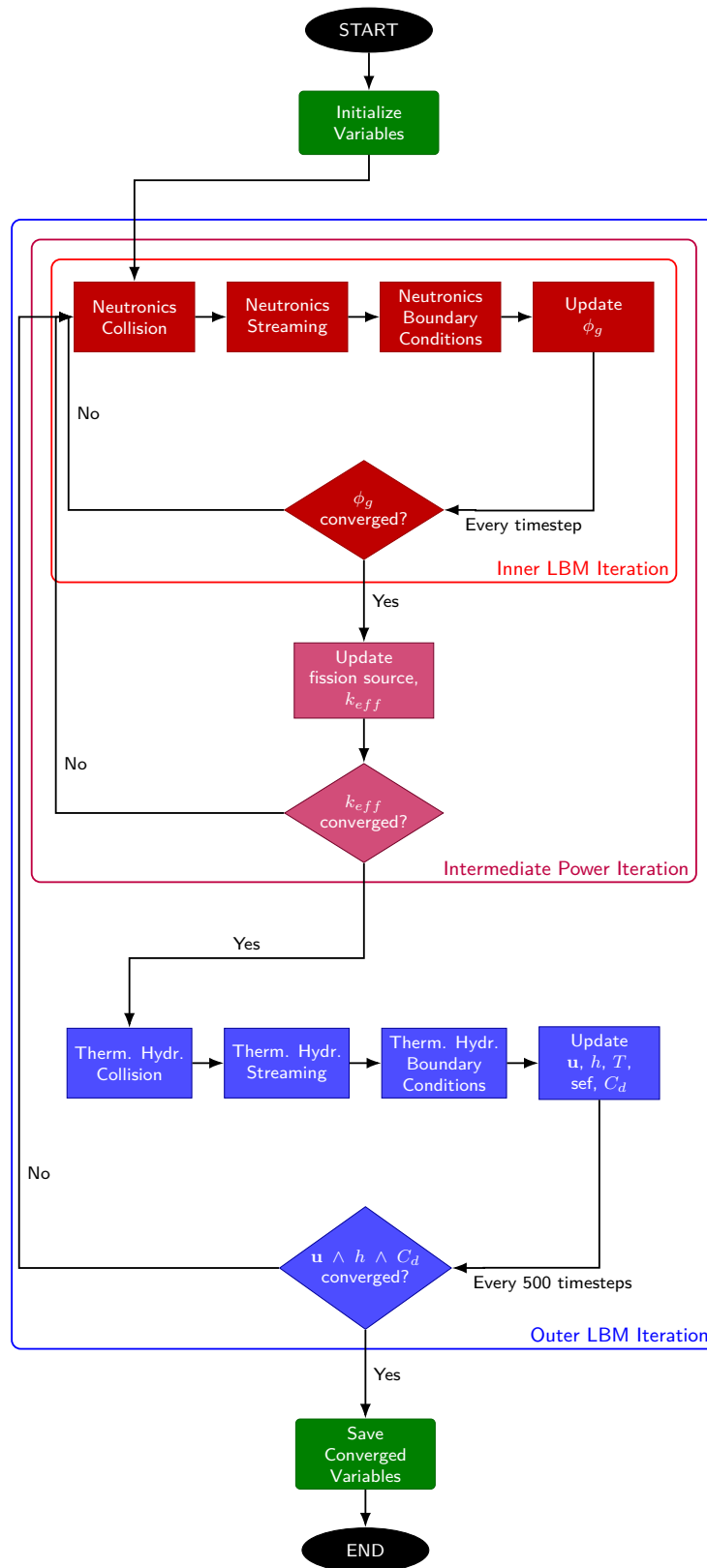


**Figure 3.13:** Solving scheme of the Filter-Matrix Lattice Boltzmann Method for coupled steady-state simulations. Note that source terms and other macroscopic variables (such as the salt expansion feedback or 'sef') are also calculated in the penultimate step of the outer LBM iteration (rightmost blue rectangle).

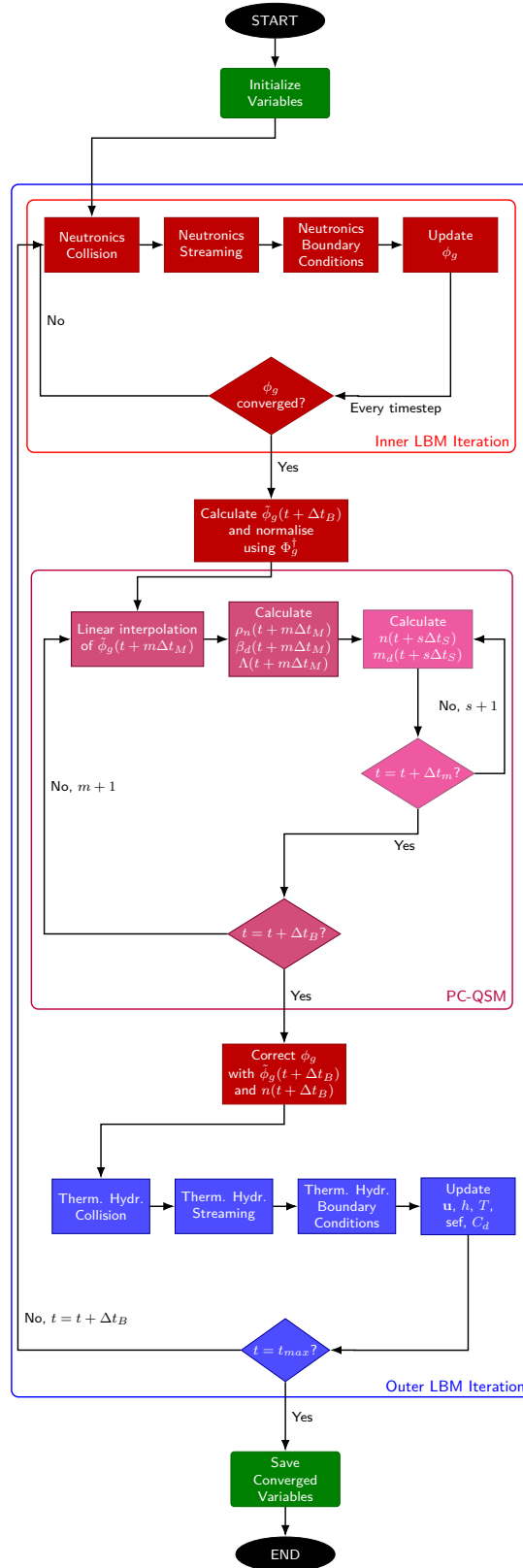


**Figure 3.14:** Solving scheme of the FM-LBM with the Predictor-Corrector Quasi-Static Method for transient neutronics. Note that in this research, the  $\Delta t_S$  loop is replaced by a matrix multiplication as explained in Section 3.7.

# 4

## Validation of Steady-State Multiphysics Models

In the absence of full-scale experimental data for molten salt fast reactors, validation of the multiphysics tool is limited to the comparison of simulation results against either the analytical solutions of simplified problems, or against the results of established code packages from existing literature applied to well-defined numerical benchmarks. One of these latter cases is the benchmark described by Tiberge et al. [60], which specifies a coupled neutronics-thermal-hydraulics problem in a simplified MSFR representation, and reports reference solutions from a range of participating code packages. This benchmark, hereafter named the 'Tiberge benchmark', is especially useful for validation work, as it is structured in various phases and steps in which the models of the different physical fields are gradually incorporated into the simulation. This makes it possible to analyse the effect of adding and coupling different physical phenomena, and to attribute any emerging source of error to a specific part of the overall multiphysics model.

In this chapter the application of the FM-LBM multiphysics tool developed in this work to *steady-state* simulations according to the Tiberge benchmark is presented and discussed, as transient cases are treated in the next chapter. In Section 4.1 a description of the benchmark is provided, along with an overview of the reference codes against which the results are compared. The results of single-physics simulations of momentum & heat transport, as well as the dynamics of the neutron flux & precursor concentrations are reported in Section 4.2. In Section 4.3 the results of simulations with coupled physical fields are presented. Heatmaps of the simulated observables in each step are shown in Appendix B. Finally, the tool's capabilities to simulate fluid dynamics in more complex geometries are assessed in Section 4.4.

### 4.1. Description of the Tiberge Benchmark

The benchmark features a square cavity of  $2\text{ m} \times 2\text{ m}$  filled with a molten salt at an initial temperature of  $T_0 = 900\text{ K}$ . A no-slip boundary condition with zero velocity is imposed on all walls except the top lid, which moves with velocity  $U_{lid}$  to the right. Furthermore, the walls are adiabatic, so homogeneous Neumann boundary conditions are imposed for the enthalpy, as well as for the concentration of delayed neutron precursors. The domain of the problem is shown in Figure 4.1. For the neutron flux, standard vacuum boundary conditions as described in Section 2.2.2 are imposed on the walls of this model reactor. The removal of heat from this setup is modelled using a volumetric heat sink:

$$q'''(\mathbf{r}) = \gamma_{th}(T_{ref} - T(\mathbf{r})), \quad (4.1)$$

where the reference temperature  $T_{ref} = T_0 = 900\text{ K}$ , and the volumetric heat transfer coefficient  $\gamma_{th}$  is uniform throughout the entire domain.

The modelled molten salt is a representation of  $\text{LiF-BeF}_2\text{-UF}_4$ . The molar composition of this mixture is shown in Table 4.1. Its fluid properties are considered constant with temperature and uniform in space, and are listed in Table 4.2. Other assumptions of the benchmark include incompressible and fully laminar flow, and buoyancy is modelled adopting the Boussinesq approximation, described in Section 2.1.1.

The implemented nuclear parameters were taken from the JEFF-3.1 library at  $T_{ref} = 900 \text{ K}$ . For six neutron groups and eight precursor families, the data was generated with Serpent. The complete set of multi-group neutronics and precursor data can be found in Appendix A as well as in [60]. While nuclear fission acts as the main source of heat production, the heat from fission products and their decay is neglected. Furthermore, Doppler broadening of the neutron absorption spectra due to increased temperatures is also omitted from the model. Therefore thermal variations only affect the neutron dynamics through the salt expansion feedback as explained in Section 2.3. As explained above, the benchmark is organised into a sequence of steps, each simulating a specific subset of the physical fields, summarised in Table 4.3.

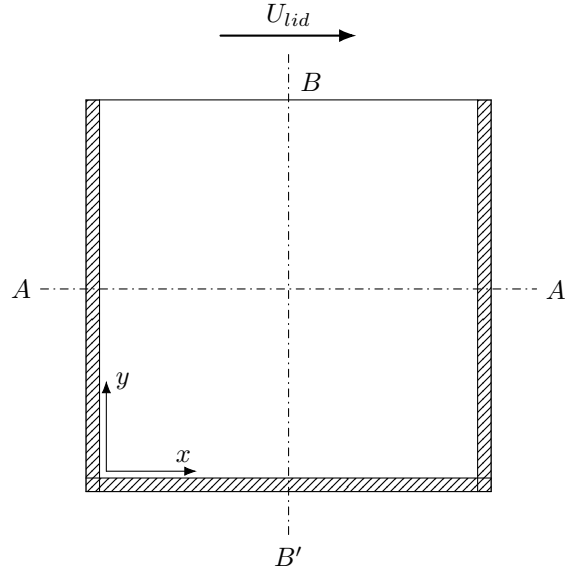


**Figure 4.1:** The domain of the Tiberge benchmark. The cavity is insulated, surrounded by vacuum, and filled with molten salt at an initial temperature of 900 K. Observables are pointwise compared along the  $AA'$  and  $BB'$  centrelines. Figure is inspired by [60].

**Table 4.1:** Fuel salt composition[60].

| Isotope             | ${}^6\text{Li}$ | ${}^7\text{Li}$ | ${}^9\text{Be}$ | ${}^{19}\text{F}$ | ${}^{235}\text{U}$ |
|---------------------|-----------------|-----------------|-----------------|-------------------|--------------------|
| Atomic fraction (%) | 2.114488        | 26.0836         | 14.0992         | 56.3969           | 1.30545            |

**Table 4.2:** Salt thermodynamic and transport properties.

| Property                      | Units                           | Value                |
|-------------------------------|---------------------------------|----------------------|
| (Reference) Density           | $\text{kg m}^3$                 | $2.0 \times 10^{-3}$ |
| Kinematic viscosity           | $\text{m}^2 \text{s}^{-1}$      | $2.5 \times 10^{-2}$ |
| Volumetric heat capacity      | $\text{J m}^{-3} \text{K}^{-1}$ | $6.15 \times 10^6$   |
| Thermal expansion coefficient | $\text{K}^{-1}$                 | $2.0 \times 10^{-4}$ |
| Prandtl number                | -                               | $3.075 \times 10^5$  |
| Schmidt number                | -                               | $2.0 \times 10^8$    |

#### 4.1.1. Reference Code Packages

The results of the FM-LBM multiphysics tool for the various benchmark cases are compared against a number of reference codes. These code packages were developed during the SAMOFAR project

| Step | Momentum | Temperature | Neutronics | Simulation Type |
|------|----------|-------------|------------|-----------------|
| 0.1  | □        |             |            | Steady-State    |
| 0.2  |          |             | □          | Steady-State    |
| 0.3  | ■        | □           | ■          | Steady-State    |
| 1.1  | ■        |             | □          | Steady-State    |
| 1.2  | ■        | □           | □          | Steady-State    |
| 1.3  | □        | □           | □          | Steady-State    |
| 1.4  | □        | □           | □          | Steady-State    |
| 2.1  | □        | □           | □          | Transient       |

**Table 4.3:** Tiberger benchmark steps. A white square (□) denotes a field that is actively solved in the step, while a black square (■) denotes a field that is prescribed or held fixed from a previous step.

by four institutions: Centre national de la recherche scientifique-Grenoble[74] (CNRS), Politecnico di Milano[75] (PoliMi), Paul Scherrer Institute[15] (PSI), and Delft University of Technology[16] (TUD). Table 4.4 gives a brief overview of the different reference codes, including the implemented neutronics scheme and spatial resolution. In the different steps of the Tiberger benchmark, the results of certain observables obtained from our simulations will be compared against the reference codes along the cavity centrelines  $AA'$  and  $BB'$ , also depicted in Figure 4.1.

**Table 4.4:** Details of the reference code packages.[74, 75, 15, 16]

| Code Package | Grid/Mesh Size   | Neutronics Model                          |
|--------------|------------------|-------------------------------------------|
| CNRS         | $200 \times 200$ | $SP_1$ & $SP_3$<br>neutron transport      |
| PoliMi       | $400 \times 400$ | Multigroup<br>neutron diffusion           |
| PSI          | $200 \times 200$ | Multigroup<br>neutron diffusion           |
| TUD          | $50 \times 50$   | Multigroup Boltzmann<br>nuclear transport |

### 4.1.2. Error Quantification

To obtain a quantified comparison of the simulations performed in this research with the results reported by the reference codes, the error of the physical observables along the  $AA'$  and  $BB'$  centrelines is measured for the different steps. Similarly to how the convergence of steady-state simulations is assessed, this error is calculated using the  $L_2$ -norm:

$$\varepsilon_Q = \sqrt{\frac{\sum_{i=1}^{N_p} (Q_S(\mathbf{r}_i) - Q_R(\mathbf{r}_i))^2}{\sum_{i=1}^{N_p} Q_R^2(\mathbf{r}_i)}}, \quad (4.2)$$

where  $\varepsilon_Q$  denotes the error in the observable  $Q$ ,  $Q_S(\mathbf{r}_i)$  the value for that observable at location  $\mathbf{r}_i$  obtained in the current work's simulation, and  $Q_R(\mathbf{r}_i)$  the ensemble average value for that observable reported by the reference codes.  $N_p$  is the number of points at which  $Q$  is evaluated; for centreline evaluations this is just the number of lattice points along the dimensions, i.e.  $N_X$  or  $N_Y$ .

## 4.2. Single-Physics Simulations

### 4.2.1. Step 0.1: Momentum

In the first phase of the benchmark, steady-state simulations are carried out without any coupling between the different physics fields. In the first step of this phase, the solution of a steady-state incompressible flow is studied. As the velocity field is driven by the moving upper lid, no external forces or coupling is needed as a momentum source. The simulation parameters chosen for this step are found in Table 4.5. Note that in this table, as well as the others in this chapter, the values of physical variables

are expressed in physical units. The procedure described in Section 3.4 can be used to convert the physical values into lattice unit values, i.e. what is actually used in the FM-LBM code.

**Table 4.5:** Simulation parameters for Step 0.1 of the Tiberge benchmark.  $N_X$  and  $N_Y$  denote the number of lattice points.

| Parameter  | Physical Unit     | Physical Value     |
|------------|-------------------|--------------------|
| $\Delta t$ | s                 | $1 \times 10^{-3}$ |
| $N_X$      | -                 | 200                |
| $N_Y$      | -                 | 200                |
| $Re$       | -                 | 40                 |
| $U_{lid}$  | $\text{m s}^{-1}$ | 0.5                |

This step is primarily aimed at testing whether the FM-LBM implementation of the Navier-Stokes Equations is capable of obtaining a correct velocity field  $\mathbf{u}$ . As such, the observables in this step are the vertical and horizontal velocity ( $u_x$  and  $u_y$ ) along the centrelines. As later phases of the benchmark use these results as fixed or initial conditions for the velocity field, the accurate modelling of this step is crucial to obtain consistent multiphysics coupling. Figures 4.2 and 4.3 show the results of the converged velocity field. The simulations show strong agreement with the data of the benchmark cases. In this step, the PSI code reports a lower maximum vertical velocity along the  $BB'$  centreline, as seen in Figure 4.3. It must be stated that this smaller peak could be reproduced by the FM-LBM algorithm by decreasing the model's grid size from  $200 \times 200$  to  $50 \times 50$ , while keeping  $\Delta t = 1 \times 10^{-3}$  s.

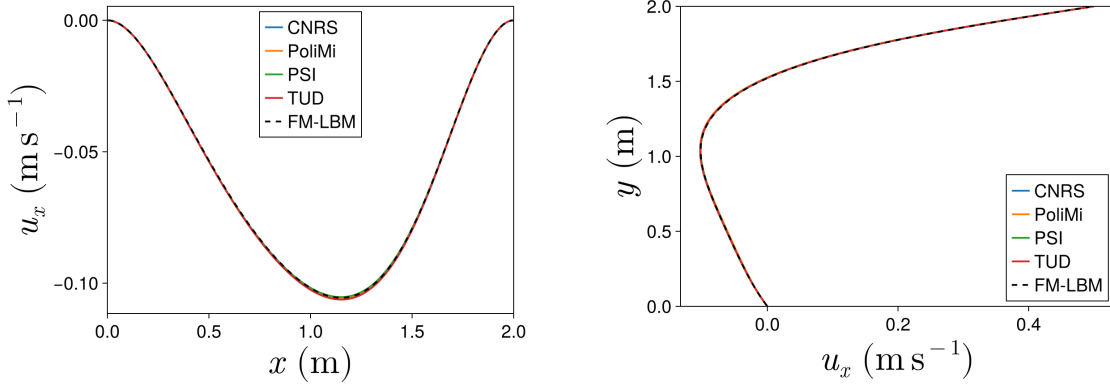


**Figure 4.2:** The horizontal velocity component for Step 0.1 of the Tiberge benchmark case along the horizontal  $AA'$  (left) and vertical  $BB'$  (right) centrelines. The simulation results are shown alongside the data from the reference codes.

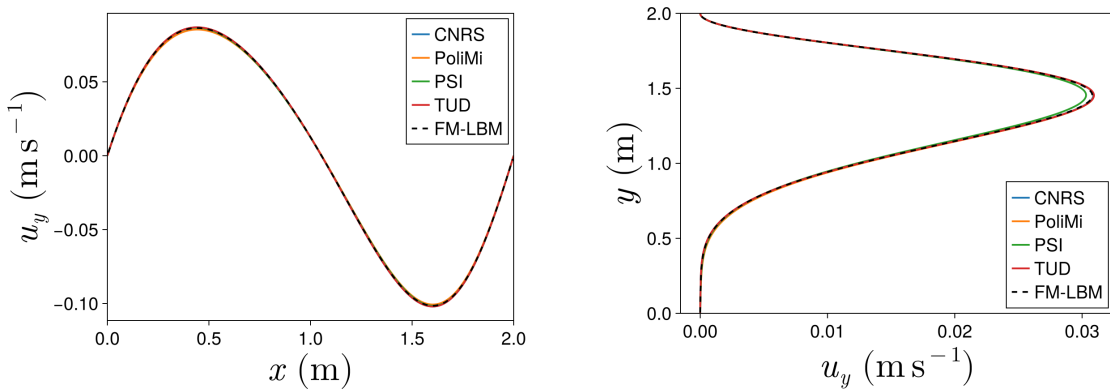


**Figure 4.3:** The vertical velocity component for Step 0.1 of the Tiberge benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

To assess the spatial accuracy of the implementation, the 0.1 step was simulated for increasing grid resolutions  $N_X$ . At first, the time step was held constant at  $\Delta t = 1 \times 10^{-3}$  s for all runs. The resulting

velocity discrepancies along the centrelines are shown in Figure 4.4a. As can be seen in the figure, the error initially decreases with increasing spatial resolution, but begins to rise again beyond approximately  $N_X = 300$ . This behaviour is a direct consequence of fixing the time step  $\Delta t$  while refining the lattice spacing  $\Delta x$ . As explained by Krüger[18], the Lattice Boltzmann Method is second-order accurate in both space and time. This means the spatial discretisation error scales like  $\Delta x^2$  and the time discretisation error scales like  $\Delta t^2$ . However, the LBM applied to the incompressible Navier-Stokes equations also carries a compressibility error of order  $\mathcal{O}(Ma^2)$ , with  $Ma$  being the lattice Mach number:

$$Ma = \frac{U}{c_s} = \frac{U\sqrt{3}\Delta t}{\Delta x} = \frac{U\sqrt{3}\Delta t N_X}{L}. \quad (4.3)$$

As  $N_X$  increases at fixed  $\Delta t$ , the Mach number grows linearly with the spatial resolution. The total error therefore behaves as:

$$\varepsilon \propto c_1 \Delta x^2 + c_2 \Delta t^2 + c_3 Ma^2. \quad (4.4)$$

Therefore, increasing the spatial resolution alone does not necessarily reduce the error, because the compressibility error will grow and dominate eventually. This causes the behaviour seen in Figure 4.4a. At low  $N_X$ , the spatial discretisation error dominates, and increasing the lattice resolution reduces the total error. At intermediate  $N_X$  the curve flattens, while at high  $N_X$  the growing incompressibility error dominates. Therefore one must scale both  $\Delta x$  and  $\Delta t$  accordingly to control the error. As stated by [18], the preferred scaling for incompressible flow is the diffusive scaling, where  $\Delta t$  scales with  $\Delta x^2$ . Under this scaling, the Mach number scales as  $Ma \propto \Delta t / \Delta x \propto \Delta x$ , so the compressibility error  $\mathcal{O}(Ma^2) \propto \Delta x^2$  decreases at the same rate as the spatial discretisation error. However, with this scaling, the Lattice Boltzmann Method is only first-order accurate in time.

Figure 4.4b shows the accuracy of the FM-LBM implementation for increasing grid refinement, now using the diffusive scaling for the spatial and temporal resolutions. As expected, the earlier upswing in error for increasing  $N_X$  has largely disappeared. However, the vertical velocity component along the vertical centreline still shows a minimum error at around  $N_X = 200$ , with slowly increasing error for increasing grid size. Due to the small error decrease of the other components with increasing grid size beyond this point, and to reduce computing time and memory usage, it was decided to simulate the rest of the benchmark cases on a  $200 \times 200$  grid.

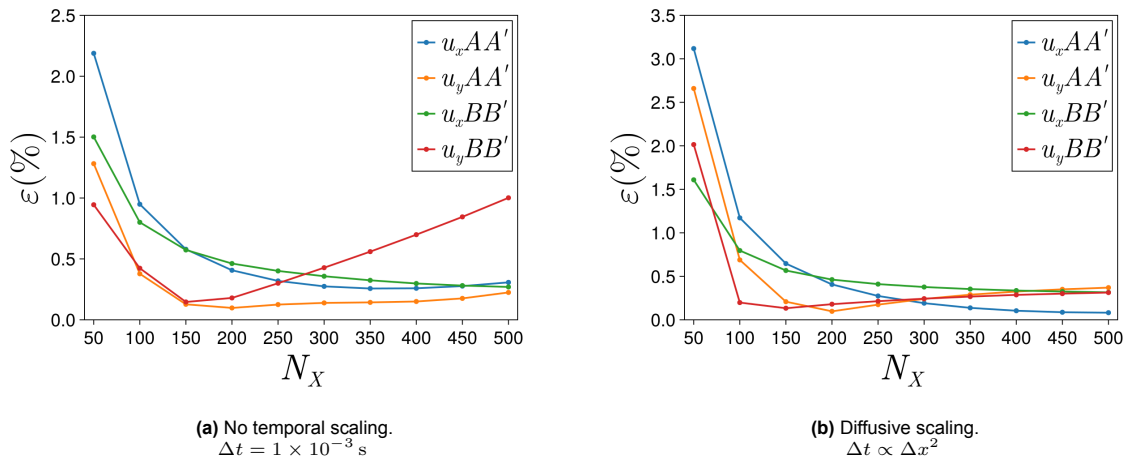


**Figure 4.4:** Discrepancy of the horizontal and vertical velocity components for Step 0.1 of the Tiberge benchmark case along the horizontal and vertical centrelines. For increasing spatial resolution, Figure 4.4a shows the discrepancy without temporal scaling, while diffusive scaling is employed in Figure 4.4b.

#### 4.2.2. Step 0.2: Neutronics

In the second step of the Tiberge benchmark, the neutronics criticality eigenvalue problem is studied with non-circulating and isothermal fuel. The reactor power is set to normalise the neutron flux. The

simulation parameters chosen for this step are found in Table 4.6. As explained in Section 3.5, the dynamics of the neutron flux occur at a much faster rate than the other fields, and therefore the timestep for the neutronics FM-LBM ( $\Delta t_N$ ) is considerably smaller than that for the thermal hydraulics ( $\Delta t_{TH}$ ).

**Table 4.6:** Simulation parameters for Step 0.2 of the Tiberge benchmark.

| Parameter       | Physical Unit | Physical Value       |
|-----------------|---------------|----------------------|
| $\Delta t_{TH}$ | s             | $1.0 \times 10^{-3}$ |
| $\Delta t_N$    | s             | $1.0 \times 10^{-9}$ |
| $N_X$           | -             | 200                  |
| $N_Y$           | -             | 200                  |
| $P$             | W             | $1.0 \times 10^9$    |
| $T$             | K             | 900                  |

The aim is to verify the neutronics solution of the model in simple, static-fuel conditions. The observables in this step are the fission rate density ( $\int_E \Sigma_f \phi dE$ ) along the centrelines, as well as the reactivity  $\rho_n$ . Minor differences in this latter value compared to the reference results might arise from the different neutronics models and approximations adopted, and are considered acceptable. Table 4.7 reports the obtained reactivity, alongside the values reported by the earlier LBM studies from the research group and the reference codes, while Figure 4.5 shows the results of the fission rate density. Again, it can be seen that the FM-LBM simulation is in very good agreement with the data of the benchmark cases. These all sit within the range of 350 pcm to 450 pcm, with the exception of the TUD codes. Their (slightly) higher reactivities can be attributed to the higher order discretisations of the neutron transport modelling, as opposed to the neutron diffusion models employed by PoliMi and PSI. With that said, the FM-LBM reactivity of 456.9 pcm sits closer to the aforementioned range than the values obtained by the other LBM studies.

| Code                 | $\rho_n$ (pcm) |
|----------------------|----------------|
| CNRS-SP <sub>1</sub> | 411.3          |
| CNRS-SP <sub>3</sub> | 353.7          |
| PoliMi               | 421.2          |
| PSI                  | 411.7          |
| TUD-S <sub>2</sub>   | 482.6          |
| TUD-S <sub>6</sub>   | 578.1          |
| LBM-Polderman        | 520.6          |
| LBM-Entes            | 578.1          |
| LBM-Pijls            | 485.5          |
| LBM-Blanken          | 456.9          |

**Table 4.7:** Simulation results for the reactivity for step 0.2 of the Tiberge benchmark case, alongside results from earlier LBM studies and the reference codes.

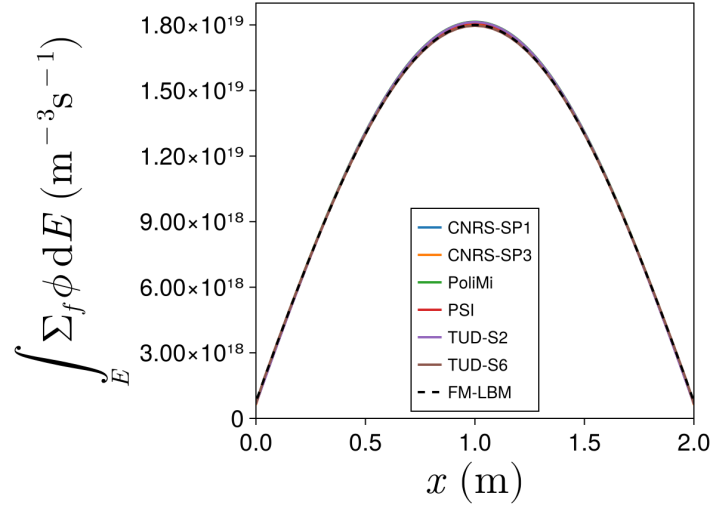


**Figure 4.5:** The fission rate density for Step 0.2 of the Tiberge benchmark case along the horizontal  $AA'$  centreline. As the domain is symmetric in this step, the results along the vertical centreline are identical. The simulation results are shown alongside data from reference codes.

Before proceeding to the enthalpy simulations, the choice of extrapolated boundary length  $\tilde{x}$  and its effect on the simulation results is studied. As explained in Section 2.2.2, this artificial length is used in the formulation of the vacuum boundary condition for neutron diffusion, which describes that  $\phi$  should be zero at the extrapolated boundary. For linear extrapolation, diffusion theory states that  $\tilde{x} = x_{wall} + 2D_n$ , while more advanced transport calculations of the extrapolated boundary indicate that one should choose  $\tilde{x} = x_{wall} + 2.1312D_n$ [52]. This latter value has been used in the previous studies that employed LBM for neutron diffusion modelling[42, 43].

The fission rate density along the horizontal centreline for both choices of  $\tilde{x}$  are presented in Figure

4.6, focused on the reactor centre (1 m, 1 m). As can be seen, the fission rate density for the model with  $\tilde{x} = 2D_n$  is slightly higher near the centre than for the model with  $\tilde{x} = 2.1312D_n$ , and agrees a little better with the reference codes. In reality, the discrepancy of both instances as calculated through Equation 4.2 amounts to just 0.37% and 0.68% respectively. The difference can be explained by the constant normalisation of the reactor power. For the case where  $\tilde{x} = 2D_n$ , the extrapolated boundary sits a little closer to the reactor wall, and so the interpolated value for the scalar flux at the wall is relatively low. This can best be understood by viewing Figure 3.5. Because the reactor output power  $P = E_{fission} \sum_g \Sigma_{f,g} \phi_g$  is held constant, this governs the maximum value of the scalar flux at the centre of the reactor. On the other hand, for  $\tilde{x} = 2.1312D_n$ , the extrapolated boundary sits a little further from the reactor wall, and so the interpolated value  $\phi_{wall}$  is relatively high. Because of the power normalisation the slightly higher fission rate density near the wall results in a slightly lower peak density at the reactor centre. Because the reference code packages model neutron transport or diffusion differently, there is no obvious choice which value for the extrapolated boundary length to use. However, as shown in Figure 4.6 and from the calculated discrepancies it can be concluded that the simulation with  $\tilde{x} = 2D_n$  agrees slightly better with the reference data. Therefore it was decided to use this value for the rest of the neutron diffusion simulations.

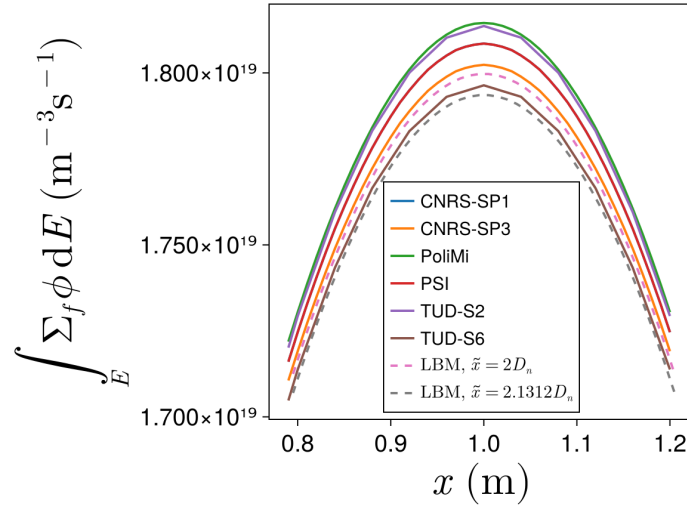


**Figure 4.6:** The fission rate density for Step 0.2 of the Tiberga benchmark case along the horizontal centreline, for both choices of extrapolated boundary length  $\tilde{x}$ , alongside data from the reference codes.

#### 4.2.3. Step 0.3: Temperature

In the third step of the Tiberga benchmark, the passive scalar transport of the codes is studied independently from the solution of the fluid flow and the neutronics. This includes the convective and diffusive behaviour of the heat transport. The flow field from Step 0.1 is set, and the neutron flux solution from Step 0.2 is used for the heat source distribution. The simulation parameters for this step are found in Table 4.8. Here the value for the thermal diffusivity  $\alpha_{th}$  was obtained by dividing the kinematic viscosity  $\nu$  by the Prandtl number of 1200.

The Tiberga benchmark describes a very high Prandtl number of  $3.075 \times 10^5$ , along with a Schmidt number of  $2.0 \times 10^8$ , but these values were reportedly used to highlight the impact of possible errors in physics coupling strategies. Realistic MSFR cores operate within a  $Pr$ -range of 1 to 35[8]. Simulating mass and thermal diffusion with these high dimensionless numbers would require a very high lattice viscosity  $\nu^*$ , which would have to be realised in a very high spatial resolution to keep the Reynolds number constant. This would immensely increase computational costs. Therefore, a Prandtl number of 1200 is chosen, which is still much higher than typical values, but more manageable in the context of computer simulations. The justification for this reduction lies in the reflection of the physical meaning of high  $Pr$  (and  $Sc$ ) numbers, as described in Table 2.1. A high Prandtl number corresponds to thermal diffusivity having a much lower effect than momentum diffusivity, and therefore convection has a dominant effect on heat distribution over diffusion. In other words, diffusion due to temperature gradi-

ents can be assumed negligible. However, due to the applied volumetric heat sink it is expected that temperature gradients will be minimal. Therefore thermal convection would stay dominant over thermal diffusion, even if the Prandtl number was significantly reduced. Moreover, to obtain stable results for such Prandtl numbers, it was found in this research that the temporal resolution  $\Delta t_{TH}$  had to be reduced from  $1.0 \times 10^{-3}$  s to  $2.5 \times 10^{-4}$  s.

To assess the accuracy of the FM-LBM implementation as a function of  $Pr$ , the 0.3 step was simulated for increasing Prandtl numbers. The results of these simulations are shown in Figure 4.7. It can be seen that even for low Prandtl numbers, the discrepancy is less than 1%, and the error initially decreases slowly for increasing Prandtl numbers. However, while running the cases for  $Pr > 1500$ , it was found that the simulations became very unstable using the simulation parameters described in Table 4.8, and therefore the convergence threshold as defined in Equation 3.43 had to be made less strict in order to retrieve output data. This resulted in higher discrepancies as can be seen in Figure 4.7.

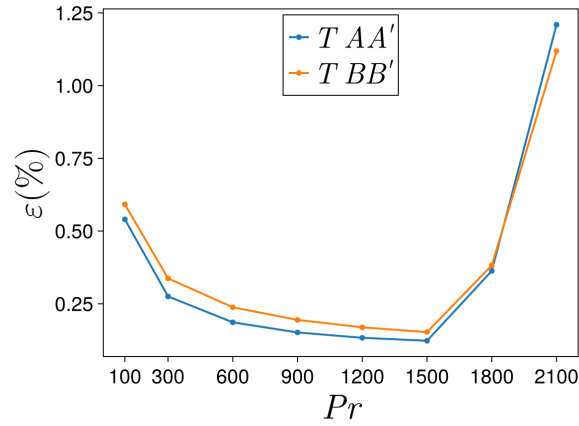


**Figure 4.7:** Discrepancy of the temperature for Step 0.3 of the benchmark case along the horizontal and vertical centrelines as a function of the dimensionless Prandtl number.

To further assess the effect of reduced Prandtl numbers on simulation errors, Figure 4.8 shows the discrepancy at each evaluation point along the reactor centrelines for different Prandtl values. For clarity the unstable results for  $Pr = 1800$  and  $2100$  are omitted from the figure. As expected, the error is greatest at the domain boundaries, as the temperature (or in the current implementation, the enthalpy) gradients are largest near the boundaries due to the Neumann boundary conditions. Moreover, it can be seen that the discrepancy is reduced for increasing Prandtl numbers, and this reduction is greatest near the boundaries and minimal near the centre of the reactor.

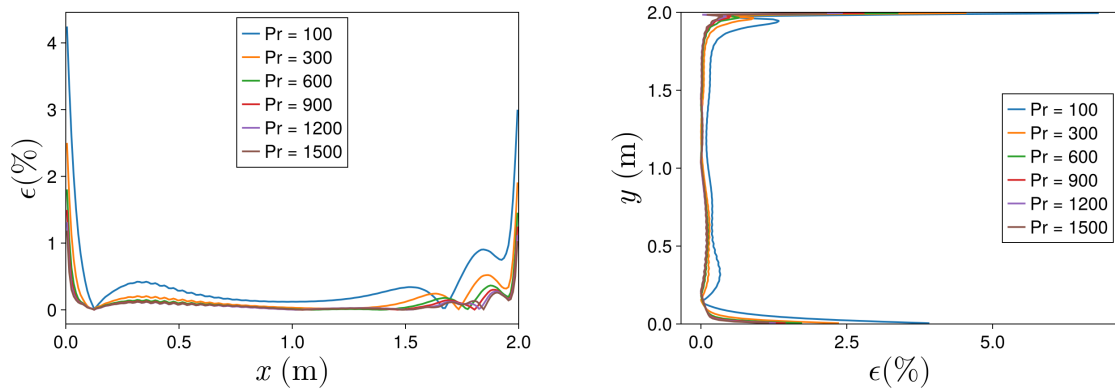


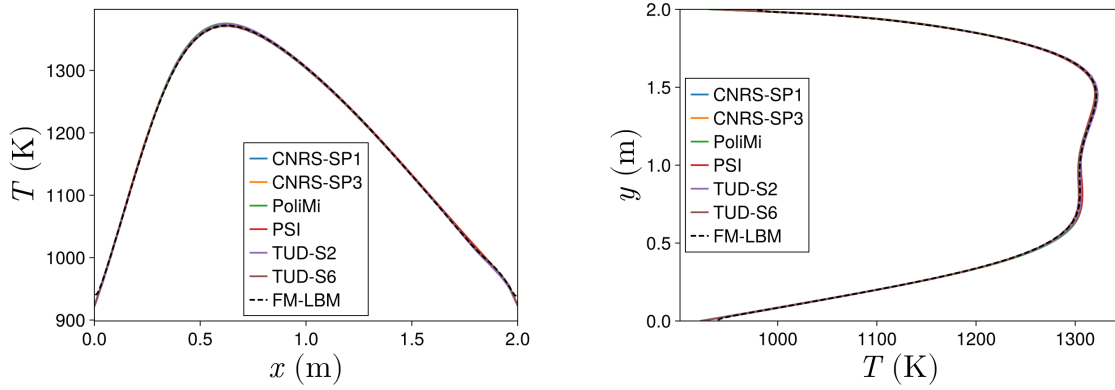
**Figure 4.8:** Pointwise discrepancy of the temperature for Step 0.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines.

The observables in this step are the temperature distributions along the centrelines. Figure 4.9 shows the obtained distributions. Again, the current FM-LBM implementation demonstrates good alignment

**Table 4.8:** Simulation parameters for Step 0.3 of the Tiberge benchmark.

| Parameter       | Physical Unit                   | Physical Value        |
|-----------------|---------------------------------|-----------------------|
| $\Delta t_{TH}$ | s                               | $2.5 \times 10^{-4}$  |
| $Pr$            | -                               | 1200                  |
| $\gamma_{th}$   | $\text{W m}^{-3} \text{K}^{-1}$ | $6.15 \times 10^{-4}$ |
| $T_{ref}$       | K                               | 900                   |
| $\alpha_{th}$   | $\text{m}^2 \text{s}^{-1}$      | $2.08 \times 10^{-5}$ |

with the reference codes, obtaining an average discrepancy of 0.13% and 0.17% for the temperature along the horizontal and vertical centreline respectively. In light of the aforementioned results, it was decided that the rest of the enthalpy simulations would use  $Pr = 1200$ , as a good trade-off between physical accuracy, numerical stability, and computation time.

**Figure 4.9:** The temperature for Step 0.3 of the Tiberge benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

#### 4.2.4. Performance

To assess the computational performance of the GPU-accelerated FM-LBM tool, the number of lattice updates per second (LUPS) was measured for the single-physics simulations. Due to the large size of these numbers, it is often more convenient to express the metric in millions of lattice updates per second (MLUPS), which is defined as

$$\text{MLUPS} = \frac{N_{Grid} N_T}{T} \times 10^{-6}, \quad (4.5)$$

where  $N_{Grid}$  denotes the total number of grid points,  $N_T$  the number of LBM time iterations performed in the simulation, and  $T$  the total simulation time. In this context, a lattice update comprises the total execution of all LBM simulation steps (collision, streaming, handling of boundary conditions, updating macroscopic variables) for a single LBM time iteration at a single lattice node.

Figures 4.10 and 4.11 show the computational performance of the previous steps for increasing grid sizes. All simulations were performed on the DelftBlue supercomputer[76] using a single NVIDIA A-100 GPU and using double-precision floating point numbers. As the FM-LBM implementation developed by Pijls most closely resembles the current work (with LBM modelling for enthalpy and neutron diffusion), the performances of both codes are shown together for comparison[43].

All three steps show an increase in MLUPS as the grid size increases. This was expected, as a highly parallelised algorithm executed on the GPU should not be limited by grid size. From a certain point on, however, the MLUPS starts to flatten out, limited by the memory access speed. It can also be seen that for a given grid size, the number of lattice updates varies greatly between the steps. For  $N_x = 500$ , Step 0.1 reports a performance of 283.41, Step 0.2 80.62, and Step 0.3 1166.30. This is another suggestion that the algorithm performance is limited by memory limitations, as the enthalpy simulation is the least

memory intensive (1 distribution array with 7 velocity directions), while the neutronics simulation is the most memory intensive (14 distribution arrays, one for each of the neutron groups and precursor families, with 7 velocity directions). This last finding might be surprising, as it was stated in Chapter 3 that the collision, streaming, and boundary kernels were also parallelised across the different neutron and precursor groups. However, as the computation of some macroscopic variables depend on the values of multiple groups (such as the delayed neutron source), the kernel for this part of the algorithm could not be parallelised across groups. Furthermore, the convergence kernel in the inner iteration for the scalar neutron flux  $\phi_g$  is launched every (small) time step, while the convergence for the thermal hydraulics quantities is only checked every 500 time steps. Apart from the LBM iterations, Step 0.2 also employs a power iteration method to obtain the multiplication factor  $k_{eff}$ , the efficiency of which is also incorporated in the performance calculation for this step. These differences explain the varying performances of the scalar field simulations.

Comparing the results with the performances reported by Pijls, it can be seen that the current FM-LBM greatly outperforms the previous code, especially at higher grid sizes. The increased performances in terms of MLUPS vary from 500% for the neutronics simulation to nearly 1100% for the enthalpy simulation. It appears that the use of the pull-scheme lookup table, introduced in Section 3.8.3, as well as the use of `StaticArrays` for the Filter-Matrix multiplication, substantially improve the runtime efficiency of the multiphysics code. When compared to other studies optimising the LBM algorithm, the current performance is in a reasonable range. For example, Tran et al.[77, 78] report simulation speeds up to 1200 MLUPS. This value is very close to the maximum performance obtained in the (D3Q7) enthalpy simulation for  $N_X = 500$  of 1166.30 MLUPS. However, their studies modelled a velocity field using a D3Q19 scheme, with the simpler BGK operator, as opposed to the current Filter-Matrix implementation. Furthermore, their simulations were executed on a different piece of hardware, namely the NVIDIA Tesla K20 GPU.

The MLUPS metric counts an entire lattice update as a single operation, irrespective of how many distribution arrays that update affects. This slightly complicates a direct comparison between the three single-physics steps, as each simulation carries a different number of lattice directions per node. For example, a single lattice update of the enthalpy field involves less computational work than one of the multi-group neutron and precursor fields, even though both count as one update under the MLUPS metric. A different measure of the raw work performed by the GPU per unit time is the number of *distribution function* or *direction* updates per second, expressed as

$$\text{MDUPS} = \text{MLUPS} \cdot N_Q \cdot N_{fields}, \quad (4.6)$$

where  $N_Q$  denotes the number of discrete velocity directions, and  $N_{fields}$  the number of fields whose distributions are advanced in one lattice update.

The MDUPS results for the three single-physics steps are shown in Figure 4.12, together with the values obtained by converting the MLUPS data of Pijls[43] on the same basis. The three curves are now closer together than in MLUPS terms, which confirms that the large MLUPS gap between the steps was caused mainly by the different number of distribution arrays, rather than by a substantial difference in solver efficiency. Step 0.3 still reaches the highest performance of over 8000 MDUPS, with Step 0.2 close behind. The apparent improvement of the neutronics solver is due to the parallelisation of the LBM's kernels across neutron groups and precursor families, as discussed earlier in this section. This is especially noticeable for low grid sizes. The remaining performance difference between the enthalpy and neutronics solvers is likely due to the costly multi-group calculation of neutron & precursor source terms, as well as the additional power iteration for  $k_{eff}$ . On the other hand, the lower performance of the momentum balance solver in Step 0.1 ( $\approx 5400$  MDUPS) can be explained by the high cost to update 19 distribution functions on a node, involving two multiplications of a  $19 \times 19$  filter matrix with a solution vector of length 19 in the collision step, compared to the  $7 \times 7$  matrix-vector multiplications for the scalar field updates. The MDUPS unit recalculation therefore does not fully eliminate the multi-field, multi-direction effect on the FM-LBM tool's performance. While the MLUPS results measure how rapidly the solver advances the lattice itself, the MDUPS results better reflect the theoretical complexity of the physical equations being solved on each field.

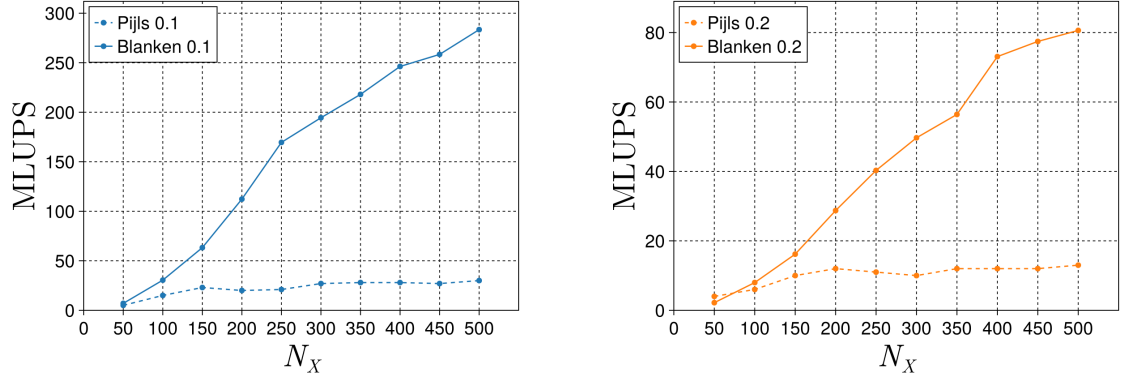


**Figure 4.10:** The computational performance of the GPU-accelerated FM-LBM algorithm for Steps 0.1 (left) and 0.2 (right) of the Tiberga benchmark case. The results are shown alongside the performance reported by Pijls[43]. Here,  $N_X$  is the number of grid points along the  $x$ -axis, since the simulations take place on a 2-dimensional grid  $N_{Grid} = N_X^2$ .

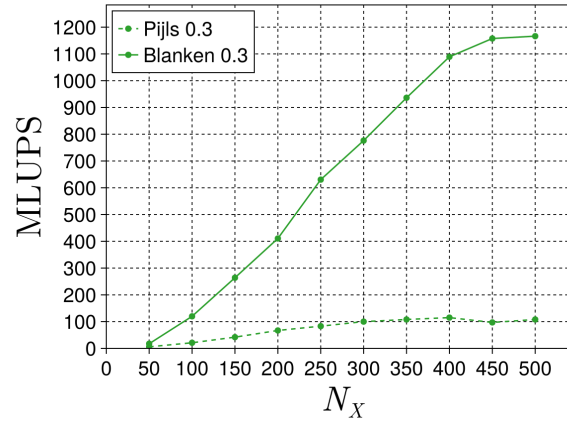


**Figure 4.11:** The computational performance of the GPU-accelerated FM-LBM algorithm for Step 0.3 of the Tiberga benchmark case. The results are shown alongside the performance reported by Pijls[43].

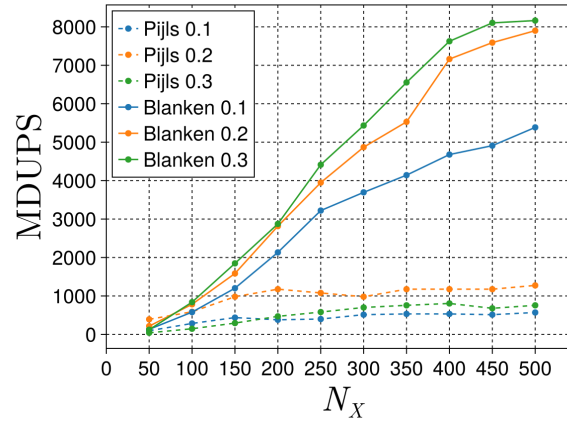


**Figure 4.12:** The computational performance of the GPU-accelerated FM-LBM algorithm for the single-physics steps of the Tiberga benchmark case, expressed in Million Direction Updates per Second (MDUPS). The results are shown alongside the performance reported by Pijls[43].

## 4.3. Coupled Simulations

### 4.3.1. Step 1.1: Circulating Fuel

In the second phase of the benchmark, steady-state solutions are found while gradually coupling the several physics phenomena characterising the molten salt fast system. In the first step of this phase, the neutronics criticality eigenvalue problem is solved in the presence of fixed fuel motion, namely the velocity field obtained in Step 0.1. Similarly to the stationary Step 0.2, the fuel temperature is considered uniform, and the reactor power is set to normalise the neutron flux. The simulation parameters chosen for this step are found in Table 4.9. It must be noted that, analogously to the Prandtl number for heat transport, the Schmidt number has been lowered to  $Sc = 1200$  as a trade-off between physical accuracy, numerical stability, and computation time.

**Table 4.9:** Simulation parameters for Step 1.1 of the Tiberge benchmark. Note that  $D_p$  denotes the mass diffusivity of the delayed neutron precursors.

| Parameter       | Physical Unit | Physical Value        |
|-----------------|---------------|-----------------------|
| $\Delta t_{TH}$ | s             | $2.5 \times 10^{-4}$  |
| $\Delta t_N$    | s             | $1.0 \times 10^{-9}$  |
| $Sc$            | -             | 1200                  |
| $D_p$           | $m^2 s^{-1}$  | $2.08 \times 10^{-5}$ |
| $P$             | W             | $1.0 \times 10^9$     |
| $T$             | K             | 900                   |

The aim of this step is to assess a correct evaluation of the effects of the fluid flow on neutronics. The observables in this step are the delayed neutron source ( $\sum_d \lambda_d C_d$ ) along the centrelines. Furthermore, to quantify the impact of precursor transport compared to the stationary case, the difference in reactivity  $\rho_n$  to the result of Step 0.2 is calculated. Figure 4.13 shows the results of the delayed neutron source, while Table 4.10 reports the obtained reactivity difference, alongside the values reported by the earlier LBM studies and the reference codes. The obtained delayed source shows strong agreement with the data of the benchmark cases. The average discrepancy of the delayed neutron source amounts to 0.8% and 0.5% along the horizontal and vertical centrelines respectively. With exception of the TUD- $S_6$  transport code, all reactivities reported by the benchmark reference codes sit within the range of 290 pcm to 420 pcm, as does the reactivity obtained in the FM-LBM implementation of this research. This is in contrast to the other LBM studies in which higher reactivities are reported for this step. The 1.1-0.2 difference of  $-60.0$  pcm is also in line with the values obtained by the reference codes and the results of Polderman and Entes. It must be noted that Pijls' value is farther off from the reference values, which could suggest that the current FM-LBM implementation is at least equally good, if not an improvement over his two-domain FM-LBM algorithm in this particular case. Either way, it can be concluded that the current implementation can accurately reproduce the effect of the precursor transport on the effective multiplication factor.

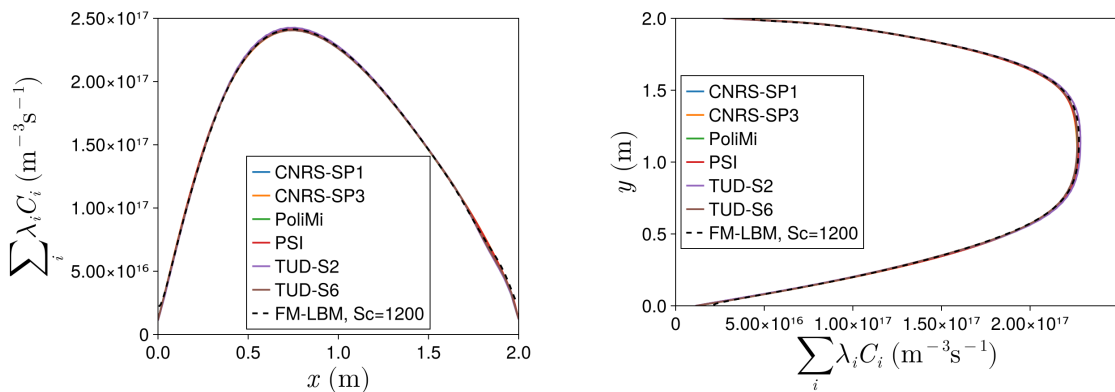


**Figure 4.13:** The delayed neutron source for Step 1.1 of the Tiberge benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

| Code          | $k_{eff}$  | $\rho_n$ (pcm) | $\rho_{n,1.1} - \rho_{n,0.2}$ (pcm) |
|---------------|------------|----------------|-------------------------------------|
| CNRS- $SP_1$  | 1.00350021 | 348.8          | -62.5                               |
| CNRS- $SP_3$  | 1.00291950 | 291.1          | -62.6                               |
| PoliMi        | 1.00360495 | 359.2          | -62.0                               |
| PSI           | 1.00349920 | 348.7          | -63.0                               |
| TUD- $S_2$    | 1.00422377 | 420.6          | -62.0                               |
| TUD- $S_6$    | 1.00520091 | 517.4          | -60.7                               |
| LBM-Polderman | 1.00457311 | 455.2          | -65.3                               |
| LBM-Entes     | 1.00516126 | 513.5          | -64.7                               |
| LBM-Pijls     | 1.00429923 | 428.1          | -57.4                               |
| LBM-Blanken   | 1.00398445 | 396.8          | -60.0                               |

**Table 4.10:** Simulation results for the reactivity difference between Steps 1.1 and 0.2 of the Tiberge benchmark case, alongside results from earlier LBM studies and the reference codes.

#### 4.3.2. Step 1.2: Power Coupling and Salt Expansion Feedback

In the second step of the coupled phase, the interaction between neutronics and thermal hydraulics is investigated with fixed fuel motion. The velocity field from Step 0.1 is fixed, the reactor power is normalised again, and the results from Step 0.3 and Step 1.1. are taken as initial values for the temperature field and the neutron flux (as well as the precursor concentration) respectively. On the one hand, temperature variations affect the neutron cross sections and diffusion coefficients due to the fuel salt expansion feedback. On the other hand, the power output from the fission events acts as a source term for the heat equation. In this step, there is no two-way coupling between the velocity field and the temperature yet. This means that buoyancy forces are ignored, so any potential errors can be attributed to the power coupling. The relevant simulation parameters chosen for this step are the same as in earlier steps, and can be found in Tables 4.6, 4.8, and 4.9.

The aim of this step is to analyse the flux shape deformation caused by the non-uniform temperature field and the effects of this deformation, in turn, on the temperature field. The observables along the centrelines in this step are the fuel temperature distribution and the change of fission rate density with respect to the solution obtained at Step 0.2, i.e.  $\int_E \Sigma_f \phi dE - \int_E \Sigma_{f,0.2} \phi_{0.2} dE$ . These are shown in Figures 4.14 and 4.15 respectively. Furthermore, to quantify the impact of the power coupling to the criticality of the simulation compared to the non-coupled simulation, the difference in reactivity  $\rho_n$  to the result of Step 1.1 is calculated. This is reported in Table 4.11. The obtained differential fission rate density agrees strongly with the benchmark results. Note that the ragged shape in the figures stems from the TUD- $S_6$  and TUD- $S_2$  codes, which make use of the Phantom- $S_N$  algorithm. This algorithm corrects the nuclear cross sections based on average temperature values per element, unlike the point-wise approach in the other reference codes. For the temperature, the average discrepancy along the horizontal and vertical centrelines amount to 0.1% and 0.2% respectively. For the change in fission rate density, these values are 0.9% and 0.9%. It can be deduced that the current FM-LBM algorithm can correctly reproduce the coupling between flux shape and temperature distribution. Considering the reported reactivities in Table 4.11, similarly to the findings in Step 1.1, both the absolute value of the obtained reactivity and the difference with the 1.1-value are in better agreement with the reference codes than the values obtained from simulations in earlier LBM studies. More concretely, the reactivity differences obtained by Entes and Pijls are more in line with the TUD- $S_6$  value, while the result of the current implementation agrees more with the other benchmark results. This is another indication that the current FM-LBM algorithm can obtain results that agree more closely to the other benchmark diffusion codes.

#### 4.3.3. Step 1.3: Buoyancy

In the third step of the Tiberge benchmark, the full multiphysics problem is analysed without an external source of momentum (i.e.  $U_{lid} = 0 \text{ m s}^{-1}$ ). The flow is thus driven only by buoyancy effects caused by the temperature gradients affecting the fuel's density through the Boussinesq approximation. The relevant simulation parameters chosen for this step are the same as in earlier steps, and can be found in Tables 4.5, 4.6, 4.8, and 4.9. Furthermore, the buoyancy is modelled with a thermal expansion coefficient of  $\beta_{th} = 2.0 \times 10^{-4} \text{ K}^{-1}$  and gravitational acceleration of  $g = 9.81 \text{ m s}^{-2}$ .

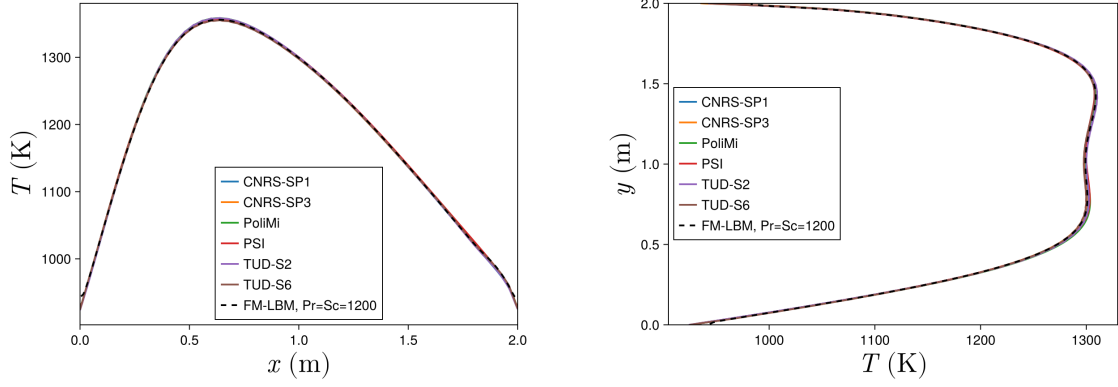


**Figure 4.14:** The temperature for Step 1.2 of the Tiberge benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

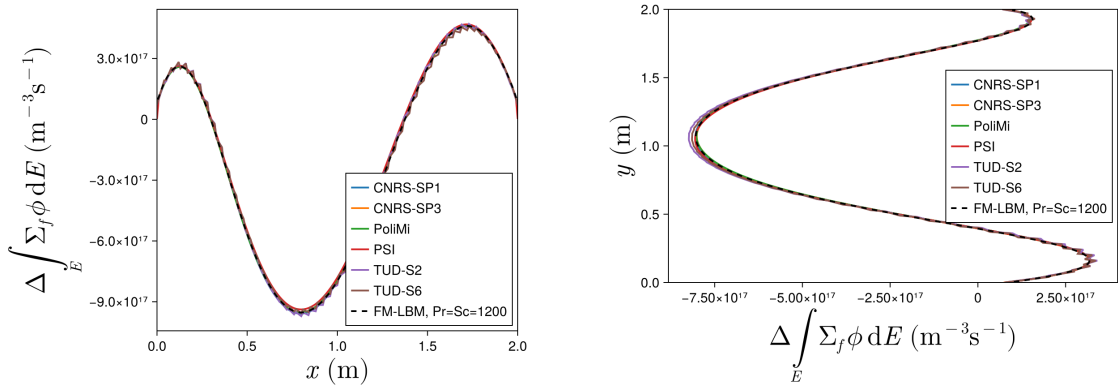


**Figure 4.15:** The change of fission rate density with respect to the results of Step 0.2 for Step 1.2 of the Tiberge benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

| Code          | $k_{eff}$  | $\rho_n$ (pcm) | $\rho_{n,1.2} - \rho_{n,1.1}$ (pcm) |
|---------------|------------|----------------|-------------------------------------|
| CNRS- $SP_1$  | 0.9920320  | -803.2         | -1152.0                             |
| CNRS- $SP_3$  | 0.9914576  | -861.6         | -1152.7                             |
| PoliMi        | 0.9920458  | -801.8         | -1161.0                             |
| PSI           | 0.9920035  | -806.1         | -1154.8                             |
| TUD- $S_2$    | 0.9928061  | -724.6         | -1145.2                             |
| TUD- $S_6$    | 0.9939903  | -604.6         | -1122.0                             |
| LBM-Polderman | 0.99288174 | -716.9         | -1172.1                             |
| LBM-Entes     | 0.9940344  | -600.2         | -1113.7                             |
| LBM-Pijls     | 0.9931081  | -694.0         | -1122.0                             |
| LBM-Blanken   | 0.9925716  | -748.40        | -1145.2                             |

**Table 4.11:** Simulation results for the reactivity difference between Steps 1.2 and 1.1 of the Tiberge benchmark case, alongside results from earlier LBM studies and the reference codes.

The aim of this step is to predict the correct velocity field induced by the fission heat source and the correct reactivity introduction due to the movement of precursors. Errors in the results of this step can be considered mainly related to buoyancy effect or to its modelling, as most of the other coupling issues for steady-state eigenvalue problems should have arisen in the simulation results for Steps 1.1 and 1.2. The observables along the centrelines include the velocity field, the temperature distribution, and the delayed neutron source. These are shown in Figures 4.16, 4.17, 4.18, and 4.19 respectively. Similarly to the earlier steps, Table 4.12 reports the difference in reactivity to the value obtained in Step 0.2. From the figures it becomes clear that the FM-LBM simulation results are in strong agreement with

the benchmark reference codes for all physics fields. This is in contrast to Pijls' FM-LBM model which yielded significantly lower maximum temperatures along the  $AA'$  and  $BB'$  centrelines[43]. The values for the average discrepancy along the horizontal and vertical centrelines are as follows: horizontal velocity: 0.2% and 0.0%, vertical velocity: 0.1% and 0.1%, temperature: 0.1% and 0.2%, delayed neutron source: 0.8% and 0.8%.

For what concerns the delayed neutron source, the profiles obtained in this research show a good agreement with the PSI and CNRS results. In the original paper by Tiberge et al.[60] the small discrepancy in these results compared with the TUD and PoliMi codes is ascribed to the non-uniform mesh used by PSI and CNRS, which was coarser in the middle of the domain. However, the grid in this research is uniform, so this line of reasoning would not be suitable. Further research could investigate the effects of varying the global or local coarseness of the grids for the thermal hydraulics and neutronics simulations. As observed in earlier steps, the obtained reactivity is in better agreement with the values reported by the benchmark reference codes than the other LBM studies, with the exception of the values reported by the TUD codes. The difference relative to the 0.2 step also lies within the range of  $-1181$  pcm to  $-1220$  pcm found by the benchmarks. This proves that the multiphysics code can correctly reproduce the effects induced by salt buoyancy on velocity, temperature, neutron flux, and delayed neutron precursor concentration.

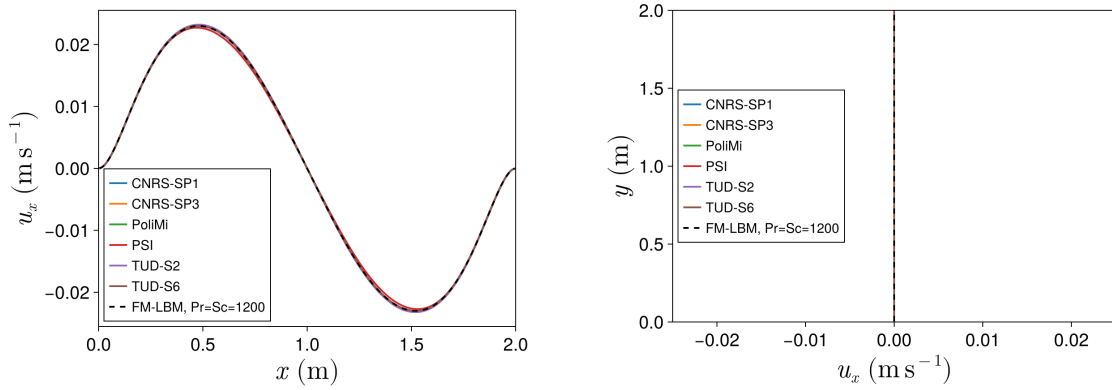


**Figure 4.16:** The horizontal velocity component for Step 1.3 of the Tiberge benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

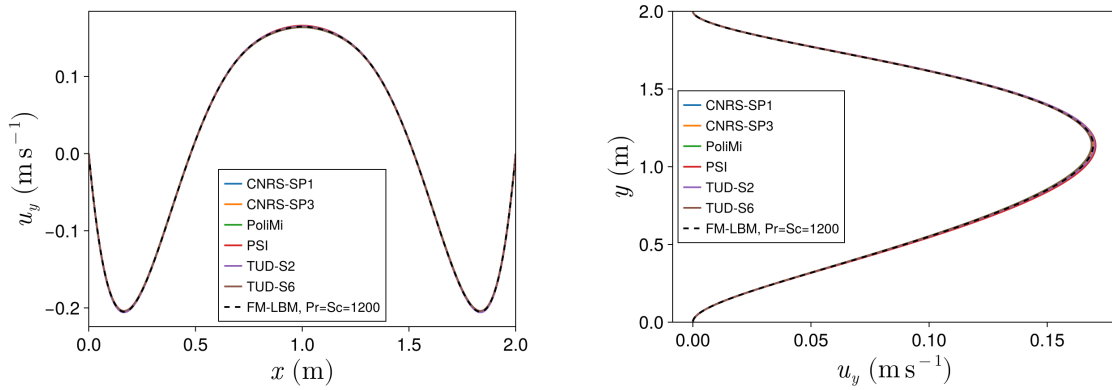


**Figure 4.17:** The vertical velocity component for Step 1.3 of the Tiberge benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

#### 4.3.4. Step 1.4: Full Coupling

In the final step of the steady-state Tiberge benchmark, the full multiphysics problem is studied with the moving lid as an external source of momentum. All the physics fields analysed separately in the previous steps are now simulated simultaneously. The observable in this step is the difference in reactivity to the result of Step 0.2 for different values of the top lid velocity  $U_{lid}$  and reactor power  $P$ .

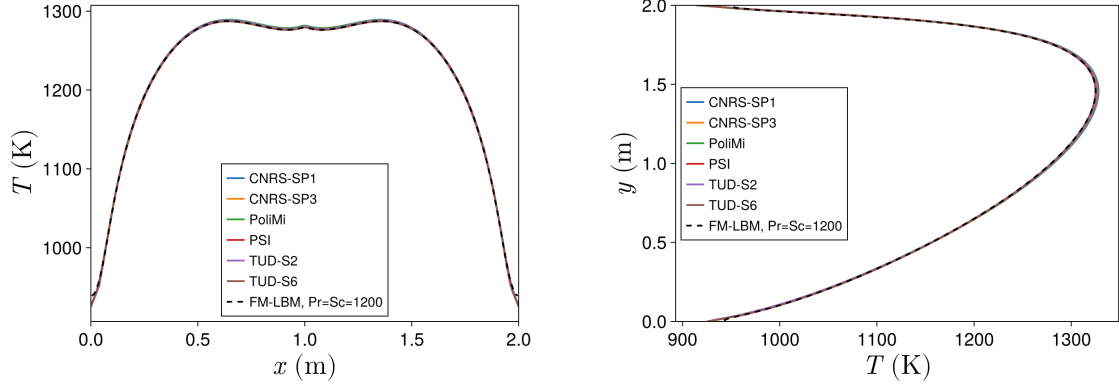


**Figure 4.18:** The temperature for Step 1.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

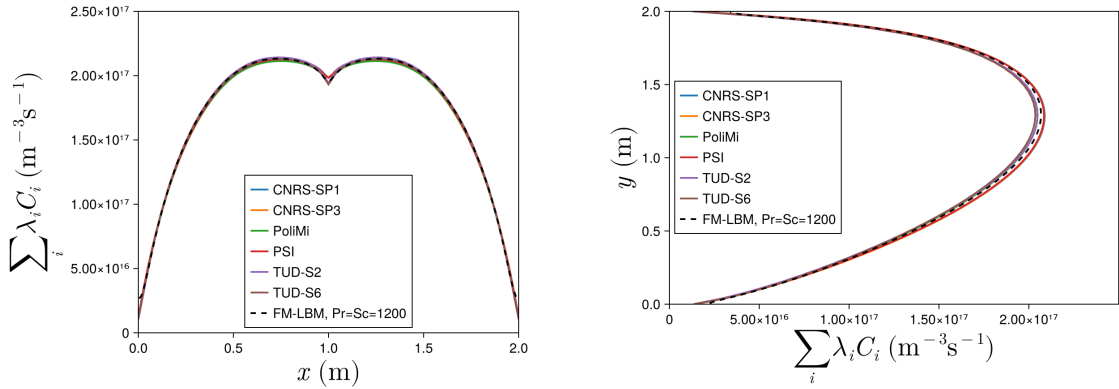


**Figure 4.19:** The delayed neutron source for Step 1.3 of the Tiberga benchmark case along the horizontal (left) and vertical (right) centrelines. The simulation results are shown alongside the data from the reference codes.

| Code          | $k_{eff}$  | $\rho_n$ (pcm) | $\rho_{n,1.3} - \rho_{n,0.2}$ (pcm) |
|---------------|------------|----------------|-------------------------------------|
| CNRS- $SP_1$  | 0.9919730  | -809.2         | -1181.4                             |
| CNRS- $SP_3$  | 0.9914045  | -867.2         | -1220.5                             |
| PoliMi        | 0.9920064  | -805.8         | -1220.7                             |
| PSI           | 0.9919858  | -807.9         | -1219.6                             |
| TUD- $S_2$    | 0.9927933  | -725.9         | -1208.5                             |
| TUD- $S_6$    | 0.9939735  | -606.3         | -1184.4                             |
| LBM-Polderman | 0.99278435 | -726.8         | -1247.4                             |
| LBM-Entes     | 0.9940039  | -603.2         | -1181.4                             |
| LBM-Pijls     | 0.9930794  | -696.9         | -1182.3                             |
| LBM-Blanken   | 0.9925533  | -750.25        | -1207.1                             |

**Table 4.12:** Simulation results for the reactivity difference between Steps 1.3 and 0.2 of the Tiberga benchmark case, alongside results from earlier LBM studies and the reference codes.

The velocity field obtained from Step 0.1 and the neutron flux and precursor concentration obtained from Step 1.2 are taken as initial values to speed up simulation convergence. The simulation parameters chosen for this step are the same as in earlier steps, and can be found in Tables 4.5, 4.6, 4.8, and 4.9.

For the sake of brevity, simulations were run for  $U_{lid} \in \{0.1, 0.3, 0.5\} \text{ m s}^{-1}$  and  $P \in \{0.2, 0.6, 1.0\} \text{ GW}$ . The results of the differences of reactivity with respect to the value obtained in Step 0.2 is shown in Table 4.13. It must be noted that in Pijls' FM-LBM implementation, unstable and non-converging results were reported for simulations with  $Pr = Sc = 1200$ , and therefore the values for both dimensionless numbers were reduced to 1000. However, this anomaly was not found in the current research, and

therefore the original 1200 value is kept.

As expected, higher power levels increase the average temperature in the system, which results in lower reactivity due to the salt expansion feedback. The lid-velocity has a dual effect on the reactivity, depending on the power level. In the case of low power, increasing the lid velocity corresponds to higher negative reactivity differences, as the concentration of precursors is shifted more from the cavity centre, where they can contribute less to the overall fission chain reaction. On the other hand, in case of high power, increasing the lid velocity also increases the forced convection of heat, and so the temperature gets (relatively) lower in the centre region of high importance. Whichever of these two mechanisms prevails over the other depends on the power level, as can be seen in the table.

Comparing the values in the various configurations for reactor power  $P$  and lid velocity  $U_{lid}$ , it can be seen that the previous LBM studies consistently overestimated the (negative) reactivity differences. As the differences obtained in this research are the lowest among the LBM results (as far as their results are reported), it can be concluded that the current implementation of the Filter-Matrix Lattice Boltzmann Method is in very good alignment with the reference codes. Specifically, for every tested configuration, the difference falls within the values reported by the reference codes.

The last step of the Tiberga benchmark completes the validation of the current FM-LBM tool for steady-state simulations. Overall, the code exhibited strong agreement with the results reported by the reference codes. In every step of the benchmark, the observables along the horizontal and vertical centre-lines displayed average discrepancies of less than 1% compared to the reference codes. Furthermore, the simulated reactivities in each benchmark step consistently aligned more closely to the values reported by the reference codes than the earlier LBM studies. Based on these findings, it can be concluded that the current implementation of the FM-LBM multiphysics tool yields satisfactory results for steady-state simulations.

| Code          | $U_{lid}$ (m s <sup>-1</sup> ) | $\rho_{n,1.4} - \rho_{n,0.2}$ (pcm) |        |         |
|---------------|--------------------------------|-------------------------------------|--------|---------|
|               |                                | $P$ (GW)                            |        |         |
|               |                                | 0.2                                 | 0.6    | 1.0     |
| CNRS- $SP_1$  | 0.1                            | -268.5                              | -783.2 | -1219.6 |
| CNRS- $SP_3$  |                                | -268.8                              | -738.2 | -1219.7 |
| PoliMi        |                                | -269.0                              | -734.0 | -1225.0 |
| PSI           |                                | -270.2                              | -738.6 | -1214.1 |
| TUD- $S_2$    |                                | -265.8                              | -730.7 | -1207.6 |
| TUD- $S_6$    |                                | -260.1                              | -716.0 | -1183.5 |
| LBM-Polderman |                                | —                                   | —      | —       |
| LBM-Entes     |                                | —                                   | —      | —       |
| LBM-Pijls     |                                | -260.7                              | -722.3 | -1195.7 |
| LBM-Blanken   |                                | -264.6                              | -729.3 | -1206.0 |
| CNRS- $SP_1$  | 0.3                            | -269.5                              | -735.2 | -1212.1 |
| CNRS- $SP_3$  |                                | -269.8                              | -735.3 | -1212.4 |
| PoliMi        |                                | -278.0                              | -734.0 | -1219.0 |
| PSI           |                                | -274.0                              | -735.6 | -1206.8 |
| TUD- $S_2$    |                                | -269.5                              | -727.7 | -1200.2 |
| TUD- $S_6$    |                                | -263.8                              | -713.2 | -1176.4 |
| LBM-Polderman |                                | —                                   | —      | —       |
| LBM-Entes     |                                | —                                   | -712.7 | -1173.1 |
| LBM-Pijls     |                                | -264.1                              | -719.2 | -1188.6 |
| LBM-Blanken   |                                | -268.0                              | -726.4 | -1193.7 |
| CNRS- $SP_1$  | 0.5                            | -276.5                              | -732.9 | -1204.8 |
| CNRS- $SP_3$  |                                | -276.8                              | -733.0 | -1205.2 |
| PoliMi        |                                | -284.0                              | -737.0 | -1214.0 |
| PSI           |                                | -278.1                              | -733.1 | -1199.8 |
| TUD- $S_2$    |                                | -273.1                              | -725.2 | -1193.0 |
| TUD- $S_6$    |                                | -267.5                              | -710.8 | -1169.7 |
| LBM-Polderman |                                | —                                   | —      | -1085.3 |
| LBM-Entes     |                                | —                                   | -708.9 | -1164.8 |
| LBM-Pijls     |                                | -268.5                              | -716.3 | -1181.5 |
| LBM-Blanken   |                                | -271.6                              | -724.0 | -1192.6 |

**Table 4.13:** Simulation results for the reactivity difference between Steps 1.4 and 0.2 of the Tiberge benchmark case for different lid velocities and reactor powers, alongside results from earlier LBM studies and the reference codes.

## 4.4. Fluid Dynamics Simulations in Complex Geometries

To validate the FM-LBM multiphysics tool's capabilities to model fluids in complex geometries, the confined laminar flow around a square obstacle is investigated. This numerical benchmark is described by Breuer et al.[79] who compared the results of CFD simulations using Lattice Boltzmann automata and the finite volume method. In their research, the resulting flow patterns were studied for different values of the Reynolds number in the laminar regime, based on velocity profiles at several positions in the flow field, as well as integral flow parameters such as the recirculation length and drag coefficients.

### 4.4.1. Description of the Breuer Benchmark

Figure 4.20 displays the geometry of the benchmark, in which a fluid flows around a square with side length  $D$  mounted centred inside a plane channel with height  $H$ . The ratio  $D/H$ , also known as the blockage ratio  $B$ , is set to  $1/8$ . In order to reduce the influence of inflow and outflow boundary conditions, the length of the channel  $L$  was set to  $50D$ . The distance between the square and the inflow boundary is  $l = L/4$ . The origin of the domain is taken to be the centre of the square, and all distance measures are expressed in terms of  $D$ . Different spatial resolutions can be chosen to study the convergence of the simulation results; the research by Breuer et al. performed simulations with lengthwise resolutions from 500 to 2000 grid points.

Instead of a moving lid, as was the case in the Tiberge benchmark, the fluid is now driven by a constant pressure gradient between the parallel walls, on which no-slip boundary conditions are imposed. Without obstructions, this would result in (plane) Poiseuille flow, which yields a parabolic velocity distribution proportional to the pressure gradient[47]. Therefore, in order to emulate a fully developed laminar channel flow upstream of the square, a parabolic velocity profile with maximum velocity  $u_{max}$  is prescribed at the channel inlet using the halfway bounce-back approach as described in Section 3.3. For the outlet, a fixed pressure is imposed through the use of the anti-bounce-back method (with downstream-extrapolated velocities), following the advice of [18]. The outflow condition ensures that possible vortices can approach the outlet without significant disturbances or reflections into the inner domain.

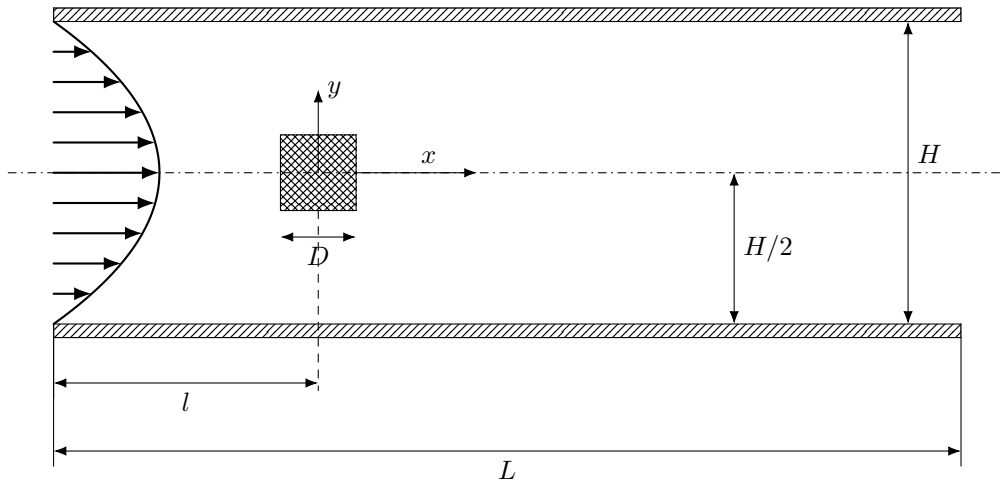


**Figure 4.20:** Schematic of the Breuer benchmark, with the centre of the square defined as the domain origin. The figure is inspired by [79].

#### 4.4.2. Flow patterns

Figure 4.21 shows the resulting velocity fields (with streamlines) in the channel with lengthwise resolution of  $N_X = 1000$  at four different Reynolds numbers ( $Re = 40, 70, 100, 200$ ), characterising different flow regimes. This Reynolds number is based on the square side length  $D$  and the maximum flow velocity  $u_{max}$ . At low  $Re$ , the fluid steadily flows around the side walls of the square, and separation happens at the trailing edge of the sharp-edged body. Just behind the square, the near-wake comprises a steady recirculation region of two symmetrical vortices on both sides. Moreover, the wake appears to lengthen for increasing  $Re$ , until it becomes unstable above a certain threshold, as can be seen in the flow pattern for  $Re = 70$ . Transverse flow oscillations shape the wake behind the square. For even larger  $Re$  vortices are periodically created behind the square and detach from alternating sides, a process which is known as vortex shedding. This shedding creates a (laminar) Kármán vortex street[79], as can be seen in the flow patterns for  $Re = 100, 200$ . A snapshot of such a vortex street is also shown on the cover of this thesis. As seen from the high-Reynolds flow patterns, the wavelength of the vortex shedding decreases with increasing  $Re$ . Moreover at  $Re = 200$  the flow separation already occurs at the leading edge of the square, which influences the frequency of the shedding. As described in the benchmark paper, simulations with even higher  $Re$  yielded deviations from the periodic vortex shedding in the far-wake, indicating the transition to turbulent flow regimes. For accurate simulations of such flow, proper turbulence models need to be incorporated in the multiphysics tool (see Chapter 6).

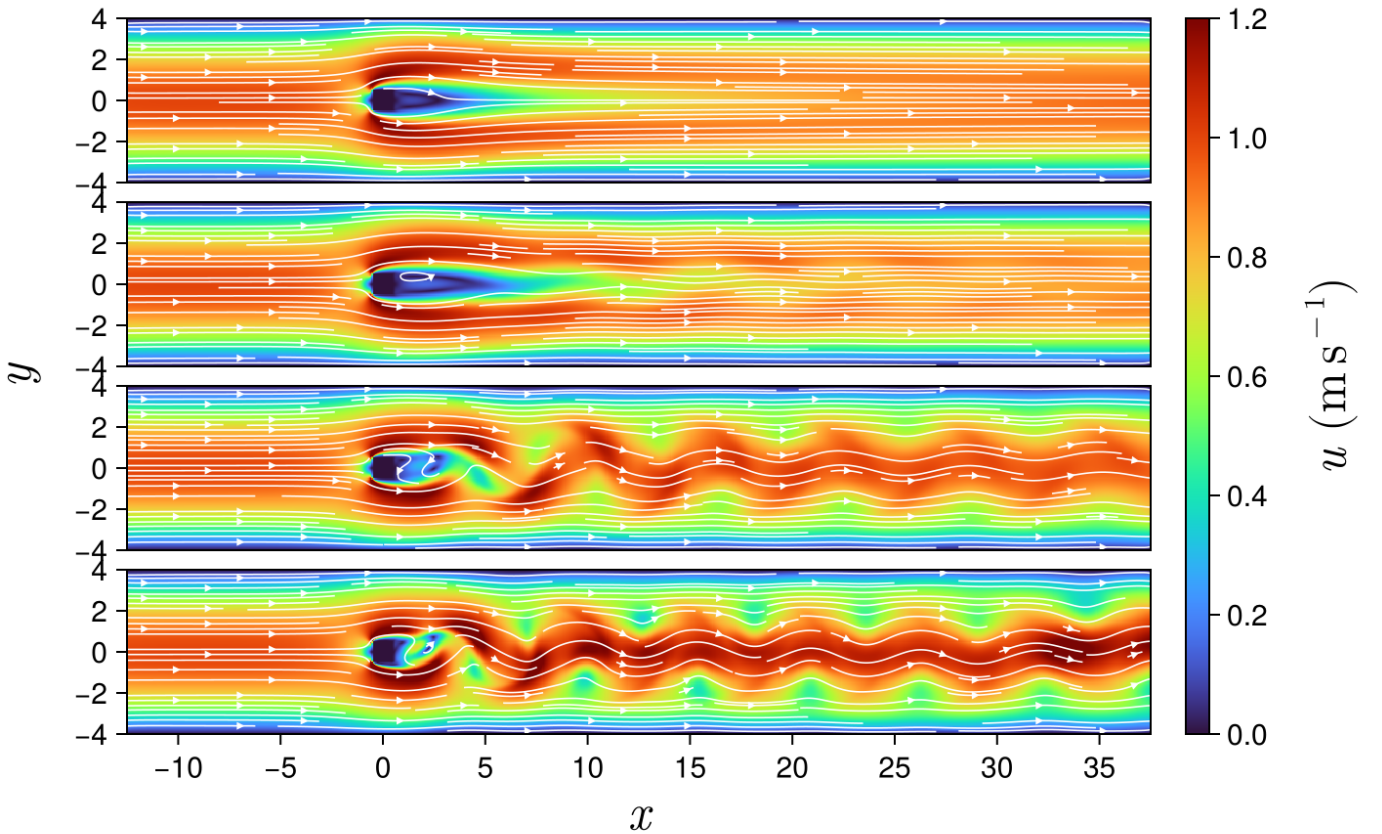


**Figure 4.21:** Velocity fields around and behind the square for (top-to-bottom) Reynolds numbers 40, 70, 100, 200. The axes are scaled to the side length  $D$  of the square, while the colorbar is scaled to the maximum inlet velocity  $u_{max}$ . For clarity, the x-axis has been reported only once at the bottom.

#### 4.4.3. Recirculation in Steady Flow

As described earlier, in the steady flow regime the near-wake behind the square is characterised by a recirculation region that grows for increasing  $Re$ . To quantify this relation, the recirculation length

$L_r$  is measured for various Reynolds numbers. Numerically,  $L_r$  can be computed along the horizontal centreline of the channel by calculating the distance between the rear side of the square and the point where the streamwise velocity  $u_x$  changes its sign from negative to positive. In the  $5 < Re < 60$  flow regime, Breuer et al. found the following linear relation using both LBM and FVM[79]:

$$L_r/D = -0.065 + 0.0554Re. \quad (4.7)$$

Figure 4.22 presents the computed recirculation lengths alongside the linear fit. Clearly the simulation results are in very good agreement with the relation obtained by Breuer. Indeed, the root mean square error against the literature fit is only  $0.021D$ , corresponding to a mean absolute percentage error of 3.16%. It can be concluded that the FM-LBM multiphysics tool is able to accurately model the near-wake for steady laminar flow.

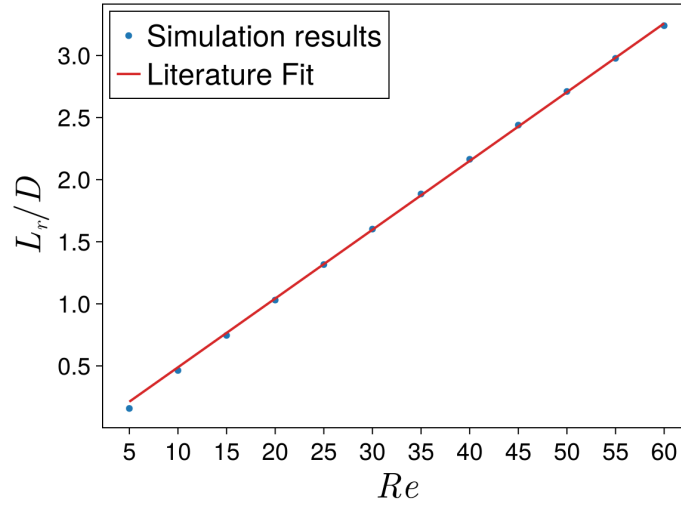


**Figure 4.22:** Computed recirculation lengths  $L_r$  for various Reynolds numbers. The linear fit reported by Breuer et al.[79] is added in red.

#### 4.4.4. Velocity profiles in Unsteady Flow

To study the unsteady flow regime, the velocity profiles at different positions in the flow field are obtained for  $Re = 100$  and compared with the profiles found by Breuer et al. As the shape of the wake evolves in time, the benchmark paper describes the evaluation moment  $t_{eval}$  as the time at which the cross-stream velocity  $u_y$  on the horizontal centreline at an axial position of  $10D$  behind the square (i.e.  $u_y(10.5, 0)$ ) changes its sign from negative to positive. Figure 4.24 shows the velocity distribution of both velocity components ( $u_x$  and  $u_y$ ) along the horizontal centreline, while Figure 4.25 show the distributions along the vertical lines positioned at the centre of the square, as well as the profiles in the near-wake and far-wake. For these simulations a grid resolution of  $N_X = 2000$  was used. Comparing the simulation results to those from the literature, the velocity profiles obtained in this research show a very good agreement with the profiles reported by Breuer et al. Even in the sections of the wake governed by the vortex street, the distributions closely follow the same trends as the reference results. This agreement is especially assuring given the fact that the flow is very unsteady, and the moment of evaluation  $t_{eval}$  therefore has a strong influence on the results. The small deviations from the reference profiles can therefore be attributed to the finite temporal and spatial resolution. It can be concluded that the FM-LBM multiphysics tool, particularly the LUT implementation for boundary treatment, is able to accurately handle both steady and unsteady laminar flow when complex geometries are present in the computational domain.

#### Spatial convergence

In the absence of explicit numerical data reported by Breuer et al., the validation of the obtained flow patterns is constrained to a qualitative comparison of the velocity profiles. Further research can investigate other flow parameters such as the lift and drag coefficient, as well as the dimensionless Strouhal

number, which describes the oscillatory nature of the vortex shedding. To still make an assessment of the model's performance, the spatial convergence of the simulation is investigated. Specifically, the velocity profiles defined in the previous section are computed for various grid sizes and compared against the simulation with the highest spatial resolution (i.e.  $N_X \times N_Y = 2000 \times 320$ ). This comparison is quantified using the same  $L_2$  discrepancy measure as was used when validating the Tiberger benchmark results (see Equation 4.2).

Figures 4.26 to 4.28 present the velocity discrepancies for grid lengths of 500 to 1900 on both a linear scale and a logarithmic scale. These simulations were all performed with  $Re = 100$ , with the same conditional evaluation time as defined above. As expected, the discrepancy decreases when the spatial resolution is increased. For the discrepancies along the horizontal centreline, this decrease is roughly linear, while the velocities along the vertical lines exhibit approximate second-order spatial convergence. This latter observation is in line with the theoretical second-order spatial accuracy of the Lattice Boltzmann Method[18]. In this case, two explanations have been found for the deviation from the ideal second-order convergence. First, the flow in this regime is very unstable due to the vortex shedding, and as explained above the time of evaluation only depends on the evolution of the vertical velocity component at one specific point in the wake. As this evaluation occurs between iterations of the algorithm,  $t_{eval}$  can differ significantly between simulations due to the finite temporal resolution. As a result, the fluid's velocity at the other points (on which  $t_{eval}$  does not rely) can deviate substantially from the results found for  $N_x = 2000$ . This is a fundamentally different source of deviation than the 'standard' discretisation error of the LBM. Second, the velocity components change signs quite often along the evaluation lines, with some regions having exactly zero velocity. This is especially visible in the vertical velocity profile of Figure 4.24. As the simulations are all performed with different grid sizes, linear interpolation of the  $N_X = 2000$  results is used to calculate the total discrepancy along the lines. This interpolation combined with the different grid sizes, can induce errors near the zero-crossings which are small in absolute terms, but large relative to the local velocity. The presence and the size of these numerical errors are difficult to predict, but they can be an explanation for the chaotic convergence of the discrepancies, especially those of the vertical velocity components. This near-zero effect is more clear when calculating the point-wise discrepancy, shown in Figure 4.23. Further research can account for this effect by restricting the discrepancy calculations to the regions where (the absolute values of) the velocity components are above a certain threshold.

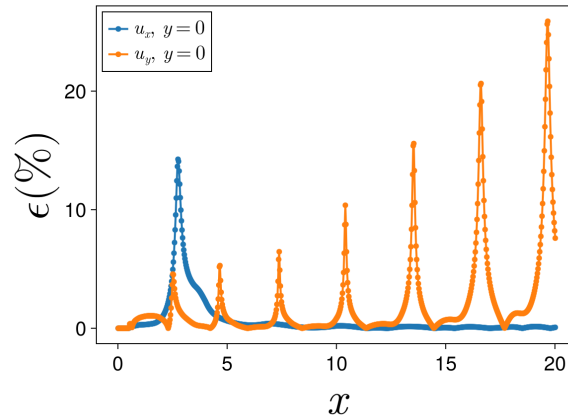


**Figure 4.23:** Pointwise discrepancy of the horizontal and vertical velocity components of the Breuer benchmark case along the horizontal centrelines between  $N_X = 1900$  and  $N_X = 2000$ . The discrepancy is highest near the zero-crossings of the velocities (compare with Figure 4.24).

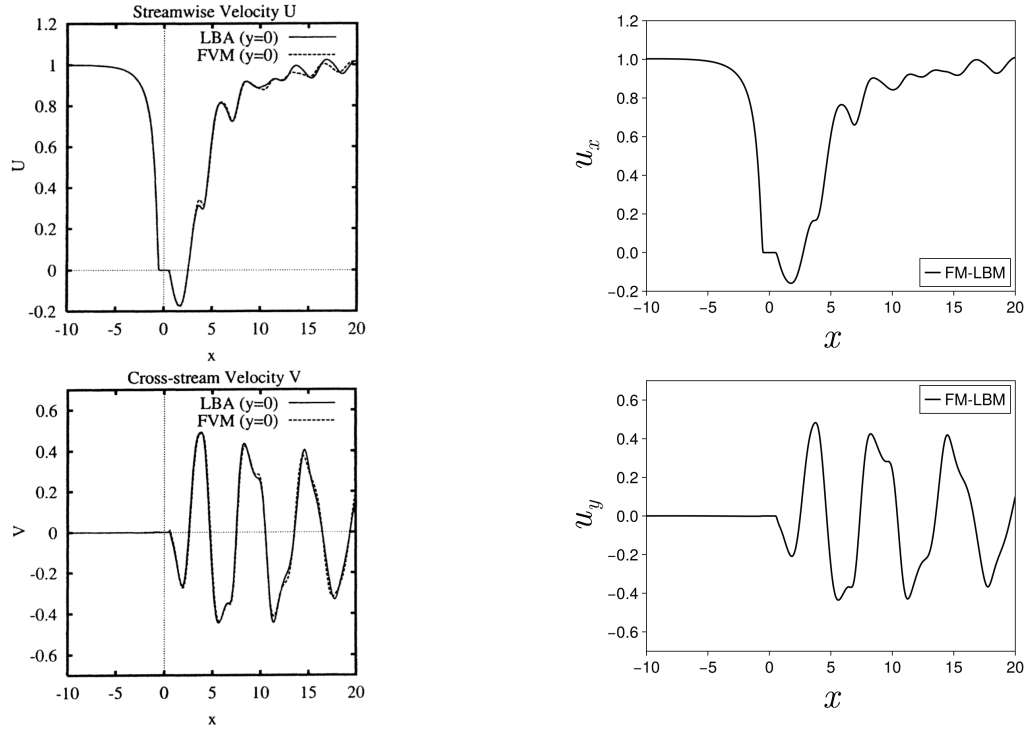


**Figure 4.24:** Streamwise (above) and cross-stream (below) velocity profiles at  $t_{eval}$  along the horizontal centreline obtained by Breuer et al.[79] (left) and the current FM-LBM implementation (right). The vertical axes are scaled to the maximum inlet velocity  $u_{max}$ .

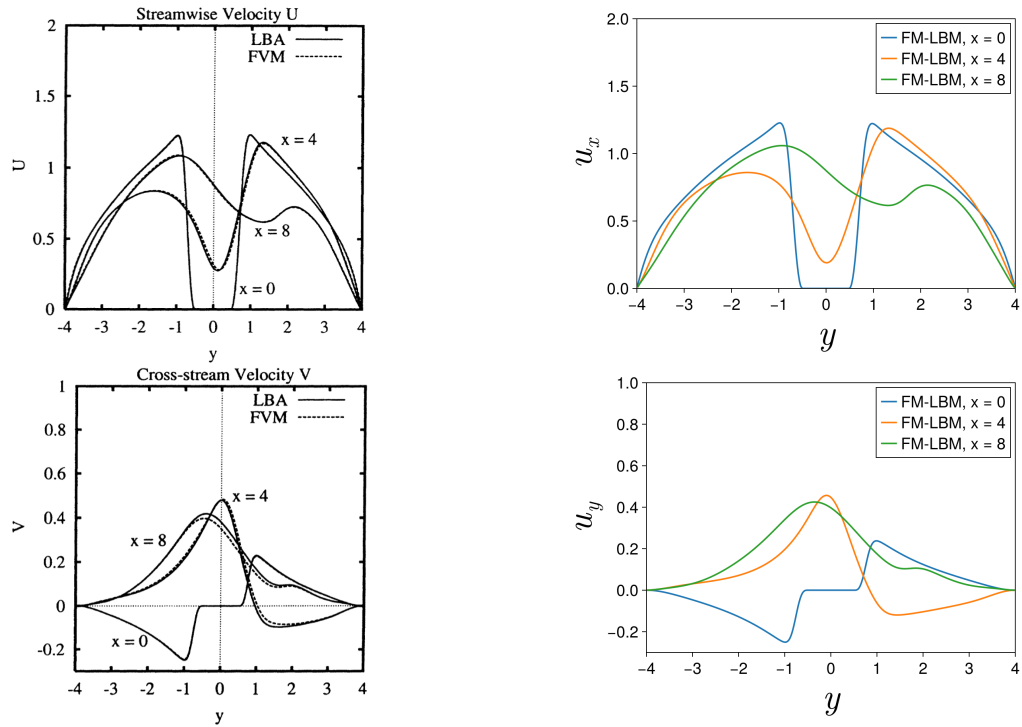


**Figure 4.25:** Streamwise (above) and cross-stream (below) velocity profiles at  $t_{eval}$  along the vertical centrelines ( $x = 0$ ,  $x = 4$ , and  $x = 8$ ) obtained by Breuer et al.[79] (left) and the current FM-LBM implementation (right).

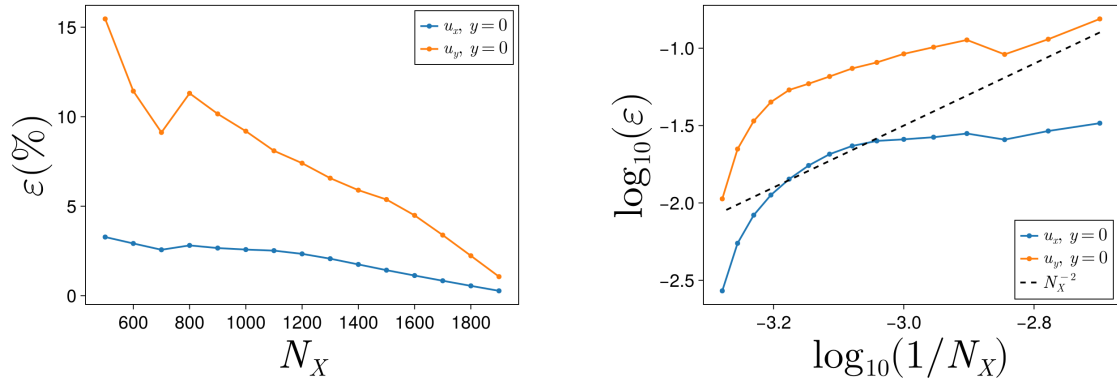


**Figure 4.26:** Discrepancy of the horizontal and vertical velocity components of the Breuer benchmark case along the horizontal centreline for different grid lengths  $N_X$ , presented on a linear (left) and logarithmic scale (right).

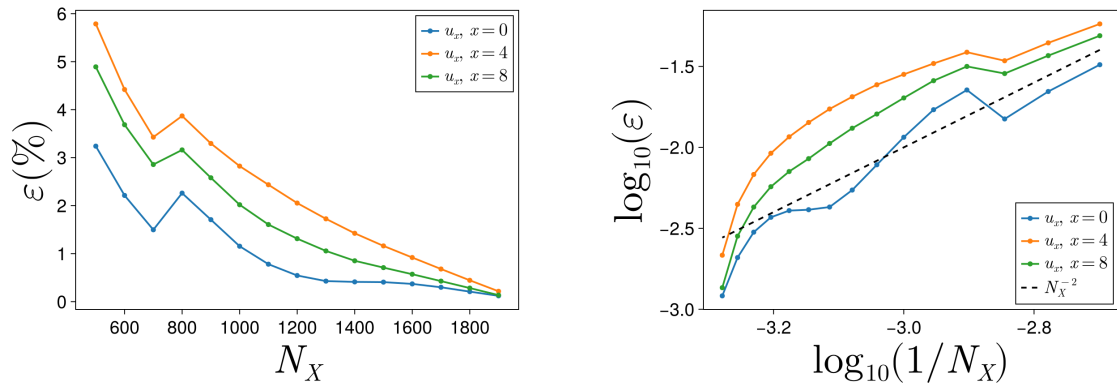


**Figure 4.27:** Discrepancy of the horizontal velocity components of the Breuer benchmark case along vertical lines for different grid lengths  $N_X$ , presented on a linear (left) and logarithmic scale (right).

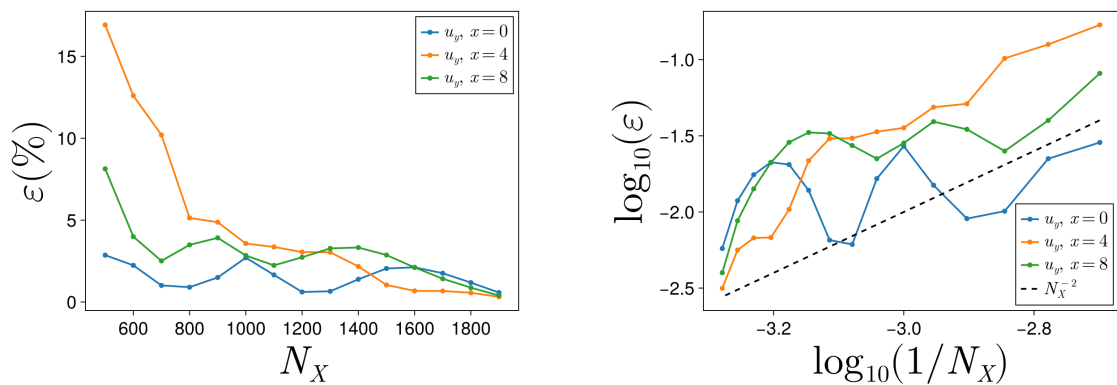


**Figure 4.28:** Discrepancy of the vertical velocity components of the Breuer benchmark case along vertical lines for different grid lengths  $N_X$ , presented on a linear (left) and logarithmic scale (right).

#### 4.4.5. Validation of the IBB implementation

The standard setup of the Breuer benchmark places the square obstacle such that it aligns with the orientation and spacing of the computational lattice grid. In this configuration, the square's boundaries are aligned perfectly halfway between lattice nodes, and therefore the bounce-back treatment of the particle distributions can be adequately modelled using the standard HBB method. In order to assess the algorithm's capabilities of handling atypical boundaries, a horizontal shift  $\delta_x$  is applied to the square's position, as displayed in Figure 4.29. This shift changes the distance between the square's sides and the boundary nodes on either side. Therefore the interpolated bounce-back method should be used to accurately describe the shifted boundaries.

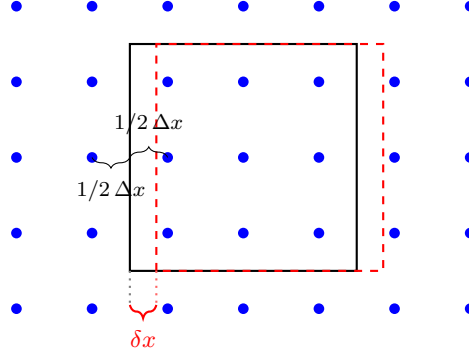


**Figure 4.29:** Schematic of the standard square obstacle boundaries in black, as well as a square shifted horizontally in the lattice, indicated with dashed red lines. Note that the  $\delta_x$  shift can be positive or negative, corresponding to a shift to the right or left respectively.

Figures 4.30 and 4.31 show the resulting streamwise and cross-stream velocity distributions for three different  $\delta_x$  shifts of the square, expressed in lattice spacings  $\Delta x$ . To mitigate the errors obtained from unsteady vortex shedding as reported in the previous section, the simulations were performed on a  $1000 \times 160$  spatial grid with  $Re = 40$ . The shift of the obstacle appears to produce a similar shift of the velocity profiles. This can be seen especially clearly in the zoomed scatter plot of the horizontal velocity component. As explained in Chapter 3, the implementation of the IBB method at the boundary nodes in the front of the square relies on its distance to the wall, which varies for the different shifts. As such, the velocity at the boundary node is lowest for the left-shifted square and highest for the right-shifted square. At the back of the square, this effect is reversed, which leads to a 'delay' in the velocity profile for the right-shifted square. The same shift can be seen in the vertical velocity profile, with a 'leading' left-shifted profile and a 'delayed' right-shifted profile. Moreover, the amplitudes of the small oscillations in the wake seem to differ between the simulations. This might come as a surprise, considering the fact that the shift of the square was purely horizontal, and therefore one might not expect changes in the vertical velocity components. However, the horizontal displacement also alters the distance of the boundary nodes at the *corners* at the left side of the square, as can be seen in Figure 4.29. Since a D3Q19 lattice velocity set is chosen for the momentum field, which contains diagonal directions (i.e.  $i = 8, 9, 10, 11$ , see Figure 3.1b), the horizontal shift also influences the vertical component of the flow.

These small shifts aside, the full-size plots show that the shifted simulations are able to produce the same trends in the velocity profiles. Therefore these results form a proof of concept that the current FM-LBM tool is able to accurately handle the momentum transport of a fluid in non-aligned complex geometries using the LUT's implementation for boundary treatment with the IBB method. While the Breuer benchmark was limited to a single square in a channel, the current implementation allows for even more obstacles and complex geometries that can be added modularly to the domain. One such example is shown in Figure 4.32.

Although the present work has restricted the validation of the complex-geometry implementation to the momentum field, the same approach can be naturally extended to the other physics fields of the multiphysics. When scaling up to full-sized, 3D reactor multiphysics simulations, one could construct a domain partitioned into multiple regions whose physical behaviour differs from one field to the next. For example, solid control rods in the reactor core are opaque to molten salt flow but transparent to neutrons, while the heat-exchanger region carries flow and energy but should contain no fissile material.

This region-dependent presence of active fields and subsequent boundary conditions might warrant the use of separate lookup tables per field, or even per region. Further research can investigate the tool's accuracy using a multi-LUT approach and its effects on the computational performance and memory footprint.

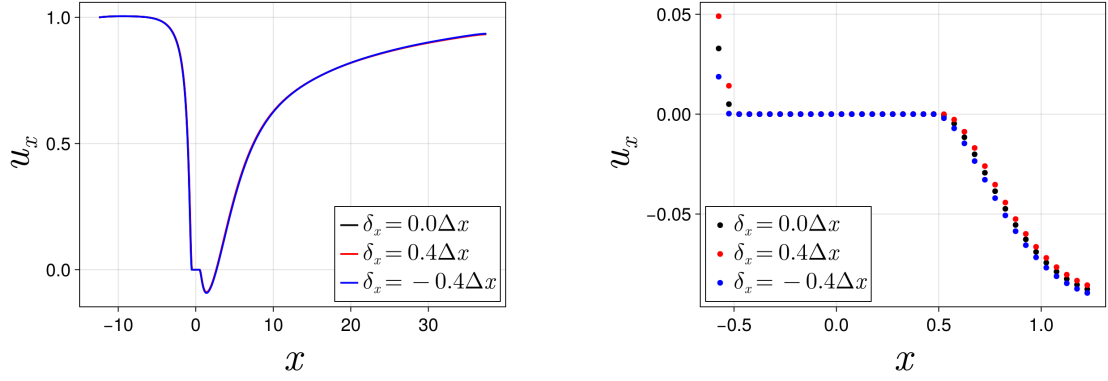


**Figure 4.30:** Streamwise velocity profile along the horizontal centreline for three shifts  $\delta_x$  of the square obstacle, in full (left) and zoomed near the square region (scatter plot, right). Note that the horizontal axis's origin is positioned at the centre of the square with  $\delta_x = 0$ . The vertical axis is scaled to the maximum inlet velocity  $u_{max}$ .

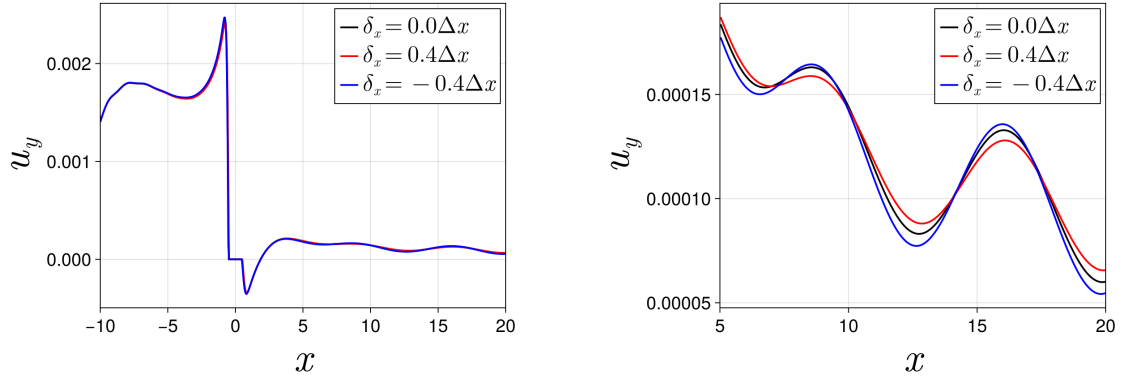


**Figure 4.31:** Cross-stream velocity profile along the horizontal centreline for three shifts  $\delta_x$  of the square obstacle, in full (left) and zoomed in the far-wake (right).

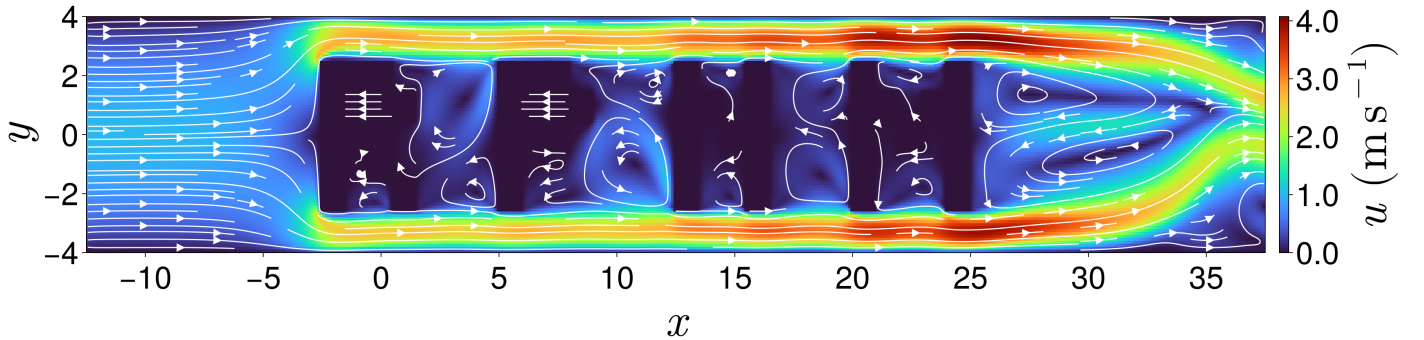


**Figure 4.32:** Velocity field around the 'RPNM' letters for Reynolds number 40.

# 5

## Validation of Transient Multiphysics Models

In this chapter the capability of the FM-LBM tool to model transient multiphysics is assessed. Before attempting to simulate the fully coupled transient phase of the Tiberge benchmark, the correct implementation of the Predictor-Corrector Quasi-Static method for the (single-physics) transient neutron flux is first verified in a simplified domain, for which a reference solution exists. Although this intermediate step might seem redundant or a step back from coupled simulations, the correct implementation of PC-QSM has been a point of some difficulty in earlier LBM work[43]. Once the simulations of the simplified case have been validated, the final phase of the Tiberge benchmark can be assessed with increased confidence in the implemented method.

In Section 5.1 the model capabilities of simulating transient neutronics in the absence of fuel flow and thermal gradients is studied. This is done in a 1-dimensional slab through a reactivity insertion. In the context of the PC-QSM, first the steady-state solutions for the neutron flux and precursor concentrations are obtained, after which the adjoint version of the problem is simulated. As explained in Section 3.6, the solution to this adjoint flux problem is used as a weight function for the point-kinetic parameters in the transient scheme. To illustrate the effect of multiple neutron energy groups on the model, the steady-state, adjoint, and transient problems are simulated with varying numbers of energy groups. Lastly, the stability of the PC-QSM is tested for different number of PK time steps. In Section 5.2 the application of the FM-LBM multiphysics tool to the final phase of the Tiberge benchmark is presented. Here the fully-coupled reactor is perturbed through variations of the volumetric heat transfer coefficient. The power response of the model is studied by varying the frequency of the perturbations. As the steady-state forward solution has already been obtained in Section 4.3.4, the solution to the adjoint problem is presented, after which the transient results are obtained.

### 5.1. Transient Neutronics Simulations

#### 5.1.1. Problem Description

To test the implementation of the PC-QSM for transient neutronics, as described in Section 3.7, the FM-LBM tool is applied to a 1-dimensional slab with no flow and constant, uniform temperature. For such simplified situations, the amplitude of the neutron and precursor density can be modelled by ignoring the shape of the distributions and reducing the reactor to a single point. In this case, reference solutions can be found for the amplitudes using forward Euler integration of the point-kinetics equations (Eqs. 2.48 and 2.49).

As this simplified problem is done in preparation of the final step of the Tiberge benchmark, the same simulation and nuclear parameters (e.g.  $\chi_g$ ,  $\beta_d$ ,  $\Sigma_x$ ) are used as earlier. A slab of 2 m is modelled by a  $100 \times 1 \times 1$  grid, with vacuum boundary conditions for the flux in the  $x$ -direction and periodic boundaries in the  $y$ - and  $z$ -directions. The thermal hydraulics time step was taken as  $\Delta t_B = 5.0 \times 10^{-4}$  s, in order to keep the same  $\Delta x/\Delta t$  ratio as used in the steady state simulations of the previous chapter. The

steady-state solutions for neutron flux  $\phi_{g,0}$  and delayed neutron precursor concentration  $C_{d,0}$  are again obtained by the FM-LBM algorithm with an inner LBM iteration for the flux, an intermediate power iteration for the  $k_{eff}$  multiplication factor, and an outer LBM iteration for the thermal hydraulics, which all ran until convergence. As there is no fuel flow or thermal variation, this outer loop only has to solve for the precursor concentration. After the solution has been obtained for the forward problem, the steady-state FM-LBM algorithm is employed again to solve for the adjoint neutron flux  $\Phi_0^\dagger$  and adjoint precursor concentration  $C_d^\dagger$ . As explained in Section 2.2.3, for the one-speed diffusion approximation in non-circulating fuel it is expected that  $\phi_0 = \Phi_0^\dagger$ , but differing solutions can occur for multiple energy groups. Finally, these solutions are used to test the model response to a perturbation. This is done through a prompt reactivity insertion of  $0.1\beta_{tot}$  at  $t = 1$  s.

### 5.1.2. Steady-State Simulations of a 1D slab

As a starting point, the simple case of one energy group and 8 precursor families is modelled. For the one-speed case, the parameters of the first (and highest-energy) neutron group of the Tiberge benchmark are chosen. As before, the power generated by this slab is set to normalise the neutron flux. The results of this simulation are shown in Figure 5.1. Similarly to the results of Step 0.2 of the square cavity problem, the steady state flux has a sinusoidal shape on the slab, which matches the analytical solution for the 1D diffusion equation with vacuum boundary conditions[52]. This is illustrated in the figure with a cosine fit. Similarly, the concentrations of all the precursor families are highest at the centre, and approximately follow the Neumann boundary condition near the edges of the slab. As these precursors are solely produced through nuclear fission and destroyed through radioactive decay according to Equation 2.24, the relative difference in concentrations between families can be explained by the different production fractions  $\beta_d$  and decay constants  $\lambda_d$ .

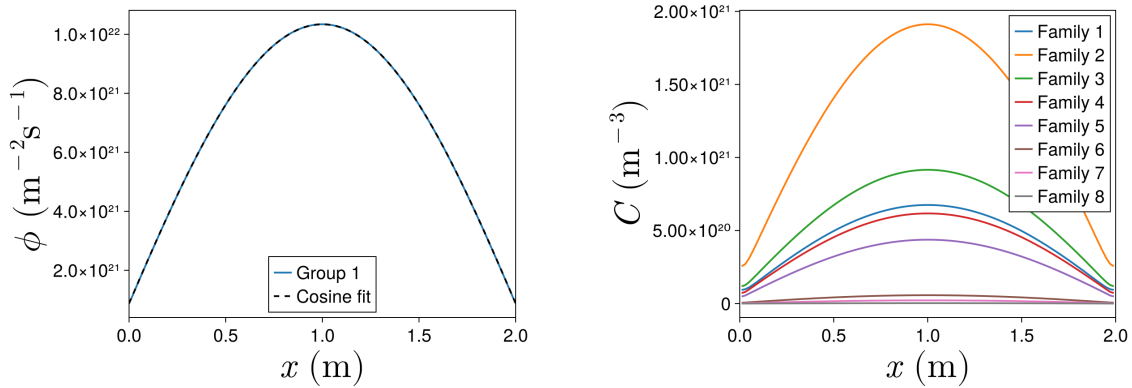


**Figure 5.1:** Steady state distributions of the scalar neutron flux (left) and delayed neutron precursor concentrations (right) for a 1D slab with 1 neutron energy group and 8 precursor families. The one-speed flux can be fitted with a cosine function.

To examine the effect of multiple energy groups, the steady state 1D simulation is repeated with 6 energy groups and 8 precursor families, the results of which are shown in Figure 5.2. The scalar fluxes of the different energy groups again exhibit the cosine shape along the slab found in the one-speed case. Similar to the difference in precursor concentrations, the relative magnitudes of the group fluxes are determined by the different macroscopic cross sections  $\Sigma_{x,g}$ , as well as the prompt and delayed fission spectra ( $\chi_g^p$  and  $\chi_g^d$  respectively).

Further comparing the results of the multi-group simulation with the one-group result, it can be noted that the scalar fluxes of the former case are significantly lower than that of the latter. In other words, if the one-group case is denoted with subscript  $1G$  and the multi-group case with  $MG$ , then  $\phi_{g,MG}(x) < \phi_{1,1G}(x)$ . Even the sum of the fluxes of all energy groups  $\sum_g \phi_{g,MG}$  is lower than the one-speed model flux. This might be surprising, as the number of energy groups should not affect the cross sections governing the NDE. In fact, increasing the number of energy groups results in a higher contribution of the in-scattering term ( $= \sum_{g'} \Sigma_{s,g' \rightarrow g} \phi_{g'}$ ) for all groups.

The reduced fluxes can be explained by the fact that in both cases the power produced by the slab, i.e.  $E_{fission} \sum_g \Sigma_{f,g} \phi_g$  is normalised to the same reactor power  $P = 1$  GW. Therefore the normalised

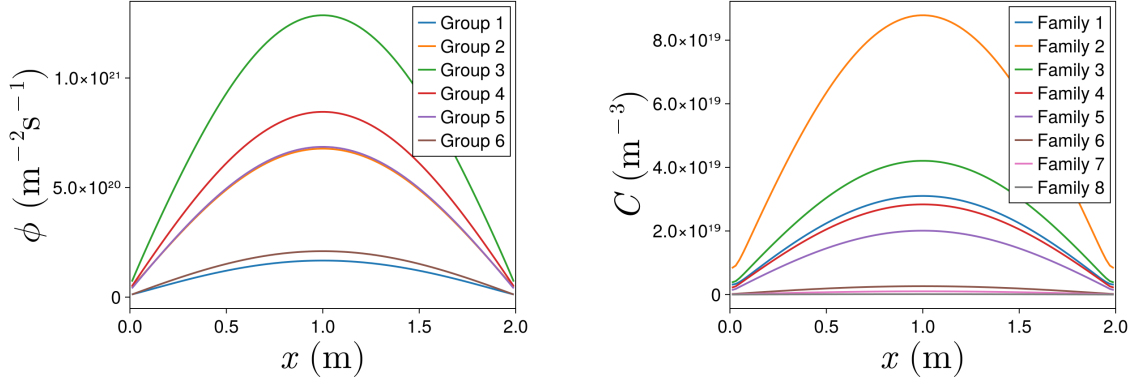


**Figure 5.2:** Steady state distributions of the scalar neutron flux (left) and delayed neutron precursor concentrations (right) for a 1D slab with 6 neutron energy groups and 8 precursor families.

fluxes can be considered 'weighted' by the (group-dependent) fission cross sections. As the value for  $\Sigma_{f,1}$  for the highest energy group is lower than the fission cross sections of the other groups (with the exception of  $\Sigma_{f,2} \approx \Sigma_{f,1}$ , see Table A.1 in the Appendix), this results in a lower total flux  $\sum_g \phi_{g,MG}$ . The inequality can be proven mathematically:

$$\begin{aligned}
 P &= E_{fission} \Sigma_{f,1} \phi_{1,SG} = E_{fission} \sum_g \Sigma_{f,g} \phi_{g,MG} \\
 &\Rightarrow \Sigma_{f,1} \phi_{1,S} = \sum_g \Sigma_{f,g} \phi_{g,MG} \\
 &= \Sigma_{f,1} \phi_{1,MG} + \Sigma_{f,2} \phi_{2,MG} + \Sigma_{f,3} \phi_{3,MG} + \Sigma_{f,4} \phi_{4,MG} + \Sigma_{f,5} \phi_{5,MG} + \Sigma_{f,6} \phi_{6,MG} \\
 &\Rightarrow \phi_{1,S} = \frac{\Sigma_{f,1}}{\Sigma_{f,1}} \phi_{1,MG} + \frac{\Sigma_{f,2}}{\Sigma_{f,1}} \phi_{2,MG} + \frac{\Sigma_{f,3}}{\Sigma_{f,1}} \phi_{3,MG} + \frac{\Sigma_{f,4}}{\Sigma_{f,1}} \phi_{4,MG} + \frac{\Sigma_{f,5}}{\Sigma_{f,1}} \phi_{5,MG} + \frac{\Sigma_{f,6}}{\Sigma_{f,1}} \phi_{6,MG} \\
 &> \phi_{1,MG} + \phi_{2,MG} + \phi_{3,MG} + \phi_{4,MG} + \phi_{5,MG} + \phi_{6,MG},
 \end{aligned} \tag{5.1}$$

where in the last step it was used that the ratios of fission cross sections were all greater or equal to one. As can be seen in Figure 5.2, this decrease of the total scalar flux also results in a reduction of the individual precursor concentrations. However, the additional neutron groups do not seem to affect the shapes or the ratios between precursor concentrations. This can be explained by the fact that the family-dependent precursor ADE is not influenced by the individual scalar fluxes, only by the magnitude of the *overall* fission source  $\sum_g (\nu \Sigma_f)_g \phi_g$ .

### 5.1.3. Adjoint Flux Simulations

To study the adjoint version of the 1D slab problem, the steady state simulation is modified to solve the differential equations for the adjoint neutron flux  $\Phi_g^\dagger$  and adjoint precursor concentration  $C_d^\dagger$  (Eqs. 2.66 and 2.67). The FM-LBM implementation to model adjoint neutronics is similar to the forward implementation and described in Section 3.6.

As with the forward case, the adjoint problem is first simulated for one neutron energy group and 8 precursor families. The results of the adjoint simulation are presented in Figure 5.3, alongside the results for the forward neutron flux  $\phi$  and forward precursor concentrations  $C_d$  for better comparison. For this one-speed model, it is apparent that the steady state solution for the adjoint flux agrees almost perfectly with the forward flux, i.e.  $\phi_g \approx \Phi_g^\dagger$ . This ties back to the fact that for the one-speed model with stationary fuel, the solutions to the forward and adjoint neutron diffusion equations are equal, as was explained in Section 2.2.4. In fact, the discrepancy between the two obtained distributions amounts to just 0.6%. However, this is not the case for the forward and adjoint precursor concentrations. Not only do the absolute values for the concentrations not match, the ratios between the precursor families are

also different, i.e.  $C_a/C_b \neq C_a^\dagger/C_b^\dagger$ . This is because the parameters governing the precursor generation and destruction ( $\beta_d$  and  $\lambda_d$ ) are used differently in the adjoint equations (Eqs. 2.66 and 2.67) than in the forward equations (Eqs. 2.24 and 2.25).

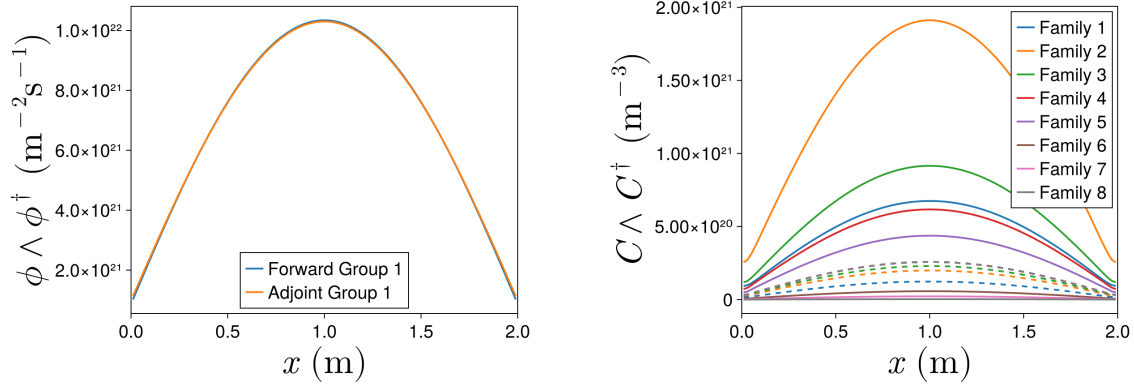


**Figure 5.3:** Steady state distributions of the adjoint neutron flux (left) and adjoint precursors concentrations (right) for a 1D slab with 1 neutron energy group and 8 precursor families. The adjoint values (dashed lines) are shown together with the solutions to the forward simulations (full lines).

Next, the adjoint problem of the 1D slab is repeated with 6 energy groups and 8 precursor families, the results of which are shown in Figure 5.4. As with the adjoint precursors, it now becomes clear that for multiple energy groups, the adjoint neutron fluxes are no longer equal to their forward counterparts. In other words, the equivalence of Equation 2.68 is not valid in the case of multi-group diffusion. Moreover, it can be concluded that the ratio between fluxes is different in the forward and adjoint case, i.e.  $\phi_a/\phi_b \neq \Phi_a^\dagger/\Phi_b^\dagger$ . As explained in Section 2.2.4, this is partly due to the reversal of the scattering source in the adjoint NDE, as well as the swapped summations over the fission sources  $(\nu\Sigma_f)_g$  and the prompt fission spectra  $\chi_g^p$ . In the context of the Predictor-Corrector Quasi-Static Method, this inequality implies that the calculation of the point-kinetic parameters is different whether the steady state solution of the forward flux or the adjoint flux is used as a weight function, which would also affect the outcome of the transient simulations. This effect is shown in the next section.

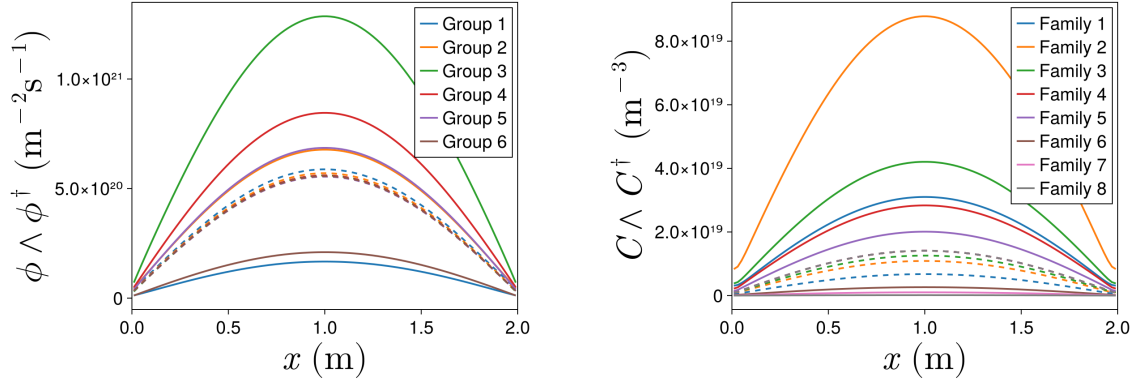


**Figure 5.4:** Steady state distributions of the adjoint neutron flux (left) and adjoint precursors concentrations (right) for a 1D slab with 6 neutron energy groups and 8 precursor families. The adjoint values (dashed lines) are shown together with the solutions to the forward simulations (full lines).

#### 5.1.4. Transient Flux Simulations

To analyse the effect of using the forward neutron flux  $\phi_g$  or the adjoint neutron flux  $\Phi_g^\dagger$  as a weight function for the PC-QSM in Equations 2.50 - 2.53, the model's response to a perturbation is studied. Specifically, the time evolution of the neutron amplitude  $n(t)$  is modelled after a reactivity insertion (or prompt jump) of  $0.1\beta_{tot}$  at  $t = 1$  s. For the 1D slab model, the reference solution for  $n(t)$  and precursor amplitude  $m_d(t)$  can be modelled by a forward Euler method of the point-kinetic equations:

$$n(t + \Delta t) = n(t) + \left( \frac{\rho_n - \beta_{tot}}{\Lambda} n(t) + \sum_d \lambda_d m_d(t) \right) \Delta t, \quad (5.2)$$

$$m_d(t + \Delta t) = m_d(t) + \left( \frac{\beta_d}{\Lambda} n(t) - \lambda_d m_d(t) \right) \Delta t. \quad (5.3)$$

Here,  $\rho_n$ ,  $\beta_{tot}$ ,  $\beta_d$ , and  $\Lambda$  are the point-kinetic parameters for the reactivity, total delayed neutron fraction, delayed neutron fraction for precursor family  $d$ , and prompt neutron generation time respectively. Figure 5.5 shows the time evolution of the neutron amplitude for the approaches using different weight functions  $W$ , relative to the initial amplitude of  $n_0$  at criticality before the prompt jump. These results were generated with ratios  $\Delta t_B/\Delta t_M = 10$  and  $\Delta t_M/\Delta t_S = 100$  for the time steps of the PC-QSM.

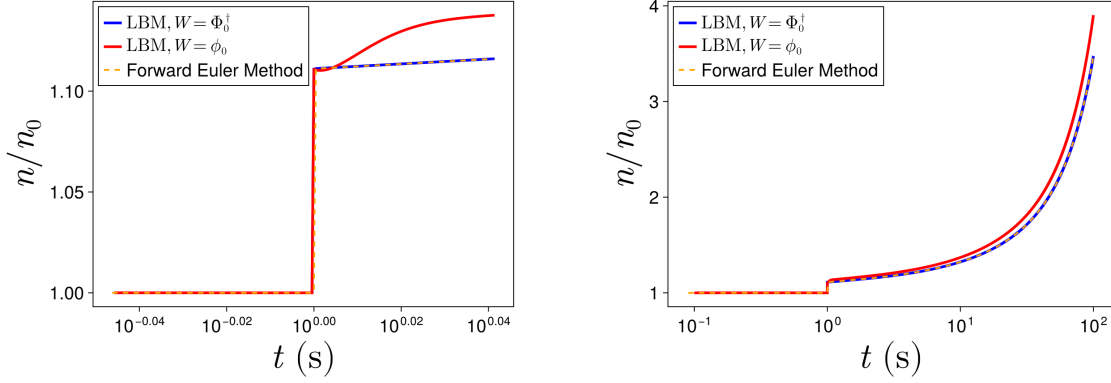


**Figure 5.5:** Neutron amplitude response for a 1D slab with 6 neutron energy groups and 8 precursor families for a reactivity insertion of  $\rho_n = 0.1\beta_{tot}$  at  $t = 1$  s. The blue and red lines correspond to the use of the steady state forward flux  $\phi_0$  or adjoint flux  $\Phi_0^\dagger$  as a weight function  $W$  respectively. The reference solution for the PK equations obtained by the forward Euler method is shown in orange (dashed line). Left: time evolution from 0.9 s to 1.1 s. Right: time evolution from 0.1 s to 100 s.

As can be seen in the figure, the magnitude of the prompt jump  $n/n_0$  is the same for both approaches, around 1.11. This can be explained as follows. Right before ( $t = 1^-$ ) and right after ( $t = 1^+$ ) the reactivity insertion, the time derivative of the neutron amplitude can be considered zero, and the precursor concentrations initially remain at the same level  $S_{d,1} = \sum_d \lambda_d m_d(1^-) = \sum_d \lambda_d m_d(1^+)$ . Using Equation 2.48, this yields:

$$\begin{aligned} \frac{dn(1^-)}{dt} = 0 &\Rightarrow n(1^-) = \frac{\Lambda}{\beta_{tot} - \rho_{n,1^-}} S_{d,1} \\ \frac{dn(1^+)}{dt} = 0 &\Rightarrow n(1^+) = \frac{\Lambda}{\beta_{tot} - \rho_{n,1^+}} S_{d,1}. \end{aligned} \quad (5.4)$$

Using that the slab is initially configured to criticality (i.e.  $\rho_{n,1^-} = 0$ ) and the insertion is  $\rho_{n,1^+} = 0.1\beta_{tot}$ , the magnitude of the prompt jump is

$$\frac{n_{1^+}}{n_0} = \frac{n_{1^+}}{n_{1^-}} = \frac{\beta_{tot} - \rho_{n,1^-}}{\beta_{tot} - \rho_{n,1^+}} = \frac{\beta_{tot} - 0}{\beta_{tot} - 0.1\beta_{tot}} = \frac{10}{9} = 1.111... \quad (5.5)$$

The above equation is true regardless of the absolute values of the PK parameters  $\rho_n$ ,  $\beta_{tot}$ , or  $\Lambda$  at the time of the reactivity insertion. Therefore the initial prompt jump is identical for both choices of weight function  $W$ .

After some time however, the shape function  $\tilde{\phi}(x, t)$  starts evolving as the transient model proceeds, partly driven by the delayed build-up of precursors in response to the higher flux. The PK parameters are then computed from the shape function and weight function  $W$  using the integrals in Equations 2.50 - 2.53. For  $W = \Phi_0^\dagger$ , this gives importance-weighted integrals, meaning neutron groups with high

importance to the chain reaction contribute more to the integrals. This is the standard derivation of the PK equations from transport theory. The resulting neutron amplitude evolves smoothly and remains close to the solution obtained by the forward Euler method. For  $W = \phi_0$ , i.e. the forward-weighted case, the integrals accumulate an error as  $\phi_0$  does not correctly suppress the contributions from neutrons groups that are kinetically less important. This causes the PK parameters to be computed with a systematic bias as the shape changes, resulting in a growing discrepancy in the neutron amplitude. In particular, after the reactivity insertion the forward-weighted amplitude begins rising faster, and this offset is maintained during the rest of the transient simulation. This shows that the effective reactivity computed with  $W = \phi_0$  is overestimated relative to the adjoint-weighted case. This error was also reported by Pijls[43] who used the forward flux  $\phi_0$  as a weight function for the PC-QSM.

It can be concluded that the incorrect choice of the steady state forward flux  $\phi_0$  as a weight function for the calculation of the point-kinetic parameters in the Predictor-Corrector Quasi-Static method results in transient behaviour of multi-group neutronics that is inconsistent with theoretical predictions. When simulating multi-group transient neutronics using PC-QSM, the steady-state solution to the adjoint version of the neutron diffusion equation ( $\Phi_0^\dagger$ ) should instead be used to produce correct results.

### 5.1.5. Stability of the PC-QSM Implementation

Before moving on to the modelling of transient multiphysics in the last step of the Tiberge benchmark, the stability of the Predictor-Corrector Quasi-Static Method is studied for varying simulation parameters. Specifically the effect of different ratios of the various timescales is analysed. As a reminder, the PC-QSM is governed by three timescales: the large timescale  $\Delta t_B$  in which the shape function  $\tilde{\phi}_g$  is calculated along with thermal hydraulics variables such as the precursor concentrations  $C_d$ , the middle timescale  $\Delta t_M$  over which  $\tilde{\phi}_g$  is linearly interpolated and the PK parameters are calculated, and the small timescale  $\Delta t_S$ , over which the neutron and precursor amplitudes ( $n(t)$  and  $m_d(t)$ ) are updated using the fixed PK parameters. In the literature there are no well-defined values for the correct ratios of the timescales; more often these are picked arbitrarily in a case-by-case basis[46], or studied in a trade-off between the increased stability and accuracy of the model versus the increasing computational costs for frequent updates of the shape and amplitude functions[40, 41, 70, 71].

In the case of the 1D slab problem, the stability of the reactivity insertion model as a function of the ratios  $N_{TM} = \Delta t_B / \Delta t_M$  and  $N_{TS} = \Delta t_M / \Delta t_S$  is investigated by varying the ratios from 1 to 100 and recording which configurations produce stable results for the transient simulation. In order to force unstable results, the large timescale  $\Delta t_B$  for thermal hydraulics and neutron shape function updates was increased from  $5.0 \times 10^{-4}$  s to  $5.0 \times 10^{-2}$  s. The results of this investigation are shown in Figure 5.6.

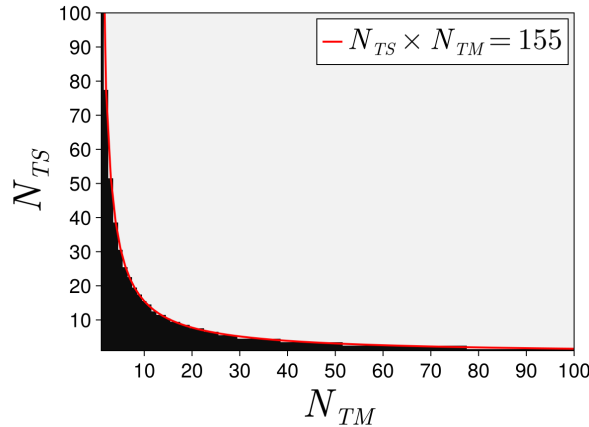


**Figure 5.6:** Stable (grey) and unstable (black) configurations for the PC-QSM in the 1D slab problem using different quantities of medium time steps per large time step ( $N_{TM}$ ) and number of small time steps per medium time step ( $N_{TS}$ ). The fit corresponding to  $N_{TM} \times N_{TS} = 155$  is added in red.

From the figure it appears that the stability of the algorithm is not tied to the absolute values of  $N_{TM}$  and  $N_{TS}$ , but instead to the product of their values. Specifically, the current implementation seems to

produce stable results for  $N_{TM} \times N_{TS} \geq 155$ , which is illustrated by the red fit in Figure 5.6. This implies that the size of the smallest time scale  $\Delta t_S = \Delta t_B / (N_{TM} \times N_{TS})$  indicates the stability of the algorithm. Indeed, in this research it was found that when the value of  $\Delta t_B$  was lowered from  $5.0 \times 10^{-2}$  s to  $5.0 \times 10^{-3}$  s, then the stability threshold for the timescale ratios also decreased from 155 to 15.5. As the  $\Delta t_S$  variable is only used in practice when updating the neutron and precursor amplitudes (as shown in Equation 3.52) in combination with the prompt neutron generation time  $\Lambda$ , the instability is attributed to the inability of too large time steps  $\Delta t_S$  to accurately model the quick generation of prompt neutrons. Further research can confirm or disprove this conjecture.

## 5.2. Transient Coupled Simulations

### 5.2.1. Problem Description

Now that the accuracy and the stability of the Predictor-Corrector Quasi-Static Method has been investigated, the capabilities of the FM-LBM tool to model transient multiphysics can be assessed. In the final phase of the benchmark by Tiberge[60], denoted as Step 2.1, the transient behaviour of a perturbed reactor is simulated. Starting from the steady-state fully coupled solution of Step 1.4 with  $U_{lid} = 0.5 \text{ m s}^{-1}$  and  $P = 1 \text{ GW}$ , the volumetric heat transfer coefficient  $\gamma_{th}$  is uniformly perturbed according to a sine wave of amplitude 10%, i.e.:

$$\gamma_{th}(t) = \gamma_{th,0} [1 + 0.1 \sin(2\pi f t)] . \quad (5.6)$$

By varying  $f \in \{0.0125, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8\} \text{ Hz}$ , the system response can be analysed in the frequency domain. The observables in this final step are the phase shift and the gain of the power as a function of the perturbation frequency:

$$\text{Gain} = \frac{(P_{max} - P_{avg})/P_{avg}}{(\gamma_{th,max} - \gamma_{th,avg})/\gamma_{th,avg}} , \quad (5.7)$$

$$\text{Phase Shift} = 2\pi f (t_{\gamma_{th,max}} - t_{P_{max}}) . \quad (5.8)$$

Here,  $P_{max}$  is the maximum of the oscillating power,  $P_{avg}$  the time-averaged power, corresponding to the steady state power of 1 GW. Similarly, the denominator in the gain equation corresponds to the maximum deviation of the heat transfer coefficient compared to its average value  $\gamma_{th,avg} = \gamma_{th,0}$ , which amounts to 10%.

### 5.2.2. Adjoint Simulations

As explained in the previous section, before the transient case of the coupled multiphysics can be simulated, one must first obtain the steady state solutions for the adjoint neutron flux  $\Phi_g^\dagger$  to use as a weighting function for the PC-QSM. Again, this quantity will be obtained by the adjoint FM-LBM implementation as described in Section 3.6.2. However, in contrast to the 1D slab problem which modelled isothermal and static fuel, for the coupled system of a molten salt reactor the effects of fuel convection and salt expansion on the adjoint quantities must be incorporated. Therefore the steady state solutions for the velocity  $\mathbf{u}$  and enthalpy  $h$  are used as fixed values in the calculations for  $\Phi_g^\dagger$  and  $C_d^\dagger$ .

To highlight the differences between the forward and adjoint solutions for the neutron flux, the total fission rate density  $\sum_g \Sigma_{f,g} \phi_g$  and delayed neutron source  $\sum_d \lambda_d C_d$  are illustrated in Figure 5.7. On the other hand, Figure 5.8 presents the adjoint versions of the fission rate density  $\sum_g \Sigma_{f,g} \Phi_g^\dagger$  and delayed neutron source  $\sum_d \lambda_d C_d^\dagger$ . Focusing first on the figures of the forward variables, it appears the fission density stays roughly centred. As explained earlier, the fast-moving neutrons are not influenced by the slow convection of the flow, while the precursors do experience convective transport due to the lid-driven circulation to the right as well as the upward buoyant effects. This is illustrated by the skewed top-right hotspot of the precursors in Figure 5.7. The forward neutron flux only feels the asymmetry of the precursors weakly through the delayed neutrons source  $\chi_g^d \sum_d \lambda_d C_d$  in the NDE, which contributes a

small fraction of the total fission source. This explains the near-symmetric fission density for circulating fuel, which was also reported in Step 1.1 of the Tiberia benchmark (see Section 4.3.1).

In the adjoint problem the situation is different. The adjoint neutron diffusion equation has no convective terms, just like the forward case. However, the adjoint fission source is coupled more strongly to the adjoint precursor concentration  $C_d^\dagger$  through the term  $(\nu\Sigma_f)_g \sum_d \beta_d C_d^\dagger / k_{eff}^\dagger$ . Moreover, the adjoint precursor concentrations are convected with the reversed flow. As explained in Chapter 2, the precursor importance is highest at locations where the generation of precursors contributes the most to the overall fission chain reaction. In circulating fuel, this corresponds to the source location of precursors that, by the time they decay and emit neutrons, have travelled downstream to the fission hotspot in the centre. Therefore the precursor importance is computed upstream, which is encoded by the negative convection  $-\mathbf{u} \cdot \nabla C_d^\dagger$ . The adjoint flux  $\Phi_g^\dagger$  is then sourced both by the (transposed) scattering and prompt fission (which would keep it centred) and by the  $(\nu\Sigma_f)_g \sum_d \beta_d C_d^\dagger / k_{eff}^\dagger$  term, which is spatially located with  $C_d^\dagger$ . Neutrons at these locations have the biggest contribution to the production of high-importance precursors, and so their importance is also greater. Therefore the adjoint flux is highest wherever the precursor importance is concentrated, which is shown in Figure 5.8.

From these figures it can be concluded that the difference between the adjoint quantities ( $\Phi_g^\dagger$  and  $C_d^\dagger$ ) and their forward counterparts is not only caused by the reversed scattering and fission terms in the adjoint multi-group NDE, as explained in the previous section, but also by the upstream convection term in the case of circulating fuel. Particularly the adjoint neutron importance is more strongly affected by the fuel convection than the forward neutron flux due to the strong coupling of the adjoint fission source with the precursor importance. The significant inequality of the steady state solutions for  $\phi_g$  and  $\Phi_g^\dagger$  in the case of circulating fuel is another indication that the accuracy of the PC-QSM for transient multiphysics simulations is heavily dependent on the choice of weighting factor.

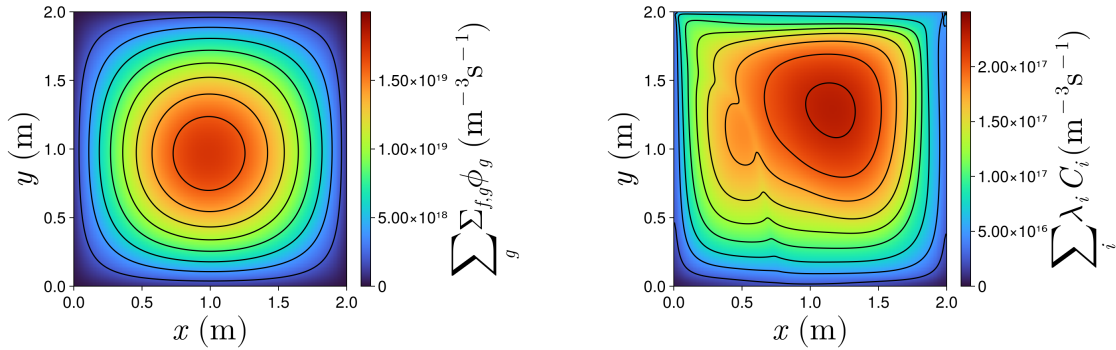


**Figure 5.7:** The forward fission rate density (left) and delayed neutron source (right) for Step 1.4 of the Tiberia benchmark case. Please note that the colorbar scale for each graph is different.

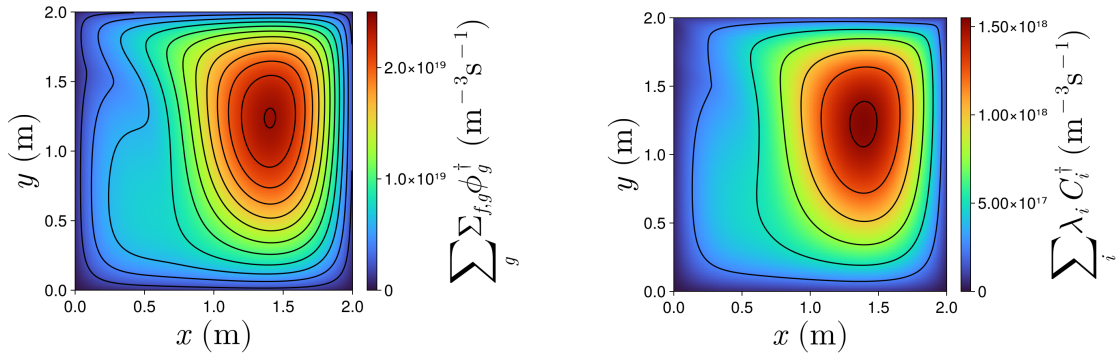


**Figure 5.8:** The adjoint fission rate density (left) and delayed neutron source (right) for Step 1.4 of the Tiberia benchmark case. Please note that the colorbar scale for each graph is different.

### 5.2.3. Transient Multiphysics Simulations

Having obtained the steady state solution for the adjoint neutron flux, the transient behaviour of the coupled multiphysics can finally be modelled for the perturbed reactor. Figure 5.9 illustrates the power output response of the system to the oscillating motion of the heat transfer coefficient. In this figure, the normalised deviations of  $P(t)$  and  $\gamma_{th}(t)$  are displayed, defined as

$$\begin{aligned}\tilde{\gamma}_{th}(t) &= \frac{\gamma_{th}(t) - \gamma_{th,avg}}{\gamma_{th,avg}} \\ \tilde{P}(t) &= \frac{P(t) - P_{avg}}{P_{avg}}.\end{aligned}\tag{5.9}$$

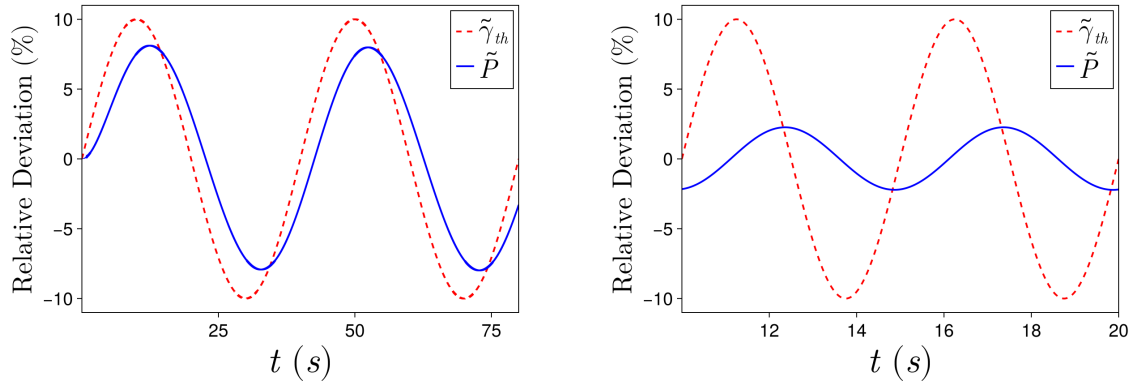


**Figure 5.9:** Snapshots of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberge benchmark case. Left:  $f = 0.025$  Hz. Right:  $f = 0.2$  Hz. Note the different scaling of the  $x$ -axis for each graph.

The figure shows how the power output experiences a delayed perturbation as a result of the imposed oscillation of the heat transfer coefficient. As  $\gamma_{th}$  increases, the contribution of the volumetric heat sink dominates the heat equation (Eq. 2.5) which has a negative effect on the temperature. This decreases the feedback of the salt expansion on the macroscopic cross sections and neutron diffusion coefficients, according to Equations 2.71 and 2.72. This in turn has a positive effect on the scalar neutron flux and the rate of fission, which increases the power output. For decreasing  $\gamma_{th}$ , the opposite cascade of effects arises, yielding higher temperatures, greater salt expansion feedback, lower fission rates, and consequently lower power output. Moreover, comparing the system response for  $f = 0.025$  Hz and  $f = 0.2$  Hz, it can be seen that higher perturbation frequencies result in a larger phase shift and a smaller gain of the power output. Snapshots of the transient simulation results for the other perturbation frequencies are shown in Appendix C.

To quantify the relation between perturbation frequency and power output, the system response is studied in the frequency domain. Figures 5.10 and 5.11 respectively present the normalised power gain and phase shift corresponding to different perturbation frequencies in Bode plots. Like with the earlier steps, the results of this research are shown together with the results of the reference codes, as well as the results of Polderman[42] and Pijls[43] who both implemented the PC-QSM for transient neutronics using the steady state solution for the forward flux  $\phi_0$  as a weight function. All cases were simulated with a PK timescale configuration of  $N_{TM} = \Delta t_B / \Delta t_M = 10$  and  $N_{TS} = \Delta t_M / \Delta t_S = 100$ , which is assumed to produce stable results following the conclusion from the previous section. From the figures it can be seen that the system response behaves according to what is expected from a theoretical viewpoint. At low frequencies, the power output follows the oscillation of the heat transfer coefficient very closely, with gain close to 1 and minimal phase shift, as the slow oscillation allows the delayed neutron precursors to reach an equilibrium in the perturbed system. On the other hand, the contribution of the delayed neutrons is suppressed for higher frequencies, as the long half-lives of some precursor families make them less sensitive to the temperature reactivity feedback. This results in a lower response amplitude and therefore a decreasing gain. Furthermore, in the case of high

frequencies the convective and diffusive terms in the time-dependent differential equations for heat, neutron flux, and precursor concentration are dominated by the high time derivative. Therefore the power output approaches a cosine wave with maximum phase shift of  $-90^\circ$ .

Comparing the result of this research to the reference codes and the previous LBM studies, it can be seen that the system response follows the same trends as the reference code results, with decreasing gain and phase shift for increasing frequencies. Moreover, the current FM-LBM algorithm is in equally good or even better agreement with the reference codes compared to the previous LBM studies. Especially the relatively high discrepancy of the gain at low frequencies reported by Pijls' FM-LBM implementation seems to be resolved, as well as the large offset in phase shift for high frequencies. The biggest discrepancies between the current research and the reference results seem to be for the phase shift. This is not entirely surprising, as the deviation of results between the reference codes is also greatest there. From these results it can be concluded that the improvements established for the 1D slab, namely the use of the adjoint neutron flux as the proper weight function in the PC-QSM, carry over to the accuracy of the fully coupled reactor case. This completes the validation of the FM-LBM multiphysics tool for transient multiphysics.

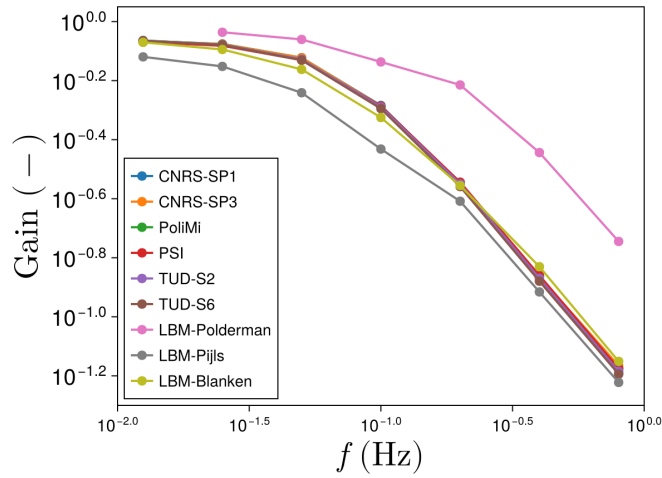


**Figure 5.10:** Gain of the frequency response of the power production  $P$  of the transient Tiberger benchmark as a function of an oscillating heat transfer coefficient  $\gamma_{th}$ . The simulation results are shown alongside the data from the reference codes and previous LBM studies.

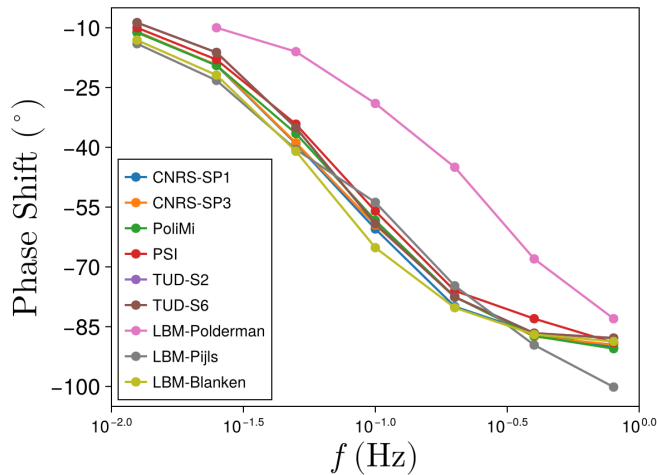


**Figure 5.11:** Phase shift of the frequency response of the power production  $P$  of the transient Tiberger benchmark as a function of an oscillating heat transfer coefficient  $\gamma_{th}$ . The simulation results are shown alongside the data from the reference codes and previous LBM studies.

## Conclusion & Recommendations

In this research, a simulation tool was developed to model the complex multiphysics behaviour of a Molten Salt Fast Reactor (MSFR) core. The tool implements a GPU-accelerated Filter-Matrix Lattice Boltzmann Method (FM-LBM) algorithm for thermal hydraulics, neutron diffusion, and precursor advection-diffusion in a molten salt. The algorithm initially consisted of 16 distribution functions (momentum, enthalpy, 6 neutron groups, and 8 precursor families), that evolve through repeated collision and streaming steps on a regular lattice grid, where the collisions are performed using matrix multiplications in which non-physical terms are filtered out to increase the stability of the simulation.

The steady-state FM-LBM algorithm was validated using the various single-physics and multiphysics steps of the Tiberga benchmark, which models a 2D square reactor with moving top lid and normalised power. The simulation results of different observables along the centrelines of the domain were compared with values reported by the reference codes. For all simulations, the average discrepancy of all observables was measured below 1%. Furthermore, the reactivity differences reported in each step were also in better agreement with the reference codes than in any of the previous LBM studies carried out by the research group. These results show that the current implementation of the FM-LBM tool yields satisfactory results for coupled steady-state simulations, and form the foundation on which the three research goals of this thesis are addressed: the handling of complex geometries in Section 6.1, the computational performance of the tool in Section 6.2, and the modelling of transient multiphysics in Section 6.3. Finally, Section 6.4 offers recommendations for future research.

### 6.1. Complex Geometries

As different versions of the Lattice Boltzmann solver for multiphysics simulations had already been investigated in previous work by the research group, the current work benefited considerably from their findings. A notable addition to the existing framework is the creation of a novel lookup table (LUT) containing the pull-streaming information for each node in the (stationary) computational domain. With this lookup table, the streaming and boundary kernels no longer need to calculate whether a node is fluid, solid, or a boundary node every time step, but instead fetch the source node and boundary distance through simple and quick array reads. This lookup table approach further enables the modelling of fluid flow around complex geometries. Discretised shapes can be added modularly during the initialisation of the computational domain, after which the CPU constructs the LUT once as a preprocessing step, before transferring all data to the GPU device. A further consequence of this approach is the elimination of ghost node padding often employed in LBM implementations.

The complex-geometry capabilities were validated against the benchmark described by Breuer, in which a square obstacle is placed in a plane channel flow. Simulations reproduced the reference solutions accurately in both the steady and the unsteady flow regimes; in the steady regime the computed recirculation lengths agreed with the reported linear fit to within a mean absolute percentage error of 3.16%. Furthermore, the interpolated bounce-back scheme was tested successfully on a shifted grid, confirming that boundaries that do not align with the lattice are treated with sufficient accuracy.

It can therefore be concluded that the FM-LBM multiphysics tool can incorporate complex geometries and reproduce accurate flow simulations of the momentum field through the LUT-based pull-streaming approach. The validation in this work was restricted to the momentum field, but the same modular, region-based approach extends naturally to the other physical fields.

## 6.2. Computational Performance

This research also assessed the computational performance of the FM-LBM tool. For the single-physics steady-state simulations, the speed of lattice updates was measured for different grid sizes. The simulations for the momentum field achieve a performance of 283.41 million lattice updates per second (MLUPS), while the simulations for neutronics and for the enthalpy field achieve 80.62 MLUPS and 1166.30 MLUPS respectively.

Compared to previous FM-LBM multiphysics study by Pijls, these performances mark an improvement ranging from roughly 500% for the neutronics simulation to nearly 1100% for the enthalpy simulation. This improvement is attributed primarily to the pull-stream lookup table and the use of stack-allocated static arrays for the filter-matrix multiplications. When compared to studies focused specifically on optimising GPU-accelerated LBM implementations, which reported performances of up to 1200 MLUPS, the performances of this research are within a reasonable range. It must be noted however, that this comparison is not perfect, as those studies modelled a single velocity field using a D3Q19 scheme with the simpler BGK operator and were executed on different hardware.

It can be concluded that the additional GPU optimisation techniques achieved significant improvements in computational performance relative to previous work, and that the tool performs competitively with GPU-accelerated LBM implementations reported in the literature.

## 6.3. Transient FM-LBM Model Development

For transient multiphysics simulations, the FM-LBM algorithm is altered to model the neutronics and thermal hydraulics at different timescales. As in previous work, the Predictor-Corrector Quasi-Static Method (PC-QSM) is employed, in which the neutron scalar flux is factorised into a shape function and an amplitude function. While the shape function is computed using the LBM solver, the amplitude function is calculated using the point-kinetic equations. The parameters used in these equations are calculated from the (linearly interpolated) shape function.

For the calculation of the point-kinetic parameters, the PC-QSM uses the solution for the adjoint neutron flux as a weight function. This is partly to ensure unique solutions to the shape and amplitude functions. To this end, two new sets of distribution functions were introduced to model the adjoint versions of the steady-state neutron diffusion equation and the precursor advection-diffusion equation. The solution of this first equation, the adjoint neutron flux, is a measure for the importance of a neutron in its role in the entire chain reaction. This quantity is sometimes used in nuclear reactor physics to study the effect of reactivity perturbations on the system.

In previous LBM studies for transient neutronics, the solution for the 'forward' neutron flux was used as the PC-QSM weight function instead, effectively assuming the two solutions are equivalent. However, simulations on a simplified 1D reactor slab performed in this research showed that, while the forward and adjoint steady-state solutions are essentially equal for the one-group, static-fuel case, they differ significantly in the case of multiple neutron energy groups. Moreover, the choice of appropriate weight function has a large effect on the accuracy of the transient behaviour of the slab in the case of a reactivity insertion. The forward-weighted amplitude overestimates the effective reactivity, while the adjoint-weighted amplitude remains close to the reference solution. This finding provides an explanation to inconsistent simulation results reported in previous studies. Furthermore, analysis on the PC-QSM indicated that its stability depends on the size of the smallest time step, over which the amplitude functions are updated through the point-kinetic equations. For time steps too large the algorithm cannot capture the short lifetime of prompt neutrons, resulting in unstable simulations.

Finally, the transient FM-LBM multiphysics solver was applied to the last step of the Tiberga benchmark, which measures the power output of the reactor as a response to oscillatory perturbations of the volumetric heat transfer coefficient. Simulating the adjoint flux as a preliminary step showed significant

differences compared to the forward scalar flux solution, consistent with the 1D slab problem. More specifically, the adjoint flux was coupled more strongly to the fuel salt convection due to increased importance of circulating precursors. This further affirms the importance of calculating the adjoint neutron flux as a weight function for the PC-QSM. Using these results to simulate the system response across different perturbation frequencies, the current implementation shows closer agreement with the results reported by the reference codes than the previous LBM studies of Polderman and Pijls. Specifically the high gain discrepancies for low frequencies and high phase shift discrepancies for high frequencies reported in earlier work were significantly reduced. These improvements are attributed mainly to the correct choice of the adjoint neutron flux as a weight function.

It can therefore be concluded that the FM-LBM multiphysics tool can be adapted to model the transient behaviour of an MSFR core with accuracy and stability, that the adjoint neutron diffusion and precursor equations can be implemented within the existing FM-LBM framework, and that using the adjoint flux as the weight function is necessary for producing physically consistent transient results in the presence of multiple energy groups or circulating fuel.

## 6.4. Recommendations for Future Research

For future research, the following extensions to the model are recommended:

- The effect of Doppler broadening due to increased temperature on the macroscopic cross sections can be studied. As the temperature dependence of the neutronics is already present in the model in the form of the salt expansion feedback, this additional effect can be easily incorporated. Similarly, the decay heat generated by precursors can be added as a source term to the heat equation to increase the realism of the neutronics model.
- Turbulence modelling was omitted from the Tiberger benchmark due to its computational complexity. However, for large-scale modelling of molten salt reactors, turbulent flow becomes a necessity. Turbulence models such as  $k - \epsilon$  or  $k - \omega$  require the additional solving of only 2 scalar quantities and should therefore be relatively straightforward to implement, although the instabilities reported in earlier turbulence work within the group (see Chapter 1) suggest this is not trivial.
- Further improvements & extensions to the current implementation of the pull-stream lookup table can be made. For more complicated domains with multiple types of boundaries, extra boundary node classes can be easily defined. Aside from interpolated bounce-back, other advanced boundary methods can be examined, such as the partially saturated bounce-back method, or direction-dependent bounce-back methods. For domains where the accessible regions for the physical fields differ (e.g. control rods), the use of multiple LUTs might prove useful.
- The current research used the neutron diffusion equation to model the transport of neutrons in the reactor core. This implementation relies on a number of assumptions, including isotropic neutron sources, slow variations of the neutron current, as well as the discretisation of neutrons into energy groups. The NDE description becomes less accurate in regions with large neutron density gradients, such as material discontinuities, localised sources, or strongly absorbing media. Future research can investigate how the implementation of the more complete neutron *transport* equation in the FM-LBM framework (as well as its angular discretisation approximations) can improve the model's accuracy in these cases.
- Other benchmarks can be studied to validate the multiphysics tool in two- and three-dimensional geometries, such as a single MSFR recirculation loop[17, 80] or the complete toroidal-shaped MSFR geometry proposed in the EVOL project[81, 82]. Different (adaptive) mesh refinements might prove useful to mitigate the large costs of 3D computations. Additionally, a multi-GPU approach can be implemented to further accelerate the simulations.

# References

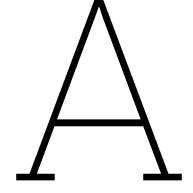
- [1] U.S. Department of Energy. *The First Reactor*. 1982.
- [2] World Nuclear Association. *Nuclear Power in the World Today*. 2026. URL: <https://world-nuclear.org/information-library/current-and-future-generation/nuclear-power-in-the-world-today> (visited on 05/27/2026).
- [3] International Energy Agency. *The Path to a New Era for Nuclear Energy*. Fuel report. Licence: CC BY 4.0. Paris: International Energy Agency (IEA), Jan. 16, 2025. URL: <https://www.iaea.org/reports/the-path-to-a-new-era-for-nuclear-energy> (visited on 10/30/2025).
- [4] Dutch Government. *NEO NL opgericht voor realisatie nieuwe kerncentrales*. 2026. URL: <https://www.overkernenergie.nl/actueel/nieuws/2026/02/16/neo-nl-opgericht-voor-realisatie-nieuwe-kerncentrales> (visited on 05/27/2026).
- [5] International Energy Agency. *Energy and AI*. report. Licence: CC BY 4.0. Paris: IEA, 2025. URL: <https://www.iaea.org/reports/energy-and-ai>.
- [6] Ersan Kabalci, Aydin Boyar, and Yasin Kabalci. “2 - Centralized power generation”. In: *Hybrid Renewable Energy Systems and Microgrids*. Ed. by Ersan Kabalci. Academic Press, 2021, pp. 47–72. ISBN: 978-0-12-821724-5. DOI: <https://doi.org/10.1016/B978-0-12-821724-5.00002-7>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128217245000027>.
- [7] Yafei Wang et al. “Chapter 9 - Molten salt reactors”. In: *Nuclear Power Reactor Designs*. Ed. by Jun Wang, Sola Talabi, and Sama Bilbao y Leon. Academic Press, 2024, pp. 163–183. ISBN: 978-0-323-99880-2. DOI: <https://doi.org/10.1016/B978-0-323-99880-2.00009-6>. URL: <https://www.sciencedirect.com/science/article/pii/B9780323998802000096>.
- [8] M. Allibert et al. “Homogeneous Molten Salt Reactors (MSRs): The Molten Salt Fast Reactor (MSFR) concept”. In: *Handbook of Generation IV Nuclear Reactors. Second edition* (2023).
- [9] World Nuclear Association. *Thorium*. 2024. URL: <https://world-nuclear.org/information-library/current-and-future-generation/thorium> (visited on 05/27/2026).
- [10] Pablo Rubiolo et al. “High temperature thermal hydraulics modeling of a molten salt: application to a molten salt fast reactor (MSFR)”. In: *ESAIM: Proceedings and Surveys* 58 (Nov. 2017), pp. 98–117. DOI: 10.1051/proc/201758098.
- [11] W B Cottrell et al. *OPERATION OF THE AIRCRAFT REACTOR EXPERIMENT*. Technical Report. United States: Oak Ridge National Lab., Tenn., Sept. 1955. DOI: 10.2172/4237975. URL: <https://www.osti.gov/biblio/4237975>.
- [12] Paul N Haubenreich and JR Engel. “Experience with the molten-salt reactor experiment”. In: *Nuclear Applications and technology* 8.2 (1970), pp. 118–136.
- [13] J. Krepel. *Overview and Update of MSR activities within GIF*. 2025. URL: <https://www.gen-4.org/resources/webinars/education-and-training-series-97-overview-and-update-msr-activities-within-gif> (visited on 10/29/2025).
- [14] Jianhui Wu et al. “A Review of Molten Salt Reactor Multi-Physics Coupling Models and Development Prospects”. In: *Energies* 15.21 (2022). ISSN: 1996-1073. DOI: 10.3390/en15218296. URL: <https://www.mdpi.com/1996-1073/15/21/8296>.
- [15] Carlo Fiorina et al. “GeN-Foam: a novel OpenFOAM® based multi-physics solver for 2D/3D transient analysis of nuclear reactors”. In: *Nuclear Engineering and Design* 294 (2015), pp. 24–37.
- [16] Marco Tiberghie, Danny Lathouwers, and Jan Leen Kloosterman. “A discontinuous Galerkin FEM multiphysics solver for the Molten Salt Fast Reactor”. In: *International Conference on Mathematics and Computational Methods applied to Nuclear Science and Engineering (M&C 2019)*, Portland, OR, USA. 2019.

- [17] Marco Tiberghie, Danny Lathouwers, and Jan Leen Kloosterman. "A multi-physics solver for liquid-fueled fast systems based on the discontinuous Galerkin FEM discretization". In: *Progress in Nuclear Energy* 127 (2020), p. 103427.
- [18] Timm Krüger et al. *The lattice Boltzmann method*. Vol. 10. 978-3. Springer, 2017.
- [19] Xiaoyi He, Shiyi Chen, and Gary D Doolen. "A novel thermal model for the lattice Boltzmann method in incompressible limit". In: *Journal of computational physics* 146.1 (1998), pp. 282–300.
- [20] Dipankar Chatterjee. "An enthalpy-based thermal lattice Boltzmann model for non-isothermal systems". In: *EPL (Europhysics Letters)* 86.1 (2009), p. 14004.
- [21] Dominique d'Humières. "Generalized lattice-Boltzmann equations". In: *Rarefied gas dynamics* (1992).
- [22] JA Somers. "Direct simulation of fluid flow with cellular automata and the lattice-Boltzmann equation". In: *Applied Scientific Research* 51.1 (1993), pp. 127–133.
- [23] Congshan Zhuo, Chengwen Zhong, and Jun Cao. "Filter-matrix lattice Boltzmann model for incompressible thermal flows". In: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 85.4 (2012), p. 046703.
- [24] Congshan Zhuo and Chengwen Zhong. "LES-based filter-matrix lattice Boltzmann model for simulating turbulent natural convection in a square cavity". In: *International Journal of Heat and Fluid Flow* 42 (2013), pp. 10–22.
- [25] Congshan Zhuo and Chengwen Zhong. "Filter-matrix lattice Boltzmann model for microchannel gas flows". In: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 88.5 (2013), p. 053311.
- [26] Congshan Zhuo and Chengwen Zhong. "LES-based filter-matrix lattice Boltzmann model for simulating fully developed turbulent channel flow". In: *International Journal of Computational Fluid Dynamics* 30.7-10 (2016), pp. 543–553.
- [27] Celeke Bus, Thorben Besseling, and Martin Rohde. "A Filter-Matrix Lattice-Boltzmann Methodology for Convective Melting and Solidification". In: *International Journal for Numerical Methods in Fluids* 97.10 (2025), pp. 1379–1393.
- [28] Tom Entes. "Development and Integration of a GPU-Accelerated Lattice Boltzmann Simulation Tool for Thermal Hydraulics with Neutronics for Multiphysics Simulations in Molten Salt Fast Reactor Cores". MSc. Thesis. TU Delft, 2025.
- [29] Viktor Sundström. "Deriving Filter Matrix Methods for Lattice Boltzmann Simulations". MSc. Thesis. Chalmers University of Technology, 2025.
- [30] Twan Terpstra. "Development of a Turbulent Lattice Boltzmann Tool for Multiphysics Simulations of a Molten Salt Fast Reactor Core". MSc. Thesis. TU Delft, 2026.
- [31] Bernard Erasmus. "The Lattice Boltzmann Method applied to linear particle transport". PhD thesis. North-West University, 2012.
- [32] Hitesh Bindra and Dhiraj V Patil. "Radiative or neutron transport modeling using a lattice Boltzmann equation framework". In: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 86.1 (2012), p. 016706.
- [33] G Agarwal et al. "Solution of Neutron Transport Equation for Criticality Estimation Using Lattice Boltzmann Method". In: *Proceedings of the Transactions of the American Nuclear Society, Washington, DC, USA* (2015), pp. 8–12.
- [34] Yahui Wang, Yu Ma, and Ming Xie. "High-order lattice Boltzmann method for multi-group neutron diffusion solution". In: *Progress in Nuclear Energy* 110 (2019), pp. 341–353.
- [35] Yahui Wang, Liming Yan, and Yu Ma. "Lattice Boltzmann solution of the transient Boltzmann transport equation in radiative and neutron transport". In: *Physical Review E* 95.6 (2017), p. 063313.
- [36] Yahui Wang and Yu Ma. "Unstructured finite-volume lattice Boltzmann method for the multi-group SP3 simulation". In: *Annals of Nuclear Energy* 170 (2022), p. 109012.
- [37] Yahui Wang, Ming Xie, and Yu Ma. "Analysis of the multi-physics approach using the unified lattice Boltzmann framework". In: *Annals of Nuclear Energy* 143 (2020), p. 107500.

- [38] Yahui Wang and Yu Ma. "IbmNTH: A unified lattice Boltzmann framework for coupled neutronics-thermal-hydraulics analysis". In: *Annals of Nuclear Energy* 166 (2022), p. 108750.
- [39] Yahui Wang, Yu Ma, and Ming Xie. "GPU accelerated lattice Boltzmann method in neutron kinetics problems". In: *Annals of Nuclear Energy* 129 (2019), pp. 350–365. ISSN: 0306-4549. DOI: <https://doi.org/10.1016/j.anucene.2019.02.009>. URL: <https://www.sciencedirect.com/science/article/pii/S0306454919300763>.
- [40] Sandra Dulla, Piero Ravetto, and MM Rostagno. "Neutron kinetics of fluid–fuel systems by the quasi-static method". In: *Annals of Nuclear Energy* 31.15 (2004), pp. 1709–1733.
- [41] Sandra Dulla, Ernest H Mund, and Piero Ravetto. "The quasi-static method revisited". In: *Progress in Nuclear Energy* 50.8 (2008), pp. 908–920.
- [42] Coco Polderman. "Developing a multi-physics tool for the neutronics-thermal-hydraulics in a molten salt reactor using the lattice Boltzmann method Coco Polderman". MSc. Thesis. TU Delft, 2024.
- [43] Thomas Pijls. "Developing a GPU-Accelerated Filter-Matrix Lattice Boltzmann Multiphysics Tool for the Transient Thermal-Hydraulics and Neutronics of a Molten Salt Fast Reactor Core". MSc. Thesis. TU Delft, 2025.
- [44] Adriano Tiribocchi et al. "Lightweight lattice Boltzmann". In: *The Journal of Chemical Physics* 158.10 (Mar. 2023). ISSN: 1089-7690. DOI: 10.1063/5.0139850. URL: <http://dx.doi.org/10.1063/5.0139850>.
- [45] Khodr Jaber, Ebenezer E Essel, and Pierre E Sullivan. "GPU-native embedding of complex geometries in adaptive octree grids applied to the lattice Boltzmann method". In: *Computer Physics Communications* (2026), p. 110155.
- [46] Yahui Wang, Yu Ma, and Ming Xie. "GPU accelerated lattice Boltzmann method in neutron kinetics problems II: Neutron transport calculation". In: *Annals of Nuclear Energy* 134 (2019), pp. 305–317. ISSN: 0306-4549. DOI: <https://doi.org/10.1016/j.anucene.2019.06.020>. URL: <https://www.sciencedirect.com/science/article/pii/S030645491930338X>.
- [47] Junuthula Narasimha Reddy. *Principles of continuum mechanics: conservation and balance laws with applications*. Cambridge University Press, 2017.
- [48] Richard Haberman. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*. Pearson, 2014.
- [49] Vaughan R Voller, M Cross, and NC Markatos. "An enthalpy method for convection/diffusion phase change". In: *International journal for numerical methods in engineering* 24.1 (1987), pp. 271–284.
- [50] Rongzong Huang, Huiying Wu, and Ping Cheng. "A new lattice Boltzmann model for solid–liquid phase change". In: *International Journal of Heat and Mass Transfer* 59 (2013), pp. 295–301.
- [51] Sydney Chapman and Thomas George Cowling. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge university press, 1990.
- [52] James J Duderstadt and Louis J Hamilton. *Nuclear reactor analysis*. John Wiley and Sons, Inc., New York, 1975.
- [53] Daniel Rozon. *Introduction to nuclear reactor kinetics*. Polytechnic International Press Canada, 1998.
- [54] Karl Otto Ott and Robert J Neuhold. *Introductory nuclear reactor dynamics*. American nuclear society, 1985.
- [55] Giovanni Lapenta, F Mattioda, and Piero Ravetto. "Point kinetic model for fluid fuel systems". In: *Annals of Nuclear Energy* 28.17 (2001), pp. 1759–1772.
- [56] F. Tarantino, R. Jacqmin, and A. Henry. "The Validity of the Point Kinetics Model During Reactor Start-Up". In: *Nuclear Simulation*. Springer-Verlag, 1990.
- [57] Alvin Martin Weinberg and Eugene Paul Wigner. "The Physical Theory of Neutron Chain Reactors". In: (*No Title*) (1958).

- [58] A Gandini. "A generalized perturbation method for bi-linear functionals of the real and adjoint neutron fluxes". In: *Journal of Nuclear Energy* 21.10 (1967), pp. 755–765.
- [59] S.A.H. Feghhi, M. Shahriari, and H. Afarideh. "Calculation of neutron importance function in fissionable assemblies using Monte Carlo method". In: *Annals of Nuclear Energy* 34.6 (2007), pp. 514–520. ISSN: 0306-4549. DOI: <https://doi.org/10.1016/j.anucene.2007.03.002>. URL: <https://www.sciencedirect.com/science/article/pii/S0306454907000643>.
- [60] Marco Tiberghien et al. "Results from a multi-physics numerical benchmark for codes dedicated to molten salt fast reactors". In: *Annals of Nuclear Energy* 142 (2020), p. 107428.
- [61] NVIDIA Corporation. *CUDA Programming Guide*. Version 13.3. Last updated May 27, 2026. 2026. URL: <https://docs.nvidia.com/cuda/cuda-programming-guide/index.html> (visited on 06/10/2026).
- [62] Kyle E Niemeyer and Chih-Jen Sung. "Recent progress and challenges in exploiting graphics processors in computational fluid dynamics". In: *The Journal of Supercomputing* 67.2 (2014), pp. 528–564.
- [63] Lin Shi et al. "A survey of GPU-based medical image computing techniques". In: *Quantitative imaging in medicine and surgery* 2.3 (2012), p. 188.
- [64] Yu-Hui Chen et al. "Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4651–4655.
- [65] Ying Sheng et al. "Flexgen: High-throughput generative inference of large language models with a single gpu". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 31094–31116.
- [66] NVIDIA Corporation. *CUDA C++ Best Practices Guide*. Version 13.2. 2025. URL: <https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html> (visited on 06/10/2026).
- [67] Tommaso Toffoli and Norman H Margolus. "Invertible cellular automata: a review". In: *Physica D: Nonlinear Phenomena* 45.1-3 (1990), pp. 229–253.
- [68] Ting Zhang et al. "General bounce-back scheme for concentration boundary condition in the lattice-Boltzmann method". In: *Phys. Rev. E* 85 (1 Jan. 2012), p. 016701. DOI: 10.1103/PhysRevE.85.016701. URL: <https://link.aps.org/doi/10.1103/PhysRevE.85.016701>.
- [69] E. Buckingham. "On Physically Similar Systems; Illustrations of the Use of Dimensional Equations". In: *Phys. Rev.* 4 (4 Oct. 1914), pp. 345–376. DOI: 10.1103/PhysRev.4.345. URL: <https://link.aps.org/doi/10.1103/PhysRev.4.345>.
- [70] Sandra Dulla, Ernest H Mund, Piero Ravetto, et al. "Accuracy of a predictor-corrector quasi-static method for space-time reactor dynamics". In: *PHYSOR'06* (2006).
- [71] Dominic Caron, Sandra Dulla, and Piero Ravetto. "New aspects in the implementation of the quasi-static method for the solution of neutron diffusion problems in the framework of a nodal method". In: *Annals of Nuclear Energy* 87 (2016), pp. 34–48.
- [72] s.n. *The Julia Language*. <https://docs.julialang.org/en/v1/>. 2025.
- [73] M'hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. "Momentum transfer of a Boltzmann-lattice fluid with boundaries". In: *Physics of fluids* 13.11 (2001), pp. 3452–3459.
- [74] JA Blanco, P Rubiolo, and E Dumonteil. "Neutronic modeling strategies for a liquid fuel transient calculation". In: *EPJ Web of Conferences*. Vol. 247. EDP Sciences. 2021, p. 06013.
- [75] Eric Cervi et al. "Multiphysics analysis of the MSFR helium bubbling system: A comparison between neutron diffusion, SP3 neutron transport and Monte Carlo approaches". In: *Annals of Nuclear Energy* 132 (2019), pp. 227–235.
- [76] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. <https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2>. 2024.
- [77] Nhat-Phuong Tran, Myungho Lee, and Dong Hoon Choi. "Memory-efficient parallelization of 3D lattice Boltzmann flow solver on a GPU". In: *2015 IEEE 22nd International Conference on High Performance Computing (HiPC)*. IEEE. 2015, pp. 315–324.

- [78] Nhat-Phuong Tran, Myungho Lee, and Sugwon Hong. "Performance optimization of 3D lattice Boltzmann flow solver on a GPU". In: *Scientific Programming* 2017.1 (2017), p. 1205892.
- [79] Michael Breuer et al. "Accurate computations of the laminar flow past a square cylinder based on two different methods: lattice-Boltzmann and finite-volume". In: *International journal of heat and fluid flow* 21.2 (2000), pp. 186–196.
- [80] Carlo Fiorina et al. "Modelling and analysis of the MSFR transient behaviour". In: *Annals of Nuclear Energy* 64 (2014), pp. 485–498.
- [81] M. Brovchenko et al. *Optimization of the Pre-Conceptual Design of the MSFR—EVOL Project*. Deliverable D2.2. EVOL Project, 2013.
- [82] H Rouch et al. "Preliminary thermal–hydraulic core design of the Molten Salt Fast Reactor (MSFR)". In: *Annals of Nuclear Energy* 64 (2014), pp. 449–456.



# Neutronics Data from the Tiberga Benchmark

This appendix contains the data for the six neutron energy groups and the eight precursor families, as used in the multiphysics simulations of the Tiberga benchmark as well as the 1D slab simulations. All data was retrieved from [60].

**Table A.1:** Definition of the neutron energy groups, along with the total and fission cross sections.

| Group, $g$ | Upper Energy Bound (MeV) | $\Sigma_{t,g}$ ( $\text{m}^{-1}$ ) | $\Sigma_{f,g}$ ( $\text{m}^{-1}$ ) |
|------------|--------------------------|------------------------------------|------------------------------------|
| 1          | $2.000 \times 10^1$      | $1.65512 \times 10^1$              | $1.11309 \times 10^{-1}$           |
| 2          | $2.231 \times 10^0$      | $2.17253 \times 10^1$              | $1.08682 \times 10^{-1}$           |
| 3          | $4.979 \times 10^{-1}$   | $3.18009 \times 10^1$              | $1.52219 \times 10^{-1}$           |
| 4          | $2.479 \times 10^{-2}$   | $2.42093 \times 10^1$              | $2.58190 \times 10^{-1}$           |
| 5          | $5.531 \times 10^{-3}$   | $2.50351 \times 10^1$              | $5.36326 \times 10^{-1}$           |
| 6          | $7.485 \times 10^{-4}$   | $2.72159 \times 10^1$              | $1.44917 \times 10^0$              |

**Table A.2:**  $P_0$  scattering cross sections.

| Group<br>$g'$ | $\Sigma_{s,0,g' \rightarrow g}$ ( $\text{m}^{-1}$ )<br>$g$ |                       |                          |                          |                          |                          |
|---------------|------------------------------------------------------------|-----------------------|--------------------------|--------------------------|--------------------------|--------------------------|
|               | 1                                                          | 2                     | 3                        | 4                        | 5                        | 6                        |
| 1             | $1.08476 \times 10^1$                                      | $5.23316 \times 10^0$ | $4.01805 \times 10^{-1}$ | $1.09869 \times 10^{-2}$ | $2.53290 \times 10^{-3}$ | $3.78334 \times 10^{-4}$ |
| 2             | 0                                                          | $1.83666 \times 10^1$ | $3.19138 \times 10^0$    | $2.34218 \times 10^{-3}$ | $2.25259 \times 10^{-4}$ | $2.00405 \times 10^{-5}$ |
| 3             | 0                                                          | 0                     | $2.98293 \times 10^1$    | $1.63470 \times 10^0$    | $1.70575 \times 10^{-3}$ | $1.24625 \times 10^{-4}$ |
| 4             | 0                                                          | 0                     | 0                        | $2.17472 \times 10^1$    | $1.90243 \times 10^0$    | $1.36858 \times 10^{-6}$ |
| 5             | 0                                                          | 0                     | 0                        | 0                        | $2.27173 \times 10^1$    | $1.05885 \times 10^0$    |
| 6             | 0                                                          | 0                     | 0                        | 0                        | 0                        | $2.37826 \times 10^1$    |

**Table A.3:**  $P_1$ -corrected diffusion coefficients and inverse neutron velocities.

| Group, $g$ | $D_g$ (m)                | $1/v_g$ ( $\text{s m}^{-1}$ ) |
|------------|--------------------------|-------------------------------|
| 1          | $2.80064 \times 10^{-2}$ | $4.00367 \times 10^{-8}$      |
| 2          | $1.84021 \times 10^{-2}$ | $7.39846 \times 10^{-8}$      |
| 3          | $1.13110 \times 10^{-2}$ | $2.61748 \times 10^{-7}$      |
| 4          | $1.44786 \times 10^{-2}$ | $6.69270 \times 10^{-7}$      |
| 5          | $1.39750 \times 10^{-2}$ | $1.55845 \times 10^{-6}$      |
| 6          | $1.28252 \times 10^{-2}$ | $4.24462 \times 10^{-6}$      |

**Table A.4:** Average total number of neutrons emitted per fission event, prompt and delayed fission spectra, and average energy emitted per fission event.

| Group, $g$ | $\nu_{n,g}$ (-) | $\chi_g^p$ (-)           | $\chi_g^d$ (-)           | $E_{fission}$ (J)          |
|------------|-----------------|--------------------------|--------------------------|----------------------------|
| 1          | 2.85517         | $3.53812 \times 10^{-1}$ | $4.30325 \times 10^{-3}$ | $3.240722 \times 10^{-11}$ |
| 2          | 2.54532         | $5.23642 \times 10^{-1}$ | $3.87734 \times 10^{-1}$ | $3.240722 \times 10^{-11}$ |
| 3          | 2.43328         | $1.21033 \times 10^{-1}$ | $5.81848 \times 10^{-1}$ | $3.240722 \times 10^{-11}$ |
| 4          | 2.43127         | $1.35457 \times 10^{-3}$ | $2.27947 \times 10^{-2}$ | $3.240722 \times 10^{-11}$ |
| 5          | 2.43330         | $1.51226 \times 10^{-4}$ | $2.89130 \times 10^{-3}$ | $3.240722 \times 10^{-11}$ |
| 6          | 2.43330         | $7.37236 \times 10^{-6}$ | $4.28935 \times 10^{-4}$ | $3.240722 \times 10^{-11}$ |

**Table A.5:** Decay constant and fraction for each family of delayed neutron precursors.

| Family, $d$ | $\lambda_d$ (s <sup>-1</sup> ) | $\beta_d$ (-)            |
|-------------|--------------------------------|--------------------------|
| 1           | $1.24667 \times 10^{-2}$       | $2.33102 \times 10^{-4}$ |
| 2           | $2.82917 \times 10^{-2}$       | $1.03262 \times 10^{-3}$ |
| 3           | $4.25244 \times 10^{-2}$       | $6.81878 \times 10^{-4}$ |
| 4           | $1.33042 \times 10^{-1}$       | $1.37726 \times 10^{-3}$ |
| 5           | $2.92467 \times 10^{-1}$       | $2.14493 \times 10^{-3}$ |
| 6           | $6.66488 \times 10^{-1}$       | $6.40917 \times 10^{-4}$ |
| 7           | $1.63478 \times 10^0$          | $6.05805 \times 10^{-4}$ |
| 8           | $3.55460 \times 10^0$          | $1.66016 \times 10^{-4}$ |

# B

## Heatmaps of Tiberga Benchmark Observables

This appendix is meant to complete Chapter 4 with heatmaps of some observables in the entire computational domain for some steps of the Tiberga benchmark. All simulation results were obtained by the FM-LBM multiphysics tool on a  $200 \times 200$  computational grid. For precursor and heat diffusion simulations,  $Pr = Sc = 1200$ .

### B.1. Single-Physics Simulations

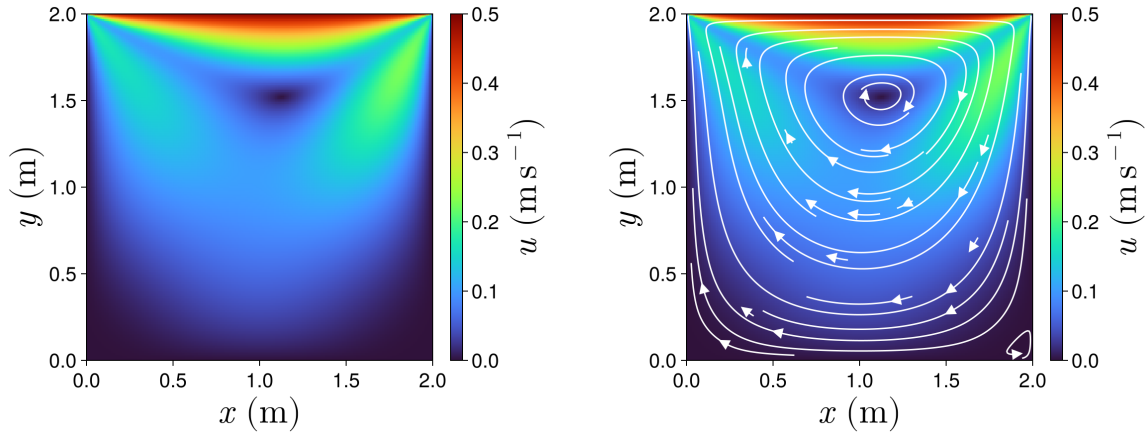


**Figure B.1:** Velocity field for Step 0.1 of the Tiberga benchmark case. Streamlines added in the right figure denote the flow direction of the molten salt.

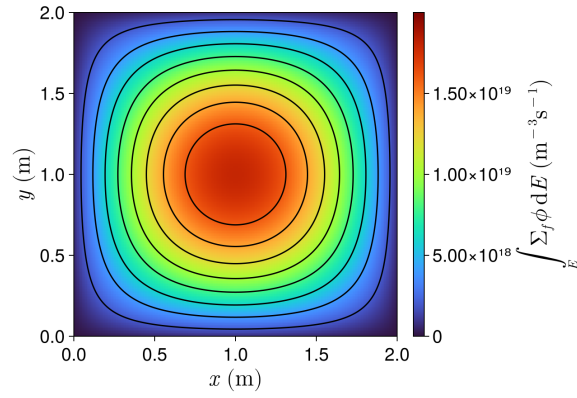


**Figure B.2:** Fission rate density for Step 0.2 of the Tiberger benchmark case. Isoline intervals of  $2.0 \times 10^{18} \text{ m}^{-3} \text{ s}^{-1}$  are added.

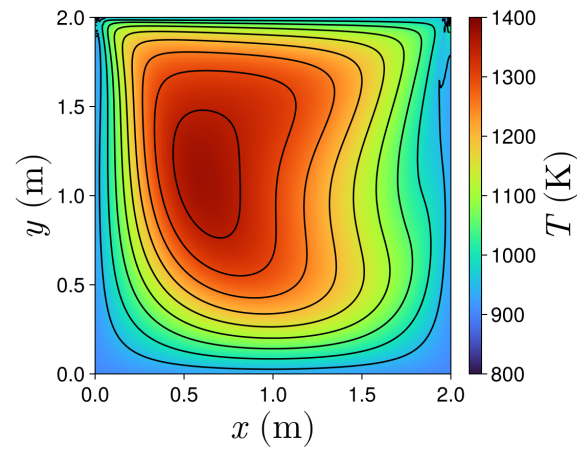


**Figure B.3:** Temperature field for Step 0.3 of the Tiberger benchmark case. Isoline intervals of 50 K are added.

## B.2. Coupled Simulations

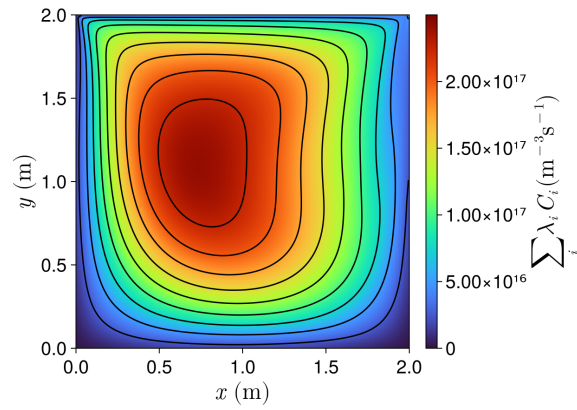


**Figure B.4:** Delayed neutron source for Step 1.1 of the Tiberger benchmark case. Isoline intervals of  $2.5 \times 10^{16} \text{ m}^{-3} \text{ s}^{-1}$  are added.

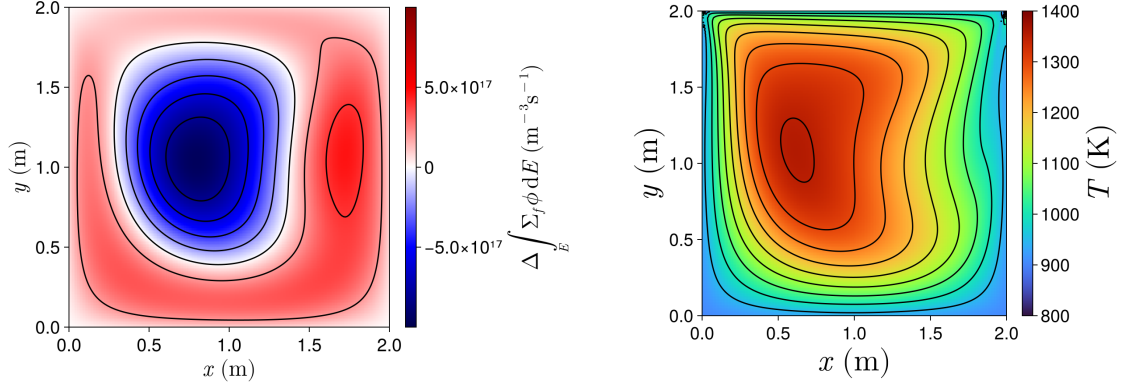


**Figure B.5:** Change of fission rate density (left) and temperature (right) for Step 1.2 of the Tiberga benchmark case. Isoline intervals of  $2.0 \times 10^{17} \text{ m}^{-3} \text{ s}^{-1}$  and 50 K are added to the left and right figure respectively.

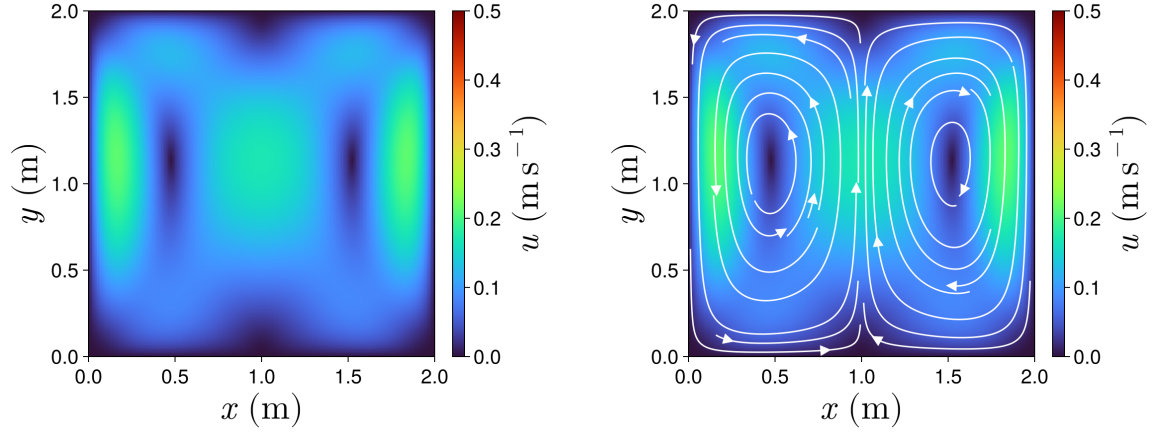


**Figure B.6:** Velocity field for Step 1.3 of the Tiberga benchmark case. Streamlines added in the right figure denote the flow direction of the molten salt.

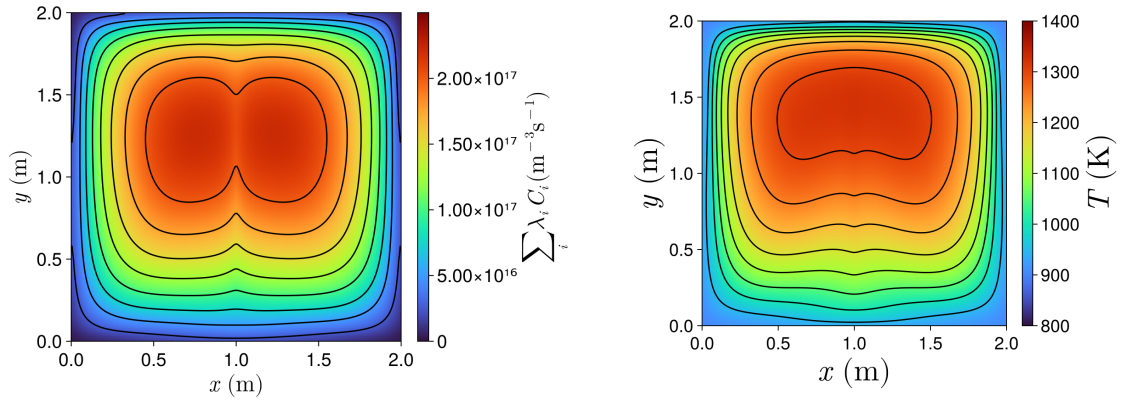


**Figure B.7:** Delayed neutron source (left) and temperature field (right) for Step 1.3 of the Tiberga benchmark case. Isoline intervals of  $2.5 \times 10^{16} \text{ m}^{-3} \text{ s}^{-1}$  and 50 K are added to the left and right figure respectively.

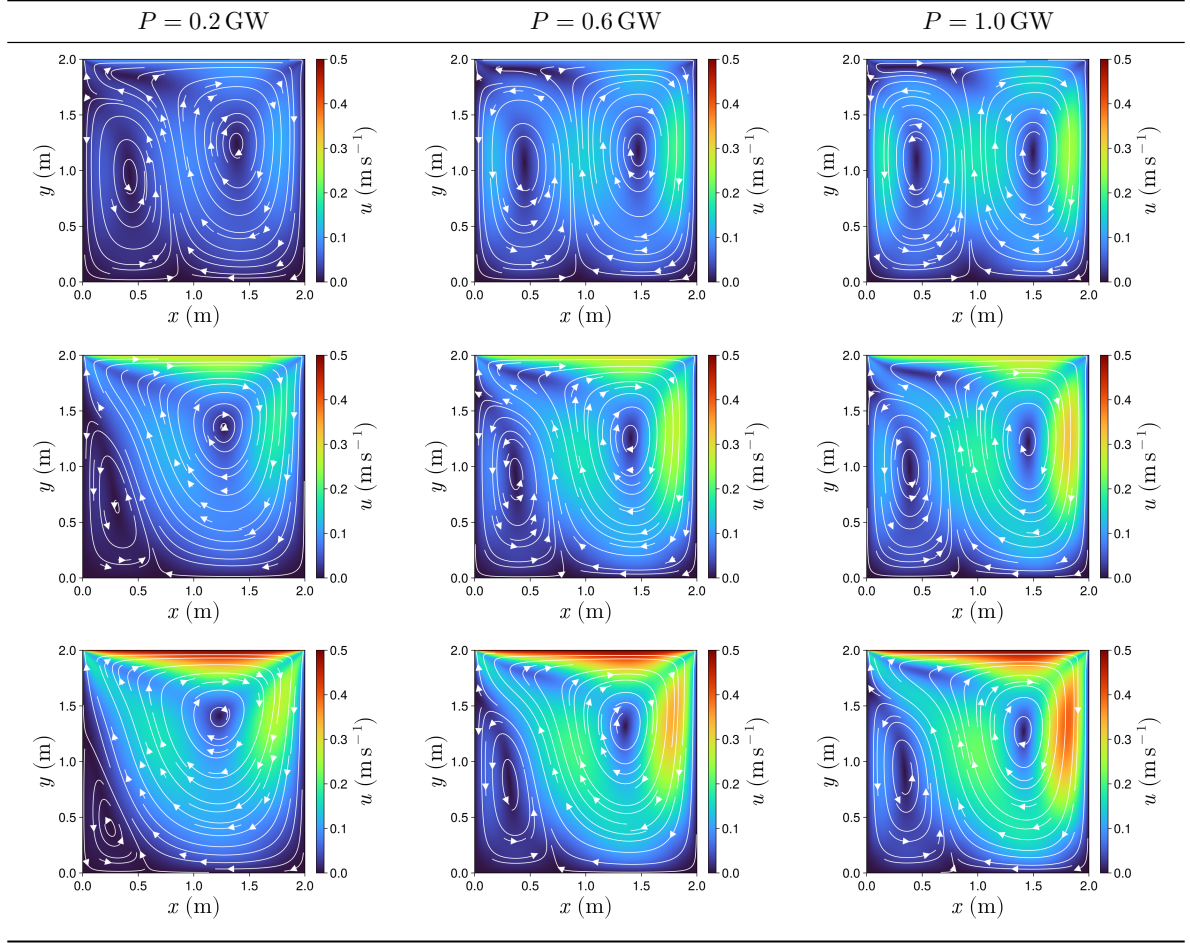


**Figure B.8:** Velocity field for Step 1.4 of the Tiberia benchmark case. For every reactor power, simulations were run with lid velocity  $U_{lid} = [0.1, 0.3, 0.5] \text{ m s}^{-1}$  shown in the top, middle, and bottom row respectively. Streamlines denote the flow direction of the molten salt.

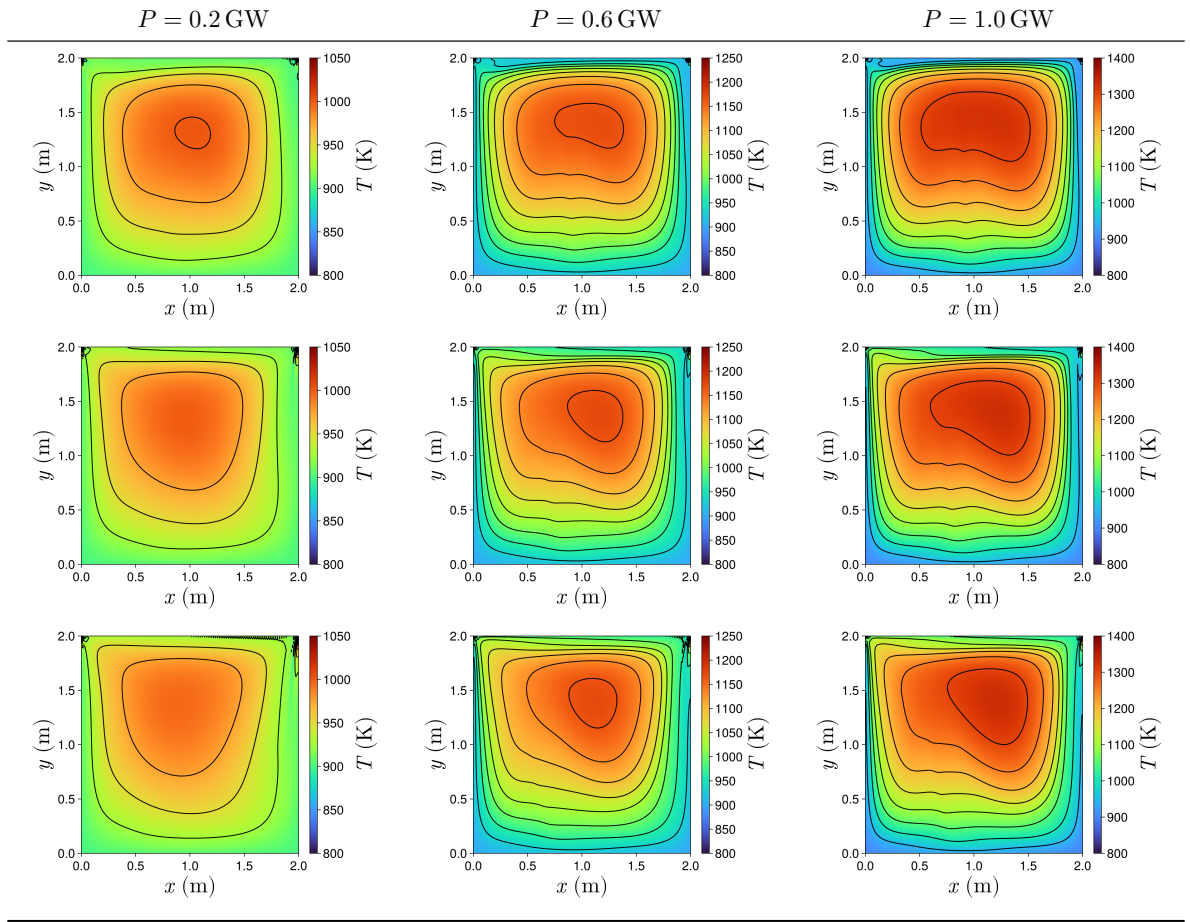


**Figure B.9:** Temperature field for Step 1.4 of the Tiberga benchmark case. For every reactor power, simulations were run with lid velocity  $U_{lid} = [0.1, 0.3, 0.5] \text{ m s}^{-1}$  shown in the top, middle, and bottom row respectively. Isoline intervals of 25 K, 33 K, and 50 K are added to the left, middle, and right column respectively. Note that the colorbar scale for each column is different.

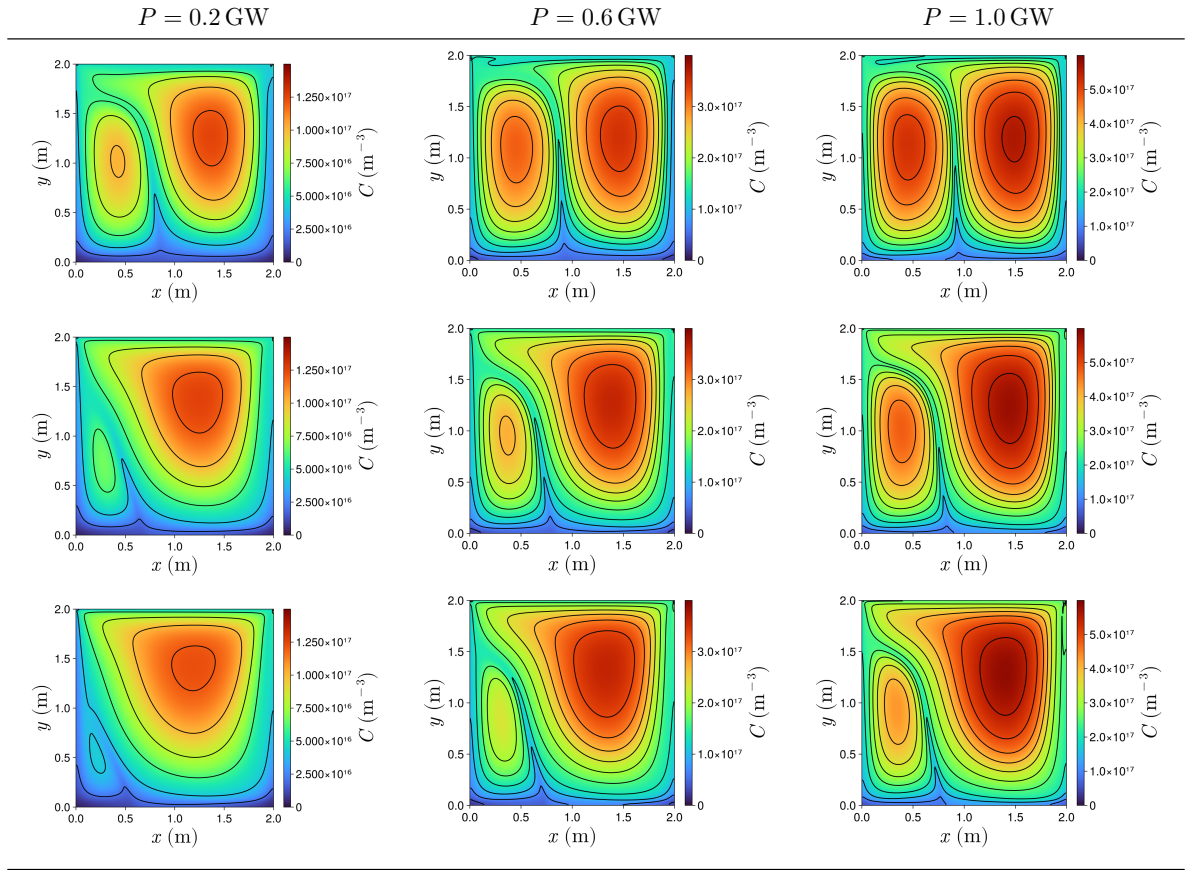


**Figure B.10:** Distribution of the first family of delayed neutron precursors for Step 1.4 of the Tiberge benchmark case. For every reactor power, simulations were run with lid velocity  $U_{lid} = [0.1, 0.3, 0.5] \text{ m s}^{-1}$  shown in the top, middle, and bottom row respectively. Isoline intervals of  $2.0 \times 10^{16} \text{ m}^{-3}$ ,  $3.3 \times 10^{16} \text{ m}^{-3}$ , and  $5.0 \times 10^{16} \text{ m}^{-3}$  are added to the left, middle, and right column respectively. Note that the colorbar scale for each column is different.

# Snapshots of Tiberga Benchmark Transient Simulation Results

This appendix is meant to complete Chapter 5 (in particular Figure 5.9) with snapshots of the power output  $\tilde{P}$  response to the oscillating heat transfer coefficient  $\tilde{\gamma}_{th}$  for Step 2.1 of the Tiberga benchmark. All simulation results were obtained by the FM-LBM multiphysics tool on a  $200 \times 200$  computational grid.

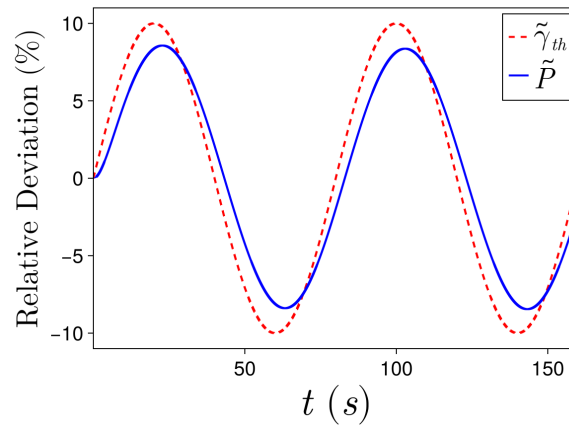


**Figure C.1:** Snapshot of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberga benchmark case, with  $f = 0.0125$  Hz.

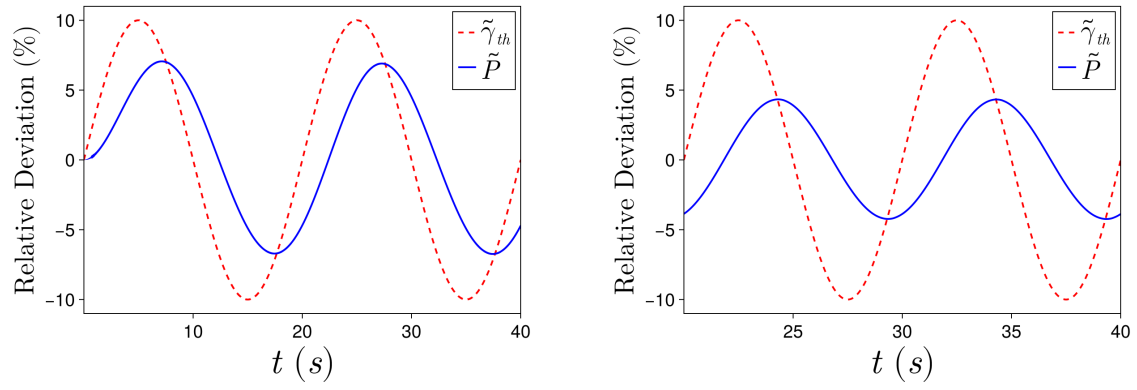


**Figure C.2:** Snapshots of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberga benchmark case. Left:  $f = 0.05$  Hz. Right:  $f = 0.1$  Hz. Note the different scaling of the  $x$ -axis for each graph.

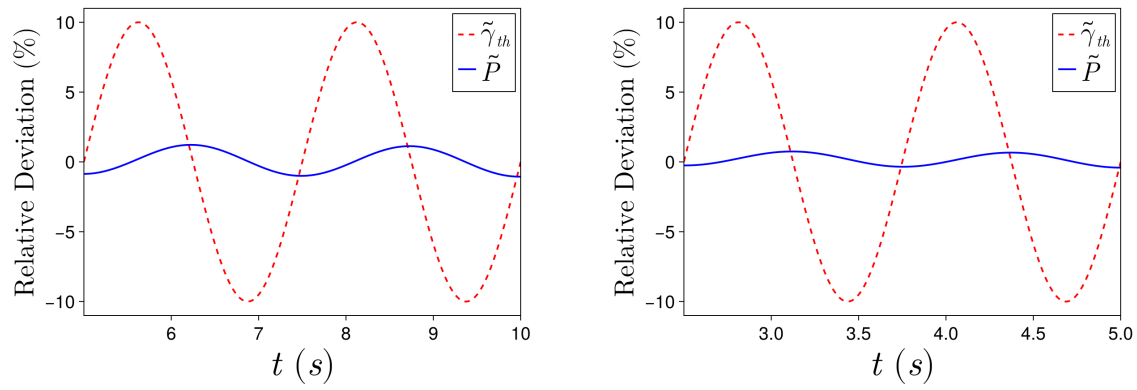


**Figure C.3:** Snapshots of the normalised deviations of the oscillating heat transfer coefficient and the resulting power output for Step 2.1 of the Tiberga benchmark case. Left:  $f = 0.4$  Hz. Right:  $f = 0.8$  Hz. Note the different scaling of the  $x$ -axis for each graph.