Master Degree Thesis in Mechanical Engineering


# Phase change materials modelling by high performance computing

**Niccolò Peruzzi**
Id: **238129**

Accademic tutor: **Pietro Asinari**
Stage tutor: **Christian Obrecht**

I would like to thank Professor Pietro Asinari as my academic tutor of Politecnico di Torino, that has followed and helped me with my work for almost a year. A special thank to Professor Christian Obrecht and all the staff of INSA de Lyon for the oppurtunity that they gave me to work on GPU programming in their facilities and for their support.

I would also like to thank my Mom and Dad as they always inspired me to never quit and always work hard. Thank to my sisters and Viviana for their unconditional love that has helped me in bad times. Finally, there are my friends. We always support each other regarding the times of the day by happily talking about things other than just our problems.

Thank you very much, everyone!

Niccolò Peruzzi

Torino, 4 December 2018.

**Abstract**

In this thesis, a multidimensional numerical model is developed to simulate the behavior of phase change materials. The approach is based on the Link-Wise Artificial Compressibility Method to solve the velocity field of the fluid domain, finite difference operators for the temperature field, and the Enthalpy method formulation to treat the phase change physical status. To verify the model, the numerical results of the mono-dimensional test case are compared with the ones obtained from the analytical Neumann's method. The verification test case is a mono-dimensional domain with simple initial and boundary conditions, also transport phenomena are omitted from the solution. Afterwards, multi-dimensional simulations are performed without omitting the effect of the buoyancy force and advection. First of all, the model is applied to simulate melting in a 3D domain in absence of gravity. Then, the same simulations are run while considering transport phenomena, to show the effects of bouyancy in melting problems.

# Contents

# List of Figures

# List of Tables

# Nomenclature

$T$ = temperature

$t$ = time

$\boldsymbol{u}$ = velocity

$L$ = latent heat

$f$ = liquid fraction

$h$ = enthalpy

$c_p$ = heat capacity at constant pressure

$c_s$ = speed of sound

$N$ = nodes

$\boldsymbol{F}$ = external force

$g$ = gravity acceleration

$x$ = space index

$w$ = stencil weight

$f$ = equilibrium function

$i, j, k$ = spatial indices

*Greek symbols*

$\kappa = (\alpha/c_p\rho)$ thermal diffusivity

$\rho$ = density

$\alpha$ = thermal conductivity

$\boldsymbol{\xi}$ = lattice velocity

$\delta x$ = mesh size or space step

$\delta t$ = time step

$\nu$ = kinetic viscosity

$\zeta$ = artificial compressibility

$\beta$ = thermal coefficient expansion

$\omega$ = relaxation frequency

*Subscripts and superscripts*

$l$ = liquid and liquid fraction threshold

$s$ = solid

$\alpha$ = stencil position

$e$ = equilibrium function

$o$ = odd part

$m$ = melting temperature

0 = cold wall index

1 = hot wall index

(') = dimensionless units

(*) = lattice units

*Dimensionless numbers*

$Ste$ = Stefan number

$Ma$ = Mach number

$Pr$ = Prandtl number

$Ra$ = Raylaigh number

$Re$ = Reynolds number

$Fo$ = Fourier number

# Introduction

Nowadays, one of the biggest issue of our society is the contrast between the increasing energy demand and the depletion of the fossil resource with their negative impact on the environment. Therefore, one key aspect to solve this problem is the role that renewable energies such as solar radiation, ocean waves, wind, and biogas have in supplying zero emissions energy for different applications. However, the intermittency of these renouvable sources is one of their main issue, that can be surpassed with energy storage systems that narrow the gap between energy availability and demand by storing different source energy that can later on be used.

In residential buildings a great amount of energy is used to control the building temperature conditions via different devices. Even if the construction materials have become more efficient in terms of insulation properties, only if used in conjunction with thermal energy storages (TES) the building energy footprint decreases with more benefits for the environment and society. TES are passive systems that use natural mechanisms to exploit external sources, like the sun's radiation, to store thermal energy, which is then released when needed, without relay on electric devices. These type of systems help to increase the thermal efficiency of the building while reducing the demand of energy, because the energy sources are "free".

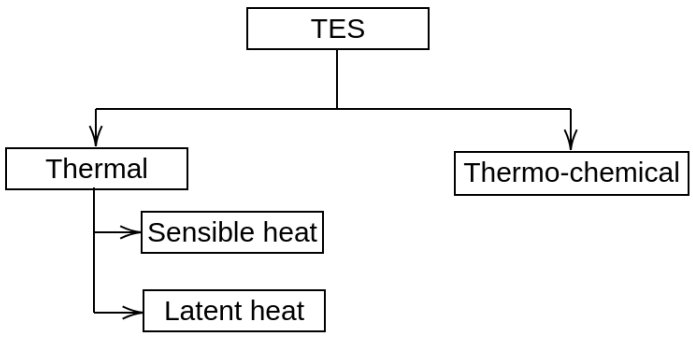



**Figure 1.** TSE classification.

TESs are classified according to the type of process and material that are used to store heat. Thermo-chemical systems work by absorbing and releasing energy while a chemical reaction occurs inside the material. Meanwhile, for thermal systems energy is stored in form of sensible or latent heat. For this applications mainly inorganic, organic and pure materials are used such as paraffis, hydrated salts or metal.

To stock sensible heat the material undergoes a temperature variation while is in contact with the environment. The amount of "capacity" of the TES is directly related to specific heat and the total temperature difference inside the material:

$$Q = \int_{\Delta T} mc_p dT \tag{1}$$

Therefore, materials with high specific heats are preferred. A very simple application of this technologies are concrete walls, that exposed to sun's radiation heat up and released the thermal energy during the night.

Thermal energy can be also stored in form of latent heat when phase change material (pcm) are used. In case of solar TES the sun's heat is absorbed by a melting material, that releases energy when it solidifies:

$$Q = \int_{\Delta T_s} mc_p dT + mfL + \int_{\Delta T_l} mc_p dT \tag{2}$$

This type of latent heat storage solutions offer higher energy storage capacity than sensible heat systems, and a major part of the sun's energy is used if compared with photo-voltaic applications.

The design of solar TES systems with pcm for thermal applications is strongly related to the computational fluid dynamic approach used to simulate phase change phenomena. However, even if the market offers different type of software, their computation performances can be increased using high performance computing as parallel programming. For this aim, using the energy equation as reference, a new numerical model is presented where the Enthalpy method is implemented on the Hybrid Thermal Link-Wise Artificial Compressibility Method (HT LW-ACM) to simulate with high accuracy the velocity field, phase status and temperature profile of melting materials. In details, this new model is able to solve phase change problems considering also the presence of the buoyancy force and mass and momentum mass transport phenomena.

This thesis is articulated in four main chapters. The first describes the phase change problem through the energy equation. The problem solution is computed developing the new Enthalpy HT LW-ACM numerical method. Also, the Neumann's analytical solution is briefly described. In the second chapter a general description of parallel computing is presented, highlighting the main computational aspects of the parallel code like the multiprocessors setting or how the memory hierarchy is handled. The third chapter is dedicated to verify the effectiveness of the numerical approach by comparing its results with Neumann's on the mono-dimensional test case. Finally, in the last chapter multidimensional simulations are run to analyze the effect of advection on phase change.

# Chapter 1

# Modelling phase change materials

## 1.1 Introduction

Phase change phenomena are classified as *moving boundary problems*, or *Stefan problems*, because they are associated with a time-dependent moving interface, which separates the two different states of the material. Therefore, the boundary position is a function of time and space and it is deeply effected by the different thermal phenomena that occur inside the material.

Furthermore, between two different phases the thermal properties change considerably: the various thermal mechanism (convection, conduction diffusion and radiation) are related to the thermophysical properties of the material and its state, resulting in different rates of energy, mass and momentum. For example, for solid state the diffusion mechanisms are not feasible and the rate of energy associated to the movement of the material is null. Therefore, those different thermal phenomena play an important role to solve the time-dependent moving boundary problems.

Initially, the only solution for the Stefan problems was achieved via analytical methods. These techniques offer an exact solution for the one-dimensional case of an infinite or semi-infinite region with simple initial and boundary conditions, where the thermal properties do not change in time. In reality, phase change problems rarely occur in one-dimension and the boundary conditions tend to be more complex. Also, this thype of approach does not assess for the various transport phenomena, that occur simultaneously with other thermal mechanisms. Thus, these models are not applicable in practice, but they are used as standard references to validate other type of models.

A different approach to solve the Stefan problems lays on the usage of numerical methods based either on finite difference or finite element schemes. In fact, finite different techniques can be used in multidimensional problems with complex geometries because of their simplicity in formulation and programming. The evolution of numerical models is deeply related with the age of modern computers: two main eras in these numerical models

can be defined. The first era is based on numerical models developed upon a single equation and omission of convection, due to the limited power of the earlier computer. In the second era, the advent of modern supercomputers had led to more sophisticated numerical models that are able to handle multi-dimensional phenomena with convection, as well as complex geometry and complex boundary conditions.

## 1.2 Energy equation

In litterature there are different method to model PCM. The phase change can be described in term of energy variation using the *energy equation*. However, its basic formulation must be adapt to asses also the presence of the phase change interface inside the material by coupling a new energy term refered as the *liquid fraction term*.

The basic energy equation is a partial differential equation that describes the heat flux inside a material due to both advection and conduction phenomena, if a certain temperature gradient is applied. Without taking into account the viscous heating and pressure effects, it is possible to define the energy equation as:

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \kappa \nabla^2 T \tag{1.1}$$

where $k\nabla^2 T$ refers to conduction and $\boldsymbol{u} \cdot \nabla T$ to advection. This last term refers to the presence of transport phenomena in the fluid phase. By solving the energy equation the temperature $T(\boldsymbol{x}, t)$ and the relative heat flux are computed. It is possible to simplify the energy equation by omitting advection[1], so the conduction driven problem is written as:

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T \tag{1.2}$$

For phase change phenomena this expression is corrected by adding a new term that assesses the phase change variation of energy in conduction driven problem. This term is a function of the latent heat $L$, the liquid fraction $f$ and its time first derivative $\partial f/\partial t$. The latent heat is responsible of the energy variation on the interface[2], while the liquid fraction defines the phase status of the material and the simultaneously presence of liquid and solid on the interface. For the fully solid and liquid regions $f = 0, 1$ respectively, while around the interface $0 < f < 1$. In particular, the movement of the boundary in the medium is described by the first time derivative of the liquid fraction. Overall, this term, referred as the liquid fraction term, describes the reason why during melting the liquid increases and a subtraction of thermal heat occurs:

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T - \frac{L}{c_p} \frac{\partial f}{\partial t} \tag{1.3}$$

[1] This simplified version of (1.1) is a combination of the Fourier's law with the conservation of energy

[2] In case of fusion the energy is absorbed, for solidification it is released. The sign of this term is negative for melting and positive for solidification.

This equation is valid for both isothermal and non-isothermal phase change transformation.

To be more precise (1.3) must be corrected again by reincorporating the advection terms:

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \kappa \nabla^2 T - \frac{L}{c_p} \frac{\partial f}{\partial t} \tag{1.4}$$

This equation is the complete differential equation to solve the moving boundary problem for phase change materials. More in details, the advection term describes the transport phenomena inside the liquid part of the material. Meanwhile, for the solid part of the domain only conduction occurs being the velocity field null. The liquid fraction time derivative is used only on the interface to chose if the material is liquid or solid, also it accounts the sudden variation in energy.

## 1.3 Analytical method

Analytical methods have been always considered the most reliable tool in modelling moving boundary problems, because their solutions are elegant and easy to implement. However, their formulations is based over some strict constrains: the dimension of the problem, the initial and boundary conditions and the constant not-variable thermophysical properties. Analytical solutions are formulated over a one-dimension infinite or semi-infinite region of space ($0 < x < \infty$), with simple initial conditions:

$$\{t = 0 : \forall x \rightarrow T(x, 0) = T_0\} \tag{1.5}$$

and easy to understand boundary conditions:

$$\{\forall t > 0 : x = 0 \rightarrow T(x, t) = T_0, \quad x = \infty \rightarrow T(x, t) = T_1\} \tag{1.6}$$

where $T_1$ is the hot wall temperature, corresponding to the first point in space to melt and $T_0$ is the cold wall temperature as well as the initial temperature. The temperatures are imposed so that the melting temperature $T_m$ lies between the two: $T_0 < T_m < T_1$. Considering constant thermophysical properties the phase change is driven by the dimensionless number called *Stefan number*:

$$Ste = c_p \Delta T / L \tag{1.7}$$

$\Delta T = T_1 - T_0$ is the maximum temperature difference that occurs in the material.

For 1D problems the energy equation is applied with some modifications. Transport phenomena are not considered and the liquid fraction is replaced with the position of the moving interface $X(t)$. In details, the position of the interface scaled over the total length of the region corresponds to the liquid fraction of the domain.

### 1.3.1 Neumann's method

The simplest case to model a PCM is the *one-phase problem*. In this approach (1.5) is modified so that $T(x,0) = T_m$ and (1.6) does not change. It is called the one-phase problems since only the liquid part increases its temperature, while the solid phase is kept at the melting conditions.

The Neumann's approach extends the one-phase problem to the *two-phase problem*. This new more realistic scenario imposes that the cold wall temperature is equal to the initial temperatures $T_0$, which is lower than $T_m$ and rises to $T_1 > T_m$ and (1.6) does not change. The problem is formulated $\forall t > 0$, recalling $X(t)$ the interface position:

- For $0 < x < X(t)$: solid region

$$\frac{\partial T}{\partial t} = \kappa_s \frac{\partial^2 T}{\partial x^2} \tag{1.8}$$

- For $X(t) < x < \infty$: liquid region

$$\frac{\partial T}{\partial t} = \kappa_l \frac{\partial^2 T}{\partial x^2} \tag{1.9}$$

- For $x = X(t)$: interface

$$T(X(t), t) = T_m \tag{1.10}$$

Imposing the Stefan condition, which corresponds to the thermal equilibrium in the interface for the melting process the differential temperature relation is:

$$\kappa_s \frac{\partial^2 T}{\partial x^2} - \kappa_l \frac{\partial^2 T}{\partial x^2} = \frac{L}{c_p} \frac{dX}{dt} \tag{1.11}$$

during the melting process the interface absorbs massive latent heat advancing at a certain speed in the medium. Moreover, for an isothermal melting process the temperature condition for $x = X(t)$ is:

$$T_l = T_s = T_m \tag{1.12}$$

The final analytical solution is written as:

- Solid temperature:

$$T_s = T_0 + (T_m - T_0) \frac{\operatorname{erfc}(x^*)}{\operatorname{erfc}(x^*_{sl})} \tag{1.13}$$

- Liquid temperature:

$$T_l = T_1 + (T_m - T_1) \frac{\operatorname{erf}(x^*)}{\operatorname{erf}(x^*_{sl})} \tag{1.14}$$

where $x^* = x/2\sqrt{\kappa t}$ is the dimensionless position and the value of the dimensionless interface position $x^*_{sl}$ is obtained solving the transcendent equation[3]:

$$\frac{T_m - T_0}{T_m - T_1}\frac{\exp(-(x^*_{sl})^2)}{\operatorname{erfc}(x^*_{sl})} + \frac{\exp(-(x^*_{sl})^2)}{\operatorname{erf}(x^*_{sl})} - \frac{\sqrt{\pi}}{Ste}x^*_{sl} = 0 \tag{1.15}$$

This analytical solution can only be obtained in a rectangular coordinate system.

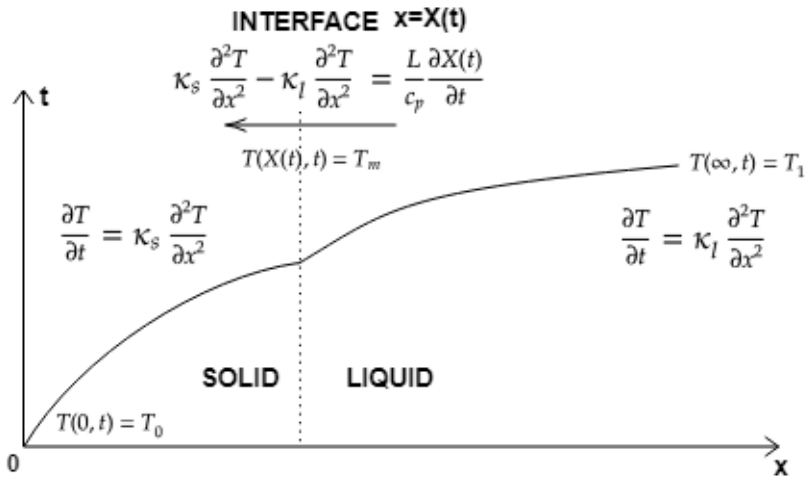



Figure 1.1: Temperature evolution of the two-phase problem.

## 1.4 Numerical method

Numerical methods are a new way to approach differential problems by using finite difference techniques, which enable to approximate differential operators with simple numerical expression. The solutions found with those methods are defined as strong numerical solutions that do not depend on the problem space dimension. Therefore numerical schemes are highly recommended for complex multi-dimensional geometries with complex boundary and initial conditions[1].

The energy equation can be numerically solved by coupling both conduction and advection, which was not allowed in the analytical solutions[4]. This is possible because the Navier-Stokes equations (INSE) are coupled with the energy equation to compute the hydrodynamic fluid properties that assess for transport and advection phenomena.

[3]Equation (1.15) is the simplified version counting on costant thermophysical properties that do not change according to phase.

[4]Recalling that the analytical solution considered the advection term null[2].

A direct numerical solution of the phase change problem can not be formulated because the liquid fraction and temperature time derivative are expressed in the same formula. Moreover, the liquid must computed according to temperature, which depends on the status of the material, but this is expressed via the liquid fraction. Hence, a *latent-heat evolution method* is used to uncoupled this two terms to define the numerical solution. However, the latent-heat method is defined as a weak numerical solution[1], which avoids to explicit the position of the moving boundary, but it focuses on reformulating it in a new way.

To properly asses the numerical solution a particular strategy is outline. First of all, the continuum space domain is divided into independent nodes. The nodes, better referred as *mesh points*, are fixed in space and defined by the following properties: velocity, density, momentum, temperature, enthalpy and liquid fraction. Secondly, the simulation time is split into discrete time steps of size $\delta t$. Finally, through updating rules, applied at each time step, the node properties are computed at each time step. Finally, recalling the discrete temperature and liquid fraction functions, the PCM behavior is modeled.

### 1.4.1 Problem formulation

The energy equation is formulated according to the physical status of the nodes, through which the domain is discretized. For fully solid nodes, only conduction occurs, while advection is added for the fully liquid ones. However, for the mesh points on the interface the problem is more complex. First of all, they are defined as the nodes where the liquid fraction $f$ changes in time: for melting it goes from 0 to 1. Secondly, the advection term can not be applied directly, but it has to be modeled according to the nodes properties. By setting a liquid fraction threshold $f_l$ it is possible to define whether a node on the moving boundary is treated as liquid or solid. Therefore, the general problem formulation is:

$$\begin{cases} \partial_t T = \kappa \nabla^2 T & \text{full solid node } f = 0 \\ \partial_t T = \kappa \nabla^2 T - \partial_t f / Ste & \text{interface solid node } f \leq f_l \\ \partial_t T + \boldsymbol{u} \cdot \nabla T = \kappa \nabla^2 T - \partial_t f / Ste & \text{interface liquid node } f > f_l \\ \partial_t T + \boldsymbol{u} \cdot \nabla T = \kappa \nabla^2 T & \text{full liquid node } f = 1 \end{cases} \tag{1.16}$$

This problem formulation considers all the thermal effects for a melting PCM. At each time step the position of the moving interface is not described, but it can be computed in the data post-processing because it is defined from the liquid fraction, ensuring the consistency with the latent-heat method.

### 1.4.2 Spatial mesh

The space domain is divided into an integer number of mesh points $N$ linked to their nearest neighboring nodes on the mesh via a *lattice*, which is based over a regular Cartesian

grid of mesh size $\delta x$. This lattice is defined as a stencil of dimensions DnQm, where "Dn" is the "n dimensions" of the space domain and "Qm" the amount of discrete velocities $\boldsymbol{\xi}_\alpha$ linked in a single stencil, with $\alpha = 0, ..., Q-1$. The stencil is chosen according to the domain dimension, for example the D1Q3 stencil is used for mono-dimensional problem, while the D3Q19 for three dimension applications where the total number of node is $N = L_x \times L_y \times L_z$, and $L_i$ is the index total nodes.

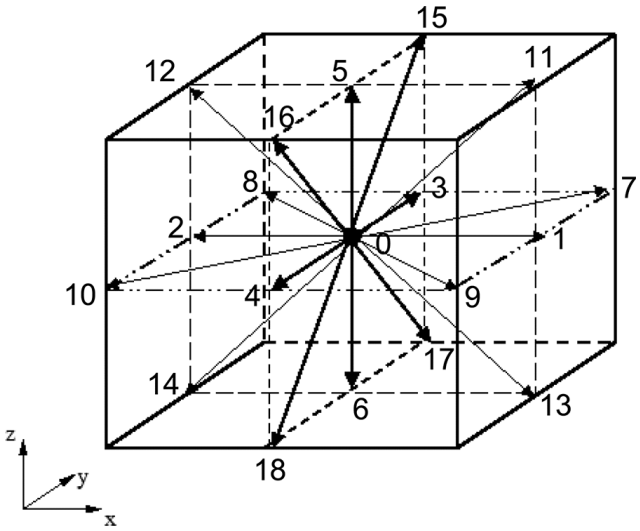



**Figure 1.2.** The D3Q19 stencil - the arrows represent the $\boldsymbol{\xi}_\alpha$ velocities.[6]

The D3Q19 stencil is represented as a cube of mesh size $\delta x$ with one central node and 18 neighboring nodes, that are disposed on its surface symmetrically to ensure mass and momentum conservation. The coordinates of the 19 velocities in this stencil are:

$$\boldsymbol{\xi}_\alpha = \begin{cases} (0,0,0) & \alpha = 0 \\ (\pm 1,0,0),(0,\pm 1,0),(0,0,\pm 1) & \alpha = 1,...,6 \\ (\pm 1,\pm 1,0),(\pm 1,0,\pm 1),(0,\pm 1,\pm 1) & \alpha = 7,...,18 \end{cases} \tag{1.17}$$

### 1.4.3 Enthalpy method

The *Enthalpy method* is the latent heat based model to evaluate the liquid fraction change rate in the energy equation. This method illustrates the physical relation between liquid fraction, temperature and enthalpy $h$ using the latent heat $L$ to account for the energy variation on the interface.

The $h - T$ relation is a step function for isothermal phase change problems[5]:

[5]Depending on the test case and material properties. For metal a non isothermal phase change occurs so the $h - T$ function is a linear curve defined between the solidus and liquidus temperature range.

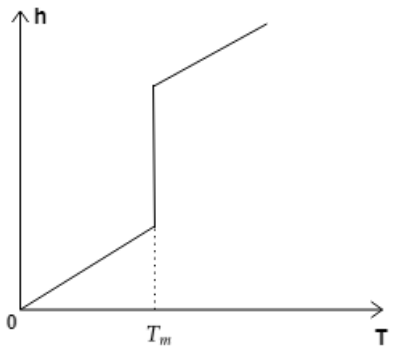



**Figure 1.3.** Enthalpy-Temperature curve for isothermal transformation.

$$h = \begin{cases} c_s T & T < T_m \text{ solid phase} \\ c_l T + L & T > T_m \text{ liquid phase} \end{cases} \tag{1.18}$$

For the interface $T = T_m$, so the enthalpy lays between the solid and liquid enthalpy limits:

$$c_s T_m \leq h \leq c_l T_m + L \tag{1.19}$$

Assuming constant thermal properties, that do not change regarding the phase status, the curve slopes of (fig:1.3) are equal, which means that the heat capacities $c_s$ and $c_l$ are the same.

The liquid fraction change rate is related to the state change rate, being the liquid fraction able to express the domain status. Similarly, the enthalpy time evolution is proportional to the state change rate. Therefore, the liquid fraction and enthlapy change are proportional in the control volume. For melting[6], it yields:

$$\frac{dh}{dt} = -L\frac{df}{dt} \tag{1.20}$$

This relation is able to predict the moving boundary with high accuracy depending on the enthlapy conditions expressed previously (1.18). Finally, the more direct $h - f$ melting function is formulated as:

$$f = \frac{h - h_s}{h_l - h_s} \tag{1.21}$$

where, $h_s = c_p T_0$ is the enthalpy of the cold wall and $h_l = c_p T_1$ the enthalpy of the hot wall.

Furthermore, the liquid fraction can not be a negative value or superior of 1: for solid $f = 0$ and for liquid $f = 1$. Thus, the previous equation is redefined as:

$$0 \leq f = \frac{h - h_s}{h_l - h_s} \leq 1 \tag{1.22}$$

[6]For solidification the sign is changed and became positive.

The update liquid fraction rule is defined starting at time step $t$ and evolving at $t+1$ for each mesh point. The enthalpy formulas (1.18) and (1.19) are expressed in the single system:

$$h^{t+1} = \begin{cases} c_p T^{t+1} & T^{t+1} < T_m \quad \text{solid phase} \\ c_p T^{t+1} + L & T^{t+1} > T_m \quad \text{liquid phase} \\ c_p T^{t+1} + L f^t & T^{t+1} = T_m \quad \text{interface} \end{cases} \tag{1.23}$$

More in details, for the nodes on the interface the enthalpy is a function of the liquid fraction computed in the previous time step. Coupling (1.22) and (1.23), where $h$ is substituted with the current time step by $h^{t+1}$ and $f$ with $f^{t+1}$, it yields:

$$f^{t+1} = \begin{cases} 0 & h^{t+1} < h_s \\ 1 & h^{t+1} > h_l \\ (c_p T_m + L f^t - h_s)/(h_l - h_s) & h_s < h^{t+1} < h_l \end{cases} \tag{1.24}$$

the liquid fraction on the interface is also express as:

$$f^{t+1} = \frac{c_p T_m + L f^t - c_p T_0}{c_p \Delta T + L} \tag{1.25}$$

which is simplified using the dimensionless Stefan number (1.7):

$$f^{t+1} = (T^{t+1} + T_m) Ste + f^t \tag{1.26}$$

In conclusion, the updating Enthalpy rule for the liquid fraction is formulated in dimensionless quantities as:

$$f^{t+1} = \begin{cases} 0 & T^{t+1} \leq T_m - f^t/Ste \\ 1 & T^{t+1} \geq T_m + (1 - f^t)/Ste \\ (T^{t+1} + T_m) Ste + f^t & \text{otherwise} \end{cases} \tag{1.27}$$

The Enthalpy method is reformulated in the general Enthalpy method algorithm. Assuming to have at disposal the temperature profile evolution $T(\boldsymbol{x}, t)$, the working algorithm is:

**Algorithm 1.** Enthalpy method formulation: updating rule

**for all** time step $t$ **do**
  **for all** mesh point $\boldsymbol{x}$ **do**
    load $T^t$
    load $f^t(\boldsymbol{x}, T)$
    **if** $T^{t+1} \leq T_m - f^t/Ste$ **then**
      compute $f^{t+1} = 0$
    **else if** $T^{t+1} \geq T_m + (1 - f^t)/Ste$ **then**
      compute $f^{t+1} = 1$
    **else**
      compute $f^{t+1} = (T^{t+1} + T_m)Ste + f^t$
    **end if**
  **end for**
**end for**

### 1.4.4 Link-Wise Artificial Compressibility method

By itself, the energy equation only describes the thermal interactions in the medium. In presence of fluid the heat flux is also function of the hydrodynamic behavior of the material. Considering an isothermal flow simulation the fluid properties are expressed in the incompressible Navier-Stoke equations (INSE). Those are solved via a particular numerical scheme called *Link-Wise Artificial Compressibility method* (LW-ACM).

The INSE are expressed as:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{F} \tag{1.28}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1.29}$$

where $\rho$ and $\nu$ are the density and kinematic viscosity, $p$ the pressure and $\boldsymbol{F}$ is the external force per unit mass due to gravity.

The LW-ACM resorts to solve the INSE without the Poison solver process, instead the continuity equation is replaced by the *artificial compressibility equation* (ACE):

$$\zeta \frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{u} = 0 \tag{1.30}$$

where $\zeta$ is the artificial compressibility, linked to the artificial density $\rho_a$ by $p = \rho_\alpha/\zeta$, yielding $c_s = 1/\sqrt{\zeta}$. The presence of the pressure time derivative enables for explicit time-integration. Therefore, the LW-ACM formulation is defined as a discrete formulation of the ACM, that operates on the spatial mesh based over the DnQm stencil.

In LW-ACM the fluid is represented by the set of Q dependent variables inside the stencil, which are expressed through the *local equilibrium functions* $\{f_\alpha\}$ defined at the

mesh point:

$$\rho = \sum_{\alpha} f_{\alpha} \tag{1.31}$$

$$\rho \boldsymbol{u} = \sum_{\alpha} f_{\alpha} \boldsymbol{\xi_{\alpha}} \tag{1.32}$$

setting $c_s = 1/\sqrt{3}$[7] the equilibria become:

$$f_{\alpha}^{(e)}(\rho, \boldsymbol{u}) = w_{\alpha}\rho \left(1 + 3\boldsymbol{u}\cdot\boldsymbol{\xi_{\alpha}} + \frac{9}{2}\left(\boldsymbol{u}\cdot\boldsymbol{\xi_{\alpha}}\right)^2 - \frac{3}{2}\boldsymbol{u}^2\right) \tag{1.33}$$

and the weights $w_{\alpha}$ associated to the velocity set for the D3Q19 stencil are:

$$w_{\alpha} = \begin{cases} 1/3 & \text{for } \alpha = 0 \\ 1/18 & \text{for } \alpha = 1, ..., 6 \\ 1/36 & \text{for } \alpha = 7, ..., 18 \end{cases} \tag{1.34}$$

Expressing the odd part of the equilibrium functions it is possible to simplify the updating rule as:

$$f_{\alpha}^{(e,o)}(\rho, \boldsymbol{u}) = \frac{1}{2}\left(f_{\alpha}^{(e)}(\rho, \boldsymbol{u}) - f_{\alpha}^{(e)}(\rho, -\boldsymbol{u})\right) \tag{1.35}$$

Therefore, the updating rule is expressed as:

$$f_{\alpha}(\boldsymbol{x}, t+1) = f_{\alpha}^{(e)}(\boldsymbol{x} - \boldsymbol{\xi}_{\alpha}, t) + 2\left(\frac{w-1}{w}\right)\left(f_{\alpha}^{(e,o)}(\boldsymbol{x}, t) - f_{\alpha}^{(e,o)}(\boldsymbol{x} - \boldsymbol{\xi}_{\alpha}, t)\right) \tag{1.36}$$

where the kinematic diffusivity is related to the relaxation frequency $w$:

$$\nu = \frac{1}{3}\left(\frac{1}{\omega} - \frac{1}{2}\right) \tag{1.37}$$

this last relation is also expressed as $1 - 6\nu = 2\left(\frac{w-1}{w}\right)$.

To sum it up, the LW-ACM is a new approach to solve the INSE modified via the ACE, which operates on the DnQm lattice to simulate isothermal flows, since the equilibrium functions are temperature independent. The LW-ACM is only applied on the mesh points that are fluid, whether for the solid part of the material the velocity is null and the density is kept constant regarding the temperature gradient.

From an implementation point of view, the updating rule is summarized in the general LW-ACM algorithm formulation. The algorithm is been developed to keep at lowest the request for the computational power. Hence, at each time step only $\rho$, $\boldsymbol{u}$ are store.

[7] To be consistent with the sound speed in the acoustic scaling.

**Algorithm 2.** LW-ACM general single step formulation

**for all** time step $t$ **do**
  **for all** mesh point $\boldsymbol{x}$ **do**
    **for all** index $\alpha$ **do**
      apply the boundary conditions
      load $\rho(\boldsymbol{x} - \boldsymbol{\xi}_\alpha, t)$ and $\boldsymbol{u}(\boldsymbol{x} - \boldsymbol{\xi}_\alpha, t)$
      compute (1.33): $f_\alpha^{(e)}(\boldsymbol{x} - \boldsymbol{\xi}_\alpha, t)$
      compute (1.35): $f_\alpha^{(e,o)}(\boldsymbol{x} - \boldsymbol{\xi}_\alpha, t)$
    **end for**
    **for all** index $\alpha$ **do**
      compute (1.36): $f_\alpha(\boldsymbol{x}, t+1)$
    **end for**
    compute (1.31) and (1.32): $\rho(\boldsymbol{x}, t+1)$ and $\boldsymbol{u}(\boldsymbol{x}, t+1)$
  **end for**
**end for**

### 1.4.5 Hybrid thermal LW-ACM

The *Hybrid thermal* (HT) method is a numerical scheme implemented on the isothermal LW-ACM to model the temperature evolution inside the control volume according to all thermal phenomena. With this new approach the fluid behaviors are described under the presence of gravity and temperature gradient in the domain.

The energy equation is simplified with finite difference schemes[6] that operate on the same grid points of the LW-ACM. Using $i, j, k$ as the mesh indices, and $t$ the time index, for the D3Q19 the temperature time dependent operator is expressed as:

$$\tilde{\partial}_t T_{i,j,k} = T^t_{i,j,k} - T^{t-1}_{i,j,k} \tag{1.38}$$

for conduction, its operator $\nabla^2 T$ is simplified in:

$$\begin{aligned}\tilde{\nabla}^2 T = 2(&T^t_{i+1,j,k} + T^t_{i-1,j,k} + T^t_{i,j+1,k} + T^t_{i,j-1,k} + T^t_{i,j,k+1} \\ &+ T^t_{i,j,k-1}) - \frac{1}{4}(T^t_{i+1,j+1,k} + T^t_{i-1,j+1,k} + T^t_{i+1,j-1,k} + T^t_{i-1,j-1,k} \\ &+ T^t_{i,j+1,k+1} + T^t_{i,j-1,k+1} + T^t_{i,j+1,k-1} + T^t_{i,j-1,k-1} + T^t_{i+1,j,k+1} \\ &+ T^t_{i-1,j,k+1} + T^t_{i+1,j,k-1} + T^t_{i-1,j,k-1}) - 9T^t_{i,j,k}\end{aligned} \tag{1.39}$$

while, for advection, which is defined using $\boldsymbol{u} \cdot \nabla T$. Where $\boldsymbol{u} = (ux, uy, uz)$ is the velocity vector of the mesh point and the temperature divergence is expressed as:

$$\begin{aligned}\tilde{\partial}_{x,y,z} T^t_{i,j,k} = &T^t_{i+1,j,k} - T^t_{i-1,j,k} \\ &- \frac{1}{8}(T^t_{i+1,j+1,k} - T^t_{i-1,j+1,k} + T^t_{i+1,j-1,k} - T^t_{i-1,j-1,k} \\ &T^t_{i+1,j,k+1} - T^t_{i-1,j,k-1} + T^t_{i+1,j,k-1} - T^t_{i-1,j,k-1})\end{aligned} \tag{1.40}$$

For the D3Q19 stencil, the numerical solution of the energy equation is formulated as:

$$
\begin{aligned}
T^{t+1}_{i,j,k} &= (1-9\kappa)T^{t}_{i,j,k} + 2\kappa(T^{t}_{i+1,j,k} + T^{t}_{i-1,j,k} + T^{t}_{i,j+1,k} + T^{t}_{i,j-1,k} + T^{t}_{i,j,k+1} \\
&+ T_{i,j,k-1}) - \frac{1}{4}\kappa(T^{t}_{i+1,j+1,k} + T^{t}_{i-1,j+1,k} + T^{t}_{i+1,j-1,k} + T^{t}_{i-1,j-1,k} + T^{t}_{i+1,j,k+1} + T^{t}_{i-1,j,k+1} \\
&+ T^{t}_{i+1,j,k-1} + T^{t}_{i-1,j,k-1} + T^{t}_{i,j+1,k+1} + T^{t}_{i,j-1,k+1} + T^{t}_{i,j+1,k-1} + T^{t}_{i,j-1,k-1}) \\
&- ux^{t}_{i,j,k}(T^{t}_{i+1,j,k} - T^{t}_{i-1,j,k} - \frac{1}{8}(T^{t}_{i+1,j+1,k} - T^{t}_{i-1,j+1,k} + T^{t}_{i+1,j-1,k} \\
&- T^{t}_{i-1,j-1,k} + T^{t}_{i-1,j,k+1} - T^{t}_{i-1,j,k-1} + T^{t}_{i+1,j,k-1} - T^{t}_{i-1,j,k-1})) \\
&- uy^{t}_{i,j,k}(T^{t}_{i,j+1,k} - T^{t}_{i,j-1,k} - \frac{1}{8}(T^{t}_{i+1,j+1,k} - T^{t}_{i+1,j-1,k} + T^{t}_{i-1,j+1,k} \\
&- T^{t}_{i-1,j-1,k} + T^{t}_{i,j+1,k+1} - T^{t}_{i,j-1,k+1} + T^{t}_{i,j+1,k-1} - T^{t}_{i,j-1,k-1})) \\
&- uz^{t}_{i,j,k}(T^{t}_{i,j,k+1} - T^{t}_{i,j,k-1} - \frac{1}{8}(T^{t}_{i+1,j,k+1} - T^{t}_{i+1,j,k-1} + T^{t}_{i-1,j,k+1} \\
&- T^{t}_{i-1,j,k-1} + T^{t}_{i,j+1,k+1} - T^{t}_{i,j-1,k+1} + T^{t}_{i,j+1,k-1} - T^{t}_{i,j-1,k-1}))
\end{aligned}
\tag{1.41}
$$

More in details, the advection phenomenon occurs when the fluid is exposed to a temperature gradient and under the action of the *buoyancy force* $\boldsymbol{F}$. Physically, the thermal gradient is responsible of the local variation of density inside the fluid, while the external force induces transport phenomena.

In solid region, since $\boldsymbol{u} = \boldsymbol{0}$ and being the density constant, the advection term of the energy equation is null.

The external force per unit of mass $\boldsymbol{F}$ is expressed as:

$$\boldsymbol{F} = -\beta(T - T_a)\boldsymbol{g} \tag{1.42}$$

where $\beta$ is the thermal coefficient expansion, $T$ the local temperature and $T_a$ the fluid average temperature, $\boldsymbol{g}$ is the external acceleration of the fluid due to gravity.

On the LW-ACM the implementation of $\boldsymbol{F}$[8] is realized by a new additional term in (1.36):

$$f^{\boldsymbol{F}}_{\alpha}(\boldsymbol{x}, t+1) = f_{\alpha}(\boldsymbol{x}, t+1) + f^{(e,o)}_{\alpha}(\rho(\boldsymbol{x}, t), \boldsymbol{F}(\boldsymbol{x}, t)) \tag{1.43}$$

The numbers of computation steps required to perform the HT LW-ACM confronted with the isothermal case are only two. The first is the correction of the equilibrium functions and the computation of the buoyancy force, the second is the coupling of the temperature operators to compute the temperature of the mesh points. Therefore, this new approach does not increase the complexity of the algorithm and the request of computation power. Also, the only variable to be stored are: the temperature, density and velocity at each time step for all the nodes.

[8] In the INSE the external force is defined, but in the previous LW-ACM this term was not expressed because the method is isothermal.

**Algorithm 3.** HT LW-ACM general formulation

**for all** time step $t$ **do**
  **for all** mesh point $\boldsymbol{x}$ **do**
    **for all** index $\alpha$ **do**
      apply the boundary conditions
      load $T(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$, $\rho(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$ and $\boldsymbol{u}(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$
      compute (1.43): $f_\alpha^{(e)}(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$ and $f_\alpha^{(e,o)}(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$
      compute (1.41): $T(\boldsymbol{x},t+1)$
      compute (1.42): $\boldsymbol{F}$
      compute $f_\alpha(\boldsymbol{x},t+1)$
    **end for**
    compute $\rho(\boldsymbol{x},t+1)$ and $\boldsymbol{u}(\boldsymbol{x},t+1)$
  **end for**
**end for**

## Enthalpy Hybrid Thermal LW-ACM

The HT LW-ACM does not solve the phase change problem. Its numerical solution can only be applied for solid or liquid domain. In fact, the temperature for the fully solid or liquid mesh points is computed via (1.41). But, for the melting nodes, which are partially solid or fluid, the Enthalpy method is implemented on the HT LW-ACM to model PCMs.

The general phase change solution is defined using (1.16), where the conduction and advection terms are already expressed with (1.39) and (1.40). The temperature time derivative is solved by a simple finite difference scheme, that is also used to express the time dependent liquid fraction term $\partial_t f$:

$$\partial_t f = f_{i,j,k}^{t} - f_{i,j,k}^{t-1} \tag{1.44}$$

The values of the liquid fraction for the nodes on the moving boundary is expressed in (1.27) as:

$$f^t = (T^t + T_m)Ste + f^{t-1} \tag{1.45}$$

It is important to notice that the updating Enthalpy rule is applied after the temperature is been computed for all the mesh points at the current time step.

To solve the energy equation, for solid and liquid nodes the temperature time solution is based on (1.41), which is extended to the melting nodes by coupling the liquid fraction term in the temperature expression $T_{i,j,k}^{t+1} = T(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$:

$$T_{i,j,k}^{t+1} = T(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t) - \frac{1}{Ste}(f_{i,j,k}^{t} - f_{i,j,k}^{t-1}) \tag{1.46}$$

The time relation $T-f$ is achieved by coupling the liquid fraction effect on temperature with one time step of difference.

The general formula $T(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$ is modified according to the nodes phase status, recalling null the advection term for solid nodes. Meanwhile, on the interface the partially

melted nodes are treated as solid or liquid based on their liquid fraction. If $f \leq 0.5$ the node is solid, otherwise liquid based on the liquid fraction threshold $f_l = 0.5$.

### 1.4.6 Boundary conditions

For the boundary nodes the link-wise formulation is modified to asses the presence of the walls through which heat is exchange. The boundary conditions are split between the geometrical boundary conditions and the thermal boundary conditions.

The first are applied only in presence of liquid boundary mesh points to assess the fluid interaction between the nodes and the walls. By contrast, the wall boundary conditions are not applied in case of solid nodes.

The geometrical boundary conditions are simulated using the *no-slip* boundary conditions: the walls are located halfway between two consecutive nodes and aligned along the mesh. The nodes of the stencil outside the control volume are counted as solid. In this case the modified *bounce-back* rule is applied, which consist in bouncing back the data between the solid and fluid nodes.

The ideal bounce back situation consists in two main step. The first is the *streaming* step, where the equilibrium functions are sent to the solid nodes. Afterwards, the *collision* step is performed on the solid node data. The new calculated equilibrium functions are then bounced back to the liquid mesh points in a streaming process. By aligning the wall halfway in the stencil, the two steps are summarized in a single operation: the equilibrium functions travel half way to the wall and after colliding are bounced back of the same distance to the central mesh point. Therefore, only two streaming process will take place.

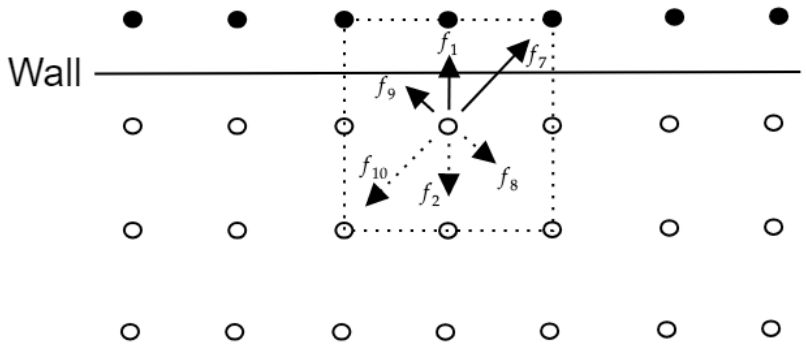



**Figure 1.4.** Bounce back graphical explanation for the D3Q19 along the $x$ direction: the continues arrows represent the streaming step, the hatched ones the collision step. In black the in-wall nodes, and in white the domain mesh points.

Finally, the bounce back operator $BB_{(\alpha)}$ is defined for boundary fluid mesh points $\boldsymbol{x}_f$ as:

$$f_{BB_{(\alpha)}}(\boldsymbol{x}_f, t) = f_{\alpha}^{(e)}(\boldsymbol{x}_f, t) + \left(2 - \frac{2}{\omega}\right) f_{BB_{(\alpha)}}^{(e,o)}(\boldsymbol{x}_f, t) \tag{1.47}$$

Furthermore, the equilibrium functions are bounced back from the wall nodes to their symmetrical nodes inside the lattice. For example, in the D3Q19 stencil, $f_{BB_{(2)}}(\boldsymbol{x}_f, t)$ is computed using $f_1(\boldsymbol{x}, t)$. Therefore, due to the highly symmetry of the stencil node 1 is bounced back in 2 and vice-versa, 3 with 4, 5 with 6 and so on.

Generally, the bounce back operations is performed when a liquid node and a solid node interact in the lattice, so this rule is also applied when a solid node is in contact with a liquid node on the phase change interface.

The thermal boundary conditions are on the second-order interpolation scheme. Refering to the nodes on the lattice, if the node lays outside the domain is temperature is set according to the thermal wall conditions and used in (1.41). For isothermal walls, the in-wall node temperature is $T_{wall}$, while for adiabatic conditions the in-wall temperature is computed using a linear function to interpolate the value of temperature.

### 1.4.7 Main algorithm

The main algorithm is the procedure that is followed to properly solve the phase change problem with the Enthalpy HT LW-ACM. The algorithm main focus is to keep at lowest the number of computational steps required and to use efficiently the computation power.

The Enthalpy method implementation on the HT LW-ACM is realized by adding a first step in (3), in which the liquid fraction variation $\Delta f$ is computed according to (1.45). If this value differs from zero then the node is melting, and based on the $f_l$ is chosen to be treated as liquid or solid. Afterwards, temperature, velocity and density are computed according to the previous conditions. Finally, the liquid fraction at the current time step is computed.

Overall, in the final algorithm the additional steps increase the request for storing and loading values, without increasing the computation demands. Also, this algorithm is highly suitable for parallel computing techniques, where the temperature and fluid values are computed in each mesh points simultaneously. This procedure decreases the computing time requirement, but it enhances the request for more complex and powerful processing units.

**Algorithm 4.** Enthalpy HT LW-ACM general formulation

**for all** time step $t$ **do**
  **for all** mesh point $\boldsymbol{x}$ **do**
    **for all** index $\alpha$ **do**
      apply the boundary conditions
      load $T(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$, $\rho(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$ and $\boldsymbol{u}(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t)$
      load $f^t$ and $f^{t-1}$
      compute $\Delta f = f^t - f^{t-1}$
      **if** $f^t \leq 0.5$ **then**
        compute $T^{t+1} = T(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t) - \frac{1}{Ste}\Delta f$
        impose $\boldsymbol{u}=\boldsymbol{0}$ and $\rho = 1$
      **else**$f^t > 0.5$
        compute $T^{t+1} = T(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,t) - \frac{1}{Ste}\Delta f$
        compute $f_\alpha^{\boldsymbol{F}}(\boldsymbol{x},t+1)$
        compute $\rho(\boldsymbol{x},t+1)$ and $\boldsymbol{u}(\boldsymbol{x},t+1)$
      **end if**
    **end for**
    compute $f^{t+1}$
  **end for**
**end for**

### 1.4.8 Physical and lattice units

The proposed solution of the phase change problem is based on the implementation of the Enthalpy and Hybrid Thermal method over the LW-ACM. In the link-wise formulation the units of quantities defined directly on $f_i$ are defined as mesh-dependent quantities called *lattice units*, because the equilibrium functions are computed on the lattice. These mesh-dependent units are not the proper choice to model PCMs, because as the mesh size tends to zero so the lattice values are affected: for instance the mesh dependent velocity fields $\boldsymbol{u} = \sum_i \boldsymbol{v}_i f_i / \sum_i f_i$ tends to zero as the mesh spacing vanishes. Therefore, to solve this problem, the corresponding mesh-independent *physical units* are computed using aimed scaling techniques.

The lattice units quantities are used to perform the computations and only during the post-processing phase the corresponding mesh-independent physical units are computed[9]. To this aim, the energy equation is formulated in its lattice units via the *acoustic scaling*, where the physical units are scaled first in the dimensionless values and then into the corresponding lattice units.

[9]The post-processing phase is not necessary since the finite-difference formulation of this method can be done directly in physical units.

| Physical | Dimensionless(') | Lattice(*) |
|---|---|---|
| $\boldsymbol{u} = \alpha/L$ | 1 | $Ma/\sqrt{3}$ |
| $L$ | 1 | $N$ |
| $t$ | $Fo = \kappa t/L^2$ | $FoN\sqrt{3}/Ma$ |
| $\nu$ | $Pr$ | $MaPrN/\sqrt{3}$ |
| $\kappa$ | 1 | $MaN/\sqrt{3}$ |
| $\delta x$ | $1/N$ | 1 |
| $\delta t$ | $Ma/(N\sqrt{3})$ | 1 |
| $c_s$ | $1/Ma$ | $1/\sqrt{3}$ |
| $T$ | $(T - T_0)/\Delta T$ | $(T - T_0)/\Delta T$ |

**Table 1.1.** Acoustic scaling table.

Applying this scheme the energy equation becomes:

$$\frac{\partial T^{(*)}}{\partial t^{(*)}} + \boldsymbol{u}^{(*)} \cdot \nabla^{(*)} T^{(*)} = \kappa^{(*)} \nabla^{2(*)} T^{(*)} - \frac{1}{Ste} \frac{\partial f}{\partial t^{(*)}} \tag{1.48}$$

where the $(*)$ is not used to simply the notation in the scaled formulas. From this scaled energy equation, the phase change problem is formulated as a function of the number of nodes $N$ of the spatial mesh and the various dimensionless numbers. Meanwhile, the liquid fraction is not scaled in lattice units, since it is a dimensionless value. The buoyancy force its scaling formula is scaled using:

$$\boldsymbol{g}^{(*)} \beta^{(*)} = \frac{Ra\nu^2}{PrN^3 \Delta T} = \frac{RaMa^2}{3N\Delta T} \tag{1.49}$$

# Chapter 2

# HPC: CUDA

## 2.1 Introduction

The term *High Performance Computing*, HPC, refers to the practice of aggregating several technologies and system software to solve computational problems with high performances in the order of *PetaFLOPS*[1]. In HPC the computational resources are optimized effectively to enhance the computation power and decrease the simulation time using *Parallel Computing*.

Parallel Computing is a way to execute many calculations or processes concurrently to effiecently work with large data sets. Its primary objective is to increase the available computation power for faster simulations, this is achieved by dividing the workload in small chunks or components, which are concurrently executed at the same time. The results are then recollected and coupled together. However, the main issue of this procedure is related to the type of problems: the more the problem can be parallelized the more HPC and Parallel Computing become the most reliable tools to solve it. For example, CFD simulations are highly parallelized due to the large data-set required to solve them.

HPC evolution is linked to the increasing demand for processing speed and power. In order to meet these requests, manufacturers, meanly Intel and AMD, started to develop CPU microprocessor architectures that have not satisfied the computational performance requested. To fill this gap, in 2006 NVIDIA firstly released the free CUDA free framework[2] for parallel computing on NVIDIA GPUS. Therefore, GPUs have emerged as highly parallel, multithreaded, manycore processor with high computational horsepower recalled as *General Purpose GPUs* (GPGPUs).

[1] *PetaFLOPS* is the acronym of: $10^{15}$ FLoating Point Operations Per Second.
[2] Nowadays its release is free as an open-source framework.

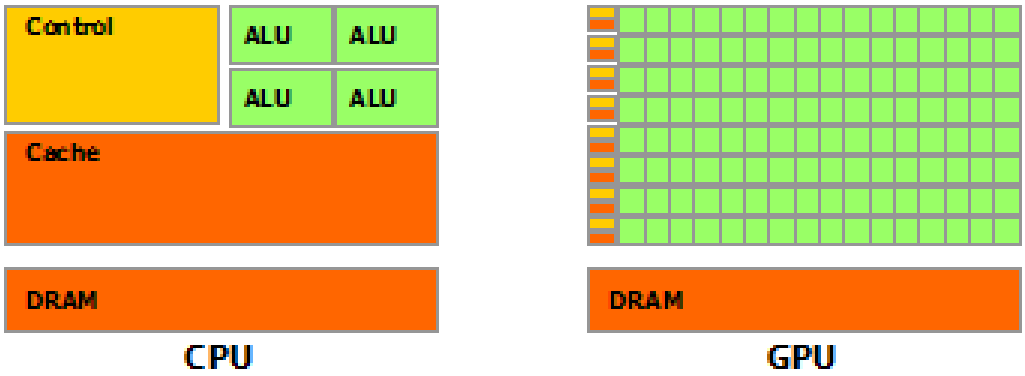



**Figure 2.1.** CPU and GPU layout comparison (In green the transistors).[11]

More in details, the reason behind the GPGPU triumphs over the CPU it is based on how data are handled. The GPU dedicates more transistors[3] to data processing by offering a compute-intensive, highly parallel computations with high arithmetic intensity[4], with less flow control and less execution time due to its local memory. Meanwhile, the CPU is responsible of data caching and flow control, having less transistors dedicated to data processing.

## 2.2 CUDA

*Compute Unified Device Architecture* (CUDA) is a general purpose parallel computing platform and programming model for parallel computing developed to work on NVIDIA GPUs. CUDA allows the user to develop the program in C[5] as the main language with minimal set of language extensions required. The main three key abstractions fo NVIDIA are: the *hierarchy of thread groups*, *shared memories* and *barrier synchronization*.

### 2.2.1 Architecture and Hardwre

The architecture model of parallel programming[6] is based over a strict hierarchy, this helps to properly handle the fluxes of data that move between the different hardware components. A CPU-based *Host* controls different *Compute Devices* as CPUs and GPUs. Each of them is made by multiple *Compute Units* and within these are multiple *Processing Elements*. At the deeper level the processing elements execute the *Kernel*, which represents a given instruction.

[3]Which represent the physical computing cores.
[4]The ratio of arithmetic operations to memory operations.
[5]Also other programming environments can be use, like FORTRAN.
[6]i.e. CUDA and OpenCL.

NVIDIA hardware definitions may change according to different documents: the computing units are called "Stream Multiprocessor" (SMs) or "CUDA cores", meanwhile the processing elements can be defined as "threads". Never the less, the architecture model and its hardware hierarchy remain untouched. NVIDIA defines a CUDA kernel as a C functions, executed in parallel by the threads.

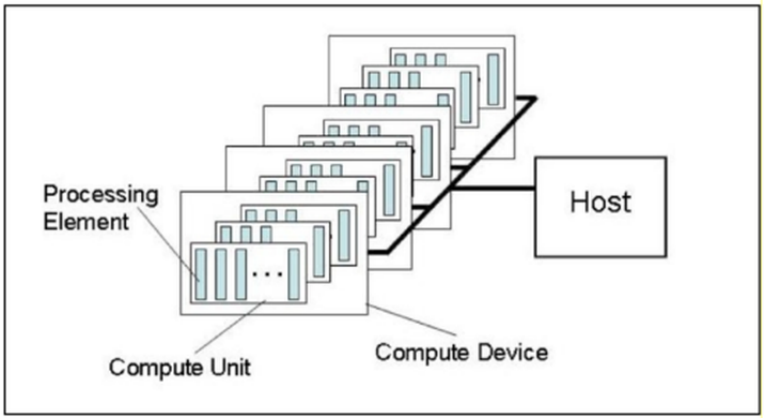



**Figure 2.2.** CUDA hierarchy.[7]

### 2.2.2 Execution Model

The execution model is the way the various devices interact in terms of data flows and computation tasks. At the top level the Host uses the CUDA API platform to query and select the compute device. Then, it submits the work instructions and it manages the workload across the different devices[7]. Meanwhile, at the lower level, in the heart of parallel computing, kernels[8] are run over each thread. This top level model can be recall as the execution hierarchy.

At the lower level, in the heart of CUDA code, kernels are executed in each processing unit over a N-dimensional computation domain. This domain, or *(CUDA) grid*, is partition into several *work-groups*, called by NVIDIA thread blocks or blocks. Within a single block a number of independent elements of execution called *work-item* or (CUDA) threads are grouped; within a work-group a thread has a unique ID accessible through the kernel. When a kernel is launched the data structure is partition into several blocks of threads that run cooperatively in parallel the instructions.

The lower execution model guides the programmers to split the problem into sub-problems that are solved independently in parallel work-groups. Inside each block a finer partition of the sub-problem is solved cooperatively in parallel by all threads. In each block

[7]The Host interacts with the GPUs via a code of instructions made in C language.
[8]CUDA kernels are written in a specific language, that can be refer as CUDA code.

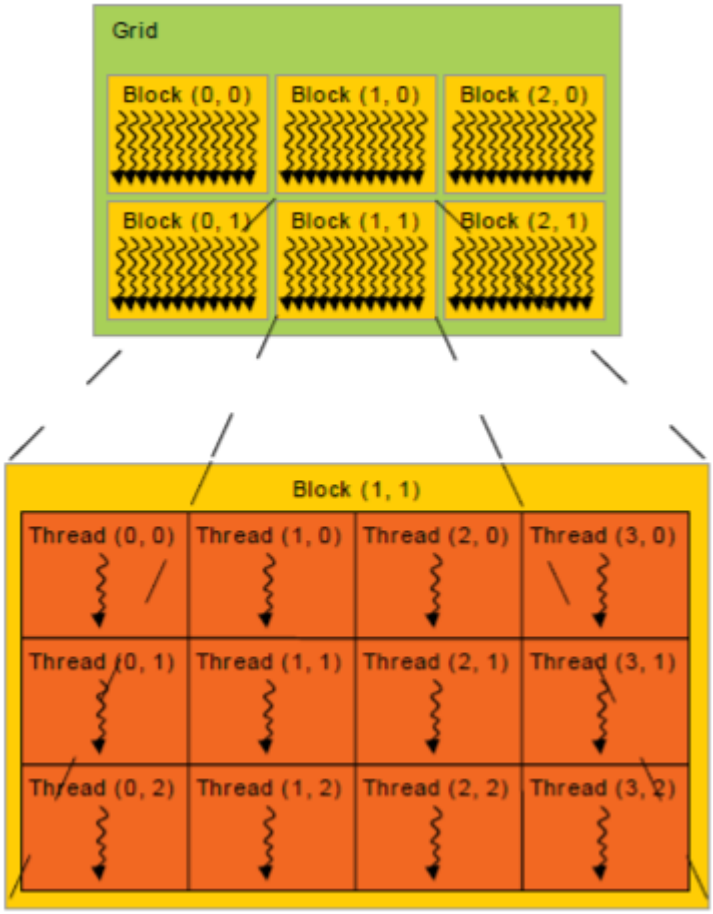



**Figure 2.3.** CUDA Grid.[11]

the execution is independent form each sub-regions. Therefore, it is possible to see that at the lower level a threads hierarchy is present, that helps to increase the computation performances.

NVIDIA GPU parallelism can be recalled as: "fine-grained[9] data parallelism and thread parallelism, nested within coarse-grained data parallelism and task parallelism"[11]. In paralleling computing a *fine-grained parallelism and thread parallelism* refers to a frequent data transfer between the processors, with small tasks in terms of code size and execution time. Meanwhile a *coarse-grained data parallelism and task parallelism* means that data communication is infrequently and takes places only after larger amounts of computation.

Having described previously the top and lower level framework, the basic pipeline for a GPGPU CUDA application is outlined:

[9]granularity is the amount of computation in relation to communication (or transfer) of data

1. The CPU host defines a N-dimensional computation region over the DRAM memory to set the index of the computation domain used to enumerate the work-groups and the work-items.

2. The host defines a group of work-items into work-groups, sharing the same local memory and working concurrently in the compute unit.

3. The hardware will load the DRAM memory in the GPU RAM and execute each block following the work queue.

4. The multiprocessor will execute only 32 threads at the time[10], when the work-group contain more threads a serialization will occur, effecting the local memory consistency.

The CUDA execution model reflects the capability of the GPU threads to be organized into working-groups sharing the same data-structure and local memory. Overall, this helps to manage the workload of the GPU and the frequent communication inside of it between the processors. Overall, this execution model reduce the working time and increase the GPU performances.

### 2.2.3 Scalable Programming Model

The latest multicore CPUs and manycores GPUs technologies aim to develop system with a large number of parallel processors. Thus to follow the increasing number of computing units available, the application softwares are becoming much complex in order to scale the massive parallelism request. Fortunately, CUDA parallel programming model is not effect by this issue, because it is design to maintain a familiar standard programming close to C. Hence, the same CUDA program can be executed with two different GPUs, with different number of SMs, with the same final results, but in terms of performances, the higher are the SMs in the GPU the lower is the computational time required.

Having a scalable programming model like CUDA involves that the CPU host must invoke a specific kernel that is used to enumerate, organize and distribute the threads into the different blocks to create the CUDA grid. Also, while this program scalability is executed a memory partitions is run to control the data flows inside the GPU. As thread blocks terminate their task, new blocks are launched on the vacated multiprocessors.

[10]NVIDIA calls them *"warp group"*.

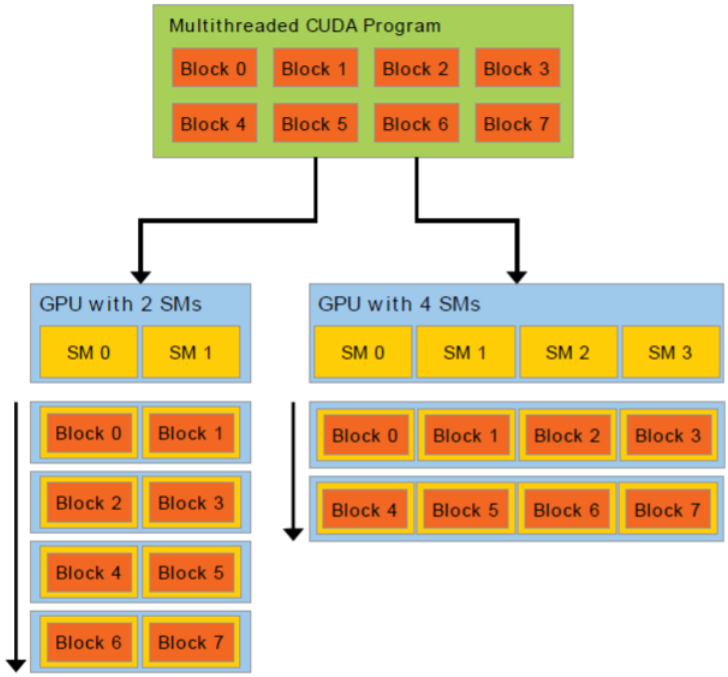



**Figure 2.4.** Scalability model.[11]

### 2.2.4 Hardware Implementation

Inside a multiprocessor unit, as a GPU, thousand of threads must be managed with unique architectures. According to the *Flynn's taxonomy*, it is possible to distinguish four parallel architectures based on how the *Instruction* and *Data Streams* are handled. These two streams can be: *Single* or *Multiple*. By combining all the possibility, four different classification are available:

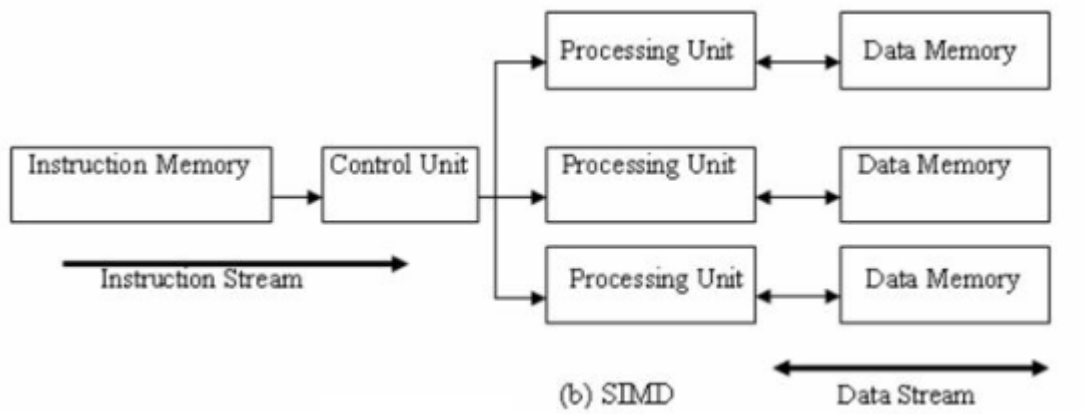
(a) Single instruction, single data (SISD)

(b) Single instruction, multiple data (SIMD)

(c) Multiple instruction, single data (MISD)

(d) Multiple instruction, multiple data (MIMD)

CUDA is based on the SIMD architecture. In this technology a single identical instruction is run over the threads that will execute the same code simultaneously with different data. In SIMD the data are store in blocks that are loaded in a single operation. Therefore, this architecture reduces the computational time considerably compared with standard CPU. Furthermore, threads are executed in groups of 32 parallel threads called *warps*. In the warp the threads have the same program address with their own specific instruction address counter and register state, being fee to branch and run independently.




ù

**Figure 2.5.** SIMD scheme.[11]

The NVIDIA warp group organization forces full computational efficiency only when all 32 threads work simultaneously. This limitation is reflected on the global problem work sizes, since the work-groups must fit evenly into the entire data structure. Furthermore, the work-group size must be less than the maximum threads that can work concurrently within a work-group[11]. If this condition is not achieved the host program will crash [12].

### 2.2.5 Memory Hierarchy

The memory hierarchy is structured in order to efficiently organize the parallel computational task. Each CUDA threads can access data from different memory space. Each thread has a private local memory, while each block has a shared accessible memory dedicates to the threads of the block during the execution routine. All threads have access to the same global memory and to two additional read-only memories: the constant and texture memory spaces. More in details, the memeoriy accesses are optimized for their memory usages. All of them are persistent when kernel are launched by the same application.

[11]The `CL_KERNEL_WORK_GROUP_SIZE` flag.
[12]In the worst case scenario the OS can crash.

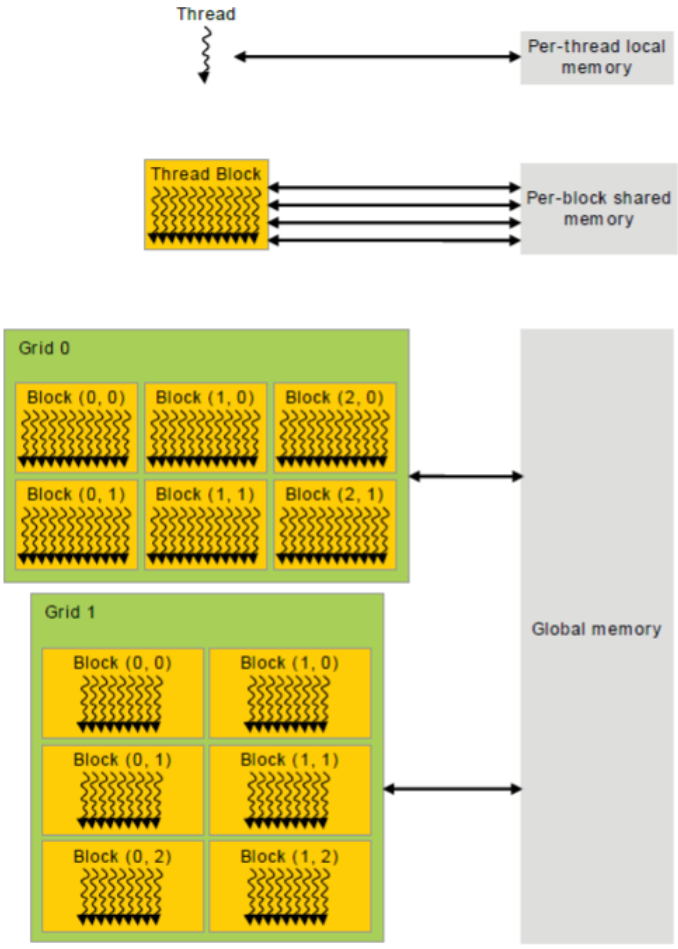



**Figure 2.6.** Memory hierarchy.[11]

### 2.2.6 Heterogeneous Programming

CUDA is defines as a "*Heterogeneous Programming*" framework because it uses more than one type of processor or core to execute the program. In fact, in the CUDA architecture the (CPU)Host and (GPU)Compute devices are different physical components. The Host device is where the program is launched and executed, meanwhile in the Compute device parallel computations are performed. Hence, the threads, that are located on the GPU, behave as coprocessors.

The two devices have their own separate memory space in DRAM: the host memory for the CPU and the computing memory device for the GPU. Therefore, the C program must have the ability to manage the device memory allocation and deallocation as well as data transfer between host and device. To simplify this program aspect a unified memory is used to bridge the host and computing device memories as a single one coherent memory image with a common address space. Thus, the CPU and GPU can easily access a common

memory space boosting the execution time by eliminating the need to explicitly mirror data on host and device.

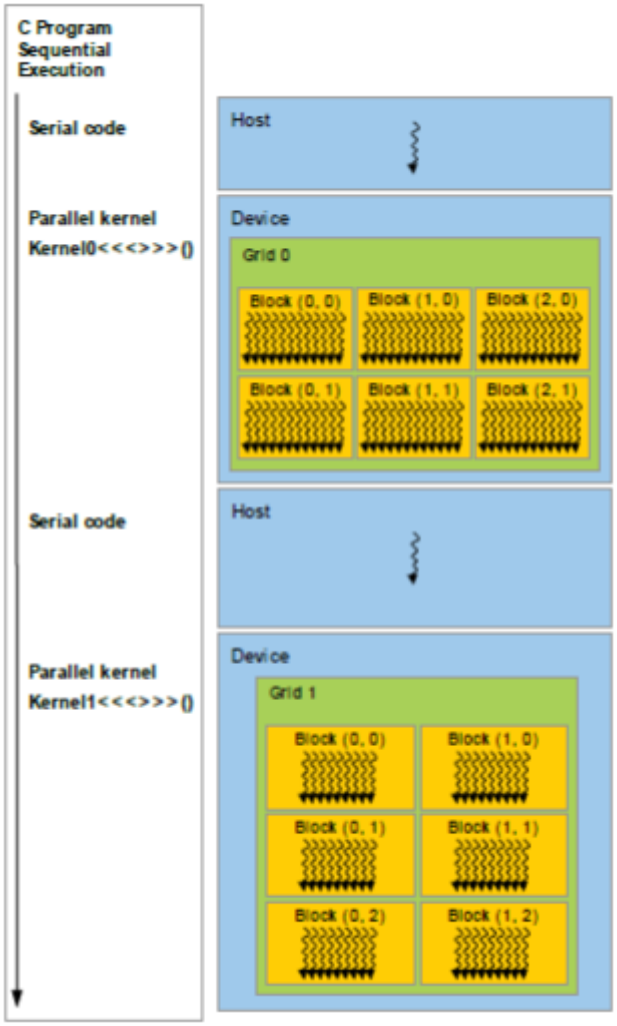



**Figure 2.7.** Heterogeneous Programming.[11]

## 2.3 Enthalpy HT LW-ACM: main code aspects

The Ethalpy HT LW-ACM formulation of **Algorithm 4** is highly suitable to be parallelized on GPUs, leading to less time consuming simulations with respect of standard methods. In practice, a code is written in C to invoke the kernels that control the GPU, that are written in the CUDA enviroment (`.cu`).

The serial implementation of the Ethalpy HT LW-ACM algorithm is realized using three `for loop`: one time loop, one loop to select the mesh point and the last to select the

surrounding nodes indexed by $\alpha$ in the lattice. Therefore, for each time step, when the central stencil node is selected the updating rules are applied to the neighboring stencil nodes one at the time. The updated local quantities of temperature, velocity, density and liquid fraction are stored. After the index `for loop` is completed the mesh points values can be computed. It is possibile to notice how the serial implementation is dependent of the number of nodes, meaning that increasing the domain dimension will negativetly effect the computation time and increase the request of computation power. By contrast, even if the time loop remains unchanged, in the parallel implementation the two spatial `for` loops are not used, because the updated properties are computed simultaneously for each node decreasing the computation time. Therefore, the parallel implementation is a much efficient way for the Enthalpy HT LW-ACM.

### 2.3.1 Main: main.c

The Host device controls the Computing device via the `main.c` C script, where the streams of data and the various instructions are launched from the CPU to the GPU. The GPU main instructions are grouped in the `compute.cu` CUDA kernel, that is launched on the `main.c` at each time step inside the time `for loop`. The main structure of the main is:

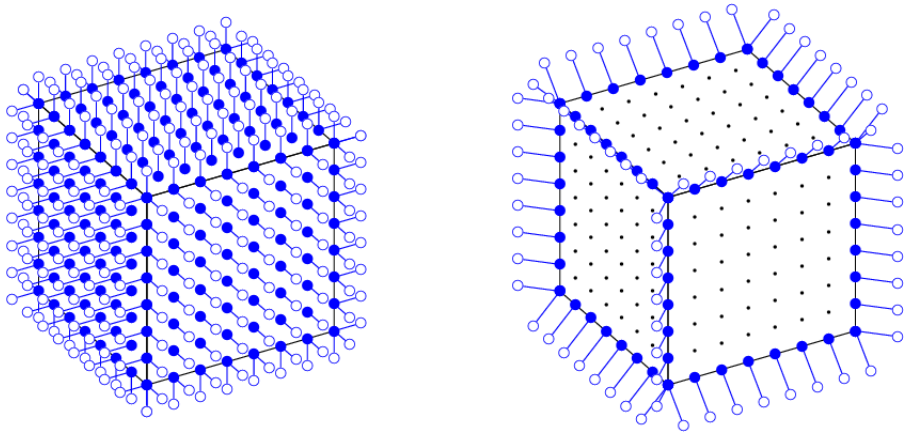
```
int main(int argc, char **argv)
{
    <Load the various parameters>
    <Load the computation domain>

    struct timespec start, stop;
    int t;
    double r = 1.;
    bool cvg = false;
    clock_gettime(CLOCK_REALTIME, &start);

    <Read the various parameters>
    <Read the computation domain>

    init_device(&D, P.device); //Device initialisation

    for (t = 0; t < P.T && !cvg; t += 2) {
    launch_compute_kernel(&D);
    ...
    }
    clock_gettime(CLOCK_REALTIME, &stop);

    <Export and print results>

    return EXIT_SUCCESS;
}
```

Furthermore, the time loop is created so that two instances of the same value apart of one time step are located in the global memory: one is used only to read data from

memory, the other to write data in memory. This specification is useful to compute the liquid fraction time difference that must be loaded in the temperature expression.

### 2.3.2 Computation kernel: compute.cu

To take advantageous of this massive parallelism, the GPU is instructed to assign each thread to a single node of the mesh to create the CUDA grid mapped from the computation domain. Inside the gird, the threads can be arrenged into a two dimensional grid of one-dimension blocks, but for three dimensional simulations a cube of size 8x8x8 is prefered to reduce the rate of data transfer increasing the efficency of the code, and in particular the threads block sizes are imposed since the coalesced memory accesses are issued by warps made of 32 threads. Also, to fit evenly the mesh points in the CUDA grid the number of nodes of the spatial mesh must be a multiple of 8 for each index. Hence, each block of threads is associated to a block of nodes[13]. The neighboring nodes of a block are referred as halos of the block, and they are used to link the different nodes blocks togheter: the threads on the surface of each single block are responsible for the nodes at the face of halo and the threads on the edges of the block handle the node at the edge of the halo.



(a) Faces of the halo. (b) Edges of the halo.

**Figure 2.8.** Data access to the halo of a block: the plain discs are the active threads, whether the hollow ones represents the nodes for which data are read.[4]

In practice, in a single block the threads first load from the device memory into the shared memory the temperature, velocity, density and liquid fraction arrays for the block and its halo. Then each thread carries out the read data operations for its single node. Once all data is available, the threads perform the computation instructions. Afterwards, the new values are written to the global memory.

[13]Since the nodes are multiple of 8, and the threads blocks are 8x8x8, the nodes block is a multiple of the threads block.

To ensure that all the threads have completed their task, which also means that all the operations inside a kernel are terminated, the synchronization barrier `__syncthreads()` is used to enque the work and wait for the completion.

```
__global__ void compute(float4* dr, float* tr, float* lr, float4* dw, float*
    tw, float* lw)
{
    // Initialisation ...

    get_coordinates(&x, &y, &z);
    g = get_geometry(x, y, z);

    <Load Momenta>
    <Load Temperature>
    <Load Liquid fraction>

    <Load the halo>
    __syncthreads();

    // Apply boundary conditions...
    <Apply BC>
    __syncthreads();

    //Computations...

    <Compute and store Temperature>
    <Compute Momenta>
    <Compute and store Velocity and Density>
    <Compute and store the Liquid fraction>
}

extern "C" void launch_compute_kernel(domain* D)
{
  <Launch kernel>
}
```

The `<Launch kernel>` operation consist in organizing the data stream. In fact the writing and reading arrays are not the same: the updated values are stored in `tw`, `dw`, `lw` arrays, that are then written on the reading arrays: `tr`, `dr`, `lr`. More in details the writing arrays store the values at the $t-2$ step, while the reading at the previous time step $t-1$:

```
    static dim3 block(S,S,S);
    static dim3 grid(D->Nx, D->Ny, D->Nz);

    compute<<<grid, block>>>(D->d0, D->t0, D->l0, D->d1, D->t1, D->l1);
    cudaThreadSynchronize();
    compute<<<grid, block>>>(D->d1, D->t1, D->l1, D->d0, D->t0, D->l0);
    cudaThreadSynchronize();
```

The momentum, temperature and liquid fraction of each mesh point are loaded in the kernel, refering to the computation node as `e00`:

```
m[e00] = dr[_(x,y,z)];        //momentum
t[e00] = tr[_(x,y,z)];        //temperature
l[e00] = lr[_(x,y,z)];        //liquid fraction
dl = l[e00] - lw[_(x,y,z)]; //Delta liquid fraction
```

The momentum array is a `float4structure` array that contains four `float` storing fields: three for the velocity and one for the density. While, the temperature and liquid fraction are two different `float` arrays. The `dl` is used to load the liquid fraction variation of the previous time step needed in (1.46).

The halo threads load the data of the neighboring nodes from the storage arrays in function of the node position inside the lattice. The node lattice index referes to the discrete velocity $\boldsymbol{\xi}_\alpha$: for instace using the D3Q19 stencil, the $15^{th}$ node, `e015`, loading data instructions are:

```
if (g & G_HALO) {
...
  m[e15] = dr[_(x,y+1,z+1)];
  t[e15] = tr[_(x,y+1,z+1)];
  l[e15] = lr[_(x,y+1,z+1)];
...
}
```

The boundary conditions are applied similarly to the way the halo nodes are assessed. According to the bounce back rule the wall node properties are bounced back to its simmetric node in the lattice. The temperature and liquid fraction wall nodes conditions are set as outlined in the previous chapter:

```
#define adia(a,b) (4/3*t[a]-1/3*t[b])

   if (g & G_WALL) {
...
//For adiabatic walls
    t[e05] = adia(e00,e06);
    l[e05] = 1;
...
//For heated walls
    t[e01] = T;
    l[e01] = 1;
}
```

After the data are loaded, the threads are free to perform the next set of instructions: temperature, velocity and density are computed[14] and their updated values are written in

[14]Those values are computed following the procedure of the computing algorithm: **Algorithm 4**.

the correspective storing arrays. Temperature is loaded in the `float tw` array, momenta in the `float4structure dw` array. Once the node properties are computed, the liquid fraction can be computed following the Enthalpy method, this values is then stored in the `float lw` array.

```
     if (tn < T0)
    lw[_(x, y, z)] = LFS;
     else if (tn > k5*(1 - l[e00]))
    lw[_(x, y, z)] = LFL;
     else
    lw[_(x, y, z)] = (tn - T0)/k5 + l[e00];
```

### 2.3.3 GPU setting: init.cu

In the main the computation kernel is launched only after the GPU settings are defined in the initialisation kernel: `init.cu`. This is also used to set the initial problem conditions and to define the size of the various arrays used in the storing and reading process.

```
__global__ void init(float4* d0, float* t0, float* l0)
{
    <Load Initial conditions>
}
void init_constants(domain* D){
   <Load the domain constant
}
extern "C" void init_device(domain* D, int id)
{
  <Set the GPU device>
  <Set the reading and storing arrays>
  <Set the  threads block>
}
```

The GPU device setting aims at defining the `id` of the selected Computing devices to launch all the computation required:

```
The GPU is set using:
    static dim3 block(S,S,S);
    static dim3 grid(D->Nx, D->Ny, D->Nz);

    D->id = id;
    cudaSetDevice(id);
    cudaSetDeviceFlags(cudaDeviceMapHost);
```

The initial conditions are imposed in the simulation inside the initialisation kernel. For melting the full solid inital domain at temperature $T_0$ is defined in the code as:

```
__global__ void init(float4* d0, float* t0, float* l0)
{
    int x, y, z;

```

```
    get_coordinates(&x, &y, &z);
    d0[_(x,y,z)] = momenta(1, 0, 0, 0);
    t0[_(x,y,z)] = T0;
    l0[_(x,y,z)] = 1;
```

The reading and storing arrays dimensions, as well as the threads block mapping are created according to the requirement dimension as:

```
extern "C" void init_device(domain* D, int id)
{
    init_constants(D);

    device_alloc(D->d0, D->size, float4);
    device_alloc(D->t0, D->size, float);
    device_alloc(D->l0, D->size, float);

    device_alloc(D->d1, D->size, float4);
    device_alloc(D->t1, D->size, float);
    device_alloc(D->l1, D->size, float);

    D->ix = NULL;
    if (D->ix_) device_alloc(D->ix, D->x_size, byte);
    D->iy = NULL;
    if (D->iy_) device_alloc(D->iy, D->y_size, byte);
    D->iz = NULL;
    if (D->iz_) device_alloc(D->iz, D->z_size, byte);
    D->jx = NULL;
    if (D->jx_) device_alloc(D->jx, D->x_size, byte);
    D->jy = NULL;
    if (D->jy_) device_alloc(D->jy, D->y_size, byte);
    D->iz = NULL;
    if (D->jz_) device_alloc(D->jz, D->z_size, byte);

    init<<<grid, block>>>(D->d0, D->t0, D->l0);
    init<<<grid, block>>>(D->d1, D->t1, D->l1);
    cudaThreadSynchronize();
}
```

# Chapter 3

# Validation

## 3.1 Introduction

The validation is a fundamental step when a new numerical method is elaborated to prove that it is a valid alternative to consolidated methods like analytical solutions. The proposed Enthalpy HT LW-ACM is a nouvelle approach that is confronted with the analytical Nuemann's method to test its effectiveness, consistency and validity. The Neumann's approach is chosen to be the reference method due to its capability on predicting the time dependent position of the moving phase change boundary based on the liquid fraction profile. Furthermore, being the thermal link-wise formulation validated in [6], and the Enthlapy method an established numerical approach, its implementation on the HT LW-ACM has to be validated.

Two main temperature based quantities are used to verify the implementation: the liquid fraction and the heat flux. The first is a *volumetric* quantity that expresses the phase change status of the domain. While, the heat flux is a *superficial* value used to describe the temperature effects on the domain.

## 3.2 Test case

For validation purposes the test case is chosen coherently to the analytical method constrains: the problem is formulated on a one-dimension domain, with simple initial and boundary conditions and constant thermophysical properties. In mono-dimensional phase change problems the buoyancy force is not assessed, omitting the advection phenomenon.

The link-wise formulation is been generalized for three-dimensional simulations on the D3Q19 stencil, but it can be rearranged for mono-dimension problem using the D1Q3 stencil, where three nodes are placed in order on the same axis.

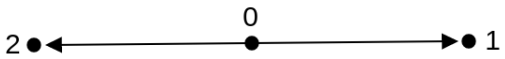



**Figure 3.1.** The D1Q3 stencil.

The coordinates of the discrete velocities and weights for this stencil are:

$$\boldsymbol{\xi}_\alpha = \begin{cases} 0 & \alpha = 0 \\ 1 & \alpha = 1 \\ -1 & \alpha = 2 \end{cases} \tag{3.1}$$

$$w_\alpha = \begin{cases} 4/6 & \text{for } \alpha = 0 \\ 1/6 & \text{for } \alpha = 1, 2 \end{cases} \tag{3.2}$$

The temperature, computed based on centered finite difference scheme, is computed omitting the advection part:

$$T_0^{t+1} = T_0^t + \kappa \left( T_1^t + T_2^t - 2T_0^t \right) - \frac{\Delta f^t}{Ste} \tag{3.3}$$

Meanwhile, the equilibrium general expression (1.33) and modified bounce back (1.47) rules are applied without any great modifications on the mesh points. Also, the Enthalpy updating rule is applied straight on, since it is used to compute the liquid fraction of one node by recalling its thermal conditions.

For consistency, to confront the analytical and numerical solutions the two results have to be scaled in the same units. The proposed numerical scheme is formulated in lattice units, while the Neumann's solution in physical units. Those can be scaled in the mesh-dependent units using the acoustic scaling techniques of (**Table (4.1)**): the thermal diffusivity $\kappa$ as $MaN/\sqrt{(3)}$, the problem size $L$ as $N$ and the temperature is expressed in the dimensionless units scaled with the $\Delta T$. Similarly, the physical evolution time is scaled in lattice units using $Fo$, through this it is possible to define the correspondence between the real time and the lattice time evolution.

The validation test case consists in a melting problem expressed in a mono-dimension domain made of 120 nodes at initial temperature $T_0 = 0$, which is completely solid: $f = 1$, $\rho = 1$ and $u = 0$ for all the nodes. The two nodes on the boundary $N = -1$ and 121 are at temperature $T_0 = 0$ and $T_1 = 1$ respectively, while the melting temperature is $T_m = 0.5$. For this test $Ste$ and $Pr$ are set to one, while $Ma = 1e-4$. The number of time iteration is $10e+4$. Since transport phenomena do not occur the buoyancy force has not to be computed through $Ra$.

## 3.3 Results

The results of the Enthalpy HT LW-ACM are confronted with Neumann's to validate it effectivness in phase change modelling. The node thermal status is shown in the temperature evolution comparison, while the liquid fraction is used for the domain phase status. From the total thermal flux $Q_t$ is possible to understand the amount of energy flux requeired for the phase change to occur.

The analytical and numerical temperature evolution at the last time step show how the two profiles have the same trend. In particular for the liquid nodes the link-wise formulation is very accurate for computing their thermal behavior. This result is also highlighted by the fact that the liquid node closer to the phase change interface has the same temperature in both models. However, these results are not obtained for the solid nodes because the two thermal profiles differs greatly.

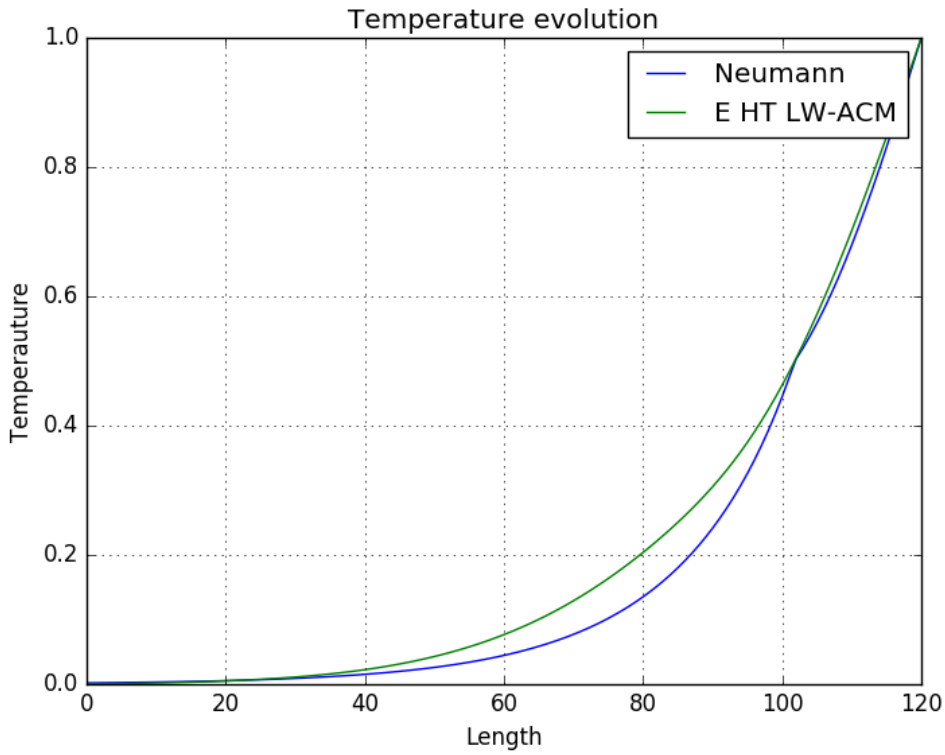



**Figure 3.2.** Temperature at $t = 10e + 4$.

The reason behind the discrepancy of the temperature profiles lays on how the two methods solve the phase change problem. Neumann's temperature is computed using (1.14) if the node is liquid otherwise (1.13) for solid, where the node status is computed based on its temperature: a node is liquid if $T_i > T_m$, whether if $T_i \leq T_m$ it is solid. By

contrast, the numerical temperature is calculated with (1.41) at each node and the phase status is based on the liquid fraction.

In terms of the thermal flux $Q_t$, the numerical and analytical results are the same, showing that the local temperature variation does not effect the global domain energy.

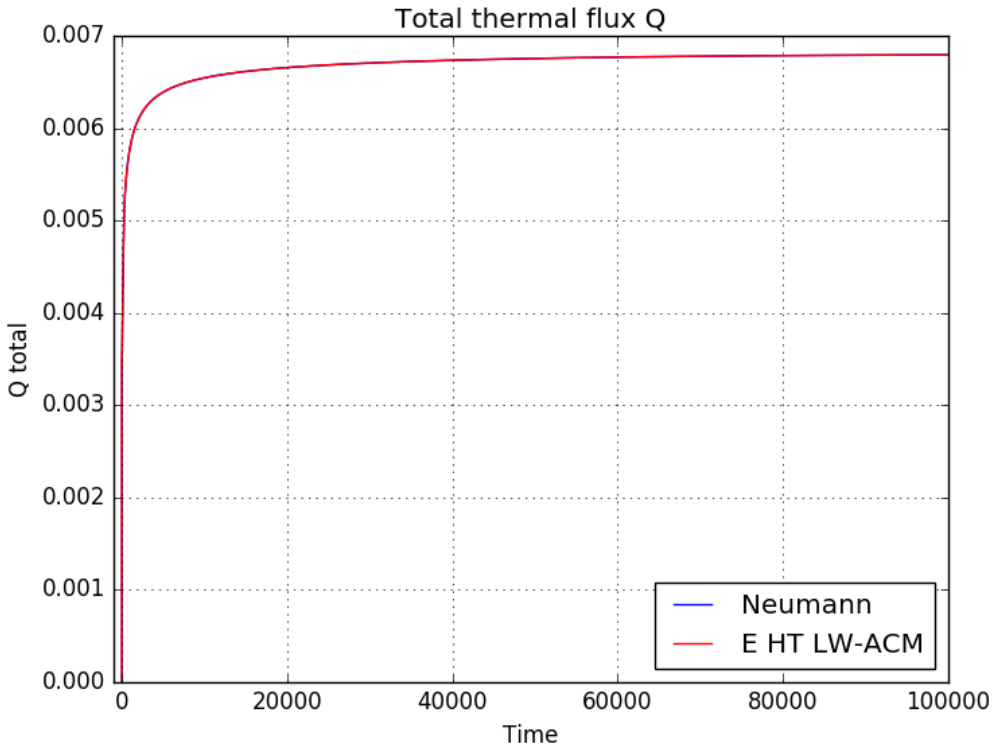



**Figure 3.3.** Total thermal flux.

The local value of the liquid fraction computed with the proposed scheme is perfectly overlapped to the numerical result, in details the nodes close to the interface have the same values for both methods, so the phase change status inside the domain of the Enthalpy HT LW-ACM follows exactly Neumann's.

Furthermore, for the numerical result the total liquid fraction follows the exact solution an globally the same amount of liquid fraction is computed at the same time step for both methods. Similarly, the position of the interface, based on the liquid fraction scaled over the domain follows the predicted position of the Neumann's method, which ensures that the phase change occurs at the time of the exact solution.

These comparisons highlight that even if the temperature profiles diverge the domain status is not effected, moreover the link-wise formulation is able to model correctly the phase evolution of the nodes being in synchronous with the Neumnann's prediction.

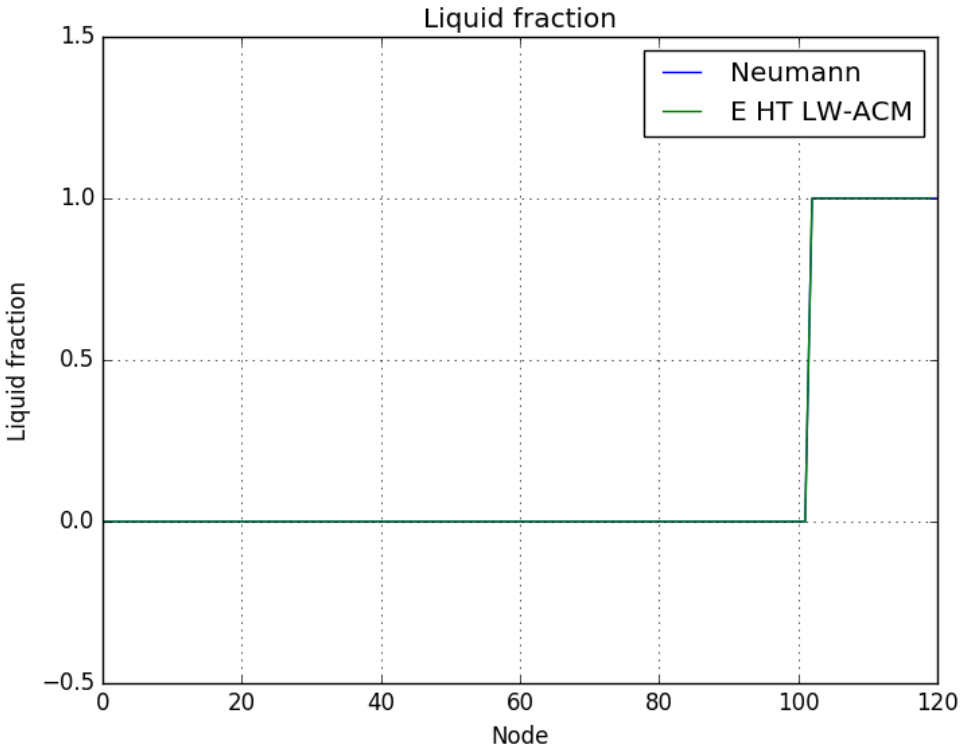



**Figure 3.4.** Liquid fraction at $t = 10e + 4$.

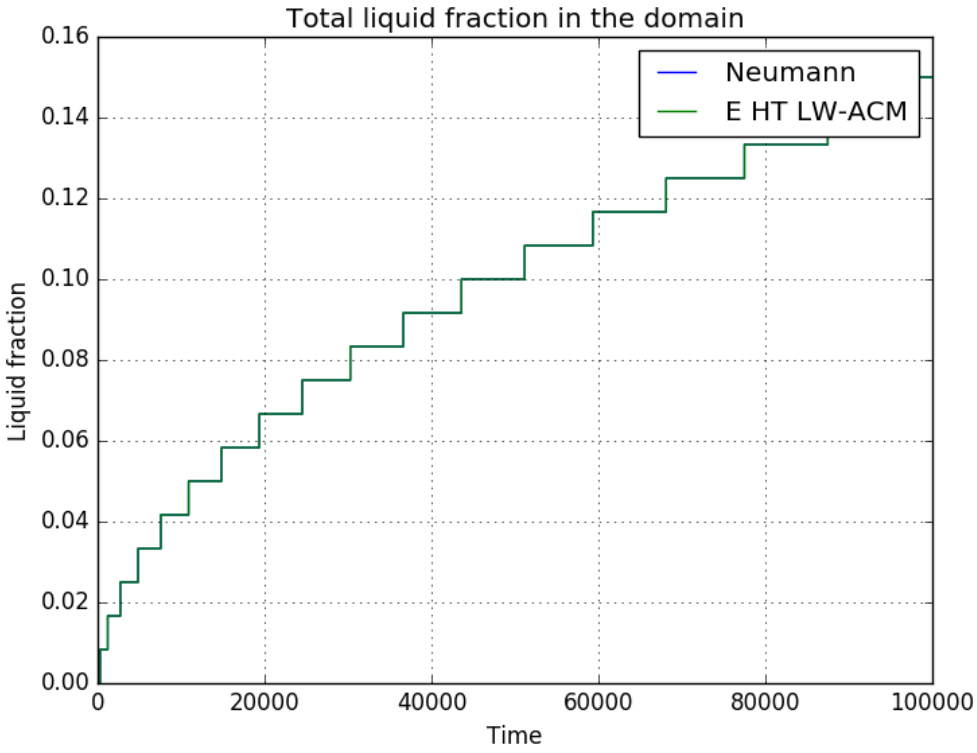



**Figure 3.5.** Total liquid fraction profile.

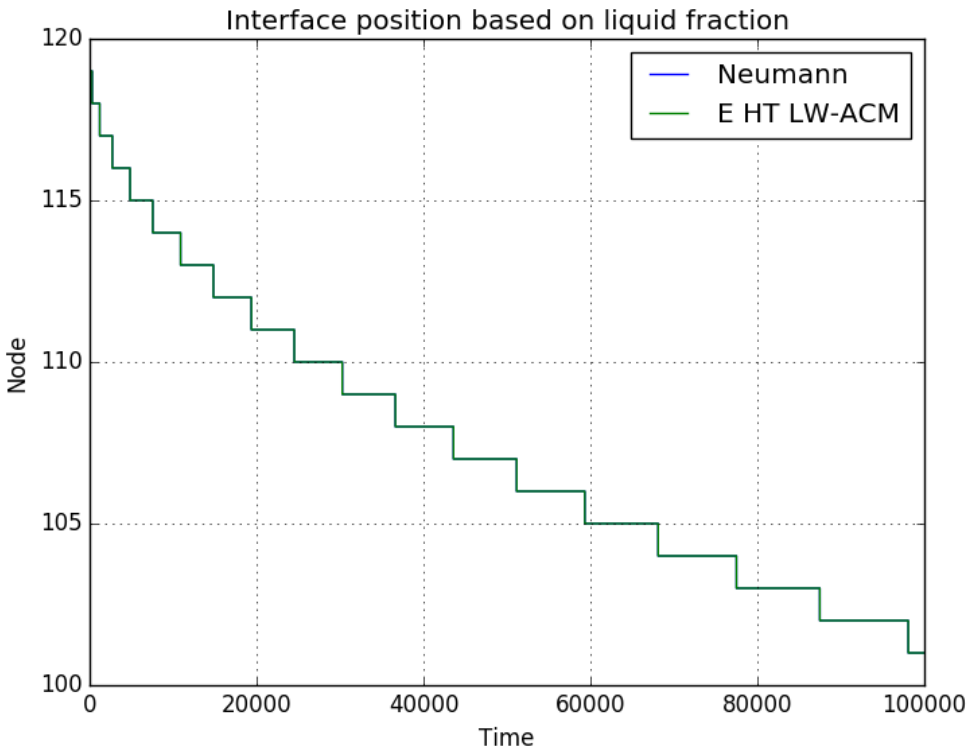



**Figure 3.6.** Interface time position.

## 3.4 Conclusions

The validation process aims at ensuring that the proposed Enthalpy implementation on the HT LW-ACM is a valid tool to model phase change problems, which is performed by confronting the numerical and exact analytical results.

The test case is chosen coherently with the analytical constrains: problem dimension, constant properties and no buoyancy force. Therefore, the one dimension test is studied by adapting the Enthalpy HT LW-ACM on the Q1D3 stencil. Furthermore, to confront the results the Neumann's solution is scaled using the acoustic scaling, which enables to formulate the exact solution in terms of lattice units.

The results show how the temperature profiles do not perfectly match Neumann's: the solid nodes temperature is higher in the numerical solution, but the node on the phase change interface is aligned with the exact value of the temperature. The reason behind the divergence in the temperature in the domain is based on how the numerical and analytical methods compute the temperature profiles: the analytical method is based on two

temperature functions depending on the node status, while the numerical temperature is computed using a single function regarding the node status updated with the Enthalpy method to compute the liquid fraction variation. Another important result is obtained with the total heat flux, which shows that the amount of energy required for both methods for the phase change is the same at each time step. Hence, in terms of energy the link-wise formulation converges to the exact solution and the temperature profiles discrepancies do not effect these simulations. Furthermore, the liquid fraction and interface confrontation consolidate these results since the domain status profile follows with great accuracy the Neumann's interface and liquid fraction. In conclusion the Enthalpy HT LW-ACM is an effective method to solve the moving boundary problem to model PCM.

# Chapter 4

# Simulations

## 4.1 Introduction

The Enthalpy HT LW-ACM is the proposed numerical model to solve real phase change problems, which occur in multidimensional domains without omitting the presence of the buoyancy force and transport phenomena.

To run a simulation the thermal properties of the material have to be set, which are scaled in lattice units, following the proper scaling technique. In practice, for the acoustic scaling the dimensionless parameters to compute the problem quantities are: the number of nodes for each index $L_x \times L_y \times L_z$, $Ma$, $Ste$ and $Pr$ for the thermal properties and $Ra$ for the magnitude of the buoyancy force. Moreover, from the energy equation and its numerical solution the stability range of the model is defined. Furthermore, for the spatial mesh the number of nodes $N$ for each indexes has to be a multiple of 8 because it has to fit evenly in the 8x8x8 CUDA grid defined on a warp of 32 threads.

Once the stability conditions are defined various simulations are performed to understand how the algorithm behaves by changing the setting parameters and the thermal conditions of the problem.

## 4.2 Numerical analysis

The melting problem main condition is that inside the material the temperature always increases until the phase change stops as the thermal equilibrium is reached. More in details, if this condition is not respected the simulation is not physically correct and the results become unstable.

The temperature for each node is computed using (1.46):

$$T_{i,j,k}^{t+1} = T(\boldsymbol{x} - \boldsymbol{\xi}_\alpha, t) - \frac{1}{Ste}(f_{i,j,k}^{t} - f_{i,j,k}^{t-1})$$

where the thermal properties are computed using the acoustic scaling (4.1). The liquid fraction variation is inversely proportionate to $Ste$, and if $T(\boldsymbol{x} - \boldsymbol{\xi}_\alpha, t)$ is constant, as $Ste \ll 1$ the mesh point temperature value decreases and become negative, which is not physical possible. Therefore, the acoustic scaling limits the range of the application due to this effect, but for real applications $Ste$ ranges between 0.5 to 0.01 depending on the $\Delta T$. Thus, to respect the melting condition and surpass the limit of stability for $Ste$, a different scaling practice is applied: the non-homogeneous acoustic scaling techniques.

| Physical | Dimensionless(') | Lattice(*) |
|---|---|---|
| $\boldsymbol{u}$ | $Ste\alpha/L$ | $Ma/\sqrt{3}Ste$ |
| $L$ | 1 | $N$ |
| $t$ | $Fo/Ste$ | $FoN\sqrt{3}/MaSte$ |
| $\nu$ | $Pr$ | $MaPrN/\sqrt{3}Ste$ |
| $\kappa$ | 1 | $MaN/\sqrt{3}Ste$ |
| $\delta x$ | $1/N$ | 1 |
| $\delta t$ | $Ma/(N\sqrt{3})$ | 1 |
| $c_s$ | $1/MaSte$ | $1/\sqrt{3}Ste$ |
| $T$ | $(T - T_0)/\Delta T$ | $(T - T_0)/\Delta T$ |

**Table 4.1.** Non-homogeneous Acoustic scaling table.

Where $N$ represents the nodes along the $y$ direction, perpendicular to the buoyancy force computed as:

$$\boldsymbol{g}^{(*)}\beta^{(*)} = \frac{Ra\nu^2}{PrN^3\Delta T} = \frac{RaMa^2}{3N\Delta TSte^2} \tag{4.1}$$

This modified version of the acoustic scaling is achieved by dividing the thermal characteristic by $Ste$ in order to express the entire phase problem as a function of this value, and the issues related to the liquid fraction weight for $Ste \ll 1$ are solved.

For the numerical temperature expression the melting condition is defined with the $T^t_{i,j,k}$ coefficients, which have to be positive to ensure correct simulations. The thermal diffusivity $k^{(*)}$ is limited as:

$$(1 - 9\kappa)T_{i,j,k} \geq 0$$

using the non-homogeneous acoustic scaling the stability condition is expressed as:

$$\kappa \leq \frac{1}{9} \Rightarrow Ma \leq 3^{-\frac{3}{2}}\frac{Ste}{N} \tag{4.2}$$

More in details, the relaxation time $\tau$ is effected by this results: using $Pr$ the $\tau - \kappa$ relation is:

$$\kappa = \frac{\nu}{Pr} = \frac{1}{2}\left(\tau - \frac{1}{2}\right)$$

and its constrain is:

$$\kappa \leq \frac{1}{9} \Rightarrow \tau \leq \frac{Pr}{3} + \frac{1}{2} \tag{4.3}$$

For $Pr \ll 1$ the relaxation time is $\tau \approx 1/2$, which is the stability limit value to force the viscosity to be positive[8]. However, even if the stability range is respected the simulations is susceptible to some numerical instabilities.

## 4.3 Multidimensional simulations

Multidimensional simulations are performed on a three dimension cube domain, which is divided into a finite numbers of nodes $L_x \times L_y \times L_z$. Initially the domain is full solid and kept at the solid wall temperature $T_0 = 0$, and the buoyancy force acts along the $z$ direction. For the boundary conditions, opposite to the cold wall, the hot wall temperature is $T_1 = 1$, while $T_m = 0.5$. The remaining faces of the cube are adiabatic.

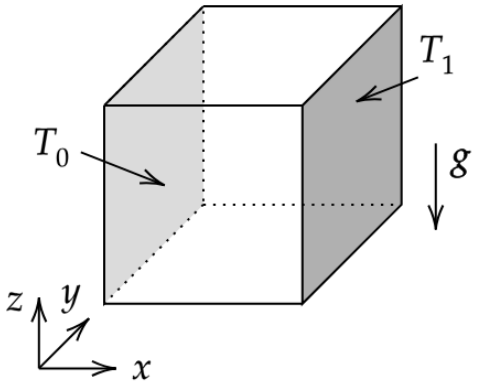



**Figure 4.1.** Three dimensional simulation domain.

The simulation parameters are set to respect the stability conditions for a 64x64x64 nodes cube domain, and different materials can be simulated using the scaling table to scale the thermal properties in the more convenient lattice units. The dimensioless parameters are: $Pr = 1$, $Ma$ spans from $10e-3$ to $10e-6$ and $Ste$ between $1-0.5-0.1$. In terms of stability as $Ste$ decreases so does $Ma$, and to avoid numerical instabilities it is chosen to be always lower respect to (4.2)[1].

### 4.3.1 Simulations without advection

The reliability of the multidimensional simulations is analyzed by comparing the numerical results with Neumann's on the $Ma - Ste$ plane. In practice, various simulations are made with variable values of $Ma$ and $Ste$, meanwhile the other setting paremeters are unchanged. The quantity that is used in the comparison is the interface position, which is computed for the three dimensions as the average value in the domain. Also, the buoyancy

[1]Sperimentally, it is been proven that if $Ma \ll 3^{-3/2} Ste/N$ the simulations are stable, and the simulation time increases.

force and the advection phenomena are not account, solving the conduction driven phase change problem.

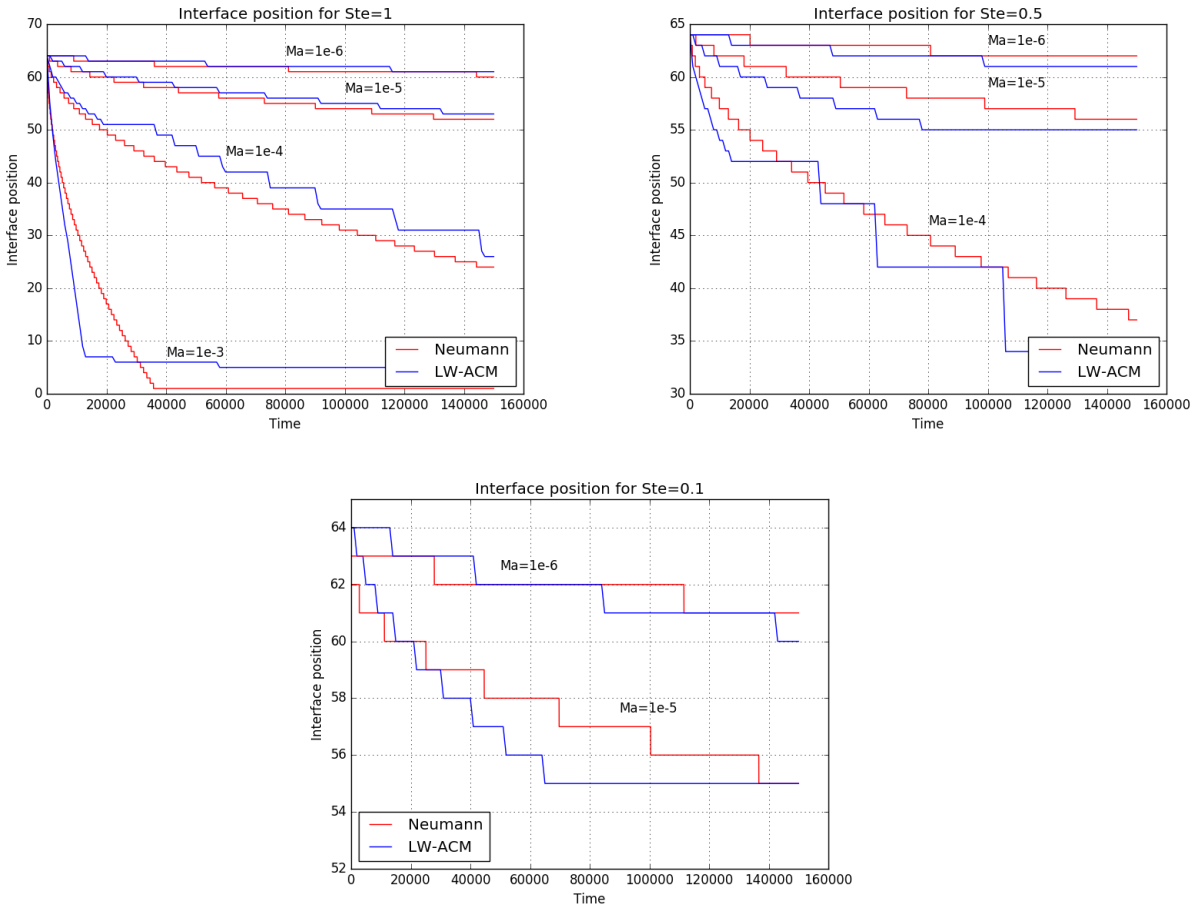



**Figure 4.2.** Interface position for simulations without advection, for variable $Ste$ and $Ma$.

The comparison demonstrates that the numerical multidimensional simulations are reliable since the link-wise formulation is able to follow the Neuamann's interface position. In particular, the velocity of fusion is directly proportionate to $Ma$: as this value decreases the interface position moves slower in the domain. This relation lays on the scaling technique for which the problem quantities are expressed in lattice units as a function of $Ma$: for example the thermal diffusivity $\kappa^{(*)}$ is proportional to $Ma$, so for small values of it the amount of thermal energy decrease slowing the melting front.

The simulations stability is directly related to $Ste$. In fact, as this value goes to zero, to obtain stable results, the maximum value of $Ma$ decreases: for example if $Ste = 0.1$ the simulations are stable only for $Ma \geq 3\ 10e - 3$, this reduces the kind of simulations that can be performed. The reduction of $Ma$ is sliglhtly compensated with the reduction of $Ste$

for the thermal properties of the material. However, it is not possible to state whether the fusion velocity decreases if both $Ste$ and $Ma$ decreases simultaneously.

### 4.3.2 Simulations with advection

Real phase change processes are subjected to transport phenomena related to the presence of the buoyancy force. In the Enthalphy HT LW-ACM it is possible to count the effects of buoyancy force by correcting the equilibrium function (1.43). Also, by comparing the interface position of the simulations with and without advection, it is possible to understand how transport phenomena really effect the melting process.

The effects of advection increases with the buoyancy force magnitude, which is directly proportional to $Ra$, that has to be positive for natural convection. In particular for the simulatons is set at $10e + 6$.

The bouyancy force scaled in lattice units (4.1) is directly proportionate to $(Ma/Ste)^2$. For stability, if $Ste$ is reduced also a smaller value of $Ma$ has to be imposed. Thus, their effects in the bouyancy force are balanced when both values decreases. Furthermore, if $Ste$ is constant and $Ma$ decreases the magnitude of the bouyancy force is reduced and the velocity of fusion is effected. By contrast, keeping $Ma$ constans while $Ste$ changes, the effects of transport phenomena increase and the fusion front moves faster.

For high values of $Ma$ the advection phenomenon plays an important role in the phase change problem increasing the velocity of melting inside the domain. More in details, in the first time step of the melting process transport phenomena do not occur because the liquid domain is not sufficient developed. For example, for $Ste = 1$ and $Ma = 1e-4$ for the $12000^{th}$ step, the melting front advances as the phase change interface without bouyancy force. However, as fusion advance the effects of advection start to deeply effect the melting front. From the liquid fraction position until the $20000^{th}$ time step the advective simulation follow the conduction driven model. By contrast, from the simulation cross section, the liquid front is effected by the presence of transport phenomena, but being the interface position computed as the average value in the domain, the two values correspond.

Small values of $Ma$ reduce the magnitude of the bouyancy force and the effects of advection. In particular, for $Ste = 0.5$ and $Ma = 1e - 5$, the melting node corresponds to the conduction simulation, but the cross section shows that even if the effect of bouyancy are reduced, still transposrt occurs in the domain. While the average interface position is comparable to the other model. Therefore, as $Ma$ and $Ste$ decrease the liquid front slows down and the thermal effects of advection are reduced.

The results show that in presence of transport phenomena the melting interface moves inside the domain faster respect to the conduction driven problem. In fact, in presence of

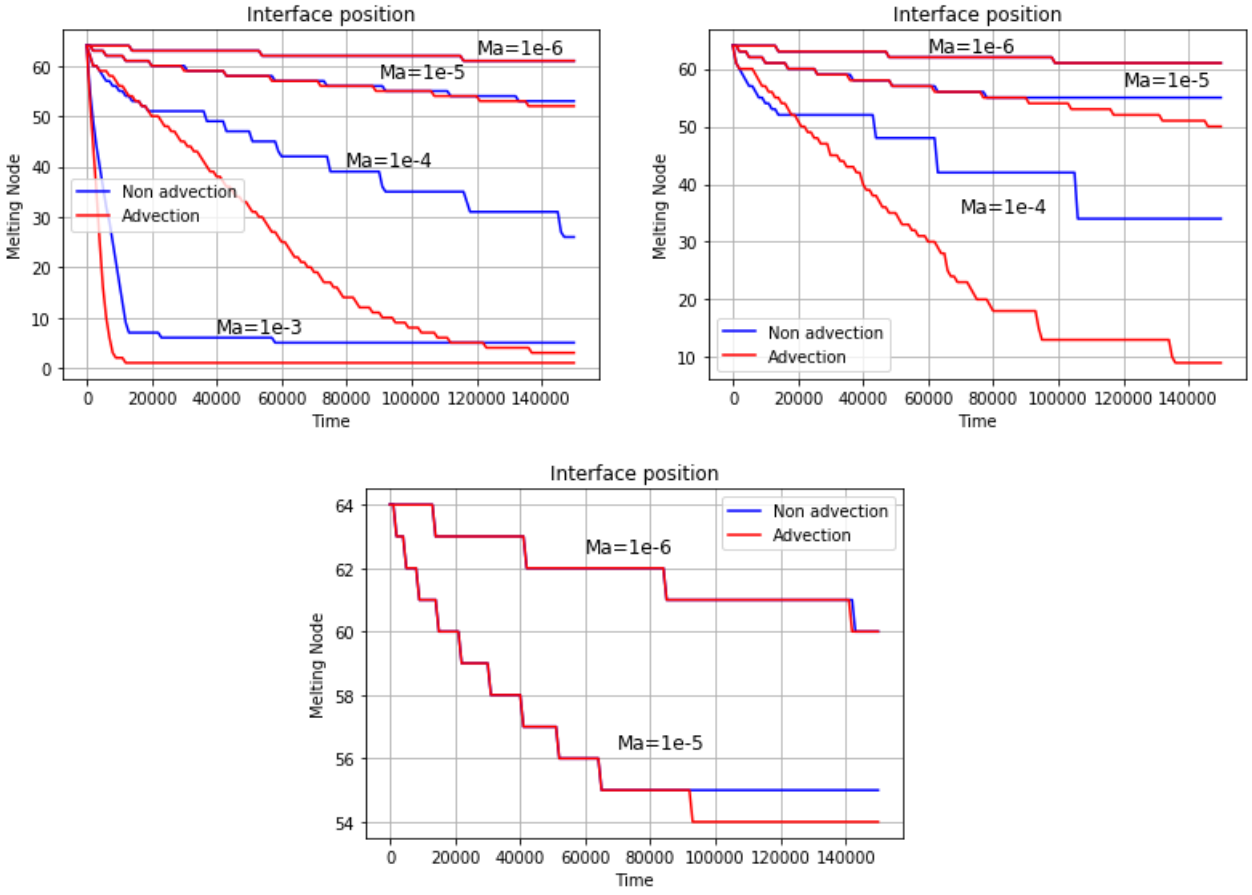



**Figure 4.3.** Interface position for simulations with advection, for variable $Ste$ and $Ma$.

transport the temperature of the node is computed counting also the effect of advection, which increases the nodes temperature. Therefore, the presence of advection add more thermal energy increasing the phase change process velocity. By contrast, as the simulation is slowed down, with a small valus of $Ma$, the effects of the buyoancy force decrease and the interface position almost matches the position of the conduction driven simulations.

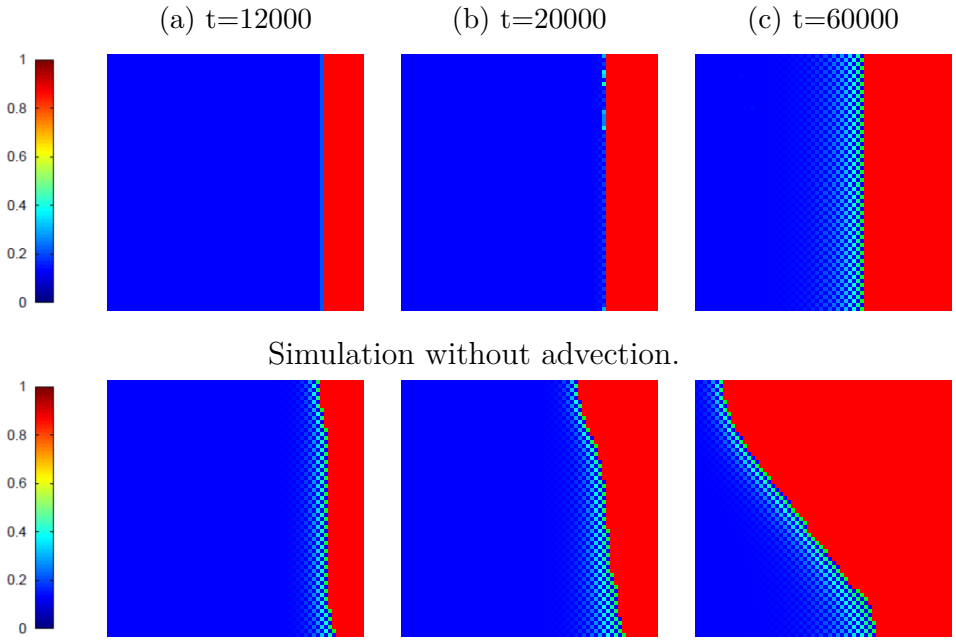



**Figure 4.4.** Liquid fraction corss section for $Ste = 1$ and $Ma = 1e - 4$.

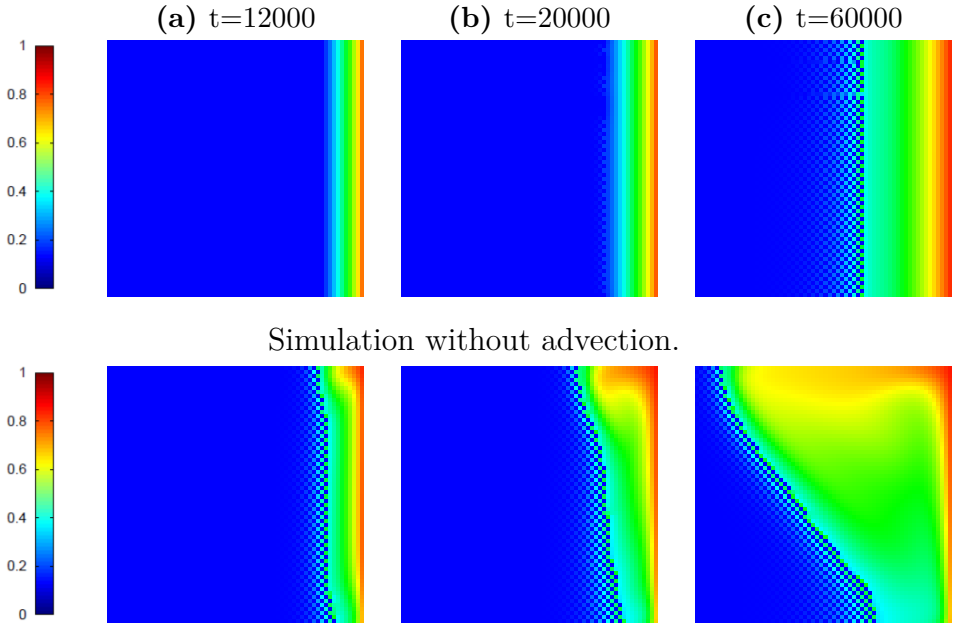



**Figure 4.5.** Isothermal cross section for $Ste = 1$ and $Ma = 1e - 4$.

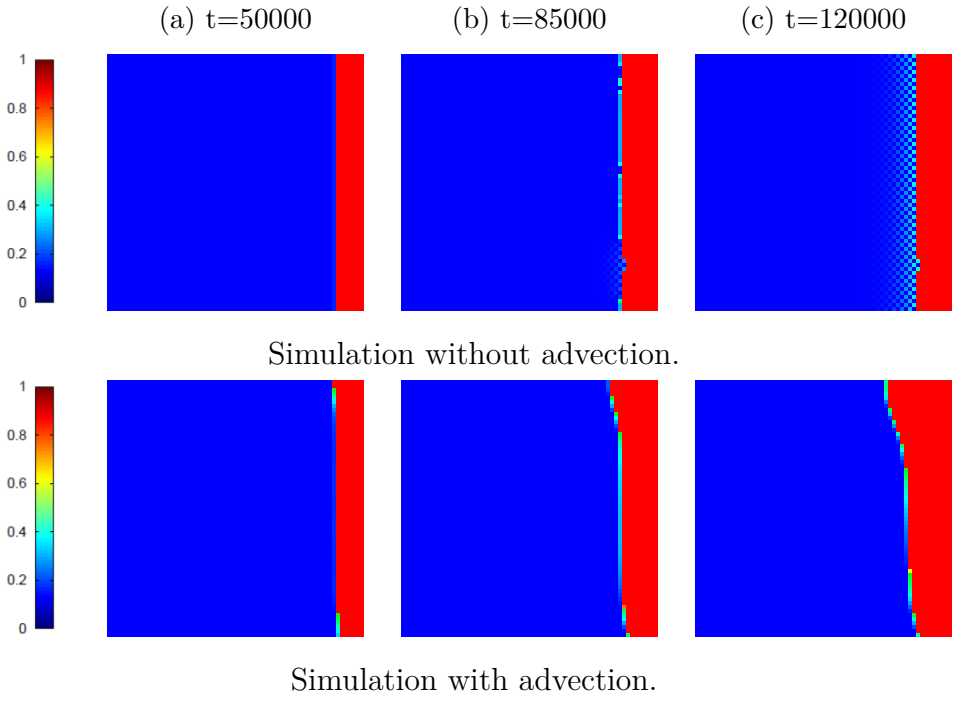



**Figure 4.6.** Liquid fraction corss section for $Ste = 0.5$ and $Ma = 1e - 5$.

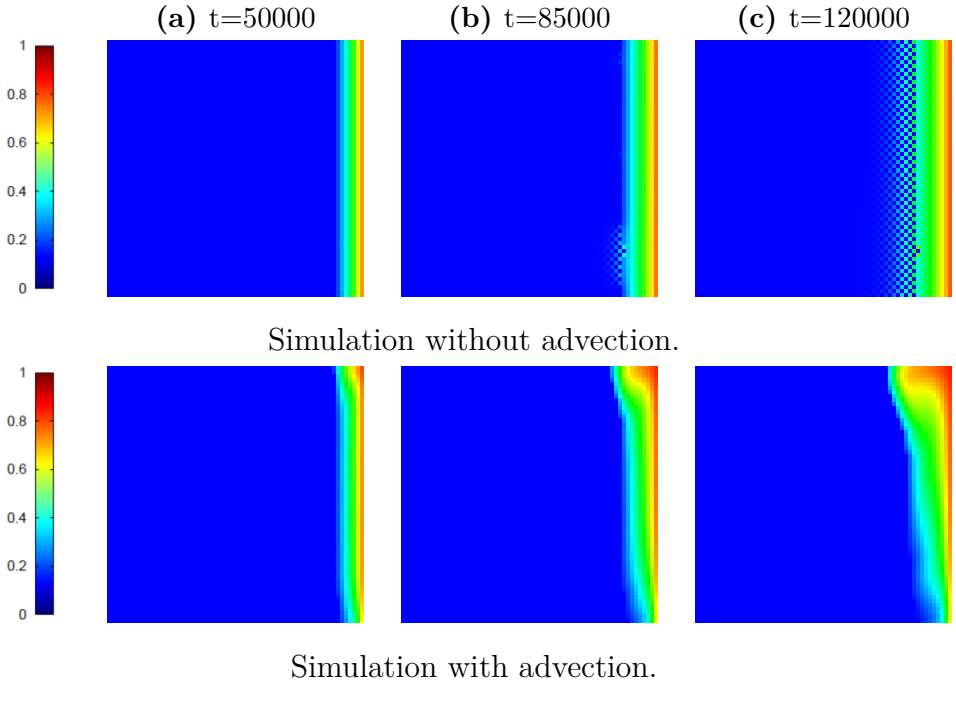



**Figure 4.7.** Isothermal cross section for $Ste = 0.5$ and $Ma = 1e - 5$.

### 4.3.3 Simulation performances

The simulation performances are typically expressed in terms of the computational time and MLUPS, which stands for *Million Lattice Updates per Second*: for example if on a one million lattice sites at a speed of one MLUPS, the lattice is updated once per second[9]. Therefore, this quantity helps to understand the computational demand of each simulation. Meanwhile, the computational time is expressed in seconds and defines the total simulation time from when the code is lunched, until the last command is executed. Overall, the performaces change singificantly according to the set of lattice parameters in the $Ma - Ste$ plane.

| $Ste$ | **Time** $[s]$ | | | |
|---|---|---|---|---|
| 1 | 37,3297 | 29,5576 | 26,4521 | 25,5506 |
| 0,5 | - | 31,077 | 27,5098 | 26,5121 |
| 0,1 | - | - | 28,6364 | 26,9949 |
| $Ma$ | 1,00e-03 | 1,00e-04 | 1,00e-05 | 1,00e-06 |

**Table 4.2.** Computational time for the only conduction driven simulations.

| $Ste$ | **Time** $[s]$ | | | |
|---|---|---|---|---|
| 1 | 36,0743 | 34,1296 | 26,4896 | 25,42 |
| 0,5 | - | 35,3026 | 28,1378 | 26,0154 |
| 0,1 | - | - | 27,3007 | 25,9232 |
| $Ma$ | 1,00e-03 | 1,00e-04 | 1,00e-05 | 1,00e-06 |

**Table 4.3.** Computation time for the complete simulations.

The simulations are performed using an NVIDIA GTX TITAN BLACK. The computational time decreases of in the best scenario 30% as $Ma$ decreases from $1e-3$ to $1e-6$. While the variation of $Ste$ does not effect considerably the simulation performances. Also, the computational time is not effected whether the advection part of the solution is considered or omitted. In details, the maximum time discrepancy for the two type of simulations is around 4.5 $[s]$ for $Ma = 1e-4$ and $Ste = 1$.

The MLUPS increase as the simulation velocity is reduced with $Ma$, with a maximum increment of almost 50% for $Ste = 1$. Simirarly with the computational time performances, the simulations are not deeply influenced by $Ste$, and by the presence of bouyancy force.

| $Ste$ | **MLUPS** | | | |
|---|---|---|---|---|
| 1 | 1053 | 1330 | 1487 | 1539 |
| 0,5 | - | 1265 | 1429 | 1483 |
| 0,1 | - | - | 1373 | 1457 |
| $Ma$ | 1,00e-03 | 1,00e-04 | 1,00e-05 | 1,00e-06 |

**Table 4.4.** MLUPS performance for the only conduction driven simulations.

| $Ste$ | **MLUPS** | | | |
|---|---|---|---|---|
| 1 | 1090 | 1152 | 1484 | 1547 |
| 0,5 | - | 1114 | 1397 | 1511 |
| 0,1 | - | - | 1440 | 1517 |
| $Ma$ | 1,00e-03 | 1,00e-04 | 1,00e-05 | 1,00e-06 |

**Table 4.5.** MLUPS performance for the complete simulations.

## 4.4 Conclusions

This chapter is dedicated to analyze the stability constrains and results of various simulations. The solution of the energy equation is possible using the Enthalpy HT LW-ACM for which the problem is scaled in the lattice units via the acoustic scaling table. However, this techinque is not suitable for real applications for which $Ste \ll 1$ because it will limit the stability range of the simulations. Hence, to solve this problem the non-homogenous acoustic scaling techinique is used for which all the lattice units are divided to $Ste$ to balance its effects when it goes to zero.

The stability conditions are expressed from the scaled temeperature function to solve the energy equation, for which the coefficiencts for the conduction term of the central node of the lattice are imposed to be always positive. Then using the scaling techinque, it is possible to compute the limit valus of $Ma$ for which the solution is stable. However, even if this conditions is respected, the simulations can be subjected to numerical instabilities. Furthermore, defining the maximum values of $Ma$ enables to define the relaxation time through $Pr$ and the realtion between the kinetic viscosity and relaxation frequency.

For fusion problems, according to the dimensioless setting parameters different types of materials and thermal conditions can be simulated. In fact, reducing $Ste$ for the same materials it means to reduce the total temperature difference inside the domain which will effect the simulations: a small difference in temperature between the hot and cold wall reduces the amount of energy at disposable in the domain and the fusion process is slowed down. The melting time increases for simulations where $Ste \ll 1$.

Meanwhile, if the value of $Ma$ decreases the melting velocity reduces, because of the acousting scaling techinques the thermal properties are directly proportionate to $Ma$. Also, if it is reduced the velocity of the liquid nodes decreases effecting the overall phase change.

The only conduction driven simulations are confronted with the results obtained with the Neumann's method. Since the numerical method follows the analytical one, the multidimensional simulations are reliable in the $Ma - Ste$ plane. Furthermore, the variation of $Ste$ effects the maximum values of $Ma$ for the simulation stability.

For more realistic simulations the bouyancy force has to be plugged in the link-wise formulation to asses the presence of transport and advection phenomena. Since, the bouyancy force is proportional to the square of the $Ma - Ste$ ratio, its magnitude is effected when these values change. However, since $Ma$ is defined according to the stability conditions, which are proportional to $Ste$, if this values reduces so $Ma$ follows, which is then compensated in the bouyancy force.

Transport and advection are directly proportionate to the bouyancy force an so to the $Ma - Ste$ ratio. In practice, reducing the relative speed of the fluid nodes decreases the effects of transport and advection. Therefore, the simulations for which $Ma \ll 1$ behaves like the conduction driven results. From the cross sections it is possible to see the presence of transport phenomena, even if the overall average liquid fraction is the same.

In conclusion, the presence of the bouyancy force plays a predominant role in the fusion velocity for high value of $Ma$ and transport phenomena deeply influece the phase change. If the fluid velocity is sufficiently high the liquid front advance faster in the domain, decreasing the total melting time respect to the conduction driven model.

# Conclusion and perspectives

The main aim of this thesis is to develop a new numerical model to simulate melting materials in a multi-dimensional domain with natural convection while studying the influence of the setting parameters to analyze different type of materials.

Phase change problems can be model using the energy equation, which correlates the conduction and advective heat fluxes with the temperature and liquid fraction variation of the domain. To solve this differential equation the domain is discretized into mesh points linked over the lattice stencil and the material properties are scaled in lattice units. The model is based on the link-wise formulation to compute the velocity field of the liquid nodes, on finite difference operators to define the temperature profile, and on the Enthalpy method to differs whether a node is liquid, solid or melting with the uploading node phase status rule.

The Enthalpy HT LW-ACM is highly suitable for parallel application to increase the computing performance. Using the CUDA enviroment, the code is structure in a such a way that each processor of the GPU is dedicated to a single mesh point and performs the task is assigned concurrently with the other processors. To increase the code efficiency the domain dimensions are forced to be a multiple of the threads block to achieve better memory access decreasing the computation time while exploiting all the computation power at disposal. To achieve those results the synchronization barrier helps to enque work and wait for the command to complete. For the code, after the setting parameters are lunched, the main kernel is called to perform the temperature, velocity, density and liquid fraction computations using the numerical expressions.

The numerical method verification is made possibile by comparing the simulation results with the ones obtained using the analytical Neumann's method. In details, the verification is necessary to test the effectiveness and validity of the Enthalpy implementation over the already esthablished HT LW-ACM. The solution is computed omitting transport and advection phenomena on a mono-dimensional domain, with simple initial and boundary conditions. From the temperature profile comparison the numerical and analytical solution behaves in the same, which is then confirmed by the total thermal flux, liquid fraction and interface position plots.

For the 3D simulations the acoustic scaling is modified to obtain simulations that are less effected by numerical instabilities when $Ste < 1$. For this part the test case is a cube with two isothermal walls at different temperature, while the remaining ones are adiabatic. To test the affidability of the multi-dimensional model on the $Ma - Ste$ plane the simulations are run considering only the conduction phenomena for melting materials. Finally, more realistic simulations are performed by plugging also the advection part of the energy equation in the solution, that shows the increment of heat transfer and of the fusion front respect to the conduction driven models. Furthermore, the computation times are effected by decresing as $Ma$ goes to zero, in contrast with the behavior of the MLPUS.

# Perspectives

The development and validation of the Enthalpy HT LW-ACM must be followed by an experimental validation as a necessary perspective, also to create a reliable reference solution that does not exists in literature. Hence, the results can serve as basis for future applications.

The various simulations show how conduction and convection effect the phase change behavior of the material, omitting the part of energy related to radiative heat transfer. In fact, the domain walls heat by conduction and radiation the inner material increasing its temperature. Also, on the fusion front, the same heat transfer mechanisms between liquid and solid occur. Therefore, more realistic phase change phenomena can be model by incorporating in the model the radiative heat[8].

# Bibliography


[1] Mathematical modelling of solidification and melting: a review. -H. Hu; S. A Argyropoulos. -Modelling and Simulation in Materials Science and Engineering, Volume 4. -Number 4(1996), pp. 371-396. Printed in the U.K. http://iopscience.iop.org/article/10.1088/0965-0393/4/4/004/pdf

[2] Mohamad Muhieddine, Edouard Canot, Ramiro March. Various Approaches for Solving Problems in Heat Conduction with Phase Change. International Journal on Finite Volumes, Institut de Mathématiques de Marseille, AMU, 2009, pp.19. <hal-01120384> https://hal.inria.fr/hal-01120384

[3] Link-Wise artificial compressibility method. -P. Asinari; T. Ohwada; E. Chiavazzo; A. F. Di Rienzo. -In: JOURNAL OF COMPUTATIONAL PHYSICS. -231(2012), pp.5109-5143. http://dx.doi.org/10.1016/j.jcp.2012.04.027

[4] High-performance implementations and large-scale validation of the link-wise artificial compressibility method. -C. Obrecht; P. Asinari; F. Kuznik; J. Roux. -In: JOURNAL OF COMPUTATIONAL PHYSICS. -275(2014), pp. 143-153. http://dx.doi.org/10.1016/j.jcp.2014.06.062

[5] Thermal link-Wise artificial compressibility method: GPU implementation and validation of a double-population method. -C. Obrecht, P. Asinari, F. Kuznik, J. Roux. -In: COMPUTERS AND MATHEMATICS WITH APPLICATIONS. -72(2016), pp. 375-385. http://dx.doi.org/10.1016/j.camwa.2015.05.022

[6] Hybrid thermal link-wise artificial compressibility method. -C. Obrecht; F. Kuznik. In: PHYSICS LETTER A. -379(2015), pp. 2224-2229. http://dx.doi.org/10.1016/j.physleta.2015.07010

[7] An introduction to the OpenCL Programming Model. -J. Tompson; K. Schlachter. https://cims.nyu.edu/~schlacht/OpenCLModel.pdf

[8] Johann Miranda Fuentes. Développement d'un modèle de Boltzmann sur gaz réseau pour l'étude du changement de phase en présence de convection naturelle et de rayonnement. INSA de Lyon, 2013. Français. . <NNT : 2013ISAL0032>. <tel-00961213> https://tel.archives-ouvertes.fr/tel-00961213/document

[9] Accelerating Lattice Boltzmann Fluid Flow Simulations Using Graphical Processors. -P. Bailey; J. Myre; S. D. C. Walsh; D. Lilja; M. O. Saar. -In: 2009 International Conference on Parallel Processing. http://dx.doi.org/10.1109/ICPP.2009.38

[10] Comprehensive Review of Thermal Energy Storage. -I. Sarbu; C. Sebarchievici. -In: Sustainability (Switzerland), Volume 10. (1-2018). http://dx.doi.org/10.3390/su10010191

[11] CUDA C PROGRAMMING GUIDE: Design Guide. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[12] CUDA by Example: An Introduction to General-Purpose GPU Programming. Book by: J. Sanders; E. Kandrot. http://www.mat.unimi.it/users/sansotte/cuda/CUDA_by_Example.pdf