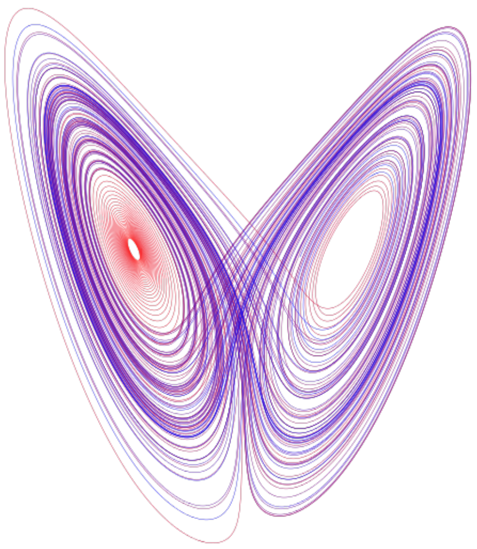
Unstable Periodic Orbits in Turbulent Hydrodynamics

Luis Alexandre Mendes Fazendeiro

<l.fazendeiro@ucl.ac.uk>



A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

Department of Chemistry

University College London (UCL)

2010

Signed declaration

I, Luis Alexandre Mendes Fazendeiro, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signed:

Luis Alexandre Mendes Fazendeiro

Abstract

In this work we describe a novel parallel space-time algorithm for the computation of periodic solutions of the driven, incompressible Navier-Stokes equations in the turbulent regime. Efforts to apply the machinery of dynamical systems theory to fluid turbulence depend on the ability to accurately and reliably compute such unstable periodic orbits (UPOs). These UPOs can be used to construct the dynamical zeta function of the system, from which very accurate turbulent averages of observables can be extracted from first principles, thus circumventing the inherently statistical description of fluid turbulence.

In order to identify these orbits we use a space-time variational principle, first introduced in 2004. This approach has not, to the best of our knowledge, been used before on dynamical systems of high dimension because of the formidable storage and computation required. In this thesis we describe the utilization of petascale high performance computation to the problem of applying this space-time algorithm to hydrodynamic turbulence.

The lattice-Boltzmann method is used to simulate the Navier-Stokes equations, due to its locality, and is implemented in a fully-parallel software package using the Message Passing Interface. This implementation, called HYPO4D, was successfully deployed on a large variety of platforms both in the UK and the US with an extremely good scalability to tens of thousands of computing cores. Based on this fluid solver other routines were developed, for the systematic location of suitable candidate spacetime minima and their numerical relaxation, using the gradient descent and conjugate gradient algorithms.

Following this methodology, several UPOs are identified in homogeneous turbulence driven by an Arnold-Beltrami-Childress force field in three spatial dimensions, at Reynolds numbers corresponding to weakly-turbulent flow. We characterize the transition to turbulence in the ABC flow and the periodic orbits computed, for a flow with $Re = 371$, after the transients have died down. The work concludes with a discussion of the potential for this approach to become a new paradigm in the study of driven dissipative dynamical systems.

The *viva voce* examination was held on the 16^{th} of September 2010. The examiners were Dr. Ian Halliday from the Materials & Engineering Research Institute, Sheffield Hallam University and Dr. Robert Bowles from the Department of Mathematics, UCL.

The picture on the title-page shows a rendering of the time integration of the Lorenz equations, a simplified model of convection rolls arising in atmospheric flows, which has played a fundamental part in modern dynamical systems theory. The attractor on these equations has been widely studied and has been found to be replete with unstable periodic orbits.

In Memoriam Rafael Patrício

(1975-2003)

“Etant données des équations ... et une solution particulière quelconque de ces équations, on peut toujours trouver une solution périodique (dont la période peut, il est vrai, étre trés longue), telle que la différence entre les deux solutions soit aussi petite qu’on le veut, pendant un temps aussi long qu’on le veut. D’ailleurs, ce qui nous rend ces solutions périodiques si précieuses, cest qu’elles sont, pour ansi dire, la seule bréche par où nous puissions esseyer de pénétrer dans une place jusqu’ici réputée inabordable.”

– Henri Poincaré [1]

“I, whose calling was really only that of a violinist storyteller, was responsible for the provision of music for our group, and I then discovered how a long time devoted to small details exalts us and increases our strength. ”

– Hermann Hesse [2]

Acknowledgements

This work would not have been possible without the help and collaboration of several people, at different stages and in completely different degrees. Apologies to anyone I may have forgotten to mention.

I must start by thanking my supervisor, Prof. Peter V. Coveney, for his immense patience and constant encouragement and, above all, for having given me the opportunity to work on such an exciting subject.

Prof. Bruce Boghosian's ideas and support were crucial at all stages of this work and collaborating with him has been by far one of the most inspiring and enriching aspects of this project. A special thanks also for his hosting my two visits to the Tufts University Campus in Medford, Massachusetts, in 2006 and 2007.

I am greatly indebted to Prof. Frank Smith for acting as my second official supervisor and for providing valuable suggestions and input at various stages of this work.

My thanks to everyone at the Centre for Computational Science. In particular, I would like to thank Radhika Saksena for discussions regarding nearly all aspects of this project. Her friendship and infectious love of science acted as a constant example and inspiration. I am grateful to Marco Mazzeo for many discussions in the earlier stages of the project, regarding the lattice-Boltzmann method as well as programming and optimization techniques.

Thanks to Owain Kenway for his generous help and availability regarding all things digital and "grid-ish". Simon Clifford's technical support was also crucial in the first stages of my work in the CCS. Thanks to Tom Hewer for tennis, sometimes under very dire circumstances indeed, and to Dave Wright and Kashif Sadiq, for poker and for sharing their enjoyment of research. I am greatly indebted to Nilufer Betik not only for her round-the-clock impeccable administrative support but also for her warmth and friendship.

Sincere thanks to all the members of the EPSRC project "User-Friendly Authentication and Authorization for Grid Environments" for their support, patience during my seemingly endless presentations and for many useful comments and suggestions. I am also specially thankful to Feng Hao and Ali Haidar for their kind friendship and encouragement. I would also like to acknowledge discussions with Jonas Lätt with whom it has been a great pleasure to work with.

I am extremely grateful to the UK National Grid Service in general and, in particular, to Matt Ford from Manchester, and Jason Lander from Leeds, for help and technical support, and Ning Li, from NAG in Oxford. In the US, I would like to thank Nick Karonis and Brian Toonen of the Argonne National Laboratory in Chicago and Jay Boisseau and the TACC support team, for their kind help in accessing and utilizing Ranger. Thanks to Pete Beckmann and Ramesh Balakrishnan, at ALCF in Argonne, for their kind help and support in accessing the IBM Blue Gene/P cluster.

On a more personal note, thanks to all my friends and family, specially to everyone at “faina londrina” and the vibrant Portuguese community in London. I would like to acknowledge the support of João Moreira ($\LaTeX$ “expert” extraordinaire), Alexandre Lindote and Luis Artur Pereira for their help and availability.

Last but not least, this work would not have been possible without the support of my dearest friend and flatmate Pedro Galvão. Thanks also to my good friends João Pedro Pinto and Azuza Dannohara for helping to make home such a pleasant environment to return to every day. And to Inch, for his special collaboration in the latter stages of writing.

I am highly indebted to the UK’s Engineering and Physical Sciences Research Council for paying my stipend in the context of grant EP/D051754/1.

This work is dedicated to my dearly missed friend Rafael Patrício, who will always remain a constant source of strength and inspiration to everyone who knew him, and who kept watching over my shoulder while I was writing it.

Published work

This thesis is the product of my own work, unless otherwise stated. It is based in part on work described in the following refereed/to be refereed publications.

- L. Fazendeiro, B. M. Boghosian, P. V. Coveney and J. Lätt, Unstable Periodic Orbits in Weak Turbulence, *Journal of Computational Science*, **1**, 13–23, 2010.
- R. S. Saksena, B. M. Boghosian, L. Fazendeiro, O.A. Kenway, S. Manos, M. D. Mazzeo, S. K. Sadiq, J. L. Suter, D. Wright and P. V. Coveney, Real Science at the Petascale, *Phil. Trans. R. Soc. A*, **367** (1897): 2557–71, 2009.
- B. M. Boghosian, P. V. Coveney, L. Fazendeiro, J. Lätt, J. Tam and H. Tang, A computational program for the Dynamical Systems Approach to Turbulence, *Preprint*, 2009.
- L. Fazendeiro, B. M. Boghosian, P. V. Coveney, J. Lätt and H. Tang, Search for Unstable periodic orbits in the Navier-Stokes equations, *Proceedings of the TeraGrid'08 conference, Las Vegas, June 9-13*, 2008, URL: http://archive.teragrid.org/events/teragrid08/Papers/papers/46.pdf.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Background

Fluid dynamics is arguably one of the most crucial areas of science and technology, present in areas ranging from the flow of blood in the human body to very large flows of plasma inside stars. The most common distinction is the one that separates laminar flows from turbulent ones. In laminar flows the velocity and pressure fields are characterized by a low time dependence. Such flows are easily predictable and smooth. Turbulent flow on the other hand exhibits highly time-dependent velocity and pressure fields which appear in large measure unpredictable. This can give rise to chaotic behaviour and thus to a wealth of complex spatio-temporal patterns that have fascinated mankind from times immemorial.

The transition between these two states can in most cases be quantified by a dimensionless control parameter that expresses the ratio between inertial and viscous forces within the fluid. In this work we are concerned with the study of turbulent flow. Whenever laminar flow is addressed it will be used as a benchmark or to better illustrate a given point in the work.

The study of turbulence can be described as one of the most important fundamental problems still faced by current research, sometimes hailed as the last great unsolved problem from classical mechanics [3]. Besides the huge theoretical challenge, it is of great practical relevance in areas as diverse as weather forecasting, transport and dispersion of pollutants, gas flows in engines, blood circulation, cosmological flows and many others.

As pointed out by several authors, e.g. [3, 4], it is remarkable that the fundamental equations which are widely believed to implicitly contain all of turbulence dynamics (the Navier-Stokes equations) have been known for more than 150 years, with so few exact results following from them.

Most engineering applications in which turbulent behaviour is relevant are usually focused in finding ways to minimize or suppress such behaviour. In a typical application, such as flow of a single component fluid through a pipe, the onset of turbulent behaviour increases the rate of energy dissipation thus requiring more work to be performed on the system in order to maintain the flow rate. However, turbulence can also play a useful role, for example, by enhancing the mixing of different components in a combustion reaction.

Computational simulation and analysis provides us with ways to test theoretical models and to visualize scales of matter (in both space and time) which are out of bounds to experimental studies. The case of turbulent hydrodynamics illustrates this well, being one of the first problems to be simulated in modern computers, from the 1940s onwards [5]. Due to the huge number of degrees of freedom active in turbulent flow, it becomes extremely difficult to track any quantities experimentally. The present work describes a computational investigation of some aspects of flow behaviour in this regime and describes and illustrates a novel way to address the problem.

The computational complexity scales with a power of $\sim 9/4$ of the number of degrees of freedom and typical flows that can be seen in everyday life are still out of the reach of present day computers [6]. As in the case of protein folding (another cutting edge problem which is also described by the equations of classical mechanics), the time taken to simulate a relatively small system can quickly become unfeasible.

Questions that can be easily stated, such as what are the heat transfer properties of a turbulent flow or the force applied by a fluid to its boundary in this regime, still continue to evade any general answers. After the onset of turbulence, fluid flow becomes intrinsically chaotic, varying randomly both in space and in time, making it extremely difficult to extract universal dynamical properties. In this context, we would like to find some easily reproducible flows that could represent the actual turbulent state.

The importance of periodic orbits in dynamical systems has been recognized at least since the work of Poincaré [1]. The attracting set [7] for a driven dissipative system can be thought of as the closure of the set of all the unstable periodic orbits (UPOs) of the system. These UPOs provide a countable sequence of orbits and can therefore provide a useful characterisation of the structure and dynamics of the attractor [8]. Following the renewed interest in dynamical systems that began in the 1960's, a concerted picture has emerged of how to obtain averages from these UPOs in a systematic fashion. An excellent introduction to the formalism and main ideas of this approach, which will be discussed in Chapter 2, can be found in the book by Cvitanović *et. al.* [9].

For the sake of clarity we will sketch here the main lines of thought, which will be developed later on in this work. It is widely accepted that a driven dissipative dynamical system will spend most of its time in the neighbourhood of the UPOs for that system. How long it will hover around one of them will depend on the stability eigenvalue of that UPO, as explained in more detail in section 2.3. These orbits are highly unstable, otherwise the whole system could easily become locked in periodic behaviour, something which is not observed in turbulent dynamics.

Following this approach, it was the purpose of this work to test a novel methodology for computing UPOs in turbulent flow, with the long-term goal to construct a digital library of these orbits, in order to compute any given observable that we wish. Although this is, in computational terms, very expensive to achieve, it needs to be done only once. Indeed, this approach has the potential to ultimately turn the study of turbulence into an activity of digital curation, since every time we wish to compute some other average of the flow we will only need to perform a summation on the UPO library, with no need to redo the initial value problem. The other great advantage is that this computation converges exactly to the true value, with an accuracy that depends on the number of lower period UPOs we include.

It is one of the main goals of this study to establish the feasibility of this methodology and discuss what further steps can be made in order to take full advantage of it.

1.2 Outline of the work

The present work is an application of the theory of dynamical systems to 3D incompressible viscous flow [10]. In section 2.2 we present the Navier-Stokes equations, and discuss some of their properties, as well as the main approaches that have been used to simulate turbulent flow. In section 2.3 we introduce some concepts of dynamical systems theory which are relevant to this work, in particular the dynamical zeta function formalism (DZF) which allows us to extract averages of the time-dependent flow from the UPOs of a given system.

The literature on the subject of UPOs in dynamical subjects comprises a huge body of work. Most of it can for practical purposes be divided in two main (broad) categories. The first one is the identification of such orbits in experimental data, usually in the form of time series. One avenue where this approach has been particularly successful is neuronal dynamics [11, 12]. So *et al.* [12] suggest that UPOs can be used as a natural symbolic representation of the states of a complex system. This methodology is then applied to the electrical activity of cells from the hippocampus of rats and the large-scale activity from human cortical electroencephalographs. Another fruitful application lies in the control of chaos in experimental situations where the knowledge of periodic orbits is used to enhance the performance of a given system [13].

The other large category where these ideas have been applied focuses on the discussion of numerical methods to track UPOs and the accuracy of the predictions thus obtained on systems with (relatively) few degrees of freedom, such as the Lorenz model [14, 15] and shell models of turbulence [16], which approximate the more difficult problem of fluid turbulence. We discuss this issue in section 2.4, where variational methods for identifying UPOs are described.

The first authors to identify UPOs within the Navier-Stokes equations were Kawahara and Kida, who published their findings in 2001 [17]. The particular configuration considered by these authors was plane Couette flow (flow between two parallel planes with a fixed relative velocity between them acting as the driving force on the viscous fluid) within a weakly-turbulent regime. In the aforementioned work and its continuation [18] the authors describe several periodic solutions and report good agreement between the averages obtained through these solutions and the whole time sequence of the flow. Examples of the computed quantities include the Reynolds shear stress and the mean velocity and vorticity components [17]. Their findings illustrate that periodic solutions can indeed be found in turbulent 3D hydrodynamics. This served as an initial inspiration for the present work and will be referred to often throughout.

Several methods to track and identify UPOs have been proposed in recent years, which will be discussed in section 2.4. In this work we chose to use the variational approach, following the method suggested by Lan and Cvitanović [19]. This variational principle is described in section 2.4.2. We discuss there how to efficiently parallelize time and space, by simultaneously searching for the UPO and its period in phase space. This is achieved by minimising a well defined functional that measures the distance in phase space between the trajectory and a neighbouring UPO.

This novel approach to turbulence studies requires a very substantial amount of resources. A collection of time slices of the system must be loaded onto the (RAM) memory of a large supercomputer. Since the flow studied here is three-dimensional this collection of slices, which we will call orbit from now on, can be said to be 4D. As shown in Chapter 5, for turbulent flow even the smallest UPOs already have periods of a magnitude that place the whole 4D orbit size in the domain of terabytes of data.

Two things follow immediately from this. The first one is that since we are dealing with such vast amounts of data the code we use must be as efficient and scalable as possible. In order to achieve this we have chosen to use the lattice-Boltzmann method to simulate incompressible fluid flow. This has several advantages, discussed in detail in Chapter 3. Perhaps the most relevant to this work lies in its requiring only communication between nearest neighbours on a computational grid, thus making it highly parallelizable. Using some of the world's current largest supercomputers we have demonstrated the scalability of the code developed in this project up to several tens of thousands of cores. It is also straightforward to apply the variational principle to the particular type of lattice-Boltzmann model used in this work, and this is carried out in Appendix A.

The other consequence of the memory requirements, already hinted at, is that this work would not be possible without access to massive computational resources. In Appendix B we present and discuss some new paradigms and technologies required to efficiently utilize these resources. More specifically, we initially studied the possibility of performing the larger simulations reported in this work over geographically distributed resources, in order to harness the required amount of memory, using the framework and techniques of grid computing [20]. With the advent of our access to two petascale resources, this possibility was temporarily abandoned. Nevertheless we believe this is definitely an avenue worth pursuing, and that the resource providers should be encouraged to unify their efforts and machines in a transparent, seamless way, since there will always be problems too big to fit onto a single machine. Therefore, part of this work is described in some detail in section B.3.

The present work has been carried out in close cooperation with computer scientists, in the framework of the EPSRC project "User-Friendly Authentication and Authorization for Grid Environments", ref. EP/D051754/1. With the emergence of grid computing, and the increasingly distributed nature of computational science, security and usability have become core aspects that cannot be overlooked and thus a discussion of some relevant issues in this area is included in section B.4.

In Chapter 4 we describe the code that we have written to simulate turbulent flow using the lattice-Boltzmann method. The software package was dubbed HYPO4D, which stands for "Hydrodynamic periodic orbits in 4 dimensions". It is written in the C programming language and uses MPI (Message Passing Interface) for parallelization. In that chapter the precise lattice-Boltzmann algorithm used is described, as well as the strategies adopted for optimizing the parallel performance of the code. We have used two typical systems, whose solutions are known, to benchmark the application, with very good results. Timing results on a variety of computational resources are presented and the effects of writing and reading data from disk (referred to as I/O from now on) discussed. Finally, in section 4.4 we describe the type of force used for the simulation of turbulent flow and the transition from laminar to time-dependent flow.

The application of the variational principle to 4D orbits is referred to as "numerical relaxation" and is described in full detail in Chapter 5. We investigated different algorithms, based on several combinations of gradient descent and conjugate gradient, well known numerical minimization tools [21]. Due to the large size of the 4D systems studied in this work, this was not always an easy task, with some algorithms proving to be more efficient than others. Some of the possible reasons for this are listed and discussed. In all minimization algorithms implemented here, extra arrays, with the same dimension as the full orbit, have to be kept in memory for the tracking of gradients as well as intermediary quantities needed to compute the value of the functional that is being minimized. This in turn increases even more the already large memory requirements and some of the strategies adopted to deal with this issue are discussed. In sections 5.3 and 5.4 the main results of this work are presented and discussed. This includes a summary of some of the larger simulations carried out and the UPOs that were found in these.

Several issues are left open by this work; the most pressing ones are discussed in Chapter 6. Some of these include the future usage of techniques from symbolic dynamics and the extension of this work to flows with a higher degree of turbulence. We then conclude with a survey of the main insights obtained through this work and their contribution to the field of turbulence studies.

CHAPTER 2

Fluid Turbulence

TURBULENCE is a paradigm problem in non-equilibrium statistical physics, with systems exhibiting large fluctuations as well as a macroscopic space-time structure. Although the Navier-Stokes equations [10, 22], which describe an incompressible viscous fluid, have been known for more than 150 years, the systematic description and physical understanding of fluid dynamics in a turbulent regime has evaded all attempts so far, being hailed as one of the great unsolved problems of classical physics [3, 4, 23–26].

In this chapter we shall address this issue by presenting the Navier-Stokes equation for incompressible fluid motion and commenting on analytical and numerical studies of the solutions of these equations. We then describe some of the main concepts in dynamical systems theory and the variational approach adopted in this work for the identification of unstable periodic orbits in forced dissipative systems.

The equations resulting from the application of this novel variational principle to the lattice-Boltzmann equation are deduced in Appendix A. This is due to organizational issues, since the lattice-Boltzmann equation itself is only discussed in Chapter 3. However, we note that these equations are also one of the main original results presented in this thesis.

2.1 Introduction

The difficulty of understanding turbulent behaviour stems from two central aspects of the problem. The first relates to the large number of degrees of freedom involved, strongly coupled by nonlinear interactions which transfer energy from the scale at which it is injected down to the damping scale where dissipation occurs. The second aspect refers to the non-equilibrium nature of turbulence. We recall that a system is said to be in thermodynamic equilibrium if it is in thermal, mechanical and chemical equilibrium. This means that there is no flow of energy in the system, no volume changes and no flow of particles due to chemical reactions [27]. However, in fully-developed turbulence systems are very far from equilibrium and the methodologies used in equilibrium statistical physics no longer apply, which in turn makes the statistics of the system much harder to predict and understand [23].

Although a great deal is known about the onset of turbulence in many different specific situations, as well as ways to suppress or enhance it, a fundamental theoretical knowledge of the problem is still lacking. The motion of incompressible, viscous Newton fluids for a driven dissipative system, is described by the Navier-Stokes equations (NSE). Most researchers would agree that these equations probably contain all of turbulence [3, 4]. For the 2D case a proof of existence and smoothness of the solutions for these equations has been known for a long time [28]. For the 3D case such proof has not been found yet, and this is widely considered to be one of the most important outstanding mathematical problems [22, 29]. Analytical solutions are known only for particular, simplified cases, such as Poiseuille and Couette flow [10, 30].

The Navier-Stokes equations possess a term which is nonlinear with respect to the velocity field. In this context, O. Reynolds, in his ground-breaking studies on the transition to turbulence in pipe flows [31], introduced a parameter that now bears his name and which gives a measure of the ratio between the nonlinear (convective) and dissipative properties of the flow. In modern dynamical systems language this can be seen as a control parameter for the system. The Reynolds number, Re, to which we shall refer often throughout this work, is defined as:

$$Re = \frac{UL}{\nu}, \tag{2.1}$$

with U and L being characteristic velocity and length scales and ν the kinematic viscosity[1] of the fluid. This parameter is specially relevant for the study of flows that exhibit maximum vorticity, $\boldsymbol{\omega}$, at their core, and not at the boundaries [32]. Vorticity is defined as the curl of the velocity field, $\mathbf{u}$:

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \times \mathbf{u}. \tag{2.2}$$

In other words, when far from the boundaries, turbulent flows (for which the vorticity field is a more relevant representation than the velocity field) behave similarly for a given value of Re, independently of the shape of the system. Moreover, the Reynolds similarity principle states that flows with the same geometry and the same value of Re will behave essentially in the same fashion. This is of major practical significance, allowing engineers to perform experiments in relatively small apparatus, like water channels and wind tunnels and extrapolate the results thus obtained to larger systems, by scaling the dimensions accordingly, and keeping a fixed value of Re [3, 32].

The typical scale of Re for fully developed turbulence in many physical systems of practical relevance, such as the boundary layer of an aircraft fuselage [33] or the motion of air a few metres above the ground, is in the order of 10^6 and higher. The Kolmogorov [34] picture for turbulence states that nonlinear interactions couple very many length scales. The energy injected in the system at the large scales is transported through progressively smaller length scales until finally the dissipation of energy

[1]Unless otherwise stated, whenever the word "viscosity" appears in the text, with no further qualification, the *kinematic* viscosity should be understood.

into heat takes over[2]. According to this picture, the smallest scale, η, still active in the flow is given by:

$$\eta \simeq \frac{L}{Re^{3/4}}. \tag{2.3}$$

As pointed out by Succi and Papetti [6], if $Re \simeq 10^8$, a typical value in atmospheric turbulence, then using Eq. (2.3) we find that the energy injected at the length scale of $1\ km$ can still be detected at the scale of $1\ mm$.

In Kolmogorov's turbulence scenario, which has been widely influential in the turbulence research of the last half-century [3], the number of degrees of freedom of the system will scale with $Re^{9/4}$. Thus, for fully developed turbulence, direct numerical simulation, in which all degrees of freedom are taken into account [35], becomes an impossible task with the current computing resources and the ones we can expect to be available for the next decade [36]. In this context and in order to reduce, for computational purposes, the huge number of degrees of freedom involved, there is great interest in the study of coherent structures within turbulent flows. These usually take the form of spiral structures, better known as vortices, possessing very high rotation. These vortices can be interpreted as meta-stable states facilitating the transfer of energy across multiple length scales until heat dissipation finally takes over [6, 37].

A further note is required at this point concerning the range of applicability of the NSE. So far we have been using the term "fluids" somewhat freely. In physics this term refers to gases, liquids and plasmas. In the present work we focus only on incompressible fluids, thus ruling out some interesting phenomena such as sound waves and supersonic flows [10]. We also do not consider reactive flows, which form another major area of research, of great relevance to industrial and environmental situations. However, following McComb [24], we are still left with a wide range of application, including the vast majority of environmental flows and even most gases, provided that the velocities in the latter are no larger than one third of the speed of sound.

The plan of this Chapter is as follows: in the next section we shall write down the NSE and briefly comment on some analytical methods that have been used to study the possible existing solutions of the equations. We then proceed to discuss some of the main methodologies used to solve them numerically, as well as commenting on the inherently statistical nature of turbulence. In section 2.3 we introduce some concepts of the theory of dynamical systems, with a special emphasis on unstable periodic orbits (UPOs) and dynamical zeta functions (DZF). Then in section 2.4 we shall discuss the spacetime variational approach used in this work to detect UPOs.

[2]In between these two limiting cases we have the *inertial* range, of which more will be said in sections 2.2.2 and 4.4.4 of this thesis.

2.2 Navier-Stokes equations

2.2.1 Some definitions

The NSE can be written as follows, assuming a Newtonian fluid, where the shear stress and strain tensors vary linearly, with the proportionality constant being the viscosity[3] [6, 24], driven by a force density field $\mathbf{F}(\mathbf{r}, t)$:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = -\frac{1}{\rho}\boldsymbol{\nabla} P + \nu \nabla^2 \mathbf{u} + \mathbf{F}(\mathbf{r}, t), \tag{2.4}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0, \tag{2.5}$$

where P is the pressure and ν is the viscosity of the fluid. Eq. (2.5) is a constraint that must be obeyed at all times, imposing a further restriction on the velocity field $\mathbf{u}$. We are here assuming an incompressible fluid of constant density, ρ. To fully define fluid flow, Eqs. (2.4)-(2.5) must be augmented with appropriate boundary conditions and initial values for the velocity field at time $t = 0$, along with suitable values for ρ, P and ν. In the derivation of the equations the conservation of mass, momentum and energy is implied.

For the case of inviscid fluid we have $\nu = 0$ in Eq. (2.4), which will then correspond to the Euler equations. These inherit most of the difficulties (mentioned in the previous section) that plague the NSE; namely, the convective term (second term on the left hand side of Eq. (2.4)) is still present and there is no proof of existence and smoothness of solutions for the 3D case either [22, 29].

If we assume that the force is divergenceless, and take the divergence of Eq. (2.4) we find that the condition of a divergenceless velocity field, Eq.(2.5), can be imposed by :

$$\nabla^2 P = -\boldsymbol{\nabla} \cdot (\mathbf{u} \cdot \boldsymbol{\nabla} \mathbf{u}). \tag{2.6}$$

This is a Poisson type equation that needs to be solved at each time step, so as to ensure condition (2.5), something which is very time consuming, introducing extra numerical difficulties and complexity in conventional computational fluid dynamics (CFD), the topic of the next section. It is one of the main advantages of the lattice-Boltzmann method (to be discussed in Chapter 3) that it does not require this extra condition.

As mentioned in the previous section, the Reynolds number is a measure of the ratio between the nonlinear and dissipative effects acting on the flow. These are described respectively by the second term on the left hand side, which represents convection, and the second term on the right hand side of Eq. (2.4). Decreasing (increasing) the velocity field or increasing (decreasing) the viscosity will result in a lower (higher) ratio between these two contributions. If this ratio goes over a certain threshold, which

[3] In the case of ”non-Newtonian” fluids the viscosity will be a function of the stress tensor (i.e., it varies depending on the applied strain rate) and memory effects assume an important role. Typical examples are ketchup, blood and paint, as well as most polymers.

will vary from one system to another, according to geometry and other factors, then the (numerical) solutions of the NSE become increasingly sensitive to initial conditions and small differences in these will be magnified exponentially. It is in this sense that fluid turbulence can be claimed to be tightly connected to modern chaos theory [7, 9, 38], and we shall explore these links a little further in section 2.3.

Before we turn to an overview of the main methods used to simulate the NSE numerically, a mention should be made of the current state of affairs regarding the problem of finding exact solutions to the NSE. In this respect, several lines of attack have been pursued over the past one hundred years, with varied degrees of success [3, 22]. Needless to say, any success in this enterprise could have significant consequences to our understanding of high Reynolds number flow.

The main goal in these analytical studies is usually to reduce as much as possible the number of degrees of freedom, by finding alternative, more tractable expressions for the equations. One such breakthrough happened with the work of Leray in the 1930s, of which an excellent account can be found in the book by Majda and Bertozzi [39]. His proposal was, in very general terms, to look for *weak solutions* of the NSE, i.e., solutions in which the velocity field may not be differentiable everywhere, and then, by providing these solutions with suitable growth properties, to investigate the existence and regularity of solutions to the NSE (see also [40] for a recent good overview of the subject). Following this line of thought, Leray showed (see [39] and references therein) that the NSE always have a weak solution with suitable growth properties. The uniqueness of these weak solutions of the Navier-Stokes equation has not however been proved, although it is known that for the Euler equations such uniqueness does not hold [29].

Another approach is to begin with an initial velocity field which is sufficiently regular so that unique smooth solutions will exist for a finite time, T. If, furthermore, the initial flow field has a small magnitude then these unique smooth solutions may persist for an unbounded amount of time. These are called *strong* solutions [40]. An interesting line of research consists in studying the sufficiency bounds which allow the solutions to persist for the longest time, T. This approach is, however, limited to low Reynolds numbers, once again, due to the requirement of the flow fields having small magnitude.

Summing up these arguments, we can say that the current state of affairs stands as follows. In three dimensions, the existence and smoothness of solutions to the NSE has been proved, provided that the initial velocity field satisfies a smallness condition [29]. Alternatively, the existence and smoothness is also known to hold without the smallness condition, but only for a small time interval $[0, T)$, where the size of the blow-up time, T, will depend on the initial values assumed. Near this blow-up time the values for the velocity field become unbounded, which means they can assume values of any conceivable magnitude, a clearly unphysical situation [29]. In a similar framework, it is the vorticity field instead which becomes unbounded, for the case of the Euler equations.

In spite of all these fundamental questions, still left wide open, we can fortunately still iterate numerically the NSE for very many situations of interest. It is to an overview of such numerical methods and the main concepts behind them that we now turn our attention to.

2.2.2 Computational Fluid Dynamics

The field of Computational Fluid Dynamics (CFD) is a very extensive one, with very many subdivisions and different methodologies being used, depending on things such as the geometry and the Reynolds number of the flows in consideration. One of the most concise introductions to the subject can be found in the book by Succi and Papetti [6], whose organization of numerical schemes we follow in this section. Due to the intrinsic nonlinearity of the NSE, these must be time forwarded on a suitably defined computational grid which discretizes the continuum spacetime assumed in those equations. This discretization can be performed in a variety of ways, each one usually giving rise to new numerical schemes, with their respective assets and drawbacks.

Following Succi and Papetti, a first distinction can be made between "grid" and "particle" methods. In the former, the spacetime continuum is discretized directly, whereas in the latter, the fluid dynamics is obtained by following bundles of particle trajectories and then summing over an ensemble of such realizations. Examples of particle methods include the lattice-Boltzmann method, which we apply in this work; molecular dynamics, which has been useful in deducing macroscopic properties such as viscosity and conductivity from the microscopic level; Monte-Carlo methods, in which a probabilistic approach is invoked; and vortex methods in which the "particles" are called "vortons" and represent the *quanta* of vorticity. It should be stressed that the term "particle" in this context is quite broad and frequently no more than a convenient representation for a given numerical approach. We shall encounter this again in Chapter 3, when discussing in detail the lattice-Boltzmann method. As pointed out by Succi and Papetti, particle methods are potentially more attractive for parallel computing since they possess a built-in kind of concurrency, in the form of particle trajectories evolving simultaneously, which grid methods do not.

Local methods

Regarding grid methods, we begin by discussing *local* ones, in which the quantities being tracked describe properties that are local to a given grid point, x_l, at time t_n. In order to simplify the notation, we do not explicitly write x_l as a vector. The three main approaches in this context are the Finite-Difference (FD), the Finite-Volume (FV) and the Finite-Element (FE) method. In the first of these methods, a given function, $f(x_l, t_n)$, is described as a set of discrete values, $f_{l,n}$, where:

$$f_{l,n} \equiv f(x_l, t_n). \tag{2.7}$$

This choice of representation does not say anything about the values of $f(x_l)$ in between two consecutive values, x_l and x_{l+1}, for a fixed time. In particular, no assumption is made as to the spatial derivatives (of both first and second order) which appear in the NSE, Eqs. (2.4)-(2.5). To estimate the value of these derivatives one must chose a proper numerical method, taking into account requirements of accuracy and stability. The simplest such choice is probably the Euler-forward representation:

$$\left.\frac{\partial f}{\partial x}\right|_{x=x_l} \equiv \frac{f_{l+1}-f_l}{\Delta x}, \tag{2.8}$$

where $\Delta x = x_{l+1} - x_l$. The derivative, as well as the values of f, are assumed to be taken at the same value of t.

The previous discretization, Eq. (2.8) will only be first-order accurate, with a discretization error of the order of Δx. In order to achieve higher-accuracy, for a given value of Δx, we must sacrifice locality, i.e., increase the number of neighbouring points being taken into account. An example of this is the central difference scheme, in which:

$$\left.\frac{\partial f}{\partial x}\right|_{x=x_l} \equiv \frac{f_{l+1}-f_{l-1}}{2\Delta x}. \tag{2.9}$$

From Eq. (2.9) we see that the derivative of f at point x_l now involves the values of f at x_{l+1} and x_{l-1}. This can be seen as a trade-off between accuracy and locality, which is to say, computational efficiency. As regards parallel implementations, the widening of the neighbourhood of the (discrete) differentiation will also have (negative) consequences in terms of scalability of the algorithms as the number of processors is increased.

The main limitation of FD schemes lies however in its inability to deal with complex geometries, namely when the boundaries of the system cannot be fitted naturally to a mesh coordinate. Although this can be overcome by adopting immersed boundary methods (see [41] for a recent review of these, and references therein), the resulting algorithms become increasingly complex and non-intuitive. The straightforward way to overcome this drawback is to move from a coordinate-based approach to an element-based one (such as FE) or a cell-based one (such as FV).

The Finite-Volume method overcomes the lack of geometrical flexibility of FD by decomposing the spatial domain into a series of non-overlapping elementary volumes (or areas, in the case of 2D flows). The particular representation chosen for these volumes must be topologically equivalent to a cube, in the sense that the spatial domain must be unambiguously covered, with no overlaps. This means that there will be a transformation between the chosen discretization and a cubic structured mesh.

One of the most important assets of this method is that it is conservative by construction, i.e., the flux leaving a given cell in one direction must be equal to the flux entering the closest neighboring cell in that direction. Another strong asset is the geometrical flexibility it exhibits due to the capability of local mesh deformations. These allow for a straightforward implementation of complex and irregular geometries, making FV one of the preferred methods not only in research software packages but also in commercial applications. One of the main difficulties associated with this approach is the need to generate an optimal grid at the beginning of the computation. Although this must be done only once for each problem (unless we consider a dynamical environment, where boundaries change with time), it can often be the most time-consuming stage of the analysis. For this reason, there is much interest in automated grid-generation methods [42].

The last local method we should mention in this context is the Finite-Element method. In this approach, the constraint of having a structured mesh is relaxed, and the fundamental geometrical entities are now defined as a collection of nodes linked by a given number of links. The number of such links for a given element will define the connectivity of that particular element. By adopting this approach the number of grid points can be clustered in the regions where flow activity is high and rarefied in regions where that is not the case. We can immediately see that this approach will be extremely useful in the structural analysis of solids, namely in the study of deformations and rheology.

An important downside of the method does exist, however, similar to the one discussed for the FV: this is the need to have a matrix that specifies the specific arrangement of the mesh. This means that each time any given calculation is performed, that matrix must be consulted in order to know the size and shape of a given element, as well its connectivity, i.e., the number of neighbouring elements and their geometrical arrangement. This places crucial importance in optimal mesh numbering strategies. Nevertheless, Finite-Element methods [43] have been used to tackle complex geometries and boundary conditions, free surfaces, turbulence effects and are widely used in industrial applications, being one of the main CFD tools.

Spectral methods

All three methods mentioned above are local ones, in the sense that the variables used represent the values of an unknown function (such as the velocity field, or the pressure) in a localized neighbourhood of a single spatial location. This focus on locality is particularly useful if the geometry of the system is a major issue. If that is not the case however, the locality condition can be somewhat relaxed with major gains in terms of computational efficiency. We then have non-local methods, in which the quantities being tracked are now global properties of the unknown function (e.g., the velocity field). Examples of such global properties are the total mass of the system, the total energy and the energy contained in a given scale of motion.

Among the non-local methods, the most important one is the Spectral Method (SM) [44]. In this method, the velocity field, $\mathbf{u}(\mathbf{r}, t)$ is decomposed in a discrete Fourier representation:

$$\mathbf{u}(\mathbf{r}, t) = \sum_{n=0}^{N} \tilde{\mathbf{u}}_n(t) e^{-i\mathbf{k}_n \cdot \mathbf{r}}, \tag{2.10}$$

where $\mathbf{k}_n$ are wave numbers, i obeys $i^2 = -1$ and $\tilde{\mathbf{u}}_n(t)$ are (discrete) Fourier coefficients (see [21] for an introduction to this subject, aimed at numerical implementation). The method is particularly well-suited for the study of homogeneous flow with periodic boundary conditions and has been the mainstay of homogeneous turbulence research for many years now, but has also been applied with great success in other areas such as global weather modelling and in investigating the transition regime in shear flows [44].

Working in Fourier space presents us with two immediate advantages. The first one is that space derivatives are now represented by algebraic diagonal operators. Using Eq. (2.10), the derivative of the j component of the velocity field in the direction m (where $j, m = 1, 2, 3$) is given by:

$$\frac{\partial u_j(\mathbf{r}, t)}{\partial r_m} = \sum_{n=0}^{N} \tilde{u}_{j_n(t)}(-ik_{m_n})e^{-i\mathbf{k}_n \cdot \mathbf{r}}, \tag{2.11}$$

and we see that derivation of the velocity field components is now reduced to an algebraic operation, with multiplying factors $-ik_{m_n}$. This will have very important efficiency gains for Eqs. (2.4), (2.5) and (2.6). The other advantage of this approach is that the process of derivation, as outlined in Eq. (2.11), can be said to be exponentially accurate, as N tends to infinity, which ties in with our discussion of finite-differences schemes, namely Eqs. (2.8) and (2.9).

However there is also a significant downside of spectral methods, since the non-linear convective term in the NSE, the second term in Eq. (2.4), becomes highly non-local in $\mathbf{k}$ space. In practice this means that any mode will interact with all others, in a fully global fashion [24]. This then gives rise to a computational complexity of order $\mathcal{O}(N^2)$, which is highly undesirable.

The way to surmount this difficulty is to replace the Fourier transform, F, of the convective term in the following way:

$$F\left[\mathbf{u} \cdot \nabla \mathbf{u}\right] \simeq F\{F^{-1}[\tilde{\mathbf{u}} \cdot F^{-1}\{i\mathbf{k}\tilde{\mathbf{u}}\}]\}, \tag{2.12}$$

where F^{-1} represents the inverse Fourier transform and $\tilde{\mathbf{u}} \equiv F[\mathbf{u}]$. Note that Eq. (2.12) is not an exact equality, and *aliasing* effects are involved, due to the fact that there is a critical frequency (referred in the literature as the *Nyquist* frequency) limiting the accuracy of the approach [21]. However, the formulation described by Eq. (2.12) reduces the computational complexity down from $\mathcal{O}(N^2)$, although at a cost of a few more Fourier transforms. This is called the pseudo-spectral method and has been the main tool in turbulence research for the last decades. It must be noted that this approach has also been extended to describe relatively simple bounded flow geometries (see [44] and references therein) as well as being combined with the Finite-Element method, discussed above, in order to utilize the best features in both methods [45].

As we have seen, both the Spectral Method and its pseudo-spectral extension rely heavily on the computation of very many Fourier transforms. It is easily seen that for a typical problem, assuming geometry is not a major issue, the time spent calculating these transforms will be the single determining factor in terms of efficiency. In this respect, it was the appearance of the Fast Fourier Transform (FFT) algorithm [21] which made this approach a winning one, not only in CFD but in very many areas where partial differential equations must be iterated extensively. The FFT algorithm reduces the numerical complexity of the spectral approach from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log_2 N)$, with N being the number of points in the lattice, for all values of N, including prime ones, which is an invaluable achievement. Nevertheless, there are still issues regarding the scalability of these approaches as we increase the number of processes for a given computation. This becomes an even more pressing point as larger and larger machines, currently in the petascale, become available to the scientific community at large.

In this context, we should mention another recent commercial development which also shows great promise, and has been generically termed “cloud computing” [46]. Several projects exist already, addressing (arguably) fundamental scientific issues, in which the idle CPU time of personal computers is effectively exploited, through the medium of distributed computing. The first of these projects, which started in 1999, is the famous “SETI@home” [47], developed at Berkeley University, in which users can donate their spare CPU time to analyze data from the Arecibo radio telescope in search of unusual signal patterns that might indicate extraterrestrial intelligent activity. More recent examples include the simulation of protein folding [48], at Stanford University, the simulation of particle trajectories on the Large Hadron Collider [49] and simulation of climatological evolution [50], at Oxford University.

Celani [36] notes that a “Turbulence@home” could also become a reality in the near future. However, parallel FFT algorithms require a very large amount of synchronized communications and would probably be highly ineffective in this context. A good alternative would be the lattice-Boltzmann method, with its inherently local nature and almost embarrassingly-parallel communication pattern, which we also use in this work and will discuss in Chapter 3.

CFD studies of Turbulence

One further important distinction that must be made in the CFD field is that between direct numerical simulation (DNS) and “non-DNS” methods. By DNS is meant that all (spatial) scales of fluid motion, down to the smallest active length scale, the Kolmogorov length, η, are taken into account. By contrast, in non-DNS methods only the larger scales of the flow are explicitly accounted for, while a given model, based on sound physical assumptions, is assumed to describe the finer scales which have been averaged out. This approach also stems from the Kolmogorov description [3] and basically relies on the assumption that the smaller eddies in turbulent flow will have a universal character, and thus may not require to be simulated in great detail.

Using Eq. (2.3) we see that the number of degrees of freedom in turbulent flow will scale as a power of $Re^{9/4}$, for a 3D simulation. The overall numerical complexity will scale as Re^3, since we require a time of $\sim Re^{3/4}$ magnitude to observe the complete sweeping of the finest scales across one dimension [3]. This becomes therefore a daunting enterprise, and for many decades effort has focused on finding models which could somehow average the finer length scales while still maintaining sufficient numerical accuracy. One such approach, whose roots can be found in the work of Smagorinsky in the 1960s [51], consists in filtering the smaller scales of the motion and finding effective equations for the motion of the larger eddies. This approach is commonly referred to as large eddy simulation (LES), for obvious reasons. The velocity field $\tilde{\mathbf{u}}(\mathbf{r}, t)$ describing the motion of eddies of size larger than Δ can be obtained by the convolution of the velocity field $\mathbf{u}(\mathbf{r}, t)$ with an appropriate kernel (acting as a filter), $G_\triangle(\mathbf{r}, \mathbf{r}')$, which eliminates the fluctuations below scale Δ:

$$\tilde{\mathbf{u}}(\mathbf{r}, t) = \int G_\triangle(\mathbf{r}, \mathbf{r}')\mathbf{u}(\mathbf{r}', t)d\mathbf{r}'. \tag{2.13}$$

The main task now is to find reasonable ways to effectively model the subgrid-scale (SGS), i.e., the effect of the eddies of size smaller than Δ, which still interact with the larger sized eddies. A good introduction to the LES models can be found in the work of Scotti and Meneveau [52]. Using this type of approach, a large field of literature exists with researchers routinely reporting results for flows with $R_\lambda \sim \mathcal{O}(10^4)$ [53]. The quantity R_λ stands for the Taylor-scale Reynolds number [54], defined as:

$$R_\lambda = \frac{u_{rms}\lambda}{\nu}, \tag{2.14}$$

where u_{rms} is the root-mean-squared velocity and the Taylor scale, λ, is defined as:

$$\lambda = \frac{u_{rms}}{\left\langle \left(\frac{\partial u}{\partial x}\right)^2 \right\rangle^{1/2}}, \tag{2.15}$$

and the derivative in the previous expression is assumed to be performed over the first component, u, of the velocity field. The Taylor-scale based Reynolds number is related [3] to the integral-scale one by:

$$R_\lambda \sim Re^{1/2}, \tag{2.16}$$

with the proportionality constant being $\sqrt{15}$ for the case of homogeneous and isotropic flow [54].

In some fields of application of fluid mechanics, such as atmospheric and geophysical flows, the relevant values of the Reynolds number can be very high indeed and the the length scales involved of a very high order of magnitude. For these cases, models coarser than LES exist to further reduce the numerical complexity, still believed to be too high in the LES. The most well-known of these methods utilizes the Reynolds-averaged Navier-Stokes (RANS) equations, which rely on the separation between the time-averaged part of the velocity and pressure fields and its fluctuations. Some methods will then use a mixture of RANS for solving the core portions of the fluid and LES for dealing with the fluid at the walls, where greater numerical accuracy is required [53].

The past decades have nevertheless seen a massive increase in the sheer capacity of the supercomputers becoming available to the scientific community, which has been continually pushing the limit of what is possible in this area. Some of the milestones in DNS of homogeneous isotropic turbulence can be seen on Table 2.1, adapted from Celani [36], where a cubic lattice with L^3 points is considered, and the respective values for R_λ are also shown.

All of the results mentioned on Table 2.1 used, without exception, some form of spectral method, which has been the main tool in CFD studies of homogeneous isotropic turbulence studies for several decades now. Looking at the more recent entries, we note that the work of the Kaneda *et al.* group [63, 64] was performed on the "Earth Simulator" machine in Japan, which was for some years the

Year	L	R_λ	Reference
1972	32	35	Orszag & Patterson [55]
1981	128	84	Rogallo [56–58]
1991	256	150	Vincent & Meneguzzi [59]; Sanada [60]
1993	512	200	She *et al.* [61]
2001	1024	460	Gotoh & Fukuyama [62]
2003	2048	730	Kaneda *et al.* [63]
2006	4096	1200	Kaneda & Ishihara [64]

Table 2.1: Progress in computing homogeneous isotropic turbulence, adapted from [36]. Cubic lattices with a number of sites given by L^3 are assumed. R_λ is the Taylor-scale based Reynolds number, defined in Eqs. (2.14)-(2.15). The first authors of each article are referenced in the fourth column, and the complete reference to each original paper can be found in the bibliography. The values are sometimes approximated. As an example, the grid size in [59] was actually 240^3, not 256^3.

fastest supercomputer in the world. Actually, in their 2003 work [63] this group already reported using a 4096^3 lattice (with an estimated value of $R_\lambda = 1201$), although with very little statistics, i.e., that particular simulation ran for much less time than for the other reported lattice sizes and values of R_λ.

Another research group which has been quite active in recent years is the one headed by D. A. Donzis and P. K. Yeung. These authors reported a value of $R_\lambda \sim 700$ using DNS applied to the mixing of passive scalars in Lagrangian[4] turbulence [65]. This is very close to the maximum experimental value reported for this case, of $R_\lambda \sim 800$. We note in passing that Lagrangian turbulence is a very active area of research currently (see [54] for a recent review of the field and references therein). More recent work explores the role of resolution effects in passive scalar mixing [66], and in dissipation and enstrophy[5] in isotropic turbulence [67], both based on DNS data obtained with grids of size up to 2048^3.

As already mentioned, we left out of this discussion several important topics. One of these is the time-dependence of the numerical schemes [6], which can be either implicit or explicit[6], since we only addressed the spatial discretization of the NSE. Another crucial approach in this field has been two-equation turbulence models, namely the $k - \epsilon$ and $k - \omega$ models [68], as they are usually referred to, where k stands for turbulent kinetic energy, ω is the turbulent dissipation and ϵ the specific dissipation. These are still within the overall framework of RANS models, but they now introduce two further

[4]Related to the Lagrangian view of fluid flows, as opposed to the Eulerian one. We shall define these terms in section 4.4.1, in the context of ABC flows.

[5]Enstrophy is a scalar quantity, defined as

$$\Omega = \frac{1}{2}\int \omega^2 d^3 r, \tag{2.17}$$

where ω is the vorticity, defined in Eq. (2.2).

[6]This issue will be briefly discussed in section 3.4.3, in the context of discussing lattice-Boltzmann methods within the CFD framework.

transport equations which can account for history effects such as convection and diffusion of turbulent energy, an approach which has found great acceptance in industrial applications [69, 70].

Finally, another method which has made some headway on the problem is the renormalization group (RG) method, which had its origin in the field of modern particle physics and quantum field theory but was afterwards generalized to several branches of statistical physics, with applications to critical phenomena and phase transitions [71] . The basic idea is the concept of a hierarchy of scales of increasing length, defined in such a way that the dynamics at a given scale will have an impact, conveniently parametrized, on the dynamics of the next scale. Due to the almost paradigmatic multiscale nature of turbulence, attempts were very quickly made of applying the RG framework to it, and for some time it seemed these showed good promise of throwing some new light into the nature of fully developed turbulence [24], but as with many trends in turbulence research the enthusiasm seems to have faded over the years.

We must finish this section by remarking that this is by no means an overview of the entire field of CFD turbulence research, which would be an impossible task in the present context. Good introductions can be found in the book by Succi and Papetti [6], on the subject of computational techniques and in the book by Uriel Frisch [3], with its focus on Kolmogorov phenomenology, the onset of turbulence and intermittency. Also of great use is the guide to the literature included in the last chapter of that book. Finally, a very accessible introduction to the subject can also be found in the review article by McComb [24]. Although heavily oriented towards the renormalization group approach, it includes a quite readable exposition of the main physical issues involved and the statistical character of turbulent flows.

2.2.3 Exact coherent structures

One line of research that has been fruitful in recent years and bears some relation to the present work is the study of “exact coherent structures”. The term was first coined by Waleffe [72] and describes exact invariant solutions of the Navier-Stokes equations, which have also been observed in direct numerical simulation and experiment. They usually take the form of travelling waves and can be seen as the physical images of the least unstable invariant solutions of the flow [73].

One of the first examples of such solutions was found by Nagata [74], in plane Couette flow. The method used consisted of continuation and bifurcation from a vortex solution of Taylor-Couette flow. In this way, two solutions were found, one with very high energy dissipation rate and a smoother one, with lower dissipation rate. These are usually referred to in the literature as “upper-branch” and “lower-branch”, respectively. He then found travelling waves for plane Couette flow [75], using a continuation method.

Following this initial work, other exact solutions have been identified in a variety of configurations. Itano and Toh [76] found travelling waves in pressure-driven channel flow, using a shooting method.

Waleffe [72, 77, 78] found exact 3D equilibria and travelling waves in both plane Couette and Poiseuille flows at several Reynolds numbers. This was then extended to pipe flow by Faisst and Eckhardt [79] and Wedin and Kerswell [80] using continuation methods.

The central idea in this line of work is the concept of "self-sustaining processes", also introduced by Waleffe [81, 82]. Very briefly, this concept describes how streamwise streaks and vortices (sometimes referred to as "rolls" in this context) interact near the boundary turbulent layer, in a quasi-cyclic manner. The insights obtained by the description of this process are now being actively explored by other authors [73] in the visualization of the space state in wall-bounded shear flows.

To conclude this brief discussion, we should mention that these travelling wave structures are also being experimentally observed in pipe flows, by the technique of high-speed stereoscopic image velocimetry [83], and their role in the transition to turbulence is being increasingly recognized.

2.2.4 Statistical description of turbulence

Returning to our main argument, after a value of Re greater than some threshold, which will depend on the particular geometry of the problem, the fluid flow becomes time-dependent and a probabilistic description must be introduced [3, 24]. This means that in order to compute relevant quantities, such as the velocity field, we must perform averages over an interval of time considered to be "long enough". The relevant time scale t_l, for a certain length scale l is given by the ratio

$$t_l \sim \frac{l}{v_l}, \tag{2.18}$$

with v_l naturally being the typical value of the velocity associated with scale l. This quantity provides us with an estimation of the typical amount of time for a structure of size $\sim l$ to undergo a significant distortion due to the motion of its components relative to one another (i.e., not their absolute, collective motion). In turbulence studies, the "large-eddy turnover time" is usually referred to, meaning the estimated turnover time for the larger coherent structures found in the fluid, as defined by Eq. (2.18). The value for observable dynamical quantities is usually estimated by taking averages from the data obtained for just a few of these time lengths [63, 65]. The quantities found this way will always be stochastic in nature, and it may be very difficult to assess whether the particular time interval chosen for computing a given average was long enough, or representative enough.

An alternative approach, which will be pursued in this thesis, based on dynamical systems theory, is to locate and characterize some of the lower-period UPOs of the equations and use these to calculate averages, via the formalism of the dynamical zeta function. These averages will then be given by an exact expansion, in analogy with a Taylor expansion in analysis, whose accuracy will depend on the number of lower-period UPOs included. Fortunately, these expansions have very good convergence properties [9] and we thus find ourselves with a tool to compute quantities in turbulent flow from first principles, as will be explained in more detail in the next section.

2.3 Unstable Periodic Orbits

2.3.1 Dynamical systems approach

In this work we adopt the view that the strange attractor [7, 84, 85] for driven dissipative systems (of which one well known example is the Lorenz attractor [14]) can be thought of as the closure of the set containing all the unstable periodic orbits (UPOs) of a given system. These UPOs provide a *countable* sequence of orbits and can therefore provide a powerful characterization of the structure and dynamics of the attractor [8].

In dynamical systems whose phase space consists of 3 or more dimensions there are infinitely many UPOs close to a strange attractor, in the neighbourhood of which the system will spend most of its time. In the case of a driven, viscous, incompressible fluid in the turbulent regime, the state space is the infinite-dimensional function space of all divergenceless vector fields. For the 2D case it has been rigorously proven that the attracting set is finite-dimensional, albeit with a dimension that scales as a power law of the Reynolds number [22, 86]. For the 3D case there exist strong indications, both theoretical [7, 86, 87] and numerical [88], that this should also be the case.

The notion of a “strange attractor” was first proposed by Ruelle and Takens, in their seminal 1971 paper, “On the nature of Turbulence” [84]. In this paper[7] the authors proposed a new scenario for the onset of turbulence in fluid motion. The main stimulus for this proposal came from their dissatisfaction with a previous scenario, suggested by Landau [10], in which turbulent behaviour is assumed to be decomposable in a series of quasi-periodic motions. According to this scenario, the physical parameters, x, describing turbulent behavior are given by

$$x(t) = f(\omega_1 t, \cdots, \omega_k t), \tag{2.19}$$

where the frequencies $\omega_1, \cdots, \omega_k$ are not rationally related, i.e., cannot be decomposed into a more fundamental description. As the control parameter for the system is increased (such as the Reynolds number, or another relevant adimensional quantity), more and more of these frequencies become active, until the superposition of very many of them will signal the onset of turbulent behavior. This had also been independently suggested by Hopf [90], and as a result these are usually called Hopf bifurcations [91].

As opposed to this picture, Ruelle and Takens argued that these quasi-periodic motions were not actually observed in turbulent flow and drawing inspiration from (then) recent advances in the qualitative theory of differential equations suggested the concept of the strange attractor underlying turbulent behavior.

[7]For a mainly non-technical (aimed at a more general audience but still highly challenging) explanation of these ideas and the motivation behind them see also the book written by Ruelle, “Chance and Chaos”, [89].

Following Eckmann [91] we can define an attractor as follows:

Definition An attractor for the flow T^t is a compact set X satisfying:

$(1)X$ is invariant under $T^t : T^t X = X$.

$(2)X$ has a shrinking neighborhood, i.e., there is an open neighborhood U of $X, U \supset X,$ such that $T^t U \subset U$ for $t > 0$ and $X = \cap_{t>0} T^t U$.

(3) The flow T^t on X is recurrent and indecomposable.

A few qualifying remarks are in order. Expression (2) actually excludes *repellors*, i.e., isolated points from which the system steers away in its time evolution. These do not play a role in experimental setups, where we are usually interested in long-term behavior. The fact that the flow T^t is recurrent means that all transients have been ignored. Again, from an experimental point of view, we are interested in the long-term behavior of a dynamical system after all transients have died out. The fact that X is indecomposable means it cannot be split into two closed invariant pieces.

A few misconceptions that sometimes arise in this context are the following.

$i)$ The fact that the flow T^t contracts volumes does not necessarily mean it will contract all lengths. There may be directions in which contraction occurs and others where stretching takes place, in phase space. In a strange attractor, in particular, there is mixing of points at all stages, which is a signal of sensitive dependence on initial conditions, one of the main ingredients of chaotic behavior [91]. It should also be noted that many strange attractors (as opposed to more trivial ones) will have a non-integer or *fractal* dimension [92], although this is not a necessary condition [9].

$ii)$ Simple dynamical systems can still have an infinity of distinct attractors. One particular such case is the Hénon map:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 1 + y_n - a x_n^2 \\ b x_n \end{pmatrix}, \tag{2.20}$$

where a and b are constants [93]. For the values of a near 1.15357 and $b = 0.3$ there occurs an infinity of attractors, corresponding to periodic points with higher and higher period [91].

$iii)$ The basins of attraction, i.e., the set of initial conditions from which the system will evolve towards the attractor, can be very complicated, and there is a large body of literature discussing these topologies [94]. The most dramatic example is probably that of *riddled* basins, i.e., chaotic attractors with basins in which at any point there are pieces of another attractor basin arbitrarily nearby, and so the basin is said to be "riddled" with holes [95].

Applying Fourier analysis on the motion of a strange attractor (e.g., on one of its coordinate components) will in general result in a continuous power spectrum. This can either be interpreted as a system exploring an infinite number of dimensions in phase space or a system evolving in a nonlinear fashion on a finite-dimensional attractor [7]. As it happened it was found in the last decades that the latter alternative frequently occurs, even in systems with very few degrees of freedom. Celebrated examples of systems exhibiting strange attractors include the Lorenz model for the convection of air in the atmosphere [14], the Hénon map [93], defined in Eq. (2.20), the Rössler equations for modeling chemical reactions [96] and the Ueda attractor found in electrical circuits described by the Duffing equation [97].

Before we move on to the description of the mathematical formalism that allows one to compute the values of observable quantities in a physical system from the knowledge of some of its UPOs, a further qualification is required. Although great progress has been made in recent years on the identification of UPOs in fluid dynamics, it is still not clear how this approach can translate to fully-developed turbulence, as noted by several authors (e.g., [7, 9]). In this thesis, as has been the case in other recent efforts in this field [17, 18, 73, 98–104], we only consider flows in a weakly-turbulent regime. The problem of fully-developed turbulence, where the values of the Reyolds number are at least of order $\mathcal{O}(10^5)$, may still elude researchers for many years to come.

2.3.2 Dynamical Zeta Function

As we have discussed throughout this chapter, although the NSE have been known for a long time, and several methodologies have been developed to simulate them numerically, very few exact results about turbulence have actually followed from them [4]. In particular, we still lack the ability to make *a priori* predictions of the turbulent averages of observables. The dynamics of the Navier-Stokes equations define a trajectory in the state space formed by the infinite-dimensional function space of all divergence-less vector fields. In the long-time limit, this trajectory is expected to settle onto a finite-dimensional attracting set [7], and the turbulent average of an observable may be thought of as its integral over a *natural measure* on this attracting set.

In low-dimensional examples, such as the Lorenz attractor [14], the attracting sets are replete with UPOs. These comprise a set of measure zero that is nonetheless dense in the attracting set. That is, the attracting set can be thought of as the closure of the set of all UPOs. The natural measure of the attractor is an eigenfunction of the Frobenius-Perron operator, $\mathcal{L}$, of the dynamics under consideration [9]. The characteristic equation of this operator is one way of defining the dynamical zeta function (DZF),

$$\zeta(z) = 1/\det\left(\mathcal{I} - z\mathcal{L}\right), \tag{2.21}$$

where $\mathcal{I}$ is the identity.

The location of the poles of $\zeta(z)$, as well as closely related functions, may be used to extract turbulent averages of observables and their correlations [9]. As is the case for other zeta functions, most notably the Riemann zeta function, $\zeta(z)$ can be expressed as an infinite product over the prime periodic orbits (i.e., periodic orbits that cannot be decomposed into a sum of smaller periodic ones) of the corresponding dynamical system. Formally this can be written as

$$\frac{1}{\zeta(z)} = \prod_p \left(1 - t_p\right), t_p = \frac{z^{n_p}}{|\Lambda_p|}, \tag{2.22}$$

where p stands for *prime* orbits, t_p is the period of the prime orbit p, n_p is the number of orbits with period t_p and Λ_p their respective stability eigenvalues. These eigenvalues are computed from the Jacobian matrix of the system, evaluated at the p periodic point with period t_p. Eigenvalues of magnitude greater then one represent unstable directions and diverging distances between orbits, magnitude less than one corresponds to stable directions (approaching distances) and a magnitude of one signals marginal directions, along which different trajectories in phase space maintain their distance.

For many systems, a good approximation to the DZF can be constructed by using the lowest-period UPOs of the flow, determined numerically [8]. As described in detail by Cvitanović *et al.*, knowledge of a finite set of low-period UPOs is often sufficient to estimate statistical averages over the natural measure of the attracting set. More recently, Kawasaki and Sasa [105] have argued that even a single UPO, with a large period, might suffice to characterize high-dimensional chaotic dynamical systems.

The main advantages of the UPO approach can be quickly summarized as follows:

$i)$ The degree of accuracy obtained can be very high and converges quickly with the number of known lower period UPOs. There is however a truncation factor, meaning that the accuracy will be limited by the first smaller period UPO we fail to include in the computation [9]. Techniques must then be developed to insure that we in fact found all of the UPOs up to a value of the period T.

$ii)$ There is no need to re-run lots of initial value problems every time we wish to compute the average of some new quantity. A long term goal is thus to construct and maintain a library of UPOs, from which all observable quantities can be systematically computed.

$iii)$ The averages thus obtained are no longer inherently stochastic since we now possess an exact expansion for their computation, as shown in great detail in [9]. This then presents us with the potential to get rid of the need for a statistical description of turbulence and thus signals the beginning of a new methodology for the study of turbulent fluid flow and possibly other driven dissipative systems as well.

The dynamical zeta function approach to the analysis of fluid turbulence has evolved through three historical stages. The first was the development of the DZF formalism in the 1970s [106]. The second was the recognition, in the late 1980s and 1990s, that this could be used as a practical numerical tool if it were only possible to compute UPOs for these systems [8]. Because the latter require petascale resources, and since those have only recently become available to academic researchers, we believe that this methodology is about to enter a third stage of research activity. By computing and classifying some of the smallest UPOs in the driven NSE we can now expect to be able to make statistical predictions for several important quantities from first principles.

In the next section we discuss several numerical methods that have been proposed to track UPOs in dynamical systems, with special emphasis on the variational approach suggested by Lan and Cvitanović [19].

2.4 Numerical methods

2.4.1 Introduction

In this section we turn our attention to the methods for locating UPOs in dynamical systems. Due to their inherent instability these periodic orbits cannot be found by forward integrating the equations of motion and special methods for their numerical detection must be devised. Several such methods have been proposed in recent years. Of these, the most commonly used by far is the Newton-Raphson method [21], in its many variants, with Poincaré sections.

The main criticism applied to this approach is that in chaotic systems the basin of attraction will shrink exponentially and Newton-Raphson will not converge unless the initial guess is already exceptionally good [107, 108]. However, this is still the main method of choice for most researchers in the field, usually using different increments to the basic Newton-Raphson method.

Examples of this include the work of Eckhardt and Ott [15] which located UPOs in the Lorenz system of equations, up to a symbolic length of 9. (For this system, the notation $\bar{0}$ and $\bar{1}$ is used to identify the different lobes in phase space traversed by the time-evolution of the system, and an orbit described by $\bar{0}\bar{1}\bar{0}$ will have a symbolic length of 3.) Using this approach good estimates for the Lyapunov exponent and the Hausdorff dimension[8] of the attractor were obtained.

Another example can be found in the work of Saiki [108] which applied the damped Newton-Raphson-Mees algorithm to the Lorenz model, obtaining more than a thousand UPOs in this fashion. Kato and Yamada [16] also used Newton-Raphson and the Newton-Raphson-Mees extension to locate UPOs in shell models of turbulence. These models use the Fourier transform representation of the NSE to reduce the number of degrees of freedom, whilst maintaining some of the basic features of turbulent behavior. Using this approach these authors found several UPOs and compared the probability density functions (PDFs) thus obtained for the velocity field with the ones obtained by straightforward statistical methods, with good agreement. More recently, Viswanath suggested the use of Newton-Krylov iterations, augmented by the locally constrained hook step, and applied this methodology to plane Couette flow, identifying several periodic orbits [100].

A very different numerical approach relies on stabilising the UPOs of the system through self-controlling feedback. The literature in this field is also quite extensive (see [110] and references therein). However, the emphasis of this line of enquiry lies rather in *eliminating* chaotic behaviour from experi-

[8]The Lyapunov exponent is a measure of the exponential separation of two adjacent trajectories, and thus of the amount of sensitivity of the system to initial conditions. The Hausdorff dimension can be defined as a generalization of the notion of the dimension of a real vector space. Trivial cases are the Hausdorff dimension of a single point, which is zero, of a line, which is one, and of a plane which is two. However, non-integral Hausdorff dimensions are also possible, and this, as we have seen, becomes relevant in the context of strange attractors, which often possess a fractal geometry. See [109] for an intuitive definition of these concepts.

mental systems and not so much on the detection of the UPOs *per se*. Finally we have the variational approach [9], which we adopt in this work, following a method that evolves from the one suggested by Lan and Cvitanović [19].

The numerical methods for finding periodic orbits in a system described by a group of differential equations have usually proceeded by the use of a *shooting method*. This procedure can be easily summarised. Let S be a surface contained in the set of total states of the system Ω (also called the phase space), and through which the orbit is known to pass. A point in S is chosen as an initial condition, $\mathbf{r}(0)$, and its time evolution followed until it returns to the surface S (usually referred to in this context as Poincaré section) at some time T. If we define the displacement $\delta = \mathbf{r}(T) - \mathbf{r}(0)$ then the rationale is to vary the initial starting point $\mathbf{r}(0)$ so as to make $\delta = 0$.

This procedure, which can be implemented using a multidimensional Newton-Raphson method, will not be likely to converge unless the initial point is already extremely close to a closed orbit (i.e., with $\delta \simeq 0$). Therefore other approaches for locating nearly periodic orbits should be considered. One of the most effective ones [8] consists in plotting the quantity

$$\Delta(t, T) = \parallel \mathbf{r}(t+T) - \mathbf{r}(t) \parallel \geq 0, \tag{2.23}$$

versus T, for several values of t, where $\Delta(t, 0) = 0$, with a suitably-defined norm. If we can find a value $T > 0$ for which $\Delta(t, T)$ is at a local minimum and has very small magnitude then the orbit will be nearly periodic. This approach will be followed throughout this work, and several plots of the quantity defined by Eq. (2.23), considering an Euclidean norm, will be shown and discussed in section 5.1.2. In the next section we discuss the next step after a suitable initial condition has been located.

2.4.2 Variational approach

Recently, progress has been made in this field by the introduction of a variational method [19] which searches for the whole periodic orbit of the system, beginning with a guess for a neighbouring closed trajectory. Based on this approach, new variational principles have been suggested [111], that extend this procedure to include the search for the period of the orbit as well. Preliminary numerical searches, on classic test cases such as the Lorenz model [14] and a limit cycle [109], suggest these principles to be well suited to the task of numerically computing UPOs. For the sake of brevity we delineate here only one of them, which illustrates the basic methodology adopted in this work.

Our aim is to find a solution $\mathbf{r}(t)$ for the differential equations

$$\dot{\mathbf{r}} = \mathbf{f}(\mathbf{r}), \tag{2.24}$$

where $\mathbf{f}$ is a vector field, $\mathbf{r}$ can contain many degrees of freedom and the dot represents differentiation with respect to time t. If T is the unknown period to be determined, one way to achieve this can then be to minimise the functional defined as:

$$\mathcal{F}[\mathbf{r},T] \equiv \frac{1}{2}\int_0^T dt|\dot{\mathbf{r}}(t)-\mathbf{f}(\mathbf{r}(t))|^2, \tag{2.25}$$

where $\mathcal{F} \geq 0$ from the definition and $\mathcal{F} = 0$ is valid only for solutions of Eq. (2.24). It will be assumed that $\mathbf{r}(t)$ is twice differentiable with respect to time. We now intend to vary the functional with respect to both $\mathbf{r}(t)$ and the unknown period T. From the Fundamental Theorem of Calculus we have:

$$\frac{\partial \mathcal{F}[\mathbf{r},T]}{\partial T} = \frac{1}{2}|\dot{\mathbf{r}}(T)-\mathbf{f}(\mathbf{r}(T))|^2, \tag{2.26}$$

and so the derivative of $\mathcal{F}$ with respect to the unknown period must vanish, as long as the equation of motion is valid at time T. Thus the problem of finding periodic solutions for Eq. (2.24) turns into the problem of minimising the functional $\mathcal{F}[\mathbf{r},T]$. The way this functional was defined invokes the L^2 norm, also called Euclidian norm, between the vector field $\mathbf{f}(\mathbf{r})$ and the time derivative of the variables $\mathbf{r}$. After a lengthy but straightforward calculation we can find expressions for the derivatives of the functional and obtain an explicit expression for the unknown period. Considering the change of variables

$$\tau \equiv \frac{t}{T} \in [0,1); \boldsymbol{\rho}(\tau) \equiv \mathbf{r}(\tau T), \tag{2.27}$$

the new variable $\boldsymbol{\rho}$ will be periodic with period one. Setting Eq. (2.26) equal to zero, the expression for the period can be found to be:

$$T = \sqrt{\frac{\int_0^1 d\sigma|\boldsymbol{\rho}'(\sigma)|^2}{\int_0^1 d\sigma|\mathbf{f}(\boldsymbol{\rho}(\sigma))|^2}}. \tag{2.28}$$

Another equation can be obtained, by computing the Fréchet derivative $\frac{\delta\mathcal{F}[\boldsymbol{\rho},T]}{\delta\boldsymbol{\rho}(\tau)}$, and setting it to zero:

$$0 = \frac{\boldsymbol{\rho}''}{T} + \{\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{\rho}(\tau)) - [\boldsymbol{\nabla}\mathbf{f}(\boldsymbol{\rho}(\tau))]^\top\}\cdot\boldsymbol{\rho}'(\tau) - \frac{1}{2}T\boldsymbol{\nabla}|\mathbf{f}(\boldsymbol{\rho}(\tau))|^2, \tag{2.29}$$

which will have periodic boundary conditions $\boldsymbol{\rho}(0) = \boldsymbol{\rho}(1)$. The superscript $\top$ denotes the transpose operator and must not be confused with the period T, which is given by Eq. (2.28).

These two equations can now be used to search for minima of the functional $\mathcal{F}$. One possible way to do this is by means of Ginzburg-Landau [112] equations, a method that has been used in previous fluid turbulence studies [37, 113]). In order to do that we introduce a new independent variable, a fictitious time, s, on which both $\boldsymbol{\rho}$ and T depend. We then have:

$$\frac{\partial\boldsymbol{\rho}(\tau,s)}{\partial s} = -\Gamma_{\boldsymbol{\rho}\boldsymbol{\rho}}\frac{\delta\mathcal{F}[\boldsymbol{\rho},T]}{\delta\boldsymbol{\rho}(\tau,s)} \tag{2.30}$$

$$\frac{dT(s)}{ds} = -\Gamma_{TT}\frac{\delta\mathcal{F}[\boldsymbol{\rho},T]}{\delta T(s)}. \tag{2.31}$$

Γ is a positive-definite linear operator, here assumed to be diagonal. The resulting equations (which we do not write here explicitly) can be simulated numerically, with the time step being s, and a suitable stopping criteria defined as the upper limit for the variation of each of the two main quantities, in successive iterations. In order to say we have found a periodic orbit we must also have $\mathcal{F} = 0$. Only then is a solution to the equations of motion effectively found. In the work previously mentioned [111] this method was applied to some canonical systems with very encouraging results. For the Lorenz equations, after a fourth-order Runge-Kutta time integration was performed for an initial condition reasonably close to an UPO, Ginzburg-Landau equations were used to evolve the estimate of the orbit and its period, with a high convergence rate.

The approach outlined here is a novel approach to the search for periodic orbits in differential equations, which parallelizes time and space. In Appendix A we apply this method to the lattice-Boltzmann method, via the lattice-BGK equation, introduced in Chapter 3, which simulates the NSE for the limit of low Mach number. First we discuss some of the preliminary work that made use of this variational principle and related computational issues.

2.4.3 Computational aspects

Based on this new variational approach, a C++ library was written by Jonas Lätt called "LUPO", which stands for "Locator for Unstable Periodic Orbits". LUPO includes modules for:

$i)$ Defining and manipulating dynamical systems;

$ii)$ Solving dynamical systems with a fourth-order Runge-Kutta integrator [21];

$iii)$ Locating unstable periodic orbits by a combination of Ginzburg-Landau relaxation, conjugate gradient solver [21], and multigrid technique in time;

$iv)$ Treating the Lorenz equations through a comprehensive list of tools;

$v)$ Searching for UPOs for 2D fluids simulated using the lattice-Boltzmann method, implemented in the OpenLB library [114].

Some of these points deserve further comment. For the Lorenz model, a comprehensive tool exists in LUPO which locates periodic orbits for a given symbolic sequence and whose results compare favourably with previous work in this area [15, 108]. Item v, mentioned above, is being actively pursued by the research collaboration based at Tufts University. Encouraging preliminary results have been obtained of UPOs in the quasi-2D experimental apparatus mentioned in [115]. This system consists of a thin layer of conducting fluid under a magnetohydrodynamic forcing and exhibits spatiotemporal chaos for sufficiently high Reynolds numbers.

Item iii, however, deserves a lengthier note. In the present work we did not make use of LUPO, although some of its philosophy and concepts were incorporated in our numerical relaxation algorithms. The main reasons for this will be made clearer in section 5.3. For now we should point out that the Lorenz equations only have 3 variables. Using LUPO, several thousands of Ginzburg-Landau and conjugate gradient iterations can be carried out on a desktop machine in a matter of just a few seconds. The computational time required will increase linearly with the length of period T of the orbit. Within the LUPO framework, the doubling of the number of discretized points for this system is carried out several times, in order to zoom in on the correct value of the period of the orbit. Again, this is still trivially performed on a serial machine.

For the case of 2D fluid simulations, memory requirements begin to be an issue and some level of parallelization is already required. Roughly speaking, for a 200^2 grid, an orbit with $T = 10^3$ will require about 6 Gigabytes of memory. As we move from the 2D to the 3D case two things happen. First of all we note that the memory requirements increase linearly by a factor of the order of the size of the (spacetime) lattice. This means that for typical 3D systems of interest we already need several Terabytes of memory, which requires the use of a rather large cluster. Indeed, the amounts of memory involved are so large that we encounter limits on the use of multigrid techniques and the doubling of grid points that LUPO performs. The second point is that the time needed to carry out either Ginzburg-Landau or conjugate gradient iterations also increases with the size of the lattice and again some severe limitations are found on the number of iterations that can be performed on realistic 3D systems. For these practical reasons our algorithms for UPO search in 3D fluids do not have the flexibility and generality that was implemented in LUPO and applied to lower dimensional problems.

With these *caveats* in mind we now return to the main argument in this thesis, which is to numerically search for unstable periodic solutions of the Navier-Stokes equations. In the next chapter we thus discuss the lattice-Boltzmann method, which we used to simulate the NSE.

CHAPTER 3

The lattice-Boltzmann Method

THE theory and concepts behind the lattice-Boltzmann method are discussed in this chapter. A brief overview is given of the hierarchy of physical approximations, going from Newton's equations of motion for individual particles to the Navier-Stokes equations, which follow the evolution in time of continuous macroscopic quantities. In between these two extremes we have "mesoscale" modelling, in which the lattice-Boltzmann method is included. By lattice-Boltzmann method we mean any numerical scheme that solves the Boltzmann equation for a distribution of interacting quasi-particles moving with constant velocities on a discrete lattice. We discuss some of the main assets and drawbacks of this approach, as well as some of the improvements that have been suggested to tackle the latter. This numerical framework has been used to simulate turbulent flow in various systems and a discussion is included of some of the main work done in this field.

3.1 Mesoscale modelling

Mesoscale models aim to describe systems which have intermediate length or time scales between a fine-grained, microscopic approach, and the macroscopic approach, such as the Navier-Stokes equations, for the case of fluid flow. These models are also particularly relevant in physical situations where processes operate at distinct scales [116], a case in point being complex liquids [117]. Another important area of application for mesoscale modelling is the study of multicomponent systems, such as amphiphilic liquids [118, 119], and immiscible, neutrally buoyant drops [120].

3.1.1 Atomistic dynamics

When considering a fluid we can view it as a collection of molecules, interacting in a given region of space. In what follows we shall ignore quantum effects, i.e., we assume that the de Broglie wavelength, $\lambda = h/p$, where h is the Planck constant and p the momentum, is much smaller than all other relevant length-scales. This assumption applies to the vast majority of fluids which surround us and are essential for life on this planet, from blood flow to atmospheric flows and ocean tides.

For a single-component fluid of non-interacting particles, this ensemble of molecules of equal mass, m, can then be described by the classical Newton equations of motion:

$$\dot{\mathbf{r}}_i = \frac{\mathbf{p}_i}{m} \quad \text{and} \quad \dot{\mathbf{p}}_i = \mathbf{F}_i, \tag{3.1}$$

relating their individual position, $\mathbf{r}_i$, and linear momentum, $\mathbf{p}_i$, with the force, $\mathbf{F}_i$ acting on each one of them. This is the starting point for all Molecular Dynamics methods in CFD. If we consider a number of N particles in 3D space we will then need to solve $6N$ differential equations. The quantity $6N$ is the dimensionality of the phase space for this system. A reasonable estimate for the order of magnitude of N is given by the Avogadro number, $Av \sim 6.022 \times 10^{23}$, that gives the number of atoms contained in 12-grams of carbon-12. This order of magnitude makes the task of solving Newton's equations of motion for each atom all but impossible. The largest existing computers on the planet are only now starting to tackle the petascale, memory-wise [121], a far cry from the order of magnitude implied by the Avogadro number.

Besides the sheer number of variables involved there is another reason why the atomistic approach is unfeasible for macroscopic systems: the dynamical instability of the phase space. By this we mean that any small uncertainty, δ_0, in the initial positions and/or momenta would grow exponentially in time, as $\delta(t) = \delta_0 e^{\chi t}$. The coefficient χ is known as the *Lyapunov exponent* and provides us with a measure of the temporal horizon of the deterministic behaviour of a system. More explicitly, at times greater than χ^{-1} the initial uncertainty will have grown in such a way that precludes any deterministic prediction of the state of the system [122].

Fortunately, we do not require such a level of resolution in order to describe properties at the macroscopic scale. It is sensible to assume that the details of how each microscopic particle behaves should not have a visible effect on length-scales removed from the microscopic one by very many orders of magnitude. The way to formalize this intuition is to adopt a statistical description instead, as was fully recognized by some of the greatest physicists of the 19^{th} century, such as Boltzmann, Maxwell and Gibbs.

3.1.2 The BBGKY Hierarchy and the Boltzmann Equation

Let us first recast Newton's equations (3.1) in a slightly different form, due to Hamilton [123]. We still assume a single component fluid, but we now include in the description the possibility of an interaction between the particles as well. A simplification often used is to consider a pairwise interaction potential varying with the distance between particles, $\Phi_{ij} = \Phi(|\mathbf{r}_i - \mathbf{r}_j|)$. The sum of the total kinetic energy T and total potential energy V for a system of N such particles is called the Hamiltonian H of the system:

$$H = T + V = \sum_{i=1}^{N} \frac{p_i^2}{2m} + \sum_{i=1}^{N} \sum_{j>i} \Phi_{ij}. \tag{3.2}$$

Hamilton's equations, which are found in any standard classical mechanics textbook, such as Goldstein's [123], can then be written as:

$$\dot{\mathbf{r}}_i = \frac{\partial H}{\partial \mathbf{p}_i}, \qquad \dot{\mathbf{p}}_i = -\frac{\partial H}{\partial \mathbf{r}_i}. \tag{3.3}$$

Again we have $6N$ differential equations to solve, and an unfeasible order of magnitude for N. We also note that these equations, as well as Newton's, are perfectly time-reversible, i.e., are invariant in regard to the reversal of time and velocities. As an important aside, $\mathbf{r}$ and $\mathbf{p}$ do not have to necessarily represent position and momentum. They are, in general, canonical coordinates in phase space, related by a canonical transformation and obeying the fundamental Poisson bracket relations [123].

The move towards a statistical description now requires us to introduce the concept of a particle distribution function. Let $\psi\left(\mathbf{r}_1, \mathbf{p}_1, \mathbf{r}_2, \mathbf{p}_2, \ldots, \mathbf{r}_N, \mathbf{p}_N; t\right)$ represent the probability of finding particle 1 at position $\mathbf{r}_1$ with momentum $\mathbf{p}_1$, particle 2 at position $\mathbf{r}_2$ with momentum $\mathbf{p}_2$, all the way to particle N at position $\mathbf{r}_N$ with momentum $\mathbf{p}_N$. Let us now define, for the sake of brevity, the quantity $\mathbf{z}_i = (\mathbf{r}_i, \mathbf{p}_i)$. The probability of each particle having a precisely defined position and momentum will be given by a delta-function. More generically, the continuous function ψ, assuming real values, can describe the dynamical evolution of a microcanonical ensemble of such well defined systems. The chain rule of derivation and the conservation of probability then imply that:

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = \frac{\partial \psi}{\partial t} + \sum_{i=1}^{N} \left(\frac{\partial \psi}{\partial \mathbf{z}_i} \frac{\partial \mathbf{z}_i}{\partial t} \right) = 0. \tag{3.4}$$

The result is called the Liouville equation in which the time derivatives $\dot{\mathbf{z}}_i$ are given by Hamilton's equations, (3.3).

The next step in this chain of thought is to define a one-body distribution function, which is the central object of kinetic theory. Let $f_1(\mathbf{r}, \mathbf{p}, t)$ describe the density of particles at position $\mathbf{r}$ with momentum $\mathbf{p}$ at time t. The one-body distribution function will be normalised in such a way that $\int f_1 \, \mathrm{d}\mathbf{z} = N$, whereas ψ, being a probability, is normalised to one. The relation between them can be seen to be:

$$f_1(\mathbf{r}_1', \mathbf{p}_1', t) = f_1(\mathbf{z}_1', t) \equiv N \int \psi(\mathbf{z}_1', \mathbf{z}_2, \ldots \mathbf{z}_N) \, \mathrm{d}\mathbf{z}_2 \cdots \mathrm{d}\mathbf{z}_N. \tag{3.5}$$

Any description taking into account only the one-body distribution function will leave out of the picture the interaction between pairs of particles. We note that Eq. (3.4) is still an exact expression which is also time-reversible. As regards its analytical solution, Bogoliubov, Born, Green, Kirkwood, and Yvon found that an infinite cascade of equations is obtained [124]. Each one of these equations will describe the dynamics of the n-particle distribution function as an integral over the $(n+1)$-particle distribution function. In other words, solving the equation for a one-body distribution, f_1, will require knowledge of the two-body distribution function, f_{12}, solving the equation for f_{12} will require knowledge of the three-body distribution function, f_{123}, and so on. This is known as the BBGKY hierarchy [124] and its framework was developed by the above-mentioned authors between the years 1935 and 1946.

In order to close this infinite hierarchy some assumption must be made about the behaviour of the molecules. This is precisely what was suggested by Ludwig Boltzmann in 1872, many decades before the BBGKY framework and working from first principles. The closure assumption made by Boltzmann is called *Stosszahlansatz* or *molecular chaos* and postulates that no correlations exist between colliding molecules [125]. Formally this is equivalent to saying:

$$f_{12} = f_1 f_2. \tag{3.6}$$

Boltzmann then derived an equation involving only one-body distribution functions which describes the dynamics of the molecules in terms of streaming and collision. The molecular interactions are described solely as localised collisions between two particles, due to a short-range potential. This celebrated equation can be written as:

$$(\partial_t + \mathbf{c}_1 \cdot \partial_{\mathbf{r}_1}) f_1 = \int g[f_1(\mathbf{r}, \mathbf{c}_1') f_1(\mathbf{r}, \mathbf{c}_2') - f_1(\mathbf{r}, \mathbf{c}_1) f_1(\mathbf{r}, \mathbf{c}_2)] \mathrm{d}b \, \mathrm{d}\mathbf{c}_2, \tag{3.7}$$

where we use the velocities $\mathbf{c}_i = \frac{\mathbf{p}_i}{m}$; primed velocities are post-collisional, and unprimed pre-collisional. In this notation, $\mathbf{r}$ represents the intermolecular distance $\mathbf{r}_1 - \mathbf{r}_2$ and g is the modulus of the relative velocity $\mathbf{c}_1 - \mathbf{c}_2$. Eq. (3.7) assumes we consider a coordinate system with z-coordinate parallel to the relative velocity $\mathbf{g}$ and impact parameter $\mathbf{b}$ perpendicular to $\mathbf{z}$. An extra term can be introduced on the left-hand side if an external force is applied to the system.

One of the most important characteristics of Boltzmann's equation is that time-reversibility has been lost. The introduction of the *Stosszahlansatz* assumption, necessary to close the BBGKY hierarchy, puts an end to the time-reversibility of the atomistic description and signals the appearance of the much-discussed *arrow of time* which is observed at macroscopic levels. Boltzmann showed that the following quantity

$$H \equiv - \int f \ln f \mathrm{d}\mathbf{c} \, \mathrm{d}\mathbf{r}. \tag{3.8}$$

always increases. The famous *H*-theorem states that

$$\frac{\mathrm{d}H}{\mathrm{d}t} \geq 0, \tag{3.9}$$

independently of the underlying microscopic potential. The implications of this have been widely discussed in the literature. Two very readable accounts can be found in the books by Coveney and Highfield [126] and by Cercignani [127]. The former traces the development of the arrow of time concept in physics and the path from reversible microscopic first-principle equations to macroscopic irreversibility. The book by Cercignani analyses in great detail the intellectual path followed by Boltzmann and the cold, even antagonistic reception his ideas encountered at the time.

This brief exposition is meant to illustrate the hierarchy of approximations involved when going from an atomistic to a macroscopic description of matter. From the Boltzmann equation there is also a

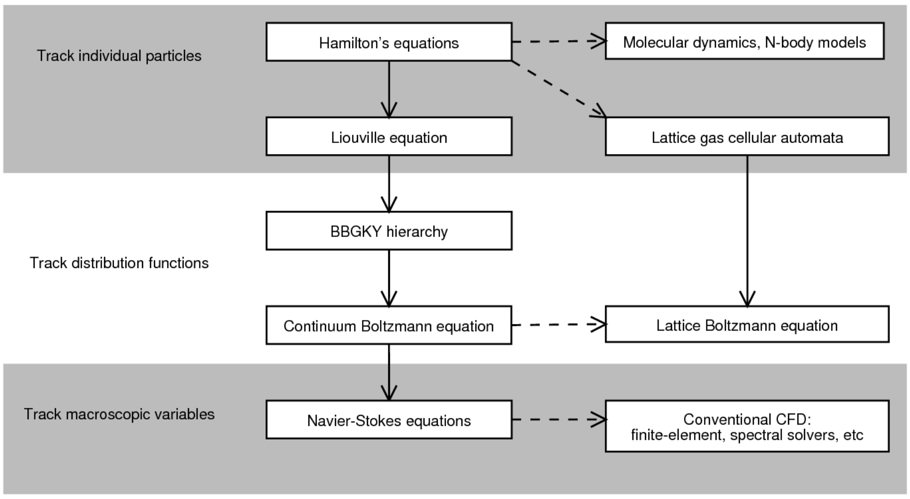

Track individual particles
Hamilton's equations
Molecular dynamics, N-body models
Liouville equation
Lattice gas cellular automata
BBGKY hierarchy
Track distribution functions
Continuum Boltzmann equation
Lattice Boltzmann equation
Track macroscopic variables
Navier-Stokes equations
Conventional CFD: finite-element, spectral solvers, etc

Figure 3.1: Levels of description in classical and computational kinetic theory.

clear path, via the Chapman-Enskog procedure [128], that leads to the Navier-Stokes equations. The latter can be seen as the mean field picture emerging from a perturbative treatment of the kinetic equations [35]. Figure 3.1, due to Jonathan Chin [129], adequately summarises this discussion and the hierarchy of approximations described. Several of the approaches used in conventional CFD have been discussed in Section 2.2. From the (continuum) Boltzmann equation we could now apply a discretization procedure and obtain the lattice-Boltzmann equation. However this was not the historical path followed, as described in the next section.

3.1.3 Lattice gas models

The origin of the lattice-Boltzmann method can be traced back to the lattice gas cellular automata (LGCA) models of the eighties, and particularly the work of Frisch, Hasslacher, and Pomeau [130] and Wolfram [131], who were amongst the first pioneers in the application of cellular automata to fluid flow simulation. Frisch *et al.* [130] showed that a simple automaton obeying conservation laws at a microscopic level could reproduce the complexity observed in macroscopic fluid flows.

It is interesting to note that the main motivation behind much of this research was the eventual simulation of turbulent flow, and even the prospect of tackling the limit of zero viscosity, as discussed in some detail by Succi [35]. The basic premise for these models is again that the macroscopic dynamics of a fluid should not be sensitive to the underlying details of the microscopic scale.

The LGCA models simulate a set of particles with discrete velocities, moving in discrete time steps on a discrete spatial lattice. All particles have the same mass and no two particles sitting on the same site can move along the same direction (exclusion principle). Mass and momentum conservation are obeyed at the collisions. In order to obtain the correct macroscopic equations it is necessary for the lattice to obey certain symmetry requirements. The correct macroscopic fluid dynamics were thus first obtained [130] on a 2D hexagonal lattice, i.e., with only just six different discrete velocities. Remarkably, it was shown that even such a simplified model could reproduce the main features of real fluid flows. A previous model, first proposed in 1973 by Hardy, Pazzis and Pomeau [132], with only four discrete velocities (square lattice), failed to achieve rotational invariance, associated with angular momentum invariance, an important feature of the NSE.

These first models considered Boolean variables for the particle distribution: any given particle had an occupation number which was either 0 or 1, leading to a Fermi-Dirac local equilibrium distribution. In this way, it can be said that each particle is still being individually tracked, as indicated in Fig. 3.1, and these models can be viewed as an ultra-simplified, spatio-temporally discrete dynamics approach.

3.2 The lattice-Boltzmann Equation

3.2.1 Historical development

The first lattice-Boltzmann equation (LBE), proposed by McNamara and Zanetti [133], aimed at eliminating the statistical noise in the LGCA model. Very briefly put, the LGCA, being an $N-$body Boolean system, was computing unnecessary many-body details. This, combined with the fact that modern-day computers are actually biased toward floating-point calculations, was resulting in the method lagging behind more conventional CFD approaches [35]. However, LGCA has found several areas of application and still survives to this day. Grosfils *et al.* [134] have shown that the statistical noise intrinsic to LGCA shares many quantitative features with the noise observed in thermodynamic systems, thus placing the method in a good position to address statistical micro-hydrodynamics.

Going back to the LBE, its basic idea was to replace the Boolean values used in the LGCA with real values for the single-particle distribution functions. In this approach, the distribution function f_i for particle i is now seen as the (real-valued) ensemble-average of the Boolean occupation numbers n_i:

$$f_i =< n_i > . \tag{3.10}$$

This represents a change in perspective in which we no longer track individual particles but instead follow the evolution in time of a series of microscopic degrees of freedom. This point of view again reverts to Boltzmann's equation, discussed in section 3.1.2, as the name "lattice-Boltzmann" already indicates. More explicitly, $f_i(\mathbf{r}, t)$ represents the probability of finding a particle whose discrete velocity is associated with index i at time step t and position $\mathbf{r}$ on the lattice. Besides getting rid of statistical noise, the LBE approach also proved in time to have several advantages over LGCA, such as allowing the simulation of flows with higher Reynolds numbers, providing a more straightforward implementation in three dimensions and the better accommodation of mesoscopic physics [35]. Nevertheless one important feature was lost in the transition: freedom from round-off errors which LGCA, due to the boolean nature of its distribution function, naturally possessed. This unconditional numerical stability was seen as a key asset of the method and one providing great hope to tackle the problem of turbulence [35].

Another crucial point is the following. Although the LBE can be obtained by a discretization of the continuum Boltzmann equation [135, 136], its collision operator is again a multibody one, as opposed to the strictly two-body collisions implied in the Boltzmann equation. This in turn means that we are no longer restricted to the regime of rarefied gas-dynamics, with its low-density and short-range interactions, but can aim instead at the modelling of more dense medium, such as liquids [35].

After the initial model of McNamara and Zanetti several lattice-Boltzmann models were proposed, in the late eighties and early nineties, all rapidly building on the previous modifications. The next major contribution was made by Higuera and Jimenez [137] who got rid of some of the spurious degrees of freedom still implied by the earlier model, thus making the LBE more amenable to three-dimensional computations. These authors proposed a linear collision operator, thus assuming that the distribution functions are close to the local equilibrium. Then Higuera, Succi and Benzi [138] proposed a slightly different version for the local collision operator and demonstrated the linear stability of this version. This last achievement opened the way to the simulation of high Reynolds number, by eliminating a previously existing lower constraint on the viscosity (still an inheritance from intrinsinc limitations of the LGCA), and thus securing a place for the LBE within the fluid turbulence recipe book.

After this period of intense developments, the method was gradually adopted by a large community of scientists, working in topics as diverse as fluid turbulence, driven cavity flows, flows in complex geometries and multiphase flows (for early reviews of the method and its applications see the articles by Benzi, Succi and Vergassola [139] and Chen and Doolen [140]).

The lattice-Boltzmann method (LBM) consists of two basic steps: the streaming of each pseudo-particle (represented by a distribution function) to a neighbouring site, whose location is precisely defined by the discrete value of its velocity, and a collision after it "arrives" there, which plays the role of a relaxation mechanism to a local equilibrium, also carefully defined. The expression for the equilibrium distribution is chosen so as to incorporate significant desired features, such as conservation of mass and momentum at each lattice site.

Following Chen and Doolen [140] and Succi [35] we point out three main advantages of using the LBE instead of other more conventional numerical methods for fluid simulation. These can be stated as follows:

$i)$ The streaming operator (corresponding to the advection of the particles) is linear; the nonlinearity (intrinsic in the hydrodynamic description) is recovered by means of multi-scale expansions of the streaming operator combined with the collision operator. This will also have dramatic advantages in terms of parallelization, which will be discussed later on (Chapter 4);

$ii)$ The pressure term, present in Eq. (2.4), is now given by an equation of state:

$$P = \rho c_s^2, \tag{3.11}$$

with c_s being the lattice sound speed and ρ the density. In conventional CFD methods the pressure usually obeys an equation of Poisson type, with velocity strains acting as sources. The need to solve this equation often produces extra numerical difficulties, requiring special treatments such as iteration or relaxation methods;

$iii)$ The fact that only a minimal set of velocities is used greatly simplifies the procedure of obtaining macroscopic quantities. These are now given by averages computed in the phase space formed by the particle distribution functions.

3.2.2 Lattice-BGK

We start by writing the lattice-Boltzmann equation, in the following way:

$$f_i(\mathbf{r}+\mathbf{c}_i, t+1) = f_i(\mathbf{r}, t) + \Omega_i(\mathbf{r}, t), \tag{3.12}$$

where $f_i(\mathbf{r}, t)$ is the distribution function of the particle located at site $\mathbf{r}$ at time t with velocity index i, $\Omega_i(\mathbf{r}, t)$ is the collision operator and $\mathbf{c}_i$ represents the velocities of each particle, with i assuming integer values $1, \ldots, Q$, with Q being the number of velocities[1] It must also be noted that the method obeys an exclusion principle: no two particles of the same species (assuming the possibility of multicomponent flows as well) with the same velocity will be at a given instant in the same lattice site.

From the particle distribution functions we can now obtain macroscopic quantities, such as density, ρ, and velocity field, $\mathbf{u}$, defined as:

$$\rho = \sum_{i=1}^{Q} f_i, \quad \mathbf{u} = \frac{1}{\rho}\sum_{i=1}^{Q} f_i \mathbf{c}_i. \tag{3.13}$$

The previous relations imply the case of a single component fluid with unit mass. This will be assumed throughout the rest of this work, unless otherwise stated.

The collision operator satisfies both mass and momentum conservation at each lattice site, i.e.:

$$\sum_{i=1}^{Q} \Omega_i = 0, \quad \sum_{i=1}^{Q} \Omega_i \mathbf{c}_i = \mathbf{0}. \tag{3.14}$$

For the collision operator we consider the assumption that the distribution is close to the local equilibrium state. This approach is called quasi-linear LBE [137] and the corresponding equation is:

$$f_i(\mathbf{r}+\mathbf{c}_i, t+1) - f_i(\mathbf{r}, t) = \mathcal{S}_{ij}(f_j(\mathbf{r}, t) - f_j^{eq}(\mathbf{r}, t)), \tag{3.15}$$

where f_j^{eq} denotes the local equilibrium state and $\mathcal{S}_{ij}$ is called the scattering matrix, determining the scattering rate between directions i and j. One further simplification, suggested almost simultaneously by several authors [141–143], is to consider a diagonal matrix and assume a single relaxation value, τ, towards the local equilibrium. This picture is based on the Bhatnagar-Gross-Krook model for the

[1]Lattice units are assumed, as will be the case from here on, unless otherwise stated.

Boltzmann equation in continuum kinetic theory [144] and is therefore dubbed lattice-BGK (LBGK). Formally, this is written as:

$$f_i(\mathbf{r}+\mathbf{c}_i, t+1) - f_i(\mathbf{r}, t) = -\frac{1}{\tau}(f_i(\mathbf{r}, t) - f_i^{eq}(\mathbf{r}, t)). \tag{3.16}$$

In this model the (kinetic) viscosity, ν, is related to the relaxation time through the expression

$$\nu = \left(\tau - \frac{\Delta t}{2}\right) c_s^2, \tag{3.17}$$

where Δt is the size of the time step which will be assumed equal to unity throughout this work.

We still need to write the local equilibrium distribution function, in order for the method to be consistently defined. Since the Navier-Stokes equations have a second order nonlinearity, an expansion up to $\mathcal{O}(u^2)$ is usually considered [140, 141]:

$$f_i^{eq}(\rho, \mathbf{u}) = \rho\left[a + b\mathbf{c}_i.\mathbf{u} + c(\mathbf{c}_i.\mathbf{u})^2 + du^2\right], \tag{3.18}$$

where ρ and $\mathbf{u}$ were defined in Eq. (3.13) and a, b, c, d are lattice constants. The values of these constants will depend on the particular model (lattice) considered and must obey the following (four) constraints:

$$\sum_{i=1}^{Q} f_i^{eq} = \rho, \quad \sum_{i=1}^{Q} f_i^{eq}\mathbf{c}_i = \rho\mathbf{u}, \tag{3.19}$$

which again impose conservation of mass and momentum, as in Eqs. (3.14). Due to its being a second order expansion, expression (3.18) will only be valid for relatively small values of the velocity $u \equiv |\mathbf{u}|$. The adimensional control parameter which is relevant here is the Mach number, defined as u/c_s, with c_s being the sound speed in the lattice.

Although historically the lattice-Boltzmann equation was introduced as a way to overcome some of the problems plaguing the LGCA models, it was later shown [135, 136] that it could also be derived from the continuum Boltzmann equation for discrete velocities. This is performed by means of a Mach number expansion, starting from the Boltzmann BGK equation.

Due to its simplicity the lattice-BGK scheme is the one most researchers in this area favour. As Sauro Succi, one of the main architects of the LBE, points out: “LBGK somehow marks the endpoint of the basic development of LBE theory (. . .) it seems fair to say that the hard-core of the basic theory lies within the 1989-1992 developments.” [35]

However, several important questions have been left out of this account. In the next section we discuss two other important lattice-Boltzmann models, which address some of the potential shortcomings of the LBGK model.

3.3 Beyond lattice-BGK

3.3.1 Multiple Relaxation Time

Going back to Eq. (3.15), which is the more general LB formulation, we do not necessarily have to assume a diagonal scattering matrix with a single eigenvalue as is done in the LBGK model. The reason why this was done is somewhat involved, but basically, in the LBE suggested by Higuera, Succi and Benzi [138], the last step before LBGK, the viscosity of the fluid was controlled by the leading nonzero eigenvalue of the scattering matrix. The other eigenvalues were related to several non-hydrodynamic modes, so-called "ghost-fields" [35]. It then seemed the next logical step to get rid of those non-hydrodynamic variables and use instead a diagonal matrix, of the form

$$\mathcal{S}_{ij}(f_j - f_j^{eq}) \equiv -\omega\delta_{ij}(f_j - f_j^{eq}), \tag{3.20}$$

where $\omega = \frac{1}{\tau}$.

At the same time that this model was proposed, another approach [145] emphasised and explored the freedom attained by having more parameters in the scattering matrix. This method, known as Multiple-Relaxation-Time Lattice-Boltzmann Equation (MRT-LBE), is reported by some authors (see d'Humières *et al.* [146] and references therein) to be more numerically stable than LBGK, due to the possibility to individually tune each different relaxation time towards an optimal stability.

Here we will only briefly sketch the main concept of the method, following the terminology adopted by d'Humières *et al.* [146]. We start by defining the set of particle distribution functions for a model with b discrete velocities in any given site on the lattice as:

$$< f \mid = (f_1, f_2, ..., f_b), \tag{3.21}$$

with the corresponding set of discrete velocities being written as

$$\mid \mathbf{c} >= (\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_b). \tag{3.22}$$

The Dirac notation of bra and ket vectors denote, respectively, row and column vectors. We now define a set of moments given by

$$m_\beta \equiv < \phi_\beta \mid f >=< f \mid \phi_\beta >, \tag{3.23}$$

where $\beta = 1, 2, ..., b$ and the set formed by all $\mid \phi_\beta >$ is an orthogonal dual basis set constructed by a Gram-Schmidt orthogonalization procedure from polynomials of the column vectors corresponding to the different velocity components. Due to this definition, the MRT-LBE is also sometimes referred to as the moment method. If we denote the velocity space by $\mathbb{V}$, spanned by $\mid f >\equiv (f_1, f_2, ..., f_b)^\top$, with $\top$ denoting the transpose operator, and the moment space by $\mathbb{M}$, spanned by $\mid m >\equiv (m_1, m_2, ..., m_b)$, there is a linear mapping M between these two spaces, such that:

$$\mid m >= M \mid f >, \mid f >= M^{-1} \mid m > . \tag{3.24}$$

If the scattering matrix $\mathcal{S}$ is chosen in such a way that the set $\{\mid \phi_\beta >\}$ are its eigenvectors, then the collision process can be described by the linear relaxation of the kinetic modes in moment space [145, 147], which allows us to rewrite Eq. (3.15) as:

$$\mid f(\mathbf{r}+\mathbf{c}_i, t+1) > - \mid f(\mathbf{r}, t) >= M^{-1}\hat{\mathcal{S}}\left(\mid m(\mathbf{r}, t) > - \mid m^{eq}(\mathbf{r}, t) >\right), \tag{3.25}$$

where $\hat{\mathcal{S}} = M\mathcal{S}M^{-1}$ is diagonal, and m_β^{eq} is the equilibrium value of the moment m_β. We will have a number b of moments, some of which are locally conserved in the collision process (and thus called "hydrodynamic") and others which are not (called "kinetic"). It must be noted that the equilibria $\{m_\beta^{eq}\}$ will be functions of the conserved moments and are invariant under the symmetry group of the lattice. For a model simulating an athermal fluid, the only conserved quantities will be mass density (a scalar) and momentum (a vector). The equilibrium values of the kinetic moments will be functions of the density and momentum.

The computational efficiency of the MRT is slightly lower than LBGK, but with good optimisation techniques it is claimed that it could be only 15% slower than the single relaxation time method [146]. The main advantages are that we now have the maximum number of adjustable relaxation times, and the maximum freedom in choosing the equilibrium function for the non-conserved moments. This in turn is claimed to allow for a greater numerical stability as previously stated.

In the exposition of the MRT-LBE made by d'Humières *et al.* [146] in 2002, the relaxation parameters of the kinetic (or ghost) modes are obtained by a linear stability analysis. A separation of scales is also implied, in which the relaxation times of the non-conserved quantities are much faster than the hydrodynamic time-scales.

A further refinement to this approach was introduced by Lätt and Chopard in 2006 [148], in which the values for the relaxation parameters are determined from physical arguments, based on the Chapman-Enskog expansion of the BGK model. These authors note that the hydrodynamic limit of the BGK model is only dependent on the values of the first three moments (density, velocity and stress tensor). In their scheme, these moments are computed at each time step at every lattice site, and this information is used to "regularise" the distribution functions at the collision. The method is thus called Regularised Lattice-Boltzmann (RLB). Both the MRT-LBE and RLB have been implemented in OpenLB [114], an open source numerical library containing several LB models and features.

3.3.2 Entropic lattice-Boltzmann

One thing that has been lost when going from the continuum Boltzmann equation to the lattice-Boltzmann one is the *H*-theorem, Eq. (3.9). In fact this was still being obeyed in the LGCA model [130]. The loss of this important property in the LBE lies in ignoring the microscopic origin of hydrodynamic behaviour and adopting instead a finite-difference approach, as pointed out by Boghosian *et al.* [149].

The fact that evolution of entropy to a maximum is no longer guaranteed implies in turn that the nonlinear stability of the LBE is also not guaranteed. This state of affairs becomes particularly relevant as we increase the Reynolds number and nonlinear effects become progressively more relevant. We note that this causal link between the existence of an *H*-theorem and the guarantee of numerical stability is not specific to the LBE, but an important guiding concept in computational physics [150].

The solution to this drawback, proposed independently by several authors in a short period of time [149, 151, 152], is to "take one step back" in the direction of kinetic physics. The resulting picture that arose from these efforts is called the Entropic lattice-Boltzmann method. Good introductory discussions to this topic can be found in the book by Succi [35], in the review article by Succi, Karlin and Chen [150] and in the paper by Boghosian *et al.* [149]. In what follows we will just sketch the main arguments involved.

The idea behind the Entropic lattice-Boltzmann method (ELB) requires the definition of an *H* function, instead of the expression for the local equilibrium distribution as was the case in the previous LBE approaches. The equilibrium distribution is then derived by extremizing the *H* function. Several different proposals exist for the form of *H*, leading to slightly different numerical realisations. One of the first proposals, by Karlin, Ferrante and Öttinger [151] was:

$$H \equiv \sum_{i=1}^{Q} f_i \, ln \left(\frac{f_i}{w_i} \right), \tag{3.26}$$

where w_i are speed-dependent weights in the equilibrium distribution function. If we write the *H* function as a sum of discrete contributions:

$$H = \sum_{i=1}^{Q} h_i, \tag{3.27}$$

we see that definition (3.26) allows for weighted contributions to H, and that h_i has the form of a (relative) Boltzmann entropy. Another approach was suggested by Boghosian *el al* [149] where the contributions h_i are uniform, with no dependence on lattice weights, although they are not Boltzmann entropies:

$$H = \sum_{i=1}^{Q} h(f_i). \tag{3.28}$$

Both models are valid in terms of restoring the *H*-theorem as well as leading to Galilean-invariant hydrodynamics, as shown in [153] and [154] respectively. As mentioned by Succi *et al.* [150], this is one of three very important properties that entropic LB models satisfy. The other two are non-negativity of the distribution function (realizability) and ease of determining the equilibrium distribution function at each site and at each time step (solvability). Regarding the former property, it is well known that negative values for the LB distribution functions easily arise for low values of the viscosity or for high velocities, close to the lattice sound speed. These non-physical values often signal the appearance of serious numerical instabilities, which can propagate quite easily on the lattice, due to the action of the streaming operator, and render the simulation meaningless, as witnessed by this author. This will be discussed in Chapter 4, dealing with LB implementation.

Going back to Eq. (3.28) we now extremize *H* with respect to the distribution functions, and impose conservation of mass and momentum, using the definitions (3.13), which gives:

$$0 = \frac{\partial}{\partial f_i}(H - A\,\rho - \mathbf{B}.\,\rho\mathbf{u}), \tag{3.29}$$

where A and $\mathbf{B}$ are Lagrange multipliers. From here we can compute an expression for the f_i^{eq} distribution functions by performing a Taylor expansion in Mach number. The details are somewhat involved and can be found in [154]. The expression thus obtained will, by construction, always increase entropy at each collision step.

Numerical tests have been performed, usually comparing the ELB with the LBGK approach. Among the first such comparisons, Ansumali and Karlin [155] focused on shock tube tests. This work showed a better numerical stability of ELB *versus* LBGK, but the authors also warned about potential implementation pitfalls for ELB, such as the extensive use of logarithmic functions, more vulnerable to round-off errors. A similar comparison was performed by Chikatamarla and Karlin [153], with the *H*-function this time constructed *via* Hermite polynomials. Again, ELB was found to be more resilient than LBGK, for the test-case of a one-dimensional shock tube.

Both the ELB and the MRT-LBE show that most of the shortcomings that plague the lattice-Boltzmann method can be overcome, putting the method on a par with more established CFD approaches. However, for a huge number of situations it is safe to say that the vast majority of researchers in the field are content with the LBGK approach, which is also much more straightforward to implement and computationally efficient. In the next section we will discuss the application of LB methods to turbulence studies, before moving on to the details of the computational implementation performed in this work.

3.4 Simulation of turbulent flow using lattice-Boltzmann methods

One common distinction of the numerical methods for the simulation of turbulent flow is that between direct numerical simulation (DNS) and methods with implicit turbulence modelling. In the latter, only the larger scales of the flow are explicitly accounted for, and a model is assumed to describe the lower scales, as mentioned in Section 2.2.2. In the present section the label "DNS" refers to LB studies of incompressible turbulent flow without any underlying turbulence modelling. As discussed below, the LBM also allows for the inclusion of such models, with quite impressive results, and an overview of work in this area is also included here.

3.4.1 Direct numerical simulation

In Section 3.2 we mentioned that one of the main advantages of lattice-Boltzmann over lattice gas methods was the possibility to have lower viscosities and thus the possibility of studying flows at higher

Reynolds numbers. This feature was explored early on in the development of the LB method. The earliest such studies focused on 2D forced isotropic turbulence [156]. In this work, Benzi and Succi compared the values obtained using the LBE (with enhanced collisions [138]) with previous numerical studies using spectral methods. A 512^2 lattice was used, with $\nu = 0.05$, corresponding to $Re \sim 10^3$. Time averages for the enstrophy (defined in Eq. (2.17)) and energy and their fluctuations were compared with the values obtained from the spectral method, with very good agreement found.

Martínez *et al.* [157] also studied 2D flows but for decaying turbulence of a shear layer, comparing the results obtained with the LBGK and a pseudo-spectral method. In this work the initial Reynolds number was 10^4 and the lattice had dimensions of 512^2, with the simulation running for 80 large-eddy turnover times. A comparison was performed between the spatial distribution, the vorticity fields and the time evolution of stream functions obtained using both methods. It was found that LBGK provided time histories of global quantities, wavenumber spectra and vorticity contour plots which were very similar to those obtained from the spectral method. Also Qian *et al.* [158] reproduced the $k^{-5/3}$ inertial range scaling in forced 2D turbulence, where energy now flows from the smaller to the larger scales (as opposed to what happens in 3D), as predicted theoretically by Kraichnan in 1967 [159].

Other workers in the field quickly moved on to the study of 3D turbulence. Chen *et al.* [160] validated the LBM for the study of 3D isotropic turbulence, looking at Beltrami flows (of which more will be said in Section 4.4), the decaying Taylor-Green vortex (see the book by Uriel Frisch [3] for a discussion of this) and decaying 3D turbulence. Again, a very good agreement was found between the spatial and time distributions of both the velocity and vorticity fields computed using the LBM and spectral methods. This was also confirmed by Treviño and Higuera [161] who studied the nonlinear stability of Kolmogorov flows [3] using both the LBM and the pseudo-spectral method at several values of the Reynolds number.

One final note should be made regarding the comparison of results obtained using LBM and spectral calculations. In lattice-Boltzamnn, space is normalised in units of length of the lattice, L_{LBM}. The relevant speed in homogeneous isotropic turbulence is

$$u_{rms} \equiv \sqrt{< u >^2}, \tag{3.30}$$

where "RMS" stands for "root-mean-squared" and the average is taken over the entire lattice [35]. For pseudo-spectral codes (PS) the most common normalisation is $L_{PS} = 2\pi$, the size of the periodic box, and $u_{PS} = 1$. In this way, the ratio $t_{PS} = \frac{L_{PS}}{2\pi u_{PS}}$, which defines the typical large eddy turn-over time, becomes one. The ratio between the LBM and PS clocks will then be given by L_{LBM}/u_{rms}. Assuming $u_{rms} \sim 0.2$ (the value of the lattice sound speed in most LB models is typically ~ 0.577, and u_{rms} should not be greater, since we are working in a nearly-incompressible regime), for $L_{LBM} = 1024$ (a typical value for the simulation of high Reynolds numbers) this gives a ratio of ~ 5000, for the simulation of the typical lifetime of a large-scale eddy. This must then be taken into account when comparing the results from the different methods [35].

3.4.2 Lattice-Boltzmann with turbulence modelling

As discussed in Section 2.2.2, it is usual in turbulence studies to consider a subgrid-scale model to deal with the smaller length scales of the fluid, whereas the larger ones are still tracked explicitly. This has also been applied to the LBM scheme with good results. One of the earliest such studies was performed by Hou *et al.* [162], who applied the Smagorinsky model [51] directly to LB, by filtering the particle distribution in Eq. (3.12) function through a standard box filter. The nonlinear term was accounted for by using an effective relaxation time of the form

$$\tau_{ef} \equiv 3\nu + C_S^2 \Delta^2 |\overline{S}|^2 + \frac{1}{2} \tag{3.31}$$

in the LBGK model, Eq. (3.16). In Eq. (3.31), ν is the kinematic viscosity, given by Eq. (3.17), Δ is the filter width, $|\overline{S}|$ is the amplitude of the filtered large-scale strain-rate tensor and C_S is the Smagorinsky constant [51]. This subgrid model was used in the simulation of flows with Re up to 10^6, using a 256^2 lattice.

Several other models have been proposed in this context. Using the LBM SGS model, Sommers *et al.* [163] studied three-dimensional pipe flow, up to $Re = 5 \times 10^4$. Succi *et al.* [164] applied the LB relaxation scheme to the solution of the $k - \epsilon$ equations [3], where k stands for the kinetic energy and ϵ for the dissipation. Eggels [165] included the turbulent stress tensor directly in the expression for the equilibrium distribution, and used the resulting model to simulate turbulent flow in a baffled stirred tank reactor, an industrial application of great practical relevance.

Lätt *et al.* [166], used the $D3Q19$ LBGK model for the simulation of weakly-turbulent flow, with Reynolds numbers of the order of $\mathcal{O}(10^3)$. In this work the authors propose a framework for the modelling of turbulence effects, by using the LB variables, and then testing their results against DNS data. Their approach consisted of applying the $k - \epsilon$ model [68], with its filtering procedure, directly to the LB distribution functions, and thus obtaining an effective viscosity.

Another recent development can be found in the work by Chen *et al.* [167]. These authors applied an effective viscosity to the LBGK scheme computed through the use of renormalization group methods. The resulting LB model was used to study the flow of air past a realistic car geometry, using the PowerFLOW code [168]. For this case the values obtained for the drag were within 5% of the experimental data. Turbulent flow in a planar channel was also studied with this model, and the results from the LB simulations compared with experimental data, for Reynolds numbers in the range $10^4 < Re < 10^6$, with an excellent agreement found.

These examples are intended to show that the lattice-Boltzmann method is quite capable of tackling some of the main challenges in turbulent research, including wall-turbulence interactions, as the last example showed.

3.4.3 Lattice-Boltzmann in the context of CFD

Some final comments are due, regarding the characterization of the lattice-Boltzmann method in the wider context of computational fluid dynamics and numerical methods in general. Following Succi [35], we can say that the LBM is an explicit, Lagrangian approximation of the Navier-Stokes equations in the nearly-incompressible limit. An explicit numerical scheme (as opposed to an implicit scheme), is one whereby the state of a given variable is computed by using the values of a number of neighbour states at a preceding time step. In an implicit scheme, there is a further dependence on the simultaneous state of the neighbourhood, which makes these methods more computationally demanding, on a per time step basis.

Other important features of the LBM include the fact that it is fully local in space and time (causality) and also second-order accurate in space and time. Besides this, it is unconditionally linearly stable, provided that the (single) relaxation time parameter is greater than $1/2$, (for which the viscosity would assume an infinite magnitude, as can be seen in Eq. (3.17)).

The accuracy and performance of the lattice-Boltzmann method have been compared to several other CFD methods. This has already been discussed on section 3.4.1 in the context of turbulence studies. Here we shall refer to the main CFD methods discussed in section 2.2.2. Noble *et al.* [169] simulated flow in an infinite periodic array of octagonal cylinders using both the LBM, with an LBGK collision operator, and a finite-difference scheme. These authors found that the streamwise and transverse velocity predictions generated by each method agreed to within 0.5% of the average streamwise velocity. Sankaranarayanan *et al.* [170] also compared the LBM with a (front-tracking) finite-difference scheme, but this time in the context of bubble rising on a $2D$ periodic box, and found quantitative agreement to within just a few percent.

Comparisons between LBM and finite-volume methods have also been performed by several authors. Bensdorf *et al.* [171] compared these two methods in the case of complex geometries and found excellent agreement between the numerical values of the flow field, on lattices/grids with comparable size. These authors also argue that as the complexity of the geometry increases the LBM eventually becomes more computationally efficient. Breuer *et al.* [172] compared the two methods in the study of laminar flow past a square cylinder, in 2D systems, with Re as high as 300 and found an excellent agreement between velocity profiles and several integral parameters.

Geller *et al.* [173] compared the LBM with both finite-element and finite-volume methods, reporting an excellent agreement between these, and a higher efficiency of the LBM in the case of weakly compressible flow. Khandai *et al.* [174] also compared LBM with FE in the simulation of a static mixer, which consists of specially designed stationary obstacles inserted in a pipe in order to promote the mixing of fluid streams flowing through it, and report an excellent agreement between the two methods.

Comparisons between LBM and spectral methods can be found in the work by Martínez *et al.* [157], in the context of decaying turbulence, and Chen *et al.* [175], who studied Beltrami flows and the decaying Taylor-Green vortex, with very good agreement reported in both cases.

The results of these various studies have confirmed that the lattice-Boltzmann method is indeed competitive with the other more established approaches. Indeed, it is actually faster in situations where a given specific accuracy is required, particularly in the context of the time-dependent simulation of large, complex systems by means of parallel implementations [176]. However, one should be aware that comparisons between the efficiency of different fluid solvers are always prone to some ambiguity since their accuracy, intrinsic speed and convergence behavior will be heavily dependent on the specific details of each implementation.

In the next chapter we shall describe the details of the particular LB implementation used in this work and its utilization in the simulation of weakly-turbulent flow.

CHAPTER 4

HYPO4D

The implementation of a fully parallel lattice-Boltzmann solver, written in the C programming language and using MPI for inter-processor communications, is described in this chapter. Its goal is to simulate turbulent flow accurately and in a computationally efficient way and then locate unstable periodic orbits for that flow. The code was written taking into account sound implementation techniques suggested by previous groups and demonstrated excellent scalability, up to tens of thousands of computational cores. The resulting software package was dubbed “HYPO4D”, which stands for “HYdrodynamic Periodic Orbits in 4 Dimensions”. In the present chapter we are mainly concerned with the (three-dimensional) fluid solver module of the code, which is at the core of the 4D module as well. The 4D aspects of the work will be presented in Chapter 5. We also describe here the numerical tests performed for validation of the fluid solver and its deployment on very many Grid resources. The results obtained for weakly-turbulent flows are discussed in the last section.

We note that the whole software package was written and developed from scratch by the present author of this thesis. The main achievements discussed in this Chapter are thus: the implementation of a fully-parallel, highly-efficient lattice-Boltzmann software package; its numerical validation using known analytical solutions of the Navier-Stokes equations, including a case with bounce-back boundary conditions; the demonstration of its excellent scalability up to 65K computational cores; and, lastly, its validation in describing weakly-turbulent flow, including a detailed analysis of an ABC flow in a periodic domain and its transition from laminar to turbulent behaviour.

4.1 Software description

4.1.1 The D3Q19 lattice

As pointed out in section 3.2.2, several lattices are possible for the LBGK scheme, consisting of different sets of discrete velocities. The guiding principle behind the various possible models is the requirement of enough symmetry to recover the Navier-Stokes equations, as was first recognized by Frisch, Hasslacher and Pomeau [130], in their lattice gas model. Formally this is equivalent to saying that the weights, ω_i

associated with each discrete velocity, $\mathbf{c}_i$, must obey the following symmetry requirements

$$\sum_{i=1}^{Q} \omega_i = 1, \tag{4.1}$$

$$\sum_{i=1}^{Q} \omega_i \mathbf{c}_i = \mathbf{0}, \tag{4.2}$$

$$\sum_{i=1}^{Q} \omega_i \mathbf{c}_i \mathbf{c}_i = AI, \tag{4.3}$$

$$\sum_{i=1}^{Q} \omega_i \mathbf{c}_i \mathbf{c}_i \mathbf{c}_i = \mathbf{0}, \tag{4.4}$$

where I is the identity, and A a normalization constant. The constraints are trivially related to mass and momentum conservation, already mentioned in section 3.2.2, as well as isotropy. However, these constraints are still not enough to completely determine a lattice, since in general we will have many more discrete velocities than the number of constraints. This freedom should thus be optimally exploited for the construction of LBGK models [35].

Qian, d'Humiéres and Lallemand [143] described a whole family of such solutions, dubbed $DnQm$, for m vectors in n dimensions. In this work we focus on 3D models only, for which $n = 3$. The number of velocities chosen will have an impact on the numerical stability and robustness of the model, as discussed in detail by Mei *et al.* [177]. A trade-off exists between numerical stability and amount of memory required, as should be intuitively expected. Mei *et al.* showed that the $D3Q19$ model represents an excellent compromise between the more instability-prone $D3Q15$ lattice and the more memory-intensive $D3Q27$.

The easiest way to visualize these different models is to keep in mind that the discrete velocities determine which directions of motion are allowed in the lattice, which is equivalent to think in terms of which nearest neighbours are being considered. This information is summarized in Table 4.1 for the three models, following Mei *et al.* [177], where the values for the lattice sound speed, c_s, allowed discrete velocities, their respective modulus and associated weights are shown.

In the streaming operation, the "rest" pseudo-particle, with discrete velocity $(0, 0, 0)$, does not suffer any change, although it will still contribute to the collision step, where its density value can vary. Each one of the other pseudo-particles (probability distribution functions) will be streamed to the neighbor site unequivocally defined by its corresponding discrete velocities. For this family of models the equilibrium distribution function, which obeys the constraints given by Eqs. (4.1)-(4.4), will have the form [143]

$$f_i^{eq}(\rho, \mathbf{u}) = \omega_i \rho [1 + \frac{1}{c_s^2}\mathbf{c}_i \cdot \mathbf{u} + \frac{1}{2c_s^4}(\mathbf{c}_i \cdot \mathbf{u})^2 - \frac{1}{2c_s^2}(\mathbf{u} \cdot \mathbf{u})], \tag{4.5}$$

where the weights ω_i have values $1/3$, $1/18$ and $1/36$ for $|\vec{c}_i|^2 = 0, 1, 2$ respectively, for the $D3Q19$ lattice used in this work, and $c_s = 1/\sqrt{3}$.

Lattice	c_s^2	Discrete velocities	Modulus	Weights (ω_i)
		$(0,0,0)$	0	2/9
$D3Q15$	1/3	$(\pm1,0,0)$,$(0,\pm1,0)$,$(0,0,\pm1)$	1	1/9
		$(\pm1,\pm1,\pm1)$	$\sqrt{3}$	1/72
		$(0,0,0)$	0	1/3
$D3Q19$	1/3	$(\pm1,0,0)$,$(0,\pm1,0)$,$(0,0,\pm1)$	1	1/18
		$(\pm1,\pm1,0)$, $(\pm1,0,\pm1)$, $(0,\pm1,\pm1)$	$\sqrt{2}$	1/36
		$(0,0,0)$	0	8/27
$D3Q27$	1/3	$(\pm1,0,0)$,$(0,\pm1,0)$,$(0,0,\pm1)$	1	2/27
		$(\pm1,\pm1,0)$, $(\pm1,0,\pm1)$, $(0,\pm1,\pm1)$	$\sqrt{2}$	1/54
		$(\pm1,\pm1,\pm1)$	$\sqrt{3}$	1/216

Table 4.1: Three $D3Qm$ LBGK lattices, with respective weights.

4.1.2 Parallelization strategies

The code implementation for the 3D lattice-Boltzmann solver was envisaged from the start as a fully parallel algorithm. There are several reasons for this. The main one was our initial expectation for the large size of even the smallest unstable periodic orbits. Even if, as we discuss later on, the 3D lattice mainly used in the present work is actually quite small (64^3), the period of the UPOs will be of the order of $\mathcal{O}(10^4)$. This will thus require the utilization of a supercomputer, or an equivalent aggregated set of resources [20], in order to implement the spacetime approach advocated in this work. Moreover, even disregarding the 4D module, thanks to this parallelization effort, HYPO4D can also be used to simulate large lattices (that could never fit on a single processor) and thus reach high Reynolds numbers. Last but not least, this project was carried out within the context of a research group with many years of expertise in the area of scientific grid computing [20, 178–184] whose activity has played a major role in pushing the boundaries of Grid computing in general.

In HYPO4D all communications between processors are handled using the MPI (Message Passing Interface) library [185]. This library allows for the creation of a specific topology (i.e., a virtual grid that can have a Cartesian or graph topology) of *processes* so that communications between these processes can occur in an unambiguous fashion. In this framework, an MPI process will be the smallest computing unit, and will be identified by a "rank", for the purpose of sending or receiving a message. This process can correspond to one core or to a multiple integer number of cores, since most currently operating clusters allow for the possibility to aggregate the memory of several cores into one single MPI process.

For the core module of HYPO4D, the LB fluid solver, a spatial domain decomposition was adopted, with each MPI process receiving a portion of the whole lattice. This strategy is particularly attractive for the lattice-Boltzmann method due to the locality of the algorithm[1]. The only quantities that must be exchanged between different processors are the values of the distribution function of particles which cross the border of the domain assigned to a particular processor, during the streaming step. Each process is attributed exactly the same subdomain size, so that there is an equal load balancing between machines (assuming they have the same characteristics). However, this can be easily altered if there is any need to do so. This subdomain decomposition is performed using the MPI function MPI_Cart_create, which sets up a Cartesian topology involving all the computing cores. This is created only once, at the beginning of each simulation.

In HYPO4D a "block" spatial decomposition was considered, which means that the dimensions of the subdomain are allowed to vary in all three directions. This ensures a lower surface to volume ratio. Other possibilities for regular sublattices are "pencil" decompositions where the size of the sublattice in one direction is held fixed and "slab" decompositions, where the size of of the sublattice is held fixed in two directions. Another important feature of the decomposition algorithm is that, of all the possible spatial decompositions, HYPO4D chooses the one in which the subdomain is most "cubelike", again in order to minimise the surface area to volume ratio. In practice this means choosing the magnitude of the dimensions of the subdomain to be as similar as possible.

The subdomain assigned to each processor is padded with a one-site deep extra layer ("halo") in all directions, which will receive the values of the particles streaming to another subdomain, at each time step. Thus there is no need to treat the boundary with any special care during the streaming step; the values of the halo are simply communicated to the respective neighbour processors after the streaming is over. This is carried out in a fixed order: each processor communicates first with its left and right neighbours, then with its up and down neighbours and, finally, with its neighbours in the z-axis. The particular order is arbitrary but, once it is chosen, must be followed strictly, for each communication stage of the algorithm. Before the next communication stage can begin, the values are reordered. Using this procedure, the corner values are transmitted indirectly, without need for any additional communications. Fig. 4.1 illustrates this process in two dimensions. The generalization to 3D is trivial and has the huge benefit that each process only need communicate (assuming the block decomposition implemented in HYPO4D) with eight nearest neighbours instead of twenty-six.

This particular halo-exchange algorithm incurs the penalty that all communications in a given direction must be fully completed before the next one can begin, otherwise the outcome would not be defined. This is implemented by using the function MPI_Waitall. However, it must be noted that this constraint is a local one and so is not a source for deadlock. HYPO4D makes exclusive use of non-blocking communications at the halo-exchange step, namely through the use of MPI functions MPI_Isend and MPI_Irecv, a factor which also contributes to prevent possible deadlock.

In order to solve the LBGK initial value problem, each processor stores two lattice arrays with the

[1]Lattice-Boltzmann can also accommodate non-local interactions, as is the case in amphiphilic fluids [118]

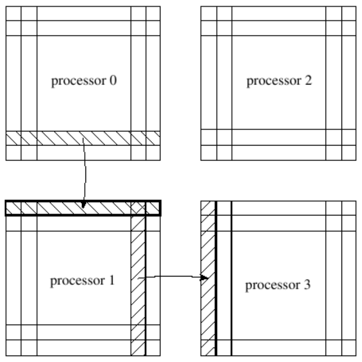


Figure 4.1: A lattice-Boltzmann halo-exchange in two dimensions. The corners are indirectly communicated. In this figure, note that the lower right shaded corner of process 0 is sent to process 3 in two communication steps.

same dimensions, latt_old and latt_new. The need for this can be seen from Eq. (3.16), since in the streaming step we will need the values of the distribution function at all lattice sites in the previous time step. HYPO4D makes extensive use of pointers, within the C programming syntax. In one iteration of the fluid solver, after collision and streaming are concluded and all relevant macroscopic quantities have been computed, the pointers for the two arrays are swapped, so that in the next time step the array latt_old will have the values determined previously for latt_new. This is of particular interest when we compare the evolution of quantities "on-the-fly", such as looking at the time-dependence of the velocity field, as we will discuss in section 4.4.

The parallel implementation of the LBGK collision and streaming constitute the core of HYPO4D. The algorithm uses non-blocking MPI communications and involves only eight nearest neighbours in its communication pattern. Our intention was to keep this computational core as simple as possible, i.e., not pursuing any aggressive optimizations, so as to deploy it as seamlessly as possible over many heterogeneous computational resources. Previous LB implementations were studied carefully, so as to adopt good implementation practices. The main ones were "LB3D" [118, 129, 186], a binary immiscible and ternary amphiphilic parallel fluid solver; "Vortonics" [37, 182] which locates and tracks vortex cores in turbulent flow; and "HemeLB" [176], that simulates fluid flow in complex geometries, and is applied to the study of human cerebral blood flow in the context of patient-specific diagnosis of aneurysms and arterio-venous malformations [187, 188].

HYPO4D allows for the specification of several parameters through an input file, without need of recompiling the code. These parameters include the dimensions of the lattice, the value of the LBGK relaxation time, the number of time steps to be simulated and the frequency (in LB time step units) at which macroscopic quantities, such as root-mean-squared (RMS) velocity are written to an output file.

Some other options, dealing with I/O, are discussed in more detail in section 4.3.3.

4.2 Numerical tests

In this section we discuss some of the numerical tests performed to assess the accuracy of the HYPO4D lattice-Boltzmann implementation. However, it must be pointed out that extensive testing of the code was performed at all stages of development, from simple consistency checks to the conservation of mass and momentum at the LB collision step. These checks can, nevertheless, be switched off by one of the parameters read from the input file, at the start of a given simulation, if the user is already sure of the numerical validity of a given parameter set. What follows are two widely used examples that provide a numerical benchmark of the application.

4.2.1 Square duct flow

The first serious numerical test conducted with HYPO4D was the simulation of square duct flow. The analytical velocity profile of fluid flow in a duct with rectangular cross section was reproduced, using two different boundary conditions and different sets of Reynolds numbers and relaxation times. In both cases the simulations were found to converge towards the known analytical profile.

For these tests, a uniform force was considered in order to drive the fluid flow. Another possible way to drive the flow is by the implementation of a pressure gradient [189]. This method was discarded for three main reasons:
i) a pressure gradient also implies a density gradient and it is our goal to study incompressible fluid flow;
ii) it has been observed that when using the pressure gradient method the maximum Reynolds number at which the simulation is still stable is smaller than that with external forcing [189];
iii) the force method has a much more straightforward implementation [35], and is thus closer to the simplicity which is such an appealing feature of the LBGK model.

The uniform force term can be introduced in the lattice-BGK equation in a simple way, by adding an extra term to the main equation, which becomes:

$$f_i(\mathbf{r}+\mathbf{c}_i, t+1) = f_i(\mathbf{r}, t) - \frac{1}{\tau}\left(f_i(\mathbf{r}, t) - f_i^{eq}(\rho, \mathbf{u})\right) + \rho F g_i, \tag{4.6}$$

with the expression for the force term [35, 190] being given by:

$$F = \frac{8\nu u_0}{L_y^2}, \tag{4.7}$$

where ν represents viscosity, u_0 the peak velocity and L_y the height of the square section perpendicular to the direction of flow.

The external force term must obey several constraints, so as to preserve the fundamental conservation laws obeyed, namely, it must inject zero mass into the fluid and ρF units of momentum per unit volume and time (assuming all particles have a mass value of $m = 1$). If we define the direction of the flow to be in the x-axis direction, this is equivalent to enforcing the following constraints:

$$\sum_i g_i = 0, \sum_i = g_i c_{iy,z} = 0, \sum_i = g_i c_{ix} = 1, \tag{4.8}$$

These in turn leave us with one free parameter [190], which can be used to fine-tune the intensity of the force term to the boundary conditions, while still having the velocity field obeying the Navier-Stokes equations. The second and third conditions state that we are imposing a unidirectional flow, in the x-axis direction.

Periodic boundary conditions were assumed at the inlet and outlet of the system, i. e., the planes $x = 0$ and $x = L_x - 1$ respectively. In the other remaining four planes which enclose the system, tangential to the direction of the flow, "no-slip" conditions were implemented in order to supply the system with the friction which characterizes simple Poiseuille flow. Two different methods were used to implement the no-slip condition. The first one was proposed by Maier, Bernard and Grunau [191], and implements the bounce-back condition [192] for particles with velocity components perpendicular to the no-slip wall, as well as enforcing zero tangential velocity at the physical wall. The other method, much simpler to implement, is mentioned by Zou and He, [189], and consists of imposing the equilibrium distribution with zero velocity at the boundaries of the flow. Formally, this can be written as:

$$f_i(\mathbf{r}_w, t) \equiv f_i^{eq}(\rho_0 = 1, \mathbf{u} = \mathbf{0}) = \omega_i, \tag{4.9}$$

where the subscript w indicates the wall lattice sites and Eq. (4.5) was used.

We considered a uniform initial state, with the values for the distribution function of each particle given by Eq. (4.5), assuming initial zero velocity and constant density, $\rho_0 = 1$. A 3D system with a rectangular cross section was considered for the validation of the algorithm. The quasi-parabolic velocity profile [30] is given by the series:

$$u(y, z) = \frac{16a^2}{\mu\pi^3}\left(-\frac{\partial P}{\partial x}\right) \sum_{i=1,3,5,\ldots}^{\infty} (-1)^{\frac{i-1}{2}} \left[1 - \frac{\cosh(\frac{i\pi z}{2a})}{\cosh(\frac{i\pi b}{2a})}\right] \frac{\cos(\frac{i\pi y}{2a})}{i^3}, \tag{4.10}$$

where P stands for pressure, $\mu \equiv \rho\nu$ is the dynamical viscosity, and $-a \leq y \leq a$ and $-b \leq z \leq b$.

For the simulations performed we imposed a convergence criteria, following previous work in this area [189, 191], in order to insure that the time iteration is carried on until a steady state in the velocity field is reached that obeys the condition:

$$\frac{\sum_{\mathbf{r}} \mid \mathbf{u}(\mathbf{r}, t+1) - \mathbf{u}(\mathbf{r}, t) \mid}{\sum_{\mathbf{r}} \mid \mathbf{u}(\mathbf{r}, t) \mid} \leq \delta, \tag{4.11}$$

where the sums are taken over all lattice sites. A value of $\delta = 10^{-10}$ was considered as the convergence criterion. In order to obey this some thousands of LB time steps may be required, depending on the parameters considered. At each time step, the previous values of the velocity field are being compared to the new ones, in every lattice site, and so two copies are kept in every iteration, with the respective pointers being swapped at the end of each iteration as described in the previous section.

The maximum relative error in velocity, ϵ, is defined as:

$$\epsilon \equiv max \frac{\sqrt{\sum_j (u_j^a - u_j)^2}}{u_0}, \tag{4.12}$$

with u_j^a being the analytical values at a given site for each velocity component, given by Eq. (4.10), u_j the measured components of the velocity and u_0 the peak velocity. The maximum is taken over the whole lattice.

For these benchmarks we considered different sets of values for the Reynolds number Re and the relaxation time τ. These are specified at the beginning of each simulation, being the only input parameters, along with the dimensions of the lattice. An analytical peak velocity is defined, by assuming it to be the relevant velocity scale and using the definition of the Reynolds number:

$$u_0 = \frac{Re}{L_y}\nu. \tag{4.13}$$

The system sizes and the values considered for the parameters Re and τ were the same ones that were used by Zou and He [189]. Different system sizes were simulated, in order to determine the order of convergence of the relative errors, using a least-squares fitting. The results obtained can be seen in Table 4.2 for both the boundary conditions described previously, labeled as I (velocity boundary condition from [191]) and II (Eq. (4.9)). For each set of values of Re and τ we indicate the corresponding analytical peak velocity, the *maximum* relative error obtained in the whole of the lattice and the coordinates where it occurred, after the convergence criterion, Eq. (4.11), has been satisfied.

Both boundary conditions show similar trends, although II has lower maximum relative errors in all the simulations reported here and slightly better convergence values. The level of optimization regarding the free parameter of Eqs. (4.8) that regulates force intensity was the same in both cases. The values obtained in these tests point to a first order convergence of the algorithm, for both boundary conditions. It is hinted in Maier *et al.* [191] that the velocity boundary condition might be of second order accuracy, although the tests reported there take into account the variation of the ratio L_z/L_y instead, and no expression for the force is given. Also in Zou *et al.* [189] a similar boundary condition is reported to be of second order accuracy, but for the case where pressure is specified at the inlet and outlet, instead of a body force driving the flow.

It must be noted that in these tests we are mainly measuring the accuracy of the boundary conditions implemented. The relative errors measured are quite small and decrease as we increase the lattice size. In order to better assess the validity of our LBGK implementation we tested a shear wave with fully-periodical boundary conditions, since these are also used for the case of isotropic homogeneous turbulence.

	L_x L_y, L_z	8 4	16 8	32 16	64 32	order of convergence
	Max. Relative Error I	3.372(-1)	1.760(-1)	1.057(-1)	7.531(-2)	7.224(-1)
	Coordinates I	0 3 2	0 5 4	0 6 8	0 11 16	
Re=5.0	Max. Relative Error II	2.810(-1)	1.345(-1)	7.650(-2)	5.380(-2)	7.969(-1)
τ= 0.8	Coordinates II	0 1 2	0 5 4	0 6 8	0 11 16	
	Analytical peak velocity	1.250(-1)	6.250(-2)	3.125(-2)	1.563(-2)	
	Max. Relative Error I	1.256(-1)	7.687(-2)	4.260(-2)	2.228(-2)	8.337(-1)
	Coordinates I	0 2 3	0 4 7	0 8 15	0 16 31	
Re=0.2	Max. Relative Error II	9.296(-2)	5.566(-2)	2.994(-2)	1.651(-2)	8.374(-1)
τ = 1.1	Coordinates II	0 2 3	0 4 7	0 8 15	0 21 16	
	Analytical peak velocity	1.000(-2)	5.000(-3)	2.500(-3)	1.250(-3)	
	Max. Relative Error I	4.870(-1)	2.600(-1)	1.461(-1)	9.618(-2)	7.852(-1)
	Coordinates I	0 1 2	0 1 4	0 6 8	0 11 16	
Re=10.0	Max. Relative Error II	4.472(-1)	2.345(-1)	1.229(-1)	7.779(-2)	8.502(-1)
τ = 0.6	Coordinates II	0 1 2	0 7 4	0 6 8	0 21 16	
	Analytical peak velocity	8.333(-2)	4.167(-2)	2.083(-2)	1.042(-2)	

Table 4.2: Maximum relative errors for 3D square duct flow. The symbol “I” identifies the velocity boundary condition and “II” the simpler equilibrium condition, given by Eq. (4.9). Also shown is the value of the coordinates where the maximum relative errors were detected, ranging from 0 to $L_i - 1$, with $i = x, y, z$. The last column shows the order of convergence for each boundary condition and combination of the values Re, τ, obtained using a least-squares fitting. In all cases the magnitude of the maximum relative error decreases as we increase the size of the lattice. The short-hand notation $(-n)$ is used for $\times 10^{-n}$.

4.2.2 Shear wave

A shear wave in a periodic domain can easily be shown to be a solution of the incompressible Navier-Stokes equations. This allows us to directly compare the velocity field obtained using our lattice-Boltzmann model with an exact analytical expression, without the need to worry about wall effects. The expression for the velocity field, $\mathbf{u}$, of the shear wave is

$$\mathbf{u}(\mathbf{r}, t) = A(t) cos\left(\frac{2\pi r_i}{L}\right)\mathbf{e}_j, \tag{4.14}$$

where $\mathbf{r}$ is the position, $\mathbf{e}_j$ is a versor, L is the size of the domain in the direction defined by $\mathbf{e}_j$, and $A(t)$ the amplitude of the wave in that direction. The condition for the shear wave is that $i \neq j$.

Inserting Eq. (4.14) into the NSE, Eq. (2.4), without a force term, we obtain the following equation

$$\frac{dA(t)}{dt} = -\nu \frac{4\pi^2}{L^2} A(t), \tag{4.15}$$

where ν is the viscosity of the fluid. Solving this straightforward differential equation, we find the expression for the amplitude to be

$$A(t) = A_0 \exp\left(-\frac{4\pi^2 \nu t}{L^2}\right), \tag{4.16}$$

with A_0 being the value of the amplitude at time $t = 0$. The combination of Eqs. (4.14) and (4.16) tells us that the wave is stationary in position and has an amplitude which decays exponentially with time.

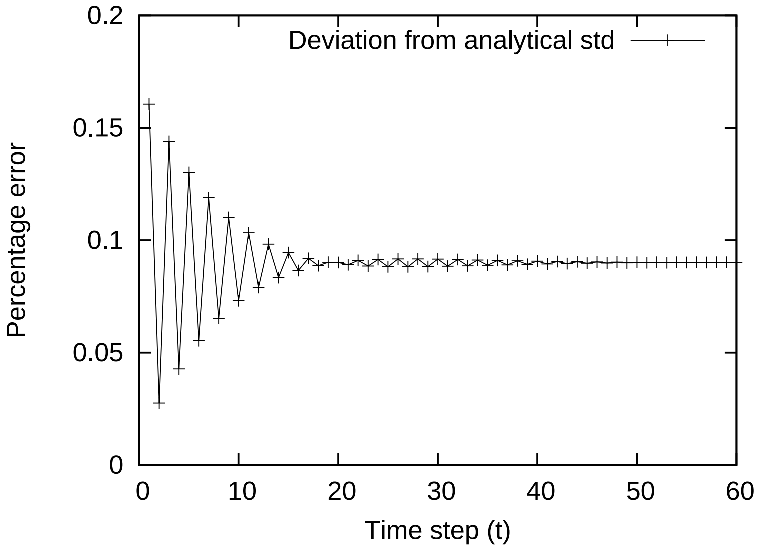


Figure 4.2: Percentage error of standard deviation for the shear wave, Eq. (4.19), in the first 60 LB time steps. Velocity field is simulated using HYPO4D, with parameters $A_0 = 0.1$, $L = 64$ (cubic lattice) and $\nu = 0.01$. Units in the x-axis are in LB time steps.

In order to simulate this in the lattice-Boltzmann framework all that is now required is to set up an initial velocity field obeying Eqs. (4.14) and (4.16), choose the values for the initial parameters, A_0, ν and L, and observe the decay of the wave. For the assessment of the numerical accuracy of our simulation we looked at the value for the standard deviation of the velocity field. We considered a wave with direction $e_j = \hat{k}$ and varying in the $r_i = x$ coordinate. The expression for the standard deviation will then be

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (u_{z_i} - \overline{u_z})^2}, \tag{4.17}$$

where $\overline{u_z}$ represents the velocity field averaged over all N lattice sites. From Eq. (4.14) it is easily seen that $\overline{u_z} = 0$. The expression for the standard deviation of the shear wave, σ_S, is then found to be

$$\sigma_S = \frac{A_0}{\sqrt{2}} \exp\left(-\frac{4\pi^2 \nu t}{L^2}\right). \tag{4.18}$$

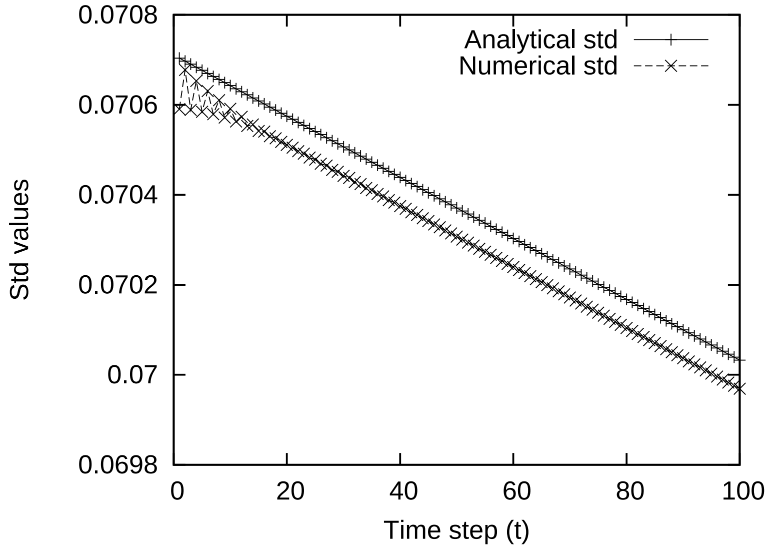


Figure 4.3: Analytical and measured shear wave standard deviations in the first 100 LB time steps. Velocity field is simulated using HYPO4D, with parameters $A_0 = 0.1$, $L = 64$ (cubic lattice) and $\nu = 0.01$. Units in the x-axis are in LB time steps. The LB clock seems to be running ahead of the analytical expression.

We compared the standard deviation in the velocity field obtained through HYPO4D, computed using Eq. (4.17), with the theoretical value given by Eq. (4.18). The input parameters were $A_0 = 0.1$, $L = 64$ (cubic lattice) and $\nu = 0.01$. A uniform mass density, with unit value, was considered over the whole lattice. We verified that the kinetic energy decays exponentially with time, as expected. In Figure 4.2 the error in percentage of the standard deviation of the velocity is shown for the first 60 iterations. The exact expression being plotted is

$$\frac{|\sigma_S - \sigma_m|}{\sigma_S} \times 100, \tag{4.19}$$

where the subscript “m” stands for “measured”. The main feature to note is the transient kinetic excitations present in the first 40 time steps, roughly speaking. After this transient the error, although quite small, actually increases with time step. The reason for this can be seen in Figure 4.3, where both the analytical and measured standard deviations are shown for the first 100 time steps. The values for both of these evolve in much the same way, but are clearly seen to be out of phase, with the LB “clock” running ahead. This discrepancy is almost certainly caused by the initial kinetic transient, visible in both Fig. 4.2 and 4.3. We found that the LB clock seems to be off by about 9.2 time steps, as compared

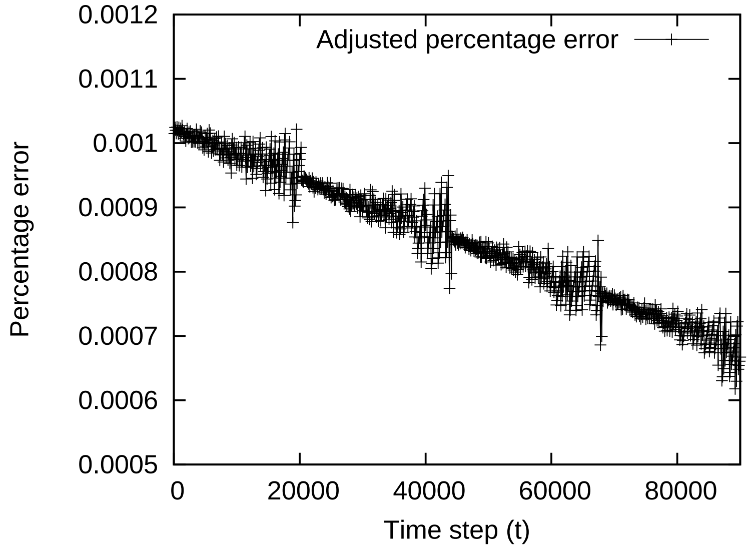


Figure 4.4: Adjusted percentage error of standard deviation for the shear wave in the first 9×10^4 LB time steps. All parameters as in the two previous figures, except that the analytical formula has been corrected by factors $t \to t + 9.2$ and $\nu = 1.0008 \times 10^{-2}$.

to the analytical expression. Besides this, the actual viscosity of the fluid being simulated is found to be not exactly 0.01 but closer to 0.010008. Taking these two factors into account, and replacing them in Eq. (4.16), the adjusted percentage error is now shown on Fig. 4.4, and is seen not only to be quite negligible, but to decrease (on average) with the number of iterations.

The analysis outlined above was repeated in the simulation of a $L = 128$ cubic lattice, with the same input parameters $A_0 = 0.1$ and $\nu = 0.01$. The errors now obtained, defined by Eq. (4.19), were reduced by a factor of ~ 10, in regard to the $L = 64$ simulation. Again we observed a strong initial kinetic transient, and the LB clock was found to be off by a factor of ~ 9.2 once more. Interestingly, the actual viscosity of the fluid was now found to be 0.010002, a correction 4 times smaller than for the $L = 64$ case. Fig. 4.5 shows the adjusted error after these two factors have been taken into account in the analytical expression. Comparison with Fig. 4.4 shows that the overall error has decreased on average by roughly one order of magnitude.

One further comment is required to finalize this discussion. At the beginning of the present section we mentioned that the shear wave considered here is a solution of the incompressible Navier-Stokes equations. However, as seen easily from the definition of the velocity field for the shear wave, Eq. (4.14) (where $i \neq j$), the convective term in the Navier-Stokes equations, $(\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u}$, will be null for this particular vector field. Thus, in the present numerical test it seems we are not explicitly simulating the *full* NSE, since the nonlinear term is not accounted for. Nevertheless, He and Luo [135], in their *a priori* derivation of the lattice Boltzmann equation, provide us with a strong argument to solve this apparent

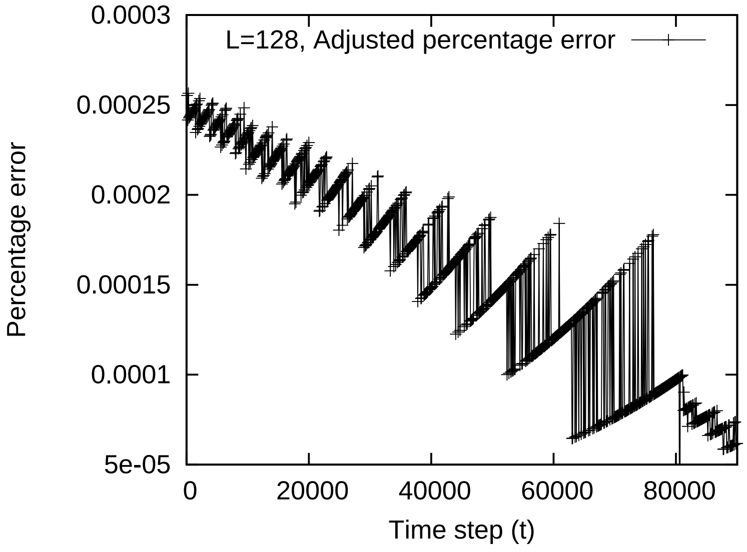


Figure 4.5: Adjusted percentage error of standard deviation for the shear wave in the first 9×10^4 LB time steps on a $L = 128$ cubic lattice. The analytical formula for the standard deviation has been corrected by factors $t \to t + 9.2$ and $\nu = 1.0002 \times 10^{-2}$. Units in the x-axis are in LB time steps.

shortcome. In their derivation, the LBE is obtained from the continuum Boltzmann BGK equation using the material derivative defined as

$$D_t \equiv \partial_t + \boldsymbol{\xi} \cdot \boldsymbol{\nabla}, \tag{4.20}$$

where $\boldsymbol{\xi}$ represents the microscopic velocity. In this way we see that the convective term is already taken into account implicitly, from the very deduction from first principles of the lattice-Boltzmann equation.[2] Thus we can safely say that the shear wave test, in a periodic domain, is an effective numerical validation of our algorithm for solving the incompressible Navier-Stokes equations.

4.3 Grid deployment and benchmarks

4.3.1 Introduction

The next logical step, after testing our lattice-Boltzmann implementation against analytical velocity profiles, was to see how its performance would scale by increasing the number of processors involved in a given simulation. This is a very important feature of the implementation, since we are aiming to simulate very large system sizes, due to the inclusion of the time component as well, in order to locate unstable periodic orbits in turbulent flow.

Amdahl's law [193] quantifies how much the gain in performance, S, can be when dividing a given

[2]We are indebted to Dr. Ian Halliday for calling our attention to this point.

task throughout a number of P processors. Formally this can be written as

$$S(P) = \frac{1}{f/P + (1-f)}, \tag{4.21}$$

where f represents the fraction of the workload that can be parallelized. Succi [35] states that in large-scale LB computations more than 99% of the workload resides within the collision step, which is perfectly parallel. This opens up good prospects for scalability up to very high core counts, as pertains to modern petascale computers.

The timing tests performed to investigate this aspect followed a hierarchy of resource sizes, ranging from a small local cluster to some of the largest currently operating supercomputers [121]. The code was also tested on a vector-architecture machine, the Cray X2 ("Black Widow") component of "HECToR" [194], part of the UK's National Supercomputing Service. In all cases, excellent scalability was found. In this section we describe only some of the main steps in this process, leading up to the deployment on Ranger at the Texas Advanced Computing Center (TACC) and Intrepid at Argonne National Laboratory (ANL), two resources that proved invaluable to this project.

All of the tests discussed in this section include only the fluid solver component of HYPO4D, since this lies at the core of the minimization algorithm discussed in Chapter 5. The architecture of each of the (main) machines where HYPO4D was deployed and tested is also briefly described. Some further aspects of this work are discussed in section B.3.

4.3.2 Timing benchmarks

The first stages of our (fully-parallel) code development were performed at our local cluster, dubbed "mavrino", which consists of a set of dual-core and quad-core AMD Opteron processors. After extensive testing and debugging we started migrating it to other platforms, beginning with the UK's NGS (National Grid Services) core resources [195], and performing scalability tests in order to demonstrate the efficiency of the code.

The first major milestone of this deployment procedure consisted of the timing tests performed on the HPCx machine [196], located at Daresbury Laboratory and led by a consortium formed by the University of Edinburgh, the Science and Technology Facilities Council and IBM. This service (no longer working since January 2010) was formed of an IBM Power5 1.5GHz cluster, consisting of 2560 cores available for scientific computation. These were coupled in logical partitions containing 16 processors each, with a super-scalar architecture, which allowed users to run parallel simulations using up to a maximum of 1024 cores. The system had a peak theoretical performance of 15.36 TeraFlops, a sustained 12.9 TeraFlops and is equipped with 5.12 TBytes of memory and 72 TBytes of disk. The operating system used was IBM's AIX.

The result of these benchmarks can be seen in Fig. 4.6 which shows the overall performance of the code when the number of cores used in the parallel simulation is increased. We use as a measure

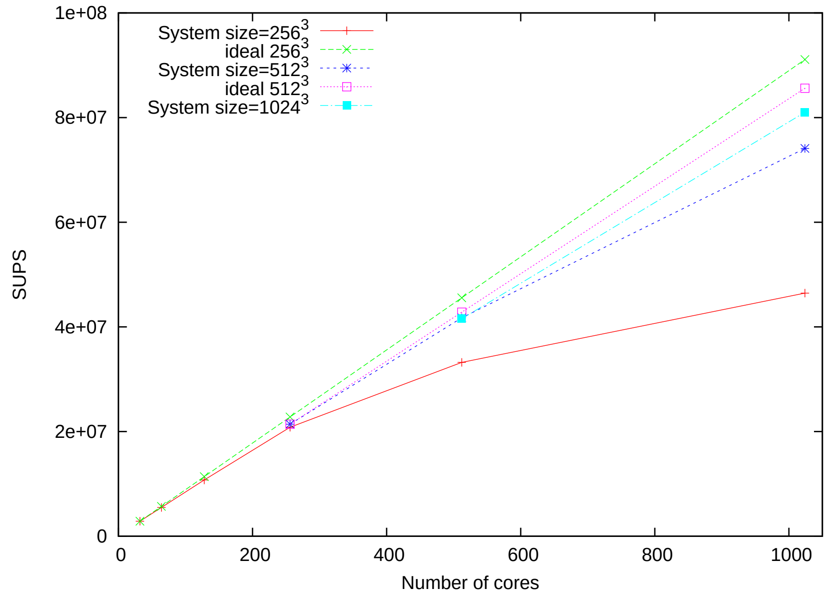


Figure 4.6: Number of total site updates per second (SUPS), given as a function of the number of cores used (ranging from 32 to 1024) for three cubic lattices with sides $L = 256, 512, 1024$, obtained on the IBM Power5 1.5GHz HPCx supercomputer. Also shown is the ideal scaling of the algorithm, in each case taking the value of SUPS obtained for the least number of processors considered as the base value.

of performance the number of site updates per second (SUPS), which, after a preliminary stage of initialisation of the simulation, reports how many lattice sites were updated per second, on average. Ideal parallel performance signifies the exact doubling of this value every time the number of cores used is doubled.

Fig. 4.6 clearly shows a linear trend for the performance of HYPO4D. Along with the measured SUPS values we also plot the ideal scaling SUPS values for the two smaller systems. This is not shown for the larger one since for that case the ideal value and the measured one are almost indistinguishable. When increasing the number of MPI ranks from 512 to 1024 we found a speed-up of 1.78 for the cubic lattice of side $L = 512$, and a speed-up of 1.95 for the cubic lattice of side $L = 1024$. This allowed for the code to be the recipient of a “Gold Star” award, the highest such award within the framework of the HPCx Capability Incentives policy, encouraging the development of efficient, scalable code.

Ranger is a 2.3 GHz AMD Opteron cluster located at TACC [197]. It has a theoretical peak of 0.579 PetaFlops and it achieved a maximal LINPACK [198] performance of 0.433 PetaFlops [121]. Its architecture consists of 82 racks, with each rack consisting of 4 chassis and each chassis consisting of 12 nodes. Each node is a Sun blade x6420 (four 16 bit AMD Opteron Quad-Core processors). The machine thus consists of $3,936$ nodes, i.e, $62,976$ cores, with a Linux operating system. It has a total aggregated memory of 126TB, which is of the utmost importance to us, since the relaxation procedure will be extremely memory intensive, as we will discuss in Chapter 5.

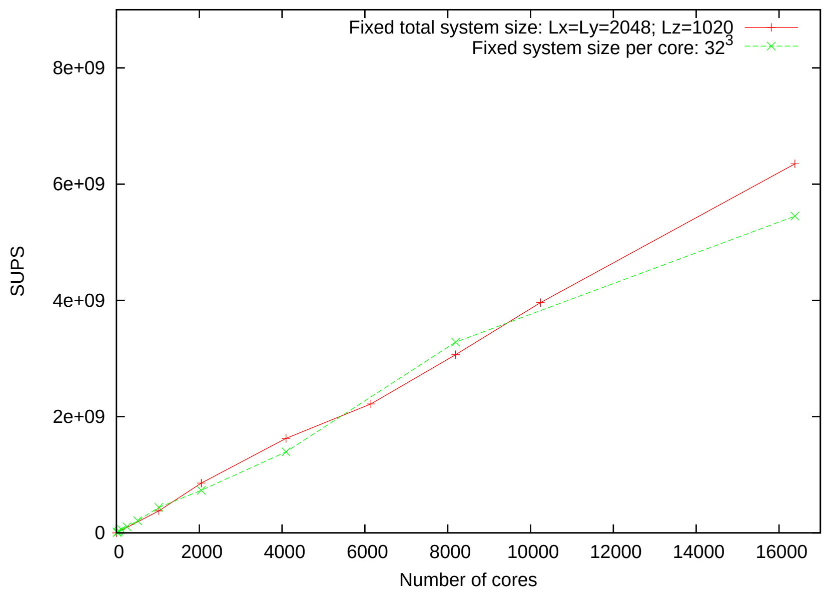


Figure 4.7: Number of total site updates per second (SUPS), given as a function of the number of cores used on Ranger. A comparison is made between "strong" (1024 to $16,384$ cores used) and "weak" (ranging from 16 to $16,384$ cores) scaling. Several of the SUPS values show a supralinear trend.

We had access to an "early-user" allocation in order to test HYPO4D on this machine, which allowed us to perform several benchmarks with very large system sizes and core counts. Fig. 4.7 shows a comparison between two different types of benchmark. In these scalability tests, and afterwards on the production runs, we used the mvapich-devel stack of the PGI 7.1 compiler suite, and the compilation flags -fast -tp barcelona-64, which target the "Barcelona" type chips used on Ranger. All tests consist of 200 lattice-Boltzmann time steps, with some of the values of several macroscopic variables and consistency tests being written to a file at every 50 time steps [3].

The full line describes timing tests performed keeping the overall size of the system fixed, so-called "strong scaling"; whereas the dotted line refers to timing tests performed keeping the portion of the system on each core fixed, so-called "weak scaling". In the latter the overall system size will increase proportionally to the number of cores being used.

Both tests have their own merits, depending on the kind of problem being studied. We found that for HYPO4D it is relatively easy to obtain linear scaling for the "strong" test scenario. This is partly due to computer memory hierarchy. If the total system size is kept fixed, by increasing the number of cores available a larger proportion of the system will be kept in cache memory, which is more readily accessible [4]. In HYPO4D this trend only starts to break down when the portion of lattice per core

[3]Unless otherwise stated all benchmarks presented will be assumed to follow this definition.

[4]This process can lead to supralinear values of performance being measured, as we observed in this work.

becomes extremely small and most of the time is now being spent in inter-processor communication, typically for values of the sub-lattice 16^3 or less, as we observed on Ranger.

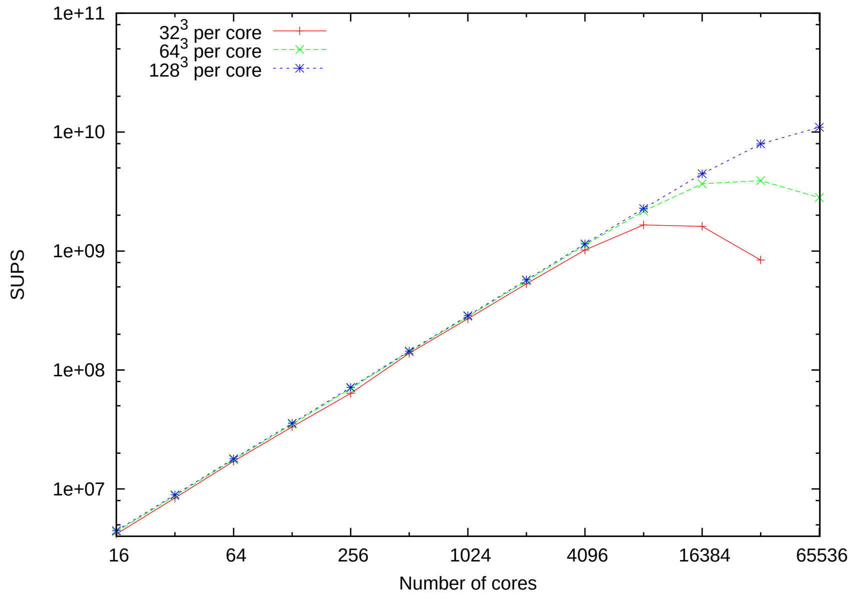


Figure 4.8: Number of total site updates per second (SUPS), given as a function of the number of cores used (ranging from 16 to $65,536$ obtained on the Blue Gene/P Intrepid machine at ANL. Optimisation level is $-O3$.

Even for the "weak scaling" case we also found very good scaling on Ranger, as shown in Fig. 4.7, up to $16,384$. As a result of this performance, HYPO4D was the recipient of a "5K Capability Computing Award" at the TeraGrid'08 conference in 2008 [199], conferred to codes scaling up to a number of at least 5000 cores.

Fig. 4.8 shows similar timing tests performed on the "Intrepid" Blue Gene/P IBM cluster at ANL [200]. This has a 0.557 PetaFlops theoretical peak and obtained a maximum LINPACK speed of 0.459 PetaFlops [121]. It consists of a total of $163,840$ PowerPC 450 type cores, each with a speed of 850 MHz (or 3.4 GigaFlops), having an aggregated memory of 80TB.

Fig. 4.8, where only weak scaling tests are shown, illustrates the scalability trend increasing with the size of the fixed load per core. Of course this must breakdown as the memory limits of any given architecture are reached. The best results can be seen to be for a 128^3 sub-lattice per core, which has a speed-up of 1.38 when going from 33K to 65K cores. Up to 33K the scaling is linear, with occasional supralinear values. The performance of HYPO4D on Intrepid in terms of SUPS per individual core is approximately two times less than on Ranger. This is consistent with the difference of processing speed between cores on each machine. It should be noted that both architectures are vastly different. Although Intrepid has more than the double of cores that Ranger has, its power consumption is only $1260K$ Watts whereas Ranger's has a value of $2000K$ Watts [121]. This is something that seems highly desirable, in an age where energy efficiency concerns play such a determinant role.

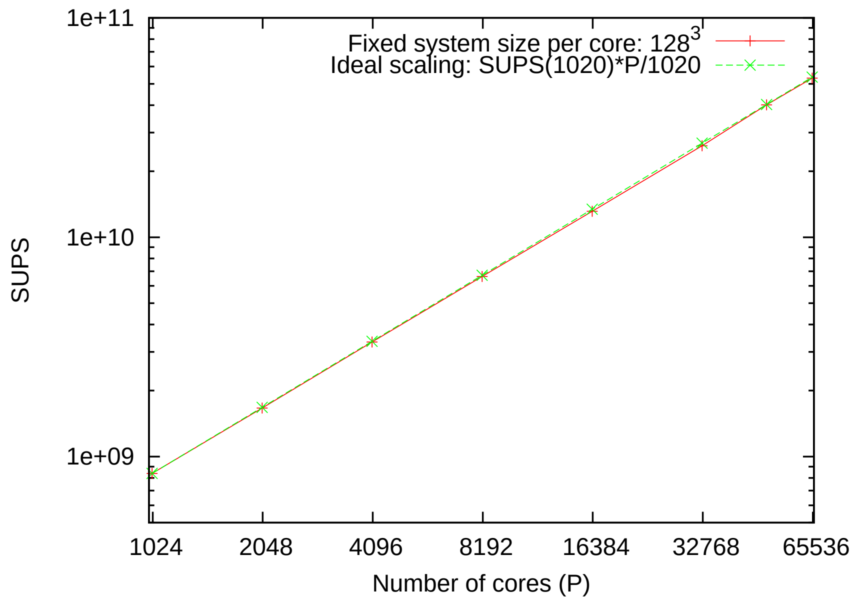


Figure 4.9: Number of total SUPS, given as a function of the number of cores used (ranging from 1024 to $65,536$ obtained on the Cray XT5 Kraken machine located at NICS. Optimisation level is $-O3$ This time a speed-up factor of $\sim$ 2.035 was measured, when going from 33K to 65K cores (see text for discussion of supralinear values).

For this range of core count, an even better scaling was obtained on the Cray XT5 Kraken machine located at the National Institute for Computational Sciences (NICS), in the USA. This machine is formed of 98928 Cray XT5-HE Opteron cores with 2.6 GHz and has a peak performance of 1.03 PetaFlops, with 832 PetaFlops of maximum LINPACK speed. It is part of a new generation of US petascale machines, which is pushing the boundaries of computational science. Fig. 4.9 shows the timing values obtained on this machine, using an $-O3$ optimization level.

Finally we should mention another computing resource, this time in the UK. At the time these benchmarks were performed HECToR [194] (short for High-End Computing Terascale Resource) was a Cray XT4 2.8 GHz cluster, with a LINPACK value of 55.0 TeraFlops and a total of $11,328$ computing cores (5664 AMD 2.8 GHz dual core Opteron). Fig. 4.10 shows the result of timing benchmarks performed on this machine. Again a value of 128^3 per core was used, since this gives the best performance for HYPO4D, as seen in Fig. 4.8. The results measured on HECToR signal one of the best HYPO4D performances, in terms of SUPS per core, along with the timings obtained on Kraken. If we focus on the SUPS value for 8192 cores, this is 0.532M for HECToR and only 0.400M for Ranger. The timing results obtained both on HECToR and on Kraken are indistinguishable from linear scaling, and in some cases supralinear. Furthermore, on Kraken HYPO4D had a sustained average of $\sim$ 0.8M SUPS per core, ranging from 1K to 65K cores.

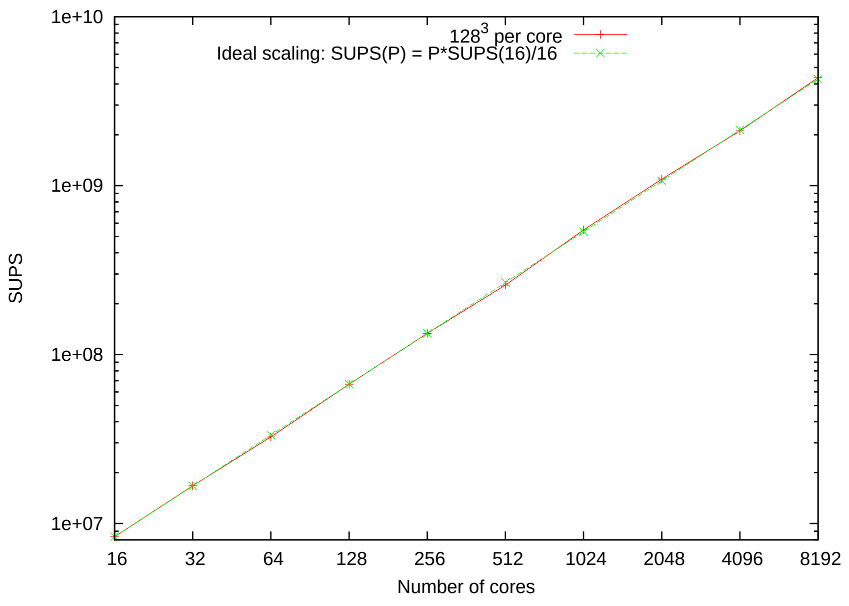


Figure 4.10: Timing tests on HECToR, XT4 component, for “weak” scaling. Optimisation level is $-O3$.

All of the timings discussed so far, have been performed on scalar architecture machines [201]. This seems to have been by far the most common architecture for the past years, in the field of supercomputing. However, for a long time, from the early 1970’s up to the 90’s, vector architectures had the lead for large scale scientific and technical computing. Briefly stated, a vector architecture is a CPU design wherein the instruction set includes operations that can perform mathematical manipulations on multiple data elements simultaneously. This is in contrast to a scalar processor, which handles one data element at a time, using multiple instructions [201].

The main reason for the loss of predominance of the vector design was the advance of CMOS (Complementary Metal Oxide Semiconductor) technology, which allowed for much higher chip packaging densities and thus a much lower cost for RISC (Reduced Instruction Set Computer) architectures. By capitalizing on the architectural simplicity of RISC and taking advantage of concurrency at the instruction level (i.e., multiple instructions being executed simultaneously by independent functional units), the label *superscalar* was coined, and this approach quickly achieved predominance over vector machines.

Recently, a vector processing component was added to HECToR, using a Cray X2 processor, developed under the code name “Black Widow”. This vector component consists of 112 Cray X2 vector processing units split into 28 vector compute nodes with 4 processing units per node. We were given early-user access to this component and thus the chance to perform timing benchmarks of HYPO4D on it. Some changes had to be made to the code, in order to run efficiently on this new architecture. These included the order in which arrays are declared and the handling of I/O which should not be

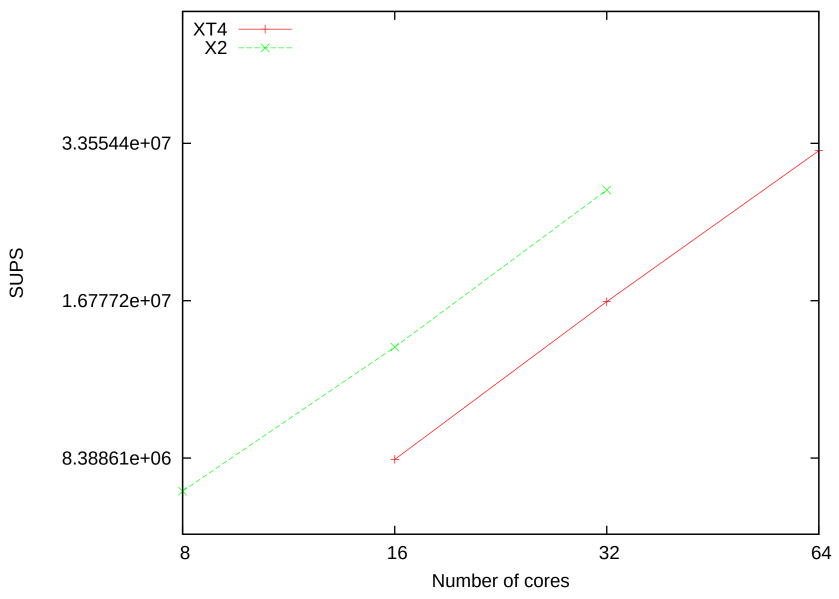


Figure 4.11: Comparison of HYPO4D performance on HECToR, between the XT4 (scalar) component and the X2 (vector) component.

inside loops in order not to break the vector architecture's pipelining of instructions [6]. The overall result of this effort consisted in a huge improvement in performance. Fig. 4.11 shows a comparison between HYPO4D performance on XT4 and X2. For the runs using 16 and 32 cores we saw an update in performance of 64%. This was possible due to the support given by NAG (Numerical Algorithms Group) [202] which is a partner in the HECToR consortium. A more detailed analysis then showed that the overall performance of HYPO4D running on a vector architecture could be ~ 4 times faster than on a scalar one. However, since HECToR's vector component is still relatively small, this avenue was not pursued further and we focused our efforts instead on much larger resources, namely Ranger and the Blue Gene/P machine.

To finalise this section, we should state that the benchmarks discussed here allow us to draw two main important conclusions. The first one is that HYPO4D is very well placed to efficiently make use of the huge computing resources (with memory being the main culprit) the spacetime approach requires. The second one is that the MPI programming paradigm still seems to be holding up to the order of magnitude of core counts of the new petascale machines, such as Ranger, Intrepid and Kraken. This is an important conclusion since several new machines, as well as upgrades to the existing ones, are on the pipeline, with the goal of obtaining several PetaFlops of *sustained* computation in the very near future, and there is every reason to believe that the MPI paradigm (as well as HYPO4D, as a specific implementation) will still be able to perform efficiently at that scale.

4.3.3 I/O and scalability

In HYPO4D all check-pointing is performed using the "XDR" (External Data Representation) protocol [203], a 1995 IETF (Internet Engineering Task Force) standard. This provides an architecture-independent data representation, allowing data to be efficiently transferred between heterogeneous systems. The files thus created are much smaller than their ASCII counterparts. Besides the distribution functions, the user can also chose to write only the velocity field. This implies reducing the storage requirements by a factor of ~ 6, since we have 19 distribution functions at each lattice site but only 3 velocity components. Several parameters on the input file control this aspect, again with no need to recompile the code.

A set of functions in the HYPO4D software package allow the user to check-point a given state of the system to disk and afterwards restart the simulation from that time step. This can be done in two different modes. The simplest mode has all MPI ranks sending its portion of the lattice to the process of rank 0 (in MPI parlance usually called "Master") who then concatenates the data and writes it to a single file. Then at the restart of the simulation, the "Master" process reads the data file and sends the respective portion of this data to the various processes arranged in a given Cartesian topology (which, by the way, may be different from the one that produced the checkpoint). This process is, however, not possible for systems too large to be kept in a single core. Therefore, in the second check-pointing mode each MPI rank will write a separate XDR file to disk, with the possibility to restart the simulation at a further date, by having each process read its respective file (in this mode, the Cartesian topology must be the same in both cases). Again, the choice between these two modes is made by the user at the start of each simulation.

One further comment is in order, and that is the impact of I/O on scalability as well as performance. For all of the benchmarks previously shown we did not keep check-points, i.e, no full configuration of the system was written to disk. We should expect this, something of obvious necessity in production runs, to somewhat degrade the performance of the code. Extensive tests on Ranger however, have shown this effect not to be as significant as might be expected.

Table 4.3 shows the impact of I/O on HYPO4D performance, measured on Ranger. The first set of timing data is the same one as was used for the soft scaling plot on Fig. 4.7. No check-pointing is performed and macroscopic average quantities are written to the standard output at every 50 time steps. In the second set of timing data, besides those macroscopic quantities still being written with the same frequency, a complete state of the system (i.e., all the LB distribution functions) is written to disk at every 50 time steps. Both these rates can be defined by a general user on an input file, at the beginning of each simulation, with no need for recompiling. For consistency, we used the "one file per core" I/O mode in all runs that performed check-pointing.

The speed-up values are defined as $SUPS(N)/SUPS(N/2)$ in both cases. Some supralinear val-

	Without checkpoints		With checkpoints		
Number of cores	SUPS	Speed-up	SUPS	Speed-up	Degradation
16	7.197876e+06		6.906664e+06		4.0%
32	1.403274e+07	1.9496	1.374042e+07	1.9894	2.0%
64	2.816367e+07	2.0070	2.734297e+07	1.9900	2.9%
128	5.441786e+07	1.9322	5.078074e+07	1.8572	6.7%
256	1.063874e+08	1.9550	1.035392e+08	2.0390	2.8%
512	2.100080e+08	1.9740	1.977605e+08	1.9100	5.8%

Table 4.3: HYPO4D timings with and without check-pointing. In all cases, a value of 32^3 lattice sites per core is considered. Speed-ups are given by $SUPS(N)/SUPS(N/2)$. The last column gives the (percentage) degradation factor in performance when check-pointing is included. All timings were measured on Ranger.

ues (i.e., larger than 2) can be seen, as discussed before. Analysis of the data displayed on Table 4.3 reveals that in all cases the SUPS value is lower when we include check-pointing, as should be expected. However, the impact of the degradation in performance, whose values are shown on the last column, is by no means large. Furthermore, the speed-up values for the runs that include check-pointing clearly show that linear scaling is still maintained.

We now turn our attention to the results obtained in the simulation of weakly turbulent flow.

4.4 Simulation of turbulent flow

4.4.1 Arnold-Beltrami-Childress flows

This section focuses on the work carried out using HYPO4D to simulate turbulent flow. All of the fluid flow simulations discussed here start from an initial state of constant (zero) velocity on all lattice sites, with a force being applied everywhere on the lattice at every time step. After a transient state, it is expected that the velocity field (in laminar flow) will closely follow the force field, since this is the only term that is driving the fluid which would otherwise be at rest. We consider periodic boundary conditions in all directions, thus excluding all wall effects. The force term can easily be included in the LBGK, in a variety of ways, by including extra terms in the two main equations, (3.16) and/or (4.5) (see Guo *et al.* [204] for a comprehensive discussion). In this work we followed the approach suggested by Shan and Chen [205] of including all of the relevant physics within the equilibrium distribution. This makes for very efficient code as well as high numerical accuracy. The exact expressions used can be found in the work of Lätt *et al.* [206].

The force field chosen to drive the flow is a vector field of ABC type, which is often used in dynamical systems studies [207–210]. Its name was first coined by Dombre *et al.* [207], referring to the mathematicians Arnold, Beltrami and Childress. The three-dimensional force field, $\mathbf{F}$, can be written as:

$$\mathbf{F} = A(\sin z + \cos y)\hat{\imath} + B(\sin x + \cos z)\hat{\jmath} + C(\sin y + \cos x)\hat{k}, \tag{4.22}$$

where $\hat{\imath}, \hat{\jmath}, \hat{k}$ represent orthogonal unit vectors and A, B, C are constants. The force term is introduced into the LBGK scheme in a way that does not artificially impart mass or momentum to the system [204]. Its overall effect thus consists of a stirring of the fluid.

The study of ABC flows actually touches on some fundamental issues in turbulence, and on the differences between the Eulerian and Lagrangian picture of fluid flows. Very briefly, in the Lagrangian picture of fluid flow the position of each fluid particle is tracked, being given by $\mathbf{x}(\mathbf{x}_0, t)$, where $\mathbf{x}_0$ may be taken as the position at some time t_0. In the Euler picture of fluid flow it is the velocity field instead that is the basic dynamic quantity being tracked. However, three-dimensional steady flows with a simple Eulerian representation can nevertheless have chaotic Lagrangian structure. This means that infinitesimally close fluid particles following the streamlines may separate exponentially in time, while remaining in a bounded domain, and thus the positions of fluid particles may become practically unpredictable for long periods of times. In 1984 Aref [211] showed that turbulent mixing of a passive scalar (i.e., a scalar quantity that is transported by the fluid motion, without influencing that motion) could occur in a simple, time-periodic, two-dimensional flow. This is also possible in three dimensions, e.g., for steady ABC flows.

This class of flows, was first studied by Arnold [212] as being an exact 3D steady-state solution to the Euler equations (Navier-Stokes without the viscous term). Moreover, for large values of the viscosity, this is the only stable solution for the NSE [208, 213]. As the viscosity decreases bifurcations arise which eventually lead to Eulerian turbulence. This process may be enhanced by the pre-existing Lagrangian turbulence. This seems to have been Arnold's original motivation for introducing these flows, as discussed by Dombre *et al.* [207].

In this work we chose to consider an equal value for all constants, $A = B = C \equiv N$, for simplicity. This regime was first studied by Childress, considering $N = 1$, who independently introduced it as a model for the kinematic dynamo effect [214]. The vector field, defined by Eq. (4.22), can easily be seen to obey the following equations:

$$\boldsymbol{\nabla} \cdot \mathbf{F} = 0, \ \boldsymbol{\nabla} \times \mathbf{F} = \mathbf{F}. \tag{4.23}$$

The second identity tells us that the force field obeys the property that characterizes Beltrami fields [86], namely:

$$\boldsymbol{\nabla} \times \mathbf{F} = \alpha \mathbf{F}, \tag{4.24}$$

with $\alpha = 1$.

Lastly we should mention that, and as pointed out by Zimmerman and Hewakandamby [210], ABC flows have several important properties which make it a mixing model with potential applications in industrial and environmental flows. Among these properties, the ABC flows have minimum energy dissipation, due to Eqs. (4.23) and (4.24). They also have maximum helicity, which is a quantity defined as

$$H = \int \mathbf{u} \cdot \boldsymbol{\omega} d^3 r, \tag{4.25}$$

where $\boldsymbol{\omega}$ is the vorticity, defined in Eq. (2.2). Since $\boldsymbol{\omega}$ and $\mathbf{u}$ are parallel for an ABC velocity field it then follows that H will assume higher values than for any other flows. This property, combined with minimum dissipation, in turn implies that these flows will have long-lived helical coherent structures [210].

4.4.2 Transition to time-dependent flow

In the simulation of ABC flows we studied the regime at which transition from laminar flow to a time-dependent velocity field occurs and sought to ascertain the value of the critical Reynolds number. Our main motivation for this was establish the lowest possible Reynolds number at which the flow is already in a turbulent (albeit weakly) regime. It is expected that flows at higher Reynolds number will have larger and more unstable UPOs, more tightly packed together, thus making them harder to locate [9].

Besides this requirement, two other criteria are of the utmost importance. The first one is numerical stability. Lattice-Boltzmann, as most numerical schemes at some point or another, is prone to numerical instabilities, as discussed in Chapter 3. These can occur for low viscosity values or very high force magnitudes and usually cause the distribution functions to assume negative (and thus nonphysical) values. In HYPO4D, this positivity constraint is by default checked at every time step and at every lattice site. For the sake of computational efficiency, this test can also be switched off, by one of the input parameters, if the user is already sure of the numerical stability of a given parameter set or range of parameters.

The other crucial criterion is the need to maintain a low Mach number (Ma) at all times, since the LBM approaches macroscopic nearly-incompressible flow with an accuracy that varies with Ma^2. In all the simulations discussed from now on $Ma \leq 0.2$, for which the maximum velocity, observed at any time step in the whole of the lattice, is used to check this criteria.

A suitable choice of input parameters (in particular the ABC force magnitude N and the relaxation time τ, which is equivalent to the viscosity, via Eq. (3.17), was reached after a protracted search for a set of values that would satisfy the above-mentioned conditions. Due to the memory requirements of the spacetime relaxation procedure we also chose to use the smallest possible (3D) lattice size that could still accurately describe weakly-turbulent behaviour. For this purpose a 64^3 lattice size was used to which, throughout the remainder of this work, all results will be referred to.

Fig. 4.12 shows the time-dependence of the velocity field for various values of the LBGK relaxation time τ and thus for various Reynolds numbers. Decreasing τ is equivalent to decreasing the kinematic viscosity ν and thus to increasing Re. In the limit of $\tau = 0.5$ we would have, from Eq. (3.17), zero viscosity and thus an infinite Reynolds number. The quantity being plotted (as a function of time in LB units) is defined as:

$$\frac{\sum_{\mathbf{r}} |\mathbf{u}(\mathbf{r}, t) - \mathbf{u}(\mathbf{r}, t-1)|}{\sum_{\mathbf{r}} |\mathbf{u}(\mathbf{r}, t-1)|}, \tag{4.26}$$

just as considered in Eq. (4.11) for the square-duct tests, and the coordinates $\mathbf{r}$ are summed over the entire lattice. In these simulations, a fixed ABC-force magnitude $N = 1/64^3$ was considered for all cases.

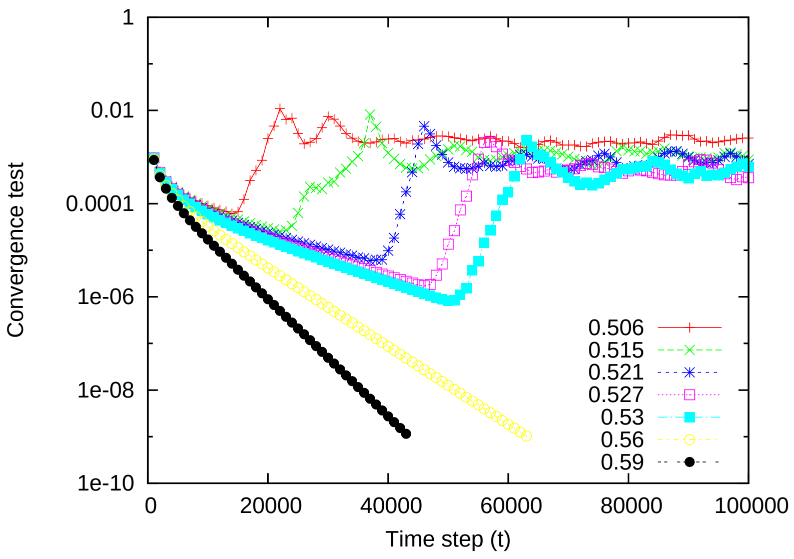


Figure 4.12: The convergence test, defined in Eq. (4.26), for the velocity field, taken between two consecutive LB time steps, up to time step 10^5, in LB units. We consider a cubic lattice with $L = 64$, constant ABC-force magnitude and several values of relaxation time, τ, shown in the plot. Units in the x-axis are in LB time steps. The sampling rate of the data has a value of 10^3.

For the values of $\tau = 0.59, 0.56$ (corresponding to $\nu = 0.03, 0.02$ respectively, by Eq. (3.17)) we see that the velocity field converges steadily towards a time-independent steady state, corresponding to laminar flow. (In Fig. 4.12 there is actually a cut-off at 10^{-9} for these two parameters, since the convergence test eventually reaches a value of zero, up to double precision.) It must be noted that since we start from an initial state of rest the velocities increase in the first tens of thousands of LBM iterations, which constitute a transient state. However, this is not shown in the figure, where only the rate of variation of the velocity field is plotted.

There is a marked transition in the behaviour of the convergence test when going from $\tau = 0.56$ to $\tau = 0.53$, i.e. $\nu = 0.02, 0.01$, respectively. For this last value, shortly after time step 5×10^4, the value of the convergence test stops decreasing and begins to increase. This marks the point at which the inertia of the fluid overcomes the energy dissipation due to the collisions. For $\nu = 0.02$, the maximum velocity is equal to 4.8482×10^{-2} in lattice units, corresponding to $Re = 155$. For $\nu = 0.01$ the flow settles into a time-dependent steady state, with RMS velocity (found by computing the average maximum modulus velocity in the lattice, where the averaging is over many thousands of time steps after transients have died out) equal to 5.7963×10^{-2} in lattice units, corresponding to $Re = 371$. As we decrease the viscosity this trend is maintained, as seen in Fig. 4.12, the main difference being that the transition to the time-dependent steady-state occurs progressively earlier, as expected.

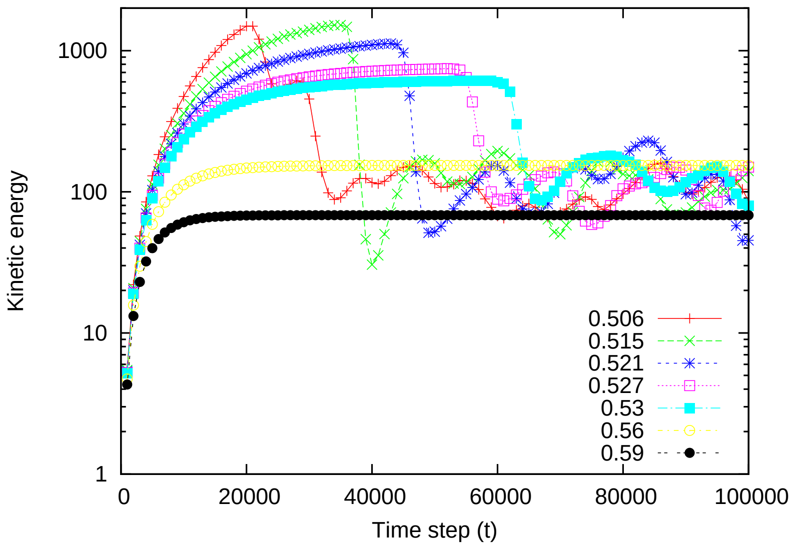


Figure 4.13: Total kinetic energy for the same values of τ as in Fig. 4.12. We consider in all cases a cubic lattice with $L = 64$ and constant ABC-force magnitude. Units in the x-axis are in LB time steps. The sampling rate has a value of 10^3.

The same behaviour can also be seen on Fig. 4.13 where the total kinetic energy is shown, for the same set of parameters as in Fig. 4.12. After an initial transient, the systems corresponding to $\tau = 0.56,\ 0.59$ settle into absolutely constant values of the total kinetic energy, corresponding to the behaviour exhibited in Fig. 4.12. As the relaxation time, and thus the kinematic viscosity, is decreased, we observe, after an initial transient, a transition to time-dependent values for the kinetic energy, already present for the case where $\tau = 0.53$. This transition then occurs progressively earlier in the simulation, as the value of τ is decreased.

Fig 4.14 exemplifies the cause of this behaviour somewhat better. The quantity being plotted, δ is defined as:

$$\delta \equiv max|\nabla \times \mathbf{u} - \mathbf{u}|, \tag{4.27}$$

and tells us how closely is the velocity field, $\mathbf{u}$, computed from the LB distribution functions using Eq. (3.13), following the ABC force driving the flow. We note that, from Eq. (4.24), after the usual initial transient, the difference between the velocity field and its curl should, in principle, be very small. This is indeed the case, for laminar flow, as shown in Fig. 4.14 for $\tau = 0.59$, but a different behaviour occurs for lower values of τ.

Fig 4.14 shows that for $\tau = 0.53$, around time step $t = 50,000$ there is an abrupt transition, with the value of δ increasing several order of magnitude in a relatively short time. It then settles into a time-dependent behaviour, as was the case in Figs. 4.12 and 4.13. For $\tau = 0.506$, the transition occurs even earlier in the simulation, and the values of δ are larger, thus showing there is even less correspondence between the driving ABC force and the measured velocity field. This demonstrates that as the inertia of the fluid becomes more important the relation between the driving force and the configuration of the flow is no longer linear [24].

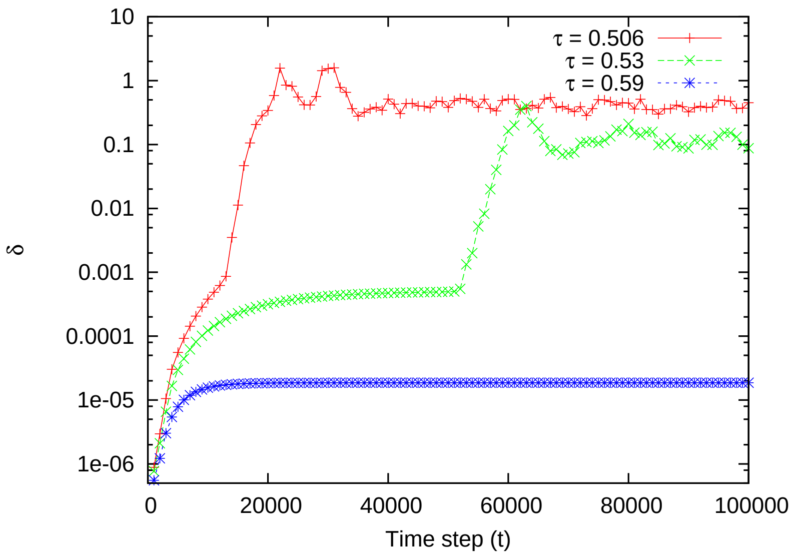


Figure 4.14: Maximum values for difference between ABC velocity field and its curl, defined in Eq. (4.27). As in previous plots, we consider a cubic lattice with $L = 64$ and constant ABC-force magnitude. The sampling rate has a value of 10^3. For the calculation of the first-order derivatives, a central finite-difference scheme, defined in Eq. (4.28), was used. Units in the x-axis are in LB time steps.

For the computation of the first derivatives involved in the curl (which was performed at a post-processing stage) we considered a five-point stencil finite central difference scheme [215]. In this approximation, the first derivatives are given by:

$$\frac{\partial u_i}{\partial x_k} \simeq \frac{-u_i(x_k + 2h) + 8u_i(x_k + h) - 8u_i(x_k - h) + u_i(x_k - 2h)}{2h}, \tag{4.28}$$

where $i, k = 1, 2, 3$. We considered $h = 1$ which means that, as can be seen from Eq. (4.28), at each point the derivative is computed, the information from the nearest four neighbours, in the direction of the derivative, was also taken into account. The accuracy of this approximation was checked by computing δ

for a “pure” ABC force term (without the lattice-Boltzmann fluid solver). This was performed on several cubic lattices of size L^3, with L ranging from 8 to 256, and δ was found to converge monotonically to zero as L increased.

4.4.3 Route to chaos in the ABC flow

After identifying the transition from a laminar to a time-dependent regime we looked more closely at the precise nature of this transition. As discussed in the previous section, the interval identified lies between LB viscosity $\nu = 0.02, 0.01$ (i.e., $\tau = 0.56, 0.53$) or Reynolds number $Re = 155, 371$, respectively. We note that these two factors do not scale in a trivial fashion, since the velocity field after the transition becomes highly nonlinear. However, this interval is still quite large and can reasonably be expected to show some interesting behaviour.

Figs. 4.15 and 4.16 show the total kinetic energy of the system in a section of the above-mentioned range for the value of the viscosity. From left to right and from top to bottom, the value of the LB viscosity decreases, between a maximum value of 0.0173 and a minimum of 0.0134, whereas the magnitude of the force term is kept constant and equal to $N = 1/64^3$. The convergence test of the velocity field also shows a similar trend, as would probably any other relevant global quantity. However this behaviour is more clearly seen when plotting the total kinetic energy, E of the system against time. This quantity is defined as

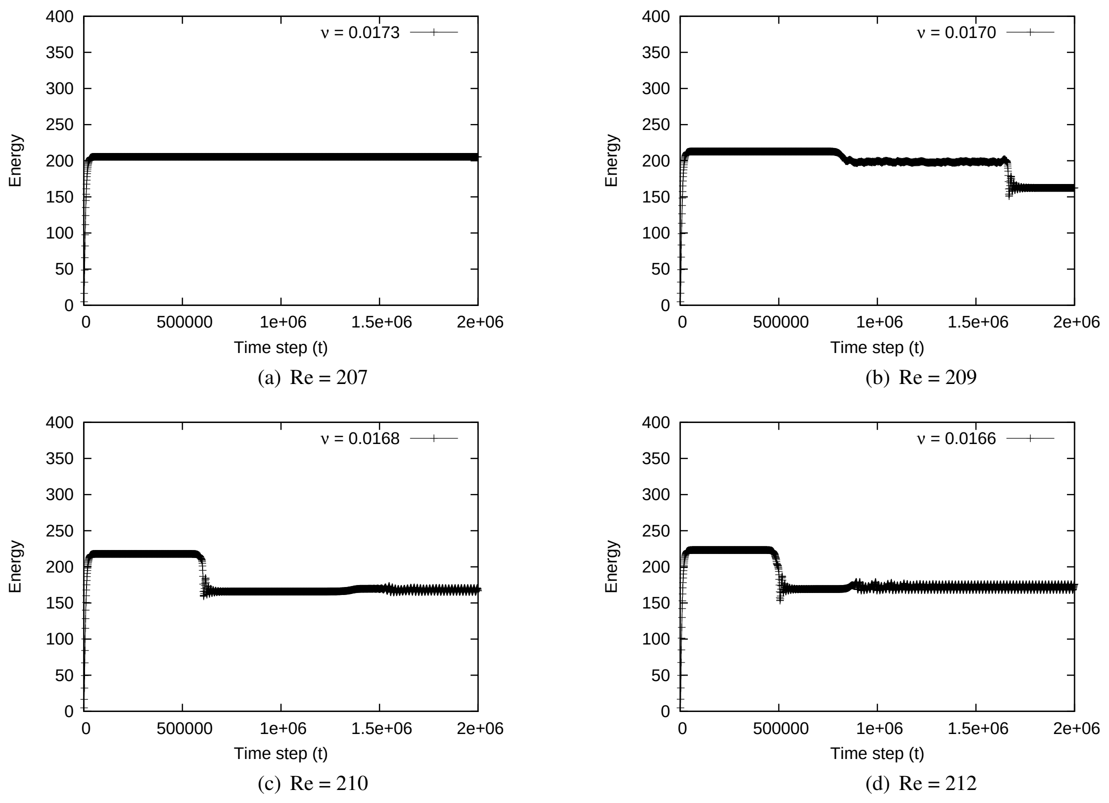
$$E(t) = \frac{1}{2}\sum_{\mathbf{r}} \left(\mathbf{u}(\mathbf{r},t)\right)^2 , \tag{4.29}$$

where the sum is taken over the whole lattice, and a unit mass is considered everywhere[5]. Since we do not consider any further interaction between the particles, this is also equal to the total energy of the system, representing the balance between the energy injected (through the continuous stirring of the ABC force) and the energy dissipated (through collisions). This is thus clearly an example of a forced dissipative system [7, 91]. The Reynolds numbers are computed by taking the average value of the maximum (modulus) velocity after the transient has died out. As was the case in Figs. 4.12 and 4.13 we observe that the duration of the transient becomes smaller as the Reynolds number increases.

Several interesting features can be seen in these energy plots. We begin with laminar behaviour, in Fig. 4.15 a). In Fig. 4.15 b) the system and goes through two meta-stable phases, but eventually settles once more into laminar flow, at constant energy, at $t \sim 1.7M$. This is no longer the case in Fig. 4.15 c) and from there on, the fluctuations in the total energy values become larger.

In Fig. 4.16 we have a quasi-periodic behaviour in the first plots, with very long values for the “quasi-period”. However, in all instances we also see a clear departure from periodicity. As viscosity decreases and the Reynolds number increases the value for this “quasi-period” decreases. Visual inspection shows

[5]As mentioned before, the Mach number was kept low, and the deviations found from perfect compressibility are of the order of numerical round-off.

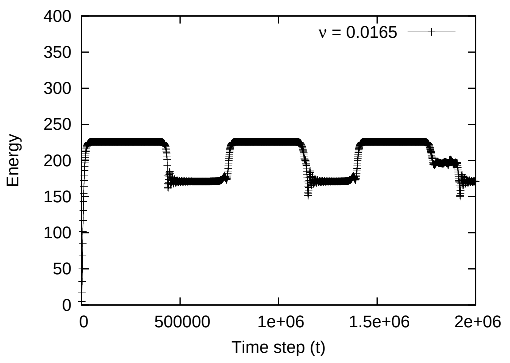


(a) Re = 207 (b) Re = 209

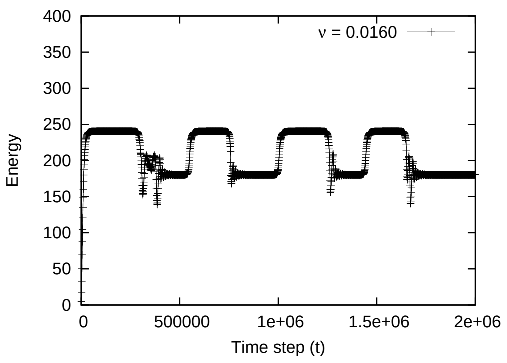
(c) Re = 210 (d) Re = 212

Figure 4.15: Total kinetic energy for different LB viscosities, displaying emergence of time-dependent behaviour. Viscosity, ν, decreases left to right and top to bottom whereas the force magnitude is kept constant. The laminar flow, observed in the first plot, becomes increasingly more unstable as the viscosity decreases. These instabilities also appear increasingly earlier in the simulation, as ν decreases. For $\nu = 0.0170$ the flow still converges to a constant kinetic energy (after a very long transient), but this is no longer the case for subsequent lower values of ν. Reynolds numbers, ranging from 207 to 212, are shown below each plot. Units in the x-axis are in LB time steps.

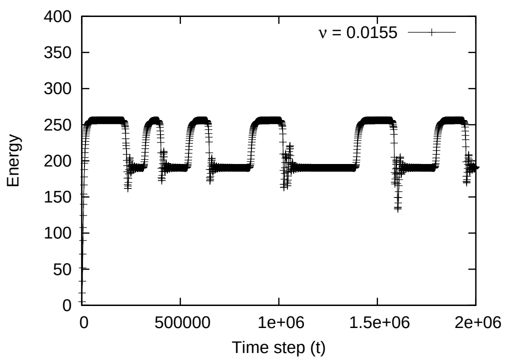
that the *average* total energy increases in all cases as ν decreases, as expected. The transitions between two quasi-stable regions (i.e., where the energy is almost constant) show increasingly larger and more unpredictable variations. The length of these quasi-stable regions decreases, converging to increasingly sharper distributions, as seen in subfigure f). After this, the variations become more and more pronounced and the behaviour increasingly chaotic. However, even in the last plot, for $\nu = 0.0134$, there are still some narrow quasi-stable regions, although they are difficult to see, at this scale of the picture. Following the terminology adopted by Eckmann in his 1981 review article [91], we can safely claim that the scenario described above signals a transition to turbulence through intermittency, with occasional windows of stability still re-appearing long after the aperiodic behaviour has settled in.



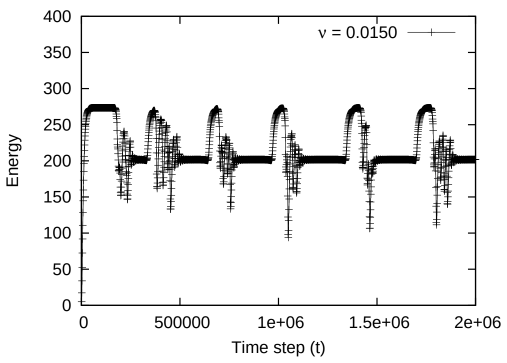
(a) Re = 213



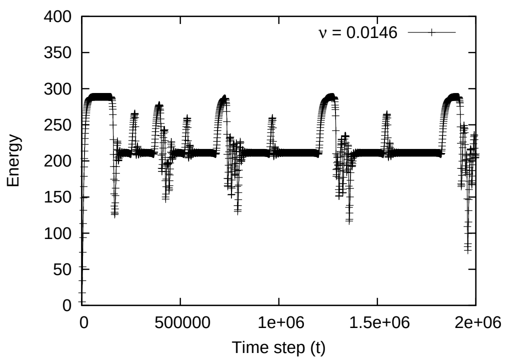
(b) Re = 219



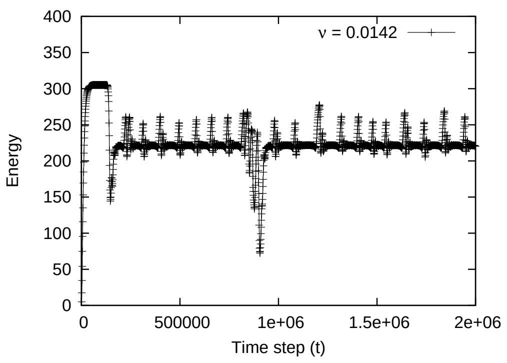
(c) Re = 230



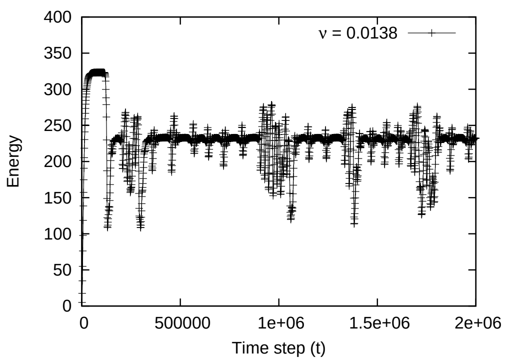
(d) Re = 234



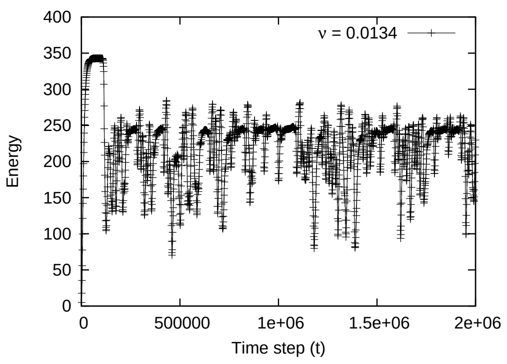
(e) Re = 246



(f) Re = 256



(g) Re = 266



(h) Re = 279

Figure 4.16: Same as in Fig. 4.15, this time displaying route to chaos. Viscosity decreases left to right and top to bottom whereas the force magnitude is kept constant. Near-periodicity, observed for the first values of ν, has a steadily decreasing period and the variation in the average kinetic energy also increases, until at $\nu = 0.0134$ we have a seemingly chaotic behaviour. Reynolds numbers, ranging from 213 to 279, are shown below each plot. Units in the x-axis are in LB time steps.

4.4.4 Energy cascade

A final test was made regarding the nature of our turbulent (ABC) flow simulations. We computed the energy spectrum of the flow and plotted it as a function of the modulus of the wave number, $\mathbf{k}$, obtained by calculating the Fourier transform of the velocity field, $\mathbf{u}$. For the NSE in a turbulent regime, a scaling exponent of $-5/3$ in the inertial range was predicted by Kolmogorov [34] and has been observed in many experimental and numerical studies [3]. This is actually one of the few mathematically rigorous analytic results, derived from first principles, known for the Navier-Stokes equations and is widely used as a test case. The exact range of application of the Kolmogorov power law has however been a subject of debate for a long time [23, 216]. A good introduction to this discussion can be found in the book by Frisch [3]. Very briefly put, Kolmogorov's power law rests on the assumption of *self-similarity* of the random velocity field at the the inertial-range scales. However, this assumption is easily broken in very many physical situations by the phenomenon of *intermittency*, by which very intense activity is displayed in very short scales of time [3].

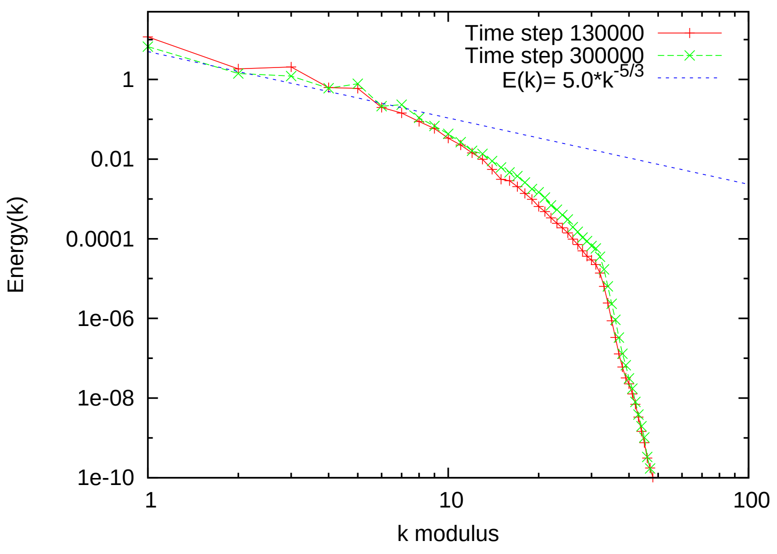


Figure 4.17: Energy as a function of the modulus of the wave number $\mathbf{k}$, Eq. (4.30), for the ABC flow, with a $-5/3$ Kolmogorov scaling also shown. The value of 5.0 (Kolmogorov constant) is arbitrary, since we are only interested here in the exponent of k. The system is a cubic lattice with $L = 64$, $\nu = 0.001$, and ABC force magnitude $N = 0.1/L^3$. The Reynolds number is 1204.

Fig. 4.17 shows the comparison between the Kolmogorov power law and data from one of our lattice-Boltzmann simulations. The quantity being plotted is

$$E(k,t) = \frac{1}{2N} \sum_{|\mathbf{k}| \equiv k} |\tilde{\mathbf{u}}(\mathbf{k},t)|^2, \tag{4.30}$$

where N is the number of lattice sites and $\tilde{\mathbf{u}}(\mathbf{k},t)$ is the Fourier transform of the velocity field:

$$\tilde{\mathbf{u}}(\mathbf{k},t) = \frac{1}{N}\sum_{\mathbf{r}} \mathbf{u}(\mathbf{r},t)e^{-i\mathbf{k}\cdot\mathbf{r}}, \tag{4.31}$$

where $\mathbf{r}$ is summed over all lattice sites.

The values of $E(k,t)$ are plotted for two time steps, both taken well beyond the transition from time-independent to time-dependent flow, as illustrated in Fig. 4.12. For time steps before that transition the inertial regime was nonexistent, the energy dropping abruptly from values of $k \sim 1$ (the length scale where energy is injected) to very low values of k, smaller than $\sim 10^{-6}$ (where energy dissipation takes over, due to the viscosity of the fluid), as seen in Fig. 4.18. This interpretation stems directly from Richardson's energy cascade picture [5] (but see also [23] for a further discussion of these ideas and their impact in modern turbulence research), summarized in his famous verse:

Big whirls have little whirls that feed on their velocity,
and little whirls have lesser whirls and so on to viscosity [5]

and which played a major historical role in the development of our understanding of turbulence [6].

Kolmogorov scaling is expected to occur for very large Reynolds numbers (fully developed turbulence), so we expect the agreement between the numerical simulations and the $-5/3$ power law to improve as we increase Re. Nevertheless, Fig. 4.17 is still a good indication that we are indeed in the region of weakly-turbulent behaviour, with $E(k)$ values on the intermediate k-values regions, which is completely non-existent in Fig 4.18. Due to the excellent scalability of the code we can confidently expect to reach much higher values of Re, possibly of the order of $\sim 10^5$, by simulating larger systems (without any sub-grid modelling). However, this falls somewhat beyond the scope of this work, since the identification of UPOs in such large systems would certainly be beyond the memory limitations of any current existing machine, as will be discussed in the next chapter.

To sum up, we have in the present Chapter described the particular LBGK model used in this work and our own parallel implementation. The bulk of this software package, entitled HYPO4D, is the fluid solver, which has been described at some length. Several timing results, obtained on a large variety of supercomputing platforms indicate that the code is well suited to efficiently utilize the kind of resources required by the (memory intensive) spacetime approach, with linear scaling obtained up to tens of thousands of cores. Numerical tests were reported, showing the accuracy of the fluid solver, with particular emphasis to shear wave decay, for which periodic boundary conditions are considered. These were also used for the simulation of the ABC flow, in which we analyzed the transition from laminar

[6]Uriel Frisch notes that Leonardo da Vinci in his celebrated notebooks seems to have already shown some understanding of this mechanism. One passage, written in his usual cryptic and reflected style of writing, reads (English translation):
"where the turbulence of water is generated
where the turbulence of water maintains for long
where the turbulence of water comes to rest" [3].

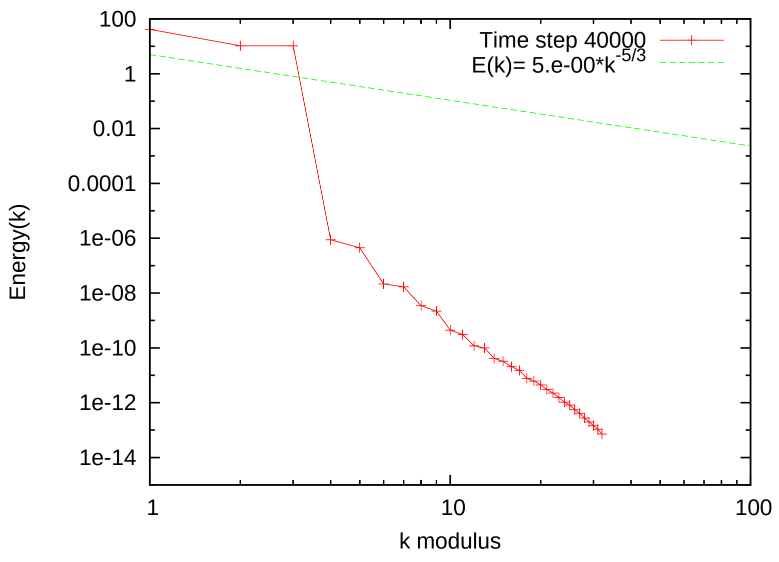


Figure 4.18: Same as in Fig. 4.17 but this time before transition to the weakly-turbulent regime occurs.

to weakly turbulent flow. In the next Chapter we shall describe the numerical implementation of the relaxation procedure and the identification of UPOs in ABC flow using the novel spacetime variational approach described in section 2.4.2.

CHAPTER 5

Numerical relaxation

THE methodology used for numerical relaxation towards an unstable periodic orbit is the main topic addressed in this chapter. We begin by describing the algorithm used for the identification of suitable starting points on which to apply this algorithm and then go on to describe the main numerical methods which were used for the minimization, or numerical relaxation, towards UPOs. Preliminary tests performed on laminar flow are discussed as well as computational aspects associated with each of the methods. This procedure requires vast amounts of memory and we discuss the strategies implemented to optimize the procedure. This is a problem that can only be addressed using petascale computing resources. The spacetime simulations thus performed are some of the largest ones ever attempted in this field, with a single job sometimes requiring in excess of a million CPU hours.

The HYPO4D software package has two modules that specifically address this problem, one for the identification of suitable candidate orbits and another for the spacetime numerical relaxation of these, using several different routines. Both these modules are fully parallel and are presented and described in detail. In the last sections we discuss the first results obtained with this methodology, for UPOs found in a $L = 64$ cubic lattice, with $Re = 371$, within weakly-turbulent ABC flow

As was the case for the $3D$ fluid solver described in the previous Chapter, the two modules of HYPO4D described in the present Chapter were also fully written and developed from scratch by the present author of this thesis. The UPOs that we present in this work all have periods between $15,000$ and $30,000$ and are described by a $L = 64$ cubic lattice. Due to the particular $19-$velocity LBGK model adopted in this work, and the use of double precision, this in turn means that one single copy of these orbits occupies in the order of one or more Terabytes of memory. The numerical relaxation procedures needed to identify them thus required the utilization of several millions of CPU hours on some of the largest existing computational resources in the world.

5.1 Methodology

In order to apply the numerical relaxation algorithm, using the novel variational principle described in Section 2.4.2, we begin by presenting the method used to identify suitable starting points and then describe the implementation of the spacetime gradient descent and conjugate gradient algorithms.

5.1.1 Definition of $\Delta(t, T)$

In this section we focus on the search for suitable minima on which to apply the numerical relaxation procedure discussed in Section 2.4.2. One very effective method to locate nearly periodic orbits, suggested by Auerbach *et al.* [8], consists in plotting the quantity

$$\Delta(t, T) = \parallel \mathbf{R}(t+T) - \mathbf{R}(t) \parallel \geq 0, \tag{5.1}$$

for very many values of t *versus* T, where $\Delta(t, 0) = 0$, and where $\parallel \cdot \parallel$ denotes a suitable choice of norm. If we can find a value $T > 0$ for which $\Delta(t, T)$ is at a local minimum and has very small magnitude then the equivalent orbit, ranging from t to $t + T$, will be nearly periodic. For this work [217] we used a simple Euclidean norm. Suitable generalizations of this can be implemented by considering a Sobolev norm (see [218], for a discussion of this approach, in the context of Ginsburg-Landau minimization), defined for generic vectors $\mathbf{P}$ and $\mathbf{Q}$ as:

$$\parallel \mathbf{P}, \mathbf{Q} \parallel_S \equiv \parallel \mathbf{P}, \mathbf{Q} \parallel_E + \gamma \parallel \boldsymbol{\nabla}\mathbf{P}, \boldsymbol{\nabla}\mathbf{Q} \parallel_E, \tag{5.2}$$

where the indices E and S stand for Euclidean and Sobolev, respectively and γ is a non-negative constant. It is assumed that $\mathbf{P}$ and $\mathbf{Q}$ have first-order derivatives and higher-order terms are ignored. With this definition, the case of $\gamma = 0$ corresponds to the standard Euclidean, L^2 norm. For the case $\gamma \neq 0$ the value of γ would have to be optimized, through global searches in the phase space spanned by the first-order derivatives of $\mathbf{P}$ and $\mathbf{Q}$. Similarly, higher-order derivatives could also be considered in Eq. (5.2), with more coefficients being introduced for each new term of higher-order.

For the case of the LB model, and considering an Euclidean norm, Eq. (5.1) becomes

$$\Delta(t, T) \equiv \sqrt{\sum_{\mathbf{r}} \sum_{i=1}^{Q} (f_i(\mathbf{r}, t+T) - f_i(\mathbf{r}, t))^2}, \tag{5.3}$$

where the coordinates $\mathbf{r}$ are summed over the entire spatial lattice. Taking into account higher order terms, in the definition of the norm, would in theory lead to more accurate minima, since we would be comparing not only the magnitude of two vectors in phase space but also their variation rates. However, the numerical difficulties involved in optimizing the extra parameters thus introduced would also be large and very likely not worth the extra computational effort. Due to the extremely high number of variables involved in our system (all the LB distribution functions) we believe that a simple Euclidean norm is already sufficient to identify suitable minima.

In order to find suitable starting points we plot the quantity defined in Eq. (5.3), with the lattice-Boltzmann distribution functions, $f_i(\mathbf{r}, t)$, being the relevant state variables. This bears some similarity with the convergence test for the velocity field, discussed in Section 4.4. The main difference is that we are no longer looking at a variation rate but rather measuring differences between states of the system (relatively) far apart in time. If a significant similarity is found between such time frames it indicates that we may be close to a spacetime (unstable) periodic orbit. More explicitly, Eq. (5.3) measures the "distance" between two vectors in a phase space with dimensionality $L^3 \times Q$, assuming a cubic lattice of dimensions L^3 and an LB model with Q discrete velocities.

5.1.2 Search for spacetime minima

Referring to our analysis of the transition to turbulent behaviour in Sections 4.4.2 and 4.4.3 we note that this takes place between the values of relaxation time $\tau = 0.56$ and $\tau = 0.53$ or, equivalently, viscosity values $\nu = 0.02$ and $\nu = 0.01$. This range of values was carefully examined, as shown in Figs. 4.15–4.16. Even though the kinetic energy shown in subfigure 4.16 h), for $\nu = 0.0134$, already shows a high degree of variation, it was found that a good deal of periodicity still remains, as can be seen from the $\Delta(t, T)$ plot shown in Fig. 5.1. For the purpose of identifying good candidate orbits for the relaxation procedure we thus found $\nu = 0.01$ ($\tau = 0.53$), was the desired (maximum) value for the viscosity. For higher values of ν, no deep minima could be found in $\Delta(t, T)$, given by Eq. (5.3).

Fig. 5.2 shows values of $\Delta(t, T)$ for such a system, with the ABC-force magnitude still being $N = 1/L^3$ and $\tau = 0.53$ for which (non-periodic) time-dependent behaviour is clearly observed. The horizontal axis in Fig. 5.2 shows the value of time, t, while the vertical axis shows the value of T, both in LB units. The colour code illustrates the magnitude of the "distance" in the phase space of the LB distribution functions between time slice $t + T$ and time slice t, as given by Eq. (5.3).

The plot shown is only a small part of a much more extensive time step comparison for this system, performed on Ranger, with values ranging from $t = 1.5 \times 10^5$ to 2.3×10^6, and $T = 5. \times 10^2$ to 1.5×10^5, in both cases separated by a sampling rate of 5×10^2, in LB time units. All the values considered for t occur well after the transition to weakly turbulent behaviour has occurred, as shown in Fig. 5.3, where we plot the rate of change of the velocity field. After transients have died out, this varies around an average value, with no well-defined periodicity. This methodology is used for all the results discussed from here on. The reason for focusing on this particular system is that the smallest UPOs of fluid flows with smaller Re should have a lower period and be more sparsely distributed throughout the state space, thus being easier to locate [9].

The search procedure illustrated by Fig. 5.2 is also fully parallel. The methodology followed consists in writing checkpoints (time steps) of the system state at a given sampling rate and then computing Eq. (5.3). Each MPI process reads a given number of time slices, and then compares each t time step with a number of $t + T$ time slices, up to a maximum value of T. The final values, $\Delta(t, T)$, are then

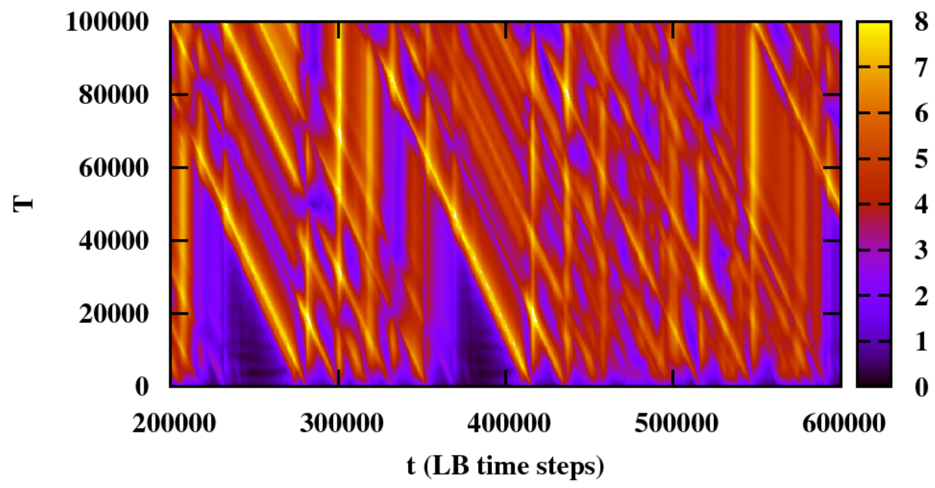


Figure 5.1: $\Delta(t, T)$, defined by Eq. (5.3), for a system with $L = 64$, $\nu = 0.0134$ (i.e., $\tau = 0.5402$) and ABC-force magnitude $1/L^3$. Darker regions indicate smaller values of $\Delta(t, T)$, which appear for many different values of T. Large periodic windows can be seen throughout. Units in both axis are in LB time steps.

communicated to a Master process (in MPI parlance) and written to the standard output. This is the only intra-processor communication required, thus making the algorithm embarrassingly parallel.

For the values shown in Fig. 5.2, a value of $\simeq 1.2 \times 10^6$ frame comparisons were performed, which required approximately half an hour on Ranger, using 4000 cores. We have followed the strategy of storing the time steps (with a given sampling rate – 500 in this case) so that other more refined algorithms may subsequently be applied, if such need arises.

The darker regions in Fig. 5.2, indicate greater similarity between different states of the time dependent system. There is a dark stripe for the smaller values of T, which results from the trivial similarity between time frames very close in time, as expected. The interesting thing to note is the appearance of darker regions for very many values of T, distinctly separated from the $T \to 0$ limit. These appear to be distributed with some regularity. We are especially interested in potential UPOs of smaller period, since these form the building blocks for larger UPOs (usually referred to as composite, or pseudo-orbits [9]). Fig. 5.4 therefore highlights one such region of interest, which can be seen in the lower left corner of Fig. 5.2. This shows two well defined minima, one centered at t slightly less than 1.8×10^5, and the other one at $t \simeq 2.1 \times 10^5$. Of these two, the first is of greater interest to us due to a broader dark region centered at a value of T smaller than that for the second minimum.

In order to evaluate how "deep" a given minimum is we compare its value of $\Delta(t, T)$ with all other

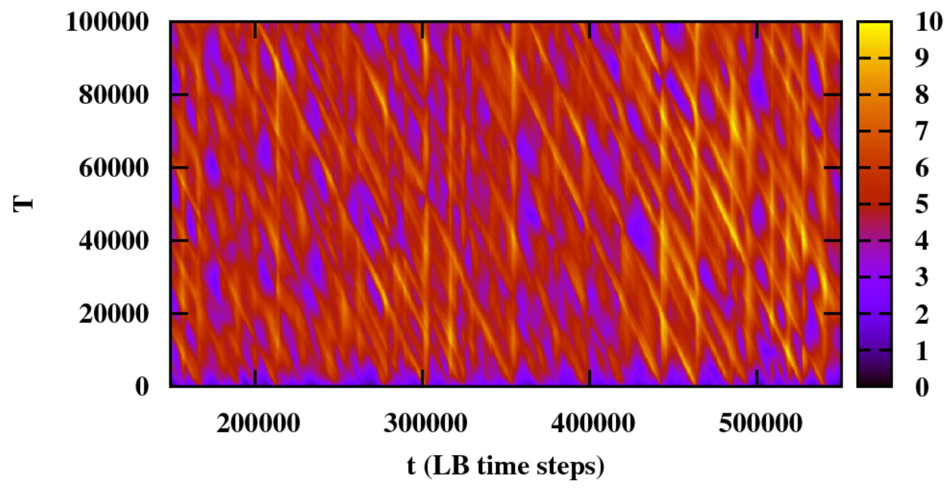


Figure 5.2: Detail of a larger $\Delta(t,T)$ plot computed on Ranger. Darker regions indicate smaller values of $\Delta(t,T)$, which appear for many different values of T. The dark stripe located at the origin of the vertical axis (very small values of T) indicates the trivial similarity of states very close in time. The system is a cubic lattice with $L = 64$, $\nu = 0.01$ (i.e., $\tau = 0.53$) and ABC-force magnitude $1/L^3$ corresponding to $Re = 371$. Units in both axis are in LB time steps.

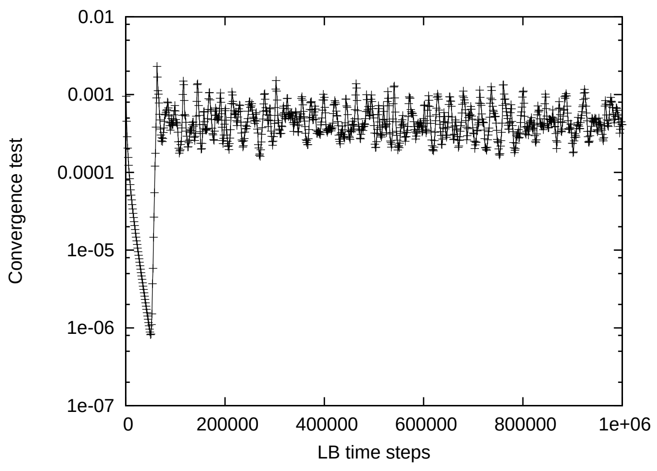


Figure 5.3: Velocity convergence test, Eq. (4.26), for the $\nu = 0.01$ and $Re = 371$ system, for 10^6 LB time steps.

such values for a given value of T. This can be seen in Fig. 5.5, for the value of $T = 27000$. For this value of T we found that the average value for $\Delta(t,T)$ was equal to 5.262909, with a standard deviation, σ, equal to 1.178410. Since $\Delta(t = 177K, T = 27K) = 2.190431$ (see Fig. 5.6) this is in

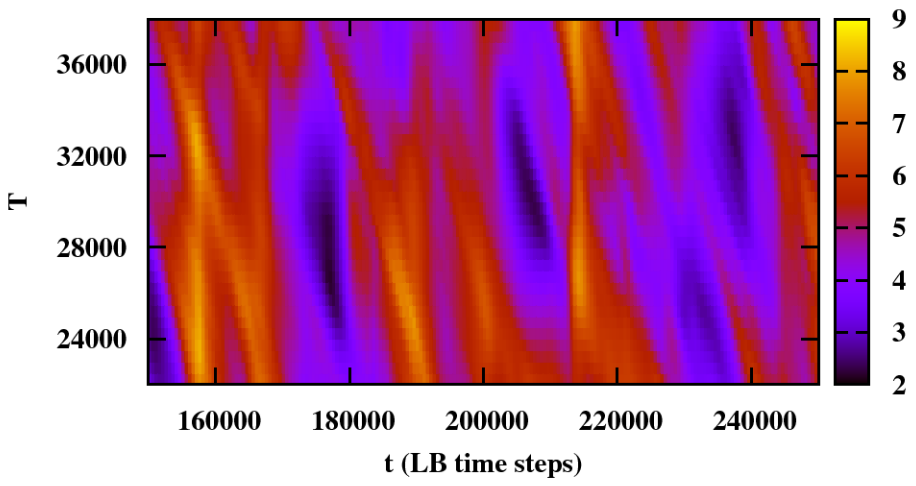


Figure 5.4: Detail from Fig. 5.2 showing a magnification of the area $t \in [1.5 \times 10^5, 2.5 \times 10^5]$ and $T \in [2.2 \times 10^4, 3.8 \times 10^4]$. The values of $\Delta(t, T)$ are given by the colour code, with darker regions indicating greater similarity between different time steps. Units in both axis are in LB time steps.

fact 3 standard deviations away from the average for this value of T.

Another similar comparison can be found within Figs. 5.7 and 5.8 for a value of $T = 24500$, for which we also found very good minima. For this case, the average value for $\Delta(t, T = 24500)$ is 5.251496, with a standard deviation of 1.160811. Since $\Delta(t = 637k, T = 24500) = 1.605092$ it is actually four standard deviations away from the average.

In Fig. 5.9, the process of zooming in on a minimum is extended and we find an optimal value of T for the first minimum in Fig. 5.4 up to an accuracy of 1 LB time step. In this case we kept the value of $t = 1.77 \times 10^5$ fixed and allowed only T to vary. The quantity being plotted in the vertical axis is again given by Eq. (5.3). The minimum of that quantity for the fixed value of t was found for $T = 26864$. Another such minimum is shown in Fig. 5.10, this time for fixed $t = 6.37 \times 10^5$ where an optimal value of $T = 24594$ was found. In both cases the values for fixed t were chosen after a rigorous global search through all the values computed for $\Delta(t, T)$.

The minima of $\Delta(t, T)$ form the starting point for the application of the 4D relaxation procedure, described in the next sections. Each one of these spacetime orbits must be loaded into the memory of a supercomputer. In order to save computing time we have found that it is easier to run an LB simulation starting from checkpoint t, with a number of time steps T, and write all time steps to disk. The relaxation procedure then begins by reading those time slices, one per computing core, as discussed in the next section. Using our $D3Q19$ LBGK model, and double precision arithmetic, each copy of an (64^3 spatial

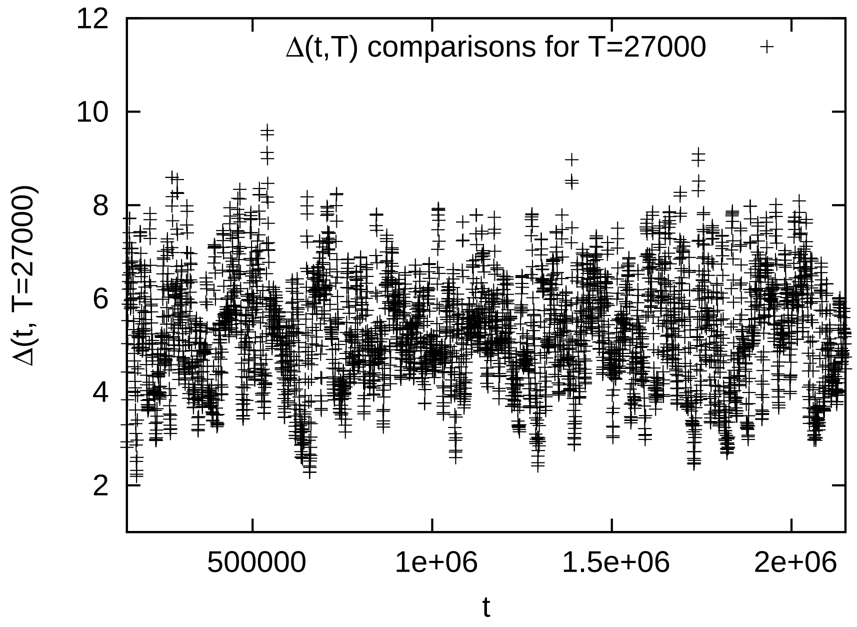


Figure 5.5: Plot of $\Delta(t, T = 27000)$ for the $Re = 371$ system. Notice the minimum near the origin of the horizontal axis, amplified in Fig. 5.6. The average value of $\Delta(t, T = 27000)$ is 5.262909, with a standard deviation equal to 1.178410. Units in the x-axis are in LB time steps and the sampling rate is 500.

lattice) orbit with period 30K time steps requires $\simeq 1.2$ TeraBytes of memory. It must be noted that these are nevertheless some of the smallest candidate orbits we could find through the search procedure outlined here. As we discuss in Section 5.1.3, there are challenging computational issues associated with the numerical relaxation of such large (spacetime) orbits.

5.1.3 Computational issues

Following the procedure outlined in Section 2.4.2, the main mathematical object which we will refer to throughout the remainder of this chapter is the functional $\mathcal{F}$ defined by:

$$\mathcal{F} \equiv \frac{1}{2} \sum_{t=0}^{T-1} \sum_{\mathbf{r}} \sum_{i=1}^{Q} |\phi_i(\mathbf{r}, t)|^2, \tag{5.4}$$

where $\phi_i(\mathbf{r}, t)$ is given by:

$$\phi_i(\mathbf{r}, t) \equiv f_i(\mathbf{r} + \mathbf{c}_i, t + 1) - f_i(\mathbf{r}, t) - \frac{1}{\tau}(f_i^{eq}(\rho, \boldsymbol{\pi}) - f_i(\mathbf{r}, t)), \tag{5.5}$$

following the LBGK model, defined in Eq. (3.16). The functional $\mathcal{F}$, defined by Eq. (5.4), is thus the LBGK equivalent of the functional defined in Eq. (2.25) for the continuum case. We immediately see

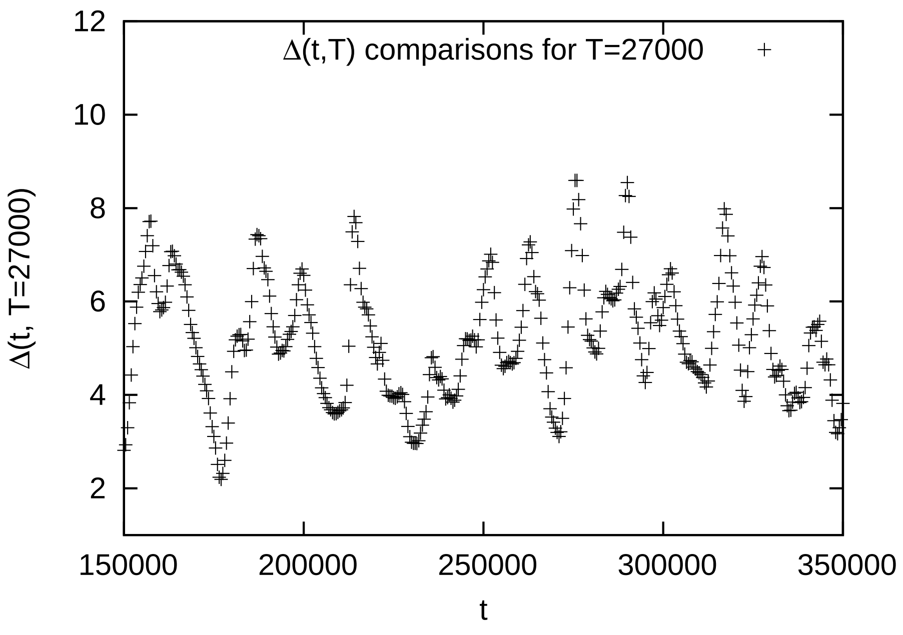


Figure 5.6: Magnification of Fig. 5.5; $\Delta(t, T = 27000)$ for $Re = 371$. $\Delta(t = 177K, T = 27000) = 2.190431$ and so lies 3σ away from average. Units in the x-axis are in LB time steps.

that $\mathcal{F}$ must, by definition, be 0 at all time steps where the LBGK equation is obeyed. However, the definition (5.4) only makes sense if we impose time-periodic boundary conditions in our system, i.e. $T - 1 + 1 \to 0$. Then we see that for the two time slices at the ends of our spacetime orbit we have:

$$\phi_i(\mathbf{r}, T-1) \equiv f_i(\mathbf{r}+\mathbf{c}_i, 0) - f_i(\mathbf{r}, T-1) - \frac{1}{\tau}(f_i^{eq}(\rho(\boldsymbol{r}, T-1), \boldsymbol{\pi}(\boldsymbol{r}, T-1)) - f_i(\mathbf{r}, T-1)) \neq 0, \quad (5.6)$$

where $\boldsymbol{\pi} \equiv \rho\mathbf{u}$ represents the momentum density. A value of $\mathcal{F} = 0$ (i.e., with $\phi_i(\mathbf{r}, t) = 0$ for all points on the spacetime orbit) will thus indicate a fully-periodic orbit, with period T. We shall often use "numerical relaxation", or "minimization", interchangeably throughout the text, meaning the procedure by which the value of $\mathcal{F}$ is minimized. A note should also be made regarding the time indices. From now on we shall refer often to the "UPO time indices", and label them from $t = 0$ to $t = T - 1$, as was the case in Eq. (5.4). These are still measured in LB time units. However, we must bear in mind that "$t = 0$" in "UPO time" has another (positive and usually high) value in terms of the LB clock, since we must insure that all transients in the initial value problem have died away.

The two main equations for the numerical relaxation procedure implemented in this work are the gradients of $\mathcal{F}$ with respect to the LBGK distribution functions and the inverse of the relaxation time. These are given in Appendix A, namely Eqs. (A.14) and (A.15). We have, however, chosen not to use Eq. (A.15) in this work, since it would raise difficult questions regarding the interpretation of the results, seeing that we would be varying the relaxation time and thus the viscosity of the fluid, which is also

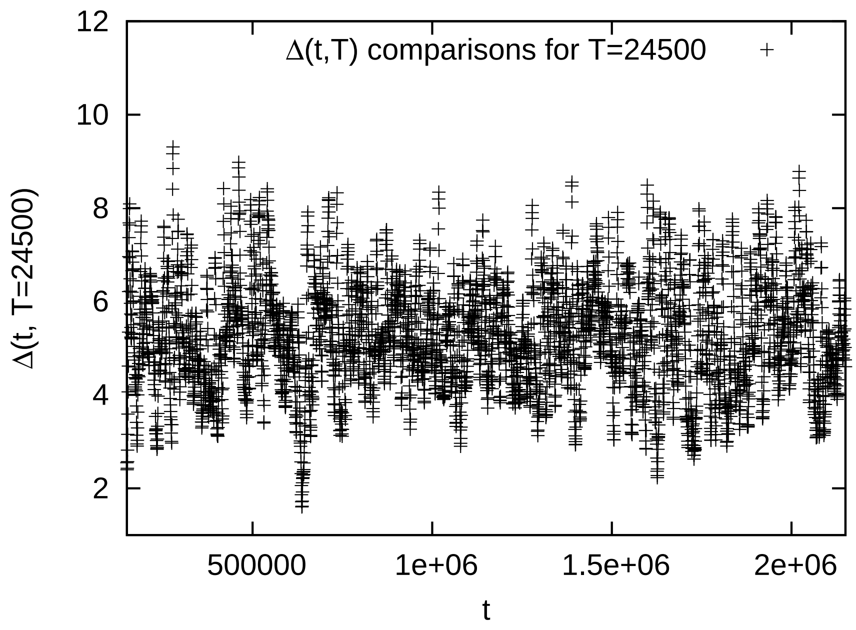


Figure 5.7: Plot of $\Delta(t, T = 24500)$ for $Re = 371$. Notice the sharp minimum near $t \sim 600K$, amplified in Fig. 5.8. The average value of $\Delta(t, T = 24500)$ is 5.251496, with $\sigma = 1.160811$. Units in the x-axis are in LB time steps and the sampling rate is 500.

used in the definition of the Reynolds number. Some numerical tests made with the numerical relaxation using Eq. (A.15) as well, show that there is very little difference in the convergence of the algorithm or the final results obtained.

There are two main difficulties associated with the numerical implementation of Eq (A.14). The first one concerns the sheer number of variables involved, which can be found by multiplying the $Q = 19$ distribution function components, by the 64^3 spatial lattice sites required, by the $T \sim 30$K time steps mentioned in the previous section. From this we see that something on the order of 10^{11} variables must be stored to represent the spacetime lattice, and thus the computational resources required are very large indeed. The second difficulty concerns the time indices in Eqs. (5.5) and subsequently (A.14). To compute the contribution to $\mathcal{F}$ at time step t, we require not only all the values of the spatial lattice at that time step but also at time $t + 1$, as seen from the first term on the right-hand side of Eq. (5.5), and at time $t - 1$, as seen from the first term on the right-hand side of Eq. (A.14).

Initially, this relaxation method was implemented by spatially decomposing the 4D lattice and keeping time local to each processing core. This meant that each core would have all of the $19 \times T$ distribution function components from at least one spatial lattice site. This was found to be not very effective computationally, in terms of the corresponding communication pattern. In addition, it placed a limit on the maximum possible value of T. This difficulty was circumvented by assigning instead all of the (64^3)

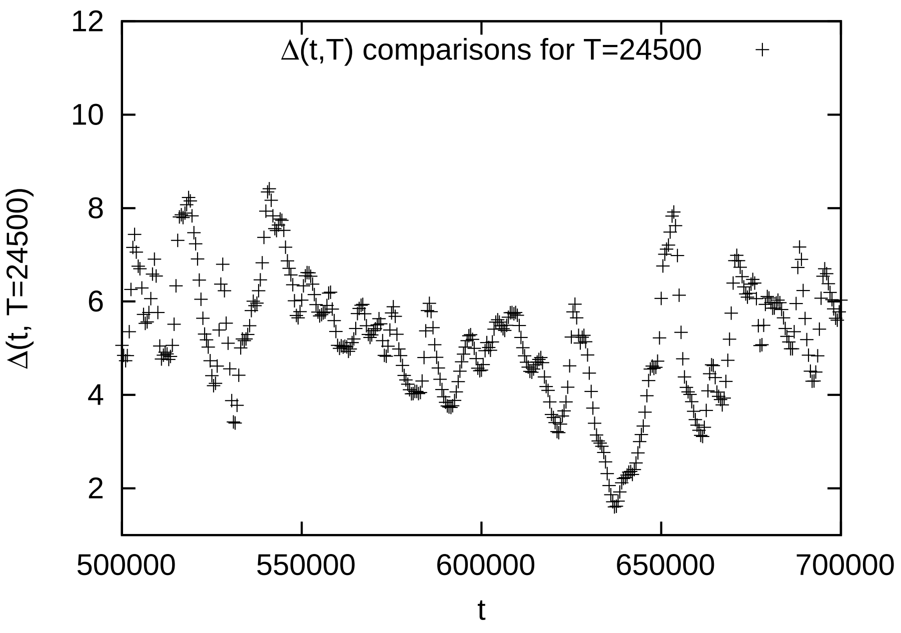


Figure 5.8: Magnification of Fig. 5.7; $\Delta(t, T = 24500)$ for $Re = 371$. $\Delta(t = 637K, T = 24500) = 1.605092$ and so lies 4σ away from average. Units in the x-axis are in LB time steps.

distribution function components $f_i(\mathbf{r}, t)$ at time t to a single MPI process. Whereas the fluid solver module of HYPO4D has a 3D Cartesian spatial topology, the resulting numerical relaxation module instead employs a 1D "ring" topology, with periodic (time) boundary conditions. Within this topology, each MPI process, corresponding to time step t, needs to exchange information with only its left $(t - 1)$ and right $(t + 1)$ neighbours at each iteration. This makes for a much more efficient communication pattern and led to a performance increase by a factor of ~ 5. Another advantage of this scheme is that no MPI communications are required to checkpoint the whole 4D orbit (as opposed to the case of the 3D Cartesian topology used in the fluid solver), since each MPI process has all the information of a given time slice t, and can thus write it to a file.

To clarify the previous discussion we now summarize the overall procedure. In schematic terms, the full algorithm for this work can now be written in the following way:

(i) Simulate weakly-turbulent flow and store many time steps at a given sampling rate;

(ii) Compute $\Delta(t, T)$, Eq. (5.3), in order to locate near-periodic orbits;

(iii) Apply a minimization procedure to the best candidate orbits found through *(ii)*, using Eq. (A.14) and a suitable numerical method to minimize $\mathcal{F}$.

The software package dubbed "HYPO4D", written to implement this program is thus formed from three main modules, each one corresponding to the three steps outlined above. The first module was

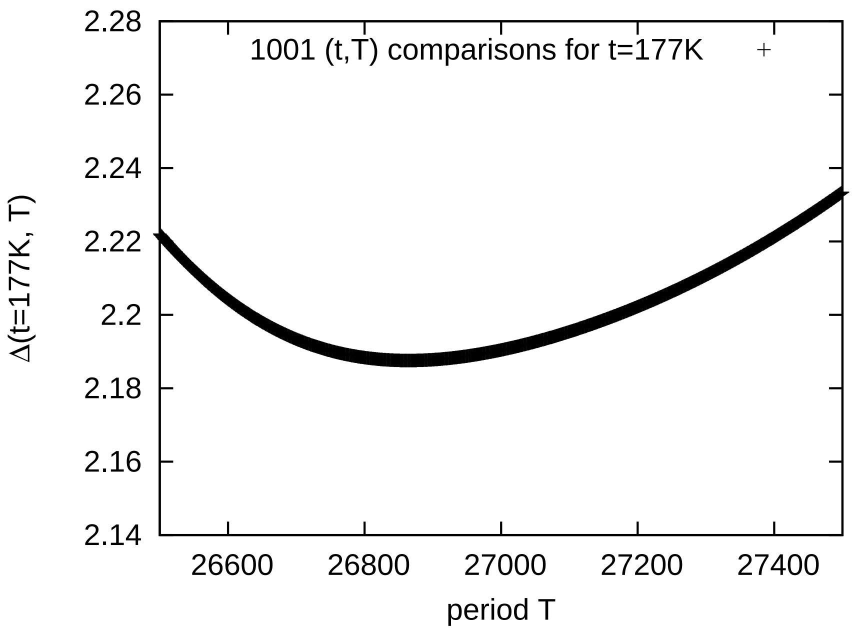


Figure 5.9: Amplification of the minima in Fig. 5.4 this time fixing $t = 177K$ and allowing only T to vary. The values of $\Delta(t, T)$ appear on the vertical axis. The sampling rate for T is now one time step. At $T = 26864$, $\Delta(t, T) = 2.1876$ is a minimum. The average $\Delta(t, T)$ value in this region of values of T is 5.2629. Units in the x-axis are in LB time steps, with the values indicating the UPO time indices.

described in great detail in Chapter 4 since it constitutes the core of the whole procedure, around which the two other modules are constructed. We now turn to the description of the two main numerical algorithms implemented to minimize $\mathcal{F}$.

5.1.4 Gradient Descent

The first method we implemented to minimize $\mathcal{F}$ via Eq. (A.14) was gradient descent (GD) sometimes also referred to as steepest descent [21, 219]. As the name implies, this is a simple minimization procedure, in which we compute the gradient of a given function and then minimize the function by taking a given step in the direction of that gradient, the direction in which the variation of the function is largest. In order to determine the exact size of the step we must perform a line search in the phase space determined by all the variables of our system.

In schematic terms, our overall algorithm can thus be written as:

(i) Fill 4D lattice with all the distribution functions, $f_i(\vec{r}, t)$, where $t = 0, T-1$, which constitutes the candidate orbit;

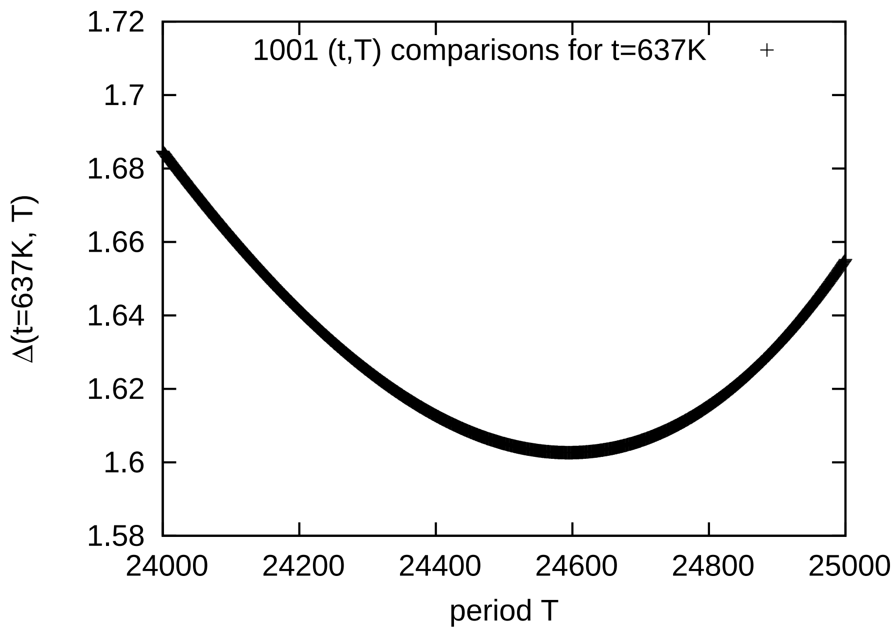


Figure 5.10: Another minimum, shown at fixed $t = 637K$ and allowing only T to vary. The values of $\Delta(t, T)$ appear on the vertical axis. The sampling rate for T is now one time step. For $T = 24594$, $\Delta(t, T) = 1.6026$ is a minimum. The average $\Delta(t, T)$ value in this region of values of T is 5.2521. Units in the x-axis are in LB time steps, with the values indicating the UPO time indices.

(ii) Implement LBGK stream and collide in this 4D lattice (with time-periodic boundary conditions);

(iii) Compute $\phi(\mathbf{r}, t)$, $\frac{\partial \mathcal{F}}{\partial f_i(\mathbf{r},t)}$ and the initial value of $\mathcal{F}$;

(iv) Apply GD until the variation of $\mathcal{F}$ falls below a certain threshold, δ;

(iv) GD algorithm:

$$f_i(\vec{r}, t) = f_i(\vec{r}, t) - \alpha * \frac{\partial \mathcal{F}}{\partial f_i(\vec{r}, t)}. \tag{5.7}$$

In order to simplify the discussion we shall consider the definitions:

$$x := f_i(\vec{r}, t) \text{ and } f' := \frac{\partial \mathcal{F}}{\partial f_i(\vec{r}, t)}, \tag{5.8}$$

For the determination of α, which gives the size of the step taken in the gradient direction, we used the golden-section line search procedure [21]. This requires that we first execute an appropriate bracketing of the minimum, i.e., we find three values $\alpha_a > \alpha_b > \alpha_c$ such that:

$$\mathcal{F}(x - \alpha_a f') > \mathcal{F}(x - \alpha_b f') < \mathcal{F}(x - \alpha_c f'). \tag{5.9}$$

After this initial step, the golden mean, the optimal quantity to perform a line search [21], is used to bisect the interval $[\alpha_a, \alpha_c]$ in order to find an optimal value for α.

In order to implement this minimization procedure three more arrays are required, one to store the values of $\phi_i(\mathbf{r}, t)$, another for the gradients f_i' and a further one to perform intermediate calculations. Since we already have an extra array associated with the LB streaming operation, this raises the number of arrays required to five. In the MPI "ring" topology discussed above, this is the number of (3D) arrays an MPI process stores in its memory. We note that, although the computations involved in computing the gradients given by Eqs. (A.14) are quite expensive, the line search procedure happens to be the part of the algorithm more computationally demanding. This is because, after the gradient has been computed, for each given value of α being tested in every iteration the functional $\mathcal{F}$ must be computed, a step which involves global communication.

5.1.5 Conjugate Gradient

The other minimization method implemented in the HYPO4D software package is conjugate gradient [21, 219]. This is a very well known routine, widely used in many types of optimization problems, which makes use of conjugate directions [219] to accelerate the procedure of locating a minimum or maximum of a given function. We used the nonlinear version of the conjugate gradient method, since the function we wish to minimize is quite complex and has a very large number of variables. Schematically, the algorithm implemented can be described in the following way, where (as defined in Eq. (5.8)) x represents all the variables in the system and we use the notation f' to denote the gradient of the functional $\mathcal{F}$:

(i) $d_0 = r_0 = -f'(x_0)$;

(ii) Find α_i, through golden-section search, that minimizes $\mathcal{F}(x_i + \alpha_i d_i)$

(iii) $x_{i+1} = x_i + \alpha_i d_i$

(iv) $r_{i+1} = -f'(x_{i+1})$

(v) $d_{i+1} = r_{i+1} + \beta_{i+1} d_i$, where β_{i+1} will be given by one of the following formulas:

$$\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} \text{ (Fletcher-Reeves) and } \beta_{i+1} = \max\{\frac{r_{i+1}^T (r_{i+1} - r_i)}{r_i^T r_i}, 0\} \text{ (Polak-Ribière),} \tag{5.10}$$

with r^T signifying the transpose of vector r.

From the previous definitions we see that not only are we now using the gradient information at each step, as in the gradient descent method, but also information from the conjugate directions, introduced in step (v) of the previous algorithm. The Polak-Ribière formula implies a restart if β attains a negative value. This restart means in practice that for that specific iteration $\beta = 0$ and the algorithm will only be performing gradient descent.

Both gradient descent and conjugate gradient have been implemented in a fully parallel fashion. Since the 4D lattice is spread out through a computational domain, all computations involve communication at some point, every time the values of $\mathcal{F}$ or its derivatives have to be computed. Furthermore, for the case of the conjugate gradient, a further three arrays are now required, in order to store information from the extra gradients being computed, as outlined above. This raises even more the already quite daunting memory requirements of the approach.

5.2 Laminar flow tests

Both the gradient descent and the conjugate gradient have been the subject of very extensive testing and optimization in a large variety of systems, before their application to the candidate orbits described in Section 5.1.2. Several strategies were used for this purpose. One of these included finding artificial minima (too small to be true UPOs) in systems with very few time steps and then attempting to minimize $\mathcal{F}$ in these systems. Another very useful testing methodology was the following: since the LB time evolution is relatively slow, if we use for our starting guess a 4D orbit with very few time steps this will automatically have a low value of $\mathcal{F}$. For this purpose it is even better to evaluate the algorithm before the transition to turbulent behaviour occurs, since the orbits are then more stable.

One such test can be seen in Fig. 5.11, for a system with only $T = 32$ time slices, using gradient descent. The system is again a 64^3 lattice, with $\tau = 0.53$ and ABC-force $N = 1/64^3$, but this time the time slices are chosen before the transition to turbulence, very close to the beginning of the initial value problem, with $t_0 = 185$. Fig. 5.11 illustrates well the typical behaviour of the minimization in these systems. We see that before we start the numerical relaxation procedure all of the discrepancy measured by $\mathcal{F}$ exists only between the first and the last time slice of the system. The quantity being plotted can be written as:

$$\mathcal{F}(t, t-1) \equiv \frac{1}{2} \sum_{\mathbf{r}} \sum_{i=1}^{Q} |\phi_i(\mathbf{r}, t)|^2. \tag{5.11}$$

This quantity differs from the one defined in Eq. (5.4) in that we have dropped the sum in the temporal index. As the minimization procedure occurs, this initial sharp discrepancy is gradually spread throughout the whole 4D lattice, with the total value of $\mathcal{F}$ also diminishing at each iteration. The latter property can be seen in Table 5.1. Another property of this minimization procedure which can be inferred from the values presented in Table 5.1 is that the convergence rate quickly decreases, after a good initial start. In the first 20 GD iterations $\mathcal{F}$ decreases by more than an order of magnitude whereas between iterations 100 and 200 $\mathcal{F}$ decreases only by a factor of ~ 1.3. This, however, is a well-established property regarding the gradient descent algorithm in general [219].

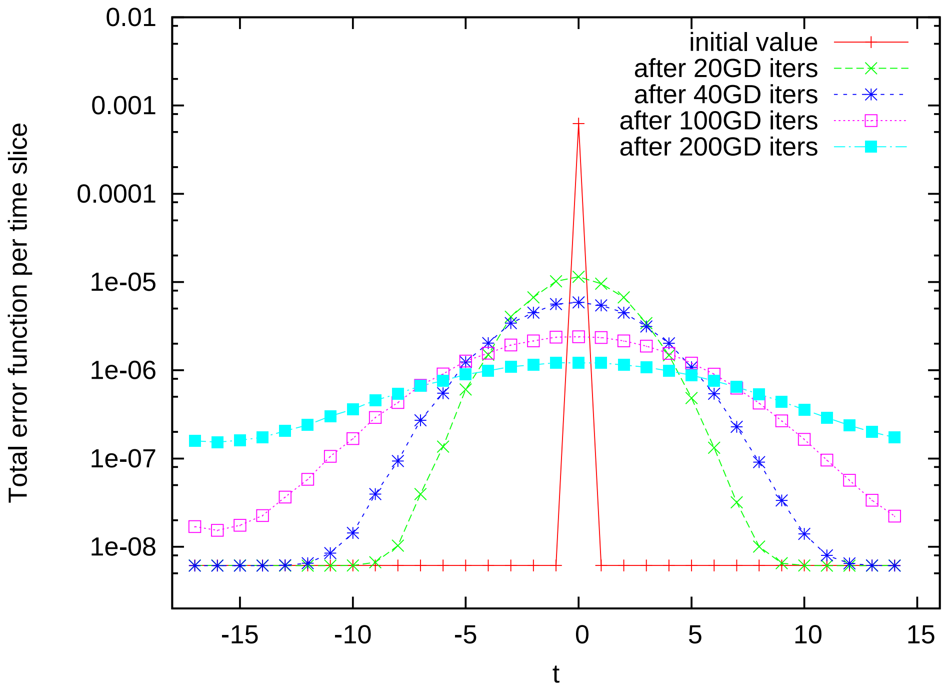


Figure 5.11: Distribution of total $\mathcal{F}$ values between two consecutive time slices, Eq. (5.11), in a test scenario with $T = 32$ (64^3 lattice with $N = 1/64^3$ and $\tau = 0.53$). As the number of gradient descent iterations increases, the initial discontinuity is spread out through the whole orbit and the total value of $\mathcal{F}$ (the integral of each curve) also decreases. The horizontal axis is centered around $t = 0$, in UPO time indices, with 0 corresponding to $\mathcal{F}(0, T-1)$. Total values of $\mathcal{F}$ are given in Table 5.1.

GD iteration	$\mathcal{F}$
0	6.256839e-04
20	5.661607e-05
40	4.077552e-05
100	2.615137e-05
200	1.966296e-05

Table 5.1: Values of $\mathcal{F}$ under GD minimization for the $T = 32$ test case shown in Fig. 5.11.

We tried to suppress this deficiency of the algorithm by using GD as a pre-conditioner and then reverting to conjugate gradient after a certain number of iterations. However, in spite of a major effort made in this aspect of the work, the conjugate gradient was always found to be prone to instability and never produced better results than gradient descent alone. Many strategies were attempted in order to produce a better performance of CG, which included using both the Fletcher-Reeves (FR) and the Polak-Ribière (PR) formulas, restarting CG periodically, using different techniques for bracketing the values of α, and a variety of other numerical tricks, but nothing quite produced the desired result. We did however see quite clearly that, for the problem described in this work, FR is substantially more stable than PR, with the latter quickly diverging in most instances.

An example of the performance of GD versus CG can be seen in Fig. 5.12, for the same test case discussed above. In the first minimization run only GD is used. In the second one, we perform 32 GD iterations as pre-conditioning (the number of time slices), in order to give time for the initial sharp discrepancy to propagate, and then revert to conjugate gradient. However, the values of $\mathcal{F}$ for the latter case vary quite abruptly, and do not converge to lower values than for "pure" GD.

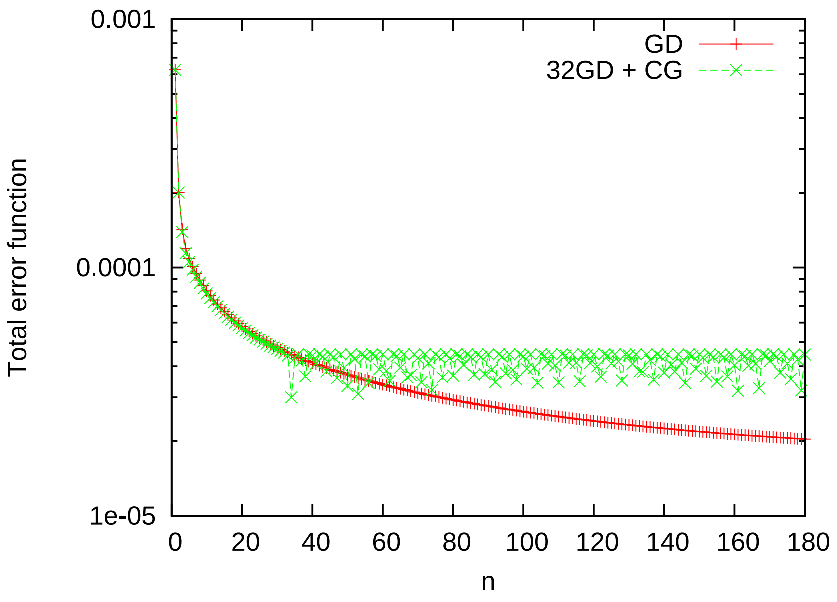


Figure 5.12: Gradient descent versus conjugate gradient minimization. The system is the same as in Fig. 5.11. The CG variant is the Fletcher-Reeves one. After 32 GD iterations, the number of time slices, in the second set of values, we revert to CG which is seen to be highly unstable. The vertical axis displays the total value of $\mathcal{F}$ and the horizontal one the number of minimization iterations.

Another interesting validation from the procedure came in a rather unexpected way. Although this was caused by a mistake, we believe that describing the workflow involved will throw some light on the procedures involved in these minimization runs. The system in question is the $T = 26864$ spacetime orbit, identified in Figs. 5.4, 5.5, 5.6 and 5.9. We have two possible ways of starting the numerical relaxation runs. One is to migrate a checkpoint, with all the values of t_0, i.e., all the LB distribution functions at that time step, the first one of the spacetime minima, and then propagate the LB values in time, by doing the LBGK stream and collide on the MPI "ring" decomposition. This is, however, highly inefficient in terms of wall clock time, since it would require that T MPI processes, each corresponding to a computing core, be involved in the computation, but with only one active at every single time step. In short, the process with MPI rank $t - t_0$ would receive the LB values from process $t - t_0 - 1$, apply LB time evolution, store the results in its memory, but also send them to its "right neighbour" in the MPI topology, process $t - t_0 + 1$.

Although this possibility is indeed implemented in HYPO4D, a much more efficient way to address the issue is to perform the whole initial value problem computation in a previous stage and temporarily store on disk the resulting set of T lattices, each with $L^3 \times Q$ distribution functions. We note that in the computation of the $\Delta(t, T)$ quantities in Section 5.1.2 we used a sampling rate of 500. If, instead we had written all the 2×10^6 time steps, this would in turn require $64^3 \times 19 \times 2 \times 10^6 \times 8 \sim 78 \times 10^{12}$ bytes of memory, an exceedingly high value, even by current petascale computing standards. This in turn means that, as is often the case in high-performance computing, a trade-off must be negotiated between memory resources and computing time.

Returning to our main argument, we then launched a set of simulations to progressively compute all of the $T = 26864$ time steps of the spacetime orbit and store them in memory. However, due to an unfortunate "bug", the magnitude of the ABC force term was 10 times smaller than it should have been. The result of this can be seen on Fig. 5.13, where the energy values are plotted. The red curve represents the correct values of the energy, in a weakly-turbulent regime. The green line, obscured by the thicker blue one, represents the energy values obtained with a force-magnitude ten times smaller. Although the first initial checkpoint, $t_0 = 177K$, is in a turbulent regime, the subsequent ones converge to laminar flow, since the force term is not strong enough to maintain turbulent activity and oppose the inertia of the system.

However, the most interesting part of Fig. 5.13 is the blue line, which shows the total energy values after 200 GD iterations have been applied to the whole system. We see that at both ends of the spacetime orbit, the relaxation procedure is indeed trying to locate a periodic orbit, albeit in an (inadvertently) artificial scenario. The high values of the energy at the beginning of time (in UPO time units) are being lowered, whereas the low values, at the end of the orbit are being increased. This provided a further satisfactory, although accidental, confirmation that the gradient descent-based algorithm was indeed performing well, even on systems with such a large number of variables.

5.3 Large-scale simulations

After the bug just referred to above was corrected, we again tried to numerically relax the $T = 26864$ minimum, corresponding to a 64^3 system with $\tau = 0.53$ and $Re = 371$, this time using the correct value for the ABC-force magnitude, $N = 1/64^3$. The results can be seen in Fig. 5.14, which shows the root-mean-squared (RMS) value of $\mathcal{F}$ per lattice site, over 340 GD numerical relaxation iterations.

Owing to the procedure described in Section 5.1.2, we already start with quite a good initial estimate. For the system in Fig. 5.14, we have an initial value of $\mathcal{F} = 2.3927$ before numerical relaxation is applied. After 340 GD iterations this value is $\mathcal{F} = 5.4697 \times 10^{-2}$. The relevant quantity in this context

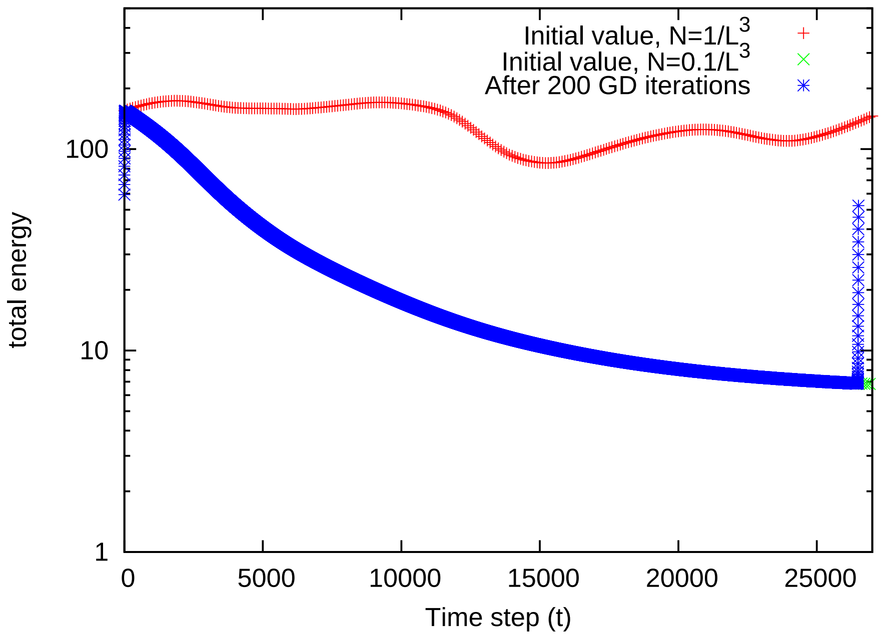


Figure 5.13: Comparison of total energy values before and after GD. The time scale is in units of the "UPO time", where t_0 corresponds to $t = 177K$ in the initial value problem. The orbit numerically minimized was (due to a bug, subsequently fixed) the one with ABC-force magnitude $N = 0.1/L^3$. Nevertheless, the numerical relaxation still tries to minimize the whole system towards a periodic orbit. The system is a 64^3 lattice, with $\tau = 0.53$ and $Re = 371$. Units in the x-axis are in LB time steps, with the values indicating the UPO time indices.

is $\mathcal{F}_{RMS}$ (per lattice site), which can be defined as:

$$\mathcal{F}_{RMS} := \sqrt{\frac{\mathcal{F}}{L^3 \times (2N+1)}}, \tag{5.12}$$

where N is the number of GD iterations. We note that the factor $2N + 1$ is due to how the minimization algorithm propagates the initial sharp discrepancy throughout the lattice, as shown in Fig. 5.11.

Using the definition (5.12) we find that after 340 GD iterations, the $T = 26864$ orbit has a $\mathcal{F}_{RMS} = 1.7504 \times 10^{-5}$. Using these same definitions we obtained, after 300 GD iterations, an RMS value per lattice site of 1.4076×10^{-5}, for the minimum shown in Fig. 5.10, with $T = 24594$. The rate of convergence was very similar to the values presented in Fig. 5.14. Since this spacetime minimum is much steeper than the one shown in Fig. 5.9 it is quite reasonable that an even lower value of (RMS per lattice site) $\mathcal{F}$ was reached with even fewer iterations. The bulk of this work was carried out on the "Intrepid" machine at ANL, although Ranger was also heavily used for preliminary work, such as the computation of the initial value problem.

We note that one of the most extensive works in this area, by Kawahara and Kida [17] imposed a convergence criterion of only 1% (although the authors pointed out that this would probably not be sufficient for the stability analysis of the periodic orbit). Although the procedure described here could

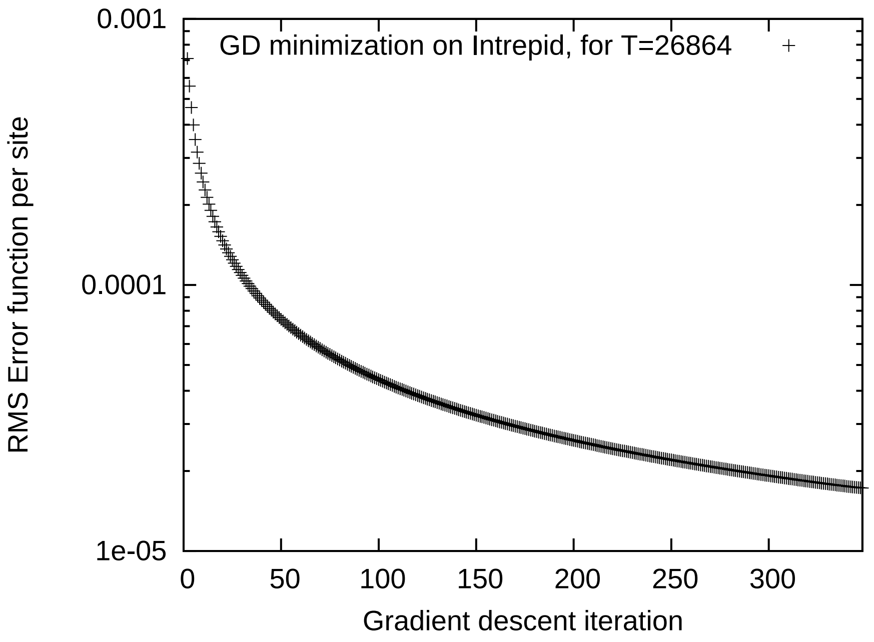


Figure 5.14: Value of $\mathcal{F}_{RMS}$ per lattice site (defined in Eq. (5.12)) with number of GD iterations. The system is the same as Fig. 5.9, with $Re = 371$. The simulation ran on 26864 computing cores of the IBM Blue Gene/P at ANL and took ~ 24 hours of wall clock time, including I/O and initialisation.

still ideally be extended, this is already an extremely low value for the error function over the whole 4D orbit.

Fig. 5.15 shows the total energy of each time slice for the full $T = 26864$ orbit, after 340 GD iterations. We note that this is not constant (which would indicate the trivial case of laminar flow), that it is periodic and reasonably smooth, and that the variations shown (for example the large depression around time slice $15K$) are much larger than the discrepancy between the two end points of the orbit. Indeed, it follows quite closely the values displayed in Fig 5.13 for the (correct) initial value problem, before any minimization was applied.

Since the values for the total energy, as well as for other relevant quantities are indeed varying in a non-trivial way, we can safely conclude that this indeed an unstable periodic orbit. The same is valid for the $t = 637K, T = 24594$ system previously discussed, for which similar findings were found. In the next section we will discuss the several UPOs computed in this work, using gradient descent numerical relaxation.

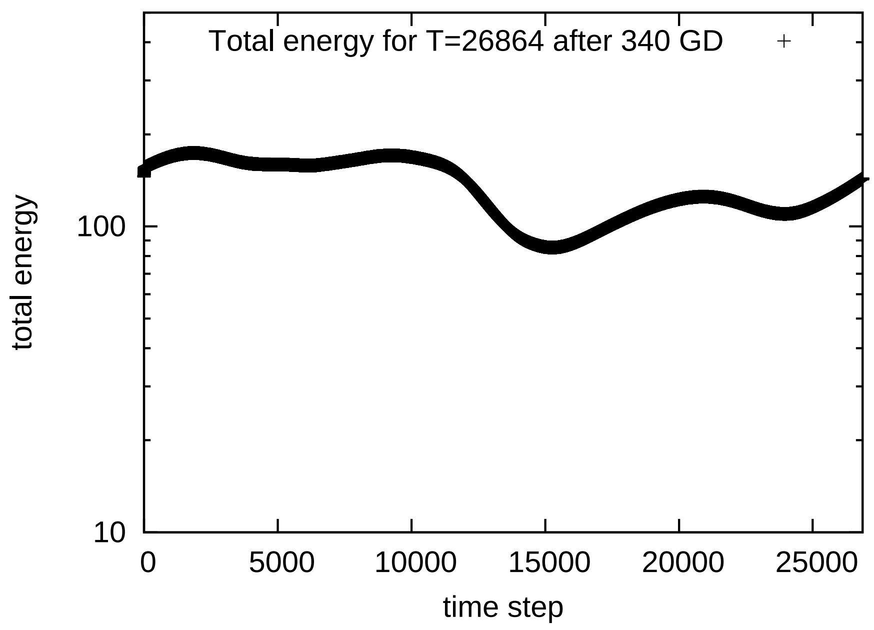


Figure 5.15: Total kinetic energy in each time slice of the orbit corresponding to the $t = 177K$, $T = 26864$ UPO obtained after 340 GD iterations. Units in the x-axis are in LB time steps, with the values indicating the UPO time indices.

5.4 UPOs in the ABC flow

In Table 5.2 we have summarized the four UPOs which have so far been identified using the numerical relaxation algorithm, with a gradient descent search. All of these have been obtained on the Intrepid Blue Gene/P petascale computer at Argonne.

The method for identifying each of the starting points has been described at length in Section 5.1.2. In all cases, the starting points for the numerical relaxation procedure were chosen by taking the lowest values of $\Delta(t, T)$ for a given value of T. However, as seen by the values of $\mathcal{F}_{RMS}$ in Table 5.2, the first two UPOs we identified (with $T = 26,864$ and $T = 24,594$), although significantly larger than the following two, represent deeper minima on the phase space.

All of these orbits have been conveniently stored, and post-processing work and analysis is still ongoing at the time of writing. Due to their very large size, the visualization of all of the time slices is not a trivial matter, but work on this is also ongoing.

Since the orbits are conveniently stored, it is a trivial matter to restart the numerical relaxation procedure from where it stopped (given in each case by the value of n in Table 5.2) and continue the minimization. However, as can be seen in Figs. 5.12 and 5.14, the decrease of the error function, $\mathcal{F}$, becomes increasingly smaller, whereas the amount of wall clock time required for these runs is indeed

t	T	n	$\mathcal{F}_{RMS}$
177,000	26,864	340	1.7504×10^{-5}
637,000	24,594	300	1.4076×10^{-5}
1,240,000	15,790	300	1.9070×10^{-5}
1,631,500	21,592	252	2.8281×10^{-5}

Table 5.2: List of UPOs located for $Re = 371$ in ABC flow. In all cases, a cubic lattice with $L = 64$ was considered with LB viscosity $\nu = 0.01$ ($\tau = 0.53$), and an ABC-force magnitude $N = 1/64^3$. The first column gives the value of t in terms of the initial value problem. The second and third columns show the values for the period, T, of the UPOs and the number, n, of GD iterations performed, respectively. $\mathcal{F}_{RMS}$ is defined in Eq. (5.12).

large. For all of the UPOs referred to in Table 5.2 a convergence criterion of 0.01% was considered. In other words, the minimization procedure was interrupted once the tolerance δ, defined as:

$$\delta \equiv \frac{\mid \mathcal{F}(n) - \mathcal{F}(n-1) \mid}{\mathcal{F}(n-1)} \tag{5.13}$$

reached a threshold of $\delta \leq 1. \times 10^{-04}$. As mentioned before, Kawahara and Kida, in their pioneering work in this area [17], considered a similar criterion which was roughly a hundred times larger, in the sense of being less strict.

As the plots of $\Delta(t, T)$, particularly Fig 5.2, indicate, there appears to be a periodicity in the distribution of the spacetime minima, which is then carried to the UPOs of the system. Heteroclinic connections between the unstable periodic orbits have been recently found in planar Couette flow [102] and we believe this may also be the case in fully-periodic flow stirred by an ABC force. Further research is however required to throw further light on this aspect.

To sum up, we have in the present Chapter described the methodology used for the identification of spacetime minima in weakly-turbulent flow and the implementation of the spacetime numerical relaxation method, based on the variational principle outlined in section 2.4.2. The computational difficulties associated with this methodology and the strategies used to overcome them were presented and discussed. We then showed the first results of our very large numerical spacetime relaxation simulations, performed on the Blue Gene/P machine at Argonne, and the UPOs thus identified in weakly-turbulent flow governed by an ABC force field. In the next Chapter we shall summarize the main achievements and future prospects of this project.

CHAPTER 6

Concluding remarks

In this work, we have presented a novel variational spacetime principle, based on the previous work of Lan and Cvitanović [19]. This was applied to the lattice-BGK equation, a variant of the lattice-Boltzmann method, which we used to simulate hydrodynamic flow, in the limit of near-incompressibility. The resulting algorithm was then applied for the numerical relaxation of spacetime candidate orbits towards UPOs, using petascale computational resources. In the remainder of this Chapter we will now discuss the main insights obtained from this work and future directions.

6.1 Discussion

The main goal of this work was the identification of unstable periodic orbits for weakly turbulent fluid flow by means of the novel spacetime variational principle described in Section 2.4.2. This algorithm is fully 4D and its application to the Navier-Stokes equations required the utilization of petascale computational resources [184], due to the vast memory requirements. To the best of our knowledge, this is the first time that such a spacetime approach has been deployed in hydrodynamics [217]. We also note that this is surely one of the largest minimization problems ever deployed on a computer, due to the huge number of variables each spacetime orbit has, as specified in Chapter 5 of this thesis.

In order for the algorithm to be successfully implemented at a scale of tens of thousands of computing cores, several difficulties had to be overcome in terms of memory load balancing and MPI communication patterns. We found that the most efficient communication pattern for this problem was obtained by a parallelization strategy that allocated one (full) time slice per core, which in turn requires a domain decomposition based on an MPI “ring” topology. Another 4D domain decomposition was also implemented, previous to this, which is based on the (spatial) locality of the lattice sites instead.

Both these two decompositions are special cases of a more general one in which both time and space indices are allowed to vary within each MPI process. This general case can also be implemented quite easily. This will in turn allow for the numerical relaxation of larger lattices, and thus much higher Reynolds numbers than the ones reported here.

However, we believe that a very important aim for this project, and one in which work is ongoing, should be to better address the transition to turbulence and specifically the role played by UPOs in this transition. This view is also reflected in the work of other researchers [17, 101, 104] who have identified UPOs in plane Couette flow in the weakly-turbulent regime, at values of the Reynolds number similar to the ones reported in the present work.

We have demonstrated in this work that the spacetime variational approach, with a gradient descent search, is indeed a valid method for the identification of UPOs. Nevertheless, it is well known that the gradient descent can very easily get "stuck" in a minima search, which is what we have observed. It would be desirable to use GD only as a pre-conditioner [219] and then revert to other more sophisticated methods, such as conjugate gradient. However, we found that CG is highly unstable for this problem, almost certainly due to the convoluted nature of the minima distribution in phase space, and thus it might be desirable to investigate the usage of other minimization algorithms in this respect.

Regarding computational deployment, we are aware that the memory requirements of this approach could be reduced by increasing computation time. One way to achieve this would be to break the spacetime orbit into "orbit segments" and use the sum of the norms of the mismatches between the end of each segment and the beginning of the next as the object function to be minimized. However, every time we needed to compute this quantity, we would still be obliged to run the time evolution along the orbit segments. If the orbit segments all have length one, the above reduces to the strategy followed in this work. Briefly stated, reducing the memory requirements by increasing computation time would cut down on our current spacetime parallelism. Currently existing computers provide us with sufficient memory to perform this computation, so we have availed ourselves of that in order to perform the numerical relaxation in as short a wall clock time as possible.

6.2 Future work

Following the example of what has recently been achieved in plane Couette flow [73, 102], it is highly desirable that the UPOs we have identified are better characterized in terms of their energy input and energy dissipation rates, connecting this to the transition to turbulence in the ABC flow. Another important issue will be to understand the role played by the UPOs of the system in this transition. In this respect, heteroclinic connections between UPOs, already hypothesized by Kawahara and Kida [17], have been found in plane Couette flow, by Gibson and co-workers [73, 102]. It is highly likely that these may also be found in homogeneous isotropic turbulence, such as we studied in this work.

In this area, visualization has proven to be a crucial tool. However, and due to the very high dimensionality involved in hydrodynamics, even of the weakly-turbulent variety, it is of the utmost importance to carefully chose the lower-dimensional representation, so that the main dynamical aspects of the system are indeed highlighted. One suitable choice, as shown by Gibson *et al.* [73] is to select orthonormal basis functions defined in terms of the equilibria of the system and its linear stability eigenfunctions. The application of this methodology to the UPOs in the ABC-driven flow we have computed in this work could provide us with some insights on the dynamical role played by these orbits.

One next logical step in this project will be the development and curation of a digital library of UPOs for several dynamical systems, with emphasis on the three-dimensional Navier-Stokes equations. Due to the sheer size of these UPOs, new computational methods will be required to effectively label and store them. This could in turn benefit greatly by the development of a symbolic dynamics for the system [9], which could then facilitate the application of the DZF formalism, mentioned in Section 2.3.2. We note that although several results have been published recently in the field of describing weakly-turbulent behaviour through the study of UPOs, nobody has yet, to the best of our knowledge, built a dynamical zeta function based on the UPOs of these systems. However, with the current ongoing work being carried out using the Blue Gene/P machine at ANL, we believe this could become a possibility in the very near future, for the ABC flow described in this work.

The main advantage of storing UPOs to represent a turbulent flow is that it needs to be done only once. After that, the turbulent average of any given quantity can be computed directly from the UPO library with high accuracy and without the need to solve an initial value problem again. We believe this methodology has the potential to become a new paradigm in the study of large driven dissipative dynamical systems, and not only for the Navier-Stokes equations.

The dynamical zeta function approach to the analysis of turbulence has evolved through three historical stages. After the discovery of DZFs in the 1970s it was later recognized, in the late 1980s and 1990s, that this could be used as a practical numerical tool once the numerical difficulties involved in the computation of UPOs were overcome. For high-dimensional dynamical systems, such as the Navier-Stokes equations, the computation of its UPOs requires the utilization of petascale resources. Since those have become available to academic researchers only recently, we believe that this methodology is about to enter a third stage of research activity. By computing and classifying some of the smallest UPOs in the driven NSE we expect to be able to make very accurate predictions from first principles for several important quantities, in a systematic fashion, in the near future. This can in turn have far-reaching consequences for the age-old problem of understanding turbulence.

APPENDIX A

The variational principle applied to the lattice-BGK equation

In what follows we shall derive the main equations used in this work for the spacetime relaxation algorithm. The variational principle, described in section 2.4.2 is here applied to the LBGK equation. We begin by rewriting the expression for the equilibrium distribution functions, Eq. (4.5), in a slightly different form:

$$f_i^{eq}(\rho,\boldsymbol{\pi}) = \omega_i \left[\rho + \frac{1}{c_s^2}\boldsymbol{\pi}\cdot\mathbf{c}_i + \frac{1}{2c_s^4\rho}\boldsymbol{\pi}\cdot(\mathbf{c}_i\mathbf{c}_i\cdot - c_s^2 I)\boldsymbol{\pi}\right], \tag{A.1}$$

where $\boldsymbol{\pi} \equiv \rho\mathbf{u}$ represents the momentum density and I is the identity operator. As defined in section 4.1.1, ρ represents the density at a given lattice point, $\mathbf{r}$ and instant t, where those indices are omitted, to simplify the expressions. The exact values for the discrete velocities, $\mathbf{c}_i$, and the weights, w_i, for the $D3Q19$ model used in this work can be found in Table 4.1.

Following the procedure of section 2.4.2 we now define the functional:

$$\mathcal{F} \equiv \frac{1}{2}\sum_{t=0}^{T-1}\sum_{\mathbf{r}}\sum_{i=1}^{Q}|\phi_i(\mathbf{r},t)|^2. \tag{A.2}$$

The quantity $\phi_i(\mathbf{r},t)$ is introduced in order to simplify the subsequent expressions, and is a local residual, which is null at the space-time coordinates $(\mathbf{r},t)$ where the LBGK is satisfied:

$$\phi_i(\mathbf{r},t) \equiv f_i(\mathbf{r}+\mathbf{c}_i,t+1) - f_i(\mathbf{r},t) - \Omega_i(\mathbf{r},t). \tag{A.3}$$

The collision operator, $\Omega_i(\mathbf{r},t)$ has been defined in Eq. (3.16) and is:

$$\Omega_i(\mathbf{r},t) = \frac{1}{\tau}(f_i^{eq}(\rho,\boldsymbol{\pi}) - f_i(\mathbf{r},t)) \tag{A.4}$$

The partial derivative of $\mathcal{F}$ with respect to the distribution function is:

$$\frac{\partial\mathcal{F}}{\partial f_k(\mathbf{s},q)} = \sum_{t=0}^{T-1}\sum_{\mathbf{r}}\sum_{i=1}^{Q}\phi(\mathbf{r},t)\left[\delta_{i,k}\delta_{q,t+1}\delta_{\mathbf{s},\mathbf{r}+\mathbf{c}_i} - \delta_{i,k}\delta_{q,t}\delta_{\mathbf{s},\mathbf{r}} - \frac{\partial\Omega_i(\mathbf{r},t)}{\partial f_k(\mathbf{s},q)}\right], \tag{A.5}$$

where $\delta_{i,j}$ represents the standard Kronecker delta function, defined as:

$$\delta_{i,j} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{A.6}$$

with $i, j \in \mathbb{N}$.

By using the expression for the collision operator as given by the right-hand side of Eq. (3.16) and making extensive use of the chain rule, we can derive an equation equivalent to Eq. (2.29), for fluid described by the LBGK. We begin by writing down the exact composition of the last term in Eq. (A.5), using the chain rule of derivation:

$$\frac{\partial \Omega_i(\mathbf{r},t)}{\partial f_k(\mathbf{s},q)} = \omega \left[\frac{\partial f_i^{eq}(\rho,\boldsymbol{\pi})}{\partial \rho} \frac{\partial \rho}{\partial f_k(\mathbf{s},q)} + \frac{\partial f_i^{eq}(\rho,\boldsymbol{\pi})}{\partial \boldsymbol{\pi}} \frac{\partial \boldsymbol{\pi}}{\partial f_k(\mathbf{s},q)} - \delta_{i,k}\delta_{q,t}\delta_{\mathbf{s},\mathbf{r}} \right], \tag{A.7}$$

where $\omega \equiv \frac{1}{\tau}$, with τ being the LBGK relaxation time. We now compute each one of the terms of Eq. (A.7) individually, using Eq. (A.1), and the definitions of mass and momentum density, Eq. (3.13). Thus:

$$\frac{\partial f_i^{eq}(\rho,\boldsymbol{\pi})}{\partial \rho} = \omega_i - \omega_i \frac{1}{2c_s^4\rho^2} \boldsymbol{\pi} \cdot (\mathbf{c}_i\mathbf{c}_i \cdot -c_s^2 I)\boldsymbol{\pi}, \tag{A.8}$$

$$\frac{\partial \rho}{\partial f_k(\mathbf{s},q)} = I, \tag{A.9}$$

$$\frac{\partial f_i^{eq}(\rho,\boldsymbol{\pi})}{\partial \boldsymbol{\pi}} = \omega_i \frac{\mathbf{c}_i}{c_s^2} + \omega_i \frac{1}{c_s^4\rho} \boldsymbol{\pi} \cdot (\mathbf{c}_i\mathbf{c}_i - c_s^2 I), \tag{A.10}$$

$$\frac{\partial \boldsymbol{\pi}}{\partial f_k(\mathbf{s},q)} = \mathbf{c}_k. \tag{A.11}$$

If we now put together Eqs. (A.5)-(A.11), the result is given by:

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial f_k(\mathbf{s},q)} = & \sum_{t=0}^{T-1} \sum_{\mathbf{r}} \sum_{i=1}^{Q} \phi_i(\mathbf{r},t) \Big(\delta_{i,k}\delta_{q,t+1}\delta_{\mathbf{s},\mathbf{r}+\mathbf{c}_i} - \delta_{i,k}\delta_{q,t}\delta_{\mathbf{s},\mathbf{r}} + \omega \delta_{i,k}\delta_{q,t}\delta_{\mathbf{s},\mathbf{r}} \\ & - \omega \left[\omega_i - \omega_i \frac{1}{2c_s^4\rho^2} \boldsymbol{\pi} \cdot \left(\mathbf{c}_i\mathbf{c}_i \cdot -c_s^2 I \right) \boldsymbol{\pi} \right] \delta_{q,t}\delta_{\mathbf{s},\mathbf{r}} \\ & - \omega \left[\omega_i \frac{\mathbf{c}_i \cdot \mathbf{c}_k}{c_s^2} + \omega_i \frac{1}{c_s^4\rho} \boldsymbol{\pi} \cdot (\mathbf{c}_i\mathbf{c}_i \cdot -c_s^2 I)\mathbf{c}_k \right] \delta_{q,t}\delta_{\mathbf{s},\mathbf{r}} \Big). \end{aligned} \tag{A.12}$$

It must be noted that we have not shown explicitly the $\mathbf{r}$ and t dependence of both ρ and $\boldsymbol{\pi}$, in order not to clutter the expressions even more. However, it is this dependence that leads to the ocurrence of $\delta_{q,t}$ and $\delta_{\mathbf{s},\mathbf{r}}$ in the last terms of the previous expression. After a little algebra, and swapping the i, k indices, as well as renaming the independent variables $\mathbf{s} \to \mathbf{r}$ and $q \to t$, we get the equation:

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial f_i(\mathbf{r},t)} = & \ \phi_i(\mathbf{r}-\mathbf{c}_i,t-1) + (\omega-1)\phi_i(\mathbf{r},t) \\ & - \omega \sum_{k=1}^{Q} \omega_k \Big(1 - \frac{1}{2c_s^4\rho^2} \boldsymbol{\pi} \cdot \left(\mathbf{c}_k\mathbf{c}_k \cdot -c_s^2 I \right) \boldsymbol{\pi} \\ & + \frac{\mathbf{c}_k . \mathbf{c}_i}{c_s^2} + \frac{1}{c_s^4\rho} \boldsymbol{\pi} \cdot (\mathbf{c}_k\mathbf{c}_k \cdot -c_s^2 I)\mathbf{c}_i \Big) \phi_k(\mathbf{r},t). \end{aligned} \tag{A.13}$$

This expression can be further simplified, to make it more amenable to direct numerical implementation, in the following way:

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial f_i(\mathbf{r},t)} &= \phi_i(\mathbf{r}-\mathbf{c}_i,t-1) + (\omega-1)\phi_i(\mathbf{r},t) \\ &\quad - \omega \sum_{k=1}^{Q} \omega_k \Big(1 - \frac{1}{2c_s^4\rho^2}\left(\boldsymbol{\pi}\cdot\mathbf{c}_k\right)^2 + \frac{1}{2c_s^2\rho^2}\left(\boldsymbol{\pi}\cdot\boldsymbol{\pi}\right) \\ &\quad + \frac{\mathbf{c}_k.\mathbf{c}_i}{c_s^2} + \frac{1}{c_s^4\rho}\left(\boldsymbol{\pi}\cdot\mathbf{c}_k\right)\left(\mathbf{c}_k\cdot\mathbf{c}_i\right) - \frac{1}{c_s^2}\left(\boldsymbol{\pi}\cdot\mathbf{c}_i\right)\Big)\phi_k(\mathbf{r},t). \qquad \text{(A.14)}\end{aligned}$$

Equation (A.14) provides the backbone of the numerical relaxation approach implemented in this work. The main aspect, in terms of numerical implementation, that should be mentioned is the explicit dependence on $t-1$ of the first term on the right-hand side. In the same way, the next two terms, involving $\phi_i(\mathbf{r},t)$ and $\phi_k(\mathbf{r},t)$, depend on values of $t = t+1$, by the definition (A.3). The difficulties this raises for numerical (parallel) implementation, and how they were dealt with in this work, are discussed at length in Section 5.1.3.

In a similar fashion, we can derive an equation based on the inverse of the relaxation time, ω:

$$\frac{\partial \mathcal{F}}{\partial \omega} = -\sum_{t=0}^{T-1}\sum_{\mathbf{r}}\sum_{i=1}^{Q}\phi(\mathbf{r},t)\left[f_i^{eq}(\rho,\boldsymbol{\pi}) - f_i(\mathbf{r},t)\right], \qquad \text{(A.15)}$$

where the relaxation time will now be allowed to vary, instead of being constant. This is equivalent to Eq. (2.31) in the continuum case. The fact that we are now using a discrete-time model, LBGK, makes the interpretation of this procedure slightly less straightforward, since the viscosity in the UPO will be different from the one in the initial value problem. For this reason we decided not to use this in the present work. Furthermore, numerical tests showed that including Eq. (A.15) in the minimization procedure did not alter the results significantly, besides making it marginally more stable.

APPENDIX B

Computational aspects

In this appendix we discuss several aspects regarding the computational resources on which the HYPO4D software package was deployed, and the current status of Grid technologies in terms of usability and security issues. It is argued that *middleware* (computer software that connects software components or applications) is a key requirement for the success of Grid computing in particular and e-Science in general. In this context we discuss three new technologies: the Application Hosting Environment (AHE), which hosts scientific applications on computational Grids [183]; the Highly-Available Robust Co-scheduler (HARC), a co-scheduling framework suitable for any resource under the control of a scheduler supporting reservations [220]; and MPIg [221], a Grid-enabled implementation of MPI. The latter two middleware technologies were used extensively, in our work through cross-site runs, described in section B.3. The last section in this appendix addresses several issues related to authentication and authorization in Grid environments, with particular emphasis on usability requirements [222, 223]. We note that the HYPO4D cross-site runs described in section B.3 constitute original work and are one of the main results of the present thesis.

B.1 Grid Computing

Grid computing can be broadly defined as "distributed computing performed transparently across multiple administrative domains" [20]. "Computing" in this context refers for any form of digital activity, such as numerical and symbolic computation, visualization, data-base access or a combination of any of these. In this context, the importance of the "transparency" requirement can not be overlooked. With the onset of larger computational Grids such as the UK's National Grid Service (NGS) [195], Europe's Distributed European Infrastructure for Supercomputing Applications (DEISA) [224] and the US TeraGrid [225], there is an ever-growing demand for usability. The process of accessing and effectively making use of the resources available at a given computing centre or site, or a set of these, across several administrative domains, should be as easy as possible, in terms of the user experience. The protocols involved, which can sometimes be quite complex, should ideally be hidden from the general user whenever that is feasible, if the use of computational Grids is to gain progressively more acceptance within the scientific community at large. This is therefore a strong argument for the development of usable middleware, as has already been addressed by several authors, such as Chin and Coveney [178], Beckles, Welch, and Basney [226] and Chin *et al.* [181].

Recent collaborations between research groups on both sides of the Atlantic [179, 182, 221, 227] have shown the immense benefits to be gained by the use of geographically distributed computing resources. Using $VORTONICS$ as a test case, a software package that simulates and tracks vortex cores in viscous hydrodynamics, Boghosian *et al.*, [227], demonstrated that problems too big to fit on any single supercomputer can be distributed over several computing sites without significant degradation in the performance of the code. More generally, Chin *et al.* [181] discussed some of the possibilities that Grids can provide, in terms of visualization and steering. In this context, the RealityGrid Steering API (Application Programming Interface) [228, 229], allows for a general user to interact in real-time with a running simulation code. This can have many advantages, such as speeding up the exploration of parameter spaces or finding logical breakpoints in the code where the application reaches a given coherent state [181]. The steering library then allows for the use of Grid services in the construction of a generic, dynamic architecture for steering and connecting visualization software to running simulations.

One fairly recent development in this field that is of the utmost importance is the emergence of the "urgent computing" paradigm. This refers to situations where a decision must be made in real-time, where the time frame may vary between a few days to a couple of hours, but the urgency is real and potentially life-saving. In such cases, very large simulations can be submitted to a computational Grid with the highest priority, with the scheduler effectively pre-empting all other jobs running at the time, so that the urgent calculation can proceed immediately.

This is already a well established practice for U.S. resource providers, with several TeraGrid sites hosting such policies, including the University of Chicago/Argonne National Laboratoy (UC/ANL) [200], the National Center for Supercomputing Applications (NCSA) [230], located at the University of Illinois at Urbana-Champaign, the San Diego Super Computer Center (SDSC) [231] and the Texas Advanced Computing Center (TACC) [197]. The middleware that mediates the urgent computing scenario is called SPRUCE: "Special PRiority and Urgent Computing Environment" [232] and has been deployed on the previously-referred sites, along with several others [225]. SPRUCE provides on-demand resource allocation, authorization, and selection capabilities for urgent computing applications that access shared Grid computing or high-performance computing resources. It allows for data centers and virtual organizations[1] [223] to use existing computing infrastructure for time-critical computations.

Patient-specific simulation [187, 233] is another area where the urgent computing paradigm is of growing relevance. In the article, "Life or Death Decision-making: The Medical Case for Large-scale,

[1]These offer a end user access to and use of high performance computing resources shared across a number of different institutions with different administrative security domains.

On-demand Grid Computing" [233], Manos *et al.* argue that "supercomputing site policies, which institute fair share system usage, are not suitable for medical applications as they stand. To support patient-specific medical simulations, where life and death decisions may be made, computational resource providers must give urgent priority to such jobs, and/or facilitate the advance reservation of such resources,". The authors then go on to discuss three patient-specific scenarios: modelling of HIV/AIDS therapies, cancer therapies, and addressing neuro-pathologies in the intracranial vasculature. The latter project, generically called GENIUS (Grid Enabled Neurosurgical Imaging Using Simulation) [234] is particularly note-worthy, as it involves collaboration between computational scientists and clinicians, based at the National Hospital for Neurology and Neurosurgery (NHNN). Combining real patient data with fluid real-time simulations, performed using a lattice-Boltzmann fluid solver called HemeLB [176], this methodology facilitates the planning of embolisation of arterio-venous malformations and aneurysms, amongst other neuro-pathologies. Taking the patient data obtained through X-ray or magnetic resonance imaging angiography, the clinician can thus perform "non-invasive virtual experiments in order to plan and study the effects of certain courses of (surgical) treatment with no danger to the patient" [233]. This, however, requires a close coordination between different computational resources, including the possibility of reserving in advance large computational resources and using urgent-computing techniques, in a life or death scenario. Some of the middleware required to facilitate this will be discussed in the next section. The issue of using real patient-specific data as the basis for Grid simulation, which can potentially span several countries and even continents, immediately raises questions of privacy and data integrity. These are addressed in section B.4.

B.2 Middleware

Going back to the definition of Grid computing, mentioned at the beginning of this chapter, the part related to "transparency across multiple administrative domains" also raises many pertinent questions, ranging from administrative issues to software compability. A parallel has been made between the evolution of the internet and of Grid computing [20]. It was the advent of new protocols, such as `htpp` and `html`, and the related development in browser technology, that made possible the leap between resources used mainly in academic and high tech facilities to a popular world wide medium, which revolutionized communication and many aspects of contemporary life in the space of a few years only, with strong evidence suggesting it will continue to do so for some years to come [46].

However, as can be seen from reports describing some of the pioneering efforts in the field of Grid computing [178, 181], there is still some way to go in order to fulfil the vision of a heterogeneous, on-demand computational Grid, as ubiquitous as the electrical power grid. The difficulties involved range from differences in the location and invocation of compilers and libraries, to non-intuitive middleware, including forms of access to the resources and monitorization of workflows. These can increase enormously the amount of work needed to successfully deploy a particular application within a Grid environment as well as managing the data it produces. Most of the middleware that has been implemented so far to address this issue can be considered “heavyweight” [235], in the sense that it is: *i)* complex to understand, configure or use; *ii)* makes use of resources disproportionately large when compared to the frequency and extent to which they are used; *iii)* has extensive dependencies; *iv)* is difficult or resource-intensive to install, deploy or administer (requiring the involvement of one or more system administrators); *v)* does not scale well. The term “lightweight”, put in a simplified manner, therefore refers in this context to software which has none of the above characteristics or only a small number of them to a very limited extent.

Following on the parallel between Grid computing and the World Wide Web, it does not seem reasonable or even desirable to expect that every scientist using Grid resources should also be an accomplished “expert user”, with a deep knowledge and understanding of the inner workings of these systems, qualities more usually expected to pertain to a systems administrator. In other words, the process of deploying a scientific Grid application should aim to be as user-friendly as possible and, fortunately, some good examples are starting to appear of middleware which tries to address these concerns. In what follows we present three middleware implementations, chosen because they have, each in different ways, been of great relevance to this project.

B.2.1 Application Hosting Environment

The Application Hosting Environment (AHE), [183], is a lightweight web-services based environment, WSRF (Web Services Resource Framework) [236] compliant, which hosts scientific applications on the Grid. Its development stemmed from the assumption that very often in a research group several people will utilize the same application, or set of applications, although they should not all be required to be expert users. Using the AHE it is sufficient that one user installs the application, or set of applications, in the desired Grid resources which then allows the other users in the group to submit jobs to a queue, monitor them and retrieve the resulting data files, using either a command line or a GUI (Graphical User Interface) client, or even a combination of both.

As we can see from this brief description, and as its name itself implies, the emphasis of the AHE is on *applications*, defined as “an entity that can be composed of multiple computational jobs, for example a simulation that consists of two coupled models which requires two jobs to instantiate it” [183]. By allowing a general user to launch and monitor jobs and retrieve any files that have been created in the process, whether the resources consist of a local workstation or a super computing resource located thousands of miles away, the whole process of Grid computing is made much more uniform and transparent.

This framework becomes even more relevant if we consider that nowadays several research groups, based at different institutions and countries, may be using the same application on a given set of computational resources. For this reason, the AHE is now part of the Virtual Physiological Human (VPH) toolkit. The VPH initiative [237] is a European project which aims to support and stimulate the progress of research in biomedical modelling and simulation of the human body. Its ultimate goal consists of enabling collaborative investigation of the human body as a single complex system, by integrating disparate structural and functional models of the living human body.

The latest version of AHE (2.0) [238], which can be downloaded from the RealityGrid project webpage[2], includes the ability to co-reserve time on resources in advance as well as launch both cross-site and steered applications. The first two of these aspects will be discussed in greater detail in the next sections. As for computational steering, although we already discussed that it is an essential component of Grid computing, we did not make use of its possibilities in this project, due to the memory-intensive nature of the HYPO4D application, as discussed in Chapter 5, which requires the use of petascale resources.

A previous version of the middleware, AHE 1.0.2, is included in the UK's OMII (Open Middleware Infrastructure Institute) [239] software release[3]. The AHE is designed to be sufficiently lightweight as to be deployed in Personal Digital Assistant (PDA) mobile devices, an avenue that has been actively pursued, with very encouraging results [240], which we experienced first-hand.

A final note should be made concerning the impact of the AHE in the main work discussed in this thesis. This was more relevant in the earlier stages of the project, and was an excellent way to gain first-hand experience of Grid computing, due to the usability of this middleware. However, it should be mentioned that the primary target applications for the AHE are legacy codes, and applications which are reasonably stable. In the case of a software package that is not only being developed as well as having a very small user base, as was the case of HYPO4D, the benefits of using the Application Hosting Environment become much smaller.

B.2.2 Highly-Available Robust Co-scheduler

The Highly-Available Robust Co-scheduler (HARC) [220, 241] middleware addresses the issue of ensuring several resources will be simultaneously available at a given time, a process known as co-scheduling which is of the utmost importance for the progress of Grid computing.

In some earlier Grid computing projects that made heavy *simultaneous* use of different resources, [179, 182], this question was addressed by means of *ad hoc* procedures. These could be as informal as simply having the phone numbers of several relevant system administrators and reserving the various resources independently for the same time, aiming for a reservation sufficiently far away in the future in order to facilitate this[4]. This procedure obviously does not scale well at all, and is neither efficient nor

[2]**http://www.realitygrid.org/AHE/**

[3]**http://omii.ac.uk/wiki/Downloads**

[4]Bruce M. Boghosian, private communication.

resilient. Among the many problems that crop up, we must mention the time-shifts between different resources, sometimes located in different continents.

The HARC middleware implements an automation of this procedure, in a way which is resilient and has been shown to be well suited for the scheduling of large scientific workflows. Co-scheduling is essentially a *distributed transaction*, by which we mean that a coordinator must ensure that a number of independent resource managers arrive at a consistent state. This procedure has only two possible outcomes: either all scheduling requests are enacted or no requests are enacted, in which case any tentative partial requests (each pertaining to a single site) must be cancelled. In order to accomplish this HARC makes use of Gray and Lamport's Paxos Commit Protocol [242], in which the coordinator process is replaced with a set of replicated processes, called *Acceptors*. The system is fault-tolerant, using a consensus algorithm that proscribes how the Acceptors will behave and eventually reach a decision regarding the booking procedure.

The code has a highly modular nature and can run on any generic resource, with only minor additions. A resource manager component exists in HARC that interacts with the reservation, or queuing, management system of the particular resource. These are usually built into existing batch processing systems, such as IBM's LoadLeveler [243] or the PBS (Portable Batch System) Pro [244]. The resource manager in HARC then exposes a similar interface to the HARC client, which the users interact with. The user can request not only advance reservations in computing resources but also the use of special, dedicated (and reservable) networks, such as the UKLight lightpath, [245], that connects the Manchester and Leeds NGS sites.

HARC has been deployed on the main NGS sites [195] as well as on most of the TeraGrid sites[5]. In section B.3 we describe a typical workflow for cross-site runs, including co-scheduling through HARC, using the HYPO4D application.

As is often the case in computer science, other implementations of the co-scheduling facility exists. One such implementation is the Grid Universal Remote (GUR) [246] which is deployed on several TeraGrid sites, including the NCSA, SDSC and UC/ANL. GUR is a python script that makes use of typical Unix ssh and scp commands in order to help users make reservations, compile programs, and co-schedule jobs [247], thus covering most of the capabilities of HARC. However, at the time of writing, it does not seem to have the same levels of flexibility and reliability as well as acceptance from resource providers that HARC has achieved. In particular, it does not allow for the reservation of netwroks as HARC does.

[5]See the GENIUS project, on the section relating to HARC, for the full (updated) list of sites where the middleware has been deployed. URL: http://wiki.realitygrid.org/wiki/GENIUS_HARC

B.2.3 MPIg

MPIg [221] is a Grid-enabled implementation of MPI, that allows to couple machines pertaining to different computing sites, with potentially heterogeneous architectures, in order to seamlessly run MPI-based applications. It is a new version of MPICH-G2 [248], both having been developed by Brian Toonen and Nick Karonis at Argonne National Laboratory. Like its predecessor, MPIg converts data in messages sent between machines of different architectures (e.g. big endian versus little endian or 64-bit versus 32-bit) and takes care of the whole communication process, selecting automatically MPI (for message passing between processors on the same cluster) or a protocol for intermachine message passing, such as TCP or UDP, if the processors communicating are located on different sites.

The two main features that affect the performance of a code running over a set of distributed resources are bandwidth (rate at which data can be transfered) and latency (the time it takes for a signal to travel between sites). Whereas the restraints on the former are mainly economical, the speed of light places a limit on the latter. The main new feature of MPIg is that it efficiently implements non-blocking communications, meaning that while the processors in a given cluster are waiting to receive data from another site they can still perform other computations, therefore helping to hide inter-site communication delay. This therefore allows for a significant improvement in the performance of Grid applications as preliminary timing results for several different scientific applications show [221].

MPIg has been deployed on the Oxford, Leeds and Manchester NGS sites, as well as on several TeraGrid sites, including NCSA, SDSC and the Louisiana Optical Network Initiative (LONI) [249]. In the next section we discuss our work with HYPO4D in the testing of cross-site runs facilitated through the usage of MPIg, in both the NGS and the TeraGrid.

B.3 HYPO4D Cross-site runs

B.3.1 Introduction

In this section we discuss the performance of the fluid solver component (described at length in Chapter 4) of HYPO4D in cross-site simulations, using MPIg and HARC middleware. The main reasons for this work are two-fold. In the first instance there was the curiosity of trying new technologies and of reporting our results to the resource providers and interested parties, in order to help push the adoption of these new paradigms. This was performed in close collaboration with the GENIUS project, including resource providers and computer scientists in both the UK and the US, and preliminary results of this work were presented at the UK e-Science All-Hands Meeting that took place in Edinburgh University on September 2008. The second main reason was that we initially believed that the only way to gather sufficient computational resources required for the memory-intensive (4D) relaxation procedure (see Chapter 5 for details) would be to aggregate several different sites. With the advent of access to petascale resources in the US, namely Ranger and Intrepid, this possibility was temporarily dropped. However, it would be highly desirable that resource providers paid closer attention to this subject.

It is our firm belief that the advantages reaped by aggregating resources would benefit all agents involved. Ideally, each time a new supercomputing resource would come online, this could be a new node on a supercomputing Grid of planetary scale, thus facilitating the simulation of cutting-edge problems on a scale never before attempted. Unfortunately the current scenario seems to be tilted towards the supercomputing sites fiercely competing amongst themselves for the glory of hosting the largest machine in the world for a brief period of time, only to be quickly succeded by a rival centre[6]. It seems that in this field some lessons might be learned from strategies adopted for commercial distributed computing that are already becoming commonplace among companies [46], although there is a need for caution and extensive testing of these new Grid technologies, with questions such as confidentiality and data integrity being of the utmost relevance [187].

B.3.2 Timing results

In order to perform cross-site runs, several intermediate steps are required, from convencing the resource providers to host the new Grid middleware such as HARC and MPIg, to recompiling the code and requesting reservations. In the following exposition we shall skip some of the details and try instead to give an overview of the process.

The first step was to recompile HYPO4D with MPIg, on a given set of resources that hosted these compilers. No major changes were made to the basic fluid solver code structure. The first timing tests were made on the NGS, using all possible (two site) combinations of the Oxford, Manchester and Leeds sites. Initially, the methodology followed was to regularly monitor these resources and look for a time when some cores on two of them might be free, in order to perform the cross-site timing experiments. As soon as HARC reservations became available on these resources we reverted to using co-scheduling instead, which has obvious advantages in terms of work planning and time efficiency. Once a co-scheduling reservation has been confirmed, the ID of that reservation can be used in a submission script, and the job, after being submitted, will start at the appointed time.

One important issue of these coordinated runs is the need to regularly test all components. It is easily seen that the more computing resources are involved, the greater the chances for failure become. If one of the components, e.g., the co-scheduling mechanism, is temporarily unavailable even on just one site, then any cross-site run involving that site will fail. On the TeraGrid, such a monitoring framework, called INCA [250], has been implemented which regularly checks the status of several key Grid components.[7] More recently, automatic INCA monitoring has also been adopted by the NGS.

[6]For an illustration of this, even a cursory look at a few editions of the bi-annual Top500 list (URL: **http://www.top500.org/**) is more than enough.

[7]A good example of this can be found in the following URL: **http://inca.teragrid.org/inca/html/ctssv3-expanded.html** where the availability of cross-site runs between two given TeraGrid resources is regularly monitored.

On the NGS the most comprehensive HYPO4D timing results were obtained between Leeds and Manchester. The reason for this was simply that the monitoring, before the adoption of INCA, of the required middleware components seemed to be more systematic on these resources. Fig. B.1 shows the value of site updates per second (SUPS), comparing the performance of HYPO4D on a single site (Manchester) with two sites (Manchester and Leeds), using MPIg in both cases. In the latter case, each site had exactly half of the total number of cores (MPI processes) requested. The results obtained are, unfortunately, not impressive, with the cross-site timings lagging very much behind the single site values as well as showing poor scaling, and must be taken more as a proof of principle. Due to the lack of regular testing on the NGS at the time this work was performed, as well as time constraints on the project, we decided to concentrate our efforts on the TeraGrid instead, which also possessed larger core counts.

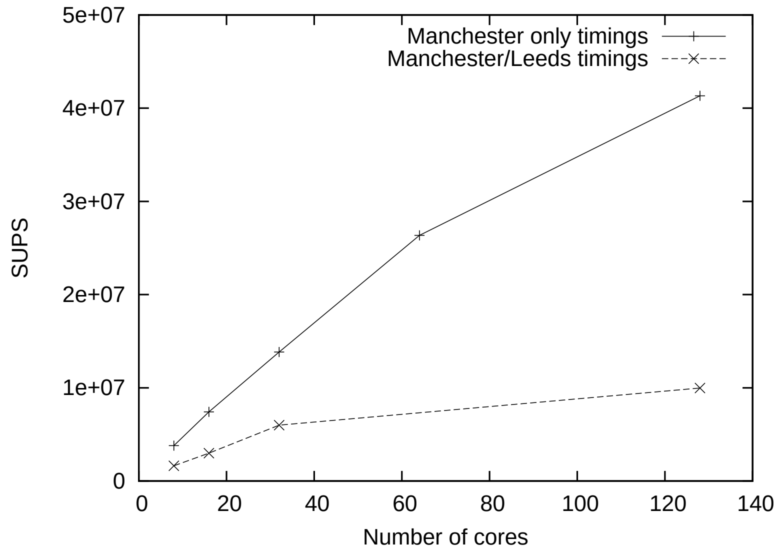


Figure B.1: Comparison between HYPO4D single-site runs (Manchester) and cross-site runs (Manchester-Leeds), using the same MPIg compiler in both cases. Number of SUPS, given as a function of the number of cores used (ranging from 8 to 128) for a 128^3 lattice of fixed size. For the cross-site timings, each site had the same number of cores in all instances.

Fig. B.2 shows a similar comparison, this time performed on the TeraGrid, between the NCSA and SDSC sites. For these tests we considered a fixed sublattice (64^3) per core (soft scaling), increasing the size of the total lattice proportionally to the number of cores. HARC reservations were used in all runs. Although there is still a clear degradation factor between the single-site and two-site performances, there is now a visible scalability trend on the cross-site timings.

Several extensions can be made to this work. We initially chose the NCSA and SDSC sites because they both possessed machines with the same architecture, namely IBM Itanium2 clusters. However, MPIg is not restricted to homogeneous resources, which opens up many possibilities. The reason why this was not pursued, as we have already mentioned, was that in the meantime acess was gained to new petascale machines, which drastically changed the emphasis of the project, in terms of resource deployment. The two main such resources were Ranger [197], to which we gained early-user access on December 2007 and Intrepid [200], on July 2008. It is however worth noticing that NCSA, based at Urbana-Champaign in Illinois, and SDSC, located in San Diego, on the West Coast are physically separated by more than 3000 kilometers. Even so, as Fig. B.2 illustrates, we were able to harness the capabilities of these two centres and use them as one single virtual machine, without any drastic loss efficiency.

In the next section we will discuss some issues related to security in Grid environments, which play an instrumental part in the gradual wider adoption of the distributed computing paradigm.

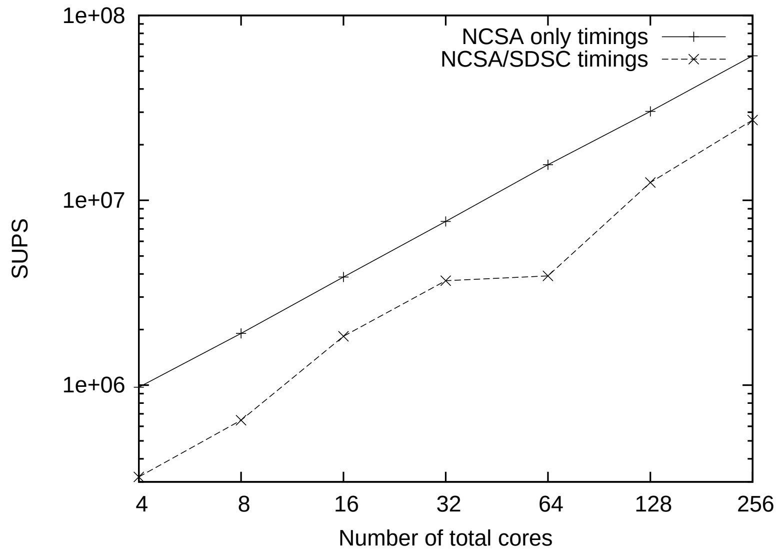


Figure B.2: Comparison between HYPO4D single-site runs (NCSA) and cross-site runs (NCSA-SDSC) on the US TeraGrid, using the same MPIg compiler in both cases. Number of SUPS, given as a function of the number of cores used (ranging from 4 to 256) for a fixed $L = 64$ cubic sublattice per core. For the cross-site timings, each site had the same number of cores in all instances.

B.4 Security in Grid environments

B.4.1 Discussion

The HYPO4D project, described in this thesis, was conducted in close collaboration with computer scientists involved in designing and implementing lightweight, user-friendly, Grid security tools and paradigms [251]. As we argued previously, midlleware is one of the key components to the progressive acceptance and popularization of Grid computing amongst the scientific community. In this context, security issues play a determinant role. A tradeoff exists between solutions which are heavy-handed and gravely hamper the learning curve of a new user, although being robust security-wise, and solutions which are extremely easy to use but may compromise the integrity of the resources provided. Ideally we would want to have the best of both options. This is what will be discussed throughout the remainder of this Appendix.

In order to systematize this discussion we consider the "security process", [252], as being composed of three main operations: authentication, authorization and auditing (see Scheneier [252] for a light introduction to the main concepts, and Gollman [253] for a more technical one). Authentication is the process of determining if the user intending to access some Grid resource really is who he or she claims to be. (The converse is also an important issue, in order that the user does not fall prey to "spoofing" attacks [253], whereby one person or a programme can impersonate a given institution. However we will not delve into this issue here.) Most current computational Grid environments use the Grid Security Infrastructure (GSI) architecture [254], as an authentication mechanism. This framework in turn makes use of the Public Key Infrastructure (PKI) paradigm, which is a security protocol involving digital $X.509$ certificates [255].

Many end-users have complained that these certificates are "cognitively difficult objects" [226], meaning that they are difficult to use as well as understand. As is often pointed out in the larger field of computer security, complexity is one of the main enemies of good security systems, if not the worst [252]. If the end-users find a certain procedure or protocol hard to use, they will, more likely than not, try to circumvent it, thus endangering the whole system. As an example, regarding the process of obtaining Grid credentials, it is widely believed, if not common practice, that many users in the UK share these *individual* certificates within peer groups, in spite of frequent admonitions by the resource providers against that practice. It is claimed as a justification for this that the procedure of obtaining individual credentials can be too difficult or lengthy. If the digital certificate is not properly managed and secured by the general user than its credibility can be compromised, thus eventually allowing for Grid resources to be accessed and explored by unauthorized parties.

In this context, it has been proposed [226, 235, 256, 257] that the user be separated from any interaction at all with the digital certificates. One alternative for implementing this would be to use a "plug-and-play" philosophy for managing PKI [226], similar to what exists for the processing of connecting an individual computer to the internet. This requires the development of lightweight, scalable middleware which must take into account input from user expectations, from its initial stages of design and planning, as opposed to the what has been the general practice so far in Grid implementations.

Regarding the authorization and auditing steps of the security process currently implemented on most Grids, they suffer from much the same faults as mentioned before. Authorization (which determines which resources or data a certain user is allowed access to) often relies on manually maintained lists [235], a procedure which scales poorly, or on heavyweight solutions, some also based on the GSI. This is a very important question in several areas, a particularly relevant one being computational projects for medical purposes, involving patient data and confidentiality, where the levels of authorization for different users accessing different fields of the data must be allowed to vary in a non-trivial fashion. Two examples of such projects, which have already been mentioned here, are GENIUS [234] and the VPH Initiative [237].

As regards auditing, once again GSI is still the principal mechanism, relying entirely on the integrity of the users' credentials, with all the possible vulnerabilities that implies. Besides the Distinguished Name (DN) of the user, a component of the digital certificate, the other information provided in the auditing mechanism is the IP address from which the authentication request appears to come. Once again this is not entirely reassuring due to the generalized practice of IP spoofing.

In a recent paper [222], Abdallah and Haidar compare three identity management schemes used in Virtual Organizations (VO) architectures. The first of these reflects *ad hoc* connections among several organizations, in the second scheme a centrally maintained database exists, whereas the third is based on PKI. The authors conclude that although the the PKI model appears to be much more reliable, the first two models are "simpler, cheaper and easier to implement". They also point out that formal methods [258] can be an important tool in modelling and understanding such complex systems, as well as clarifying the sometimes implicit assumptions about the parties involved.

It seems inevitable that as scientific computing continues to play an increasingly larger part in overall research and wider collaborations between institutions are forged, the problems briefly outlined here will likely become even more pressing.

B.4.2 User-Friendly Security Solutions

Several of the concepts, which have only been briefly sketched here, were actively addressed in an EPSRC-funded project with the title "User-Friendly Security Solutions for Grid Environments" [251]. The aim of this project was to develop *usable* middleware that addressed some of the authentication and authorisation concerns in the field of Grid Computing.

Some of the good practices that deserve mention in this area include the use of formal methods [258] for assistance in modelling and validation and the fact that user input should be taken into account from the earlier stages of software development [223]. In order to circumvent the practice of rogue certificate sharing, this project proposed instead a model of "group certificates", whereby one unique certificate can be shared by several members of a research project, with authorization being managed and fine-grained by an expert user or system administrator.

This solution thus removes digital certificates from the experience of the common user. Instead he or she will now log on to a local Gateway service, that authenticates the user, and serves as a launching point to a set of Grid resources. Since this work shared several of the concerns that guided the design of the AHE, discussed in section B.2.1, in particular the strong emphasis on usability, the two midlleware projects evolved in a highly coordinated fashion. At the time of writing, a prototype for this project is in its final stages of development and will be incorporated within the AHE.

We conclude this brief overview of existing Grid technologies and the challenges they address by stating that middleware is indeed a central component to the long-term success of this new computing paradigm, whose aim is to make computing resources as ubiquitous and unobtrusive as the electrical power grid. New paradigms of organization within Grid domains, such as the concept of virtual organizations [223], are now emerging which make the need for usable, scalable, lightweight middleware an even more pressing issue.

Bibliography

[1] H. Poincaré. *Les méthodes nouvelles de la méchanique céleste*. Guthier-Villars, Paris, 1892–1899.

[2] H. Hesse. *Journey to the East (Die Morgenlandfahrt)*. Samuel Fischer, Berlin, 1932.

[3] U. Frisch. *Turbulence: The legacy of A. N. Kolmogorov*. Cambridge: Cambridge University Press, 1995.

[4] S. R. Sreenivasan. Fluid Turbulence. *Rev. Mod. Phys.*, 71:383–395, 1999.

[5] L. F. Richardson. *Weather Prediction by Numerical Process*. Cambridge: Cambridge University Press; 2nd edition (2007), 1922.

[6] S. Succi and F. Papetti. *An introduction to parallel computational fluid dynamics*. Nova Science Publishers, New York, 1996.

[7] J.-P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.*, 57:617–656, 1985.

[8] D. Auerbach, P. Cvitanović, J.-P. Eckmann, G. Gunaratne, and I. Procaccia. Exploring chaotic motion through periodic orbits. *Phys. Rev. Lett.*, 58(23):2387–89, 1987.

[9] P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner, G. Vattay, N. Whelan, and A. Wirzba. *Chaos: Classical and Quantum (http://www.chaosbook.org/)*. Niels Bohr Institute, Copenhagen, 2005.

[10] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics*. Elsevier, Oxford, 2nd edition, 1987.

[11] P. So, E. Ott, T. Sauer, B. J. Gluckman, C. Grebogi, and S. J. Schiff. Extracting unstable periodic orbits from chaotic time series data. *Phys. Rev. E*, 55(5):5398–5417, 1997.

[12] P. So, J. T. Francis, T. I. Netoff, B. J. Gluckman, and S. J. Schiff. Periodic orbits: a new language for neuronal dynamics. *Biophys. J.*, 74:2776–85, 1998.

[13] E. Ott, C. Grebogi, and J. A. Yorke. Controlling chaos. *Phys. Rev. Lett.*, 64(11):1196–99, 1990.

[14] E. N. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20:130–141, 1963.

[15] B. Eckhardt and G. Ott. Periodic orbit analysis of the Lorenz attractor. *Z. Phys. B*, 93:259–266, 1994.

[16] S. Kato and M. Yamada. Unstable periodic orbits embedded in a shell model of turbulence. *Phys. Rev. E*, 68(2):025302(R), 2003.

[17] G. Kawahara and S. Kida. Periodic motion embedded in plane Couette turbulence: regeneration cycle and burst. *J. Fluid Mech.*, 449:291–300, 2001.

[18] L. van Veen, S. Kida, and G. Kawahara. Periodic motion representing isotropic turbulence. *Fluid Dyn. Res.*, 83:19–46, 2006.

[19] Y. Lan and P. Cvitanović. Variational method for finding periodic orbits in a general flow. *Phys. Rev. E*, 69(1):016217, 2004.

[20] P. V. Coveney. Scientific Grid Computing. *Phil. Trans. R. Soc. A*, 363:1707–13, 2005.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C, Second Edition*. Cambridge: Cambridge University Press, 1992.

[22] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis*. Providence, R.I. : AMS Chelsea Publishing, 1984.

[23] G. Falkovich, K. Gawędzki, and M. Vergassola. Particles and fields in fluid turbulence. *Rev. Mod. Phys.*, 73:913–975, 2001.

[24] W. D. McComb. Theory of Turbulence. *Rep. Prog. Phys.*, 58:1117–1205, 1995.

[25] A. S. Monin and A. M. Yaglom. *Statistical fluid mechanics: mechanics of turbulence*, volume 1. M.I.T. Press, Cambridge MA, 1971.

[26] A. S. Monin and A. M. Yaglom. *Statistical fluid mechanics: mechanics of turbulence*, volume 2. M.I.T. Press, Cambridge MA, 1975.

[27] R. Bowley and M. Sánchez. *Introductory Statistical Mechanics*. Clarendon Press, Oxford, 1996.

[28] O. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Fluids*. Gordon and Breach, New York, 2nd edition, 1969.

[29] C. Fefferman. Existence and smoothness of the Navier-Stokes equation. Clay Millenium Prize problem description. http://www.claymath.org/millennium/Navier-Stokes_Equations/, 2000.

[30] F. M. White. *Viscous Fluid Flow*. McGraw-Hill, New York, 1974.

[31] O. Reynolds. An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Phil. Trans. R. Soc. Lond.*, 174:935–82, 1883.

[32] C. Foias, O. Manley, R. Rosa, and R. Temam. *Navier-Stokes Equations and Turbulence*, volume 83 of *Encyclopedia of Mathematics and its applications*. Cambridge: Cambridge University Press, 2001.

[33] R. I. Bowles. Transition to turbulent flow in aerodynamics. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 358(1765):245–260, 2000.

[34] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Dokl. Akad. Nauk SSSR*, 30:9–13, 1941.

[35] S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, Oxford, 2001.

[36] A. Celani. The frontiers of computing in turbulence: challenges and perspectives. *J. Turbul.*, 8(34), 2007.

[37] L. Finn, B. M. Boghosian, and C. N. Kottke. Vortex core identification in viscous hydrodynamics. *Phil. Trans. R. Soc. A*, 363:1937–48, 2005.

[38] P. Cvitanović (ed.). *Universality in chaos : a reprint selection*. Bristol: Hilger, 2nd edition, 1989.

[39] A. Majda and A. Bertozzi. *Vorticity and incompressible flow*. Cambridge: Cambridge University Press, 2002.

[40] C. R. Doering. The 3D Navier-Stokes problem. *Annu. Rev. Fluid Mech.*, 41:109–128, 2009.

[41] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[42] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation: Foundations and Applications*. North-Holland: Elsevier Science Pub. Co., New York, 1985.

[43] R. Glowinski and O. Pironneau. Finite element methods for Navier-Stokes equations. *Annu. Rev. Fluid Mech.*, 24:167–204, 1992.

[44] M. Y. Hussaini and T. A. Zang. Spectral methods in fluid dynamics. *Annu. Rev. Fluid Mech.*, 19:339–367, 1987.

[45] A. Patera. A spectral element method for fluid dynamics: laminar flow in a channel expansion. *J. Comp. Phys.*, 54:468–488, 1984.

[46] N. Carr. *The Big Switch: Rewiring the World, from Edison to Google*. W. W. Norton & Co., New York, 2008.

[47] http://setiathome.berkeley.edu/.

[48] http://folding.stanford.edu/.

[49] http://lhcathome.cern.ch/.

[50] http://climateprediction.net/.

[51] J. Smagorinsky. General circulation experiments with the primitive equations. I. The basic experiment. *Mon. Weather Rev.*, 91(3):99–164, 1963.

[52] A. Scotti and C. Meneveau. A fractal model for large eddy simulation of turbulent flow. *Physica D*, 127:198–232, 1999.

[53] U. Piomelli and E. Balaras. Wall-layer models for large-eddy simulations. *Annu. Rev. Fluid Mech.*, 34:349–374, 2002.

[54] F. Toschi and E. Bodenschatz. Lagrangian properties of particles in turbulence. *Annu. Rev. Fluid Mech.*, 41:375–404, 2009.

[55] S. A. Orszag and G. S. Patterson, Jr. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett.*, 28(2):76–79, 1972.

[56] R. S. Rogallo. Numerical experiments in homogeneous turbulence. *NASA Tech. Rep., TM-81315, NASA*, 1981.

[57] R. S. Rogallo and P. Moin. Numerical simulation of turbulent flows. *Annu. Rev. Fluid Mech.*, 16:99–137, 1984.

[58] J. A. Domaradzki, R. W. Metcalfe, R. S. Rogallo, and J. J. Riley. Analysis of subgrid-scale eddy viscosity with use of results from direct numerical simulations. *Phys. Rev. Lett.*, 58(6):547–550, 1987.

[59] A. Vincent and M. Meneguzzi. The spatial structure and statistical properties of homogeneous turbulence. *J. Fluid Mech.*, 225:1–20, 1991.

[60] T. Sanada. Cluster statistics of homogeneous turbulence. *Phys. Rev. A*, 44(10):6480–89, 1991.

[61] Z.-S. She, S. Chen, G. Doolen, R. H. Kraichnan, and S. A. Orszag. Reynolds number dependence of isotropic Navier-Stokes turbulence. *Phys. Rev. Lett.*, 70(21):3251–54, 1993.

[62] T. Gotoh and D. Fukayama. Pressure spectrum in homogeneous turbulence. *Phys. Rev. Lett.*, 86(17):3775–78, 2001.

[63] Y. Kaneda, T. Ishihara, M. Yokokawa, K. Itakura, and A. Uno. Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box. *Phys. Fluids*, 15(2):L21–L24, 2003.

[64] Y. Kaneda and T. Ishihara. High-resolution direct numerical simulation of turbulence. *J. Turb.*, 7(20), 2006.

[65] P. K. Yeung, D. A. Donzis, and K. R. Sreenivasan. High-Reynolds-number simulation of turbulent mixing. *Phys. Fluids*, 17:081703, 2005.

[66] D. A. Donzis and P. K. Yeung. Resolution effects and scaling in numerical simulations of passive scalar mixing in turbulence. *Physica D: Nonlinear Phenomena*, 239(14):1278–87, 2010.

[67] D. A. Donzis, P. K. Yeung, and K. R. Sreenivasan. Dissipation and enstrophy in isotropic turbulence: Resolution effects and scaling in direct numerical simulations. *Phys. Fluids*, 20:045108, 2008.

[68] B. E. Launder and D. B. Spalding. *Lectures in mathematical models of turbulence*. Academic Press, London, 1972.

[69] http://www.cfd-online.com/wiki/.

[70] J. E. Bardina, P. G. Huang, and T. J. Coakley. Turbulence modeling validation, testing, and development. *NASA Technical Memorandum 110446, NASA*, 1997.

[71] K. G. Wilson. Renormalization group and critical phenomena. I. Renormalization group and the Kadanoff scaling picture. *Phys. Rev. B*, 4(9):3174–83, 1971.

[72] F. Waleffe. Exact coherent structures in channel flow. *J. Fluid Mech.*, 435:93–102, 2001.

[73] J. F. Gibson, J. Halcrow, and P. Cvitanović. Visualizing the geometry of space state in plane Couette flow. *J. Fluid Mech.*, 611:107–130, 2008.

[74] M. Nagata. Three-dimensional finite-amplitude solutions in plane Couette flow: bifurcation from infinity. *J. Fluid Mech.*, 217:519–527, 1990.

[75] M. Nagata. Three-dimensional traveling-wave solutions in plane Couette flow. *Phys. Rev. E*, 55(2):2023–25, 1997.

[76] T. Itano and S. Toh. The dynamics of bursting process in wall turbulence. *J. Phys. Soc. Japan*, 70(3):703–716, 2001.

[77] F. Waleffe. Three-dimensional coherent states in plane shear flows. *Phys. Rev. Lett*, 81(19):4140–43, 1998.

[78] F. Waleffe. Homotopy of exact coherent structures in plane shear flows. *Phys. Fluids*, 15:1517–43, 2003.

[79] H. Faisst and B. Eckhardt. Travelling waves in pipe flow. *Phys. Rev. Lett*, 91(22):224502, 2003.

[80] H. Wedin and R. R. Kerswell. Exact coherent structures in pipe flow: travelling wave solutions. *J. Fluid Mech.*, 508:333–371, 2004.

[81] F. Waleffe. Hydrodynamic stability and turbulence: beyond transients to a self-sustaining process. *St. Appl. Math.*, 95:319–343, 1998.

[82] F. Waleffe. On a self-sustaining process in shear flows. *Phys. Fluids*, 9:883–900, 1997.

[83] B. Hof, C. W. H. van Doorne, J. Westerweel, F. T. M. Nieuwstadt, H. Faisst, B. Eckhardt, H. Wedin, R. R. Kerswell, and F. Waleffe. Experimental observation of nonlinear traveling waves in turbulent pipe flow. *Science*, 305(5690):1594–98, 2004.

[84] D. Ruelle and F. Takens. On the nature of turbulence. *Commun. Math. Phys*, 20:167–192, 1971.

[85] O. E. Lanford III. The strange attractor theory of turbulence. *Annu. Rev. Fluid Mech.*, 14:347–364, 1982.

[86] V. I. Arnold and B. A. Khesin. *Topological Methods in Hydrodynamics*. New York: Springer, 1998.

[87] R. Temam. Approximation of attractors, large eddy simulations and multiscale methods. *Proc. R. Soc. A*, 434:23–39, 1991.

[88] L. Keefe, P. Moin, and J. Kim. The dimension of attractors underlying periodic turbulent Poiseuille flow. *J. Fluid Mech.*, 242:1–29, 1992.

[89] D. Ruelle. *Chance and Chaos*. Princeton, N.J.: Princeton University Press, 1991.

[90] E. Hopf. A mathematical example displaying features of turbulence. *Commun. Pure Appli. Math.*, 1(4):303–322, 1948.

[91] J.-P. Eckmann. Roads to turbulence in dissipative dynamical systems. *Rev. Mod. Phys.*, 53(4):643–654, Oct 1981.

[92] B. B. Mandelbrot. *The fractal geometry of nature*. Elsevier, New York: W.H. Freeman, 1982.

[93] M. Hénon. A two-dimensional mapping with a strange attractor. *Commun. in Math. Phys.*, 50(1):69–77, 1976.

[94] http://www.scholarpedia.org/article/basin_of_attraction.

[95] E. Ott, J. C. Alexander, I. Kan, J. C. Sommerer, and J. A. Yorke. The transition to chaotic attractors with riddled basins. *Physica D: Nonlinear Phenomena*, 76(4):384–410, 1994.

[96] O. E. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[97] Y. Ueda. Randomly transitional phenomena in the system governed by Duffing's equation. *J. Stat. Phys*, 20(2):181–196, 1979.

[98] G. Kawahara. Laminarization of minimal plane Couette flow: Going beyond the basin of attraction of turbulence. *Phys. Fluids*, 17:041702, 2004.

[99] J. Wang, J. F. Gibson, and F. Waleffe. Lower branch coherent states: transition and control. *Phys. Rev. Lett.*, 98(20):204501, 2008.

[100] D. Viswanath. Recurrent motions within plane Couette turbulence. *J. Fluid Mech.*, 580:339–358, 2007.

[101] T. M. Schneider, J. F. Gibson, M. Lagha, F. De Lillo, and B. Eckhardt. Laminar-turbulent boundary in plane Couette flow. *Phys. Rev. E*, 78(3):037301, 2008.

[102] J. Halcrow, J. F. Gibson, P. Cvitanović, and D. Viswanath. Heteroclinic connections in plane Couette flow. *J. Fluid Mech.*, 621:365–376, 2009.

[103] J. F. Gibson, J. Halcrow, and P. Cvitanović. Equilibrium and travelling-wave solutions of plane Couette flow. *J. Fluid Mech.*, 638:243–266, 2009.

[104] T. M. Schneider, J. F. Gibson, and J. Burke. Snakes and ladders: Localized solutions of plane Couette flow. *Phys. Rev. Lett.*, 104(10):104501, 2010.

[105] M. Kawasaki and S. Sasa. Statistics of unstable periodic orbits of a chaotical dynamical system with a large number of degrees of freedom. *Phys. Rev. E*, 72(3):037202, 2005.

[106] D. Ruelle. *Thermodynamic formalism: the mathematical structure of equilibrium statistical mechanics*. Cambridge: Cambridge University Press, 2nd edition, 2004.

[107] Y. W. Koh and K. Takatsuka. Finding periodic orbits of higher-dimensional flows by including tangential components of trajectory motion. *Phys. Rev. E*, 76(6):066205, 2007.

[108] Y. Saiki. Numerical detection of unstable periodic orbits in continuous-time dynamical systems with chaotic behaviors. *Nonlin. Processes Geophys.*, 14:615–620, 2007.

[109] R. C. Hilborn. *Chaos and non-linear dynamics: an introduction for scientists and engineers*. New York: Oxford University Press, 1994.

[110] K. Pyragas. Continuous control of chaos by self-controlling feedback. *Phys. Lett. A*, 170:421–428, 1992.

[111] B. M. Boghosian, P. V. Coveney, L. Fazendeiro, J. Lätt, J. Tam, and H. Tang. A computational program for the dynamical systems approach to turbulence. *Preprint*, 2009.

[112] V. L. Ginzburg and L. D. Landau. *Zh. Eksp. Teor. Fiz.*, 20:1064, 1950.

[113] L. Finn and B. M. Boghosian. A global variational approach to vortex core identification. *Physica A*, 362:11–16, 2006.

[114] http://www.openlb.org/.

[115] N. T. Ouellette and J. P. Gollub. Curvature fields, topology, and the dynamics of spatiotemporal chaos. *Phys. Rev. Lett.*, 99(19):194502, 2007.

[116] P. Koumoutsakos. Multiscale flow simulations using particle methods. *Annu. Rev. Fluid Mech.*, 37:457–487, 2005.

[117] R. Delgado-Buscalioni, P. V. Coveney, and G. De Fabritiis. Towards multi-scale modelling of complex liquids using hybrid particle-continuum schemes. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 222(5):769–776, 2008.

[118] P. J. Love, M. Nekovee, P. V. Coveney, J. Chin, N. González-Segredo, and J. M. R. Martin. Simulations of amphiphilic fluids using mesoscale lattice-Boltzmann and lattice-gas methods. *Comp. Phys. Commun.*, 153(3):340–358, 2003.

[119] R. S. Saksena and P. V. Coveney. Shear rheology of amphiphilic cubic liquid crystals from large-scale kinetic lattice-Boltzmann simulations. *Soft Matter*, 5(22):4446–63, 2009.

[120] I. Halliday, T. J. Spencer, and C. M. Care. Validation of multicomponent lattice Boltzmann equation simulations using theoretical calculations of immiscible drop shape. *Phys. Rev. E*, 79(1):016706, 2009.

[121] http://www.top500.org/.

[122] T. Bohr, M. H. Jensen, G. Paladin, and A. Vulpiani. *Dynamical Systems Approach to Turbulence (Cambridge Nonlinear Science Series)*. Cambridge: Cambridge University Press, 1998.

[123] S. Goldstein. *Classical Mechanics*. Addison-Wesley, London, 1959.

[124] K. Huang. *Statistical Mechanics*. Wiley, New York, 1987.

[125] C. Cercignani. *Theory and Application of the Boltzmann Equation*. Elsevier, New York, 1975.

[126] P. V. Coveney and R. Highfield. *The Arrow of Time: the quest to solve science's greatest mystery*. Flamingo, London, 1991.

[127] C. Cercignani. *Ludwig Boltzmann: The Man Who Trusted Atoms*. Oxford: Oxford University Press, 1998.

[128] S. Chapman and T. G. Cowling. *The Mathematical Theory of Non-uniform Gases*. Cambridge: Cambridge University Press, 2nd edition, 1952.

[129] J. Chin. *Mesoscale fluid simulation with the Lattice Boltzmann method*. PhD thesis, Queen Mary, University of London, 2005.

[130] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Lett.*, 56(14):1505–08, 1986.

[131] S. Wolfram. Cellular automaton fluids 1: Basic theory. *J. Stat. Phys.*, 45:471–526, 1986.

[132] J. Hardy, O. de Pazzis, and Y. Pomeau. Molecular dynamics of a lattice gas: transport properties and time correlation functions. *Phys. Rev. A*, 13(5):1949–61, 1976.

[133] G. McNamara and G. Zanetti. Use of the Boltzmann equation to simulate lattice-gas automata. *Phys. Rev. Lett.*, 61(20):2332–35, 1988.

[134] P. Grosfils, J.-P. Boon, and P. Lallemand. Spontaneous fluctuation correlations in thermal lattice-gas automata. *Phys. Rev. Lett.*, 68(7):1077–80, 1992.

[135] X. He and L.-S. Luo. *A priori* derivation of the lattice Boltzmann equation. *Phys. Rev. E*, 55(6):R6333–36, 1997.

[136] T. Abe. Derivation of the lattice Boltzmann method by means of the discrete ordinate method for the Boltzmann equation. *J. Comp. Phys.*, 131:241–246, 1997.

[137] F. J. Higuera and J. Jimenez. Boltzmann approach to lattice gas simulations. *Europhys. Lett.*, 9:663–668, 1989.

[138] F. J. Higuera, S. Succi, and R. Benzi. Lattice gas dynamics with enhanced collisions. *Europhys. Lett.*, 9:345–349, 1989.

[139] R. Benzi, S. Succi, and M. Vergassola. The lattice boltzmann equation: theory and applications. *Physics Reports*, 222(3):145–197, 1992.

[140] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329–364, 1998.

[141] S. Chen, H. D. Chen, D. Martínez, and W. Matthaeus. Lattice Boltzmann model for simulation of magnetohydrodynamics. *Phys. Rev. Lett.*, 67(27):3776–79, 1991.

[142] J. M. Koelman. A simple lattice Boltzmann scheme for Navier-Stokes fluid flow. *Europhys. Lett.*, 15:603–607, 1991.

[143] Y. H. Qian, D. d'Humières, and P. Lallemand. Lattice BGK models for the Navier-Stokes equation. *Europhys. Lett.*, 17:479–484, 1992.

[144] P. Bhatnagar, E. Gross, and M. Krook. A model for collision processes in gases i: small amplitude processes in charged and neutral one-component systems. *Phys. Rev.*, 94:511–525, 1954.

[145] D. d'Humières. Rarefied gas dynamics: theory and simulations. *Prog. Aeronaut. Astronaut.*, 159:450–458, 1992.

[146] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Yuo. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Phil. Trans. R. Soc. Lond. A*, 360:437–451, 2002.

[147] P. Lallemand and L.-S. Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. *Phys. Rev. E*, 61(6):6546–62, 2000.

[148] J. Lätt and B. Chopard. Lattice Boltzmann method with regularized pre-collision distribution functions. *Math. Comp. Sim.*, 72:165–168, 2006.

[149] B. M. Boghosian, J. Yepez, P. V. Coveney, and A. Wagner. Entropic lattice Boltzmann methods. *Proc. R. Soc. London A*, 457:717, 2001.

[150] S. Succi, I. V. Karlin, and H. Chen. Colloquium: Role of the *H* theorem in lattice Boltzmann hydrodynamic simulations. *Rev. Mod. Phys.*, 74:1203–20, 2002.

[151] I. V. Karlin, A. Ferrante, and H. C. Öttinger. Perfect entropy functions of the lattice Boltzmann method. *Europhys. Lett*, 47(2):182–188, 1999.

[152] H. Chen and C. Teixeira. *H*-theorem and origins of instability in thermal lattice Boltzmann models. *Comput. Phys. Commun.*, 129:21–31, 2000.

[153] S. S. Chikatamarla and I. V. Karlin. Entropy and Galilean invariance of lattice Boltzmann theories. *Phys. Rev. Lett.*, 97(19):190601, 2006.

[154] B. M. Boghosian, P. J. Love, P. V. Coveney, I. V. Karlin, S. Succi, and J. Yepez. Galilean-invariant lattice-Boltzmann models with *H* theorem. *Phys. Rev. E*, 68(2):025103(R), 2003.

[155] S. Ansumali and I. V. Karlin. Stabilization of the lattice Boltzmann method by the *H* theorem: A numerical test. *Phys. Rev. E*, 62(6):7999–8003, 2000.

[156] R. Benzi and S. Succi. Two-dimensional turbulence with the lattice Boltzmann equation. *J. Phys. A*, 23:L1–L5, 1990.

[157] D. O. Martínez, W. H. Matthaeus, S. Chen, and D. C. Montgomery. Comparison of spectral method and lattice Boltzmann simulations of two-dimensional hydrodynamics. *Phys. Fluids*, 6(3):1285–98, 1994.

[158] Y. H. Qian, S. Succi, and S. A. Orszag. Recent advances in lattice Boltzmann computing. *Annu. Rev. Comp. Phys.*, 3:195–242, 1995.

[159] R. H. Kraichnan. Inertial ranges in two-dimensional turbulence. *Phys. Fluids*, 10:1417–23, 1967.

[160] H. Chen, S. Chen, and W. H. Matthaeus. Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method. *Phys. Rev. A.*, 45:R5339–42, 1992.

[161] C. Treviño and F. Higuera. Lattice Boltzmann and spectral simulations of non-linear stability of Kolmogorov flows. *Rev. Mex. Fis.*, 40:878–890, 1994.

[162] S. Hou, J. Sterling, S. Chen S, and G. D. Doolen. A lattice Boltzmann subgrid model for high Reynolds number flows. *Fields Inst. Comm.*, 6:151–166, 1996.

[163] J. A. Sommers. Direct simulation of fluid flow with cellular automata and the lattice- Boltzmann equation. *Applied Sci. Res.*, 51:127–133, 1993.

[164] S. Succi, G. Amati, and R. Benzi. Challenges in lattice Boltzmann computing. *J. Stat. Phys.*, 81:5–16, 1995.

[165] J. G. M. Eggels. Direct and large-eddy simulation of turbulent fluid flow using the lattice-Boltzmann scheme. *Int. J. Heat Fluid Flow*, 17:307–323, 1996.

[166] J. Lätt, B. Chopard, S. Succi, and F. Toschi. Numerical analysis of the averaged flow field in a turbulent lattice Boltzmann simulation. *Physica A*, 362:6–10, 2006.

[167] H. Chen, S. Kandasamy, S. Orszag, R. Shock, S. Succi, and V. Yakhot. Extended Boltzmann kinetic equation for turbulent flows. *Science*, 301:633–636, 2003.

[168] http://www.exa.com/pages/pflow/pflow_main.html.

[169] D. R. Noble, J. G. Georgiadis, and R. O. Buchius. Comparison of accuracy and performance for lattice Boltzmann and finite difference simulations of steady viscous flow. *Int. J. Num. Meth. Fluids*, 23:1–18, 2003.

[170] K. Sankaranarayanan, I. G. Kevrekidis, S. Sundaresan, J. Lu, and G. Tryggvason. A comparative study of lattice boltzmann and front-tracking finite-difference methods for bubble simulations. *Int. J. Multiphase Flow*, 29:109–116, 2003.

[171] J. Bernsdorf, F. Durst, and M. Schäfer. Comparison of cellular automata and finite volume techniques for simulation of incompressible flows in complex geometries. *Int. J. Num. Meth. Fluids*, 29:251–264, 1999.

[172] M. Breuer, J. Bernsdorf, T. Zeiser, and F. Durst. Accurate computations of the laminar flow past a square cylinder based on two different methods: Lattice-Boltzmann and finite-volume. *Int. J. Heat Fluid Flow*, 21:186–196, 2000.

[173] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron. Benchmark computations based on lattice boltzmann, finite element and finite volume methods for laminar flows. *Computers & Fluids*, 35:888–897, 2006.

[174] D. Kandhai, D. J.-E. Vidal, A. G. Hoekstra, H. Hoefsloot, R. Iedema, and P. M. A. Sloot. Lattice-Boltzmann and finite element simulations of fluid flow in a SMRX static mixer reactor. *Int. J. Num. Meth. Fluids*, 31:1019–33, 1999.

[175] S. Chen, Z. Wang, X. Shan, and G. D. Doolen. Lattice Boltzmann computational fluid dynamics in three dimensions. *J. Stat. Phys.*, 68:379–400, 1992.

[176] M. D. Mazzeo and P. V. Coveney. HemeLB: A high performance parallel lattice-Boltzmann code for large scale fluid flow in complex geometries. *Comp. Phys. Commun.*, 178(12):894–914, 2008.

[177] R. Mei, W. Shyy, D. Lu, and L.-S. Luo. Lattice Boltzmann method for 3-D flows with curved boundary. *J. Comput. Phys.*, 161:680–699, 2000.

[178] J. Chin and P. V. Coveney. Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware. UK e-Science Technical Report, number UKeS-2004-01, http://www.realitygrid.org/lgpaper.html, 2004.

[179] R. J. Blake, P. V. Coveney, P. Clarke, and S. M. Pickles. The TeraGyroid Experiment - Supercomputing 2003. *Scientific Computing*, 13(1):1–17, 2005.

[180] B. M. Boghosian and P. V. Coveney. Scientific applications of Grid computing. *Computing in Science and Engineering*, 7(5):10–13, 2005.

[181] J. Chin, M. J. Harvey, S. Jha, and P. V. Coveney. Scientific grid computing: The first generation. *Computing in Science and Engineering*, 7(5):24–32, 2005.

[182] B. M. Boghosian, P. V. Coveney, S. Dong, L. Finn, S. Jha, G. Karniadakis, and N. Karonis. NEKTAR, SPICE and Vortonics: Using federated Grids for large scale scientific applications. *Cluster Computing*, 10(3):351–364, 2007.

[183] P. V. Coveney, R. S. Saksena, S. J. Zasada, M. McKeown, and S. Pickles. The application hosting environment: Lightweight middleware for grid-based computational science. *Comp. Phys. Commun.*, 176(6):406–418, 2007.

[184] R. S. Saksena, B. M. Boghosian, L. Fazendeiro, O.A. Kenway, S. Manos, M. D. Mazzeo, S. K. Sadiq, J. L. Suter, D. Wright, and P. V. Coveney. Real Science at the Petascale. *Phil. Trans. R. Soc. A*, 367(1897):2557–71, 2009.

[185] M. Snir, S. W. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI - The Complete Reference*. MIT Press, Cambridge, MA, 1998.

[186] J. Harting, M. J. Harvey, J. Chin, and P. V. Coveney. Detection and tracking of defects in the gyroid mesophase. *Comp. Phys. Commun.*, 165(2):97–109, 2005.

[187] S. K. Sadiq, M. D. Mazzeo, S. J. Zasada, S. Manos, I. Stoica, C. V. Gale, S. J. Watson, P. Kellam, S. Brew, and P. V. Coveney. Patient-specific simulation as a basis for clinical decision-making. *Phil. Trans. R. Soc. A*, 366(1878):3199–3219, 2008.

[188] M. D. Mazzeo, S. Manos, and P. V. Coveney. In situ ray tracing and computational steering for interactive blood flow simulation. *Comp. Phys. Commun.*, 181(2):355–370, 2010.

[189] Q. Zou and X. He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Phys. Fluids*, 9(6):1591–98, 1997.

[190] M. Sbragaglia and S. Succi. Analytical calculation of slip flow in lattice Boltzmann models with kinetic boundary conditions. *Phys. Fluids*, 17:093602, 2005.

[191] R. S. Maier, R. S. Bernard, and D. W. Grunau. Boundary conditions for the lattice Boltzmann method. *Phys. Fluids*, 8(7):1788–1801, 1996.

[192] S. Chen, D. Martínez, and R. Mei. On boundary conditions in lattice Boltzmann methods. *Phys. Fluids*, 8(9):2527–36, 1996.

[193] G. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Conference Proceedings*, 30:483–485, 1967.

[194] http://www.hector.ac.uk/.

[195] http://www.grid-support.ac.uk/.

[196] http://www.hpcx.ac.uk/.

[197] http://www.tacc.utexas.edu/.

[198] http://www.netlib.org/linpack/.

[199] L. Fazendeiro, B. M. Boghosian, P. V. Coveney, J. Lätt, and H. Tang. Search for unstable periodic orbits in the Navier-Stokes equations. *Proceedings of the TeraGrid'08 conference, Las Vegas, June 9-13, 2008, http://archive.teragrid.org/events/teragrid08/Papers/papers/46.pdf.*

[200] http://www.anl.gov/.

[201] H. S. Morse. *Practical parallel computing*. AP Professional, Boston, 1994.

[202] http://www.nag.co.uk/.

[203] http://www.ietf.org/rfc/rfc1832.txt.

[204] Z. Guo, C. Zheng, and B. Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Phys. Rev. E*, 65(4):046308, 2002.

[205] X. Shan and S. Chen. Lattice Boltzmann method for simulating flows with multiple phases and components. *Phys. Rev. E*, 47(3):1815–19, 1993.

[206] J. Lätt, O. Malaspinas, and B. Chopard. External force and boundary conditions in lattice Boltzmann. *Phys. Rev. E*, Submitted, 2010.

[207] T. Dombre, U. Frisch, J. M. Greene, M. Hénon, A. Mehr, and A. M. Soward. Chaotic streamlines in the ABC flows. *J. Fluid Mech.*, 167:353–391, 1986.

[208] X.-H. Zhao, K.-H. Kwek, J.-B. Lin, and K.-L. Huang. Chaotic and resonant streamlines in the ABC flow. *SIAM J. Appl. Math.*, 53(1):71–77, 1993.

[209] L. Finn. *A Variational Approach to Vortex Core Identification in Viscous Hydrodynamics*. PhD thesis, Tufts University, 2006.

[210] W. B. Zimmerman and B. N. Hewakandamby. On the mixing properties of spherically symmetric helical coherent structures in turbulence. *Chemical Engineering Science*, 61(9):2826–34, 2006.

[211] H. Aref. Stirring by chaotic advection. *J. Fluid Mech.*, 143:1–21, 1984.

[212] V. I. Arnold. Sur la topologie des écoulements stationnaires des fluides parfaits. *C.R. Acad. Sci. Paris*, 261:17–20, 1965.

[213] D. Galloway and U. Frisch. A note on the stability of a family of space-periodic Beltrami flows. *J. Fluid Mech.*, 180:557–564, 1987.

[214] S. Childress. New solutions of the kinematic dynamo problem. *J . Math. Phys.*, 11:3063–71, 1970.

[215] M. Abramowitz and I. A. Stegun (eds.). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1964.

[216] B. B. Mandelbrot. Intermittent turbulence in self-similar cascades: divergence of high moments and dimension of the carrier. *J. Fluid Mech.*, 62:331–358, 1974.

[217] L. Fazendeiro, B. M. Boghosian, P. V. Coveney, and J. Lätt. Unstable periodic orbits in weak turbulence. *Journal of Computational Science*, 1:13–23, 2010.

[218] J. W. Neuberger and R. J. Renka. Sobolev gradients and the Ginzburg–Landau functional. *SIAM Journal on Scientific Computing*, 20(2):582–590, 1998.

[219] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.418, 1994.

[220] J. MacLaren. HARC: The highly-available resource co-allocator. In *OTM Conferences (2)*, volume 4804 of *Lecture Notes in Computer Science*, pages 1385–1402. Springer, 2007.

[221] S. Manos, M. Mazzeo, O. Kenway, P. V. Coveney, N. T. Karonis, and B. Toonen. Distributed MPI cross-site run performance using MPIg. In *HPDC '08: Proceedings of the 17th International Symposium on high performance distributed computing*, pages 229–230. ACM, New York, NY, USA, 2008.

[222] A. E. Abdallah and A. N. Haidar. Usability evaluation of Identity Management schemes in three Virtual Organisation architectures. *Journal of Information Assurance and Security*, 4(6):560–570, 2009.

[223] A. N. Haidar, A. E. Abdallah, P. Y. A. Ryan, P. V. Coveney, B. Beckles, J. M. Brooke, and M. A. S. Jones. Formal modelling of a usable identity management solution for virtual organisations. *Electronic Proceedings in Theoretical Computer Science*, (16):41–50, 2010.

[224] http://www.deisa.eu/.

[225] http://www.teragrid.org/.

[226] B. Beckles, V. Welch, and J. Basney. Mechanisms for increasing the usability of grid security. *Int. J. Human-Computer Studies*, (63):74–101, 2005.

[227] B. M. Boghosian, L. Finn, and P. V. Coveney. Moving the data to the computation: multi-site distributed parallel computation. RealityGrid, Tech. Rep., http://www.realitygrid.org/publications.shtml, 2006.

[228] J. M. Brooke, P. V. Coveney, J. Harting, S. Jha, S. M. Pickles, R. L. Pinning, and A. R. Porter. Computational Steering in RealityGrid. *Proceedings of the UK e-Science All Hands Meeting, http://www.nesc.ac.uk/events/ahm2003/AHMCD/*, 2003.

[229] S. M. Pickles, R. Haines, R. L. Pinning, and A. R. Porter. A practical toolkit for computational steering. *Phil. Trans. R. Soc. A*, 363:1843–53, 2005.

[230] http://www.ncsa.illinois.edu/.

[231] http://www.sdsc.edu/.

[232] P. Beckman, I. Beschatnikh, S. Nadella, and N. Trebon. Building an infrastructure for urgent computing. In *High Performance Computing and Grids in Action*, number 3834, pages 146–156. IOS, Amsterdam, The Netherlands, 2007.

[233] S. Manos, S. Zasada, and P. V. Coveney. Life or death decision-making: the medical case for large-scale, on-demand grid computing. *CTWatch Q. J.*, 4:35–45, 2008.

[234] http://wiki.realitygrid.org/wiki/genius.

[235] B. Beckles, P. V. Coveney, P. Y. A. Ryan, A. E. Abdallah, S. M. Pickles, J. M. Brooke, and M. McKeown. A user-friendly approach to computational grid security. *Proceedings of the UK e-Science All Hands Meeting, http://www.allhands.org.uk/2006/proceedings/papers/636.pdf*, 2006.

[236] S. Graham, A. Karmarkar, J. Mischkinsky, I. Robinson, and I. Sedukin. Web Services Resource Framework. OASIS technical report. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf, 2006.

[237] http://www.vph-noe.eu/.

[238] S. J. Zasada and P. V. Coveney. Virtualizing Access to Scientific Applications with the Application Hosting Environment. *Comp. Phys. Commun.*, 180(12):2513–25, 2009.

[239] http://omii.ac.uk/.

[240] I. Holmes and R. S. Kalawsky. The RealityGrid PDA and Smartphone clients: Developing effective handheld user interfaces for e-Science. *Proceedings of the UK e-Science All Hands Meeting, http://www.allhands.org.uk/2006/proceedings/papers/590.pdf*, 2006.

[241] J. MacLaren and M. McKeown. HARC: A Highly-Available Robust Co-scheduler. http://www.realitygrid.org/middleware.shtml/.

[242] J. Gray and L. Lamport. Consensus on transaction commit. *ACM Transactions on Database Systems*, 31(1):133–160, 2006.

[243] http://www-01.ibm.com/software/tivoli/products/scheduler-loadleveler/.

[244] http://www.pbsgridworks.com/.

[245] http://www.ja.net/.

[246] K. Yoshimoto, P. Kovatch, and P. Andrews. Co-scheduling with user-settable reservations. In *Job scheduling strategies for parallel processing*, number 3834 in Lecture Notes in Computer Sciences, pages 146–156. Springer, Berlin, Germany, 2005.

[247] http://www.teragrid.org/userinfo/jobs/gur.php.

[248] N. T. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-enabled implementation of the Message Passing Interface. *J. Parallel Distrib. Comput.*, 63(5):551–563, 2003.

[249] http://www.loni.org/.

[250] http://inca.sdsc.edu/drupal/.

[251] http://www.realitygrid.org/uf-security/.

[252] B. Schneier. *Secrets and Lies: Digital Security in a Networked World.* Indianapolis: John Wiley & Sons, 2000.

[253] D. Gollmann. *Computer Security, Second Edition.* Chichester: John Wiley & Sons, 2006.

[254] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92. ACM Press, New York, NY, USA, 1998.

[255] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *IETF RFC 3280*, (URL: http://www.ietf.org/rfc/rfc3280.txt), 2002.

[256] B. Beckles. Removing digital certificates from the end-user's experience of grid environments. *Proceedings of the UK e-Science All Hands Meeting,*
URL: http://www.allhands.org.uk/2004/submissions/papers/250.pdf, 2004.

[257] B. Beckles. Re-factoring grid computing for usability. *Proceedings of the UK e-Science All Hands Meeting, URL: http://www.allhands.org.uk/2005/proceedings/papers/565.pdf*, 2005.

[258] J. F. Monin and M. G. Hinchey. *Understanding formal methods*. Springer-Verlag New York, Inc., 2001.