Exploiting the structure of turbulence with tensor networks

Nikita Gourianov

Keble College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

Abstract

Turbulence is among the most important unsolved problems of physics. Its numerical solution is hindered by the extreme number of computational variables needed to accurately resolve the broad range of length-scales that become relevant during turbulent flow. It turns out that a similar challenge appears in a completely different branch of physics. Quantum many-body systems are described by elements of a vector space whose dimension grows exponentially with the number of particles, making direct simulation infeasible. However, the actual information contained in realistic quantum states is typically only polynomially large in the number of particles, which in principle makes it possible to represent them using a polynomial number of parameters. Tensor network methods do precisely this for quantum systems with local interactions by removing unrealised long-distance correlations from the solution space, which in turn enables accurate simulation of otherwise intractable quantum systems. In this thesis, a simple tensor network formalism known as the matrix product state (MPS) ansatz is transferred from quantum physics onto fluid dynamics and used to numerically examine two paradigmatic turbulent flows: the 2D temporally decaying jet and 3D collapse of the Taylor-Green vortex. We find both flows to be structured according to the classical, scale-local view of turbulence, where flow features of disparate scales are largely uncorrelated. We eliminate these unrealistic interscale correlations from the solution space through our MPS encoding of the velocity field, and then formulate a MPS algorithm for simulating turbulence. With this algorithm, we find that the incompressible Navier-Stokes equations can be accurately solved even when reducing the number of computational parameters by more than one order of magnitude compared to traditional direct numerical simulation. The outlook is threefold. Further work towards harnessing the power of tensor networks for turbulence simulation holds the promise of computational fluid dynamics calculations that are yet inconceivable in scale. Moreover, the close connection that our MPS algorithm has to quantum physics points towards the exciting prospect of solving the Navier-Stokes equations on

a quantum computer. Finally, from a theoretical standpoint, this work also lays the foundations for studying the structures of turbulence using tensor network theory. One topic of particular interest here is what tensor network geometry is most appropriate for turbulence simulations and why. Answering this question will illuminate the structure of turbulence from a completely new angle, and perhaps help unravel the old riddle that is predicting the dynamics of turbulent flows.

Acknowledgements

This thesis is the final piece of work of my doctoral degree in atomic and laser physics, done at Keble College, University of Oxford during the period of October 2018 – March 2022. The journey was challenging, but also incredibly giving in terms of what I learned and experienced. I am thankful for all the important lessons in mathematics, computer science and quantum physics that I received from my brilliant Oxford research colleagues. A few names that spring to mind are Michael Lubasch, Paolo Rosson, Jonathan Coulthard, Quincy van den Berg, Joseph Tindall and Frank Schlawin. I also have incredibly grateful to my collaborators in Bath and Pittsburgh, namely Peyman Givi, Sergey Dolgov and Hessam Babaee, who taught me intricacies of numerical optimisation and practically all that I know about turbulence. I am especially grateful to Michael and Peyman, my friends. I thank my supervisor Dieter Jaksch for providing insights and ideas and his insistence on high standards. Last, but not least, I thank my thesis examiners Andrew Daley and Gianluca Gregori for a great viva that significantly improved the quality of this thesis.

This degree would not have been a success without the strength I received from friends and family, both inside Oxford and beyond. I would like to particularly thank my friends in Oxford University Judo Club with whom I partook in the beautiful and exciting sport that is Judo. My friends from Keble are also owed gratitude for their generous sharing of forbidden knowledge and black humour. I am also grateful to my family, particularly my mother Marina, for their support and patience. Lastly, thank you, Eilish. I do not know where I would be without your love and kindness.

Contents

List of figures

Chapter 1

Introduction

Many decades ago, Richard Feynman remarked "turbulence is the last, great unsolved problem of classical physics" [1]. Being able to model the chaotic motion of fluids known as turbulence is of paramount importance to understanding nature, as such flow is ubiquitous and can be found in the atmosphere, the oceans, rivers, stars, and in both the wake of a car as well as inside the combustion chamber of its engine. However, modelling turbulence accurately requires overcoming the seemingly insurmountable challenge of solving the Navier-Stokes equations at high Reynolds numbers (Re). After two centuries of intense effort, it has become quite clear that purely analytical approaches are unlikely to be sufficient. The rise of modern computing in the mid 20th century permitted numerical solution of these equations, which initially resulted in a rapid burst of progress. But by now it is becoming increasingly obvious that accurate simulation of realistic turbulence is not computationally feasible with the current approaches of computational fluid dynamics (CFD) [2]. Feynman's remark still rings true today.

The central difficulty in simulating turbulence is rooted in its multiscale nature: high Re flows contain mutually interacting eddies across an extremely broad range of length and time scales [3, 4]. These spatial scales range from the largest size of the energy containing eddies ℓ (i.e. the integral scale), to the smallest one, known as the Kolmogorov microscale η, and the separation between these scales is $\ell/\eta \sim \mathrm{Re}^{3/4}$ according to phenomenological theory [5]. The direct numerical simulation (DNS) of turbulence therefore requires the number of gridpoints M to scale with Reynolds number as $M \sim (\ell/\eta)^K \sim \mathrm{Re}^{3K/4}$ for all the scales in K-dimensional flow to be resolved. This is a severe scaling which renders DNS infeasible for the scale of Reynolds numbers

encountered in realistic applications. Consider the example in [6]. A transport aircraft is flying at a cruise speed of $u = 250\text{m/s}$, and altitude of 10^4m at which the kinematic viscosity of air is about $\nu = 3.5 \cdot 10^{-5}\text{m}^2\text{s}^{-1}$. If the plane's chord length is $\ell = 5\text{m}$, the Reynolds number of the airflow around its wing is about

$$\text{Re} = \frac{u\ell}{\nu} = \frac{250\text{m}\,\text{s}^{-1} \cdot 5\text{m}}{3.5 \cdot 10^{-5}\text{m}^2\text{s}^{-1}} \approx 10^6. \tag{1.1}$$

DNS of the flow around the wing then requires $M \approx (10\ell/\eta)^3 \approx 10^3 \cdot \text{Re}^{9/4} \approx 10^{16}$ spatial gridpoints, which means each second must be discretised by about 10^6 temporal gridpoints for numerical stability to be maintained. This means that computing just a single second of flight time involves operating upon the order of 10^{22} floating point numbers, which would take thousands of years even on a 1 teraflop computer!

A broad range of methods have been created in an attempt to remedy this problem. One such method is that of adaptive mesh refinement [7], which works by allocating many gridpoints to areas in the computational domain where the fluid is highly turbulent, and few to the places where the fluid is more laminar. This can result in computational savings when the turbulence is concentrated in only certain areas of the domain (e.g. around the trailing edge of an airfoil). Another approach is based upon the Reynolds decomposition, which is the idea of decomposing the instantaneous quantities into their time-averaged and fluctuating components. After doing this, the Navier-Stokes equations can be time-averaged to produce the Reynolds averaged Navier-Stokes (RANS) equations, which couple mean flow quantities with a turbulence term. Modelling the turbulence term then allows one to solve the RANS equations for the mean flow and thus understand the time-averaged properties of the flow. RANS solvers are widely used in industry due to their modest cost requirements, but struggle with accuracy [8]. An important method that is often used to deal with multiphase flows is that of Lagrangian particle tracking [9]. It functions by repeatedly simulating the trajectory of individual particles (e.g. bubbles, dust, spray) embedded in some turbulent flow field, and then statistically analysing the results to characterise the dynamics of the particles within the flow field.

The currently most prominent method (except DNS) is arguably that of large eddy simulation (LES) [10]. LES was created about half a century ago and works by only resolving the flow down to some intermediate scale whilst modelling the remaining "subgrid-scales". However, since the general coupling between the unresolved and resolved scales is not fully understood, the currently available LES models are of

limited accuracy and reliability [8, 11]. Both DNS and LES are *scale-resolving* approaches, where increasing the number of numerical variables (e.g. grid points) translates to resolving finer and finer scales. And, while DNS and LES together arguably form the backbone of modern turbulence simulation [8], neither of them are able to reliably capture turbulence. Simulating all the scales between ℓ and η results in an unacceptable computational cost (DNS), but restricting the set of scales under consideration undermines the accuracy of the simulations (LES).

Resolving all the scales of turbulence thus requires an extreme number of computational variables, making direct simulation of turbulence computationally infeasible. As it turns out, a similar challenge precludes exact simulation of quantum many-body physics.

Being able to accurately simulate quantum many-body systems is important for understanding how the microscopic dynamics of quantum particles lead to macroscopic phenomena (such as high-T_c superconductivity), and as such the quantum many-body problem lies at the core of modern condensed matter physics. This problem essentially boils down to solving the Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}\left|\Psi(t)\right\rangle = H\left|\Psi(t)\right\rangle, \tag{1.2}$$

for a general N-particle Hamiltonian H and quantum state

$$\left|\Psi(t)\right\rangle = \sum_{i_1 i_2 \ldots i_N} C_{i_1 i_2 \ldots i_N}(t)\left|i_1\right\rangle\left|i_2\right\rangle \ldots \left|i_N\right\rangle. \tag{1.3}$$

The amplitude of the basis states constituting $|\Psi\rangle$ is given by the complex (and exponentially large in N) tensor C, with the state being normalised such that $\langle\Psi|\Psi\rangle = 1$, and $|\Psi\rangle$ itself lives in the N-particle Hilbert space $\mathcal{H}$. This Hilbert space is composed of the Hilbert spaces of the N individual particles in the form of

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \cdots \otimes \mathcal{H}_N, \tag{1.4}$$

with $|i_k\rangle \in \mathcal{H}_k$ being a set of orthonormal basis states that span $\mathcal{H}_k$. Equation (1.4) implies the dimensionality of $\mathcal{H}$ explodes exponentially into $|\mathcal{H}| = |\mathcal{H}_1| \cdot |\mathcal{H}_2| \ldots |\mathcal{H}_N|$ which in turn makes $|\Psi\rangle$ an exponentially large vector. This makes it infeasible to explicitly store, let alone time-evolve, $|\Psi\rangle$. For example, $\mathcal{H}$ of a magnetic system of N spin-1/2 particles is spanned by $|\mathcal{H}| = 2^N$ basis states. Solving this system within the entirety of $\mathcal{H}$ through e.g. exact diagonalisation of H can in the worst case carry

an $\sim 8^N$ computational cost, making this algorithm impossible to run on a desktop machine beyond just $N \approx 20$ spins.

However, the immensity of the Hilbert space is somewhat of an illusion. While *in theory* quantum many-body states live in an $\exp(N)$-sized Hilbert space, *in practice* often just a tiny sub-manifold of the Hilbert space is actually accessible during realistic time-evolution, when starting from some fiducial state (such as a product state or vaccuum state). Indeed, in [12] it is rigorously shown that only $\text{poly}(N)$ parameters are ever needed to represent the realistically-occurring states[1] of locally-interacting systems. In other words, the real information present in these physical quantum many-body states is polynomially large, albeit the states are embedded in an exponentially-large space. This opens the possibility of efficiently simulating quantum many-body systems by restricting oneself to just the polynomially large solution space that encompasses the realistic states (note, however, that this does *not* mean all locally-interacting quantum many-body systems can be efficiently simulated on classical machines).

The most powerful way of targeting this realistic manifold is the quantum computer. In 1980 Yuri Manin [13], and later Richard Feynman in 1982 [14], conjectured that a quantum computer – a computing machine based upon qubits and quantum gates instead of bits and transistors – would be able to simulate the dynamics of locally-interacting quantum systems *exponentially efficiently* such that the computational cost scales as $\sim \text{poly}(N)$ instead of $\sim \exp(N)$. This is done by representing the wavefunction with the qubits of the quantum machine, and then time-evolving this wavefunction according to some time-dependent Hamiltonian realised through quantum gates (e.g. lasers on ion-based quantum computers). Seth Lloyd later showed the conjecture of Manin and Feynman to be true in his 1996 article [15], where he analytically demonstrated that any local quantum system can be efficiently simulated on a quantum computer. However, as of today it remains unclear when (if ever) a fully fault-tolerant quantum computer will be constructed [16].

This lack of quantum computers forces us to make do with schemes running on classical computers instead. One such method is that of *tensor networks*, where the wavefunction is encoded in the form of a tensor network ansatz and time-evolved exclusively within this representation. Tensor network methods exploit that the entanglement (a type of correlation) between quantum particles is often highly structured. In

[1] With "realistically-occurring states", we refer to quantum states that can be produced in experiments in $\text{poly}(N)$ time.

particular in quantum systems where the particles are only entangled locally [17], near-locally [18] or just weakly [19], all the realistically occurring states will be efficiently parameterised by tensor network ansätze such as matrix product states (MPSs) [20, 21], projected entangled pair states (PEPSs) [22, 23] or the multigrid renromalisation ansatz (MERA) [18], allowing such systems to be simulated at just $\sim \text{poly}(N)$ cost in place of $\sim \exp(N)$. This reduction in computational cost enabled by tensor networks has provided significant insights into antiferromagnetic materials [24], quantum phase transition dynamics [25], superconducting systems [26] and quantum chemistry [27].

While tensor networks were born in the field of quantum physics [21], they have by now migrated into other fields. This includes cosmology [28], applied numerical mathematics [29], computer science [30], machine learning [31] and even linguistics [32]. A good overview of these recent trends is provided in [33].

In general, tensor networks work well when correlations in the underlying data are of a *local* nature. Local in the sense that each section of the data is only strongly coupled to another *restricted* section, as opposed to the *whole remainder* of the data. The aforementioned quantum systems where the particles are locally entangled is only one such example of locality.

Locality (as a seemingly omnipresent concept of physics [34]) in fact also features prominently in fluid dynamics. In 1941 Kolmogorov postulated that at large Re, three-dimensional turbulent flows undergo an energy cascade which is scale-local in the sense that eddies of a given size predominately transfer energy to other eddies of similar sizes [5]. After decades of debate, strong evidence has been found in favour of this postulate [35–37]. Furthermore, two-dimensional flows also undergo an energy cascade affected by locality [38]. Thus, high Re turbulence is characterised by locality between length scales of flow similarly to how interactions between quantum particles are local.

If the scale-local nature of interactions in turbulent flows leads to the various length scales being locally correlated too, it would in turn imply that turbulent flows are of similar structure to quantum many-body systems where the particles are locally entangled. Since such (quantum) locality can be efficiently exploited using tensor networks (as previously discussed), it is natural to ask whether the tensor network approach can also be used to exploit the scale-locality of turbulence. This is the central question of this thesis.

1.1 Thesis structure

The manifestation of locality in quantum many-body physics and fluid dynamics is considered in Chapter 2. The chapter begins with an overall introduction to the principle of locality in physics and how its presence can be exploited to reduce the dimensionality of a system. Then it is specifically explained how quantum systems get structured according to *area laws of entanglement* when the constituent particles are strictly locally correlated. Afterwards it is argued that the scale-locality of turbulence should also impose structure on turbulent fluid fields. This structure is more closely investigated by measuring the amount of interscale correlations found in two paradigmatic flows: a 2D Kelvin-Helmholtz instability and the 3D Taylor-Green vortex. The chapter finishes with a discussion on how the scale-locality of turbulence can be exploited with MPSs.

Chapter 3 introduces tensor networks in general and the MPS ansatz in particular. Topics include tensor algebra, how tensor networks can be used for data compression, what MPSs are and how they can be produced and various other aspects of MPSs that are relevant to this thesis. The operators used for manipulating MPSs, matrix product operators, are also covered. The mathematical machinery outlined here is actively used in the subsequent chapters of the thesis.

Chapter 4 presents a framework for encoding turbulence with MPSs. The first two sections deal with how scalar functions and vector fields can be represented in MPS form. Then a fine-graining algorithm is presented for prolonging vector fields onto finer grids, followed by an explanation of how finite difference matrices can be encoded as matrix product operators, and then finally it is explained how to calculate the Hadamard (i.e., elementwise) product of two MPSs. Reading Chapter 4 is important to both mathematically understand how the MPS ansatz is well-adapted to the multiscale nature of turbulence, and to follow the subsequent derivation of the MPS-based fluid simulation algorithm in Chapter 5.

Our MPS scheme for simulating fluid dynamics is derived in Chapter 5. The chapter begins by reformulating the Navier-Stokes equations into a cost function to be minimised over the MPS manifold. Then a Runge-Kutta time-marching scheme for minimising this cost function is derived, followed by a detailed explanation on how this time-marching scheme can be implemented in practice. The computational complexity of our MPS algorithm is also discussed in detail. Chapter 5 finishes with a remark on

how the MPS algorithm can be easily adjusted to solve Burgers' equation in place of the Navier-Stokes equations.

Burgers' equation is an *analytically solvable* simplification of the full Navier-Stokes equations and therefore present an excellent toy-problem on which to evaluate our MPS algorithm. In this spirit, Chapter 6 revolves around the simulation of 1D shock-waves evolving under Burgers' equation. The chapter begins with a general introduction to the Burgers' equation and how it can be analytically solved. Burgers' equation is known for generating shock-waves at large Reynolds numbers, which are very difficult for DNS to handle. In the middle of Chapter 6 it is shown analytically that these shock-waves are easily captured by the MPS representation, and afterwards these analytical results are confirmed numerically in the final parts of the chapter.

In Chapter 7 the MPS algorithm is used to simulate the two paradigmatic flows of Chapter 2 at high Reynolds number. The results of these simulations are presented and discussed, and they indicate that the MPS algorithm is capable of accurately resolving the turbulence of both the Kelvin-Helmholtz instability and the Taylor-Green vortex using only a small fraction of the variables required by DNS. It is also considered what the discarded variables represent and why they are unnecessary for maintaining the accuracy of the simulations.

We discuss alternative tensor networks for simulating turbulence in Chapter 8. This includes various geometries that are better than MPSs at capturing correlation between more distant length scales, and ansätze which are better suited to capture the structure of interscale correlations we have observed during our investigation of 3D turbulence. A quantum circuits representation of turbulence is also briefly mentioned.

Chapter 9 concludes our investigation on the utility of using tensor networks to exploit turbulence structure. The main findings are summarised and it is underlined how they move the fields of fluid dynamics and tensor networks forward. Suggestions for future work are made.

1.2 Publications

The central results of this thesis are published in "N. Gourianov, M. Lubasch, S. Dolgov, Q. Y. van den Berg, H. Babaee, P. Givi, M. Kiffner and D. Jaksch; A

quantum-inspired approach to exploit turbulence structures; *Nature Computational Science*, 2022". Particularly Chapters 2, 4, 5 and 7 are in large parts based on this paper.

Additionally, N. Gourianov during his D. Phil. also significantly contributed to the manuscript "P. Secular, N. Gourianov, M. Lubasch, S. Dolgov, S. R. Clark, D. Jaksch; Parallel time-dependent variational principle algorithm for matrix product states; *Physical Review B*, 2020".

Chapter 2

Locality in quantum physics and fluid dynamics

2.1 Introduction: principle of locality

A central facet of modern physics is the principle of locality. Articulated about two centuries ago [34], it posits that any interaction between particles must be mediated by the space separating them. In stark contrast to the old Newtonian ideas of *instant force at a distance* (e.g. Newton's law of gravitation), the principle of locality both guided Maxwell in his formulation of the laws of electrodynamics and turned out to be a key axiom in Einstein's theories of relativity.

The presence of locality in a system induces structure ("structure" in the sense that the system is governed by a set of constraints). Consider for instance the theory of special relativity, where locality demands that no signal can travel faster than the speed of light in vacuum (as otherwise we would be dealing with instantaneous action): this restriction imposes a causal structure upon the universe that forbids any relationship between events that are sufficiently separated in spacetime.

Locality also features prominently in both quantum theory and fluid dynamics, albeit in a different way from special relativity. In special relativity, one must assume the presence of locality for the causal structure of spacetime to be maintained, as otherwise, if information could travel faster than the speed of light, it would become possible for effects to occur *before* their causes. However, locality in entanglement and the scale-locality of turbulence instead *emerge* from the equations of motion,

the Schrödinger equation and the Navier-Stokes equations, respectively. Though this difference does not change that the presence of locality in a system always leads to structure.

Because the structure of a system by definition limits its dynamics, it also reduces the degrees of freedom needed to describe the system. That is, the solution space within which the states of the system lie is *smaller* than it would otherwise be, and this can readily be exploited for simulation by reducing the computational resources required to both represent the states and time-evolve them. In this way, algorithms that account for locality can allow for cheaper simulations than more naive schemes that do not.

2.1.1 Chapter structure

This chapter explores locality-induced structures in quantum many-body physics and fluid dynamics, and how they can be numerically exploited. Sections 2.2 and 2.3 discuss how quantum many-body systems become structured according to so-called *area laws* whenever the entanglement between particles is local. Tensor network methods were specifically designed to handle such quantum systems. Moving on, the scale-local nature of turbulent flows is introduced in Section 2.4. To investigate whether scale-locality reflects a structure that tensor networks can exploit, in Sections 2.5, 2.6 and 2.7 we investigate the interscale correlations of two paradigmatic turbulent flows using methods of quantum information theory. The results of these investigations imply that tensor networks can be used to simulate fluid flows using vastly fewer variables than DNS (i.e., within a data-compressed format), as is discussed in Section 2.8. The chapter is concluded in Section 2.9.

2.2 The Schmidt decomposition in quantum theory

In quantum many-body physics, the Schmidt decomposition is often used for studying the entanglement – i.e. quantum correlation – between different sections of quantum systems [17, 20, 21]. The Schmidt decomposition is mathematically equivalent to performing an exact singular value decomposition (SVD) [39] of a complex vector. For

example, assume that we have an N-particle pure-state

$$|\Psi\rangle = \sum_{i_1 i_2 \ldots i_N} C_{i_1 i_2 \ldots i_N} |i_1\rangle |i_2\rangle \ldots |i_N\rangle , \tag{2.1}$$

which lives in the N-particle Hilbert space[1] $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \cdots \otimes \mathcal{H}_N$, with $|i_k\rangle \in \mathcal{H}_k$. The amplitude of the basis states constituting $|\Psi\rangle$ is given by the complex (and exponentially large in N) tensor C, with the state being normalised such that $\langle\Psi|\Psi\rangle = 1$. If we wish to study the entanglement between the set of particles 1 to n and the remaining particles $n+1$ to N, we can bipartition the Hilbert space between sub-system A consisting of particles $1 \ldots n$ and the complementing sub-system B of particles $n+1 \ldots N$ as

$$\mathcal{H} = \mathcal{H}^{\mathrm{A}} \otimes \mathcal{H}^{\mathrm{B}}, \tag{2.2}$$

with any state $|\Psi^A\rangle$ of the set of particles $1 \ldots n$ living in $\mathcal{H}^{\mathrm{A}}$, while the corresponding state $|\Psi^B\rangle$ of the particles $n+1 \ldots N$ lives in $\mathcal{H}^{\mathrm{B}}$. By reshaping C into a matrix where the rows are associated with the states of the particles of A and the columns with those of B, and then performing an exact SVD between these rows and columns, $|\Psi\rangle$ will be rewritten as

$$|\Psi\rangle = \sum_{\alpha=1}^{\Gamma} \lambda_\alpha |\psi_\alpha^A\rangle |\psi_\alpha^B\rangle , \tag{2.3}$$

with the singular values (or Schmidt values) λ_α being real, non-negative scalars sorted in descending order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\Gamma$, $|\psi_\alpha^A\rangle$, $|\psi_\alpha^B\rangle$ being orthonormal states such that $\langle\psi_\alpha^A|\psi_\beta^A\rangle = \langle\psi_\alpha^B|\psi_\beta^B\rangle = \delta_{\alpha\beta}$ where $\delta_{\alpha\beta}$ is the Kronecker delta and $\Gamma = \min\{|\mathcal{H}^{\mathrm{A}}|, |\mathcal{H}^{\mathrm{B}}|\}$ being the Schmidt number. We have now illustrated the Schmidt decomposition by putting $|\Psi\rangle$ into the Schmidt-form of Equation (2.3).

The Schmidt spectrum λ_α determines the entanglement between sub-systems A and B. This entanglement is traditionally measured in units of von Neumann entanglement entropy

$$S_{AB} = -\sum_{\alpha=1}^{\Gamma} \lambda_\alpha^2 \log_2 \lambda_\alpha^2. \tag{2.4}$$

It is interesting to note that the von Neumann Entropy is at its maximum if $\lambda_\alpha^2 = 1/\Gamma$, at which point the entropy equals $\log_2 \Gamma$. Or equivalently,

$$S_{AB} \leq \log_2 \Gamma, \tag{2.5}$$

[1]Note that we here only refer to the quantum state and Hilbert space as "N-particle" for the sake of simplicity; this Hilbert space could just as well represent the states of a quantum system of N lattice sites, like the Hubbard model, where each lattice site can accommodate multiple particles.

which implies the maximum entanglement is limited by the volume (i.e. dimensionality) of the smallest of the two sub-system A and B. Thus, in general, the maximum possible entanglement entropy a many-body quantum system can contain scales with its volume.

The central benefit of Schmidt decomposing $|\Psi\rangle$ is that it allows us to systematically approximate $|\Psi\rangle$ by truncating the least important singular values. Remember, the singular values λ_α are sorted in descending order, while the states $|\psi_\alpha^A\rangle, |\psi_\alpha^B\rangle$ are orthonormal; this means that if we were to truncate the sum from Γ down to some scalar $\chi \leq \Gamma$, we would only keep the χ most important orthonormal basis functions, weighted by their respective singular values. Thus $|\Psi\rangle$ would be approximated as

$$|\Psi\rangle \approx |\Phi\rangle = \sum_{\alpha=1}^{\chi} \lambda_\alpha |\psi_\alpha^A\rangle |\psi_\alpha^B\rangle, \tag{2.6}$$

such that the squared L2-norm error ϵ^2 is minimal and equals

$$\begin{aligned} \epsilon^2 &= \big\| |\Psi\rangle - |\Phi\rangle \big\|_2^2 = \langle\Psi|\Psi\rangle + \langle\Phi|\Phi\rangle - (\langle\Psi|\Phi\rangle + \langle\Phi|\Psi\rangle) \\ &= \sum_{\alpha,\beta=1}^{\Gamma} \lambda_\alpha\lambda_\beta \langle\psi_\alpha^A\psi_\alpha^B|\psi_\beta^A\psi_\beta^B\rangle + \sum_{\alpha,\beta=1}^{\chi} \lambda_\alpha\lambda_\beta \langle\psi_\alpha^A\psi_\alpha^B|\psi_\beta^A\psi_\beta^B\rangle \\ &- \sum_{\alpha=1}^{\Gamma}\sum_{\beta=1}^{\chi} \lambda_\alpha\lambda_\beta \left(\langle\psi_\alpha^A\psi_\alpha^B|\psi_\beta^A\psi_\beta^B\rangle + \langle\psi_\beta^A\psi_\beta^B|\psi_\alpha^A\psi_\alpha^B\rangle\right) = \sum_{\alpha=\chi+1}^{\Gamma} \lambda_\alpha^2. \end{aligned} \tag{2.7}$$

2.3 Area law of entanglement

Remarkably, nature itself sometimes truncates these singular values. Assume now the sub-systems A and B of Equation (2.4) are separated by a continuous boundary as illustrated in Figure 2.1. Consider quantum systems wherein the interactions are local and a finite gap exists between the energies of the ground state and excited states. As it turns out, these systems' low-energy states will normally be characterised by local entanglement, where particles close to each other entangle while the correlations between far-away particles is exponentially small [17]. This means that the singular values in Equation (2.4) which are associated with long-range entangled modes must be approximately zero (truncated by nature), which in turn implies that the entanglement entropy S_{AB} between the two sub-systems scales not with their volume, but rather with the area separating them. This phenomena is what is known as the area law of entanglement [17, 19, 40, 41].

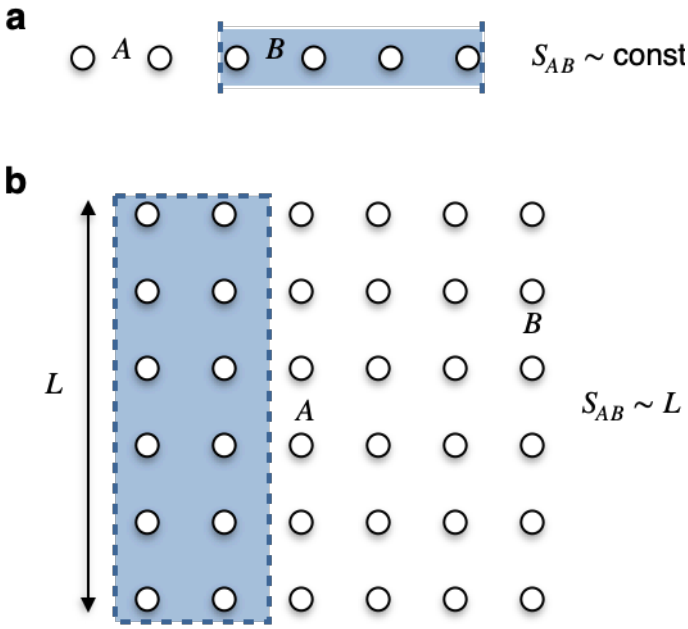

Figure 2.1: **An illustration of the area law for 1D and 2D quantum lattice systems**. In area law following systems, the bipartite von Neumann entanglement entropy scales with the area separating the sub-systems instead of their volumes. For the 1D system of **a**, this means that the entanglement entropy S_{AB} between sub-systems A and B saturates towards a constant as the lattice lengthens. However, if the 2D quantum lattice of **b** starts expanding e.g. vertically, such that the boundary between sub-systems A and B grows, then the entanglement will grow linearly with L.

In a naive simulation of an N-particle quantum many-body system, for instance through exact diagonalisation, the solution space equals the exp(N)-sized Hilbert space. Such a simulation will exactly resolve (up to machine precision) all of the singular values at all bipartitions of the quantum system. But if nature truncates many of the singular values, then why should they at all be represented during simulation? The answer is that they should not – resolving them is a waste of computational resources. Hence the locality in entanglement of quantum systems leads to a structure in the quantum state that restricts the number of singular values required to accurately represent the state. One practical way of discarding these unnecessary singular values during numerical simulation is by representing the state as a *tensor network* and simulating it entirely within such a form, as is explained in detail in the following Chapter 3.

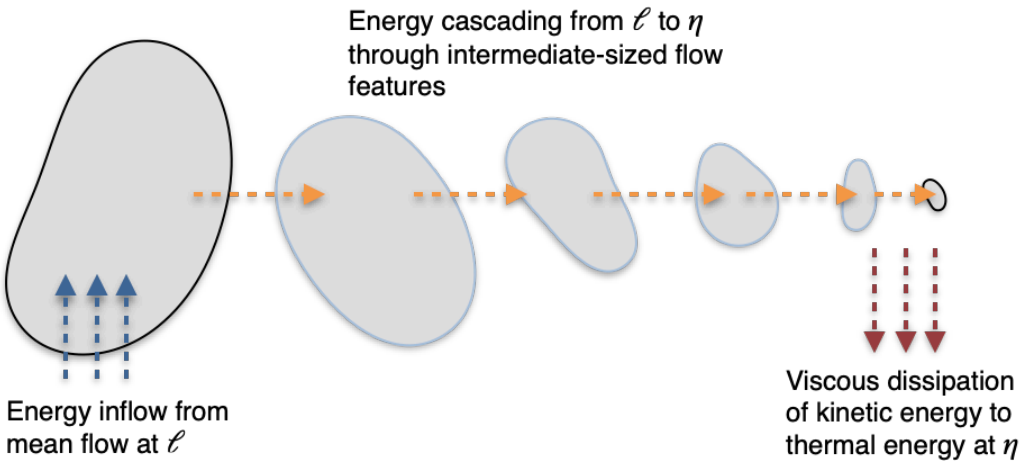

Figure 2.2: **The turbulent energy cascade.** The energy cascade in three-dimensional space is caricatured in this figure. ℓ and η here denote respectively the integral and Kolmogorov length scales. These scale are related with the Reynolds number as $\ell/\eta \sim \mathrm{Re}^{3/4}$.

2.4 Scale-locality of the turbulent cascade

While in quantum physics the locality is between particles, in this section we argue that in fluid dynamics locality appears between *length scales of flow*.

The poem "Big whirls have little whirls that feed on velocity, and little whirls have lesser whirls and so on to viscosity" was written over a century ago by Lewis Richardson [42], yet it still correctly encapsulates how energy is transferred and dissipated in turbulent flows. The first major analytical explanation of this phenomena is Kolmogorov's 1941 theory of the turbulent energy cascade [5], which says that energy flows from the integral spatial and temporal scales to the Kolmogorov microscales (where the energy is then dissipated) in a *scale-local* fashion through the intermediate scales. Scale-local in the sense that flow features at a given scale only interact with flow features that are roughly of the same scale, which forces the energy to cascade through a succession of scales, as illustrated in Figure 2.2, before reaching the dissipative microscale.

The interscale dynamics of turbulence have historically been studied by simply equating scale with wavenumber [37]. To understand how, however, we first need to define the equations of motion. If $\boldsymbol{V}(t,\boldsymbol{r}) = \sum_{i=1}^{K} \boldsymbol{e}_i u_i(t,\boldsymbol{r})$ is the K-dimensional velocity field at time t and position $\boldsymbol{r}$ within some box of length L_{box}, then the time-evolution of $\boldsymbol{V}(t,\boldsymbol{r})$ is governed by the Navier-Stokes equations. They assume the fluid is Newtonian and

viscous, and consist of conservation equations for momentum and mass along with an equation of state that couples the temperature, density and pressure of the fluid. Assuming the fluid is water-like, i.e. of constant density, will simplify the equations without removing their turbulent properties. Doing this results in the *incompressible* Navier-Stokes equations

$$\begin{aligned} &\nabla.\boldsymbol{V} = 0, \\ &\frac{\partial \boldsymbol{V}}{\partial t} + \big(\boldsymbol{V}.\nabla\big)\boldsymbol{V} = -\nabla p + \nu\nabla^2\boldsymbol{V}\,, \end{aligned} \tag{2.8}$$

where $p(t, \boldsymbol{r})$ is the pressure and ν the kinematic viscosity (which is constant throughout space and time). Now, the Fourier transform of each component of $\boldsymbol{V}$ gives

$$\widehat{u}_i(t, \boldsymbol{k}) = \int u_i(\boldsymbol{r}, t) e^{\mathrm{i}\boldsymbol{k}.\boldsymbol{r}} \mathrm{d}\boldsymbol{r}, \tag{2.9}$$

with $\boldsymbol{k} = \sum_{i=1}^{K} \boldsymbol{e}_i k_i$ being the wavenumber. Assuming periodic boundary conditions, rewriting the velocity field as $u_i = \sum_{\boldsymbol{k}} \widehat{u}_i(\boldsymbol{k}) e^{\mathrm{i}\boldsymbol{k}.\boldsymbol{r}}$ and making use of the fact that the pressure is nothing more than a Lagrange multiplier for the incompressibility condition [43], in turn transforms Equation (2.8) into

$$\Big(\frac{\partial}{\partial t} + \nu|\boldsymbol{k}|^2\Big)\widehat{u}_i(t, \boldsymbol{k}) = -\mathrm{i}\sum_{j,m=1}^{K} k_m P_{ij}(\boldsymbol{k}) \sum_{\boldsymbol{k}=\boldsymbol{k}'+\boldsymbol{k}''} \widehat{u}_j(t, \boldsymbol{k}')\widehat{u}_m(t, \boldsymbol{k}''), \tag{2.10}$$

with the projector P_{ij} being

$$P_{ij}(\boldsymbol{k}) = \delta_{ij} - \frac{k_i k_j}{|\boldsymbol{k}|^2}, \tag{2.11}$$

where δ_{ij} is the Kronecker-delta. Equation (2.10) reveals that the velocity field at wavenumber $\boldsymbol{k}$ in principle depends upon all possible interactions between wavenumbers $\boldsymbol{k}'$ and $\boldsymbol{k}''$ as long as $\boldsymbol{k}' + \boldsymbol{k}'' - \boldsymbol{k} = 0$. These so-called *triadic wavenumber interactions* offer a perspective on the interscale dynamics occurring within turbulence and have therefore been the focal point in the theoretical study of the scale-locality of turbulence.

A central question is how close $\boldsymbol{k}$, $\boldsymbol{k}'$ and $\boldsymbol{k}''$ are to each other when their respective modes exchange energy amongst themselves [37]. To this end, multiplying with the complex-conjugated $\widehat{u}_i^*(t, \boldsymbol{k})$ on both sides of Equation (2.10) will result in an expression for the time-evolution of the energy spectrum $E(t, \boldsymbol{k}) = \frac{1}{2}\,|\widehat{u}_i(t, \boldsymbol{k})|^2$:

$$\Big(\frac{\partial}{\partial t} + 2\nu|\boldsymbol{k}|^2\Big)E(t, \boldsymbol{k}) = T(t, \boldsymbol{k}). \tag{2.12}$$

Here the interscale energy transfer function $T(t, \boldsymbol{k})$ completely determines the coupling between disparate scales and it equals

$$T(t, \boldsymbol{k}) = \sum_{j,m=1}^{K} k_m P_{ij}(|\boldsymbol{k}|) \sum_{\boldsymbol{k}=\boldsymbol{k}'+\boldsymbol{k}''} \mathrm{Im}\left(\widehat{u}_i^*(t, \boldsymbol{k})\widehat{u}_j(t, \boldsymbol{k}')\widehat{u}_m(t, \boldsymbol{k}'')\right), \tag{2.13}$$

with $\mathrm{Im}(\cdot)$ being the imaginary part of a complex quantity.

The nature of $T(t, \boldsymbol{k})$ has been debated for over half a century. Kolmogorov's scale-local view of 3D turbulence supposes that energy is exchanged only when $\boldsymbol{k}$, $\boldsymbol{k}'$ and $\boldsymbol{k}''$ are close to each other, a view endorsed by the subsequent theories of Obukhov, Onsager and Heisenberg [44–46]. They proposed that pairs of modes at similar wavenumbers $\boldsymbol{k}'$ and $\boldsymbol{k}''$ transfer energy to double-sized wavenumbers $\boldsymbol{k} \approx 2\boldsymbol{k}' \approx 2\boldsymbol{k}''$, thus driving the energy cascade by successively transferring energy from large to small scales. This picture was later refined through detailed calculations by Kraichnan, who found that the energy transfer between modes at disparate wavenumbers decays as a *power-law*. To see this, we define the parameter

$$s(|\boldsymbol{k}|, |\boldsymbol{k}'|, |\boldsymbol{k}''|) = \frac{\max(|\boldsymbol{k}|, |\boldsymbol{k}'|, |\boldsymbol{k}''|)}{\min(|\boldsymbol{k}|, |\boldsymbol{k}'|, |\boldsymbol{k}''|)}, \tag{2.14}$$

which gives the disparity between the wavenumbers. Kraichnan showed [47, 48] that the contribution to the energy cascade flux across $|\boldsymbol{k}|$ from disparate modes decays asymptotically with the disparity as $\sim s^{-4/3}$. The explosion of computing power in the late 1980s permitted numeric evaluation of these theories, but the simulations turned out to seemingly claim evidence both in favour [49, 50] and against [51] scale-locality. In the late 2000s, rigorous studies came out that aimed to settle the controversy in favour of the scale-local $\sim s^{-4/3}$ view [35–37].

The dominant physical mechanisms behind the three-dimensional energy cascade are *vortex stretching* and *strain self-amplification*. With regards to the former, turbulent fluctuations tend to on average elongate vortical structures in incompressible flow [52], and when this happens the law of angular momentum conservation demands the vortices get thinner (in the sense of a decrease in their radii) and spin faster (akin to how a figure skater can increase her spin by drawing in her arms). This thinning energises larger wavenumbers, leading to a transfer of energy from large scales to small. Vortex stretching has traditionally been thought to be the main mechanism for transferring energy down the cascade, but it was recently revealed that strain self-amplification is in fact at least as important [53–55]. Strain self-amplification

occurs in regions of steep velocity gradients. There, slow-moving fluid is overtaken by faster fluid, which causes fluid elements to be squeezed into sheet-like bodies (akin to how a water-filled balloon thrown into a wall gets squeezed before bursting). This squeezing of fluid parcels serves to energise larger wavenumbers (just as the aforementioned vortex thinning does), hence helping drive the Kolmogorov energy cascade.

One might guess that two-dimensional turbulence is merely a simplification of the 3D problem. This is however incorrect, as both vortex stretching and strain self-amplification vanish in two dimensions [55]. This leaves two remaining mechanisms for transporting energy between scales that are not mentioned above (due to them being of little importance in 3D flow compared to vortex stretching and strain self-amplification), and not only do these two mechanisms *not* vanish in 2D, they in fact serve to make the dynamics of 2D flow completely different from 3D. The first of these processes is *vortex merging*, where small vortices of like rotation combine to form larger vortices, and then the larger vortices in turn combine with other similarly large vortices and so on until the resulting eddies become as large as is possible to fit into the domain. An important point is that the interaction between the voritces is scale-local in the sense that small vortices do not interact with large vortices (instead, they simply get swept along the large ones), but do indeed strongly interact and become deformed when colliding with similarly-sized vortices [56]. The second important process in 2D turbulence is that of *vortex thinning*. When small vortices are caught in larger-scale strain rates in the flow, the strain-rate tends to flatten ("thin") the 2D vortex. Kelvin's circulation theorem then demands that this leads to a decrease in the velocity around the vortex, which in turns also reduces its energy. But the energy does not dissipate, it rather transfers to the large scale flow features responsible for the strain-rate [38]. Furthermore, this mechanism is weakly scale-local [38]. Together, vortex thinning and merging serve to drive energy up from small scales of flow to larger ones, leading to what is known as the *inverse energy cascade* of two-dimensional flow. Thus while energy flows from large scales to small in 3D flow, it is the complete opposite in 2D.

We have now discussed the scale-locality of high Re flow. In conclusion, it is widely believed that the 3D Kolmogorov cascade is scale-local, while for 2D fluid dynamics, the evidence points towards scale-locality being present in a weaker form. Thus the interactions between the modes corresponding to wavenumbers $\boldsymbol{k}'$ and $\boldsymbol{k}''$ in the RHS of Eqs. (2.10) and (2.12) are scale-local in turbulence, though the exact nature of

these interscale interactions still remains an open research question [37]. But what about the low Re case?

At low Reynolds numbers, equivalent to large ν, the flow turns *laminar*. Laminar flow is a highly ordered collection of layers of fluid particles moving in the same direction without intermixing between the layers. As ν gets larger (and increasingly larger scales become subject to viscous dissipation), the RHS of Eqs. (2.10) and (2.12) shrink in amplitude compared to the viscous term, making the interscale interactions decrease in relative importance. In the limit of $\nu \to \infty$ we get the extreme case of unsteady Stokes flow where the RHS has vanished and the different length scales of flow have become completely decoupled from one another. Therefore, the Reynolds number controls the amount of interactions between different length scales of flow. At very low Re, these interactions are turned off and all different wavenumbers decouple, while at very high Re the turbulent and multiscale[2] flow contains (scale-local) interactions between the wavenumbers.

The scale-local nature of interactions in fluid dynamics is said to be reflected in a particular structure of turbulence. When excitations are transferred in a chaotic chain of steps from wavenumber to wavenumber, the conditions at the large scales of flow presumably uncouple from the dynamics at the small scales. This is the reasoning behind the famous *universality hypothesis of turbulence*, which states that at a sufficiently high Reynolds number, the dynamics of small-scale flow are universal and independent of the large scale flow [57, 58]. This universality hypothesis is a main motivator behind LES. Afterall, if the physics of the small scales are universal across turbulent flows, it ought to be possible to model them while resolving just the large scale flow.

In a sense, LES exploits the structure induced by locality to remove unnecessary variables (i.e., the large wavenumbers) from consideration. Namely, to do data-compression. Unfortunately however, nobody has yet been able to formulate a generally valid theory of turbulence and, as such, no single subgrid-scale LES model exists which is accurate for all flows. The structure of many-body quantum systems discussed in Section 2.3 is also imposed by locality, and numerical methods exist which actively exploit this locality to simulate the physics within vastly reduced (i.e. data-compressed) solution spaces. A prominent such method that has seen great

[2]Recall that at high Re the dissipative scale becomes small compared to the integral scale, which activates the many intermediate flow scales between the integral and dissipative scales.

success in quantum many-body physics is that of *tensor networks*, begging the question of whether they can also be used to exploit the scale-locality of turbulent flows.

2.5 A quantum-inspired scale decomposition

To understand the utility of tensor networks for CFD, we here study the structure of turbulence from a quantum information perspective. This entails, for the sake of convenience, using language and terminology taken from quantum theory even though we are dealing with classical physics.

We begin by investigating the interscale correlations in turbulence, this time not using a wavenumbers approach as in the previous Section 2.4, but rather the Schmidt (or singular value) decomposition [Section 2.2] to bipartition flow fields into coarse and fine length scales (though a wavenumbers-based investigation into the interscale correlations is also provided as a side-note in Appendix B).

The very first step is to discretise the computational domain. We discretise each spatial dimension by 2^N gridpoints, where N is a positive integer. In this way, the velocity field $\boldsymbol{V}(t, \boldsymbol{r})$ becomes discrete functions of position $\boldsymbol{r}_q$ ala

$$\boldsymbol{V}(t, \boldsymbol{r}_q) = \sum_{i=1}^{K} u_i(t, \boldsymbol{r}_q)\boldsymbol{e}_i, \tag{2.15}$$

where K is the spatial dimension and $\boldsymbol{e}_i$ are Cartesian unit vectors. This computational grid will now be systematically divided into subgrids by means of the Schmidt decomposition.

Consider a 1D system ($K = 1$) and scale all lengths with its spatial dimension L_{box}. Let us momentarily ignore time. The spatial domain $[0, 1]$ of the velocity u is discretised with N bits into 2^N gridpoints as $r_q = q/2^N$ with $q = 0, 1, \ldots, 2^N - 1$ (see Figure 2.3 for a $N = 5$ example). This grid can be bipartioned into coarse and fine subgrids. Let $n = 1, \ldots, N - 1$. Then, for a given n, the coarse subgrid comprises the points $X_k = k/2^n$ with $k = 0, \ldots, 2^n - 1$. The spacing between neighbouring points is 2^{-n} and this therefore defines a coarse length scale, as indicated by the red gridpoints in Figure 2.3. At each coarse grid point X_k a fine subgrid is attached with points $x_l = l/2^N$, $l = 0, 1, \ldots, 2^{(N-n)} - 1$, and adjacent points are separated by the fine length scale 2^{-N}, as the blue gridpoints in Figure 2.3. In this way, any point r_q of the

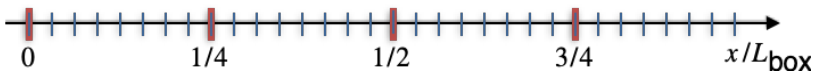

Figure 2.3: **A 1D grid**. A line of length L_{box} is here discretised using a Cartesian grid with 32 gridpoints. The subgrid structure when decomposing a function u according to Equation (2.16) for $n = 2, N = 5$ is illustrated here. The 4 gridpoints X_k of the coarse grid are here denoted by the red points. Each X_k has a fine subgrid x_k of 8 points attached to it, and these points are denoted in blue.

1D grid can be written as $r_q = X_k + x_l$. Next, the function values $u(r_q) = u(X_k + x_l)$ are arranged into a $2^n \times 2^{N-n}$ matrix where the rows and columns correspond to increments along the coarse and fine grids, respectively. Performing an un-truncated SVD on this matrix gives the exact Schmidt decomposition of $u(r_q)$ at bipartition n:

$$u(r_q) = \sum_{\alpha=1}^{d(n)} \lambda_\alpha R_\alpha(X_k) f_\alpha(x_l), \quad r_q = X_k + x_l, \tag{2.16}$$

where the Schmidt number $d(n) = \min(2^n, 2^{N-n})$ is maximal (due to the SVD being un-truncated), and the functions R_α and f_α obey the orthonormality condition of

$$\sum_k R_\alpha(t, X_k) R_\beta(t, X_k) = \sum_l f_\alpha(t, x_l) f_\beta(t, x_l) = \delta_{\alpha\beta}\,, \tag{2.17}$$

with $\delta_{\alpha\beta}$ being the Kronecker delta. Each product $R_\alpha f_\alpha$ is weighted by a Schmidt coefficient $\lambda_\alpha \geq 0$, and these coefficients appear in descending order $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_{d(n)}$.

The SVD can also be done in an approximate (truncated) manner by setting $d(n) = \chi < \min(2^n, 2^{N-n})$, resulting in

$$u(r_q) \approx v(r_q, \chi) = \sum_{\alpha=1}^{\chi} \lambda_\alpha R_\alpha(X_k) f_\alpha(x_l). \tag{2.18}$$

Decreasing the Schmidt number to χ results in only the χ largest singular values and their associated orthonormal basis functions being kept, ensuring that the error due to this approximation is minimal in the Frobenius norm (analogously to the quantum case in Section 2.2). The total error-squared ϵ^2 is

$$\epsilon^2 = \|u(r_q) - v(r_q, \chi)\|_2^2 = \sum_{\alpha=\chi+1}^{\Gamma^{\text{1-D}}(n)} \lambda_\alpha^2. \tag{2.19}$$

Truncating the Schmidt decomposition in this manner approximates u in an orthonormal time-dependent basis that evolves with the fluid flow to optimally capture spatially

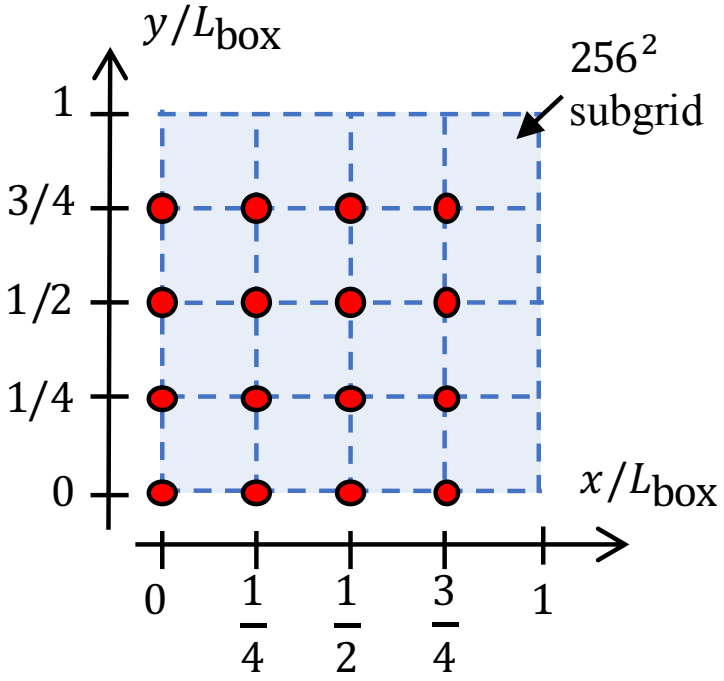

Figure 2.4: **A quadratic grid**. A square with edge length L_{box} is here discretised with a 1024×1024 grid. The subgrid structure when decomposing a function u_i according to Equation (2.21) for $n = 2, N = 10$ is illustrated here. Red dots are the 4×4 gridpoints $\boldsymbol{X}_k$ of the coarse grid. Each blue square attached to the $\boldsymbol{X}_k$ indicates the quadratic subgrid with the 256×256 gridpoints $\boldsymbol{x}_k$ of the fine grid.

correlated structures. This is in contrast to classical DNS (as implemented through e.g. finite difference or spectral methods), where the bases are chosen a priori and do not automatically adapt to any changes in the structure of the solution, though somewhat similar to the method of adaptive mesh refinement [7], with the caveat that adapative mesh refinement removes excess gridpoints from specific locations in the spatial domain, while truncating singular values removes excess basis functions which cover the whole spatial domain.

This procedure can be straightforwardly generalised by replacing bits with quarternaries (2D) or octals (3D), i.e., by replacing 1D line segments with squares (2D) or cubes (3D). The maximal possible Schmidt number at the n-th bipartition in K dimensions is in general given by

$$\Gamma^{K\text{-D}}(n) = \min(2^{Kn}, 2^{K(N-n)}). \tag{2.20}$$

Thus the maximal Schmidt number in 2D is $\Gamma^{2\text{-D}}(n) = \min(4^n, 4^{N-n})$ and $\Gamma^{3\text{-D}}(n) = \min(8^n, 8^{N-n})$ for 3D.

To demonstrate, we illustrate in Figure 2.4 the Schmidt decomposition for a 2D ($K = 2$) flow field. Each component u_i on this $2^N \times 2^N$ grid is decomposed into functions $R_{i,\alpha}$ and $f_{i,\alpha}$ on a coarse and a fine subgrid, respectively, giving

$$u_i(t, \boldsymbol{r}_q) = \sum_{\alpha=1}^{\Gamma^{2\text{-D}}(n)} \lambda_{i,\alpha}(t) R_{i,\alpha}(t, \boldsymbol{X}_k) f_{i,\alpha}(t, \boldsymbol{x}_l), \quad \boldsymbol{r}_q = \boldsymbol{X}_k + \boldsymbol{x}_l. \tag{2.21}$$

The positions $\boldsymbol{X}_k$ correspond to a quadratic grid with $2^n \times 2^n$ points (coarse grid), and $\boldsymbol{x}_l$ correspond to a fine subgrid with $2^{N-n} \times 2^{N-n}$ gridpoints. As in the 1D case, the functions $R_{i,\alpha}$ and $f_{i,\alpha}$ obey an orthonormality condition and their products are weighted by the associated Schmidt coefficient $\lambda_{i,\alpha}$.

2.6 Interscale correlations in turbulence

Let us now employ the Schmidt decomposition to examine interscale correlations within actual turbulence. To see how, consider the Schmidt spectrum $\lambda_\alpha(n)$ at a given bipartition n of the lattice. If all coefficients λ_α for $\alpha > 1$ were insignificantly small at this bipartition, then we would essentially be dealing with an uncorrelated product state[3] where there is no correlation between the length scales that are coarser and finer than n. While on the other hand if a large number of the coefficients were of large magnitude [e.g. $\lambda_\alpha(n)$ being a fat-tailed distribution in α], then the state will contain major amounts of correlation between the aforementioned coarse and fine scales.

We consider the Schmidt spectra of two paradigmatic turbulent flows: the 2D temporally decaying jet (TDJ) [59] and the 3D Taylor-Green vortex (TGV) [60]. The TDJ comprises a central jet flow in the x-direction that eventually collapses from the presence of Kelvin-Helmholtz instabilities in the jet's boundary layers. In the TGV, a single, large, ordered fluctuation collapses into a turbulent flurry of small scale structures. While these flows are simple to initialise and simulate, they still exhibit the core features of turbulence when the Reynolds number is high, making them an ideal starting point for our investigations into the interscale correlations in turbulent flows. To this end, DNS of these two flow cases is performed at high Reynolds numbers in the following Sections 2.6.1 and 2.6.2, and then the Schmidt spectra of the resulting DNS solutions are examined in Sections 2.6.3 and 2.6.4.

[3]Note that while the product state exhibits no interscale correlations, it is still highly correlated in space due to its fine grid dependence being repeated.

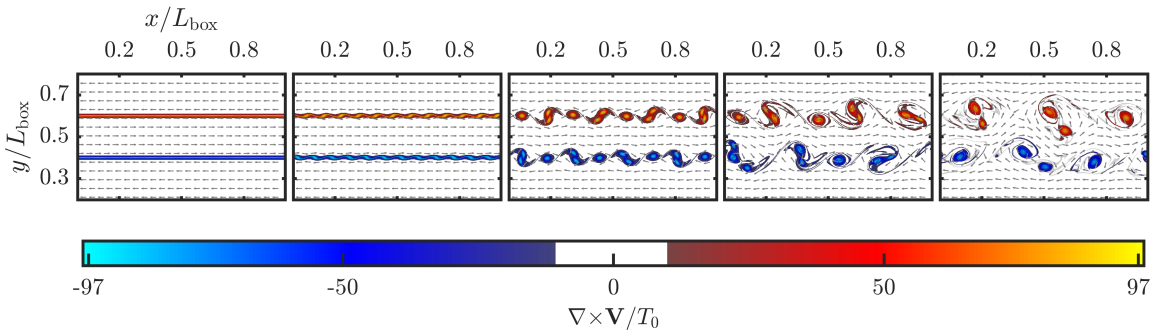

Figure 2.5: **DNS of the temporally developing jet flow**. Dynamical simulation of the incompressible Navier-Stokes equations in 2D for a planar jet streaming along x at Re $= 1000$. The five snapshots illustrate the velocity field at times $t/T_0 = 0, 0.25, 0.75, 1.25, 1.75$ (left to right), T_0. Red corresponds to positive vorticity (counter-clockwise flow) and blue to negative (clockwise). See Section 2.6.1 for the definitions of Re, L_{box} and T_0. The simulation was done on a 1024×1024 spatial grid.

2.6.1 Temporally developing jet flow

The 2D TDJ is a classical flow problem centered around the dynamics of a planar fluid-jet propagating through another fluid. When the jet is perturbed at a sufficiently high Reynolds number, these perturbation will grow until the vortex sheet rolls up and forms a turbulent shear layer of interacting vortical structures. These vortices pair up, orbit each other and merge into larger vortices repeatedly until the shear layers saturate and become unable to drive further pairings and mergers. As time goes on, the inverse energy cascade carries energy from small scales to large, resulting in a collection of increasingly fewer and larger eddies.

The set-up of the TDJ simulations is as follows. Consider a square with edge length L_{box} with periodic boundary conditions and the initial conditions

$$\boldsymbol{V}(t = 0, x, y) = \boldsymbol{J}(y) + \boldsymbol{D}(x, y)\,, \tag{2.22}$$

where $\boldsymbol{J}(y)$ is the initial jet profile

$$\boldsymbol{J}(y) = \boldsymbol{e}_1 \frac{u_0}{2}\left[\tanh\left(\frac{y - y_{\min}}{h}\right) - \tanh\left(\frac{y - y_{\max}}{h}\right) - 1\right] \tag{2.23}$$

with the streamwise direction along $\boldsymbol{e}_1$. u_0 is the magnitude of the velocity differential between the jet and its surroundings, $y_{\min}$ and $y_{\max}$ describe the extent of the jet and h is the initial thickness of the vortex sheet. These parameters define the time scale

$T_0 = L_{\text{box}}/u_0$ and Reynolds number $\text{Re} = u_0 h/\nu$. In this work, the TDJ simulations were performed for $t/T_0 \in [0, 2]$. The function

$$\boldsymbol{D} = \delta(\boldsymbol{e}_1 d_1 + \boldsymbol{e}_2 d_2) \tag{2.24}$$

in Equation (2.22) is a small disturbance of miscellaneous wave modes needed to initiate the roll-up of the jet that is statistically homogeneous along $\boldsymbol{e}_1$ and divergence free. $\boldsymbol{D}$ was in this work set to

$$\begin{aligned} d_1(x,y) &= 2\frac{L_{\text{box}}}{h^2}\left[(y-y_{\max})e^{-(y-y_{\max})^2/h^2} + (y-y_{\min})e^{-(y-y_{\min})^2/h^2}\right] \\ &\quad \times \left[\sin(8\pi x/L_{\text{box}}) + \sin(24\pi x/L_{\text{box}}) + \sin(6\pi x/L_{\text{box}})\right], \\ d_2(x,y) &= \pi\left[e^{-(y-y_{\max})^2/h^2} + e^{-(y-y_{\min})^2/h^2}\right] \\ &\quad \times \left[8\cos(8\pi x/L_{\text{box}}) + 24\cos(24\pi x/L_{\text{box}}) + 6\cos(6\pi x/L_{\text{box}})\right], \end{aligned} \tag{2.25}$$

with $\delta = u_0/(40A)$ and $A = \max_{x,y} \sqrt{d_1^2 + d_2^2}$. Scaling all lengths with L_{box}, velocities with u_0 and time with T_0, we set $y_{\min} = 0.4$, $y_{\max} = 0.6$, $h = 1/200$.

The temporal dynamics of the TDJ for the above set-up are illustrated in Figure 2.5 for $\text{Re} = 1000$. There, the background perturbations in the shear-layer are amplified ($t/T_0 = 0, 0.25$) leading to the layer rolling up into vortices ($t/T_0 = 0.75$) which in turn pair-up and merge and become progressively larger ($t/T_0 = 1.25$) until $t/T_0 = 1.75$, at which point the pairing is terminated.

2.6.2 Taylor-Green vortex flow

Moving on to three dimensional space enables new mechanisms that now make the energy flow from large scales to small, in the style of Kolmogorov's energy cascade. The most famous of these mechanisms is that of vortex stretching, and it is particularly pronounced in the TGV flow case. There, a single, large, ordered fluctuation grows increasingly unstable until it eventually breaks down into a chaotic flurry of small scale structures. Vortex stretching drives the formation of progressively smaller vortices until the Kolmogorov microscale is reached and they dissipate. As this viscous dissipation continuously drains kinetic energy (transferring it into thermal energy) from the system, the flow will decay and settle down as time goes on.

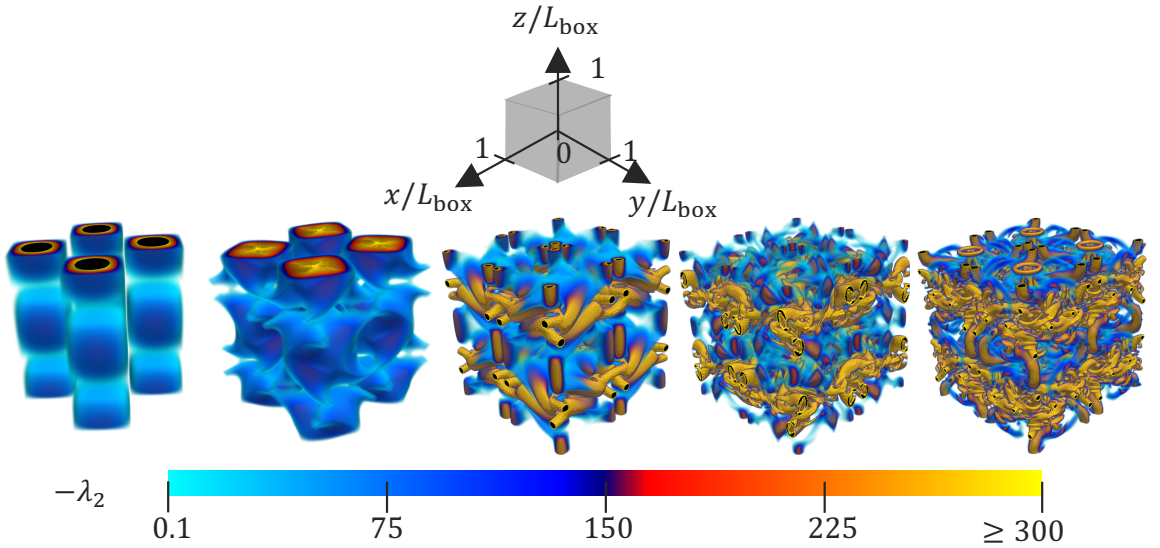

Figure 2.6: **DNS of the 3D Taylor-Green vortex**. Dynamical simulation of the incompressible Navier-Stokes equations in 3D for the TGV flow case at Re = 800. Vortical structures rendered using the λ_2 method [61] are shown at times $t/T_0 = 0, 0.2, 0.8, 1.4, 2$ (left to right). See Section 2.6.2 for the definitions of Re, L_{box} and T_0. This simulation was performed using a $256 \times 256 \times 256$ spatial grid.

The TGV simulations are conducted on a cube with edge length L_{box} with periodic boundary conditions. The initial flow field is given as

$$\begin{aligned} u_1(t=0,\boldsymbol{r}) &= -u_0 \sin(k_0 x)\cos(k_0 y)\cos(k_0 z)\,, \\ u_2(t=0,\boldsymbol{r}) &= u_0 \cos(k_0 x)\sin(k_0 y)\cos(k_0 z)\,, \\ u_3(t=0,\boldsymbol{r}) &= 0\,, \end{aligned} \tag{2.26}$$

where u_0 is the velocity amplitude of the initial vortex and its wavenumber is $k_0 = 2\pi/L_{\text{box}}$. The corresponding energy at $t = 0$ is $E_0 = u_0^2/2$, the Reynolds number is defined using the integral scale as $\text{Re} = u_0/(k_0\nu)$ and T_0 is set to $T_0 = L_{\text{box}}/u_0$, with $t/T_0 \in [0, 2]$.

A DNS of the TGV flow at Re = 800 is illustrated in Figure 2.6. There, the original vortex ($t/T_0 = 0$) collapses ($t/T_0 = 0.2$) into turbulent worm-like structures ($t/T_0 = 0.8$) that grow progressively more turbulent ($t/T_0 = 1.4$) until eventually being dissipated by viscosity ($t/T_0 = 2$).

2.6.3 Schmidt spectrum

We now investigate the Schmidt spectra of the TGV and TDJ flows.

Let us start with the DNS solutions of the TDJ at Re=1000 shown in Fig. 2.5. Each velocity component u_i is decomposed according to Equation (2.21) at every bipartition n and at times $t = 0.25, 0.75, 1.25, 1.75$ in an exact manner, i.e. such that $d(n) = \Gamma^{2-\mathrm{D}}(n)$. This produces the Schmidt spectrum $\lambda_{i,\alpha}(n,t)$, and it is (after being normalised) illustrated in Figure 2.7. A contour corresponding to the quantity $d_{99}(n,t) \leq \Gamma^{2-\mathrm{D}}(n)$ is also included in the plots of this figure. $d_{99}(n,t)$ is the Schmidt number at which the L2-norm error of the representation of the *whole* velocity field is $< 1\%$. From Figure 2.7 it is clearly seen that $d_{99}(n,t)$ lie far below their maximal values for $n > 1$ for all t. This implies that the great majority of Schmidt coefficients are associated with relatively unimportant modes, which is in turn equivalent to the interscale correlations of the TDJ flow being limited throughout the course of the simulation. To be more specific, let $\chi_{99} = \max d_{99}(n,t)$ be defined as the maximal value of d_{99} for all n and t. We find that $\chi_{99} = 25$ for the DNS of the TDJ, which is quite small compared to $\Gamma^{2-\mathrm{D}}(5) = 4^5 = 1024$.

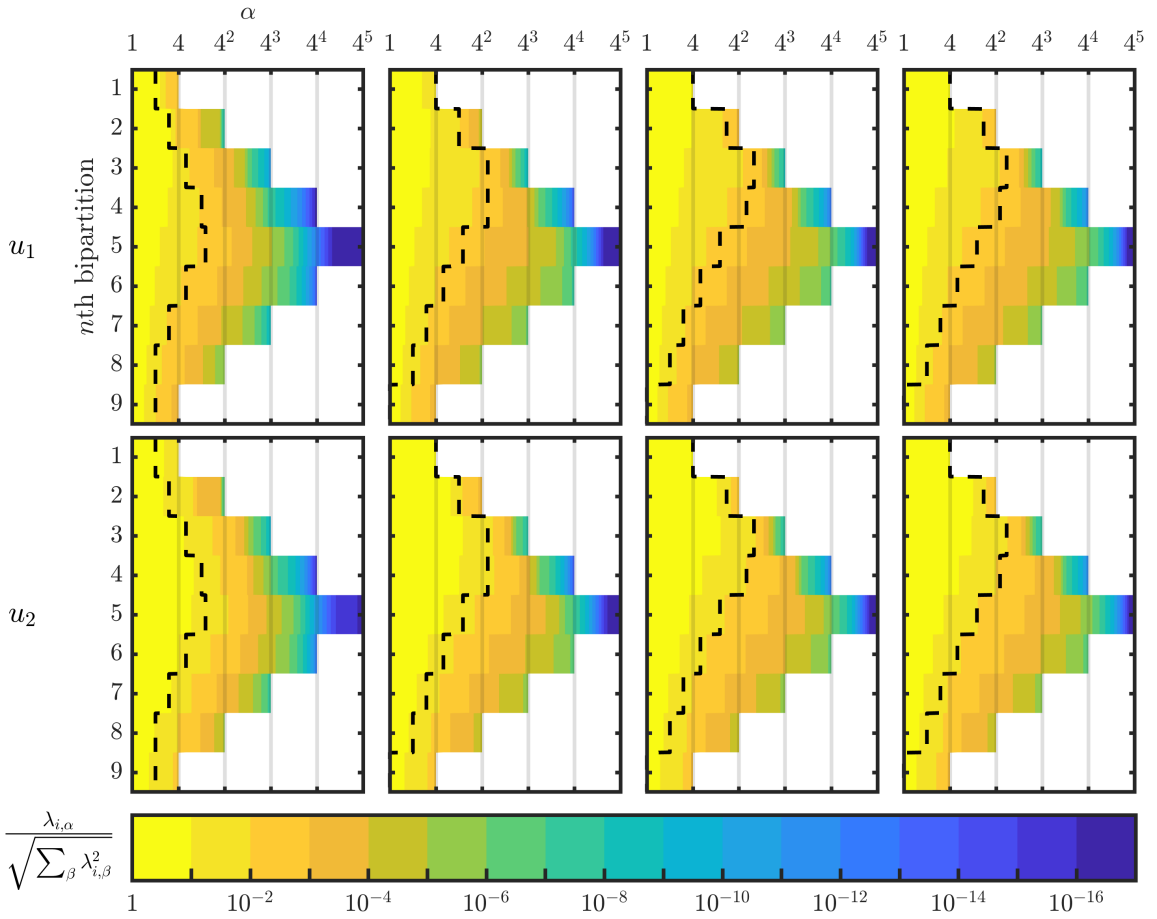

Figure 2.7: **Schmidt spectrum from DNS of the TDJ**. For the TDJ flow at Re=1000, the Schmidt coefficients $\lambda_{i,\alpha}$ obtained from each velocity component u_i are here, after being normalised, shown at times $t/T_0 = 0.25, 0.75, 1.25, 1.75$ (left to right) for each of the 9 bipartitions available on the 1024×1024 DNS grid. The Schmidt coefficients are normalised such that the sum of their squares equals 1, and are sorted in descending order. The black, dashed lines denote the $d_{99}(n,t)$ of Section 2.6.3, which is the Schmidt number at which the L2-norm error of the representation of the whole flow field is at least 99% accurate.

We now move on to the DNS of the TGV at Re=800 from Fig. 2.6. Performing an exact Schmidt decomposition of each of the now three velocity components u_i at times $t = 0.2, 0.8, 1.4, 2$, yields the Schmidt spectrum $\lambda_{i,\alpha}$, which is shown (post normalisation) in Figure 2.8. $d_{99}(n,t)$ (now of a 3D velocity field) is again much smaller than its maximal possible value, implying that the interscale correlations are severely limited also for the 3D TGV flow. Indeed, we find that the χ_{99} of the TGV simulations is $\chi_{99} = 207$, which is small compared to $\Gamma^{3-\mathrm{D}}(4) = 8^4 = 4096$.

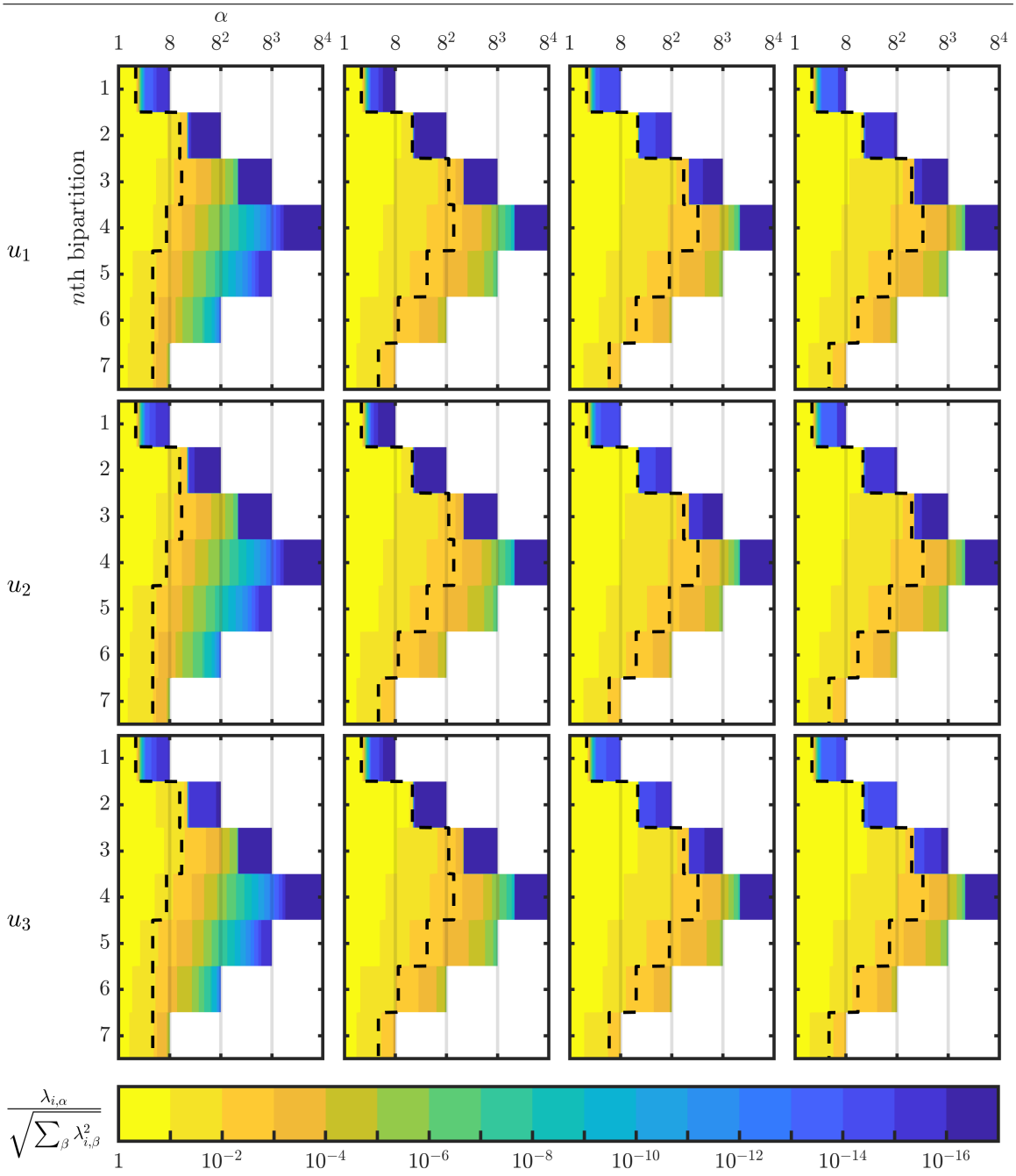

Figure 2.8: **Schmidt spectrum from DNS of the TGV**. For the TGV flow at Re=800, the Schmidt coefficients $\lambda_{i,\alpha}$ obtained from each velocity component u_i are here, after being normalised, shown at times $t/T_0 = 0.2, 0.8, 1.4, 2$ (left to right) for each of the 7 bipartitions available on the $256 \times 256 \times 256$ DNS grid. The Schmidt coefficients are normalised such that the sum of their squares equals 1, and are sorted in descending order. The black, dashed lines denote the $d_{99}(n,t)$ of Section 2.6.3, which is the Schmidt number at which the L2-norm error of the representation of the whole flow field is at least 99% accurate.

2.6.4 The von Neumann entanglement entropy

The interscale correlations of the Schmidt spectrum can be directly quantified using the von Neumann entanglement entropy analogously to how it is used to quantify entanglement in Section 2.2. Following the precedent of Equation (2.4), if the velocity component $u_i(\boldsymbol{r}_q, t)$ has the Schmidt spectrum $\lambda_{i,\alpha} = \lambda_{i,\alpha}(n,t)$, then its von Neumann entanglement entropy $S_i(n,t)$ can be defined as

$$S_i(n,t) = -\sum_{\alpha=1}^{\Gamma^{K\text{-D}}(n)} \frac{\lambda_{i,\alpha}(n,t)^2}{\sum_{\beta=1}^{\Gamma^{K\text{-D}}(n)} \lambda_{i,\beta}(n,t)^2} \log_2 \left(\frac{\lambda_{i,\alpha}(n,t)^2}{\sum_{\beta=1}^{\Gamma^{K\text{-D}}(n)} \lambda_{i,\beta}(n,t)^2} \right). \tag{2.27}$$

Note that the normalisation factor $\sum_{\beta=1}^{\Gamma^{K\text{-D}}(n)} \lambda_{i,\beta}(n,t)^2 = \|u_i(t,\boldsymbol{r}_q)\|_2^2$ is just the square of the L2 norm of the relevant velocity component.

The von Neumann entanglement entropy can be physically interpreted as a type of uncertainty, or Shannon entropy. In the quantum physics context, the entanglement entropy quantifies the uncertainty in measurement outcomes of the state of a quantum sub-system due to its entanglement with the rest of the system. In our fluid dynamics case, the entanglement entropy quantifies how much a velocity field at a given range of length scales (e.g. the fine subgrid) varies (i.e., how uncertain it is) due to its coupling with the flow at the other length scales (e.g. the coarse subrgrid).

The entanglement entropy for the 2D TDJ flow shown in Figure 2.9a shifts towards bipartitions between coarser length scales (i.e. lower n) with increasing time. This behaviour is consistent with a 2D inverse energy cascade [38] where energy is carried from fine to coarse length scales as time progresses, e.g. through vortex merging. These dynamics are particularly pronounced for the cross-stream u_2 velocity component, for which a large number of fine scale disturbances become energised by the shear and grow in size ($t/T_0 \approx 0.25, 0.75$) until the eventual collapse of the jet and the saturation of the shear layer ($t/T_0 \approx 1.25, 1.75$). At later times no further growth of the disturbances occur. Remarkably, these physics are visible in the dynamics of the entanglement entropy. At $t/T_0 = 0.25$ the entanglement entropy is large for all bipartitions between $n = 1$ and $n = 7$, indicating significant correlations between all length scales. At later times ($t/T_0 = 1.25, 1.75$), when the energy increasingly flows towards coarser length scales, also the entanglement entropy shifts towards lower n bipartitions as shown in Figure 2.9a.

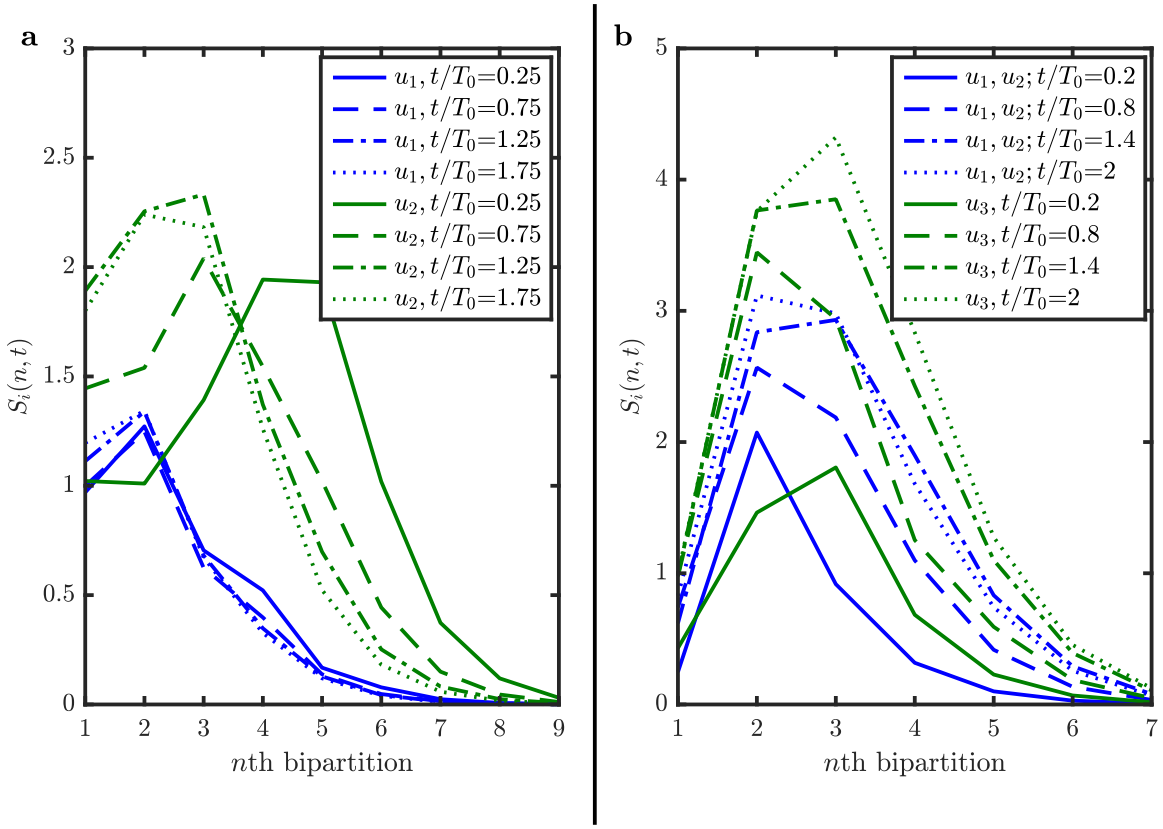

Figure 2.9: **Von Neumann entanglement entropy between length scales in the TDJ and TGV flows**. The entanglement entropies $S_i(n,t)$ for the velocity fields u_i at each available bipartition n are calculated from the DNS solutions of the TDJ **a** and TGV **b** flows and illustrated at various times t. The 1024×1024 grid of the DNS of the TDJ are bipartitioned along 9 length scales, while 7 bipartitions are used for the $256 \times 256 \times 256$ grid DNS of the TGV. A symmetry exists between the u_1 and u_2 components of the TGV velocity field, making their entropies overlap within **b**.

For the 3D TGV flow shown in Figure 2.9b the opposite happens. There, fine length scales become energised with increasing time. Correspondingly, the entanglement entropy increases at larger values of n with time. This increase is consistent with the hypothesis of a direct energy cascade in 3D turbulent flows [5], where mechanisms such as vortex stretching transports energy to progressively finer and finer length scales until the Kolmogorov microscale is reached and the energy starts being dissipated by viscosity. However, unlike in the 2D TDJ case, the outflow of energy is not accompanied by a corresponding reduction of interscale correlations. Instead, the entanglement entropy increases with time for all bipartitions, a result of the increasing disorder as the TGV collapses into various worm-like vortical structures.

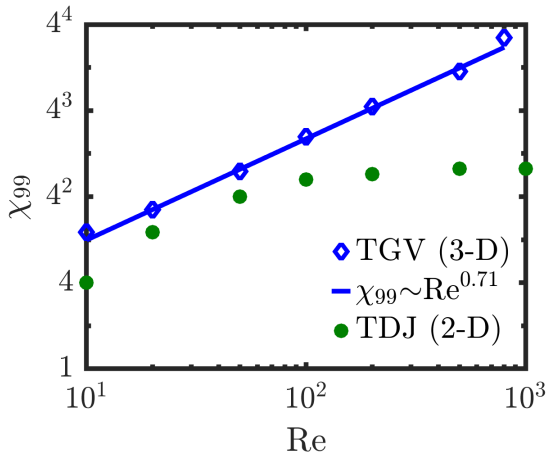

Figure 2.10: **Relation between amount of interscale correlation and Reynolds number**. Scaling of $\chi_{99} = \max d_{99}(n,t)$, with the Reynolds number of the 2D TDJ and 3D TGV flows defined in Sections 2.6.1 and 2.6.2.

Note that these dynamics of the entropies $S_i(n,t)$ for each of the velocity components u_i are all consistent with those of d_{99} for both the TDJ and TGV. Compare Figures 2.7 and 2.8 to Figure 2.9, where we see how flatter singular values distributions, implying a high degree of interscale correlation, correspond to larger entanglement entropies.

2.7 Scaling of interscale correlations with Reynolds number

A fundamental question is how the amount of interscale correlations scales with Reynolds number, the single parameter (along with initial and boundary conditions, but these are kept constant here) which fully defines the dynamics of incompressible flows. To investigate, we calculated how χ_{99}, our main quantifier of interscale correlation, varies with Re. The resulting plot is provided in Figure 2.10.

Interestingly, in Figure 2.10, χ_{99} saturates in the TDJ case for Re $\gtrsim 200$. This suggests that the interscale correlations of 2D flows are bounded in a fashion analogous to quantum correlations of the locally interacting, gapped 1D quantum systems discussed in Section 2.3. A possible explanation for this might be rooted in the inverse energy cascade. As the cascade carries energy from the fine scales to the coarse, the flow field becomes increasingly larger-scale (i.e. smoother) as time moves on. This removal of the finest flow scales reduces the span of length scales available to correlate with each other, which consequently keeps the flow scale-local and, by extension, bounds the amount of interscale correlation. Recall now from Section 2.4 that increasing the Reynolds number does not change the overall physics of the energy cascades, it

only reduces the dissipative microscale η relative to L_{box}. Thus once Re is made sufficiently large to remove dissipation from consideration, only the inverse cascade is left, and since the fundamental physics behind the inverse cascade do not change with increasing Re, it stands to reason that χ_{99} must eventually saturate with Re precisely as it does in Figure 2.10.

χ_{99} increases according to a power law in the 3D case as $\chi_{99} \sim \text{Re}^{0.71}$. This is reminiscent of the polynomial scaling of the separation between the largest and smallest scales in 3D turbulence, $\ell/\eta \sim \text{Re}^{3/4}$, which implies that the number of variables employed in 3D DNS goes as $\sim M \sim \text{Re}^{9/4}$, as discussed in Chapter 1. By truncating all Schmidt numbers at every bipartition down to χ_{99}, one may define a new parameterisation wherein the modes associated with the least important Schmidt coefficients are not resolved. The number of variables required for this $d(n) \leq \chi_{99}$ representation scales with χ_{99} as $\chi_{99}^2 \log M$, as is explained in the following Section 2.8. In the case of the TGV flow, this becomes $\text{Re}^{1.42} \log \text{Re}$, since $\chi_{99} \sim \text{Re}^{0.71}$ and $M \sim \text{Re}^{3/4}$.

Assume the TGV scaling behaviours holds for large Re. Then for growing Reynolds number, the asymptotic ratio between the variables required in the $d(n) \leq \chi_{99}$ parameterisation versus DNS goes as

$$\sim \frac{\chi_{99}^2 \log M}{M^3} \sim \frac{\text{Re}^{1.42} \log \text{Re}}{\text{Re}^{9/4}} \sim \frac{\log \text{Re}}{\text{Re}^{0.83}} \tag{2.28}$$

for the 3D TGV flow. Equation (2.28) suggests that as Re grows larger, the above ratio decays roughly polynomially and thus a smaller and smaller fraction of the variables used by DNS are actually necessary to represent 3D turbulent flows – a result that is consistent with the notion of scale-locality imposing structure upon turbulence.

2.8 Exploiting the scale-locality of turbulence

The preceding sections established that turbulent flows can be accurately resolved at highly restricted Schmidt numbers. This reality, which is a reflection of the scale-local nature of turbulence, can immediately be exploited for simulation by expressing each velocity component in the compressed tensor network format known as matrix product state (MPS) or tensor train decomposition [21, 29]. MPSs represent a natural approach for systematically restricting the Schmidt numbers of Equation (2.21) across

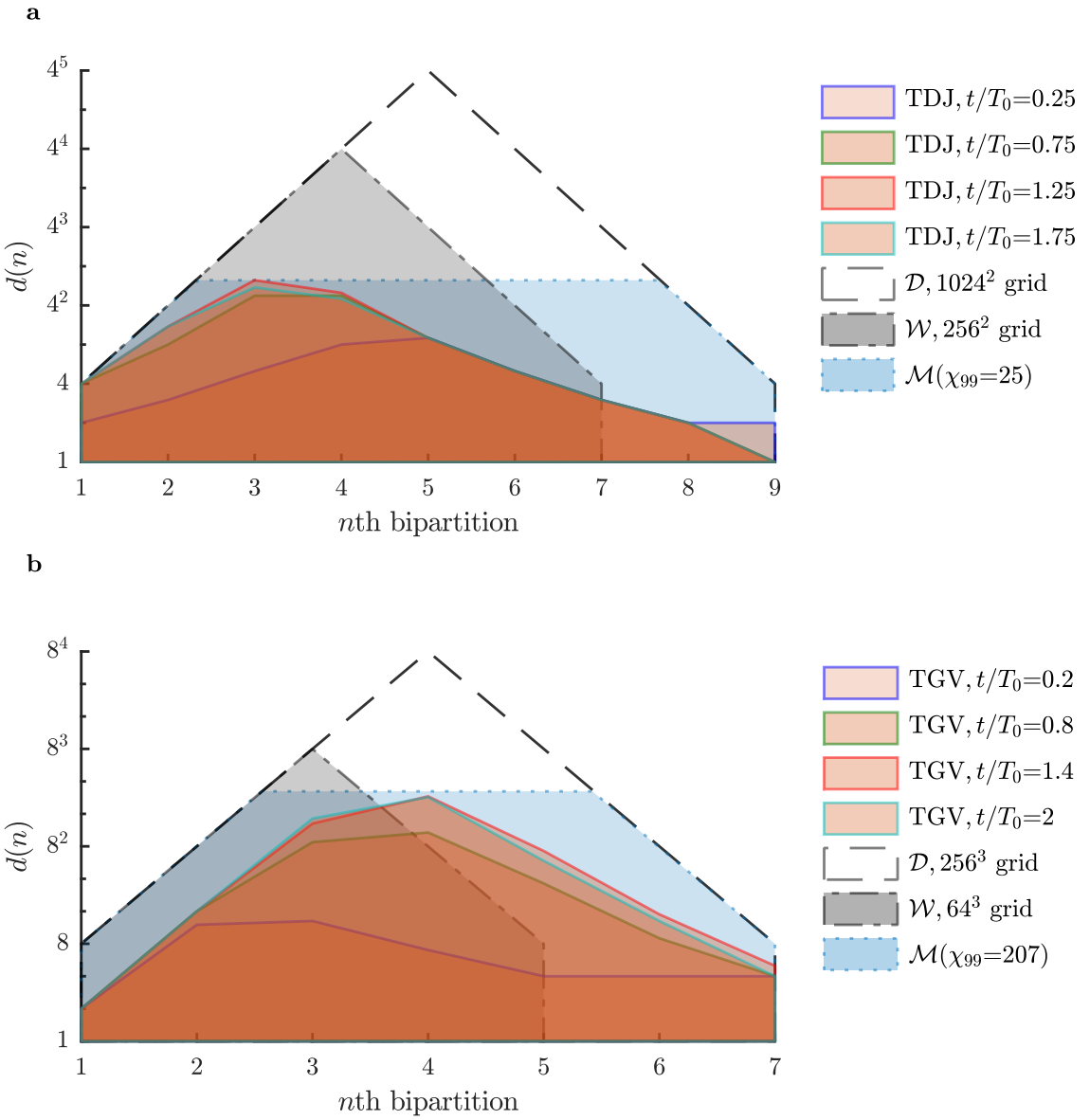

Figure 2.11: **Turbulence interscale correlations**. **a** shows the Schmidt numbers $d_{99}(n,t)$ on a logarithmic scale such that the decomposition in Equation (2.21) results in a 99% accurate representation of DNS solutions to the 2D TDJ at four different times, on a 1024×1024 spatial grid [see Figure 2.5]. $\mathcal{D}$, indicated by the black dashed line, is the solution-space of DNS. The grey area $\mathcal{W}$ is for solutions on a 256×256 grid. The blue shaded area $\mathcal{M}$ is the solution-space when $d(n) \leq \chi_{99}$ in Equation (2.21), with $\chi_{99} = \max d_{99}(n,t)$. **b** Shows $d_{99}(n,t)$ Schmidt numbers for DNS solutions to the 3D TGV flow at four different times on a $256 \times 256 \times 256$ spatial grid [see Figure 2.6].

all n, thus limiting the interscale correlations. The MPS representation comprises products of N tensors, with the nth tensor corresponding to a length scale of $L_{\text{box}}/2^n$

and being of dimension $d(n-1) \times 2^K \times d(n)$, where 2^N is the number of gridpoints in each spatial direction, $d(0) = d(N) = 1$ [62] and $d(n) = \min\left(\Gamma^{\text{2-D}}(n), \chi\right)$ in 2D and $d(n) = \min\left(\Gamma^{\text{3-D}}(n), \chi\right)$ in 3D. This means the number of variables in the MPS representation goes as $\sim 2^K \chi^2 N \sim 2^K \chi^2 \log M$, which is only a logarithmic increase with the total number of gridpoints M, if χ is kept constant, resulting in an exponential reduction in the number of variables parameterising the solution. A detailed outline of the MPS decomposition for real-space functions is provided in Chapter 4.

The parameter χ is known as the bond-dimension and can be freely varied. By reducing χ from its maximal value, the least important interscale correlations are systematically discarded and the solution space goes from $\mathcal{D}$ (of DNS) to the so-called MPS manifold $\mathcal{M}(\chi) \subseteq \mathcal{D}$. χ thus controls the amount of interscale correlations resolved in $\mathcal{M}$ as well as the level of compression achieved in the MPS representation.

$\mathcal{M}(\chi)$ is illustrated in Figure 2.11. There, the interscale correlations of the 2D (3D) flow captured by an MPS with bond-dimension $\chi = 25$ ($\chi = 207$) correspond to the MPS manifold in Figure 2.11a (Figure 2.11b). These figures make it clear that to cover all the relevant length scales of flow, DNS must operate within the wastefully large $\mathcal{D}$, where unimportant interscale correlations are resolved. In contrast, all the relevant length scales can be captured within $\mathcal{M}$ without having to resolve unimportant interscale correlations. Further, note that truncating the scales resolved in DNS leads to a solution space $\mathcal{W} \subseteq \mathcal{D}$ which is only capable of representing coarse solutions. Operating within $\mathcal{W}$ is ineffectual as many relevant length scales are ignored (as also illustrated in 2.11). On the other hand, $\mathcal{M}$ can be truncated (by reducing χ) without affecting the range of length scales it covers.

In order to fully utilise the promised dimensionality reduction of MPS for numerical simulations on large grids, we have devised a MPS algorithm for simulating fluid dynamics. This algorithm variationally solves the incompressible Navier-Stokes equations within the compressed MPS manifold $\mathcal{M}$ and is outlined in detail in Chapter 5. Resulting MPS simulations using it are presented for the TDJ and TGV flow case towards the end of this thesis, in Chapter 7.

2.9 Conclusion

Locality is a ubiquitous phenomenon of nature whose presence in physical systems can induce structure, structure which in turn allows the states of these systems to be described using fewer variables than would otherwise be the case (i.e., the states get data-compressed). A prime example of this is how local entanglement between particles results in area laws that can be exploited by tensor network methods to facilitate an *exponential* reduction in the number of variables required for accurate simulation.

Locality is also a prime characteristic of turbulence. Not in terms of correlations between quantum particles, but rather between *length scales of flow*: in turbulent fluids, energy is transferred between the largest and smallest length scales in a cascade moving through all the intermediary scales. This *scale-local* turbulent energy cascade structures the flow such that eddies of significantly different size become less correlated with one another. We examined this structure by numerically computing the interscale correlations present within two paradigmatic turbulent flows, the 2D TDJ and the 3D TGV, using the Schmidt decomposition. Our investigation reveals that both flows can be accurately represented using relatively low Schmidt numbers, implying that the amount of correlations present between the scales is limited. Indeed, by removing unphysical interscale correlations, we find that the 3D TGV flow can be data-compressed at a ratio scaling with Re as $\sim \frac{\log \mathrm{Re}}{\mathrm{Re}^{0.83}}$. In the 2D TDJ flow, the interscale correlations stop increasing with the Reynolds number after a certain point, implying they are bounded in a fashion reminiscent to how entanglement is limited in area law following quantum systems. These results reflect the scale-locality of turbulence and imply that turbulent flows can be accurately simulated in a massively data-compressed format by simply truncating the least important Schmidt coefficients from the solution space.

A straightforward approach to restricting the Schmidt numbers (and by extension, the interscale correlations) in this fashion is the tensor network known as the MPS ansatz. By representing the flow in the form of MPSs, the scale-locality of turbulence can be exploited to perform simulations within the highly data-compressed MPS manifold. Furthermore, the simplicity of MPSs allows us to formulate an elegant algorithm for solving the Navier-Stokes equations, as shall become apparent in the coming Chapters 3, 4 and 5.

Chapter 3

Tensor network theory

3.1 Introduction

In this chapter we momentarily take a step back from fluid dynamics to have a closer look at tensor networks. The theory described here will be useful in the later chapters where the tensor network approach is used to formulate a MPS scheme for solving the incompressible Navier-Stokes equations.

In Ch. 1, we discussed how the states of quantum many-body systems with local interactions become structured [12] in a way that restricts all physically-occurring states to a vanishingly small volume of the full Hilbert space. Sometimes, this structure takes the form of an *area law of entanglement* [19], where quantum correlations are concentrated at the boundaries between different parts of the system [Section 2.3].

Such area laws are often exploited to simulate quantum many-body systems that would otherwise be intractable due to the exponential size of the Hilbert space. This is done by restricting the simulation to the tiny part of the Hilbert space that is actually relevant through the method of tensor networks [17, 20, 21]. Encoding quantum states in the form of tensor networks and simulating them entirely within this form allows the systematic elimination of unphysical long-distance correlations from the computational domain, whilst still resolving the *physical* short-distance correlations. This can result in an *exponential* reduction in the computational complexity compared to direct methods such as exact diagonalisation, whose solution space is the whole N-particle Hilbert space. In particular, the ground states of 1D lattice systems with area law structure are perfectly captured by MPSs [20, 21] while their 2D analogues

can be captured by PEPSs [22, 23], permitting these ground states to be computed at only $\sim \text{poly}(N)$ cost instead of $\sim \exp(N)$ cost.

However, the utility of tensor network methods is not restricted to just finding ground states: they can also be used to efficiently time-evolve quantum systems with short-range interactions for brief times. The reason why is rooted in the Lieb-Robinson theorem, which states that the speed at which information propagates through a quantum system is bounded [21, 63] when particles only locally interact (though this actually turns out to hold true even for some specific forms of long-range interactions [64]). This means, assuming the initial state contains no entanglement between distant particles, that the entanglement between particles will remain short-range during the first part of the time-evolution, allowing these initial dynamics to be captured with poly-cost tensor network simulations. Full time-evolution of quantum systems (even those with just short-range interactions) is however widely believed to require a quantum computer [65].

Tensor networks also work well for weakly entangled systems in general [19]. For instance, in quantum phase transitions the bipartite entanglement scales with the area separating the sub-systems *multiplied* by the logarithm of the volume of the smallest sub-system. Not only can such logarithmic violations of the area law *still* be handled by the MPS and PEPS ansätze at $\sim \text{poly}(N)$ cost, the MERA geometry is specifically designed to reproduce this logarithmic violation by being inherently capable of capturing the power law decaying quantum correlation seen in quantum critical states [18, 66].

MPS, PEPS and MERA are however just a few famous examples of tensor networks. A whole zoo of alternative geometries exist. For instance, tensor renormalisation group [67], tensor-entanglement renormalisation group [68], tensor product variational formulation [69], weighted graph states [70], string-bond states [71], entangled-plaquette states [72], tree tensor networks [73], continuous MPS [74] and continuous MERA [75]. There also exists infinite-MPS [76, 77] and infinite-PEPS [78] for representing lattice systems in the thermodynamic limit. Indeed, the branching MERA [79] geometry is even capable of capturing quantum states states where the bipartite entanglement scales with the *volume* of the smallest sub-system (i.e. which follow a *volume law* instead of an area law)!

While all of these tensor networks are unique in their own ways, the underlying strategy

remains identical: by representing the quantum state in tensor network form, the numerical variables are restricted in an intelligent and systematic manner such that the simulation is carried within a polynomially large solution space which contains the realistic quantum states. This, in turn, brings down the computational cost of the simulation from $\sim \exp(N)$ to $\sim \text{poly}(N)$.

Finally, it must be mentioned that although tensor networks were born in the field of quantum information theory, they have by now migrated into other fields. For example, tensor networks are applied in numerical mathematics [29], computer science [30], machine learning [31], the study of quantum gravity [28] and even linguistics [32]. For a more complete overview of the use of tensor networks outside of quantum physics, see [33].

3.1.1 Chapter structure

In the remainder of this chapter we introduce tensor networks in more concrete terms. A mathematical and graphical description of tensor networks is provided in Section 3.2. Section 3.3 outlines precisely how quantum many-body states are represented as tensor networks and why this leads to an exponential reduction in the number of variables. Then the focus of the chapter shifts toward the simplest of tensor networks, the MPS ansatz. The MPS vectors are covered in Section 3.4, whilst the matrix product operators (MPOs) acting upon them are presented in Section 3.5. Finally, the chapter is concluded in Section 3.6.

3.2 Tensor algebra and tensor diagrams

Let us now take a more concrete consideration of tensor networks. From a computational perspective, a tensor is essentially just a multidimensional array of complex numbers. The tensor order denotes the number of indices. For example, an order-0 tensor represents a scalar, order-1 tensor is a vector, an order-2 tensor a matrix, etc. Furthermore, each of these indices are associated with their own dimension. A tensor network is then simply a number of tensors which are contracted with each other in some specific order. For example, let $A_{\alpha\beta}$ be an order-2 tensor which is contracted through its second index with some order-1 tensor a_γ. This is nothing other than a

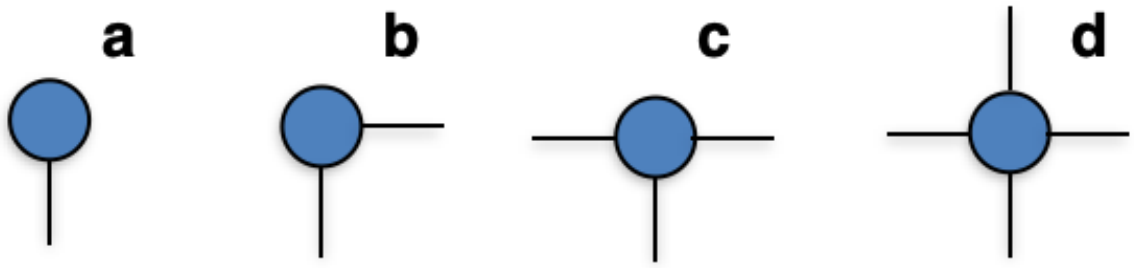

Figure 3.1: **The diagrammatic notation for tensors**. In **a** a vector is denoted, in **b** a matrix, in **c** an order-3 tensor while in **d** we have an order-4 tensor.

matrix-vector multiplication

$$\sum_{\beta=1}^{d} A_{\alpha\beta} a_\beta = b_\alpha, \tag{3.1}$$

resulting in the vector b, with $d \in \mathbb{Z}$ here being the number of rows of a. Equation (3.1) represents a very simple tensor network consisting of a matrix contracted with a vector.

But tensor networks can of course also be far more complicated. Consider for instance the following contraction

$$\sum_{\gamma,\epsilon,\zeta,\eta=1}^{d} A_{\alpha\beta\gamma} B_{\gamma\delta\epsilon\zeta\eta} C_{\eta\theta} D_{\epsilon\zeta} = E_{\alpha\beta\delta\theta}, \tag{3.2}$$

where the dimension of each index equals d, which results in the order-4 tensor E. Even for this network of four tensors, it is already challenging to keep track of which index is being contracted with which. And as the tensor networks becomes more involved, e.g. when many high-order tensors are contracted with each other through various indices, it quickly becomes impractical to keep track of the contractions through equations. Therefore, for the sake of legibility, the convention is to represent tensor contractions *diagrammatically*.

In such diagrams the tensors are represented by nodes (which can be of arbitrary shape – circle, square, triangle, etc.) and their indices as lines (or legs), as drawn in Figure 3.1. Contracted indices are represented by a line that connects the relevant tensor pair, whilst uncontracted indices are represented by unconnected lines, as shown in Figure 3.2. Furthermore, various numerical operations can also straightforwardly be pictorially represented, such as the SVD and QR decompositions [39] illustrated in Figure 3.3.

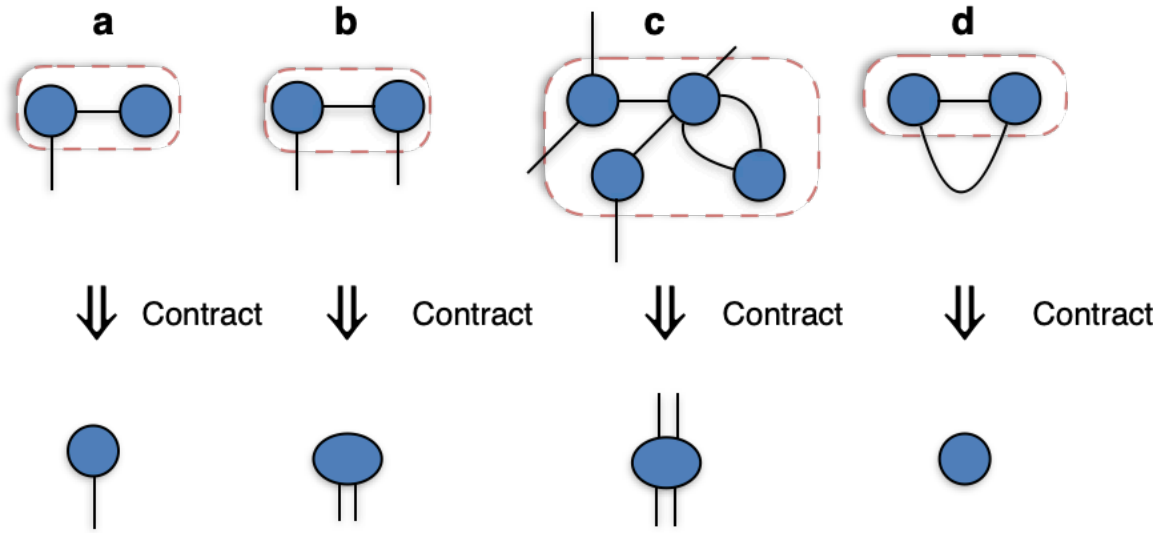

Figure 3.2: **Tensor network diagrams representing various contractions**. **a** denotes a vector-matrix multiplication while in **b** there is a matrix-matrix multiplication. The contraction of Equation (3.2) is pictorially represented in **c**. **d** represents the trace of a matrix-matrix product.

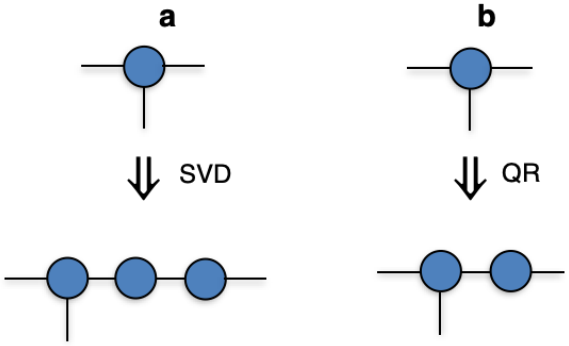

Figure 3.3: **Diagrammatic representation of two numerical operations**. **a** denotes a singular values decomposition and **b** a QR decomposition, both acting on the same tensor. During both decompositions, the tensor is considered as a matrix with the left and down indices corresponding to the rows whilst the right index corresponds to the columns.

Finally, it must be noted that the number of operations which is required to perform a contraction of a tensor network strongly depends on the order of contractions. For instance, in Figure 3.4, Equation (3.2) is contracted in two different ways at widely different computational cost. In general, minimising the cost of a tensor network

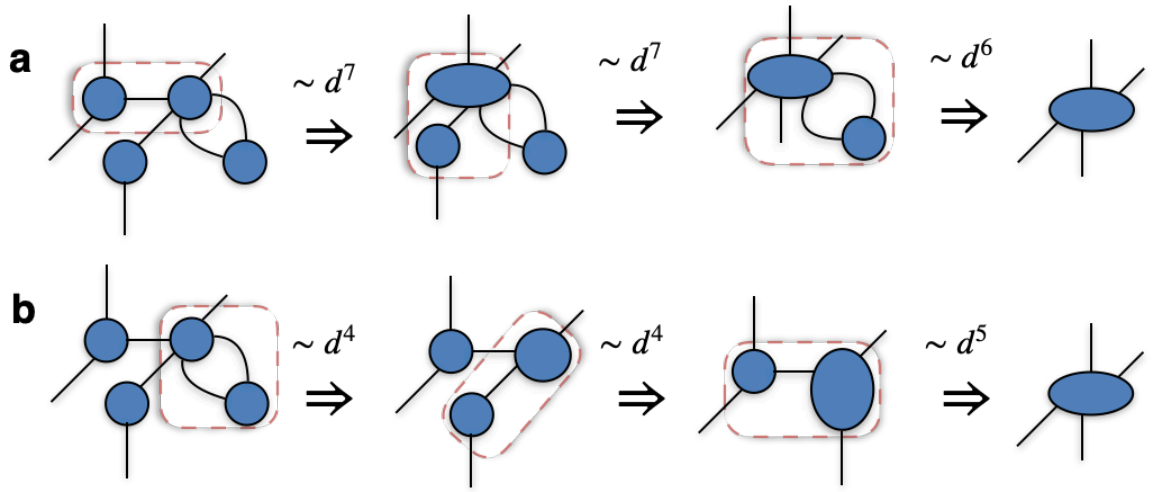

Figure 3.4: **Two different strategies for calculating Equation** (3.2). The contraction order utilised in **a** requires $\sim d^7$ operations, whilst that of **b** needs only $\sim d^5$. All indices are assumed to be of dimension d.

contraction requires one to optimise over all possible contraction orderings. This is in some cases a computationally hard problem [80] while in other cases easy and intuitive (e.g. MPS). Keeping in mind that the computational cost is highly dependent on contraction orders is crucially important when designing tensor network algorithms.

3.3 Exponential compression

In this section we explain how representing states as tensor networks can result in an exponential reduction in the number of variables. Let us take a quantum state as our example, e.g. the general many-body quantum state $|\Psi\rangle$ from Equation (2.1). The order-N amplitude tensor C contains $\sim \exp(N)$ elements, one for each of the exponentially numerous basis vectors forming the N-particle Hilbert space. The exponential number of elements contained in C makes it highly inefficient to describe many-body states by explicitly specifying each element of C. However, after reading Section 3.1 we know that the physically relevant quantum states of many important quantum systems are restricted to a tiny speck of the full Hilbert space. Therefore, in these cases, it is not only infeasible to specify each element of C, but also unnecessary. This is the fundamental idea behind tensor networks: representing an exponentially large state as a tensor network opens the path to go from an exponentially expensive

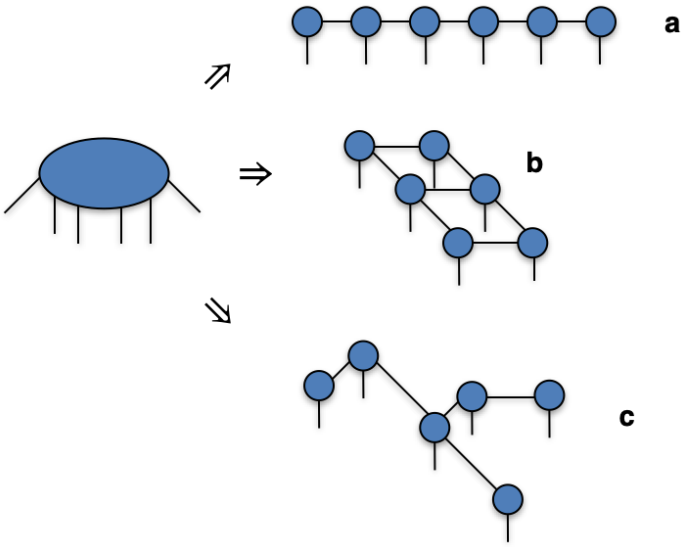

Figure 3.5: **Three tensor network decompositions of a tensor**. In **a** we have a MPS, in **b** a PEPS whilst **c** depicts some arbitrary geometry.

description to just a polynomially expensive one, without significantly affecting the accuracy when the state is structured appropriately (e.g. according to an area law of entanglement).

Let us discuss the polynomial scaling of tensor networks more concretely. Assume C is represented in the form of some tensor network like the ones seen in Figure 3.5. Then the total number of parameters p_{tot} of this representation is given by

$$p_{\text{tot}} = \sum_{t=1}^{N_{\text{tens}}} p(t), \tag{3.3}$$

with $p(t)$ the number of parameters in tensor t and N_{tens} being the total number of tensors in the tensor network. In general, N_{tens} must be polynomial in the number of particles N for the ansatz to make sense, i.e. $N_{\text{tens}} \sim \text{poly}(N)$. $p(t)$ itself is given by multiplying the number of elements associated with each index of tensor t as

$$p(t) = \prod_{i=1}^{\text{order}(t)} d(t,i), \tag{3.4}$$

where $d(t,i)$ is the dimension of index i, with i running from $i = 1$ all the way up to the order of the tensor order(t). Finally, let us now look at the scaling of p_{tot}. If $\chi = \max_{i,t}\{d(i,t)\}$ is the largest of all the dimensions, then $p(t)$ is bounded by $\sim \text{poly}(\chi)$

in the form of

$$p_{\text{tot}} \sim \text{poly}(\chi)\text{poly}(N), \tag{3.5}$$

as long as the order of all tensors is also bounded by some constant. Thus if χ is fixed while N is allowed to increase, the tensor network approximation of C will achieve exponential compression.

For such a polynomial tensor network approximation of C to be accurate, the original $\sim \exp(N)$ coefficients in C must be highly dependent on each other, i.e. C highly structured. A famous example of such structure is the bounding of entanglement according to area laws discussed in Sections 2.3 and 3.1, in which case C is ideally represented using e.g. the MPS ansatz in 1-D (Figure 3.5a) or PEPS in 2-D (Figure 3.5b). Taking the MPS as an example, it is straightforward to show that the number of parameters scales as $\sim Ng\chi^2$, with N being its length and g the dimension of the open indices when assuming (for the sake of simplicity) that all of them are of the same dimension g. Yet the contraction of the MPS will yield a tensor containing g^N coefficients – thus quantum states of amenable structure can be encoded in the form of a MPS containing just polynomially many parameters, despite the states living in exponentially large Hilbert spaces. The same argument applies for PEPS and other tensor networks.

3.4 Matrix product states

The main tensor network considered in this thesis is the MPS ansatz. The simplicity of the MPS architecture has turned MPSs into a widely used ansatz backed by a highly developed mathematical machinery. Examples of powerful MPS algorithms include the density matrix renormalisation group [81, 82] for calculating ground states of quantum systems, time-evolving block decimation for time-evolution of quantum systems wherein the interactions are local [65] and the time-dependant variational principle for time-evolution with non-local interactions [83]. Various variations of these schemes also exist such as adaptations to infinite systems [76, 77] and MPI-parallelised variants [84–86].

While MPS have traditionally been used for simulating either weakly entangled or area law following 1D lattice systems from quantum many-body physics [17, 19–21], in recent years it has also been introduced into the applied numerical mathematics [29]

and computer science [30] communities (where MPSs go under the name of "tensor trains" or "quantics tensor trains"). In this thesis, MPSs are applied to problems in fluid dynamics where the relevant states are vector fields made up of time-dependant scalar functions instead of quantum states.

The text below is structured as follows. First, the MPS decomposition is derived for a general tensor in Section 3.4.1. Section 3.4.2 addresses the number of variables available within an MPS. Afterwards, in Section 3.4.3, we discuss the close connection between MPSs and the Schmidt decomposition of Section 2.2. Finally, for context, we mention a few exact MPS representations in Section 3.4.4.

3.4.1 Decomposition

The MPS ansatz is traditionally illustrated by simply decomposing a tensor into a MPS. In this spirit, let A be a order-N tensor with complex-valued elements $A_{i_1 i_2 \ldots i_N}$. A will now be decomposed into a MPS of N sites with the help of repeated SVDs.

We begin by reshaping A into a matrix where i_1 runs across the rows, whilst $i_2, i_3, \ldots, i_N$ run through the columns. Performing a truncated SVD such that the χ largest singular values are kept results in

$$A_{i_1 i_2 \ldots i_N} \approx \sum_{\alpha_1=1}^{\leq \chi} U^{i_1}_{\alpha_1}[1] S_{\alpha_1 \alpha_1}[1] V^{\dagger}_{\alpha_1 i_2 i_3 \ldots i_N}[1]. \tag{3.6}$$

Here S is the so-called singular values matrix and its entries are real and placed in descending order solely along the diagonal, while U and V are both unitary matrices. We now multiply $S[1]$ with $V^{\dagger}[1]$ together to form the remainder tensor $R[1] = S[1]V^{\dagger}[1]$. These operations are illustrated in the top three rows of Figure 3.6.

Continuing, we now reshape $R[1]$ such that the multi-index $i_2\alpha_1$ runs over the rows whilst the remaining indices run across the columns. Performing another χ-restricted truncated SVD on $R[1]$ yields

$$A_{i_1 i_2 \ldots i_N} \approx \sum_{\alpha_1=1}^{\leq \chi} \sum_{\alpha_2=1}^{\leq \chi} U^{i_1}_{\alpha_1}[1] U^{i_2}_{\alpha_1 \alpha_2}[2] S_{\alpha_2 \alpha_2}[2] V^{\dagger}_{\alpha_2 i_2 i_3 \ldots i_N}[2]. \tag{3.7}$$

See the fourth row of Figure 3.6. Analogously to before, multiplying $S[2]$ and $V^{\dagger}[2]$ together will form the remainder tensor $R[2] = S[2]V^{\dagger}[2]$, as shown in the fifth row of

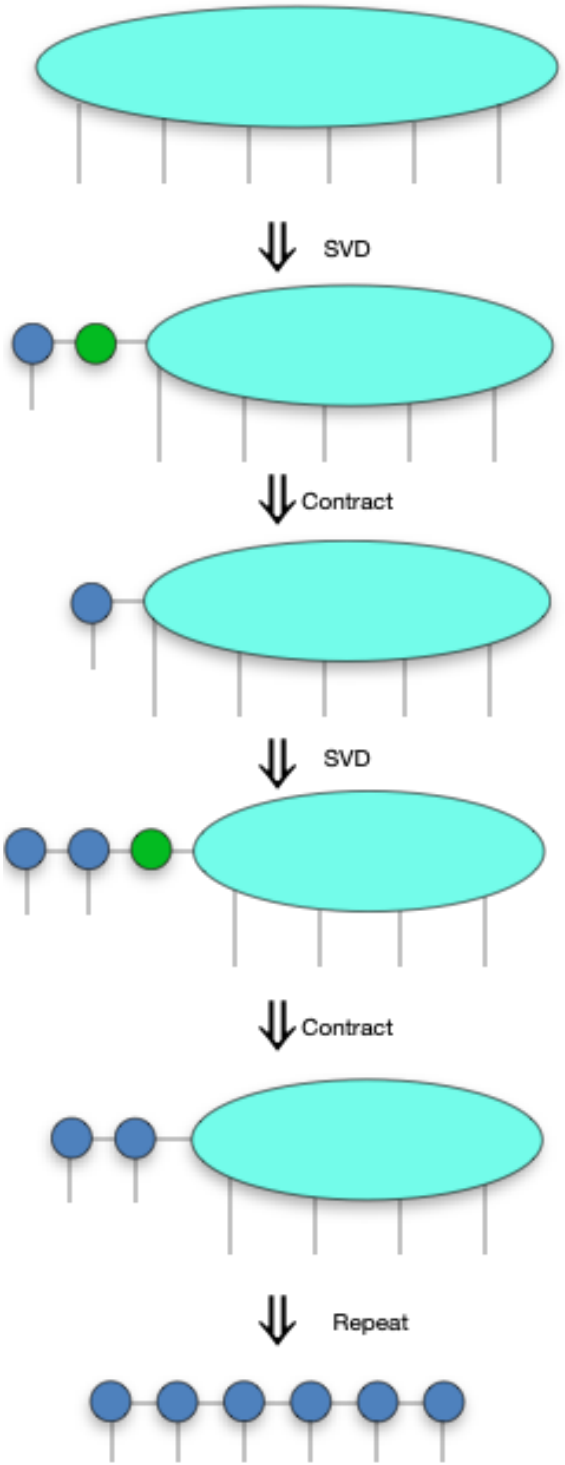

Figure 3.6: **Graphical illustration of the MPS decomposition.** The top row represents the original tensor $A_{i_1 i_2 \ldots i_6}$ (cyan). The second row shows the result of the first SVD between indices i_1 and $(i_2 i_3 \ldots i_6)$, with the first tensor $U[1]$ being marked in blue, the singular values tensor $S[1]$ in green and the remaining factor $V^\dagger[1]$ in cyan. These latter two tensors have been contracted with each other in the third row, resulting in the cyan tensor. The fourth row illustrates the result of the second SVD between indices i_2 and $i_3 i_4 \ldots i_6$. In this row, $U[1]$ and $U[2]$ are shown in blue, $S[2]$ in green and $V^\dagger[2]$ in cyan. In the fifth row, these have been contracted and produced the cyan tensor. The sixth row illustrates the final MPS after another three analogous SVDs and tensor contractions.

Figure 3.6. Doing another truncated SVD results in

$$A_{i_1 i_2 \ldots i_N} \approx \sum_{\alpha_1=1}^{\leq\chi} \sum_{\alpha_2=1}^{\leq\chi} \sum_{\alpha_3=1}^{\leq\chi} U^{i_1}_{\alpha_1}[1] U^{i_2}_{\alpha_1\alpha_2}[2] U^{i_3}_{\alpha_2\alpha_3}[3] S_{\alpha_3\alpha_3}[3] V^{\dagger}_{\alpha_3 i_3 \ldots i_N}[3]. \tag{3.8}$$

A clear pattern emerges: performing consecutive tensor-reshapes and truncating SVDs will result in an increasingly long chain of matrix products that approximate A. After $N-1$ such reshapes and truncated SVDs, the result will be

$$A_{i_1 i_2 \ldots i_N} \approx \sum_{\{\alpha_n\}=1}^{\leq\chi} U^{i_1}_{\alpha_1}[1] U^{i_2}_{\alpha_1\alpha_2}[2] \ldots U^{i_{N-1}}_{\alpha_{N-2}\alpha_{N-1}}[N-1] R^{i_N}_{\alpha_{N-1}}[N], \tag{3.9}$$

with $R[N] = S[N]V^{\dagger}[N]$. This is the MPS decomposition of A, and it is illustrated for $N = 6$ in the very last row of Figure 3.6.

Before moving on, note that the MPS decomposition here is illustrated for solely pedagogical purposes and is in fact never explicitly carried out in practical algorithms. This is because the MPS decomposition carries an *exponential* computational cost in N (to see why, simply consider the cost of the SVD of $V^{\dagger}[\lfloor N/2 \rfloor]$), which defeats the purpose of using tensor networks in the first place. Instead, in actual algorithms, every MPS is instead initialised either tensor by tensor or through alternative strategies such as the prolongation algorithm of Section 4.4.

3.4.2 Number of variables

Employing the above MPS representation can result in enormous data-compression. Whilst the number of variables needed to describe A exactly scales as $\sim \exp(N)$, the amount of variables contained inside the MPS of Equation (3.9) goes merely as $\sim \chi^2 N$. Thus, if χ is kept constant for increasing N, the compression in the number of variables becomes exponential (as previously outlined in Section 3.3). It must however be noted that the number of *physical* variables is in general different from the number of parameters defining the tensor network. This is because all tensor network representations, including MPS, contain inherent gauge freedoms at each bond (see e.g. [86]). In the case of MPS, the exact number of available variables Q_{MPS} within the MPS of Equation (3.9) is given by

$$Q_{\text{MPS}} = \sum_{n=1}^{N} g(n) d(n-1) d(n) - \sum_{n=1}^{N-1} d(n)^2. \tag{3.10}$$

Here $g(n)$ and $d(n) \leq \chi$ are, respectively, the physical and internal bond-dimensions at tensor n. $g(n)$ corresponds to the dimension of the original tensor A along index i_n. $d(1)$ and $d(N)$ both equal 1, whilst for $1 < n < N$, $d(n)$ equals the number of singular values kept during the nth singular values decomposition. The first term of Equation (3.10) is the total number of parameters contained within the tensors, while the second is a substraction of the gauge degrees of freedom inherent in the MPS representation [87]. For a sufficiently large χ, the MPS representation of A becomes *exact* and $Q_{\text{MPS}} = \prod_{n=1}^{N} g(n)$.

3.4.3 Canonical form and Schmidt decomposition

The MPS decomposition is naturally related to the Schmidt decomposition presented in Section 2.2. This can be seen by exploiting the gauge freedom of tensor networks to bring the MPS of Equation (3.9) into a mixed canonical form [21] with the canonical centre placed at the nth bond:

$$\begin{aligned} A_{i_1 i_2 \ldots i_N} \approx \sum_{\{\alpha\}=1}^{\leq\chi} & U^{i_1}_{\alpha_1}[1] U^{i_2}_{\alpha_1\alpha_2}[2] \ldots U^{i_n}_{\alpha_{n-1}\alpha_n}[n] S_{\alpha_n\alpha_n} \\ & \times (V^{i_{n+1}}_{\alpha_n\alpha_{n+1}}[n+1])^\dagger (V^{i_{n+2}}_{\alpha_{n+1}\alpha_{n+2}}[n+2])^\dagger \ldots (V^{i_N}_{\alpha_{N-1}}[N])^\dagger. \end{aligned} \tag{3.11}$$

In this form, S is a singular values matrix (whose entries are real and placed along the diagonal in descending order) whilst $U[n]$ and $V[n]$ are both unitary matrices with the properties of

$$\begin{aligned} \sum_{i_n=0}^{g(n)-1} \sum_{\alpha=1}^{\leq\chi} \left(U^{i_n}_{\beta\alpha}[n]\right)^* U^{i_n}_{\alpha\gamma}[n] &= \delta_{\beta\gamma}, \\ \sum_{i_n=0}^{g(n)-1} \sum_{\beta=1}^{\leq\chi} V^{i_n}_{\alpha\beta}[n] \left(V^{i_n}_{\beta\gamma}[n]\right)^* &= \delta_{\alpha\gamma}, \end{aligned} \tag{3.12}$$

where $(\cdot)^*$ denotes the complex conjugate operation and $\delta_{\alpha\beta}$ is the Kronecker delta. This mixed canonical form is in practice reached by first doing a "left to right" sweep across the entire network as described in Section 3.4.1, followed by sweeping from "right to left" through successive SVDs and contractions until the nth bond is reached. The properties in Equation (3.12) ensure that the product of the left-unitary tensors U also produce a single, unitary tensor

$$U^{i_1}[1] U^{i_2}[2] \ldots U^{i_n}[n] = L_{i_1 i_2 \ldots i_n}, \tag{3.13}$$

whilst the product of the right-unitary tensors $V^\dagger$ results in

$$(V^{i_{n+1}}[n+1])^\dagger (V^{i_{n+2}}[n+2])^\dagger \dots (V^{i_N}[N])^\dagger = R^\dagger_{i_{n+1} i_{n+2} \dots i_N}. \tag{3.14}$$

Furthermore, let now $A_{i_1 i_2 \dots i_N}$ be the amplitude tensor of some state $|\Psi\rangle$ such that $|\Psi\rangle = \sum_{i_1 i_2 \dots i_N} A_{i_1 i_2 \dots i_N} |i_1 i_2 \dots i_N\rangle$. $|\Psi\rangle$ is normalised and lives in the complex Hilbert space of $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \dots \mathcal{H}_N$ with $|i_k\rangle \in \mathcal{H}_k$. Combining Equations (3.12) and (3.13) and (3.14) along with Equation (3.11) results in

$$|\Psi\rangle \approx \sum_{\alpha=1}^{\leq \chi} L^\alpha_{i_1 i_2 \dots i_n} |i_1 i_2 \dots i_n\rangle \, S_{\alpha\alpha} R^\alpha_{i_{n+1} i_{n+2} \dots i_N} |i_{n+1} i_{n+2} \dots i_N\rangle \,. \tag{3.15}$$

Defining $\lambda_\alpha = S_{\alpha\alpha}$ and the orthonormal basis functions of $|\psi^A_\alpha\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \dots \mathcal{H}_n$ and $|\psi^B_\alpha\rangle \in \mathcal{H}_{n+1} \otimes \mathcal{H}_{n+2} \otimes \dots \mathcal{H}_N$ as

$$\begin{aligned} |\psi^A_\alpha\rangle &= L^\alpha_{i_1 i_2 \dots i_n} |i_1 i_2 \dots i_n\rangle \,, \\ |\psi^B_\alpha\rangle &= R^\alpha_{i_{n+1} i_{n+2} \dots i_N} |i_{n+1} i_{n+2} \dots i_N\rangle \,, \end{aligned} \tag{3.16}$$

allows Equation (3.15) to be rewritten into

$$|\Psi\rangle \approx \sum_{\alpha=1}^{\leq \chi} \lambda_\alpha |\psi^A_\alpha\rangle \, |\psi^B_\alpha\rangle \,, \tag{3.17}$$

which establishes that the mixed canonical form of the MPS of a quantum state *is* precisely the Schmidt decomposition of Equation (2.6).

This Schmidt decomposition offers a direct insight into why MPSs are an ideal ansatz for area law following 1D quantum systems. Recall from Section 2.3 and Figure 2.1 that locally entangled quantum systems follow an area law where the entanglement entropy S_{AB} of any two bipartitions A and B does not scale with the number of particles N, but is rather bounded by some sufficiently large constant. This implies, when seen in the context of Equations (3.17) and (2.4), that a sufficiently large χ can be found that will adequately capture the entanglement entropy of the system for *any* N. Thus, quantum states whose entanglement is restricted according to an area law can be represented exponentially efficiently as an MPS.

3.4.4 Representative power

While increasing χ sufficiently much will always make the relationship in Equation (3.9) exact, the MPS representation can in fact also be exact even for quite small χ. For

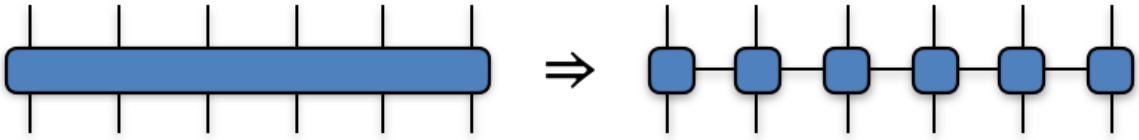

Figure 3.7: **An illustration of the MPO decomposition**. The left object is a tensor representing a 6 site operator. In the right, it has been decomposed into MPO form.

instance, it was previously discussed in this chapter that 1D quantum systems which follow area laws are extremely well described by MPS [17, 19, 21]. Three famous 1-D examples of this are the GHZ state, which can be written as a $\chi = 2$ MPS, and the AKLT and Majumdar-Gosh models, whose ground states can both be captured exactly by MPS of respective bond-dimensions $\chi = 2$ and $\chi = 3$ [17]. Furthermore, mathematicians have also shown that various classical functions, such as trigonometric, polynomial and certain fractal functions, can be exactly represented as MPSs once χ becomes appropriately high [88–90]. For example, any Fourier series $f(x) \in \mathbb{C}, x \in \mathbb{R}$ of order n,

$$f(x) = A_0 + \sum_{i=1}^{n} A_i \sin(k_i x + \phi_i), \quad A_i, k_i, \phi_i \in \mathbb{C}, \tag{3.18}$$

can be represented exactly with $\chi = 2n + 1$, while any polynomial of order n requires $\chi = n + 1$.

3.5 Matrix product operators

Just like matrices can operate upon vectors, the so-called matrix product operators (MPOs) can operate upon MPSs. These tensor networks are defined across the whole N-site Hilbert space and encompass both local (e.g. a Hadamard gate [91] acting on a single qubit) and non-local operations (e.g. quantum Ising Hamiltonian [92] coupling N qubits). The crucial advantage of MPOs is that their variable numbers can grow logarithmically in the size of the Hilbert space (just as with MPS).

3.5.1 Decomposition

MPOs can be produced in a manner analogous to the MPS decomposition in Section 3.4.1. Let some N-site operator be represented through the order $2N$ tensor of $O^{j_1 j_2 \cdots j_N}_{i_1 i_2 \ldots i_N}$. An SVD between the first pair of sites of this tensor and the rest results in

$$O^{j_1 j_2 \ldots j_n}_{i_1 i_2 \ldots i_N} \approx \sum_{\alpha_1=1}^{\leq c} A^{i_1 j_1}_{\alpha_1}[1] S_{\alpha_1 \alpha_1}[1] \big(V^{j_2 j_3 \ldots j_N}_{\alpha_1 i_2 i_3 \ldots i_N}[2]\big)^\dagger, \tag{3.19}$$

with c being the bond-dimension, i.e. maximum number of singular values kept. Continuing the SVDs in a manner identical to Section 3.4.1 will result in the MPO

$$O^{j_1 j_2 \ldots j_n}_{i_1 i_2 \ldots i_N} \approx \sum_{\{\alpha_n\}=1}^{\leq c} A^{i_1 j_1}_{\alpha_1}[1] A^{i_2 j_2}_{\alpha_1 \alpha_2}[2] \ldots A^{i_N j_N}_{\alpha_{N-1}}[N]. \tag{3.20}$$

This MPO decomposition is illustrated graphically in Figure 3.7.

Note that directly performing this MPO decomposition carries an *exponential* computational complexity in N (just as with the MPS decomposition). For actual simulations, MPOs should not be produced in this manner – instead, they ought to be initialised tensor by tensor at a polynomial cost. See for instance how we initialise our finite difference MPOs in Section 4.5.

3.5.2 Number of variables

The number of variables contained within the MPO of Equation (3.20) only grows *polynomially* in N when c is set to be constant, which is an exponential reduction compared to the number of variables within O. This exponential compression is essentially identical to that of the MPS representation in Sections 3.4.1 and 3.4.2.

For the sake of concreteness we now provide the exact expression for the number of physical variables in a MPO. Let $g(n)$ be the dimension of the open bond index i_n and $h(n)$ be that of j_n, and $d(n) \leq c$ the dimension of the internal bond index α_n. Then the number of variables of the MPO is given by

$$Q_{\text{MPO}} = \sum_{n=1}^{N} g(n)h(n)d(n-1)d(n) - \sum_{n=1}^{N-1} d(n)^2. \tag{3.21}$$

Here $d(1) = d(N) = 1$, whilst for $1 < n < N$, $d(n)$ equals the number of singular values kept during the nth singular values decomposition. The first term is the total number of parameters contained within the MPO, while the second is a substraction of the gauge degrees of freedom (just as with MPS in Section 3.4.2). For sufficiently large c, $Q_{\text{MPO}} = \prod_{n=1}^{N} g(n)h(n)$ and the MPO representation of O becomes *exact*.

3.5.3 Operating upon matrix product states

The MPO of Equation (3.20) can operate upon a corresponding N-site MPS at *logarithmic* (i.e. $\sim \text{poly}(N)$) cost. To illustrate this, assume for simplicity that the open bonds of the above MPO and the MPS in Equation (3.9) all have dimension g and then connect them together as illustrated in Figure 3.8a. If the respective bond-dimensions of the MPS and MPO are χ and c, it is straightforward to show that the computational cost of performing the contraction in Figure 3.8b goes as $\sim Ng^2\chi^2c^2$. In comparison, if the MPO and MPS were in the form of a matrix and vector, contracting them would be equivalent to performing a matrix-vector multiplication at the exponential cost of $\sim g^{2N}$, or $\sim g^N$ if the matrix happens to be sparse.

Contracting an MPS with an MPO will, however, lead to its bond-dimension growing from χ to $c\chi$. Repeatedly performing such contractions, as is prominently done in e.g. the time-evolving block decimation algorithm, therefore results in both the bond-dimension and the computational cost increasing exponentially with runtime. Fortunately however, $c\chi$ can often (but not always) be truncated back down to χ without significant loss of accuracy. The naive way of doing this is simply setting the MPS to either left or right canonical form and then performing a sweep of truncating SVDs [93] at a cost of $\sim Ngc^3\chi^3$, assuming $c < \chi$. A more efficient strategy, which reduces the cost down to $\sim Ngc\chi^3$, when $c < \chi$, is the zip-up algorithm [93]. The zip-up algorithm works by sweeping through the MPS-MPO network from left to right (or vice-versa), where at every step a connected pair of MPS and MPO nodes are contracted followed by an SVD truncation of the bond-dimension down to an intermediate $\widetilde{\chi} \approx 2\chi$ (immediately truncating down to χ might lead to an unacceptable loss of accuracy since to the canonical form was broken by the contraction). Upon reaching the right boundary, a leftwards return sweep of SVDs truncates the bond-dimension fully down to the original χ. Moving on, it is also possible to employ variational schemes to keep the bond-dimension from increasing. The cost then

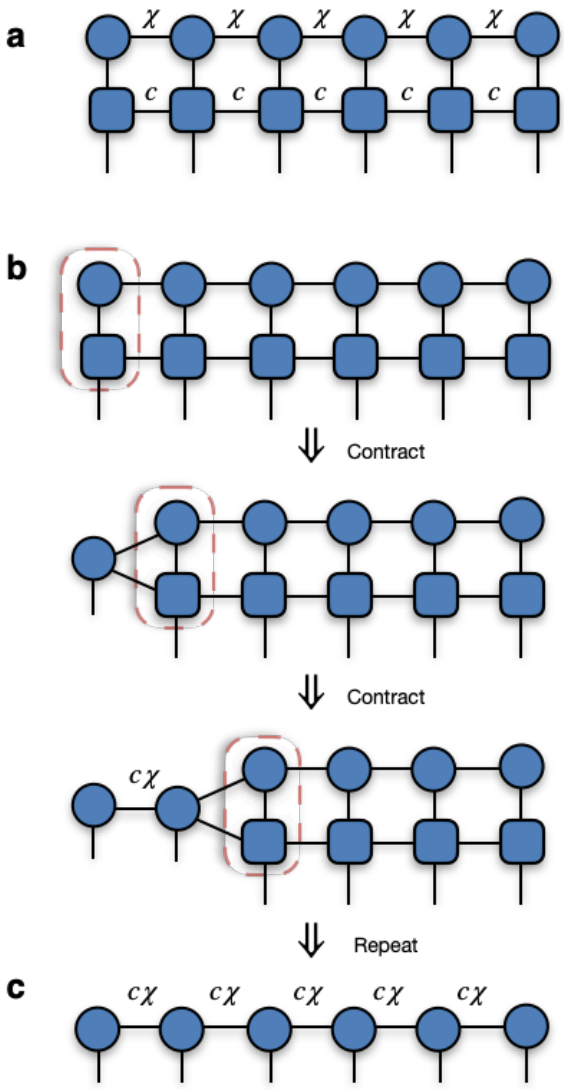

Figure 3.8: **An MPS-MPO contraction**. **a** shows a tensor network consisting of a six site MPS (circular nodes) connected to a corresponding MPO (square nodes). The internal bonds of the MPS have dimension χ and those of the MPO have c, and so if the contraction shown in **b** is done *exactly*, the internal bond-dimension of the resulting MPS in **c** becomes $c\chi$.

scales as $\sim N\text{poly}(g)(c^2\chi^2 + c\chi^3)$ [93], with the dependence on g given by how many variational sites are employed during each update.

MPS-MPO contractions are just one of many MPO operations. Others operations include MPO-MPO addition, MPO-MPO contractions, elementwise MPO-MPO multiplication and so on. There also exist algorithms devised for time-evolving MPOs [94]. More detailed discussions on these operations are available in e.g. [17, 21].

3.6 Conclusion

This chapter covers the background and motivation behind tensor networks. Namely, that tensor networks originated from quantum theory as a means to represent and simulate at *polynomial* cost many-body systems living in *exponentially* large Hilbert spaces. A particularly well-known and simple tensor network is the MPS ansatz, which has seen enormous success in the study of strongly correlated 1-D quantum systems. MPS is thoroughly covered in this chapter due to it being the central tensor network used in this thesis. In the next chapter we explain why by demonstrating how fluid flow fields can be encoded in a scale-by-scale manner using MPSs.

Chapter 4

Encoding flow fields using matrix product states

4.1 Introduction

An important problem of quantum physics is how to deal with the N-body quantum state. While representing it as a vector in Hilbert space is *mathematically* convenient, such a representation (as discussed in the previous chapters) is extremely impractical *numerically* due to the $\sim \exp(N)$ growth of the Hilbert space. Fortunately however, not all regions of the Hilbert space are created equal: realistic quantum states are often structured in a manner that restricts them to a tiny, exponentially small fraction of the full Hilbert space (see Chapter 1 and [12]). The elements of this volume of realistic states can be numerically encoded and manipulated at $\sim \text{poly}(N)$ cost using either quantum circuits on quantum machines, or, in certain cases, tensor networks on classical computers.

Similarly in fluid dynamics, while it is mathematically convenient to represent turbulence as vector fields discretised on a Cartesian grid, it is completely impractical to do so numerically due to the multiscale nature of turbulent flows: in turbulence, the separation between the largest scales l and the smallest η typically goes as $\sim l/\eta \sim \text{Re}^{3/4}$. Thus, straightforwardly discretising a K-dimensional velocity field on a Cartesian grid will require $M \sim (l/\eta)^K \sim \text{Re}^{3K/4}$ gridpoints. This severe scaling with Re will quickly render M infeasibly large, as demonstrated in the transport aircraft example of Chapter 1. In essence, representing turbulence on standard grids faces broadly

similar numerical difficulties to encoding quantum states as vectors in Hilbert space, due to neither of these representations accounting for underlying physical structures[1].

As argued in Chapter 2, physical structures reflecting the phenomena of locality appear in both turbulence and many-body quantum physics. In many important quantum systems, locality of interactions between quantum particles leads to the entanglement between them also becoming local in turn, while a salient feature of turbulent flows is the scale-locality of the turbulent energy cascade, in the sense that eddies at a given length scale predominately interact only with other eddies of similar scale. This scale-locality (as discussed in Section 2.4) justifies the universality hypothesis of turbulence, which supposes that as the Reynolds number increases, the eddies at the finest scales become increasingly uncorrelated to those at the coarse scales, and thus that the flow dynamics of the finest scales are *universal* across all flows at sufficiently high Re. This is reminiscent of a quantum system wherein no entanglement exists between far-away particles, i.e. where the entanglement is short-range and structured according to an area law (recall Section 2.3).

The method of tensor networks is able to simulate many-body quantum systems at polynomial cost (in contrast to the exponential cost of direct methods) precisely when the entanglement between particles is local (or just plain weak). Given that turbulence is itself characterised by locality, but between eddies at different length scales instead of between quantum particles, it follows that this locality is associated with flow structures that tensor networks can be used to exploit, as discussed in Sections 2.4 through 2.8. Indeed, Section 2.8 establishes that specifically the MPS ansatz, the simplest of tensor networks, can be used to encode turbulent flow fields in a way that systematically discards unimportant interscale correlations, allowing for a data-compressed representation of turbulence that exploits its scale-local nature.

4.1.1 Chapter structure

Following the suggestions of Section 2.8, this Chapter presents a scale-by-scale MPS encoding of flow fields that is meant to exploit the scale-local structure of turbulence. The chapter begins with Section 4.2 where a scale-by-scale decomposition of scalar functions is described. The decomposition is used in the following Section 4.3 to

[1]However, recall from Chapters 1, 2 and 3 that there do exist schemes in both computational quantum physics and computational fluid dynamics that aim at exploiting structure.

decompose vector fields. The next pair of sections describe procedures required for the CFD algorithm outlined in Chapter 5: in Section 4.4, a strategy for fine-graining MPS representations of vector fields is provided. Section 4.5 describes how to represent finite difference operators as MPOs. Then Section 4.6 describes how to perform Hadamard multiplication of two MPSs. Finally, this chapter is concluded in Section 4.7.

4.2 Scalar functions as matrix product states

In this section we derive the MPS decomposition for a general K-dimensional scalar function $f(\boldsymbol{r})$. Let us begin by discretising $f(\boldsymbol{r})$ onto a K-dimensional cube of edge length L_{box} where each spatial dimension is discretised by 2^N gridpoints. The whole K-dimensional Cartesian grid thus comprises $M = 2^{KN}$ gridpoints $\boldsymbol{r}_q$ and f is now discretised onto the gridpoint vectors $\boldsymbol{r}_q$ as $f(\boldsymbol{r}_q)$. These $\boldsymbol{r}_q$ can be defined through a tuple of positive integers $(q^1, q^2, \ldots, q^K)$ as:

$$\boldsymbol{r}_q = L_{\text{box}} \sum_{i=1}^{K} \frac{q^i}{2^N} \boldsymbol{e}_i, \quad q^i \in \{0, \ldots, 2^N - 1\}, \tag{4.1}$$

with q^i simply being the index of the gridpoint in the direction $\boldsymbol{e}_i$. The binary representation $(\ldots)_2$ of these indices q^i requires N bits,

$$q^i = \left(b_1^i, b_2^i, \ldots, b_N^i\right)_2 , \tag{4.2}$$

where $b_n^i \in \{0, 1\}$, $n = 1, \ldots, N, i \in \{1, \ldots, K\}$ and b_1^i and b_N^i are the most and least significant bits, respectively.

The mappings of Equations (4.1) and (4.2) imply that $f(\boldsymbol{r}_q)$ can equivalently be written as a function of the bits, i.e. $f(\boldsymbol{r}_q) \equiv f(b_1^1, \ldots, b_N^K)$. There are however many ways to map b_n^i into $\boldsymbol{r}_q$. Because each bit is associated with a length scale, one sensible mapping is to combine all indices associated with the same bit into an object

$$\omega_n = \left(b_n^1, b_n^2, \ldots, b_n^K\right)_2 , \tag{4.3}$$

such that $\omega_n \in \{0, \ldots, 2^K - 1\}$. The object ω_n is a composed of K bits. Thus for $K = 1$ it is just a bit, for $K = 2$ a quartenary, for $K = 3$ an octal, and so on. This "sensible" grouping of bits is the one we employ throughout this thesis.

Numerically, $f(\boldsymbol{r}_q)$ can be represented in a order K tensor A of M elements as:

$$f(\boldsymbol{r}_q) = A_{q_1 q_2 \ldots q_K}. \tag{4.4}$$

By exploiting the relations in Equations (4.2) and (4.3), $f(\boldsymbol{r}_q)$ can be reformulated into

$$f(\boldsymbol{r}_q) = f(\omega_1, \omega_2, \ldots, \omega_N) = A_{\omega_1\omega_2\ldots\omega_N}. \tag{4.5}$$

Representing $f(\boldsymbol{r}_q)$ in this form is equivalent to decomposing the spatial grid into N separate length scales – thus we have gone from discretising f in terms of the gridpoint vectors $\boldsymbol{r}_q$ to instead discretising f in terms of length scales ω_n.

We now adopt the bra-ket framework of quantum physics. This will become useful later on as we transfer the machinery of MPS from quantum physics to fluid dynamics. In this bra-ket form, $f(\boldsymbol{r}_q)$ can be expressed as

$$|u\rangle = \sum_{\{\sigma_n\}=0}^{2^K-1} A_{\sigma_1\sigma_2\ldots\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle\,, \tag{4.6}$$

where $|\sigma_n\rangle$ is a vector living in some Hilbert space $\mathcal{A}'$ defined through $\text{span}(\mathcal{A}') = \{|0\rangle, |1\rangle, \ldots, |2^K-1\rangle\}$, with $|i\rangle$ being a set of orthonormal basis vectors such that $\langle i|j\rangle = \delta_{ij}$. Thus, $|u\rangle$ is an element of the composite Hilbert space $\mathcal{A} = (\mathcal{A}')^{\otimes N}$ containing M elements. $f(\boldsymbol{r}_q) = f(\omega_1, \omega_2, \ldots, \omega_N)$ is recoverable through the inner-product of

$$\langle\omega_1\omega_2\ldots\omega_N|u\rangle = A_{\omega_1\omega_2\ldots\omega_N} = f(\omega_1, \omega_2, \ldots, \omega_N). \tag{4.7}$$

Finally, note that while Equation (4.6) is in the form of a quantum state, in this case $|u\rangle$ encodes the discretised function $f(\boldsymbol{r}_q)$ instead of some quantum state.

The representations of f in Equations (4.5) and (4.6) show that $f = f(\omega_1, \omega_2, \ldots, \omega_N)$ is an N-dimensional function. The recipe of Section 3.4.1 allows A to be represented in the form of the N-site MPS

$$A_{\omega_1\omega_2\ldots\omega_N} \approx \sum_{\{\alpha_n\}=1}^{\leq\chi} A^{\omega_1}_{\alpha_1}[1] A^{\omega_2}_{\alpha_1\alpha_2}[2] \ldots A^{\omega_{N-1}}_{\alpha_{N-2}\alpha_{N-1}}[N{-}1] A^{\omega_N}_{\alpha_{N-1}}[N], \tag{4.8}$$

where the tensor $A[n]$ encapsulates f at the $L_{\text{box}}/2^n$ length scale and is of dimension $d(n{-}1){\times}2^K{\times}d(n)$, with $d(n) = \min\left(\chi, 2^{Kn}, 2^{K(N-n)}\right)$. In bra-ket form, Equation (4.8) becomes

$$\begin{aligned} &|u\rangle \approx |v(\chi)\rangle = \\ &\sum_{\{\alpha_n\}=1}^{\leq\chi} \sum_{\{\sigma_N\}=0}^{2^K-1} A^{\sigma_1}_{\alpha_1}[1] A^{\sigma_2}_{\alpha_1\alpha_2}[2] \ldots A^{\sigma_{N-1}}_{\alpha_{N-2}\alpha_{N-1}}[N{-}1] A^{\sigma_N}_{\alpha_{N-1}}[N] \,|\sigma_1\sigma_2\ldots\sigma_N\rangle\,. \end{aligned} \tag{4.9}$$

$|v(\chi)\rangle$ is now a vector within the MPS manifold of $\mathcal{B}(\chi) \subseteq \mathcal{A}$ restricted by the bond-dimension χ.

The number of variables in the MPS representation of f scales as $\sim N2^K\chi^2$. When $\chi, K \sim$ const, this becomes an exponential reduction in N compared to the $M = 2^{KN}$ variables of the original representation. Furthermore, if χ is set to (the maximal value of) $\chi = 2^{K\lfloor N/2 \rfloor}$, $\mathcal{B}(2^{K\lfloor N/2 \rfloor}) = \mathcal{A}$ and the relationships in Equations (4.8) and (4.9) become exact. Cf. with Equation (2.20) of Section 2.5 and note that $2^{K\lfloor N/2 \rfloor} = \Gamma^{K\text{-D}}(\lfloor N/2 \rfloor)$. The precise number of variables contained within this MPS representation is found through Equation (3.10).

4.3 Vector fields as matrix product states

Let $\mathbf{V}(\mathbf{r}) = \mathbf{e}_1 u_1(\mathbf{r}) + \mathbf{e}_2 u_2(\mathbf{r}) + \cdots + \mathbf{e}_K u_K(\mathbf{r})$ be a K dimensional continuous vector field. Following the recipe outlined in the previous Section 4.2 permits the discretisation of $\mathbf{V}$ into $\mathbf{V} \to \sum_{i=1}^{K} \boldsymbol{e}_i |u_i\rangle$. After doing this, we perform a component by component MPS decomposition at a pre-chosen bond-dimension χ, resulting in

$$\begin{aligned} &\sum_{i=1}^{K} \boldsymbol{e}_i |u_i\rangle \approx \sum_{i=1}^{K} \boldsymbol{e}_i |v_i(\chi)\rangle = \\ &\sum_{i=1}^{K} \boldsymbol{e}_i \sum_{\{\alpha_n\}=1}^{\leq\chi} \sum_{\{\sigma_n\}=0}^{2^K-1} A^{\sigma_1}_{\alpha_1}[1,i] A^{\sigma_2}_{\alpha_1\alpha_2}[2,i] \ldots A^{\sigma_N}_{\alpha_{N-1}}[N,i] \, |\sigma_1\sigma_2\ldots\sigma_N\rangle \, . \end{aligned} \tag{4.10}$$

The vector field $\sum_{i=1}^{K} \boldsymbol{e}_i |u_i\rangle$ lives in $\mathcal{D} = \mathcal{A}^{\oplus K}$, while $\sum_{i=1}^{K} \boldsymbol{e}_i |v_i\rangle$ is in the MPS manifold $\mathcal{M}(\chi) = \mathcal{B}(\chi)^{\oplus K} \subseteq \mathcal{D}$ bounded by χ. Furthermore, when χ is maximal, the relationship in Equation (4.10) is exact and $\mathcal{M} = \mathcal{D}$. We have now generalised the MPS decomposition to also include vector fields.

4.4 Fine-graining of vector fields

Imagine we want to exploit the exponential compression of the MPS representation to produce an MPS algorithm of cost $\sim \text{poly}(N) \sim \log(M)$ for e.g. time-evolving some initial velocity field. The first step of such an algorithm must be to prepare the initial MPS. To do this, it is not sufficient to merely discretise the initial vector field $V(t = 0)$ on a cube with $M = 2^{KN}$ gridpoints and then MPS decompose it, as that would lead

to a $\sim \exp(N)$ computational cost [Section 3.4.1]. Instead, a $\sim \text{poly}(N)$ algorithm for preparing the initial MPS must be used.

The simplest strategy for efficiently preparing the initial MPS is to write it out tensor-by-tensor. This is reasonable when the initial state can be expressed as a simple MPS of limited χ, as is the case for e.g. polynomials or Fourier series (see Section 3.4.4). When the initial state is more complicated, this strategy becomes infeasible.

A more general strategy is that of *fine-graining* (or prolongation). The first step in this strategy goes as follows: the initial velocity field $\mathbf{V}(t=0,\mathbf{r}) = \mathbf{e}_1 u_1(t=0,\mathbf{r}) + \mathbf{e}_2 u_2(t=0,\mathbf{r}) + \cdots + \mathbf{e}_k u_k(t=0,\mathbf{r})$ is decomposed into a MPS with bond-dimension χ using the procedure of Section 4.3, but on a *rough* grid where each dimension is discretised using 2^{N_R} gridpoints instead of the full 2^N, with $0 < N_R < N$. This makes $\mathbf{V}(t=0,\mathbf{r}) \to \sum_{i=1}^{K} \boldsymbol{e}_i \left|v_i^R(\chi)\right\rangle$ with

$$\left|v_i^R\right\rangle = \sum_{\{\alpha_n\}=1}^{\leq\chi} \sum_{\{\sigma_n\}=0}^{2^K-1} A^{\sigma_1}_{\alpha_1}[1,i] A^{\sigma_2}_{\alpha_1\alpha_2}[2,i] \ldots A^{\sigma_{N_R}}_{\alpha_{N_R-1}}[N_R,i] \left|\sigma_1\sigma_2\ldots\sigma_{N_R}\right\rangle . \tag{4.11}$$

The cost of this MPS decomposition is $\sim \min\left(\chi^2 2^{K\lfloor N_R/2\rfloor}, \chi 2^{KN_R}\right) \sim \exp(N_R)$, which is still exponential, but in N_R instead of N. The second (and final) step is to use interpolation to *prolongate* $\left|v_i^R\right\rangle$ onto the full grid wherein each direction is discretised with 2^N gridpoints. The number of gridpoints used to represent every $\left|v_i^R\right\rangle$ can straightforwardly be doubled by contracting it with the prolongation MPO

$$\begin{aligned}
&\left[P^m\right]^{\tau_1\tau_2\ldots\tau_m\tau_{m+1}}_{\sigma_1\sigma_2\ldots\sigma_m} = \\
&\sum_{\{\beta_n\}=1}^{c} \sum_{\{\tau_n\}=0}^{2^K-1} \sum_{\{\sigma_n\}=0}^{2^K-1} L_{\beta_0} B^{\tau_1\sigma_1}_{\beta_0\beta_1} B^{\tau_2\sigma_2}_{\beta_1\beta_2} \ldots B^{\tau_m\sigma_m}_{\beta_{m-1}\beta_m} R^{\tau_{m+1}}_{\beta_m} \left|\tau_1\tau_2\ldots\tau_{m+1}\right\rangle \left\langle\sigma_1\sigma_2\ldots\sigma_m\right| ,
\end{aligned} \tag{4.12}$$

with c being its bond-dimension and L, B and R defined in Equation (4.14) below, as

$$\begin{aligned}
&P^{N_R}\left|v_i^R\right\rangle = \\
&\sum_{\{\alpha_n\}=1}^{\leq\chi} \sum_{\{\beta_n\}=1}^{c} \sum_{\{\sigma_n\}=0}^{2^K-1} \sum_{\{\tau_n\}=0}^{2^K-1} A^{\sigma_1}_{\alpha_1}[1,i] A^{\sigma_2}_{\alpha_1\alpha_2}[2,i] \ldots A^{\sigma_{N_R}}_{\alpha_{N_R-1}}[N_R,i] \\
&L_{\beta_0} B^{\tau_1\sigma_1}_{\beta_0\beta_1} B^{\tau_2\sigma_2}_{\beta_1\beta_2} \ldots B^{\tau_{N_R}\sigma_{N_R}}_{\beta_{N_R-1}\beta_{N_R}} R^{\tau_{N_R+1}}_{\beta_{N_R}} \left|\tau_1\tau_2\ldots\tau_{N_R+1}\right\rangle .
\end{aligned} \tag{4.13}$$

However, carrying out this contraction exactly will increase the bond-dimension from χ to $c\chi$, which is not sustainable for the reasons provided in Section 3.5.3. To keep the bond-dimension from growing, one can instead employ approximate contractions

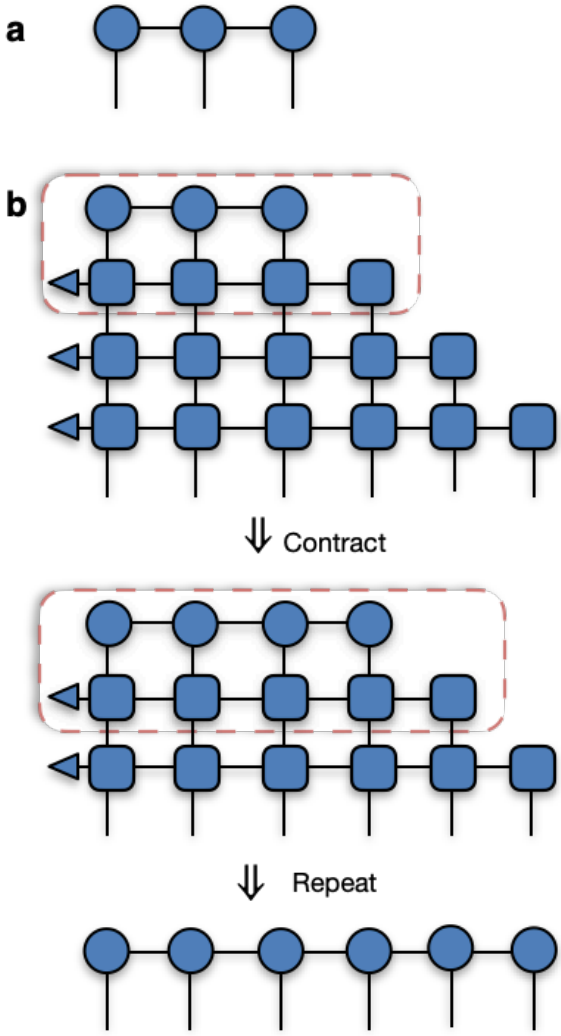

Figure 4.1: **Fine-graining of a MPS**. The length $N_R = 3$ MPS (circles) in **a** is a scalar function on a rough K-dimensional grid wherein each direction is discretised with 2^{N_R} gridpoints. In **b**, the MPS is mapped onto a refined grid where each direction is discretised with $2^N = 2^6$ gridpoints. This is done by applying prolongation MPOs (triangles and squares) in the manner of Equation (4.13) $N - N_R = 3$ times.

through e.g. the zip-up algorithm (see Section 3.5.3) at a cost of $\sim Ngc\chi^3$. Controlling the bond-dimension in this way whilst performing the contraction in Equation (4.13) will fine-grain the initial state by doubling the number of gridpoints (from 2^{KN_R} to 2^{K2N_R}) without altering the bond-dimension. Repeatedly applying this recipe $N - N_R$ times will map the $|v_i^R\rangle$ velocity components onto an *exponentially* large grid ($M = 2^{KN}$ gridpoints) at *polynomial* cost, thus completing the preparation of the initial state. The full algorithm is illustrated diagrammatically in Figure 4.1.

The tensors L_{β_0}, $B^{\tau_n\sigma_n}_{\beta_n\beta_{n+1}}$ and $R^{\tau_{m+1}}_{\beta_m}$ that constitute P^m differ depending on the geometry and preferred interpolation scheme. For example, in the case of periodic boundary conditions and $K = 1$, P^m can be made to perform linear interpolation by setting every element in the three tensors to zero except

$$\begin{aligned} L_1 &= L_2 = 1, \\ B^{00}_{11} &= B^{11}_{11} = 1, \\ B^{10}_{22} &= B^{01}_{12} = 1, \\ R^0_1 &= 1, \\ R^1_1 &= R^1_2 = 1/2, \end{aligned} \tag{4.14}$$

as noted in [62, pp. 588 – 590]. Storing these values requires the bond-dimension of P^m to be $c = 2$.

The reason why L, B and R take the values they do in Equation (4.14) can be understood by viewing the MPO as a finite state machine [77] whose instructions are initiated by the right-most tensor R. This means that at the *old* gridpoints corresponding to $\tau_{m+1} = 0$, $R = 1$ for $\beta_m = 1$ (and is otherwise zero), which instructs the remaining tensors B, L to not perform any interpolation. For the *new* gridpoints corresponding to $\tau_{m+1} = 1$, however, $R = 1/2$ for both $\beta_m = 0, 1$ and it can be shown that this instructs the remainder of the MPO to calculate the values at the new gridpoints using second order accurate linear interpolation. A formal recipe for generating prolongation MPOs with higher-order interpolation is provided in [95, pp. 747 – 748].

4.5 Differentiation with matrix product operators

While it is now clear how vector fields may be represented in MPS form, to solve partial differential equations inside the MPS manifold it must also be possible to differentiate MPSs. In this thesis, all such differentiation is carried out using finite difference operators recast into the form of MPOs.

In the standard method of finite differences, these operators are represented as sparse matrices. Recall for instance $f(\boldsymbol{r}_q)$ from Equation (4.4) and let $\boldsymbol{f} = \mathrm{vec}\,(f(\boldsymbol{r}_q))$ be an array populated by every value of the function at each gridpoint $\boldsymbol{r}_q$. Setting $K = 1$ and assuming periodic boundary conditions, the function can be differentiated using

for example a 2nd-order accurate central finite difference scheme by left-multiplying $\boldsymbol{f}$ with the matrix

$$D = \frac{1/2}{L_{\text{box}}/2^N} \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}, \tag{4.15}$$

giving the discrete derivative $D\boldsymbol{f}$.

Such differentiation can also straightforwardly be carried out in the MPS manifold. Take $|v(\chi)\rangle$ from Equation (4.9). Assuming, again, $K = 1$ and periodic boundary conditions, D from the above Equation (4.15) can be reformulated into the MPO

$$D^{\sigma_1\sigma_2\dots\sigma_N}_{\tau_1\tau_2\dots\tau_N} = \sum_{\{\alpha_n\}=1}^{c} \sum_{\{\tau_n\}=0}^{1} \sum_{\{\sigma_n\}=0}^{1} L_{\alpha_0} B^{\tau_1\sigma_1}_{\alpha_0\alpha_1} B^{\tau_2\sigma_2}_{\alpha_1\alpha_2} \dots B^{\tau_N\sigma_N}_{\alpha_{N-1}\alpha_N} R_{\alpha_N} |\tau_1\tau_2\dots\tau_N\rangle \langle\sigma_1\sigma_2\dots\sigma_N|, \tag{4.16}$$

and contracted with $|v(\chi)\rangle$ to calculate the derivative $D|v(\chi)\rangle$. The tensors $L^{\tau_n\sigma_n}_{\alpha_n}$, $B^{\tau_n\sigma_n}_{\alpha_n\alpha_{n+1}}$ and $R^{\tau_n}_{\alpha_n}$ constituting D are now given by

$$\begin{aligned} &L_1 = L_2 = L_3 = 1, \\ &B^{00}_{11} = B^{11}_{11} = B^{22}_{11} = B^{33}_{11} = 1, \\ &B^{10}_{12} = B^{01}_{22} = B^{10}_{33} = B^{01}_{13} = 1, \\ &R_1 = \frac{1/2}{L_{\text{box}}/2^N}, R_2 = \frac{1/2}{L_{\text{box}}/2^N}. \end{aligned} \tag{4.17}$$

with the remaining elements all being zero. Storing these values requires the bond-dimension of D to be $c = 3$, which, in fact, also holds for $K > 1$. A general framework for generating MPOs out of tridiagonal Toeplitz matrices (which is how finite difference operators are traditional expressed) is provided in [95, pp. 747 – 751]. See also [62, p. 591] and [96, p. 22] for further information on how the method of finite differences can be implemented in MPO form.

4.6 Hadamard multiplication

To solve *linear* partial differential equations in MPS form, it is in principle sufficient to know how to represent vector fields as MPSs and differentiate them while they are in

that form. However, the Navier-Stokes equations are *nonlinear* due to the presence of the convective term (namely $(\mathbf{V}.\nabla)\mathbf{V}$ from Equation (2.8)). As will become apparent in Chapter 5, calculating such terms requires the use of the Hadamard product.

The Hadamard product is nothing other than the elementwise product of two arrays. If $\boldsymbol{f}$ and $\boldsymbol{g}$ are two arrays of length 2^N, then the Hadamard product

$$\mathbf{h} = \mathbf{f} \odot \mathbf{g} \tag{4.18}$$

between them is defined as

$$h_i = f_i g_i, \quad i \in \{1, \ldots, 2^N\}, \tag{4.19}$$

with h_i, f_i and g_i being the ith elements of respectively $\mathbf{h}$, $\mathbf{f}$ and $\mathbf{g}$.

This definition carries over for MPSs. To see, let $\mathbf{f}$ and $\mathbf{g}$ be encoded per Section 4.2 into the two respective MPSs

$$\begin{aligned} |f(\chi)\rangle &= \sum_{\{\alpha_n\}=1}^{\leq\chi} \sum_{\{\sigma_n\}=0}^{2^K-1} A^{\sigma_1}_{\alpha_1}[1] A^{\sigma_2}_{\alpha_1\alpha_2}[2] \ldots A^{\sigma_N}_{\alpha_{N-1}}[N] \, |\sigma_1\sigma_2\ldots\sigma_N\rangle \,, \\ |g(\chi)\rangle &= \sum_{\{\beta_n\}=1}^{\leq\chi} \sum_{\{\tau_n\}=0}^{2^K-1} B^{\tau_1}_{\beta_1}[1] B^{\tau_2}_{\beta_1\beta_2}[2] \ldots B^{\tau_N}_{\beta_{N-1}}[N] \, |\tau_1\tau_2\ldots\tau_N\rangle \,, \end{aligned} \tag{4.20}$$

both of bond-dimension χ (for the sake of simplicity). The elementwise product

$$|h(\chi')\rangle = |f(\chi)\rangle \odot |g(\chi)\rangle \tag{4.21}$$

between them can straightforwardly be calculated by inserting Kronecker-delta tensors between each of the N sites of $|f(\chi)\rangle$ and $|g(\chi)\rangle$ as

$$\begin{aligned} &|h(\chi')\rangle = \\ &\sum_{\{\alpha_n\}=1}^{\leq\chi} \sum_{\{\beta_n\}=1}^{\leq\chi} \sum_{\{\sigma_n\}=0}^{2^K-1} \sum_{\{\tau_n\}=0}^{2^K-1} \sum_{\{\upsilon_n\}=0}^{2^K-1} A^{\sigma_1}_{\alpha_1}[1] A^{\sigma_2}_{\alpha_1\alpha_2}[2] \ldots A^{\sigma_N}_{\alpha_{N-1}}[N] \\ &\delta_{\sigma_1\tau_1\upsilon_1}\delta_{\sigma_2\tau_2\upsilon_2}\ldots\delta_{\sigma_N\tau_N\upsilon_N} B^{\tau_1}_{\beta_1}[1] B^{\tau_2}_{\beta_1\beta_2}[2] \ldots B^{\tau_N}_{\beta_{N-1}}[N] \, |\upsilon_1\upsilon_2\ldots\upsilon_N\rangle \,, \end{aligned} \tag{4.22}$$

with all elements of the Kronecker-delta tensor δ_{ijk} equalling zero except when $i = j = k$, in which case $\delta_{ijk} = 1$. Equation (4.22) is illustrated in Figure 4.2.

The computational cost of calculating $|h(\chi')\rangle$ scales with the bond-dimension as $\sim \chi^4$ when using the zip-up or variational algorithms of Section 3.5.3. These algorithms

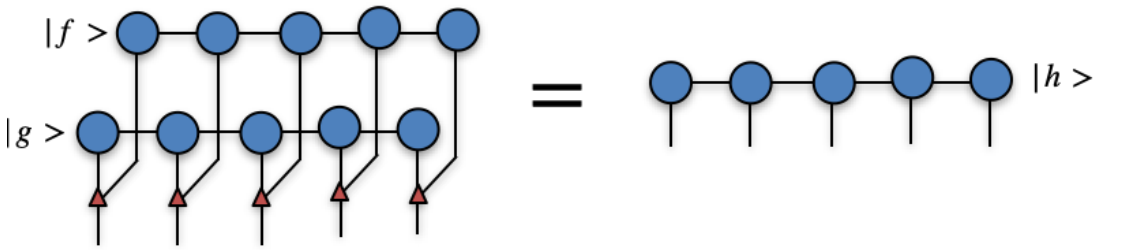

Figure 4.2: **Elementwise product between two MPSs**. This figure illustrates the elementwise product of $|h\rangle = |f\rangle \odot |g\rangle$ as seen in Equation (4.22) for $N = 5$. The red triangles denote Kronecker-delta tensors.

can perform the contraction either exactly or approximately, but doing it exactly will make the bond-dimension grow to an unacceptable $\chi' = \chi^2$. Fortunately, for all the flow cases discussed in this thesis, we find (using a variational scheme) that these contractions can be performed approximately such that $\chi' = \chi$ without significantly affecting accuracy.

4.7 Conclusion

While the MPS ansatz is traditionally used to simulate quantum systems, this chapter demonstrates how also flow fields can be encoded in MPS form. The encoding is done in a scale-by-scale fashion suitable for exploiting the scale-local structures of turbulence along the lines recommended in Section 2.8. Furthermore, it is also explained how to fine-grain and differentiate the MPS-encoded flow fields entirely within the MPS manifold, and perform elementwise multiplication between MPSs. These procedures open the way for the MPS algorithm of the following Chapter 5.

Chapter 5

A matrix product state algorithm for fluid simulation

5.1 Introduction

In this chapter a MPS algorithm is presented for simulating turbulence. Chapter 2 (Section 2.8 in particular) suggests the MPS ansatz can be used to exploit turbulence structures, motivating the schemes presented in Chapter 4 for representing and manipulating velocity fields entirely in MPS form. These schemes are in turn used in this chapter to create the MPS algorithm for variationally solving the incompressible Navier-Stokes equations.

5.1.1 Chapter structure

This chapter begins with Section 5.2, where the incompressible Navier-Stokes equations are reformulated into a variational problem. An algorithm for solving this problem is subsequently presented in Section 5.3, and the algorithm's computational complexity and arithmetic intensity are then described and demonstrated in Sections 5.4, 5.5 and 5.6. Section 5.7 explains how the algorithm can be easily modified to also solve Burgers' equation in place of the Navier-Stokes equations. Finally, the chapter is rounded off in Section 5.8.

5.2 Defining the variational problem

For the MPS algorithm to be accurate, real-space must be discretised in such a way that all length scales between ℓ and η are still resolved. We do this by resolving the flow $\mathbf{V}(t, \mathbf{r})$ on a K-dimensional Cartesian grid where each of the K spatial dimensions are discretised with $2^N \approx \ell/\eta$ gridpoints. Then, following the general framework of Chapter 4, we put the incompressible Navier-Stokes equations, i.e. Equations (2.8), into the form

$$\begin{aligned} &\sum_{i=1}^{K} \left(D_i \left|u_i\right\rangle\right) = 0, \\ &\frac{\partial \left|u_i\right\rangle}{\partial t} + D_i \left|p\right\rangle + \sum_{j=1}^{K} \left(\left|u_j\right\rangle \odot D_j \left|u_i\right\rangle - \nu D_j^2 \left|u_i\right\rangle\right) = 0, \end{aligned} \tag{5.1}$$

where $\left[\sum_{i=1}^{K} \mathbf{e}_i \left|u_i(t)\right\rangle\right] \in \mathcal{D}$ is an exact encoding of the flow field in the Hilbert space $\mathcal{D}$ of dimension $K2^{KN}$, D_k and D_k^2 represent respectively the first and second derivatives along unit vector $\boldsymbol{e}_k$ discretised using eight-order-accurate central finite difference stencils [97] and $\odot$ is the Hadamard (elementwise multiplication) operation [Section 4.6].

This discretisation permits us to reformulate the incompressible Navier-Stokes equations into a variational problem on the MPS manifold. We use the procedure in Section 4.3 to represent $\sum_{i=1}^{K} \mathbf{e}_i \left|u_i(t)\right\rangle$ as the MPSs $\left[\sum_{i=1}^{K} \mathbf{e}_i \left|v_i(t,\chi)\right\rangle\right] \in \mathcal{M}(\chi) \subseteq \mathcal{D}$, which is an exact representation if $\chi = \Gamma^{K\text{-D}}(\lfloor N/2 \rfloor)$. After-which we encode D_k and D_k^2 into MPO form per the recipe provided in Section 4.5. Note now that the two coupled Equations (5.1) are both set to zero. This means they can be recast into the variational problem of minimising across time and space the cost-function

$$\begin{aligned} \Omega\left(\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle\right) = \mu \left\| \sum_{i=1}^{K} D_i \left|\tilde{v}_i\right\rangle \right\|^2 + \\ \left\| \sum_{i=1}^{K} \mathbf{e}_i \left[\frac{\partial \left|\tilde{v}_i\right\rangle}{\partial t} + \sum_{j=1}^{K} \left(\left|\tilde{v}_j\right\rangle \odot D_j \left|\tilde{v}_i\right\rangle - \nu D_j^2 \left|\tilde{v}_i\right\rangle\right)\right] \right\|^2 \end{aligned} \tag{5.2}$$

for the trial solution $\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i(t,\chi)\right\rangle \in \mathcal{M}$ to find the solution

$$\sum_{i=1}^{K} \mathbf{e}_i \left|v_i(t,\chi)\right\rangle = \underset{\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle}{\operatorname{argmin}} \Omega\left(\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle\right) \tag{5.3}$$

to the incompressible Navier-Stokes equations on the MPS manifold $\mathcal{M}$. The incompressibility condition is now enforced through the penalty method [98, 99], which replaces the pressure term (whose role in the original equations is solely that of a Lagrange multiplier for the incompressibility condition [43]) by the penalty term $\mu \left\| \sum_{i=1}^{K} D_i \left| \tilde{v}_i \right\rangle \right\|^2$ that now instead enforces incompressibility when the penalty coefficient μ is set to a sufficiently large number. On a final note, while the nonlinear term is here (for the sake of simplicity) represented in convective form as $\sum_{j=1}^{K} \left| \tilde{v}_j \right\rangle \odot D_j \left| \tilde{v}_i \right\rangle$, the skew-symmetric form $\sum_{j=1}^{K} \frac{1}{2} \big[\left| \tilde{v}_j \right\rangle \odot D_j \left| \tilde{v}_i \right\rangle + D_j (\left| \tilde{v}_j \right\rangle \odot \left| \tilde{v}_i \right\rangle) \big]$ should be used during actual numerical simulation [100].

The exact solution to the incompressible Navier-Stokes equations is found when Ω is minimised all the way down to zero (within machine precision), but this is only possible when χ is maximal. If χ is not, what will instead be found is the closest approximation (per the L2 norm) within the MPS manifold of the actual solution. The minimum of Ω is then no longer necessarily zero, but it can always be quantified and controlled (through χ) such that the approximate solution is accurate.

To simulate fluid flows on computers, time must also be discretised. There are two straightforward ways of doing this. The first is to expand on the encoding in Section 4.2 such that the temporal dimension is discretised with 2^N gridpoints similarly to the spatial dimensions, with the different time-scales being incorporated into MPS tensors of matching length-scales. This would expand the MPS manifold to cover all of spacetime and allow the incompressible Navier-Stokes equations to be solved (after recasting the time-derivative into MPO form) as a boundary value problem. However, this approach requires knowing beforehand the configurations of the initial and final flow fields, which is an impractical requirement since typically only the initial field is known. Therefore, for the sake of simplicity, in this thesis we instead went with the second (more traditional) option where the incompressible Navier-Stokes equations are treated as an initial values problem to be solved with a time-marching scheme.

We chose an explicit Runge-Kutta 2 time-marching scheme with which to solve the variational problem. If the initial and final times are set to $t = 0$ and $t = T_f$, time can be discretised into

$$t_q = q\Delta t, \quad q = 0, 1, \ldots, T_f/\Delta t, \tag{5.4}$$

with the number of timesteps $T_f/\Delta t \in \mathbb{Z}^+$. This allows the time derivative in

Equation (5.2) to be approximated to first-order as

$$\frac{\partial \left|\tilde{v}_i(t)\right\rangle}{\partial t} \approx \frac{\left|\tilde{v}_i(t_{q+1})\right\rangle - \left|\tilde{v}_i(t_q)\right\rangle}{\Delta t}. \tag{5.5}$$

Two such steps can then be compounded into a single explicit midpoint Runge-Kutta 2 step:

$$\begin{aligned}
\sum_{i=1}^{K} \mathbf{e}_i \left|v_i^{0.5}\right\rangle &= \underset{\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle}{\operatorname{argmin}} \Theta\left(\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle, \sum_{i=1}^{K} \mathbf{e}_i \left|v_i(t_q)\right\rangle, \sum_{i=1}^{K} \mathbf{e}_i \left|v_i(t_q)\right\rangle, \Delta t/2\right), \\
\sum_{i=1}^{K} \mathbf{e}_i \left|v_i(t_{q+1})\right\rangle &= \underset{\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle}{\operatorname{argmin}} \Theta\left(\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle, \sum_{i=1}^{K} \mathbf{e}_i \left|v_i(t_q)\right\rangle, \sum_{i=1}^{K} \mathbf{e}_i \left|v_i^{0.5}\right\rangle, \Delta t\right),
\end{aligned} \tag{5.6}$$

where $\left|\tilde{v}_i\right\rangle = \left|\tilde{v}_i(\chi)\right\rangle$ is now independent of time and with the variational function Θ given through

$$\begin{aligned}
\Theta\left(\sum_{i=1}^{K} \mathbf{e}_i \left|\tilde{v}_i\right\rangle, \sum_{i=1}^{K} \mathbf{e}_i \left|\alpha_i\right\rangle, \sum_{i=1}^{K} \mathbf{e}_i \left|\beta_i\right\rangle, \tau\right) = \mu \left\| \sum_{i=1}^{K} D_i \left|\tilde{v}_i\right\rangle \right\|^2 + \\
\left\| \sum_{i=1}^{K} \mathbf{e}_i \left[\frac{\left|\tilde{v}_i\right\rangle - \left|\alpha_i\right\rangle}{\tau} + \sum_{j=1}^{K} \left(\left|\beta_j\right\rangle \odot D_j \left|\beta_i\right\rangle - \nu D_j^2 \left|\beta_i\right\rangle \right) \right] \right\|^2.
\end{aligned} \tag{5.7}$$

Performing $T_f/\Delta t$ such Runge-Kutta 2 steps will result in a solution that approximately minimises Ω to second-order accuracy in time.

5.3 Solving the variational problem

Let us now derive the variational algorithm that solves Equation (5.7). The first step is to expand the equation into

$$\begin{aligned}
\Theta = \sum_{i,j=1}^{K} \Bigg\{ & \mu \left\langle \tilde{v}_i \right| D_i^\dagger D_j \left| \tilde{v}_j \right\rangle + \frac{\left\langle \tilde{v}_i | \tilde{v}_i \right\rangle}{\tau^2} \\
& + \frac{\left\langle \tilde{v}_i \right|}{\tau} \left(\frac{-\left|\alpha_i\right\rangle}{\tau} + \left|\beta_j\right\rangle \odot D_j \left|\beta_i\right\rangle - \nu D_j^2 \left|\beta_i\right\rangle \right) \\
& + \left(\frac{-\left|\alpha_i\right\rangle}{\tau} + \left|\beta_j\right\rangle \odot D_j \left|\beta_i\right\rangle - \nu D_j^2 \left|\beta_i\right\rangle \right)^\dagger \frac{\left|\tilde{v}_i\right\rangle}{\tau} \Bigg\} + \Big[\dots \Big],
\end{aligned} \tag{5.8}$$

where $[\dots]$ denotes the remaining part of Θ that is independent of $\left|\tilde{v}_i\right\rangle$ (the contents of $[\dots]$ will later vanish during differentiation). Let now $\left|\tilde{v}_i\right\rangle$ be in a canonical form

with canonical centre at site n, i.e.

$$
\begin{aligned}
|\tilde{v}_i\rangle = \sum_{\{\sigma_n\}=0}^{2^K-1} \sum_{\{\alpha_n\}=1}^{\leq\chi} & U^{\sigma_1}_{\alpha_1}[1,i] U^{\sigma_2}_{\alpha_1\alpha_2}[2,i] \dots U^{\sigma_{n-1}}_{\alpha_{n-2}\alpha_{n-1}}[n-1,i] C^{\sigma_n}_{\alpha_{n-1}\alpha_n}[n,i] \\
& \times (V^{\sigma_{n+1}}_{\alpha_n\alpha_{n+1}}[n+1,i])^\dagger (V^{\sigma_{n+2}}_{\alpha_{n+1}\alpha_{n+2}}[n+2,i])^\dagger \dots (V^{\sigma_N}_{\alpha_{N-1}}[N,i])^\dagger \, |\sigma_1\sigma_2\dots\sigma_N\rangle \,,
\end{aligned}
\tag{5.9}
$$

with the unitary tensors U and $V^\dagger$ respectively satisfying the left and right orthonormality conditions of Section 3.4.3. For ease of notation, we now define $|\tilde{v}_i\rangle = |l^n_i\rangle\,|c^n_i\rangle\,|r^n_i\rangle$, where

$$
\begin{aligned}
|l^n_i\rangle &= \sum_{\{\sigma_k\}=1}^{2^K-1} U^{\sigma_1}[1,i] U^{\sigma_2}[2,i] \dots U^{\sigma_{n-1}}[n-1,i] \, |\sigma_1\sigma_2\dots\sigma_{n-1}\rangle \,, \\
|r^n_i\rangle &= (V^{\sigma_{n+1}}[n+1,i])^\dagger (V^{\sigma_{n+2}}[n+2,i])^\dagger \dots (V^{\sigma_N}[N,i])^\dagger \, |\sigma_{n+1}\sigma_{n+2}\dots\sigma_N\rangle \,, \\
|c^n_i\rangle &= \sum_{\sigma_n=1}^{2^K-1} C^{\sigma_n}[n,i] \, |\sigma_n\rangle \,.
\end{aligned}
\tag{5.10}
$$

Lastly, note that D_i is a skew-symmetric operator in the sense that $D_i^\dagger = -D_i$. These definitions allow us to rewrite Equation (5.8) into

$$
\begin{aligned}
\Theta = \sum_{i,j=1}^{K} \Bigg\{ & -\mu \, \langle r^n_i c^n_i l^n_i| \, D_i D_j \, |l^n_j c^n_j r^n_j\rangle \\
& + \frac{\langle r^n_i c^n_i l^n_i | l^n_i c^n_i r^n_i\rangle}{\tau^2} + \frac{\langle r^n_i c^n_i l^n_i|}{\tau} \left(\frac{-\,|\alpha_i\rangle}{\tau} + |\beta_j\rangle \odot D_j \, |\beta_i\rangle - \nu D_j^2 \, |\beta_i\rangle \right) \\
& + \left(\frac{-\,|\alpha_i\rangle}{\tau} + |\beta_j\rangle \odot D_j \, |\beta_i\rangle - \nu D_j^2 \, |\beta_i\rangle \right)^\dagger \frac{|l^n_i c^n_i r^n_i\rangle}{\tau} \Bigg\} + \Bigg[\dots \Bigg].
\end{aligned}
\tag{5.11}
$$

Now, the minimum of Θ is found at the stationary point where the gradient of Θ with regards to its variational variables vanishes. Here the variational variables in question are the tensors constituting $\sum_{i=1}^K \mathbf{e}_i \, |\tilde{v}_i\rangle$, and so we require that

$$
\frac{\partial \Theta}{\partial C^{\sigma_n}[n,i]} = 0 \tag{5.12}
$$

simultaneously for all n and i. The solution to Equation (5.12) is given by

$$
\begin{aligned}
\langle r^n_i \sigma_n l^n_i| \sum_{j=1}^{K} \Bigg\{ & \mu\tau D_i D_j \, |l^n_j c^n_j r^n_j\rangle - \frac{|l^n_i c^n_i r^n_i\rangle}{\tau} \\
& + \frac{|\alpha_i\rangle}{\tau} - |\beta_j\rangle \odot D_j \, |\beta_i\rangle + \nu D_j^2 \, |\beta_i\rangle \Bigg\} = 0,
\end{aligned}
\tag{5.13}
$$

which can be rearranged into

$$\begin{aligned} &C^{\sigma_n}[n,i] - \mu\tau^2 \sum_{j=1}^{K} \langle r_i^n \sigma_n l_i^n |\, D_i D_j \,| l_j^n \sigma_n r_j^n \rangle\, C^{\sigma_n}[n,i] \\ &= \langle r_i^n \sigma_n l_i^n | \alpha_i \rangle - \tau \langle r_i^n \sigma_n l_i^n | \sum_{j=1}^{K} \Big\{ |\beta_j\rangle \odot D_j \,|\beta_i\rangle + \nu D_j^2 \,|\beta_i\rangle \Big\}, \end{aligned} \tag{5.14}$$

after making use of the fact that $\langle r_i^n l_i^n | l_i^n r_i^n \rangle = \mathbb{1}$.

Solving Equation (5.14) will locally minimise Θ with respect to the nth tensor of the ith flow component. We define

$$\begin{aligned} H_{ij}^{\sigma_n} &= \langle r_i^n \sigma_n l_i^n |\, D_i D_j \,| l_j^n \sigma_n r_j^n \rangle\,, \\ \alpha_i^{\sigma_n} &= C^{\sigma_n}[n,i], \\ \beta_i^{\sigma_n} &= \langle r_i^n \sigma_n l_i^n | \alpha_i \rangle - \tau \langle r_i^n \sigma_n l_i^n | \sum_{j=1}^{K} \Big\{ |\beta_j\rangle \odot D_j \,|\beta_i\rangle + \nu D_j^2 \,|\beta_i\rangle \Big\}, \end{aligned} \tag{5.15}$$

and collect these into vectors $\boldsymbol{\alpha} = \mathrm{vec}(\alpha_i^{\sigma_n}), \boldsymbol{\beta} = \mathrm{vec}(\beta_i^{\sigma_n})$ while constructing the matrix H from elements $H_{ij}^{\sigma_n}$. Inserting $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and H into Equation (5.14) gives a set of linear equations for $\boldsymbol{\alpha}$

$$\left(\mathbb{1} - \mu\Delta t^2 H\right) \boldsymbol{\alpha} = \boldsymbol{\beta}. \tag{5.16}$$

$-H$ is here a positive semi-definite matrix, as can be seen by considering

$$-\boldsymbol{\alpha}^t H \boldsymbol{\alpha} = -\sum_{i,j=1}^{K} \langle \tilde{v}_i |\, D_i D_j \,| \tilde{v}_j \rangle = \sum_{i,j=1}^{K} \langle D_i \tilde{v}_i | D_j \tilde{v}_j \rangle\,. \tag{5.17}$$

Because $\sum_{i,j=1}^{K} \langle D_i \tilde{v}_i | D_j \tilde{v}_j \rangle \approx \|\nabla . \boldsymbol{V}\|_2^2 \geq 0$, H must be negative semi-definite, making $(\mathbb{1} - \mu\Delta t^2 H)$ as a whole positive definite (recall that $\Delta t, \mu > 0$). Furthermore, it can be shown that constructing H costs $\mathcal{O}\left(N\chi^4\right)$, while computing its action on $\boldsymbol{\alpha}$ costs just $\mathcal{O}\left(N\chi^3\right)$. We therefore employed the iterative method of conjugate gradient descent (CGD) [101] to solve the linear problem of Equation (5.16).

The global minimum of Θ can be obtained by adapting well-established techniques for calculating the ground state energy of quantum many-body systems. We start with $\sum_{i=1}^{K} \mathbf{e}_i \,|\tilde{v}\rangle$ in right-canonical form and minimise Θ with respect to the $n = 1$ tensor using Equation (5.16). Similarly to the density matrix renormalisation group (DMRG) algorithm [21], we subsequently perform a QR decomposition on site $n = 1$ and shift the canonical centre to site $n = 2$, and optimise Θ with respect to $n = 2$. We iterate this procedure and sweep through the sites of $\sum_{i=1}^{K} \mathbf{e}_i \,|\tilde{v}\rangle$ up until $n = N$,

and then sweep back to $n = 1$. After a certain number of such sweeps, convergence will be achieved and the minimum of Θ found.

The above minimisation procedure will complete a single Runge-Kutta 2 timestep per Equation (5.6) and time-evolve the flow field from $t = t_q$ to $t = t_{q+1}$. After $T_f/\Delta t$ such timesteps, the algorithm will have solved the incompressible Navier-Stokes equations within the MPS manifold of $\mathcal{M}(\chi)$ to second order accuracy in time.

5.4 Computational complexity

The computational complexity of our algorithm for each individual timestep is a product of the computational cost of the global sweeps multiplied with that of the local minimisations. Let $\overline{m}_1$ be the average number of sweeps required for the minimisation of Θ to converge, and q be the cost associated with shifting the canonical centre as described at the end of Section 5.3, and c the computational cost of a local minimisation. Then the scaling will be $\mathcal{O}\left[\overline{m}_1 N(q + c)\right]$ because the cost of the sweeps scales linearly with the number of matrices swept, which is N. Shifting the canonical centre rightwards from matrix n to $n + 1$ (or leftwards, to $n - 1$) is done in two parts. First, a QR decomposition is performed, followed by a tensor contraction between the upper triangular matrix and the next site. It is straightforward to show [21] that the cost of both goes as $\sim \chi^3$, which in turn implies $q \sim \chi^3$.

The computational cost of the local minimisation equals the cost needed to first explicitly calculate $\boldsymbol{\beta}$ in Equation (5.15) plus the subsequent cost associated with the CGD iterations needed to solve Equation (5.16). Regarding the latter, let $\overline{m}_2$ be the mean number of iterations CGD requires to converge to the solution of Equation (5.16) within the desired precision. The cost of these iterations is then on average a product of $\overline{m}_2$ and the cost of computing the action of H on $\boldsymbol{\alpha}$. The last operation can be executed as a tensor contraction where the matrix H is never explicitly formulated, but, instead, the tensor network of $H\boldsymbol{\alpha}$ is contracted using standard tensor contraction techniques [17] at $\mathcal{O}(\chi^3)$ cost. Calculating $\boldsymbol{\beta}$ is however more expensive due to each β_i containing the nonlinear advective term $\sum_{j=1}^{K} |\beta_j\rangle \odot D_j |\beta_i\rangle$. Straightforwardly constructing it requires the use of tensor contractions that scale as $\sim \chi^4$, as noted in Section 4.6. Thus the full cost of each local minimisation step goes as $c \sim \chi^4 + \overline{m}_2\chi^3$.

The above implies the computational complexity of each timestep to be

$$\mathcal{O}\left[\overline{m}_1 N(q+c)\right] = \mathcal{O}\left[\overline{m}_1 N\left(\chi^4 + \chi^3(\text{const} + \overline{m}_2\right)\right]. \tag{5.18}$$

The numerical precision is controlled by $\overline{m}_1$ and $\overline{m}_2$. However, in our experience, the number of sweeps or CGD iterations required to achieve a given precision does not change with increasing N or Reynolds number. Dropping these prefactors along with the non-dominant terms leads to the total computational complexity of

$$\mathcal{O}\left[\overline{m}_1 N\left(\chi^4 + \chi^3(\text{const} + \overline{m}_2\right)\right] = \mathcal{O}\left(N\chi^4\right) \tag{5.19}$$

per timestep. This complexity is equivalent to $\mathcal{O}(\chi^4 \log M)$, since the number of gridpoints M is related with N through $M = 2^{KN}$. This is compared with the $\mathcal{O}(M \log M)$ computational complexity of DNS [see Appendix A for more details].

This quartic scaling in χ is a central bottleneck of our algorithm; though it might be possible to resolve it. As already mentioned, we compute the elementwise product in the advective term directly at a cost of $\sim \chi^4$. However, this elementwise product can also be *estimated* at a cost of $\sim \chi^3$ using the sampling algorithm of [102, Step 2, Section 4], which will bring down the cost of calculating $\sum_{j=1}^{K} |\beta_j\rangle \odot D_j |\beta_i\rangle$, and by extension also the cost of whole variational algorithm, down to just $\sim \chi^3$. It is however unclear how accurately the sampling algorithm would estimate $\sum_{j=1}^{K} |\beta_j\rangle \odot D_j |\beta_i\rangle$, and the sampling algorithm is also difficult to implement. For these reasons, we leave it to future work to explore whether this sampling algorithm can be used to speed up our variational scheme.

5.5 Demonstration of computational scaling

Among the most interesting aspects of the MPS algorithm is how the computational complexity of our variational algorithm scales logarithmically with M. To practically demonstrate this scaling, Figure 5.1 plots the actual CPU time our algorithm requires to perform 10 timesteps simulating the 2-D Navier-Stokes equation at various values of M and χ. The simulations were all carried out on a `MacBook Pro (Retina, 15-inch, Mid 2015)` running `macOS v. 11.6` on a `64-bit 2,5 GHz Quad-Core i7 CPU` with `16 GB 1600 MHz DDR3 RAM`, using `C` code compiled with a `GCC/6.5.0` compiler on `O3` optimisation, with the underlying linear algebra operations being performed by `IMKL/2017` routines. Note how all the curves in the figure saturate for sufficiently

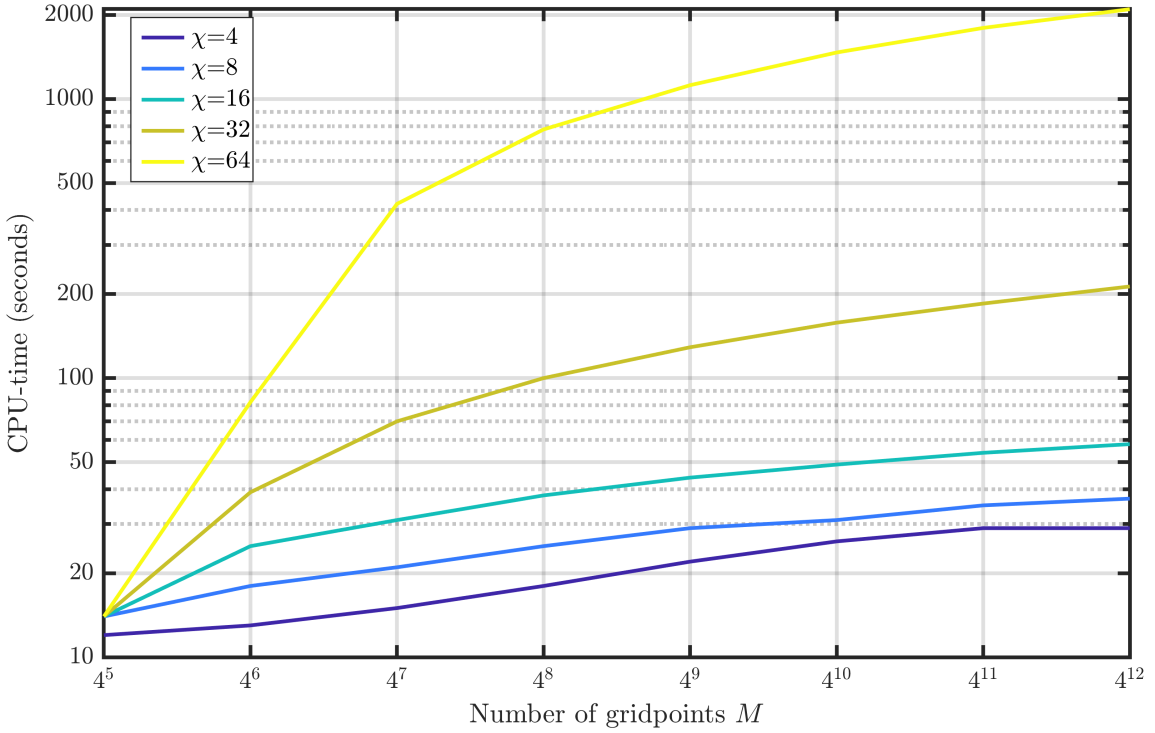

Figure 5.1: **Demonstration of MPS computational scaling**. 10 timesteps of the 2-D Navier-Stokes equations were simulated with our MPS algorithm. The CPU time (in seconds) required to perform these 10 timesteps is plotted for five values of χ against eight different grid-sizes.

many gridpoints; this implies a *sub-polynomial* scaling in M of the computational complexity that is consistent with Equation (5.19).

5.6 Arithmetic intensity of the algorithm

For completeness we also briefly discuss how memory-efficient our algorithm is. This can be done by studying a quantity known as the arithmetic intensity I. I is the ratio between the amount of arithmetic operations performed (in units of FLOPs) to the required memory traffic (in bytes) of an algorithm. Many algorithms running on modern computing systems are memory bound in the sense that their performance is limited by memory bandwidth rather than computing power. Thus, algorithms with a high value of I can more optimally utilize modern hardware than those with a low arithmetic intensity. The main sub-routines of our algorithm are QR decomposition, matrix-matrix multiplication, matrix-vector multiplication and matrix-reshapes. Let

us denote their respective arithmetic intensities as I_{QR}, $I_{\text{m-m}}$, $I_{\text{m-v}}$ and $I_{\text{m-r}}$. When operating on matrices of size $m \times m$, or vectors of size m, the intensities of these schemes go as:

$$\begin{aligned} I_{\mathrm{QR}} &\sim m, \\ I_{\text{m-m}} &\sim m, \\ I_{\text{m-v}} &\sim \text{const}, \\ I_{\text{m-r}} &\sim \text{const}. \end{aligned} \tag{5.20}$$

As discussed in Section 5.4, our algorithmic cost is dominated by the calculation of the nonlinear term. This in practice boils down to repeatedly performing matrix-matrix multiplications between pairs of non-square matrices of type $(\text{const} \cdot \chi^2) \times \chi$ and $\chi \times (\text{const} \cdot \chi)$. Theoretically, the arithmetic intensity of such operations goes as

$$I \sim \frac{\mathcal{O}(\chi^4)}{\mathcal{O}\left(\chi^2(\chi + \text{const})\right)} = \mathcal{O}(\chi) \tag{5.21}$$

for large χ when memory is efficiently utilised. If, however, the memory is *not* efficiently utilised due to shortcomings in either hardware (e.g. inadequate cache size) or software (e.g. failure to use cache blocking), the scaling in Equation (5.21) will fail to hold.

Fortunately, the problem of matrix-matrix multiplication is among the most important and thoroughly studied problems of numerical linear algebra. Highly optimised hardware (along with the associated software) beyond standard central processing units exist for such operations, ranging from graphics processing units [103] to even tensor processing units [104, 105]. Thus, the dominant operation of our algorithm is characterised by a high I. This in turn makes our algorithm capable of efficiently utilising the power of modern parallelised computing hardware.

5.7 Burgers' equation

The Burgers' equation (explained in detail in the following Section 6.1) can be solved in place of the incompressible Navier-Stokes equations by trivially modifying the algorithm. Setting $\mu = 0$ will remove the incompressibility condition from consideration such that the algorithm just solves the momentum equations without the pressure term (whose role is anyway just to enforce incompressibility), which is nothing more than Burgers' equation.

We note that setting $\mu = 0$ increases the performance of our algorithm. When $\mu = 0$, we get that $\overline{m}_2 = 1$ because the CGD procedure used to solve Equation (5.16) is replaced by the much cheaper substitution $\boldsymbol{\alpha} = \boldsymbol{\beta}$. This is an unsurprising result, given that the solutions of the Navier-Stokes equation are complicated and turbulent, whilst those of the Burgers' equation are simple and non-turbulent. However, the advective term must still be computed, hence the computational scaling of the algorithm remains $\sim \chi^4 \log M$.

5.8 Conclusion

In this chapter an MPS algorithm for simulating fluid flows has been derived. The derivation is based upon using the MPS encoding of Chapter 4 to solve the incompressible Navier-Stokes equations (or Burgers' equation if $\mu = 0$) on the MPS manifold with a variational scheme. The cost of the scheme scales as $\sim \chi^4 \log M$, albeit it might be viable to bring the scaling down to $\sim \chi^3 \log M$ (as discussed in Section 5.4), which we leave to future work to investigate.

In Chapter 2 it is asserted that the scale-local structure of turbulence allows for it to be accurately simulated within the MPS manifold whilst using far fewer variables than DNS. This assertion can be tested in practice with the MPS algorithm formulated here, and this is in fact precisely what is done in Chapter 7. But before looking at actual turbulent flow, we investigate the efficacy of our algorithm in the following Chapter 6 by setting $\mu = 0$ and simulating the much simpler fluid dynamics of Burgers' equation.

Chapter 6

Simulating shock waves

6.1 Introduction: Burgers' equation

A central difficulty in the study of fluids is that their dynamics are simultaneously nonlinear and (spatially) non-local. Take the case of a Newtonian, incompressible fluid, whose governing equations are the incompressible Navier-Stokes equations of Equation (2.8). The advective term $\big(\mathbf{V}.\nabla\big)\mathbf{V}$ in these equations induces nonlinearity. As for the non-locality, consider that the speed of sound in a fluid is given by the derivative of the pressure field with regards to the fluid density. Because the density is constant in an incompressible fluid, the speed of sound becomes infinite and thus information is transferred non-locally in space. This non-locality is enforced through the pressure, which serves as a Lagrange-multiplier for the incompressibility condition [43].

In Jan Burgers' 1948 study of turbulence [106], he simplified the incompressible Navier-Stokes equations by removing the pressure term and thus eliminating the non-locality. This resulted in what is now known as Burgers' equation:

$$\frac{\partial \mathbf{V}}{\partial t} + \big(\mathbf{V}.\nabla\big)\mathbf{V} = \nu\nabla^2\mathbf{V}. \tag{6.1}$$

Equation (6.1) is typically studied in 1D (one spatial coordinate x and time), giving

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{6.2}$$

with $u(x,t)$ being the sole velocity component of flow. Burgers' idea was that removing the non-locality will make solving the equations feasible while hopefully still preserving

at least some of the features of turbulence due to the remaining nonlinearity. However, this hypothesis unfortunately turned out to be false. Some years later Hopf [107] and Cole [108] demonstrated that Burgers' equation can be mapped into a standard linear heat equation (which has a known analytical solution) that embodies none of the core features of turbulence (i.e., chaos, rapid mixing and multiscale physics). However, Burgers' equation remains important in turbulence research precisely because it is analytically solvable while still containing the nonlinear advective term. These features make it an excellent testbed for new numerical methods, since the results of numerical simulations of Burgers' equation can straightforwardly and directly be compared against analytical solutions.

6.1.1 Chapter structure

We use that Burgers' equation is analytically solveable to evaluate the utility of using tensor networks for simulating fluid flows. First we present the known mathematical solution of the 1D Burgers' equation in Section 6.2 for arbitrary initial conditions. We then narrow down by using the δ function as the initial flow field, which generate so-called "hump solutions" to Burgers' equation. They are derived analytically in Section 6.3. Then, in Section 6.4 it is shown that tensor networks are able to represent all hump solutions with exponential efficiency. In particular, in the asymptotic limit of $\nu \to 0$, these hump solutions become triangular waves, i.e. prototypical shock waves. An exact MPS description of these triangular waves is provided in Section 6.5, where it turns out the bond-dimension is independent of the grid size. This result implies that shock waves are simulateable within a highly (exponentially) compressed MPS manifold, and this is confirmed through actual such simulations in Section 6.6. The chapter concludes in Section 6.7.

6.2 General analytical solution of Burgers' equation

We now present the analytical solution to the 1D Burgers' equation. By defining a function $w = w(x,t)$ via

$$u(x,t) = -2\nu \frac{1}{w}\frac{\partial w}{\partial x} = -2\nu \frac{\partial \log(w)}{\partial x}, \tag{6.3}$$

also known as the Hopf-Cole transformation [107, 108], Equation (6.2) simplifies to the heat equation

$$\frac{\partial w}{\partial t} = \nu \frac{\partial^2 w}{\partial x^2}. \tag{6.4}$$

The heat equation has the general mathematical solution

$$w(x,t) = \frac{1}{2\sqrt{\pi \nu t}} \int_{-\infty}^{\infty} w(\alpha, 0) e^{-\frac{(x-\alpha)^2}{4\nu t}} \mathrm{d}\alpha, \tag{6.5}$$

where $w(\alpha, 0)$ denotes the initial condition at time $t = 0$. $w(\alpha, 0)$ is obtained from the initial condition to Burgers' equation $u(x, 0)$ by inverting Equation (6.3) for $t = 0$:

$$w(x,0) = e^{-\frac{1}{2\nu} \int_a^x u(y,0) \mathrm{d}y}, \tag{6.6}$$

where a can be chosen freely. Combining the above equations results in the following solution to Burgers' equation:

$$u(x,t) = \frac{1}{t} \frac{\int_{-\infty}^{\infty} w(\alpha,0)(x-\alpha) e^{-\frac{(x-\alpha)^2}{4\nu t}} \mathrm{d}\alpha}{\int_{-\infty}^{\infty} w(\alpha,0) e^{-\frac{(x-\alpha)^2}{4\nu t}} \mathrm{d}\alpha}, \tag{6.7}$$

where $w(\alpha, 0)$ represents the initial condition defined in Equation (6.6).

6.3 Analytical solution for initial δ function

Following [109], we now investigate the so-called hump solution to Burgers' equation. It is produced when using the δ function initial condition of

$$u(x,0) = u_0 \delta(x - x_0), \tag{6.8}$$

where u_0 is a normalising constant. Plugging this into Equation (6.6) gives

$$w(x,0) = e^{-\frac{u_0}{2\nu} \int_a^x \delta(y - x_0) \mathrm{d}y}. \tag{6.9}$$

Now we choose $a = x_0 + \epsilon$, where ϵ denotes an infinitesimally small positive number, so that

$$\begin{aligned} w(x,0) &= e^{\frac{u_0}{2\nu}}, \quad x \leq x_0, \\ w(x,0) &= 1, \quad x > x_0. \end{aligned} \tag{6.10}$$

The integration is now split into two parts as:

$$\int_{-\infty}^{\infty} w(\alpha,0) \ldots \mathrm{d}\alpha = e^{\frac{u_0}{2\nu}} \int_{-\infty}^{x_0} \ldots \mathrm{d}\alpha + \int_{x_0+\epsilon}^{\infty} \ldots \mathrm{d}\alpha. \tag{6.11}$$

We from now on ignore ϵ to simplify the notation. The evaluation of the integral in the denominator of Equation (6.7) is straightforward, and the integral in the numerator can be evaluated by making use of the substitution $\beta = (x-\alpha)/(2\sqrt{\nu t})$. Ultimately, these calculations lead to the result

$$u(x,t) = \sqrt{\frac{\nu}{t}} \frac{(e^{\frac{u_0}{2\nu}} - 1)e^{-\frac{(x-x_0)^2}{4\nu t}}}{\sqrt{\pi} + \frac{\sqrt{\pi}}{2}(e^{\frac{u_0}{2\nu}} - 1)\mathrm{erfc}(\frac{x-x_0}{2\sqrt{\nu t}})}, \tag{6.12}$$

where $\mathrm{erfc}(x) = (2/\sqrt{\pi}) \int_x^\infty \exp(-\alpha^2)\mathrm{d}\alpha$ is the complementary error function.

We now analyse the solution (6.12) in the limit of $\nu \to 0$. First consider $x \leq x_0$, for which

$$\begin{aligned} \lim_{\nu\to 0} \mathrm{erfc}\left(\frac{x-x_0}{2\sqrt{\nu t}}\right) &= 2, \quad x < x_0, \\ \lim_{\nu\to 0} \mathrm{erfc}\left(\frac{x-x_0}{2\sqrt{\nu t}}\right) &= 1, \quad x = x_0, \end{aligned} \tag{6.13}$$

making Equation (6.12) become

$$\lim_{\nu\to 0} u(x,t) = 0, \quad x \leq x_0. \tag{6.14}$$

For $x > x_0$ we use the $x \to \infty$ asymptotic expansion of the complementary error function

$$\mathrm{erfc}(x) = \frac{e^{-x^2}}{\sqrt{\pi}x} + \mathcal{O}(x^{-3}e^{-x^2}), \tag{6.15}$$

so that for $\nu \to 0$

$$\mathrm{erfc}\left(\frac{x-x_0}{2\sqrt{\nu t}}\right) \approx 2\sqrt{\nu t}\frac{e^{-\frac{(x-x_0)^2}{4\nu t}}}{\sqrt{\pi}(x-x_0)}, \quad x > x_0, \tag{6.16}$$

which transforms Equation (6.12) into

$$u(x,t) = \sqrt{\frac{\nu}{t}} \frac{(e^{\frac{u_0}{2\nu}} - 1)e^{-\frac{(x-x_0)^2}{4\nu t}}}{\sqrt{\pi} + \sqrt{\nu t}(e^{\frac{u_0}{2\nu}} - 1)\frac{e^{-\frac{(x-x_0)^2}{4\nu t}}}{x-x_0}}. \tag{6.17}$$

It is observed that

$$\begin{aligned} \lim_{\nu\to 0} e^{-\frac{(x-x_0)^2}{4\nu t}} &= 0, \quad x > x_0, \\ \lim_{\nu\to 0} \left(e^{\frac{u_0}{2\nu} - \frac{(x-x_0)^2}{4\nu t}}\right) &= \infty, \quad x_0 < x < x_0 + \sqrt{2u_0 t}, \\ \lim_{\nu\to 0} \left(e^{\frac{u_0}{2\nu} - \frac{(x-x_0)^2}{4\nu t}}\right) &= 1, \quad x = x_0 + \sqrt{2u_0 t}, \\ \lim_{\nu\to 0} \left(e^{\frac{u_0}{2\nu} - \frac{(x-x_0)^2}{4\nu t}}\right) &= 0, \quad x > x_0 + \sqrt{2u_0 t}. \end{aligned} \tag{6.18}$$

Using these limit solutions in Equation (6.17) gives that in the limit $\nu \to 0$:

$$\begin{aligned} u(x,t) &= \frac{x - x_0}{t}, \quad x_0 < x < x_0 + \sqrt{2u_0 t}, \\ u(x,t) &= 0, \quad \text{otherwise.} \end{aligned} \tag{6.19}$$

As ν grows large, it is straightforward to show that Equation (6.12) approaches the Gaussian

$$u(x,t) = \frac{u_0}{2\sqrt{\pi\nu t}} e^{-\frac{(x-x_0)^2}{4\nu t}}. \tag{6.20}$$

And for $\nu \to \infty$, this Gaussian will approach a uniform function with an amplitude tending towards zero.

6.4 Solutions in matrix product state form

The Burgers' equation can be solved with exponential efficiency in the number of grid points through the MPS framework. Recall the MPS representation of Section 4.2 and consider it for a 1D function discretised with $M = 2^N$ gridpoints. In this case, the physical index in Equation (4.3) collapses to the range $\omega_n \equiv b_n^1 \in \{0, 1\}$, which means that all grid points can be defined as $\boldsymbol{r}_q = x_q = q/2^N$, where $q = 0, 1, \ldots, 2^N - 1$. In the following text we show that both the aforementioned δ hump initial flow condition and the later flow it develops into can both be efficiently represented in MPS form.

Let us begin with the MPS representation of the δ hump. The initial discretised δ-function $u_0(x_q) = u_0\delta(q - j)$ with normalising constant u_0 and peak position $j = (b_1, \ldots, b_N)_2$, $b_n \in \{0, 1\}$ can be exactly represented as a MPS per Equation (4.8). The bond-dimension of this MPS is just $\chi = 1$ and its tensors $A[n]$ are given by

$$A_{b_1}^{\omega_1}[1] = \begin{cases} u_0, & \omega_1 = b_1, \\ 0, & \text{otherwise,} \end{cases} \qquad \text{and} \qquad A_{b_n}^{\omega_n}[n] = \begin{cases} 1, & \omega_n = b_n, \\ 0, & \text{otherwise,} \end{cases} \tag{6.21}$$

for $n = 2, \ldots, N$.

Time evolving this initial δ function according to Burgers' equation results in the solution of Equation (6.12). For very large ν, this solution becomes a Gaussian (tending towards a uniform function as $\nu \to \infty$), which has an exponentially convergent MPS approximation [110]. For $0 < \nu < \infty$ it can be shown that the function of Equation (6.7) remains holomorphic in x, and hence regularity arguments similar to those in [111] can be used to prove an exponential convergence of the polynomial approximation

of Equation (6.7). In turn, polynomials of degree p sampled on an equidistant grid admit an exact MPS representation [88, Thm. 6] with $\chi \leq p+1$, and hence this MPS is also an exponentially converging approximation of Equation (6.7). The solution for $\nu \to 0$ is a triangular wave which has an exact MPS representation of bond-dimension $\chi = 3$, as proven in Section 6.5.

Therefore, we conclude that the MPS description of solutions of the initial values problem considered here are exponentially convergent in the number of variables used. In other words, a MPS scheme would require exponentially fewer variables than e.g. a standard finite difference scheme, as illustrated in the following section for the case of $\nu \to 0$.

6.5 Triangular waves as matrix product states

We now demonstrate the exponential advantage that the MPS ansatz can provide over DNS for the $\nu \to 0$ triangular-wave solution of Equation (6.12) by analytically deriving its MPS representation. This limit solution is a prototypical shock wave and (as all shock waves) it is discontinuous, which slows down the convergence of both polynomial and Fourier approximations. In contrast, a MPS with a bond-dimension of just 3 can represent this function exactly, as we shall now see.

Definition. A Heaviside vector of length $J \in \mathbb{N}$ with step position $j \in \mathbb{Z}$ is defined element-wise as

$$\theta_q^j := \begin{cases} 1, & q \leq j, \\ 0, & \text{otherwise}, \end{cases} \tag{6.22}$$

for $q \in \{0, 1, \ldots, J-1\}$.

Definition. A unit vector at position j, with $j \in \{0, \ldots, J-1\}$, $J \in \mathbb{N}$, is defined element-wise as

$$e_q^j := \begin{cases} 1, & q = j, \\ 0, & \text{otherwise}, \end{cases} \tag{6.23}$$

for $q \in \{0, \ldots, J-1\}$.

Lemma. Let $j = (b_1, \ldots, b_N)_2$ and $q = (\omega_1, \ldots, \omega_N)_2$, with $b_n, \omega_n \in \{0, 1\}$. Then the j-th Heaviside vector can be written as the MPS

$$\theta_{\omega_1 \ldots \omega_N}^{b_1 \ldots b_N} = T_{\omega_1}^{b_1}[1] \cdots T_{\omega_N}^{b_N}[N] \tag{6.24}$$

with bond-dimensions $d(n) = 2$, $n = 1, \ldots, N-1$, where

$$T^{b_1}_{\omega_1}[1] = \begin{bmatrix} e^{b_1}_{\omega_1} & \theta^{b_1-1}_{\omega_1} \end{bmatrix}, \tag{6.25}$$

$$T^{b_n}_{\omega_n}[n] = \begin{bmatrix} e^{b_n}_{\omega_n} & \theta^{b_n-1}_{\omega_n} \\ 0 & 1 \end{bmatrix}, \tag{6.26}$$

for $n = 2, \ldots, N-1$, and

$$T^{b_N}_{\omega_N}[N] = \begin{bmatrix} \theta^{b_N}_{\omega_N} \\ 1 \end{bmatrix}. \tag{6.27}$$

Proof. Consider two indices first, and prove that $\theta^{b_1 b_2}_{\omega_1\omega_2} = \theta^{b_1-1}_{\omega_1} + \theta^{b_2}_{\omega_2} e^{b_1}_{\omega_1}$.

- If $\omega_1 > b_1$, we obtain 0, as expected from (6.22).
- If $\omega_1 = b_1$, we get $\theta^{b_2}_{\omega_2}$. This becomes 0 when $\omega_2 > b_2$, and 1 when $\omega_2 \leq b_2$.
- If $\omega_1 < b_1$, we are left with $\theta^{b_1-1}_{\omega_1} = 1$.

All cases are thus in agreement with (6.24) for $N = 2$. Multiplying the last two factors of the Heaviside MPS gives

$$T^{b_{N-1}}_{\omega_{N-1}}[N-1] T^{b_N}_{\omega_N}[N] = \begin{bmatrix} \theta^{b_{N-1}-1}_{\omega_{N-1}} + \theta^{b_N}_{\omega_N} e^{b_{N-1}}_{\omega_{N-1}} \\ 1 \end{bmatrix} = \begin{bmatrix} \theta^{b_{N-1}, b_N}_{\omega_{N-1}, \omega_N} \\ 1 \end{bmatrix}. \tag{6.28}$$

Similarly, assuming that

$$T^{b_n}_{\omega_n}[n] \cdots T^{b_N}_{\omega_N}[N] = \begin{bmatrix} \theta^{b_n \ldots b_N}_{\omega_n \ldots \omega_N} \\ 1 \end{bmatrix}, \tag{6.29}$$

gives the induction step for $T[n-1] \cdots T[N]$, and eventually, since $T^{b_1}_{\omega_1}[1]$ is just one row,

$$T^{b_1}_{\omega_1}[1] \cdots T^{b_N}_{\omega_N}[N] = \theta^{b_1 \ldots b_N}_{\omega_1 \ldots \omega_N}, \tag{6.30}$$

as expected.

Definition. A vector whose elements are $x_q = q = (\omega_1, \ldots, \omega_n)_2$ can be expressed by the MPS

$$X_q \equiv X_{\omega_1 \ldots \omega_N} = \begin{bmatrix} 1 & 2^{N-1}\omega_1 \end{bmatrix} \cdots \begin{bmatrix} 1 & 2^{N-n}\omega_n \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} \omega_N \\ 1 \end{bmatrix} \tag{6.31}$$

of bond-dimension $\chi = 2$, per [88].

Theorem. A triangular wave vector with elements

$$w^{b_1\ldots b_N}_{\omega_1\ldots\omega_N} := X_q \cdot \theta^{b_1\ldots b_N}_{\omega_1\ldots\omega_N} \tag{6.32}$$

can be written as a MPS of bond-dimension $\chi = 3$, $n = 1, \ldots, N-1$.

Proof. Multiplying the MPS $X_{q_1\ldots q_N}$ with $\theta^{b_1\ldots b_N}_{q_1\ldots q_N}$ tensor by tensor will result in a MPS $\widehat{W}[1]\cdots\widehat{W}[N]$ of bond-dimension 4. However, this decomposition is redundant. For example, the first factor reads

$$\widehat{W}^{b_1}_{\omega_1}[1] = \begin{bmatrix} e^{b_1}_{\omega_1} & e^{b_1}_{\omega_1} 2^{N-1}\omega_1 & \theta^{b_1-1}_{\omega_1} & \theta^{b_1-1}_{\omega_1} 2^{N-1}\omega_1 \end{bmatrix}, \tag{6.33}$$

albeit $e^{b_1}_{\omega_1}\omega_1 = b_1 e^{b_1}_{\omega_1}$ which means that

$$\widehat{W}^{b_1}_{\omega_1}[1] = \underbrace{\begin{bmatrix} e^{b_1}_{\omega_1} & \theta^{b_1-1}_{\omega_1} & \theta^{b_1-1}_{\omega_1} 2^{N-1}\omega_1 \end{bmatrix}}_{W^{b_1}_{\omega_1}[1]} \underbrace{\begin{bmatrix} 1 & b_1 2^{N-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R^{b_1}[1]}. \tag{6.34}$$

$W[1]$ (of bond-dimension 3) can be considered the first non-redundant MPS tensor of $w^{b_1\ldots b_N}_{\omega_1\ldots\omega_N}$. Multiplying $R[1]$ with $\widehat{W}[2]$ will continue the reduction and produce $W[2]$. Assuming that $W[1], \ldots, W[n-1]$ have already been obtained, the next step gives

$$R^{b_{n-1}}[n-1]\widehat{W}^{b_n}_{\omega_n}[n] :=$$

$$\begin{bmatrix} 1 & c_{n-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e^{b_n}_{\omega_n} & 2^{N-n}\omega_n e^{b_n}_{\omega_n} & \theta^{b_n-1}_{\omega_n} & 2^{N-n}\omega_n\theta^{b_n-1}_{\omega_n} \\ 0 & e^{b_n}_{\omega_n} & 0 & \theta^{b_n-1}_{\omega_n} \\ 0 & 0 & 1 & 2^{N-n}\omega_n \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.35}$$

where c_{n-1} is a scalar, with $c_1 = b_1 2^{N-1}$. This gives

$$R^{b_{n-1}}[n-1]\widehat{W}^{b_n}_{\omega_n}[n] =$$

$$\begin{bmatrix} e^{b_n}_{\omega_n} & (2^{N-n}b_n + c_{n-1})e^{b_n}_{\omega_n} & \theta^{b_n-1}_{\omega_n} & (2^{N-n}\omega_n + c_{n-1})\theta^{b_n-1}_{\omega_n} \\ 0 & 0 & 1 & 2^{N-n}\omega_n \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.36}$$

Notice that the first two columns are linearly dependent, allowing us to rewrite the above expression into

$$R^{b_{n-1}}[n-1]\widehat{W}^{b_n}_{\omega_n}[n] =$$

$$\underbrace{\begin{bmatrix} e^{b_n}_{\omega_n} & \theta^{b_n-1}_{\omega_n} & (2^{N-n}\omega_n + c_{n-1})\theta^{b_n-1}_{\omega_n} \\ 0 & 1 & 2^{N-n}\omega_n \\ 0 & 0 & 1 \end{bmatrix}}_{W^{b_n}_{\omega_n}[n]} \underbrace{\begin{bmatrix} 1 & 2^{N-n}b_n + c_{n-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R^{b_n}[n]}. \tag{6.37}$$

Since $R[n]$ is identical to $R[n-1]$ except for the element $c_n = 2^{N-n}b_n + c_{n-1}$ at position $(1,2)$, the recursion can continue all the way until $\widehat{W}[N]$. This will result in the factors $W[n]$ of bond-dimensions 3 that together form the non-redundant representation of $w^{b_1 \ldots b_N}_{\omega_1 \ldots \omega_N}$.

6.6 Simulation in matrix product state manifold

The analytical results of the previous sections are now checked *numerically*. We do this by using the MPS algorithm of Section 5.7 to time-evolve the δ hump initial condition under Burgers' equation at large ν and compare the results against DNS. Furthermore, to stress that an extremely dense grid is required to correctly resolve shock waves as previously pointed out in Section 6.4, under-resolved DNS (URDNS) is also performed on overly coarse grids that fail to cover the finest length scales of the flow (see Appendix A for a more involved explanation of URDNS and DNS).

The set-up of our simulations is as follows. Burgers' equation is solved within the spatial domain of $x \in [0, 1)$ and temporal domain $t \in [0, 1]$. The initial δ hump is approximated using the Gaussian function

$$u(x,0) = \frac{u_0}{\sqrt{2\pi}\sigma} e^{-\left(\frac{x-x_0}{\sqrt{2}\sigma}\right)^2} \approx u_0 \delta(x - x_0), \tag{6.38}$$

where $u_0 = 1$, $x_0 = 1/2$ and $\sigma = 10^{-3}/\sqrt{2}$. The viscosity is set to $\nu = 1.3 \times 10^5$, for which DNS requires a spatial grid of $M = 8192$ points to resolve all relevant scales. The MPS simulations were carried out on this underlying grid at various bond-dimensions.

The flows resulting from the simulations are shown in Figure 6.1 for DNS, MPS (at various bond-dimensions) and URDNS (at different grid sizes). From this figure, it is clear that all the URDNS calculations using 2048 grid points or less are unreliable, as can be seen from the numerical oscillations around the shock at $t = 1/10$. This is especially the case for the 512 grid, where the oscillations visibly propagate through time. In comparison, the MPS simulations are consistently accurate except for the very lowest bond-dimension ($\chi = 2$). Especially the $\chi \geq 4$ simulations are indistinguishable from DNS with the naked eye.

As Burgers' equation is non-chaotic, the accuracy of each simulation can be quantified by evaluating how close the final velocity field is to the DNS solution. We do this in

two ways. First, we calculate the normalised distance in the L2-norm between the velocity field of the DNS solution at $t = 1$ to the velocity fields calculated through URDNS and MPS:

$$\rho(s,c) = \sqrt{\frac{\int_0^1 \left[u_{\mathrm{DNS}}(x,t=1) - u_{s,c}(x,t=1)\right]^2 \mathrm{d}x}{\int_0^1 u_{\mathrm{DNS}}(x,t=1)^2 \mathrm{d}x}}. \tag{6.39}$$

Since the DNS solution is nearly exact, $\rho(s,c)$ can be used to gauge the accuracy of scheme s at compression ratio c, and is tabulated in Table 6.1 for the simulations in Figure 6.1. From the table we see that URDNS at $c = 1{:}8$ is already about thrice as inaccurate as MPS at $c = 1{:}52$, which is in accord with the above discussion. Secondly, we also measured the power spectrum $E(k)$ at $t = 1$ for each simulation. $E(k)$ is defined as

$$\begin{aligned} E(k) &= \frac{1}{2}|\widehat{u}(k,1)|^2, \\ \widehat{u}(k,t) &= \int_0^1 u(x,t) e^{-\mathrm{i}2\pi k x} \mathrm{d}x, \end{aligned} \tag{6.40}$$

and is plotted in Figure 6.2.

Compression	Scheme	Inaccuracy
1:4	URDNS	0.0003
1:8	URDNS	0.0122
1:16	URDNS	0.5151
1:26	MPS	0.0006
1:52	MPS	0.0043
1:85	MPS	0.0703
1:171	MPS	0.6727

Table 6.1: **Quantitative evaluation of the accuracy of MPS and URDNS simulations for the 1D Burgers' equation shockwave**. The quantity $\rho(s,c)$ of Equation (6.39) is tabulated here, which measures how distant in the L2-norm the $u(x, t = 1)$ velocity field calculated through DNS is to that of URDNS and MPS. The nearer $\rho(s,c)$ is to 0, the closer the solution of the scheme in question is to the correct solution given by DNS.

The power spectrums in Figure 6.2 further substantiate that MPS is much more accurate than URDNS. The minimum bond-dimension required for reasonable accuracy is (unsurprisingly) $\chi = 3$, while going up to $\chi = 4$ results in a solution at DNS-level accuracy until about $k \approx 10^3$. $\chi = 4$ is equivalent to operating entirely within a MPS manifold parameterised by only 1:52 as many variables as that of the DNS solution space. In comparison, URDNS at the 512 grid (1:16 data compression ratio) fails

completely, while the accuracy of the MPS scheme only truly breaks down at $\chi = 2$ (1:171 data compression ratio).

The results in Table 6.1 and Figures 6.1 and 6.2 confirm the assertion made in Section 6.5 that $\chi = 3$ is sufficient for MPS to accurately simulate a triangular wave. In the figures we see that the shock wave is overall correctly resolved and time-evolved, albeit with some numerical artefacts occurring due to ν not being exactly set to zero. Letting $M \to \infty$, $\sigma \to 0$ and $\nu \to 0$ should make the $\chi = 3$ MPS simulations converge to a perfectly resolved triangular wave.

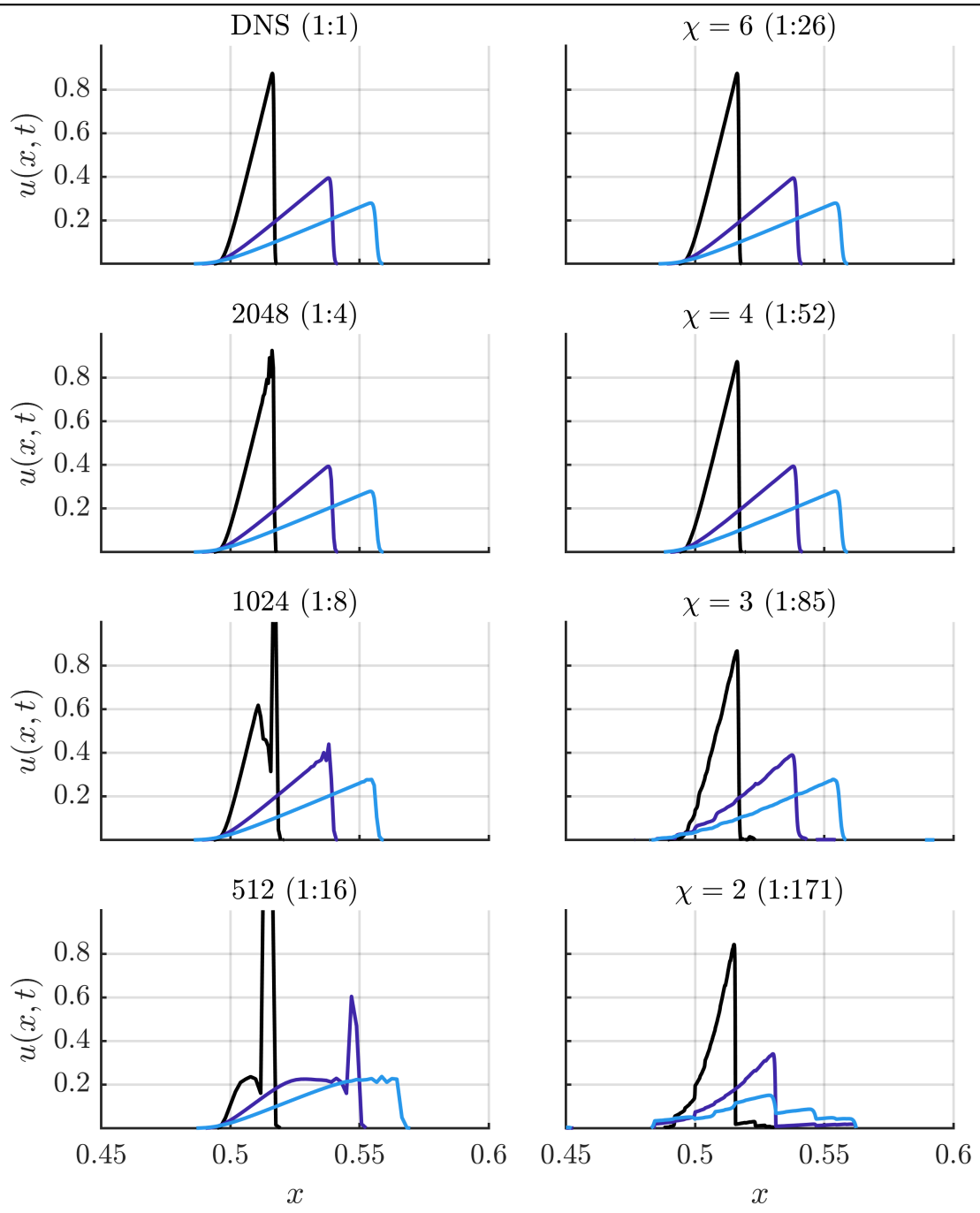

Figure 6.1: **Shock wave formation under the 1D Burgers' equation**. Here an initial δ hump flow profile results in the formation of a shock wave. The black curves correspond to the field at $t = 1/10$, blue to $t = 1/2$ and cyan to $t = 1$. In the left column, the simulations were performed using DNS (8192 spatial grid) and URDNS (at 2048, 1024 and 512 spatial grids). The simulations in the right columns are from MPS (at bond-dimensions $\chi = 6, 4, 3, 2$). The compression ratio in the number of variables compared to DNS is marked in parentheses.

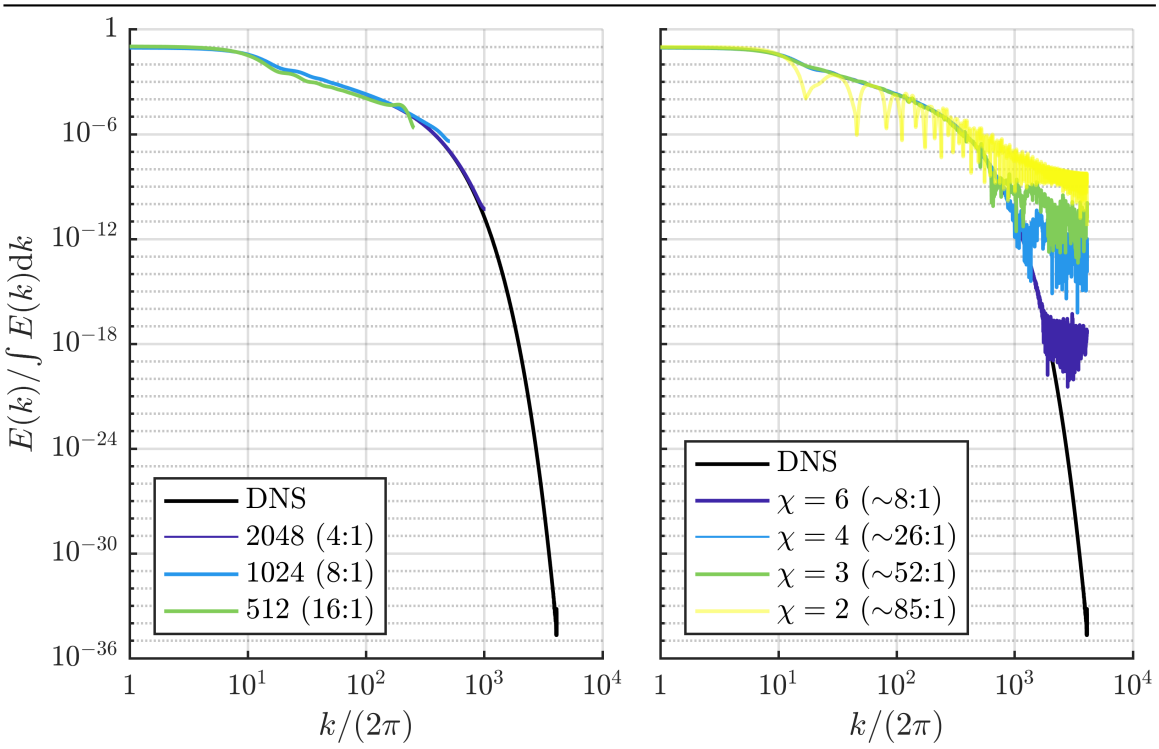

Figure 6.2: **Power spectrum of shock wave formed under Burgers' equation**. Power spectra are shown for DNS, URDNS and MPS simulations at $t = 1$ for the shock wave which results when time-evolving a δ hump initial flow field under the 1D Burgers' equation. DNS was done at a 8192 spatial grid. The power spectra have been normalised by the total kinetic energy present at each flow field. The compression ratio in the number of variables compared to DNS is marked in parentheses.

6.7 Conclusion

This chapter investigates the efficacy of the MPS ansatz by considering a 1D shock wave evolving under Burgers' equation. The equation is first solved using the well-known Hopf-Cole transformation, after which it is investigated how well the resulting solution can be encoded in MPS form. It turns out that for a delta-hump initial condition, the MPS representation should achieve *exponential* data-compression. This data-compression is then studied numerically by considering the asymptotic limit of $\nu \to 0$, where the solution converges to a triangular wave, i.e. a prototypical shock wave. While such solutions are extremely challenging to resolve using standard DNS schemes, it turns out they can be accurately represented by MPSs of bond-dimension $\chi = 3$. This makes the MPS algorithm scale *exponentially* better than DNS for this flow case, since the computational cost of DNS is here $\sim M$ [Appendix A] whilst that of the MPS is $\sim \chi^4 \log M$ [Section 5.4].

These results indicate that the structure of flow governed by Burgers' equation can be well-represented in MPS form, a promising first check on the effectiveness of MPS for CFD. The natural next step is to employ the MPS algorithm to simulate the more realistic Navier-Stokes equations.

Chapter 7

Simulating turbulence with matrix product states

7.1 Introduction

The preceding chapters have finally put us in the position to use MPSs to exploit the structures of turbulence for simulation. Chapter 2 argues that turbulence is characterised by scale-local structures which can be resolved using a specific scale-by-scale MPS encoding. This encoding is defined in detail in Chapter 4, and then used in Chapter 5 to formulate an algorithm for simulating fluid dynamics within the data-compressed MPS format. And now that the efficacy of the MPS algorithm is demonstrated in Chapter 6, we can finally use it to simulate turbulence.

We consider the turbulent TDJ and TGV flows of Chapter 2. Because these two flow cases contain scale-local structure that can be captured by the MPS decomposition (recall Figure 2.11 and Section 2.8), it should be possible to accurately simulate these flows in MPS manifolds $\mathcal{M}(\chi)$ at highly limited bond-dimensions χ. To test this assertion, we perform MPS simulations at various χ and compare the resulting accuracy against DNS. Reducing χ is equivalent to decreasing the number of variables parameterising the MPS solution, and the analog to reducing the bond-dimension in DNS is to perform URDNS on a coarse grid that does not cover all relevant length scales (as a side-note, URDNS can be considered to be the simplest form of LES – see Appendix A). For a fair comparison, we therefore choose for every χ a corresponding URDNS grid such that both methods parameterise the solution with approximately the same number of variables.

7.1.1 Chapter structure

The MPS, URDNS and DNS simulations are compared *qualitatively* for the TDJ and TGV flows in Sections 7.2 and 7.3, while in the following Section 7.4 a more *quantitative* comparison is provided. The main findings of this chapter are summarised and put into a larger context in Section 7.5.

7.2 2D turbulence: temporally developing jet

We begin by considering the accuracy of MPS relative to DNS for the TDJ flow described in Section 2.6.1. In the units defined there, we set the penalty coefficient for MPS simulations to be $\mu = 6.25 \times 10^4$. The results for the different solvers are shown in Figure 7.1. The top row in Figure 7.1a corresponds to DNS and illustrates how the background perturbations in the shear-layer are amplified ($t/T_0 = 0.25$) until the layer rolls up into vortices which in turn pair-up and merge into progressively larger vortices ($t/T_0 = 0.75, 1.25$) until $t/T_0 = 1.75$, when pairing is terminated.

The stress exerted upon the mean flow by turbulent fluctuations is given by the Reynolds stress tensor

$$\tau_{ij}(y,t) = \langle u_i' u_j' \rangle \,, \tag{7.1}$$

where $u_i' = u_i - \langle u \rangle_i$ is the fluctuating part of the velocity component and $\langle \cdot \rangle$ denotes the ensemble-average across the statistically homogeneous streamwise direction: $\langle u \rangle_i = \frac{1}{L_{\text{box}}} \int_0^{L_{\text{box}}} u_i(x,y,t) \mathrm{d}x$. The τ_{12} component is plotted in Figure 7.1b. The famous Boussinesq approximation, used in LES and RANS, models τ_{12} as

$$-\tau_{12} \sim \frac{\partial \langle u_1 \rangle}{\partial y} + \frac{\partial \langle u_2 \rangle}{\partial x}, \tag{7.2}$$

and interestingly, in Figure 7.1b the Reynolds stresses are largely of the opposite sign to the mean streamwise velocity gradients along the cross-stream direction, which is in accord with the Boussinesq approximation. The exception is when vortex pairing is terminated and the growth of the shear-layer is temporarily paused. It is important that ensemble-averaged quantities such as τ_{12} are correctly resolved if the simulation is to be physically valid.

Let us now compare the accuracy between the MPS and URDNS simulations. Rows 2–4 in Figure 7.1 show MPS results for $\chi = 33, 74$ and 118, corresponding to compression

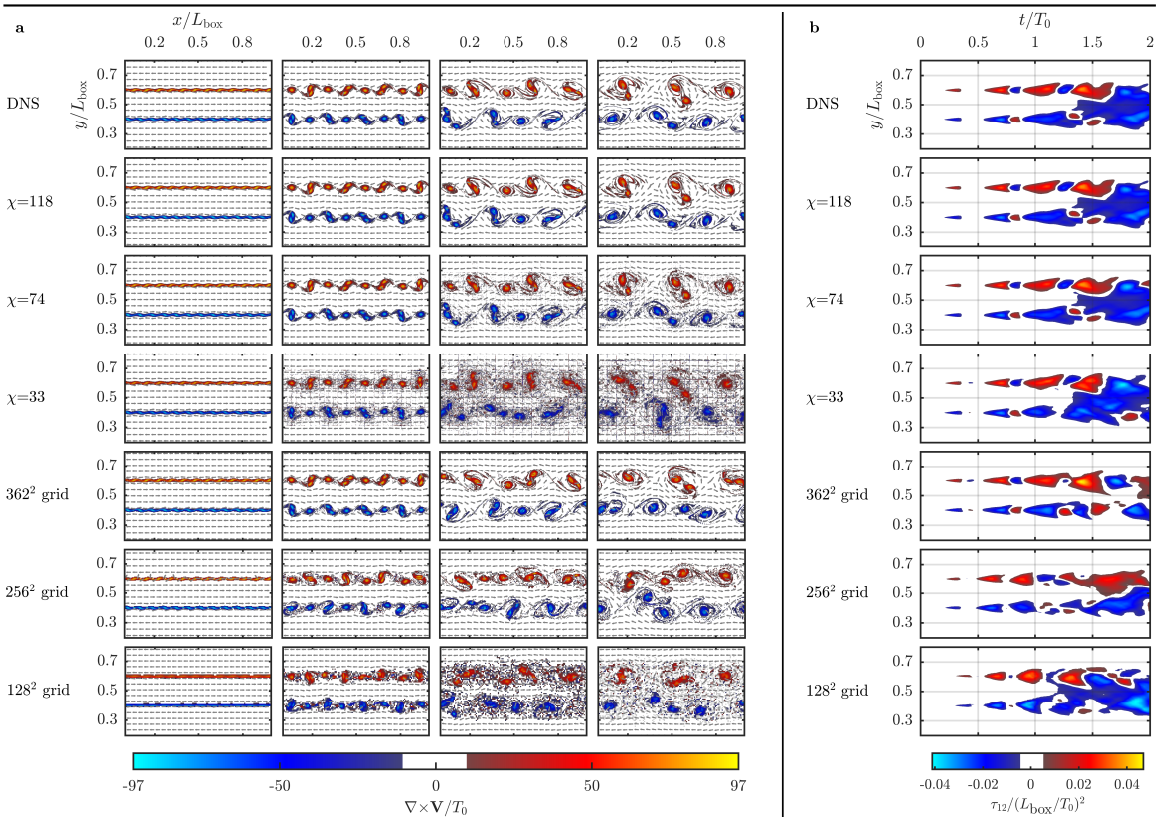

Figure 7.1: **2D Temporally developing jet.** Dynamical simulation of the incompressible Navier-Stokes equations in 2D for a planar jet streaming along x with $\text{Re} = 1000$ per the flow-conditions defined in Section 2.6.1. **a** shows snapshots of the vorticity and velocity fields taken at $t/T_0 = 0.25, 0.75, 1.25, 1.75$ (left to right). Red corresponds to positive vorticity (counter-clockwise flow) and blue to negative (clockwise). Top row corresponds to DNS results (also plotted in Figure 2.6.1) on a quadratic $2^{10} \times 2^{10}$ grid. Rows 2–4 are MPS results with different maximal bond dimensions χ. Bottom three rows are for URDNS on quadratic grids as indicated. **b** Reynolds stress τ_{12} [see Equation (7.1)] between the streamwise and cross-stream directions as a function of time and y coordinate. Red (blue) corresponds to positive (negative) stress.

ratios of approximately 1:64, 1:16 and 1:8 compared to DNS, respectively. These results show that the large-scale dynamics of the jet are correctly captured by all the MPS simulations, with $\chi = 74$ and 118 practically indistinguishable from DNS. The bottom three rows in Figure 7.1 show the URDNS results for different grid sizes. We find that the 362^2 grid (corresponding to $\chi = 118$) is only accurate until $t/T_0 \approx 1.2$ while the lower-resolution URDNS already fails at $t/T_0 \approx 0.75$. This is observed in both the instantaneous vorticity dynamics results and the Reynolds-stresses. The MPS algorithm achieves a much higher accuracy than URDNS for the same number

of variables (see Table 7.1 for quantitative results). This result can also be understood in terms of the interscale correlations shown in Figure 2.11a, where the domain corresponding to URDNS on the 256^2 grid is shown by the grey-shaded area $\mathcal{W}$. A significant amount of correlations present in the DNS solutions are outside of $\mathcal{W}$, which is consistent with our finding that URDNS cannot accurately represent the DNS solutions.

7.3 3D turbulence: Taylor-Green vortex

We now move on the TGV flow outlined in Section 2.6.2. In the units defined there, $\mu = 2.5 \times 10^5$ in all the MPS simulations. The top row in Figure 7.2a corresponds to DNS and illustrates how the original vortex collapses ($t/T_0 = 0.2$) into turbulent worm-like structures ($t/T_0 = 0.8$) which become progressively more turbulent ($t/T_0 = 1.4$) until viscosity eventually dissipates these vortical structures ($t/T_0 = 2$). Rows 2–4 and 5–7 in Figure 7.2a correspond to the results of our MPS algorithm and to URDNS, respectively. The bond dimension and grid sizes have been chosen such that the compression ratios compared to DNS are approximately 1:25 (rows 2 and 5), 1:49 (rows 3 and 6) and 1:78 (rows 4 and 7). While MPS produces a solution comparable to DNS for a compression ratio of 1:49 ($\chi = 128$), the corresponding URDNS results clearly deviate from DNS. Discrepancies between URDNS and DNS are even visible for the largest URDNS grid (compression 1:25).

A more quantitative analysis of the performance of MPS vs. URDNS is shown in Figure 7.2b. In the non-DNS simulations, a portion of the energy is erroneously lost to numerical diffusion. The amount of numerical diffusion can be measured by comparing the observed kinetic energy dissipation

$$\varepsilon(t) = -\frac{1}{2}\frac{d}{dt}\int_{\mathcal{V}} \left|\boldsymbol{V}(\boldsymbol{r},t)\right|^2 \mathrm{d}\mathbf{r} \tag{7.3}$$

to the physical global dissipation (enstrophy)

$$\zeta(t) = \nu \int_{\mathcal{V}} \left|\nabla \times \boldsymbol{V}(\boldsymbol{r},t)\right|^2 \mathrm{d}\mathbf{r}, \tag{7.4}$$

where we integrate over the whole space $\mathcal{V}$. ζ is related to the viscous dissipation of kinetic energy [112]. For incompressible flows with periodic boundary conditions, the

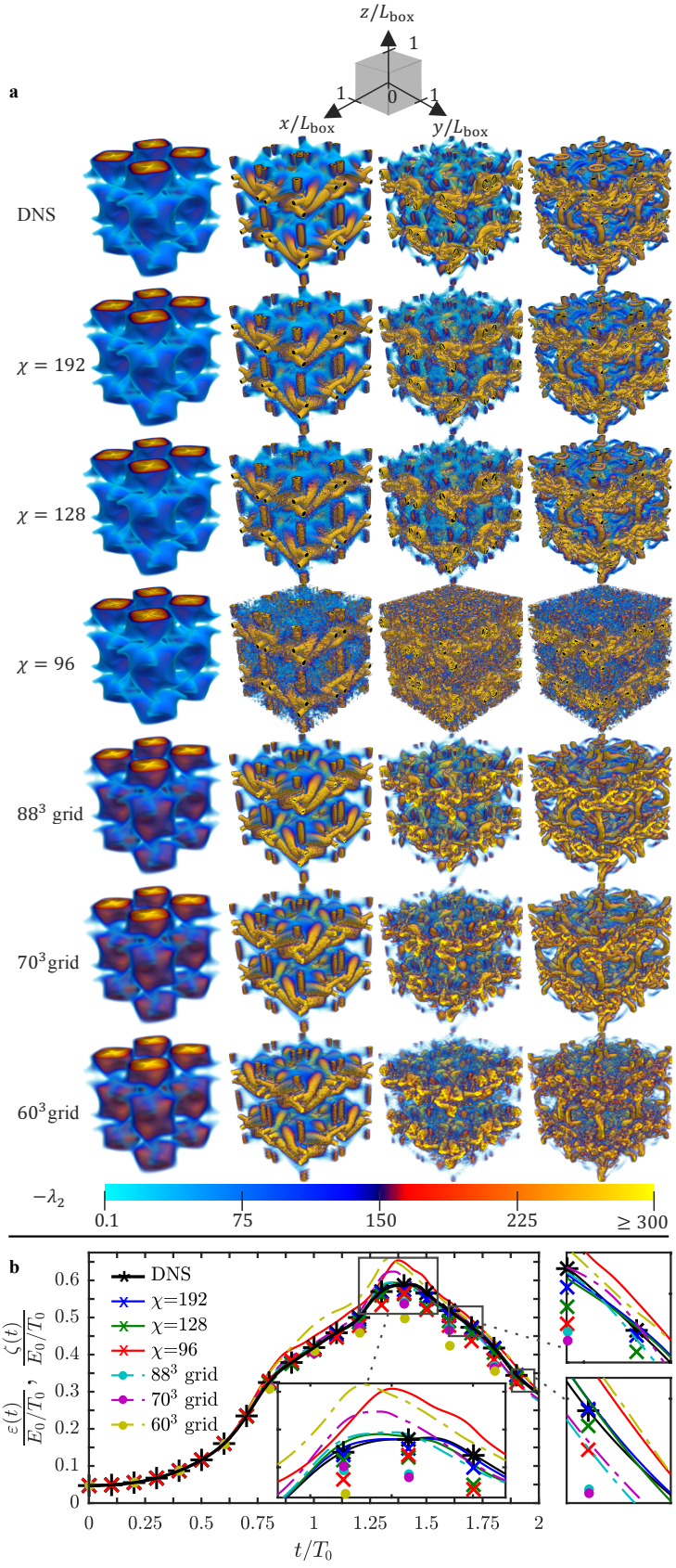

Figure 7.2: **3D Taylor-Green vortex.** Dynamical simulations of the incompressible Navier-Stokes equations in 3D for the Taylor-Green vortex at Re = 800 for the parameters defined in Section 2.6.2. **a** Vortical structures rendered using the standard λ_2 method [61] are shown at times $t/T_0 = 0.2, 0.8, 1.4, 2$ (left to right). Top row is for DNS on a $2^8 \times 2^8 \times 2^8$ grid (also plotted in Figure 2.6). Rows 2–4 are for MPS simulations with different χ, and bottom three rows are for URDNS on cubic grids as indicated. **b** shows the enstrophy $\zeta(t)$ (asterisks/crosses/circles) and the energy dissipation $\epsilon(t)$ (lines) as a function of time, with E_0 being the total energy at $t = 0$.

incompressible Navier-Stokes equations imply $\varepsilon(t) = \zeta(t)$, however, restricting the solution space results in numerical diffusion that violates this equality. The $\varepsilon(t)$ versus $\zeta(t)$ predictions for MPS at $\chi \geq 128$ are consistent with both the DNS results here and in previous work [60]. We find that the MPS simulations with $\chi = 128$ and 192 dissipate the energy more accurately than any of the URDNS results, especially for $t/T_0 \geq 1.4$, see Table 7.1. As in the 2D case, this outcome is in line with the interscale correlations shown in Figure 2.11b, where the domain corresponding to URDNS on the 64^3 grid is shown by the grey-shaded area $\mathcal{W}$. The bipartitions at $n = 5, 6, 7$ are associated with comparatively large Schmidt numbers for $t/T_0 \geq 0.8$, and hence these interscale correlations cannot be captured by URDNS.

7.4 Quantitative comparison between simulations

Here we provide a more quantitative comparison between the MPS and URDNS accuracy. The accuracy of MPS and URDNS simulations are gauged by comparing the ensemble-aggregated quantities of Figures 7.1b and 7.2b against DNS (statistical quantities such as these are needed for the comparisons to be rigorous, due to the chaotic nature of turbulence). In this section, we quantify this comparison by integrating the discrepancy DNS has to MPS and URDNS in Figures 7.1b and 7.2b.

The *visual* difference between the Figure 7.1b Reynolds stress of DNS (row 1) to that of MPS (rows 2–4) and URDNS (rows 5–7) can be numerically evaluated through the variable

$$\sigma(s,c) = \sqrt{\frac{\int_{L_{\text{box}}/5}^{4L_{\text{box}}/5}\int_0^{2T_0}\left[\tau_{12}^{\text{DNS}}(y,t) - \tau_{12}^{s,c}(y,t)\right]^2 \mathrm{d}y\mathrm{d}t}{\frac{6}{5}L_{\text{box}}T_0\Big[\max\limits_{y,t}\left(\tau_{12}^{\text{DNS}}(y,t)\right) - \min\limits_{y,t}\left(\tau_{12}^{\text{DNS}}(y,t)\right)\Big]^2}}, \tag{7.5}$$

with s being the scheme in question and c the compression ratio of said scheme compared to DNS. $\sigma(s,c)$ quantifies the root-mean-square of the *visual* difference between the subplots in Figure 7.1b, and the closer $\sigma(s,c)$ is to 0, the nearer the Reynolds stress in question is to that of DNS. $\sigma(s,c)$ is tabulated in Table 7.1, and it indicates MPS is more accurate than URDNS when the same number of variables is used in both schemes to parameterise the solution. The difference is particularly striking at $c = 1{:}8$, where $\sigma(\text{MPS}, 1{:}8)$ is less than a twentieth of $\sigma(\text{URDNS}, 1{:}8)$.

We also quantify the accuracy of URDNS and MPS against DNS for the TGV flow by integrating the numerical diffusion $|\zeta(t) - \epsilon(t)|$ of Figure 7.2b. This is equivalent to

$$e(s,c) = \frac{1}{E_0}\int_0^{2T_0} |\zeta_{s,c}(t) - \epsilon_{s,c}(t)|\mathrm{d}t, \tag{7.6}$$

when normalised by the initial total kinetic energy E_0. The lower $e(s,c)$ is, the better is the accuracy of the relevant simulation. $e(s,c)$ is tabulated in Table 7.1, where the errors of the MPS and URDNS simulations are compared to the (small) error of DNS, which is $e(\text{DNS}) = 0.0020$, equivalent to $\approx 0.2\%$ of the initial kinetic energy being lost to the (unphysical) numerical diffusion during the course of the entire simulation. From the table we see that MPS does relatively well compared to URDNS, with both $e(\text{MPS}, 1{:}25)$ and $e(\text{MPS}, 1{:}49)$ being below 10%, which is in accord with the assertion that MPS is capable of capturing the key properties of the TGV flow, despite the presence of fine-scale noise.

Case	Compression	Scheme	Inaccuracy
TDJ	1:1	DNS	0
TDJ	1:8	MPS	0.0119
TDJ	1:8	URDNS	0.2612
TDJ	1:16	MPS	0.0485
TDJ	1:16	URDNS	0.3333
TDJ	1:64	MPS	0.2404
TDJ	1:64	URDNS	0.3201
TGV	1:1	DNS	0.0020
TGV	1:25	MPS	0.0385
TGV	1:25	URDNS	0.1599
TGV	1:49	MPS	0.0844
TGV	1:49	URDNS	0.2133
TGV	1:78	MPS	0.2618
TGV	1:78	URDNS	0.4563

Table 7.1: **Quantitative comparison between MPS and URDNS simulations for the TDJ and TGV flow cases**. The rows corresponding to the TDJ flow tabulate $\sigma(s,c)$, as defined in Equation (7.5). $\sigma(s,c)$ measures the discrepancy of the URDNS and MPS Reynolds stresses to that of DNS in Figure 7.1b. The TGV flow case rows tabulate $e(s,c)$, which is defined in Equation (7.6) and represents the total numerical diffusion in Figure 7.2b. The nearer $\sigma(s,c)$ or $e(s,c)$ is to 0, the more accurate the simulation in question is.

7.5 Conclusion

The results of this chapter demonstrate that the paradigmatic TDJ and TGV flows contain scale-local structure that can be exploited with MPSs. The structures is reflected in the limited amount of interscale correlations present in these turbulent flows (recall Figure 2.11) and is the reason why the MPS simulations remain highly accurate even at low χ. On the other hand, coarsening the grid and running URDNS fails to accurately simulate any of the flow cases.

The inaccuracy of URDNS exemplifies the incongruence between scale-resolving CFD approaches (i.e. DNS/URDNS, LES) and the multiscale nature of turbulence. While coarsening the grid removes from the solution space only the smallest and least energetic eddies, these flow-structures are still important in realistic flow, since they interact and correlate with larger eddies and thus help drive turbulent flow (as Figure 2.11 shows). Removing them from consideration therefore leads to inaccuracy.

In contrast, our structure-resolving MPS scheme does not truncate any length scales. It instead discards large amounts of unphysical interscale correlations (i.e. the upper triangles where $d(n) > \chi_{99}$ in Figure 2.11) from the solution space, thus reducing the dimensionality of turbulence simulations without significantly impacting accuracy.

The structure-resolving properties of MPSs can reduce computational costs. The computational cost of our MPS algorithm goes as $\sim \chi^4 \log M$, as explained in Section 5.4. Resolving down to the Kolmogorov microscale requires $M \sim (\ell/\eta)^K \sim \mathrm{Re}^{3K/4}$ grid points. Assuming $\chi \sim \mathrm{Re}^\gamma$, the overall MPS algorithm scales as $\sim \mathrm{Re}^{4\gamma} \log \mathrm{Re}$. Comparing this to the cost of DNS, which is $\sim M \log M \sim \mathrm{Re}^{3K/4} \log \mathrm{Re}$ [Appendix A], implies the MPS algorithm outperforms DNS when $\gamma < 3K/16$. For our 2D example, Figure 2.10 suggests $\gamma \approx 0$ which leads to an exponential speedup of the MPS algorithm over DNS for sufficiently large Re. For the TGV flow $\gamma \approx 0.71$, which is larger than $3K/16$. However, we note that numerical methods for manipulating tensors are an active field of research and that it might be viable to reduce the cost of our MPS algorithm along the lines discussed in Section 5.4.

Chapter 8

Alternative tensor network geometries

8.1 Introduction

The previous Chapter 7 demonstrates that turbulence can be simulated in a highly data-compressed MPS form. But the MPS ansatz is only one specific type of tensor network that can be used to exploit turbulence structures.

The MPS structure is ideal for capturing correlations between nearby sites (i.e. nearest-neighbour correlation) regardless of whether the sites represent length scales or quantum particles. This is due to the connectivity of MPS solely being between neighbouring sites (see e.g. Figure 8.1a). However, in actual turbulent flows, it cannot always be expected that these correlations are strictly nearest-neighbour. In Section 2.4 it is noted that while turbulent interactions are considered "scale-local" due to them decaying polynomially with separation in scale, this polynomial decay still means that *all* the scales interact with each-other to a non-negligible extent. Such interactions might generate relatively long-range correlations that are challenging to capture with MPS (as seen in Figure 2.10 for the 3D TGV flow case), motivating the need to consider more complicated tensor network geometries for simulating turbulence.

8.1.1 Chapter structure

Many tensor network geometries exist (both known and unknown) that might potentially be useful for representing turbulence. For the sake of simplicity, we only provide a discussion on two geometries: tree tensor networks (TTNs) in Section 8.2 and the

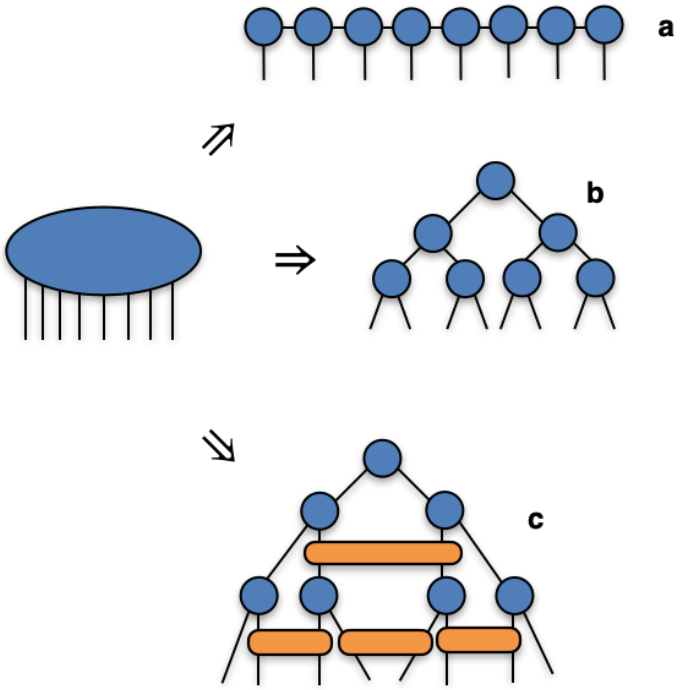

Figure 8.1: **Three possible decompositions from one tensor**. In **a**, an eight-order tensor has been decomposed into an $N = 8$ MPS where only the nearest neighbours are connected. In **b**, the same tensor has been decomposed into a TTN, which connects distant sites. The MERA decomposition is shown in **c**. Note both the increased connectivity compared to MPS and TTN, as well as the presence of loops.

multiscale entanglement renormalisation ansatz (MERA) in Section 8.3. The *ideal* ansatz must be capable of perfectly capturing the correlation structure of turbulent flows, as is discussed in Section 8.4. The chapter is concluded in Section 8.5.

8.2 Tree tensor networks

It is noted in the above that the MPS connectivity is ideally suited for resolving nearest-neighbour correlations, while more long-range correlations necessitate alternative tensor network geometries that connect more distant sites. The TTN [73, 113, 114] is one such geometry and it is illustrated in Figure 8.1b. The tree structure has much better connectivity between distant sites than the MPS geometry. For instance in Figure 8.1, to move from the TTN site associated with the first (leftmost) open bond to that which is associated with the last (rightmost) open bond, one only needs to pass through four

closed bonds, while in the equivalent eight-site MPS one would have to pass through seven closed bonds. This advantage might allow TTNs to represent turbulent flows with a *lower* bond dimension χ than MPS.

However, compared to MPSs the increased connectivity of TTNs has two drawbacks. First, it decreases the connectivity between neighbouring sites. To see, consider the second site in the TTN of Figure 8.1b. Moving from the second to third (neighbouring) site requires going through four bonds, which is significantly longer than a corresponding move in the MPS of Figure 8.1a. This disadvantage in connectivity between neighbouring sites increases with N as $\sim \log N$ and is in discord with the scale-local nature of turbulence. The second drawback is that manipulating TTNs is in general more computationally expensive than manipulating MPSs. For instance, the computational cost of just putting a TTN into canonical form is $\sim N\chi^4$ [73], while doing the same for MPS only costs $\sim N\chi^3$.

8.3 Multiscale entanglement renormalisation ansatz

The MERA network generalises TTNs and is illustrated in Figure 8.1c. MERA connects distant sites (just like TTNs) while simultaneously maintaining connectivity between neighbouring sites (like MPS). This connectivity allows MERA to capture correlations that are both near and distant, and should enable MERA networks to represent turbulent flow fields with a lower bond-dimension than MPS, especially if the correlations are long-range.

However, the increased connectivity of MERA unfortunately leads to the presence of loops (e.g. it is possible to walk along the outer edge of the MERA to form a triangular loop). These loops create various numerical difficulties, such as making it impossible to straightforwardly Schmidt-decompose the network and truncate the bond dimension in a globally optimal manner, that require complicated numerical methods to resolve [115]. Furthermore, contracting MERA networks is in general very expensive [116].

8.4 Structure of interscale correlation in turbulence

Till now we have only evaluated tensor network ansätze based upon their connectivity. In addition to connectivity, it is also important to understand how the bond dimension χ required to represent turbulence must scale with increasing Reynolds number. The case of MPS, for instance, is illustrated in Figure 2.10 of Section 2.7 for both the TDJ and TGV flows. Let us investigate what this tells us about the structure of the interscale correlations in these flows and what tensor network geometry is most appropriate for representing turbulence.

Consider Figure 2.10 in the context of the von Neumann entanglement entropy (previously outlined in Sections 2.2 and 2.6). Equations (2.4) and (2.5) imply that the entanglement entropy is at most $S(n,t) \leq \log d(n)$. Applying the MPS ansatz and setting $d(n) \leq \chi$ gives

$$S(n,t) \leq \log \chi. \tag{8.1}$$

In the case of the TDJ flow, Figure 2.10 implies that the bond dimension required to resolve the flow at 99% accuracy within $\mathcal{M}$ saturates with Re as $\chi \sim \text{const}$. This indicates

$$S_{\max} \sim \log \chi \sim \text{const}, \tag{8.2}$$

with $S_{\max} = \max_{n,t}[S(n,t)]$. When the entanglement entropy saturates with the system size like this, Equation (3.21) implies that the MPS representation becomes exponentially more efficient than that of DNS. This is the case for both this 2D TDJ flow example and for 1D quantum systems behaving according to the area law (Section 2.3). For the TGV flow, Figure 2.10 implies that the bond dimension must go as $\chi \sim \text{Re}^{0.71}$ if the flow is to be resolved at 99% accuracy within $\mathcal{M}$. This indicates

$$S_{\max} \sim \log \text{Re}^{0.71} \sim \log \text{Re}. \tag{8.3}$$

Furthermore, Kolmogorov's theory states that the number of gridpoints 8^N scales with Reynolds number as $8^N \sim \text{Re}^{9/4}$. Inserting these into Equation (8.3) gives

$$S_{\max} \sim \log 8^{N \cdot 0.71/2.25} \sim N, \tag{8.4}$$

which means the entanglement entropy scales linearly in the number of sites N (or equivalently, in the number of length scales). This $S_{\max} \sim N$ scaling is known as a *volume law*, since the entanglement entropy increases with the overall size ("volume") of the MPS.

A volume law can be effectively captured by neither MPSs, TTNs nor MERA. TTNs, like MPSs, are ideal only when $S_{\max} \sim$ const, whilst MERA can at most handle $S_{\max} \sim \log N$.

There do however exist more sophisticated ansätze capable of capturing entanglement entropy increasing like it does in Equation (8.4). The most obvious example is the 2D generalisation of MPS, PEPS, which exploits that what seems like a volume law on a 1D MPS lattice, could actually be an area law on a 2D PEPS lattice. Furthermore, the PEPS geometry is far better suited at representing correlations between distant sites than is MPS [17], which (similarly to TTNs and MERA) might be useful for representing turbulent flow fields characterised by significant correlation between faraway scales. The branching MERA [79] is another example of a tensor network scheme capable of efficiently capturing volume laws, but only when the entanglement is structured in a specific way. Quantum computers might also be useful, given that they are intrinsically able to represent large amounts of entanglement. To this end, a tensor network inspired ansatz capable of representing velocity fields on quantum hardware is described in [117, 118]. If an ansatz is found that precisely matches the structure of the TGV flow, the bond-dimension (or number of variational parameters, in the case of quantum circuits) would increase sub-polynomially with Re instead of the $\chi \sim$ poly(Re) scaling when MPSs, TTNs or MERA are used to encode the velocity field.

8.5 Conclusion

In this chapter it is briefly discussed what advantages alternative tensor network geometries hold over MPSs for encoding turbulence. We find that TTNs and MERA networks allow for better connectivity between distant length scales, and might therefore require lower bond-dimensions than MPSs to perform accurate fluids simulations. Furthermore, the PEPS and branching MERA geometries are capable of capturing the entanglement entropy scaling observed in the TGV case, which could in the best case allow PEPS and branching-MERA to simulate 3D flows such as the TGV using a bond-dimension which scales *exponentially* better with Re than the MPS bond-dimension. However, while these alternative geometries might require lower bond-dimensions during simulation, algorithms based upon them are significantly more complicated to implement and are much more computationally expensive to use than

the MPS scheme of this thesis. We leave it to future work to decide on whether these alternative candidates are truly better suited at encoding and simulating turbulence than the MPS ansatz.

Chapter 9

Conclusion and outlook

This thesis demonstrates that tensor networks are capable of efficiently taking advantage of the structure of turbulence. The scale-local nature of turbulence often leads to correlations between distant length scales being limited, which we capitalise on by encoding the different length scales of flow into individual tensors on MPS chains where the bond-dimension χ is limited. The number of variables required by the MPS encoding goes as $\sim \chi^2 \log(\ell/\eta)$, allowing us to circumvent a central roadblock of traditional scale-resolving CFD methods, namely their inability to resolve the extremely broad range of length scales between ℓ and η that typically becomes energised during turbulence (e.g. DNS requires $\sim (\ell/\eta)^K$ variables to accurately represent K–dimensional flow). When the flow can be accurately described at low χ, the logarithmic scaling of our encoding opens the possibility for MPS simulations of fluid dynamics that are exponentially faster than traditional CFD.

We investigated the practical utility of MPSs for CFD by formulating a variational MPS algorithm whose cost scales as $\sim \chi^4 \log(\ell/\eta)$ and used it to simulate fluid dynamics in one, two and three dimensions. Our algorithm was first used to simulate the shock-waves generated by Burgers' equations (a non-turbulent simplification of the Navier-Stokes equations). For this problem, we showed both analytically and numerically that the MPS scheme only requires $\chi = 3$ to maintain accuracy, making it exponentially advantageous compared to DNS. Moving on, we then simulated the 2D Kelvin-Helmholtz instability. The results indicate that MPSs can also here be used to exponentially reduce the computational cost compared to DNS. It would be interesting to investigate whether this is just a unique case for the TDJ flow or if it is a more general property of 2D turbulence, because if this exponential reduction

in cost is indeed general, it would have significant practical consequences in e.g. the simulation of atmospheric flows. We also investigated the turbulent collapse of the 3D Taylor-Green vortex, which we found our MPS encoding can accurately capture using just a tiny fraction of the variables required by DNS. For this specific flow case, however, the MPS representation required an overly large χ to maintain accuracy and therefore our MPS algorithm was unable to demonstrate any reduction in computational complexity compared to DNS. This issue might potentially be overcome by following the steps outlined in Section 5.4 to reduce the scaling in χ of our MPS scheme down to $\sim \chi^3 \log(\ell/\eta)$.

If traditional CFD approaches (e.g. DNS, LES) are *scale*-resolving, tensor networks methods are instead *structure*-resolving. Structure-resolving methodologies are those which represent flow in terms of turbulent structures that span all scales between ℓ and η, and have previously been used for analysing turbulence [119] (i.e., diagnostics), but not simulating it (i.e., prediction). Simulating turbulence with structure-resolving schemes has historically been hampered by difficulties in identifying physically valid structures with which to represent the flow [117, 119]. Our scheme overcomes this problem by representing the flow in terms of a MPS structure that is valid when the flow is scale-locally correlated. As such, this MPS algorithm is to our knowledge the first structure-resolving scheme capable of robustly simulating turbulent flows.

Our MPS encoding captures the scale-local aspect of the structure of turbulence; however, other aspects might be better exploited using alternative ansätze. For instance, MPS can turn numerically inefficient when correlations between distant scales become relevant, because then a very large χ will be needed to maintain an accurate description of the flow. In such cases, χ might be reduced by switching to tensor networks that better connect distant length scales together, like TTNs or MERA. However, for any of these networks to encode the observed 3D turbulence accurately, the required bond-dimension must still scale as $\chi \sim \text{poly}(\text{Re})$ [Section 8.4]. It might be possible to drastically reduce this scaling, possibly even down to $\chi \sim \text{const}$, if an ansatz is found that corresponds precisely to the structure of turbulence (like how the MPS ansatz matches the structure of low-energy states of certain quantum systems). A good starting point might be to investigate how well PEPS and branching MERA can encode the 3D TGV flow [Section 8.4].

The utility of tensor networks in fluid dynamics goes beyond the incompressible Navier-Stokes equations. Future avenues of investigation for MPSs include compressible

flows, in which the Mach number is an important parameter, and transport of scalar quantities under both passive and chemically reactive conditions where the effects of Prandtl, Peclet and Damkohler numbers [120] must be taken into account. It would be interesting to examine how these parameters affect the fidelity of low χ MPS simulations. Furthermore, as tensor network methods are naturally suited to tackle high-dimensional problems, their applicability to the transported probability density function (PDF) of turbulent reactive flows [121] should be considered. In these flows, in addition to temporal and spatial variations, the PDF is a function of the three-dimensional velocity field along with the pertinent scalar variables (energy, pressure and species mass fractions) [122]. With just 10 species (a very simple chemical kinetics model), the unsteady PDF must be resolved in a 17-dimensional space. High-fidelity modelling and simulation of such complex flows can potentially be enabled through a well-chosen tensor network ansatz.

The close connection of our tensor network-based approach to quantum physics points towards the prospect of solving the Navier-Stokes equations on a quantum computer [117, 123]. Recently, several algorithms for solving nonlinear partial differential equations on quantum computers have been proposed [118, 124, 125]. The elegant scheme of [124] linearises nonlinear differential equations by exploiting the connection between the nonlinear Schrödinger equation and the (linear) Schrödinger equation governing Bose-Einstein condensates. In the highly rigorous study of [125], nonlinear differential equations are mapped into higher-dimensional linear problems using Carleman linearisation. The advantage of these two schemes is that their linearisations allows them to leverage exponentially efficient [126–128] quantum methods for solving linear differential equations. The disadvantage is that they can only be expected to remain accurate when the nonlinearity is weak, i.e. for low Re in the context of the Navier-Stokes equations. One approach that might *not* be limited to low Re however, is that of [118]. This work introduces a tensor network based programming paradigm for quantum computers that allows our MPS algorithm to be ported onto quantum hardware. This would involve encoding the flow fields as quantum states and using the formalism of [129] to minimise Equation (5.7) on quantum computers. Carrying out this minimisation within the MPS manifold will immediately reduce the scaling with bond-dimension to $\sim \chi^2$. Further, the variational manifold available to a quantum circuit is more general than just the MPS manifold [118] due to the intrinsic ability of quantum computers to represent large amounts of entanglement, and so the accuracy of our algorithm should also improve when it is run on quantum hardware.

In summary, our work demonstrates that tensor network methods can be used to efficiently exploit the structures of turbulence to simulate fluid dynamics on both classical and quantum hardware. Future work along this avenue holds the promise of enabling large-scale CFD calculations that are well beyond the scope of current approaches.

Appendix A

Direct numerical simulation algorithm

We here briefly outline our DNS scheme. To solve the incompressible Navier-Stokes equations, it employs second-order Runge-Kutta temporal discretisation combined with an eighth-order central finite difference discretisation of the spatial derivatives [97] on a Cartesian grid to solve Burgers' equation, whilst enforcing the incompressibility condition by continuously projecting the velocity field onto the divergence-free subspace as per Chorin's method [130]. Disabling this projection step will result in the algorithm simply solving Burgers' equation in place of the incompressible Navier-Stokes equations.

The computational cost of the DNS scheme is $\sim M \log M$. This is because it is dominated by the projection step, which is performed through repeated fast Fourier transform and inverse fast Fourier transforms that scale as $\sim M \log M$. In comparison, the matrix-vector multiplications in the DNS algorithm (which is all that is needed to solve Burgers' equation) scale as just $\sim M$ when the sparsity of the finite difference stencils is exploited.

If there are just enough gridpoints to resolve all scales from ℓ to η, the scheme is true DNS and solves the incompressible Navier-Stokes equations *exactly* within a sufficiently large solution space $\mathcal{D}$ (see Figures 2.11a and 2.11b). If the finest scales are removed however (without invoking any subgrid scale model) such that the smallest remaining resolved scale is significantly larger than η, then the scheme becomes an URDNS operating within the scale-restricted $\mathcal{W} \subseteq \mathcal{D}$. The Navier-Stokes equations cannot be solved exactly within $\mathcal{W}$ due to the finest scales being subject to unphysical numerical dissipation. In comparison, the MPS algorithm operates within the MPS manifold of $\mathcal{M} \subseteq \mathcal{D}$ where the interscale correlations are limited while all the scales

between ℓ and η are still present.

URDNS can be considered as the most basic form of large eddy simulations [8, 10, 11], where no explicit models are employed to account for the disregarded subgrid scales.

Appendix B

Two-point correlations between wavenumbers

Recall that in Section 2.4 we argued that fluid flows are characterised by scale-local interactions which in turn should lead to scale-local correlations. This scale-locality can be illustrated by studying the two-point correlations between various scales in wavenumber space in the above two flow cases. Let u_i' be the velocity fluctuation

$$u_i'(t, \boldsymbol{r}) = u_i(t, \boldsymbol{r}) - \langle u_i \rangle (t), \tag{B.1}$$

with $\langle \cdot \rangle$ representing an averaging along all real-space directions in which the flow is statistically homogeneous. For the 2D TDJ, that would be along the x-axis, whilst for the 3D TGV flow this would be along *all* axes due to the isotropy of the flow. $\widehat{u}_i'$ is the Fourier transform of the i-th flow component,

$$\widehat{u}_i'(t, \boldsymbol{k}) = \int_{\mathcal{V}} u_i'(t, \boldsymbol{r}) e^{-\mathrm{i}\boldsymbol{k}.\boldsymbol{r}} \mathrm{d}\boldsymbol{r}, \tag{B.2}$$

with $\mathcal{V}$ being the relevant spatial domain. The two-point correlation can then be expressed through the tensor

$$C_{ij}(t, \delta\boldsymbol{k}) = \int_{\widehat{\mathcal{V}}} \widehat{u}_i'(t, \boldsymbol{k} + \delta\boldsymbol{k}) \big(\widehat{u}_j'(t, \boldsymbol{k})\big)^* \mathrm{d}\boldsymbol{k}, \tag{B.3}$$

with $\widehat{\mathcal{V}}$ being the Fourier domain.

Before continuing, we briefly remark on the side that C_{ij} is linked to the energy of the flow. To see, let E_{ij} be the inverse Fourier transform of C_{ij} as

$$E_{ij}(t, \boldsymbol{r}) = \frac{1}{2\pi} \int_{\widehat{\mathcal{V}}} C_{ij}(t, \delta\boldsymbol{k}) e^{\mathrm{i}\delta\boldsymbol{k}.\boldsymbol{r}} \mathrm{d}\delta\boldsymbol{k}. \tag{B.4}$$

E_{ij} and C_{ij} must now be a Fourier transform pair, which implies that

$$C_{ij}(t, \delta\boldsymbol{k}) = \int_{\mathcal{V}} E_{ij}(t, \boldsymbol{r}) e^{-\mathrm{i}\delta\boldsymbol{k}.\boldsymbol{r}} \mathrm{d}\boldsymbol{r}. \tag{B.5}$$

Setting $i = j$ and $\delta\boldsymbol{k} = 0$ in Eqs. (B.3) and (B.5) instantly gives

$$\int_{\widehat{\mathcal{V}}} |\widehat{u_i'}(t, \boldsymbol{k})|^2 \mathrm{d}\boldsymbol{k} = \int_{\mathcal{V}} E_{ii}(t, \boldsymbol{r}) \mathrm{d}\boldsymbol{r}, \tag{B.6}$$

allowing us to set

$$E_{ii}(t, \boldsymbol{r}) = |u_i'|^2(t, \boldsymbol{r}). \tag{B.7}$$

Therefore, the Fourier-space two-point correlations of any component of the turbulent velocity fluctuations is simply the Fourier transform of the real-space distribution of the turbulent kinetic energy of said component[1].

Moving on, we now normalise C_{ij} in the traditional manner by dividing it with the standard deviations of the fluctuations, giving

$$C_{ij}'(t, \delta\boldsymbol{k}) = \frac{C_{ij}(t, \delta\boldsymbol{k})}{\sqrt{C_{ii}(t, 0) C_{jj}(t, 0)}}. \tag{B.8}$$

The Schwartz inequality now commands that $C_{ij}' \leq 1$. At this point, recall that we are only interested in the potential scale-locality of correlations between different scales, yet Equation (B.8) also contains information on the direction of $\delta\boldsymbol{k}$. Let us simplify our analysis by eliminating this directional information by integrating Equation (B.8) over spheres $\mathcal{S}$ of radii $|\delta\boldsymbol{k}|$:

$$c_{ij}(t, |\delta\boldsymbol{k}|) = \oint C_{ij}'(t, \delta\boldsymbol{k}) \mathrm{d}\mathcal{S}(|\delta\boldsymbol{k}|). \tag{B.9}$$

The resulting correlation tensor $c_{ij}(t, |\delta\boldsymbol{k}|)$ is now a normalised object describing the autocorrelation between wavenumbers.

Let us now use this tensor to study the inter-wavenumber correlations within the TDJ and TGV flows of Sections 2.6.1 and 2.6.2. In Figures B.1 and B.2, the diagonal components of $c(t, |\delta\boldsymbol{k}|)$ are plotted for the TDJ and TGV flows at various times t/T_0. The immediate observation is that $c(t, |\delta\boldsymbol{k}|)$ significantly decays with increasing $|\delta\boldsymbol{k}|$ in all cases. In all the TDJ and TGV plots, except the TGV plot at time $t/T_0 = 0.2$, the correlations remain steady until about $|\delta\boldsymbol{k}| \sim 10^1$, but afterwards decay at a rate of about two to three orders of magnitude per wavenumber decade. Thus once the

[1]This result is essentially a restatement of the Wiener-Khinchin theorem.

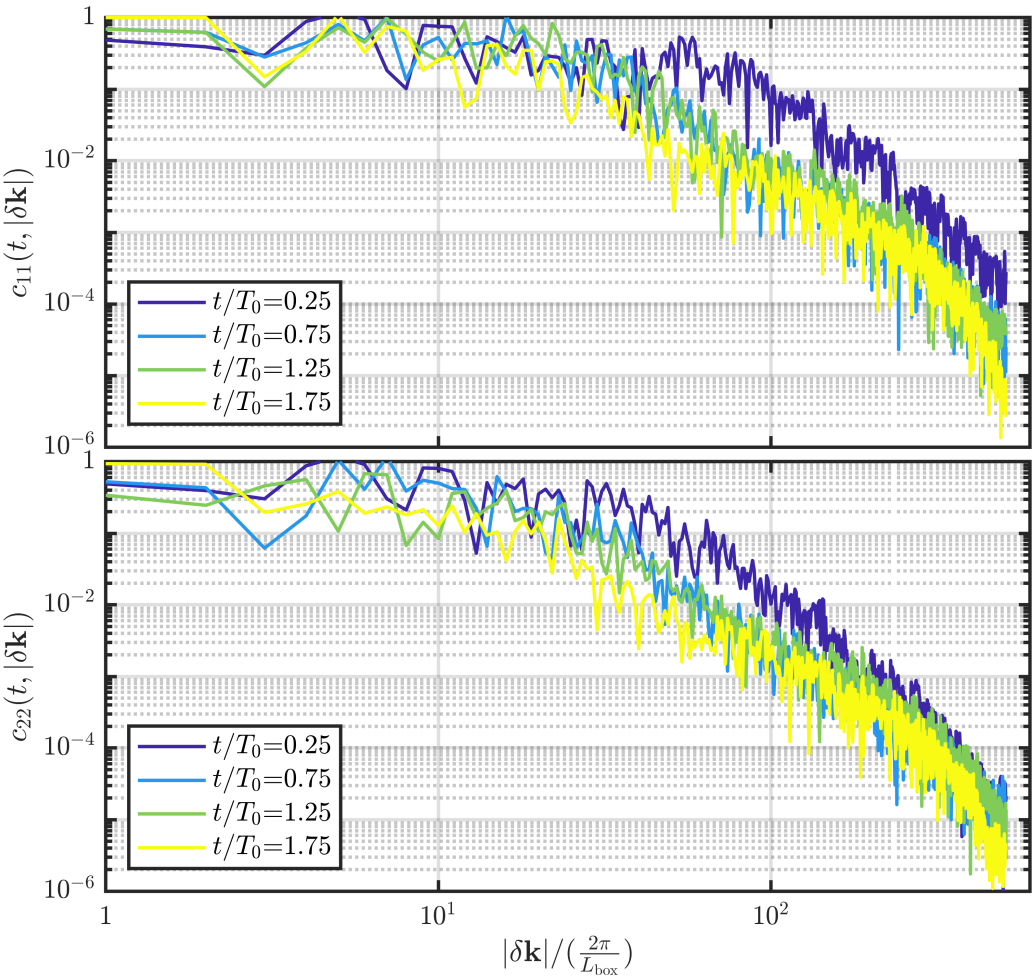

Figure B.1: **Wavenumber autocorrelations of the 2D TDJ.** The autocorrelations tensor $c_{ij}(t, |\delta\boldsymbol{k}|)$, as defined in Equation (B.9), is plotted across wavenumber-separations $|\delta\boldsymbol{k}|$ for $i = j$ at four different times.

instabilities of the TDJ and TGV flows lead to the large-scale structures collapsing into turbulence, the wavenumbers remain correlated for only about a decade. This points to the correlations between different scales being scale-local, in line with the scale-locality of turbulent interactions [Section 2.4]. Furthermore, this scale-local nature of $c(t, |\delta\boldsymbol{k}|)$ illustrates the aforementioned universality hypothesis of turbulence: when $c(t, |\delta\boldsymbol{k}|)$ consistently decays with increasing $|\delta\boldsymbol{k}|$, the motions at large and small scales become increasingly unrelated to each other.

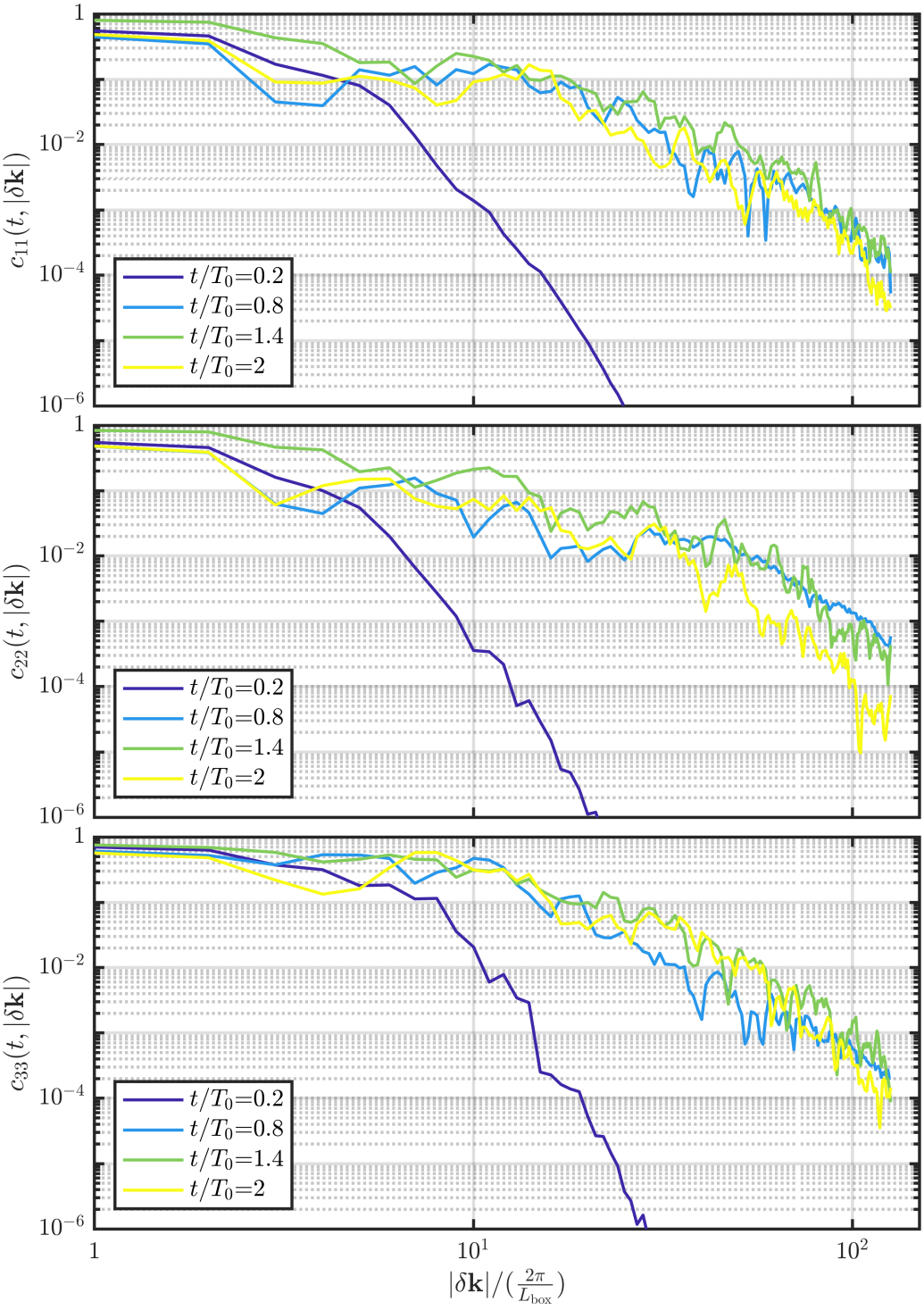

Figure B.2: **Wavenumber autocorrelations of the 3D TGV.** The autocorrelations tensor $c_{ij}(t, |\delta\boldsymbol{k}|)$, as defined in Equation (B.9), is plotted across wavenumber-separations $|\delta\boldsymbol{k}|$ for $i = j$ at four different times.

Bibliography

1. Feynman, R. P., Leighton, R. B. & Sands, M. *The Feynman lectures on physics; New millennium ed.* (Basic Books, New York, NY, 2010).
2. Slotnick, J. *et al. CFD Vision 2030 Study: A path to revolutionary computational aerosciences* tech. rep. (NASA, 2014).
3. Monin, A. & Yaglom, A. *Statistical fluid dynamics: mechanics of turbulence, vol. I* (Dover, New York, 2007).
4. Monin, A. & Yaglom, A. *Statistical fluid dynamics: mechanics of turbulence, vol. II* (Dover, New York, 2007).
5. Kolmogorov, A. The local structure of turbulence in incompressible viscous fluid for very large Reynolds' numbers. *Dokl. Akad. Nauk SSSR* **30,** 301–305 (1941).
6. Moin, P. & Kim, J. Tackling turbulence with supercomputers. *Sci. Am.* **276,** 62–68 (1997).
7. Berger, M. J. & Oliger, J. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.* **53,** 484–512 (1984).
8. Zhiyin, Y. Large-eddy simulation: past, present and the future. *Chin. J. Aeronaut.* **28,** 11–24 (2015).
9. Berlemont, A., Desjonqueres, P. & Gouesbet, G. Particle lagrangian simulation in turbulent flows. *Int. J. Multiph. Flow* **16,** 19–34 (1990).
10. Sagaut, P. *Large eddy simulation for incompressible flows: an introduction* (Springer, Berlin, Heidelberg, 2006).
11. Pope, S. Ten questions concerning the large-eddy simulation of turbulent flows. *New J. Phys.* **6,** 35–35 (2004).
12. Poulin, D., Qarry, A., Somma, R. & Verstraete, F. Quantum simulation of time-dependent Hamiltonians and the convenient illusion of Hilbert space. *Phys. Rev. Lett.* **106,** 170501 (2011).
13. Manin, Y. *Computable and non-computable* tech. rep. (Sovietskoye Radio, 1980).
14. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys.* **21,** 467–488 (1982).
15. Lloyd, S. Universal quantum simulators. *Science* **273,** 1073–1078 (1996).

16. Sarma, S. D. *Quantum computing has a hype problem* tech. rep. (MIT Technol. rev., 2022).

17. Orús, R. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Ann. Phys.* **349,** 117–158 (2014).

18. Vidal, G. Entanglement renormalization. *Phys. Rev. Lett.* **99,** 220405 (2007).

19. Eisert, J., Cramer, M. & Plenio, M. Colloquium: area laws for the entanglement entropy. *Rev. Mod. Phys.* **82,** 277–306 (2010).

20. Verstraete, F., Murg, V. & Cirac, J. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Adv. Phys.* **57,** 143 (2008).

21. Schollwöck, U. The density-matrix renormalization group in the age of matrix product states. *Ann. Phys.* **326,** 96–192 (2011).

22. Verstraete, F. & Cirac, J. I. Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions. Preprint: arXiv:cond-mat/0407066.

23. Lubasch, M., Cirac, J. I. & Bañuls, M. C. Unifying projected entangled pair state contractions. *New J. Phys.* **16,** 033014 (2014).

24. Yan, S., Huse, D. A. & White, S. R. Spin-liquid ground state of the S = 1/2 Kagome Heisenberg antiferromagnet. *Science* **332,** 1173–1176 (2011).

25. Clark, S. R. & Jaksch, D. Dynamics of the superfluid to Mott-insulator transition in one dimension. *Phys. Rev. A* **70,** 043612 (2004).

26. Corboz, P. Improved energy extrapolation with infinite projected entangled-pair states applied to the two-dimensional Hubbard model. *Phys. Rev. B* **93,** 045116 (2016).

27. Szalay, S. *et al.* Tensor product methods and entanglement optimization for ab initio quantum chemistry. *Int. J. Quantum Chem.* **115,** 1342–1391 (2015).

28. Swingle, B. Entanglement renormalization and holography. *Phys. Rev. D* **86,** 065007 (2012).

29. Oseledets, I. Tensor-train decomposition. *SIAM J. Sci. Comput.* **33,** 2295 (2011).

30. Cichocki, A. *et al.* Tensor networks for dimensionality reduction and large-scale optimization: part 1 low-rank tensor decompositions. *Found. Trends Mach. Learn.* **9,** 249–429 (2016).

31. Stoudenmire, E. M. & Schwab, D. J. Supervised learning with quantum-inspired tensor networks. Preprint: arXiv:1605.05775.

32. Gallego, Á. J. & Orus, R. The physical structure of grammatical correlations: equivalences, formalizations and consequences. Preprint: arXiv:1708.01525.

33. Orús, R. Tensor networks for complex quantum systems. *Nat. Rev. Phys.* **1,** 538–550 (2019).

34. Haag, R. *Local quantum physics* 9–27 (Springer, Berlin, Heidelberg, 1992).

35. Eyink, G. & Aluie, H. Localness of energy cascade in hydrodynamic turbulence. I. Smooth coarse graining. *Phys. Fluids* **21,** 115107 (2009).

36. Aluie, H. & Eyink, G. Localness of energy cascade in hydrodynamic turbulence. II. Sharp spectral filter. *Phys. Fluids* **21,** 115108 (2009).

37. Cardesa, J., Vela-Martin, A. & Jimenez, J. The turbulent cascade in five dimensions. *Science* **357,** 782–784 (2017).

38. Chen, S. *et al.* Physical mechanism of the two-dimensional inverse energy cascade. *Phys. Rev. Lett.* **96,** 084502 (2006).

39. Demmel, J. *Applied numerical linear algebra* (SIAM, 1997).

40. Hastings, M. B. Solving gapped Hamiltonians locally. *Phys. Rev. B* **73,** 085115 (2006).

41. Wolf, M. M., Verstraete, F., Hastings, M. B. & Cirac, J. I. Area laws in quantum systems: mutual information and correlations. *Phys. Rev. Lett.* **100,** 070502 (2008).

42. Richardson, L. F. Weather prediction by numerical process. *Q. J. Roy. Meteor. Soc.* **48,** 282–284 (1922).

43. Chorin, A. Numerical solution of the Navier-Stokes equations. *Math. Comp.* **22,** 745–762 (1968).

44. Obukhov, A. M. Spectral energy distribution in a turbulent flow. *Dokl. Akad. Nauk SSSR* **32** (1941).

45. Onsager, L. Statistical hydrodynamics. *Nuovo Cim.* **6,** 279–287 (1949).

46. Heisenberg, W. Zur statistischen Theorie der Turbulenz. *Z. Phys.* **124,** 628–657 (1948).

47. Kraichnan, R. H. Isotropic turbulence and inertial-range structure. *Phys. Fluids* **9,** 1728–1752 (1966).

48. Kraichnan, R. H. An almost-Markovian Galilean-invariant turbulence model. *J. Fluid Mech.* **47,** 513–524 (1971).

49. Zhou, Y. Degrees of locality of energy transfer in the inertial range. *Phys. Fluids A: Fluid* **5,** 1092–1094 (1993).

50. Domaradzki, J. A., Liu, W. & Brachet, M. E. An analysis of subgrid-scale interactions in numerically simulated isotropic turbulence. *Phys. Fluids A: Fluid* **5,** 1747–1759 (1993).

51. Yeung, P. K. & Brasseur, J. G. The response of isotropic turbulence to isotropic and anisotropic forcing at the large scales. *Phys. Fluids A: Fluid* **3,** 884–897 (1991).

52. Tennekes, H. & Lumley, J. L. *A first course in turbulence* (M.I.T. Press, Cambridge, 1972).

53. Tsinober, A. *An informal conceptual introduction to turbulence* (J. Wiley, Dodrecht, 2009).

54. Carbone, M. & Bragg, A. D. Is vortex stretching the main cause of the turbulent energy cascade? *J. Fluid Mech.* **883,** R2 (2020).

55. Johnson, P. L. On the role of vorticity stretching and strain self-amplification in the turbulence energy cascade. *J. Fluid Mech.* **922** (2021).

56. Rivera, M. K. *The inverse energy cascade of two-dimensional turbulence* PhD thesis (University of Pittsburgh, 2000).

57. Nelkin, M. In what sense is turbulence an unsolved problem? *Science* **255,** 566–570 (1992).

58. Schumacher, J. *et al.* Small-scale universality in fluid turbulence. *P. Natl. Acad. Sci. USA* **111,** 10961–10965 (2014).

59. Givi, P. Model-free simulations of turbulent reactive flows. *Prog. Energ. Combust. Sci.* **15,** 1–107 (1989).

60. Brachet, M. *et al.* Small-scale structure of the Taylor–Green vortex. *J. Fluid. Mech.* **130,** 411–452 (1983).

61. Jeong, J. & Hussain, F. On the identification of a vortex. *J. Fluid. Mech.* **285,** 69–94 (1995).

62. Lubasch, M., Moinier, P. & Jaksch, D. Multigrid renormalization. *J. Comput. Phys.* **372,** 587–602 (2018).

63. Lieb, E. H. & Robinson, D. W. The finite group velocity of quantum spin systems. *Commun. Math. Phys.* **28,** 251–257 (1972).

64. Hastings, M. B. & Koma, T. Spectral gap and exponential decay of correlations. *Commun. Math. Phys.* **265,** 781–804 (2006).

65. Vidal, G. Efficient classical simulation of slightly entangled quantum computations. *Phys. Rev. Lett.* **91,** 147902 (2003).

66. Vidal, G. Class of quantum many-body states that can be efficiently simulated. *Phys. Rev. Lett.* **101,** 110501 (2008).

67. Levin, M. & Nave, C. P. Tensor renormalization group approach to two-dimensional classical lattice models. *Phys. Rev. Lett.* **99,** 120601 (2007).

68. Gu, Z. C., Levin, M. & Wen, X. G. Tensor-entanglement renormalization group approach as a unified method for symmetry breaking and topological phase transitions. *Phys. Rev. B* **78,** 205116 (2008).

69. Nishino, T. *et al.* Two-dimensional tensor product variational formulation. *Prog. Theor. Phys.* **105,** 409–417 (2001).

70. Anders, S., Briegel, H. J. & Dür, W. A variational method based on weighted graph states. *New J. Phys.* **9,** 361–361 (2007).

71. Schuch, N., Wolf, M. M., Verstraete, F. & Cirac, J. I. Simulation of quantum many-body systems with strings of operators and Monte Carlo tensor contractions. *Phys. Rev. Lett.* **100,** 040501 (2008).

72. Mezzacapo, F., Schuch, N., Boninsegni, M. & Cirac, J. I. Ground-state properties of quantum many-body systems: entangled-plaquette states and variational Monte Carlo. *New J. Phys.* **11,** 083026 (2009).

73. Shi, Y. Y., Duan, L. M. & Vidal, G. Classical simulation of quantum many-body systems with a tree tensor network. *Phys. Rev. A* **74,** 022320 (2006).

74. Verstraete, F. & Cirac, J. I. Continuous matrix product states for quantum fields. *Phys. Rev. Lett.* **104,** 190405 (2010).

75. Haegeman, J., Osborne, T. J., Verschelde, H. & Verstraete, F. Entanglement renormalization for quantum fields in real space. *Phys. Rev. Lett.* **110,** 100402 (2013).

76. Vidal, G. Classical simulation of infinite-size quantum lattice systems in one spatial dimension. *Phys. Rev. Lett.* **98,** 070201 (2007).

77. Crosswhite, G. M., Doherty, A. C. & Vidal, G. Applying matrix product operators to model systems with long-range interactions. *Phys. Rev. B* **78,** 035116 (2008).

78. Jordan, J., Orús, R., Vidal, G., Verstraete, F. & Cirac, J. I. Classical simulation of infinite-size quantum lattice systems in two spatial dimensions. *Phys. Rev. Lett.* **101,** 250602 (2008).

79. Evenbly, G. & Vidal, G. Class of highly entangled many-body states that can be efficiently simulated. *Phys. Rev. Lett.* **112,** 240502 (2014).

80. Schuch, N., Wolf, M. M., Verstraete, F. & Cirac, J. I. Computational complexity of projected entangled pair states. *Phys. Rev. Lett.* **98,** 140506 (2007).

81. Verstraete, F., Porras, D. & Cirac, J. I. Density matrix renormalization group and periodic boundary conditions: a quantum information perspective. *Phys. Rev. Lett.* **93,** 227205 (2004).

82. White, S. R. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.* **69,** 2863–2866 (1992).

83. Haegeman, J., Lubich, C., Oseledets, I., Vandereycken, B. & Verstraete, F. Unifying time evolution and optimization with matrix product states. *Phys. Rev. B* **94,** 165116 (2016).

84. Stoudenmire, E. M. & White, S. R. Real-space parallel density matrix renormalization group. *Phys. Rev. B* **87,** 155137 (2013).

85. Urbanek, M. & Soldán, P. Parallel implementation of the time-evolving block decimation algorithm for the Bose–Hubbard model. *Comput. Phys. Commun.* **199,** 170–177 (2016).

86. Secular, P. *et al.* Parallel time-dependent variational principle algorithm for matrix product states. *Phys. Rev. B* **101,** 235123 (2020).

87. Holtz, S., Rohwedder, T. & Schneider, R. On manifolds of tensors of fixed TT-rank. *Numer. Math.* **120,** 701–731 (2012).

88. Oseledets, I. Constructive representation of functions in low-rank tensor formats. *Constr. Approx.* **37,** 1–18 (2013).

89. Dolgov, S., Khoromskij, B. & Savostyanov, D. Superfast Fourier transform using QTT approximation. *J. Fourier Anal. Appl.* **18,** 915–953 (2012).

90. Khoromskij, B. $\mathcal{O}(d \log N)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling. *Constr. Approx.* **34,** 257–280 (2011).

91. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).

92. Dziarmaga, J. Dynamics of a quantum phase transition: exact solution of the quantum Ising model. *Phys. Rev. Lett.* **95,** 245701 (2005).

93. Stoudenmire, E. M. & White, S. R. Minimally entangled typical thermal state algorithms. *New J. Phys.* **12,** 055026 (2010).

94. Zwolak, M. & Vidal, G. Mixed-state dynamics in one-dimensional quantum lattice systems: a time-dependent superoperator renormalization algorithm. *Phys. Rev. Lett.* **93,** 207205 (2004).

95. Kazeev, V. A. & Khoromskij, B. N. Low-rank explicit QTT representation of the Laplace operator and its inverse. *SIAM J. Matrix Anal. Appl.* **33,** 742–758 (2012).

96. García-Ripoll, J. J. Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equations. *Quantum* **5,** 431 (2021).

97. Fornberg, B. Generation of finite difference formulas on arbitrarily spaced grids. *Math. Comput.* **51,** 699–706 (1988).

98. Fiacco, A. & McCormick, G. *Nonlinear programming: sequential unconstrained minimization techniques* (J. Wiley, New York, 1968).

99. Fiacco, A. Penalty methods for mathematical programming in E^n with general constraint sets. *J. Optim. Theor. Appl.* **6,** 252–268 (1970).

100. Zhang, T. A. On the rotation and skew-symmetric forms for incompressible flow simulations. *Appl. Numer. Math.* **7,** 27–40 (1991).

101. Shewchuk, J. *An introduction to the conjugate gradient method without the agonizing pain* tech. rep. (Carnegie Mellon University, 1994).

102. Rakhuba, M. V. & Oseledets, I. V. Fast multidimensional convolution in low-rank tensor formats via cross approximation. *SIAM J. Sci. Comput.* **37,** A565–A582 (2015).

103. Ernst, D., Hager, G., Thies, J. & Wellein, G. Performance engineering for real and complex tall & skinny matrix multiplication kernels on GPUs. *Int. J. High Perform. Comput. Appl.* **35,** 5–19 (2021).

104. Jouppi, N., Young, C., Patil, N. & Patterson, D. Motivation for and evaluation of the first tensor processing unit. *IEEE Micro* **38,** 10–19 (2018).

105. Ganahl, M., Beall, J., Hauru, M., Lewis, A. G. M., Yoo, J. H., Zou, Y. & G. Vidal. Density matrix renormalization group with tensor processing units. Preprint: arXiv:2204.05693.

106. Burgers, J. M. A mathematical model illustrating the theory of turbulence. *Adv. Appl. Mech.* **1,** 171–199 (1948).

107. Hopf, E. The partial differential equation $u_t + uu_x = \mu_{xx}$. *Commun. Pure Appl. Math.* **3,** 201–230 (1950).

108. Cole, J. On a quasi-linear parabolic equation occurring in aerodynamics. *Quart. Appl. Math.* **9,** 225–236 (1951).

109. Whitham, G. B. *Linear and Nonlinear Waves* (J. Wiley, New York, 1974).

110. Dolgov, S. V., Khoromskij, B. N. & Oseledets, I. V. Fast solution of multi-dimensional parabolic problems in the tensor train/quantized tensor train–format with initial application to the Fokker-Planck equation. *SIAM J. Sci. Comput.* **34,** A3016–A3038 (2012).

111. Herrmann, L., Schwab, C. & Zech, J. Deep neural network expression of posterior expectations in Bayesian PDE inversion. *Inverse Probl.* **36,** 125011 (2020).

112. Foias, C., Manley, O., Rosa, R. & Temam, R. *Navier-Stokes equations and turbulence* (Cambridge University Press, 2001).

113. Tagliacozzo, L., Evenbly, G. & Vidal, G. Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law. *Phys. Rev. B* **80,** 235127 (2009).

114. Murg, V., Verstraete, F., Legeza, Ö. & Noack, R. M. Simulating strongly correlated quantum systems with tree tensor networks. *Phys. Rev. B* **82,** 205105 (2010).

115. Evenbly, G. Gauge fixing, canonical forms, and optimal truncations in tensor networks with closed loops. *Phys. Rev. B* **98,** 085155 (2018).

116. Pfeifer, R. N. C., Haegeman, J. & Verstraete, F. Faster identification of optimal contraction sequences for tensor networks. *Phys. Rev. E* **90,** 033315 (2014).

117. Gourianov, N. *et al.* A quantum-inspired approach to exploit turbulence structures. *Nat. Comput. Sci.* **2,** 30–37 (2022).

118. Lubasch, M., Joo, J., Moinier, P., Kiffner, M. & Jaksch, D. Variational quantum algorithms for nonlinear problems. *Phys. Rev. A* **101,** 010301(R) (2020).

119. Taira, K. *et al.* Modal analysis of fluid flows: an overview. *AIAA J.* **55,** 4013–4041 (2017).

120. Libby, P. A. *et al. Turbulent reacting flows* (Academic Press, London, 1994).

121. Pope, S. B. PDF methods for turbulent reactive flows. *Prog. Energ. Combust.* **11,** 119–192 (1985).

122. Nouri, A. G., Nik, M. B., Givi, P., Livescu, D. & Pope, S. B. Self-contained filtered density function. *Phys. Rev. Fluids* **2,** 094603 (2017).

123. Fukagata, K. Towards quantum computing of turbulence. *Nat. Comput. Sci.* (2022).

124. Lloyd, S., De Palma, G., Gokler, C., Kiani, B., Liu, Z. W., Marvian, M., Tennie, F. & Palmer, T. Quantum algorithm for nonlinear differential equations. Preprint: arXiv:2011.06571.

125. Liu, J. P. *et al.* Efficient quantum algorithm for dissipative nonlinear differential equations. *Proc. Natl. Acad. Sci. U.S.A.* **118** (2021).

126. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103,** 150502 (2009).

127. Berry, D. W. High-order quantum algorithm for solving linear differential equations. *J. Phys. A* **47,** 105301 (2014).

128. Berry, D. W., Childs, A. M., Ostrander, A. & Wang, G. Quantum algorithm for linear differential equations with exponentially improved dependence on precision. *Commun. Math. Phys.* **356,** 1057–1081 (2017).

129. Cerezo, M. *et al.* Variational quantum algorithms. *Nat. Rev. Phys.,* 1–20 (2021).

130. Chorin, A. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. Am. Math. Soc.* **73,** 928–931 (1967).