High-Order Discontinuous Galerkin Hydrodynamics for Supersonic Astrophysical Turbulence

Miha Cernetic

München 2024

High-Order Discontinuous Galerkin Hydrodynamics for Supersonic Astrophysical Turbulence

Miha Cernetic

Dissertation
an der Fakultät für Physik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Miha Cernetic
aus Postojna, Slovenija

München, den 21.12.2023

Erstgutachter: Prof. Dr. Volker Springel
Zweitgutachter: PD Dr. Klaus Dolag
Tag der mündlichen Prüfung: 23.02.2024

To my dad, in loving memory.

Contents

List of Figures

List of Tables

Zusammenfassung

Im gesamten Universum ist Turbulenz einer der wichtigsten Prozesse, um kinetische Energie von großen Skalen auf die viskose Skala zu übertragen, wo sie Gas erhitzen, chemische Elemente durchmischen, oder die Sternbildung regulieren kann. Turbulenz ist in allen astrophysikalischen Systemen allgegenwärtig und spielt eine vielfältige Rolle, von der Regulierung der Stern- und Planetenbildung über die Flammenausbreitung in Typ Ia Supernovae bis hin zur Ausbreitung der kosmischen Strahlung und dem Auftreten von großräumigen galaktischen Winden. Die numerische Simulation von Turbulenz ist jedoch schwierig, da man einen großen dynamischen Bereich auflösen muss, von der Erzeugung auf großen Skalen bis hin zu viskosen Skalen, um den Einfluss, den Turbulenz auf das betreffende astrophysikalische System hat, richtig zu beschreiben.

Um solche Systeme zu simulieren und gleichzeitig turbulente Bewegungen ausreichend aufzulösen, werden neue Algorithmen und größere Computer benötigt. Seit den 1970er Jahren hat sich die Anzahl der Transistoren in einem integrierten Schaltkreis etwa alle zwei Jahre verdoppelt – allgemein bekannt als Moore'sches Gesetz – und dies hat lange Zeit zu einer ähnlichen Steigerung der Rechenleistung geführt. Seit Mitte der 2010er Jahre hat sich dieser Fortschritt jedoch verlangsamt, und die Computerindustrie hat eine Reihe von Ansätzen entwickelt, um die Anzahl der Gleitkommaoperationen pro Sekunde, die ihre Chips leisten können, dennoch weiter zu erhöhen. Ein wichtiger aktueller Trend in dieser Hinsicht ist die Verwendung von Grafikprozessoren (GPUs) und deren Umwidmung für den Einsatz im Hochleistungsrechnen. Dieser Ansatz wird in den ersten Exascale-Supercomputer verwendet, die in letzter Zeit auf den Markt gekommen sind. Das einfachere Chipdesign von GPUs stellt jedoch grundlegende Herausforderungen für (astrophysikalische) Codes dar, da man sie nicht einfach für diese Systeme neu kompilieren kann. Stattdessen ist eine grundlegende Neuschreibung erforderlich, die eine Änderung der Datenstrukturen, der Speicherzugriffe, der Codelogik und der Kommunikationsstrategie beinhaltet.

Motiviert sowohl durch diese Hardware-Trends als auch durch den Bedarf an effizienteren numerischen Algorithmen untersucht diese Arbeit die Verwendung von diskontinuierlichen Galerkin (dG)-Methoden für Simulationen astrophysikalischer Turbulenz auf GPUs. Die numerische Technik von dG verspricht Konvergenz hoher Ordnung bei vergleichsweise geringem Kommunikationsbedarf, was ihr möglicherweise eine überlegene Rechenleistung und Genauigkeit verleiht, insbesondere wenn sie auf moderner, mit GPUs beschleunigter Computerhardware realisiert wird.

Um diese Möglichkeit zu untersuchen, habe ich eine dG-Methode hoher Ordnung in der

GPU-nativen “Compute Unified Device Architecture” (CUDA) implementiert, kombiniert mit einer Parallelisierung über mehrere verteilte Rechnerknoten mit dem “Message Passing Interface” (MPI). Zunächst habe ich die Methode für subsonische Strömungen realisiert, wo sie eine vielversprechende exponentielle Konvergenz mit zunehmender räumlicher Ordnung aufweist. Dies ermöglicht die Simulation von subsonischen Strömungen mit sehr hoher Auflösung bei geringem Rechenaufwand. Außerdem habe ich die klassische dG-Methode mit einem rekonstruktionsbasierten Navier-Stokes-Löser kombiniert, was sich in früheren Arbeiten als schwierig erwiesen hatte. Zusätzlich habe ich eine neue Methode zur Auflösung von Schocks innerhalb von Zellen vorgeschlagen, die auf einer geeigneten Injektion von künstlicher Viskosität basiert.

Aufbauend auf diesen Schritten habe ich mich dann mit dem Fall der supersonischen Turbulenz befasst. Das Netzwerk von stark interagierenden Schocks, das in solchen Simulationen auftritt, erforderte weitere Verbesserungen des Algorithmus und des Codes. Ich habe eine vereinfachte Implementierung der künstlichen Viskosität auf der Grundlage der klassischen von Neumann-Richtmeyer-Formulierung eingeführt und das Stabilitätsproblem von dG für supersonische Strömungen gelöst, indem ich einen neuartigen Projektionsansatz für die primitiven Variablen vorschlug, so dass sie stabil auf Zelloberflächen extrapoliert werden können.

Die Entwicklungen dieser Arbeit machen somit zum ersten Mal GPU-beschleunigte diskontinuierliche Galerkin-Methoden hoher Ordnung für eine breite und im Wesentlichen uneingeschränkte Palette von Forschungsanwendungen in der Astrophysik anwendbar. Dazu gehören zum Beispiel Untersuchungen der subsonischen Turbulenz in Galaxienhaufen, die mit dG-Methoden hoher Ordnung viel präziser und kostengünstiger dargestellt werden kann als mit traditionellen Finite-Volumen-Verfahren. Wichtig ist, dass dies nun auch Systeme einschließt, die starke Schocks enthalten oder sogar von Netzwerken von Schocks beherrscht werden, wie z.B. Regionen mit supersonischer Turbulenz, ein Regime, das bisher nicht effektiv mit dG-Hydrodynamik hoher Ordnung behandelt werden konnte.

Abstract

Throughout the universe a primary way of transferring large scale kinetic energy down to viscous scales where it can heat gas, mix chemical elemental abundances, regulate star formation, etc., is turbulence. Turbulence is ubiquitous across astrophysical objects, from regulating star and planet formation, to being the main driver of flame propagation in Type Ia supernovae, to controlling the propagation of cosmic rays and launching outflows and winds in galaxies. However, turbulence is difficult to simulate as one needs to resolve a wide dynamic range, from the stirring at large scales down to viscous scales, to properly account for the influence it has on the astrophysical system in question.

To be able to simulate such systems while sufficiently resolving turbulent motions new algorithms and bigger computers are needed. Since the 1970s the number of transistors in an integrated circuit has doubled about every two years – commonly known as Moore's law – and this has given rise to a similar expansion of their computational performance. Since the mid 2010s, however, this progress has slowed down, and the computer industry has turned to a variety of approaches for still increasing the number of floating point operations per second their chips can deliver. One important current trend along this line is to use graphical processing units (GPUs) and re-purpose them for use in high performance computing. In fact, this approach is powering the first exascale supercomputers that have emerged recently. The simpler chip design of GPUs poses however fundamental challenges to (astrophysical) codes as one cannot simply recompile them for these systems. Instead, a fundamental rewrite is necessary, including a change of data structures, memory accesses, code logic and communication strategy.

Motivated both by these hardware trends and by the need for more efficient numerical algorithms, this thesis investigates the use of Discontinuous Galerkin (DG) methods for simulations of astrophysical turbulence on GPUs. The numerical technique of DG promises high-order convergence with comparatively small communication needs, potentially giving it superior computational performance and accuracy, particularly when realized on modern computer hardware accelerated with GPUs.

To investigate this prospect, I have implemented a high-order DG method in the GPU-native Compute Unified Device Architecture (CUDA), combined with a parallelization across multiple distributed memory nodes with the Message Passing Interface (MPI). First, I realized the method for subsonic flows where it exhibits a very promising exponential convergence with increasing spatial order. This allows for subsonic flows to be simulated at very high resolution at a low computational cost. I furthermore improved the classic

DG method with a reconstruction-based Navier-Stokes solver, something that had proven difficult in previous work. I also proposed a new sub-cell shock resolving method based on an appropriate injection of artificial viscosity.

Building on these steps, I have then addressed the case of supersonic turbulence. The network of strong interacting shocks appearing in such simulations required further algorithm and code improvements. I introduced a simplified artificial viscosity implementation based on the classic von Neumann-Richtmeyer formulation and solved the stability problem of DG for supersonic flows by proposing a novel projection approach for the primitive variables so that they can be stably extrapolated to cell surfaces.

The developments of this thesis thus make high-order, GPU-accelerated Discontinuous Galerkin methods for the first time applicable to a wide und essentially unrestricted range of research applications in astrophysics. This includes, for example, studies of the subsonic turbulence in galaxy clusters which can be represented much more precisely and in a more cost effective way with high-order DG methods than with traditional finite volume techniques. Importantly, this now also includes systems that contain strong shocks, or are even governed by networks of shocks such as regions filled with supersonic turbulence, a regime that could previously not be treated effectively with high-order DG hydrodynamics.

Chapter 1

Introduction

On all scales, the dynamics of astrophysical fluid flows is shaped by turbulent motions. These complex and chaotic motions arise due to the nonlinear nature of the gas flow, described by the Navier-Stokes equations, which capture the interplay of momenta, pressure, and viscosity. Turbulence is known for its ability to transfer large scale energy down to small scales through the energy cascade. Big whirls akin to Kelvin-Helmholtz instabilities give rise to ever smaller whirls until the viscous scale is reached. There these whirls get converted into heat. This energy transfer is crucial for understanding many astrophysical processes. It also makes such an understanding difficult from the computational point of view as it is necessary to resolve an immense range of scales for proper treatment of turbulence. Whether it be in the context of star formation, the interstellar-, circumgalactic- or intracluster-medium, planet formation, supernovae, atmospheric turbulence, river currents, or the tumultuous flows within stars, turbulent motions and their resulting gas dynamics govern these systems. Understanding turbulence is essential for comprehending the behavior of astrophysical systems on scales ranging from the microscopic to the cosmic. It is crucial for the shaping of structures throughout the universe and our understanding of them.

1.1 Turbulent systems in the Universe

1.1.1 Star formation

Star formation is a complex process occurring across multiple scales of the interstellar medium. The final stages take place in the densest clouds consisting of dense $n > 10^2\,\mathrm{cm}^{-3}$ and cold $T \sim 10 - 30\,\mathrm{K}$ gas (Larson, 1981). The complex and nonlinear interplay between turbulence, gravity, pressure and magnetic fields takes place at scales from thousands of astronomical units to tens of parsecs within these cold clouds in star-forming regions and plays a crucial role in determining the distribution of masses of the pre-stellar cores (e.g. Chabrier, 2003; Mac Low & Klessen, 2004; McKee & Ostriker, 2007).

1.1.2 Stellar evolution

Resolving the temporal evolution of a single star entails taking into account three very different timescales, a dynamical timescale — from seconds to days, in extreme cases years, the thermal timescale — 10^7 years, and the nuclear timescale — 10^{10} for the Sun. These processes are strongly coupled and all need to be resolved in tandem. Computationally this is only possible in 1D. For many applications in stellar evolution studies 1D models are sufficient and the cheap computational cost is also attractive because it allows for easy parameter space exploration of many key processes of interest. Such processes include but are not limited to fusion in the core, the formation of heavier elements, the intricate balance between gravity and pressure, stellar wind ejection and the study of both radiative transport in the radiative zone and convection in the star's convective zones (e.g. Paxton et al., 2011; Jermyn et al., 2023). And it is the transition between the two zones that has proven very challenging for 1D models to describe (e.g. Paxton et al., 2019). The so called boundary layer can be clearly seen on the Kippenhahn diagram of a 1D simulation by Steindl et al. (2022) shown in Fig. 1.1 and as a slice through a full 3D box in a simulation by Herwig et al. (2023) in Fig. 1.2 as the red circle segment. The advection layer is turbulent by nature and therefore requires a proper 3D treatment. While models calibrated with observations exist to sufficiently describe the convection zone itself the boundary layer between it and the radiative zone cannot be described with a simple empirically calibrated model. Full 3D simulations of stellar evolution exist but are currently limited to simulating the star for a few days which is far from the lifespan of even the shortest living stars (e.g. Rizzuti et al., 2023; Herwig et al., 2023). Further large scale 3D studies of turbulence are therefore crucial for advancing our understanding of stellar evolution.

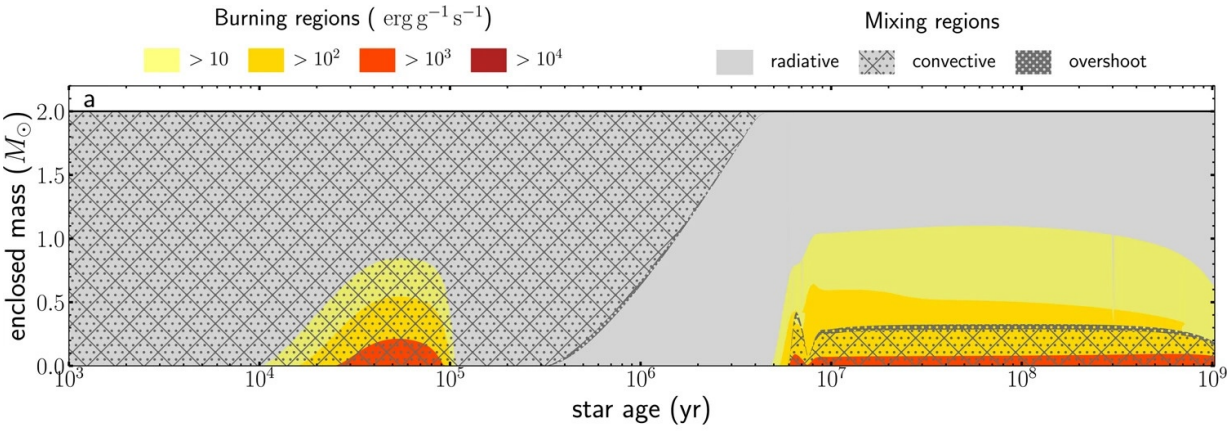


Figure 1.1: Kippenhahn diagram (adopted from Steindl et al., 2022) showing the evolution of the internal structure of a 2 $M_\odot$ star. The boundary layer is denoted as the "overshoot" on the top right legend.

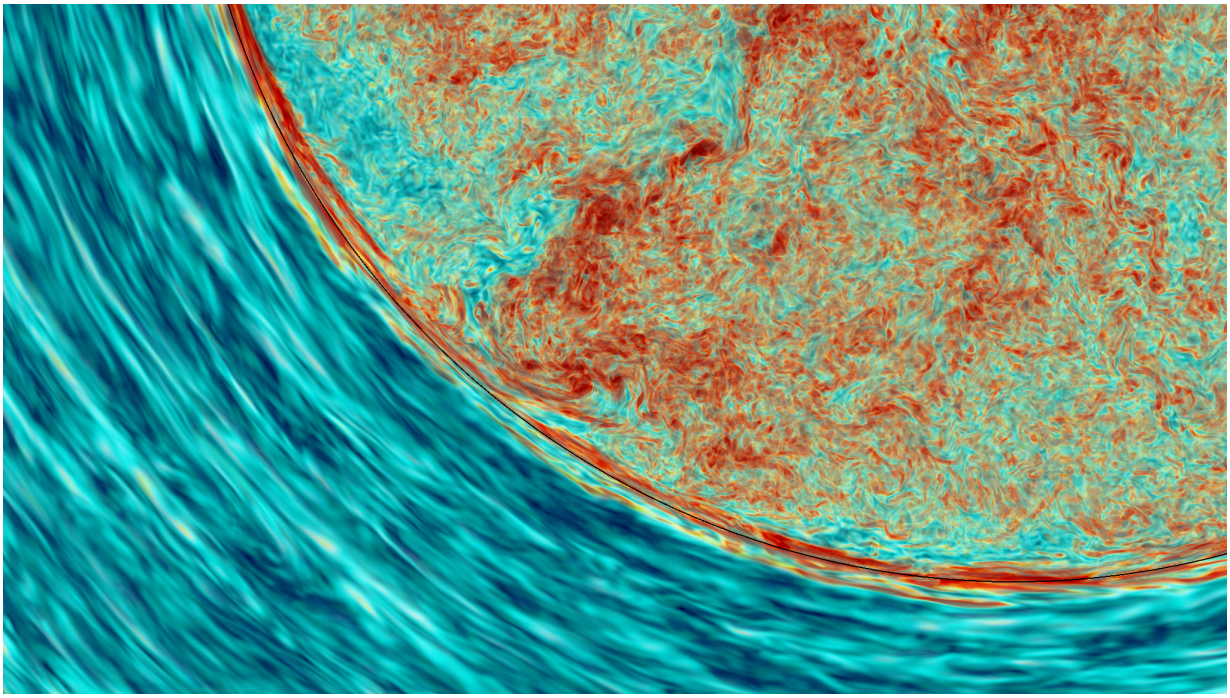


Figure 1.2: Vorticity magnitude in the convective boundary region of a 3D stellar evolution simulation (adopted from Herwig et al., 2023).

1.1.3 Intracluster medium

Galaxy clusters are the largest gravitationally bound objects with virial masses and radii

$$M_{\rm vir} \sim 10^{14-15}\,{\rm M}_\odot$$

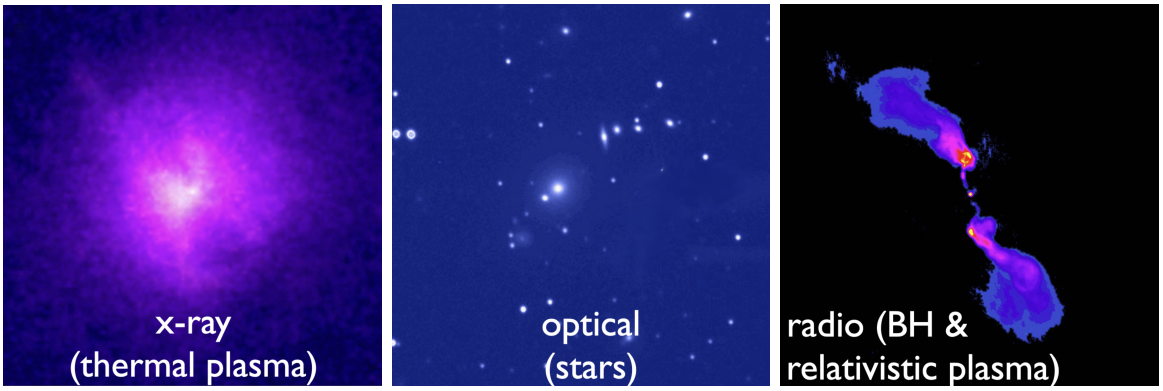
$$R_{\rm vir} \sim 1-3\,{\rm Mpc}.$$

They host the most massive galaxies and black holes, making them of utmost importance in studying galaxy formation and hierarchical clustering on the largest of scales. They are composed mainly of Dark Matter which makes up about 84% of their mass, followed by hot plasma at 14% and stars at about 2% (see a review by Kravtsov & Borgani, 2012). The three components of the Hydra cluster are shown in Fig. 1.3. The relatively high fraction of gas in these systems is curious and has been studied extensively. Piffaretti et al. (2005) found that galaxy clusters have densities $n \sim 10^{-4}$ cm^{-3} with temperatures $T \sim 10^8$ K. This conditions result in a large electron mean free path, which makes thermal conduction an important contributor to gas dynamics in these systems

$$\ell_e \simeq 2\left(\frac{T}{3\ {\rm keV}}\right)^2\left(\frac{n}{0.01\ {\rm cm}^{-3}}\right)^{-1}{\rm kpc}.$$

About half of galaxy clusters have cooling times below 1 Gyr, while exhibiting lower than expected star formation and cooling rates (Rawle et al., 2012). Such clusters should be actively forming stars but something is suppressing star formation. One possible explanation

is the presence of constant subsonic turbulence in the galaxy cluster plasmas, which provides pressure support go the intracluster gas, preventing its collapse (Dennis & Chandran, 2005). To study this turbulent heating heating it is necessary to resolve all processes that determine the physical viscous scale in galaxy cluster atmospheres. The dilute, weakly magnetized plasmas of the ICM are buoyantly unstable and very susceptible to instabilities caused by large temperature gradients, like heat-flux-driven buoyancy instability (HBI; Quataert, 2008) and magnetothermal instability (MTI; Balbus, 2000, 2001). Perrone & Latter (2022) have show the potential of HBI and MTI in driving turbulence in galaxy clusters but future higher resolution numerical studies with physical viscosity and conduction are needed to determine the viscous scale in galaxy clusters and unravel all sources of heating.



(a) (b) (c)

Figure 1.3: A view of the Hydra cluster in x-ray, optical and radio. Credits: X-ray: NASA/CXC/SAO; Optical: Instituto de Astrofísica de Canarias; Radio: Greg Taylor (NRAO).

1.1.4 Supernovae

Turbulence not only plays a crucial role in triggering Type Ia supernova (e.g. Reinecke et al., 2002; Pan et al., 2008; Schmidt et al., 2010), the largest explosions in the universe also drive turbulence on galactic scales and with sufficient ferocity to change the host galaxies' star formation rate within a matter of Myrs (e.g. Herbst & Assousa, 1979; Nagakura et al., 2009; Bluck et al., 2019).

Triggering Type Ia

Binary systems consisting of a white dwarf and a companion star can result in type Ia supernovae, which are famous standard candles in our Universe, crucial for accurately determining distances (Riess et al., 1998; Perlmutter et al., 1999). They are reliable indicators at large distances, even exceeding 1000 Mpc. They happen when a slowly rotating carbon-oxygen white dwarf accretes mass from the companion star. As it gains mass the growing gravitational force is countered by the increase of temperature and pressure in

the dwarf's core. At some point during this process carbon fusion generates a deflagration flame front which is dramatically accelerated by its interaction in turbulence. In a matter of seconds a substantial fraction of C and O undergoes fusion to heavier elements and in the process enough energy is generated to gravitationally unbound the star. The mechanisms of turbulent propagation of the deflagration flame front are not fully understood and are crucial for improving our understanding of Type Ia supernova (e.g. Hillebrandt & Niemeyer, 2000; Schmidt et al., 2010). A deflagration driven explosion is shown in Fig. 1.4.

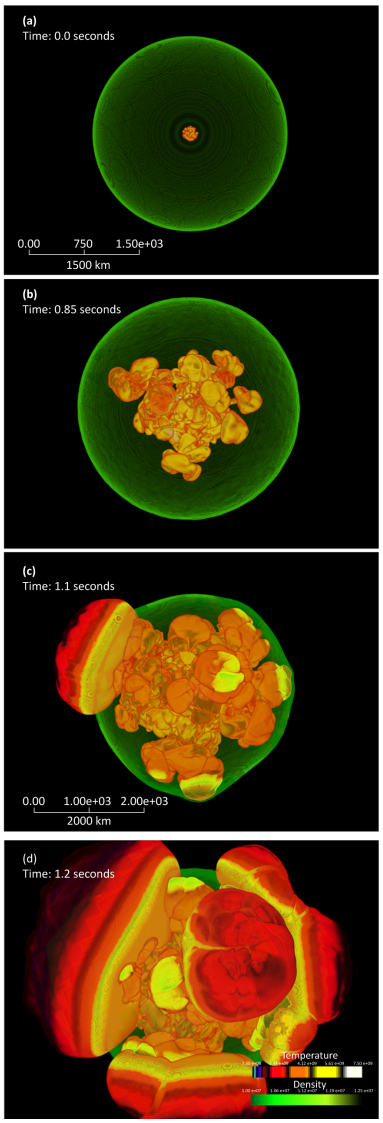


Figure 1.4: Four snapshots during a simulation of the explosion phase of the deflagration-to-detonation model of nuclear-powered Type Ia supernovae. The images show extremely hot matter (ash or unburned fuel) and the surface of the star (green). Ignition of the nuclear flame was assumed to occur simultaneously at 63 points randomly distributed inside a 128-km sphere at the center of the white dwarf star. Image: Argonne National Laboratory.

Stirring supernovae

When a star explodes as a supernova, it deposits a tremendous amount of energy into its surroundings. This energy can disperse the dense molecular clouds of gas within the galaxy, which are the primary sites for star formation. By dispersing these clouds, the supernova can effectively reduce the amount of available material for new star formation, thereby influencing the overall SFR of the host galaxy. Despite this, supernovae can also act as catalysts for new star formation. The large scale shock waves quickly break down into a network of smaller shocks as shown in Fig. 1.5. They will further decay down to smaller and smaller eddies and eventually turbulently still a large volume around the supernova. The density of this volume will adopt a lognormal distribution where the densest tail of the distribution can end up forming stars by triggering cold dense clouds to collapse under their own gravity and form new stars. This process is known as "triggered" or "induced" star formation.

All types of supernovae, not just the Type Ia described above, have a profound effect on the interstellar medium of the host galaxy and can change the galaxy's star formation in the long run. The sudden release of energy during a supernova creates powerful shock waves that propagate through the surrounding interstellar medium. These shock waves violently stir the surrounding gas giving rise to turbulent motions. This turbulence in turn determines the future evolution of the interstellar medium, influencing star formation in the galaxy, affecting the recycling of baryons, etc. In this way, supernovas act as cosmic drivers of turbulence, influencing the nature of the interstellar medium on both local and galactic scales (e.g. Herbst & Assousa, 1979; Nagakura et al., 2009; Bluck et al., 2019).

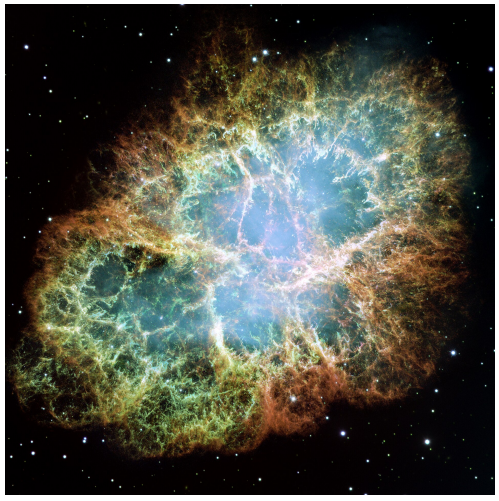


Figure 1.5: Crab nebula imaged by the Hubble telescope. The image consists of 24 individual Wide Field and Planetary Camera 2 exposures taken in October 1999, January 2000, and December 2000. The colors in the image indicate the different elements that were expelled during the explosion. Blue in the filaments in the outer part of the nebula represents neutral oxygen, green is singly-ionized sulfur, and red indicates doubly-ionized oxygen. NASA, ESA, J. Hester and A. Loll (Arizona State University)

1.1.5 AGN

Active galactic nuclei are compact galaxy centers emitting a large amount of energy across the electromagnetic spectrum, from radio and microwave through optical all the way to gamma ray frequencies. In the past we had to rely on spectral analysis which indicated the source of this energy emission does not come from stars but rather from matter infalling on a supermassive black hole (SMBH) at the center of the host galaxy (Gebhardt et al., 2000; Häring & Rix, 2004; McConnell & Ma, 2013; Kormendy & Ho, 2013). But recently we got the first visual proof of the existence of a black hole at the centre of the M87 galaxy (Event Horizon Telescope Collaboration et al., 2019). During a typical lifetime a SMBH will release an amount of energy on the order of hundreds of binding energies of its host galaxy (e.g. Fabian, 2012; King & Pounds, 2015). It has been shown that super-Eddington accretion onto the central black hole is crucial for sustaining large luminosities observed from AGNs. To drive accretion one needs some kind of viscosity to efficiently transfer angular momentum from the inner part of the accretion disk outwards (Armitage, 1998). In highly ionized AGN disks the source of viscosity is the magnetorotational instability (MRI) driven turbulence. In this case turbulence mixes material locally at different radii and as such acts as an effective source of viscosity. In this way a sustained high rate of accretion on the central black hole can be achieved. AGNs accelerate ejecta in two colimated jets emanating along the normal of the galactic disk (see Fig. 1.6). Recently Medina-Torrejón et al. (2023) have demonstrated that magnetic reconnection, driven by turbulence, can accelerate particles to extreme energies in magnetically dominated flows, reaching significant fractions of the speed of light. This kind of jets deposits a large amount of kinetic energy at large scales, which significantly influences the future evolution of the host galaxy (e.g. Costa et al., 2020, 2022). Here again the large scale energy is transferred down the energy cascade with the help of turbulence.

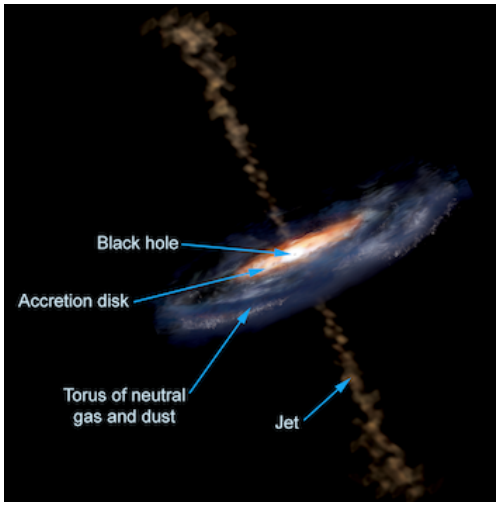


Figure 1.6: This illustration shows the different features of an active galactic nucleus (AGN). The extreme luminosity of an AGN is powered by accretion onto a supermassive black hole. Some AGN have jets, while others do not. (Credit: Aurore Simonnet, Sonoma State University)

1.1.6 Cosmic rays

Cosmic rays propagate through the interstellar medium by scattering off Alfvén waves (MHD waves). This scattering process can isotropize the cosmic rays and reduce their streaming velocity to the Alfvén wave velocity. However, the presence of turbulence can damp these Alfvén waves which in turn suppresses the propagation of cosmic rays (e.g. Lazarian, 2016; Lazarian & Xu, 2022).

1.1.7 Outlook on turbulence simulations

Many of the astrophysical systems described above cannot be understood properly without better resolving and understanding the turbulence present within them. Doing so requires larger simulations with more degrees of freedom, and ideally also with more accurate numerical techniques. Why the usual approach of just buying a larger supercomputer broke down in just the last few years will be discussed in the next section. Later, we will also address how new numerical techniques can improve the faithfulness with which simulations track turbulence for a given number of computational cells.

1.2 GPU computing

Moore's Law states that the number of transistors on an integrated circuit doubles approximately every two years. A recent history of the law is shown in Fig. 1.7. Until about the mid-2010s, astrophysics codes could reliably double their simulation sizes in step with improving hardware. It might be necessary to use a hardware-specific compiler, but the overall logic of the code would remain the same. The last decade has seen a market shift in hardware. Instead of few ten very high performance cores modern supercomputers use GPUs which can run thousands of concurrent threads, albeit at lower frequency. This trend is clearly shown by the green and black symbols in Fig. 1.7. The same trend can be seen in the TOP500 list published twice a year by the TOP500 project. It ranks and details the 500 most powerful non-distributed computer systems in the world. The evolution of the performance share of pure CPU machines compared to supercomputers with accelerators is shown in Fig. 1.8. By November 2023, only 30% of the total performance of the world's 500 fastest supercomputers will come from CPU-only machines. In addition, all current and future exascale supercomputers use accelerators. It is clear that in order to harness current and especially future supercomputers astrophysical codes must support GPUs. Unfortunately, a simple change of compiler is not feasible in this case. The architectural differences between CPUs and GPUs require a thorough code rewrite to change not only the execution logic but also the memory layout of the code, and sometimes even the underlying core algorithm.

The fundamental differences between CPUs and GPUs stem from their origins. GPUs were originally developed for rendering graphics, and as such they excel in tasks such as shading, texturing, and rendering independent polygons that make up 3D objects. On the other hand, CPUs are designed for handling any general-purpose program, regardless

of whether it involves intensive number crunching or not. Because of this, GPUs ended up having a higher number of processing units, greater aggregate memory bandwidth but higher latency and slower, simpler individual cores. CPUs have much higher clock speeds, more advanced instruction processing and more cache per core. The hardware architecture disparities between CPUs and GPUs are depicted in Fig 1.9. The transistor counts associated with various functions are abstractly represented by the relative sizes of different shaded areas. The figure provides a visual representation of the diverse components of a computer system. Computation is represented in green, instruction processing in gold, L1 cache in purple, higher-level cache in blue, and memory (DRAM) in orange, with the latter being notably larger than the caches.

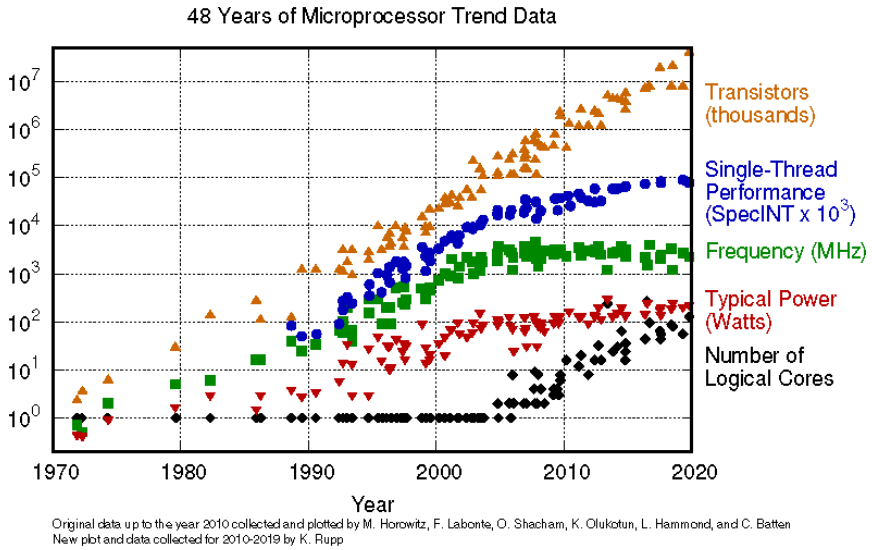


Figure 1.7: Moore's Law Timeline, including Moore's Bend with Transistors/CPU Inflected with Multi-Core CPUs beginning in 2005. The number of transistors is shown with orange triangles, the single thread performance is shown by blue circles, the frequency by red upside down triangles and the number of logical cores per integrated circuit is indicated by black diamonds.

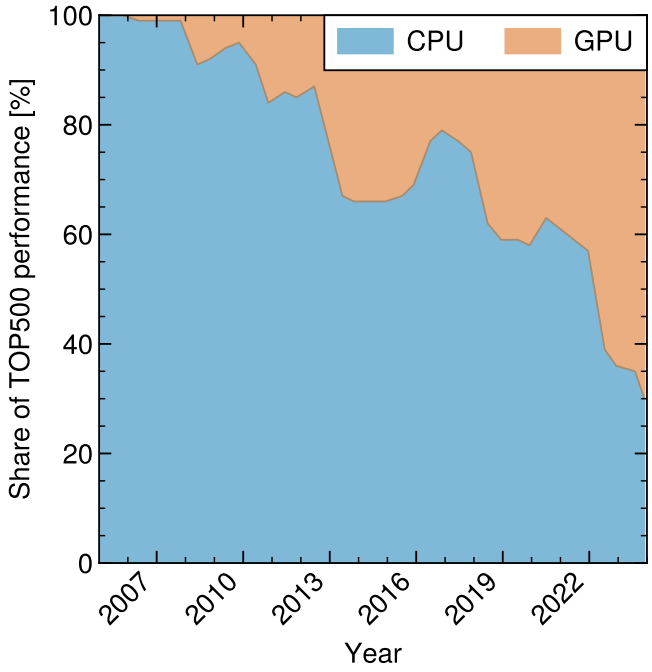


Figure 1.8: Evolution of the performance share in the TOP500 list from November 2004 until November 2023.

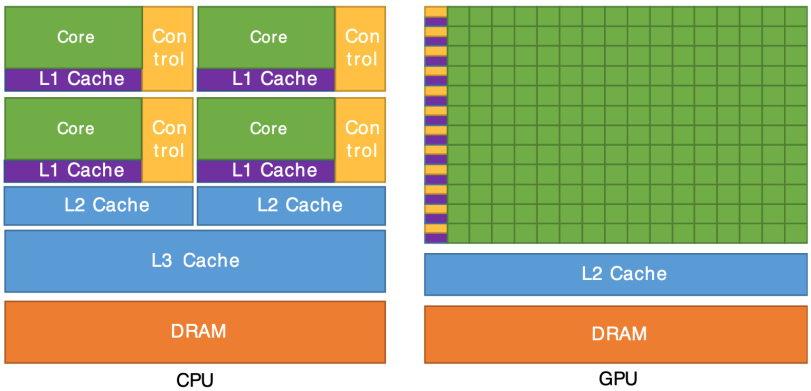


Figure 1.9: An illustration of the components of an example CPU on the left and GPU on the right. The relative allocation of transistors to different functions is represented by the relative sizes of different shaded areas. Computation is shown in green, instruction processing in gold, L1 cache in purple, higher level cache in blue and memory (DRAM) in orange. Figure obtained from NVIDIACorporation (2021).

1.3 Numerical fluid dynamics

In this part, we will discuss the equations that describe the (turbulent) flow of gas using the fluid approximation. This approximation assumes that the gas is collisional, meaning that the velocity distribution of the gas particles is Maxwellian. For a sufficient number of collisions to occur, the mean free path and the time between collisions must be significantly smaller than the length and time scales of interest. In our case, these scales are determined by the periodic simulation box in which we drive our turbulence, and the time at which we sample the turbulence in the box. Both of these scales are large enough to allow the fluid approximation to hold.

The fluid approximation enables us to define a density ρ, velocity $\boldsymbol{v}$, and pressure P at any point $\boldsymbol{x}$ in space. While it is possible to have multiple fluid species with individual densities and velocities that interact with each other, in this work, we will focus on the single-fluid case. This means that fluid quantities are represented in terms of, for example, the total density ρ and the velocity at the center of mass of the resolution elements. If we ignore viscous effects, the evolution of these quantities is dictated by the Euler equations.

We do not consider the evolution of magnetic nor radiation fields in this thesis. In the final chapter we allow for a simplified cooling description.

Quantities

Before we turn to the equations let us define the quantities of interest.

Name	Symbol	Unit (SI)
Gas density	ρ	$\mathrm{kg/m^3}$
Particle number density	N	$1/\mathrm{m^3}$
Velocity	$\vec{v}$	m/s
Temperature	T	K
Sound speed	C_s	m/s
Isothermal sound speed	c_s	m/s
Pressure	P	$\mathrm{N/m^2}$
Internal energy density	E	$\mathrm{J/m^3}$
Internal specific energy	e	J/kg
Internal specific enthalpy	h	J/kg
Total specific energy	e_{tot}	J/kg
Total specific enthalpy	h_{tot}	J/kg

(1.1)

Among these variables only five of them are independent

$$\rho, \qquad \vec{v}, \qquad e \tag{1.2}$$

All other variables in Table 1.1 can be expressed from our five chosen variables. The number density is connected to density via the mean weight of gas particles μ

$$\rho = N\mu, \tag{1.3}$$

Similarly the internal energy density is connected to the specific internal energy

$$E = \rho e \tag{1.4}$$

with the total specific energy being the sum of the internal energy and the kinetic energy:

$$e_{\text{tot}} = e + \frac{1}{2}|\vec{v}|^2 \tag{1.5}$$

Similarly with specific enthalpy we first define it as

$$h = e + \frac{pV}{\rho} \tag{1.6}$$

and the total specific enthalpy

$$h_{\text{tot}} = h + \frac{1}{2}|\vec{v}|^2 \tag{1.7}$$

To connect energy to pressure we need the so called equation of state. In this work we only concern ourselves with ideal gases.

1.3.1 Ideal gas

An ideal gas is a simplification of real gas dynamics that allows us to treat gas particles as point particles moving randomly. The only interparticle interactions are perfectly elastic point-like collisions. Such a gas obeys the ideal gas law

$$pv = nRT \tag{1.8}$$

and the resulting simplified equation of state

$$p = (\gamma - 1)\,\rho e, \tag{1.9}$$

where γ the adiabatic index of the gas. It tells us about the number of degrees of freedom of each gas particle f

$$\gamma = 1 + \frac{2}{f} \tag{1.10}$$

Monoatomic gas has three translational degrees of freedom, one per spatial dimension and a resulting $\gamma = 5/3 = 1.67$. Diatomic gas has two additional rotational degrees of freedom for a $\gamma = 1.4$. The adiabatic index is also known as the heat capacity ratio because we can also define it as the ratio of the heat capacity at constant pressure to heat capacity at constant volume:

$$\gamma = \frac{c_p}{c_v} \tag{1.11}$$

In adiabatic processes of ideal gases, γ relates pressure and density as

$$p = K\rho^\gamma \tag{1.12}$$

with K being a constant related to entropy. K stays constant in the absence of viscosity, shocks, cooling and heating. We can now also derive the adiabatic sound speed of ideal gas as:

$$C_s^2 = \frac{\partial P}{\partial \rho} = \gamma \frac{P}{\rho} = \gamma(\gamma - 1)e. \tag{1.13}$$

1.3.2 Euler equations

To describe the time evolution of gas flows we turn to conservation laws. Namely the conservation of mass, momentum and energy. The laws can be expressed as a system of partial differential equations (PDEs) or in the form of integral equations.

Mass conservation

An arbitrary volume V is delineated from the rest of the space by its surface $\partial V = S$ with a normal unit vector $\vec{n}$ at each and differential surface element as dS. Absent any sources or sinks any change of mass in this volume only happens through mass flux through the surface S.

$$\frac{\partial}{\partial t}\int \rho \; dV + \int_{\partial V} \rho\vec{v}\cdot\vec{n} \; dS = 0 \tag{1.14}$$

Applying the divergence theorem to the second term of the equation above converts the surface integral of the flux to the volume integral of the divergence

$$\frac{\partial}{\partial t}\int \rho \; dV + \int_{V} \nabla\cdot(\rho\vec{u}) \; dV = 0 \tag{1.15}$$

The above equation holds true for any choice of dV, therefore we have arrived at the PDE

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\vec{v}) = 0 \tag{1.16}$$

this is also known as the continuity equation.

Momentum conservation

The conservation of momentum density $\rho\vec{v}$ can be derived in a similar way to the conservation of mass with a slight complication due to the fact that we have to take into account all forces that act on the surface by the gas surrounding the volume. We start the same way as before, except here we replace ρ with $\rho\vec{v}$

$$\frac{\partial}{\partial t}\int \rho\vec{v} \; dV + \int_{\partial V} \rho\vec{v}\vec{v}\cdot\vec{n} \; dS = 0 \tag{1.17}$$

Except this is not the full story, as we need to account for external forces from other gas bordering our dV. Force is simply $F = pS$ so we get another term.

$$\frac{\partial}{\partial t}\int \rho\vec{v} \; dV + \int_{\partial V} \rho\vec{v}\vec{v}\cdot\vec{n} \; dS + \int_{\partial V} p\vec{n} \; dS = 0 \tag{1.18}$$

By introducing the unit matrix $\mathbf{I}$ and plugging it in the third term, rewriting it as $p\mathbf{I}\cdot\vec{n}$ allows us to apply the divergence theorem.

$$\frac{\partial}{\partial t}\int \rho\vec{v}\ dV + \int_V \nabla\cdot(\rho\vec{v}\vec{v} + \mathbf{I}p)\ dV = 0 \tag{1.19}$$

because this can be applied to any dV we can write the PDE for momentum conservation

$$\frac{\partial \rho\vec{v}}{\partial t} + \nabla\cdot(\rho\vec{v}\vec{v}) + \nabla p = 0 \tag{1.20}$$

Energy conservation

The total specific energy of a gas parcel e_{tot} has two constituent parts. The internal specific energy which is related the the gas temperature and the specific kinetic energy which depends on the velocity of the parcel. Like with the derivation of momentum conservation we need to account for the influences from surrounding gas on our parcel dV.

$$\frac{\partial}{\partial t}\int \rho\left(e + \frac{v^2}{2}\right)\ dV + \int_{\partial V} \rho\left(e + \frac{v^2}{2}\right)\vec{v}\cdot\vec{n}\ dS + \int_{\partial V} p\vec{v}\cdot\vec{n}\ dS = 0 \tag{1.21}$$

Applying the divergence theorem and merging the second and third terms

$$\frac{\partial}{\partial t}\int \rho\left(e + \frac{v^2}{2}\right)\ dV + \int_{\partial V} \nabla\cdot\left\{\left(\rho e + \frac{1}{2}\rho\vec{v}^2 + p\right)\vec{v}\right\} = 0 \tag{1.22}$$

The resulting PDE reads

$$\frac{\partial \rho e_{\text{tot}}}{\partial t} + \nabla\cdot\left\{\left(\rho e_{\text{tot}} + p\right)\vec{v}\right\} = 0. \tag{1.23}$$

Having derived this third and final conservation law we now have the necessary mathematical framework to study the inviscid flow of gas in the universe. To study viscous flows one needs to use the Navier-Stokes equation.

1.3.3 Navier-Stokes equations

The Navier-Stokes equations listed below are a set of parabolic PDEs which describe the flow of viscous fluids. They are thought to govern the motion of all fluids in every context, from the cosmological and galactic, through design of planes, water turbines to weather forecasting. Despite their vast importance and even wider applicability it is currently unknown whether for a given set of initial conditions there always exists a smooth solution for the Navier-Stokes equations. Since May 2020 the Clay Mathematics Institute identified this *Navier–Stokes existence and smoothness* problem as one of the Millennium Prize Problems whose solution carries a $1M prize. We do not concern ourselves with this and define the Navier-Stokes equation as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla\cdot\boldsymbol{F} = \nabla\cdot\boldsymbol{F}_{\text{NS}}, \tag{1.24}$$

with $\boldsymbol{F}_{\mathrm{NS}}$ being the Navier-Stokes flux vector. It is a non-linear function both of the state vector $\boldsymbol{u}$ and its gradient $\boldsymbol{\nabla u}$. In the canonical form

$$\boldsymbol{F}_{\mathrm{NS}} = \begin{pmatrix} 0 \\ \boldsymbol{\Pi} \\ \boldsymbol{v} \cdot \boldsymbol{\Pi} + \chi(\gamma - 1)\rho \nabla u \end{pmatrix}, \tag{1.25}$$

with a viscous tensor

$$\boldsymbol{\Pi} = \nu\rho \left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^{\mathrm{T}} + \frac{2}{3} \nabla \cdot \boldsymbol{v} \right). \tag{1.26}$$

The presence of ∇ which dissipates shear motions with viscosity ν and heat conduction with thermal diffusivity χ fluids now generate entropy as they flow and not only in the presence of shows which we discuss in the next section.

1.3.4 Shocks and other discontinuities

Since a large fraction of this thesis deals with treating shocks we will briefly introduce these intriguing phenomena here.

As alluded by Eq. (1.12) a fluid parcel's entropy normally stays constant. In the presence of shocks this assumption breaks down even for completely inviscid flows. Intuitively one can think of shock waves like extremely steep sound waves. They can form from strong sound waves because the sound speed is proportional to $\sqrt{T}$, and as a consequence the warmer top of the wave will start "catching up" with the cooler preceding valley and steepen. Eventually the sound wave will become steep enough to form a discontinuity.

Assuming a 1D shock moving through a medium we put ourselves in the reference frame of the shock and define the pre- and post-shock regions of gas as the region which has not passed through the shock and the region which has, respectively. In our example, a gas particle will only ever go from the pre-shock to the post-shock region, never the other way around. The pre-shock region has a density ρ_1 and the post-shock ρ_2. The two regions are separated by the shock front and their properties connected by the so called shock jump conditions, also known as the Rankine-Hugoniot conditions. We will derive them below.

We start with the equations of mass flux, momentum flux, and energy flux conservation across the shock front:

$$\rho_1 v_1 = \rho_2 v_2 \tag{1.27}$$

$$p_1 + \rho_1 v_1^2 = p_2 + \rho_2 v_2^2 \tag{1.28}$$

$$\frac{1}{2} v_1^2 + \frac{\gamma}{\gamma - 1} \frac{p_1}{\rho_1} = \frac{1}{2} v_2^2 + \frac{\gamma}{\gamma - 1} \frac{p_2}{\rho_2}. \tag{1.29}$$

Next we define auxiliary variables representing specific volumes as $V_i = \rho_i^{-1}$ and the associated mass flux $j = \rho_1 v_1 = \rho_2 v_2$. Substituting them into the equations above yields

$$\begin{aligned} v_1 &= jV_1 \qquad v_2 = jV_2 \\ p_1 + j^2 V_1 &= p_2 + j^2 V_2 \end{aligned} \tag{1.30}$$

Combining these two together results in

$$j^2 = (p_1 - p_2) / (V_2 - V_1) . \tag{1.31}$$

This quadratic equation has two solutions. One where $p_1 > p_2$ and another where $p_1 < p_2$. The former solution is unphysical and will turn into a rarefraction wave. So we focus on the latter one where the post-shock region has the higher pressure.

Substituting $v_2 - v_1 = j\,(v_2 - v_1)$ into the equation above and only considering the positive root because of our decision to only focus on the shock solution gives us

$$v_1 - v_2 = \sqrt{(p_2 - p_1)\,(V_1 - V_2)} \tag{1.32}$$

We have arrived at the relation between the pre- and post-shock gas velocities. Now we turn our attention to how the energy changes across the jump.

First we take Eq. (1.29) and rewrite it using the total specific enthalpy $h_{\rm tot}$ as

$$\rho_1 h_{\rm tot} v_1 = \rho_2 h_{\rm tot} v_2. \tag{1.33}$$

Using the definition of $h_{\rm tot}$ gives

$$\rho_1 \left(h_1 + \frac{1}{2} v_1^2 \right) v_1 = \rho_2 \left(h_2 + \frac{1}{2} v_2^2 \right) v_2. \tag{1.34}$$

Using the conservation of mass from Eq. (1.27) yields

$$h_1 + \frac{1}{2} v_1^2 = h_2 + \frac{1}{2} v_2^2, \tag{1.35}$$

and rearranging h_1 and h_2 on one side and again using the relation $v_2 - v_1 = j\,(V_2 - V_1)$ we get

$$h_2 - h_1 = \frac{j^2}{2} \left(V_1^2 - V_2^2 \right), \tag{1.36}$$

which when substituting in Eq. (1.31) and using the definition of enthalpy from Eq. (1.6) results in

$$e_2 - e_1 = \frac{1}{2} (V_1 - V_2)\,(p_2 - p_1) . \tag{1.37}$$

We have arrived at the relation between the pre- and post-shock energy of the gas.

In their seminal book on fluid dynamics, Landau & Lifshitz (1959) derive the jump conditions for polytropic gases as

$$\begin{aligned} \frac{\rho_2}{\rho_1} &= \frac{u_1}{u_2} = \frac{(\gamma+1) M_1^2}{(\gamma-1) M_1^2 + 2} \\ \frac{P_2}{P_1} &= \frac{2\gamma M_1^2}{\gamma+1} - \frac{\gamma-1}{\gamma+1} \\ M_2^2 &= \frac{2 + (\gamma-1) M_1^2}{2\gamma M_1^2 - (\gamma-1)} \end{aligned} \tag{1.38}$$

with $M_i = |v_i|/C_S$ being the Mach number. From these relations we can draw some general conclusions about the behaviour of gases undergoing shocks. Changing our frame of reference from the shock to the laboratory frame where the pre-shock region is at rest and the shock is moving through it from left to right we can conclude:

1. the maximum gas compression ρ_2/ρ_1 is:
   (a) monoatomic gas: 4
   (b) diatomic gas: 6
2. the Mach number of the pre-shock region is $M_1 < 1$, and for the post-shock region it is $M_2 > 1$
3. shocks generate entropy and are the only source of entropy for the integral form of hydro equations without viscosity.

1.4 Discontinuous Galerkin Method

In Section 1.3.2 we have derived the Euler equations. They are a system of hyperbolic partial differential equations which describe the conservation of mass, momentum and total energy of a fluid as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{\alpha=1}^{d} \frac{\partial \boldsymbol{f}_\alpha(\boldsymbol{u})}{\partial x_\alpha} = 0, \tag{1.39}$$

We can define a state vector $\mathbf{u}$ in which we collect the five independent variables from Eq. (1.2):

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho \boldsymbol{v} \\ e \end{bmatrix}, \quad e = \rho u + \frac{1}{2}\rho \boldsymbol{v}^2. \tag{1.40}$$

By defining the ideal gas equation of state,

$$p = \rho u \left(\gamma - 1\right), \tag{1.41}$$

we make our system complete. The last missing ingredient from Eq. (1.39) are the fluxes $\boldsymbol{f_\alpha}(\boldsymbol{u})$:

$$\boldsymbol{f}_1 = \begin{pmatrix} \rho v_x \\ \rho v_x v_x + p \\ \rho v_x v_y \\ \rho v_x v_z \\ (\rho e + p) v_x \end{pmatrix}, \ \boldsymbol{f}_2 = \begin{pmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y v_y + p \\ \rho v_y v_z \\ (\rho e + p) v_y \end{pmatrix}, \ \boldsymbol{f}_3 = \begin{pmatrix} \rho v_z \\ \rho v_x v_z \\ \rho v_y v_z \\ \rho v_z v_z + p \\ (\rho e + p) v_z \end{pmatrix}. \tag{1.42}$$

Assembling the flux vectors into $\boldsymbol{F} = (\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3)$, we can write the Euler equation in its compact form,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{F} = 0. \tag{1.43}$$

In classic finite volume methods this equation is solved on a grid with each cell holding exactly one value for each element of the state vector. Below we introduce how this is done for our choice of a high-order Discontinuous Galerkin (DG) method.

1.4.1 Representation of conserved variables

In the Discontinuous Galerkin approach, the state vector $\boldsymbol{u}^K(\boldsymbol{x}, t)$ in each cell K is expressed as a linear combination of time-independent, basis functions $\phi_l^K(\boldsymbol{x})$ and $\boldsymbol{w}_l^K(t)$ N time dependent weights.

$$\boldsymbol{u}^K(\boldsymbol{x}, t) = \sum_{l=1}^{N} \boldsymbol{w}_l^K(t)\, \phi_l^K(\boldsymbol{x}). \tag{1.44}$$

As such the global state of the simulation is fully described by the set of all weights. The "Discontinuous" part of the method's name originates from the lack of any constraint on the continuity of the solution across cell boundaries.

The implementation of DG in this thesis uses Legendre polynomials to form its orthonormal basis functions. The rectangular cells used are all of equal size forming a classic Cartesian grid. Note that both order of the basis function as well as the cell size and shape can be generalized in DG.

1.4.2 Time evolution

The derivation of time evolution of the time dependent weights w_l^K starts by taking the Euler equation (1.39), multiplying it with a basis function and integrating it over cell K:

$$\int_K \phi_l^K \frac{\partial \boldsymbol{u}}{\partial t} \mathrm{d}\boldsymbol{x} + \int_K \phi_l^K\, \boldsymbol{\nabla} \boldsymbol{F}\, \mathrm{d}\boldsymbol{x} = 0. \tag{1.45}$$

Now integrating the second term by parts and applying the divergence theorem to the intermediate result leads to the so-called weak formulation of the conservation law:

$$\int_K \phi_l^K \frac{\partial \boldsymbol{u}}{\partial t} \mathrm{d}\boldsymbol{x} + \int_{\partial K} \phi_l^K\, \boldsymbol{F}\, \mathrm{d}\boldsymbol{n} - \int_K \boldsymbol{\nabla} \phi_l^K\, \boldsymbol{F}\, \mathrm{d}\boldsymbol{x} = 0. \tag{1.46}$$

with $|K|$ being the cell volume.

We now insert the definition $\boldsymbol{u}$ and use the orthonormal property of our set of basis functions to arrive at a differential equation for the time evolution of the weights:

$$|K| \frac{\mathrm{d}\boldsymbol{w}_l^K}{\mathrm{d}t} = \int_K \boldsymbol{\nabla} \phi_l^K\, \boldsymbol{F}\, \mathrm{d}\boldsymbol{x} - \int_{\partial K} \phi_l^K\, \boldsymbol{F}^\star(\boldsymbol{u}^+, \boldsymbol{u}^-)\, \mathrm{d}\boldsymbol{n}. \tag{1.47}$$

The discontinuity of states across cell interfaces in the third term is addressed by replacing $\boldsymbol{F}(\boldsymbol{u})$ on cell surfaces with a flux function $\boldsymbol{F}^\star(\boldsymbol{u}^+, \boldsymbol{u}^-)$ where $\boldsymbol{u}^+$ and $\boldsymbol{u}^-$ are the state vectors parallel and anti-parallel to $\boldsymbol{n}$. Here we can reach for a tool commonly deployed in finite volume methods – the Riemann solver. More precisely, we use the approximate Riemann HLLC solver solver by Toro (2009) as implemented in the AREPO code (Springel, 2010; Weinberger et al., 2020).

1.5 Turbulence

In this section we will introduce the theory of turbulence. What characterises turbulent flows, how do we study them, which scales are important in it and how the energy is transferred by turbulence.

An important number describing how a given fluid flow is susceptible to turbulence is its Reynolds number. Defined as the ratio between the inertial force to viscous or friction force

$$\mathrm{Re} = \frac{uL}{\nu} \tag{1.48}$$

where u is the flow speed, L a characteristic length and ν is the fluid's kinematic viscosity. Flows with low Reynolds numbers are stabilised by viscous forces and are therefore more likely to be smooth, laminar. While high Reynolds number flows are turbulent – characterised by vortices, eddies and other instabilities, due to being dominated by inertial forces. The chaotic flow characteristic of turbulence defies classical fluid description and we have to resort to a statistical description (Pope, 2000).

Now consider a high Reynolds number flow. Under the assumption of a subsonic flow, where we can assume the fluid is practically incompressible and only stirred by solenoidal shear modes. A shear flow is known to give rise to various fluid instabilities, chief among them the Kelvin-Helmholtz instability. It gives rise to many vortices of various sizes and it is exactly these swirling motions on which Kolmogorov (1941) based his theory of turbulence which we will briefly introduce below.

1.5.1 Kolmogorov's theory of incompressible turbulence

Let us consider a turbulent flow in a quasi-stationary state established by a constant time-averaged energy injection rate ϵ [J/g]. The eddies that make up this flow vary in size l and have a corresponding characteristic velocity that is a function of their size $u(l)$ with a corresponding timescale $\tau(l) = \frac{l}{u}$. We can now define a scale dependent Reynolds number

$$Re(l) = \frac{lu(l)}{\nu}. \tag{1.49}$$

The process of large eddies, which are unaffected by viscosity, to create ever smaller ones continues until $Re(l)$ approaches unity, at which point viscous forces take over and efficiently convert the kinetic motions into heat through dissipation. This transfer of large scale kinetic energy to smaller scales is known as the energy cascade and is one of the main features of Kolmogorov's theory.

For high Reynolds number flows Kolmogorov hypothesised that the small scale motions (compared to the driving scale) are statistically isotropic and do not hold any information about larger scales. And consequently the statistical properties of these small scale motions can depend only on the energy injection rate ϵ and shear viscosity ν.

Using dimensional analysis we define Kolmogorov length and the associated velocity and timescale

$$\eta = \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}, \tag{1.50}$$

$$u_\eta = (\epsilon\nu)^{\frac{1}{4}}, \tag{1.51}$$

$$\tau_\eta = \sqrt{\left(\frac{\nu}{\epsilon}\right)}. \tag{1.52}$$

Plugging in the just defined characteristic scales into Eq. (1.49) we can see that at Kolmogorov length scales

$$Re(\eta) = \frac{\eta\ u_\eta}{\nu} = 1, \tag{1.53}$$

the Reynolds number is unity and as such the Kolmogorov length describes the dissipation range.

The second Kolmogorov's similarity hypothesis states that for a range of scales between the driving and the dissipation range $L_0 > l > \eta$ where the statistics are determined exclusively by energy injection ϵ.

In this case we can combine ϵ with a eddy size l, where $L_0 > l > \eta$ and construct

$$u(l) = (\epsilon l)^{\frac{1}{3}}, \tag{1.54}$$

$$\tau(l) = \left(\frac{l^2}{\epsilon} l\right)^{\frac{1}{3}}. \tag{1.55}$$

In the inertial range the energy transfer rate is expected to be scale invariant. First we define the transfer rate

$$T(l) \sim \frac{u^2(l)}{\tau(l)}, \tag{1.56}$$

and under the assumption of scale invariance

$$T(l)\ \epsilon, \tag{1.57}$$

meaning that all injected energy simply moves through the inertial range. This gives us

$$\epsilon \sim \frac{U_0^3}{L_0}, \tag{1.58}$$

where U_0 is the characteristic velocity of large scale, L_0 sized eddies.

This allows us to derive the scaling relations between the all three characteristic flow properties in the inertial range and Re,

$$\frac{\eta}{L_0} = Re^{-\frac{3}{4}}, \tag{1.59}$$

$$\frac{u_\eta}{U_0} = Re^{-\frac{1}{4}}, \tag{1.60}$$

$$\frac{\tau_\eta}{\tau} = Re^{-\frac{1}{2}}, \tag{1.61}$$

and conclude that the properties of the inertial range are completely set by the Reynolds number!

Energy dissipation of Kolmogorov turbulence

Wave number k is defined as $k = 2\pi/l$ and the amount of kinetic energy stored in turbulent motions between two wave numbers is

$$\triangle E = \int_{k_1}^{k_2} E(k)\ dk, \tag{1.62}$$

with $E(k)$ being the energy spectrum of Kolmogorov turbulence. In the case of inertial range we know that every statistic depends only on ϵ and l or in this case $k(l)$. As such it follows

$$E(k) = C\ \epsilon^a\ b^b, \tag{1.63}$$

with C being a numerical constant determined from simulations and experiments to be ~ 1.5 (Pope, 2000) and $a,\ b$ numbers obtained through dimensional analysis. We quickly see that $a = 2/3$ and $b = -5/3$. We have arrived at the famous $-5/3$ slope characteristic of energy power spectrum of subsonic turbulence.

1.6 Challenges in modelling turbulence and overview of this thesis

The thesis is structured as follows: In Chapter 2 we describe out first implementation of the DG method on GPUs with sub-cell shock capturing. We extensively tested our implementation using various standard fluid dynamics tests and driven isothermal turbulence and implemented a novel Navier-Stokes solver, which has proven difficult for DG methods so far. In Chapter 3 we apply the method from the previous chapter to the problem of supersonic driven isothermal turbulence. We present an improven shock capturing technique necessary to robustly threat the network of shocks present in a $\mathcal{M} > 10$ turbulent box. Additionally we employ a novel projection of primitive variables to cell edges which proves crucial to stability. Finally, at the end of the thesis, we summarise in Chapter 4 our results and discuss possible extensions of the work and future research directions.

Chapter 2

High-order Discontinuous Galerkin hydrodynamics with sub-cell shock capturing on GPUs

This work has been published in the Monthly Notices of the Royal Astronomical Society, Volume 522, Issue 1, Pages 982-1008.

Hydrodynamical numerical methods that converge with high-order hold particular promise for astrophysical studies, as they can in principle reach prescribed accuracy goals with higher computational efficiency than standard second- or third-order approaches. Here we consider the performance and accuracy benefits of Discontinuous Galerkin (DG) methods, which offer a particularly straightforward approach to reach extremely high order. Also, their computational stencil maps well to modern GPU devices, further raising the attractiveness of this approach. However, a traditional weakness of this method lies in the treatment of physical discontinuities such as shocks. We address this by invoking an artificial viscosity field to supply required dissipation where needed, and which can be augmented, if desired, with physical viscosity and thermal conductivity, yielding a high-order treatment of the Navier-Stokes equations for compressible fluids. We show that our approach results in sub-cell shock capturing ability, unlike traditional limiting schemes that tend to defeat the benefits of going to high order in DG in problems featuring many shocks. We demonstrate exponential convergence of our solver as a function of order when applied to smooth flows, such as the Kelvin-Helmholtz reference problem of Lecoanet et al. (2016). We also demonstrate excellent scalability of our GPU implementation up to hundreds of GPUs distributed on different compute nodes. In a first application to driven, sub-sonic turbulence, we highlight the accuracy advantages of high-order DG compared to traditional second-order accurate methods, and we stress the importance of physical viscosity for obtaining accurate velocity power spectra.

2.1 Introduction

Computational fluid dynamics has become a central technique in modern astrophysical research (for reviews, see, e.g., Trac & Pen, 2003; Vogelsberger et al., 2020; Andersson & Comer, 2021). It is used in numerical simulations to advance the understanding of countless systems, ranging from planet formation (e.g. Nelson et al., 2000) over the evolution of stars (e.g. Edelmann et al., 2019), and the interplay of gas, black holes and stars in galaxy formation (e.g. Weinberger et al., 2017), up to extremely large scales involving clusters of galaxies (e.g. Dolag et al., 2009) or the filaments in the cosmic web (e.g. Mandelker et al., 2019).

This wide breadth of scientific applications is also mirrored in a bewildering diversity of numerical discretization schemes. Even so the underlying equations for thin, non-viscous gases – the Euler equations – are the same in a broad class of astrophysical studies, the commonly applied numerical methods come in many different flavors, and are sometimes based on radically different principles. At a basic level, one often distinguishes between Lagrangian and Eulerian discretization schemes. The former partition the gas into elements of (nearly) constant mass, as done for example in the popular smoothed particle hydrodynamics (SPH) approach (e.g. Monaghan, 1992) and its many derivatives. In contrast, the latter discretize the volume using a stationary (often Cartesian) mesh (e.g. Stone & Norman, 1992), such that the fluid is represented as a field. Hybrid approaches, which for example use an unstructured moving-mesh (Springel, 2010) are also possible.

For mesh-based codes, finite-volume and finite-element methods are particularly popular. In the finite-volume approach, one records the averaged state in a cell, which is updated in time by the numerical scheme. This approach combines particularly nicely with the conservative character of the Euler equations, because the updates of the conserved quantities in each cell can be expressed as pair-wise fluxes through cell boundaries, yielding not only a manifestly conservative approach but also a physically intuitive formulation of the numerical method. In finite-element approaches one instead expands the fluid state in terms of basis functions. In spectral methods, the support of the basis functions can be the full simulation domain, for example if Fourier series are used to represent the system.

Discontinuous Galerkin (DG) approaches (first introduced for non-linear problems by Cockburn & Shu, 1989), which are the topic of this thesis, are a particular kind of finite-element approaches in which a series expansion for the solution is carried out separately within each computational cell (which can have a fairly general shape). Inside a cell, it is thus simply a truncated spectral method. The solutions for each of the cells are coupled with each other, however, at the surfaces of the cells. Interestingly, high-order accuracy of global solutions can be obtained simply through the high order of the spectral method applied inside a cell, while it does not require continuity of the solutions at the cell interfaces. This makes it particularly straightforward to extend DG schemes to essentially arbitrarily high order, because this does not make the coupling at cell interfaces any more complicated. This is quite different from high-order finite volume schemes, where the reconstruction step requires progressively deeper stencils at high order (Janett et al., 2019).

Another advantage of the DG approach is that it allows in principle cells of different convergence order to be directly next to each other (Schaal et al., 2015). This makes a spatially varying mesh resolution, or a spatially varying expansion order, more straightforward to implement than in high-order extensions of finite volume methods, where typically the high-order convergence property is compromised at resolution changes unless preserved with special treatments.

Despite these advantages, DG methods have only recently begun to be considered in astrophysics. First implementations and applications include Mocz et al. (2014); Schaal et al. (2015); Kidder et al. (2017); Velasco Romero et al. (2018); Guillet et al. (2019), as well as more recently Lombart & Laibe (2021); Markert et al. (2022); Deppe et al. (2022). We here focus on exploring a new implementation of DG that we developed from the ground up for use with graphical processing units (GPUs). The recent advent of exascale supercomputers has been enabled through the use of graphical processing units (GPUs) or various other types of accelerator units. The common feature of these accelerators is the capability to execute a large number of floating point operations at the expense of lower memory bandwidth and total memory per computing unit (few MBs compared to few GBs on an ordinary compute node) compared to the CPU. Another peculiarity of accelerators is that they have hundreds of computing units (roughly equivalent to CPU cores) which execute operations in a single instruction, multiple data (SIMD) mode. Since many of the newest and largest supercomputers use such accelerators, it becomes imperative to either modify existing simulation codes for their efficient use, or to write new codes optimized for this hardware from scratch.

While there are already many successes in the literature for both approaches (e.g. Schneider & Robertson, 2015; Ocvirk et al., 2016; Wibking & Krumholz, 2022), most current simulation work in the astrophysical literature is still being carried out with CPU codes. Certainly one reason is that large existing code bases are not easily migrated to GPUs. Another is that not all numerical solvers easily map to GPUs, making it hard or potentially impossible to port certain simulation applications to GPUs.

However, there are also numerous central numerical problems where GPU computing should be applicable and yield sizable speed-ups. One is the study of hydrodynamics with uniform grid resolutions, as needed for turbulence. In this work, we thus focus on developing a new implementation of DG that is designed to run on GPUs. We base our implementation of DG on Schaal et al. (2015) and Guillet et al. (2019), with one critical difference. We do not apply the limiting schemes described in these studies as they defeat the benefits of high-order approaches when strong shocks are present. Rather, we will revert to the idea of deliberately introducing a small amount of artificial viscosity to capture shocks, i.e. to add required numerical viscosity just where it is needed, and ideally with the smallest amount necessary to suppress unphysical oscillatory solutions. As we will show, with this approach the high-order approach can still be applied well to problems involving shocks, without having to sacrifice all high-order information on the stake of a slope limiter.

This chapter is structured as follows. In Section 2.2, we detail the mathematical basis of the Discontinuois Galerkin discretization of hydrodynamics as used by us. In Section 2.3,

we generalize the treatment to include source terms which involve derivatives of the fluid states, such as needed for the Navier-Stokes equations, or for our artificial viscosity treatment for that matter. We then turn to a discussion of shock capturing and oscillation control in Section 2.4. The following Section 2.5 is devoted to elementary tests, such as shock tubes and convergence tests for smooth problems. In Section 2.6 we then show results for "resolved" Kelvin-Helmholtz instabilities, and in Section 2.7, we give results for driven isothermal turbulence and discuss to what extent DG methods improve the numerical accuracy and efficiency of such simulations. Implementation and parallelization issues of our code, in particular with respect to using GPUs, are described in Section 2.8, while in Section 2.9, we discuss the performance and scalability of our new GPU-based hydrodynamical code. Finally, we give a summary and our conclusions in Section 2.10.

2.2 Discontinuous Galerkin discretization of the Euler equations

The Euler equations are a system of hyperbolic partial differential equations. They encapsulate the conservation laws for mass, momentum and total energy of a fluid, and can be expressed as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{\alpha=1}^{d} \frac{\partial \boldsymbol{f}_\alpha(\boldsymbol{u})}{\partial x_\alpha} = 0, \tag{2.1}$$

where the sum runs over the d dimensions of the considered problem. The state vector $\mathbf{u}$ holds the conserved variables: density, momentum density, and total energy density:

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho \boldsymbol{v} \\ e \end{bmatrix}, \quad e = \rho u + \frac{1}{2}\rho \boldsymbol{v}^2. \tag{2.2}$$

To make our system complete we need an equation of state which connects the hydrodynamics pressure p with the specific internal energy u. If γ is the adiabatic index, i.e. the ratio of the specific heat of the gas at a constant pressure C_p to its specific heat at a constant volume C_v, the ideal gas equation of state is

$$p = \rho u \left(\gamma - 1\right). \tag{2.3}$$

We also need to specify the second term of Eq. (2.1). The fluxes $\boldsymbol{f_\alpha}(\boldsymbol{u})$ in three dimensions are:

$$\boldsymbol{f}_1 = \begin{pmatrix} \rho v_x \\ \rho v_x v_x + p \\ \rho v_x v_y \\ \rho v_x v_z \\ (\rho e + p) v_x \end{pmatrix}, \; \boldsymbol{f}_2 = \begin{pmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y v_y + p \\ \rho v_y v_z \\ (\rho e + p) v_y \end{pmatrix}, \; \boldsymbol{f}_3 = \begin{pmatrix} \rho v_z \\ \rho v_x v_z \\ \rho v_y v_z \\ \rho v_z v_z + p \\ (\rho e + p) v_z \end{pmatrix}. \tag{2.4}$$

By summarizing the flux vectors into $\boldsymbol{F} = (\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3)$, we can also write the Euler equations in the compact form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{F} = 0, \tag{2.5}$$

which highlights their conservative character. Numerically solving this set of non-linear, hyperbolic partial differential equations is at the heart of computational fluid dynamics. Here we shall consider the specific choice of a high-order Discontinuous Galerkin (DG) method.

2.2.1 Representation of conserved variables in DG

In the Discontinuous Galerkin approach, the state vector $\boldsymbol{u}^K(\boldsymbol{x}, t)$ in each cell K is expressed as a linear combination of time-independent, differentiable basis functions $\phi_l^K(\boldsymbol{x})$,

$$\boldsymbol{u}^K(\boldsymbol{x}, t) = \sum_{l=1}^{N} \boldsymbol{w}_l^K(t)\, \phi_l^K(\boldsymbol{x}), \tag{2.6}$$

where the $\boldsymbol{w}_l^K(t)$ are N time dependent weights. Since the expansion is carried out for each component of our state vector separately, the weights $\boldsymbol{w}_l^K$ are really vector-valued quantities with 5 different values in 3D for each basis l. Each of these components is a single scalar function with support in the cell K.

The union of cells forms a non-overlapping tessellation of the simulated domain, and the global numerical solution is fully specified by the set of all weights. Importantly, no requirement is made that the piece-wise smooth solutions within cells are continuous across cell boundaries.

We shall use a set of orthonormal basis functions that is equal in all cells (apart from a translation to the cell's location), and we specialize our treatment in this chapter to Cartesian cells of constant size. The DG approach can however be readily generalized to other mesh geometries, and to meshes with variable cell sizes. Also, we will here use a constant number N of basis functions that is equal for all cells, and determined only by the global order p of the employed scheme. In principle, however, DG schemes allow this be varied from cell to cell (so-called p-refinement).

2.2.2 Time evolution

To derive the equations governing the time evolution of the DG weights w_l^K, we start with the original Euler equation from Eq. (2.5), multiply it with one of the basis functions and integrate over the corresponding cell K:

$$\int_K \phi_l^K \frac{\partial \boldsymbol{u}}{\partial t} \mathrm{d}\boldsymbol{x} + \int_K \phi_l^K\, \boldsymbol{\nabla} \boldsymbol{F}\, \mathrm{d}\boldsymbol{x} = 0. \tag{2.7}$$

Integration by parts of the second term and applying the divergence theorem leads to the so-called weak formulation of the conservation law:

$$\int_K \phi_l^K \frac{\partial \boldsymbol{u}}{\partial t} \mathrm{d}\boldsymbol{x} + \int_{\partial K} \phi_l^K \, \boldsymbol{F} \, \mathrm{d}\boldsymbol{n} - \int_K \boldsymbol{\nabla} \phi_l^K \, \boldsymbol{F} \, \mathrm{d}\boldsymbol{x} = 0. \tag{2.8}$$

where $|K|$ stands for the volume of the cell (or area in 2D).

If we now insert the basis function expansion of $\boldsymbol{u}$ and make use of the orthonormal property of our set of basis functions,

$$\int_K \phi_l^K(\boldsymbol{x}) \phi_m^K(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \delta_{l,k} |K|, \tag{2.9}$$

we obtain a differential equation for the time evolution of the weights:

$$|K| \frac{\mathrm{d}\boldsymbol{w}_l^K}{\mathrm{d}t} = \int_K \boldsymbol{\nabla} \phi_l^K \, \boldsymbol{F} \, \mathrm{d}\boldsymbol{x} - \int_{\partial K} \phi_l^K \, \boldsymbol{F}^\star(\boldsymbol{u}^+, \boldsymbol{u}^-) \, \mathrm{d}\boldsymbol{n}. \tag{2.10}$$

Here we also considered that the flux function at the surface of cells is not uniquely defined if the states that meet at cell interfaces are discontinuous. We address this by replacing $\boldsymbol{F}(\boldsymbol{u})$ on cell surfaces with a flux function $\boldsymbol{F}^\star(\boldsymbol{u}^+, \boldsymbol{u}^-)$ that depends on both states at the interface, where $\boldsymbol{u}^+$ is the outwards facing state relative to $\boldsymbol{n}$ (from the neighbouring cell), and $\boldsymbol{u}^-$ is the state just inside the cell. We will typically use a Riemann solver for determining $\boldsymbol{F}^\star$, making this akin to Godunov's approach in finite volume methods. In fact, the same type of exact or approximate Riemann solvers can be used here as well. We use for ordinary gas dynamics a simplified version of the Riemann HLLC solver by Toro (2009) as implemented in the AREPO code (Springel, 2010; Weinberger et al., 2020). We have also included an exact Riemann solver in case an isothermal equation of state is specified.

What remains to be done to make an evaluation of Eq. (2.10) practical is to approximate both the volume and surface integrals numerically, and to choose a specific realization for the basis functions. We shall briefly discuss both aspects below. Another ingredient is the definition of the weights for the initial conditions. Thanks to the completeness of the basis, they can be computed by projecting the state vector $\boldsymbol{u}(\boldsymbol{x})$ of the initial conditions onto the basis functions ϕ_l^K of each cell:

$$\boldsymbol{w}_l^K = \frac{1}{|K|} \int_K \boldsymbol{u} \, \phi_l^K \, \mathrm{d}V. \tag{2.11}$$

If a finite number N of basis functions is used to approximate the numerical solution, the total approximation error is then

$$L1 = \frac{1}{|K|} \int_K \left| \boldsymbol{u}(\boldsymbol{x}) - \sum_{l=1}^{N} \boldsymbol{w}_l^K \, \phi_l^K(\boldsymbol{x}) \right| \, \mathrm{d}V. \tag{2.12}$$

We shall use this L1 norm to examine the accuracy of our code when analytic solutions are known.

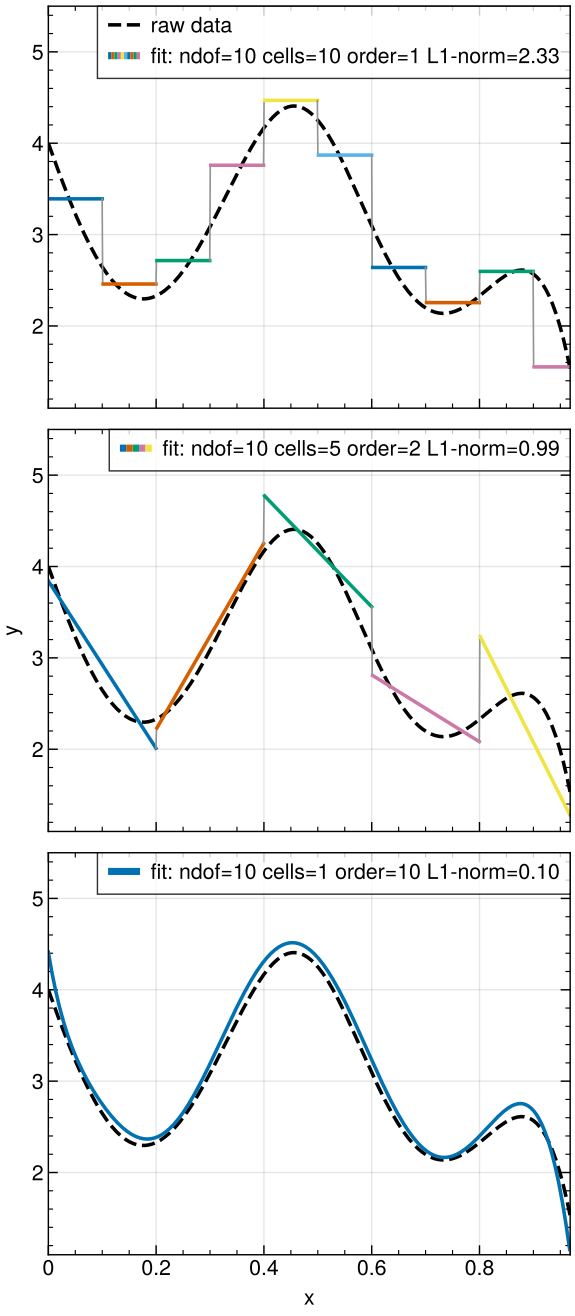


Figure 2.1: An example of fitting an arbitrary, smooth function $y = f(x)$ with 10 degrees of freedom, but varying number of cells and polynomial orders used for these cells, as labelled in the different panels. The L1 error norm for approximating the function is highest in case piece-wise constant approximations are used, while it drops when fewer, but piece-wise linear cells are used, and finally reaches its lowest value when a single cell with a single 10th order polynomial is used.

2.2.3 Legendre basis function

Following Schaal et al. (2015), we select Legendre polynomials $P_l(\xi)$ to construct our set of basis functions. They are defined on a canonical interval $[-1, 1]$ and can be scaled such that they form an orthogonal basis with normalization chosen as:

$$\int_{-1}^{1} P_l(\xi) P_m(\xi) \mathrm{d}\xi = 2\,\delta_{l,m}. \tag{2.13}$$

Note that the 0-th order Legendre polynomial is just a constant term, while the 1-st order features a simple pure linear dependence. In general, $P_l(\xi)$ is a polynomial of degree l.

Within each cell, we define local coordinates $\boldsymbol{\xi} \in [-1, 1]^d$. The translation between global coordinates $\boldsymbol{x}$ to local cell coordinates $\boldsymbol{\xi}$ is:

$$\boldsymbol{\xi}^K = \frac{2}{h}\left(\boldsymbol{x} - \boldsymbol{x}_c^K\right), \tag{2.14}$$

with h being the cell size in one dimension, and $\boldsymbol{x}_c^K$ is the cell centre in world coordinates. Multi-dimensional basis functions are simply defined as Cartesian products of Legendre polynomials, for example in three dimensions as follows:

$$\phi_l^K(\boldsymbol{x}) = P_l^{\mathrm{3D}}[\boldsymbol{\xi}^K(\boldsymbol{x})], \tag{2.15}$$

with

$$P_l^{\mathrm{3D}}[\boldsymbol{\xi}^K] \equiv P_{l_x}(\xi_x^K) \cdot P_{l_y}(\xi_y^K) \cdot P_{l_z}(\xi_z^K), \tag{2.16}$$

where the generalized index l enumerates different combinations of Legendre polynomials $l_x(l)$, $l_y(l)$, and $l_z(l)$ in the different directions. In practice, we truncate the expansion at a predefined order n, and discard all tensor products in which the degree of the resulting polynomial exceeds n. This means that we end up in 3D with

$$N^{\mathrm{3D}}(n) = \frac{1}{6}(n+1)(n+2)(n+3) \tag{2.17}$$

basis functions, each a product of three Legendre polynomials of orders $l_{z,y,z} \in \{0, \ldots, n\}$. In 2D, we have

$$N^{\mathrm{2D}}(n) = \frac{1}{2}(n+1)(n+2), \tag{2.18}$$

and in 1D the number is $N^{\mathrm{1D}}(n) = n + 1$. The expected spatial convergence order due to the leading truncation error is in each case $p = n + 1$. From now on we will refer to p as the order of our DG scheme, with $n = p - 1$ being the highest degree among the involved Legendre polynomials.

In Figure 2.1, we show an example of approximating a smooth function with Legendre polynomials of different order and with a different number of cells, but keeping the number of degrees constant. In this case, the approximation error tends to be reduced by going to higher order, even when this implies using fewer cells.

2.2.4 Gaussian quadrature

An integration of a general function $f(x)$ over the interval $[-1,1]$ can be approximated by Gaussian quadrature rules, as

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \simeq \sum_{j=1}^{n_g} g_j\, f(x_j) \tag{2.19}$$

for a set of evaluation points x_j and suitably chosen quadrature weights g_j. We use ordinary Gaussian quadrature with internal points only. The corresponding integration rule with n_g evaluation points is exact for polynomials up to degree $2n_g - 1$. If we use Legendre polynomials up to order n, we therefore should use at least $n_g \geq (n+1)/2$ integration points. Note, however, that the nonlinear dependence of the flux function on the state vector $\boldsymbol{u}$ means that we actually encounter rational functions as integrands and not just simple polynomials. As a result, we need unfortunately a more conservative number of integration points for sufficient accuracy and stability in practice. A good heuristic is to take the number of basis functions used for the one-dimensional case as a guide, so that one effectively employs at least one function evaluation per basis function. This means we pick $n_g = n+1$ in what follows.

Multi-dimensional integrations, as needed for the surface and volume integrals in our Cartesian setup, can be carried out through tensor products of Gaussian integrations. We denote the corresponding function evaluation points as $\boldsymbol{\xi}_{\boldsymbol{j}}^{\mathrm{vol}} = (x_{j_1}, x_{j_2}, x_{j_3})$ and Gaussian weights as $g_{\boldsymbol{j}}^{\mathrm{vol}} = g_{j_1} \cdot g_{j_2} \cdot g_{j_3}$ for the combination $\boldsymbol{j} = (j_1, j_2, j_3)$ of Gaussian quadrature points needed for integrations over the cell volume in 3D. For surface integrations over our cubical cells, we correspondingly define $\boldsymbol{\xi}_{\boldsymbol{k},x+}^{\mathrm{sur}} = (+1, x_{k_1}, x_{k_2})$, and $\boldsymbol{\xi}_{\boldsymbol{k},x-}^{\mathrm{sur}} = (-1, x_{k_1}, x_{k_2})$ for evaluation points on the right and left surface in the x-direction of one of our cubical cells, with $\boldsymbol{k} = (k_1, k_2)$ and likewise for the y- and z-directions. The corresponding Gaussian quadrature weights are given by $g_{\boldsymbol{k}}^{\mathrm{sur}} = g_{k_1} \cdot g_{k_2}$.

Putting everything together, we arrive at a full set of discretized evolutionary equations for the weights. For definiteness, we specify this here for the three dimensional case:

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{w}_l^K}{\mathrm{d}t} = {} & \frac{1}{4}\sum_{\alpha=1}^{3} \sum_{\substack{\boldsymbol{j}\in \\ [1,n_g]^3}} \left\{ \boldsymbol{f}_\alpha[\boldsymbol{u}^K(\boldsymbol{\xi}_{\boldsymbol{j}}^{\mathrm{vol}})] \cdot \frac{\partial P_l^{3\mathrm{D}}(\boldsymbol{\xi}_{\boldsymbol{j}}^{\mathrm{vol}})}{\partial \xi_\alpha} \right\} g_{\boldsymbol{j}}^{\mathrm{vol}} \\
& -\frac{1}{8}\sum_{\alpha=1}^{3} \sum_{\substack{\boldsymbol{k}\in \\ [1,n_g]^2}} \left\{ P_l^{3\mathrm{D}}(\boldsymbol{\xi}_{\boldsymbol{k},\alpha+}^{\mathrm{sur}})\, \boldsymbol{f}_\alpha^\star \left[\boldsymbol{u}^{K,\alpha+}(\xi_{\boldsymbol{k},\alpha-}^{\mathrm{sur}}), \boldsymbol{u}^K(\boldsymbol{\xi}_{\boldsymbol{k},\alpha+}^{\mathrm{sur}})\right] \right. \\
& \qquad \left. - P_l^{3\mathrm{D}}(\boldsymbol{\xi}_{\boldsymbol{k},\alpha+}^{\mathrm{sur}})\, \boldsymbol{f}_\alpha^\star \left[\boldsymbol{u}^K(\boldsymbol{\xi}_{\boldsymbol{k},\alpha-}^{\mathrm{sur}}), \boldsymbol{u}^{K,\alpha-}(\boldsymbol{\xi}_{\boldsymbol{k},\alpha+}^{\mathrm{sur}})\right] \right\} g_{\boldsymbol{k}}^{\mathrm{sur}}.
\end{aligned} \tag{2.20}$$

Here the notation $\boldsymbol{u}^{K,\alpha+}$ and $\boldsymbol{u}^{K,\alpha-}$ refer to the state vectors evaluated for the right and left neighbouring cells of cell K in the direction of axis α, respectively. The state vector

evaluations themselves are given by

$$\boldsymbol{u}^K(\boldsymbol{\xi}) = \sum_{l=1}^{N} \boldsymbol{w}_l^K \, P_l^{\mathrm{3D}}(\boldsymbol{\xi}). \tag{2.21}$$

Note that the prefactor $1/|K|$ in front of the surface integral terms in Eq. (2.20) turns into $1/8$ as a result of the change of integration variables mediated by Eq. (2.15). The volume integral acquires a factor of $2/h$ from the coordinate transformation, thus the final prefactor becomes $1/4$. The numerical computation of the time derivative of the weights based on a current set of weights is in principle straightforward using Eq. (2.20), but evidently becomes more elaborate at high-order, involving numerous sums per cell.

In passing we note that instead of just counting the number of cells per dimensions, both the storage effort and the numerical work needed is better measured in terms of the number of degrees of freedom per dimension. A fixed number of degrees of freedom (and thus storage space) can be achieved with different combinations of cell size and expansion order. The hope in using high-order methods is that they deliver better accuracy for a fixed number of degrees of freedom, or arguably even more importantly, better accuracy at fixed computational expense.

2.2.5 Time integration

With

$$\dot{\boldsymbol{w}} \equiv \frac{\mathrm{d}\boldsymbol{w}_l^K}{\mathrm{d}t} \tag{2.22}$$

in hand, standard ODE integration methods such as the broad class of Runge-Kutta integrations can be used to advance the solution forward in time. We follow standard procedure and employ strongly positivity preserving (SPP) Runge-Kutta integration rules as defined in Schaal et al. (2015, Appendix D). Note that when higher spatial order is used, we correspondingly use a higher order time integration method, such that the time integration errors do not start dominating over spatial discretization errors. The highest time integration method we use is a 5 stage 4-th order SSP RK method.

The timestep size Δt is set conservatively as

$$\Delta t_{\max} = f_{\mathrm{CFL}} \frac{h}{2p\,(c_{s,\,\max} + v_{\max})}, \tag{2.23}$$

where h is the cell size, f_{CFL} is the Courant–Friedrichs–Lewy factor, $c_{s,\,\max}$ denotes the global maximum sound speed and $v_{\max}$ is the global maximum kinematic velocity, respectively. We use a f_{CFL} of 0.5 for all problems except the shock tube, Sedov blast wave and double blast wave where a more conservative 0.3 was used instead.

For high order runs ($p > 4$) we did not see time integration errors to start dominating over the spatial discretization errors, despite employing only a 4-th order RK scheme. We attribute this to our use of a low Courant factor and to including global maximum velocities in the timestep criterion. Once the errors from time integration would start to

dominate at high order, we could recover sufficient accuracy of our time integration scheme by appropriately scaling the time-step size as $h^{r/4}$.

2.3 Treatment of viscous source terms

As we will discuss later on, our approach for capturing physical discontinuities (i.e. shocks and contact discontinuities) in gas flows deviates from the classical slope-limiting approach and instead relies on a localized enabling of artificial viscosity. Furthermore, we will generalize our method to also account for physical dissipative terms, so that we arrive at a treatment of the full compressive Navier-Stokes equations.

To introduce these methods, we start with a generalized set of Euler equations in 3D that are augmented with a diffusion term in all fluid variables,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F} = \nabla \cdot (\varepsilon \nabla \boldsymbol{u}), \tag{2.24}$$

where $\boldsymbol{u}$ and $\boldsymbol{F}$ are the state vector (2.6) and the flux matrix (2.4), respectively.

The crucial difference between the normal Euler equations (2.1) and this dissipative form is the introduction of a second derivative on the right-hand side, which modifies the character of the problem from being purely hyperbolic to an elliptic type, while retaining manifest conversation of mass, momentum and energy. This second derivative can however not be readily accommodated in our weight update equation obtained thus far. Recall, the reason we applied integration by parts and the Gauss' theorem going from Eq. (2.5) to Eq. (2.8) was to eliminate the spatial derivative of the fluxes. If we apply the same approach to $\nabla \cdot (\varepsilon \nabla \boldsymbol{u})$ we are still left with one ∇-operator acting on the fluid state.

2.3.1 The uplifting approach

In a seminal paper, Bassi & Rebay (1997) suggested a particular treatment of this second derivative inspired by how one typically reduces second (or higher) order ordinary differential equations (ODEs) to first order ODEs. Bassi & Rebay (1997) reduce the order of Eq. (2.24) by introducing the gradient of the state vector, $\boldsymbol{S} \equiv \nabla \boldsymbol{u}$, as an auxiliary set of unknowns. This yields a system of two partial differential equations:

$$\boldsymbol{S} - \nabla \boldsymbol{u} = 0, \tag{2.25}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{F} - \epsilon \boldsymbol{S}) = 0. \tag{2.26}$$

Interestingly, if we consider a basis function expansion for $\boldsymbol{S}$ for each cell in the same way as done for the state vector, then the weak formulation of the first equation can be solved with the DG formalism using as input only the series expansion of the current state $\boldsymbol{u}$. This entails again an integration by parts that yields volume and surface integrations for each cell. To compute the latter, one needs to adopt a surface state $\boldsymbol{u}^\star$ for potentially discontinuous jumps $\boldsymbol{u}^+$ and $\boldsymbol{u}^-$ across the cell boundaries. Bassi & Rebay (1997) suggest to

use the arithmetic mean $\boldsymbol{u}^\star = [\boldsymbol{u}^- + \boldsymbol{u}^+]/2$ for this, so that obtaining the series expansion coefficients for $\boldsymbol{S}$ is straightforward. One can then proceed to solve Eq. (2.26), with a largely identical procedure than for the Euler equation, except that the ordinary flux $\boldsymbol{F}$ is modified by subtracting the viscous flux $\boldsymbol{F}_{\text{visc}} = \epsilon \boldsymbol{S}$. At cell interfaces one furthermore needs to define the viscous flux uniquely somehow, because $\boldsymbol{S}$ can still be discontinuous in general at cell interfaces. Here Bassi & Rebay (1997) suggest to use the arithmetic mean again.

A clear disadvantage of this procedure, which we initially implemented in our code, is that it significantly increases the computational cost, memory requirements and code complexity, because the computation of $\boldsymbol{S}$ involves the same set of volume and surface integrals that are characteristic of the DG approach, except that it actually has to be done *three times* as often than for $\boldsymbol{u}$ in 3D, once for each spatial dimension. But more importantly, we have found that this method is prone to robustness problems, in particular if the initial conditions already contain large discontinuities across cells. In this case, the estimated derivatives inside a cell can reach unphysically large values by the jumps seen on the outer sides of a cell.

In hindsight, this is perhaps not too surprising. For a continuous solution, there is arguably little if anything to be gained by solving Eq. (2.25) with the DG algorithm if a polynomial basis is in use. Because this must then return a solution identical to simply taking the derivatives of the basis functions (which are analytically known) and retaining the coefficients of the expansion. On the other hand, if there are discontinuities in $\boldsymbol{u}$ at the boundaries, the solution for $\boldsymbol{S}$ sensitively depends on the (to a certain degree arbitrary) choice made for resolving the jumps in the computation of the surface integrals for $\boldsymbol{S}$. In particular, there is no guarantee that using the arithmetic mean does not induce large oscillations or unphysical values for $\boldsymbol{S}$ in the interior of cells in certain cases.

For all these reasons we have ultimately abandoned the Bassi & Rebay (1997) method, because it does not yield a robust solution for the diffusion part or the equations in all situations, and does not converge rapidly at high order either. Instead, we conjecture that the key to high order convergence of the diffusive part of the PDE system is the availability of a consistently defined continuous solution across cell boundaries.

2.3.2 Surface derivatives

For internal evaluations of the viscous flux (which in general may depend on $\boldsymbol{u}$ and $\nabla \boldsymbol{u}$) within a cell, we use the current basis function expansion of the solution in the cell and simply obtain the derivative by analytically differentiating the basis functions. We argue that this is the most natural choice as the same interior solution $\boldsymbol{u}$ is used for computing the ordinary hydrodynamical flux.

The problem, however, lies with the surface terms of the viscous flux, as here neither the value of the state vector nor the gradient are uniquely defined, and unlike for the hyperbolic part of the equation, there is no suitable 'Riemann solver' to define a robust flux for the diffusion part of the equation. Simply taking arithmetic averages of the two values that meet at the interface for the purpose of evaluating the surface viscous flux is

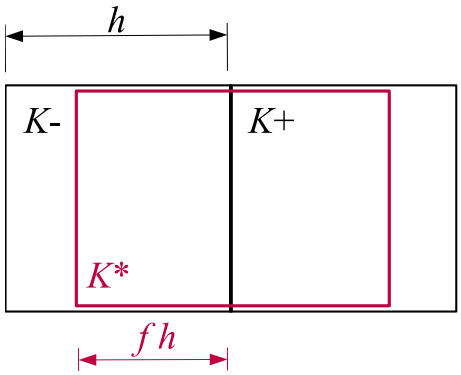


Figure 2.2: Two cells K^- and K^+ that meet at a joint face. The corresponding polynomial solutions $u-$ and $u+$ are in general discontinuous at the interface. To unambiguously define a joint solution and its gradient on the interface, we construct an interpolant solution on a domain $K*$ placed symmetrically around the interface. In the normal direction, a fraction f of both cells is covered (we pick either $f = 3/4$ or $f = 1$ in practice), in the the transverse direction(s), the cells are covered in full.

not accurate and robust in practice.

We address this problem by constructing a new *continuous* solution across a cell interface by considering the current solutions in the two adjacent cells of the interface, and projecting them onto a new joint polynomial expansion in a rectangular domain that covers part (or all) of the two adjacent cells. This approach is similar to the recovery method proposed by van Leer & Nomura (2005) in their work on solving the diffusion equation in DG. This interpolated solution minimizes the L_2 difference to the original (in general discontinuous) solutions in the two cells, but it is continuous and differentiable at the cell interface by construction. The quantities $\boldsymbol{u}$ and $\nabla \boldsymbol{u}$ needed for the evaluation of the viscous surface flux are then computed by evaluating the new basis function expansion at the interface itself.

A sketch of the adopted procedure is shown in Figure 2.2. The two solutions in the two adjacent cells are given by

$$\boldsymbol{u}^{K^-}(\boldsymbol{x}) = \sum_{l=1}^{N} \boldsymbol{w}_l^{K^-} \, \phi^{K^-}(\boldsymbol{x}). \tag{2.27}$$

and

$$\boldsymbol{u}^{K^+}(\boldsymbol{x}) = \sum_{l=1}^{N} \boldsymbol{w}_l^{K^+} \, \phi^{K^+}(\boldsymbol{x}). \tag{2.28}$$

We now seek an interpolated solution in terms of a set of new basis functions $\psi^{K^\star}$ defined on the domain $K^\star$, i.e.

$$\tilde{\boldsymbol{u}}^{K^\star}(\boldsymbol{x}) = \sum_{l=1}^{N^\star} \boldsymbol{q}_l^{K^\star} \, \psi^{K^\star}(\boldsymbol{x}). \tag{2.29}$$

In order to avoid a degradation of accuracy if the solution is smooth, and to provide sufficient accuracy for the gradient, we adopt order $n+1$ for the polynomial basis of $\tilde{\boldsymbol{u}}^{K^\star}$. As for ordinary cells, the generalized index l enumerates different combinations $[l_x(l), l_y(l), l_z(l)]$ of Legendre polynomials and their Cartesian products in the multidimensional case. If, for example, the two cells are oriented along the x-axis, we define

$$\psi_l^{K^\star}(\boldsymbol{x}) = P_{l_x}(\xi_x^{K^\star}) \cdot P_{l_y}(\xi_y^K) \cdot P_{l_z}(\xi_z^K), \tag{2.30}$$

where now the mapping of the x-extension of the domain $K^\star$ into the standard interval $[-1, 1]$ is correspondingly modified as

$$\xi_x^{K^\star} = \frac{1}{f\,h}\left(x - \frac{x_c^{K^-} + x_c^{K^+}}{2}\right), \tag{2.31}$$

where f is the fraction of overlap of each of the two cells (see Fig. 2.2). The coefficients $\boldsymbol{q}_l^{K^\star}$ can then be readily obtained by carrying out the projection integrals

$$\begin{aligned} \boldsymbol{q}_l^{K^\star} &= \frac{1}{|K^\star|}\int_{K^\star} \boldsymbol{u}(\boldsymbol{x})\, \psi_l^{K^\star}(\boldsymbol{x})\,\mathrm{d}V \\ &= \frac{1}{|K^\star|}\sum_{m=1}^{N}\left[\boldsymbol{w}_m^{K^-}\int_{K^-}\phi_m^{K^-}\psi_l^{K^\star}\,\mathrm{d}V + \boldsymbol{w}_m^{K^+}\int_{K^+}\phi_m^{K^+}\psi_l^{K^\star}\,\mathrm{d}V\right]. \end{aligned} \tag{2.32}$$

The projection is a linear operation, and the overlap integrals of the Legendre basis functions can be precomputed ahead of time. In fact, many evaluate to zero due to the orthogonality of our Legendre basis. In particular, this is the case for the transverse basis functions if their order is not equal, so that the projection effectively becomes a sparse matrix operation that expresses the new expansion coefficients in the normal direction as a sum of one or several old expansion coefficients in the normal direction. This can be more explicitly seen by defining Legendre overlap integrals as

$$A_{m,l}^- = \int_{-1}^{0} P_m(2fx+1)P_l(x)\,\mathrm{d}x, \tag{2.33}$$

$$A_{m,l}^+ = \int_{0}^{1} P_m(2fx+1)P_l(x)\,\mathrm{d}x. \tag{2.34}$$

Then the new coefficients can be computed as follows

$$\boldsymbol{q}_{(l_x,l_y,l_z)}^{K^\star} = \frac{1}{2f}\sum_{m_x=0}^{l_x}\left[A_{m_x,l_x}^-\boldsymbol{w}_{(m_x,l_y,l_z)}^{K^-} + A_{m_x,l_x}^+\boldsymbol{w}_{(m_x,l_y,l_z)}^{K^+}\right]. \tag{2.35}$$

Note that for transverse dimensions, only the original Legendre polynomials contribute, hence the new coefficients are simply linear combinations of coefficients that differ only in the order of the Legendre polynomial in the x-direction. Also note that for the transverse

dimensions, the highest Legendre orders l_y and l_z that are non-zero are the same as for the original coefficients, i.e. the fact that we extend the order to $n+1$ becomes only relevant for the direction connecting the two cells.

Another point to note is that the basis function projection can be carried out independently for the left and right side of an interface (corresponding to the first and second part of the sum in eqn. 2.35), each yielding a partial result that can be used in turn to evaluate partial results for $\tilde{\boldsymbol{u}}\ \nabla\tilde{\boldsymbol{u}}$ at the interface. Adding up these partial results then yields the final interface state and and interface gradient. This means that this scheme does not require to send the coefficients $\boldsymbol{w}^{K^\pm}$ to other processors in case K^- and K^+ happen to be stored on different CPUs or GPUs, only "left" and "right" states for $\tilde{\boldsymbol{u}}$ and $\nabla\tilde{\boldsymbol{u}}$ need to be exchanged (which are the partial results that are then summed instead of taking their average), implying the same communication costs as, for example, methods that would rely on taking arithmetic averages of the values obtained separately for the K^- and K^+ sides.

Finally, we choose $f = 3/4$ for the size of the overlap region for $n \leq 2$, but $f = 1$ for higher order $n > 2$. For the choice of $f = 3/4$, the estimate for the first derivative of the interpolated solution ends up being

$$\nabla\tilde{\boldsymbol{u}} = \frac{\boldsymbol{u}^+ - \boldsymbol{u}^-}{h}\boldsymbol{n}, \tag{2.36}$$

for piece-wise constant states, where h is the cell spacing, $\boldsymbol{n}$ is the normal vector of the interface, and $\boldsymbol{u}^\pm$ are the average states in the two cells. This intuitively makes sense for low order. In particular, this will pick up a reasonable gradient even if one starts with a piece-wise constant initial conditions, and even if $n = 0$ (corresponding to DG order $p = 1$) is used. We also obtain the expected convergence orders for diffusion problems (see below) with this choice when $n \leq 2$ is used. On the other hand, we have found that it is necessary to include the full available information of the two adjacent cells by adopting $f = 1$ for still higher order in order to obtain the expected high-order convergence rates for diffusion problems also for $n > 2$.

2.3.3 The Navier-Stokes equations

While we will use the above form of the dissipative terms for our treatment of artificial viscosity (see below), we also consider the full Navier-Stokes equations. They are given by:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F} = \nabla \cdot \boldsymbol{F}_{\mathrm{NS}}, \tag{2.37}$$

where now the Navier-Stokes flux vector $\boldsymbol{F}_{\mathrm{NS}}$ is a non-linear function both of the state vector $\boldsymbol{u}$ and its gradient $\boldsymbol{\nabla u}$. We pick the canonical form

$$\boldsymbol{F}_{\mathrm{NS}} = \begin{pmatrix} 0 \\ \boldsymbol{\Pi} \\ \boldsymbol{v}\cdot\boldsymbol{\Pi} + \chi(\gamma-1)\rho\nabla u \end{pmatrix}, \tag{2.38}$$

with a viscous tensor

$$\boldsymbol{\Pi} = \nu\rho\left(\nabla\boldsymbol{v} + \nabla\boldsymbol{v}^{\mathrm{T}} + \frac{2}{3}\nabla\cdot\boldsymbol{v}\right) \tag{2.39}$$

that dissipates shear motions with viscosity ν. We also include optional heat conduction with thermal diffusivity χ. Note that the derivatives of the primitive variables can be easily obtained from the derivatives of the conservative variables when needed, for example $\boldsymbol{\nabla v} = [\nabla(\rho\boldsymbol{v}) - \boldsymbol{v}\nabla\rho]/\rho$, and one can thus express the velocity gradient $\boldsymbol{\nabla v}$ in terms of $\boldsymbol{\nabla u}$ and $\boldsymbol{u}$.

2.3.4 Passive tracer

Finally, for later application to the Kelvin-Helmholtz problem, we follow Lecoanet et al. (2016) and add a passive, conserved tracer variable to the fluid equations. The density of the tracer is $c\rho$, with c being its dimensionless relative concentration. It can be added as a further row to the state vector $\boldsymbol{u}$. Since the tracer is conserved and simply advected with the local velocity, the corresponding entry in the flux vector is $c\rho\boldsymbol{v}$. Further, we can also allow for a diffusion of the tracer with diffusivity η, by adding $\eta\rho\nabla c$ in the corresponding row of the Navier-Stokes flux vector. The governing equation for the passive tracer dye is hence

$$\frac{\partial(c\rho)}{\partial t} + \nabla\cdot(c\rho\boldsymbol{v}) = \nabla(\eta\rho\nabla c). \tag{2.40}$$

2.4 Shock capturing and oscillation control

2.4.1 Artificial viscosity

High-order numerical methods are prone to oscillatory behaviour around sharp jumps of density or pressure. Such physical discontinuities arise naturally at shocks in supersonic fluid motion, and they are an ubiquitous phenomenon in astrophysical gas dynamics. In fact, the Euler equations have the interesting property that perfectly initial conditions can evolve with time into states that feature real discontinuities. The physical dissipation that must happen in these jumps is implicitly dictated by the conservation laws, but discrete numerical methods may not always produce the required level of dissipation, such that postshock oscillations are produced that are reminiscent of the Gibbs phenomenon in Fourier series expansion around jump discontinuities.

Our DG code produces these kinds of oscillations with increasing prominence at higher and higher order when discontinuities are present. And once the oscillations appear, they do not necessarily get quickly damped because of the very low numerical dissipation of high-order DG. Shocks, in particular, seed new oscillations with time, because inside cells the smooth *inviscid* Euler equations are evolved – in which there is no dissipation at all. Thus the entropy production required by shocks is simply not possible. Note that the oscillations are not only physically wrong, they can even cause negative density or pressure fluctuations in some cells, crashing the code.

One approach to prevent this are so-called slope limiters. In particular, the family of minmod slope limiters is highly successfully used in second-order finite volume methods. While use of them in DG methods is possible, applying them in high order settings by discarding the high-order expansion coefficients whenever the slope limiter kicks in (see Schaal et al., 2015; Guillet et al., 2019) is defeating much of the effort to going to high order in the first place. Somehow constructing less aggressive high-order limiters that can avoid this is a topic that has seen much effort in the literature, but arguably only with still limited success. In fact, the problem of coping with shocks in high-order DG is fundamentally an issue that still awaits a compelling and reasonably simple solution. Recent advanced treatments had to resort to replacing troubled cells with finite volume solutions computed on small grid patches that are then blended with the DG solution (e.g. Zanotti et al., 2015; Markert et al., 2021).

We here return to the idea that this problem may actually be best addressed by resurrecting the old idea of artificial viscosity (Persson & Peraire, 2006). In other inviscid hydrodynamical methods, in particular in the Lagrangian technique of smoothed particle hydrodynamics, it is evident and long accepted that artificial viscosity must be added to capture shocks. Because the conservation laws ultimately dictate the amount of entropy that needs to be created in shocks, the exact procedure for adding artificial viscosity is not overly critical. What is critical, however, is that the there is a channel for dissipation and entropy production. It is also clear that shocks in DG can be captured in a sub-cell fashion only if the required dissipation is provided somehow, either through artificial viscosity that is ideally present only at the place of the shock front itself where it is really needed, or by literally capturing the shock by subjecting the "troubled cell" to a special procedure in which it is, for example, remapped to grid of finite volume cells.

Persson & Peraire (2006) suggested to use a discontinuity (or rather oscillation) sensor to detect the need for artificial viscosity in a given cell. For this, they proposed to measure the relative contribution of the highest order Legendre basis functions in representing the state of the conserved fields in a cell. A solution of a smooth problem is expected to be dominated by the lower order weight coefficients, and statistically the low order weights should be much larger than their high order counterparts. In contrast, for highly oscillatory solutions in a cell (which often are created as pathological side-effects of discontinuities), the high order coefficients are more strongly expressed.

We adopt the same discontinuity sensor as Persson & Peraire (2006). For every cell K, we can calculate the conserved variables $\boldsymbol{u}(\boldsymbol{x})$ using either the full basis in the normal way,

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_{i=1}^{N(p)} \boldsymbol{w}_l^K \phi_l$$

or by omitting the highest order basis functions that are not present at the next lower expansion order, as

$$\hat{\boldsymbol{u}}(\boldsymbol{x}) = \sum_{i=1}^{N(p-1)} \boldsymbol{w}_l^K \phi_l$$

The discontinuity/oscillatory sensor S^K in cell K can now be defined as

$$S^K = \frac{\int_K (u - \hat{u})(u - \hat{u}) \mathrm{d}V}{\int_K u(x)u(x)\,\mathrm{d}V}, \tag{2.41}$$

where we restrict ourselves to one component of the state vector, the density field. Note that due to the orthogonality of our basis functions, this can be readily evaluated as

$$S^K = \frac{\sum_{l=N_{p-1}}^{N_p} [w_l^K]^2}{\sum_{l=1}^{N_p} [w_l^K]^2} \tag{2.42}$$

in terms of sums over the squared expansion coefficients. While we have $0 \leq S^K \leq 1$, we expect S^K to generally assume relatively small values even if significant oscillatory behaviour is already present in K, simply because the natural magnitude of the expansion coefficients declines with their order rapidly. Persson & Peraire (2006) argue that the coefficients should scale as $1/p^2$ in analogy with the scaling of Fourier coefficients in 1D, so that typical values for S^K in case oscillatory solutions are present may scale as $1/p^4$. Our tests indicate a somewhat weaker scaling dependence, however, for oscillatory solutions developing for identical ICs, where the troubled cells scale approximately as $S^K \sim 1/p^2$ as a function of order.

In the approach of Persson & Peraire (2006), artificial viscosity is invoked in cells once their S^K value exceeds a threshold value, above which it is ramped up smoothly as a function of S^K to a predefined maximum value. While this approach shows some success in controlling shocks in DG, it is problematic that strong oscillations need to be present in the first place *before* the artificial viscosity is injected to damp them. In a sense, some damage must have already happened before the fix is applied.

For capturing shocks we therefore argue it makes more sense to resort to a physical shock sensor which detects rapid, non-adiabatic compressions in which dissipation should occur. We therefore propose here to adapt ideas widely used in the SPH literature (Morris & Monaghan, 1997; Cullen & Dehnen, 2010), namely to consider a time-dependent artificial viscosity field that is integrated in time using suitable source and sink functions. Adopting a dimensionless viscosity strength $\alpha(\boldsymbol{x}, t)$, we propose the evolutionary equation

$$\frac{\partial \alpha}{\partial t} = \dot{\alpha}_{\mathrm{shock}} + \dot{\alpha}_{\mathrm{wiggles}} - \frac{\alpha}{\tau} \tag{2.43}$$

for steering the spatially and temporarily variable viscosity. For the moment we use a simple shock sensor $\dot{\alpha}_{\mathrm{shock}} = f_v \max(0, -\boldsymbol{\nabla} \cdot \boldsymbol{v})$ based on detecting compression, where $f_v \sim 1.0$ can be modified to influence how rapidly the viscosity should increase upon strong compression. In the absence of sources, the viscosity decays exponentially on a timescale

$$\tau = f_\tau \frac{h}{p\, c_s}, \tag{2.44}$$

where h/p is the expected effective spatial resolution at order p, c_s is the local sound speed, and $f_\tau \sim 0.5$ is a user-controlled parameter for setting how rapidly the viscosity decays again after a shock transition.

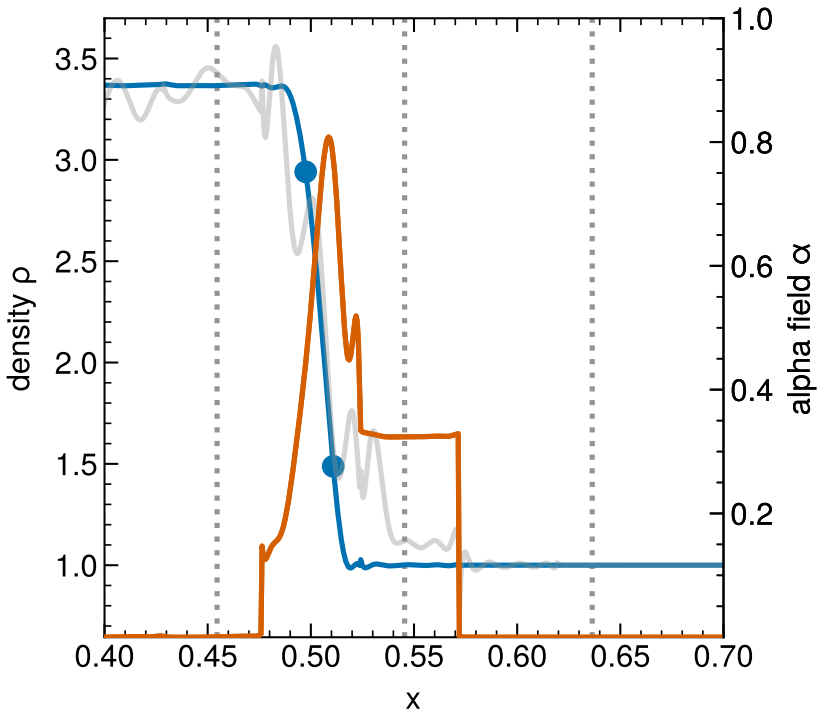


Figure 2.3: Zoom into a Mach number $\mathcal{M} = 4$ shock that is simulated with order $p = 9$. The upstream gas has unit density and unit pressure. Individual mesh cell boundaries are indicated with dotted lines. The density field obtained with artificial viscosity included is shown as a solid blue, while the result without artificial viscosity is shown as a grey line in the background. The artificial viscosity field itself is shown as orange line (scale on the right). The analytic shock position at the displayed time is at $x = 0.5$, in the middle of one of the mesh cells. The circles mark the locations where the density has reached 20 and 80 percent, respectively, of the shock's density jump. We use the distance Δx_{shock} of the corresponding points as a measure of the shock width.

Finally, the term $\dot{\alpha}_{\text{wiggles}}$ in Equation (2.43) is a further source term added to address the occurrence of oscillatory behaviour away from shocks. In fact, this typically is seeded directly ahead of strong shocks, for example when the high-order polynomials in a cell with a shock trigger oscillations in the DG cell directly ahead of the shock through coupling at the interface. Another typical situation where oscillations can occur are sharp, moving contact discontinuities. Here the shock sensor would not be effective in supplying the needed viscosity as there is no shock in the first place. We address this problem by considering the *rate of change* of the oscillatory senor S^K as a source for viscosity, in the form

$$\dot{\alpha}_{\text{wiggles}} = f_w \max\left(0, \frac{\mathrm{d}\log S^K}{\mathrm{d}t}\right), \tag{2.45}$$

for $S^K > S_{\text{onset}}$, otherwise $\dot{\alpha}_{\text{wiggles}} = 0$. When $\mathrm{d}\log S^K/\mathrm{d}t$ is positive and large, oscillatory behaviour is about to grow and the cell is on its way to become a troubled cell, indicating

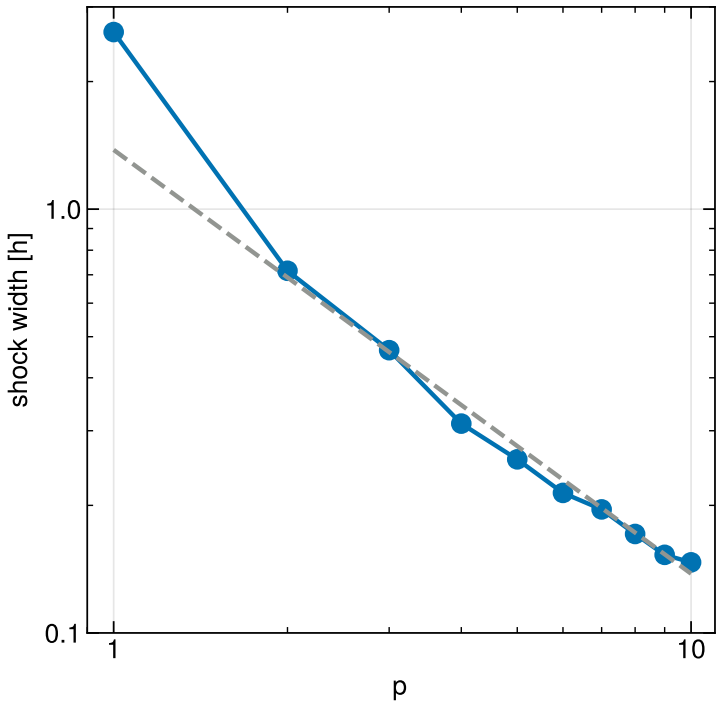


Figure 2.4: Shock width in units of the cell size as a function of the order p of our DG code, for a Mach number $\mathcal{M} = 4$ shock that runs into gas at rest. The dashed line marks a $\Delta x_{\text{shock}} \propto 1/p$ power law, which accurately describes our measurements, except for the lowest order result with piece-wise constant states, which is so highly diffusive that it does not require any artificial viscosity.

that this should better be prevented with local viscosity. In this way, oscillatory solutions can be much more effectively controlled than waiting until they already reached a substantial size. It is nevertheless prudent to restrict the action of this viscosity trigger to cells that have S_K above a minimum value S_{onset}, otherwise the code would try to suppress even tiny wiggles, which would invariably lead to very viscous behaviour. In practice, we set $S_{\text{onset}} = 10^{-4}/p^2$, and we compute $\mathrm{d}\log S^K/\mathrm{d}t$ based on the time derivatives of the weights of the previous timestep.

We add α as a further field component to our state vector $\boldsymbol{u}$, meaning that it is spatially variable and is expanded in our set of basis functions. We do not advect the α field with the local flow velocity as to allow it to fall behind moving discontinuities and to fully suppress any excited oscillations there. Also, advecting the α field at high order would require a limiting scheme for this field itself. Note that in the post-/pre-shock region we can assume the first term of Eq. (2.43) to be unimportant. Once the wiggles are suppressed the second term disappears as well, so that then the default choice of parameters suppresses any existing α field to percent level in a handful of time steps. Only the shock sensor source function is actually variable in a cell, whereas our oscillatory sensor affects the viscosity throughout a cell.

Finally, the actual viscosity applied in the viscous flux of Eqn. (2.37) is parameterized

as

$$\epsilon = \alpha c_s \frac{h}{p}, \tag{2.46}$$

and we impose a maximum allowed value of $\alpha_{\max} = 1$, primarily as a means to prevent overstepping and making the scheme violate the the von Neumann stability requirement for explicit integration of the diffusion equation, which would cause immediate numerical instability. Since our timestep obeys the Courant condition, this is fortunately not implying a significant restriction for effectively applying the artificial viscosity scheme, but it imposes an upper bound that can be used safely without making the time-integration unstable.

We have found that the above parameterisation works quite reliably, injecting viscosity only at discontinuities and when spurious oscillations need to be suppressed, while at the same time not smoothing out solutions excessively. Figure 2.3 shows an example for a Mach number $\mathcal{M} = 3$ shock that is incident from the left on gas with unit density and unity pressure, and adiabatic index $\gamma = 5/3$. The simulation has been computed at order $p = 9$, and at the displayed time, the shock position should be at $x = 0.5$, for a mesh resolution of $h = 1.0/21$. We show our DG result as a thick blue line, and also give the viscosity field $\alpha(x)$ as a red line. Clearly, the shock is captured at a fraction of the cell size, with negligible ringing in the pre- and post-shock regions. This is achieved thanks to the artificial viscosity, which peaks close to the shock center, augmented by additional weaker viscosity in the cell ahead of the shock, which would otherwise show significant oscillations as well. This becomes clear when looking at the solution without artificial viscosity, which is included as a grey line in the background.

The blue circles in Fig. 2.3 mark the places in which the solution has reached 20 and 80 percent of the height of the shock's density jump. We can operationally define the difference in the corresponding x-coordinates as the width Δx_{shock} with which the shock is numerically resolved. In Figure 2.4 we show measurements of the shock width for the same set-up, except for varying the employed order p. We see that the shock width declines with higher order, accurately following the desired relationship $\Delta x_{\text{shock}} \propto 1/p$, except for the lowest order $p = 1$, which deviates towards broader width compared to the general trend. The importance of this result for the DG approach can hardly be overstated, given that it has been a nagging problem for decades to reliably capture shocks at sub-cell resolution in DG without having to throw away much of the higher resolution information. The result of Figure 2.4 essentially implies that shocks are resolved with the same width for a fixed number of degrees of freedom, independent of the employed order p. Whereas using higher order at a fixed number of degrees of freedom is thus not providing much of an advantage for making shocks thinner compared to using more cells, it at least does not degrade the solution. But smooth parts of a solution can then still benefit from the use of higher order.

In total our artificial viscosity method uses five parameters, one for each of the three terms of Eq. (2.43), a further general scaling factor α which is applied to the total viscous flux as defined in Eq. (2.46), and an onset threshold S_{onset}. In this way we are able to individually control the suppression of shocks, wiggles and the decay time of the viscous field as well as the total magnitude of viscous flux. The default values we adopted for these parameters throughout this work are $\alpha = 1.0$, $f_v = 2.5$, $f_\tau = 0.5$, $f_w = 0.2$, and

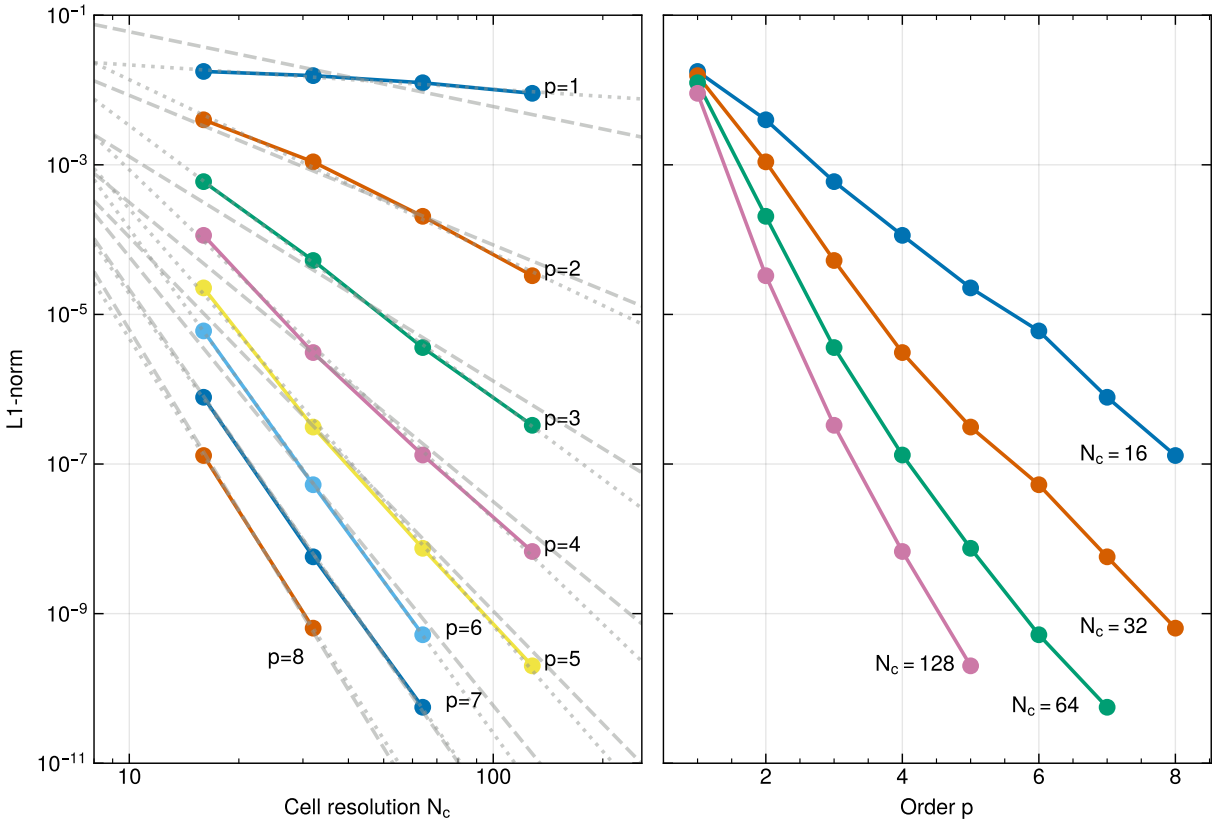


Figure 2.5: Convergence of the Yee et al. (1999, 2000) vortex when evolved for $t = 10.0$ time units. The left panel shows the error norm in the density fields as a function of spatial grid resolution, for 8 different orders p of our DG scheme. The measured convergence orders for $L1$ (colored lines) are close to the expected $L1 \propto N_c^{-p}$ power-laws (dashed grey lines). The actually achieved convergence orders (fitted power-laws, shown as dotted lines) are typically even slightly better than expected, except for the lowest order $p = 1$. The panel on the right-hand side shows the same data, but as a function of DG order p, using a log-linear plot. For fixed grid resolution, the error declines *exponentially* with the order p of the scheme, highlighting the very fast improvement of accuracy when the DG order is increased. We note that the imposed periodic boundaries for the chosen box size of 10 lead to an edge effect which puts the lower boundary of the L1-norm to $\sim 10^{-11}$.

$S_{\text{onset}} = 10^{-4}$.

2.4.2 Positivity limiter

With our artificial viscosity approach described above we intend to introduce the necessary numerical viscosity where needed, such that slope limiting becomes obsolete. However, for further increasing robustness of our code, it is desirable that it also runs stably if a too weak or no artificial viscosity is specified, or if its strength is perhaps locally not sufficient

for some reason in a particularly challenging flow situation. To prevent a breakdown of the time evolution in this case, we consider an optional positivity limiter following Zhang & Shu (2010) and Schaal et al. (2015). This can be viewed as a kind of last line of defense against the occurrence of oscillations in a solution that ventures into the regime of unphysical values, such as negative density or pressure. The latter can happen even for arbitrarily small timesteps, especially when higher order methods are used where such robustness problems tend to be more acute.

Finite-element and finite-volume hydrodynamical codes typically employ procedures such as slope limiters to cope with these situations, this means they locally reduce the order of the scheme (effectively making it more diffusive) by discarding high-order information. A similar approach is followed by the positivity limiter described here, which is based on Schaal et al. (2015), with an important difference in how we select the evaluation points. We stress however that the positivity limiter is not designed to prevent oscillations, only to reduce them to a point that still allows the calculation to proceed.

For a given cell, we first determine the average density $\overline{\rho}$ in the cell, which is simply given by the 0-th order expansion coefficient for the density field of the given cell, and we likewise determine the average pressure $\overline{p}$ of the cell. If either $\overline{\rho}$ or $\overline{p}$ is negative, a code crash is unavoidable.

Otherwise, we define a lowest permissible density $\rho_{\text{bottom}} = 10^{-6}\bar{\rho}$. Next, we consider the full set of quadrature evaluation points $\{\boldsymbol{x}_i\}$ relevant for the cell, which is the union of the points used for internal volume integrations and the points used for surface integrals on the outer boundaries of the cell. We then determine the minimum density ρ_{min} occurring for the field expansions among these points. In case $\rho_{\text{min}} < \rho_{\text{bottom}}$, which includes the possibility that ρ_{min} is negative, we calculate a reduction factor $f = (\bar{\rho} - \rho_{\text{bottom}}) \,/\, (\bar{\rho} - \rho_{\text{min}})$ and replace all higher order weights of the cell with

$$\boldsymbol{w}_l'^K = f\, \boldsymbol{w}_l^K \quad \text{for} \quad l > 1. \tag{2.47}$$

This limits the minimum density appearing in any of the discrete calculations to ρ_{bottom}. By applying the correction factor f to all fields and not just the density, we avoid to potentially amplify relative fluctuations in the velocity and pressure fields.

We proceed similarly for limiting pressure oscillations, except that here no simple reduction factor can be computed to ensure that p_{min} stays above p_{bottom}, due to the non-linear dependence of the pressure on the energy, momentum and density fields. Instead, we simply adopt $f = 0.5$ and repeatedly apply the pressure limiter until $p_{\text{min}} \geq p_{\text{bottom}}$.

In our test simulations the positivity limiter, as expected, does not trigger for inherently smooth problems and thus is in principle not needed. However, when starting simulations with significant discontinuities in the initial conditions, the positivity limiter usually kicks in at the start for a couple of timesteps, especially for high order simulations, until the artificial viscosity is able to tame the spurious oscillations, making the positivity limiter superfluous in the subsequent evolution.

2.5 Basic tests

In this section we consider a set of basic tests problems that establish the accuracy of our new code both for smooth problems, as well as for problems containing strong discontinuities such as shocks or contact discontinuities. We shall begin with a smooth hydrodynamic problem that is suitable for verifying code accuracy for the inviscid Euler equations. We then turn to testing the diffusion solver of the code, as an indirect means to test the ability of our approach to stably and accurately solve the viscous diffusion inherent in the Navier-Stokes equations. We then consider shocks and the supersonic advection of a discontinuous top-hat profile to verify the stability of our high-order approach when dealing with such flow features. Applications to Kelvin-Helmholtz instabilities and driven turbulence are treated in separate sections.

2.5.1 Isentropic vortex

The isentropic vortex problem of Yee et al. (1999, 2000) is a time-independent smooth vortex flow, making it a particularly useful test for the accuracy of higher-order methods, because they should reach their theoretically optimal spatial convergence order if everything is working well (e.g. Schaal et al., 2015; Pakmor et al., 2016). We follow here the original setup used in Yee et al. (1999) by employing a domain with extension $[-5,5]^2$ in 2D and an initial state given by:

$$v_x(\boldsymbol{r}) = -\frac{\beta y}{2\pi}\exp\left(\frac{1-r^2}{2}\right) \tag{2.48}$$

$$v_y(\boldsymbol{r}) = \frac{\beta x}{2\pi}\exp\left(\frac{1-r^2}{2}\right) \tag{2.49}$$

$$u(\boldsymbol{r}) = 1-\frac{\beta^2}{8\gamma\pi^2}\exp\left(1-r^2\right) \tag{2.50}$$

$$\rho(\boldsymbol{r}) = [(\gamma-1)u(\boldsymbol{r})]^{\frac{1}{\gamma-1}} \tag{2.51}$$

where we choose $\gamma = 1.4$, and $\beta = 5$. We evolve the vortex with different DG expansion order n and different mesh resolutions $N^2_{\rm grid}$ until time $t = 10$, and then measure the resulting L1 approximation error of the numerical result for the density field relative to the analytic solution (which is identical to the initial conditions). In order to make the actual measurement of $L1$ independent of discretization effects, we use $n+2$ Gaussian quadrature for evaluating the volume integral appearing in Eq. (2.12). Likewise, we use this elevated order when projecting the initial conditions onto the discrete realization of DG weights of our mesh.

In Figure 2.5 we show measurements of the L1 error as a function of grid resolution $N_{\rm grid}$, for different expansion order from $p = 1$ to $p = 8$. The left panel shows that the errors decrease as power laws with spatial resolution for fixed n, closely following the expected convergence order $L1 \propto N^{-p}_{\rm grid}$ in all cases (except for the $p = 1$ resolution, which exhibits

slightly worse behavior – but this order is never used in practice because of its dismal convergence properties).

Interestingly, the data also shows that for a given grid resolution, the L1 error goes down *exponentially* with the order of the scheme. This is shown in the right panel of Fig. 2.5, which shows the L1 error in a log-linear plot as a function of order p, so that exponential convergence manifests in a straight decline. This particularly rapid decline of the error with p for smooth problems makes it intuitively clear that it can be advantageous to go to higher resolution if the problem at hand is free of true physical discontinuities.

2.5.2 Diffusion of a Gaussian pulse

To test our procedures for simulating the diffusion part of our equations, in particular our treatment for estimating surface gradients at interfaces of cells, we first consider the diffusion of a Gaussian pulse, with otherwise stationary gas properties. For simplicity, we consider gas at rest and with uniform density and pressure, and we consider the evolution of a small Gaussian concentration of a passive tracer dye under the action of a constant diffusivity.

For definiteness, we consider a tracer concentration $c(\vec{x}, t)$ given by

$$c(\vec{x}, t) = c_b + \sum_{\vec{j}} \frac{c_g}{2\pi\sigma^2} \exp\left(-\frac{(\vec{x} - \vec{j})^2}{2\sigma^2}\right), \quad \text{with } \sigma^2 = 2\eta t, \tag{2.52}$$

placed in a unit domain $[-0.5, 0.5]^2$ in 2D with periodic boundary conditions. Here the sum over $\vec{j}$ effectively accounts for a Cartesian grid of Gaussian pulses spaced one box size apart in all dimensions to properly take care of the periodic boundary conditions. If we adopt a fixed diffusivity η and initialize $c(\vec{x}, t)$ at some time t_0, then the analytic solution of equation (2.40) tells us that eqn. (2.52) will also describe the dye concentration at all subsequent times $t > t_0$.

For definiteness, we choose $\eta = 1/128$, $c_b = 1/10$, $c_g = 1$, and $t_0 = 1$, and examine the numerically obtained results at time $t = t_0 + 3 = 4$ by computing their L1 error norm with respect to the analytic solution. In the top panel of Fig. 2.6, we show the convergence of this diffusion process as a function of the number of grid cells used, for the first five DG expansion orders. Reassuringly, the L1 error norm decays as a power-law with the cell size, in each case with the expected theoretical optimum $L1 \propto N_{\text{cells}}^{-p}$. This shows that our treatment of the surface derivatives is not only stable and robust, but is also able to deliver high-order convergence.

The bottom panel of Figure 2.6 shows that this also manifests itself in an exponential convergence as function of DG expansion order when the mesh resolution is kept fixed. For this result, we adopted $N_{\text{cells}} = 8$ and went all the way to 10-th order.

While these results do not directly prove that our implementation is able to solve the full Navier-Stokes equations at high-order, they represent an encouraging prerequisite. Also, we note that both the version without viscous source terms (i.e. the Euler equations), as well as the viscous term itself when treated in isolation converges at high order. We will later

on compare to a literature result for the Kelvin-Helmholtz instability in a fully viscous simulation to back up this further and to test a situation where the full Navier-Stokes equations are used.

2.5.3 Double blast wave

To test the ability of our DG approach to cope with strong shocks, particularly at high order, we look at the classic double blast wave problem of Woodward & Colella (1984). The initial conditions are defined in the one-dimensional domain $[0, 1]$ for a gas of unit density and adiabatic index $\gamma = 7/5$, which is initially at rest. By prescribing two regions of very high pressure, $P = 1000$ for $x < 0.1$, and $P = 100$ for $x > 0.9$, in an otherwise low-pressure $P = 0.1$ background, the time evolution is characterized by the launching of very strong shock and rarefaction waves that collide and interact in complicated ways. Because of the difficulty of this test for shock-capturing approaches, it has often been studied in previous work to examine code accuracy and robustness (e.g. Stone et al., 2008; Springel, 2010).

In order to highlight differences due to different DG orders, we have run deliberately low-resolution realizations of the problem, using 100 cells of equal size within the region $[0, 1]$. We have then evolved the initial conditions with order$p = 2$, $p = 4$, or $p = 8$. Furthermore, we examine a run done with four times as many cells carried out at order $p = 2$. This latter simulation has the same number of degrees of freedom as the $p = 8$ simulation, and thus should have a similar effective spatial resolution. For comparison purposes, we use a simulation carried out with 10000 cells at order $p = 2$, which can be taken as a result close to a converged solution. All simulations were run with our artificial viscosity implementation using our default settings for the method (which do not depend on order p).

In Figure 2.7, we show the density profile at the time $t = 0.038$, as done in many previous works, based on our 100 cell runs. Clearly, the shock fronts and contact discontinuities of the problem are quite heavily smoothed out for the $p = 2$ run with 100 cells, due to the low resolution of this setup. However, the quality of the result can be progressively improved by going to higher order while keeping the number of cells fixed, as seen by the results for $p = 4$ and $p = 8$. This is in itself important. It shows that even problems dominated by very strong physical discontinuities are better treated by our code when higher order is used. The additional information this brings is not eliminated by slope-limiting in our approach, thanks to the sub-cell shock capturing allowed by our artificial viscosity technique.

Finally, in Figure 2.8 we compare the $p = 7$, 100 cell result to the $p = 1$ result using 400 cells. Recall that the order of the method is $p+1$ and the total number of degrees of freedom of the two simulations is therefore the same. We find essentially the same quality of the results, which is another important finding. This demonstrates that to first order only the number of degrees of freedom per dimension is important for determining the ability of our DG code to resolve shocks. Putting degrees of freedom into higher expansion order instead of into a larger number of cells is thus not problematic for representing shocks. At the same, it also does not bring a clear advantage for such flow structures. This is because shocks are

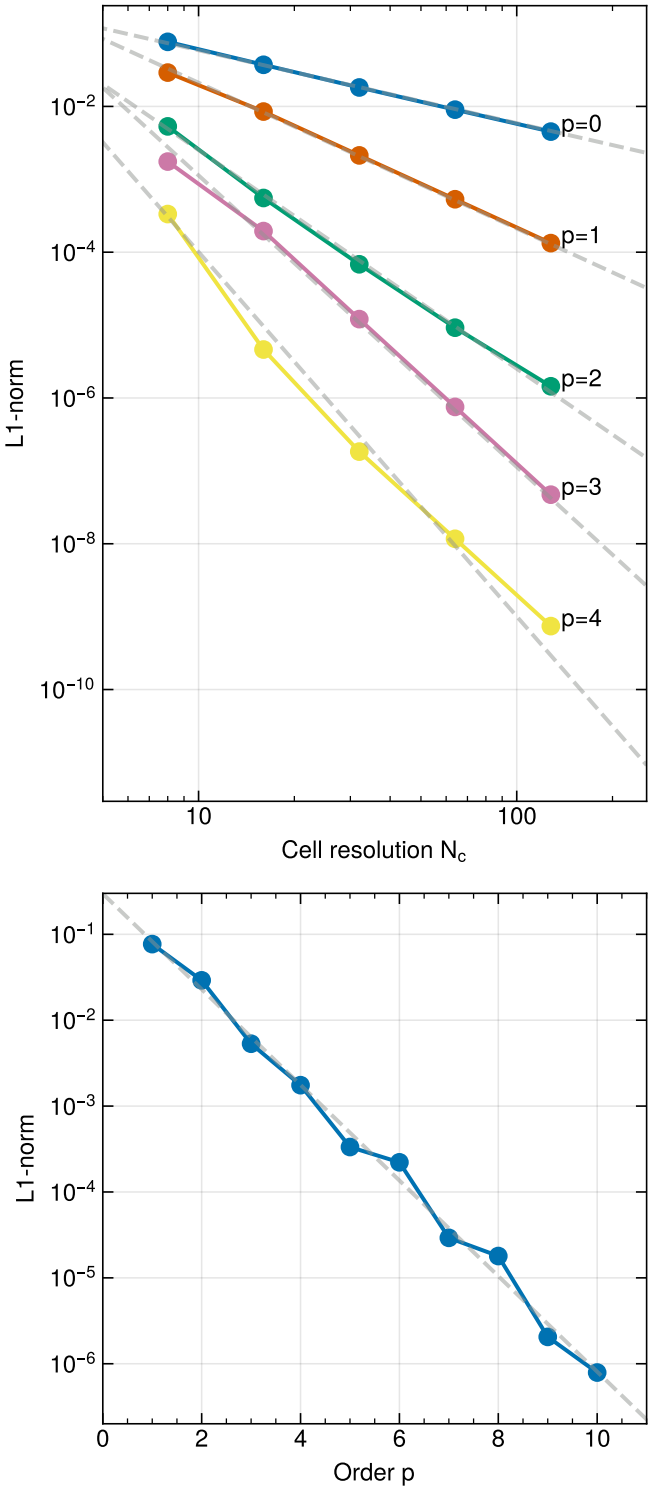


Figure 2.6: Convergence of the diffusion process of a Gaussian profile when started from a smooth state. The top panel shows results for runs carried out at different mesh resolution N_c and DG expansion order p, as labelled. For fixed expansion order, the L1 error declines as a power law as a function of the spatial grid resolution, with the slope of the the expected convergence rate. In the bottom panel, we show the error as a function of order at a fixed grid resolution of $N_c = 8$. In this case, the error declines exponentially as a function of the expansion order.

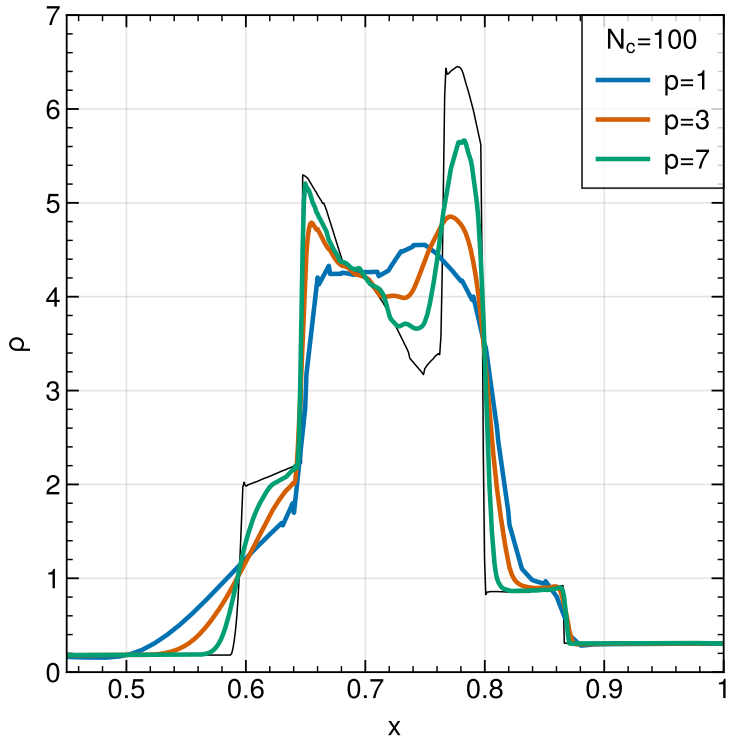


Figure 2.7: Double blast wave problem at fixed spatial resolution, but for increasing DG order. This shows clearly that our new artificial viscosity method can cope with strong shocks, and that adding higher order information is still worthwhile in treating problems with strongly interacting shocks. For reference, a high resolution result with $N_c = 10000$, $p = 1$ is shown as thin black line.

ultimately always broadened to at least the spatial resolution limit. Real discontinuities therefore only converge with 1st order in spatial resolution, and high-order DG schemes do not provide a magic solution for this limitation as their effective resolution is set by the degrees of freedom. Still, as our results show, DG can be straightforwardly applied to problems with strong shocks using our artificial viscosity formulation. When there is a mixture of smooth regions and shocks in a flow, the smooth parts can still benefit from the higher order accuracy while the shocks are rendered with approximately the same accuracy as done with a second-order method with the same number of degrees of freedom.

It is interesting to compare our results to other results in the literature. First we compare to an older DG implementation with a moment limiter by Krivodonova (2007). At low orders their mode by mode limiter performs marginally better than our implementation, but as it was pointed by Vilar (2019) the mode by mode limiting does not work well for higher orders. In contrast, our method remains stable and offers steadily improving results as the order is increased. Vilar (2019) uses a DG implementation with *a posteriori* limiting where troubled cells are detected at the end of the time-step and then recomputed using a finite-volume method and flux reconstruction. Their approach is also able to resolve the

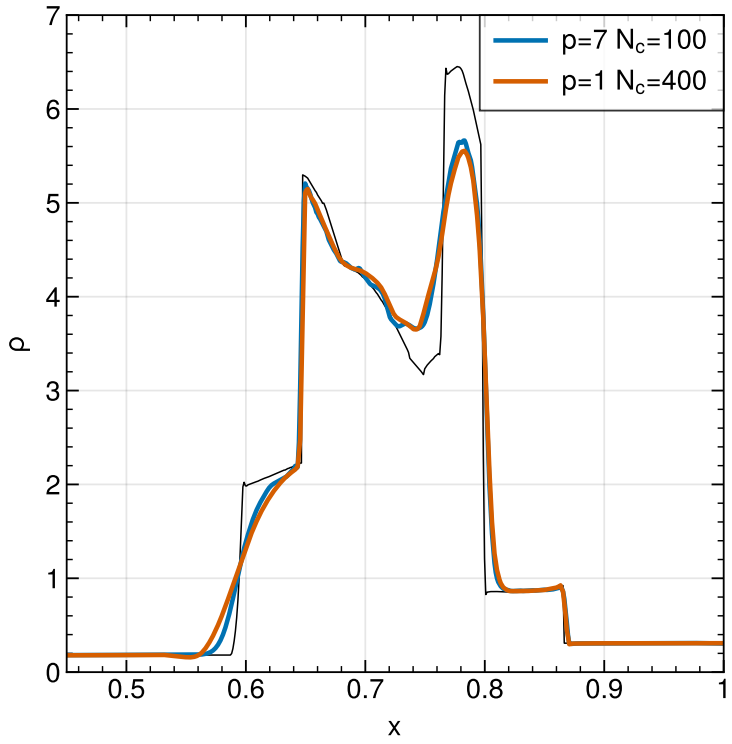


Figure 2.8: Double blast wave problem at fixed number of degrees of freedom for two different combinations of order and spatial resolution. This shows that for strong shocks the total number of degrees of freedom determines accuracy of our solution. For reference, a high resolution result with $N_c = 10000$, $p = 1$ is shown as thin black line.

complicated interactions of shocks and rarefaction waves and yields a steadily improving result with higher resolution as well. To demonstrate that our DG implementation is competitive with state-of-art weighted essentially non-oscillatory (WENO) schemes we compare our results with those reported by Zhao et al. (2017). They simulated this problem using an 8-th order WENO scheme with 400 grid cells. The WENO implementation performs here somewhat better at a given number of degrees of freedom compared to our DG method. Note that the number of degrees of freedom in a WENO scheme is order independent and therefore our $p = 7$ run at at 100 cells has twice the number of degrees of freedom as their 8-th order run with 400 cells, yet their result is closer to the ground truth than ours.

2.5.4 Advection of a top-hat pulse

Next we consider the problem of super-sonically advecting a strong contact discontinuity in the form of an overdense square that is in pressure equilibrium with the background. This tests the ability of our code to cope with a physical discontinuity that is not self-steeping, unlike a shock, i.e. once the contact discontinuity is (excessively) broadened by numerical

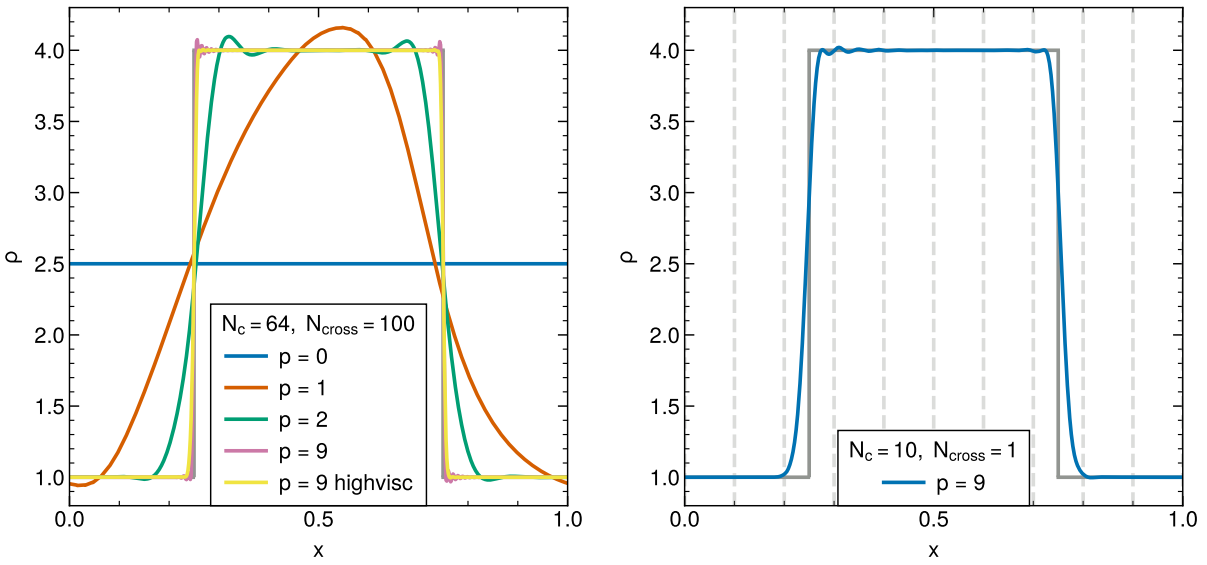


Figure 2.9: *Top panel:* Square advection problem at $t = 1.0$ for different expansion orders p using 64 grid cells in each case. At this time, the top hat profile has been advected 100 times through the box. The initial profile, which is the analytic solution in this case, is shown as a solid grey line in the background. Different numerical results are given for polynomial orders $p = 0, 1, 2$, and $p = 9$, as well as for $p = 9$ with a higher artificial viscosity setting for stronger wiggle suppression. *Bottom panel:* Square advection problem at $t = 0.01$ for $p = 9$ using 10 grid cells. The profile has been advected through the box once. The dotted vertical lines indicate grid cell borders. Sub-cell shock capturing can be observed.

viscosity, it will invariably retain the acquired thickness. In fact, the advection errors inherently present in any Eulerian mesh-based numerical method will continue to slowly broaden a moving contact discontinuity with time, in contrast to Lagrangian methods, which can cope with this situation in principle free of any error.

A problem that starts with a perfectly sharp initial discontinuity furthermore tests the ability of our DG approach to cope with a situation where strong oscillatory behaviour is sourced in the higher order terms, an effect that is especially strong if the motion is supersonic and the system's state is characterized by large discontinuities. Here any naive implementation that does not include any type of limiter or artificial viscosity terms will invariably crash due to the occurrence of unphysical values for density and/or pressure. The square advection problem is thus also a sensitive stability test for our high-order Discontinuous Galerkin method.

In our test we follow the setup-up of Schaal et al. (2015), but see also Hopkins (2015) for a discussion of results obtained with particle-based Lagrangian codes. In 2D, we consider a domain $[0, 1]^2$ with pressure $P = 2.5$ and $\gamma = 7/5$. The density inside the central square of side-length 0.5 is set to $\rho = 4$, and outside of it to $\rho = 1$. A velocity of $v_x = 100$ is added to all the gas, and in the y-direction we add $v_y = 50$. We simulate until $t = 1.0$, at which point

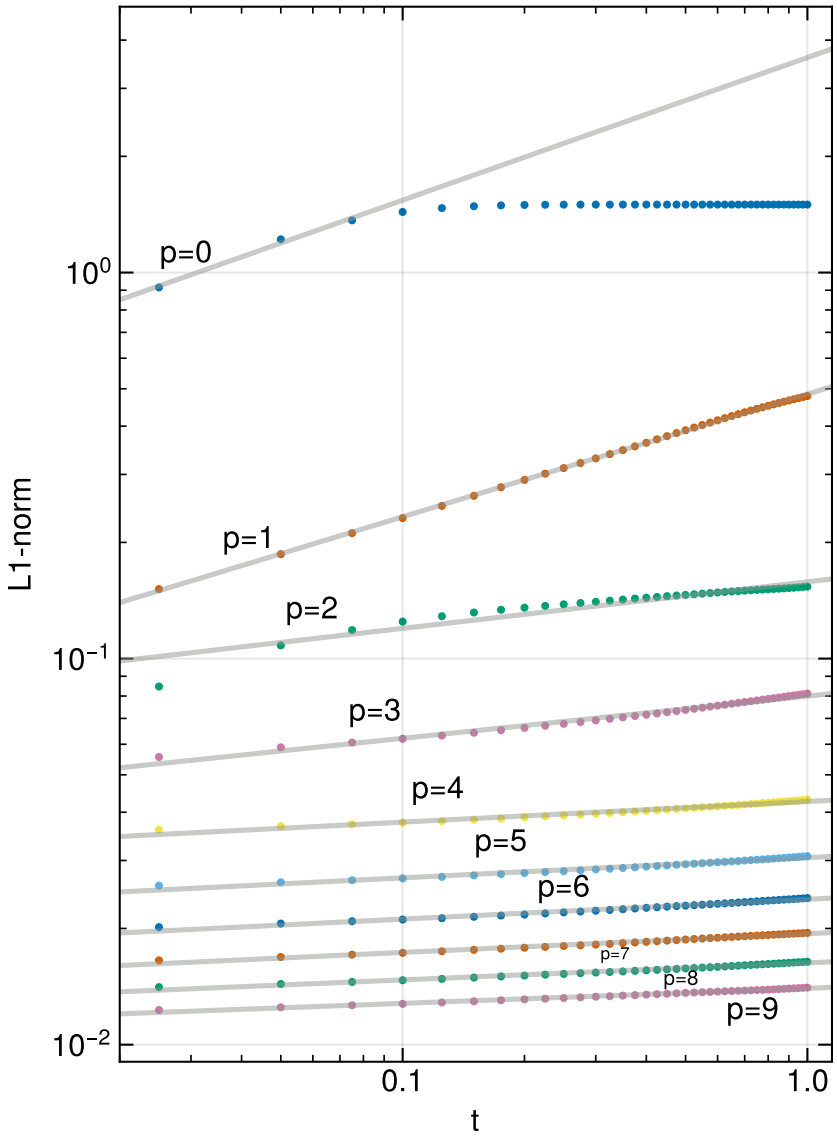


Figure 2.10: Time evolution of the L1 error norm for the density in the square advection problem, calculated for polynomial orders $p = 0$ to $p = 9$ (from top to bottom) using 64 grid cells in each case. The individual measurements for numerical outputs are shown with filled circles, the lines are power-law fits $L1 \propto t^n$. Note that not only the absolute error at any given time declines with increasing order p, also the slopes n become progressively shallower. This means that the numerical diffusivity of the code becomes smaller for higher order, reducing advection errors substantially. The measured slopes n for the $p = 0$ to $p = 9$ cases are in that sequence: 0.427, 0.335, 0.172, 0.056, 0.054, 0.049, 0.048, 0.046, 0.039, and 0.028. In the $p = 0$ case, only the first three points were used to measure the slope.

the pulse has been advected 100 times through the periodic box in the x-direction and half that in the y-direction, and it should have returned exactly to where it started. We have also run the same test problem generalized to 3D, with an additional velocity of $v_z = -25$ in the z-direction, and as well in 1D, where only the motion in the x-direction is present. In general, the multi-dimensional tests behave very similarly to the one-dimensional tests, with the size of the overall error being determined by the largest velocity. For simplicity, we therefore here restrict ourselves in the following to report explicit results for the 1D case only.

In Figure 2.9, we show the density profile of the pulse at $t = 1.0$ when 10 cells per dimension are used, for different DG expansion orders p. A second-order accurate method, $p = 1$, which is equivalent or slightly better than common second-order accurate finite volume methods (see also Schaal et al., 2015) has already washed out the profile substantially at this time. Already order $p = 2$ does substantially better, with $p = 3$ results starting to resemble a top hat profile. The 10th order run with $p = 9$ is able to retain the profile very sharply, albeit with a small amount of ringing right at the discontinuities. Similarly to our results for shocks, we thus find that our code is able to make good use of higher order terms if they are available in the expansion basis. Applying simple limiting schemes such as minmod in the high-order case is in contrast prone to lose much of the benefit of high-order when string discontinuities are present in the simulation, simply because these schemes tend to discard subcell information beyond linear slopes. We also show a $p = 9$ order run with higher artificial viscosity injection in regions with wiggles by using a lower $S_{\text{onset}} = 10^{-6}$. Spurious oscillations get dampened at the cost of a slightly wider shock front.

Comparing our results in the bottom part of Fig. 2.9 to the DG implementation with posteriori correction by Vilar (2019, their figs/subsonic. 5 and 7a), we can see that both methods successfully capture the sharp transition in a sub-cell fashion. For this comparison it is worth noting that in our case the shock is resolved within one cell, while in the setup of Vilar (2019) the discontinuity occurs at the border of two cells.

To look more quantitatively at the errors, we show in Figure 2.10 the L1 error norm of the density field as a function of time, for all orders from $p = 0$ to $p = 9$. We see that the lowest order does very poorly on this problem, due to its large advection errors. In fact, $p = 0$ loses the profile completely after about 10 box transitions, yielding a uniform average density throughout the whole box. When one uses higher order, both the absolute error at any given time but also the rate of residual growth of the error with time drop progressively. The latter can be described by a power-law $L1 \propto t^n$, with a slope n that we measure to be just 0.028 for $p = 9$, while it is still 0.335 for a second-order, $p = 1$ method. The longer a simulation runs, the larger the accuracy advantage of a high-order method over lower-order methods thus becomes.

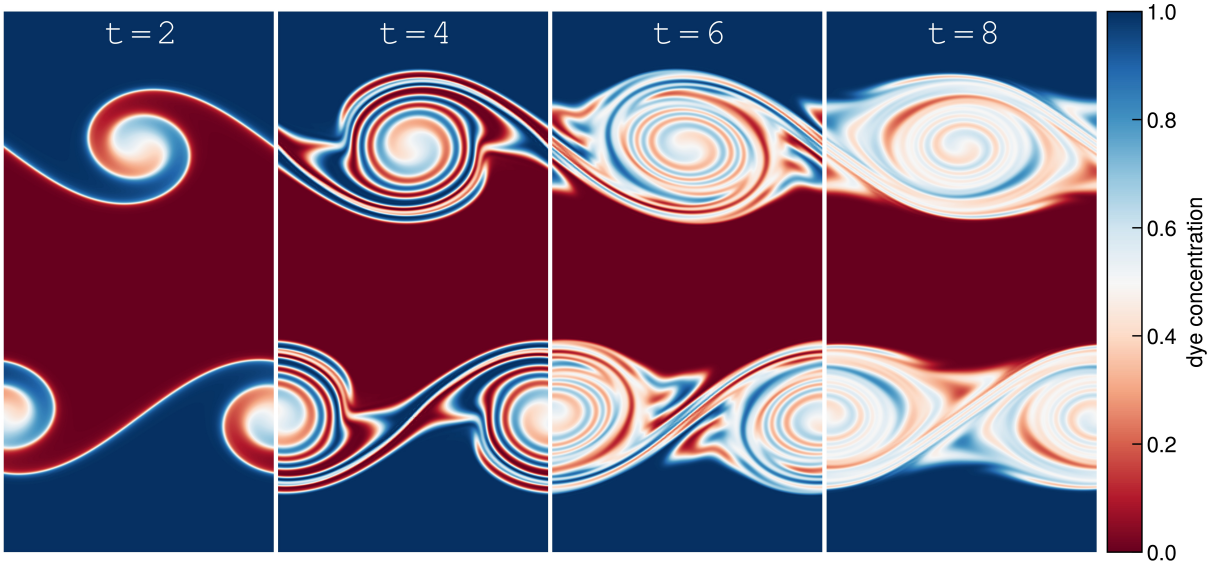


Figure 2.11: Time evolution of the dye concentration in a Kelvin-Helmholtz simulation using 64 DG-cells along the x-range $[0, 1]$, at order $p = 5$, using a viscosity setting of $\mathrm{Re} = 10^5$ and $\Delta\rho/\rho_0 = 0$.

2.6 Kelvin-Helmholtz instabilities

Simulations of the Kelvin-Helmholtz (KH) instability have become a particularly popular test of hydrodynamical codes (e.g. Price, 2008; Springel, 2010; Junk et al., 2010; Valcke et al., 2010; Cha et al., 2010; Berlok & Pfrommer, 2019; Tricco, 2019; Borrow et al., 2022), arguably initiated by an influential comparison of SPH and Eulerian codes by Agertz et al. (2007), where significant discrepancies in the growth of the perturbations in different numerical methods had been identified. One principal complication, however, is that for initial conditions with an arbitrarily sharp discontinuity the non-linear outcome is fundamentally ill-posed at the discretized level (e.g. Robertson et al., 2010; McNally et al., 2012), because for an ideal gas the shortest wavelengths have the fastest growth rates, so that one can easily end up with KH billows that are seeded by numerical noise at the resolution limit, rendering a comparison of the non-linear evolution between different methods unreliable. Furthermore, in the inviscid case, the non-linear outcome is fundamentally dependent on the numerical resolution so a converged solution does not even exist.

Lecoanet et al. (2016) have therefore argued that using smooth initial conditions across the whole domain combined with an explicit physical viscosity is a better choice, because this allows in principle converged numerical solutions to be reached. We follow their approach here, and in particular compare to the reference solution determined by Lecoanet et al. (2016) using the spectral code DEDALUS (Burns et al., 2020) at high resolution.

Specifically, following Lecoanet et al. (2016) we adopt as initial conditions a shear flow

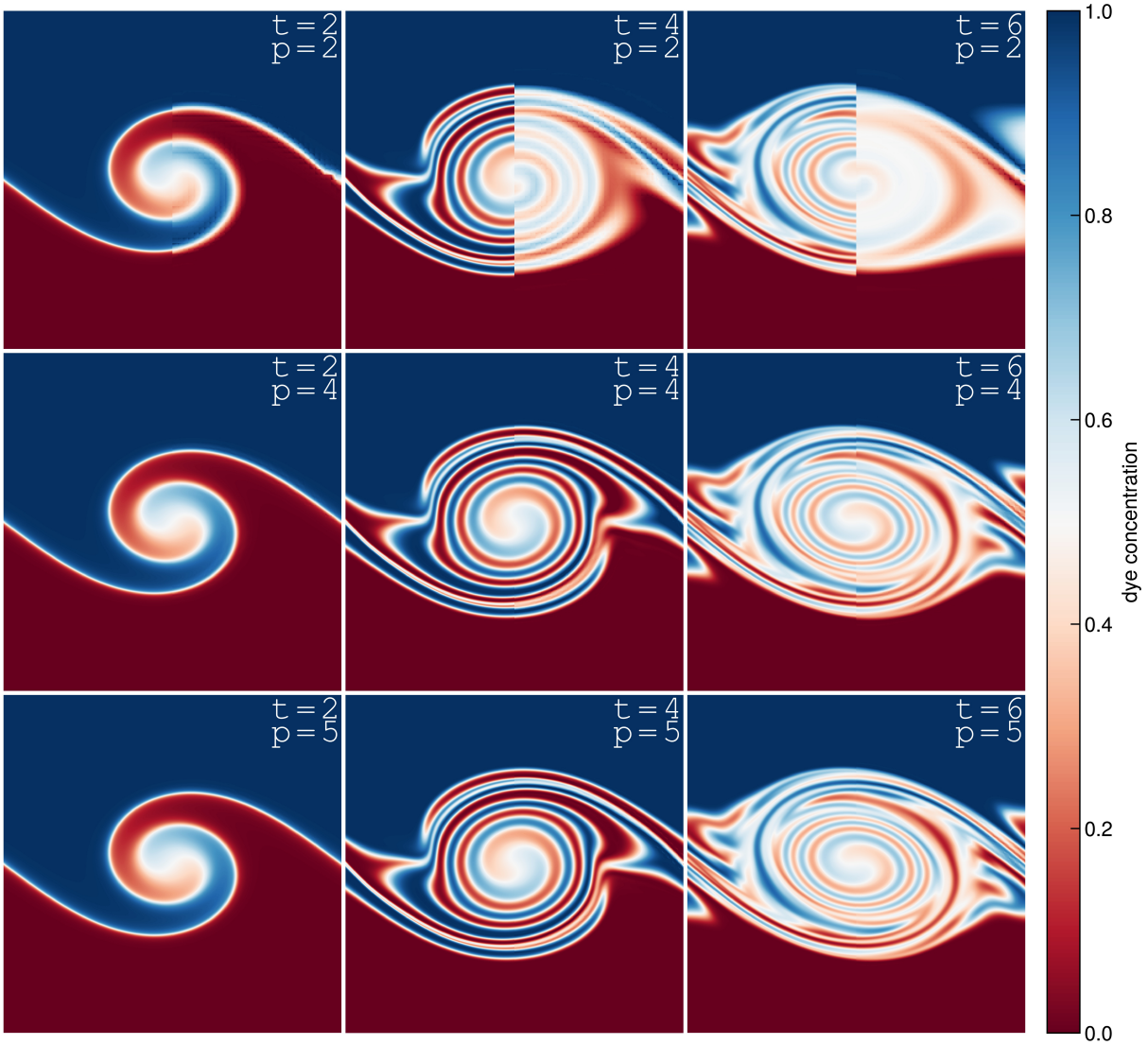


Figure 2.12: Dye concentration in Kelvin-Helmholtz simulations, using $\mathrm{Re} = 10^5$ and $\Delta\rho/\rho_0 = 0$, compared at fixed grid resolution but different times t and order p. Each of the nine panels shows the high-resolution DEDALUS reference result (Lecoanet et al., 2016) in the left half, and our DG result (at different order p as labelled) in the right half. All DG simulations were done with $N_c = 64$ grid cells.

with a smoothed density and velocity transition:

$$\begin{aligned} \rho(x,y) &= 1 + \frac{\Delta\rho}{\rho_0}\frac{1}{2}\left[\tanh\left(\frac{y-y_1}{a}\right) - \tanh\left(\frac{y-y_2}{a}\right)\right], \\ v_x(x,y) &= u_{\mathrm{flow}}\left[\tanh\left(\frac{y-y_1}{a}\right) - \tanh\left(\frac{y-y_2}{a}\right) - 1\right], \end{aligned} \tag{2.53}$$

with $u_{\text{flow}} = 1$, $a = 0.05$, $y_1 = 0.5$ and $y_2 = 1.5$ in a periodic domain with side length $L = 2$. This is perturbed with a small velocity component in the y-direction to seed a KH billow on large scales:

$$v_y(x, y) = A \sin(2\pi x) \left[\exp\left(-\frac{(y - y_1)^2}{\sigma^2} \right) + \exp\left(-\frac{(y - y_2)^2}{\sigma^2} \right) \right], \tag{2.54}$$

where $A = 0.01$ and $\sigma = 0.2$ is chosen. The pressure is initialized everywhere to a constant value, $P(x, y) = P_0$, with $P_0 = 10$. With these choices, the flow stays in the subsonic regime with a Mach number $\mathcal{M} \sim 0.25$.

The free parameter $\Delta\rho/\rho_0$ describes the presence or absence of a density "jump" across the two fluid phases that stream past each other. By adding a passive tracer field

$$c(x, y) = \frac{1}{2} \left[\tanh\left(\frac{y - y_2}{a} \right) - \tanh\left(\frac{y - y_1}{a} \right) + 2 \right] \tag{2.55}$$

to the initial conditions, we can study the KH instability also easily for the case of a vanishing density jump. In fact, we shall focus on the case $\Delta\rho/\rho_0 = 0$ here as it is free of the particularly subtle inner vortex instability in the late non-linear evolution of the KH problem (Lecoanet et al., 2016), which further complicates the comparison of different codes.

To realize the above initial conditions we evaluate them within each cell of our chosen mesh at multiple quadrature points in order to project them onto our DG basis. We perform this initial projection using a Gauss integration that is 2 orders higher than that employed in the run itself. This ensures that integration errors from the projection of the initial conditions onto our DG basis are subdominant compared to the errors incurred by the time evolution, and are thus unimportant.

We choose identical values for shear viscosity ν, thermal diffusivity χ, and dye diffusivity η. Below, we mostly focus on discussing results for a Reynolds number $\text{Re} = 10^5$ for which we set $\nu = \chi = \eta = 2u_{\text{flow}}/\text{Re} = 2 \times 10^{-5}$. We have also carried out simulations with a higher Reynolds number $\text{Re} = 10^6$, obtaining qualitatively similar results, although these simulations require higher resolution for convergence and thus tend to be more expensive.

2.6.1 Visual comparison

A visual illustration of the time evolution of the dye concentration for a simulation with $\text{Re} = 10^5$ and $\Delta\rho/\rho_0 = 0$ is shown in Fig. 2.11. In this calculation, 64 DG cells were used to cover the x-range $[0, 1]$, which is the relevant number to compare to the resolution information in Lecoanet et al. (2016). Expansion order $p = 6$ has been used in this particular run. It is nicely seen that the KH billow seeded in the initial conditions is amplified in linear evolution until a time $t \sim 1 - 2$, then it rolls up multiple times in a highly non-linear evolution, before finally strong mixing sets in that progressively washes out the dye concentration throughout the vortex.

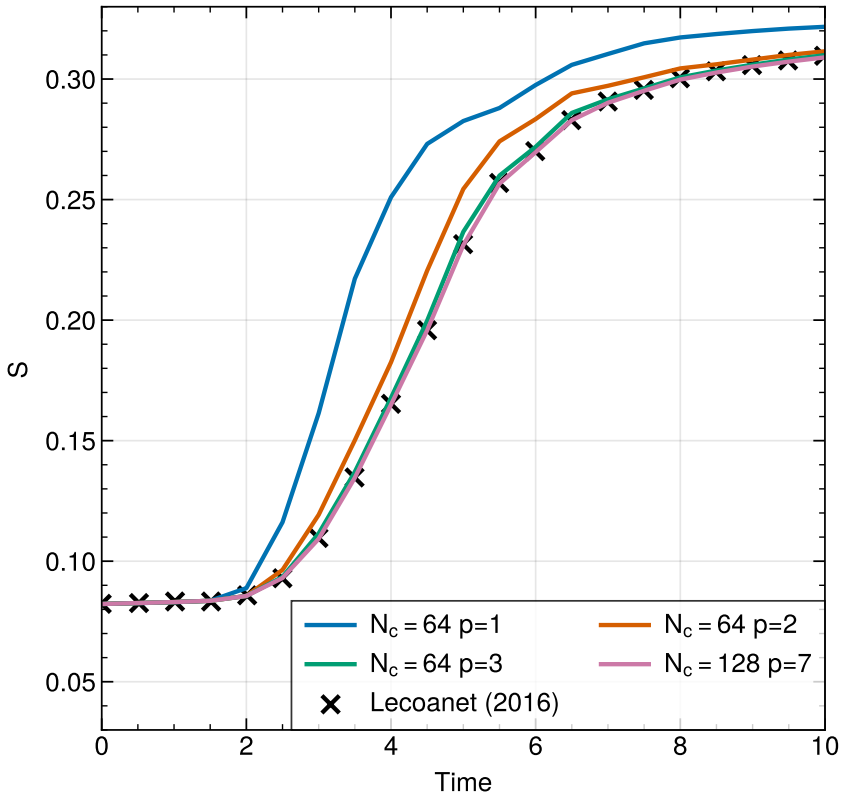


Figure 2.13: Volume integrated dye entropy as a function of time. We show our DG simulation results with 64 cells using orders $p = 1$ to $p = 3$, and a calculation with 128 cells and order $p = 7$. All simulations were ran with $\mathrm{Re} = 10^5$ and a density jump $\Delta\rho/\rho_0 = 0$. Already the run with 64 cells and $p = 3$ shows an essentially converged result; still better resolutions yield perfect agreement with the very high resolution results obtained by Lecoanet et al. (2016) with the DEDALUS and ATHENA codes.

Upon visual inspection, this time evolution compares very closely to that reported by Lecoanet et al. (2016). In Figure 2.12 we make this comparison more explicit by showing results obtained for different order p at a number of times 'face-to-face' with their reference simulation. In each of the panels, the left half of the picture contains the DEDALUS result at resolution 3096×6144, while the right half gives our results at 64×128 resolution, but with different orders p. We have deliberately chosen this modest resolution for this comparison in order to allow some differences to be seen after all. They show up clearly only at second-order in the top row, while at $p = 4$ they are only discernible at times $t = 4$ and $t = 6$ as faint discontinuities at the middle of the images, where the result from DEDALUS meets that from our DG code. Already by $p = 5$, visual inspection is insufficient to identify clear differences. We note that for higher DG grid resolutions, this becomes rapidly extremely difficult already for lower orders.

2.6.2 Dye entropy

An interesting more quantitative comparison of our simulations to those of Lecoanet et al. (2016) can be carried out by considering the evolution of the passive scalar "dye" in some detail. The technical implementation of this passive tracer is described in Section 2.3.4.

A dye entropy per unit mass can be defined as $s = -c \ln c$, and its volume integral is the dye entropy

$$S = \int \rho s \, \mathrm{d}V, \tag{2.56}$$

which can only monotonically increase with time. The dye entropy can be viewed as a useful quantitative measure for the degree of mixing that occurs as a result of the non-linear KH instability. To guarantee an accurate measurement of the dye entropy we perform the integral above at two times higher spatial order than employed in the actual simulation run. We also note than when computing the dye entropy we use our entire simulation domain (although left and right halves give identical values), and we then normalize to half of the volume to make our values directly comparable to those of Lecoanet et al. (2016).

In Figure 2.13, we show measurements of the dye entropy evolution for several of our runs, compared to the converged results obtained by Lecoanet et al. (2016) consistently with the DEDALUS and ATHENA codes. We obtain excellent agreement already for 64 cells and order $p = 4$, corresponding to 256 degrees of freedom per dimension. Our under-resolved simulations with fewer cells and/or degrees of freedom show an excess of mixing and higher dye entropy, as expected.

We note that Tricco (2019) have also studied this same reference problem using SPH. Interestingly, they find that simulations that are carried out at lower resolution than required for (approximate) convergence show an underestimate of dye mixing, marking an important qualitative difference to the mesh-based computations. The SPH simulations also require a substantially higher number of resolution elements to obtain an approximately converged result. Tricco (2019) get close to achieving this for the dye concentration by using 2048 particles per dimension, but even then the dye entropy of their result falls slightly below the converged result at $t = 8$.

2.6.3 Error norm

Finally, we consider a direct comparison of the dye entropy fields obtained in our simulations to the DEDALUS reference solution made publicly[1] available by Lecoanet et al. (2016) at a grid resolution of 3096×6144 points. To perform a quantitative comparison, we consider the L_2-norm of the difference in the dye fields, defined as

$$L_2 = \left[\frac{1}{V} \int \left(c_{\mathrm{DG}} - c_{\mathrm{Lecoanet}} \right)^2 \mathrm{d}V \right]^{1/2}. \tag{2.57}$$

[1] https://doi.org/10.5281/zenodo.5803759

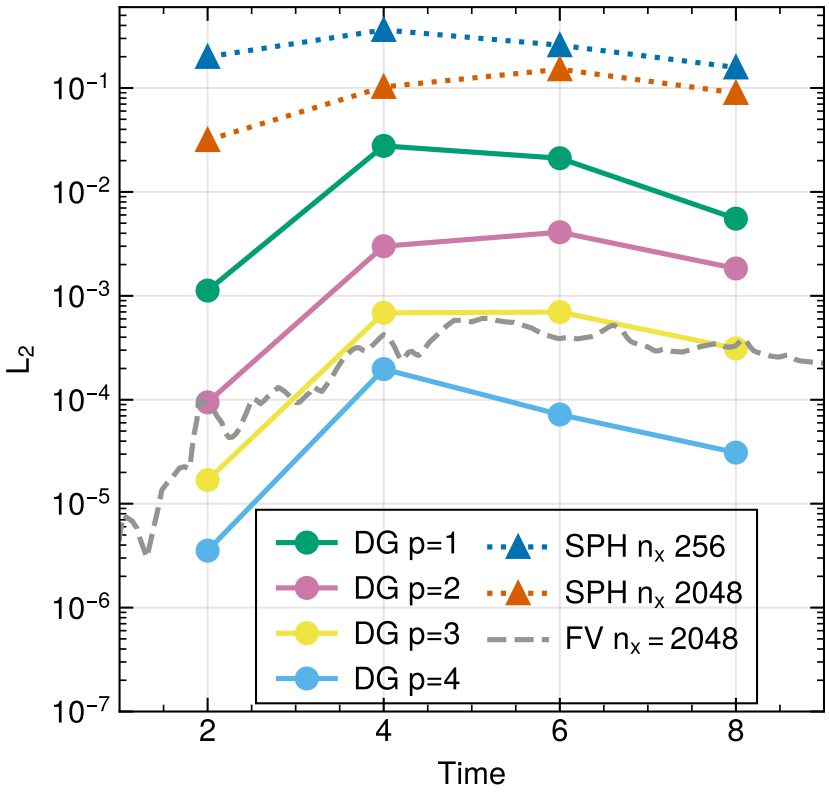


Figure 2.14: Volume-averaged L_2-error norm of the difference in the dye concentration relative to a high-resolution spectral result as a function of time, for a set of DG simulations carried out with 64 cells and different expansion order $p = 1$ to $p = 4$ (as labelled), for $\mathrm{Re} = 10^5$ and a density jump $\Delta\rho/\rho_0 = 0$. The DG results are presented with filled circles at the four available output times of the spectral simulation, the connecting lines are there simply to guide the eye. Similarly, we include SPH results by Tricco (2019) as triangles at two different resolutions. Finally, the dashed line refers to the result obtained by Lecoanet et al. (2016) using the finite-volume code ATHENA with 2048 cells.

In Figure 2.14 we show first the time evolution of the L_2-norm, for DG simulations carried out with 64 cells and orders $p = 2$ to $p = 5$. We also include results reported by Lecoanet et al. (2016) for the ATHENA mesh code at a resolution of 1024 cells, as well as SPH results by Tricco (2019) at particle resolutions of 256 and 2048, respectively. Our $p = 4$ results with 64 cells are already as good as ATHENA with 2048 cells, demonstrating that far fewer degrees of freedom are sufficient when a high order method is used for this smooth problem. In contrast, a relatively noisy method such as SPH really struggles to obtain truly accurate results. Even at the 2048 resolution, the errors are orders of magnitude larger than for the mesh-based methods, and the sluggish convergence rate of SPH will make it incredibly costly, if possible at all, to push the error down to the level of what our DG code, or ATHENA, comparably easily achieve.

In Figure 2.15, we examine the error as a function of the employed DG expansion

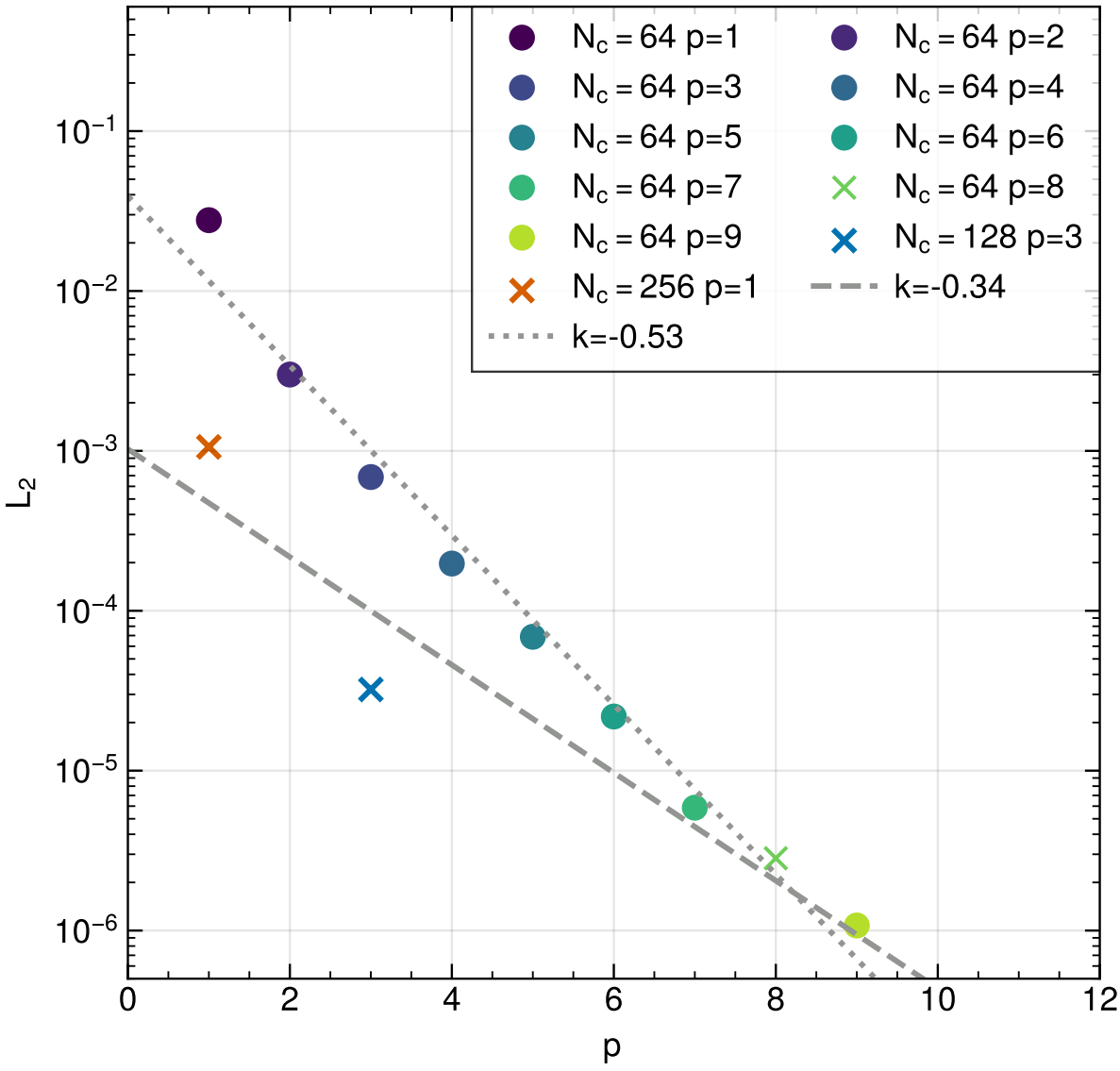


Figure 2.15: Volume-averaged L_2 error norm of the dye concentration field as a function of DG order p for a set of simulations with $\mathrm{Re} = 10^5$ and a density jump $\Delta\rho/\rho_0 = 0$ at $t = 4$. The circles show simulations with $N_c = 64$ cells with progressively increasing order p (the run with $p = 8$ is shown with a cross symbol while still being a member of the sequence of simulations with circles). The crosses highlight three simulations with the same number of degrees of freedom, reached with different combinations of N_c and p. The dotted line is a fit showing the rapid convergence we achieve with increasing order p at $N_c = 64$. The dashed line indicates the convergence rate for three simulations with equal number of degrees of freedom, as we increase the order. Among the three runs with an equal number of degrees of freedom, the one with the highest order p achieves the lowest L_2-norm.

order. For a fixed number $N_c = 64$ of cells, we show the L_2 error at time $t = 4$, for orders $p = 1$ up to $p = 9$. It is reassuring that we again find exponential convergence for this problem, where the error drops approximately linearly with p on this log-linear plot. This demonstrates that we can fully retain the ability to converge at high-order for our compressible Navier-Stokes solver, which is additionally augmented with thermal and dye diffusion processes. We consider this to be a very important validation of our numerical methodology and actual code implementation.

Another interesting comparison is to consider simulations that have an equal number of degrees of freedom, but different cell numbers and expansion orders. In the figure (marked with crosses), we also include results for the three cases $N_c = 64/p = 8$, $N_c = 128/p = 4$, and $N_c = 256/p = 2$, which all have the same number of degrees of freedom per dimension. Strictly speaking, the higher order ones have actually slightly less, given that the number $N^{2\mathrm{D}}(p) = p(p+1)/2$ of degrees of freedom per cell is slightly less than p^2 for $p > 1$, see Equation (2.18). Regardless, the run with $N_c = 64$ clearly has the lowest error. This confirms once more that for a smooth problem it is typically worthwhile in terms of yielding the biggest gain in accuracy to invest additional degrees of freedom into higher order rather than additional cells.

2.7 Driven sub-sonic turbulence

The phenomenon of turbulence describes the notion of an unsteady, random flow that is characterized by the overlap of swirling motions on a variety of scales (e.g. Pope, 2000). In three-dimensions, one finds that if fluid motion is excited on a certain scale (the injection scale) it tends to decay into complex flow features on ever smaller scales, helped by fluid instabilities such as the Kelvin-Helmholtz instability. Eventually, the vortical motions become so small that they are eliminated by viscosity on the so-called dissipation scale. If the injection of kinetic energy on large scales persists and is quasi-stationary, a fully turbulent state develops which effectively exhibits a transport of energy from the injection to the dissipation scale. For incompressible isotropic, subsonic turbulence, the statistics of velocity fluctuations in such a turbulent flow are described by the Kolmogorov velocity power spectrum, which has a power law shape in the inertial range, and a universal shape in the dissipative regime.

For astrophysics, turbulence plays a critical role in many environments, including the intracluster medium, the interstellar medium, or the buoyantly unstable regions in stars. Numerical simulations need to be able to accurately follow turbulent flows, for example in order to correctly describe the mixing of different fluid components, or the amplification of magnetic fields. However, this is often a significant challenge as the scale separation between injection and dissipation scales in astrophysical settings can be extremely large, while for three-dimensional simulation codes it is already difficult to resolve even a moderate difference between injection and dissipation scales. In addition, most astrophysical simulations to date rely on numerical viscosity exclusively instead of including an explicit physical viscosity, something that can in principal modify the shape of the dissipative part

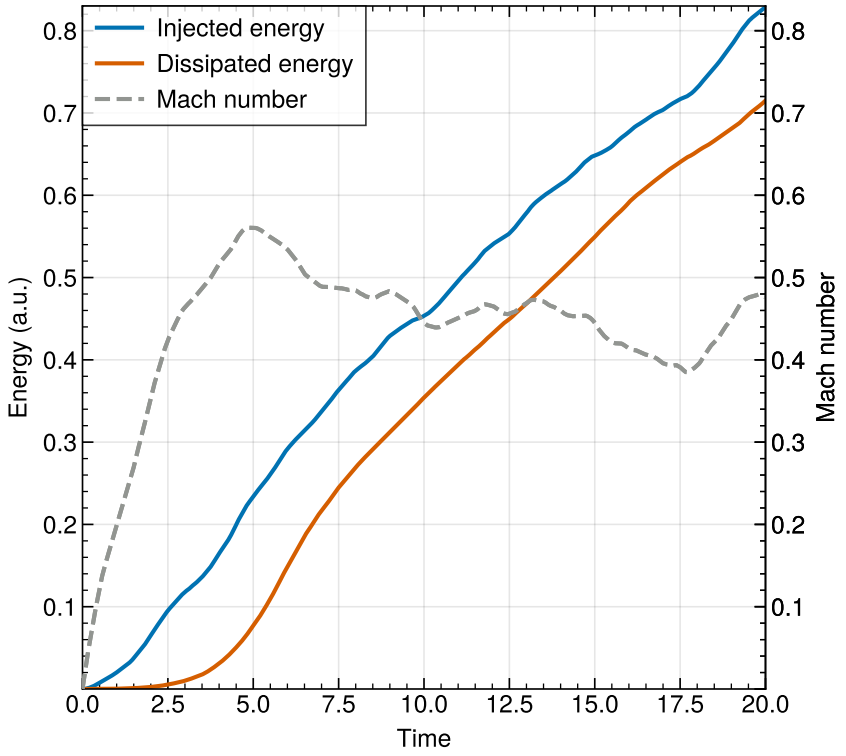


Figure 2.16: Cumulative injected and dissipated energy, as well as global Mach number, as a function of time in one of our driven turbulence simulations. The gas is initially at rest, and put into motion by the driving. Eventually, energy injection is balanced by dissipation in a time-averaged fashion, and the difference between the cumulative injected and dissipated energy is reflected in the kinetic energy as measured by the Mach number.

of the turbulent power spectrum, thereby creating turbulent velocity statistics that differ from the expected universal form because they are directly affected by aspects of the numerical method.

Of course, the general accuracy of a numerical method is also important for how well turbulence can be represented. For example, Bauer & Springel (2012) have pointed out that the comparatively large noise in SPH makes it difficult for this technique to accurately represent subsonic turbulence. While this can in principle be overcome with sufficiently high numerical effort, it is clear that methods that have a low degree of numerical viscosity combined with the ability to accurately account for physical viscosity should be ideal for turbulence simulations. Our DG approach has these features, and especially in the regime of subsonic turbulence, where shocks are expected to play only a negligible role, the DG method should be particularly powerful.

This motivates us to test this idea in this section by considering isothermal, subsonic, driven turbulence in periodic boxes of unit density. The subsonic state refers to the average kinetic energy of the flow in units of the soundspeed, as measured through the Mach number. Instead of directly imposing an isothermal equation of state, we simulate gas

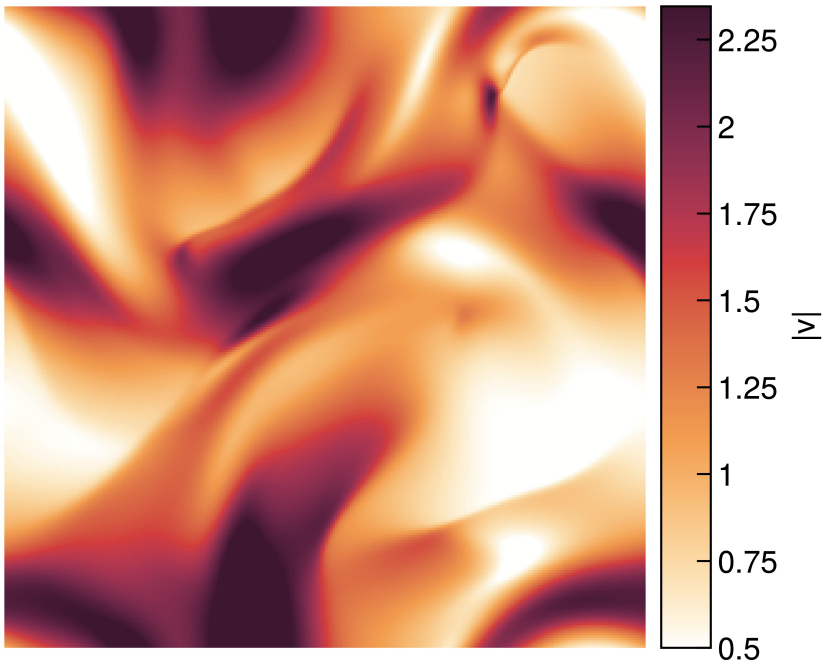


Figure 2.17: Two-dimensional slice through a driven, isothermal, subsonic 3D turbulence simulation depicting the velocity amplitude $|v| = (v_x^2 + v_y^2 + v_z^2)^{1/2}$ at $t = 20.48$, for a simulation with $N_c = 128$, $p = 4$, and $\mathrm{Re} = 10^5$.

with a normal ideal gas equation of state and reset the temperature every timestep such that a prescribed sound speed is retained. We have checked that this does not make a difference for any of our results, but this approach allows us to use our approximate, fast HLLC Riemann solver instead of having to employ our exact, but slower isothermal Riemann solver.

2.7.1 Driving

To create the turbulence, we drive fluid motions on large scales. To do this consistently at high order, we add a source function $\boldsymbol{s}(\boldsymbol{x}, t)$ to the right-hand side of the Euler equations, both in the momentum equation and as work function $\boldsymbol{s} \cdot \boldsymbol{v}$ in the energy equation. These source terms have to be integrated with Gaussian quadrature over cell volumes to retain the high-order discretization.

For setting up the driving field $\boldsymbol{s}(\boldsymbol{x}, t)$, we follow standard techniques as implemented in Bauer & Springel (2012), which in turn are directly based on Schmidt et al. (2006); Federrath et al. (2008, 2009). The acceleration field is constructed in Fourier space by populating modes in the narrow range $2\pi/L \leq k \leq 2 \times 2\pi/L$, with amplitudes scaled to follow $\propto k^{-5/3}$ over this range. The phases of the forcing modes are drawn from an Ornstein–Uhlenbeck process. They are periodically updated whenever a time interval Δt has elapsed, while keeping a temporal correlation over a timescale t_s with the previous

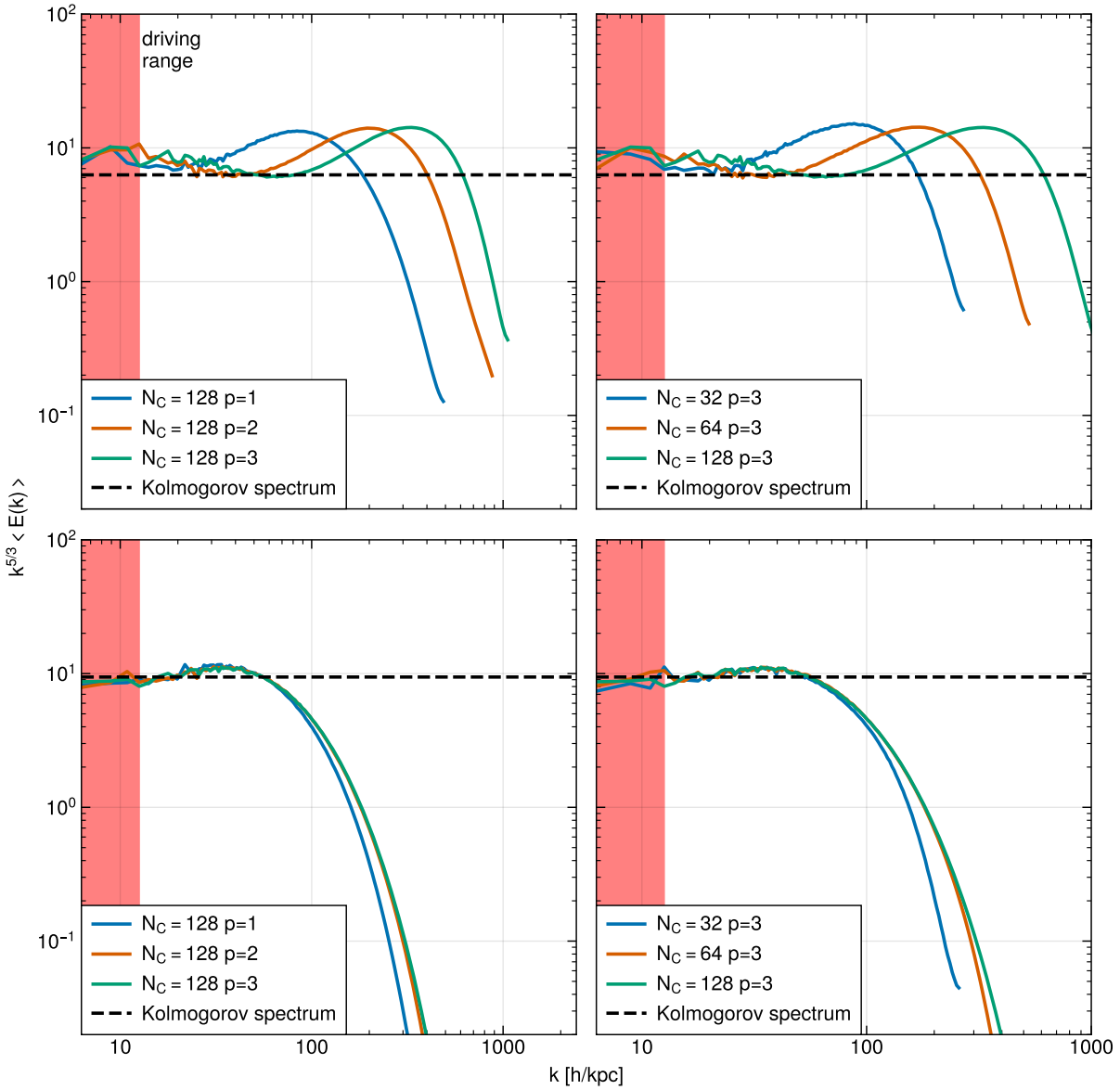


Figure 2.18: Compensated velocity power spectra of driven turbulence simulations as a function of wavenumber for varying numbers of cells, and varying spatial order. The panels in the top row show simulations where the Euler equations were solved, whereas the bottom two panels give results where the full compressible Navier-Stokes equations with a prescribed physical viscosity were used. The region marked with a red shade is the driving range.

phases. This effectively yields a smoothly varying, random driving field. Our specific settings for update frequency, coherence timescale and distribution function for drawing the driving phases are as in Bauer & Springel (2012, their table 1, left column).

We here also restrict ourselves to include only solenoidal driving, i.e. we project out all compressive modes in Fourier space by a Helmholtz decomposition. Specifically, if $\boldsymbol{s}$ is the

principal acceleration field constructed in the above fashion, we project it as

$$\hat{\boldsymbol{s}}(\boldsymbol{k}) = \left(\delta_{ij} - \frac{k_i k_j}{\boldsymbol{k}^2}\right) \boldsymbol{s}(\boldsymbol{k}) \tag{2.58}$$

in Fourier space to end up with an acceleration field $\hat{\boldsymbol{s}}$ that is free of compressive modes, which would only produce a spectrum of additional sound waves in our subsonic case.

2.7.2 Results for subsonic turbulence

All our turbulence simulations are started with gas of uniform density at rest. We monitor the average kinetic energy, as well as the total cumulative injected kinetic energy and the total cumulative dissipated energy, allowing us to verify the establishment of a quasi-stationary state. An example for this is shown in Figure 2.16, where we illustrate the build-up of the turbulent state in terms of the total energies. There is an initial ramp up phase of the turbulence until $t \sim 5$, during which the Mach number grows nearly linearly to its final quasi-stationary time-averaged value of $\mathcal{M} \simeq 0.47$. The cumulative injected energy grows approximately linearly with time, whereas the dissipated energy tracks it with a time lag, because the initial evolution until $t \sim 2.5$ does not yet show any significant dissipation. The difference between the injected and dissipated energies is the current kinetic energy of the gas, and thus is effectively given by the Mach number.

In Figure 2.17, we show a visual example of the quasi-stationary turbulent state established after some time, here simulated with $N_c = 128$ cells and order $p = 4$. The slice through the magnitude of the velocity field illustrates the chaotic structures characteristic of turbulence. Even though there are some steep gradients in the velocity field, the velocity varies smoothly overall, reflecting the absence of strong shock waves in this subsonic case.

To statistically analyse the turbulent state we turn to measuring power spectra of the velocity field at multiple output times, and then consider a time-average spectrum to reduce the influence of intermittency. To calculate the final power spectrum of a simulation, we average over 64 velocity power spectrum measurements over the time interval $5.12 < t < 20.48$.

Inviscid treatment of gas

The behaviour of inviscid gas is described by the Euler equations of eqn. (2.2). Because of the simplicity of this model and the desire to run simulations with as little viscosity as possible to maximize the intertial range of the turbulence, it is a popular choice for the study of turbulence. For example, the largest driven turbulence simulation to date by Federrath et al. (2016) were performed using inviscid gas, as well as many other studies in the field (e.g. Schmidt et al., 2006; Bauer & Springel, 2012; Bauer et al., 2016; Federrath et al., 2008, 2010; Price & Federrath, 2010).

In the top two panels of Fig. 2.18, we show such simulations carried out with our DG code. In all such simulation, the energy injected at large scales follows the Kolmogorov spectrum and cascades from large to small scales. This part of the spectra is called the

inertial range and it follows the $k^{-5/3}$ Kolmogorov spectrum closely, even though our gas is compressible and the density fluctuations for our Mach number are not negligible any more. Note that all our plots are compensated with a $k^{5/3}$ factor, such that the Kolmogorov spectrum corresponds to a horizontal line. The extent of the inertial range is primarily determined by the total number of degrees of freedom in an inviscid simulation. However, as we transition from the inertial range to the dissipation portion of the spectra, a noticeable bump can be seen in which the spectrum significantly exceeds the power-law extrapolation from larger scales. As energy is being transferred from larger to smaller scales, creating ever smaller eddies, it eventually reaches scales at which the code cannot resolve smaller eddies any more. This leads to a build-up of an energy excess at this characteristic scale, until the implicit numerical viscosity terms become strong enough to dissipate away the arriving energy flux. This effect is commonly known in numerical studies of turbulence and referred to as the "bottleneck" effect. It should be pointed out that experimental determinations of turbulent velocity spectra also show a weak form of this effect (see Verma & Donzis, 2007, and references therein). Küchler et al. (2019) later even measured the relation between the amplitude of the bump and R_λ of the flow. The problem of numerical simulations of inviscid gas is however that the shape of the bump is determined by numerical details of the hydrodynamic code and that it is usually excessively pronounced.

The bottleneck effect cannot be fixed by using higher resolution, or higher order for that matter. Indeed, in the top two panels of Figure 2.18 we can see that the bottleneck moves to ever smaller scales with increasing cell number at a fixed spatial order, and similarly it moves towards smaller scales if we increase the spatial order of our method at a fixed number of cells. While both avenues of adding further degrees of freedom successfully widen the inertial range and push the dissipative regime to smaller scales, they unfortunately cannot eliminate the "bump" in the bottleneck, or address the equally incorrect detailed shape of the dissipation regime itself. This detailed shape changes slightly as we vary the order p because the precise way of how numerical dissipation interacts with the flow is modified by this, while in contrast increasing the number of cells leaves the shape unchanged, because this just moves the dissipation regime to smaller scales in a scale-invariant fashion.

The only way around this and to get closer to velocity spectra seen in experimental studies of turbulence is to solve the full compressible Navier-Stokes equation, where the dissipative regime is set not by numerics, but by the physical viscosity of the gas itself. If this viscosity is large enough, it will effectively dissipate energy at scales larger than our numerical viscosity. We consider this case in the following subsection.

Viscous treatment of gas

We now consider driven turbulence results akin to the simulations just discussed, with the only difference being that we are now solving the full compressible Navier-Stokes equations as described in Sec. 2.3.3. In the bottom two panels of Fig. 2.18, we display compensated velocity power spectra with physical viscosity added. Such full Navier-Stokes simulations exhibit the proper behaviour of the "bottlenect" effect, as the location and shape of the

bump become resolution-independent and do not depend on numerical code details any more. Such simulations are in the literature referred to as "direct numerical simulations" or DNS. Our code can achieve DNS for turbulence by either increasing the resolution or the spatial order, as is evident in the bottom two panels of Fig. 2.18.

To determine if increasing the order of our method or its resolution is more beneficial, we compare three simulations with approximately the same number of degrees of freedom, but different resolutions and orders in Fig. 2.19. The orange line shows a run we can consider a converged DNS result with $N_c = 128$ and $p = 3$. A simulation with identical N_c but lower p in blue fails to fully converge. On the other hand, the green dashed line shows a simulation with *eight times fewer* total number of cells, but at a higher spatial order. It has as many degrees of freedom as the simulation shown in blue, and yet its power spectra matches that of the simulation shown in orange. We can therefore conclude that running driven turbulence at higher order is preferable to increasing the cell resolution. Or to put it another way, if there is a limited number of degrees of freedom that can be represented due to memory constraints, it is better to "spend" the memory on higher p than N_c. In the present case, a comparison of the wall-clock time between the high cell resolution and high order runs shows an about 2x faster calculation time at high order vs using a higher cell resolution. For even high order, this CPU-time advantage may not persist, but the memory advantage will. Given that turbulence simulations tend to be memory-bound, this in itself can already be a significant advantage.

2.8 Code details

2.8.1 Parallelization strategy

Modern supercomputers consist by now of thousands to millions of computing cores, a trend which is bound to continue. Recently, however, the most significant gains in computational performance (measured in floating point operations per second – FLOPS) have came from dedicated accelerator cards. These are most commonly, but not always, graphics processing units (GPUs) that have been repurposed to do general computational work. Accelerators achieve a large number of FLOPS by foregoing large, per-core caches and advanced control circuitry for single compute units, while at the same time they are able to execute large sets of threads concurrently in a data-parallel fashion.

Current GPU-accelerated computers typically consist of normal, CPU-equipped compute nodes that are outfitted with attached GPU cards. Utilising the power of both, CPUs and GPUs, efficiently with such heterogeneous machines is challenging. It requires not only a suitable subdivision of the work, but often also an algorithmic restructuring of the computations such that they can be mapped efficiently onto the massively parallel execution model of GPUs, as well as prescriptions for data placement and movement between the separate memory of CPU and GPUs. The problem becomes even harder when multiple compute nodes with distributed memory, each with their own GPUs, are supposed to work together on a tightly coupled problem. Efficient and scalable massively parallel codes for

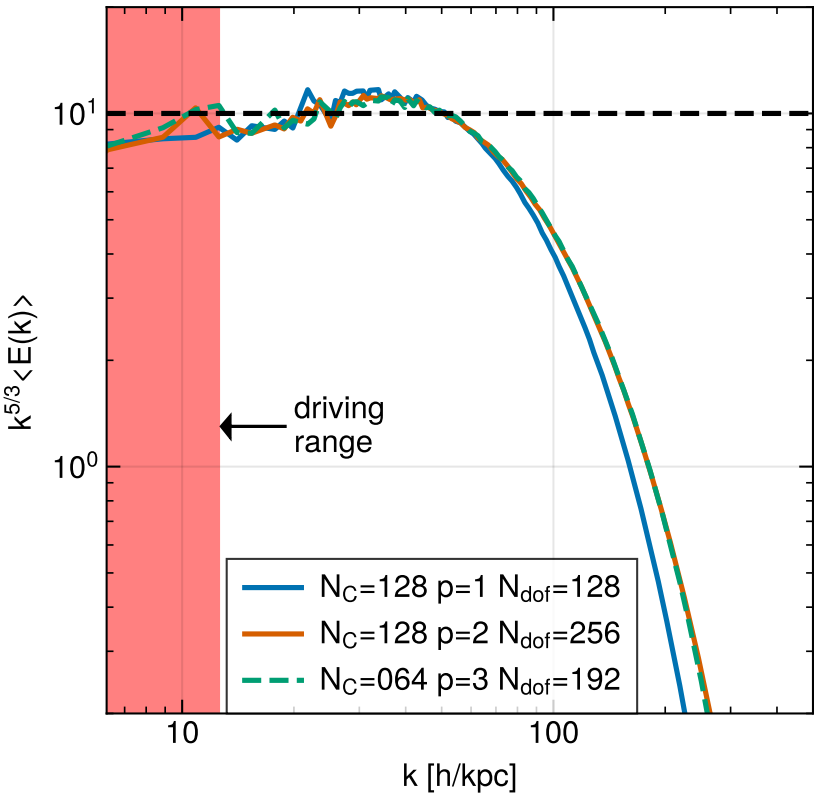


Figure 2.19: Compensated velocity power spectra as a function of wavenumber for a similar number of degrees of freedom, but varying the spatial order and the number of cells. The total wall-clock time for the simulation runs $128^3|p=2$, $128^2|p=3$, and $64^3|p=4$ on 16 A100 GPUs were 0.9, 3.9, and 1.8 hours, respectively. We note that one can keep the converged result obtained with $N_c = 128$ and $p = 3$ by going to fewer cells and higher order (the $N_c = 64$ and $p = 4$ run), while still achieving a speed-up.

such machines must decompose the problem into multiple parts, distribute the parts among the available compute units, and only exchange data between various parts when really needed.

In the present version of our code TENETGPU[2], we address this by an implementation that can execute a given hydrodynamical problem flexibly either on one or several GPUs, on one or multiple CPU cores, or a mixture thereof. Independent on how GPUs and CPU-cores are distributed onto different compute nodes, TENETGPU can in this way make use of whatever is available, up to extremely powerful systems such as the first exascale supercomputers that are presently put into service (which are GPU-accelerated, such as 'Frontier', ranked the fastest in the world according to the Top500 list released May 30,

[2]While our code is written from scratch for GPUs, its first version has been heavily inspired by the code TENET of Schaal et al. (2015), hence we named ours TENETGPU. Source code: https://bitbucket.org/Migelo/gpu_testbed

2022).

To achieve this flexibility, we split the mesh along the x-axis into different slabs, which can have different thickness, if desired. Each slab is either computed by a different GPU, or by one CPU core. The communication between slabs, which is realized with the Message Passing Interface (MPI), thus needs to happen along the x-dimension between neighboring slabs only, as all the needed data along the other two axes is locally available for the corresponding slab. The data that is communicated consists of surface states or surface fluxes at Gauss points needed for integrations over cell areas. For driving the GPU computations, each GPU requires a separate CPU core as well. For example, if one has a compute node with 32 cores and 2 GPUs as accelerator cards, a simulation with 256^3 mesh cells could be run by assigning slabs with a thickness of 98 cells to each of the GPUs, while letting the remaining 30 compute cores each work on slabs with a thickness of 2 cells each. Of course, this particular mixed execution example would only make sense if each of the GPUs would be around ~ 50 times faster than a single CPU core. In practice, the speed difference is typically considerably larger, so most of the work should typically be assigned to GPUs if those are available.

We also note that for the moment our code supports only meshes with uniform and fixed resolution. However, a more general domain decomposition than just a slab-based decomposition is planned for the future and in principle straightforward. This can, in particular, remove the obvious scaling limitation of our current approach, where the number of cells per dimension sets the maximum number of GPUs or CPU cores that could be employed.

2.8.2 GPU computing implementation

The above parallelization strategy makes it clear that our code is neither a plain CPU code nor a pure GPU code. Rather, it implements its core compute functionality where needed twice, in a CPU-only version and in a GPU-only version. Both versions can be interchangeably used for any given slab taken from the global computational mesh, and they produce the same results. While this approach evidently requires some extra coding, we have found that this is actually quite helpful for code validation, as well as for quantifying the relative performance of CPU and GPU versions. Further, the extra coding effort can be greatly alleviated by using wherever possible functions that can be compiled and executed both by GPUs and CPUs based on a single implementation.

For the GPU code, we have used the CUDA programming model available for Nvidia GPU devices. All our code is written in low level C++, and we presently do not make use of programming models such as OpenACC, special GPU-accelerated libraries, or new C++ language features that allow GPU-based execution of standard libraries via execution policies. Our programming model is thus best described as MPI-parallel C++, accelerated with CUDA[3] when GPUs are available. If no GPUs are available, the code can still be

[3]We presently use the CUDA toolkit version 11.4, the GNU g++ 11 compiler and the C++17 standard. For message passing, we prefer the OpenMPI-4 library, for Fourier transforms we use FFTW-3 and for

compiled into a CPU-only version.

For storing static data such as coefficients of Legendre polynomials or Gaussian quadrature weights, we try to make use of the special constant memory on GPUs, which offers particularly high performance, also in comparison to the ordinary general memory. Likewise, for computing parallel reductions across individual cells, we make use of the special shared memory. However, the size of the corresponding memory spaces is quite limited, and varies between different GPU hardware models. This can necessitate adjustments of the used algorithms at compile time, depending on code settings such as the expansion order and on which execution platform is used. We address this by defining appropriate compile-time switches, such that these adjustments are largely automatic.

We note that the data of slabs that are computed with GPUs need to fit completely on GPUs as we refrain from transmitting the data from the front end host computer to the GPU on every timestep. Instead, the data remains on the GPU for maximum performance, and only when a simulation is finished or a temporary result should be output to disk it is copied back from the GPU to the front end host. Wherever such transfers are needed, we use pinned memory on the front end to achieve maximum bandwidth between the host and GPUs. GPUs can access such pinned memory directly, without going through the host CPU first. The problem sizes we are able to efficiently tackle with GPUs are therefore limited by the total combined GPU memory available to a run. Modern GPUs typically have some 10 GBs of main memory, but the detailed amount can vary greatly depending on the model, and is course a matter of price as well. The communication between adjacent slabs is organized such that communication and computation can in principle overlap. This is done such that first the surface states are computed and a corresponding MPI exchange with the neighbouring slabs is initiated. While this proceeds, the volume integrals for slabs are carried out by the GPU, and only once this is completed, the work continues with the received surface data.

Because slabs that are computed on GPUs need to be executed in a massively thread-parallel fashion with shared-memory algorithms, some changes in the execution logic compared to the effectively serial CPU code are required. For example, to avoid race-conditions in our GPU code without needing to introduce explicit locks, we process the mesh in a red-black checkerboard fashion. Finally, we note that we also implemented a scheme that makes our results binary identical when the number of mesh slabs is changed. This ultimately relates to the question about how the wrap-around between the leftmost and rightmost planes of the mesh in our periodic domain is implemented. Here the order in which fluxes from the left and right neighboring cells is added to cells needs to be unique and independent of the location and number of slabs in the box in order to avoid that different floating point rounding errors can be introduced when the number of slabs is changed.

random number generation we rely on GSL 2.4.

N_c	p	min. memory need
128	1	512 MB
128	2	1440 MB
128	3	3520 MB
128	5	9856 MB
128	9	37.81 GB
2048	1	2048 GB
2048	2	5760 GB
2048	3	13.75 TB
2048	5	38.5 TB
2048	9	151.3 TB

Table 2.1: Minimum memory need for our DG code when a 3D simulation is assumed with $(N_c)^3$ cells and expansion order p, including allowing for an artificial viscosity field. Here double precision with 8 bytes per floating point number has been assumed.

2.8.3 Memory usage

Before closing this section, it is perhaps worthwhile to discuss the memory need of our DG simulations, as this is ultimately determining the maximum size of simulations that can be done for a given number of GPUs. To represent a scalar field such as the density ρ at order p, we need for every cell a certain number of basis function weights $N^{d\mathrm{D}}(p)$, were d is the number of spatial dimensions, see equations (2.17) and (2.18). When multiplied with the number of cells, we obtain the number of degrees of freedom, which is identical to the number of floating point variables needed to stored the full density field. If we write the total number of cells as $(N_c)^d$, then the total number of variables that need to be stored for the DG weight vector is

$$N_w = (2+d)(N_c)^d N^{d\mathrm{D}}(p). \tag{2.59}$$

Here we assumed that we simulate the plain Euler equations without viscosity, where we need $(2+d)$ conserved fields to describe the flow. If we account for our artificial viscosity field, which will always be required for problems involving shocks, this number goes up by one further unit, yielding

$$N_w = (3+d)(N_c)^d N^{d\mathrm{D}}(p). \tag{2.60}$$

A passive tracer field, if activated, would add a further unit in the prefactor. In 2D and 3D, a conservative upper bound for $N^{d\mathrm{D}}(p)$ is p^d, but this is not particularly tight. Already for $p = 2$, $N^{3\mathrm{D}}$ is lower than p^3 by a factor of 2, for $p = 4$ this grows to a factor 3.2, and for $p = 10$ the difference is more than a factor 4.5.

Another significant source of memory need lies in our timestepping algorithm. At present we use stability preserving Runge-Kutta schemes that require a temporary storage of the time derivatives of the weights, evaluated at several different points in time, depending on the order of the Runge-Kutta scheme, which we adjust according to the chosen p.

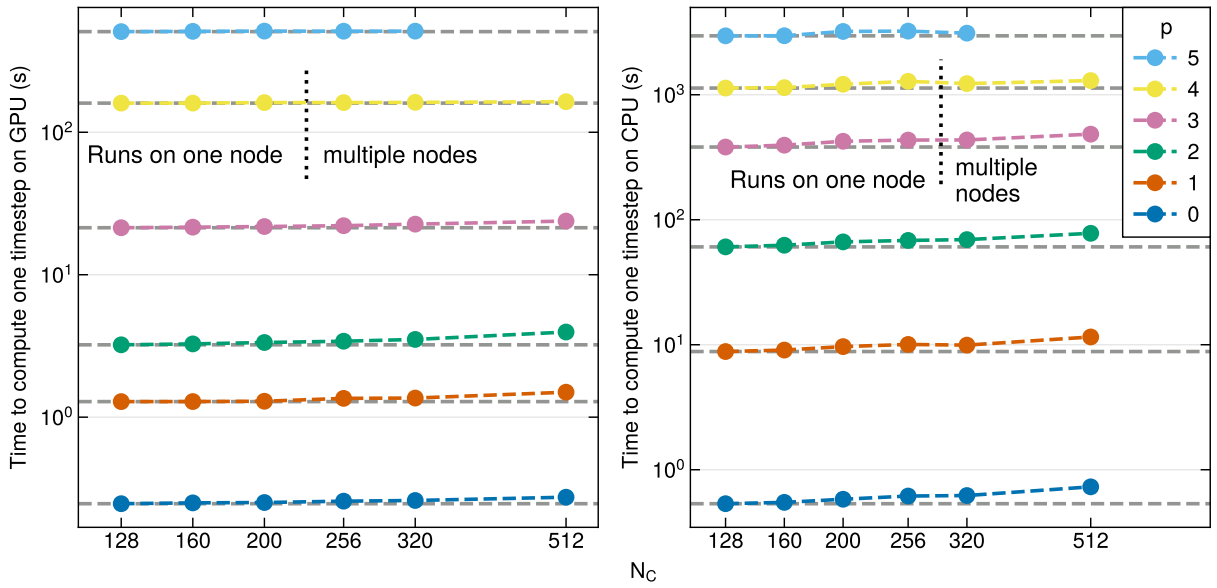


Figure 2.20: Weak scaling of TENETGPU for a 3D test problem. The y-axis shows the time taken to compute one timestep averaged over a small number of timesteps. The left panel shows results for GPU execution when the problem size N_c^3, measured in terms of the number of cells N_c per dimension, increases in several steps by close to a factor of two from $N_c = 128$ to $N_c = 512$ cells, and when between 1 to 64 GPUs are applied to the problem. In contrast, the right hand panel gives results when the problems are executed on CPUs instead, using from 4 to 256 cores, again keeping in each case the load per computational element constant. We carry out the measurements for different expansion order, from $p = 0$ to $p = 5$. Ideal weak scaling corresponds to horizontal lines (dashed). The dotted vertical line marks the transition between using CPU cores or GPUs associated with a single compute node of our cluster, and the use of multiple nodes in which MPI data exchange via the Intel Omni-Path takes place. The missing measurement at $p = 5$ is due to the large memory required to store communication buffers, which make the $N_c = 512$ problem not fit onto 64 GPUs. The missing data points at $N_c = 400$ are due to 400 not being divisible by 32, as this would lead to uneven distribution of work across the GPUs we did not consider these runs.

The required temporary storage space $N_{\dot{w}}$ is thus a multiple of N_w, with a prefactor that depends on the chosen order p, i.e.

$$N_{\dot{w}} = f_t(p) N_w. \tag{2.61}$$

Here $f_t(p)$ depends on the number of stages in the Runge-Kutta scheme. Presently, we use a setup where $f_t(p) = p$ for $p \leq 3$, and $f_t(p) = 5$ otherwise. The minimum amount of total storage (in terms of needed floating point numbers) required by the code is thus

$$N_w = [3 + d + f_t(p)](N_c)^d N^{\mathrm{dD}}(p). \tag{2.62}$$

During execution of our code using multiple GPUs or CPU cores, some temporary buffer space is furthermore required to hold, in particular, send and receive buffers for

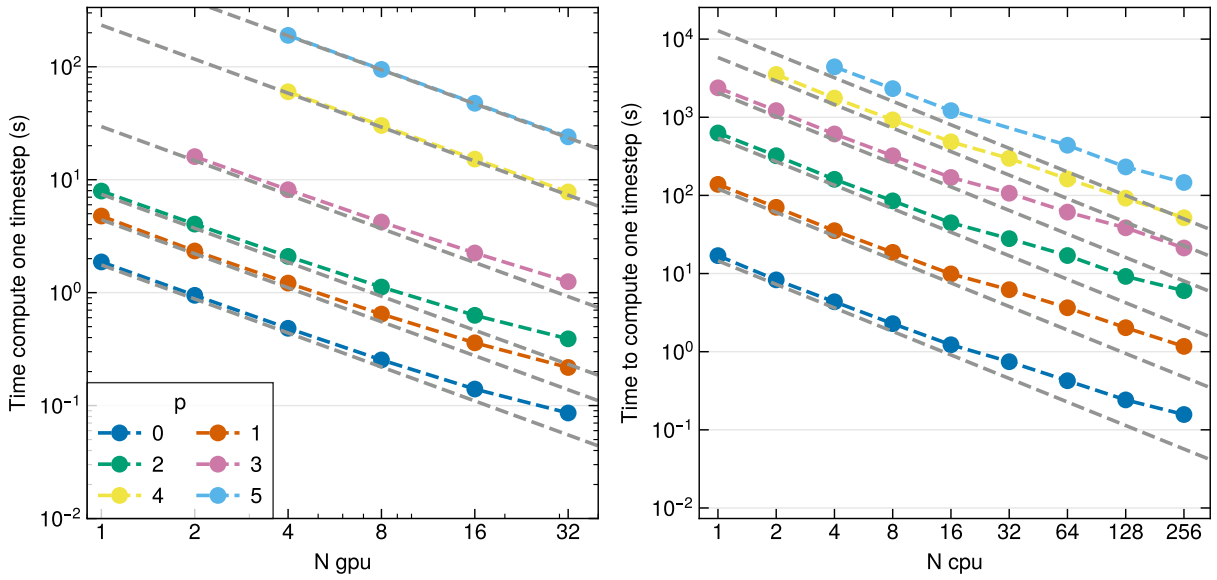


Figure 2.21: Strong scaling of TENETGPU for a 3D test problem of size 256^3 cells. The y-axis shows the average time taken to carry out one timestep. The left panel shows timing results when between 1 and 16 Nvidia A100 GPUs are used, while the right panel gives results when between 1 to 256 ordinary Intel Xeon-6138 cores are used. Ideal strong scalability corresponds to the dashed lines indicated in the panels. Missing data points at high orders and low number of compute devices are due to the fact that such large problems do not fit on a single GPU / node.

fluid states or fluxes along slab surfaces orthogonal to the x-direction. These tend to be subdominant, however, compared to the memory requirements to store the weights and their time derivatives themselves. The latter thus represent the quantities that need to be primarily examined to decide about the feasibility of a simulation in terms of its memory needs. When we use the oscillatory sensor for controlling artificial viscosity, some further temporary storage is needed as well, but since this is again small compared to N_w since only two scalar quantities per cell are needed, this conclusion is not changed. Note that our DG approach does not need to store gradient fields for any of the fields, which is different from many finite volume methods such as, for example, AREPO. Also, use of the Navier-Stokes solver instead of simulating just the Euler equations does not increase the primary memory needs in any significant way.

In Table 2.1, we give a few examples of the memory need for a small set of simulation sizes and simulation orders, which illustrates the memory needs of the code, and which can be easily scaled to other problem sizes of interest. A single Nvidia A100 GPU with 40 GB of RAM could thus still run a $N_c = 128$ problem at order $p = 9$, or a 512^3 problem at quadratic order $p = 1$. For carrying out a 2048^3 simulation at $p = 1$, a cluster offering at least 52 such devices would already be necessary.

2.9 Code performance

In order to fully utilise large parallel supercomputers, a code has to be able to run efficiently not only on a single core on one CPU, but also on hundreds to thousands of cores on many CPUs. The degree to which this can accelerate the total runtime of a computation is encapsulated by the concept of parallel scalability. Similarly, for a GPU-accelerated code it is of interest to what extent the use of a GPU can speed up a computation compared to using an ordinary CPU. If more than a single GPU is used, one is furthermore interested in whether a code can efficiently make simultaneous use of several, perhaps hundreds of GPUs. In this section we examine these aspects and present results of weak- and strong scaling tests of our new code.

2.9.1 Weak scaling

Weak scaling performance describes a situation where a set of simulations of increasing size is run and compared, but where the load per computational unit, be it a CPU core or a GPU in our case, is kept constant. The time to perform a single timestep should remain constant in this case, increasing only due to communication-related overhead, through work-load imbalances, or through other types of parallelization losses, for example if a code contains residual serial work that scales with the problem size.

Weak scaling results of our code are shown on Fig. 2.20. We run a 3D box with constant density and pressure using the Navier-Stokes equations, the positivity limiter and artificial viscosity. This setup is computationally very close to problems we are running in production. We consider problem sizes of 128^3, 160^3, 200^3, 256^3, 320^3, and 512^3 cells, forming a sequence that approximately doubles in size, with a factor of 64 enlargement from the smallest to the largest runs. To compensate for the fact that the problem size does not exactly double every time we increase the number of cells, we apply a correction factor to the timing results at each resolution[4]. Correspondingly, we execute these problems with one Nvidia A100 GPU for the smallest mesh size, and 64 GPUs for the largest mesh size, keeping the load per GPU roughly constant. The results are shown in the left panel of Fig. 2.20. For comparison, we also measure the execution speed if instead every GPU is replaced by four CPU cores of Intel Xeon-6138 processors. The corresponding results are shown in the right panel of Fig. 2.20. Finally, we repeat these measurements for different DG expansion orders $p = 0 - 5$.

The results in the figure show generally good weak scalability, but also highlight some performance losses for large problem sizes. These arise in part because our domain is split into slabs and not cubes. Larger problems lead to ever thinner slabs with a larger surface-to-volume ratio and thus more communication between different slabs. We also see the influence of enhanced communication on weak scalability when data needs to be

[4]The current version of the GPU part of the code can only run if N_c and the number of slabs in the x-direction per rank are even. This and the fact that N_c has to be an integer in any case prevents ideal doubling of problem size. The correction factors we apply are: 128^3: 1.0, 160^3: 0.977, 200^3: 0.954, 256^3: 1.0, 320^3: 0.977, and 512^3: 1.0.

transferred across node boundaries. At higher orders the weak scaling is generally better, as the compute-to-communicate time ratio shifts strongly to the compute side.

2.9.2 Strong scaling

Strong scaling is a test where one runs a problem of given size on an ever increasing number of compute units. Contrary to weak scaling, the load per compute unit decreases in this test, and the time to perform a single timestep should decrease in inverse proportion to the increasing computational power applied to solve the problem.

We show a strong scaling result in Fig. 2.21, again carried out for a 3D box with constant density and pressure using the Navier-Stokes equations, the positivity limiter and artificial viscosity. For definiteness, we use a simulation with 256^3 cells, and consider orders $p = 0$ to $p = 5$. The left panel of Figure 2.21 shows the average execution time for a single step when 1, 2, 4, 8, or 16 Nvidia A100 GPUs are used. In contrast, the right panel of Figure 2.21 shows the average execution time when CPU cores on a cluster with 2 Intel Xeon-6138 CPUs are used, with 40 cores per node. We show results from 1 core to 256 cores. Especially in the latter case, one sees clear limits of strong scalability, as communication costs become quite large if the problem is decomposed into slabs that are just a single cell wide. This stresses that there is always a limit for strong scalability, something that is known as Ahmdahl's law. By enlarging the problem size, this limit can however usually be pushed to larger parallel partition sizes. Another major contributor to the degradation that happens when going from 16 to 32 cores is the saturation of available memory bandwidth of a single 20-core socket. We verified this using the STREAM benchmark[5].

2.9.3 CPU vs GPU benchmark

Another interesting question is how the absolute speed of GPU execution of our code compares to running it only on ordinary CPU cores. To estimate this speedup we take the average execution times to compute a timestep from our weak scaling results for both the GPU and CPU runs and consider their ratio. We do this for the three considered DG orders $p = 2$ to $p = 4$, and for the varying problem sizes and number of compute units used. Since we had used 4 CPU cores to pair up with 1 GPU, we rescale the results in two different ways, to either compare the execution performance of four Nvidia A100 GPUs with 40 Intel Xeon-6138 cores – which is how one of our compute nodes is equipped – or to the performance of a single GPU compared to one CPU core (which thus gives 10 times higher values).

The corresponding results are illustrated in Fig. 2.22. The speedup of GPU execution at the node-level is modest for order $p = 2$, as there are not enough floating point operations to fill up the GPUs. At $p = 2,\ 3$ we reach the highest node-level speedup observed among this set of runs, it peaks at just over 8x the CPU speed for large problems. This runs show better performance because there are a lot of floating points operations to perform at the

[5] https://github.com/intel/memory-bandwidth-benchmarks

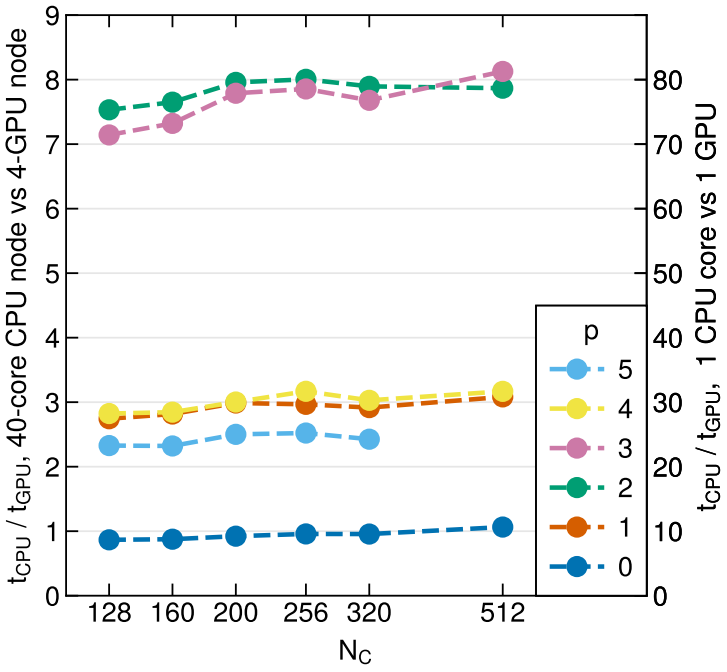


Figure 2.22: Ratio of time taken to calculate one timestep of test simulations with the Navier-Stokes solver on GPUs or CPUs, based on our weak scaling test runs. The left vertical scale shows results when we normalize them to the speed ratio for using 4 Nvidia A100 GPUs versus 40 Intel Xeon 6138 CPU cores, while the right scale normalizes the speed results to a comparison of 1 GPU vs 1 CPU core.

same time, and all intermediate results still fit into the GPU's limited shared memory. Such shared memory is "on chip" and therefore about ∼100x faster than global memory. Once the intermediate results become too large to fit into shared memory, the code determines the maximum number of quadrature points it can process at once and proceeds forward in batches of n quadrature points. At this point, a single GPU is about 80 times as fast as a CPU core, but when comparing a fully equipped GPU node to a fully equipped CPU node, more realistic numbers are in the ballpark of ~ 8. Note that this speedup metric is based on the specific hardware configuration of the cluster the authors had access to throughout this project. While the configuration of four Nvidia A100 GPUs paired with about 40 Intel Xeon cores quite typically reflects the general HPC situation in 2021 and 2022, the corresponding hardware characteristics are not universal and can be expected to evolve substantially in future generations of CPU-GPU systems. In any case, the performances we find are not far away from the ratio of the nominal peak performances of the involved compute devices for double precision arithmetic (which we have used here throughout), but this comparison also suggests that there is still some modest room for improvement in the performance of our GPU implementation.

2.10 Summary and Conclusions

In this study, we have described a novel hydrodynamical simulation code which is based on the mathematical Discontinuous Galerkin approach. The fluid state is expanded in this method into a set of spatially varying basis functions with time-variable weights, yielding a separation of the temporal and spatial dependencies. The time evolution of the weights is obtained in a weak formulation of the underlying partial differential equations of fluid dynamics.

Our work builds up on the earlier development of a DG code by Schaal et al. (2015) and Guillet et al. (2019), but extends it into several crucial directions. First of all, we have developed a novel GPU implementation from scratch, thereby demonstrating the substantial potential of these acceleration devices for achieving higher computational performance in astrophysical applications. This potential has already been identified in a few first finite-volume hydrodynamical GPU codes in astrophysics, but ours is the first one that can carry out DG calculations of the full Navier-Stokes equations at very high order of $p = 10$ and beyond.

Secondly, we have introduced a novel approach to shock-capturing at high order, solving the long-standing problem that standard slope-limiting techniques do not work well at high order and tend to discard in troubled cells much of the advantage that is supposed to be delivered by a high order approach. The latter can only be rescued if the DG method is able to capture physical discontinuities in a sub-cell fashion. By means of our new source routines for a time-dependent artificial viscosity field, we have demonstrated very good shock-capturing ability of our code, with a shock broadening that closely tracks the effective spatial resolution h/p that we expect from the method based on its number of degrees of freedom per dimension. While this does not necessarily give high-order approaches an advantage for representing a shock compared with a lower order method with the same number of degrees of freedom, at least it also is not worse – using a high-order approach will however in any case still be beneficial for all smooth parts of a flow. If it performs at the same time as well as a lower order method in places where there is a shock, this can be a significant advantage. For contact discontinuities, similar considerations apply, but here high-order methods have the additional advantage of exhibiting greatly reduced numerical diffusivity. Contact discontinuities that move over substantial timespans therefore also benefit from the use of higher order.

Third, we have stressed that the use of physical viscosity is often a key to make problems well posed and amenable to direct numerical solutions. Here we have introduced a novel method to define the viscous surface fluxes at cell interfaces. This is based on arriving at unambiguous derivatives at interfaces by projecting the two piece-wise solutions in the adjacent cells onto a continuous basis function expansion covering both cells. The derivatives can then be computed in terms of analytic derivatives of the basis functions. We have shown that this technique is robust, consumes much less memory and computational effort than the uplifting technique, and most importantly, it converges at the expected rapid convergence rate when high order is used.

In fact, in several of our test problems, we could show that our DG code shows for

smooth problems exponential convergence as a function of expansion order p, while for fixed order, the L_1 error norm declines as a power-law of the spatial resolution, $L_1 \propto h^p$. These favourable properties suggest that it is often worthwhile to invest additional degrees of freedom into the use of higher expansion order rather than employing more cells. However, since every DG cell effectively represents a small spectral problem in which the required solution evaluations and volume integrations are carried out in real space, the computational cost to advance a single cell also increases rapidly with order p. In practice, this can make the optimal order quite problem dependent.

With our present implementation we could obtain excellent agreement with the reference Kelvin-Helmholtz solution computed by Lecoanet et al. (2016) with the spectral code DEDALUS. Remarkably, we achieved this already with 64 cells and order $p = 4$, for which our results are equally as accurate as those obtained with the finite volume code ATHENA at second order using 2048 cells. This again shows the potential of the DG approach. Given that in this work we could overcome one of its greatest weaknesses in an accurate, simple, and robust way – namely the treatment of shocks at high order – we are confident that the DG method could soon turn into a method of choice in astrophysical applications, rivaling the traditional finite volume techniques. Our next planned steps to make this a reality are to add additional physics such as radiative cooling and self-gravity to our code, and to provide functionality for local refinement and derefinement (h-adaptivity), as well as to allow for varying the expansion order used in a single cell (p-adaptivity). The high performance we could realize with our GPU implementation, which outperforms modern multi-core CPUs by a significant factor, furthermore strengthens the case to push into this direction, which seems also a necessity to eventually be able to harness the power of the most powerful supercomputers at the exascale level for unsolved problems in astrophysical research.

Chapter 3

Supersonic turbulence simulations with GPU-based high-order Discontinuous Galerkin hydrodynamics

This work has been submitted to the Monthly Notices of the Royal Astronomical Society.

We investigate the numerical performance of a Discontinuous Galerkin (DG) hydrodynamics implementation when applied to the problem of driven, isothermal supersonic turbulence. While the high-order element-based spectral approach of DG is known to efficiently produce accurate results for smooth problems (exponential convergence with expansion order), physical discontinuities in solutions, like shocks, prove challenging and may significantly diminish DG's applicability to practical astrophysical applications. We consider whether DG is able to retain its accuracy and stability for highly supersonic turbulence, characterized by a network of shocks. We find that our new implementation, which regularizes shocks at sub-cell resolution with artificial viscosity, still performs well compared to standard second-order schemes for moderately high Mach number turbulence, provided we also employ an additional projection of the primitive variables onto the polynomial basis to regularize the extrapolated values at cell interfaces. However, the accuracy advantage of DG diminishes significantly in the highly supersonic regime. Nevertheless, in turbulence simulations with a wide dynamic range that start with supersonic Mach numbers and can resolve the sonic point, the low numerical dissipation of DG schemes still proves advantageous in the subsonic regime. Our results thus support the practical applicability of DG schemes for demanding astrophysical problems that involve strong shocks and turbulence, such as star formation in the interstellar medium. We also discuss the substantial computational cost of DG when going to high order, which needs to be weighted against the resulting accuracy gain. For problems containing shocks, this favours the use of comparatively low DG order.

3.1 Introduction

Turbulence is a fundamental physical phenomenon that appears universally in fluid flow (e.g. Launder, 1974; Larson, 1981; Mellor & Yamada, 1982; Kim et al., 1987; Menter, 1994; Frisch, 1995; Goldreich & Sridhar, 1995; Balbus & Hawley, 1998; Pope, 2000; Brandenburg & Åke Nordlund, 2011), and thus affects many fields of study, including meteorology, engineering, and, of course, astrophysics. For example, there is turbulence in and around the Sun, something that will be further characterized by a recently approved NASA Medium-Class Explorer mission (Klein et al., 2023). In our Galaxy, the interstellar medium (ISM) is characterized by supersonic turbulent motions that shape the gas distribution and gas kinematics, and that play a fundamental role in regulating star formation, as first observed by Larson (1981), with a recent review on the topic by Ballesteros-Paredes et al. (2020). Multiple recent works used ALMA to study the influence of turbulence on star formation (e.g. Li et al., 2020; Gómez et al., 2021; Bhadari et al., 2023). The ongoing PASIPHAE (Tassis et al., 2018) and POSSUM (Anderson et al., 2021) surveys will soon produce a full tomographic map of the galactic magnetic field, shedding new light on the nature ISM and CGM turbulence.

The observational interest in ISM turbulence is matched only by the vast number of theoretical investigations. Because turbulence has a commanding influence on the distribution of gas in the ISM, many studies, dating back decades, have looked into this (e.g. Scalo et al., 1998; Passot & Vázquez-Semadeni, 1998; Ostriker et al., 1999; Klessen, 2000; Wada & Norman, 2001; Ballesteros-Paredes & Mac Low, 2002; Li et al., 2003; Kravtsov, 2003; Mac Low et al., 2005; Federrath et al., 2009, 2021; Mathew et al., 2023; Rabatin & Collins, 2023). Using the results from these studies a series of new star formation recipes were proposed by, e.g., Kretschmer & Teyssier (2020) and Girma & Teyssier (2024), among others.

Hydrodynamical simulations are a primary tool for the study of such highly non-linear physics. But numerical effects and resolution limitations strongly influence the quality of the obtained hydrodynamical results, motivating a constant search for improvements in numerical schemes and likewise demanding careful validation of new techniques.

In this chapter, we investigate the main properties and effects of high-order Discontinuous Galerkin (DG) methods when applied to supersonic turbulence. The DG approach is a general tool of applied mathematics first used to solve the equations of neuron transport by Reed & Hill (1973) and then robustly defined in a series of five papers by Cockburn & Shu (1988, 1989); Cockburn et al. (1989, 1990); Cockburn & Shu (1998). DG has been gaining traction as a key method for solving partial different equations, such as the Euler equations of fluid flow used in multiple recent works (Schaal et al., 2015; Velasco Romero et al., 2018; Guillet et al., 2019).

In a companion study (Cernetic et al., 2023), we have presented an implementation of a GPU-accelerated, MPI-parallel DG code for solving the Navier-Stokes equations. We could confirm the high accuracy and computational efficiency of this approach in a variety of test problems, even showing exponential convergence as a function of the employed spectral order. We also demonstrated that shocks and physical discontinuities can be handled by

an artificial viscosity field at sub-cell resolution. The width of these continuities is however fundamentally limited by the effective spatial resolution of the scheme, and thus only linearly improves with higher spatial resolution, as is the case with ordinary finite volume schemes. This raises the important question whether the advantages of DG are defeated in problems containing many shocks, a question we seek to address in this chapter.

A physical setting where this perhaps can be answered in a particularly succinct way is supersonic, isothermal turbulence. The density probability distribution function (PDF) and the power spectrum of compressible, supersonic turbulence play an important role especially in theories of star formation. However, super- and hypersonic turbulence are particularly challenging for Eulerian mesh codes given the extremely high ram pressures, strong shocks, and huge density contrasts that develop in this regime, in addition to regions of nearly vanishing density. This makes it hard to capture the inertial range of supersonic turbulence accurately, even more so than for subsonic turbulence.

In DG methods, the solution inside cells is approximated by smooth, high-order polynomials. It is clear that strong shocks passing through cells may play havoc with such polynomials, causing strong Gibbs-like oscillations, and worse, potentially trigger so wide oscillations that unphysical values of fluid variables occur. To address this, we in this chapter introduce a modification of our sub-cell shock capturing scheme – actually simplifying it considerably compared to our previous approach – by resorting to a classic von Neumann-Richtmyer viscosity (Von Neumann & Richtmyer, 1950). In addition, we introduce a novel regularlisation of the primitive variables at cell boundaries, which proves critical to stably and accurately evolve high Mach number turbulence with DG at high order.

In this chapter, we demonstrate the accuracy of these new implementations by considering a number of test problems containing strong shocks. We then move on to simulations of driven, isothermal turbulence. We vary the Mach number systematically from low values to Mach numbers beyond ten, comparing at each stage DG with a standard, second-order finite volume method based on piece-wise linear reconstruction. We analyze velocity power spectra, structure functions and density PDFs in order to examine the advantages brought by going to DG as compared to classic finite volume (FV) methods with the same number of cells. We also include an analysis of high dynamic range simulations that can resolve the sonic point.

The chapter is structured as follows. First, in Section 3.2, we summarize the DG approach in general and the particular implementation we have developed in our GPU-based code. Then, in Sections 3.3 and 3.4, we introduce two method improvements in the form of a new artificial viscosity treatment and a projection method for the primitive fluid variables. Combined, they allow sustained high mach number turbulence simulations with DG. In Section 3.5 we detail our implementation of turbulence driving and our measurements of basic turbulence statistics. Section 3.6 presents our main simulation results in the form of a systematic suite of turbulence simulations, from the subsonic to the supersonic regimes. For a specific choice of driving, we extend the dynamic range in Section 3.7 substantially by going to higher resolution, allowing us to see the transition from supersonic to subsonic turbulence at the sonic point. We discuss the computational cost of high-order DG in Section 3.8, and conclude by summarizing our results in Section 3.9.

3.2 Discontinuous Galerkin hydrodynamics

The Discontinuous Galerkin (DG) approach is a general high-order finite element method for numerically solving partial differential equations (e.g. Cockburn & Shu, 1989). Here we apply it to the Euler and Navier-Stokes equations for numerical hydrodynamics. Consider the Euler equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{\alpha=1}^{d} \frac{\partial \boldsymbol{f}_\alpha(\boldsymbol{u})}{\partial x_\alpha} = 0, \tag{3.1}$$

with the state vector $\boldsymbol{u}$ storing the conserved variables of each cell, and the sum α running over their spatial dimensions and $\boldsymbol{f}_\alpha(\boldsymbol{u})$ being the analytical flux matrix. The state vector consists of

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho\boldsymbol{v} \\ e \end{bmatrix}, \quad e = \rho u + \frac{1}{2}\rho\boldsymbol{v}^2, \tag{3.2}$$

with u being the specific internal thermal energy, while ρ denotes the fluid density, $\boldsymbol{v}$ its velocity, and e its total energy density. To completely describe the gas we also need an equation of state connecting the pressure p with u and ρ. For this we employ the ideal gas equation,

$$p = \rho u \left(\gamma - 1\right), \tag{3.3}$$

where γ is the ratio of specific heats at constant pressure and constant volume, respectively, commonly known as the adiabatic index. The flux matrix $\boldsymbol{f}_\alpha(\boldsymbol{u})$, spelled out explicitly in 3D, is given by

$$\boldsymbol{f}_1 = \begin{pmatrix} \rho v_x \\ \rho v_x v_x + p \\ \rho v_x v_y \\ \rho v_x v_z \\ (\rho e + p) v_x \end{pmatrix}, \; \boldsymbol{f}_2 = \begin{pmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y v_y + p \\ \rho v_y v_z \\ (\rho e + p) v_y \end{pmatrix}, \; \boldsymbol{f}_3 = \begin{pmatrix} \rho v_z \\ \rho v_x v_z \\ \rho v_y v_z \\ \rho v_z v_z + p \\ (\rho e + p) v_z \end{pmatrix}. \tag{3.4}$$

The key starting point of the DG method is to approximate the solution of the Euler equations (3.1) in each cell of interest by projecting the state vector (3.2) onto a set of orthogonal basis functions for each cell. The resulting solution representation is allowed to be discontinuous across element boundaries, i.e. each cell has its own projection that is independent of that in neighbouring cells. The time evolution of the solution in each cell and the coupling of the solutions across cell boundaries are derived from a weak form of the underlying differential equations. At cell boundaries, this gives rise to numerical flux functions that can be computed with the help of Riemann solvers, similarly to how this is done in finite volume discretizations with Godunov's method.

3.2.1 Basis expansion

To be more explicit, we express the state vector $\boldsymbol{u}^K(\boldsymbol{x}, t)$ in each cell K as a linear combination of time-independent, differentiable basis functions $\phi_l^K(\boldsymbol{x})$,

$$\boldsymbol{u}^K(\boldsymbol{x}, t) = \sum_{l=1}^{N} \boldsymbol{w}_l^K(t)\, \phi_l^K(\boldsymbol{x}), \tag{3.5}$$

where the $\boldsymbol{w}_l^K(t)$ are N time dependent weights. Since the expansion is carried out for each component of our state vector separately, the weights $\boldsymbol{w}_l^K$ are really vector-valued quantities with 5 different values in 3D for each basis function l. Each of these components is a single scalar function with support in the cell K.

We decompose our simulation domain into a set of non-overlapping cells of equal size, and we pick tensor-products of Legendre polynomials as basis, so that each cell has a smooth polynomial solution within it. The solution may in general jump across the cell boundaries, and a special treatment is needed for the diffusion equation in this case due to its second spatial derivatives (and likewise for the Navier Stokes equations), which we will briefly specify below. In any case, at a given time the global numerical solution is fully determined by the set of all weights.

In the following, we only consider Cartesian cells of uniform size and a fixed number of basis functions per cell. It is possible to generalise the DG approach to a variety of other cell geometries, to spatially vary the cell size (h-refinement), and to modify the expansion order applied to individual cell's (so-called p-refinement). For more details on DG implementations that realize adaptive mesh refinement, see for example Schaal et al. (2015) and Guillet et al. (2019). For DG methods with local p-refinement see Mossier et al. (2022) and references within.

3.2.2 Time evolution

To evolve the simulation in time we need to derive a way for evolving the time-dependent weights from one time-step to another. Starting with the Euler equations (3.1), we multiply it with a test function, e.g. one of our basis functions ϕ_l, and integrate over a cell K, yielding

$$\int_K \phi_l^K \frac{\partial \boldsymbol{u}}{\partial t} \mathrm{d}\boldsymbol{x} + \int_K \phi_l^K\, \boldsymbol{\nabla} \boldsymbol{F}\, \mathrm{d}\boldsymbol{x} = 0. \tag{3.6}$$

Integrating the second term by parts and using Gauss's theorem we can transform the integral over the cell into an integral over volume and its outer surfaces, respectively, yielding the so-called weak formulation of the hyperbolic conservation laws of the Euler equations:

$$\int_K \phi_l^K \frac{\partial \boldsymbol{u}}{\partial t} \mathrm{d}\boldsymbol{x} + \int_{\partial K} \phi_l^K\, \boldsymbol{F}\, \mathrm{d}\boldsymbol{n} - \int_K \boldsymbol{\nabla} \phi_l^K\, \boldsymbol{F}\, \mathrm{d}\boldsymbol{x} = 0, \tag{3.7}$$

where $|K|$ stands for the volume/area/length of the cell.

Using the orthornomality of our Legendre basis,

$$\int_K \phi_l^K(\boldsymbol{x})\phi_m^K(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \delta_{l,k}|K|, \tag{3.8}$$

we can simplify the integrals and obtain a differential equation for the time evolution of the weights:

$$|K|\frac{\mathrm{d}\boldsymbol{w}_l^K}{\mathrm{d}t} = \int_K \boldsymbol{\nabla}\phi_l^K\, \boldsymbol{F}\,\mathrm{d}\boldsymbol{x} - \int_{\partial K} \phi_l^K\, \boldsymbol{F}^\star(\boldsymbol{u}^+, \boldsymbol{u}^-)\,\mathrm{d}\boldsymbol{n}. \tag{3.9}$$

Here we also considered that the flux function at the surface of cells is not uniquely defined if the states that meet at cell interfaces are discontinuous. We address this by replacing $\boldsymbol{F}(\boldsymbol{u})$ on cell surfaces with a flux function $\boldsymbol{F}^\star(\boldsymbol{u}^+, \boldsymbol{u}^-)$ that depends on both states at the interface, where $\boldsymbol{u}^+$ is the outwards facing state relative to $\boldsymbol{n}$ (from the neighbouring cell), and $\boldsymbol{u}^-$ is the state just inside the cell. We typically use an approximate Riemann solver for determining $\boldsymbol{F}^\star$, but of course an exact Riemann solvers can be used as well. In the remainder of this work, we use the Riemann HLLC solver by Toro (2009) as implemented in the AREPO code (Springel, 2010; Weinberger et al., 2020).

What remains to be done to make an evaluation of Eqn. (3.9) practical is to approximate both the volume and surface integrals numerically. For the integrations, we employ Gaussian quadrature that turns the volume and surface integrals into discrete sums. The number of Gauss points needs to be chosen consistently with the selected expansion order p (see Schaal et al., 2015) such that the L_1-error norm,

$$L_1 = \frac{1}{|K|}\int_K \left| \boldsymbol{u}(\boldsymbol{x}) - \sum_{l=1}^{N} \boldsymbol{w}_l^K\, \phi_l^K(\boldsymbol{x}) \right| \mathrm{d}V, \tag{3.10}$$

of the total approximation error declines as $L_1 \propto h^{-(p+1)}$ with spatial resolution h. Similarly, the time integration method of the differential equation for $\mathrm{d}\boldsymbol{w}_l^K/\mathrm{d}t$ needs to be of sufficiently high order to avoid that time integration errors dominate the total error budget. We choose Runge-Kutta schemes of appropriate order to achieve this goal. For full details, in particular for the location of the Gauss points and for the specific enumeration of the basis functions we have chosen, we refer to our earlier study. There, also other practical aspects, such as the definition of the weights for given initial conditions, are discussed.

3.2.3 Diffusion operator across cell boundaries

To generalize the above approach to treat the full Navier-Stokes equations (hereafter NS), or a general diffusion operator $\nabla \cdot (\varepsilon \nabla \boldsymbol{u})$ that we used in our previous work (Cernetic et al., 2023) to introduce artificial viscosity for shock capturing, we add the corresponding dissipative term as a source term to the basic Euler equation, so that it reads, for example, as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F} = \nabla \cdot (\varepsilon \nabla \boldsymbol{u}), \tag{3.11}$$

with $\boldsymbol{u}$ being the state vector (3.5) and $\boldsymbol{F}$ the flux matrix (3.4).

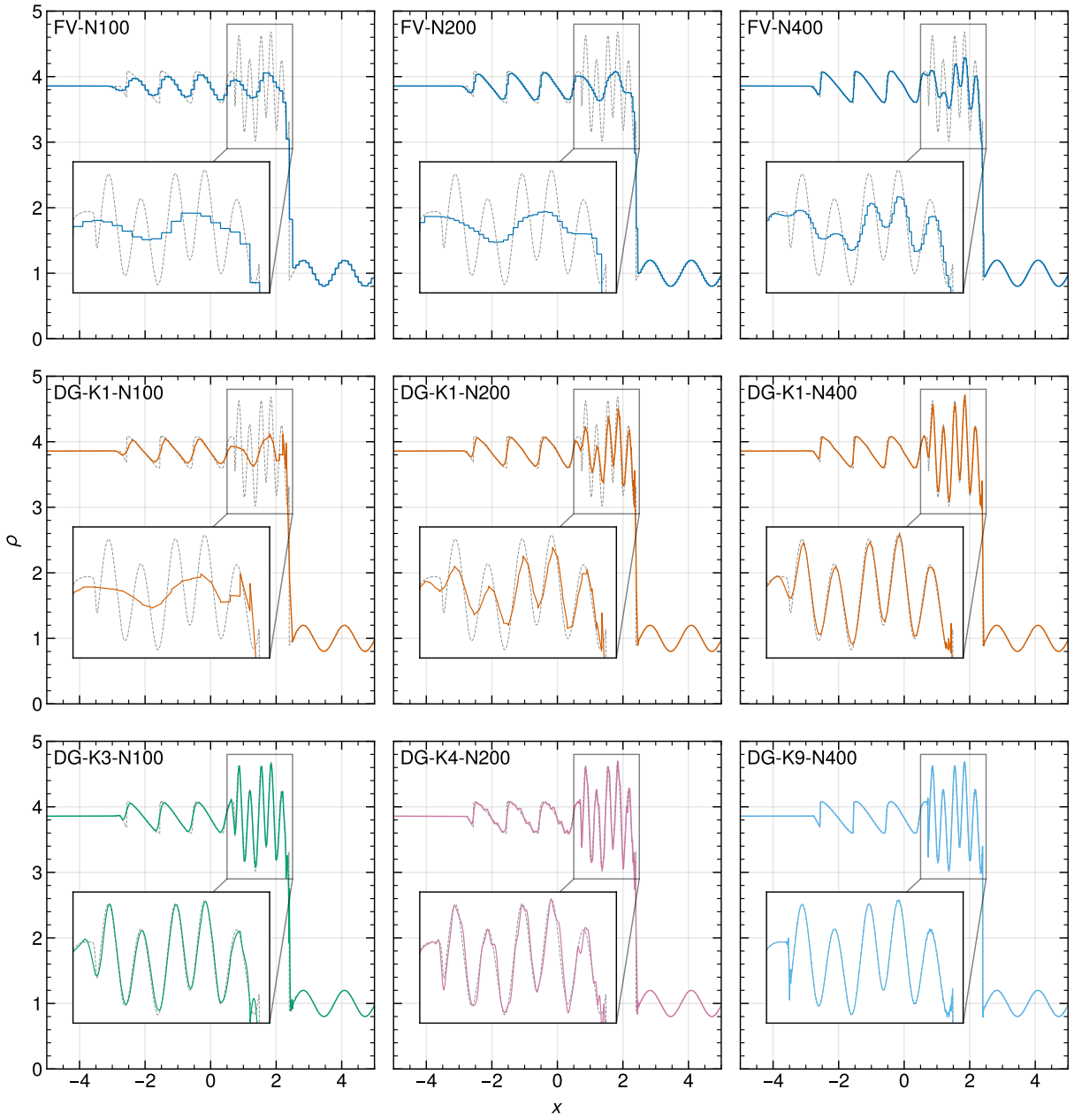


Figure 3.1: Shu-Osher shock interaction test problem at time $t = 1.8$, for different resolutions and numerical schemes. The initial conditions contain a Mach number $\mathcal{M} = 3$ shock wave that is incident on a sinusoidal density perturbation. The top row shows the problem when simulated at different resolutions (as labelled, where the number following 'N' is the number of cells over a domain length of 10 units) with a conventional finite volume (FV) method with piece-wise linear reconstruction. Even with 400 cells, the short-wavelength wiggles (see the enlarged insets) in the solution (dotted line) are only poorly resolved. In the middle row, we show equivalent DG computations at order $k = 1$, i.e. also with a linear expansion inside cells. The results especially for the 200 and 400 cell resolutions are drastically improved. In the bottom row, we extend the results to higher order DG schemes, up to a tenth-order accurate scheme ($k = 9$), demonstrating that our implementation can robustly treat strong shocks at high order thanks to our new artificial viscosity scheme.

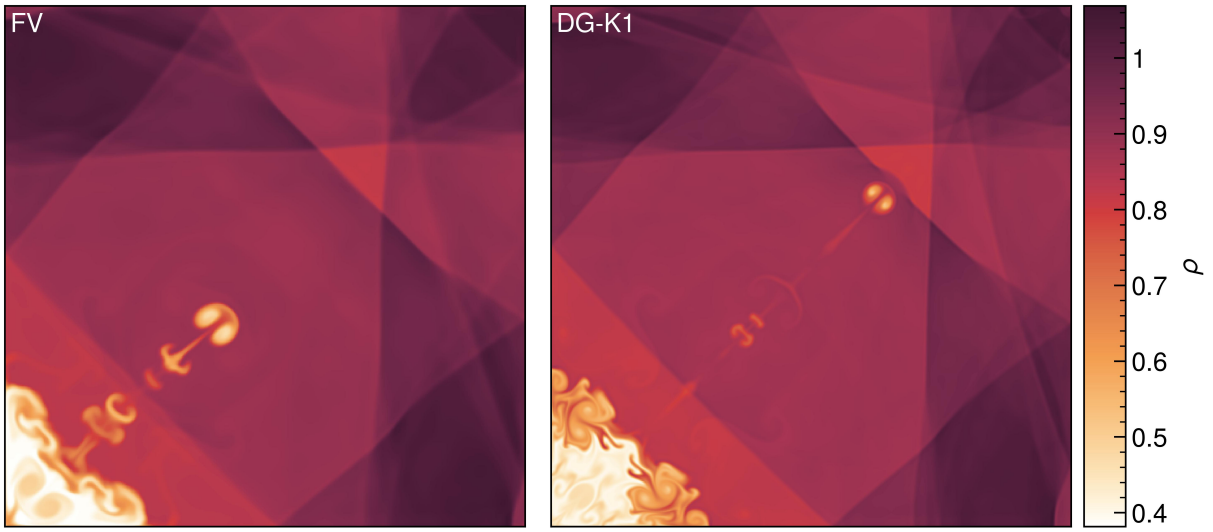


Figure 3.2: Density field of the Liska & Wendroff (2003) implosion test at time $t = 2.5$, simulated with 400×400 cells either with DG at order $k = 1$ (right panel), or with a finite volume scheme (left panel). Both methods describe the fluid with linear functions inside cells. The initial conditions contain a region of strongly reduced density and pressure in the lower left corner. This launches a shock towards the origin which reflects at the reflecting boundaries of the domain. The interaction of the shocks at the corner and the diagonal produces a jet of dense gas along the diagonal direction. The test is very sensitive to numerical diffusion, which tends to limit the length of the diagonal jet. As our results demonstrate, our DG scheme is not only capable of capturing the strong shock interactions while accurately maintaining the symmetry of the system, it also shows clearly less numerical diffusion than the equivalent finite volume scheme.

The crucial difference between the normal Euler equations (3.1) and this dissipative form is the introduction of a second derivative on the right-hand side, which modifies the character of the problem from being purely hyperbolic to an elliptic type, while retaining manifest conversation of mass, momentum and energy. This second derivative can not be readily accommodated in our weight update equation obtained thus far. Recall, the reason we applied integration by parts and the Gauss' theorem going from Eq. (3.6) to Eq. (3.7) was to eliminate the spatial derivative of the fluxes. If we apply the same approach to $\nabla \cdot (\varepsilon \nabla \boldsymbol{u})$ we are still left with one ∇-operator acting on the fluid state.

Our method for addressing this effectively works by constructing a new continuous solution of $\boldsymbol{u}$ across all pairs of adjacent cells. To this end we create a "virtual" cell that overlaps partially or in full with the two constituent cells. By evaluating each cell's weights and projecting them onto the common overlapping basis we obtain the basis of the virtual cell. Note that this projection is a sparse matrix operation in which the new coefficients are a sum of the old expansion coefficients, making the estimation of second derivatives at cell interfaces reasonably efficient.

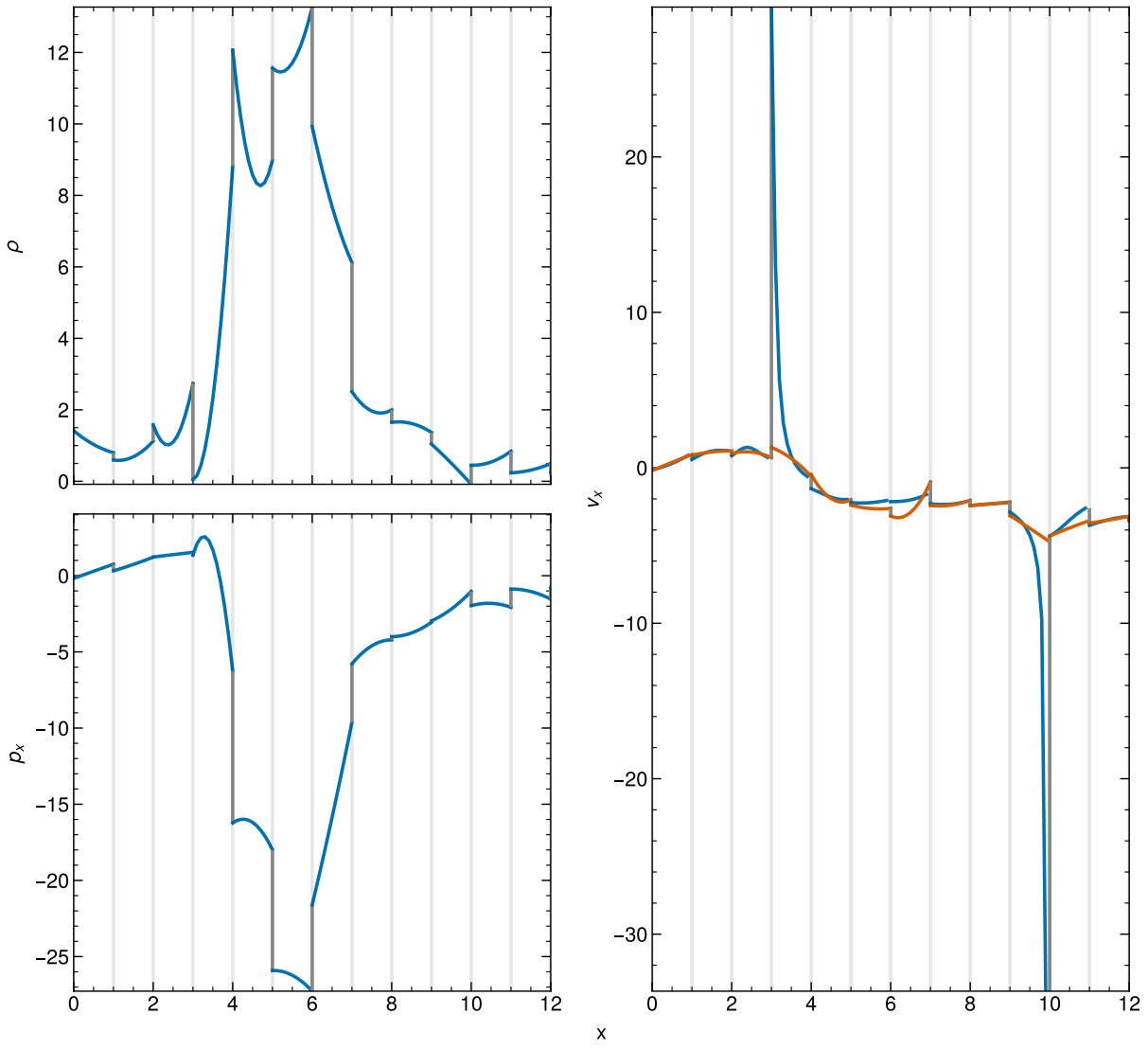


Figure 3.3: Illustration of the occurrence of problematic, extrapolated primitive variable values at cell boundaries when derived naively from the conservative variables. All panels show a skewer through a 3D, driven-turbulence simulation of high Mach number with vertical lines delineating different cells. The upper left panel shows density, the lower left panel shows the momentum p_x along the x-direction. The right panel displays the velocity (blue lines) calculated by taking the ratio of the left panels. This is compared to the velocity calculated with our new method (described in Sec. 3.4), shown in orange. The latter approach projects the velocity itself on the polynomial basis, based on the values attained at the internal Gauss points within a cell.

3.2.4 Parallelisation on GPUs

Compared to ordinary finite volume schemes, DG approaches require the evaluation of polynomial expansions at a variety of Gauss points, and the cell evolution is described not

only by cell averages but instead by multi-valued expansion vectors for each fluid variable. Calculating the time evolution of these high-order weights increases the computational work needed per cell. At the same time, the coupling to neighboring cells at arbitrary order only ever involves surface states. In contrast to finite volume codes, where ever deep stencils are needed for higher and higher order reconstructions. As such the algorithm therefore features a comparatively high computational intensity with only modest communication needs in comparison to high-order finite volume approaches. These characteristics are in principle favourable for reaching a high fraction of the theoretical peak performance on modern computing hardware which operates in a Single Instruction, Multiple Data (SIMD) mode. And since much of the work on different cells can be done fully in parallel, it is attractive to consider GPUs as computational engines for DG methods.

We have therefore developed our DG implementation from the ground up to use GPUs. Otherwise, CPUs can also be used. Parallelisation over multiple GPUs is achieved through the message passing interface (MPI), i.e. clusters of compute nodes each equipped with one or several GPUs can be employed. In principle, our code architecture also allows a mixed operation of CPUs and GPUs, although this is typically not a preferable strategy in practice as their relative speeds will in general not be well matched, and our work-load decomposition between the two is static and fixed at start-up. Full technical details of our code are described in Section 9 of our previous study (Cernetic et al., 2023). In the present work we focus primarily on the algorithmic efficiency of DG for problems involving many shocks and not on absolute code speed. The latter is of course also quite sensitive to implementation details and the employed computing hardware.

3.3 Viscous shock capturing

One important conceptual feature of DG is that there is no source of viscosity in the sub-cell evolution, because DG is designed to evolve a *smooth*, differentiable field of the conservative variables under the *inviscid* Euler equations as accurately as possible. By construction, there is no source of entropy in this evolution. It follows that a true physical discontinuity, in the form of a shock wave in which the inviscid assumption breaks down, cannot be represented correctly – because this would require that entropy is produced by *irreversibly* converting some of the kinetic energy to heat.

In our previous study we have addressed this by introducing an explicit viscosity field that was treated with a special high-order DG solver for a diffusive source term added to the Euler equations (i.e. turning them effectively into a generalized form of the Navier-Stokes equations). This artificial viscosity field could then be used for the purpose of shock capturing, besides optionally adding physical viscosity and/or heat diffusion. To steer the strength of the artificial viscosity, we had introduced both a simple shock sensor based on the rate of local compression and a 'wiggle sensor' that was meant to detect rapid, spurious oscillations in the flow. Each of them could ramp up the local viscosity, while without such a sensor trigger the strength of the artificial viscosity was made to decay again to zero on a short timescale.

We could demonstrate that this approach allowed a capturing of shock waves at sub-cell resolution. Still, this scheme is quite complicated and technically involved, as the treatment of the viscous source function introduces additional computational cost as well as memory overhead. Another disadvantage is that some of the viscosity was effectively added as a type of post hoc damage control, namely only when the solution already exhibited a strongly oscillatory character. The simulation thus first needed to develop a problematic local character before this is "healed" again by supplying needed dissipation, while it would evidently be better to prevent the occurrence of local problems in the first place.

We have therefore reconsidered the parametrisation of our artificial viscosity. One should perhaps first comment that the word "artificial" is really a bit of a misnomer in this context. While we stick to using this term for consistency with the literature, a better name would arguably be "required viscosity", because having no dissipation in a DG-cell that features a shock is physically plainly wrong. Adding the viscosity that needs to be there is hence in principal "natural" not artificial.

In any case, we here resort to a version of the well-known von Neumann-Richtmyer viscosity first described in Von Neumann & Richtmyer (1950), which has been exploited successfully in the field for decades (Wilkins, 1980), and incidentally has also motivated the parametrisation of artificial viscosity commonly employed in smoothed particle hydrodynamics (Monaghan & Gingold, 1983). The von Neumann-Richtmyer viscosity is based on the idea to introduce a viscous pressure Π in rapidly compressing parts of the flow (indicating a region undergoing a shock), and to add it to the ordinary thermal pressure, so that the sum of the two pressures enters in the usual place in the momentum and energy equations. The effect of this will be that the compression is slowed, with kinetic energy being converted to internal energy in an energy-conserving fashion. However, since the excess pressure Π is only added during the compression phase, the produced heat energy does not give rise to the same pressure when the gas can expand again, thus the thermal energy cannot be converted back to kinetic energy in full, unlike for ordinary adiabatic compression and expansion. Such an irreversible conversion of kinetic energy to heat is exactly what happens at a shock, and it is a process that is associated with entropy production.

More explicitly, if we label the entropy per unit mass of the gas through an entropic function, $A = p/\rho^\gamma$, then the Euler equations show that the volume density ρA of the entropic function is a conserved quantity outside of shocks (e.g. Springel & Hernquist, 2002), governed by the additional conservation law

$$\frac{\partial}{\partial t}(\rho A) + \vec{\nabla} \cdot (\rho A \boldsymbol{v}) = 0. \tag{3.12}$$

Adding a viscous pressure in the Euler equations as described above gives rise to

$$\frac{\mathrm{d}A}{\mathrm{d}t} = -\frac{1}{2}\frac{\gamma - 1}{\rho^\gamma}\Pi \nabla \cdot \boldsymbol{v}, \tag{3.13}$$

where $\mathrm{d}/\mathrm{d}t$ is the convective derivative. Hence, a judiciously chosen Π can inject the required entropy.

The basic parametrisation of the von Neumann-Richtmyer viscosity we adopt is the classic form

$$\Pi = \alpha_{\rm visc}\,\rho\left(\frac{h}{p}\right)^2 |\nabla\cdot\boldsymbol{v}|^2, \tag{3.14}$$

for $\nabla\cdot\boldsymbol{v} < 0$, otherwise Π is zero. Here h/p gives the expected spatial resolution of our DG scheme of order p (with h being the cell size). The parameter $\alpha_{\rm visc}$ is dimensionless and roughly determines over how many resolution elements a shock is resolved. Typical values should be in the range $\alpha_{\rm visc} \simeq 1.0 - 3.0$. Note that the precise value will not be important for determining the properties of the post-shock flow, as the total dissipation occurring at a shock is prescribed by the conservation laws, i.e. the effective shock profile auto-adjusts such that that the correct total dissipation occurs. However, the sharpness of the shock and the degree to which there may be residual postshock-oscillations still depend on $\alpha_{\rm visc}$ and the functional form adopted for Π.

The quadratic dependence on $\nabla\cdot\boldsymbol{v}$ proves effective in selectively adding viscosity in shocks, while introducing only negligible viscosity in other places of the flow. However, the above parametrisation can still leave some postshock oscillations downstream of a shock, essentially because the viscosity shuts off too rapidly after passing through the strongest rate of compression. To mitigate this, one can augment the viscosity with a small additional bulk viscosity contribution, of the form

$$\Pi = \beta_{\rm visc}\,\rho c_s\left(\frac{h}{p}\right)|\nabla\cdot\boldsymbol{v}|, \tag{3.15}$$

where c_s is the sound speed. As the latter increases in a shock, this preferentially affects the shock region past the maximum compression rate, and thus helps to damp out post-shock oscillations. This viscosity parametrisation is however less specific than that with a quadratic dependence on the velocity divergence, and hence can lead to an unwanted damping of flow features such as sounds waves when used with a non-negligible value of $\beta_{\rm visc}$. We thus typically either set $\beta_{\rm visc} = 0$, or choose a value around $\beta_{\rm visc} \simeq 0.1\,\alpha_{\rm visc}$.

In most practical applications we have found that $\alpha_{\rm visc} \sim 2.0$ and $\beta_{\rm visc} \sim 0.2$ provide a good compromise between stability, narrowness of shocks, and the damping of postshock oscillations, largely independent of flow type and DG-order p. To protect against the possibility that the viscous force applied in one timestep could become so large that it would *reverse* the compressive motion, we limit Π against a maximum value of

$$\Pi_{\rm max} = \frac{1}{2}\rho\left(\frac{h}{p}\right)^2\frac{|\nabla\cdot\boldsymbol{v}|}{\Delta t}, \tag{3.16}$$

where Δt is the prescribed timestep at the beginning of the step. A similar type of limiter is used in the SPH code GADGET (Springel et al., 2001).

In practical terms, we simply add Π to the pressure computed for the fluid state at all *internal* Gauss-points used in the volume integration over the flux function, on the grounds that here the inviscid Euler equations need to be augmented with dissipative terms to introduce entropy production where necessary. Π is calculated using the quadrature point

specific ρ and $|\nabla \cdot \boldsymbol{v}|$. In the surface integrals, we do not introduce any artificial viscosity. This is because the Riemann solver computes a wave solution that injects entropy into the downstream cell when appropriate, or in other words, here the inviscid Euler equations are implicitly already supplemented with a means to irreversibly convert kinetic energy to heat.

Recall for comparison that in finite volume methods there are two ways to produce entropy. One is through the Riemann problems solved at cell interfaces, and this is present in equivalent form in our DG approach as just mentioned. The other is through the implicit averaging step that is done at the end of every timestep, where only the average state of cells is retained (to be followed by a reconstruction step from scratch the the beginning of the next step). This averaging step also produces entropy in general, for example when it mixes gas phases of different temperature that have streamed into a cell. The Discontinuous Galerkin approach misses this source of entropy (likewise this is absent in SPH). In many situations this is advantageous, for example in pure advection, while for shocks it is an impediment – here DG needs to be augmented with a suitable channel to entropy production, and this is exactly what we achieve with the artificial viscosity.

In order to be able to directly compare our DG implementation with artificial viscosity shock-capturing to a classic second-order accurate finite volume (FV) scheme, we have added such a scheme to our code as well. The FV approach can in essence be viewed as a DG-scheme of order $K = 0$, i.e. where only cell-averages of the conserved variables are stored, but which is augmented with a reconstruction step that computes linear slopes of the fluid variables for each cell (through piece-wise linear reconstruction), raising it again to the description of the fluid as done by a $K = 1$ DG-scheme. Then these slopes are used to compute the interface states left and right of all cell interfaces, which are in turn fed to the Riemann solver to compute the fluxes between cells. Unlike in a DG scheme of order $K = 1$, the slopes are not evolved in time, but rather discarded after every step and then re-estimated. The FV scheme therefore does not need to compute fluxes inside cells, unlike the corresponding DG scheme.

For carrying out the piece-wise linear reconstruction in our FV scheme, we estimate the slopes for the primitive variables in each spatial direction, preventing over- and undershoots with a monotonised central slope limiter. This limiter still has the so-called total variation diminishing (TVD) property, but it is substantially less diffusive than, for example, the minmod limiter. Note, however, that the scheme is not guaranteed to be positivity preserving, so that in simulations with extreme density variations (such as in supersonic turbulence) we have introduced an additional slope-liming criterion based on a troubled cell indicator in order to be able to robustly run simulations in all situations. In particular, if a cell ends up with negative density in a timestep, such a cell is flagged as a 'troubled cell' for this step, meaning that its slope estimate is set to zero, and the corresponding timestep calculation is simply repeated. Because for flat slopes positivity can be guaranteed for reasonable timesteps, this then allows the simulation to proceed.

For our general DG implementation, we employ a similar approach to guarantee positivity and code stability in case challenging local flow situations should arise. We here verify positivity at all Gauss points also in all intermediate steps of the Runge-Kutta time

integration. If negative density or pressure values occurs, we apply a positivity limiter that in the first instance tries to scale all high-order weights such that the negative values can be avoided. The computation of the timestep is then repeated. If even a flat expansion inside a cell is not able to rectify the situation, we reduce the timestep size and try again.

Before closing this section, we consider two illustrative tests of the new artificial viscosity treatment in DG. In Figure 3.1 the well-known Shu-Osher shock tube problem (Shu & Osher, 1989, their test problem 8). This describes the interaction of a strong incoming Mach $\mathcal{M} = 3$ shock wave with an adiabatic standing wave in density. The result is a complicated oscillatory pattern in the downstream region of the shock, which is challenging for numerical schemes to resolve accurately. The initial conditions are given by, for $z < -4$, as $\rho = 3.857143$, $v_x = 2.629369$, and $P = 10.33333$, and for $x \geq -4$ as $\rho = 1 + 0.2\sin(5x)$, $v_x = 0$ and $P = 1$, with and adiabatic index $\gamma = 1.4$.

The solution domain at $t = 1.8$ has five zones, from left to right they are the flat initial section, followed by wide waves, followed by very sharp narrow waves, then a sharp density jump and finishing with a set of wide sine-like oscillations. At $N_{\text{cell}} = 100$ the finite volume scheme with linear reconstruction is able to reliably resolve zone one and five. The same order DG method shown in the middle row performs slightly better in the wide waves section and much better resolving the last smooth waves section. Going to $k = 3$ which results in a 4-th order method shown in the bottom row, the blue line only has slight deviations from the analytic solution in zone two where it fails to resolve the sharp edges of zig-zag waves. Moving on to $N_{\text{cell}} = 200$, the FV method resolves the wide zig-zag waves better, while the sharp narrow waves do not improve significantly. On the other hand, the same zone improves significantly with $k = 1$ almost fully resolving the narrow waves. At $N_{\text{cell}} = 400$, FV has basically fully resolved the zig-zag waves, akin to DG. But it still fails to resolve zone three with its sharp narrow waves. The same order DG method resolves the waves much better while also accounting for the very sharp upper bump between the waves and the shock, albeit with some ringing. Moving on to the last row where we show the performance of high order DG methods to showcase their stability. It is interesting to see the ringing at $N_{\text{cell}} = 200$ with $k = 5$ at the zig-zag waves and slightly at the narrow waves. The same ringing is missing at in an odd order method at $N_{\text{cell}} = 400$ with $k = 9$. This highly refined method does not suffer from any ringing, showcasing the robustness of our DG also in the presence of strong shocks at high order, made possible here by the use of the artificial viscosity.

As second problem we consider the implosion test of Liska & Wendroff (2003), which consists of a square-shaped 2D domain of extension $[0, 0.3]^2$ with reflective boundary conditions in which the region $x+y < 0.15$ has initially density $\rho = 0.125$ and pressure $P = 0.14$, while all the other gas has $\rho = 1$ and $P = 1.0$. The gas is at rest in the beginning and has an adiabatic index of $\gamma = 1.4$. When the system evolves in time, the region of strongly reduced density and pressure in the lower left corner produces a shock towards the origin which undergoes a double reflection at the domain walls. The interaction of the shocks at the corner and the diagonal produces a jet of dense gas along the diagonal direction. In addition, further shocks bounce off at the opposite sides of the domain, and the Richtmyer-Meshkov instability produces intricate flow features as shocks cross contact discontinuities

in the problem.

In Figure 3.2 we show the state at time $t = 2.5$ for a DG simulation with 400×400 cells at order $k = 1$ (i.e. with a linear run of the fluid variables inside cells), and we compare to an equivalent finite volume simulation with the same number of cells. This test is very sensitive to numerical diffusion, which tends to limit the length of the diagonal jet. The comparison highlights that the DG scheme is able to accurately capture the strong shocks in the system based on the artificial viscosity treatment, and it is does so with noticeable less numerical diffusivity than the finite volume scheme. In fact, our second-order DG result appears close to or even better than the third-order finite result based on piece-wise parabolic reconstruction reported by Stone et al. (2008) for the ATHENA code. We have also carried out this test with higher order DG schemes and higher cell resolutions (not shown), which reveal still finer detail in the fluid evolution, as expected.

3.4 Primitive variables at cell interfaces

In simulations of supersonic turbulence, extreme density contrasts and networks of strong, interacting and overlapping shocks are encountered that put any numerical scheme to a stress-test in terms of robustness. We already mentioned that this is even the case for simple finite volume schemes that use piece-wise linear reconstructions, but these problems become even more acute in high-order approaches such as our DG scheme. Only extremely diffusive, first order schemes are free of such troubles.

One particular issue we noticed in supersonic turbulence calculations with DG is that our default approach to compute the primitive variables at the cell interfaces can sometimes produce extreme velocity values that are basically unphysical. After being processed by the Riemann solver, the resulting inaccurate fluxes then pollute the solutions inside the cells. The problem originates in our definition of the velocity field inside cells as ratio of the polynomial describing the momentum density and the polynomial describing the density, which both are evolved separately. The ratio of two polynomials is a rational function that can be well outside the space of our underlying polynomial basis functions. While the values obtained at the interior Gauss points should be reasonably well behaved, because these velocities enter the internal flux computation, the *extrapolated* values at cell interfaces are much less well constrained. And indeed, in cells that show large excursions of density and/or momentum density from the mean (perhaps even in opposite directions), the velocities one obtains at cell interfaces by dividing the two polynomial expansions can become quite extreme, especially if the density itself approaches very small values.

This is illustrated in Figure 3.3 along a one-dimensional skewer through a low-resolution DG simulation (32^3 cells with expansion order $K = 2$) of Mach number $\mathcal{M} \simeq 3$ turbulence. The two panels on the left show the mass density and the x-component of the momentum density, respectively, while the right panel shows their ratio (black), i.e. the inferred v_x-velocity. The cell boundaries are indicated with vertical dotted lines.

It is clearly seen, and expected, that the polynomial solutions inside cells can in general give rise to discontinuous jumps of the conserved and primitive variables at cell boundaries.

These jumps are no problem for the DG scheme, and they preferentially tend to occur when there are shocks and contact discontinuities. However, what is nevertheless a problem are the extreme velocity values that can result at cell interfaces when the momentum and density values are divided by each other, as seen for example in the two cells between $x = 6$ and $x = 7$, and $x = 11$ and $x = 12$, respectively.

We have solved this issue by defining reconstructed primitive variable fields in each cell, simply by computing a polynomial expansion of the primitive variables themselves based on the values they assume at the internal Gauss points of the cells. The procedure for obtaining the corresponding coefficients is akin to how one would project initial conditions onto the polynomial expansion by exploiting the completeness of the basis. The projection of an arbitrary field $\boldsymbol{f}(\boldsymbol{x})$ onto the basis functions ϕ_l^K of a cell can be done by

$$\boldsymbol{w}_l^K = \frac{1}{|K|} \int_K \boldsymbol{f}\, \phi_l^K \, \mathrm{d}V, \tag{3.17}$$

and the volume integration can be approximated by Gaussian quadrature. We can now set for $\boldsymbol{f}$ the vector of primitive variable fields expressed through the polynomial expansion of the conserved variables (i.e. for the velocity this will be a rational function obtained as the ratio of momentum density and mass density), and use our standard order for approximating the volume integral with Gaussian quadrature, so that ultimately only the values of the conservative variables at the Gauss points enter. This will then mean that we, for example, obtain an approximation for the velocity field by a polynomial of the same order as used for the conservative variables, and this polynomial goes in principle[1] through the velocity values obtained at the Gauss points, whereas the extrapolated values at the cell interfaces are now bounded and in general better behaved than obtained for the ratio of momentum density and mass density at these same points. Note that for the density field itself, this procedure just returns the same field again, because the density is already a representable polynomial function at the given order.

When we use these 'polynomial extrapolations' for the flux computation at cell interfaces we find that this drastically improves the robustness and the quality of results for our supersonic turbulence and strong shock simulations, whereas for all smooth problems it does not make any tangible difference. Figure 3.3 illustrates this clearly by also including the projected velocity field obtained in this fashion. At those cell interfaces where the ratio of momentum and mass density produced large excursions of the predicted velocities the extrapolations obtained from the projected velocity field are much more reasonable and well behaved. Everywhere else there is no substantial difference.

[1] Since in 2D and 3D we discard in the tensor product of Legendre polynomials cross terms of higher order than imposed in 1D, this is not exactly true, and the polynomial projection in essence entails some small level of smoothing.

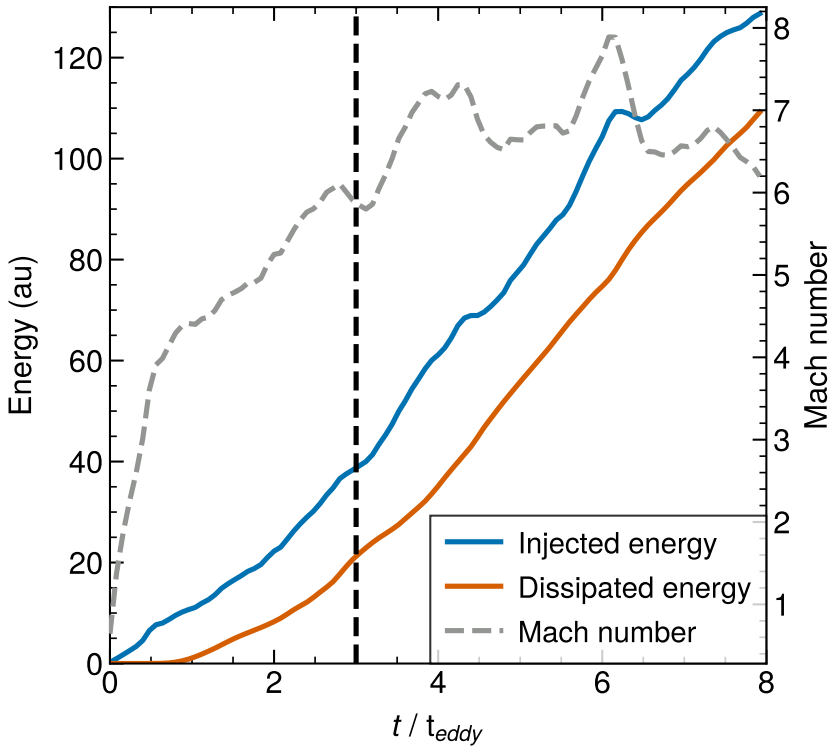


Figure 3.4: Cumulative injected and dissipated energy, as well as global volume averaged Mach number, as a function of time in one of our driven turbulence simulations. The vertical dashed line indicates the time at which we start our power spectra measurements. The gas is initially at rest, and put into motion by the driving. Eventually, energy injection is balanced by dissipation in a time-averaged fashion, and the difference between the cumulative injected and dissipated energy is reflected in the kinetic energy as measured by the Mach number.

3.5 Driving and measuring turbulence

In the following, we collect the definitions of some basic quantities to characterize the statistical properties of turbulence and describe our method for driving turbulence. We also detail our measurement techniques for turbulence-related quantities that we examine later on.

3.5.1 Basic statistics of supersonic and subsonic turbulence

The Mach number of turbulence is often defined as

$$\mathcal{M} = \left\langle \boldsymbol{v}^2/c_s^2 \right\rangle^{1/2}, \tag{3.18}$$

which is a volume-weighted quantity that only depends on the velocity field $\boldsymbol{v}$ in units of the sound-speed c_s. It is also possible to define a density-weighted Mach number, given by $\mathcal{M}_\rho = \left\langle \rho \boldsymbol{v}^2/\overline{\rho} \right\rangle^{1/2}/c_s$, which can also be expressed in terms of the total kinetic energy

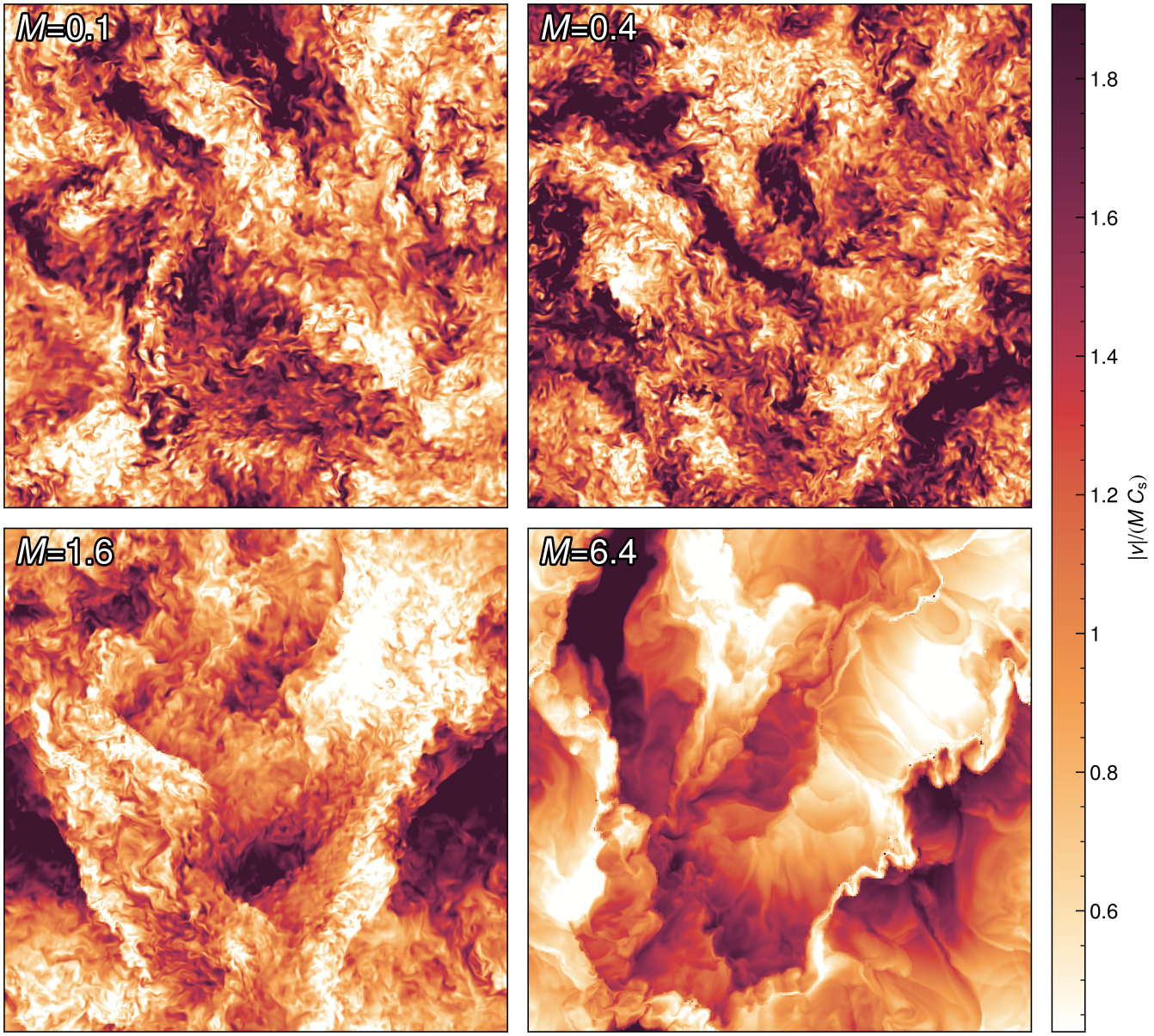


Figure 3.5: Slices through the turbulent velocity field of simulations with different Mach number, here $\mathcal{M} = 0.1$, $\mathcal{M} = 0.4$, $\mathcal{M} = 1.6$, and $\mathcal{M} = 6.4$, as labelled. In each case, the color map shows the velocity amplitude $|v| = (v_x^2 + v_y^2 + v_z^2)^{1/2}$ in units of the corresponding characteristic velocity, here taken as the Mach number times the sound speed. For definiteness, the DG calculations have used 256^3 cells and $k = 2$, and each panel shows the state after the same number of eddy turnover times after the start of the simulations. The subsonic simulations show a nearly self-similar behaviour, as expected for this setup. However, as we transition into the supersonic regime, it is evident that the character of the turbulence qualitatively changes.

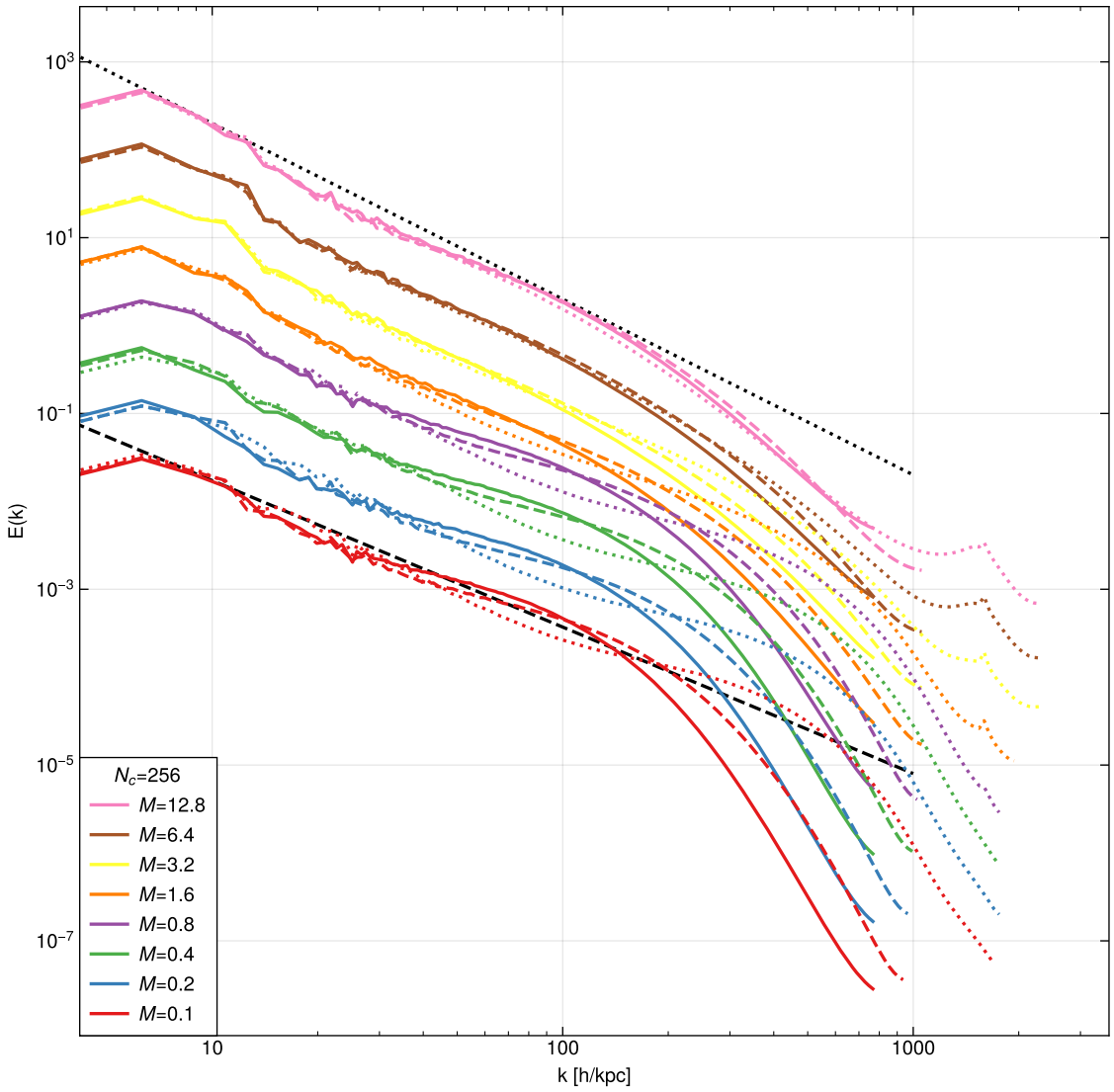


Figure 3.6: Velocity power spectra for different turbulent Mach numbers, from the subsonic to the highly supersonic regime, as labelled. For each driving strength, we compare DG simulations with order $p = 2$ (dashed) and $p = 3$ (dotted) with corresponding finite volume simulation (solid). The black dashed line indicates the Kolmogorov $E(k) \propto k^{-5/3}$ power-law slope, indicative of the subsonic cascade, whereas the dotted black line shows the Burgers $E(k) \propto k^{-2}$ scaling indicative of supersonic turbulence where dissipation is part of the self-similar cascade. The simulations here use only 256^3 cells and thus have a fairly limited dynamic range that can only resolve a very small part of the turbulent cascade before entering the dissipative regime. Nevertheless, the sequence clearly shows a steeping of the slope towards the supersonic regime, marking the transition from Kolmogorov to Burgers turbulence. Also, the second-order DG runs can resolve the turbulence to higher wave number than the second-order accurate finite volume scheme, reflecting DG's higher accuracy and reduced numerical dissipation. Interestingly, while third-order DG likewise does better than second-order DG in the subsonic regime, this advantage nearly vanishes in the supersonic regime.

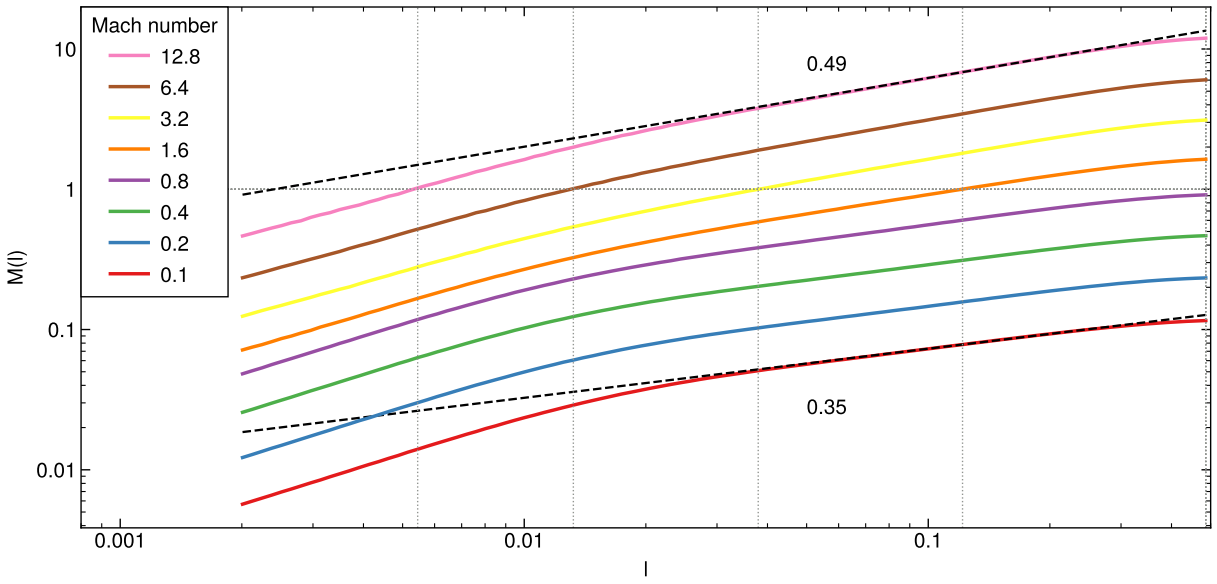


Figure 3.7: Velocity structure function for different turbulent Mach numbers, from the subsonic to the highly supersonic regime, as labelled. For each driving strength, we show DG simulations with order $p = 2$. The black dashed lines indicate fits done between $0.1 < l < 0.25$ for the most subsonic and the most highly supersonic runs. The vertical and horizontal dotted grey lines indicate the super- to subsonic transitions for simulations where it happens. The simulations here use only 256^3 cells and thus have a fairly limited dynamic range that can only resolve a very small part of the turbulent cascade before entering the dissipative regime. Nevertheless, the sequence clearly shows a steeping of the slope towards the supersonic regime, marking the transition from Kolmogorov to Burgers turbulence. In particular, we measure slopes of 0.35 and 0.49 for our two fits, quite close to the expected scalings of 1/3 and 1/2 for subsonic and supersonic turbulence, respectively.

of the flow, $\mathcal{M}_\rho = (2E_{\rm kin}/M_{\rm tot})^{1/2}/c_s$. While for subsonic turbulence both measures are equal, $\mathcal{M} \simeq \mathcal{M}_\rho$, for supersonic turbulence there is a small difference, with $\mathcal{M}_\rho$ being generally slightly smaller than $\mathcal{M}$. Pan et al. (2022) cite $\mathcal{M}_\rho \simeq \mathcal{M}^{0.96}$ for the relation between the two quantities in the supersonic regime, which matches our own findings very closely. Note that we consider in this chapter only isothermal flows in which c_s is constant, which simplifies the discussion considerably.

The characteristic velocity, $v(l)$, of the turbulent velocity field is scale-dependent, and we define this quantity (following Federrath et al., 2021) in terms of the total second-order velocity structure function, as follows

$$v(l) = \frac{1}{2}\left[\left\langle |\boldsymbol{v}(\boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x}+\boldsymbol{l})|^2 \right\rangle_{\boldsymbol{x},l=|\boldsymbol{l}|}\right]^{1/2}, \tag{3.19}$$

where we average over a large number of random pairs that are separated by a fixed distance $l = |\boldsymbol{l}|$. Based on this quantity, we can also define a scale-dependent Mach

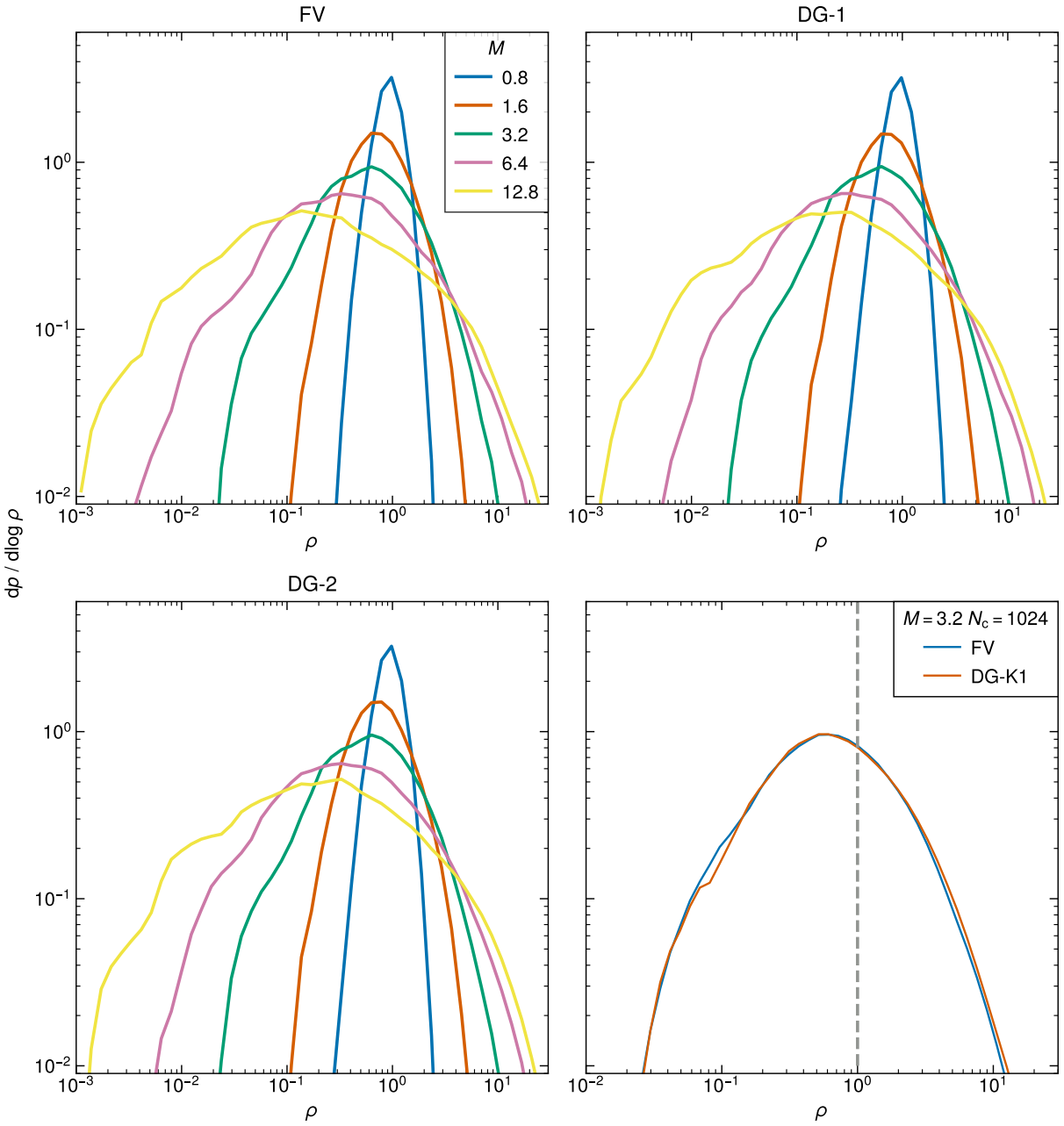


Figure 3.8: Density probability distribution functions (PDFs) in different turbulence simulations, carried out for a variety of Mach numbers and different numerical schemes. In the top two and the bottom left panel, we compare FV, DG at order $k = 1$, and DG at order $k = 2$, as labelled, for a suite of 256^3 simulations at Mach numbers from 0.8 to 12.8. All three numerical schemes show a qualitatively very similar behaviour in which the shape of the density PDF transitions from an approximately normal form in density in the subsonic regime to a log-normal shape in the supersonic regime (note that we use $\log_{10}$ in the PDF's vertical normalization), with a width that grows with Mach number. The bottom right panel compares PDFs at a fixed Mach number of $\mathcal{M} = 3.2$ for higher resolution runs of 1024^3 cells carried out with FV and DG-1. Here we see that the PDFs are not identical after all, but that the DG scheme is able to resolve slightly higher densities than the corresponding FV scheme.

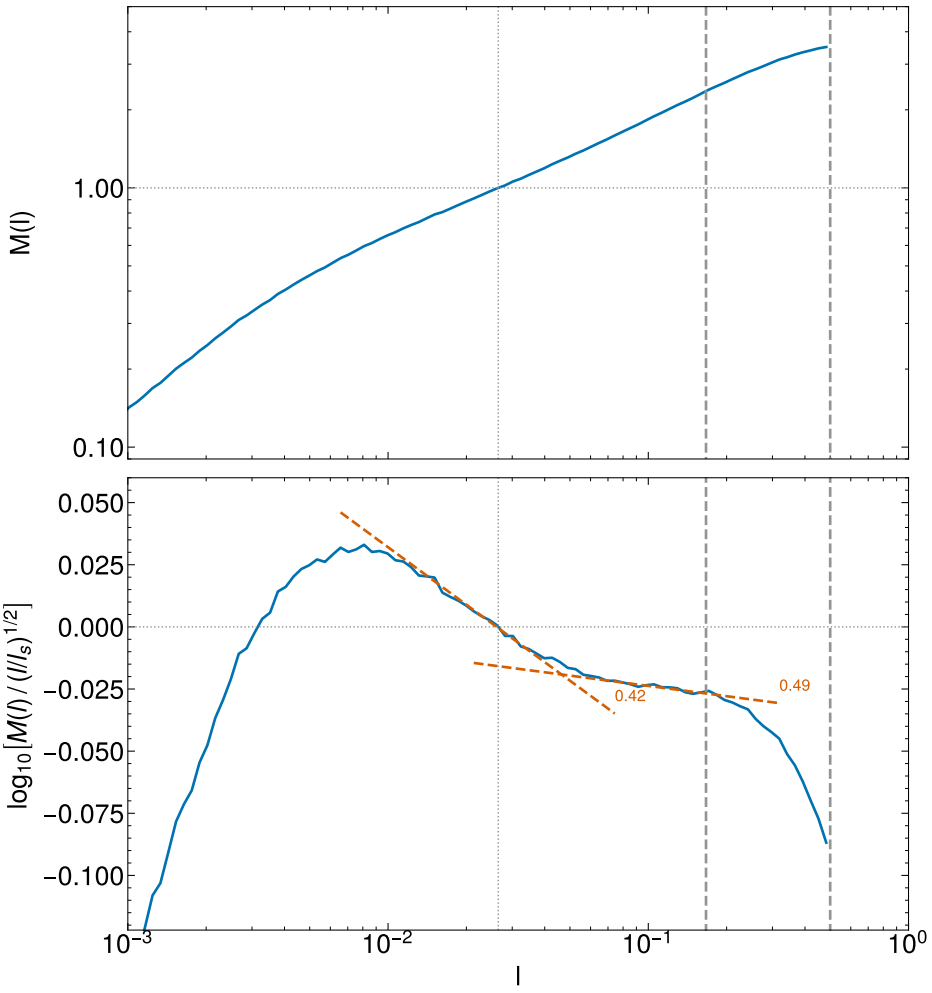


Figure 3.9: Velocity structure function (top panel) for a high-resolution DG run with 1024^3 cells and $k = 1$, for driven turbulence with Mach number $\mathcal{M} = 3.2$. For pair distances equal to half the box size (right-most dashed vertical line), the structure function starts out at values close to the box-averaged Mach number. From this driving scale, it takes until at least three times smaller scales (marked by the middle dashed vertical line) before a self-similar turbulent cascade develops. The structure function then first drops relatively steeply towards smaller scales, close to the expected $M(l) \propto l^{1/2}$ scaling for Burgers turbulence. Around the sonic point at l_s, where $M(l_s) = 1$, the scaling flattens as the turbulence transitions into the subsonic regime. Here a scaling $M(l) \propto l^{1/3}$ would be expected if an extended inertial range is present, until a strong steeping sets in when the dissipation regime is entered. The bottom panel shows the velocity structure function in a compensated form, where it is multiplied by the factor $(l/l_s)^{-0.5}$ which brings out subtle shape difference more clearly. Right when the supersonic turbulence cascade sets in, we measure a slope of 0.49 for $M(l)$, close to the expectation. Furthermore, there is a clear break around the sonic scale where the structure function flattens. Out fit in this region returns a slope of $\simeq 0.42$, somewhat steeper than expected. However, this is not really surprising as the still fairly limited dynamic range of this calculation and the influence of the bottleneck effect are likely causes for this small difference.

number, $\mathcal{M}(l) = v(l)/c_s$. On the largest scales, $l = L/2$, where L is the box size, we expect the velocities of the pairs to be antiparallel on average, reflecting the corresponding property of the driving field, so that there $\mathcal{M}(l)$ should approach the total Mach number of the simulation box.

Previous work has demonstrated a scaling of $v(l) \propto l^\alpha$ with $\alpha \simeq 1/2$ in the supersonic regime, while this flattens to $\alpha = 1/3$ in the subsonic regime. If turbulence is supersonic on the largest length scales, we thus expect the existence of a "sonic scale" l_s where the characteristic velocities have fallen to the sound speed, with $\mathcal{M}(l_s) = 1$. Based on the velocity scaling in the supersonic regime, we expect this roughly for $l_s = L_{\rm inj}/\mathcal{M}^2$, or in terms of wave number, for

$$k_s \simeq k_{\rm inj}\mathcal{M}^2. \tag{3.20}$$

This estimate assumes that the supersonic cascade is already well developed right at the injection scale, which is however typically not the case in practice as some range of scales is required before the self-similarity of the cascade is fully established. In any case, unambiguously identifying the sonic point in a turbulence calculation is challenging as it requires to resolve an inertial range *both* in the supersonic regime and in the subsonic regime, which demands very high dynamic range. Federrath et al. (2021) have recently accomplished this in a simulation of ground-breaking size, using a grid size of 10048^3 cells. We shall later try to identify the sonic scale in DG simulations of considerably smaller size.

Besides characterizing the statistics of the velocity field in real-space through structure functions, it is also common to consider its correlation functions, for example the two-point correlations $\langle \boldsymbol{v}(\boldsymbol{x}+\boldsymbol{l})\,\boldsymbol{v}(\boldsymbol{x})\rangle_{\boldsymbol{x}}$ and its Fourier-transform, the velocity power spectrum. The latter can be defined as

$$E_v(\boldsymbol{k}) = \left(\frac{2\pi}{L}\right)^3 |\hat{\boldsymbol{v}}(\boldsymbol{k})|^2, \tag{3.21}$$

where $\hat{\boldsymbol{v}}$ is the Fourier transform of the velocity field. For a statistically isotropic velocity field, it is customary to define the $\boldsymbol{k}$-shell averaged energy spectrum $E(k)$ through

$$E(k) = 4\pi k^2 \left\langle E_v(\boldsymbol{k})\right\rangle . \tag{3.22}$$

The total velocity dispersion is then given as the integral over $E(k)$. In particular we have

$$\mathcal{M} = \frac{1}{c_s^2}\int E(k)\,\mathrm{d}k. \tag{3.23}$$

In the subsonic regime, we expect the Kolmogorov (1941) scaling $E(k) \propto k^{-5/3}$ of the velocity power spectrum, while in the supersonic regime this is expected to steepen to Burgers (1948) turbulence with $E(k) \propto k^{-2}$.

3.5.2 Driving isothermal turbulence

We drive turbulence following the same approach as in our previous work on subsonic turbulence (Cernetic et al., 2023) which in turn follows closely the procedure described in

many previous works, such as Schmidt et al. (2006); Federrath et al. (2008, 2009, 2010); Price & Federrath (2010) and Bauer & Springel (2012).

The acceleration field is constructed in Fourier space between the fundamental mode of the box, $k_{\rm min} = 2\pi/L$, and $k_{\rm max} = 4\pi/L = 2k_{\rm min}$, with Fourier mode phases chosen at random from an Ornstein–Uhlenbeck process. As injection scale we can thus define $k_{\rm inj} \simeq k_{\rm max}$, corresponding to half the box size in real space. The Ornstein–Uhlenbeck process is used because it is temporally homogeneous, meaning its variance and mean remain constant over time. This type of frequent but correlated driving results it a semi-stationary turbulent field which simplifies its sampling. The randomly chosen phases are updated every timestep Δt, yielding a discrete time evolution update prescription for the Fourier phases $\boldsymbol{x}_t$, as follows:

$$\boldsymbol{x}_t = f\,\boldsymbol{x}_{t-\Delta t} + \sigma\sqrt{(1-f^2)}\,\boldsymbol{z}_n, \tag{3.24}$$

where f is a decay factor defined as $f = \exp(-\Delta t/t_c)$, with t_c being the correlation timescale. $\boldsymbol{z}_n$ is a Gaussian random variable and σ is the variance.

Through the use of a Helmholtz decomposition, the driving can be made either fully solenoidal, fully compressive, or a combination of the two. To stay consistent with our previous work on subsonic turbulence (Cernetic et al., 2023) we retain the same purely solenoidal driving. In the subsonic regime, compression modes created by compressive driving would result in sound waves propagating through the simulation. Such large-scale sound waves start coupling to smaller scales only when their non-linear steepening starts to dominate. For supersonic turbulence, compressive driving has however a more important influence on the properties of turbulence (e.g Federrath et al., 2008, 2010; Federrath, 2013).

The driving has three free parameters which have to be chosen carefully to quickly establish a quasi-stationary turbulent field that faithfully represents the statistical properties of turbulence at the intended Mach number. We can define the eddy turn-over timescale on the injection scale as

$$T = \frac{L}{2c_s\mathcal{M}}. \tag{3.25}$$

The correlation timescale t_c in the driving prescription is the characteristic lifetime of Fourier modes of the driving field, and thus should ideally be of the order or slightly smaller than the eddy turnover time. Based on this we set the correlation timescale as $t_c \simeq T$. We furthermore set the mode update frequency Δt to be 100 times smaller than t_c to assure a smooth transition from one mode to the next.

The third parameter in Eqn. (3.24), σ, determines the strength of the turbulent driving, and as such the achieved Mach number. To get intuition for this parameter, let us consider the relation between the driving strength and the Mach number in the quasi-stationary end state. We start with the energy injection rate per unit mass, ϵ, which scales as

$$\epsilon \propto \sigma^2\,\Delta t \tag{3.26}$$

based on the driving prescription itself. Guided by this expression, we define the strength of the driving through a parameter

$$E_{\rm inj} = \sigma^2\,t_c, \tag{3.27}$$

and express σ in terms of $E_{\rm inj}$. Then the achieved energy injection rate scales linearly with the prescribed parameter $E_{\rm inj}$ as

$$\epsilon \propto E_{\rm inj}, \tag{3.28}$$

approximately independently of t_c. In the regime of subsonic Kolmogorov turbulence, the driving creates characteristic velocities that are expected to scale with length scale and energy injection rate as

$$v(l) \propto (\epsilon l)^{1/3}, \tag{3.29}$$

which means that the achieved Mach number varies as

$$\mathcal{M} \propto \epsilon^{1/3} \propto E_{\rm inj}^{1/3}. \tag{3.30}$$

For doubling the Mach number, we thus need to triple our driving strength $E_{\rm inj}$ while t_c and Δt should be halved.

After driving sets in from gas at rest, turbulence tends to become fully developed only for times $t \geq 2T$. We thus analyze turbulence by averaging the results for a large number of outputs between $3T < t < 8T$ (as, e.g., in Federrath et al., 2021), which gives us enough independent samplings of the box to get robust and converged results with respect to temporal averaging. We note that it also requires some range of scales between the driving scale and the onset of a fully developed self-similar turbulent cascade. One can therefore not expect to immediately obtain proper turbulent scaling right at the scale where the driving ends, but rather needs to go to somewhat smaller scales. For example, Federrath et al. (2021) conservatively estimate that the turbulent supersonic cascade becomes fully developed for $l < L/8$ when the driving is centered at a scale of $L/2$, which is similar to our work.

We note that we typically not simulate an isothermal gas directly in this work, but rather one with an adiabatic index of $\gamma = 1.0001$. After every timestep, we restore a uniform temperature by extracting (or adding) thermal energy as needed. This allows us to measure the actual energy injection rate by determining the volume integral of the work the external driving field does on the gas, and a dissipation rate by accounting for the energy we need to extract to maintain a uniform temperature. The difference between these time integrated rates is then the instantaneous total turbulent kinetic energy of the gas if it started from rest. In Figure 3.4 we show an example of the time evolution of the total Mach number and the cumulative injected and dissipated energies for a turbulence simulation with Mach number $\mathcal{M} \simeq 6.4$. We see that the cumulative injected energy grows approximately linearly with time, and this evolution is tracked by the dissipated energy, albeit with some time delay. Dissipation only starts to set in after about one eddy turnover time, while it takes until about $t \simeq 3\,t_{\rm eddy}$ before a quasi-stationary turbulent state has developed where the Mach number does not grow anymore but rather fluctuates around a long-term average value.

3.5.3 Measuring structure functions and power spectra

To measure velocity structure functions, we define a logarithmic set of radial bins between half the box size as maximum distance, and the nominal resolution limit, $L/[(k+1)N]$, as minimum distance, where L is the box size, N is the number of cells per dimension, and k the DG order. Typically we adopt 100 such bins. Upon each output time, we then draw for each bin a fixed number (typically 10^5) of random positions in the box, plus a set of random directions uniformly distributed over the unit sphere. A second position paired with each point is then determined relative to the first one based on these directions with the distance of the corresponding radial bin, taking periodic wrap-around in the simulation box into account. The velocity values at the two selected coordinates are then evaluated based on the polynomial expansions of the cells the points fall into. The corresponding squared velocity differences are summed for each bin, averaged, and converted to the velocity structure function as defined in Eqn. (3.19). To reduce statistical noise in the measurement for a single output and obtain a robust statistical characterization of the quasi-stationary turbulent state, many measurements over an extended time period are averaged, as described earlier.

For measuring velocity field power spectra, we adopt a Fourier mesh with dimension $N_{\mathrm{grid}} = N(1+k)$. Velocity values at the regular grid positions are then evaluated based on the polynomial expansions within the corresponding DG cells. We use the MPI-parallel FFTW library to transform the velocity field to Fourier space, separately for each spatial velocity dimension. In each case, the corresponding velocity mode powers are summed up in finely binned k-space shells, so that the average mode power of the 3D velocity field can be computed. We typically use a fine set of 2000 logarithmically spaced bins in k-space. These fine shells can later be adaptively rebinned in a plotting script to form larger bins, as desired. Here we typically rebin such that bins containing just a few modes are accepted for low-k (otherwise the k-bins would get too wide there), while for high-k the bins can be made narrower while still having large mode counts and thus good statistics. Following standard conventions in the field, we present the velocity power spectrum in terms of the quantity $E(k)$ as defined in Eqn. (3.22).

3.6 Turbulence with DG in the supersonic and subsonic regimes

In this section we want to investigate whether our new DG scheme – with artificial viscosity shock capturing and an auxiliary projection of the primitive variables to deal with extrapolations to cell boundaries – is capable of robustly and accurately simulating driven isothermal turbulence well into the supersonic regime. To this end we first consider a set of simulations where we vary the Mach number systematically but keep otherwise all relevant numerical parameters the same. For definiteness we consider $N^3 = 128^3$ cells and DG orders $k = 1$ and $k = 2$, and we compare to matching finite-volume simulations with piece-wise linear reconstruction as conventional base-line results. Our simulation sequence

starts with Mach number $\mathcal{M} = 0.1$, and then we modify the driving strength and the time correlation parameters of the driving routine systematically to create a sequence of simulations in which the Mach number doubles in each step, until we reach $\mathcal{M} \simeq 12.8$. We note that in the subsonic regime, the Mach numbers realized by this procedure accurately match the expected doubling in each step, while for $\mathcal{M}$ significantly above unity, they start to fall slightly short. This is of course expected at some level due to the stronger dissipation in the supersonic case already in the driving regime. This could be corrected for by a correspondingly stronger increase in the driving strength in the supersonic regime, something that we however found not really necessary yet over the limited range in Mach numbers explored here. Note that in each case we simulate for the same number of eddy turn-over times, which however corresponds to different absolute timespans.

In Figure 3.5, we visually illustrate the turbulent velocity field for the $\mathcal{M} = 0.1$, $\mathcal{M} = 0.4$, $\mathcal{M} = 1.6$, and $\mathcal{M} = 6.4$ cases. In each panel, we show the corresponding simulations after the same number of eddy turn-over times after the start of the simulations. While the two subsonic simulations look qualitatively very similar, with the most important difference being the amplitude of the velocity field, the character of the turbulent field clearly starts to differ as we transition into the supersonic regime. This is accompanied by the appearance of strong velocity discontinuities (i.e. shocks), and the velocity field begins to exhibit sharper gradients as well.

This difference also becomes readily apparent in a quantitative way when we consider the velocity power spectra of this sequence of simulations, which we show in Figure 3.6. We here compare the simulations with the finite volume approach (solid lines) to corresponding runs carried out with DG at orders $k = 1$ (dashed) and $k = 2$ (dotted). All Mach numbers from $\mathcal{M} = 0.1$ to $\mathcal{M} = 12.8$ are shown in a single diagram, which is readily possible as they are offset vertically due to their systematically different velocity amplitudes. The common plot makes it evident that the shape of the power spectra systematically changes when transitioning into the supersonic regime. While the inertial range outside the driving range is small due to the limited dynamic range of these simulations, it is still sufficient to show a transition from a $E(k) \propto k^{-5/3}$ Kolmogorov-spectrum in the subsonic case to a $E(k) \propto k^{-2}$ Burger spectrum in the supersonic regimes. This trend is reproduced consistently both by the DG simulations and the finite volume scheme.

Another important and interesting trend is seen for the relative difference between the DG and the FV simulations. At given cell resolution, going from FV to DG significantly extends the dynamic range over which the turbulence can be followed. This is already the case for $k = 1$, and even more so for $k = 2$, with the latter showing also signs of a more pronounced bottleneck effect, which reflects the different and generally lower numerical dissipation in this scheme. The detailed dissipation processes are also the reason why some of the DG runs show slightly enhanced velocity power again on the smallest scales, within cells. As this happens deep in the dissipation regime anyway, it is however not of concern for the practical applicability of the DG method.

Importantly, the improvement in the dynamic range brought about by $k = 1$ and $k = 2$ DG in the subsonic regime nearly disappears in the supersonic regime. Clearly, the accuracy advantages of DG do not play out as effectively any more for supersonic turbulence, if at all.

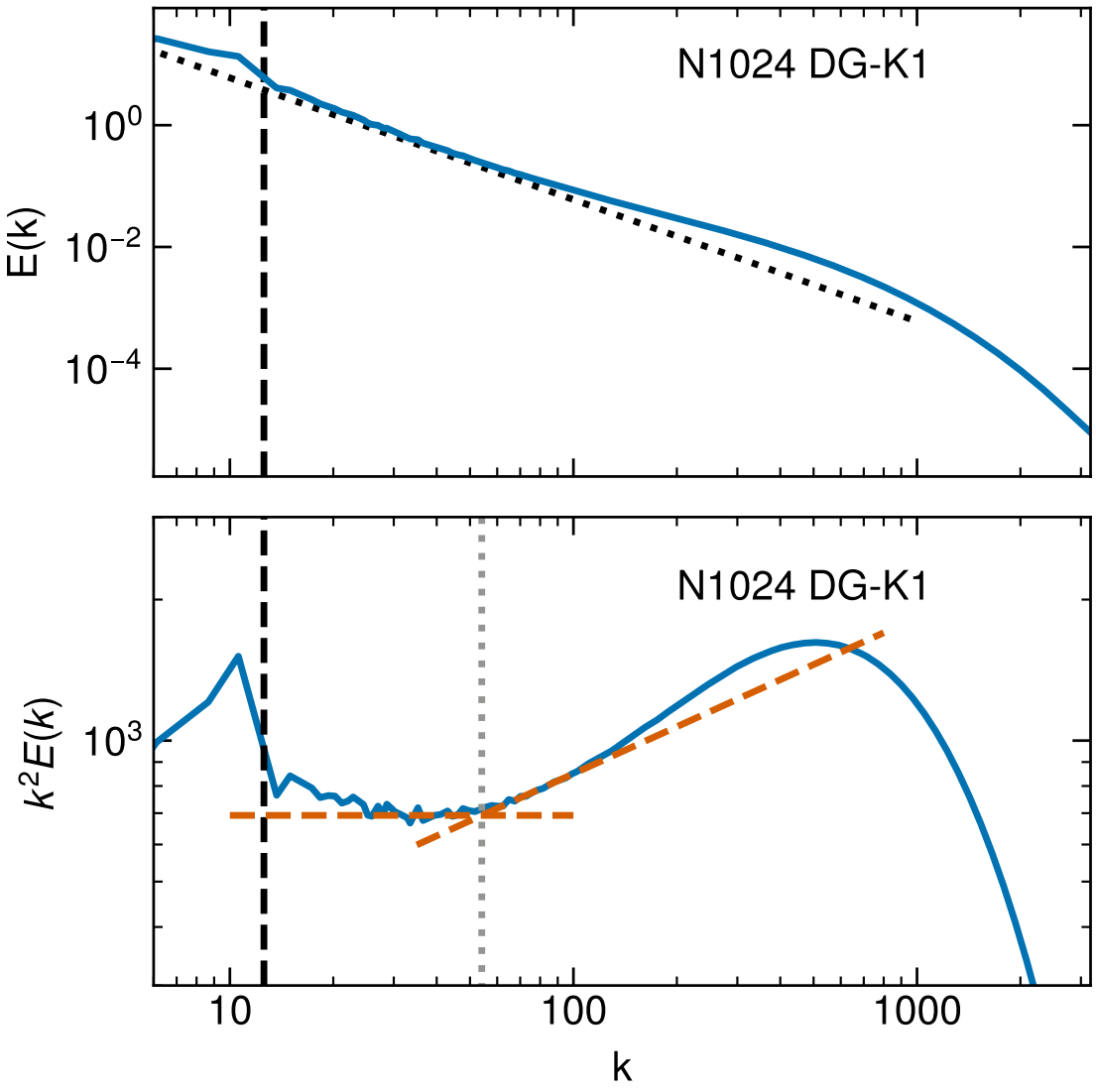


Figure 3.10: Velocity power spectrum of the turbulence simulation shown in Fig. 3.9, i.e. for a DG run with $k = 1$ and 1024^3 cells. The top panel shows $E(k)$ directly, whereas the bottom panel displays the same data again, but this time compensated by a factor k^2 to compress the vertical dynamic range and highlight subtle changes in shape. The dashed vertical line marks the end of our driving range, which can be discerned as a region of elevated power. At slightly larger k than this injection scale, a region with a fully developed supersonic turbulent cascade develops. This is indicated by the dashed horizontal line in the bottom panel, which has the $E(k) \propto k^{-2}$ slope of Burgers turbulence. At still smaller scales, the spectrum becomes flatter again, close to the $E(k) \propto k^{-5/3}$ expected for Kolmogorov turbulence. We have indicated this slope as an inclinded dashed line in the bottom panel, with the dotted line marking the scale where extrapolations of the two power laws intersect. This intersection is reasonably close to the sonic scale inferred from the velocity structure function. We also note that there is a prominent bottleneck effect (as expected) with a small shoulder in the power spectrum before $E(k)$ drops rapidly in the dissipative regime.

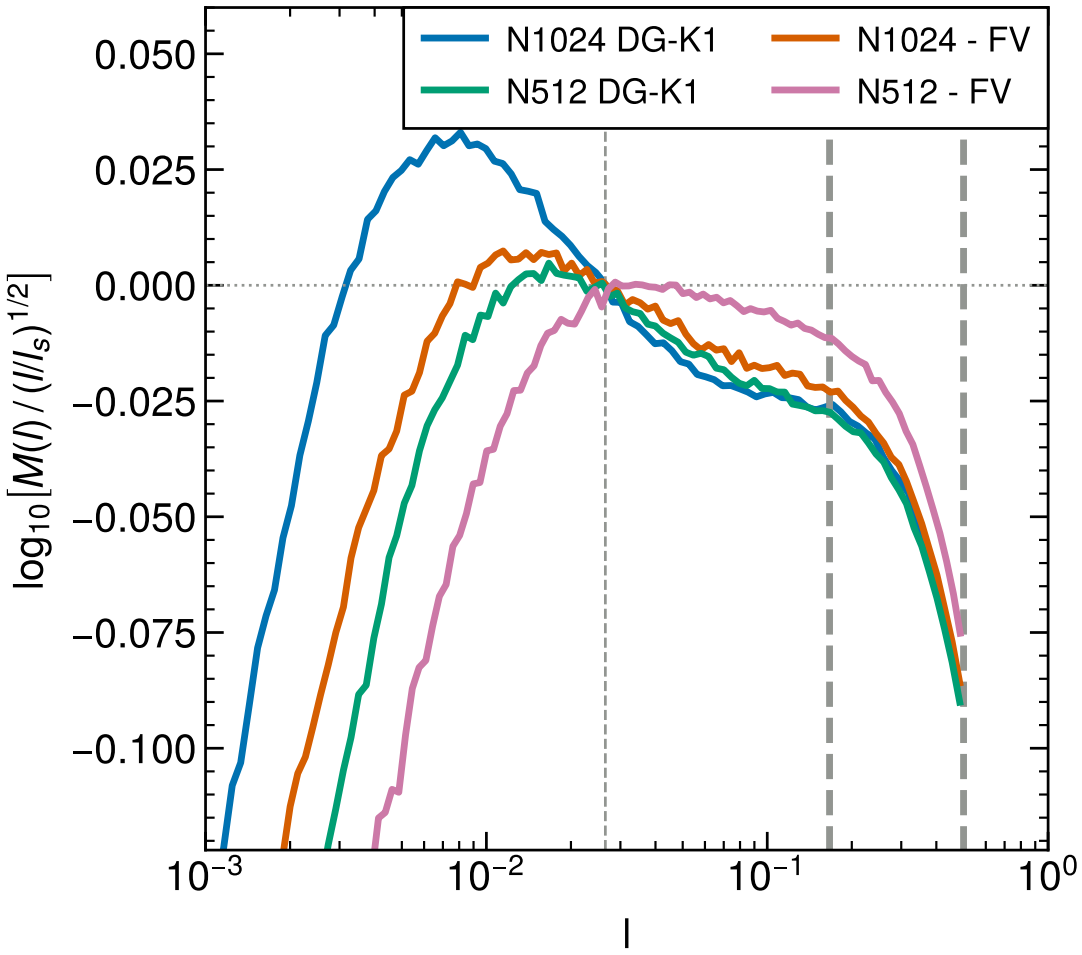


Figure 3.11: Convergence of the velocity structure function (shown in compensated form as in the bottom panel of Fig. 3.9) between calculations that use 512^3 or 1024^3 cells, and either finite volume (FV) or DG with order $k = 1$, respectively, as labelled. The sonic scale used for rescaling the plots is the same for all lines and corresponds to the value measured for the DG run at the 1024^3 resolution. Interestingly, the 512^3 simulation with DG does nearly as well as the 1024^3 run with FV, but both show at most a very feeble hint for a transition between the supersonic and subsonic regimes of turbulence. This is because of the closeness of the dissipation regime at this resolution, which already affects the region around the sonic scale strongly. For the 512^3 run with FV, the dynamic range is clearly insufficient to resolve the region around the sonic point properly. In contrast, the high-resolution DG run is already able to distinguish different slopes of the cascade in the supersonic and supersonic regimes, although it is clear that also this calculation can still be expected to be influenced by resolution effects in the transition region.

Of course, this does not come as a complete surprise in light of our earlier discussion about the challenges involved in capturing true discontinuities with high-order DG methods. In fact, based on this we can already view it as a success that DG can robustly treat supersonic turbulence after all, with an accuracy that is at least comparable or even slightly better than that of a finite volume method. Since any supersonic cascade will eventually transition into the subsonic regime again, this is ultimately encouraging, because it means that in simulations that offer sufficiently high dynamic range, the advantages of DG can still become important again in the subsonic regime. We will explicitly return on this point in Section 3.7.

Besides considering the velocity statistics, it is also interesting to look for systematic differences in the density PDF, and in the velocity structure function. In Figure 3.8, we show density probability distribution functions obtained by considering the density fields in different high-resolution 2D slices through the simulation boxes at different times, and then averaging the results. We compute the histograms in terms of logarithmic density, i.e. a strictly parabolic shape of our measured density PDF would therefore correspond to a lognormal distribution.

3.7 Simulating the super- to subsonic transition

As we have just seen, DG simulations offer comparatively little accuracy gains in the supersonic regime of turbulence, but they can yield considerably more accurate results in the subsonic regime by moving the numerical dissipation scale in smooth flows to smaller scales. This raises the question whether DG can still be advantageous in a simulation of supersonic turbulence when it has enough dynamic range to transition into the subsonic regime. Then we may expect that perhaps both, the transition around the so-called sonic point, as well as the subsonic part of the turbulent cascade, are represented better by the DG approach compared with traditional FV methods.

To examine this question we consider two reasonably high resolution simulations of supersonic $\mathcal{M} = 3.2$ turbulence, carried out with 1024^3 cells and the $k = 1$ DG order, and with a piece-wise linear finite volume approach, for comparison. This resolution is a far cry from the large 10048^3 simulation recently used by Federrath et al. (2021) to resolve the location of the sonic point. Since we employ a somewhat smaller Mach number to begin with, we should in principle have, however, a chance to see something if the numerical technique is less dissipative than standard finite volume approaches, given that according to Eqn. (3.20) the sonic point should be for $\mathcal{M} = 3.2$ only about a factor of 10 away from the injection scale.

In Figure 3.9 we show the velocity structure function of the $k = 1$ DG simulation, with the top panel directly showing the measured Mach number as a function of scale as defined in Eqn. (3.19), while the bottom panel shows the structure function in a compensated way by dividing it with a $l^{1/2}$ dependence, which is the expected scaling in the regime of a self-similar supersonic turbulence cascade. Thanks to a large compression of the vertical scale, this compensated version brings out a number of important details in the shape of

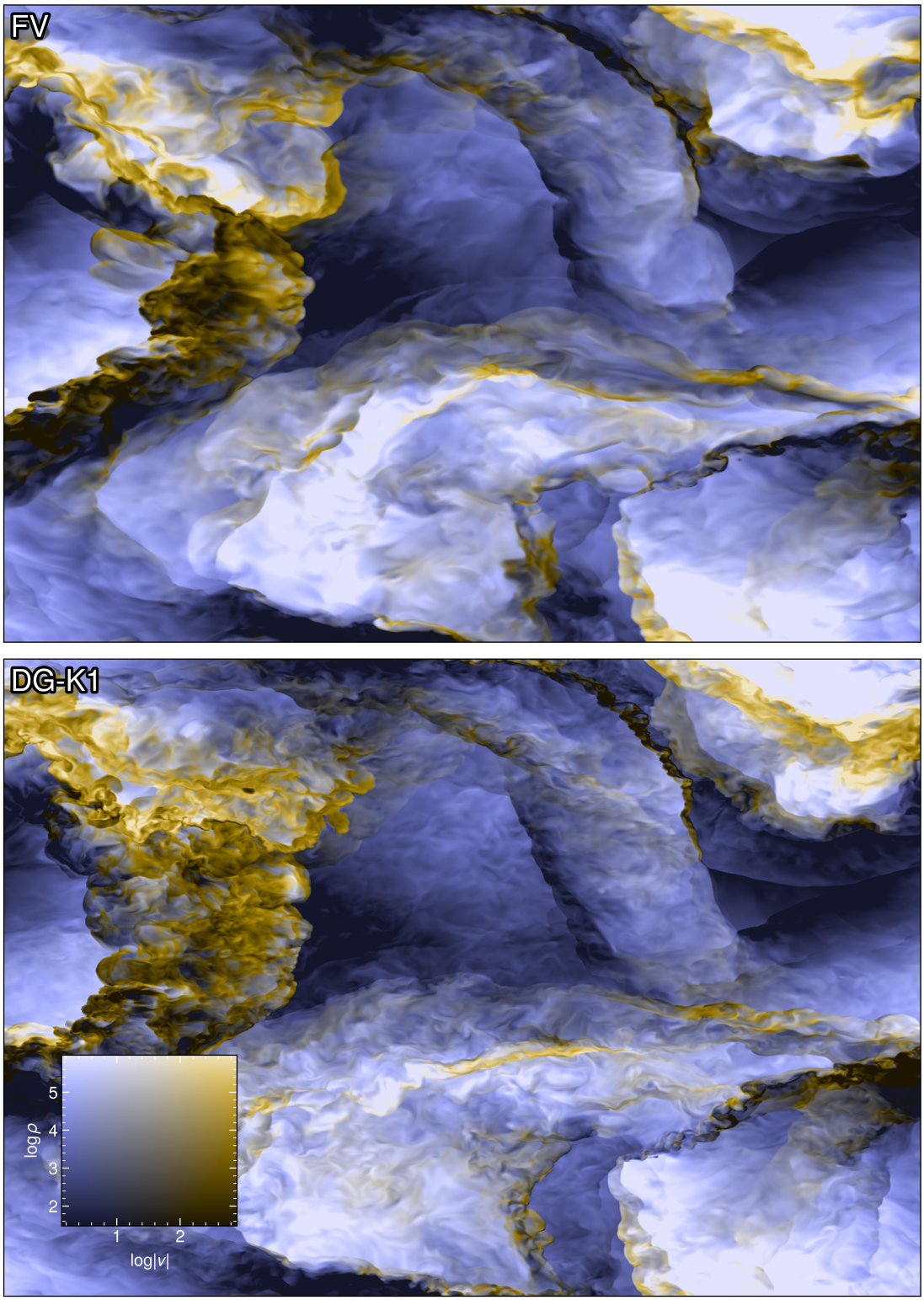


Figure 3.12: Visualization of the turbulence for FV (top panel) and DG (bottom panel) simulations at Mach number $\mathcal{M} = 3.2$. We use a two-dimensional color map, where the logarithm of density is mapped to brightness while the logarithm of the gas velocity is mapped to color hue, as indicated. The fields are shown at the same time, using 1024^3 cells. Superficially the images look quite similar, but closer inspection reveals a richer and more pronounced small-scale structure in the DG simulation.

the structure function. In particular, on large scales, we see a settling region that ranges between the driving scale at $L/2$ down to about $\sim L/12$. Only at still smaller scales, the supersonic cascade has fully developed. Interestingly, this then follows quite accurately a $v(l) \propto l^{0.5}$ slope until there is a quite sudden change in slope to $v(l) \propto l^{0.4}$, which is the expected slope in the region of the subsonic cascade. We thus think that this clear kink in the structure function identifies the sonic point – the supersonic to subsonic transition. We have also identified a scale l_s in the structure function where $v(l_s) = c_s$. This scale lies close to the place where we detect the change in slope in the structure function, but it is not identical to it in our results. Towards still smaller scales, the compensated structure function eventually reaches a maximum after following the $v(l) \propto l^{0.4}$ inertial range for a while, and then declines rapidly due to a steeping of the slope.

A similar behaviour can be inferred from the velocity power spectrum, which we show in Figure 3.10 for the same simulation. Again, we show in the top panel the plain power spectrum, while in the bottom panel we plot it in a compensated form where the power spectum has been multiplied with a k^2 factor. The latter is the expected slope for Burgers turbulence in a supersonic cascade. Indeed, our results for the velocity power spectrum accurately follow this slope over a small dynamic range beginning slightly out of the driving region, once the turbulence had a chance to fully develop in a self-similar fashion. Eventually the power spectrum flattens to the $k^{-5/3}$ slope of Kolmogorov turbulence, which manifests as a rise in the componsated version of the plot. There is thus a clear kink in the spectrum which we can again interpret as a manifestation of the sonic point. The subsonic intertial range is however not very large in our simulation due to its limited numerical resolution, and thus the bottleneck effect from the onset of the dissipation range influences a good part of it. Towards still smaller scales, the power spectrum eventually transitions fully into the dissipation due to the numerical viscosity of the discretization method.

It is interesting to compare the shape of the structure function between DG and a finite volume scheme, something we present in Figure 3.11. There we show both the $k = 1$ DG simulation as well as the corresponding FV simulation for 1024^3 cells in the same plot. In addition, we also include two further simulations computed instead with a lower 512^3 resolution. Interestingly, while the FV simulation with 1024^3 cells also shows a kink in the structure function where the sonic point is expected, the supersonic slope is flatter than expected, and the feature appears somewhat washed out compared to the $k = 1$ DG result. In addition, the transition to the dissipation regime appears considerably earlier. It is thus evident that the DG approach represents the sonic point much more accurately than the FV scheme, and it is also much less diffusive on small scales, allowing it to capture a larger part of the subsonic turbulent cascade.

Remarkably, the 512^3 result for $k = 1$ DG is nearly as good as the 1024^3 FV result, and it likewise shows evidence for the sonic point, albeit not as cleanly as the 1024^3 DG result. In contrast, the 512^3 finite volume result fails to yield any trace of the sonic point. Apparently it is simply already too diffusive. Our comparison thus confirms that DG can offer advantages even in simulations of supersonic turbulence, provided the resolution is high enough that the transiton to subsonic turbulence can be resolved and the associated subsonic cascade is of interest for the analysis of the simulation.

Finally, it is also instructive to visually verify the better small-scale resolving power of the DG approach in maps of the turbulence field. In Figure 3.12, we compare visualizations of the turbulent density field in slices through the box of the 1024^3 DG and FV runs. We show the fields at the same time, with the logarithm of the density encoded as pixel brightness while the logarithm of the gas velocity is mapped to color hue. Superficially the images look quite similar, but closer inspection reveals a richer and more pronounced small scale structure in the DG simulation. This confirms our earlier quantitative findings for the velocity structure function and the velocity power spectrum, and reflects the better resolving power of the DG approach for the subsonic part of the turbulent cascade. There is thus no question that DG simulations of supersonic turbulence deliver higher accuracy than FV simulations for an equal number of cells. But the relative computational cost of the methods is also an important aspect that needs to be considered. We will turn to this question in the next section.

3.8 Discussion on computational cost

Let us now discuss the accuracy and computational cost of DG. We have shown that DG is applicable to supersonic turbulence, and that it is also fairly accurate in this regime, but apparently not significantly more so than a finite volume technique. But is it then worthwhile to go to high order given the computational cost of DG? Unlike for smooth problems, no exponential convergence can be expected in the supersonic regime, instead the width of shocks is expected to decline only as the effective spatial resolution of the scheme. Presumably this means that low-order methods are computationally more efficient if shocks play a prominent role for the evolution of the simulated system. To make this aspect more specific, we discuss in the following the expected computational cost of the DG method as a function of the employed order. This should shed some light on where potential sweetspots lie for different types of simulations.

For definiteness, we consider simulations carried out with N^3 cells at DG order k, using ideal hydrodynamics.This means we use $b_k = \frac{1}{6}(k+1)(k+2)(k+3)$ Legendre coefficients to describe each of the $f = 5$ five conserved fields (mass density, three spatial momentum densities, and energy density). The total number of degrees of freedom (DOF) is thus $D_{\rm DG} = f b_k N^3$, which is also the storage footprint of the scheme and equals the total length of the vector of weights. Note that the information content that can be captured by any simulation is arguably best given by this quantity, since the value of each degree of freedom is in principle independent of all others.

For one evaluation of the time derivative of the weights, we need to carry out an internal integration over the fluxes, as well as a flux computation over the surfaces of the cells, with the latter involving a Riemann solver. To carry out the volume integrals over the DG cells we need $v_k = (k+1)^3$ internal Gauss points each, while integrating the fluxes over each face of a cell requires $s_k = (k+1)^2$ Gauss points.

Let us first consider the volume integrations and try to estimate the number of floating point operations required for this. We shall treat them all as equivalent in cost (disregarding

that divisions are more expensive), for simplicity, and will be content with an approximate count. Optimized code implementations may perhaps reach a slightly smaller operation count, for example through clever reuse of partial results, but should not be able to change the overall scaling. At each internal Gauss point, we first of all need an evolution of the field expansion to get the conserved variables of the fluid at the corresponding coordinate, which requires of order $2\,b_k\,f$ floating point operations. Conversion of the conserved states to the full hydrodynamical flux then requires of order $c_F \simeq 15$ operations at each Gauss point. This flux is then contracted with spatial derivatives of the Legengre basis functions to give a contribution to the time derivative of the weights, the cost of this is about $6\,fb_k$ per Gauss point. Multiplying each of these partial contributions with a Gaussian quadrature weight and then adding them up to yield the overall contribution to the time derivative of the weights gives another $2\,fb_k$ operations per Gauss point. Summing this up for all internal Gauss points, we thus need about $v_k(10\,b_k\,f + c_F)$ operations for the internal volume integration.

For the surface fluxes, we need $6\,s_k$ Gauss points in total per cell. The evolution of the field expansions costs again $2\,b_k\,f$ operations per Gauss point. These fluid states in conserved variablles are then converted to primitive variables, and are fed together with the state from the neighbouring cell to a Riemann solver, yielding the flux vector. Computing this very conservatively costs at least $\sim 2\,c_F$ for a (very) approximate Riemann solver. But since this cost has to be spent only once per interface for the two adjacent cells and Gauss points, we can can approximate the cost of this part of the calculation again with $\simeq c_F$ operations per Gauss point. Contracting the fluxes with a surface normal does not require a full scalar product in this case since we know that exactly one component of the normal vector is unity, but we still need to multiply with a Legendre function value, a Gaussian quadrature weight, and add things up to the total partial result for the weight change, requiring about $3\,fb_k$ per surface Gauss point, yielding a total cost of $6\,s_k(5\,b_k\,f + c_F)$ per cell.

The cost of the computation of the time derivative of the DOFs (i.e. the vector of weights, $\mathrm{d}\boldsymbol{w}/\mathrm{d}t$), is thus approximately

$$\begin{aligned} C_{\dot{\boldsymbol{w}}} &= N^3[v_k(10\,b_k\,f + c_F) + 6s_k(5\,b_k\,f + c_F)] \\ &= \frac{5}{3}N^3(k+1)^2(5k^4 + 50k^3 + 175k^2 + 259k + 183), \end{aligned} \tag{3.31}$$

which reveals a steep $C_{\dot{\boldsymbol{w}}}/N^3 \propto k^6$ increase of the computational cost per cell when going to high-order. Furthermore, note that the time integration itself requires several evaluations of this quantity if one aims for a consistently high order of the time integration. For the stabillity-preserving Runge-Kutta schemes we use, we need, for example, 2 stages for a second-order accurate scheme ($k = 1$), and 3-stages for a third order scheme. For a fourth order scheme, one would ideally like to use a 4-stage Runge-Kutta scheme, which however does not exist in a purely forward form, so we need to use a 5-stage scheme instead. At still higher order, similar compromises may need to be made but we ignore some of these details here by estimating that we need $(k + 1)$ evaluations of the time derivative of the DOFs to complete one timestep. Staging intermediate results according to the Butcher

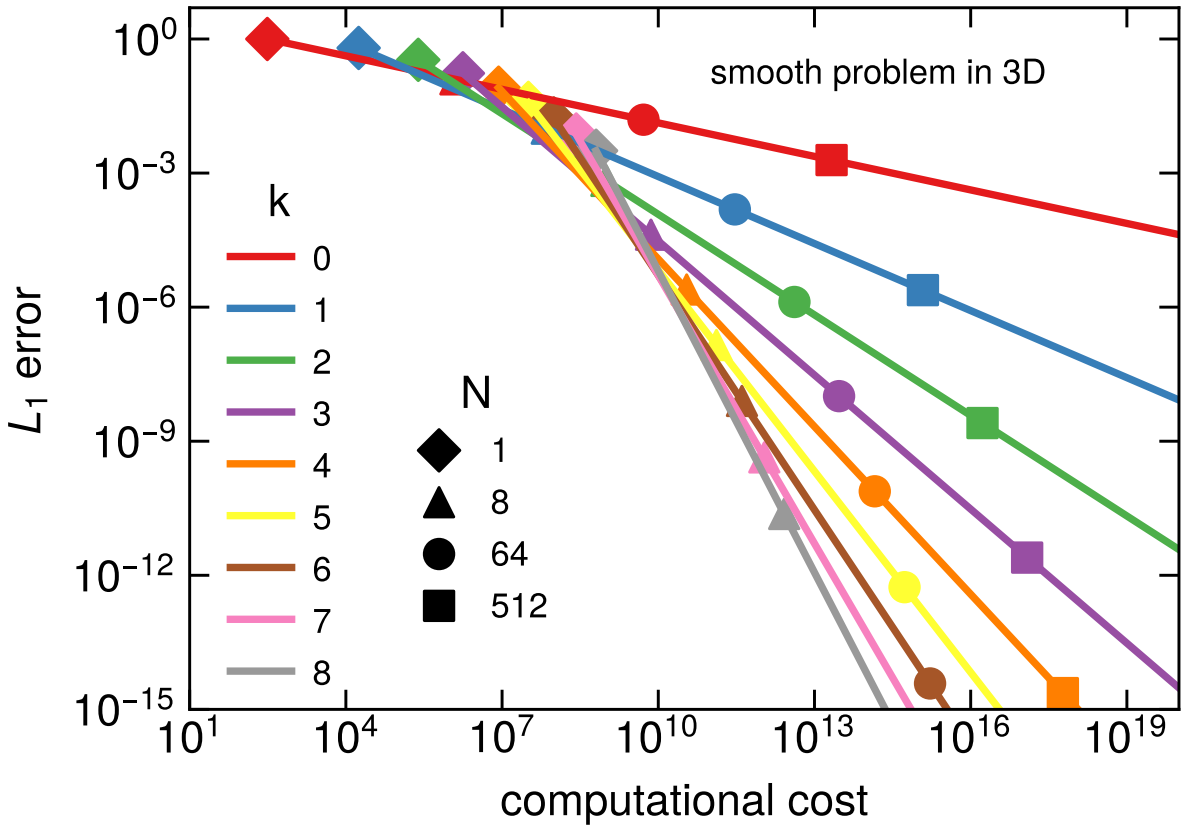


Figure 3.13: Expected scaling of the total numerical error as a function of the invested computational effort for a *smooth* hydrodynamical problem simulated with DG at different order k (coloured solid lines) in a three-dimensional box with N^3 cells, based on Eqn. (3.33). A few illustrative problem sizes are marked with symbols, as labelled. High-order methods incur a substantially higher computational cost for a given number of cells, but they are also able to approximate a smooth solution more accurately. The error drops progressively faster as a function of resolution for higher order methods, in fact so fast that they become the method of choice – in the sense of requiring the lowest computational cost – for large enough problem sizes and sufficiently small target error.

tableau of the Runge-Kutta scheme, and adding up the time derivatives with the Butcher weights to yield the final result at the end of the step, needs about $(k+1)^2 \times D_{\rm DG}$ floating point operations, so that we end up with a cost per timestep of about

$$C_{\rm step} = (k+1)C_{\boldsymbol{\dot{w}}} + (k+1)^2 f b_k N^3. \tag{3.32}$$

Finally, the permissable Courant timestep also becomes smaller at higher order, as $\Delta t \propto h/(k+1)$, with $h \propto 1/N$. The total number of timesteps needed to evolve a system over some finite time interval T therefore scales in proportion to $(1+k)N$. The resulting total

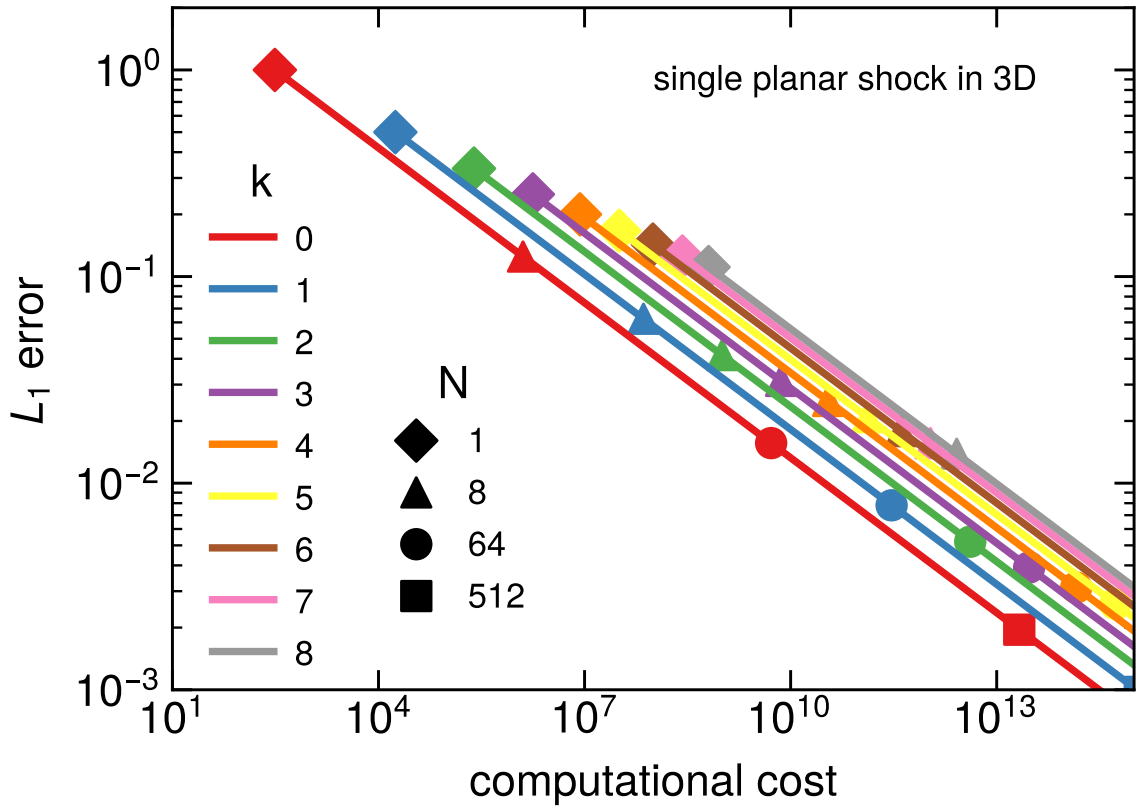


Figure 3.14: Expected scaling of the computational cost and total numerical error for a *planar shock* problem simulated with DG at different order k (coloured solid lines) in a three-dimensional box with N^3 cells that otherwise exhibits a homogeneous fluid state everywhere outside the shock. In this situation, only the numerically broadened shock itself is contributing to the error budget, which thus declines only with the linear spatial resolution as $L_1 \propto N^{-1}(k+1)^{-1}$. As a result, higher-order methods do not provide a scaling advantage of their numerical error, i.e. the relative accuracy of low and high order methods stays invariant as a function of resolution, and their higher baseline computational cost per cell (compare the illustrative problem sizes marked with symbols) is not worthwhile. Note, however, that problems of practical interest do not consist of shocks only, rather they also have non-trivial smooth regions in between shocks, where the considerations of Fig. 3.13 apply. In general, which order is computationally most cost efficient is therefore problem-dependent.

computational cost thus scales as

$$\begin{aligned} C_{\mathrm{tot}} &= (1+k)N\left[(k+1)C_{\dot{w}} + (k+1)^2 f b_k N^3\right] \\ &= \frac{5}{6}N^4(1+k)^4(10k^4+100k^3+315k^2+523k+372). \end{aligned} \tag{3.33}$$

To leading order, the computational cost therefore scales as $C_{\mathrm{tot}} \propto N^4 k^8$ in 3D applications of DG. This makes going to very high-order impractical, and moderate order is only

worthwhile if this indeed delivers a correspondingly high accuracy.

To get a better idea of the critical trade off between computational cost and reached accuracy, we consider a fiducial error norm $L_1 = a_k N^{-(k+1)}$ as a function of computational cost for different orders k of the DG-scheme. A decline of the error as $L_1 \propto N^{-(k+1)}$ is expected for smooth problems, and reflects the decrease of the error norm with spatial resolution at fixed order, while for fixed resolution, the error declines exponentially with order $p = k + 1$. Empirically, the coefficient a_k in front typically shows only a weak dependence on order. Instead of simply taking it to be constant, we here set a_0 =1, and for order $k > 1$ we assume that L_1 for $N = 1$ and order k is equal to the error norm for $k-1$ at the same number of degrees of freedom. In other words, we assume that for $N \simeq 1$ trading in spatial degrees of freedom for expansion order coefficients keeps the accuracy roughly fixed, which is reasonable (in fact, for smooth problems, we typically expect that the accuracy improves in this case for the high-order scheme, see Fig. 15 in Cernetic et al. (2023)).

In Figure 3.13 we now show the expected L_1 error norm based on this prescription as a function of computational cost, for different order. Note that increasing the cost implicitly means considering larger problem sizes N^3. Despite the steep increase of the cost with order, for large problem sizes the high-order schemes tend to be advantageous, i.e. they deliver higher accuracy at a given computational cost, or conversely, they can reach a given computational L_1 error for lower computational cost. For intermendiate problem sizes, or low target accuracies, the situation is less clear cut, and here intermediate or lower order can be computationally advantagous. This is also confirmed by experimental findings, as reported for example in Schaal and Bauer. We thus conlude again that for smooth problems, where $L_1 \propto N^{-(k+1)}$ holds, high-order DG methods tend to be cost effective. Importantly, they also have other attractive properties, such as much lower advection errors and excellent angular momentum conservation.

However, an important flipside to this discussion, which is relevant for the present chapter, is that once the error norm declines more weakly with spatial resolution due to the presence of true physical discontinuities such as shocks, this cost advantage of DG methods is defeated. We can make this more explicit by considering a fiducial simulation with a single planar shock wave in a 3D box with an otherwise homogenous fluid state. This could be realized, for example, by inflow and outflow boundary conditions on opposite sides of the box, and by matching the inflow and outflow states with the Rankine-Hugoinot to the jump conditions of a strong shock. In such a situation we expect the error norm to decline as $L_1 = b_k N^{-1}(k+1)^{-1}$, with an approximately constant prefactor (in the following we set $b_k = 1$). The shock does become narrower with higher spatial resolution or high order, but it does so only linearly. This profoundly alters the relation between invested computational cost and expected accuracy, as illustrated in Figure 3.14 for this situation. It is now always computationally favorable to reach a desired accuracy (here equivalent to shock width) by investing into higher N rather than into higher order k.

This does not automatically imply that higher-order DG methods are worthless for problems involving shocks, because in non-trivial flow problems there is typically a lot of interesting structure in regions outside of shocks, and these are rendered less accuractely

by low order methods. Also, recall that *most of the volume* will always be outside of shocks, given that shocks are in principle arbitrarily thin transition layers, so the volume fraction occupied by them is very small even when taking numerical broadening into account. So since one will typically be still interested in reaching high accuracy in smooth parts of the flow, a high order DG method can often be the most efficient choice. One simply then has to pay the price that some of the computational effort is consumed for an inefficient representation of shocks. In principle, this deficiency could be overcome by an approach where one applies h-refinement (i.e. increasing the grid resolution while lowering order $p = k+1$) in places with shocks, while in smooth regions one should rather use p-refinement (increasing order p while reducing mesh resolution h). Whether this is readily feasible in practice is however problem dependent.

3.9 Conclusions

One of the challenges of using DG methods for simulating supersonic turbulence is the appearance of Gibbs-like phenomena, especially at higher orders. These phenomena can cause spurious oscillations and numerical instabilities in the presence of shocks. The artificial viscosity method proposed by Cernetic et al. (2023) as well as the simpler artificial viscosity parametrisation proposed in the present chapter can mitigate this problem, but they alone are not sufficient to stably evolve highly supersonic flows with DG at high order.

In such flows, multiple shocks can interact, and very steep gradients of fluid variables develop inside cells. Especially for the computation of fluid states at cell surfaces, which in essence can be viewed as a polynomial extrapolation from the Gauss points inside the cells, this can frequently lead to problems. Because our code evolves the conserved variables $\boldsymbol{u} = (\rho, \rho\boldsymbol{v}, e)$, to obtain the velocity we need to divide the polynomial expansion of momentum density by that for the density, but this can lead to unphysical values at cell surfaces in the presence of very strong field variations, particularly when the density becomes very low. Similarly for the pressure, where the kinetic energy density needs to be subtracted from the total energy density, so that the pressure becomes a complicated rational function which is not necessarily as well behaved as the polynomial basis functions. This destroys the robustness of DG for highly supersonic flow.

To solve this issue we have introduced a novel projection approach of the primitive variables. They are first evaluated based on the conserved variables at the internal Gauss points, and the resulting values are used to define a polynomial expansion of the primitive variables over the cells. This regularizes the extrapolation to cell surfaces and avoids unphysical values there. Together with our simplified artificiall viscosity method this makes it possible to successfully simulate non-trivial test problems involving multiple interacting shocks as well as driven supersonic turbulence.

Our main findings can be summarized as follows:

- We have introduced a simple but effective approach to capture shocks in high-order Discontinuous Galerkin discretisations of the Euler equations of fluid dynamics. It

relies on the familiar von Neumann-Richtmyer viscosity applied at the internal Gauss points of each cell. This approach works well at all orders, is robust, has no storage overhead, and allows shock-capturing with sub-cell resolution.

- Simulations with very strong shocks become much more robust and accurate if the primitive variables at cell interfaces are not simply derived from the extrapolated conserved variables there, but rather from a polynomial expansion of the primitives themselves, which is uniquely obtained from the values they assume at the internal Gauss points.

- Using these two methodological advances, we have succeeded to stably run supersonic turbulence simulations of isothermal gas at Mach number 12.8 with high-order DG. This has previously been met with severe stability problems, and thus this represents in its own right an important advance for the practical applicability of DG approaches in astrophysics.

- While DG offers significant accuracy gains over finite volume schemes in the subsonic regime (which manifests itself in an extended intertial range and reduced numerical dissipation), this advantage is progressively diminished and nearly lost in the supersonic regime. However, once the sonic point is approached, the higher accuracy of DG starts to be noticeable again, and for simulations that have enough dynamic range to also resolve parts of the subsonic cascade, DG begins to shine again.

- We have given a simple analysis of the numerical cost of our DG implementation at order k, based on estimating the required number of floating point operations to carry out a 3D simulation with N^3 cells over a fixed timespan T. This cost increases rapidly with order and resolution, as $\propto N^4 k^8$. For smooth problems, the error declines so rapidly for high k that it is in principle still worthwhile to employ high order. The physical discontinuity of a shock defeats this scaling, however, and here low-order is more cost efficient.

Our findings in this chapter make DG fully applicable to astrophysical problems involving strong shocks and contact discontinuities. Particularly the very low advection errors and accurate angular momentum conservation of DG compared with finite volume schemes should make this method interesting for many applications, for example for problems involving multiphase has. However, as soon as many shocks are present, it appears unlikely that high-order DG methods with $k > 2$ are computational cost effective. Rather, a sweetspot can be expected for $k = 1$ or $k = 2$, and it is probably worthwhile to further optimize production codes for this regime.

Chapter 4

Discussion and outlook

In this chapter, we summarize the results of the thesis and give an outlook for possible studies in the future. For a general overview and the background of this thesis, we refer to Section 1.6.

4.1 Summary of this thesis

The goal of this thesis was to assess how high order numerical methods on GPUs perform in simulations of sub- and supersonic driven turbulence. We also insights insights on how to write massively parallel scientific codes that are able to harness all CPUs and GPUs of modern exa-scale machines. We concentrated on studying treatment of shocks and making the fragile Discontinous Galerkin method robust.

In Chapter 2 I demonstrate that the Discontinuous Galerkin method can be used to simulate subsonic turbulence with a high accuracy, even exhibiting exponential convergence with higher order integration.

In Chapter 3, I performed a parameter study of supersonic turbulence with the Discontinuous Galerkin method and found that the method is robust against shocks and can be used to simulate supersonic turbulence with a high accuracy despite the presence of a network of strong shocks. Up to a Mach number of $\mathcal{M} = 12.8$ the first order DG method offers significant benefits over the classic finite volume method with linear reconstruction.

4.2 Future extension of this thesis

The developed high-order hydrodynamics code that runs natively on GPUs has a wide range of possible future applications both because it can utilize the largest GPU accelerated supercomputers while at the same time using modern hydrodynamic methods capable of resolving smaller scales at the same number of degrees of freedom.

The benefits of exponential convergence with spatial order can be readily exploited in studying the source of heating in galaxy clusters, which exhibit very short cooling times $t_{\rm cool} < 1$ Gyr but abnormally low corresponding star formation (Rawle et al., 2012).

Additionally, the hot, low density, weakly magnetised plasma that makes up atmospheres of galaxy clusters leads to large electron mean free paths, making thermal conduction an important factor. By using the existing Navier-Stokes solver with shear viscosity and thermal diffusivity enhanced by the DG MHD implementation from Guillet et al. (2019) I will run high resolution simulations of the ICM to fully resolve the dissipation scale and constrain sources of heating in galaxy clusters.

The stability of this DG implementation for supersonic flows opens the possibility to study a wide variety of astrophysical systems. One problem where the capability of DG to better resolve the smooth parts of the simulation despite the presence of strong shocks would be an analysis of star formation in supersonic and multiphase ISM (Veilleux et al., 2020). To this end I will implement a cooling prescription proposed by (Townsend, 2009), set up a global high resolution (∼pc) ISM simulation and explore the injection of turbulence in the ISM by stellar feedback, winds and supernovae. A similar setup could be used in conjunction with the AGN model of Costa et al. (2020) to study how the ISM turbulence responds to AGN outflows.

Staying with AGNs, another possible application of the code is to study cosmic ray production in AGN jets and a detailed investigation of the jet's wake. Recently Mbarek & Caprioli (2019) showed the potential of relativistic jets of powerful AGNs to produce ultra-high-energy cosmic rays (UHECRs) up to 1020 eV through reacceleration in the jets – the espresso mechanism. I would simulate a single AGN jet in a z-extended box and investigate its influence on cosmic ray acceleration at a resolution a few times higher than the current state-of-art. Such a simulation would at the same time allow to study the jet's wake. It is thought the wake can to pull down ejected gas, forcing it to be reaccreted and thus further fueling the AGN.

From the algorithmic point of view one extension of this thesis would be to implement a p-refinement of the DG elements. This way the order of computational cells undergoing shocks could be dynamically decreased, while the polynomial order of cells in the smooth parts of the computational domain could be increased. This way a much higher dynamic range could be obtained at the same amount of total degrees of freedom. Such an implementation would be compared to the h-refinement based approach by Schaal et al. (2015).

Another natural algorithmic extension of this thesis would be to implement purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics proposed by Schlottke-Lakemper et al. (2021) and compare it to other established gravity solvers like TreePM (Xu, 1995; Bagla, 2002; Springel et al., 2005), fast multipole method (FMM) introduced by Greengard & Rokhlin (1987) and a hybrid FMM-PM approach by Springel et al. (2021).

Bibliography

Agertz, O., Moore, B., Stadel, J., Potter, D., Miniati, F., Read, J., Mayer, L., Gawryszczak, A., Kravtsov, A., Nordlund, Å., Pearce, F., Quilis, V., Rudd, D., Springel, V., Stone, J., Tasker, E., Teyssier, R., Wadsley, J., Walder, R. (2007), *Fundamental differences between SPH and grid methods*, MNRAS, 380(3), 963

Anderson, C. S., Heald, G. H., Eilek, J. A., Lenc, E., Gaensler, B. M., Rudnick, L., Van Eck, C. L., O'Sullivan, S. P., Stil, J. M., Chippendale, A., Riseley, C. J., Carretti, E., West, J., Farnes, J., Harvey-Smith, L., McClure-Griffiths, N. M., Bock, D. C. J., Bunton, J. D., Koribalski, B., Tremblay, C. D., Voronkov, M. A., Warhurst, K. (2021), *Early Science from POSSUM: Shocks, turbulence, and a massive new reservoir of ionised gas in the Fornax cluster*, Publ. Astron. Soc. Australia, 38, e020

Andersson, N., Comer, G. L. (2021), *Relativistic fluid dynamics: physics for many different scales*, Living Reviews in Relativity, 24(1), 3

Armitage, P. J. (1998), *Turbulence and Angular Momentum Transport in a Global Accretion Disk Simulation*, ApJ, 501(2), L189

Bagla, J. S. (2002), *TreePM: A Code for Cosmological N-Body Simulations*, Journal of Astrophysics and Astronomy, 23(3-4), 185

Balbus, S. A. (2000), *Stability, Instability, and "Backward" Transport in Stratified Fluids*, ApJ, 534(1), 420

Balbus, S. A. (2001), *Convective and Rotational Stability of a Dilute Plasma*, ApJ, 562(2), 909

Balbus, S. A., Hawley, J. F. (1998), *Instability, turbulence, and enhanced transport in accretion disks*, Reviews of Modern Physics, 70(1), 1

Ballesteros-Paredes, J., André, P., Hennebelle, P., Klessen, R. S., Kruijssen, J. M. D., Chevance, M., Nakamura, F., Adamo, A., Vázquez-Semadeni, E. (2020), *From Diffuse Gas to Dense Molecular Cloud Cores*, Space Sci. Rev., 216(5), 76

Ballesteros-Paredes, J., Mac Low, M.-M. (2002), *Physical versus Observational Properties of Clouds in Turbulent Molecular Cloud Models*, The Astrophysical Journal, 570, 734, ADS Bibcode: 2002ApJ...570..734B

Bassi, F., Rebay, S. (1997), *A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier Stokes Equations*, Journal of Computational Physics, 131(2), 267

Bauer, A., Schaal, K., Springel, V., Chandrashekar, P., Pakmor, R., Klingenberg, C. (2016), *Simulating Turbulence Using the Astrophysical Discontinuous Galerkin Code TENET*, in *Software for Exascale Computing - SPPEXA 2013-2015*, edited by H. Bungartz, P. Neumann, W. E. Nagel, volume 113 of *Lecture Notes in Computational Science and Engineering*, 381–402, Springer

Bauer, A., Springel, V. (2012), *Subsonic turbulence in smoothed particle hydrodynamics and moving-mesh simulations*, MNRAS, 423(3), 2558

Berlok, T., Pfrommer, C. (2019), *On the Kelvin-Helmholtz instability with smooth initial conditions - linear theory and simulations*, MNRAS, 485(1), 908

Bhadari, N. K., Dewangan, L. K., Pirogov, L. E., Pazukhin, A. G., Zinchenko, I. I., Maity, A. K., Sharma, S. (2023), *Fragmentation and dynamics of dense gas structures in the proximity of massive young stellar object W42-MME*, MNRAS, 526(3), 4402

Bluck, A. F. L., Maiolino, R., Sánchez, S. F., Ellison, S. L., Thorp, M. D., Piotrowska, J. M., Teimoorinia, H., Bundy, K. A. (2019), *Are galactic star formation and quenching governed by local, global, or environmental phenomena?*, Monthly Notices of the Royal Astronomical Society, 492(1), 96

Borrow, J., Schaller, M., Bower, R. G., Schaye, J. (2022), *SPHENIX: smoothed particle hydrodynamics for the next generation of galaxy formation simulations*, MNRAS, 511(2), 2367

Brandenburg, A., Åke Nordlund (2011), *Astrophysical turbulence modeling*, Reports on Progress in Physics, 74(4), 046901

Burgers, J. (1948), *A Mathematical Model Illustrating the Theory of Turbulence*, Advances in Applied Mechanics, 1, 171

Burns, K. J., Vasil, G. M., Oishi, J. S., Lecoanet, D., Brown, B. P. (2020), *Dedalus: A flexible framework for numerical simulations with spectral methods*, Physical Review Research, 2(2), 023068

Cernetic, M., Springel, V., Guillet, T., Pakmor, R. (2023), *High-order discontinuous Galerkin hydrodynamics with sub-cell shock capturing on GPUs*, MNRAS, 522(1), 982

Cha, S.-H., Inutsuka, S.-I., Nayakshin, S. (2010), *Kelvin-Helmholtz instabilities with Godunov smoothed particle hydrodynamics*, MNRAS, 403(3), 1165

Chabrier, G. (2003), *Galactic Stellar and Substellar Initial Mass Function*, PASP, 115(809), 763

Cockburn, B., Hou, S., Shu, C.-W. (1990), *The Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws. IV: The Multidimensional Case*, Mathematics of Computation, 54(190), 545

Cockburn, B., Lin, S.-Y., Shu, C.-W. (1989), *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems*, Journal of Computational Physics, 84(1), 90

Cockburn, B., Shu, C.-W. (1988), *The Runge-Kutta local projection P1-discontinuous-Galerkin finite element method for scalar conservation laws*, in *1st National Fluid Dynamics Conference*, 636

Cockburn, B., Shu, C.-W. (1989), *TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws II: General Framework*, Mathematics of Computation, 52(186), 411

Cockburn, B., Shu, C.-W. (1998), *The Runge-Kutta Discontinuous Galerkin Method for Conservation Laws V: Multidimensional Systems*, Journal of Computational Physics, 141(2), 199

Costa, T., Arrigoni Battaia, F., Farina, E. P., Keating, L. C., Rosdahl, J., Kimm, T. (2022), *AGN-driven outflows and the formation of Lyα nebulae around high-z quasars*, MNRAS, 517(2), 1767

Costa, T., Pakmor, R., Springel, V. (2020), *Powering galactic superwinds with small-scale AGN winds*, MNRAS, 497(4), 5229

Cullen, L., Dehnen, W. (2010), *Inviscid smoothed particle hydrodynamics*, MNRAS, 408(2), 669

Dennis, T. J., Chandran, B. D. G. (2005), *Turbulent Heating of Galaxy-Cluster Plasmas*, ApJ, 622(1), 205

Deppe, N., Hébert, F., Kidder, L. E., Throwe, W., Anantpurkar, I., Armaza, C., Bonilla, G. S., Boyle, M., Chaudhary, H., Duez, M. D., Vu, N. L., Foucart, F., Giesler, M., Guo, J. S., Kim, Y., Kumar, P., Legred, I., Li, D., Lovelace, G., Ma, S., Macedo, A., Melchor, D., Morales, M., Moxon, J., Nelli, K. C., O'Shea, E., Pfeiffer, H. P., Ramirez, T., Rüter, H. R., Sanchez, J., Scheel, M. A., Thomas, S., Vieira, D., Wittek, N. A., Wlodarczyk, T., Teukolsky, S. A. (2022), *Simulating magnetized neutron stars with discontinuous Galerkin methods*, Phys. Rev. D, 105(12), 123031

Dolag, K., Borgani, S., Murante, G., Springel, V. (2009), *Substructures in hydrodynamical cluster simulations*, MNRAS, 399(2), 497

Edelmann, P. V. F., Ratnasingam, R. P., Pedersen, M. G., Bowman, D. M., Prat, V., Rogers, T. M. (2019), *Three-dimensional Simulations of Massive Stars. I. Wave Generation and Propagation*, ApJ, 876(1), 4

Event Horizon Telescope Collaboration, Akiyama, K., Alberdi, A., Alef, W., Asada, K., Azulay, R., Baczko, A.-K., Ball, D., Baloković, M., Barrett, J., Bintley, D., Blackburn, L., Boland, W., Bouman, K. L., Bower, G. C., Bremer, M., Brinkerink, C. D., Brissenden, R., Britzen, S., Broderick, A. E., Broguiere, D., Bronzwaer, T., Byun, D.-Y., Carlstrom, J. E., Chael, A., Chan, C.-k., Chatterjee, S., Chatterjee, K., Chen, M.-T., Chen, Y., Cho, I., Christian, P., Conway, J. E., Cordes, J. M., Crew, G. B., Cui, Y., Davelaar, J., De Laurentis, M., Deane, R., Dempsey, J., Desvignes, G., Dexter, J., Doeleman, S. S., Eatough, R. P., Falcke, H., Fish, V. L., Fomalont, E., Fraga-Encinas, R., Freeman, W. T., Friberg, P., Fromm, C. M., Gómez, J. L., Galison, P., Gammie, C. F., García, R., Gentaz, O., Georgiev, B., Goddi, C., Gold, R., Gu, M., Gurwell, M., Hada, K., Hecht, M. H., Hesper, R., Ho, L. C., Ho, P., Honma, M., Huang, C.-W. L., Huang, L., Hughes, D. H., Ikeda, S., Inoue, M., Issaoun, S., James, D. J., Jannuzi, B. T., Janssen, M., Jeter, B., Jiang, W., Johnson, M. D., Jorstad, S., Jung, T., Karami, M., Karuppusamy, R., Kawashima, T., Keating, G. K., Kettenis, M., Kim, J.-Y., Kim, J., Kim, J., Kino, M., Koay, J. Y., Koch, P. M., Koyama, S., Kramer, M., Kramer, C., Krichbaum, T. P., Kuo, C.-Y., Lauer, T. R., Lee, S.-S., Li, Y.-R., Li, Z., Lindqvist, M., Liu, K., Liuzzo, E., Lo, W.-P., Lobanov, A. P., Loinard, L., Lonsdale, C., Lu, R.-S., MacDonald, N. R., Mao, J., Markoff, S., Marrone, D. P., Marscher, A. P., Martí-Vidal, I., Matsushita, S., Matthews, L. D., Medeiros, L., Menten, K. M., Mizuno, Y., Mizuno, I., Moran, J. M., Moriyama, K., Moscibrodzka, M., Müller, C., Nagai, H., Nagar, N. M., Nakamura, M., Narayan, R., Narayanan, G., Natarajan, I., Neri, R., Ni, C., Noutsos, A., Okino, H., Olivares, H., Ortiz-León, G. N., Oyama, T., Özel, F., Palumbo, D. C. M., Patel, N., Pen, U.-L., Pesce, D. W., Piétu, V., Plambeck, R., PopStefanija, A., Porth, O., Prather, B., Preciado-López, J. A., Psaltis, D., Pu, H.-Y., Ramakrishnan, V., Rao, R., Rawlings, M. G., Raymond, A. W., Rezzolla, L., Ripperda, B., Roelofs, F., Rogers, A., Ros, E., Rose, M., Roshanineshat, A., Rottmann, H., Roy, A. L., Ruszczyk, C., Ryan, B. R., Rygl, K. L. J., Sánchez, S., Sánchez-Arguelles, D., Sasada, M., Savolainen, T., Schloerb, F. P., Schuster, K.-F., Shao, L., Shen, Z., Small, D., Sohn, B. W., SooHoo, J., Tazaki, F., Tiede, P., Tilanus, R. P. J., Titus, M., Toma, K., Torne, P., Trent, T., Trippe, S., Tsuda, S., van Bemmel, I., van Langevelde, H. J., van Rossum, D. R., Wagner, J., Wardle, J., Weintroub, J., Wex, N., Wharton, R., Wielgus, M., Wong, G. N., Wu, Q., Young, K., Young, A., Younsi, Z., Yuan, F., Yuan, Y.-F., Zensus, J. A., Zhao, G., Zhao, S.-S., Zhu, Z., Algaba, J.-C., Allardi, A., Amestica, R., Anczarski, J., Bach, U., Baganoff, F. K., Beaudoin, C., Benson, B. A., Berthold, R., Blanchard, J. M., Blundell, R., Bustamente, S., Cappallo, R., Castillo-Domínguez, E., Chang, C.-C., Chang, S.-H., Chang, S.-C., Chen, C.-C., Chilson, R., Chuter, T. C., Córdova Rosado, R., Coulson, I. M., Crawford, T. M., Crowley, J., David, J., Derome, M., Dexter, M., Dornbusch, S., Dudevoir, K. A., Dzib, S. A., Eckart, A., Eckert, C., Erickson, N. R., Everett, W. B., Faber, A., Farah, J. R., Fath, V., Folkers, T. W., Forbes, D. C., Freund, R., Gómez-Ruiz, A. I., Gale, D. M., Gao, F., Geertsema, G., Graham, D. A., Greer, C. H., Grosslein, R., Gueth, F., Haggard, D., Halverson, N. W., Han, C.-C., Han, K.-C., Hao, J., Hasegawa, Y., Henning, J. W., Hernández-Gómez, A., Herrero-Illana, R., Heyminck, S., Hirota, A., Hoge, J., Huang, Y.-D., Impellizzeri, C. M. V., Jiang, H., Kamble, A., Keisler, R.,

Kimura, K., Kono, Y., Kubo, D., Kuroda, J., Lacasse, R., Laing, R. A., Leitch, E. M., Li, C.-T., Lin, L. C. C., Liu, C.-T., Liu, K.-Y., Lu, L.-M., Marson, R. G., Martin-Cocher, P. L., Massingill, K. D., Matulonis, C., McColl, M. P., McWhirter, S. R., Messias, H., Meyer-Zhao, Z., Michalik, D., Montaña, A., Montgomerie, W., Mora-Klein, M., Muders, D., Nadolski, A., Navarro, S., Neilsen, J., Nguyen, C. H., Nishioka, H., Norton, T., Nowak, M. A., Nystrom, G., Ogawa, H., Oshiro, P., Oyama, T., Parsons, H., Paine, S. N., Peñalver, J., Phillips, N. M., Poirier, M., Pradel, N., Primiani, R. A., Raffin, P. A., Rahlin, A. S., Reiland, G., Risacher, C., Ruiz, I., Sáez-Madaín, A. F., Sassella, R., Schellart, P., Shaw, P., Silva, K. M., Shiokawa, H., Smith, D. R., Snow, W., Souccar, K., Sousa, D., Sridharan, T. K., Srinivasan, R., Stahm, W., Stark, A. A., Story, K., Timmer, S. T., Vertatschitsch, L., Walther, C., Wei, T.-S., Whitehorn, N., Whitney, A. R., Woody, D. P., Wouterloot, J. G. A., Wright, M., Yamaguchi, P., Yu, C.-Y., Zeballos, M., Zhang, S., Ziurys, L. (2019), *First M87 Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole*, ApJ, 875(1), L1

Fabian, A. C. (2012), *Observational Evidence of Active Galactic Nuclei Feedback*, ARA&A, 50, 455

Federrath, C. (2013), *On the universality of supersonic turbulence*, MNRAS, 436(2), 1245

Federrath, C., Klessen, R. S., Iapichino, L., Beattie, J. R. (2021), *The sonic scale of interstellar turbulence*, Nature Astronomy, 5, 365

Federrath, C., Klessen, R. S., Iapichino, L., Hammer, N. J. (2016), *The world's largest turbulence simulations*, arXiv e-prints, arXiv:1607.00630

Federrath, C., Klessen, R. S., Schmidt, W. (2008), *The Density Probability Distribution in Compressible Isothermal Turbulence: Solenoidal versus Compressive Forcing*, ApJ, 688(2), L79

Federrath, C., Klessen, R. S., Schmidt, W. (2009), *The Fractal Density Structure in Supersonic Isothermal Turbulence: Solenoidal Versus Compressive Energy Injection*, ApJ, 692(1), 364

Federrath, C., Roman-Duval, J., Klessen, R. S., Schmidt, W., Mac Low, M. M. (2010), *Comparing the statistics of interstellar turbulence in simulations and observations. Solenoidal versus compressive turbulence forcing*, A&A, 512, A81

Frisch, U. (1995), *Turbulence. The legacy of A.N. Kolmogorov*

Gebhardt, K., Bender, R., Bower, G., Dressler, A., Faber, S. M., Filippenko, A. V., Green, R., Grillmair, C., Ho, L. C., Kormendy, J., Lauer, T. R., Magorrian, J., Pinkney, J., Richstone, D., Tremaine, S. (2000), *A Relationship between Nuclear Black Hole Mass and Galaxy Velocity Dispersion*, ApJ, 539(1), L13

Girma, E., Teyssier, R. (2024), *A new star formation recipe for magnetohydrodynamics simulations of galaxy formation*, MNRAS, 527(3), 6779

Goldreich, P., Sridhar, S. (1995), *Toward a Theory of Interstellar Turbulence. II. Strong Alfvenic Turbulence*, ApJ, 438, 763

Gómez, G. C., Vázquez-Semadeni, E., Palau, A. (2021), *Density profile evolution during prestellar core collapse: collapse starts at the large scale*, MNRAS, 502(4), 4963

Greengard, L., Rokhlin, V. (1987), *A fast algorithm for particle simulations*, Journal of Computational Physics, 73(2), 325

Guillet, T., Pakmor, R., Springel, V., Chandrashekar, P., Klingenberg, C. (2019), *High-order magnetohydrodynamics for astrophysics with an adaptive mesh refinement discontinuous Galerkin scheme*, MNRAS, 485(3), 4209

Häring, N., Rix, H.-W. (2004), *On the Black Hole Mass-Bulge Mass Relation*, ApJ, 604(2), L89

Herbst, W., Assousa, G. E. (1979), *Supernovas and Star Formation*, Scientific American, 241(2), 138

Herwig, F., Woodward, P. R., Mao, H., Thompson, W. R., Denissenkov, P., Lau, J., Blouin, S., Andrassy, R., Paul, A. (2023), *3D hydrodynamic simulations of massive main-sequence stars - I. Dynamics and mixing of convection and internal gravity waves*, MNRAS, 525(2), 1601

Hillebrandt, W., Niemeyer, J. C. (2000), *Type Ia Supernova Explosion Models*, Annual Review of Astronomy and Astrophysics, 38(1), 191

Hopkins, P. F. (2015), *A new class of accurate, mesh-free hydrodynamic simulation methods*, MNRAS, 450(1), 53

Janett, G., Steiner, O., Alsina Ballester, E., Belluzzi, L., Mishra, S. (2019), *A novel fourth-order WENO interpolation technique. A possible new tool designed for radiative transfer*, A&A, 624, A104

Jermyn, A. S., Bauer, E. B., Schwab, J., Farmer, R., Ball, W. H., Bellinger, E. P., Dotter, A., Joyce, M., Marchant, P., Mombarg, J. S. G., Wolf, W. M., Sunny Wong, T. L., Cinquegrana, G. C., Farrell, E., Smolec, R., Thoul, A., Cantiello, M., Herwig, F., Toloza, O., Bildsten, L., Townsend, R. H. D., Timmes, F. X. (2023), *Modules for Experiments in Stellar Astrophysics (MESA): Time-dependent Convection, Energy Conservation, Automatic Differentiation, and Infrastructure*, ApJS, 265(1), 15

Junk, V., Walch, S., Heitsch, F., Burkert, A., Wetzstein, M., Schartmann, M., Price, D. (2010), *Modelling shear flows with smoothed particle hydrodynamics and grid-based methods*, MNRAS, 407(3), 1933

Kidder, L. E., Field, S. E., Foucart, F., Schnetter, E., Teukolsky, S. A., Bohn, A., Deppe, N., Diener, P., Hébert, F., Lippuner, J., Miller, J., Ott, C. D., Scheel, M. A., Vincent, T. (2017), *SpECTRE: A task-based discontinuous Galerkin code for relativistic astrophysics*, Journal of Computational Physics, 335, 84

Kim, J., Moin, P., Moser, R. (1987), *Turbulence statistics in fully developed channel flow at low Reynolds number*, Journal of Fluid Mechanics, 177, 133

King, A., Pounds, K. (2015), *Powerful Outflows and Feedback from Active Galactic Nuclei*, ARA&A, 53, 115

Klein, K. G., Spence, H., Alexandrova, O., Argall, M., Arzamasskiy, L., Bookbinder, J., Broeren, T., Caprioli, D., Case, A., Chandran, B., Chen, L.-J., Dors, I., Eastwood, J., Forsyth, C., Galvin, A., Genot, V., Halekas, J., Hesse, M., Hine, B., Horbury, T., Jian, L., Kasper, J., Kretzschmar, M., Kunz, M., Lavraud, B., Le Contel, O., Mallet, A., Maruca, B., Matthaeus, W., Niehof, J., O'Brien, H., Owen, C., Retinò, A., Reynolds, C., Roberts, O., Schekochihin, A., Skoug, R., Smith, C., Smith, S., Steinberg, J., Stevens, M., Szabo, A., TenBarge, J., Torbert, R., Vasquez, B., Verscharen, D., Whittlesey, P., Wickizer, B., Zank, G., Zweibel, E. (2023), *HelioSwarm: A Multipoint, Multiscale Mission to Characterize Turbulence*, Space Sci. Rev., 219(8), 74

Klessen, R. S. (2000), *One-Point Probability Distribution Functions of Supersonic Turbulent Flows in Self-gravitating Media*, The Astrophysical Journal, 535, 869, ADS Bibcode: 2000ApJ...535..869K

Kolmogorov, A. (1941), *The Local Structure of Turbulence in Incompressible Viscous Fluid for Very Large Reynolds' Numbers*, Akademiia Nauk SSSR Doklady, 30, 301

Kormendy, J., Ho, L. C. (2013), *Coevolution (Or Not) of Supermassive Black Holes and Host Galaxies*, ARA&A, 51(1), 511

Kravtsov, A. V. (2003), *On the Origin of the Global Schmidt Law of Star Formation*, The Astrophysical Journal, 590, L1, ADS Bibcode: 2003ApJ...590L...1K

Kravtsov, A. V., Borgani, S. (2012), *Formation of Galaxy Clusters*, Annual Review of Astronomy and Astrophysics, 50(1), 353

Kretschmer, M., Teyssier, R. (2020), *Forming early-type galaxies without AGN feedback: a combination of merger-driven outflows and inefficient star formation*, MNRAS, 492(1), 1385

Krivodonova, L. (2007), *Limiters for high-order discontinuous Galerkin methods*, Journal of Computational Physics, 226(1), 879

Küchler, C., Bewley, G., Bodenschatz, E. (2019), *Experimental Study of the Bottleneck in Fully Developed Turbulence*, Journal of Statistical Physics, 175(3-4), 617

Landau, L. D., Lifshitz, E. M. (1959), *Fluid Mechanics*, Course of theoretical physics / by L. D. Landau and E. M. Lifshitz, Vol. 6, New York: Pergamon, second edition

Larson, R. B. (1981), *Turbulence and star formation in molecular clouds.*, MNRAS, 194, 809

Launder, B. (1974), *The numerical computation of turbulent flows*, Computer Methods in Applied Mechanics and Engineering, 3(2), 269

Lazarian, A. (2016), *Damping of Alfvén Waves by Turbulence and Its Consequences: From Cosmic-ray Streaming to Launching Winds*, ApJ, 833(2), 131

Lazarian, A., Xu, S. (2022), *Damping of Alfvén Waves in MHD Turbulence and Implications for Cosmic Ray Streaming Instability and Galactic Winds*, Frontiers in Physics, 10, 702799

Lecoanet, D., McCourt, M., Quataert, E., Burns, K. J., Vasil, G. M., Oishi, J. S., Brown, B. P., Stone, J. M., O'Leary, R. M. (2016), *A validated non-linear Kelvin-Helmholtz benchmark for numerical hydrodynamics*, MNRAS, 455(4), 4274

Li, S., Zhang, Q., Liu, H. B., Beuther, H., Palau, A., Girart, J. M., Smith, H., Hora, J. L., Lin, Y., Qiu, K., Strom, S., Wang, J., Li, F., Yue, N. (2020), *ALMA Observations of NGC 6334S. I. Forming Massive Stars and Clusters in Subsonic and Transonic Filamentary Clouds*, ApJ, 896(2), 110

Li, Y., Klessen, R. S., Mac Low, M.-M. (2003), *The Formation of Stellar Clusters in Turbulent Molecular Clouds: Effects of the Equation of State*, The Astrophysical Journal, 592, 975, ADS Bibcode: 2003ApJ...592..975L

Liska, R., Wendroff, B. (2003), *Comparison of Several Difference Schemes on 1D and 2D Test Problems for the Euler Equations*, SIAM Journal on Scientific Computing, 25(3), 995

Lombart, M., Laibe, G. (2021), *Grain growth for astrophysics with discontinuous Galerkin schemes*, MNRAS, 501(3), 4298

Mac Low, M.-M., Balsara, D. S., Kim, J., de Avillez, M. A. (2005), *The Distribution of Pressures in a Supernova-driven Interstellar Medium. I. Magnetized Medium*, The Astrophysical Journal, 626, 864, ADS Bibcode: 2005ApJ...626..864M

Mac Low, M.-M., Klessen, R. S. (2004), *Control of star formation by supersonic turbulence*, Reviews of Modern Physics, 76(1), 125

Mandelker, N., van den Bosch, F. C., Springel, V., van de Voort, F. (2019), *Shattering of Cosmic Sheets due to Thermal Instabilities: A Formation Channel for Metal-free Lyman Limit Systems*, ApJ, 881(1), L20

Markert, J., Gassner, G., Walch, S. (2021), *A Sub-element Adaptive Shock Capturing Approach for Discontinuous Galerkin Methods*, Communications on Applied Mathematics and Computation, doi: 10.1007/s42967-021-00120-x

Markert, J., Walch, S., Gassner, G. (2022), *A discontinuous Galerkin solver in the FLASH multiphysics framework*, MNRAS, 511(3), 4179

Mathew, S. S., Federrath, C., Seta, A. (2023), *The role of the turbulence driving mode for the initial mass function*, MNRAS, 518(4), 5190

Mbarek, R., Caprioli, D. (2019), *Bottom-up Acceleration of Ultra-high-energy Cosmic Rays in the Jets of Active Galactic Nuclei*, The Astrophysical Journal, 886(1), 8

McConnell, N. J., Ma, C.-P. (2013), *Revisiting the Scaling Relations of Black Hole Masses and Host Galaxy Properties*, ApJ, 764(2), 184

McKee, C. F., Ostriker, E. C. (2007), *Theory of Star Formation*, ARA&A, 45(1), 565

McNally, C. P., Lyra, W., Passy, J.-C. (2012), *A Well-posed Kelvin-Helmholtz Instability Test and Comparison*, ApJS, 201(2), 18

Medina-Torrejón, T. E., de Gouveia Dal Pino, E. M., Kowal, G. (2023), *Particle Acceleration by Magnetic Reconnection in Relativistic Jets: The Transition from Small to Large Scales*, ApJ, 952(2), 168

Mellor, G. L., Yamada, T. (1982), *Development of a Turbulence Closure Model for Geophysical Fluid Problems (Paper 2R0808)*, Reviews of Geophysics and Space Physics, 20, 851

Menter, F. R. (1994), *Two-equation eddy-viscosity turbulence models for engineering applications*, AIAA Journal, 32(8), 1598

Mocz, P., Vogelsberger, M., Sijacki, D., Pakmor, R., Hernquist, L. (2014), *A discontinuous Galerkin method for solving the fluid and magnetohydrodynamic equations in astrophysical simulations*, MNRAS, 437(1), 397

Monaghan, J. J. (1992), *Smoothed particle hydrodynamics.*, ARA&A, 30, 543

Monaghan, J. J., Gingold, R. A. (1983), *Shock Simulation by the Particle Method SPH*, Journal of Computational Physics, 52(2), 374

Morris, J. P., Monaghan, J. J. (1997), *A Switch to Reduce SPH Viscosity*, Journal of Computational Physics, 136(1), 41

Mossier, P., Beck, A., Munz, C.-D. (2022), *A p-Adaptive Discontinuous Galerkin Method with hp-Shock Capturing*, Journal of Scientific Computing, 91(1), 4

Nagakura, T., Hosokawa, T., Omukai, K. (2009), *Star formation triggered by supernova explosions in young galaxies*, Monthly Notices of the Royal Astronomical Society, 399(4), 2183

Nelson, R. P., Papaloizou, J. C. B., Masset, F., Kley, W. (2000), *The migration and growth of protoplanets in protostellar discs*, MNRAS, 318(1), 18

NVIDIACorporation (2021), *CUDA C++ Programming Guide: Design Guide*, https://docs.nvidia.com/cuda/archive/11.2.0/pdf/CUDA_C_Programming_Guide.pdf

Ocvirk, P., Gillet, N., Shapiro, P. R., Aubert, D., Iliev, I. T., Teyssier, R., Yepes, G., Choi, J.-H., Sullivan, D., Knebe, A., Gottlöber, S., D'Aloisio, A., Park, H., Hoffman, Y., Stranex, T. (2016), *Cosmic Dawn (CoDa): the First Radiation-Hydrodynamics Simulation of Reionization and Galaxy Formation in the Local Universe*, MNRAS, 463(2), 1462

Ostriker, E. C., Gammie, C. F., Stone, J. M. (1999), *Kinetic and Structural Evolution of Self-gravitating, Magnetized Clouds: 2.5-dimensional Simulations of Decaying Turbulence*, The Astrophysical Journal, 513, 259, ADS Bibcode: 1999ApJ...513..259O

Pakmor, R., Springel, V., Bauer, A., Mocz, P., Munoz, D. J., Ohlmann, S. T., Schaal, K., Zhu, C. (2016), *Improving the convergence properties of the moving-mesh code AREPO*, MNRAS, 455(1), 1134

Pan, L., Ju, W., Chen, J.-H. (2022), *An exact relation for density fluctuations in compressible turbulence*, MNRAS, 514(1), 105

Pan, L., Wheeler, J. C., Scalo, J. (2008), *The Effect of Turbulent Intermittency on the Deflagration to Detonation Transition in Supernova Ia Explosions*, The Astrophysical Journal, 681(1), 470

Passot, T., Vázquez-Semadeni, E. (1998), *Density probability distribution in one-dimensional polytropic gas dynamics*, Physical Review E, 58(4), 4501

Paxton, B., Bildsten, L., Dotter, A., Herwig, F., Lesaffre, P., Timmes, F. (2011), *Modules for Experiments in Stellar Astrophysics (MESA)*, ApJS, 192(1), 3

Paxton, B., Smolec, R., Schwab, J., Gautschy, A., Bildsten, L., Cantiello, M., Dotter, A., Farmer, R., Goldberg, J. A., Jermyn, A. S., Kanbur, S. M., Marchant, P., Thoul, A., Townsend, R. H. D., Wolf, W. M., Zhang, M., Timmes, F. X. (2019), *Modules for Experiments in Stellar Astrophysics (MESA): Pulsating Variable Stars, Rotation, Convective Boundaries, and Energy Conservation*, ApJS, 243(1), 10

Perlmutter, S., Aldering, G., Goldhaber, G., Knop, R. A., Nugent, P., Castro, P. G., Deustua, S., Fabbro, S., Goobar, A., Groom, D. E., Hook, I. M., Kim, A. G., Kim, M. Y., Lee, J. C., Nunes, N. J., Pain, R., Pennypacker, C. R., Quimby, R., Lidman, C.,

Ellis, R. S., Irwin, M., McMahon, R. G., Ruiz-Lapuente, P., Walton, N., Schaefer, B., Boyle, B. J., Filippenko, A. V., Matheson, T., Fruchter, A. S., Panagia, N., Newberg, H. J. M., Couch, W. J., Project, T. S. C. (1999), *Measurements of Omega and Lambda from 42 High-Redshift Supernovae*, The Astrophysical Journal, 517(2), 565

Perrone, L. M., Latter, H. (2022), *Magneto-thermal instability in galaxy clusters - I. Theory and two-dimensional simulations*, MNRAS, 513(3), 4605

Persson, P.-O., Peraire, J. (2006), *Sub-Cell Shock Capturing for Discontinuous Galerkin Methods*, AIAA Inc., Reston, VA

Piffaretti, R., Jetzer, P., Kaastra, J. S., Tamura, T. (2005), *Temperature and entropy profiles of nearby cooling flow clusters observed with XMM-Newton*, A&A, 433(1), 101

Pope, S. B. (2000), *Turbulent Flows*, Cambridge University Press

Price, D. J. (2008), *Modelling discontinuities and Kelvin Helmholtz instabilities in SPH*, Journal of Computational Physics, 227(24), 10040

Price, D. J., Federrath, C. (2010), *A comparison between grid and particle methods on the statistics of driven, supersonic, isothermal turbulence*, MNRAS, 406(3), 1659

Quataert, E. (2008), *Buoyancy Instabilities in Weakly Magnetized Low-Collisionality Plasmas*, ApJ, 673(2), 758

Rabatin, B., Collins, D. C. (2023), *Density and velocity correlations in isothermal supersonic turbulence*, MNRAS, 525(1), 297

Rawle, T. D., Edge, A. C., Egami, E., Rex, M., Smith, G. P., Altieri, B., Fiedler, A., Haines, C. P., Pereira, M. J., Pérez-González, P. G., Portouw, J., Valtchanov, I., Walth, G., van der Werf, P. P., Zemcov, M. (2012), *The Relation between Cool Cluster Cores and Herschel-detected Star Formation in Brightest Cluster Galaxies*, ApJ, 747(1), 29

Reed, W. H., Hill, T. R. (1973), *Triangular mesh methods for the neutron transport equation*

Reinecke, M., Hillebrandt, W., Niemeyer, J. C. (2002), *Three-dimensional simulations of type Ia supernovae*, A&A, 391, 1167

Riess, A. G., Filippenko, A. V., Challis, P., Clocchiatti, A., Diercks, A., Garnavich, P. M., Gilliland, R. L., Hogan, C. J., Jha, S., Kirshner, R. P., Leibundgut, B., Phillips, M. M., Reiss, D., Schmidt, B. P., Schommer, R. A., Smith, R. C., Spyromilio, J., Stubbs, C., Suntzeff, N. B., Tonry, J. (1998), *Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant*, The Astronomical Journal, 116(3), 1009

Rizzuti, F., Hirschi, R., Arnett, W. D., Georgy, C., Meakin, C., Murphy, A. S., Rauscher, T., Varma, V. (2023), *3D stellar evolution: hydrodynamic simulations of a complete burning phase in a massive star*, Monthly Notices of the Royal Astronomical Society, 523(2), 2317

Robertson, B. E., Kravtsov, A. V., Gnedin, N. Y., Abel, T., Rudd, D. H. (2010), *Computational Eulerian hydrodynamics and Galilean invariance*, MNRAS, 401(4), 2463

Scalo, J., Vázquez-Semadeni, E., Chappell, D., Passot, T. (1998), *On the Probability Density Function of Galactic Gas. I. Numerical Simulations and the Significance of the Polytropic Index*, The Astrophysical Journal, 504, 835, ADS Bibcode: 1998ApJ...504..835S

Schaal, K., Bauer, A., Chandrashekar, P., Pakmor, R., Klingenberg, C., Springel, V. (2015), *Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement*, MNRAS, 453(4), 4278

Schlottke-Lakemper, M., Winters, A. R., Ranocha, H., Gassner, G. J. (2021), *A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics*, Journal of Computational Physics, 442, 110467

Schmidt, W., Ciaraldi-Schoolmann, F., Niemeyer, J. C., Röpke, F. K., Hillebrandt, W. (2010), *Turbulence in a Three-Dimensional Deflagration Model For Type Ia Supernovae. II. Intermittency and the Deflagration-to-Detonation Transition Probability*, ApJ, 710(2), 1683

Schmidt, W., Hillebrandt, W., Niemeyer, J. C. (2006), *Numerical dissipation and the bottleneck effect in simulations of compressible isotropic turbulence*, Computers & Fluids, 35(4), 353

Schmidt, W., Niemeyer, J. C., Hillebrandt, W. (2006), *A localised subgrid scale model for fluid dynamical simulations in astrophysics. I. Theory and numerical tests*, A&A, 450(1), 265

Schneider, E. E., Robertson, B. E. (2015), *CHOLLA: A New Massively Parallel Hydrodynamics Code for Astrophysical Simulation*, ApJS, 217(2), 24

Shu, C.-W., Osher, S. (1989), *Efficient Implementation of Essentially Non-oscillatory Shock-Capturing Schemes, II*, Journal of Computational Physics, 83(1), 32

Springel, V. (2010), *E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh*, MNRAS, 401(2), 791

Springel, V., Di Matteo, T., Hernquist, L. (2005), *Modelling feedback from stars and black holes in galaxy mergers*, MNRAS, 361(3), 776

Springel, V., Hernquist, L. (2002), *Cosmological smoothed particle hydrodynamics simulations: the entropy equation*, MNRAS, 333(3), 649

Springel, V., Pakmor, R., Zier, O., Reinecke, M. (2021), *Simulating cosmic structure formation with the GADGET-4 code*, MNRAS, 506(2), 2871

Springel, V., Yoshida, N., White, S. D. M. (2001), *GADGET: a code for collisionless and gasdynamical cosmological simulations*, New Astron., 6(2), 79

Steindl, T., Zwintz, K., Vorobyov, E. (2022), *The imprint of star formation on stellar pulsations*, Nature Communications, 13(1), 5355

Stone, J. M., Gardiner, T. A., Teuben, P., Hawley, J. F., Simon, J. B. (2008), *Athena: A New Code for Astrophysical MHD*, ApJS, 178(1), 137

Stone, J. M., Norman, M. L. (1992), *ZEUS-2D: A Radiation Magnetohydrodynamics Code for Astrophysical Flows in Two Space Dimensions. I. The Hydrodynamic Algorithms and Tests*, ApJS, 80, 753

Tassis, K., Ramaprakash, A. N., Readhead, A. C. S., Potter, S. B., Wehus, I. K., Panopoulou, G. V., Blinov, D., Eriksen, H. K., Hensley, B., Karakci, A., Kypriotakis, J. A., Maharana, S., Ntormousi, E., Pavlidou, V., Pearson, T. J., Skalidis, R. (2018), *PASIPHAE: A high-Galactic-latitude, high-accuracy optopolarimetric survey*, arXiv e-prints, arXiv:1810.05652

Toro, E. (2009), *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, XXIV, 724, Springer Berlin, Heidelberg

Townsend, R. H. D. (2009), *An Exact Integration Scheme for Radiative Cooling in Hydrodynamical Simulations*, ApJS, 181(2), 391

Trac, H., Pen, U.-L. (2003), *A Primer on Eulerian Computational Fluid Dynamics for Astrophysics*, PASP, 115(805), 303

Tricco, T. S. (2019), *The Kelvin-Helmholtz instability and smoothed particle hydrodynamics*, MNRAS, 488(4), 5210

Valcke, S., de Rijcke, S., Rödiger, E., Dejonghe, H. (2010), *Kelvin-Helmholtz instabilities in smoothed particle hydrodynamics*, MNRAS, 408(1), 71

van Leer, B., Nomura, S. (2005), *17th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Toronto, Ontario, Canada

Veilleux, S., Maiolino, R., Bolatto, A. D., Aalto, S. (2020), *Cool outflows in galaxies and their implications*, A&ARv, 28(1), 2

Velasco Romero, D. A., Han Veiga, M., Teyssier, R., Masset, F. S. (2018), *Planet-disc interactions with discontinuous Galerkin methods using GPUs*, MNRAS, 478(2), 1855

Verma, M. K., Donzis, D. (2007), *Energy transfer and bottleneck effect in turbulence*, Journal of Physics A Mathematical General, 40(16), 4401

Vilar, F. (2019), *A posteriori correction of high-order discontinuous Galerkin scheme through subcell finite volume formulation and flux reconstruction*, Journal of Computational Physics, 387, 245

Vogelsberger, M., Marinacci, F., Torrey, P., Puchwein, E. (2020), *Cosmological simulations of galaxy formation*, Nature Reviews Physics, 2(1), 42

Von Neumann, J., Richtmyer, R. D. (1950), *A Method for the Numerical Calculation of Hydrodynamic Shocks*, Journal of Applied Physics, 21(3), 232

Wada, K., Norman, C. A. (2001), *Numerical Models of the Multiphase Interstellar Matter with Stellar Energy Feedback on a Galactic Scale*, The Astrophysical Journal, 547, 172, ADS Bibcode: 2001ApJ...547..172W

Weinberger, R., Springel, V., Hernquist, L., Pillepich, A., Marinacci, F., Pakmor, R., Nelson, D., Genel, S., Vogelsberger, M., Naiman, J., Torrey, P. (2017), *Simulating galaxy formation with black hole driven thermal and kinetic feedback*, MNRAS, 465(3), 3291

Weinberger, R., Springel, V., Pakmor, R. (2020), *The AREPO Public Code Release*, ApJS, 248(2), 32

Wibking, B. D., Krumholz, M. R. (2022), *QUOKKA: a code for two-moment AMR radiation hydrodynamics on GPUs*, MNRAS, 512(1), 1430

Wilkins, M. L. (1980), *Use of artificial viscosity in multidimensional fluid dynamic calculations*, Journal of Computational Physics, 36(3), 281

Woodward, P., Colella, P. (1984), *The Numerical Stimulation of Two-Dimensional Fluid Flow with Strong Shocks*, Journal of Computational Physics, 54(1), 115

Xu, G. (1995), *A New Parallel N-Body Gravity Solver: TPM*, ApJS, 98, 355

Yee, H. C., Sandham, N. D., Djomehri, M. J. (1999), *Low-Dissipative High-Order Shock-Capturing Methods Using Characteristic-Based Filters*, Journal of Computational Physics, 150(1), 199

Yee, H. C., Vinokur, M., Djomehri, M. J. (2000), *Entropy Splitting and Numerical Dissipation*, Journal of Computational Physics, 162(1), 33

Zanotti, O., Fambri, F., Dumbser, M., Hidalgo, A. (2015), *Space–time adaptive ADER discontinuous Galerkin finite element schemes with a posteriori sub-cell finite volume limiting*, Computers & Fluids, 118, 204

Zhang, X., Shu, C.-W. (2010), *On positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes*, Journal of Computational Physics, 229(23), 8918

Zhao, F., Pan, L., Li, Z., Wang, S. (2017), *A new class of high-order weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, Computers & Fluids, 159, 81

Acknowledgements

I am deeply thankful to my supervisor, Volker Springel, without his extensive supervision, guidance and magic C++ MPI skills I would still be still stuck in 1D. I really appreciated his availability to answer any and all questions I ever had, regardless of his immensely busy schedule.

I am also grateful to Rüdiger Pakmor for relentless willingness to help in all matters of numerics and career. And to Klaus Dolag for his role as the second assessor for my thesis.

Additional thanks go to Thorsten Naab with whom I did an internship at MPA back in 2017 during which I decided MPA is *the place* where I want to do my PhD. Thank you for all the personal and career support throughout my MPA journey. Looking back to 2017 reminded me of how we were still young and blissfully unaware of what was waiting for us in 2020 when I started my PhD.

Talking about the pandemic™ I am first immensely grateful to my parents Tanja and Simon who not only exfiltrated me from Germany just hours before it closed its borders on March 16th, but alongside my brother Blaž warmly welcomed me back to my childhood home during the first lockdown. It was a surprise and a pleasure to spend about half a year in the same household setup that we had back in my high school times.

I could not have made it through the pandemic™ without the many friends I made at MPA. Before starting my PhD I had the pleasure of getting to known Matteo, Max, Ivan, Timo and Bernhard. I have to express gratitude to the many fellow PhD students and postdocs with whom we've created countless forever memories. Thank you fellow PhDs; Chris B., Monica, Nahir, Aniket, Oliver, Simon, Christian, Jessie, Teresa, and postdocs; Tiara, Tiago, Enrico, Daniela and Deepika.

Thank you Hitesh and Fulvio for caring about the MPA community and taking over The Commonroom, I look forward to sharing a cold beverage with you during my future visits to MPA.

Thank you Chris D. for the almost PhD long D&D session! Anna and Adam, one could not ask for better co-adventurers!

A big shout-out to the Commonroom thief thanks to whom I got to experience the German judicial system.

Andrej, thank you for your support me through the intense period of grief following my father's death.

My thanks go to all the secretaries, thank you Maria, Cornelia, Gabi, and Sonja. Their hard work, dedication and efficiency are greatly appreciated.

Finally, I want to thank my family for their support and encouragement throughout my PhD. I am especially grateful to my parents and grandparents for their unconditional love and support.